

# Simatic S7 to Logix5000 Application Conversion Guide



Application Solution

## Important User Information

Solid state equipment has operational characteristics differing from those of electromechanical equipment. Safety Guidelines for the Application, Installation and Maintenance of Solid State Controls (publication SGI-1.1 available from your local Rockwell Automation sales office or online at <http://literature.rockwellautomation.com>) describes some important differences between solid state equipment and hard-wired electromechanical devices. Because of this difference, and also because of the wide variety of uses for solid state equipment, all persons responsible for applying this equipment must satisfy themselves that each intended application of this equipment is acceptable.






In no event will Rockwell Automation, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Rockwell Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by Rockwell Automation, Inc. with respect to use of information, circuits, equipment, or software described in this manual.

Reproduction of the contents of this manual, in whole or in part, without written permission of Rockwell Automation, Inc., is prohibited.

Throughout this manual, when necessary, we use notes to make you aware of safety considerations.

|   |  |
|---|--|
|    | Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.                             |
|     | Identifies information that is critical for successful application and understanding of the product.   |
|  | Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence |
|  | Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.  |
|  | Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.   |

Allen-Bradley, Rockwell Automation, and TechConnect are trademarks of Rockwell Automation, Inc.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

|  |  |    |
|--|--|----|
| <b>Preface</b>   | Purpose . . . . .  | 7  |
|  | Conversion versus Translation . . . . .                            | 7  |
|  | Terminology . . . . .  | 8  |
|  | Additional Resources . . . . .                                     | 8  |
|  | PLC Logic Conversion Services Provided by Rockwell Automation . .  | 9  |
|  | Service Features . . . . .   | 9  |
|  | One-stop PLC Program Conversion Services . . . . .                 | 9  |
|  | Service Benefits . . . . .   | 10 |
|  | Services Offered. . . . .  | 10 |
|  | Basic Conversion Package. . . . .                                  | 10 |
|  | Conversion Plus Initial Clean-up Package . . . . .                 | 10 |
|  | Additional Options . . . . .                                       | 11 |
|  | Additional Program Conversions Available . . . . .                 | 11 |
|  | <br>   |    |
|  | <b>Chapter 1</b>   |    |
| <b>Hardware Conversion</b>                                 | Introduction . . . . .   | 13 |
|  | S7 Controllers . . . . .   | 13 |
|  | I/O Systems . . . . .  | 14 |
|  | S7 Local I/O . . . . .   | 14 |
|  | Selection and Configuration of S7 I/O Components . . . . .         | 14 |
|  | Logix Local I/O. . . . .   | 16 |
|  | Selection and Configuration of Logix I/O Components . . . . .      | 18 |
|  | S7 Remote I/O . . . . .  | 20 |
|  | Configuration of S7 Profibus DP Remote I/O . . . . .               | 21 |
|  | Logix Distributed I/O. . . . .                                     | 22 |
|  | Configuration of Logix Distributed I/O . . . . .                   | 22 |
|  | Networks . . . . .   | 25 |
|  | Networks in S7 . . . . .   | 25 |
|  | Networks in Logix. . . . .   | 27 |
|  | Conversion of HMI . . . . .  | 31 |
|  | Conversion of Systems Containing Distributed Controllers . . . . . | 32 |
|  | Hardware and Software Implementation . . . . .                     | 32 |
|  | Connecting Siemens and Rockwell Automation Devices . . . . .       | 34 |
|  | Controllers . . . . .  | 34 |
|  | Distributed Devices . . . . .                                      | 34 |
|  | <br>   |    |
|  | <b>Chapter 2</b>   |    |
| <b>Logix Features that May Not be Familiar to S7 Users</b> | Introduction . . . . .   | 35 |
|  | S7 Organization Blocks Compared to Logix Tasks . . . . .           | 36 |
|  | Organization Blocks in S7. . . . .                                 | 36 |
|  | Tasks in Logix . . . . .   | 41 |
|  | Task Monitor . . . . .   | 46 |
|  | Tags Not Addresses . . . . .                                       | 47 |
|  | Data Areas in S7. . . . .  | 47 |
|  | Data in Logix . . . . .  | 50 |
|  | I/O and Alias Tags. . . . .  | 51 |
|  | Programming Languages . . . . .                                    | 53 |

|   |   |    |
|---|---|----|
|   | Logix Ladder Diagram . . . . .                        | 54 |
|   | Logix Structured Text . . . . .                       | 54 |
|   | Logix Function Block Diagram . . . . .                | 55 |
|   | Logix Sequential Function Chart . . . . .             | 55 |
|   | Conversion of STEP 7 Code to Logix . . . . .          | 55 |
|   | Arrays not Pointers . . . . .                         | 56 |
|   | Add-On Instructions . . . . .                         | 57 |
|   | Add-On Instruction Summary . . . . .                  | 57 |
|   | Backing Tags . . . . .                                | 58 |
|   | The Common Industrial Protocol (CIP) . . . . .        | 58 |
|   | Viewing the Network . . . . .                         | 59 |
|   | Data Exchange between Controllers . . . . .           | 60 |
|   | Send / Receive in STEP 7 . . . . .                    | 60 |
|   | Produced / Consumed Tags in Logix . . . . .           | 60 |
|   | User-Defined Data Types . . . . .                     | 61 |
|   | Asynchronous I/O Updating . . . . .                   | 62 |
|   | The DINT Data Type . . . . .                          | 62 |
|   | Phase Manager . . . . .                               | 63 |
|   | Phase Management in STEP 7 . . . . .                  | 63 |
|   | PhaseManager in Logix . . . . .                       | 63 |
|   | Coordinated System Time (CST) . . . . .               | 65 |
|   | Timestamped Inputs . . . . .                          | 65 |
|   | Scheduled Outputs . . . . .                           | 65 |
|   | No Temporary Variables . . . . .                      | 66 |
|   | No Accumulators or Special Registers needed . . . . . | 66 |
|   | <b>Chapter 3</b>                                      |    |
| <b>Conversion of System Software<br/>and Standard Functions</b> | Introduction . . . . .                                | 67 |
|   | Logix System Functions . . . . .                      | 68 |
|   | Copy . . . . .  | 68 |
|   | Date and Time Setting and Reading . . . . .           | 69 |
|   | Read System Time . . . . .                            | 69 |
|   | Handling of Interrupts . . . . .                      | 70 |
|   | Errors . . . . .                                      | 70 |
|   | Status – Controller . . . . .                         | 71 |
|   | Status – Module . . . . .                             | 71 |
|   | Status – for OBs and Tasks . . . . .                  | 72 |
|   | Timers . . . . .                                      | 72 |
|   | Conversion Routines . . . . .                         | 73 |
|   | String Handling Routines . . . . .                    | 73 |
|   | Examples of System Function Calls . . . . .           | 74 |
|   | Setting the Clock . . . . .                           | 74 |
|   | Disabling Interrupts . . . . .                        | 76 |
| Read System Time . . . . .                                      | 78  |    |
| Get Faults . . . . .  | 79  |    |
| Module Information . . . . .                                    | 80  |    |

|   |  |     |
|---|--|-----|
|   | Get Scan Time . . . . .  | 81  |
| <b>Conversion of Typical Program Structures</b> | <b>Chapter 4</b>   |     |
|   | Introduction . . . . .   | 83  |
|   | Conversion Code Examples . . . . .                                 | 83  |
|   | Ladder Logic Translation . . . . .                                 | 83  |
|   | Jumps and Decision Making . . . . .                                | 90  |
|   | Arrays . . . . .   | 94  |
|   | User Data Types . . . . .  | 99  |
|   | Pointers and Arrays . . . . .                                      | 102 |
|   | State Machine . . . . .  | 103 |
|   | STEP 7 State Machine . . . . .                                     | 104 |
|   | Strings . . . . .  | 108 |
|   | STEP 7 Temporary Variables . . . . .                               | 110 |
|   | Functions . . . . .  | 110 |
|   | Block Copy, COP and CPS . . . . .                                  | 114 |
|   | Mathematical Expressions . . . . .                                 | 116 |
|   | Other Topics Related to Programming . . . . .                      | 120 |
|   | Scope of Variables . . . . .                                       | 120 |
|   | OBs, Tasks, and Scheduling . . . . .                               | 120 |
|   | A Larger Example - Control Module . . . . .                        | 121 |
|   | Components of the CM . . . . .                                     | 121 |
|   | User Data Type Valve . . . . .                                     | 122 |
|   | The Add-On Instruction . . . . .                                   | 123 |
|   | Add-On Instruction Local Data . . . . .                            | 124 |
|   | Call-up . . . . .  | 127 |
| <b>Common Mistakes when Converting to Logix</b> | <b>Chapter 5</b>   |     |
|   | Introduction . . . . .   | 129 |
|   | Not Selecting Appropriate Hardware . . . . .                       | 129 |
|   | Underestimating Impact of Task Scheduling . . . . .                | 130 |
|   | Performing Translation Instead of Conversion . . . . .             | 130 |
|   | Not Using the Most Appropriate Logix Languages . . . . .           | 130 |
|   | Implementation of Incorrect Data Types – DINT versus INT . . . . . | 131 |
|   | Add DINTs . . . . .  | 131 |
|   | Add INTs . . . . .   | 131 |
|   | Timing Results . . . . .   | 131 |
|   | User Code Emulating Existing Instructions . . . . .                | 132 |
|   | User Code . . . . .  | 132 |
|   | COP Instruction . . . . .  | 132 |
|   | Incorrect Usage of COP, MOV, and CPS . . . . .                     | 133 |
|   | Incorrect Usage of CPT . . . . .                                   | 133 |
|   | Not Handling Strings in Optimal Way . . . . .                      | 133 |
|   | Extensive Usage of Jumps . . . . .                                 | 133 |
|   | Not Using Aliased Tags . . . . .                                   | 133 |

|   |  |
|---|--|
| <b>S7 to Logix Glossary</b>                       | <p><b>Chapter 6</b></p> <p>Introduction. . . . . 135</p> <p>Hardware Terminology . . . . . 135</p> <p>Software Terminology . . . . . 136</p>   |
| <b>S7 300 and S7 400 Parts and RA Equivalents</b> | <p><b>Appendix A</b></p> <p>Introduction . . . . . 139</p> <p>Compact S7 300 CPUs. . . . . 140</p> <p>Standard S7 300 CPUs. . . . . 140</p> <p>Technology S7 300 CPUs . . . . . 141</p> <p>Fail-Safe S7 300 CPUs . . . . . 142</p> <p>S7 300 Digital Input Modules . . . . . 142</p> <p>S7 300 Digital Output Modules. . . . . 143</p> <p>S7 300 Relay Output Modules. . . . . 144</p> <p>S7 300 Digital Combo Modules . . . . . 144</p> <p>S7 300 Analog Input Modules. . . . . 144</p> <p>S7 300 Analog Output Modules . . . . . 145</p> <p>S7 300 Analog Combo Modules . . . . . 146</p> <p>S7 400 Standard Controllers . . . . . 146</p> <p>Redundant and Fail Safe Controllers. . . . . 147</p> <p>Digital Input Modules . . . . . 147</p> <p>Digital Output Modules. . . . . 147</p> <p>Analog Input Modules. . . . . 148</p> <p>Analog Output Modules . . . . . 148</p> |
| <b>Siemens HMI Cross Reference Table</b>          | <p><b>Appendix B</b></p> <p>SIMATIC Micro Panels and Rockwell Automation Equivalents . . . 149</p> <p>SIMATIC Panels - 7x Series and Rockwell Automation<br/>Equivalents . . . . . 151</p> <p>SIMATIC Panels - 17x Series and Rockwell Automation<br/>Equivalents . . . . . 152</p> <p>SIMATIC Panels - 27x Series and Rockwell Automation<br/>Equivalents . . . . . 155</p> <p>SIMATIC Multi Panels - 27x Series and Rockwell Automation<br/>Equivalents . . . . . 157</p> <p>SIMATIC Multi Panels - 37x Series and Rockwell Automation<br/>Equivalents . . . . . 159</p>   |

### Purpose

This user manual provides guidance for users and engineers who have used control systems based on one of these two platforms:

- Siemens S7 Controller
- Rockwell Automation Logix Programmable Automation Controller (PAC)

And in addition:

- have a desire or a need to take advantage of the PAC features, or are in the early stages of migrating a S7 to Logix.
- have specific STEP 7 program code that they wish to convert to effective and efficient RSLogix 5000 code.

Use this manual to help you adopt good practices and to avoid common mistakes when converting the project to Logix.

### Conversion versus Translation

The theme of conversion versus translation is one that is repeated in this application conversion guide. Simple translation is focusing only on the line of code and finding an equivalent in the Logix languages. To convert an application optimally, you have to do more than just translate. For instance, you may benefit from choosing a different programming language, utilizing different programming techniques, and designing a different scheduling scheme to solve the same task. So, conversion is performed in a context of a higher level design and knowledge of the strengths of the Logix system.

If you have application code to convert, you will need to understand your STEP 7 program before you start conversion – either by having been involved yourself in its development, or by reading documentation of the program and of the process that it controls. If the program or the process is unfamiliar or poorly documented, proper conversion will be difficult – it will be mere translation and is less likely to succeed. For example, in Logix, there is a global name space, whereas in the Siemens environment there are data blocks that can be loaded/unloaded by application code. Appreciation of this helps you design a strategy for conversion.

In some cases, if the documentation of both the process and program is poor, it may be more efficient in terms of the overall project duration/cost to draw up a new specification and begin your Logix program with minimal time spent on translation from the old program.

## Terminology

STEP 7 is the programming software environment for Siemens SIMATIC S7 controllers. RSLogix 5000 software is used with Rockwell Automation Logix programmable automation controllers. We refer to Logix as a programmable automation controller because it does so much more than a traditional general-purpose PLC. It provides an excellent control platform for multi-discipline control, a common namespace, Coordinated System Time for truly scalable multi-CPU architectures, user-defined data types, and full NetLinx connectivity.

The term “Logix” is used to refer to any of the ControlLogix, CompactLogix, GuardLogix, FlexLogix, DriveLogix or SoftLogix controllers, or the RSLogix 5000 programming environment where it is clear from the context which is being referred to.

## Additional Resources

Every section of this application conversion guide references other Rockwell Automation user manuals, selection guides, and documents in which more information can be found.

| <b>Publication Number</b> | <b>Publication Title</b>   |
|---------------------------|--|
| 1756-SG001                | ControlLogix Controllers Selection Guide                             |
| 1769-SG001                | 1769 CompactLogix Controllers Selection Guide                        |
| 1768-UM001                | 1768 CompactLogix Controllers User Manual                            |
| 1769-SG002                | Compact I/O Selection Guide  |
| 1756-RM094                | Logix5000 Controllers Design Considerations Programming Manual       |
| 1756-PM001                | Logix5000 Controllers Common Procedures Programming Manual           |
| 1756-RM003                | Logix5000 Controllers General Instructions Reference Manual          |
| 1734-SG001                | POINT I/O Selection Guide  |
| 1738-SG001                | ArmorPoint I/O Selection Guide                                       |
| 1792-SG001                | ArmorBlock MaXum I/O and ArmorBlock I/O Selection Guide              |
| 1794-SG002                | FLEX I/O and FLEX Ex Selection Guide                                 |
| NETS-SG001                | NetLinx Selection Guide  |
| VIEW-SG001                | Visualization Platforms Selection Guide                              |
| IA-RM001                  | Integrated Architecture: Foundations of Modular Programming          |
| 6873-SG004                | Encompass Program Product Directory                                  |
| 1756-PM010                | Logix5000 Controllers Add-On Instructions Programming Manual         |
| 1756-RM087                | Logix5000 Controllers Execution Time and Memory Use Reference Manual |
| IASIMP-RM001              | IA Recommended Literature Reference Manual                           |



## PLC Logic Conversion Services Provided by Rockwell Automation

Rockwell Automation provides additional services for PLC logic conversion.

- Service Features
- One-stop PLC Program Conversion Services
- Service Benefits
- Services Offered
- Basic Conversion Package
- Conversion Plus Initial Clean-up Package
- Additional Program Conversions Available

### Service Features

Program Conversion Services will convert your legacy Allen-Bradley brand PLC or third-party programmable controller program to run on a Logix programmable automation control system, or the SLC 500/MicroLogix or PLC-5 programmable controllers.

Legacy products are often expensive to support and are difficult to repair, which can increase downtime and decrease production. For this reason, Rockwell Automation Customer Support now offers Program Conversion Services. These services are designed to reduce the cost and the time it takes to migrate from a legacy PLC to one of our current PAC or PLC-control platform families.

### One-stop PLC Program Conversion Services

Migration to a current Allen-Bradley control platform from a legacy product will improve your manufacturing process, system reliability and flexibility, give you more access to application processing power, and reduce equipment repair costs and spares inventory. With Program Conversion Services from Rockwell Automation Customer Support, your existing programmable controller program will be expertly and quickly converted to the new controller family. Rockwell Automation customer support engineers can help in the migration of legacy Allen-Bradley equipment or convert your PLC systems to Rockwell Automation products while minimizing downtime and maximizing operational success.

## Service Benefits

Specialists for each of the product platforms will be involved during the program conversion process. There are no hard to find anomalies in the logic caused by typing errors. In most cases, the entire data table is reproduced and no data is lost, as well as the original documentation is preserved, no re-typing of comments and symbols. Original Allen-Bradley brand programs can be in 6200, APS, or AI series format. New programs will be in the appropriate RSLogix format.

## Services Offered

Two program-conversion packages are available as well as project specific custom packages done on a case-by-case basis.

## Basic Conversion Package

- The original programmable controller program will be converted to the appropriate ControlLogix, CompactLogix, PLC-5, or SLC 500/MicroLogix format.
- The package provides an error listing generated during the conversion that includes instructions that are not directly convertible and any addresses that may not have been converted, which could include pointers and indirect addressing.
- The program and error listing would be returned to the customer for manual debugging and correction.

## Conversion Plus Initial Clean-up Package

- The original programmable controller program will be converted to the appropriate ControlLogix, PLC-5, or SLC 500/MicroLogix format.
- We will correct and convert any instruction and/or addressing errors to the new processor family.
- The completed program will then be returned to the customer for final startup and debugging.

---

## Additional Options

Additional options to either of the packages include the following:

- Application-level telephone support during the start-up and debugging phase of the project.
- Consultation on system re-engineering, operator interface, architecture and communication strategies, to take full advantage of the new platform's control capabilities that are not part of a code translation effort, training, and onsite startup is available as an added value from your local Global Sales and Solutions (GSS) office.
- Complete turn-key migration or upgrades are available from your local GSS/Engineered Systems Office.

## Additional Program Conversions Available

- PLC-2 format to ControlLogix, CompactLogix, PLC-5, SLC500/MicroLogix format
- PLC-3 format to ControlLogix, CompactLogix, or PLC-5 format
- PLC-5/250 format to ControlLogix or CompactLogix format
- Modicon – Quantum, 984, 584, 380, 381, 480, 485, 780, 785 to ControlLogix or CompactLogix format
- Siemens – S-5, S-7 to ControlLogix or CompactLogix format
- TI - 520, 520C, 525, 530, 530C, 535, 560, 560/565, 565, 560/560T, 560T, 545, 555, 575 to ControlLogix or CompactLogix format
- GE Series 6 to ControlLogix or CompactLogix format

Program conversions of other third-party programmable controllers to Allen-Bradley controller programs are also available. Contact technical support for details.

To schedule a conversion project, or learn more about the Program Conversion Services, contact your local Rockwell Automation sales office or authorized distributor: email us at [raprogramconversion@ra.rockwell.com](mailto:raprogramconversion@ra.rockwell.com), or visit <http://support.rockwellautomation.com/> and view KnowledgeBase Document G19154.

---

**IMPORTANT**

Use consultation services for re-engineering, typically to expand the system functionality and not to change out hardware due to obsolete or related reasons. SLC to Logix format and PLC-5 to Logix format conversions and PCE comment generation are built into RSLogix 5000 software.

---

**Notes:**

## Hardware Conversion

### Introduction

The objective of this chapter is to provide guidance to a user or engineer who needs to determine the correct Logix hardware as a replacement for the existing S7 equipment.

The chapter describes how to select controllers, local I/O, remote I/O, networks, and HMI, includes a section on distributed controller architecture, and provides HW conversion examples of the most often used S7 modules.

| Topic  | Page |
|--|------|
| S7 Controllers   | 13   |
| I/O Systems  | 14   |
| Networks   | 25   |
| Conversion of HMI  | 31   |
| Conversion of Systems Containing Distributed Controllers | 32   |
| Connecting Siemens and Rockwell Automation Devices       | 34   |

### S7 Controllers

This table lists a relevant sample selection of current Siemens S7 controllers, which are used to cover a wide range of applications.

#### Sample Selection of Current Siemens S7 Controllers

| Controller  | Part Number         | Logix Equivalent     |
|-------------|---------------------|----------------------|
| 313C        | 6ES7 313-5BF03-0AB0 | L23 Serial           |
| 314C-DP     | 6ES7 314-6CG03-0AB0 | L23 EtherNet/IP, L31 |
| 315-2 DP    | 6ES7 315-2AG10-0AB0 | L32E, L32C           |
| 317-2 DP    | 6ES7 317-6TJ10-0AB0 | L35CR, L35E          |
| 317T-2 DP   | 6ES7 317-6TJ10-0AB0 | L43, L45             |
| 319-3 PN/DP | 6ES7 318-3EL00-0AB0 | L45, L61             |
| 414-2       | 6ES7 414-2XK05-0AB0 | L61, L62             |
| 414-3       | 6ES7 414-3XM05-0AB0 | L62, L63, L64, L65   |
| 414-3 PN/DP | 6ES7 414-3EM05-0AB0 |                      |

**Sample Selection of Current Siemens S7 Controllers**

|                              |  |   |
|------------------------------|--|---|
| 315F-2 PN/DP (Safety)        | 6ES7 315-2FH13-0AB0<br>6ES7 317-2FK13-0AB0 | GuardLogix L61S, L62S, L63S   |
| 414-H (Redundant)<br>417-H   | 6ES7 414-4HM14-0AB0<br>6ES7 417-4HT14-0AB0 | L61-L65 with SRM  |
| PCS7 – Uses 417-4 controller |  | L3x, L4x, L6x +<br>FactoryTalk View,<br>FactoryTalk Batch<br>software |

A guide to the suitability of some of the most commonly used S7 controllers follows:

- S7 315-2DP – Small to medium-sized machines.
- S7 317-2DP – Medium to medium - large sized machines, small to medium process control applications.
- S7 414-2 – Demanding machine control, process control applications.
- S7 414-3 – Demanding machine control, large process control applications.

The complete range of S7 controllers is listed in [Appendix A](#).

**I/O Systems**

These sections describe Logix I/O systems to replace existing S7 equipment.

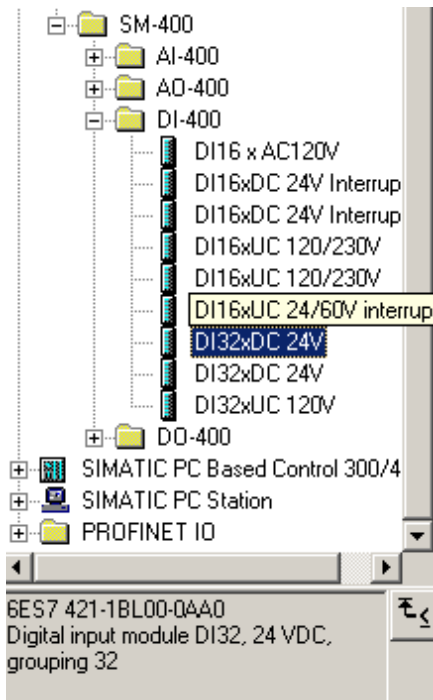
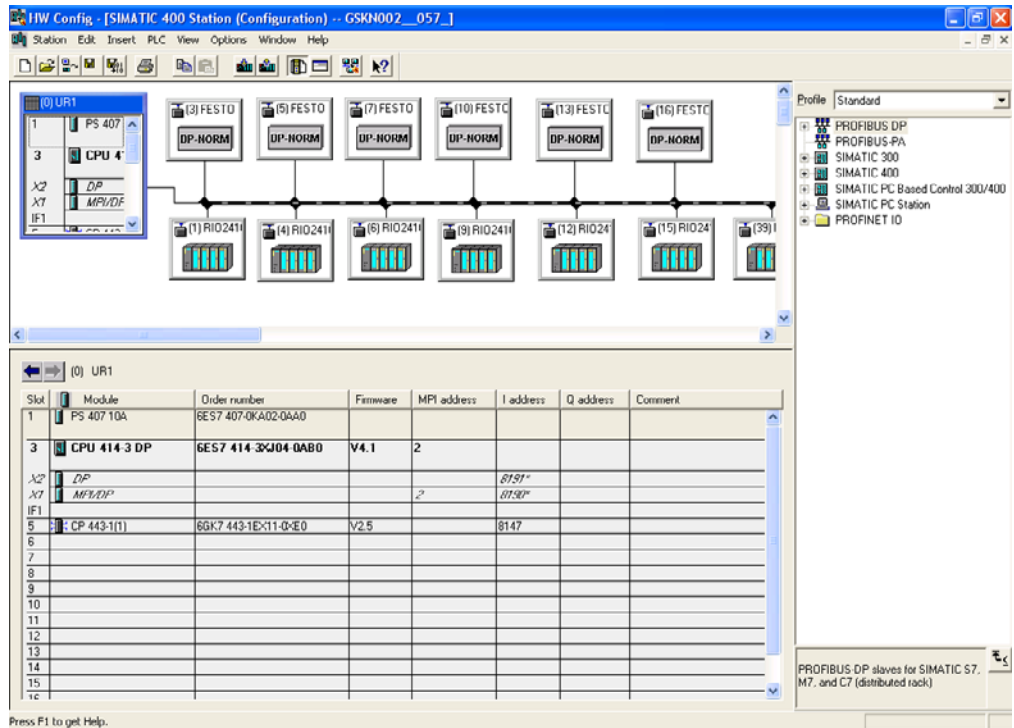
**S7 Local I/O**

There is a wide range of S7-300 and S7-400 I/O modules. S7-300 modules are mounted to standard DIN rail and connected to adjacent cards by using U-connectors, which are supplied with the modules. S7-400 modules are mounted to the S7-400 rack.

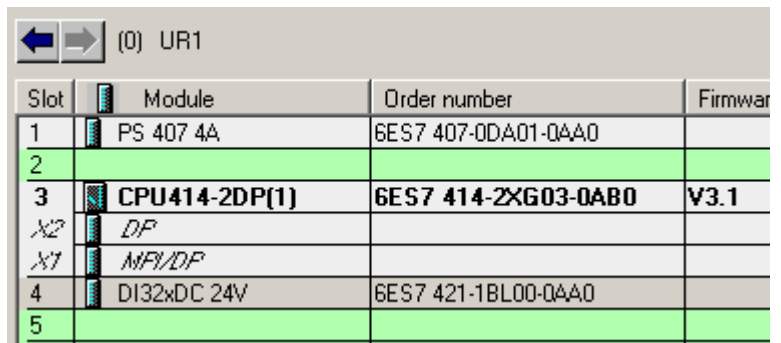
**Selection and Configuration of S7 I/O Components**

The screen shots that follow are from the STEP 7 Hardware Configuration program, a separate program in the STEP 7 application collection. In RSLogix 5000 software, this functionality is fully integrated as you will see later in this user manual.

### STEP 7 Hardware Configuration Program



Drag the selected module to the rack configuration screen.



## Logix Local I/O

A wide range of ControlLogix and CompactLogix I/O modules is available. 1769 I/O is cost-optimized for just-enough functionality as often requested by OEMs, while the 1756 I/O family provides high feature/functionality for the most demanding applications, as often requested by end users and sometimes required to meet specific performance levels.

CompactLogix modules are mounted to standard DIN rail and a special coupling system secures electrical and mechanical connection to adjacent modules. Engineers may welcome the mechanical coupling system – with the S7-300, modules are fixed to a special rail only and not to each other (other than by the electrical U-connector).

ControlLogix modules are mounted in the 1756 racks.

- For 1769-L31, 1769-L32C, 1769-L32E, and 1768-L43 controllers, the maximum number of I/O modules attached to the controller's rack is 16, in up to 3 banks.
- For 1769-L35CR, 1769-L35E, and 1768-L45 controllers, the maximum number of I/O modules attached to the controller's rack is 30, also in 3 banks.
- For 1756 controllers, the number of slots in the rack defines the maximum number of local I/O modules. It can be 4, 7, 10, 13, or 17.

On both platforms, further I/O can be networked via CIP networks, where EtherNet/IP and ControlNet networks provide the tightest, seamless I/O integration.



This table lists the Logix equivalents for some popular S7 I/O modules.

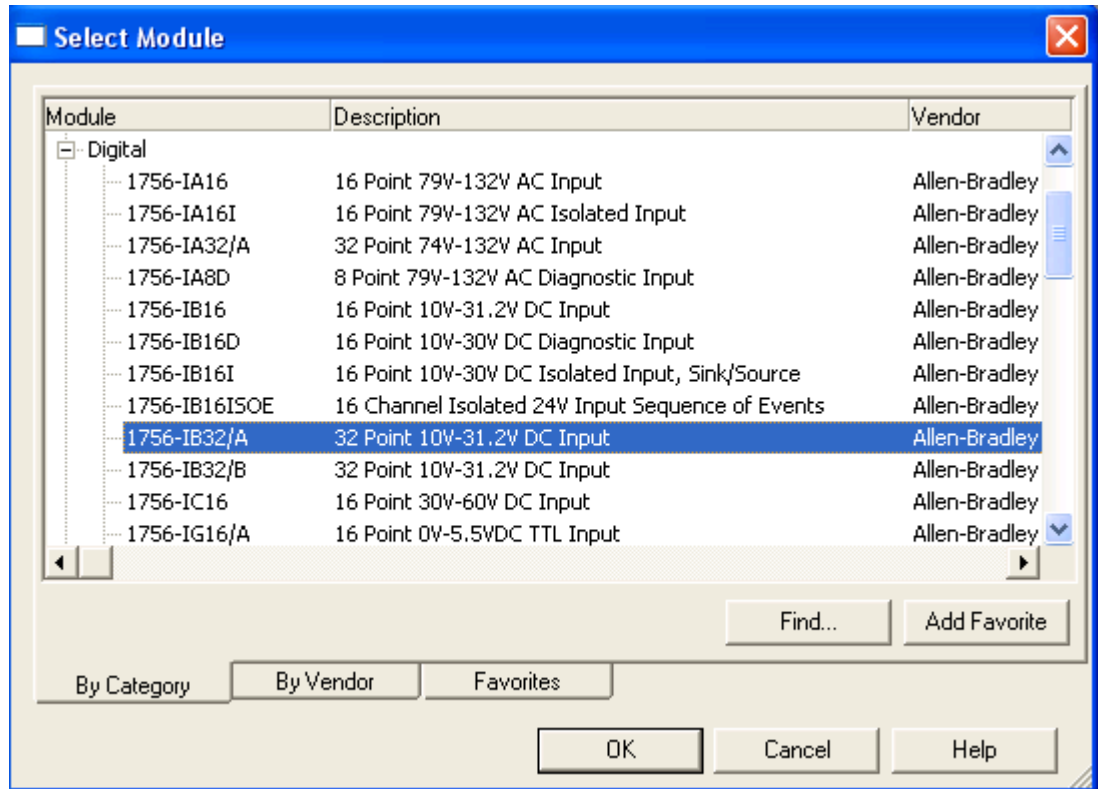
**Logix Equivalents for S7 I/O Modules**

| <b>S7 I/O module</b>  | <b>Description</b>               | <b>Logix Equivalent</b> | <b>Description</b>                     |
|-----------------------|----------------------------------|-------------------------|--|
| 6ES7 321-1BL00-0AA0   | S7-300 32 channel digital input  | 1769-IQ32               | CompactLogix 32 channel digital input  |
| 6ES7 322 - 1BH01-0AA0 | S7-300 16 channel digital output | 1769-OB16               | CompactLogix 16 channel digital output |
| 6ES7 421-1BL01-0AA0   | S7-400 32 channel digital input  | 1756-IB32               | ControlLogix 32 channel digital input  |
| 6ES7 422-1BH01-0AA0   | S7-400 16 channel digital output | 1756-OB16E              | ControlLogix 16 channel digital output |

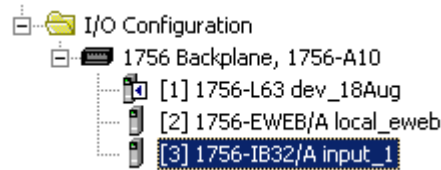
Refer to [Appendix A](#) for more detailed conversion tables of I/O modules.

## Selection and Configuration of Logix I/O Components

From the I/O Configuration branch of your project tree, the Logix library of device profiles can be accessed. These profiles provide full wizard-driven configuration for complete, easy-to-use integration into the data table and intuitive programmable control over each module's functionality, such as scaling, alarming, and diagnostics.



Select an item and it will appear in the rack in your I/O configuration.



The device profile tags for the new I/O module have been added automatically to the controller scope tag database.

|   |           |  |                |
|---|-----------|--|----------------|
| + | Local:3:C |  | AB:1756_DI:C:0 |
| + | Local:3:I |  | AB:1756_DI:I:0 |

The view below shows the tags partly expanded.

|                                 |  |                |
|---------------------------------|--|----------------|
| [-] Local:3:C                   |  | AB:1756_DI:C:0 |
| [+] Local:3:C.FilterOffOn_0_7   |  | SINT           |
| [+] Local:3:C.FilterOnOff_0_7   |  | SINT           |
| [+] Local:3:C.FilterOffOn_8_15  |  | SINT           |
| [+] Local:3:C.FilterOnOff_8_15  |  | SINT           |
| [+] Local:3:C.FilterOffOn_16_23 |  | SINT           |
| [+] Local:3:C.FilterOnOff_16_23 |  | SINT           |
| [+] Local:3:C.FilterOffOn_24_31 |  | SINT           |
| [+] Local:3:C.FilterOnOff_24_31 |  | SINT           |
| [+] Local:3:C.COSOnOffEn        |  | DINT           |
| [+] Local:3:C.COSOffOnEn        |  | DINT           |
| [+] Local:3:I                   |  | AB:1756_DI:I:0 |

The profile contains configuration and status data as well as I/O data.

|                                 |  |                |
|---------------------------------|--|----------------|
| [-] Local:0:C                   |  | AB:1756_DI:C:0 |
| [+] Local:0:C.FilterOffOn_0_7   |  | SINT           |
| [+] Local:0:C.FilterOnOff_0_7   |  | SINT           |
| [+] Local:0:C.FilterOffOn_8_15  |  | SINT           |
| [+] Local:0:C.FilterOnOff_8_15  |  | SINT           |
| [+] Local:0:C.FilterOffOn_16_23 |  | SINT           |
| [+] Local:0:C.FilterOnOff_16_23 |  | SINT           |
| [+] Local:0:C.FilterOffOn_24_31 |  | SINT           |
| [+] Local:0:C.FilterOnOff_24_31 |  | SINT           |
| [+] Local:0:C.COSOnOffEn        |  | DINT           |
| [+] Local:0:C.COSOffOnEn        |  | DINT           |
| [-] Local:0:I                   |  | AB:1756_DI:I:0 |
| [+] Local:0:I.Fault             |  | DINT           |
| [+] Local:0:I.Data              |  | DINT           |
| [-] Local:0:O                   |  | DINT           |

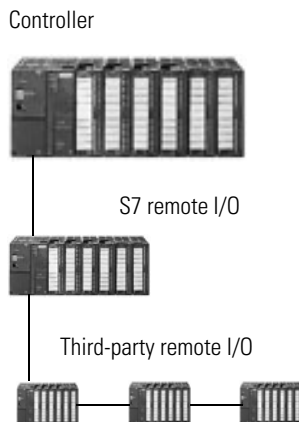
Refer to [Chapter 4](#) for more information.

## S7 Remote I/O

It is common to divide I/O between the controller's local rack and remote I/O stations, with communication under the Profibus DP network. These are the types of Profibus DP nodes:

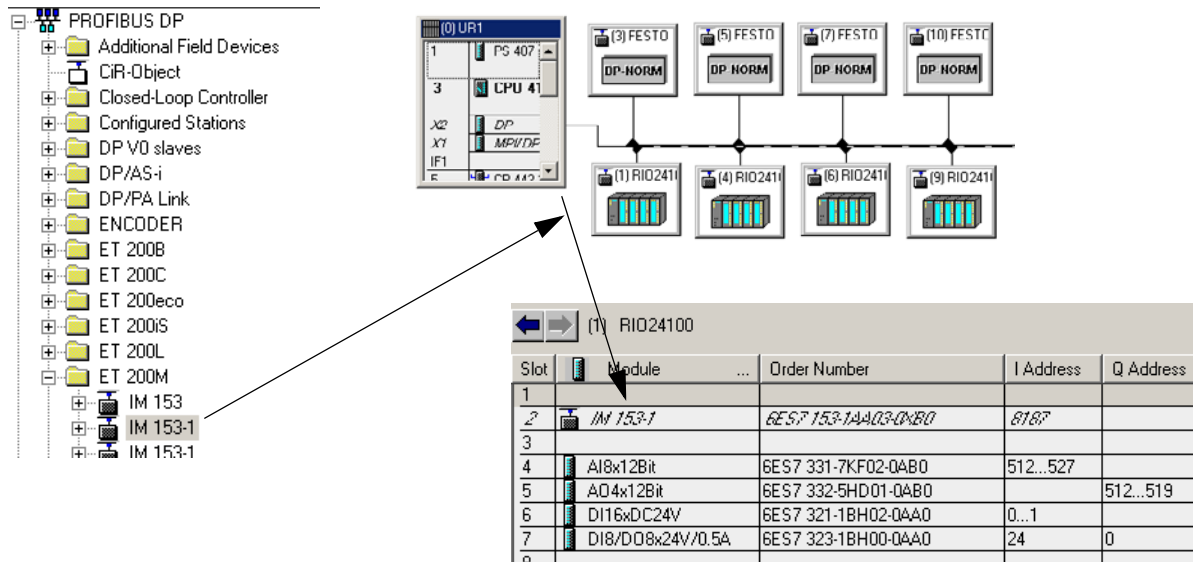
- S7 remote I/O, in which case standard S7-300 I/O modules are mounted in a remote I/O panel and interface with the Profibus DP bus via a special module. The controller sees this I/O as local I/O and assigns standard I/O addresses. This is called ET200M.
- Other Siemens remote I/O, such as ET200S (similar to the POINT I/O system) and ET200L (similar to the FLEX I/O system).
- Third-party remote I/O. A number of manufacturers of I/O and valves produce an interface to link their systems to the Profibus DP bus in the same way as S7 remote I/O. For these systems, a special integration file (GSD file) may need to be imported to your STEP 7 installation.
- Some manufacturers of more complex devices, such as, weigh scales and variable speed drives (VSD), produce Profibus DP interfaces for their products. For these systems, a special integration file (GSD file) will need to be imported to your STEP 7 installation. It is often necessary to refer to the manufacturer's documentation to learn the meaning of the data areas.

### Typical S7 I/O Configuration



## Configuration of S7 Profibus DP Remote I/O

A Profibus DP interface module can be installed in the hardware configuration by dragging from the hardware catalogue to the graphic of the Profibus DP bus. Once the interface module is installed, it can be opened and standard S7-300 modules added as if it were local I/O.



The screenshot shows the hardware configuration interface. On the left is a tree view of the hardware catalogue under 'PROFIBUS DP', with 'IM 153-1' selected. In the center, a rack diagram shows a 'UR1' rack with a 'CPU 41' and 'DP' module. A Profibus DP bus is connected to four 'DP-NORM' modules (3, 5, 7, 11) and four 'RIO241I' modules (1, 4, 6, 9). An arrow points from the 'IM 153-1' in the catalogue to the rack diagram. Below the rack diagram is a table for the 'RIO24100' module.

| Slot | Module           | Order Number        | I Address | Q Address |
|------|------------------|---------------------|-----------|-----------|
| 1    |                  |                     |           |           |
| 2    | IM 153-1         | 6ES7 153-1AA03-0AB0 | 8187      |           |
| 3    |                  |                     |           |           |
| 4    | A18x12Bit        | 6ES7 331-7KF02-0AB0 | 512...527 |           |
| 5    | A04x12Bit        | 6ES7 332-5HD01-0AB0 |           | 512...519 |
| 6    | D116xDC24V       | 6ES7 321-1BH02-0AA0 | 0...1     |           |
| 7    | D18/DO8x24V/0.5A | 6ES7 323-1BH00-0AA0 | 24        | 0         |
| 8    |                  |                     |           |           |

The data table defines the I/O addresses associated with the drive. Symbols for these addresses would be added manually in the Symbol Table. Hardware configuration is now complete.

It is possible to use remote devices on the Profibus DP network along with Logix, but with the same constraints/usability limitations you experience in the S7 environment.

## Logix Distributed I/O

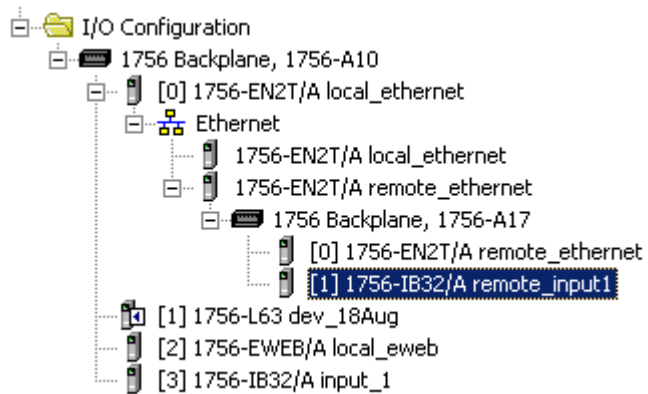
Rockwell Automation distributed I/O includes remote I/O using 1756 or 1769 I/O modules and various distributed I/O platforms, such as POINT I/O, FLEX I/O, ArmorPoint, and ArmorBlock systems.

The I/O modules are connected to the network by using a communication module or communication adapter, or directly by using a built-in communication interface.

## Configuration of Logix Distributed I/O

All I/O configuration is done in the project tree of RSLogix 5000 software. From the I/O Configuration branch, insert a communication module for your chosen network type.

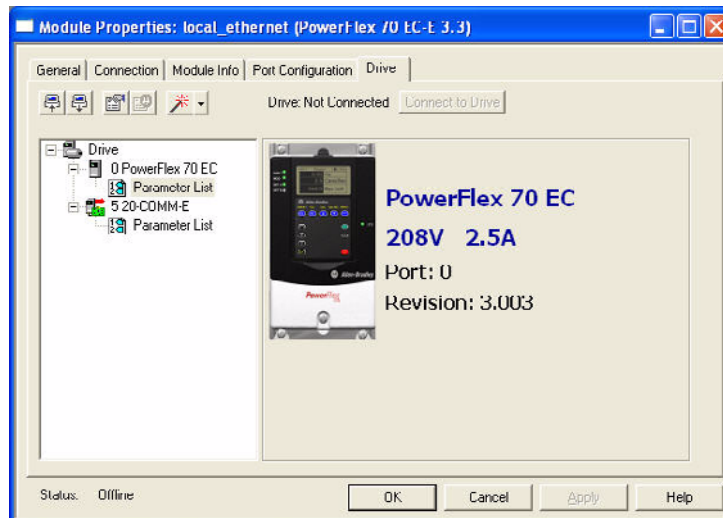
The screen shot shows an addition of a remote 1756-IB32 I/O module connected via an EtherNet/IP network.



Notice that tags corresponding to the remote I/O module have been added automatically to the controller scope tag database.

|                                      |  |  |                           |
|--------------------------------------|--|--|---------------------------|
| [-] remote_ethernet:I                |  |  | AB:1756_ENET_17SLOT:I:0   |
| [+] remote_ethernet:I.SlotStatusBits |  |  | DINT                      |
| [+] remote_ethernet:I.Slot           |  |  | AB:1756_ENET_SLOT:I:0[17] |

A networked variable speed drive, such as PowerFlex drive, can be added in the same way.



Again, RSLogix 5000 software will generate the new tags automatically for any device with a profile in RSLogix 5000 software and connected on an EtherNet/IP or ControlNet network. For the DeviceNet network, GuardLogix Safety I/O is integrated in the same way. Other DeviceNet devices need to be set up by using the RSNetWorx configuration software and EDS files that operate essentially equivalent to the STEP 7 Profibus manager software and GSD files.

Shown below are device profile tags in RSLogix 5000 software, available for hundreds of Rockwell Automation devices.

|   |  |  |
|---|--|--|
| - | PowerFlex_Drive:I                          |  |
| + | PowerFlex_Drive:I.DriveStatus              |  |
| - | PowerFlex_Drive:I.DriveStatus_Ready        |  |
| - | PowerFlex_Drive:I.DriveStatus_Active       |  |
| - | PowerFlex_Drive:I.DriveStatus_CommandDir   |  |
| - | PowerFlex_Drive:I.DriveStatus_ActualDir    |  |
| - | PowerFlex_Drive:I.DriveStatus_Accelerating |  |
| - | PowerFlex_Drive:I.DriveStatus_Decelerating |  |
| - | PowerFlex_Drive:I.DriveStatus_Alarm        |  |
| - | PowerFlex_Drive:I.DriveStatus_Faulted      |  |
| - | PowerFlex_Drive:I.DriveStatus_AtSpeed      |  |
| - | PowerFlex_Drive:I.DriveStatus_LocalID0     |  |
| - | PowerFlex_Drive:I.DriveStatus_LocalID1     |  |
| - | PowerFlex_Drive:I.DriveStatus_LocalID2     |  |
| - | PowerFlex_Drive:I.DriveStatus_SpdRefID0    |  |
| - | PowerFlex_Drive:I.DriveStatus_SpdRefID1    |  |
| - | PowerFlex_Drive:I.DriveStatus_SpdRefID2    |  |
| - | PowerFlex_Drive:I.DriveStatus_SpdRefID3    |  |
| + | PowerFlex_Drive:I.OutputFreq               |  |
| - | PowerFlex_Drive:O                          |  |
| + | PowerFlex_Drive:O.DriveLogicRslt           |  |
| - | PowerFlex_Drive:O.DriveLogicRslt_Stop      |  |
| - | PowerFlex_Drive:O.DriveLogicRslt_Start     |  |



## Networks

Refer to these sections for information about the networks.

### Networks in S7

#### *Profibus DP Network, DPV1, DPV3*

In the S7 world, the principal network type for communication with devices is the Profibus DP network in a variety of implementations. Some higher-range S7-300 and all S7-400 controllers have built-in Profibus master ports.

#### *Profibus Network - Other*

Profibus FMS and FDL are for data communication between controllers. They perform a similar function to the industrial Ethernet network, and the configuration is nearly identical. The differences are that Profibus communication processors are required rather than the Ethernet network, and that Profibus cabling will be used.

Profibus DPv2 can be used to connect to servo drives in the S7-315T and S7-317T controllers for low end motion control.

#### *Industrial Ethernet Network*

Siemens industrial Ethernet network is the Siemens variety of the Ethernet network in an industrial environment. It is used mainly for communication between controllers, and for controller-to-programming computer communication.

Apart from some of recent controllers equipped for Profinet, S7 controllers do not have built-in Ethernet ports. An S7 system using Industrial Ethernet will have communication processors mounted in the racks.

Depending on the communication processor, the following protocols can be used:

- S7 (Proprietary protocol for communication between S7 controllers)
- TCP (Transmission Control Protocol) Raw Sockets
- ISO-on-TCP (Extended TCP with additional checking)
- UDP (User Datagram Protocol) Raw Sockets

Application code is required to manage most aspects of communication on these networks.

In the Rockwell Automation environment, this functionality can be implemented using integrated EtherNet/IP ports, EtherNet/IP Bridge modules and/or EWEB modules.

### *Profinet*

Profinet provides for similar Profibus DP functionality on an Industrial Ethernet with the same programming overhead requirements. A network using Profinet is similar to Profibus except for different cable and connectors, and use Ethernet field interface modules rather than Profibus. Controllers with a built-in Profinet interface or a communication processor that is equipped for Profinet are used to connect to the network.

Alternatively, an existing Profibus DP network can be bridged to Profinet, either with a proxy or by using the Profibus DP port of a Profinet-equipped controller.

Some Profinet field interface modules have multiple RJ45 ports with an integrated switch, to allow a Profibus-type line bus topology, if required.

Profinet provides these three communication possibilities:

- Profinet CBA (Component Based Automation), which is primarily used for controller to controller communication and uses standard Ethernet hardware and the TCP/IP software stack.
- Profinet IO for scheduled transfers such as Drives or I/O modules and uses standard Ethernet hardware, but bypasses the TCP/IP software stack.
- Profinet IRT (Isochronous Real Time) for motion control applications which uses Profinet specific hardware and also bypasses the TCP/IP software stack and must exist on a protected network segment.

If the Profinet CBA framework is used, then Profibus, Profinet and Industrial Ethernet networks can be integrated by graphical configuration, with reduced need for additional programming. Rockwell Automation EtherNet/IP networks provide this functionality using standard hardware and the standard TCP/IP software stack utilizing built-in functions like the Message (MSG) instruction and produced/consumed tags.

## Networks in Logix

NetLinx is the term identifying the Rockwell Automation solution in the area of networking technologies. The following are the primary networks used in Logix systems:

- EtherNet/IP
- ControlNet
- DeviceNet

These networks have a variety of notable features. All are designed under the Common Industrial Protocol (CIP), which enables you to control, configure and collect data over any of the NetLinx networks. As a result, data can flow between different networks without any need for protocol translation software or proxies.

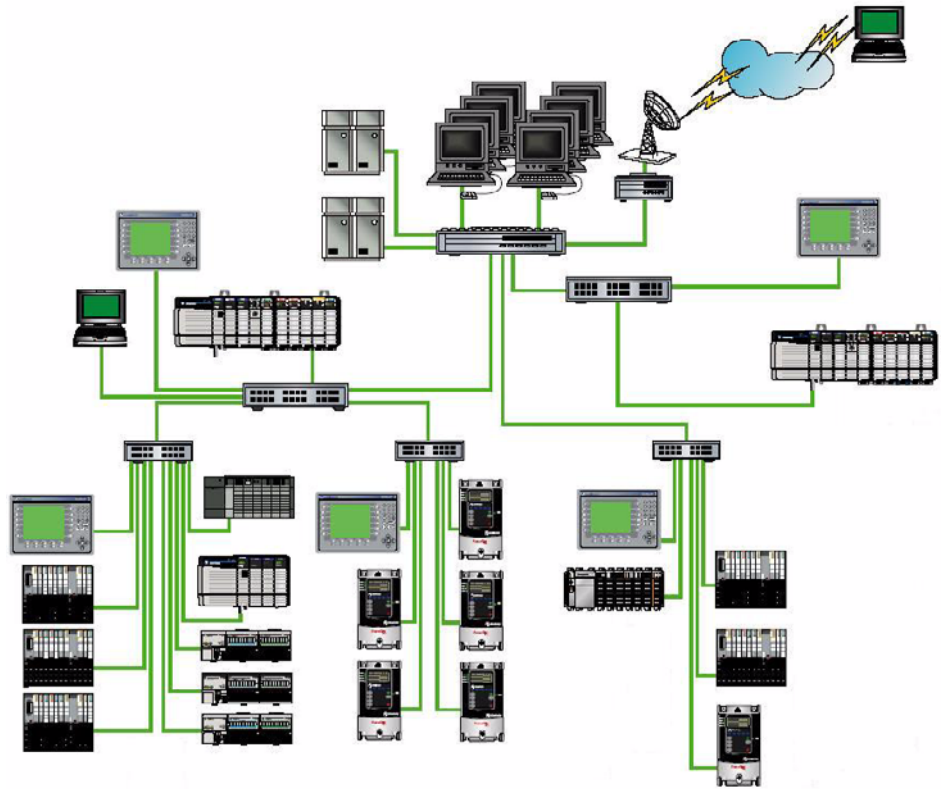
An engineer who is becoming familiar with Logix systems may be impressed by the integrated nature and elegance in configuration of Logix networks.

### *EtherNet/IP Network*

The EtherNet/IP network offers a full suite of control, configuration, and data collection services. It uses TCP/IP for general messaging/information exchange and UDP/IP for I/O messaging. It is most often used in these types of configurations:

- General I/O control
- Data exchange among controllers
- Connecting many computers
- Connecting many devices
- Connectivity to enterprise systems
- Integration of Safety devices
- Motion control (future)

### Typical Ethernet/IP Example

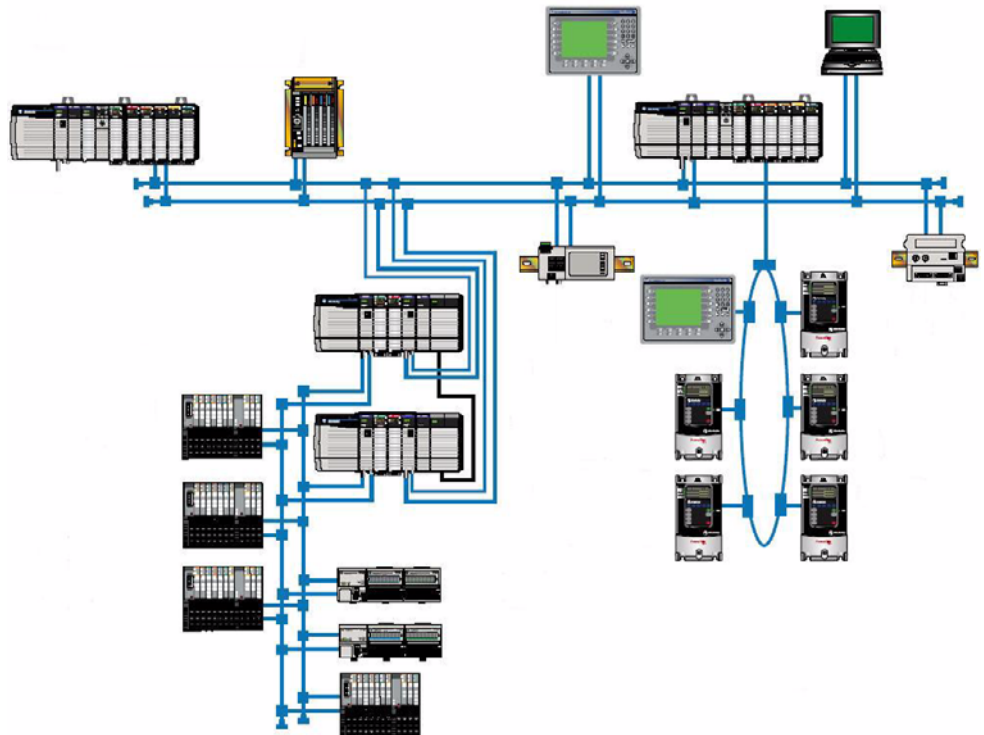


## *ControlNet Network*

ControlNet is a real-time control network that provides transport of both time-critical I/O and interlocking data and messaging data, including upload/download of programming and configuration data on a single physical media link. It is most often used in these types of configurations:

- General I/O control
- Data exchange among controllers
- Backbone to multiple distributed DeviceNet networks

### **Typical ControlNet Example**

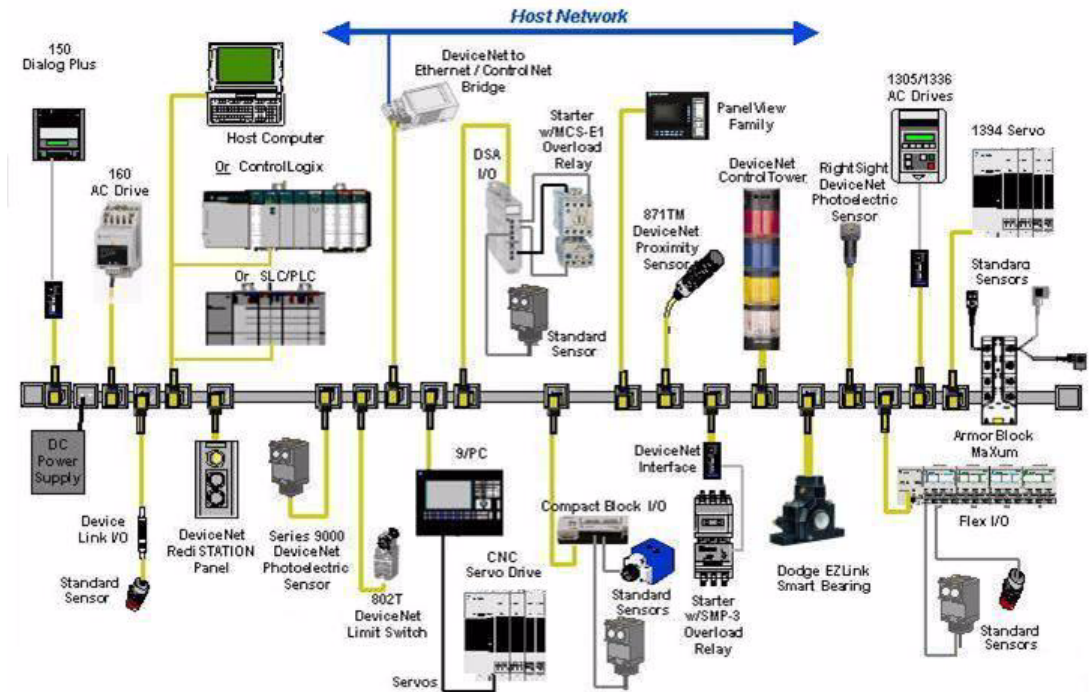


### DeviceNet Network

The DeviceNet network is a solution for low-level industrial device networking. Designed for devices with a low data volume per device for real time operation. It is most often used in these types of configurations:

- Applications containing distributed devices with a few points
- Network of third-party drives and other “simple” third-party devices
- Systems in which devices need to be connected directly to the network with data and power in the same connection
- When advanced diagnostic information is required

### Typical DeviceNet Example



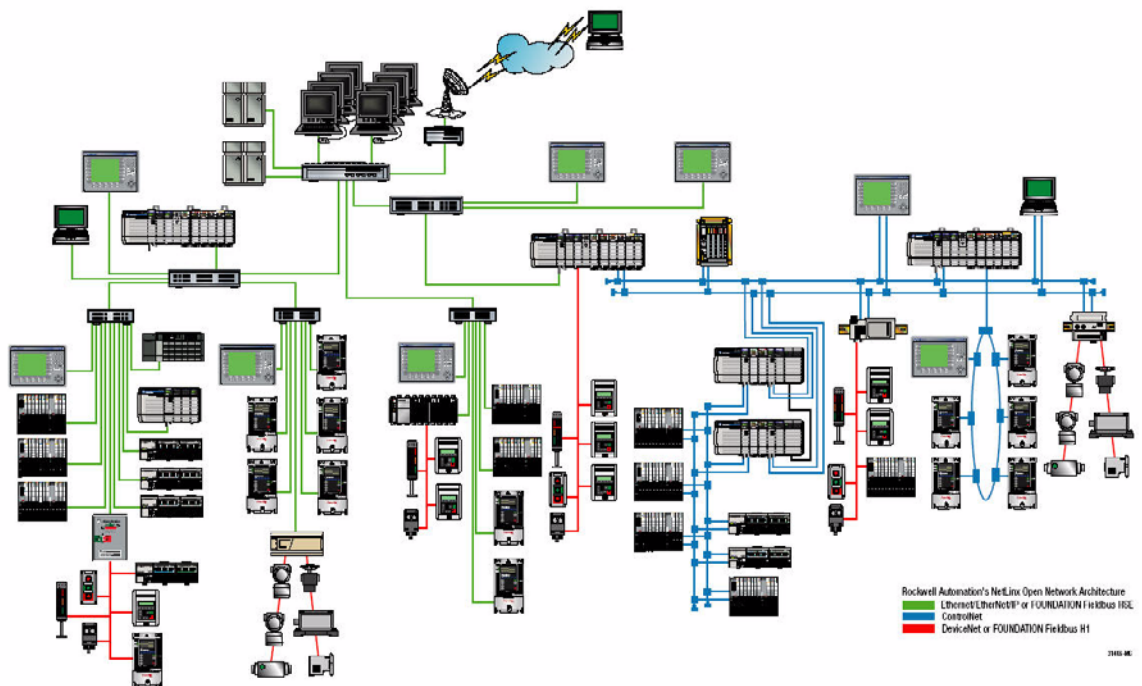
## Interconnecting NetLinx Networks

There are two common ways to interconnect NetLinx networks.

- Communication backplane, allowing multiple network links at once.
- Communication linking devices, linking two networks together in a seamless fashion.

Neither any controller nor any programming is required in either of these approaches.

### Example of a Control System Based on the NetLinx Networks



## Conversion of HMI

Refer to [Appendix B](#).

## Conversion of Systems Containing Distributed Controllers

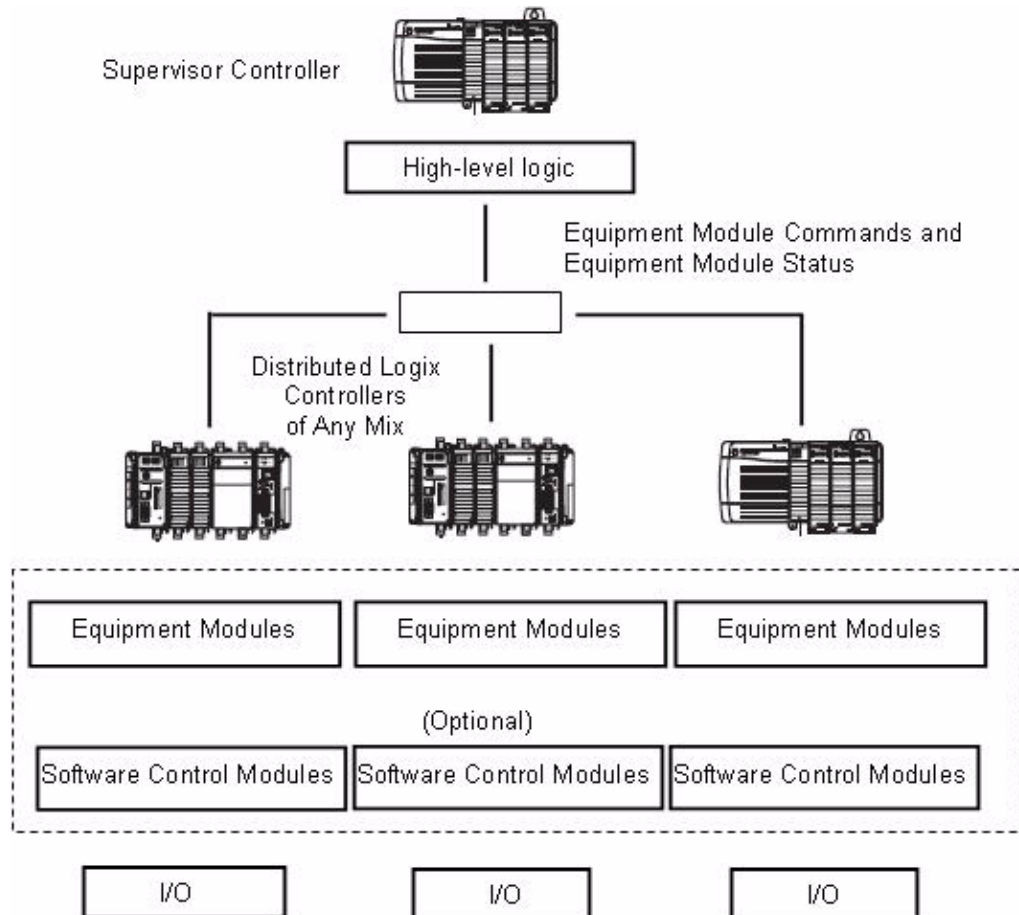
This section covers:

- how a general discrete control application containing a group of functional units can be built using multiple controllers.
- how a similar method can be applied to a process control application that is designed to the S88 standard.

### Hardware and Software Implementation

#### *General Discrete Control*

The hardware and software model for distributed logic for general discrete control is shown below. In this case the supervisory role will be played by a controller. EtherNet/IP or ControlNet network can be used to interconnect the controllers. Produce-Consume or explicit messaging can be used to exchange data within the system.

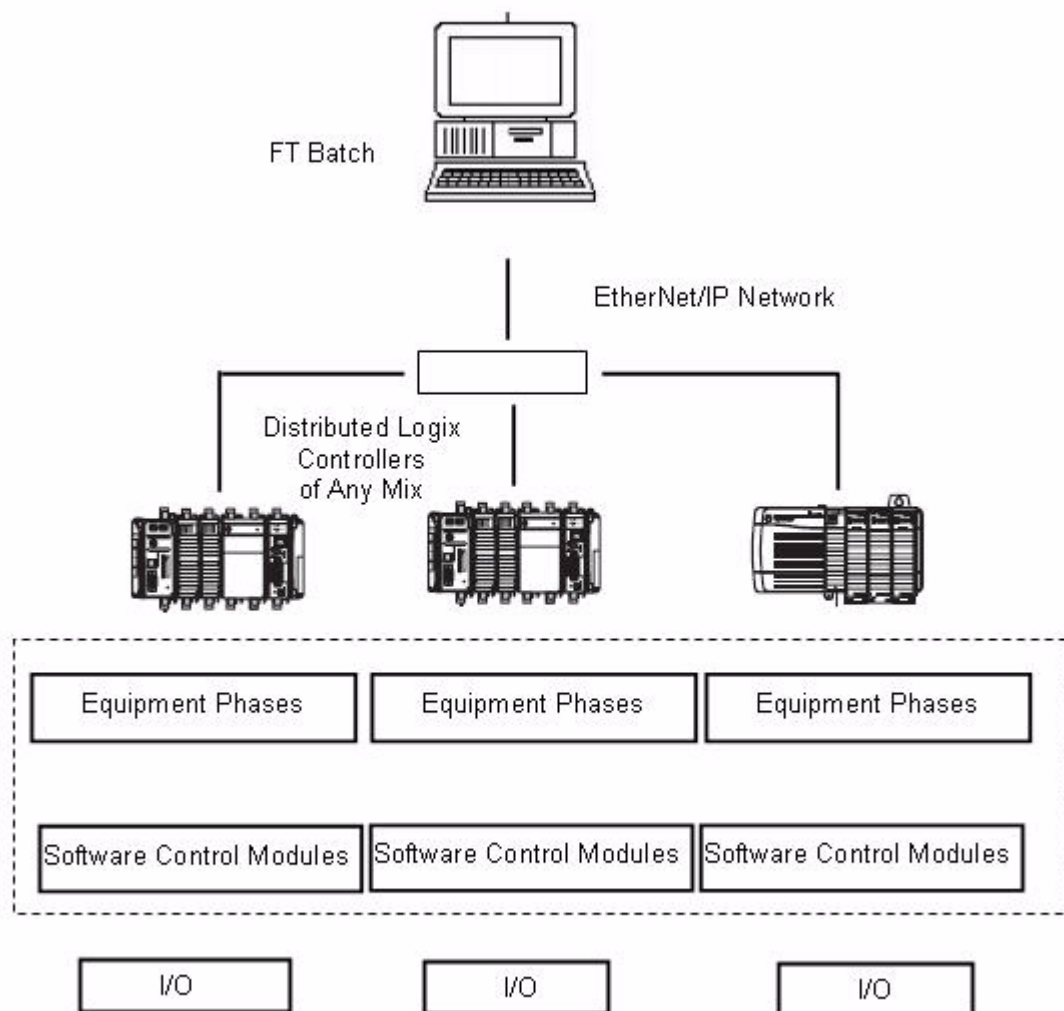




### Process Control

The diagram below illustrates the hardware and software structure for a S88 process control application. The PC will be running FactoryTalk Batch software, which is a software package for running production batches by means of recipes. FactoryTalk Batch software resides in a PC and communicates with each controller via the EtherNet/IP network.

Equipment phases are configured under PhaseManager as described later on in [Chapter 2](#). They execute the phase logic and communicate with the control system I/O via control modules.



## Connecting Siemens and Rockwell Automation Devices

There are situations in which you need to interconnect Siemens and Rockwell Automation equipment. We recommend that you use products from partnering companies grouped in the Encompass program.

### Controllers

Logix controllers can be connected to S7 networks by using:

- in-rack modules.
- standalone communication gateways.

### Distributed Devices

Some of Rockwell Automation's range of I/O systems, PowerFlex drives and HMI terminals connects to Profibus via communication adapters, built-in interfaces or interface modules.

## Logix Features that May Not be Familiar to S7 Users

### Introduction

This chapter describes Logix features that may not be familiar to S7 users.

| Topic  | Page |
|--|------|
| S7 Organization Blocks Compared to Logix Tasks | 36   |
| Tags Not Addresses                             | 47   |
| I/O and Alias Tags                             | 51   |
| Programming Languages                          | 53   |
| Add-On Instructions                            | 57   |
| The Common Industrial Protocol (CIP)           | 58   |
| Data Exchange between Controllers              | 60   |
| User-Defined Data Types                        | 61   |
| Asynchronous I/O Updating                      | 62   |
| The DINT Data Type                             | 62   |
| Phase Manager                                  | 63   |
| Coordinated System Time (CST)                  | 65   |
| Timestamped Inputs                             | 65   |
| Scheduled Outputs                              | 65   |
| No Temporary Variables                         | 66   |
| No Accumulators or Special Registers needed    | 66   |

Certain features of the Logix system are easier to use and maintain than S7 – for instance, data is organized into tag databases with no absolute addresses, whereas in S7 data items have absolute addresses that are selected by the programmer in defined memory areas.

In other respects, the structure of Logix is quite similar to S7, but it is presented differently – for instance beneath the surface the Task structure is similar to S7's Organization Blocks.

This Chapter contrasts those features that are different (such as tags) and compares those features that have underlying similarities (such as tasks).

The objective is to:

- provide the S7 user converting to Logix with information that will make the design process easier and quicker.
- show what Logix can do so engineers do not attempt to re-create what exists within controller firmware.

## **S7 Organization Blocks Compared to Logix Tasks**

This comparison of organization blocks and tasks will introduce the structure of a Logix program to the S7 user.

Organization blocks and tasks are similar in that both are called by the controller's operating system, rather than by the user program. In STEP 7 (and Logix), there are three types of organization block (task in Logix).

- Program Cycle OB (organization Continuous Task in Logix) where the OB re-commences from the beginning when it has finished.
- Cyclic Interrupt OB (Periodic Task in Logix) where the OB executes at a pre-configured time period.
- Hardware Interrupt OBs (Event Task in Logix) execute in response to some hardware stimulus.

Many STEP 7 programmers do not use Cyclic Interrupt OBs.

Logix provides a user configurable multi-tasking operating system which allows the CPU power to allocate as required by the application.

### **Organization Blocks in S7**

The type of OB is defined by its number – they are continuously executed (OB1 only), periodically executed (OB30 – OB38), they can be executed on events (OB40 – OB47) or they can execute on certain faults arising. With Logix, tasks are not numbered but are identified by a user-defined name.

A meaningful name can be attached to a STEP 7 OB if required.

### *OB1 Program Cycle*

OB1 cycles continuously. When it has finished executing, the output image table values are sent to the outputs, the input image table is updated from the outputs and OB1 starts again.

A STEP 7 program doesn't have to include OB1, but if it is included, it will be continuous.

#### **Typical OB1 Fragment:**

**Network 3** : Title:

```
callup valve and motor control module
```

```
CALL "ValveMotor_Calls"
```

**Network 4** : Title:

```
callup switch control module
```

```
CALL "Switch_Calls"
```

**Network 5** : Title:

```
callup flow totalisers control module
```

```
CALL "Totaliser_calls"
```

**Network 6** : Title:

```
callup analogue input control module
```

```
CALL "AnalogueIn_calls"
```

OB1 is the root of the call hierarchy for all continuously executed code.

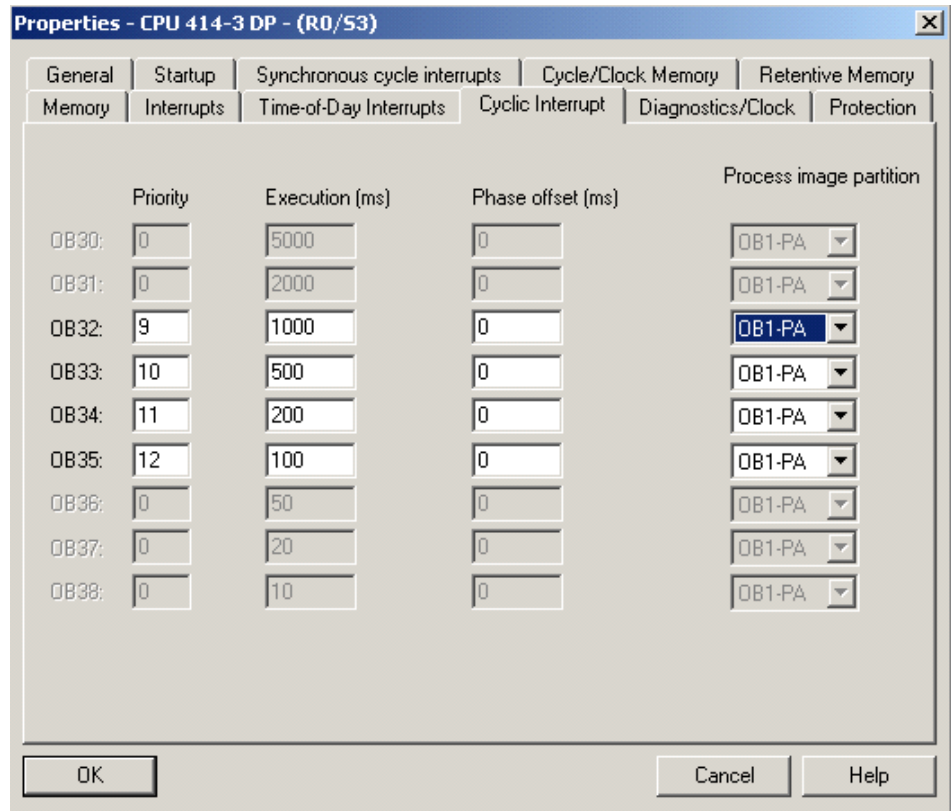
OB1 resembles the (only one possible of course) **continuous task** in Logix. In S7 terminology, OB1 is described as "Program Cycle".

For readers who are more familiar with Logix than STEP 7, it may be useful to know that in STEP 7 ladder logic, a network is the same as a Logix rung. In STEP 7 statement list, the networks are still there but they only serve to improve the code's appearance. They break up the code into sections and enable comments to be added. All of the code could be placed in one network if desired – it would compile and run perfectly well.

*OB30 – OB38 Cyclic Interrupts*

These OBs execute at fixed, configurable intervals. Their priority may also be configured. Higher priority OBs will interrupt any lower priority ones that are running.

**How Periodically Called OBs are Configured**



The number of periodic OBs available depends on the type of controller. Lower priority number represents higher interrupt priority (priority selection is only available with S7 400 controllers). Execution (ms) is the execution period for the OB. Phase offset allows phasing the triggering of periodic interrupts relative to each another. The process image partition selection allows the I/O image table to be partitioned and that partition only updated when the interrupt occurs (this feature is available with S7 400 controllers only). Default is the full table. In Logix, see the task I/O update selection and IOT commands.

The content of a periodic interrupt OB typically resembles the content of OB1. It will consist of calls to functions and function blocks that are to be executed at the periodicity of the OB.

These OBs resemble **periodic tasks** in Logix. In S7 terminology OB30 – OB38 are called Cyclic Interrupt OBs.

#### *OB40 – OB47 Hardware Interrupt OBs*

These OBs may be configured to trigger on an input event. Their priority may also be configured.

These are **event tasks** in Logix. In S7 terminology OB40 – OB47 are called Hardware Interrupts.

For example, the most simple hardware event that could be handled by a Hardware Interrupt OB (or Event Task) is a change of state of a digital input. A Hardware Interrupt (or Event Task) would guarantee a very fast response to the change.

Event tasks are more flexible than Hardware Interrupt OBs, with triggers not only from I/O, but also from network events, program instructions and motion events.

### *Program Structure in STEP 7*

A typical program includes Organization Blocks (OB), Function Blocks (FB), Functions (FC) and Data Blocks (DB). System Function Blocks (SFB) and System Functions (SFC) will usually be present.

- From Organization Blocks (Program Cycle or Cyclic Interrupt or both), calls are made to Function Blocks and Functions.
- A Function Block contains code, and is associated with a Data Block that contains the static data the FB requires. In addition to static data, the FB has temporary data. FBs are used when the logic must preserve values between executions.
- A Function contains code but has no static data. It has temporary data. FCs are used when the logic completes on a single execution – it has no need to preserve values.
- Data Blocks are areas for storage of static data. They will be described in the next section.
- SFBs and SFCs are System Function Blocks and System Functions. They can be copied from libraries that are included with a STEP 7 installation and placed in a project.
- When this has been done, they can be called from anywhere in the program.

In STEP 7 there isn't an equivalent structure to Logix's Program/Routine. The OB will be the root of the call chain to FBs and FCs, but how that is done is up to the programmer.



## Tasks in Logix

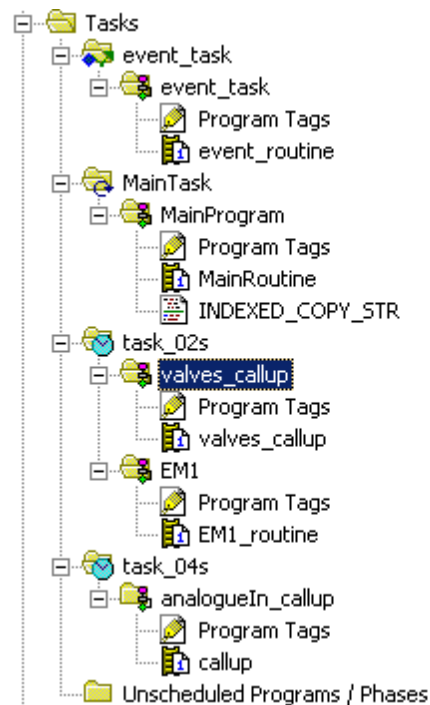
Tasks are called by the operating system. A task provides scheduling and priority for one or more programs. Each Program contains a data section and one or more code routines.

The tasks may be periodic, event or continuous. Each task may be assigned a priority. The continuous task, if present, is always of the lowest priority.

A Logix project will have one task whose default name is MainTask. This task can be continuous, periodic or event. You can change its name if you wish.

### *Task and Program Structure in Logix*

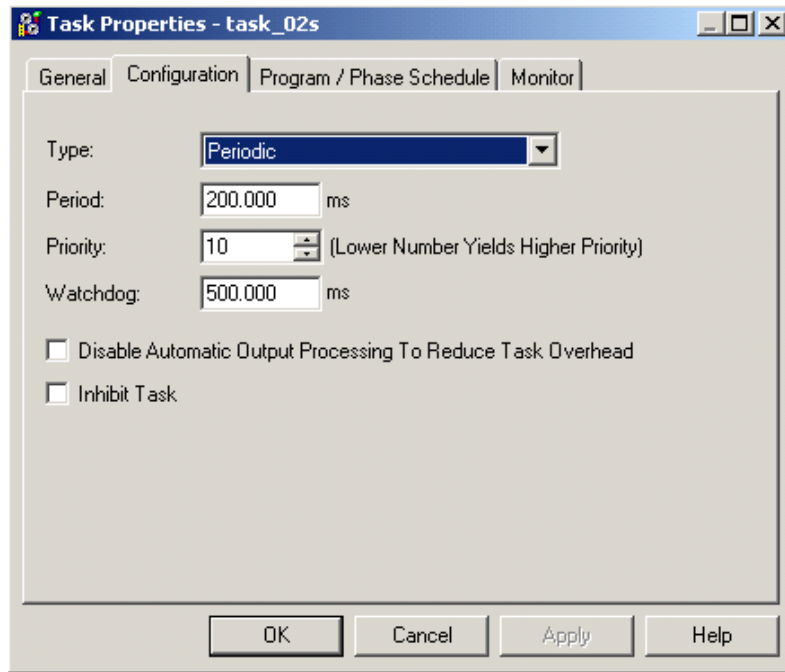
This snapshot from a sample RSLogix 5000 project tree helps illustrate how tasks and programs are structured.



In the screen shot above, the icon to the left of “event\_task” signifies an event task. The icon to the left of “MainTask” signifies a continuous task and the icon to the left of “task\_02s” signifies a periodic task.

### *Periodic Tasks*

Periodic tasks will trigger at a constant configured interval. Configuration of the period and priority is shown below.



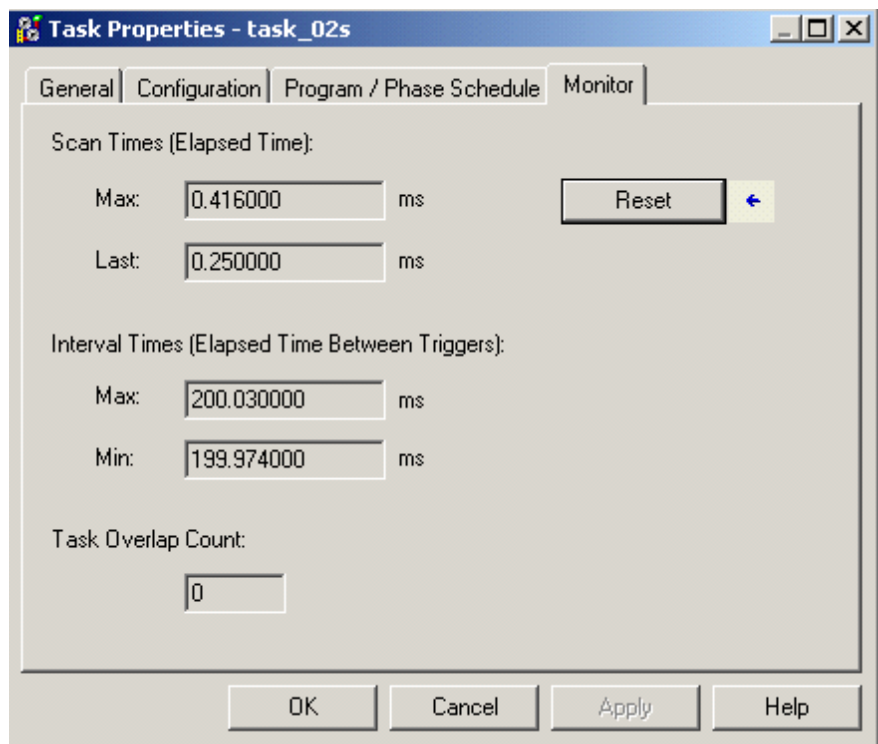
Configuration is similar to the OB30 – OB38 configuration page that was described in the section “OB30 – OB38 Cyclic Interrupts”.

### *Scheduling of Periodic Tasks*

The purpose of the Task system is:

- to allow the programmer to choose appropriate frequencies for the execution of Programs. By executing code no more frequently than is needed, the controller CPUs power is used more efficiently for application priorities.
- to use the priority system to allow critical tasks to interrupt lower priority ones, therefore giving them a better chance of executing at the intended frequency.

It is easy to check these times from Task Properties /Monitor.



What will happen if a trigger occurs while a task is running?

- If the new trigger is for a task with a higher priority than the one running, the running task will be interrupted by the new one, and will resume when the higher priority task is complete.
- If the new trigger is for a task with a lower priority than the one running, the running task will continue, and the new task will wait until no task of a higher priority is running.
- If the new trigger is for a task with a same priority as the one running, the controller will run both tasks by switching between them at 1ms intervals.
- If the new trigger is for the same task as the one that is running, the new trigger will be discarded. This is an **overlap** condition.

The number of overlaps that occurred since the counter was last reset is shown in the task properties window. A non-zero number indicates that the interrupt period needs to be increased.

**TIP**

Avoid switching tasks unnecessarily, due to the amount of processing power that is wasted during unnecessary switching.

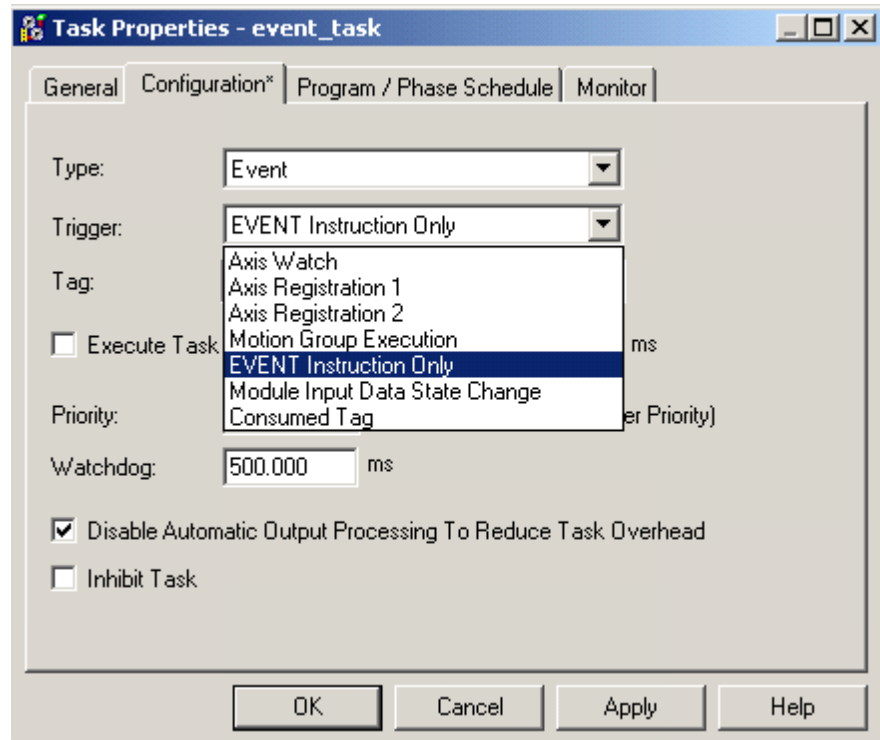
When you program periodic interrupts in Logix, note these similarities and differences with STEP 7:

- In STEP 7, calls will be made from the OB that is configured to execute at the chosen frequency to the Functions and Function Blocks you wish to execute at this frequency. In Logix, you will insert programs and routines in the project tree under the Task.
- In both STEP 7 and Logix, the actual application code will not differ greatly from the code in a continuous execution task. Note that the constant and known frequency of a periodic task gives programmers the opportunity to turn a simple variable increment into a timer.
- In both systems, you will need to check for overlaps as you develop and test your code. The execution time of the OB or Task must be much less than its execution period.
- Checking the execution time for Logix tasks is easy. Use the task properties screen shown above. In STEP 7 you will need to sample the system clock at the start and end of the OB, subtract the values and store the result in a variable for monitoring.
- In a S7 controller, overlaps will cause the controller to stop unless a fault OB is added that traps the fault. Logix is less strict and merely counts the number of overlaps.

- In STEP 7, it is possible to phase the execution of periodic OBs relative to one another. This is not available with Logix Tasks.

### *Event Tasks*

Event Tasks will execute when a configured trigger event occurs. Normally they would be given higher priority than periodic tasks.



An event task is configured by opening the task's Task Properties page and selecting Type Event. Different types of event task triggers can be used for different Logix controllers.

### Continuous Task

A Logix controller supports one continuous task, but a project doesn't have to include the continuous task. You can if you wish run your entire program under Periodic and Event Tasks.

You can configure whether the continuous task updates outputs at the end of its execution.

You can if you wish adjust the percentage of your CPU's time that is spent on unscheduled communication as a percentage of time dedicated to the continuous task.

### Task Monitor

RSLogix 5000 software includes a tool called Task Monitor that can help with analyzing scheduled tasks, and much more.

The screen shot below shows how information about your controller's tasks can be viewed in one table.

The screenshot shows the 'Logix5000 Task Monitor' window with the 'User Tasks' tab selected. The table below displays the following data:

| Name     | Rate        | CPU   | Priority | Last Scan | Max Scan | Watchdog  | Overlap | Stat  |
|----------|-------------|-------|----------|-----------|----------|-----------|---------|-------|
| MainTask | * 500,000us | 0.70% | Lowest   | 3,524us   | 5,852us  | 500,000us | 0       | Runr  |
| task_02s | 200,000us   | 0.16% | 9        | 548us     | 894us    | 500,000us | 0       | Runr  |
| task_04s | 400,000us   | 0.04% | 10       | 764us     | 1,182us  | 500,000us | 0       | Runr  |
| task_01s | 100,000us   | 0.17% | 5        | 134us     | 426us    | 500,000us | 0       | Runr  |
| event    | 10,000us    | 0.00% | 10       | 0         | 0        | 500,000us | 0       | Stopp |

The other tabs provide a wealth of other system-level information on your controller's performance. The tool is included as standard on the RSLogix 5000 installation disk.

## Tags Not Addresses

One of the first major differences that a S7 user will notice when starting to work with Logix is that data doesn't have addresses. Data items are created in a tag database, and RSLogix 5000 software allocates addresses “behind the scenes”. This makes it unnecessary for users to understand and manage memory addresses. This section describes data allocation in the two systems.

## Data Areas in S7

### Data Areas in S7 Controllers

| Address Area               | S7 Notation | Unit Size          |
|----------------------------|-------------|--------------------|
| Process Image Input Table  | I           | Input Bit          |
|                            | IB          | Input Byte         |
|                            | IW          | Input Word         |
|                            | ID          | Input Double Word  |
| Process Image Output Table | Q           | Output Bit         |
|                            | QB          | Output Byte        |
|                            | QW          | Output Word        |
|                            | QD          | Output Double Word |
| Bit Memory                 | M           | Memory Bit         |
|                            | MB          | Memory Byte        |
|                            | MW          | Memory Word        |
|                            | MD          | Memory Double Word |
| Timers                     | T           |                    |
| Counters                   | C           |                    |
| Data Block                 | DBX         | Data Bit           |
|                            | DBB         | Data Byte          |
|                            | DBW         | Data Word          |

The sections below say more about the two most commonly used areas in programming – bit memory and data blocks.

### Bit Memory

“Bit Memory” locations are denoted Mx where, for example:

- **M5.3** is a bit.
- **MB6** is a byte (BYTE).
- **MW8** is a 16 bit word (WORD).
- **MD10** is a 32 bit word (DWORD).

Bit memory locations can be labelled in the Symbol Table (similar to a PLC-5 or SLC Symbol Table), as shown in the following screen shot.

| Status | Symbol ▲             | Address | Data type | Comment                                     |
|--------|----------------------|---------|-----------|---|
|        | EXT_ZONE2_ON         | Q 28.2  | BOOL      | EXTRUDER_ZONE2 ON                           |
|        | EXT_ZONE3_ON         | Q 28.3  | BOOL      | EXTRUDER_ZONE3 ON                           |
|        | EXT_ZONE4_ON         | Q 28.4  | BOOL      | EXTRUDER_ZONE4 ON                           |
|        | EXT_ZONE5_ON         | Q 28.5  | BOOL      | EXTRUDER_ZONE5 ON                           |
|        | FALSE                | M 0.0   | BOOL      |   |
|        | FlowTotaliser        | UDT 10  | UDT 10    |   |
|        | FSL24001             | I 10.0  | BOOL      |   |
|        | FSL24002             | I 41.0  | BOOL      |   |
|        | FSL24003             | I 10.6  | BOOL      |   |
|        | FT24001              | PIW 530 | INT       |   |
|        | FT24002              | PIW 526 | INT       |   |
|        | FT24006              | PIW 570 | INT       |   |
|        | GET_INDEXED_REFE...  | FC 111  | FC 111    |   |
|        | Global_Data          | DB 99   | DB 99     |   |
|        | IFIX_alarms          | DB 71   | DB 71     |   |
|        | INDEXED_COMPARE      | FC 102  | FC 102    |   |
|        | INDEXED_COPY         | FC 101  | FC 101    |   |
|        | Interlocks_Handler   | FB 70   | FB 70     |   |
|        | Interrupt_Execution  | OB 35   | OB 35     |   |
|        | JUNK_BIT             | M 0.2   | BOOL      |   |
|        | KTRON_CmdsFromM...   | DB 33   | DB 33     | Commands & Setpoints From Manufacturing PLC |
|        | KTRON_StatusToMan... | DB 32   | DB 32     | Feeder Status to ManufacturingPLC           |
|        | LF24001A             | Q 17.0  | BOOL      | PIG LAUNCH VALVE                            |
|        | LF24002A             | Q 18.5  | BOOL      | PIG LAUNCHER                                |
|        | LF24003A             | Q 10.4  | BOOL      | PIG LAUNCHER                                |



### Data Blocks

Data Blocks have similar status to other blocks – Organization Blocks, Function Blocks and Functions – except that they contain data rather than program code. The memory in Data Blocks is static – the data retains its value until it is changed.

#### Example of a Data Block

| Address | Name                    | Type   | Initial value | Comments |
|---------|-------------------------|--------|---------------|----------|
| 0.0     |                         | STRUCT |               |          |
| +0.0    | GrantrezSalts_SP        | REAL   | 0.000000e+000 | Grant    |
| +4.0    | GrantrezSalts_FeedFact  | REAL   | 0.000000e+000 | Grant    |
| +8.0    | CMC_SP                  | REAL   | 0.000000e+000 | CMC S    |
| +12.0   | CMC_FeedFact1           | REAL   | 0.000000e+000 | CMC I    |
| +16.0   | SiliconDioxide_SP       | REAL   | 0.000000e+000 | Silic    |
| +20.0   | SiliconDioxide_FeedFact | REAL   | 0.000000e+000 | Silic    |
| +24.0   | MaleicAcid_SP           | REAL   | 0.000000e+000 | Malei    |
| +28.0   | MaleicAcid_FeedFact     | REAL   | 0.000000e+000 | Malei    |
| +32.0   | Sequence_Cmnds          | STRUCT |               |          |
| +0.0    | PrimeMaterials          | BOOL   | FALSE         | Loads    |
| +0.1    | StartProduction         | BOOL   | FALSE         | Start    |
| +0.2    | EndProduction           | BOOL   | FALSE         | EndPr    |
| +0.3    | Stop                    | BOOL   | FALSE         | Stop     |
| +0.4    | FaultAck                | BOOL   | FALSE         | Ackno    |
| +0.5    | Spare1                  | BOOL   | FALSE         |          |

Data Block symbols do not appear in the Symbol Table, but the name of the Data Block does.

Data Blocks can be assigned to hold the data used by Function Blocks. These are called Instance Data Blocks.

## Data in Logix

In the RSLogix 5000 programming environment, data is set up in a tag database. Memory addresses are hidden from the programmer, which makes things easier for the programmer.

### Tag Database

| Controller Tags - dev_18Aug(controller) |                  |                  |                          |          |                     |  |  |
|---|------------------|------------------|--------------------------|----------|---------------------|--|--|
| Scope: dev_18Aug                        |                  | Show...          |                          | Show All |                     |  |  |
| Name                                    | Alias For        | Base Tag         | Data Type                | Style    | Description         |  |  |
| + analogIn_1                            |                  |                  | DINT                     | Decimal  |                     |  |  |
| Blue_Button                             | Local:3:I.Data.0 | Local:3:I.Data.0 | BOOL                     | Decimal  |                     |  |  |
| + CompactLogix_1_consume                |                  |                  | UDT_STEP_SEQUENCE        |          | Data - step sequ... |  |  |
| + ControlLogix_1_produce                |                  |                  | UDT_STEP_SEQUENCE        |          | Data - step sequ... |  |  |
| + Drive:I                               |                  |                  | AB:PowerFlex70EC_Driv... |          |                     |  |  |

### Select a Tag from a Pull-down Menu While Programming

The screenshot shows a software interface for programming a controller. A dialog box titled 'Controller' is open, displaying a list of tags with columns for Name, Data Type, and Description. The tags listed are em1Data.step[0] through em1Data.step[4], all of type BOOL. A pull-down menu is open over the list, showing 'em1Data.step[0]' selected. Below the dialog box, a ladder logic diagram is visible. It shows a 'JMP' instruction with the label 'STEP 1 /Jump to step 3'. The instruction is connected to a terminal labeled 'end\_seq' and another terminal labeled 'em1Data'. The 'em1Data' terminal is connected to a coil labeled 'em1Data'.

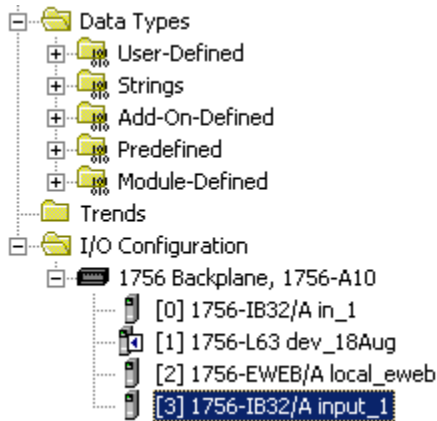
In Logix, there is a **controller-scope tag database** and **program-scope tag databases** associated with each Program.

- Tags in the controller-scope database are global and can be accessed by routines in any part of the program.
- Program-scope tags can only be accessed by routines in that Program.

## I/O and Alias Tags

An alias tag lets you represent another tag, while both tags share the same value. One of the purposes of aliases is to reference the I/O tags as described below.

I/O modules can be added to a project by adding the module to the controller backplane in the project folder.

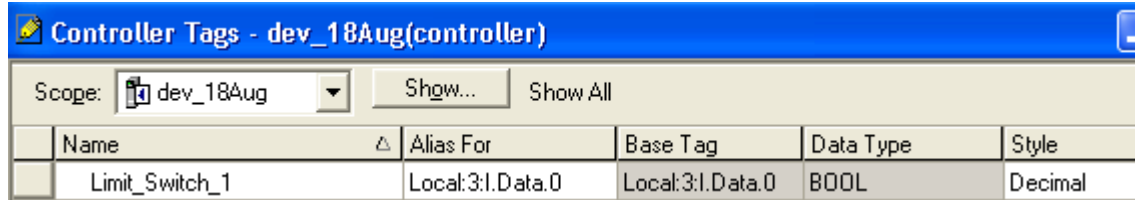


In this case a 32-point input card has been added at slot 3. The slot number is in square brackets at the beginning of the line. “1756-IB32/A” is the part number of the card. “input\_1” is a name for the card that is configured when the card is first added to the rack.

Having added the card, RSLogix 5000 software will automatically generate the relevant device profile tags to the Controller-scope tag database. They are the Local:3:I input and Local:3:C configuration tags below.

| Controller Tags - dev_18Aug(controller) |           |          |                |          |  |
|---|-----------|----------|----------------|----------|--|
| Scope:  dev_18Aug                       |           | Show...  |                | Show All |  |
| Name                                    | Alias For | Base Tag | Data Type      | Style    |  |
| [-] Local:3:I                           |           |          | AB:1756_DI:I:0 |          |  |
| [-] Local:3:I.Fault                     |           |          | DINT           | Binary   |  |
| [-] Local:3:I.Data                      |           |          | DINT           | Binary   |  |
| Local:3:I.Data.0                        |           |          | BOOL           | Decimal  |  |
| Local:3:I.Data.1                        |           |          | BOOL           | Decimal  |  |
| Local:3:I.Data.2                        |           |          | BOOL           | Decimal  |  |

You can create a new alias tag with a more descriptive name. For instance, an alias for the first input can be created called Limit\_Switch\_1, which physically describes this input.



| Name           | Alias For        | Base Tag         | Data Type | Style   |
|----------------|------------------|------------------|-----------|---------|
| Limit_Switch_1 | Local:3:I.Data.0 | Local:3:I.Data.0 | BOOL      | Decimal |

In STEP 7, the hardware configuration tool will assign addresses to an I/O card when it is added to the system. For example, a digital input card might be assigned bytes I16 and I17. Then the programmer will identify the bit address for each input and enter a name against it in the symbol table. Once that is done, the program will automatically make the association I16.5 = “ZSC2036”.

## Programming Languages

This section describes the programming languages that are available with STEP 7 and RSLogix 5000 software. All languages are not standard; it depends upon the version of the software purchased. Selection of the Logix language most suitable to the task will result in easier program design, more rapid coding and a program that is easier to understand.

There is one significant difference between the S7 and Logix languages. In S7, Statement List is the “native” language of the controller. Other languages are translated to STL. In Logix, all the languages are “native” languages in the controller – each is compiled without reference to any of the others. The benefit of this is that when you upload a program from the controller, you view it in the language it was written in.

**STEP 7** has three standard languages:

- Statement List (STL) – could be described as high-level assembler.
- Ladder Logic (LAD)
- Function Block Diagram (FBD)

And some optional languages:

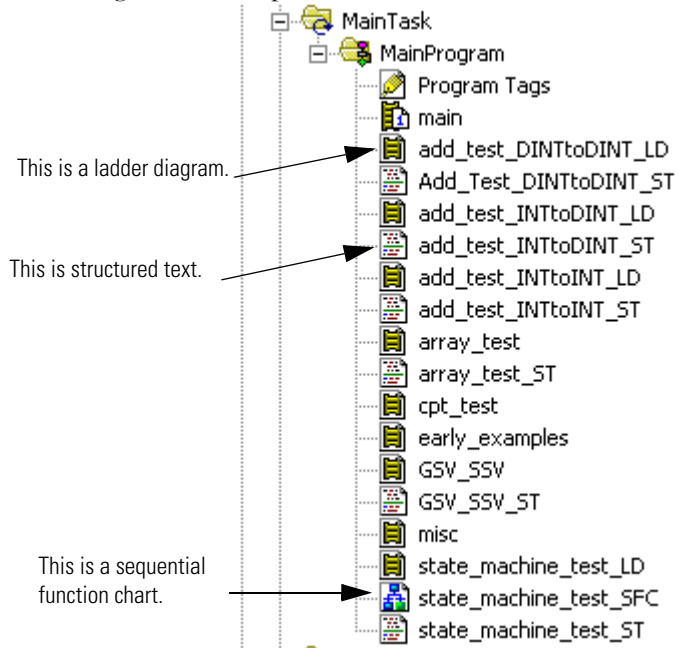
- Structured Text (ST)
- CFC – Continuous Flow Chart for process-type applications
- HiGraph – Sequential control via Graphing software
- ML – Motion Language – similar to GML in the older Rockwell Automation dedicated 1394 motion controller

A program can consist of Function Blocks and Functions written in different languages.

RSLogix 5000 software has four programming languages:

- Ladder Diagram (LD) – comparable to Siemens LD, with an expanded instruction set.
- Structured Text (ST) - Equivalent to Siemens ST
- Function Block Diagram (FBD) – Equivalent to Siemens CFC
- Sequential Function Chart (SFC) – Comparable to Siemens hiGraph.

A Routine – the basic section of code in Logix – can be in any of these, and a program can be made of routines written in different languages. The following screen shot gives an example.



## Logix Ladder Diagram

Traditionally, Ladder Diagram is used for implementing Boolean combinational logic. In Logix, it can also be used for sequential logic, motion, data manipulation and mathematical calculations, although other languages may be more convenient for these tasks.

## Logix Structured Text

Structured Text is a high level procedural language that will be easy to learn for anyone with experience in Basic, Pascal or one of the 'C' family of languages. It is used primarily for data manipulation and mathematical calculations, although motion, combinational, and sequential logic can also be easily programmed in ST.

## Logix Function Block Diagram

Function Block Diagram describes graphically a function (Boolean or mathematical) relating input variables and output variables. Input and output variables are connected to blocks by connection lines. An output of a block may also be connected to an input of another block.

It is good practice to program PID loops in FBD. It is the most convenient language for process control.

## Logix Sequential Function Chart

SFC is a graphical tool for describing sequential logic as a set of states and transitions. Outputs may be assigned to a state, and Boolean conditions for transitions to other states defined.

## Conversion of STEP 7 Code to Logix

- If you have STEP 7 ladder logic code that you wish to convert it to Logix, LD should be your first choice. The meaning of LD is similar in both systems.
- If you have STEP 7 function block diagram code that you wish to convert it to Logix, FBD should be your first choice.
- Note that the standard Logix FBD is more advanced than STEP 7 FBD, and is equivalent to the optional STEP 7 language CFC.
- If you have STEP 7 Statement List code that you wish to convert to Logix, the most suitable language will depend on the nature of the STL block. If the STL block contains mainly Boolean evaluations, LD would probably be the best Logix language to convert to. If the STL block contains pointers to access and manipulate data, or executes mathematical calculations, ST would probably be the best Logix language to convert to. If the STL block contains sequential logic, SFC should be considered, although sequential logic can also be easily implemented in ST and LD.

## Arrays not Pointers

In STEP 7, arrays can be defined exactly as they would be in Pascal or C, but the basic languages (STL, LD and FBD) do not have high-level support for accessing them. Instead, pointer routines must be constructed.

STEP 7 library functions lack support for array access. Programmers who are comfortable with pointers can write their own functions such as FC101 “INDEXED\_COPY” (see below) but it requires skill and time.

“INDEXED\_COPY” in STEP 7 does the same as the Logix instruction COP for indexed copying.

```
CALL "INDEXED_COPY"          FC101
  indexSrc:=#index_in
  source  :="Instance_FB2".table  P#DB4.DBX0.0
  indexDst:=1
  dest    :="Instance_FB2".target P#DB4.DBX96.0
  len     :=8
```

FC111 below will access an array.

```
CALL "GET_INDEXED_REFERENCE" FC111
  refArray :="Instance_FB2".table P#DB4.DBX0.0
  index    :=#index_in
  byteIncr :=32
  startIndex:=TRUE
  retVal   :=#ptr
```

The pointer to the object is returned in parameter #ptr, which can then be dereferenced to get the data.

In Logix, arrays can be both defined and accessed in the usual way of a high-level computer language, as the fragment below illustrates.

```
// copy a string from a table of strings #table
// to a target string #target. The index is #index_in
COP(table[index_in], target, target.LEN);
```



## Add-On Instructions

## Add-On Instruction Summary

Add-on Instructions are the equivalent of STEP 7 Function Blocks, with private data and advanced parameter choices. In particular, the INOUT parameter type or “pass by reference” makes it possible to efficiently pass data structures to the code.

Because the Add-On Instruction is so similar to the STEP 7 Function Block, it is likely that the S7 programmer who is converting to Logix will make use of it quite readily.

Comparison between FBs and Add-On Instructions:

- Both can be called as named functions from anywhere in the program.
- Both contain a private data area of static data, although it is not truly private in the case of STEP 7.
- A STEP 7 function block also has a temporary data area.
- In the Add-On Instruction, local static data will do the same.

Both have three types of parameters – input (pass by value), output (pass by value) and in-out (pass by reference). The pass by reference parameter is a considerable benefit, since it allows large data structures to be passed efficiently.

The Add-On Instruction will automatically maintain a change history by recording a timestamp and the Windows user name at the time of the change. This is not available with STEP 7 Function Blocks

With the Add-On Instruction a pre-scan routine can be configured to run when the controller goes from Program mode to Run mode, or powers up in Run mode. Under these conditions, the pre-scan routine will run once, and can typically be used to initialise data. In STEP 7 the Organization block OB100 does the same, but the pre-scan code cannot be specifically attached to a FB.

If the Add-On Instruction is called from a SFC step and the SFC is configured for Automatic Reset, a post-scan routine defined in the Add-On Instruction will execute once when the SFC exits that step. It could be used for resetting data. A STEP 7 FB has no built-in equivalent (although it is easy to program).

An Add-On Instruction can have an EnableInFalse routine, which will be called (if present) when the rung condition at the Add-On Instruction call is false. In this case, the input and output parameters will pass values. A STEP 7 FB has no equivalent.

Add-on Instructions are explored further in [Chapter 4](#).

## Backing Tags

Many instructions and data types use backing tags – tags that are created specifically for the instance of the instruction or data types that you are instantiating. Add-On Instructions, timers, counters, messages, and PID control all use backing tags. RSLogix 5000 software will generate the corresponding structure of elements for you anytime you create a tag of that type so you do not have to create the elements on your own.

| Name         | Alias For | Base Tag | Data Type | Style   |
|--------------|-----------|----------|-----------|---------|
| -Timer1      |           |          | TIMER     |         |
| + Timer1.PRE |           |          | DINT      | Decimal |
| + Timer1.ACC |           |          | DINT      | Decimal |
| Timer1.EN    |           |          | BOOL      | Decimal |
| Timer1.TT    |           |          | BOOL      | Decimal |
| Timer1.DN    |           |          | BOOL      | Decimal |

## The Common Industrial Protocol (CIP)

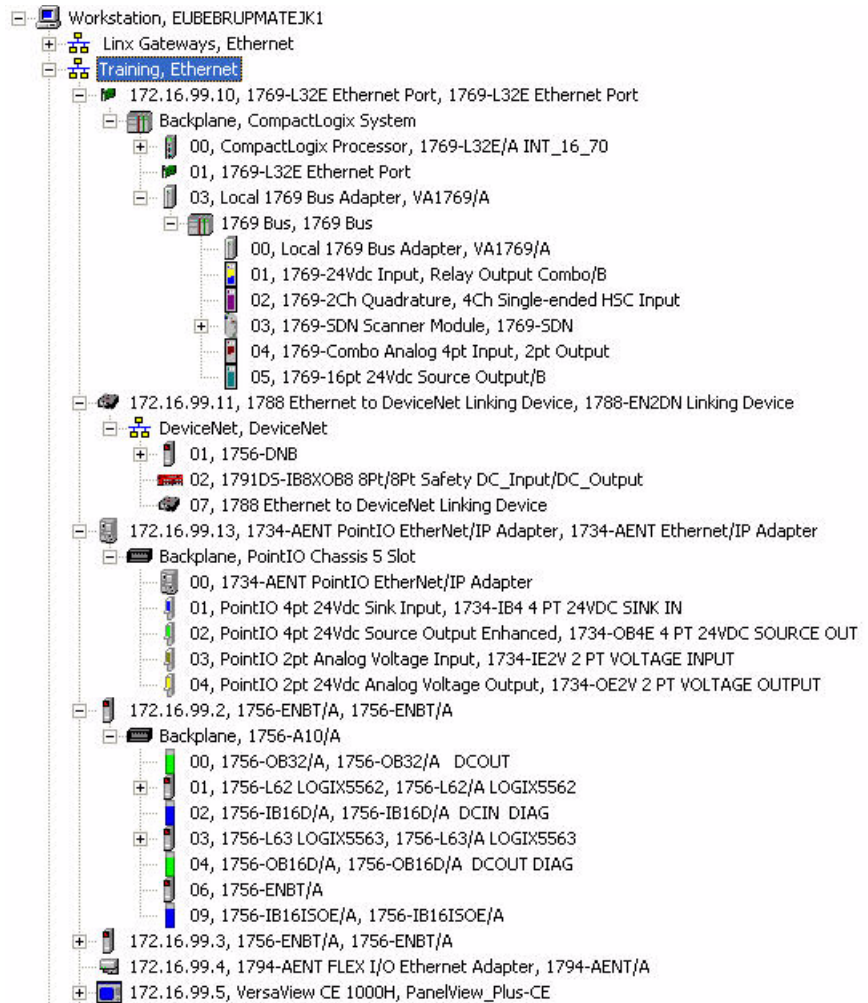
Logix uses three main networks - Ethernet/IP, ControlNet and DeviceNet. Each has characteristics suitable for different areas of application. The three network types share a protocol, the ‘Common Industrial Protocol’.

The CIP makes it possible to transfer data via any of the three types of networks supported by Logix with a nearly identical configuration and programming interface for all three. Also, data can be transferred through a network built from more than one of the three network types without any need for the programmer to translate protocols.

In “traditional” S7 the two main protocols are Industrial Ethernet, for networking to IT and to other controllers, and Profibus DP for networking to field systems. These two protocols are separate at the hardware level and the data level. With the latest S7 hardware and software, “Profinet CBA” integrates Industrial Ethernet, Profinet and Profibus.

## Viewing the Network

S7 users may find the Logix network configuration and management striking. As an example, the tree below shows the devices actually connected to the system. This tree was produced by going online – nothing was configured.



Networks are described further in [Chapter 1](#).

## Data Exchange between Controllers

### Send / Receive in STEP 7

To prepare controller to controller communication in STEP 7, these steps are taken.

1. The remote stations are configured graphically in a STEP 7 component called NetPro.
2. A connection table is built in NetPro specifying the protocols and parameters for each of the connections.
3. The library functions FC5 AG\_SEND and FC6 AG\_RECV are copied into the project.
4. Calls are made from the user program to AG\_SEND and AG\_RECV, specifying connection parameters and the data areas that are being used to source and receive the data.

### Produced / Consumed Tags in Logix

Produced and consumed tags are the way that critical data is transferred between networked Logix controllers every defined time period. Produced and consumed tags can transmit over Ethernet/IP or ControlNet and on the backplane of ControlLogix controllers.

Produced and consumed tags are tags that are configured as produced or consumed when they are created. If a tag is marked as produced, then its value will be multicast to a EtherNet/IP or ControlNet network that the controller is connected to. If it is marked as consumed, then the controller that the tag requires data from will be identified as part of the configuration, and the consumed tag will receive its value from the equivalent produced tag in that controller.

There are separate channels for send and receive. Changing the value of a consuming tag will have no effect on the producing tag. This resembles controller-to-controller communication in S7 and differs from controller-to-SCADA communication, where any change will be reflected at the other end.

No programming is required to set up produce/consume connections. This contrasts with S7, where some coding is needed for controller-to-controller (SEND/RECEIVE) communication.

## User-Defined Data Types

In Logix, User-Defined Data Types can be configured. This allows the structure of a complex data type to be declared as a type. Instances of that type can then be defined in the program.

Logix User-Defined Data Types have very similar configuration and usage to STEP 7 User-Defined Data Types.

### Logix UDT

Name:

Description: 

Ramps a real variable from its current value to a new value at a specified rate.

Members: Data Type Size: 28 byte(s)

|                          | Name           | Data Type | Style   | Description                   |
|--------------------------|----------------|-----------|---------|-------------------------------|
| <input type="checkbox"/> | initial_output | REAL      | Float   | saved initial output          |
| <input type="checkbox"/> | increment      | REAL      | Float   | calculated increment          |
| <input type="checkbox"/> | RAMP_RATE_ABS  | REAL      | Float   | per second - (set always +ve) |
| <input type="checkbox"/> | RAMP_TARGET    | REAL      | Float   | final value - (set)           |
| <input type="checkbox"/> | change         | REAL      | Float   | calculated change over ramp   |
| <input type="checkbox"/> | counter        | DINT      | Decimal | internal counter              |
| <input type="checkbox"/> | complete       | BOOL      | Decimal | ramping is complete           |
| <input type="checkbox"/> | _enable        | BOOL      | Decimal | for enable one shot           |
| <input type="checkbox"/> | enabled        | BOOL      | Decimal | ramper enabled                |
| <input type="checkbox"/> |                |           |         |                               |

## Asynchronous I/O Updating

In Logix systems, I/O is updated asynchronously with respect to program execution periods, in contrast with the traditional PLC approach as used in S7 where an I/O image table is updated at the start of the cycle and input values do not change during an execution of the program.

The Logix programmer will need to consider whether there is any need to buffer input data, so that its value remains constant during program execution.

It is quite common to “consume” inputs once only by passing them as parameters to a code module. The inputs will not be used anywhere else in the program. This removes any need for buffering. See the Control Module example in [Chapter 4](#).

## The DINT Data Type

Logix controllers operate on DINT (32 bit integer) tags more efficiently than on INT (16 bit integer) or SINT (8 bit integer). Use DINT whenever possible, even if the range of values you are working with would fit in an INT or a SINT. These data types are provided for IEC61131-3 compatibility reasons, but are internally converted to DINTS before being used by the program, so code will execute more efficiently in most situations.

## Phase Manager

### Phase Management in STEP 7

STEP 7 possesses no built-in tools to perform phase management. The necessary structures must be programmed in a set of routines, typically referred to as the PLI or Phase Logic Interface. The components for a PLI program based on S88 are:

- A step sequencer whose behavior complies with the S88 state model. Certain steps or ranges of steps define the S88 state. Sequencer commands are also as specified by S88, and the sequencer will respond only when the state model permits. A sequencer with these properties is called a phase.
- A set of data for each phase that is used to record the status of the phase and to receive incoming commands from the recipe manager. The recipe manager communicates with this data. The format of the data will depend on the recipe manager.
- A logic module that translates the phase status into the format required by the recipe manager, and translates commands from the recipe manager into phase commands.

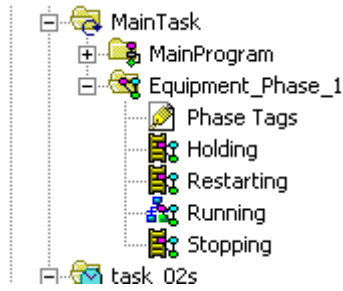
### PhaseManager in Logix

In an S88 Equipment Phase, there are specified states of the phase as well as the transitions between these states. The PhaseManager is a functionality of RSLogix 5000 software that allows you to do three things:

- Allocate the code for each phase state to a different routine.
- Run a state machine “behind the scenes” that handles the transitions between states of the phase.
- Manage the running of the phase using a set of Logix commands.

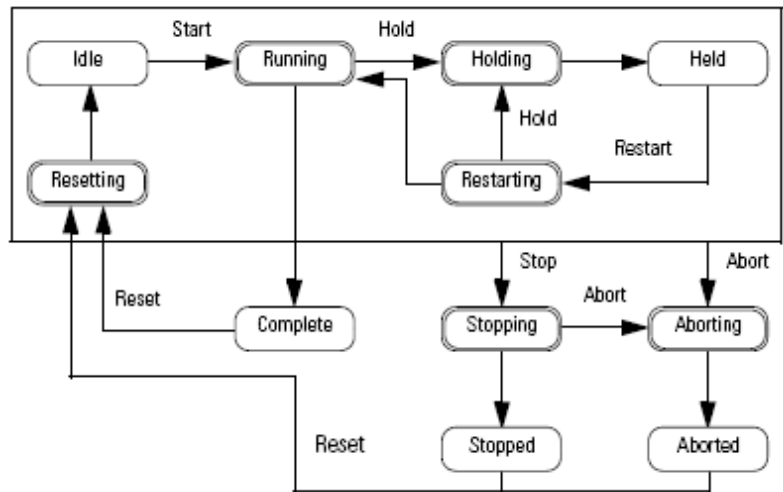
It is used in a variety of application spaces, including but not limited to Process Control and Packaging, because it allows for clean separation of Device/Equipment control and of Procedural control, hence making code far more modularized and efficient, especially for larger systems where standardization.

### Equipment Phase in the Project Tree



The code for each state of the phase can be written in any of the Logix languages.

This is the phase state machine. It is almost identical to the S88 state model.



If you have programmed a S88 compliant STEP 7 phase manager / PLI routine and wish to convert it to Logix, it may be possible to avoid translation by using the Logix PhaseManager.



## **Coordinated System Time (CST)**

S7 has a system clock, which is represented using 32 bits and counts in milliseconds. Its value can be obtained (and stored) by making a call to the operating system, which is useful for accurate measurement of time intervals.

Logix use Coordinated System Time which is a 64 bit number that measures the number of microseconds since the controller was last started. As with S7, intervals can be measured by making calls to the operating system to get the CST value. It provides the foundation for clock synchronization for multi-CPU systems, accurate motion control functionality, scheduled output switching to 100  $\mu$ s accurate, input event timestamping, scheduled analog sampling, safety I/O monitoring and communication, motion cam position calculations, and Wall Clock Time.

## **Timestamped Inputs**

Timestamp is a functionality that records a change in input data with a relative time of when that change occurred. With digital input modules, you can configure a time stamp for changes of data. You can use the CST timestamp to compare the relative time between data samples.

This allows the programmer to achieve unparalleled accuracy in linking input signals to time references for applications such as commonly used in motion control, without putting a huge burden on the communication and logic processing systems and related application code.

## **Scheduled Outputs**

With Digital Output Modules, you can configure the module to set the outputs at a scheduled time.

This allows the programmer to achieve unparalleled accuracy in linking outputs to time references for applications such as axis positions in motion control, or process control functions, without putting a huge burden on the communication and logic processing systems and related application code.

## No Temporary Variables

S7 has a category of variables called Temporary Variables. Their scope is the program block in which they are defined and their lifetime is the execution of the program block in which they are defined.

Logix does not have an equivalent to the Temporary Variable. All variables are static – they retain their values until changed.

To achieve the functionality typically targeted in S7 applications, use for example one of the following approaches:

- Use program-scope tags.
- If you are programming an Add-On Instruction, use Local Tags (part of the Add-On Instruction data).

## No Accumulators or Special Registers needed

If you program in STEP 7 Statement List, you will be familiar with the accumulators and the AR1 and AR2 pointer registers. There are no equivalents in Logix. All operands are tags.

To achieve the functionality typically targeted in S7 applications, use for example one of the following approaches:

- Use program-scope tags.
- If you are programming an Add-On Instruction, use Local Tags (part of the Add-On Instruction data).
- Consider whether you need Logix equivalents of the S7 accumulators and special registers. They are there because of the low-level nature of S7 Statement List, and in a language such as Structured Text, it is unlikely that they will be needed.

# Conversion of System Software and Standard Functions

## Introduction

This chapter lists the more commonly used S7 System Functions, explains how the equivalent is done in Logix and provides several specific examples.

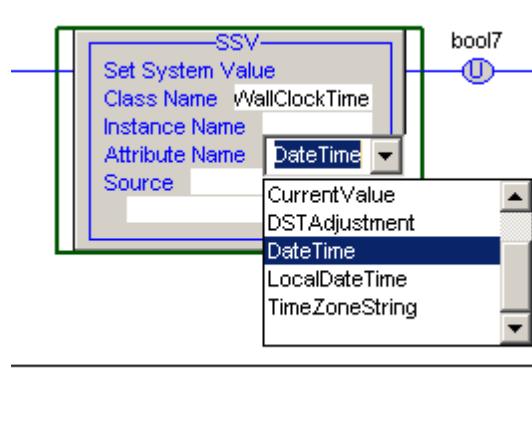
| <b>Topic</b>                      | <b>Page</b> |
|-----------------------------------|-------------|
| Logix System Functions            | 68          |
| Copy                              | 68          |
| Date and Time Setting and Reading | 69          |
| Read System Time                  | 69          |
| Handling of Interrupts            | 70          |
| Errors                            | 70          |
| Status – Controller               | 71          |
| Status – Module                   | 71          |
| Status – for OBs and Tasks        | 72          |
| Timers                            | 72          |
| Conversion Routines               | 73          |
| String Handling Routines          | 73          |
| Examples of System Function Calls | 74          |

The purpose of this chapter is to make you aware of the dedicated instructions available in Logix, so you do not waste time developing solutions that already exist.

## Logix System Functions

In Logix, the equivalent of most S7 System Functions will be the GSV (Get System Value) and the SSV (Set System Value) instructions. These instructions access a hierarchy of objects (Classes, Instances and Attributes) built-in to Logix controllers. If you program GSV and SSV, drop-down menus will guide you through parameter selection.

### SSV Instruction



Once the basics of GSV and SSV have been learned, the new Logix user may find that access to the operating system is easier than with S7 SFCs.

## Copy

Used for copying complex data structures - arrays of instances of User Data Types.

| S7             | Comment   | Logix                             | Comment   |
|----------------|---|-----------------------------------|---|
| SFC20 BLKMOV   | With BLKMOV, the addresses must be defined at compile time.   | COP (instruction)                 | If COP is used to copy between arrays, the start of the block (source or destination) may include an array index to address the element whose value is evaluated at run time. |
| SFC81 UBLKMOV  | Uninterruptible version – to ensure that source data cannot change during copy.                     | CPS (instruction)                 | Uninterruptible version – to ensure that source data cannot change during copy.   |
| SFC14 DPRD_DAT | If Profibus DP device has communications data area > 4 bytes the SFC will ensure consistent reads.  | CPS (ControlNet and Ethernet /IP) | Not required for DeviceNet  |
| SFC15 DPWR_DAT | If Profibus DP device has communications data area > 4 bytes the SFC will ensure consistent writes. | CPS (ControlNet and Ethernet /IP) | Not required for DeviceNet  |

## Date and Time Setting and Reading

The controllers of both systems have a real-time clock, which can be read or set.

| <b>S7</b>     | <b>Comment</b>                                       | <b>Logix</b>              | <b>Comment</b>  |
|---------------|--|---------------------------|---|
| SFC0 SET_CLK  | Values passed in an instance of type DT (DateTime)   | SSV<br>(Set System Value) | SSV Class - WallClockTime<br>SSV Attribute - DateTime<br>SSV source - specify element[0] of DINT[7] |
| SFC1 READ_CLK | Values returned in an instance of type DT (DateTime) | GSV<br>(Get System Value) | GSV Class - WallClockTime<br>GSV Attribute - DateTime<br>GSV dest – element[0] of DINT[7]           |

## Read System Time

The controllers of both these systems have a system clock, which starts at the time the controller starts. In the S7 system the time is in milliseconds, in Logix it is microseconds.

| <b>S7</b>      | <b>Comment</b>                           | <b>Logix</b>              | <b>Comment</b>   |
|----------------|--|---------------------------|--|
| SFC64 TIME_TCK | Returns system time in range 0...2.31 ms | GSV<br>(Get System Value) | Returns system time in range 0...2.63 $\mu$ s<br>GSV Class - CST<br>GSV Attribute - CurrentValue<br>GSV dest - specify element[0] of DINT[2]<br>DINT[0] - lower 32 bits<br>DINT[1] - upper 32 bits |

## Handling of Interrupts

Interrupts can be enabled and disabled by the user program making calls to system functions.

| S7             | Comment   | Logix                           | Comment  |
|----------------|---|---------------------------------|--|
| SFC39 DIS_IRT  | Disables interrupts handled by a specified OB. Interrupt requests are lost.                 | SSV<br>Inhibits specified task. | SSV Class - Task<br>SSV Instance - Task name<br>SSV Attribute - InhibitTask<br>SSV source - DINT variable set to 1 |
| SFC39 EN_IRT   | Enables interrupts handled by a specified OB  | SSV<br>Enables specified task.  | SSV Class - Task<br>SSV Instance - Task name<br>SSV Attribute - InhibitTask<br>SSV source - DINT variable set to 0 |
| SFC41 DIS_AIRT | Disables interrupts handled by a specified OB. Interrupt requests are delayed.              | UID                             | Disables interruption of the current task by a higher priority task  |
| SFC42 EN_AIRT  | Enables interrupts handled by a specified OB. Any interrupts delayed by SFC41 are executed. | UIE                             | Enables interrupts of the current task.  |

## Errors

These system calls return bit fields in the case of S7, or an integer in the case of Logix, representing error codes.

| S7             | Comment  | Logix  | Comment  |
|----------------|--|--|--|
| SFC38 READ_ERR | Reads and clears error bits. The type of error to be queried can be selected with a filtering field. | GSV<br>(Use SSV to reset counters or faults) | GSV Class - FaultLog<br>GSV Attribute:<br>MajorEvents – No of major events<br>MinorEvents – No of minor events<br>MajorFaultBits – current major fault<br>MinorFaultBits – current minor fault<br>GSV Target – INT or DINT to receive data |

## Status – Controller

The SFC (S7) and GSV call (Logix) will return data on the controller. Note – SFC51 requires some learning before it can be used. GSV in this case is more accessible.

| S7            | Comment   | Logix | Comment  |
|---------------|---|-------|--|
| SFC51 RDSYSST | <p>Input parameters specify the class of information to be read, and possibly an instance number if there are several objects.</p> <p>Output parameters are a pointer to a list with the returned information, and the number and size of the elements in the list.</p> | GSV   | <p>Modules with a direct connection: Examine 'Fault' or 'ChannelFault' member, if present. Modules with a rack optimized connection: Examine the 'SlotStatusBits' member of the adapter input data or the 'Fault' member of the card as above. For all other cards: Execute GSV:</p> <p>Class – Module<br/>Instance – ModuleName<br/>Attribute - Entrystatus</p> |

## Status – Module

The SFC (S7) and GSV call (Logix) will return data on the installed modules.

| S7            | Comment   | Logix | Comment   |
|---------------|---|-------|---|
| SFC51 RDSYSST | <p>Input parameters specify the class of information to be read, and possibly an instance number if there are several objects.</p> <p>Output parameters are a pointer to a list with the returned information, and the number and size of the elements in the list.</p> | GSV   | <p>GSV Class - Module<br/>GSV Attribute:<br/>EntryStatus (relationship of the Module object to the Module)<br/>FaultCode<br/>FaultInfo<br/>ForceStatus<br/>LEDStatus<br/>Mode (SSV also)<br/>GSV Target – depends on attribute chosen</p> |

You can monitor fault information in the Logix tags that are created when the module is inserted into the I/O Configuration. Similarly with STEP 7, if you go to the hardware configuration and switch to “Open ONLINE”, fault information for modules will be displayed.

## Status – for OBs and Tasks

| S7        | Comment   | Logix     | Comment  |
|-----------|---|-----------|--|
| OB Header | Status data for OBs is stored in Temporary variables that are automatically generated by the OB header. These may be directly accessed by the OB code, and transferred to static data areas if access is required from outside the OB.<br>See an example below. | GSV / SSV | GSV Class - Task<br>GSV instance – Task name<br>GSV attribute:<br>DisableUpdateOutputs (at the end of the Task)<br>EnableTimeOut<br>InhibitTask<br>Instance<br>LastScanTime (microseconds)<br>MaxIntervaln (between successive executions of Task)<br>OverlapCount (triggered while executing)<br>Priority<br>Rate (period in microseconds)<br>StartTime (value of WallClockTime when task was last started)<br>Status (3 status bits)<br>Watchdog (microseconds)<br>GSV Source / Target – depends on attribute chosen |

## Timers

| S7       | Comment   | Logix  | Comment                  |
|----------|---|--|--------------------------|
| SFB4 TON | On-delay timer                                  | TON (LD)<br>TONR (ST & FBD)                  | On-delay timer           |
|          |   | RTO (LD)<br>RTOR (LD & ST)                   | Retentive on-delay timer |
| SFB5 TOF | Off-delay timer                                 | TOF (LD)<br>TOFR (ST & FBD)                  | Off-delay timer          |
| SFB3 TP  | Generates a pulse that will run unconditionally | Bit of the accumulator of a free-running TON |                          |



## Conversion Routines

| S7                       | Comment                  | Logix               | Comment   |
|--------------------------|--------------------------|---------------------|---|
| <b>Library functions</b> |                          | <b>Instructions</b> |   |
| FC16 I_STRNG             | Integer to string        | DTOS                | INT can be used as a source tag instead of DINT |
| FC5 DI_STRNG             | Double integer to string | DTOS                | DINT to string                                  |
| FC30 R_STRG              | Real to string           | RTOS                | Real to String                                  |
| FC38 STRG_I              | String to Integer        | DTOS                |   |
| FC37 STRG_DI             | String to double integer | STOD                | String to DINT                                  |
| FC39 STRG_R              | String to real           | STOR                | String to real                                  |

## String Handling Routines

| S7               | Comment   | Logix               | Comment                               |
|------------------|---|---------------------|---------------------------------------|
|                  | Library functions   |                     | Instructions                          |
| FC10 EQ_STRNG    | Compare strings for equality                                | EQU                 | Compare strings for equality          |
| FC13 GE_STRNG    | compare strings for >=                                      | GEQ (LD)<br>>= (ST) | compare strings for >=                |
| FC15 GT_STRNG    | compare strings for >                                       | GRT (LD)            | compare strings for >                 |
| FC19 LE_STRNG    | compare strings for <=                                      | LEQ (LD)<br><= (ST) | compare strings for <=                |
| FC24 LT_STRNG    | compare strings for <                                       | LES (LD)<br>< (ST)  | compare strings for <                 |
| FC29<br>NE_STRNG | Compare strings for <>                                      | NEQ (LD)<br><> (ST) | Compare strings for <>                |
| FC21 LEN         | Length of string  | .LEN                | Property of any string instance       |
| FC26 MID         | returns a middle section of string                          | MID                 | returns a middle section of string    |
| FC2 CONCAT       | concatenate two strings                                     | CONCAT              | concatenate two strings               |
|                  | Can be done with FC31<br>REPLACE                            | DELETE              | Delete a section of a string          |
| FC17 INSERT      | Insert source string in target<br>string                    | INSERT              | Insert source string in target string |
| FC31 REPLACE     | Replace n characters of target<br>string with source string | Use DELETE / INSERT |                                       |
| FC11 FIND        | Find a string in another string                             | FIND                | Find a string in another string       |

There is no equivalent in STEP 7 to Logix's ASCII Serial Port instructions – neither in the instruction set nor in the function library. These would have to be programmed in STL if required.

## Examples of System Function Calls

These examples are intended primarily to illustrate the use of the GSV/SSV instructions.

### Setting the Clock

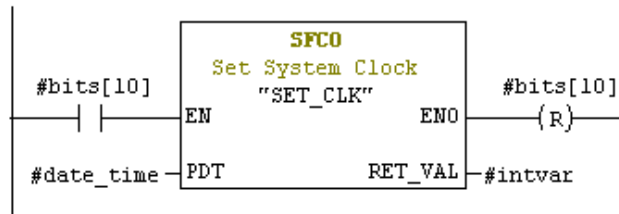
#### STEP 7

This call to SFC0 will set the clock. The time and data is entered in #date\_time.

The data and time are stored in 8 bytes following #data\_time in BCD format.

#### Network 14 : Title:

set the clock to the value stored in "date\_time"



0 – year

1 – month

2 – day

3 – hour

4 – minute

5 – second

6 – 2 most significant digits of milliseconds

7 – 1 least significant digit of milliseconds and day of week

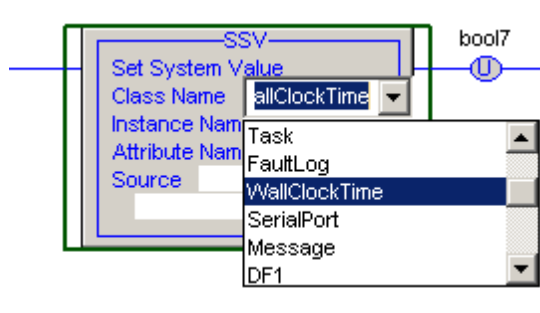
*Logix*

The date and time values are stored in the seven DINTs following #date\_time.

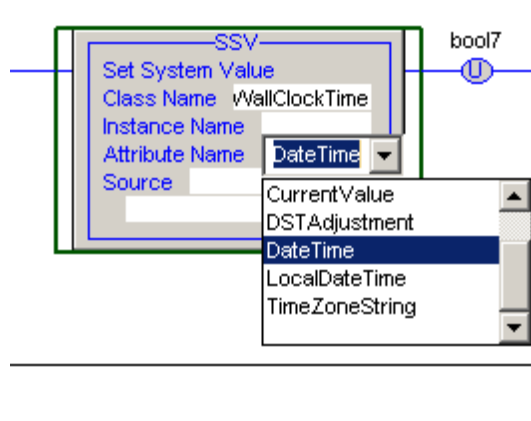


- 0- year
- 1 – month
- 2 – day
- 3 – hour
- 6 – minute
- 5 – second
- 6 - microsecond

The screen shot for Logix shows the data structure associated with GSV and SSV. Select class from a pull-down menu as follows.



Select Attribute from the pull-down menu, as follows.



Finally, select the tag that will be the source (SSV) or destination (GSV) of the data.

## Disabling Interrupts

### STEP 7

**Network 2 : Title:**

```

Disable interrupts for the Interrupt Execution (ie Periodic) 0B35

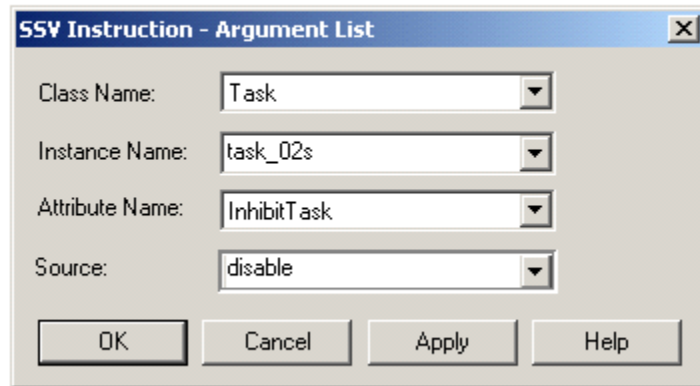
CALL "DIS_IRT"                SFC39          -- Disable New Int
MODE   :=B#16#2
OB_NR  :=35
RET VAL:=#intVar
    
```

## Logix

This example shows SSV in Structured Text.

If you type “gsv” then “alt-A” the following parameter selection screen will pop up.

```
// disable task_0.2s  
ssv();
```



Once the parameters are entered, click “OK” and the actual parameters will be completed.

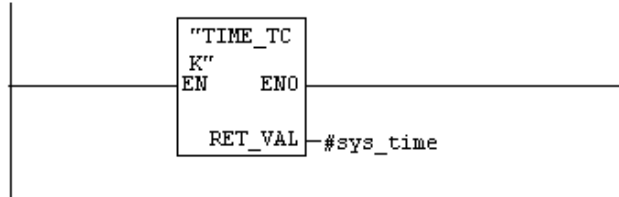
```
// disable task_0.2s  
ssv(Task,task_02s,InhibitTask,disable);
```

## Read System Time

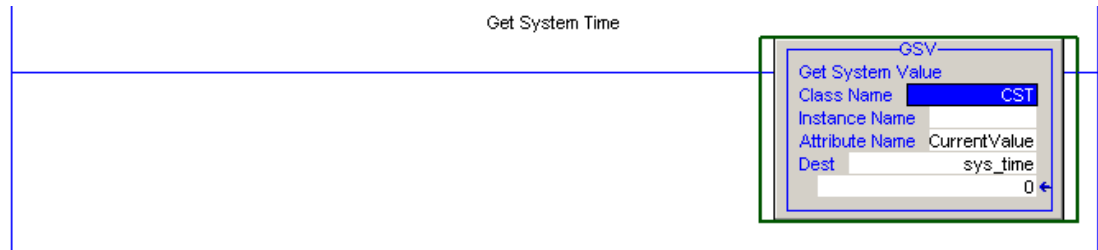
STEP 7

Network 15: Title:

read system time



Logix

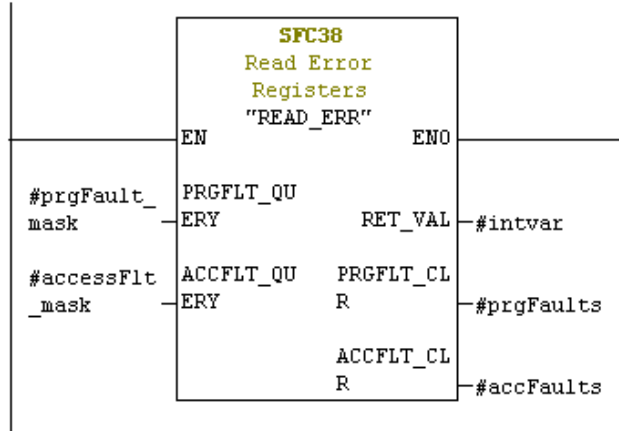


## Get Faults

### STEP 7

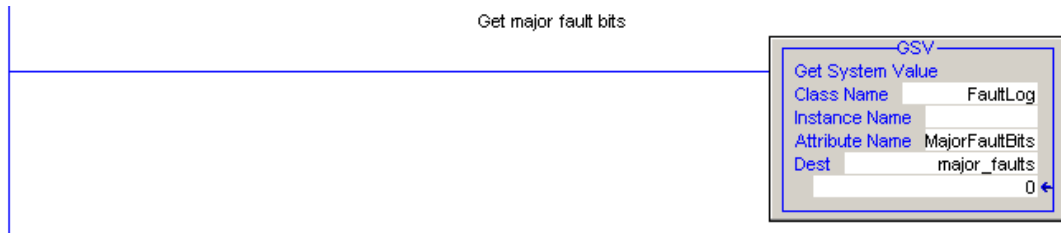
**Network 16 : Title:**

Get programming faults and I/O access faults



The bit pattern in the input parameters acts as a filter to select the faults that are to be queried. The faults returned are the **masked** faults – masking prevents them stopping the controller or calling a fault OB.

### Logix



## Module Information

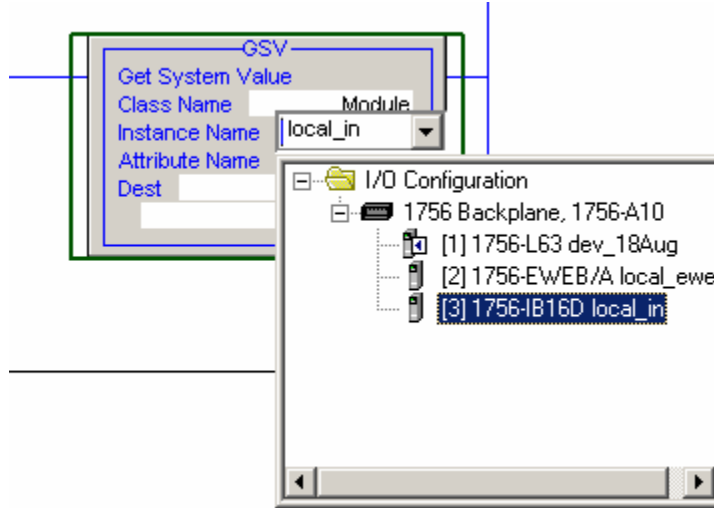
The easiest way is to inspect the module device profile tags, which contain fault/diagnostic information.

### 1756-IT6I2 Thermocouple Analog-input Card Tag

| Name                         | Alias For | Base Tag | Data Type                |
|------------------------------|-----------|----------|--------------------------|
| + Local:4:C                  |           |          | AB:1756_AI6_Float:C:0    |
| - Local:4:I                  |           |          | AB:1756_AI6_CJ_Float:I:0 |
| + Local:4:I.ChannelFaults    |           |          | INT                      |
| - Local:4:I.Ch0Fault         |           |          | BOOL                     |
| - Local:4:I.Ch1Fault         |           |          | BOOL                     |
| - Local:4:I.Ch2Fault         |           |          | BOOL                     |
| - Local:4:I.Ch3Fault         |           |          | BOOL                     |
| - Local:4:I.Ch4Fault         |           |          | BOOL                     |
| - Local:4:I.Ch5Fault         |           |          | BOOL                     |
| + Local:4:I.ModuleFaults     |           |          | INT                      |
| - Local:4:I.AnalogGroupFault |           |          | BOOL                     |
| - Local:4:I.InGroupFault     |           |          | BOOL                     |
| - Local:4:I.Calibrating      |           |          | BOOL                     |
| - Local:4:I.CalFault         |           |          | BOOL                     |
| - Local:4:I.CJUnderrange     |           |          | BOOL                     |
| - Local:4:I.CJOverrange      |           |          | BOOL                     |
| + Local:4:I.Ch0Status        |           |          | SINT                     |
| - Local:4:I.Ch0CalFault      |           |          | BOOL                     |
| - Local:4:I.Ch0Underrange    |           |          | BOOL                     |
| - Local:4:I.Ch0Overrange     |           |          | BOOL                     |
| - Local:4:I.Ch0RateAlarm     |           |          | BOOL                     |
| - Local:4:I.Ch0LAlarm        |           |          | BOOL                     |
| - Local:4:I.Ch0HAlarm        |           |          | BOOL                     |
| - Local:4:I.Ch0LLAlarm       |           |          | BOOL                     |
| - Local:4:I.Ch0HHAlarm       |           |          | BOOL                     |



Another way is to use the GSV instruction to read module objects. The screen shot below shows how to use GSV to obtain information regarding the 1756-IB16D digital input module.



## Get Scan Time

### STEP 7

This is a screen shot of the Temporary Variables header for OB1.

| Contents Of: 'Environment\Interface\TEMP' |             |         |   |  |
|---|-------------|---------|---|--|
| Name                                      | Data Type   | Address | Comment   |  |
| OB1_SCAN_1                                | Byte        | 1.0     | 1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1) |  |
| OB1_PRIORITY                              | Byte        | 2.0     | Priority of OB Execution                              |  |
| OB1_OB_NUMBR                              | Byte        | 3.0     | 1 (Organization block 1, OB1)                         |  |
| OB1_RESERVED_1                            | Byte        | 4.0     | Reserved for system                                   |  |
| OB1_RESERVED_2                            | Byte        | 5.0     | Reserved for system                                   |  |
| OB1_PREV_CYCLE                            | Int         | 6.0     | Cycle time of previous OB1 scan (milliseconds)        |  |
| OB1_MIN_CYCLE                             | Int         | 8.0     | Minimum cycle time of OB1 (milliseconds)              |  |
| OB1_MAX_CYCLE                             | Int         | 10.0    | Maximum cycle time of OB1 (milliseconds)              |  |
| OB1_DATE_TIME                             | Date_And... | 12.0    | Date and time OB1 started                             |  |

#OB1\_PREV\_CYCLE is the scan time. As a temporary variable, it ceases to exist when the execution of OB1 is complete. To store the scan time, copy #OB1\_PREV\_CYCLE to a static memory location.

*Logix*

The execution time can be retrieved for each Logix task.



With S7, you can directly get the scan time for OB1 from #OB1\_PREV\_CYCLE. However, for periodic OBs, there is no equivalent to #OB1\_PREV\_CYCLE. To get the execution time for periodic OBs, you will need to insert calls to SFC64 TIME\_TCK at the start and end of the OB, and subtract the system clock times returned by the SFC.

## Conversion of Typical Program Structures

### Introduction

The objective of this section is to demonstrate how some typical programming tasks in STEP 7 can be performed in RSLogix 5000 software. The discussion is based mainly on code fragments, but there are also some complete examples.

| Topic                               | Page |
|-------------------------------------|------|
| Conversion Code Examples            | 83   |
| Other Topics Related to Programming | 120  |
| A Larger Example - Control Module   | 121  |

There is also some discussion of matters related to programming, such as the scope and visibility of variables, and the scheduling of code sections.

### Conversion Code Examples These examples show conversion code.

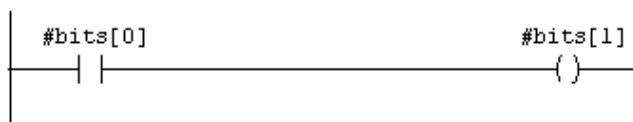
#### Ladder Logic Translation

This section describes a few examples of comparison between STEP 7 LAD and Logix LD.

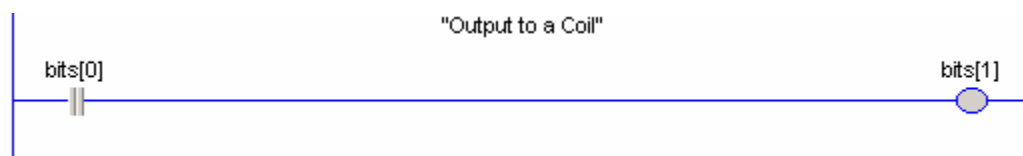
##### *Writing to a Coil*

STEP 7

Output to a coil



LOGIX

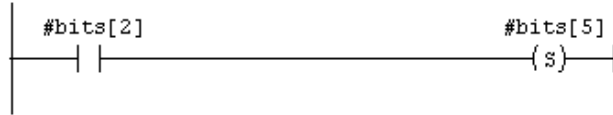


### Set and Reset

STEP 7

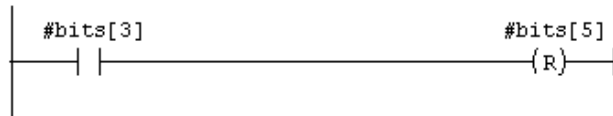
**Network 3 : Title:**

set bit

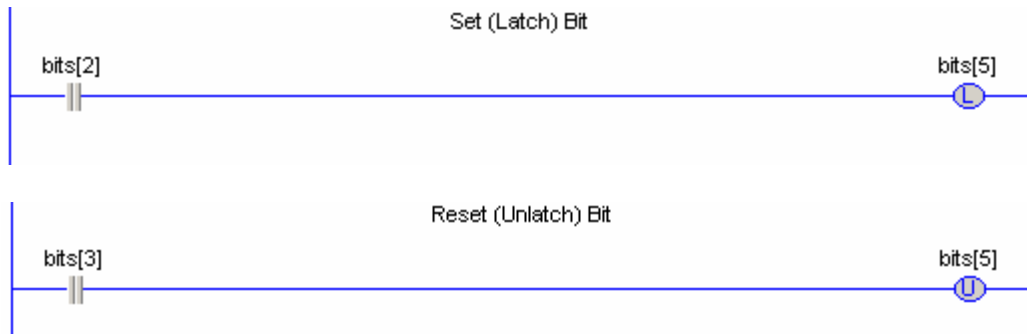


**Network 4 : Title:**

reset bit



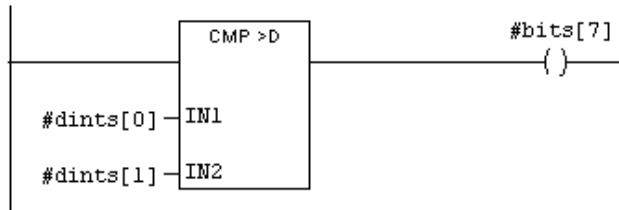
LOGIX



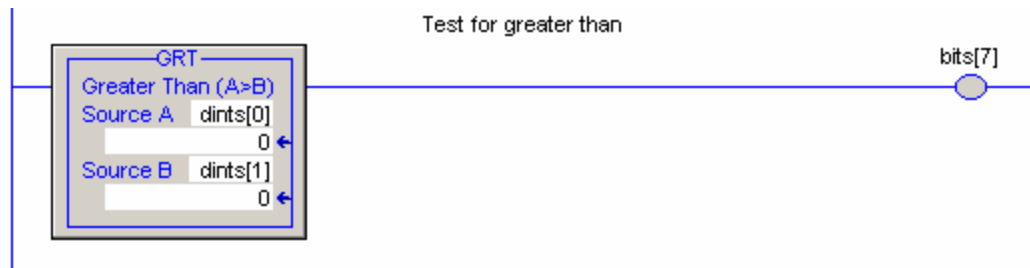
*Test for Greater Than*

STEP 7

test for greater than



LOGIX

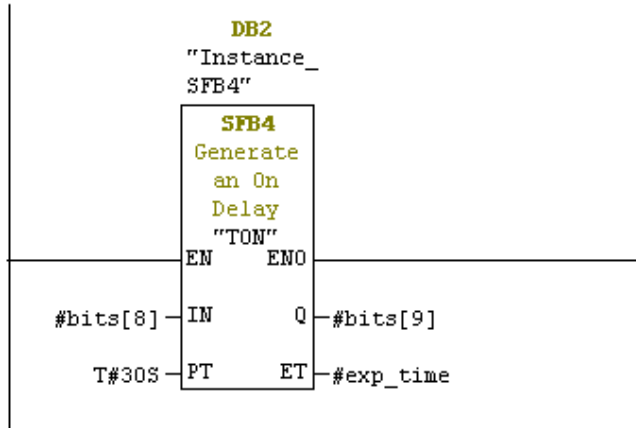


As before, use the CMP instruction if the expression is more complex than just comparing two numbers.

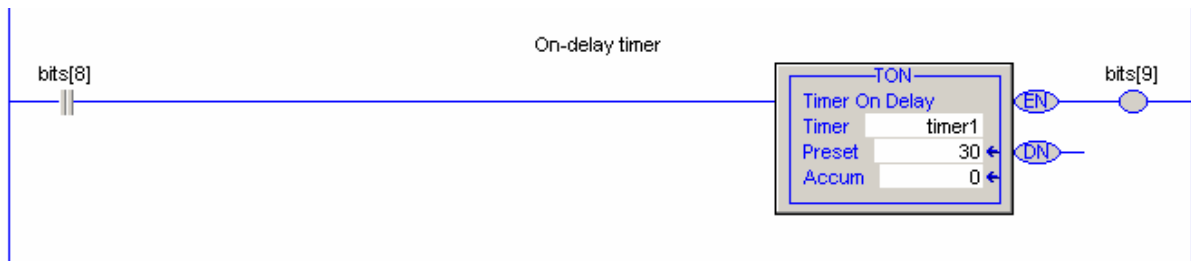
*On Timer Delay*

STEP 7

On delay timer



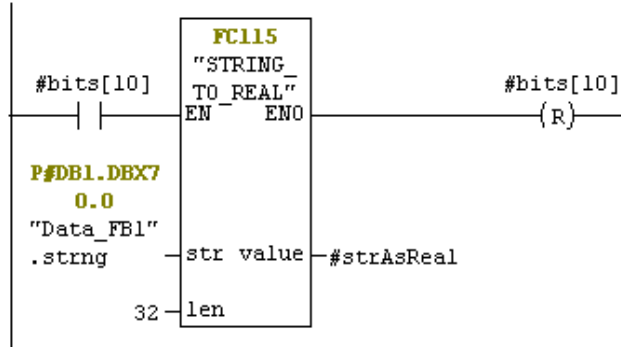
LOGIX



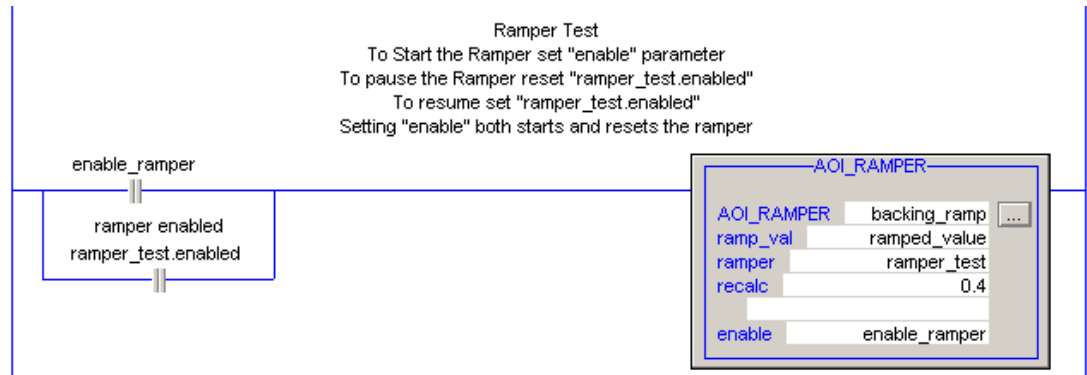
*Call to User Function*

STEP 7

user-function call



LOGIX

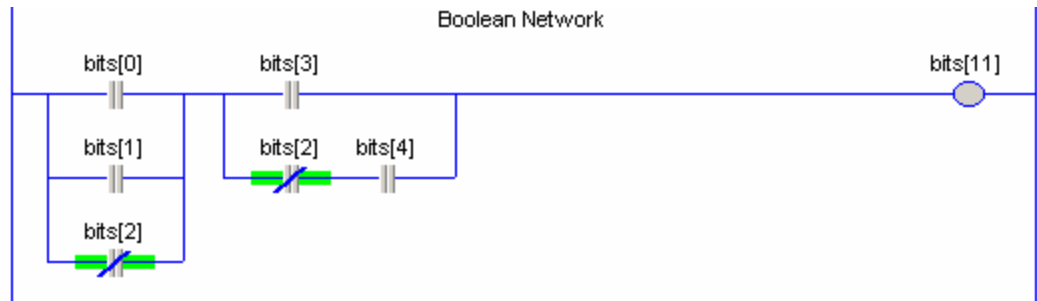


*Boolean Network*

STEP 7



LOGIX



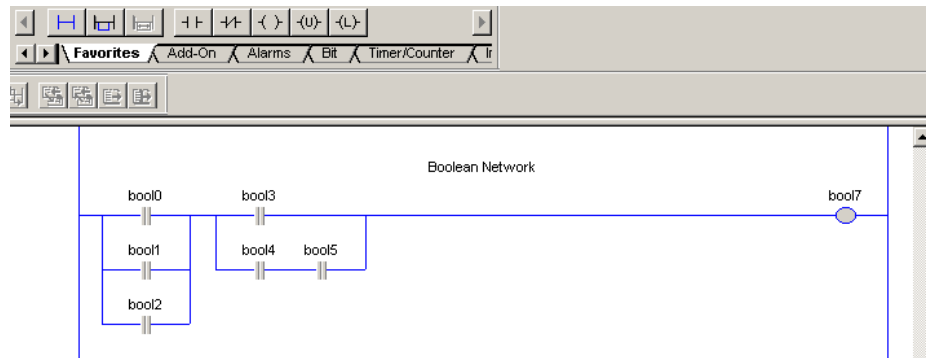
There is sufficient similarity between STEP 7 LAD and Logix LD to make translation at the level of routines fairly straightforward.



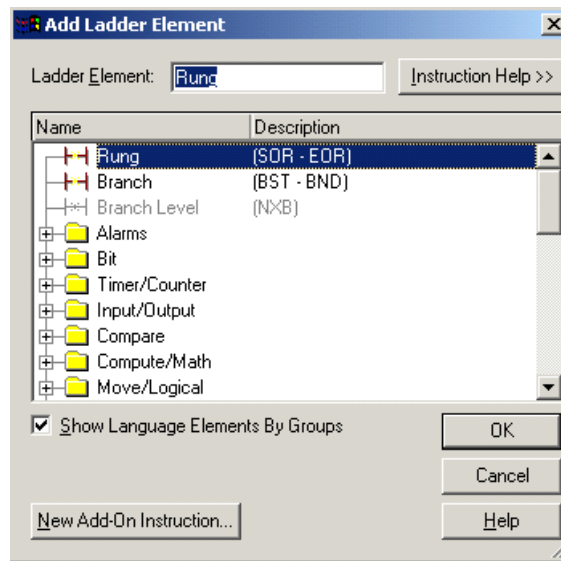
### The Logix LD Editor

There are no less than seven ways to select LD instructions. Two methods which are fairly similar to the way it is done in STEP 7 are described below.

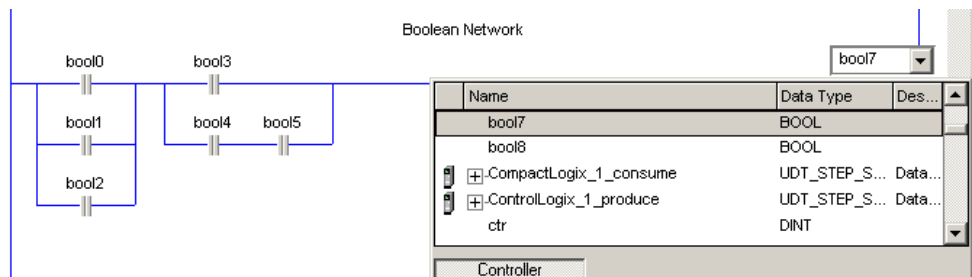
You can select from a palette above the LD worksheet.



If you type Alt+Insert, this selection pop-up will appear.



When configuring instructions, pull-down menus are available to allow you to select the tag to be entered.



## Jumps and Decision Making

### STEP 7 - Conventional Jump Sequence

The following example task is explained in the Network comment. Two S7 versions are shown because both are often used.

#### Network 1: Multi-way selection

```
if #input is 5 set #target to 8
else if #input is 6 set #target to 10
else if #input is 7 set #target to 16
else set #input to 0
```

```

L   #input
L   5
==I
JCN _001

L   8
T   #target

JU  end

_001: L   #input
L   6
==I
JCN _002

L   10
T   #target

JU  end

_002: L   #input
L   7
==I
JCN _003

L   16
T   #target

JU  end

_003: L   0
T   #target

end: NOP 0
```

The value of #input is compared with the set of constants until comparison is found. Then the action is performed, and comparison ceases. A default action is executed if #input does not compare with any value in the set.

*STEP 7 - Jump List*

In this example the task is the same, but a Jump List is used. This is similar to a microprocessor jump table, and transfers execution to a label depending on the value of a variable.

**Network 2 : Title:**

```
if #input is 5 set #target to 8
else if #input is 6 set #target to 10
else if #input is 7 set #target to 16
else set #input to 0
```

```

L    #input
L    5
-I
JL   rng
JU   d5
JU   d6
JU   d7
rng: L    0
T    #target
JU   cont

d5:  L    8
T    #target
JU   cont

d6:  L    10
T    #target
JU   cont

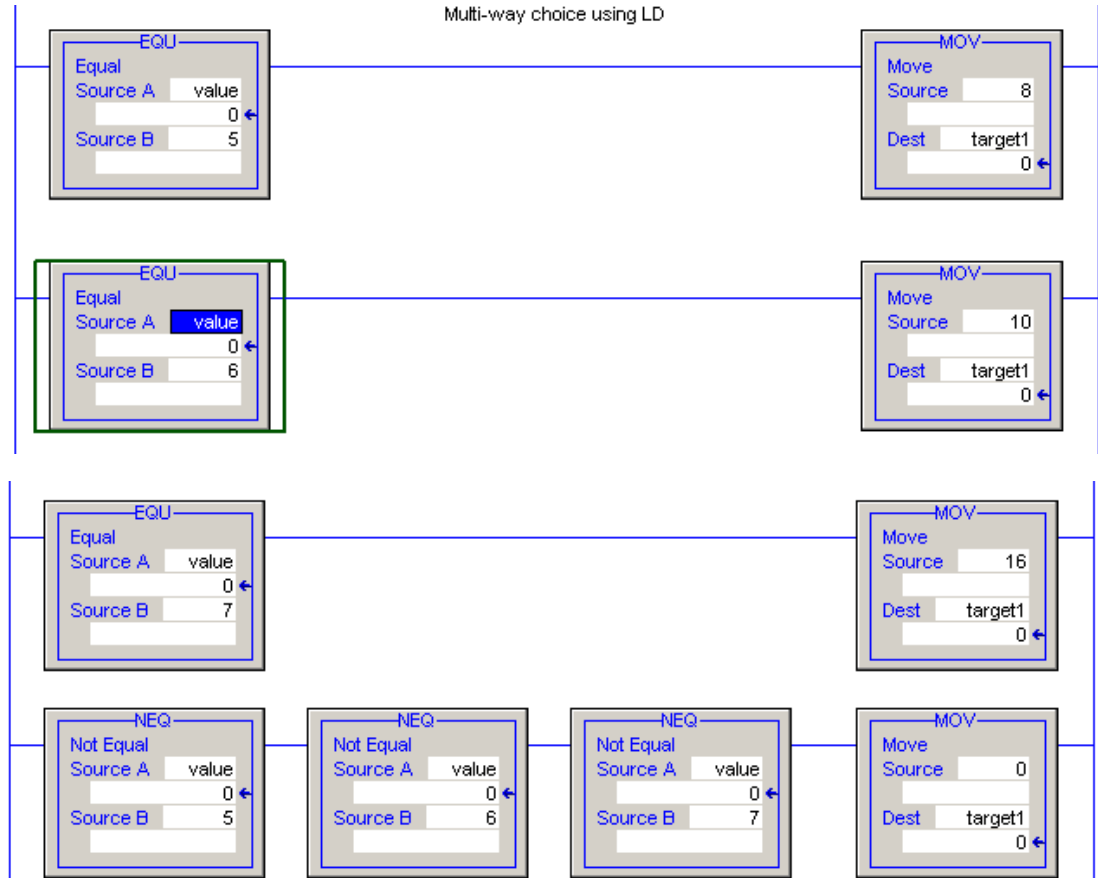
d7:  L    16
T    #target

cont: NOP 0
```

This is more readable than the conventional jump sequence, and is more efficient because only the code at the target label is executed.

*Logix - Ladder Logic*

This shows multi-way choice using LD.



*Logix - Structured Text If...Then...Else*

Anyone familiar with a programming language in the Basic/Pascal/C families will understand this without difficulty.

```
// multi-way choice using Structured Text  
  
if (value = 5) then target := 8;  
elsif (value = 6) then target := 10;  
elsif (value = 7) then target := 16;  
else target := 0;  
end_if;
```

Brackets around the “if” condition are not compulsory.

*Logix Structured Text CASE statement*

This is another variant in ST that does the same task. It is sufficiently compact and clean that there is little need for additional comment.

```
// multi-way choice using Structured Text CASE  
  
case value of  
  5: target := 8;  
  6: target := 10;  
  7: target := 16;  
else target := 0;  
end_case;
```

All solutions work but this is the preferred Logix solution. It is compact and sufficiently clear that no further documentation is needed.

## Arrays

STEP 7 and Logix both allow arrays of simple or complex objects to be created in memory. Logix has high-level support for accessing arrays. In STEP 7 however, low-level programming is needed.

### STEP 7 Array Creation

The following screen shot shows two arrays that have been created in an instance data block. Simple\_array is an array of 10 elements. UDT\_array is an array of 10 structures of type test\_UDT1, where test\_UDT1 is a user data type containing a few other types, not shown.

| Contents Of: 'Environment\Interface\STAT' |                 |         |               |  |
|---|-----------------|---------|---------------|--|
| Name                                      | Data Type       | Address | Initial Value |  |
| input                                     | Int             | 6.0     | 0             |  |
| target                                    | Int             | 8.0     | 0             |  |
| simple_array                              | Array [0..9...] | 10.0    |               |  |
| UDT_array                                 | Array [0..9...] | 50.0    |               |  |
| state                                     | Int             | 170.0   | 0             |  |
| error                                     | Bool            | 172.0   | FALSE         |  |

### Logix Array Creation

This is exactly the same in Logix.

| Name         | Alias For | Base Tag | Data Type     | Style   | Description                     |
|--------------|-----------|----------|---------------|---------|---------------------------------|
| target       |           |          | DINT          | Decimal |                                 |
| value        |           |          | DINT          | Decimal |                                 |
| simple_array |           |          | DINT[10]      | Decimal |                                 |
| UDT_array    |           |          | test_UDT1[10] |         | For testing Step7->Controllo... |
| index        |           |          | DINT          | Decimal |                                 |

### Array Declaration Syntax

STEP 7 uses the declaration syntax ARRAY[0...15] OF REAL. Logix uses REAL[15].

STEP 7 has a special syntax for strings. STRING[32] is a 32 character string in STEP 7 whereas in Logix STRING[32] is an array of thirty two strings, each one containing 82 characters.

*Array Access in STEP 7*

This example is to execute a simple task on the two arrays `simple_array[]` and `UDT_array[]`. The task is described in the network comment.

In STEP 7, it is not possible to access arrays using the normal `array[]` notation. Instead you have to use low-level operations with pointers. In the fragment below, a function “GET\_INDEXED\_REFERENCE” makes the task much easier by returning a pointer to the array element that is to be accessed.

**Network 3 : Title:**

```
array operations
-----
if (simple_array[2] = simple_array[5]) then
  UDT_array[8].boolean1 := 1;
else
  UDT_array[8].boolean1 := 0;
end_if;

// 1. compare simple_array[2] with simple_array[5]
CALL "GET_INDEXED_REFERENCE"          FC111
refArray := "Data_test".simple_array  P#DB1.DBX10.0
index    := 2
byteIncr := 4
startIndex:=FALSE
retVal   := #ptr1                      [BOOL]

CALL "GET_INDEXED_REFERENCE"          FC111
refArray := "Data_test".simple_array  P#DB1.DBX10.0
index    := 5
byteIncr := 4
startIndex:=FALSE
retVal   := #ptr2

OPN "Data_test"                        DB1
L   DID [#ptr1]
L   DID [#ptr2]
==D
=   #compare

// 2. get pointer to UDT_array
CALL "GET_INDEXED_REFERENCE"          FC111
refArray := "Data_test".UDT_array    P#DB1.DBX44.0
index    := 8
byteIncr := 12                       [INT]
startIndex:=FALSE
retVal   := #ptr1

L   #ptr1
LAR1

// 3. set or reset the bit
A   #compare
=   DIX [AR1,P#0.0]
```

In this case, the actual Logix Structure Text code was used as the Network comment, demonstrating just how intuitive Logix code is.

*STEP 7 - Looping Through Array Elements*

The objective in this example is to clear the float field in each structure in UDT\_array[]. This is not difficult, but confidence in using pointers is clearly required.

```
Array Operations
-----
```

```
Clear all float elements at UDT offset P#6.0 in array UDT_array
```

```
// transfer pointer to UDT_array to AR1
L    P##UDT_array
LAR1
// initialise counter
L    0
T    #ctr

// end if #ctr > 9
loop: L    #ctr
      L    9
      >I
      JC    end2
// clear the float field at offset p#6.0
L    0.000000e+000
T    DID [AR1,P#6.0]
// increment AR1 by size of the UDT
+AR1 P#12.0
// increment counter
L    #ctr
INC  1
T    #ctr
// loop back
JU   loop

end2: NOP  0
```



*Logix - Array Operations in Structured Text*

The following ST fragment performs the tasks described in the preceding two sections.

```
// array access in ST
if (simple_array[2] = simple_array[5]) then
    UDT_array[8].boolean1 := 1;
else
    UDT_array[8].boolean1 := 0;
end_if;

// clearing array elements
if (simple_array[0] = 5) then
    index := 0;
    while (index <= 9) do
        UDT_array[index].real1 := 0.0;
        index := index + 1;
    end_while;
end_if;
```

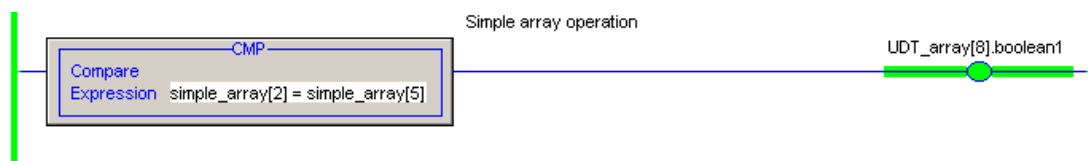
No additional comments are needed to describe how this works.

If you find yourself switching Boolean variables with if...then...else statements, consider writing a Boolean equation instead:

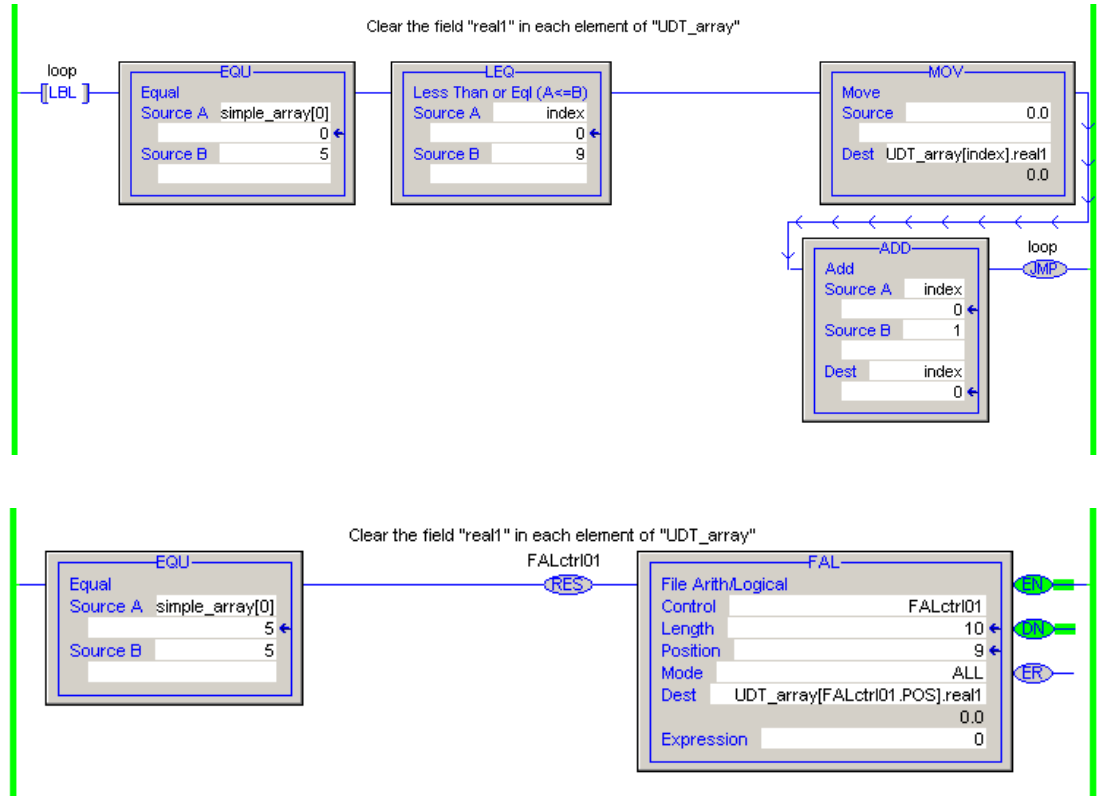
```
// array access in ST
UDT_array[8].boolean1 := simple_array[2] = simple_array[5];
```

*Logix - Array Operations in Ladder Diagram*

The examples of the previous section can be written in LD using the CMP (Compare) instruction as follows.



The second one (clearing the real field in the array of UDTs) can be done either of these ways.



The first approach to clearing the array elements is a translation from While Loop of the ST code. The second uses the advanced FAL instruction for array operations.

## User Data Types

Configuring and using User Data Types (UDTs) in STEP 7 and Logix is very similar.

Below is a UDT in STEP 7.

| Address | Name     | Type       | Initial value | Comment |
|---------|----------|------------|---------------|---------|
| 0.0     |          | STRUCT     |               |         |
| +0.0    | boolean1 | BOOL       | FALSE         |         |
| +0.1    | boolean2 | BOOL       | FALSE         |         |
| +2.0    | dint1    | DINT       | L#0           |         |
| +6.0    | real1    | REAL       | 0.000000e+000 |         |
| +10.0   | spare    | WORD       | W#16#0        |         |
| =12.0   |          | END_STRUCT |               |         |

Below is a UDT in Logix.

Name:

Description:

Members: Data Type Size: 16 byte(s)

| Name                   | Data Type | Style   | Description |
|------------------------|-----------|---------|-------------|
| boolean1               | BOOL      | Decimal |             |
| boolean2               | BOOL      | Decimal |             |
| dint1                  | DINT      | Decimal |             |
| real1                  | REAL      | Float   |             |
| spare                  | DINT      | Decimal |             |
| <small>10# 0#0</small> |           |         |             |

In both systems, UDTs can be used to declare and define variables.

Here is a declaration involving a UDT in STEP 7.

| Contents Of: 'Environment\Interface\STAT' |                       |         |               |  |
|---|-----------------------|---------|---------------|--|
| Name                                      | Data Type             | Address | Initial Value |  |
| input                                     | Int                   | 6.0     | 0             |  |
| target                                    | Int                   | 8.0     | 0             |  |
| simple_array                              | Array [0..9] Of Dint  | 10.0    |               |  |
| UDT_array                                 | Array [0..9] Of UDT 1 | 50.0    |               |  |

Here is a declaration involving a UDT in Logix.

| Name                | Alias For        | Base Tag         | Data Type      | Style   |
|---------------------|------------------|------------------|----------------|---------|
| Limit_Switch_1      | Local:3:I.Data.0 | Local:3:I.Data.0 | BOOL           | Decimal |
| Local:3:C           |                  |                  | AB:1756_DI:C:0 |         |
| Local:3:I           |                  |                  | AB:1756_DI:I:0 |         |
| conveyor_1          |                  |                  | UDT1           |         |
| conveyor_1.boolean1 |                  |                  | BOOL           | Decimal |
| conveyor_1.boolean2 |                  |                  | BOOL           | Decimal |
| conveyor_1.dint1    |                  |                  | DINT           | Decimal |
| conveyor_1.real1    |                  |                  | REAL           | Float   |
| conveyor_1.spare    |                  |                  | DINT           | Decimal |

One minor difference between the two systems is as follows:

In STEP 7 you can declare a variable of type “struct”.

| Contents Of: 'Environment\Interface\STAT' |                       |         |               |  |
|---|-----------------------|---------|---------------|--|
| Name                                      | Data Type             | Address | Initial Value |  |
| target                                    | Int                   | 8.0     | 0             |  |
| simple_array                              | Array [0..9] Of Dint  | 10.0    |               |  |
| UDT_array                                 | Array [0..9] Of UDT 1 | 50.0    |               |  |
| state                                     | Int                   | 170.0   | 0             |  |
| error                                     | Bool                  | 172.0   | FALSE         |  |
| transition01                              | Bool                  | 172.1   | FALSE         |  |
| transition12                              | Bool                  | 172.2   | FALSE         |  |
| transition13                              | Bool                  | 172.3   | FALSE         |  |
| transition24                              | Bool                  | 172.4   | FALSE         |  |
| transition43                              | Bool                  | 172.5   | FALSE         |  |
| transition31                              | Bool                  | 172.6   | FALSE         |  |
| str                                       | String[46]            | 174.0   | "             |  |
| table                                     | Struct                | 222.0   |               |  |

Notice the entry “table” of type Struct. Inside “table” can be a collection (ordered or unordered) of any combination of types.

In Logix, this would be done by defining “Struct” as a UDT containing the desired data structure and then declaring “table” as type Struct.

| Name                 | △ | Alias For        | Base Tag         | Data Type      | Style   |
|----------------------|---|------------------|------------------|----------------|---------|
| Limit_Switch_1       |   | Local:3:I.Data.0 | Local:3:I.Data.0 | BOOL           | Decimal |
| + Local:3:C          |   |                  |                  | AB:1756_DI:C:0 |         |
| + Local:3:I          |   |                  |                  | AB:1756_DI:I:0 |         |
| - conveyor_1         |   |                  |                  | UDT1           |         |
| conveyor_1.boolean1  |   |                  |                  | BOOL           | Decimal |
| conveyor_1.boolean2  |   |                  |                  | BOOL           | Decimal |
| + conveyor_1.dint1   |   |                  |                  | DINT           | Decimal |
| conveyor_1.real1     |   |                  |                  | REAL           | Float   |
| + conveyor_1.spare   |   |                  |                  | DINT           | Decimal |
| - table              |   |                  |                  | Struct         |         |
| table.status_bit1    |   |                  |                  | BOOL           | Decimal |
| table.status_bit2    |   |                  |                  | BOOL           | Decimal |
| table.status_bit3    |   |                  |                  | BOOL           | Decimal |
| + table.dwell_timer1 |   |                  |                  | TIMER          |         |
| + table.dwell_timer2 |   |                  |                  | TIMER          |         |
| table.speed          |   |                  |                  | REAL           | Float   |

## Pointers and Arrays

A STEP 7 program can have pointers to any data object. Indirect access to data blocks is also allowed but there are no pointers to functions (except in a restricted way by the JL (Jump List) instruction). The data pointer is unusual in that it is a pointer to a bit. Hence its value is eight times that of a normal pointer to a byte. This reflects the importance of bits in control systems programming.

In Logix there are no pointers. Arrays perform the same function as pointers, but are simpler and safer.

Will the S7 programmer be able to do a full range of tasks in Logix without pointers? In computer programming, pointers to data are used principally for three purposes:

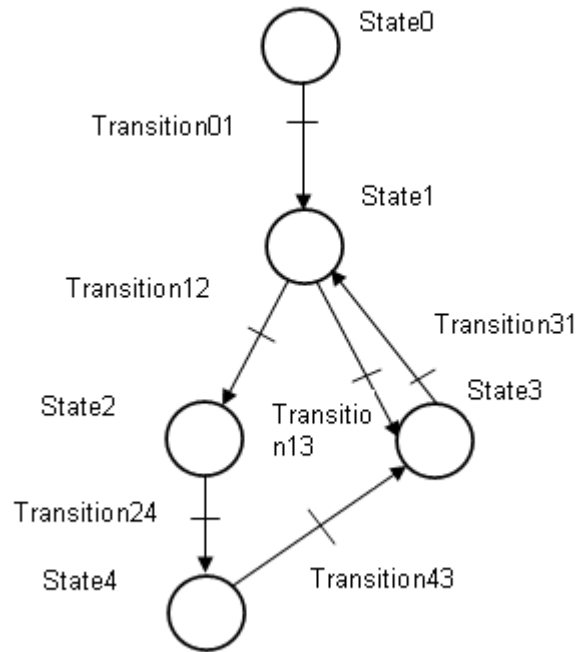
- Operations on sequentially ordered data items (arrays of objects, strings).
- Allocating, accessing and deleting dynamically allocated objects.
- Passing references to objects as parameters in function calls.

In Logix, the first purpose is covered by arrays. The second purpose is not relevant in control software because we do not have dynamically allocated objects. The third is covered by “inout” parameters in both STEP 7 function blocks and Logix Add-On Instructions.

It is concluded therefore that the absence of explicit pointers is not a limitation for Logix programmers. STEP 7 programmers should also discover that coding using arrays can be more quickly performed in Structured Text using arrays than in STL using pointers.

## State Machine

The State Machine is an important construct in control systems software because it greatly simplifies the task of programming sequential control.



## STEP 7 State Machine

STEP 7 offers a graphical Sequential Function Chart as an optional extra to the basic application. If the graphical SFC is not available, Statement List will do the job.

**Network 4:** Title:

```
state machine
```

```

      L    #state
      JL   rngl
      JU   st0
      JU   st1
      JU   st2
rngl: SET
      S    #error
      BEU

st0:  L    1
      A    #transition01
      JC   next
      JU   ovr

st1:  L    2
      A    #transition12
      JC   next

      L    3
      A    #transition13
      JC   next
      JU   ovr

st2:  L    4
      A    #transition24
      JC   next
      JU   ovr

st3:  L    0
      A    #transition31
      JC   next
      JU   ovr

st4:  L    3
      A    #transition43
      JC   next
      JU   ovr

next: T    #state

ovr:  NOP  0
```



The variable #state contains the state number. The Jump List instruction causes execution to jump to the label relevant to the value of #state. If a transition condition from that state is True, the new state value is loaded in the accumulator and execution jumps to label “next”, where the new state number is transferred to variable #state.

### *Logix State Machine in Structured Text*

Here is the same state machine in Structured Text, using the CASE statement. As with the other ST examples, it would be hard to write a clearer description than the code itself.

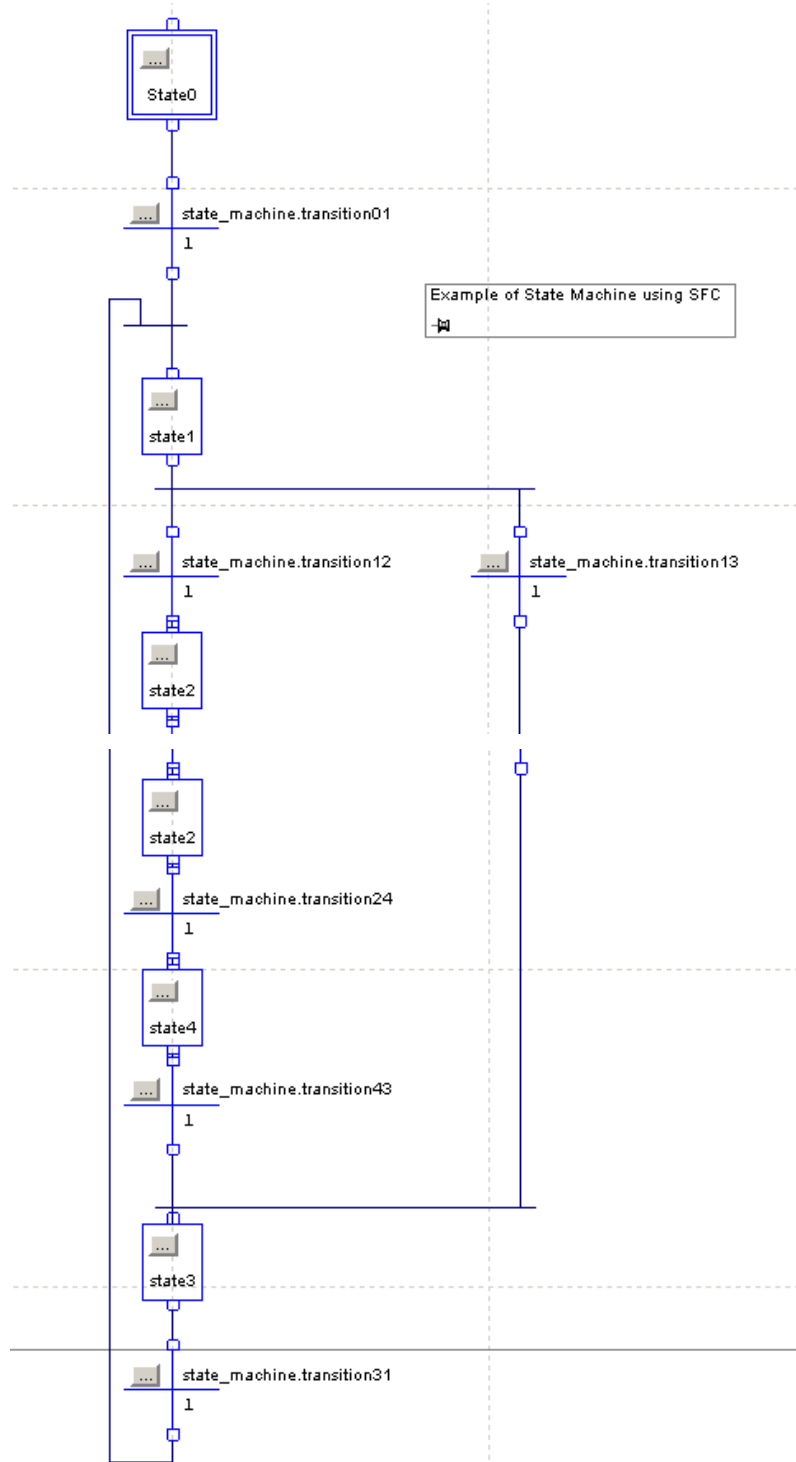
```
// implementation of State Machine using CASE in ST

case state_machine.state of
  0: if state_machine.transition01 then
      state_machine.state := 1;
    end_if;
  1: if state_machine.transition12 then
      state_machine.state := 2;
    elsif state_machine.transition13 then
      state_machine.state := 3;
    end_if;
  2: if state_machine.transition24 then
      state_machine.state := 4;
    end_if;
  3: if state_machine.transition31 then
      state_machine.state := 1;
    end_if;
  4: if state_machine.transition43 then
      state_machine.state := 3;
    end_if;
end_case;
```

### Logix State Machine in Sequential Function Chart

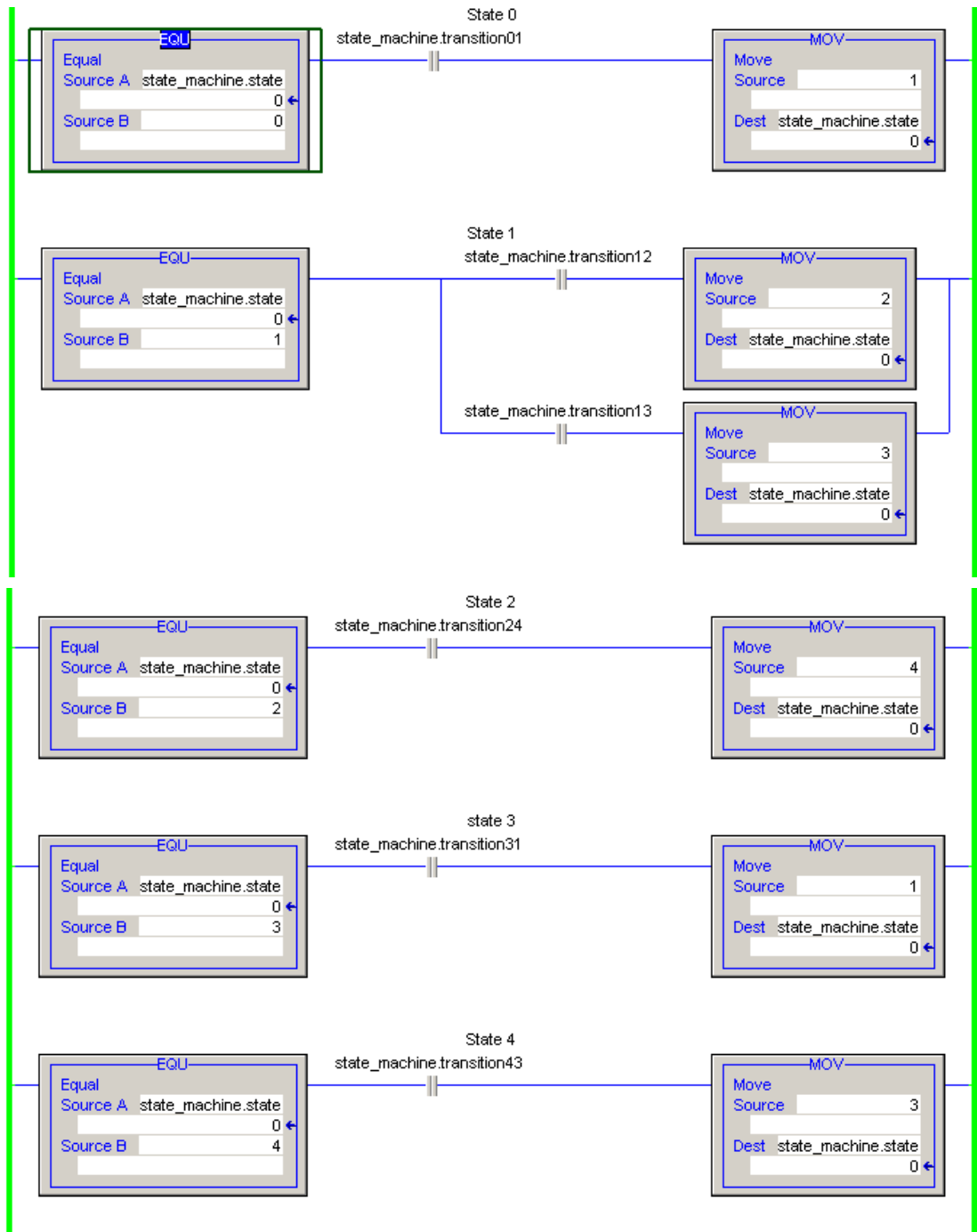
Logix provides a graphical SFC as one of its standard suite of languages. Shown below is the state machine in SFC.

Implementation of State Machine Using SFC chart



### State Machine in Ladder Diagram

The screen shot below shows how the state machine can be implemented in LD.



## Strings

### String Definition in STEP 7

| Contents Of: 'Environment\Interface\STAT' |            |         |                             |  |
|---|------------|---------|-----------------------------|--|
| Name                                      | Data Type  | Address | Initial Value               |  |
| transition12                              | Bool       | 172.2   | FALSE                       |  |
| transition13                              | Bool       | 172.3   | FALSE                       |  |
| transition24                              | Bool       | 172.4   | FALSE                       |  |
| transition43                              | Bool       | 172.5   | FALSE                       |  |
| transition31                              | Bool       | 172.6   | FALSE                       |  |
| str                                       | String[46] | 174.0   | 'This is an example string' |  |
| table                                     | Struct     | 222.0   |                             |  |

The data header shows how strings are defined. The length of the string is entered in brackets [] after the String data type. The initial value of the string is typed in the “Initial Value” column.

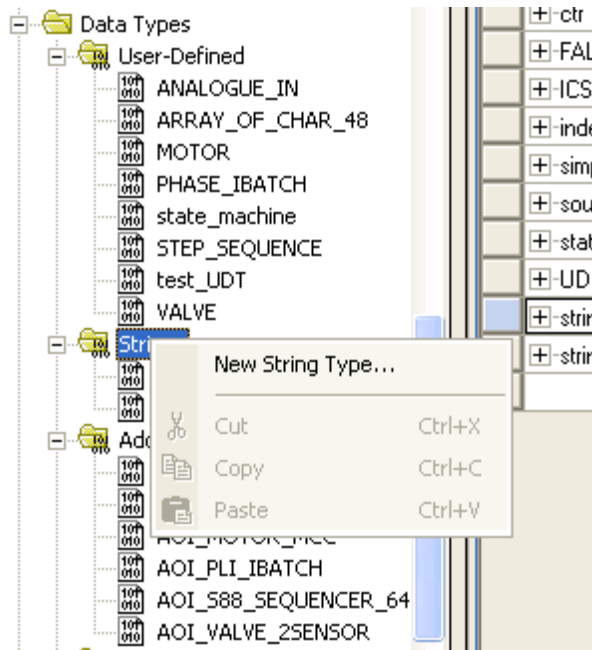
It is possible to create an array of strings, but each could not be given an initial value. An alternative definition to avoid this problem is shown by the entry “table” in the data header. “Table” is a structure. The contents of the structure, not shown, are five instances of string[46], and each has been given an initial value.

### String Definition in Logix

The extract from the tag configuration table below shows how string are defined in Logix.

|                   |  |              |
|-------------------|--|--------------|
| +UDT_array        |  | test_UDT[10] |
| +string_of_82char |  | STRING       |
| +string_of_40char |  | STRING 40    |

If you wish to create a string of a different length than the 82-character default, right-click on “strings” in your project tree (as shown below).



Then configure the properties as below.

Name:

Description:

Maximum Characters:

Members: Data Type Size: 52 byte(s)

| Name | Data Type | Style   | Description |
|------|-----------|---------|-------------|
| LEN  | DINT      | Decimal |             |
| DATA | SINT[48]  | ASCII   |             |

Having done this, you can define instances of your new type.

|   |  |  |           |  |
|---|--|--|-----------|--|
| <input type="checkbox"/> string_of_82char |  |  | STRING    |  |
| <input type="checkbox"/> string_of_48char |  |  | STRING_48 |  |

With instances of type `STRING` or `STRING_48`, there is a `LEN` field that automatically updates when a string constant is entered or when the string is manipulated by `ASCII` or `STRING` instructions.

## STEP 7 Temporary Variables

One of the categories of variable in STEP 7 is the Temporary variable. They can be created in any Organization Block, Function or Function Block.

Temporary variables are used for local, temporary storage of intermediate values, and for pointers. They only exist while their block is executing, and their values are lost when the block terminates.

Logix does not have Temporary variables. All storage is static - that is, values are retained between code executions.

If you use Add-On Instructions, you will notice that Local variables can be created for an Add-On Instruction. These variables can be used in the same way as temporary variables.

## Functions

If the STEP 7 programmer uses Statement List, he might have to develop low-level routines that are time consuming to write and require careful testing. Functions are important because the development of such routines need be done only once, and having been done, both the originator of the function and other programmers can do the same thing in a fraction of the time.

This section discusses how functions can be implemented in Logix.

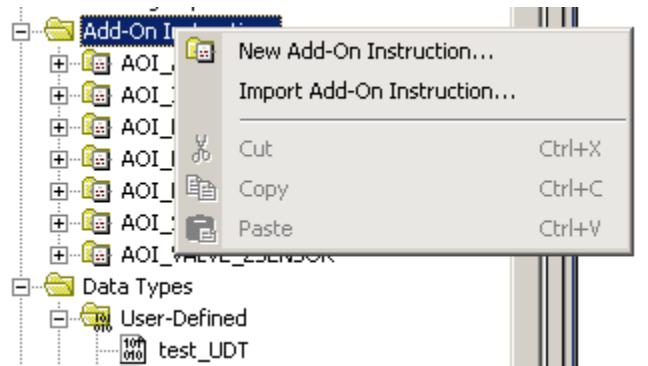
### *Function as Add-On Instruction in Logix*

STEP 7 Functions and Function Blocks are similar in their structure to the Logix Add-On Instruction. The Add-On Instruction has the same types of parameters as the FB (Input, Output and InOut) and it has its own data area. Once coded and tested, an Add-On Instruction can be used from anywhere in a program, and is sufficiently self-contained to be exported to other projects or placed in a code library.

*Example - a Ramp Function*

This example takes a real variable and ramps it linearly from its current value to a new value at a specified rate.

Go to the Add-On Instructions branch of your project tree and right-click Add-On Instruction.



This form appears.

**New Add-On Instruction**

Name:

Description:

Type:

Revision: Major  Minor  Extended Text

Revision Note:

Vendor:

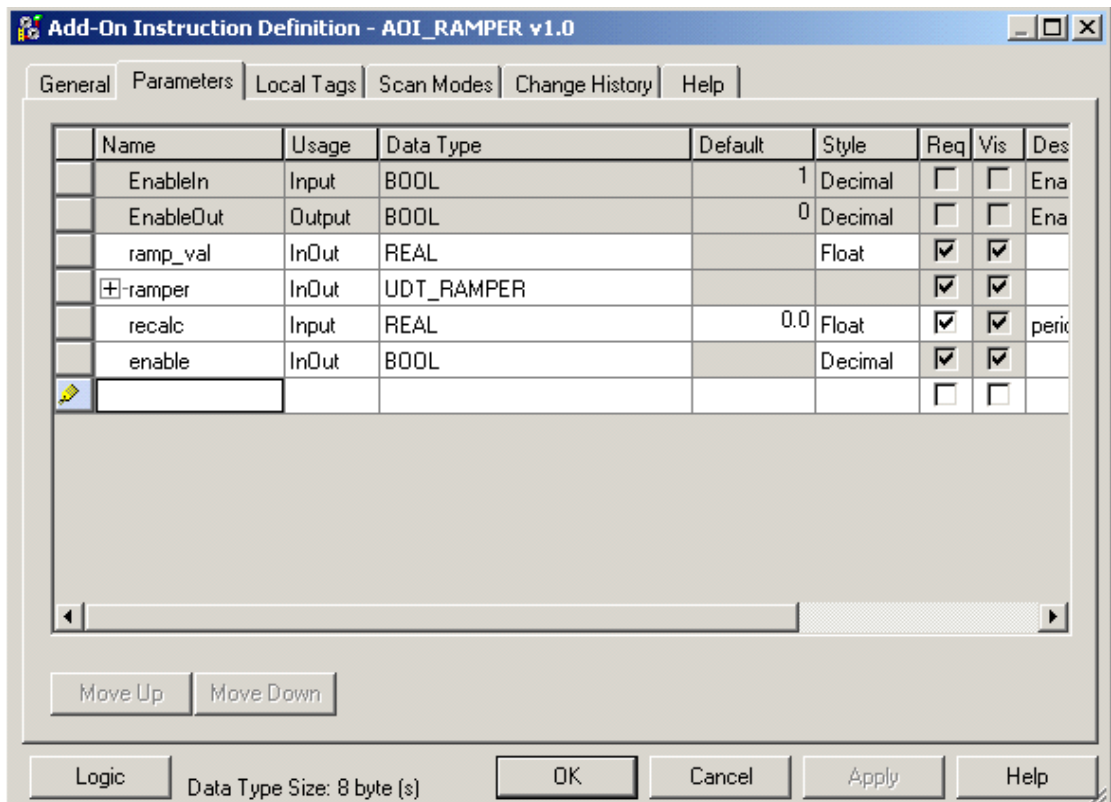
Open Logic Routine

Open Definition

OK Cancel Help

Enter the name of the Add-On Instruction and specify the language its code section will be written in.

Choose the Parameters tab.

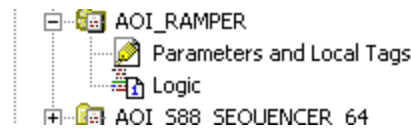


As in STEP 7, Input parameters are values from the program to the Add-On Instruction, Output parameters are values from the Add-On Instruction to the program and InOut parameters are for variables that will be modified by the Add-On Instruction. If you have any data structures, choose the InOut type anyway because they are passed by reference and this is more efficient.

| Members:       |           |         |                               | Data Type Size: 28 byte(s) |
|----------------|-----------|---------|-------------------------------|----------------------------|
| Name           | Data Type | Style   | Description                   |                            |
| initial_output | REAL      | Float   | saved initial output          |                            |
| increment      | REAL      | Float   | calculated increment          |                            |
| RAMP_RATE_ABS  | REAL      | Float   | per second - (set always +ve) |                            |
| RAMP_TARGET    | REAL      | Float   | final value - (set)           |                            |
| change         | REAL      | Float   | calculated change over ramp   |                            |
| counter        | DINT      | Decimal | internal counter              |                            |
| complete       | BOOL      | Decimal | rammin is complete            |                            |



In the project tree for AOI\_RAMPER, there is a logic section.



Open it to see the code for this Add-On Instruction.

```
// Ramps a real variable from its current value to a new value at a
// specified rate.

// Parameters:
//   ramp_val   - variable to be ramped
//   ramper     - instance of UDT UDT_RAMPER
//   recalcalc  - code recalculation period (s)
//   enable     - start signal

// To use - set the target value in ramper.RAMP_TARGET_ABS
//         - set the ramp rate in ramper.RAMP_RATE_ABS
//         - to Start the Ramper set "enable" parameter
//         - to pause the Ramper reset "ramper.enabled"
//         - to resume set "ramper.enabled"
//         - setting "enable" both starts and resets the ramper

// on completion, the UDT field "complete" is set and the UDT field
// "enabled" is reset

// when enable is set, initialise
if (enable & (enable xor ramper._enable)) then
    ramper.initial_output := ramp_val;
    ramper.change := ramper.RAMP_TARGET - ramp_val;
    ramper.increment := ramper.change / abs(ramper.change)
                      * ramper.RAMP_RATE_ABS * recalcalc;

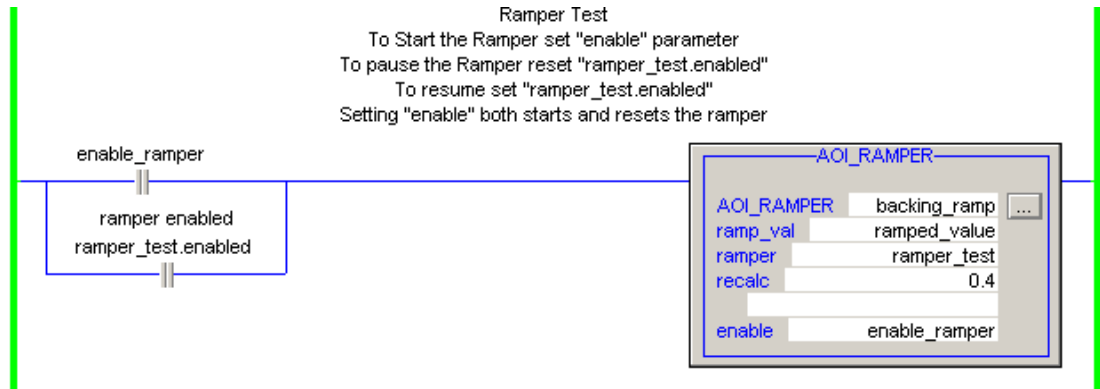
    ramper.counter := 0;
    ramper.complete := 0;
    enable := 0;
    ramper.enabled := 1;
end_if;

ramper._enable := enable;

// ramp calculations
if (ramper.enabled) then
    ramp_val := ramper.initial_output + (ramper.counter
                                         * ramper.increment);

    ramper.counter := ramper.counter + 1;
    if (abs(ramper.counter * ramper.increment)
        > abs(ramper.change)) then
        ramp_val := ramper.RAMP_TARGET;
        ramper.complete := 1;
        ramper.enabled := 0;
    end_if;
end_if;
```

The Add-On Instruction can be called from any routine.



Note that with Add-On Instructions, you will need to create a tag of type Add-On Instruction in a data area that is visible to the routine. This is called a backing tag.

Before you write an Add-On Instruction, check through the Instruction Help in RSLogix 5000 software. You might find that there is an existing instruction that will do the job. The following section will illustrate this.

### Block Copy, COP and CPS

In STEP 7, it is common to use the system function SFC20 “BLKMOV” to copy a block of data between locations.

```

CALL "BLKMOV"
SRCBLK := "Data_EMI".stepMsgs.step5
RET_VAL:=#intVar
DSTBLK := "Data_EMI".actualStep
    
```

The instruction copies the string from the fifth location in an array of strings to a destination string.

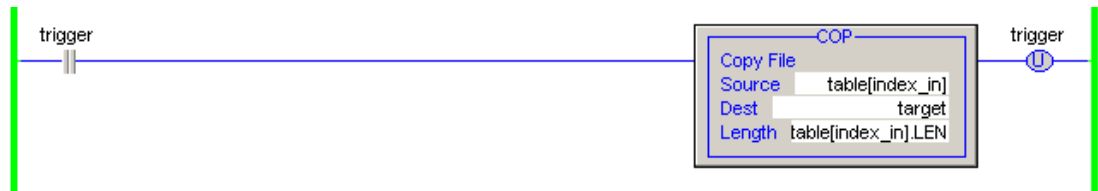
Often we want to copy the i-th element from an array, where “i” can vary as the program executes. “BLKMOV” cannot do this.

The STEP 7 programmer would write a function to meet his requirement.

```
// copy step number descriptor to SCADA display area (EM faceplate)
CALL "INDEXED_COPY"
indexSource := "Data_EMs".EM1.stepNumber
sourceRef   := "Data_EM1".stepMsgs
indexDest   := 1
destRef     := "Data_EM1".actualStep
recordLength:=8           // 32 bytes
```

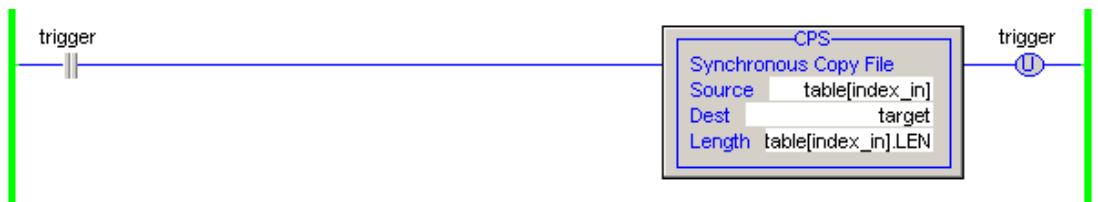
In this case, the copy is between two arrays and the indexes are defined by indexSource and indexDest.

In Logix, the COP built-in instruction will save all the work.



Because the source and destination specifications can include variable array indexes, COP will do the job. It is the equivalent of "INDEXED\_COPY".

The CPS instruction is the same as COP but with one difference.



The instruction cannot be interrupted. Therefore, the source and destination data will remain constant throughout its execution. If you wish to move data that could change, use CPS.

Examples are:

- copying input data to a buffer, from where the program will operate on the data.
- copying consumed tags to a buffer, from where the program will operate on the data.

## Mathematical Expressions

This section will describe how the S7 programmer can perform mathematical computations in Logix. An example will be used - the expression “ $\sqrt{\cos(x)^2 + \sin(x)^2}$ ”. The result of this expression is always exactly 1, so it's easy to check that you are getting the correct answer.

### STEP 7 - STL

Math code in STEP 7 STL is efficient, but perhaps not too clear for someone who is unfamiliar with STL.

**Network 1:** Title:

calculate:

```
(SIN(x)^2 + COS(x)^2) ^0.5
```

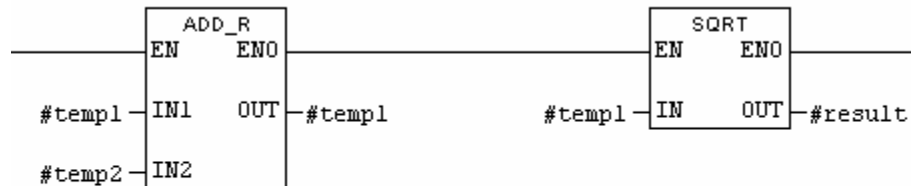
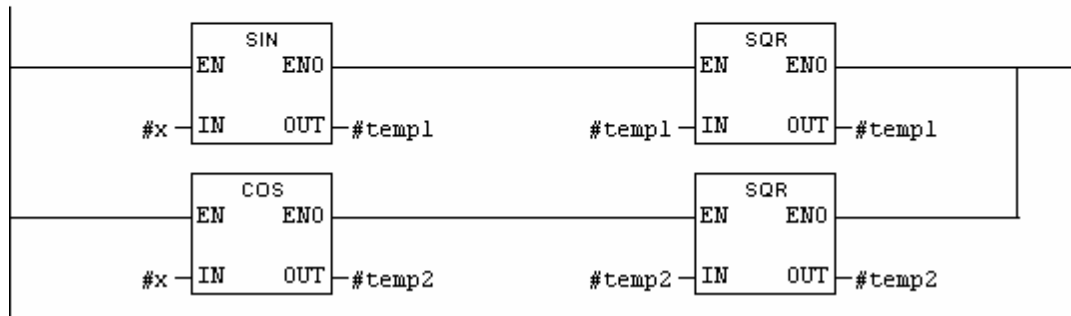
```
L    #x
SIN
SQR
L    #x
COS
SQR
+R
SQRT
T    #result
```

*STEP 7 - LAD*

Math evaluation in LAD follows a conventional pattern of combining functions.

**Network 1:** Title:

```
calculate:
(SIN(x)^2 + COS(x)^2) ^0.5
```



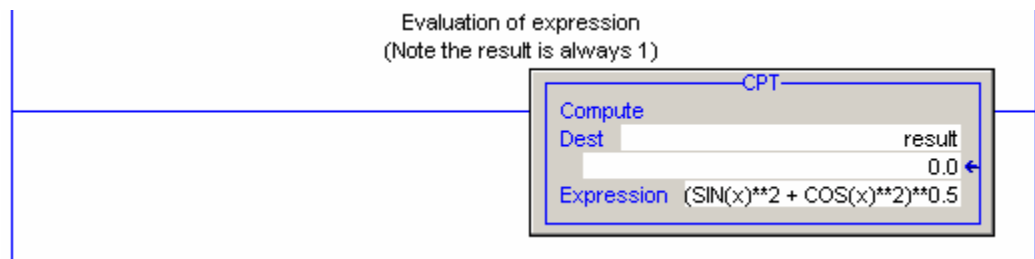
TEMP: REAL

*Logix - ST*

The expression is entered in the same way as with any other high-level language.

```
// evaluation of mathematical expression in Structured Text
result := (SIN(x)**2 + COS(x)**2)**0.5;
```

*Logix - LD*



The CPT instruction enables the expression to be entered in a high-level manner, which most people will understand more easily than a network (rung) of separate instructions.

*STEP 7 - User Function*

This function block has been written to do much the same as Logix CPT.

```
CALL "CPT" , "Data_CPT"          FB119 / DB119
str := "test_formulae".test_string15 P#DB16.DBX476.0
x   := 3.300000e+001
a   := 0.000000e+000
b   := 0.000000e+000
c   := 0.000000e+000
result:=#realVar
```

```
-- ((cos x)^2 + (sin x)^2) ^0.5 --
```

| IN          |  | OUT |             |
|-------------|--|-----|-------------|
| +3.300e+001 |  |     |             |
| +0.000e+000 |  |     |             |
| +0.000e+000 |  |     |             |
| +0.000e+000 |  |     |             |
|             |  |     | +1.000e+000 |

It reads and evaluates an expression string that is stored in a data block. It has a limitation compared with Logix CPT - the expression is written in reverse Polish notation, which will not suit everyone.

The main problems with writing a function block such as this are that it takes time and is not for beginning programmers. With Logix, the CPT instruction is available for everyone to use as soon as RSLogix 5000 software is installed.

### *Type Checking*

With both STEP 7 and Logix, parameters to Functions, Function Blocks, Instructions and Add-On Instructions are strictly type-checked by their compilers.

There are differences with mathematical expressions.

Logix distinguishes between Numeric and Boolean values. The compiler will reject expressions that illogically mix numeric and boolean values. When it encounters expressions of mixed numeric type, it will make conversions to produce a result of the type of the declared result variable. Hence it will interpret \* as integer multiplication if the result is to be an integer and as real multiplication if the result is to be a real.

In STEP 7 the type of arithmetic operations must be specified. There are for example \*I (multiply two 16 bit integers) \*D (multiply two 32 bit integer) and \*R (multiply two reals). It is up to the programmer to ensure that the two numbers that are the operands of a \*R instruction are reals. If they are not, the compiler will not complain but the result will be nonsense.

### *Conclusion*

The Logix methods of programming mathematical expressions are clearer, and by separating math code from other logic, will simplify testing and validation.

## Other Topics Related to Programming

### Scope of Variables

This is an area where Logix differs considerably from STEP 7.

#### *Rules for STEP 7*

- Temporary variables are invisible outside the block in which they are declared.
- Global static variables are visible throughout the program.
- Static variables that are declared as instance data to a function block have a special status in the FB, but they can be accessed from other parts of the program.

#### *Rules for Logix*

Execution in Logix is divided into Tasks. Each Task may have several Programs and each Program may have several Routines. Each Program may have its own tag section.

- Controller scope tags are visible throughout all Routines in all Programs.
- Program scope tags are visible only in the Routines in the Program in which they are defined. This means that if a routine in one Program is to share data with a routine in another Program, it must use Controller scope data.
- Add-On Instruction Local Tags are only visible to that Add-On Instruction's logic.

### OBs, Tasks, and Scheduling

Organization Blocks, Tasks and Scheduling are described in [Chapter 2](#).



## **A Larger Example - Control Module**

This example will assemble some of the different topics illustrated in the previous sections. The term “Control Module” (CM) comes from the influential S88 Batch Control standard. S88 has encouraged controller software design to be more “object oriented”. This Control Module is for a binary valve. The Add-On Instruction is suitable for this type of programming.

### **Components of the CM**

These are:

- a UDT called UDT\_VALVE.
- an Add-On Instruction called AOI\_VALVE\_2SENSOR
- a new Program under “task\_02s” called “valves\_callup”, which contains program tags section and a routine.

## User Data Type Valve

The UDT is shown below.

Name:

Description:

Members: Data Type Size: 24 byte(s)

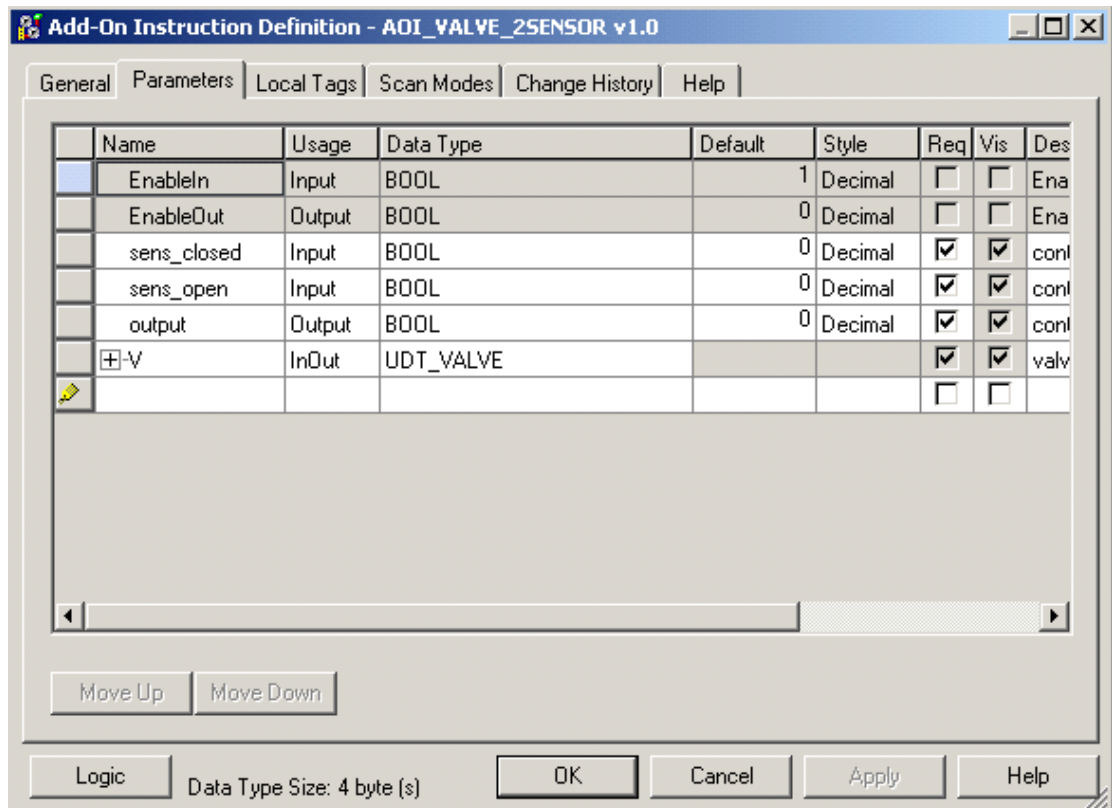
|                          | Name           | Data Type | Style   | Description                                     |
|--------------------------|----------------|-----------|---------|---|
| <input type="checkbox"/> | opening_preset | DINT      | Decimal | max preset for opening                          |
| <input type="checkbox"/> | closing_preset | DINT      | Decimal | max preset for closing                          |
| <input type="checkbox"/> | state          | DINT      | Decimal | state of valve (for internal logic)             |
| <input type="checkbox"/> | state_saved    | DINT      | Decimal | for evaluation of edge                          |
| <input type="checkbox"/> | timecount      | DINT      | Decimal | valve timer                                     |
| <input type="checkbox"/> | auto           | BOOL      | Decimal | auto mode (set from SCADA)                      |
| <input type="checkbox"/> | manual         | BOOL      | Decimal | manual mode (set from SCADA)                    |
| <input type="checkbox"/> | closed         | BOOL      | Decimal | state of valve                                  |
| <input type="checkbox"/> | open           | BOOL      | Decimal | state of valve                                  |
| <input type="checkbox"/> | fault_closing  | BOOL      | Decimal | closed sensor feedback not received             |
| <input type="checkbox"/> | fault_opening  | BOOL      | Decimal | open sensor feedback not received               |
| <input type="checkbox"/> | fault_sensors  | BOOL      | Decimal | sensors and logical state of valve do not agree |
| <input type="checkbox"/> | acquired       | BOOL      | Decimal | acquired by EM                                  |
| <input type="checkbox"/> | interlocked    | BOOL      | Decimal | interlocked - de-energise                       |
| <input type="checkbox"/> | fail_open      | BOOL      | Decimal | property - fails open                           |

Building the UDT should be the first step - it includes all the data that is necessary to model the valve.

## The Add-On Instruction

### *Add-On Instruction Parameters*

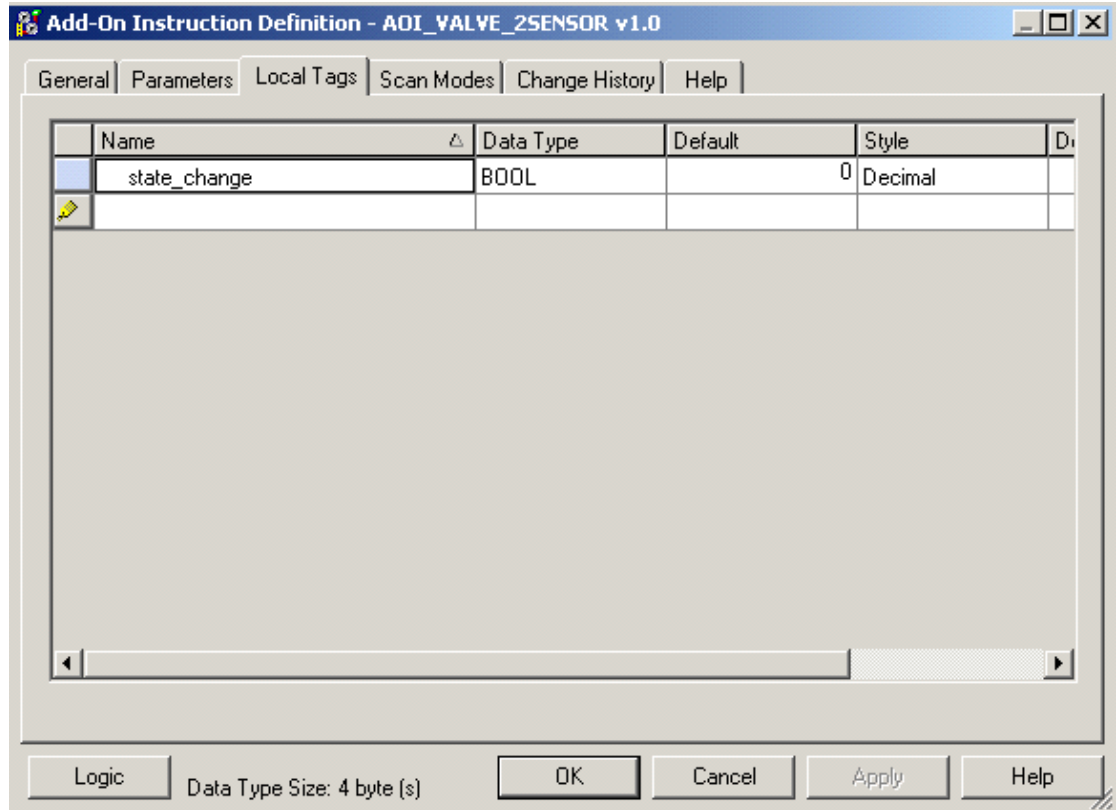
The screen shot shows the parameter configuration screen.



The parameters that have been added are the I/O for the valve and an object of type “UDT\_VALVE”. “V” must be an InOut parameter.

### Add-On Instruction Local Data

The screen shot below shows the configuration page for the Add-On Instruction local data.



*Add-On Instruction Logic*

The screen shot below shows the logic for this Add-On Instruction.

```

// Control Module Valve_2sensor
// -----
// Implements logic for a valve with an open and a closed sensor and one output

// See UDT Valve for data structure.

// Note the open/close command V.open_command must be set or reset externally
// and then left until the next activation is required. Do not continuously
// hold the flag set or reset.

// increment timer counter
V.timecount := V.timecount + 1;

// evaluate change of state (state machine)
state_change := V.state <> V.state_saved;
V.state_saved := V.state;

// set output
output := (V.fail_open xor V.open_command) and not
           (V.interlocked or V.faulted);

// valve is faulted
V.faulted := V.fault_opening or V.fault_closing or V.fault_sensors;

// action on fault or interlock
if V.faulted or V.interlocked then
  if V.fail_open then
    V.state := 3;
    V.open_command := 1;
  else
    V.state := 0;
    V.open_command := 0;
  end_if;
end_if;

// state machine:
// the state machine does not set outputs - it monitors inputs
// to set status and faults.
case V.state of
  // state 0 - valve is closed - wait for open command
  0: V.closed := 1;
     V.open := 0;
     if (V.open_command) then
       V.state := 1;
     // fault sensors
     else
       V.fault_sensors := (not sens_closed) or (sens_open);
     end_if;
  // state 1 -
  1: V.state := 2;
  // state 2 - waiting for open sensor
  2: if (sens_open & not sens_closed) then
       V.state := 3;

```

```

        // possible close command while waiting to open
        elsif not V.open_command then
            V.state := 0;
        // fault opening
        else
            V.fault_opening := (V.timecount > V.opening_preset);
        end_if;
// state 3 - open - wait for close command
3: V.closed := 0;
   V.open := 1;
   if (not V.open_command) then
       V.state := 4;
   // fault sensors
   else
       V.fault_sensors := (sens_closed) or (not sens_open);
   end_if;
// state 4 -
4: V.state := 5;
// state 5 - wait for closed sensor
5: if (sens_closed & not sens_open) then
    V.state := 0;
    // possible open command while waiting to close
    elsif V.open_command then
        V.state := 3;
    // fault closing
    else
        V.fault_closing := (V.timecount > V.closing_preset);
    end_if;
else;
end_case;
// end state machine

// reset timer if change of state
if (state_change) then V.timecount := 0;
end_if;

// external fault reset
if (V.clear_faults) then
    V.fault_opening := 0;
    V.fault_closing := 0;
    V.fault_sensors := 0;
    V.clear_faults := 0;
end_if;

```

The tags referred to in this logic are all parameters or local tags. This means that the Add-On Instruction could be used in any program (provided the UDT Valve is also present).

## Call-up

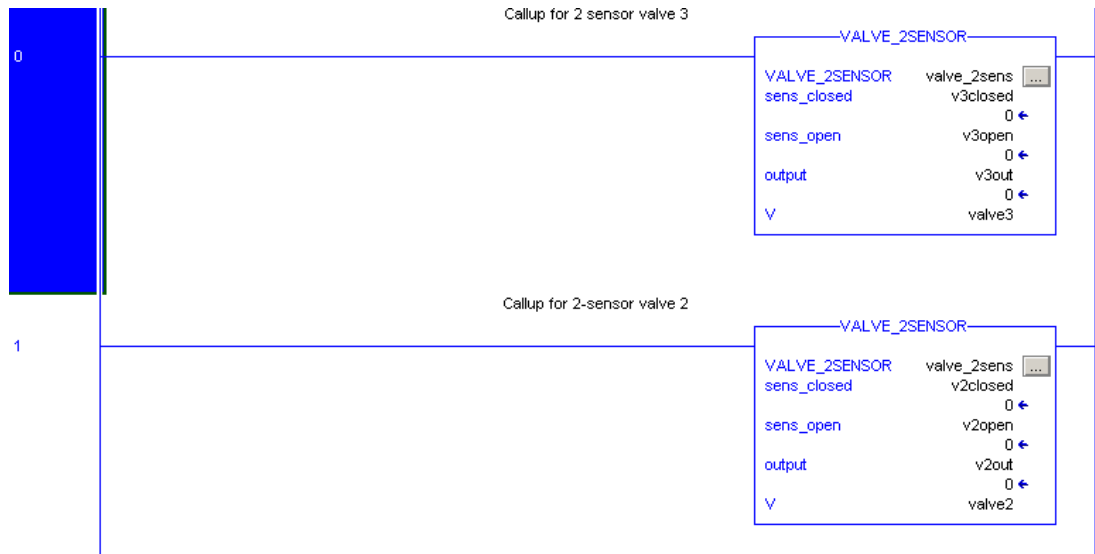
Both the call-up code and the instances of UDT Valve are located in the program “valves\_callup”, which runs under task\_02s. The frequency with which the call-up code is executed depends on the application and the size of the valve.

The screen shot below shows the data instances.

| Scope: valves_callup |   |           |          |               |
|----------------------|---|-----------|----------|---------------|
| Show... Show All     |   |           |          |               |
| Name                 | △ | Alias For | Base Tag | Data Type     |
| + valve_2sens        |   |           |          | valve_2sensor |
| + valve1             |   |           |          | Valve         |
| + valve2             |   |           |          | Valve         |
| + valve3             |   |           |          | Valve         |
| + valve4             |   |           |          | Valve         |
| + valve5             |   |           |          | Valve         |

Add an instance of type Valve for each physical valve. The first tag is the required “backing tag” for the Add-On Instruction.

The screen shot below shows the call-up code.



Call the Add-On Instruction once for each valve. The actual parameters are the actual I/O tags for the valve's sensors and solenoid, and the instance of UDT "valve".

The I/O tags will only appear in the call to the Add-On Instruction. They will not be used anywhere else in the program. Apart from being tidier from the software structure point of view, this cancels any risk of problems arising from asynchronous updating of I/O.

Remember that with Logix controllers the I/O are scanned asynchronously.



## Common Mistakes when Converting to Logix

### Introduction

The objective of this section is to point out some of the design and programming mistakes that S7 users often make when converting applications to Logix. These mistakes have been identified by examination of Logix programs that have been converted from STEP 7.

| Topic  | Page |
|--|------|
| Not Selecting Appropriate Hardware                       | 129  |
| Underestimating Impact of Task Scheduling                | 130  |
| Performing Translation Instead of Conversion             | 130  |
| Not Using the Most Appropriate Logix Languages           | 130  |
| Implementation of Incorrect Data Types – DINT versus INT | 131  |
| User Code Emulating Existing Instructions                | 132  |
| Incorrect Usage of COP, MOV, and CPS                     | 133  |
| Incorrect Usage of CPT                                   | 133  |
| Not Handling Strings in Optimal Way                      | 133  |
| Extensive Usage of Jumps                                 | 133  |
| Not Using Aliased Tags                                   | 133  |

Programming mistakes fall into these two categories:

- Programming that leads to reduced controller efficiency.
- Programming that leads to a control system that is difficult to understand, maintain, and develop.

In most cases, coding for efficiency will also improve the readability and modularity of your program. Conversely, improving the program's structure should also make it more efficient.

### Not Selecting Appropriate Hardware

This chapter is concerned mainly with software. Remember, however, that the correct selection of hardware is a requirement for satisfactory operation. It is possible that the number of controllers and racks may not be the same as for an equivalent S7 system.

Read [Chapter 1](#) and [Appendix A](#) for more about hardware. More information can be found in [Appendices A](#) and [B](#).

## Underestimating Impact of Task Scheduling

In the area of scheduling and interrupts, there isn't much difference in the capability of the two systems. However, in the Logix world, scheduling is more actively encouraged.

It is quite common for STEP 7 programmers to neglect scheduling when working with Logix controllers. Please see [Chapter 2](#) for a more detailed account of scheduling in Logix.

## Performing Translation Instead of Conversion

It is a common mistake to translate line-by-line a STEP 7 program to Logix.

Instead, a more thorough process is needed, which is described as conversion. This will cover choice of languages, scheduling and choice of code routines.

By converting rather than translating your STEP 7 programs, you will make better use of the capability of your Logix system.

## Not Using the Most Appropriate Logix Languages

Programmers often neglect Logix languages other than ladder logic.

Read [Chapter 2](#) for a discussion of how to choose a Logix language and [Chapter 4](#) for examples of STEP 7 code translated into Logix.

## Implementation of Incorrect Data Types – DINT versus INT

It is commonly advised to use DINT rather than INT.

The example below shows an addition of two DINTs v. addition of two INTs.

### Add DINTs

```
// add two DINTs and assign to a third DINT

for index := 0 to 999 do
    result_DINT := operandA_DINT + operandB_DINT;
end_for;
```

### Add INTs

```
// add two INTs and assign to a third INT

for index := 0 to 999 do
    result_INT := operandA_INT + operandB_INT;
end_for;
```

### Timing Results

The table shows relative times (smaller number is faster). The numbers here are only for comparison with other numbers in the table. They should not be compared with entries in other tables.

| Method                     | Relative Times |
|----------------------------|----------------|
| Add DINTs with ST For Loop | 53             |
| Add INTs with ST For Loop  | 100            |

For comparison, the same test was done with an S7 controller. In this case, results were identical for DINTs and INTs.

The lesson is to use DINT for all integer work in Logix. Only use INT or SINT if you are interfacing to an external system that requires the use of INTs or SINTs.

## User Code Emulating Existing Instructions

Programmers often write user code when an existing instruction will do the job. As an example, compare copying an array with user code and with the COP instruction.

### User Code

```
for index := 0 to 99 do
  target_DINT[index] := source_DINT[index];
end_for;
```

### COP Instruction

```
cop(source_DINT[0], target_DINT[0], 100);
```

Below are the relative timings for the two methods. Again, the numbers here are only for comparison with other numbers in the table. They should not be compared with entries in other tables.

| Method                                   | Relative Timing |
|--|-----------------|
| Copy array of DINTs with structured text | 100             |
| Copy array of DINTs with COP             | 18              |

To perform operations like copy arrays, STEP 7 library functions that are written in Statement List are used. If the library function doesn't do what is required, a new one can be written. The functions written can be almost as efficient as the ones that STEP 7 provides.

However in Logix, it is impossible for a programmer to write a copying function that is as efficient as the built-in COP. The lesson for S7 programmers is to check the Instruction Help in RSLogix 5000 software carefully before doing it yourself.

## **Incorrect Usage of COP, MOV, and CPS**

MOV copies a simple value (immediate or tag) to a simple tag type – DINT, INT, SINT, or REAL. COP can do the same as MOV (the source cannot be an immediate value), but its more important use is to copy complex data types.

It would be a minor programming mistake to use COP to copy simple data types.

A mistake that is seen often is to use multiple MOVs to copy a data structure when one COP could be used.

If your source data could change during copying due to asynchronous I/O updates, use CPS instead. This instruction cannot be interrupted so source data will remain constant while copying.

## **Incorrect Usage of CPT**

In Logix, the CPT instruction can be used to evaluate expressions. The expression is entered in one of the fields of the instruction. It is very convenient.

However, CPT should only be used if more than one arithmetic instruction would be required to evaluate the expression. If a single instruction is sufficient, it will be faster than CPT.

You can read more about CPT in [Chapter 4](#).

## **Not Handling Strings in Optimal Way**

If you want to define a new String type, for example with a different number of characters than the default 82, it would be a mistake to create a new 'User Data Type'. Instead, create a new String data type. The advantage of doing it this way is that the 'LEN' field will automatically update as the length of the string changes.

## **Extensive Usage of Jumps**

In Logix, jumps can only occur in Ladder Logic. It is recommended that JMP instruction be used sparingly. Jumps in ladder logic often make the program difficult to read.

## **Not Using Aliased Tags**

Remember to create aliased tags for the I/O tags that RSLogix 5000 software creates for you. They will make your program easier to read. See [Chapter 2](#).

**Notes:**

## S7 to Logix Glossary

### Introduction

This chapter provides a glossary of S7 terms and their Logix equivalents.

### Hardware Terminology

| S7 Term                  | Definition   | Logix Term                              | Definition   |
|--------------------------|--|---|--|
| Communications Processor | Comms module   | Bridge                                  |  |
| Controller               | The controller   | Controller                              |  |
| CPU                      | Central Processing Unit                                | CPU or Controller                       |  |
| Failsafe CPU             | CPU 315F-2 DP Implements PROFISAFE version of DP       | GuardLogix                              | L61S, L62S, L63S   |
| Industrial Ethernet      | Siemens version of Ethernet                            | Ethernet /IP<br>ControlNet              | Both of these have the same (or better) functionality as Industrial Ethernet |
| MPI                      | Multi-Point Interface – a serial bus                   | Serial                                  | DF1 or DH485 protocols   |
| Programmable Controller  |  | Controller or PAC                       |  |
| PROFIBUS DP              | Commonly used field bus                                | Ethernet /IP<br>ControlNet<br>DeviceNet |  |
| PROFIBUS PA              | Variety of Profibus specializing in Process Automation | As Profibus DP                          |  |
| PROFINET                 | Profibus over Ethernet                                 | Ethernet /IP                            |  |
| PROFISAFE                | Failsafe version of PROFIBUS DP                        | GuardLogix                              |  |
| S7-200                   | Low-end controllers                                    | MicroLogix                              |  |
| S7-300                   | Mid-range controllers                                  | CompactLogix                            |  |
| S7-400                   | High-end controllers                                   | ControlLogix                            |  |
| SIMATIC                  | Brand name for Siemens automation products             | Logix                                   |  |

## Software Terminology

| S7 Term             | Definition  | Nearest Logix Term   | Definition   |
|---------------------|---|--|--|
| Accumulator         | Used in STL   | N/A  | In Logix languages, there is no need to access low-level structures of the CPU                     |
| AR1, AR2            | Pointer registers                                       | N/A  | In Logix languages, there is no need to access low-level structures of the CPU                     |
| Array               | Syntax ARRAY[0...7] OF REAL                             | Array  | Syntax REAL[8]<br>Indexing always starts at 0  |
| Bit Memory          | Addresses M...  | N/A  | Use tags   |
| Block Transfer      | Copy block of data.<br>SFC20 BLK_MOV                    | COP  | Instruction<br>(use MOV for a simple variable)   |
| BOOL                |   | BOOL   |  |
| BYTE                | 8 bit word  | SINT   | Use is deprecated (it's slower than DINT) except when required (for example, characters of string) |
| CFC                 | Optional process control language                       | FBD  | Standard function block language.  |
| CHAR                | Byte as character                                       | SINT   |  |
| Cycle_Execution     | OB1 – Continuously executed                             | Continuous Task  | Continuously executed  |
| Data Block          | Unit of static data memory                              | Controller-Scope Tag database<br>or Program-scope tag database | Global<br><br>visible within the Program that the database is linked to                            |
| DINT                | Double integer  | DINT   | Double integer   |
| DWORD               | 32 bit word   | DINT   |  |
| FBD                 | Function Block Diagram                                  | FBD  | Function Block Diagram   |
| Function            | Program Unit with Temporary memory but no Static memory | Routine<br>AOI (Add-On Instruction)                            | Both of these <b>could</b> correspond to a function  |
| Function Block      | Program Unit with Temporary memory and Static memory    | Routine<br>AOI (Add-On Instruction)<br>Program                 | All of these <b>could</b> correspond to a function block   |
| GRAPH               | Optional graphical language                             | Sequential Function Chart                                      | Standard Graphical Language  |
| HW Config           | Hardware Configuration – component of STEP 7            | I/O Configuration  | Branch of Controller Organiser   |
| INT                 | Integer   | INT  | Use is deprecated (it's slower than DINT)  |
| Interrupt_Execution | Periodically executing OB                               | Periodic Task  | Periodically executing Task  |
| LAD                 | Ladder Logic  | LD   | Ladder Logic   |
| Library             | System functions  | GSV, SSV   | Instructions –<br>Get System Value<br>Set System Value   |



| <b>S7 Term</b>     | <b>Definition</b>                          | <b>Nearest Logix Term</b> | <b>Definition</b>  |
|--------------------|--|---------------------------|--|
| NetPro             | Network Configurator                       | N/A                       | Part of I/O Configuration branch of controller organiser.                                    |
| Organization Block | Program unit called by Operating System    | Task                      | Program unit called by Operating System  |
| Pointer            | Data pointer used in STL                   | N/A                       | Use arrays   |
| REAL               | 32 bit floating point number               | REAL                      | 32 bit floating point number   |
| SCL                | Optional high-level language               | Structured Text           | Standard Language  |
| Simatic Manager    | Component of STEP 7                        | Controller Organiser      | Component of RSLogix 5000  |
| STEP 7             | Development and monitoring software for S7 | RSLogix 5000              | Development and monitoring software for Logix  |
| STL                | Statement List                             | N/A                       | Use Structured Text or Ladder Logic or Sequential Function Chart                             |
| STRING             | Sequence of CHARs. Default Length 254      | STRING                    | Sequence of SINTs. Default length 82. String object also contains its length as property.LEN |
| STRUCT             | Untyped collection of data                 | N/A                       | In Logix a structure is an instance of type (UDT)  |
| Symbol             | Name for data memory address               | Tag                       | Tag defines the structure of the variable <i>and</i> reserves memory                         |
| Temporary memory   | Memory created on run-time stack           | N/A                       | Use tags   |
| WORD               | 16 bit word                                | INT                       |  |
| UDT                | User Data Type                             | UDT                       | User Data Type   |

**Notes:**

## **S7 300 and S7 400 Parts and RA Equivalents**

### **Introduction**

This appendix lists Siemens products and their Rockwell Automation equivalents.

| <b>Topic</b>                        | <b>Page</b> |
|-------------------------------------|-------------|
| Compact S7 300 CPUs                 | 140         |
| Standard S7 300 CPUs                | 140         |
| Technology S7 300 CPUs              | 141         |
| Fail-Safe S7 300 CPUs               | 142         |
| S7 300 Digital Input Modules        | 142         |
| S7 300 Digital Output Modules       | 143         |
| S7 300 Relay Output Modules         | 144         |
| S7 300 Digital Combo Modules        | 144         |
| S7 300 Analog Input Modules         | 144         |
| S7 300 Analog Output Modules        | 145         |
| S7 300 Analog Combo Modules         | 146         |
| S7 300 Analog Output Modules        | 146         |
| Redundant and Fail Safe Controllers | 147         |
| Digital Input Modules               | 147         |
| Digital Output Modules              | 147         |
| Analog Input Modules                | 148         |
| Analog Output Modules               | 148         |

## Compact S7 300 CPUs

| Siemens Catalogue Number | Siemens Short Reference | Memory              |     | Comms Ports |           | Max MMC Size |    | Embedded I/O |    |    | RA Solution                   |
|--------------------------|-------------------------|---------------------|-----|-------------|-----------|--------------|----|--------------|----|----|-------------------------------|
|                          |                         |                     | MPI | DP          | Serial    |              | DI | DO           | AI | AO |                               |
| 6ES7 312-5BE0x-xxxx      | S7-312C                 | 32K                 | Y   | N           | N         | 4 MB         | 10 | 6            |    |    | 1769-L31 + Compact I/O ML1500 |
| 6ES7 313-5BF0x-xxxx      | S7-313C                 | 64K Yes No No       | Y   | N           | N         | 8 MB         | 24 | 16           | 4  | 2  | 1769-L31 + Compact I/O ML1500 |
| 6ES7 313-6BF0x-xxxx      | S7-313C-PtP             | 64K                 | Y   | N           | RS422/485 | 8 MB         | 16 | 16           |    |    | 1769-L31 + Compact I/O ML1500 |
| 6ES7 313-6CF0x-xxxx      | S7-313C-DP              | 64 K                | Y   | Y           | N         | 8 MB         | 16 | 16           |    |    | 1769-L31 + Compact I/O ML1500 |
| 6ES7 314-6BG0x-xxx x     | S7-314C-PtP             | 96K                 | Y   | N           | RS422/485 | 8 MB         | 24 | 16           | 4  | 2  | 1769-L31 + Compact I/O ML1500 |
| 6ES7 314-6CG0x-xxx x     | S7-314C-DP              | 96K Yes Yes No 8 MB | Y   | Y           | N         | 8 MB         | 24 | 16           | 4  | 2  | 1769-L31 + Compact I/O ML1500 |

## Standard S7 300 CPUs

| Siemens Catalogue Number | Siemens Short Reference | Memory |     | Comms Ports |    | Max Load Memory Size (RAM) | RA Solution            |
|--------------------------|-------------------------|--------|-----|-------------|----|----------------------------|------------------------|
|                          |                         |        | MPI | DP          | PN |                            |                        |
| 6ES7 312-1AE1x-xxxx      | S7-312                  | 32K    | Y   | N           | N  | 4 MB                       | 1769-L31               |
| 6ES7 314-1AG1x-xxxx      | S7-314                  | 96K    | Y   | N           | N  | 8 MB                       | 1769-L31               |
| 6ES7 315-2AG1x-xxxx      | S7-315-2 DP             | 128K   | Y   | Y           | N  | 8 MB                       | 1769-L3xE or 1769-L3xC |

| Siemens<br>Catalogue<br>Number | Siemens Short<br>Reference | Memory | Comms<br>Ports |    |    | Max Load<br>Memory<br>Size (RAM) | RA Solution               |
|--------------------------------|----------------------------|--------|----------------|----|----|----------------------------------|---------------------------|
|                                |                            |        | MPI            | DP | PN |                                  |                           |
| 6ES7<br>315-2EH1x-xxxx         | S7-315-2 PN/DP             | 256K   | Y              | Y  | Y  | 8 MB                             | 1769-L3xE or<br>1769-L3xC |
| 6ES7<br>317-2AJ1x-xxxx         | S7-317-2 DP                | 512K   | Y              | Y  | N  | 8 MB                             | 1769-L3xE or<br>1769-L3xC |
| 6ES7<br>317-2EK1x-xxxx         | S7-317-2 PN/DP             | 1 MB   | Y              | Y  | Y  | 8 MB                             | 1769-L3xE or<br>1769-L3xC |
| 6ES7<br>319-3ELOx-xxxx         | S7-319-3 PN/DP             | 1.4 MB | Y              | Y  | Y  | 8 MB                             | 1769-L3xE or<br>1769-L3xC |

## Technology S7 300 CPUs

| Siemens<br>Catalogue<br>Number | Siemens Short<br>Reference | Memory | Comms<br>Ports |    |    | Max Load<br>Memory<br>Size (RAM) | RA Solution |
|--------------------------------|----------------------------|--------|----------------|----|----|----------------------------------|-------------|
|                                |                            |        | MPI            | DP | PN |                                  |             |
| 6ES7<br>315-6TG1x-xxxx         | S7-315T-2 DP               | 128K   | Y              | Y  | Y  | 4 or 8 MB                        | 1768-L43    |
| 6ES7<br>317-6TJ1x-xxxx         | S7-317T-2 DP               | 512K   | Y              | Y  | Y  | 4 or 8 MB                        | 1768-L43    |

## Fail-Safe S7 300 CPUs

| Siemens Catalogue Number | Siemens Short Reference | Memory |     | Comms Ports |    | Max Load Memory Size (RAM) | RA Solution ControlLogix     |
|--------------------------|-------------------------|--------|-----|-------------|----|----------------------------|------------------------------|
|                          |                         |        | MPI | DP          | PN |                            |                              |
| 6ES7 315-6FF1x-xxxx      | S7-315F-2 DP            | 192K   | Y   | Y           | N  | 8 MB                       | GuardLogix or SmartGuard 600 |
| 6ES7 315-2FH1x-xxxx      | S7-315F-2 PN/DP         | 256K   | Y   | Y           | Y  | 8 MB                       | GuardLogix or SmartGuard 600 |
| 6ES7 317-6FF0x-xxxx      | S7-317F-2 DP            | 1 MB   | Y   | Y           | N  | 8 MB                       | GuardLogix or SmartGuard 600 |
| 6ES7 317-2FK1x-xxxx      | S7-317F-2 PN/DP         | 1 MB   | Y   | Y           | Y  | 8 MB                       | GuardLogix or SmartGuard 600 |

## S7 300 Digital Input Modules

| Siemens Catalogue Number | Front Connector | Points | Range           | RA Solution             | Comments                      |
|--------------------------|-----------------|--------|-----------------|-------------------------|-------------------------------|
| 6ES7 321-1BH0x-xxxx      | 20-pin          | 16     | 24 VDC          | 1769-IQ16<br>1769-IQ16F |                               |
| 6ES7 321-1BH5x-xxxx      | 20-pin          | 16     | 24 VDC          | 1769-IQ16<br>1769-IQ16F |                               |
| 6ES7 321-1BL0x-xxxx      | 40-pin          | 32     | 24 VDC          | 1769-IQ32<br>1769-IQ32T |                               |
| 6ES7 321-1CH0x-xxxx      | 40-pin          | 16     | 24 ... 48 V     | n/a                     |                               |
| 6ES7 321-1CH2x-xxxx      | 20-pin          | 16     | 48 ... 125 VDC  | n/a                     |                               |
| 6ES7 321-1BH1x-xxxx      | 20-pin          | 16     | 24 VDC          | 1769-IQ16<br>1769-IQ16F |                               |
| 6ES7 321-7BH0x-xxxx      | 20-pin          | 16     | 24 VDC          | 1769-IQ16<br>1769-IQ16F |                               |
| 6ES7 321-1FH0x-xxxx      | 20-pin          | 16     | 120 ... 230 VAC | 1769-IA16               | 1769-IA16 only admits 120 VAC |
| 6ES7 321-1FF0x-xxxx      | 20-pin          | 8      | 120 ... 230 VAC | 1769-IM12               | 1769-IM12 only admits 230 VAC |

|                     |        |    |                 |           |                                     |
|---------------------|--------|----|-----------------|-----------|-------------------------------------|
| 6ES7 321-1FF1x-xxxx | 40-pin | 8  | 120 ... 230 VAC | 1769-IA8I | 1769-IA8I<br>only admits<br>120 VAC |
| 6ES7 321-1E0x-xxxx  | 40-pin | 32 | 120 VAC         | n/a       |                                     |
| n/a                 |        | 16 | 5 VDC TTL       | 1769-IG16 |                                     |

## S7 300 Digital Output Modules

| Siemens Catalogue Number | Front Connector | Points | Range          | Output Current | RA Solution             | Comments                    |
|--------------------------|-----------------|--------|----------------|----------------|-------------------------|-----------------------------|
| 6ES7 332-1FH0x-xxxx      | 20-pin          | 16     | 120/230 VAC    | 0.5 A          | 1769-OA16               |                             |
| 6ES7 332-1FF0x-xxxx      | 20-pin          | 8      | 120/230 VAC    | 2 A            | 1769-OA8                | S7-300 has fuse per group   |
| 6ES7 332-5FF0x-xxxx      | 40-pin          | 8      | 120/230 VAC    | 2 A            | 1769-OA8                | S7-300 comes in groups of 1 |
| 6ES7 322-1BH0x-xxxx      | 20-pin          | 16     | 24 VDC         | 0.5 A          | 1769-OB16<br>1769-OB16P |                             |
| 6ES7 322-1BH1x-xxxx      | 20-pin          | 16     | 24 VDC         | 0.5 A          | n/a                     | High Speed                  |
| 6ES7 322-1BL0x-xxxx      | 40-pin          | 32     | 24 VDC         | 0.5 A          | 1769-OB32<br>1769-OB32T |                             |
| 6ES7 322-1BF0x-xxxx      | 20-pin          | 8      | 24 VDC         | 2 A            | 1769-OB8                |                             |
| 6ES7 322-8BF0x-xxxx      | 20-pin          | 8      | 24 VDC         | 0.5 A          | 1769-OB8                |                             |
| 6ES7 332-1FL0x-xxxx      | 2x20-pin        | 32     | 120 VAC        | 1 A            | n/a                     |                             |
| 6ES7 332-5GH0x-xxxx      | 40-pin          | 16     | 24/48 V        | 0.5 A          | n/a                     |                             |
| 6ES7 332-1CF0x-xxxx      | 20-pin          | 8      | 48 ... 125 VDC |                | n/a                     |                             |
| n/a                      |                 | 16     | 5 VDC TTL      |                | 1769-OG16               |                             |
| n/a                      |                 | 16     | 24 VDC         |                | 1769-OV16               |                             |
| n/a                      |                 | 32     | 24 VDC         |                | 1769-OV32T              |                             |
| n/a                      |                 | 16     | 24 VDC         |                | 1769-OB16P              |                             |

## S7 300 Relay Output Modules

| Siemens Catalogue Number | Front Connector | Points | Output Current | RA Solution | Comments  |
|--------------------------|-----------------|--------|----------------|-------------|---|
| 6ES7 322-1HH0x-xxxx      | 20-pin          | 16     | 2 A            | 1769-OW16   |   |
| 6ES7 322-1HF0x-xxxx      | 20-pin          | 8      | 5 A            | 1769-OW8    |   |
| 6ES7 322-1HF1x-xxxx      | 40-pin          | 8      | 5 A            | 1769-OW8I   |   |
| 6ES7 322-5HF0x-xxxx      | 40-pin          | 8      | 8 A            | 1769-OW8I   | S7-300 module comes with RC filter and overvoltage protection |

## S7 300 Digital Combo Modules

| Siemens Catalogue Number | Front Connector | Points  | Range Inputs | Output Current | RA Solution  | Comments  |
|--------------------------|-----------------|---------|--------------|----------------|--------------|---|
| 6ES7 323-1BH0x-xxxx      | 20-pin          | 8 / 8   | 24 VDC       | 24 VDC / 0.5 A | 1769-IQ6XOW4 | Compact I/O has less I/Os and outputs are relay |
| 6ES7 323-1BL0x-xxxx      | 40-pin          | 16 / 16 | 24 VDC       | 24 VDC / 0.5 A | n/a          |   |
| 6ES7 327-1BH0x-xxxx      | 20-pin          | 8 / 8   | 24 VDC       | 24 VDC / 0.5 A | n/a          | 8 inputs; 8 inputs or outputs (configurable)    |

## S7 300 Analog Input Modules

| Siemens Catalogue Number | Front Connector | Points | Resolution (bits) | Type                                     | Compact I/O Solution     | Comments |
|--------------------------|-----------------|--------|-------------------|--|--------------------------|----------|
| 6ES7 331-1KF0x-xxxx      | 40              | 8      | 13                | Voltage, Current, Resistance Temperature | 1769sc-IF8U<br>1769-IF8U |          |
| 6ES7 331-7KF0x-xxxx      | 20              | 8      | 9 / 12 / 14       | Voltage, Current, Resistance Temperature | 1769sc-IF8U<br>1769-IF8U |          |



|                     |    |   |             |   |                         |  |
|---------------------|----|---|-------------|---|-------------------------|--|
| 6ES7 331-7KB0x-xxxx | 20 | 2 | 9 / 12 / 14 | Voltage,<br>Current,<br>Resistance<br>Temperature | 1769sc-IF8U<br>1769-IF4 |  |
| 6ES7 331-7NF0x-xxxx | 40 | 8 | 16          | Voltage<br>Voltage                                | 1769-IF8                |  |
| 6ES7 331-7NF1x-xxxx | 40 | 8 | 16          | Voltage<br>Voltage                                | 1769-IF8                | Includes<br>hardware<br>interrupt at end<br>of cycle vs 6ES7<br>331-7NF0x-xxxx |
| 6ES7 331-7HF0x-xxxx | 20 | 8 | 14          | Voltage<br>Voltage                                | 1769-IF8                |  |
| 6ES7 331-7PF0x-xxxx | 40 | 8 |             | RTD<br>Resistance                                 | 1769-IR6                |  |
| 6ES7 331-7PF1x-xxxx | 40 | 8 |             | Thermocouple                                      | 1769-IT6                |  |
| n/a                 |    |   |             |   | 1769-IF4I               |  |

## S7 300 Analog Output Modules

| Siemens Catalogue Number | Front Connector | Points | Resolution (bits) | Type               | RA Solution              | Comments |
|--------------------------|-----------------|--------|-------------------|--------------------|--------------------------|----------|
| 6ES7 332-5HD0x-xxxx      | 40              | 4      | 12                | Voltage<br>Current | 1769-OF4VI<br>1769-OF4CI |          |
| 6ES7 332-7ND0x-xxxx      | 20              | 4      | 16                | Voltage<br>Current | 1769-OF4VI<br>1769-OF4CI |          |
| 6ES7 332-5HB0x-xxxx      | 20              | 2      | 12                | Voltage<br>Current | 1769-OF2                 |          |
| 6ES7 332-5HF0x-xxxx      | 20              | 8      | 12                | Voltage<br>Current | 1769-OF8V<br>1769-OF8C   |          |

## S7 300 Analog Combo Modules

| Siemens Catalogue Number | Front Connector | Points | Resolution (bits) | Type   | RA Solution  | Comments                |
|--------------------------|-----------------|--------|-------------------|--|--------------|-------------------------|
| 6ES7 334-0KE0x-xxxx      | 20              | 4 / 2  | 12                | Voltage<br>Current<br>Pt 100                 |              | Outputs only<br>Voltage |
| 6ES7 334-0CE0x-xxxx      | 20              | 4 / 2  | 8                 | Voltage and<br>Current (Inputs<br>& Outputs) | 1769-IF4XOF2 |                         |

## S7 400 Standard Controllers

| Siemens Catalogue Number | Siemens Short Reference | Work Memory Size | Comms Ports |    |    | Max Load Memory Size (RAM) | RA Solution ControlLogix |
|--------------------------|-------------------------|------------------|-------------|----|----|----------------------------|--------------------------|
|                          |                         |                  | MPI         | DP | PN |                            |                          |
| 6ES7 412-1XF04-0AB0      | CPU 412-1               | 144KB            | Y           | Y  | N  | 64MB                       | 1756-L61                 |
| 6ES7 412-2GX04-0AB0      | CPU 412-2               | 256KB            | Y           | Y  | N  | 64MB                       | 1756-L61                 |
| 6ES7 414-2GX04-0AB0      | CPU 414-2               | 512KB            | Y           | Y  | N  | 64MB                       | 1756-L62                 |
| 6ES7 414-3XJ04-0AB0      | CPU 414-3               | 1.4MB            | Y           | Y  | N  | 64MB                       | 1756-L63                 |
| 6ES7 414-3EM05-0AB0      | CPU 414-3<br>PN/DP      | 2.8MB            | Y           | Y  | Y  | 64MB                       | 1756-L63                 |
| 6ES7 416-3XK04-0AB0      | CPU 416-2               | 2.8MB            | Y           | Y  | N  | 64MB                       | 1756-L63                 |
| 6ES7 416-3XL04-0AB0      | CPU 416-3               | 5.6MB            | Y           | Y  | N  | 64MB                       | 1756-L64                 |
| 6ES7 416-3ER05-0AB0      | CPU 416-3<br>PN/DP      | 11.2 MB          | Y           | Y  | Y  | 64MB                       | 1756-L64                 |
| 6ES7 417-4XL04-0AB0      | CPU 417-4               | 20MB             | Y           | Y  | N  | 64MB                       | 1756-L64                 |

## Redundant and Fail Safe Controllers

| Siemens Catalogue Number | Siemens Short Reference | Work Memory Size |     | Comms Ports |    |            | Max Load Memory Size (RAM) | RA Solution ControlLogix |
|--------------------------|-------------------------|------------------|-----|-------------|----|------------|----------------------------|--------------------------|
|                          |                         |                  | MPI | DP          | PN | Sync ports |                            |                          |
| 6ES7 414-4HJ04-0AB0      | CPU 414-4H              | 1.4MB            | Y   | Y           | N  | Y          | 64MB                       | 1756-L63                 |
| 6ES7 417-4HL04-0AB0      | CPU 417-4H              | 20MB             | Y   | Y           | N  | Y          | 64MB                       | 1756-L64                 |
| 6ES7 416-2FK04-0AB0      | CPU-416F-2              | 2.6MB            | Y   | Y           | N  | N          | 64MB                       | 1756-L61S                |

## Digital Input Modules

| Siemens Catalogue Number                       | Front Connector | Points | Range        | RA Solution | Comments |
|--|-----------------|--------|--------------|-------------|----------|
| 6ES7 421-7BH01-0AB0<br>(Interrupt/ diagnostic) | 48 pin          | 16     | 24V DC       | 1756-IB16D  |          |
| 6ES7 421-1BL01-0AA0                            | 48 pin          | 32     | 24V DC       | 1756-IB32   |          |
| 6ES7 421-1EL00-0AA0                            | 48 pin          | 32     | 120V AC/DC   | 1756-IA32   |          |
| 6ES7 421-1FH20-0AA0                            | 48 pin          | 16     | 230V AC/DC   | 1756-IM161  |          |
| 6ES7 421-7DH00 0AB0<br>(Interrupt/ diagnostic) | 48 pin          | 32     | 24-60V AC/DC |             |          |

## Digital Output Modules

| Siemens Catalogue Number            | Front Connector | Points | Range                     | Current | RA Solution             | Comments |
|-------------------------------------|-----------------|--------|---------------------------|---------|-------------------------|----------|
| 6ES7 422-1FH00-0AA0                 | 48 pin          | 16     | 230VAC                    | 2A      | 1756-OA16               |          |
| 6ES7 422-1HH00-0AA0                 | 48 pin          | 16     | 60V DC/230V AC<br>(relay) | 5A      | 1756-OW16I              |          |
| 6ES7 422 1BH11-0AA0                 | 48 pin          | 16     | 24VDC                     | 2A      | 1756-OB16E              |          |
| 6ES7 422-1BL00-0AA0                 | 48 pin          | 32     | 24VDC                     | 0.5A    | 1756-OB32               |          |
| 6ES7 422-7BL00-0AB0<br>(Diagnostic) | 48 pin          | 32     | 24VDC                     | 0.5A    | 1756-OB16D<br>1756-OB32 |          |

## Analog Input Modules

| Siemens Catalogue Number           | Front Connector | Channels | Resolution (bits) | Type  | RA Solution            | Comments                               |
|------------------------------------|-----------------|----------|-------------------|---|------------------------|--|
| 6ES7 431-0HH0-0AB0                 | 48 pin          | 16       | 13                | Voltage<br>Current  | 1756-IF16              | 16 bits                                |
| 6ES7 431-1KF00-0AB0                | 48 pin          | 8        | 13                | Voltage<br>Current<br>Impedance                                   | 1756-IF8               | 16 bits<br>4 differential inputs       |
| 6ES7 431-1KF10-0AB0                | 48 pin          | 8        | 14-16             | Voltage<br>Current<br>Thermocouple<br>Thermoresistor<br>Impedance | 1756-IR6I<br>1756-IT6I | 6 RTD<br>6 Thermocouple<br>Both 16 bit |
| 6ES7 431-1FK20-0AB0                | 48 pin          | 8        | 14                | Voltage<br>Current<br>Impedance                                   | 1756-IF16              | 16 bit                                 |
| 6ES7 431-7QH00-0AB0<br>(Interrupt) | 48 pin          | 16       | 16                | Voltage<br>Current<br>Thermocouple<br>Thermoresistor<br>Impedance | 1756-IR6I<br>1756-IT6I | 6 RTD<br>6 Thermocouple                |
| 6ES7 431-7KF00-0AB0                | 48 pin          | 8        | 16                | Voltage<br>Current<br>Thermocouple                                | 1756-IT6I              | 6 channels                             |
| 6ES7 431-7KF01-0AB0                | 48 pin          | 8        | 16                | Thermoresistor  | 1756-IR6I              | 5 channels                             |

## Analog Output Modules

| Siemens Catalogue Number | Front Connector | Channels | Resolution (bits) | Type               | RA Solution | Comments |
|--------------------------|-----------------|----------|-------------------|--------------------|-------------|----------|
| 6ES7 432-1HF00-0AB0      | 48 pin          | 8        | 13                | Voltage<br>Current | 1756-OF8    | 15 bits  |

## Siemens HMI Cross Reference Table

Use this appendix to compare Rockwell Automation panels to specific types of Siemens panels.

| Topic   | Page |
|---|------|
| SIMATIC Micro Panels and Rockwell Automation Equivalents              | 149  |
| SIMATIC Panels - 7x Series and Rockwell Automation Equivalents        | 151  |
| SIMATIC Panels - 17x Series and Rockwell Automation Equivalents       | 152  |
| SIMATIC Panels - 27x Series and Rockwell Automation Equivalents       | 155  |
| SIMATIC Multi Panels - 27x Series and Rockwell Automation Equivalents | 157  |
| SIMATIC Multi Panels - 37x Series and Rockwell Automation Equivalents | 159  |

### SIMATIC Micro Panels and Rockwell Automation Equivalents

| SIMATIC Micro Panels   |   |   |        |   | Rockwell Automation Solution       |                                     |  |
|------------------------|---|---|--------|---|------------------------------------|-------------------------------------|--|
| Siemens Catalog Number | Short Reference                           | Description   | Mem.   | Comm. Options                               | Rockwell Automation Catalog Number | Name                                | Description  |
| 6AV6640-0BA11-0AX0     | SIMATIC OP 73MICRO                        | 3 in. STN monochrome display, 160x48 pixels, keypad, 24V DC only              | 128 KB | 1xRS485, S7-200 compatible, no printer port | 2711P-K4M5D                        | PanelView Plus 400 grayscale keypad | 3.8 in. STN 32-level grayscale display, 320 x 240 pixels, RS-232 communication, keypad, 24V DC, 64 MB flash, USB printing capabilities |
| 6AV6545-0AA15-2AX0     | SIMATIC TP070<br>Phased out in April 2007 | 5.7 in. STN display, Blue mode (4 levels), 320x240 pixels, touch, 24V DC only | 128 KB | 1xRS485, S7-200 compatible, no printer port | 2711P-T6M5D                        | PanelView Plus 600 grayscale touch  | 5.5 in. STN 32-level grayscale display, 320 x 240 pixels, RS-232 communication, touch, 24V DC, USB printing capabilities               |

| SIMATIC Micro Panels   |   |  |        |   | Rockwell Automation Solution       |  |  |
|------------------------|---|--|--------|---|------------------------------------|--|--|
| Siemens Catalog Number | Short Reference                                     | Description  | Mem.   | Comm. Options                               | Rockwell Automation Catalog Number | Name                                   | Description  |
| 6AV6640-OCA01-0AX0     | SIMATIC TP 170MICRO<br><br>Phased out in April 2007 | 5.7 in. STN display, Blue mode (4 levels), 320x240 pixels, touch, 24V DC only, limited application functionality | 256 KB | 1xRS485, S7-200 compatible, no printer port | 2711P-T6M5D                        | PanelView Plus 600 grayscale touch     | 5.5 in. STN 32-level grayscale display, 320x240 pixels, RS-232 communication, touch, 24V DC, USB printing capabilities |
| 6AV6640-OCA11-0AX0     | SIMATIC TP 177MICRO                                 | 5.7 in. STN display, Blue mode (4 levels), 320x240 pixels, touch, 24V DC only                                    | 256 KB | 1xRS485, S7-200 compatible, no printer port | 2711P-T6M5D                        | PanelView Plus 600 grayscale touch     | 5.5 in. STN 32-level grayscale display, 320x240 pixels, RS-232 communication, touch, 24V DC, USB printing capabilities |
| 6AV6610-OAA01-1CA8     | WINCC FLEXIBLE MICRO software                       | Configuration and programming software for Simatic micro panels only   | N/A    | N/A   | 9701-VWSTMENE                      | RSView Studio Machine Edition software | RSView Studio for Machine Edition configuration software for developing and testing machine level HMI applications     |

## SIMATIC Panels - 7x Series and Rockwell Automation Equivalents

| SIMATIC Panels - 7x Series |                                 |   |        |  | Rockwell Automation Solution       |  |  |
|----------------------------|---------------------------------|---|--------|--|------------------------------------|--|--|
| Siemens Catalog Number     | Short Reference                 | Description   | Mem.   | Comm. Options  | Rockwell Automation Catalog Number | Name                                       | Description  |
| 6AV6641-0AA11-0AX0         | SIMATIC OP73                    | 3 in. STN monochrome display, 160x48 pixels, keypad, 24V DC only                    | 256 KB | 1x RS485, S7-200, S7-300/400 compatible, no printer port                   | 2711P-K4M5D                        | PanelView Plus 400 grayscale keypad        | 3.8 in. STN 32-level grayscale display, 320x240 pixels, RS-232 communication, keypad, 24V DC, 64 MB flash, USB printing capabilities |
| 6AV6641-0BA11-0AX0         | SIMATIC OP77A                   | 4.5 in. STN monochrome display, 160x64 pixels, keypad, 24V DC only                  | 256 KB | 1xRS422, 1xRS485, S7-200, S7-300/400, no printer port                      | 2711P-K4M5D                        | PanelView Plus 400 grayscale keypad        | 3.8 in. STN 32-level grayscale display, 320x240 pixels, RS-232 communication, keypad, 24V DC, 64 MB flash, USB printing capabilities |
| 6AV6641-0CA01-0AX0         | SIMATIC OP77B                   | 4.5 in. STN monochrome display, 160x64 pixels, keypad, 24V DC only                  | 1 MB   | 1xRS232, 1xRS422, 1xRS485, USB, S7-200, S7-300/400, printer port available | 2711P-K4M5D                        | PanelView Plus 400 grayscale keypad        | 3.8 in. STN 32-level grayscale display 320x240 pixels, RS-232 communication, keypad, 24V DC, 64 MB flash, USB printing capabilities  |
| 6AV6621-0AA01-0AA0         | WINCC FLEXIBLE COMPACT software | Configuration and programming software for Simatic OP77, OP/TP170, and micro panels | N/A    | N/A  | 9701-VWSTMENE                      | RSView Studio for Machine Edition software | RSView Studio Machine Edition configuration software for developing and testing machine level HMI applications                       |

## SIMATIC Panels - 17x Series and Rockwell Automation Equivalents

| SIMATIC Panels - 17x Series |  |  |        |  | Rockwell Automation Solution       |   |   |
|-----------------------------|--|--|--------|--|------------------------------------|---|---|
| Siemens Catalog Number      | Short Reference  | Description  | Mem.   | Comm. Options  | Rockwell Automation Catalog Number | Name  | Description   |
| 6AV6545-OBA15-2AX0          | SIMATIC TP170A Blue mode<br><br>Phased out in April 2007 | 5.7 in. STN display, Blue mode (4 levels), 320x240 pixels, touch, 24V DC only            | 320 KB | 1xRS232, 1xRS422, 1xRS485, S5, S7-200, S7-300/400, and third-party controllers, no printer port              | 2711P-T6M20D                       | PanelView Plus 600 grayscale touch            | 5.5 in. STN 32-level grayscale display, 320x240 pixels, EtherNet/IP, RS-232 communication, touch, 24V DC, 64 MB flash, USB printing capabilities            |
| 6AV6545-0BB15-2AX0          | SIMATIC TP170B Blue mode<br><br>Phased out in April 2007 | 5.7 in. STN display, Blue mode (4 levels), 320x240 pixels, touch, 24V DC only            | 768 KB | 2xRS232, 1xRS422, 1xRS485, S5, S7-200, S7-300/400, and third-party controllers, printer port available       | 2711P-T6M20D                       | PanelView Plus 600 grayscale touch            | 5.5 in. STN 32-level grayscale display, 320x240 pixels, EtherNet/IP, RS-232 communication, touch, 24V DC, 64 MB flash, USB printing capabilities            |
| 6AV6545-0BC15-2AX0          | SIMATIC TP170B color<br><br>Phased out in April 2007     | 5.7 in. STN display, color (256 colors), 320x240 pixels, touch. 24V DC only              | 768 KB | 2xRS232, 1xRS422, 1xRS485, S5, S7-200, S7-300/400, and third-party controllers, printer port available       | 2711P-T6C20D                       | PanelView Plus 600 color touch                | 5.5 in. TFT color display, 320x240 pixels, 18-bit color depth, EtherNet/IP, RS-232 communication, touch, 24V DC, 64 MB flash, USB printing capabilities     |
| 6AV6542-0BB15-2AX0          | SIMATIC OP170B Blue mode<br><br>Phased out in April 2007 | 5.7 in. STN display, Blue mode (4 levels), 320x240 pixels, keypad and touch, 24V DC only | 768 KB | 2xRS232, 1xRS422, 1xRS485, S5, S7-200, S7-300/400, and third-party controllers, printer port available       | 2711P-B6M20D                       | PanelView Plus 600 grayscale touch and keypad | 5.5 in. STN 32-level grayscale display, 320x240 pixels, EtherNet/IP, RS-232 communication, touch and keypad, 24V DC, 64 MB flash, USB printing capabilities |
| 6AV6642-0DC01-1AX0          | SIMATIC OP177B Blue mode                                 | 5.7 in. STN display, Blue mode (4 levels), 320x240 pixels, keypad and touch, 24V DC only | 2 MB   | 1xRS422, 1xRS485, USB, Ethernet, S5, S7-200, S7-300/400, and third-party controllers, printer port available | 2711P-B6M20D                       | PanelView Plus 600 grayscale touch and keypad | 5.5 in. STN 32-level grayscale display, 320x240 pixels, EtherNet/IP, RS-232 communication, touch and keypad, 24V DC, 64 MB flash, USB printing capabilities |



| SIMATIC Panels - 17x Series |                          |   |        |  | Rockwell Automation Solution       |                                    |  |
|-----------------------------|--------------------------|---|--------|--|------------------------------------|------------------------------------|--|
| Siemens Catalog Number      | Short Reference          | Description   | Mem.   | Comm. Options  | Rockwell Automation Catalog Number | Name                               | Description  |
| 6AV6642-0AA11-0AX0          | SIMATIC TP177A Blue mode | 5.7 in. STN display, Blue mode (4 levels), 320x240 pixels, touch, 24V DC only | 512 KB | 1xRS422, 1xRS485, S7-200, S7-300/400 compatible, no printer port   | 2711P-T6M20 D                      | PanelView Plus 600 Grayscale Touch | 5.5-inch STN 32-level Grayscale Display, Display 320 x 240, pixels, EtherNet/IP, RS-232 Communications, Touch, 24VDC, 64 MB Flash, USB Printing capabilities |
| 6AV6642-0BA01-1AX0          | SIMATIC TP177B color     | 5.7 in. STN display, color (256 colors), 320x240 pixels, touch. 24V DC only   | 2 MB   | 1xRS422, 1xRS485, USB, Ethernet, S5, S7-200, S7-300/400, and third-party controllers, printer port available | 2711P-T6C20 D                      | PanelView Plus 600 Color Touch     | 5.5-inch TFT Color display; 320 x 240 pixels, 18-bit Color depth, EtherNet/IP, RS-232 Communications, Touch, 24VDC, 64 MB Flash, USB Printing capabilities   |
| 6AV6642-0BC01-1AX0          | SIMATIC TP177B Blue mode | 5.7 in. STN display, Blue mode (4 levels), 320x240 pixels, touch, 24V DC only | 2 MB   | 1xRS422, 1xRS485, USB, S5, S7-200, S7-300/400, and third-party controllers, printer port available           | 2711P-T6M20 D                      | PanelView Plus 600 Grayscale Touch | 5.5-inch STN 32-level Grayscale Display, Display 320 x 240, pixels, EtherNet/IP, RS-232 Communications, Touch, 24VDC, 64 MB Flash, USB Printing capabilities |

| SIMATIC Panels - 17x Series |                                      |  |      |  | Rockwell Automation Solution       |   |  |
|-----------------------------|--------------------------------------|--|------|--|------------------------------------|---|--|
| Siemens Catalog Number      | Short Reference                      | Description  | Mem. | Comm. Options  | Rockwell Automation Catalog Number | Name                                      | Description  |
| 6AV6642-8BA10-0AA0          | SIMATIC TP177B color stainless steel | 5.7 in. STN display, color (256 colors), 320x240 pixels, touch, 24V DC only, stainless steel bezel | 2 MB | 1xRS422, 1xRS485, USB, Ethernet, S5, S7-200, S7-300/400, and third-party controllers, printer port available | 2711P-T6C20D                       | PanelView Plus 600 color touch            | 5.5 in. TFT color display, 320x240 pixels, 18-bit color depth, EtherNet/IP, RS-232 communication, touch, 24V DC, 64 MB flash, USB printing capabilities            |
| 6AV6642-0DA01-1AX0          | SIMATIC OP177B color                 | 5.7 in. STN display, color (256 colors), 320x240 pixels, keypad and touch, 24V DC only             | 2 MB | 1xRS422, 1xRS485, USB, Ethernet, S5, S7-200, S7-300/400, and third-party controllers, printer port available | 2711P-B6C20D                       | PanelView Plus 600 color touch and keypad | 5.5 in. TFT color display, 320x240 pixels, 18-bit color depth, EtherNet/IP, RS-232 communication, touch and keypad, 24V DC, 64 MB flash, USB printing capabilities |
| 6AV6621-0AA01-0AA0          | WINCC FLEXIBLE COMPACT software      | Configuration and programming software for Simatic OP77, OP/TP170 & micro panels                   | N/A  | N/A  | 9701-VWSTMENE                      | RSView Studio Machine Edition software    | RSView Studio Machine Edition configuration software for developing and testing machine level HMI applications   |

## SIMATIC Panels - 27x Series and Rockwell Automation Equivalents

| SIMATIC Panels - 27x Series |  |   |      |  | Rockwell Automation solution       |                                  |  |
|-----------------------------|--|---|------|--|------------------------------------|----------------------------------|--|
| Siemens Catalog Number      | Short Reference  | Description   | Mem. | Comm. Options  | Rockwell Automation Catalog Number | Name                             | Description  |
| 6AV6545-0CA10-0AX0          | SIMATIC TP270 6 in. color<br><br>Phased out in October 2006  | 5.7 in. STN display, color (256 colors), 320x240 pixels, touch, 24V DC only   | 2 MB | 2xRS232, 1xRS422, 1xRS485, USB, S5, S7-200, S7-300/400, and third-party controllers, printer port available. | 2711P-T6C20D                       | PanelView Plus 600 color touch   | 5.5 in. TFT color display, 320x240 pixels, 18-bit color depth, EtherNet/IP, RS-232 communication, touch, 24V DC, 64 MB flash, USB printing capabilities  |
| 6AV6545-0CC10-0AX0          | SIMATIC TP270 10 in. color<br><br>Phased out in October 2006 | 10.4 in. STN display, color (256 colors), 640x480 pixels, touch, 24V DC only  | 2 MB | 2xRS232, 1xRS422, 1xRS485, USB, S5, S7-200, S7-300/400, and third-party controllers, printer port available  | 2711P-T10C4D1                      | PanelView Plus 1000 color touch  | 10.4 in. TFT display, 640x480 pixels, 18-bit color, EtherNet/IP and RS-232, touch, 24V DC, 64 MB flash, USB printing capabilities                        |
| 6AV6542-0CA10-0AX0          | SIMATIC OP270 6 in. color<br><br>Phased out in October 2006  | 5.7 in. STN display, color (256 colors), 320x240 pixels, keypad, 24V DC only  | 2 MB | 2xRS232, 1xRS422, 1xRS485, USB, S5, S7-200, S7-300/400, and third-party controllers, printer port available  | 2711P-K6C20D                       | PanelView Plus 600 color         | 5.5 in. TFT color display, 320x240 pixels, 18-bit color depth, EtherNet/IP, RS-232 communication, keypad, 24V DC, 64 MB flash, USB printing capabilities |
| 6AV6542-0CC10-0AX0          | SIMATIC OP270 10 in. color<br><br>Phased out in October 2006 | 10.4 in. STN display, color (256 colors), 640x480 pixels, keypad, 24V DC only | 2 MB | 2xRS232, 1xRS422, 1xRS485, USB, S5, S7-200, S7-300/400, and third-party controllers, printer port available  | 2711P-K10C4D1                      | PanelView Plus 1000 color keypad | 10.4 in. TFT display, 640x480 pixels, 18-bit color, EtherNet/IP and RS-232, keypad, 24V DC, 64 MB flash, USB printing capabilities                       |

| SIMATIC Panels - 27x Series |                                  |  |      |   | Rockwell Automation solution       |  |  |
|-----------------------------|----------------------------------|--|------|---|------------------------------------|--|--|
| Siemens Catalog Number      | Short Reference                  | Description  | Mem. | Comm. Options   | Rockwell Automation Catalog Number | Name                                   | Description  |
| 6AV6643-0AA01-1AX0          | SIMATIC TP 277 6 in. color       | 5.7 in. STN display, color (256 colors), 320x240 pixels, touch, 24V DC only                            | 4 MB | 1xRS422, 1xRS485, USB, Ethernet: S5, S7-200, S7-300/400, and third-party controllers, printer port available  | 2711P-T6C20D                       | PanelView Plus 600 color touch         | 5.5 in. TFT color display, 320x240 pixels, 18-bit color depth, EtherNet/IP, RS-232 communication, touch, 24V DC, 64 MB flash, USB printing capabilities  |
| 6AV6643-0BA01-1AX0          | SIMATIC OP 277 6 in. color       | 5.7 in. STN display, color (256 colors), 320x240 pixels, keypad, 24V DC only                           | 4 MB | 1xRS422, 1xRS485, USB, Ethernet, S5, S7-200, S7-300/400, and third-party controllers, printer port available. | 2711P-K6C20D                       | PanelView Plus 600 color               | 5.5 in. TFT color display, 320x240 pixels, 18-bit color depth, EtherNet/IP, RS-232 communication, keypad, 24V DC, 64 MB flash, USB printing capabilities |
| 6AV6622-0BA01-0AA0          | WINCC FLEXIBLE STANDARD software | Configuration and programming software for Simatic OP/TP/MP270, MP370, OP77, OP/TP170 and micro panels | N/A  | N/A   | 9701-VWSTMENE                      | RSView Studio Machine Edition software | RSView Studio Machine Edition configuration software for developing and testing machine level HMI applications   |

## SIMATIC Multi Panels - 27x Series and Rockwell Automation Equivalents

| SIMATIC Multi Panels - 27x Series |  |  |      |  | Rockwell Automation Solution       |                                  |  |
|-----------------------------------|--|--|------|--|------------------------------------|----------------------------------|--|
| Siemens Catalog Number            | Short Reference  | Description  | Mem. | Comm. Options  | Rockwell Automation Catalog Number | Name                             | Description  |
| 6AV6542-0AG10-0AX0                | SIMATIC MP270B keypad 10 in.<br><br>Phased out in October 2006 | 10.4 in. TFT display, color (64 k colors), 640x480 pixels, keypad, 24V DC only | 5 MB | 2xRS422, 1xRS485, USB, Ethernet, S5, S7-200, S7-300/400, and third-party controllers, printer port available   | 2711P-K10C4D1                      | PanelView Plus 1000 color keypad | 10.4 in. TFT display, 640x480 pixels, 18-bit color, EtherNet/IP and RS-232, keypad, 24V DC, 64 MB flash, USB printing capabilities                       |
| 6AV6545-0AG10-0AX0                | SIMATIC MP270B touch, 10 in.<br>Phased out in October 2006     | 10.4 in. TFT display, color (64 k colors), 640x480 pixels, touch, 24V DC only  | 5 MB | 2xRS422, 1xRS485, USB, Ethernet, S5, S7-200, S7-300/400, and third-party controllers, printer port available   | 2711P-T10C4D1                      | PanelView Plus 1000 color touch  | 10.4 in. TFT display, 640x480 pixels, 18-bit color, EtherNet/IP and RS-232, touch, 24V DC, 64 MB flash, USB printing capabilities                        |
| 6AV6545-0AH10-0AX0                | SIMATIC MP270B touch, 6 in.<br><br>Phased out in October 2006  | 5.7 in. TFT display, color (64 k colors), 320x240 pixels, touch, 24V DC only   | 5 MB | 2xRS422, 1xRS485, USB, Ethernet, S5, S7-200, S7-300/400, and third-party controllers, printer port available   | 2711P-K6C20D                       | PanelView Plus 600 color         | 5.5 in. TFT color display, 320x240 pixels, 18-bit color depth, EtherNet/IP, RS-232 communication, keypad, 24V DC, 64 MB flash, USB printing capabilities |
| 6AV6643-0CB01-1AX0                | SIMATIC MP 277 touch, 8 in.                                    | 7.5 in. TFT display, color (64 k colors), 640x480 pixels, touch, 24V DC only   | 6 MB | 1xRS422, 1xRS485, 2xUSB, Ethernet, S5, S7-200, S7-300/400, and third-party controllers, printer port available | 2711P-T7C4D1                       | PanelView Plus 700 color touch   | 6.5 in. TFT display, 640x480 pixels, 18-bit color, EtherNet/IP and RS-232, touch, 24V DC, 64 MB flash, USB printing capabilities                         |

| <b>SIMATIC Multi Panels - 27x Series</b> |   |  |             |  | <b>Rockwell Automation Solution</b>       |  |  |
|--|---|--|-------------|--|---|--|--|
| <b>Siemens Catalog Number</b>            | <b>Short Reference</b>                        | <b>Description</b>   | <b>Mem.</b> | <b>Comm. Options</b>   | <b>Rockwell Automation Catalog Number</b> | <b>Name</b>                            | <b>Description</b>   |
| 6AV6643-OC001-1AX0                       | SIMATIC MP 277 touch, 10 in.                  | 10.4 in. TFT display, color (64 k colors), 640x480 pixels, touch, 24V DC only                              | 6 MB        | 1xRS422, 1xRS485, 2xUSB, Ethernet: S5, S7-200, S7-300/400, and third-party controllers, printer port available | 2711P-T10C4D1                             | PanelView Plus 1000 color touch        | 10.4 in. TFT display, 640x480 pixels, 18-bit color, EtherNet/IP and RS-232, touch, 24V DC, 64 MB flash, USB printing capabilities  |
| —  | SIMATIC MP 277 touch, 10 in., stainless steel | 10.4 in. TFT display, color (64 k colors), 640x480 pixels, touch, 24V DC only, stainless steel bezel, IP66 | 6 MB        | 1xRS422, 1xRS485, 2xUSB, Ethernet, S5, S7-200, S7-300/400, and third-party controllers, printer port available | 2711P-T10C4D1                             | PanelView Plus 1000 color touch        | 10.4 in. TFT display, 640x480 pixels, 18-bit color, EtherNet/IP and RS-232, touch, 24V DC, 64 MB flash, USB printing capabilities  |
| 6AV6643-ODB01-1AX0                       | SIMATIC MP 277 keypad, 8 in.                  | 7.5 in. TFT display, color (64 k colors), 640x480 pixels, keypad, 24V DC only                              | 6 MB        | 1xRS422, 1xRS485, 2xUSB, Ethernet, S5, S7-200, S7-300/400, and third-party controllers, printer port available | 2711P-K7C4D1                              | PanelView Plus 700 color keypad        | 6.5 in. TFT display, 640x480 pixels, 18-bit color, EtherNet/IP and RS-232, keypad, 24V DC, 64 MB flash, USB printing capabilities  |
| 6AV6643-ODD01-1AX0                       | SIMATIC MP 277 keypad, 10 in.                 | 10.5 in. TFT display, color (64 k colors), 640x480 pixels, keypad, 24V DC only                             | 6 MB        | 1xRS422, 1xRS485, 2xUSB, Ethernet, S5, S7-200, S7-300/400, and third-party controllers, printer port available | 2711P-K10C4D1                             | PanelView Plus 1000 color keypad       | 10.4 in. TFT display, 640x480 pixels, 18-bit color, EtherNet/IP and RS-232, keypad, 24V DC, 64 MB flash, USB printing capabilities |
| 6AV6622-0BA01-0AA0                       | WINCC FLEXIBLE STANDARD software              | Configuration and programming software for Simatic OP/TP/MP270, MP370, OP77, OP/TP170 and micro panels     | N/A         | N/A  | 9701-VVSTMENE                             | RSView Studio Machine Edition software | RSView Studio Machine Edition configuration software for developing and testing machine level HMI applications                     |

## SIMATIC Multi Panels - 37x Series and Rockwell Automation Equivalents

| SIMATIC Multi Panels - 37x Series |  |  |         |  | Rockwell Automation Solution       |                                  |  |
|-----------------------------------|--|--|---------|--|------------------------------------|----------------------------------|--|
| Siemens Catalog Number            | Short Reference                              | Description  | Mem.    | Comm. Options  | Rockwell Automation Catalog Number | Name                             | Description  |
| 6AV6542-0DA10-0AX0                | SIMATIC MP370 keypad, 12 in.                 | 12.1 in. TFT display, color (256 colors), 800x600 pixels, keypad, 24V DC only                              | 12.5 MB | 1xTTY, 2xRS232, 1xRS422, 1xRS485, 1xUSB, Ethernet, S5, S7-200, S7-300/400, and third-party controllers, printer port | 2711P-K12C4D1                      | PanelView Plus 1250 color keypad | 12.1 in. TFT display, 800x600 pixels, 18-bit color, EtherNet/IP and RS-232, keypad, 24V DC, 64 MB flash, USB printing capabilities |
| 6AV6545-0DA10-0AX0                | SIMATIC MP370 touch, 12 in.                  | 12.1 in. TFT display, color (256 colors), 800x600 pixels, touch, 24V DC only                               | 12.5 MB | 1xTTY, 2xRS232, 1xRS422, 1xRS485, 1xUSB, Ethernet, S5, S7-200, S7-300/400, and third-party controllers, printer port | 2711P-T12C4D1                      | PanelView Plus 1250 color touch  | 12.1 in. TFT display, 800x600 pixels, 18-bit color, EtherNet/IP and RS-232, touch, 24V DC, 64 MB flash, USB printing capabilities  |
| 6AV6545-0DB10-0AX0                | SIMATIC MP370 touch, 15 in.                  | 15.1 in. TFT display, color (256 colors), 1024x768 pixels, touch, 24V DC only                              | 12.5 MB | 1xTTY, 2xRS232, 1xRS422, 1xRS485, 1xUSB, Ethernet, S5, S7-200, S7-300/400, and third-party controllers, printer port | 2711P-T15C4D1                      | PanelView Plus 1500 color touch  | 15 in. TFT display, 1024x768 pixels, 18-bit color, EtherNet/IP and RS-232, touch, 24V DC, 64 MB flash, USB printing capabilities   |
| 6AV6545-8DB10-0AA0                | SIMATIC MP370 touch, 15 in., stainless steel | 15.1 in. TFT display, color (256 colors), 1024x768 pixels, touch, 24V DC only, stainless steel bezel, IP66 | 12.5 MB | 1xTTY, 2xRS232, 1xRS422, 1xRS485, 1xUSB, Ethernet, S5, S7-200, S7-300/400, and third-party controllers, printer port | 2711P-T15C4D1                      | PanelView Plus 1500 color touch  | 15 in. TFT display, 1024x768 pixels, 18-bit color, EtherNet/IP and RS-232, touch, 24V DC, 64 MB flash, USB printing capabilities   |

| SIMATIC Multi Panels - 37x Series |                                  |  |         |  | Rockwell Automation Solution       |  |  |
|-----------------------------------|----------------------------------|--|---------|--|------------------------------------|--|--|
| Siemens Catalog Number            | Short Reference                  | Description  | Mem.    | Comm. Options  | Rockwell Automation Catalog Number | Name                                   | Description  |
| 6AV6 644-0AA01-2AX0               | SIMATIC MP377 touch 12.1 in.     | 12.1 in. TFT display, 65,536 colors, 800x600 pixels, touch, 24V DC only                                | 12.5 MB | 1xTTY, 2xRS232, 1xRS422, 1xRS485, 2xUSB, 2xEthernet, S5, S7-200, S7-300/400, and third-party controllers, printer port | 2711P-T12C4D1                      | PanelView Plus 1250 color touch        | 12.1 in. TFT display, 800x600 pixels, 18-bit color, EtherNet/IP and RS-232, touch, 24V DC, 64 MB flash, USB printing capabilities  |
| 6AV6 644-0BA01-2AX0               | SIMATIC MP377 keypad, 12.1 in.   | 12.1 in. TFT display, 65,536 colors, 800x600 pixels, keypad, 24V DC only                               | 12.5 MB | 1xTTY, 2xRS232, 1xRS422, 1xRS485, 2xUSB, 2xEthernet, S5, S7-200, S7-300/400, and third-party controllers, printer port | 2711P-K12C4D1                      | PanelView Plus 1250 color keypad       | 12.1 in. TFT display, 800x600 pixels, 18-bit color, EtherNet/IP and RS-232, keypad, 24V DC, 64 MB flash, USB printing capabilities |
| 6AV6 644-0AB01-2AX0               | SIMATIC MP377 touch, 15 in.      | 15 in. TFT display, 65,536 colors, 1024x768 pixels, touch, 24V DC only                                 | 12.5 MB | 1xTTY, 2xRS232, 1xRS422, 1xRS485, 2xUSB, 2xEthernet, S5, S7-200, S7-300/400, and third-party controllers, printer port | 2711P-T15C4D1                      | PanelView Plus 1500 color touch        | 15 in. TFT display, 1024x768 pixels, 18-bit color, EtherNet/IP and RS-232, touch, 24V DC, 64 MB flash, USB printing capabilities   |
| 6AV6 644-0BA01-2AX0               | SIMATIC MP377 touch, 19 in.      | 19 in. TFT display, 65,536 colors, 1280x1024 pixels, touch, 24V DC only                                | 12.5 MB | 1xTTY, 2xRS232, 1xRS422, 1xRS485, 2xUSB, 2xEthernet, S5, S7-200, S7-300/400, and third-party controllers, printer port | 2711P-T15C4D1                      | PanelView Plus 1500 color touch        | 15 in. TFT display, 1024x768 pixels, 18-bit color, EtherNet/IP and RS-232, touch, 24V DC, 64 MB flash, USB printing capabilities   |
| 6AV6622-0BA01-0AA0                | WINCC FLEXIBLE STANDARD software | Configuration and programming software for Simatic OP/TP/MP270, MP370, OP77, OP/TP170 and micro panels | N/A     | N/A  | 9701-VWSTMENE                      | RSView Studio Machine Edition software | RSView Studio Machine Edition configuration software for developing and testing machine level HMI applications                     |



**Notes:**





# Rockwell Automation Support

Rockwell Automation provides technical information on the Web to assist you in using its products. At <http://support.rockwellautomation.com>, you can find technical manuals, a knowledge base of FAQs, technical and application notes, sample code and links to software service packs, and a MySupport feature that you can customize to make the best use of these tools.

For an additional level of technical phone support for installation, configuration, and troubleshooting, we offer TechConnect support programs. For more information, contact your local distributor or Rockwell Automation representative, or visit <http://support.rockwellautomation.com>.

## Installation Assistance

If you experience a problem within the first 24 hours of installation, please review the information that's contained in this manual. You can also contact a special Customer Support number for initial help in getting your product up and running.

|                       |  |
|-----------------------|--|
| United States         | 1.440.646.3434<br>Monday – Friday, 8am – 5pm EST   |
| Outside United States | Please contact your local Rockwell Automation representative for any technical support issues. |

## New Product Satisfaction Return

Rockwell Automation tests all of its products to ensure that they are fully operational when shipped from the manufacturing facility. However, if your product is not functioning and needs to be returned, follow these procedures.

|                       |  |
|-----------------------|--|
| United States         | Contact your distributor. You must provide a Customer Support case number (call the phone number above to obtain one) to your distributor in order to complete the return process. |
| Outside United States | Please contact your local Rockwell Automation representative for the return procedure.   |

[www.rockwellautomation.com](http://www.rockwellautomation.com)

### Power, Control and Information Solutions Headquarters

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444

Europe/Middle East/Africa: Rockwell Automation, Vorstlaan/Boulevard du Souverain 36, 1170 Brussels, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640

Asia Pacific: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication LOGIX-AP008B-EN-P - June 2008

Supersedes publication LOGIX-AP008B-EN-P

Copyright © 2008 Rockwell Automation, Inc. All rights reserved. Printed in the U.S.A.