

Converting PLC-5 or SLC 500 Logic to Logix-Based Logic

1756 ControlLogix, 1756 GuardLogix, 1769 CompactLogix, 1769 Compact GuardLogix, 1789 SoftLogix, 5069 CompactLogix, 5069 Compact GuardLogix, Studio 5000 Logix Emulate

Publication 1756-RM085E-EN-P

Rockwell Automation

Reference Manual

Original Instructions

Important User Information

Read this document and the documents listed in the additional resources section about installation, configuration, and operation of this equipment before you install, configure, operate, or maintain this product. Users are required to familiarize themselves with installation and wiring instructions in addition to requirements of all applicable codes, laws, and standards.

Activities including installation, adjustments, putting into service, use, assembly, disassembly, and maintenance are required to be carried out by suitably trained personnel in accordance with applicable code of practice.

If this equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

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.

WARNING: 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.

ATTENTION: 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.

IMPORTANT Identifies information that is critical for successful application and understanding of the product.

Labels may also be on or inside the equipment to provide specific precautions.

SHOCK HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.

BURN HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.

ARC FLASH HAZARD: Labels may be on or inside the equipment, for example, a motor control center, to alert people to potential Arc Flash. Arc Flash will cause severe injury or death. Wear proper Personal Protective Equipment (PPE). Follow ALL Regulatory requirements for safe work practices and for Personal Protective Equipment (PPE).

This manual includes new and updated information. Use these reference tables to locate changed information.

Grammatical and editorial style changes are not included in this summary.

Global changes

This table identifies changes that apply to all information about a subject in the manual and the reason for the change. For example, the addition of new supported hardware, a software design change, or additional reference material would result in changes to all of the topics that deal with that subject.

Subject	Reason
Updated tables that provide Tip, Note, and Important information.	Formatting change.
The Legal Noticeshave been updated.	Removed redundant information.

New or enhanced features

None in this version.

Chapter 2

Converting Program Structure

Chapter 3

Converting Data

Studio 5000 environment

The Studio 5000 Automation Engineering & Design Environment® combines engineering and design elements into a common environment. The first element is the Studio 5000 Logix Designer® application. The Logix Designer application is the rebranding of RSLogix 5000® software and will continue to be the product to program Logix 5000™ controllers for discrete, process, batch, motion, safety, and drive-based solutions.

The Studio 5000® environment is the foundation for the future of Rockwell Automation® engineering design tools and capabilities. The Studio 5000 environment is the one place for design engineers to develop all elements of their control system.

Additional resources

These documents contain additional information concerning related Rockwell Automation products.

Resource	Description
Industrial Automation Wiring and Grounding Guidelines, publication 1770-4.1	Provides general guidelines for installing a Rockwell Automation industrial system.
Product Certifications webpage, available at http://ab.rockwellautomation.com	Provides declarations of conformity, certificates, and other certification details.

View or download publications at http://www.rockwellautomation.com/literature. To order paper copies of technical documentation, contact the local Rockwell Automation distributor or sales representative.

Legal Notices

Rockwell Automation publishes legal notices, such as privacy policies, license agreements, trademark disclosures, and other terms and conditions on the Legal Notices page of the Rockwell Automation website.

End User License Agreement (EULA)

You can view the Rockwell Automation End-User License Agreement ("EULA") by opening the License.rtf file located in your product's install folder on your hard drive.

Open Source Licenses

The software included in this product contains copyrighted software that is licensed under one or more open source licenses. Copies of those licenses are included with the software. Corresponding Source code for open source packages included in this product are located at their respective web site(s).

Alternately, obtain complete Corresponding Source code by contacting Rockwell Automation via the Contact form on the Rockwell Automation website: http://www.rockwellautomation.com/global/about-us/contact/contact.page
Please include "Open Source" as part of the request text.

A full list of all open source software used in this product and their corresponding licenses can be found in the OPENSOURCE folder. The default installed location of these licenses is `C:\Program Files (x86)\Common Files\Rockwell\Help\FactoryTalk Services Platform\Release Notes\OPENSOURCE\index.htm`.

Converting a PLC-5 or SLC 500 Program into a Logix Project

Introduction

The RSLogix Project Migrator converts a PLC-5 or SLC 500 import/export file (PC5 or SLC extension) into a complete import/export file (L5K extension).

This manual describes the RSLogix Project Migrator. This chapter describes the pre-migration file preparation and post-migration examples and tasks.

IMPORTANT The Project Migrator converts only ladder instructions. SFC and structured text files are not converted.

The Project Migrator can be downloaded from the Rockwell Automation website.

What to expect from the Project Migrator

The goal of the Project Migrator is to reduce the amount of work involved in migrating a PLC-5 or SLC 500 program to a Logix project. The Project Migrator automatically converts the program logic, but it is not the complete solution. Depending on the application, you may need to do additional work to make the converted logic work properly.

The Project Migrator produces a syntactically correct import/export file, but the exact intent of the original application could be lost. This loss could be due to differences in rules. (For example, rules of precedence, rules of indexed addressing, or rules of I/O addressing). When there is an error in the migration, the Project Migrator records the error in the rung of the Logix routine in which it occurred. You can use that error message to analyze and fix the error.

IMPORTANT After running the conversion process, the resulting import/export file still requires further manipulation. You must map the I/O and use BTD, MOV, or CPS instructions to place this mapped data into the structures created by the conversion process.

Comparing PLC-5/SLC 500 architecture to Logix architecture

The Logix architecture differs in several ways from that of the PLC-5 and SLC 500 processors. The Project Migrator converts this legacy architecture as it best fits into the Logix architecture. Because of the architectural differences, you may have to rework the converted Logix project to make sure it operates properly.

The most significant differences in architecture are listed in the following table:

Architectural issue	Comparison
CPU	The PLC-5 and SLC 500 processor is based on 16-bit operations. Logix controllers use 32-bit operations. The Project Migrator converts legacy logic into its 32-bit equivalent.
operating system	The PLC-5 and SLC 500 processors support individual program files that can be configured as selectable timed interrupts (STIs) or input interrupts (DIIs/PIIs). In addition, the PLC-5 processor supports multiple main control programs (MCPs). A Logix controller combines these into it's task, program, and routine organization. The Project Migrator converts the legacy program types into appropriate Logix tasks.
input and outputs	The PLC-5 and SLC 500 processor map I/O memory into I and O data table files. The I/O data is updated synchronously to the program scan so you know you have current values each time the processor begins a scan. A Logix controller references I/O which is updated asynchronously to the logic scan. For a Logix controller, use the synchronous copy (CPS) instruction to create an I/O data buffer to use for static values during logic execution and update the buffer as needed. After the conversion is complete, you must add instructions to copy the I/O data into the I and O arrays. Do this at the beginning or ending of a program to buffer the data so that it is presented synchronously to the program scan.
data	The PLC-5 and SLC 500 processors store all data in global data tables. You access this data by specifying the address of the data you want. A Logix controller supports data that is local to a program and data that is global to all the tasks within the controller. A Logix controller can also share data with other controllers, and instead of addresses, you use tags to access the data you want. Each PLC-5 and SLC 500 data table file can store several words of related data. A Logix controller uses arrays to store related data. The Project Migrator converts the PLC-5 and SLC 500 data table files into Logix arrays.
s	The PLC-5 and SLC 500 s are based on their 16-bit architecture and can have different time bases. A Logix controller is based on its 32-bit architecture and only supports a 1 msec time base. The Project Migrator converts the legacy s as they best fit into the Logix architecture. Converted s might require rework to make sure they operate properly.
communications	The PLC-5 processor supports block-transfer read and write (BTR and BTW) instructions, ControlNet I/O (CIO), and message (MSG) instructions. The SLC 500 processor supports block-transfer read and write (BTR and BTW) instructions and MSG (MSG) instructions. The Logix 5000 controllers support MSG instructions of various types. You will need to verify and complete all MSG instructions after migration.

The conversion/migration process

The entire conversion process involves the following steps:

Conversion step	Page
Preparing RSLogix 5 or RSLogix 500 files for Migration on page 13	9 on page 13
Exporting a PLC-5 or SLC 500 Program on page 13	9 on page 13
Using the Project Migrator Wizard to Convert a PLC-5 or SLC 500 Program on page 16	13 on page 16
Working with PCE Instructions on page 19	16 on page 19
Working with UNK Instructions on page 20	17 on page 20

Conversion step	Page
Configuring the Controller and Chassis on page 20	18 on page 20
Mapping the I/O on page 21	19 on page 21
Completing the MSG Configuration on page 22	20 on page 22

The rest of the chapter describes these steps in detail.

Preparing RSLogix 5 or RSLogix 500 files for migration

Before using the Project Migrator, it's best to prepare the RSLogix5 and RSLogix 500 files.

- To save memory, remove unused references from the PLC-5 and SLC 500 application files. The following options are available for you in RSLogix 5 or RSLogix 500 software:
  - Delete unused memory. (Tools> Delete Unused Memory)
  - Delete unused addresses. (Tools> Database>Delete Unused Addresses)
- To help avoid syntax errors that the Project Migrator will not convert if encountered in the PC5 file, remove SFC and STX routines.

Exporting a PLC-5 or SLC 500 Program

Before you can convert PLC-5 or SLC 500 logic to its Logix equivalent, you must first export the logic to an ASCII text file with a PC5 extension for a PLC-5 file or a SLC extension for an SLC 500 file.

If you elect to convert comments and symbols, you also need the TXT file, which is the standard 6200 programming software format for a documentation file.

Use the RSLogix 5 or RSLogix 500 export procedure to produce two types of files:

- Database files (TXT). These files contain the application's symbols, address comments, instruction comments, and rung comments. Three TXT files are produced.
  - <program name>.txt - Address Comment and Symbols
  - <program name>1.txt - Instruction Comments (These are ignored by the Project Migrator.)
  - <program name>2.txt - Rung Comments (These are ignored by the Project Migrator. Rung comments within PC5/SLC file are used.)
- Program file format (PC5 or SLC). This file contains an application's data, RLL statements, and rung comments.

Export options

There are a few ways that you can export files for migration using RSLogix 5 or RSLogix 500 software as described on the following pages.

Export Option 1: Create two separate exports

For this option, you export the PC5/SLC files and then export the TXT file.

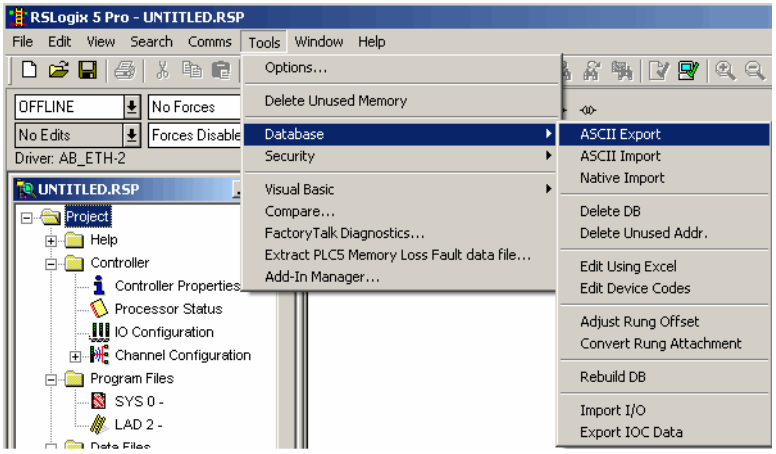
First, create the PC5 or SLC files for export, using the following steps.

1. In RSLogix 5 or RSLogix 500, select File>Save As. The Save Program As dialog box appears.

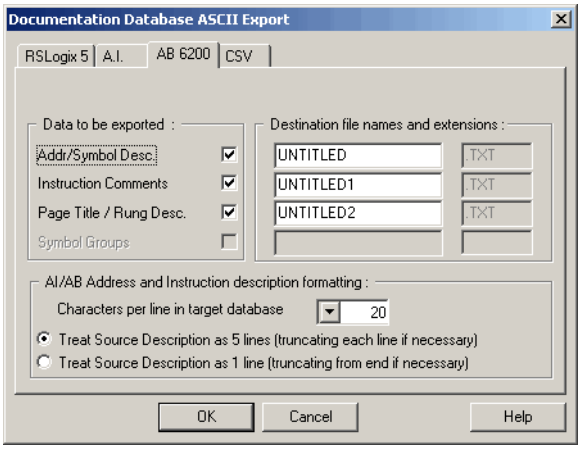
2. In the Save in field, select the program you want to export.
   By default, the software points to the Project folder for the destination. You can enter a different destination directory.
3. In the Save As type field, select the Library Files format (PC5 or SLC).
4. Check the 'Save data base as external file' checkbox so that the comments and symbols are included in the export.
5. Click **Save**. The Export PC5 Format or the Export SLC5000 dialog box appears.
6. On the export format dialog box, use the following steps.
   a. Select **Complete Program Save**.
   b. Select all the export options.
   c. Click **OK**.

Then create a TXT file for export, using the following steps.

1. In RSLogix5 or RSLogix 500, from the Tools menu, choose **Database > ASCII Export**.



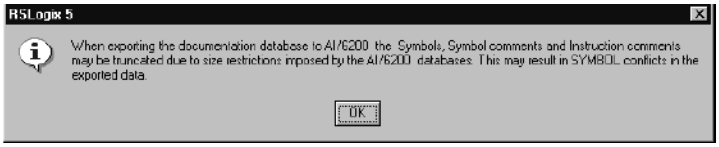
The Documentation Database ASCII Export dialog box appears.



2. Select the **AB 6200** tab, make your selections, and then click **OK**.

The Select Export Destination Directory dialog box appears.

3. Under Directories, select the directory where the PC5 or SLC file resides.
4. Accept the warning about comments and symbols. Click **OK**.



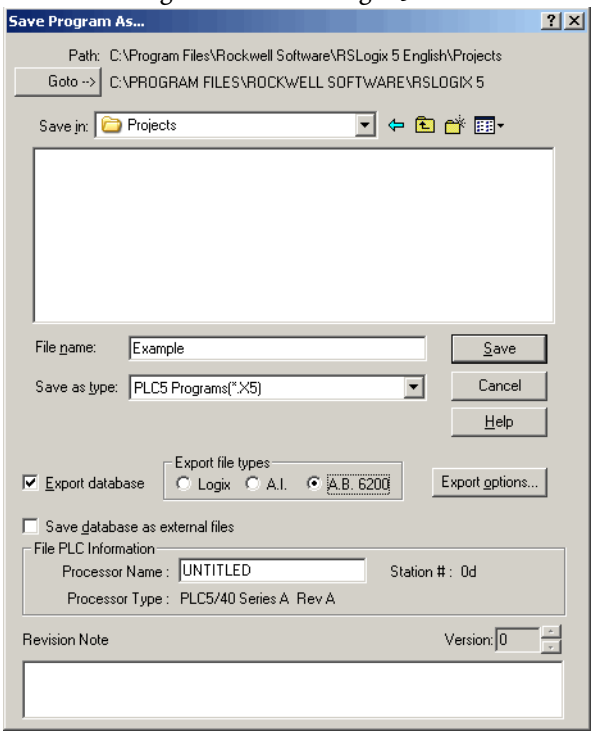
RSLogix 5 programming software stores PLC-5 programs using RSP file extensions. RSLogix 500 programming software stores SLC 500 programs using RSS file extensions.

Export Option 2: Create TXT files and PC5/SLC files for export at the same time

To create both files for export at the same time, use the following steps.

1. In RSLogix 5 or RSLogix 500, select **File>Save As**.

   The Save Program As dialog box appears. The example below shows the Save As dialog box from RSLogix 5.



2. In the Save as type field, select X5 or ACH to activate the "Export database" checkbox.
3. Check the "Export database" checkbox.

4. Under Export file type, select one of the following, depending on the programming software you are using.
   - For RSLogix 5, select A.B. 6200
   - For RSLogix 500, select A.P. S.
5. In the Save as type field, change the file type from X5 or ACH to PC5 or SLC. Even though the A.B. 6200 format is dimmed, the database will be exported in that format.
6. Click **Save**. The Export PC5 Format or the Export SLC 500 Format dialog box appears.
7. On the export format dialog box, use the following steps.

   d. Select **Complete Program Save**.

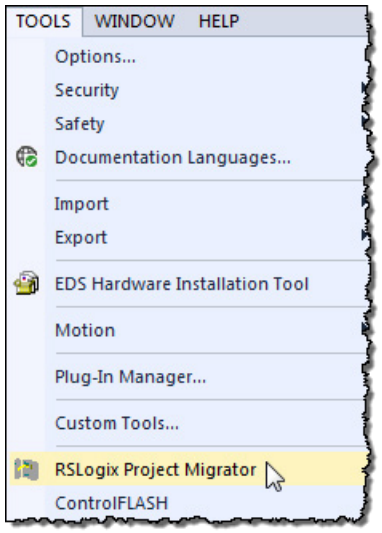
   e. Select *all* the export options.

   f. Click **OK**.

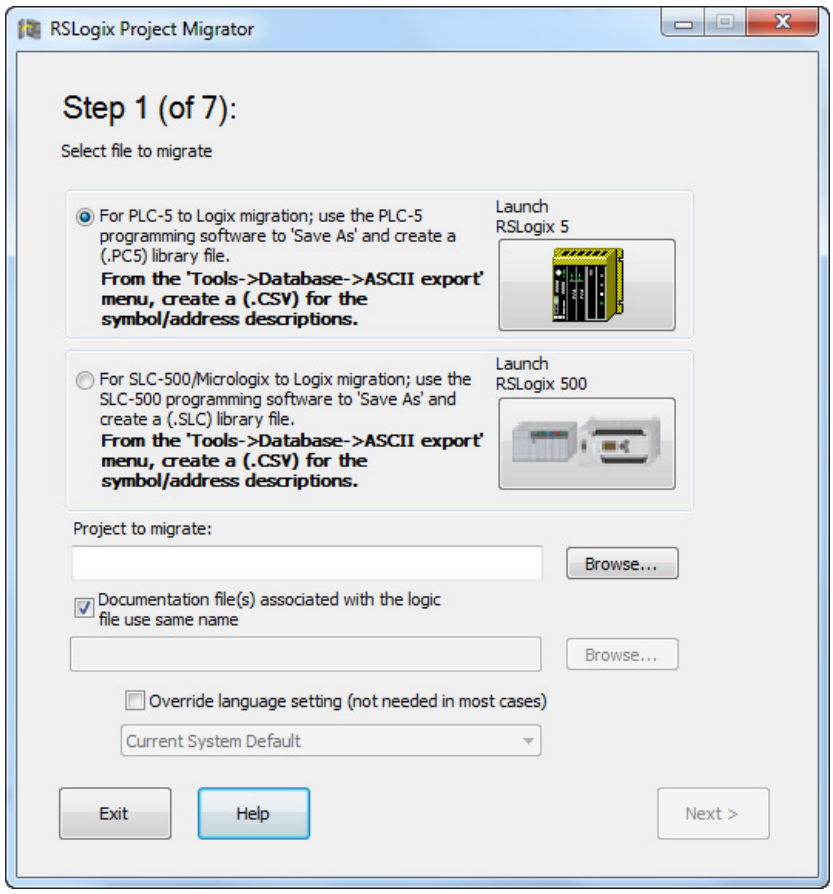
Use the Project Migrator Wizard to Convert a PLC-5 or SLC 500 Program

Once you have the ASCII text file of the PLC-5 or SLC 500 program file, you can convert the logic to its Logix equivalent. In the Logix Designer application, use the following steps.

1. From the **Tools** menu, choose **RSLogix Project Migrator**.



2. The wizard appears. Use the wizard to walk through steps 1 through 7.



Wizard step 1 notes

- To start, select one of the option buttons.
  - For **PLC-5 to Logix...** button - Select this option to browse for PC5 files.
  - For **SLC-500 to Logix...** button - Select this option to browse for SLC files.
- To use a different TXT file name than the program file name, clear the **Documentation file(s) associated with the logic file use the same name** checkbox. Browse to the first database file name (TXT).
- RSLogix 5 or RSLogix 500 can also be launched from the wizard. To do so, click the **Launch RSLogix 5** or the **Launch RSLogix 500** icon at the top of the dialog box.

  For this to work, the software must be installed on the same computer as the Project Migrator.
- Browse to the file to be migrated.
- Click **Next**.

Wizard step 2 notes

Select or clear the check box for the following option:

- **Create Alias Tag for existing PLC-5/SLC Symbols** -- This option creates alias tags for all symbols found in the database files. Otherwise the symbols are added as address comments to converted legacy file types.

Wizard step 3 notes

- Click **Migrate**. Since every PLC-5 and SLC500 application is unique, there may be syntax errors.
- If syntax errors occur during migration, they appear in the **Status Log** pane and include the line at which the error occurred. Choose one these actions to deal with errors:
  - Edit the error immediately and click **Save & Retry** to restart the migration.
  - Examine the original application to decide if the area where the syntax error is occurring is something that can be deleted permanently or if it is something that can be removed and then later be recreated in the Logix Designer application.
  - Edit the PC5, SLC, and TXT files using a text editor such as Notepad.
  - Review the table that follows for the most common syntax errors and their descriptions.

Wizard step 4 notes

- Specify a file name and folder destination for the migrated file. The default file name is the same as the legacy file with an .L5k extension, and the default folder location is the same as the location of the legacy file.
- Select the Logix controller and version of the Logix Designer application. The version selected should correspond to a version of the Logix Designer application that is supported on the PC.
- Click **Next**.

Common Syntax Errors

The Project Migrator might run into syntax errors within the program and database files. If so, you must correct the errors to continue the conversion. The following table describes common migration errors.

Syntax Error	Description	How to Fix the Error
Invalid symbol name	The Project Migrator expects the symbol names to be alphanumeric. RSLogix 5 and RSLogix 500 software enforce these rules, but using the 6200 software or manually editing the database files may cause these rules to be broken.	Search for symbol names that are not alphanumeric.
" (quote) within a " (quote)	Quotes are used to denote the start and end of string values or rung/instruction/address comments. If a rung/instruction/address comment contains a quote, the Project Migrator does not recognize that it is not the end of the string.	Either remove the quote or make it a double quote (""). The Project Migrator will translate the double quote as a single quote in the Logix Designer application.
% within a %	% characters are used to denote the start and end of comments within the program file and occasionally in database files. This type of comment is ignored by the Project Migrator. If a comment contains another %, the Project Migrator does not recognize that it is not the end of the comment.	Remove the extra % or make it a double %. The Project Migrator will treat the double %% as consecutive comments.

Syntax Error	Description	How to Fix the Error
Errant characters	The program or database file contains a random character or two that does not fit the syntax of the program or database files. This is more common with manually-edited files than a direct export from RSLogix 5 or RSLogix 500.	Remove the errant characters.
Invalid rung syntax	The rung has invalid syntax, such as unmatched parentheses.	Check and fix the rung syntax.

Wizard step 5 notes

Step 5 appears only for SLC and MicroLogix migrations. Select an I/O migration option:

- **Keep existing I/O in a separate rack** - Keeps the legacy I/O modules as they exist in the .SLC file, attaching them remotely via EtherNet in a legacy chassis.
- **Replace all I/O with equivalent newer models** - Displays a list of all I/O modules detected in the legacy file. Select the desired converted I/O module, onto which the Project Migrator maps the legacy I/O. Manually verify compatibility with the chosen output module. You can opt to create a placeholder tag rather than selecting an I/O module. When you select this option you must manually add the desired I/O and handle any copying operations from the migrated output file.

Wizard step 6 notes

When the migration is complete, click **Exit** to close the Project Migrator, or click **Launch RSLogix 5000** to open the Logix Designer application and import the migrated project.

Working with PCE Instructions

The Project Migrator inserts a Possible Conversion Error (PCE) instruction within the appropriate ladder rung to help you identify possible errors with the conversion. To complete the conversion process, you will want to locate, analyze, and fix any discrepancies using the PCE instructions.

For a list of PCE instruction errors, see Appendix A Programming Conversion Errors (PCE) Messages on page 65

Recognizing the instructions

Text is appended to the rung comments that have the PCE instruction. The message text begins with asterisks (*) and the words "Generated by Translation Tool", and ends with asterisks.

An example of a PCE instruction follows:

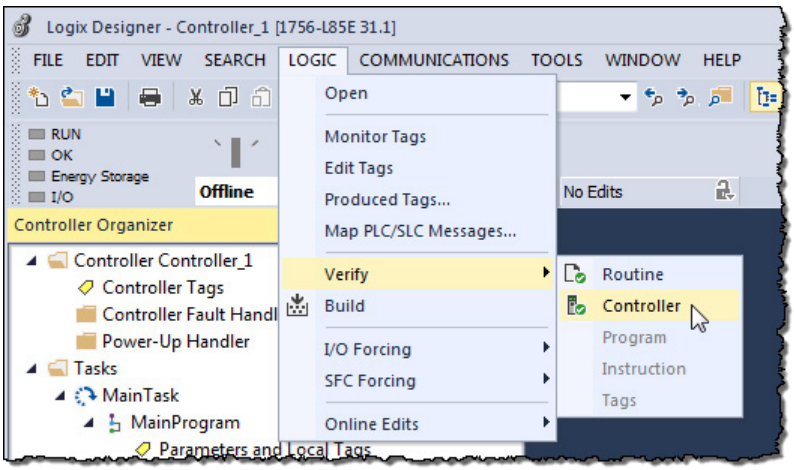
```
 *** Generated by Translation Tool: Source and destination
types may differ *** ";

N: PCE(120, PCE011) COP(I1_008, N23[0], 4);
```

Locating PCE instructions

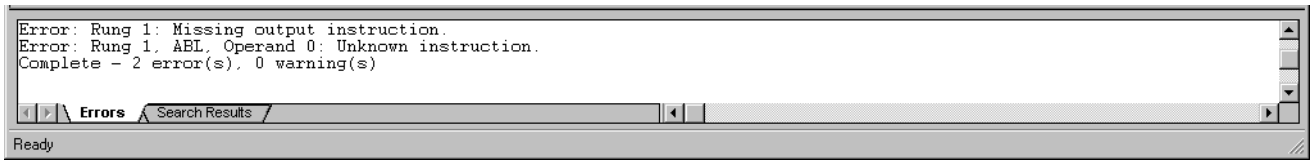
You can also locate all of the PCE instructions by verifying the logic. The **Verify > Controller** task compiles the Logix program and checks for errors. This is an easy way to see where all the PCE instructions are because the error checking will point them out. To locate the PCE instructions, use the following steps.

1. From the **Logic** menu, choose **Verify > Controller**.



The bottom of the screen displays the results.

2. Double-click the error shown in the error window to go directly to the rung where the error resides.



Resolving PCE Instructions

Once you import the converted Logix project, find each PCE instruction. A PCE instruction highlights a possible conversion error. Delete each PCE instruction and replace it with the appropriate, corrected logic.

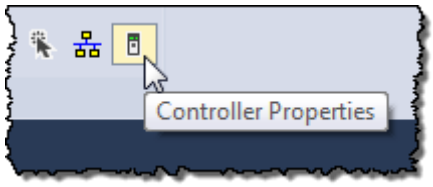
Working with UNK Instructions

The Project Migrator converts some PLC-5 and SLC 500 instructions that have no equivalent in the Logix architecture. Once you import these instructions into the Logix project, they appear as UNK instructions. You must delete each UNK instruction and replace it with the appropriate corrected logic.

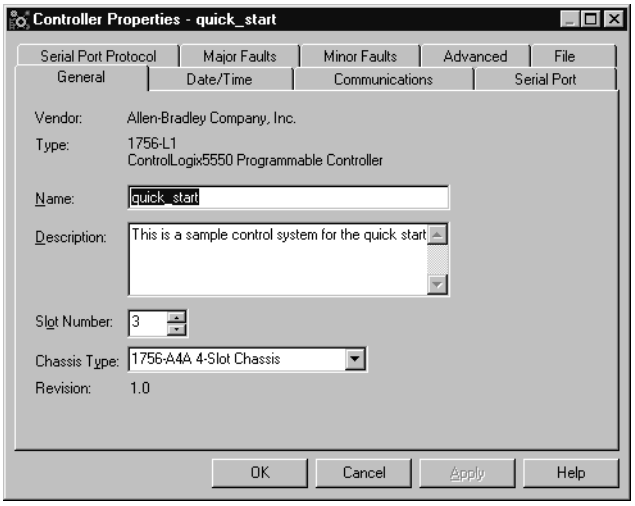
Configuring the Controller and Chassis

Once you have resolved any errors, continue the conversion process by using the **Controller Properties** dialog box in the Logix Designer application to assign the chassis size and slot number of the controller. Use the steps that follow.

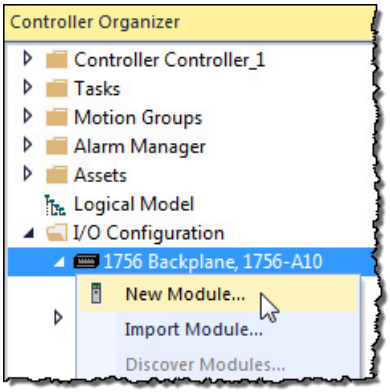
1. Click the **Controller Properties** icon to open the **Controller Properties** dialog box.



2. Select **Properties**. The Controller Properties dialog box appears.



3. Configure the controller by specifying the slot number of the controller and the chassis size.
4. Click **OK**.
5. Continue to use the **Controller Organizer** to specify the I/O modules and other devices for the controller. The example that follows shows how to specify the I/O module.



g. Select the backplane.

h. Right-click and select **New Module**.

Mapping the I/O

The file structure in a Logix controller is tag-based. To facilitate the conversion, the Project Migrator creates tags and arrays of tags to align and map the PLC-5 files. For example:

PLC-5 address	Maps to:
N7:500	N7[500]
N17:25	N17[25]
R6:100	R6[100]
I:002	I[2]
O:001	O[1]

The tags created for physical I/O (For example, I.2) are empty at the end of the conversion process.

- To continue with the conversion process, use the Logix Designer application to add all the I/O modules to the tree structure for a Logix controller.
- Then, program instructions to map the Logix I/O tags to the converted tags.
  - For example, if you add a 16-point input module in slot 2 of the local chassis, the programming software creates these I/O tag structures:

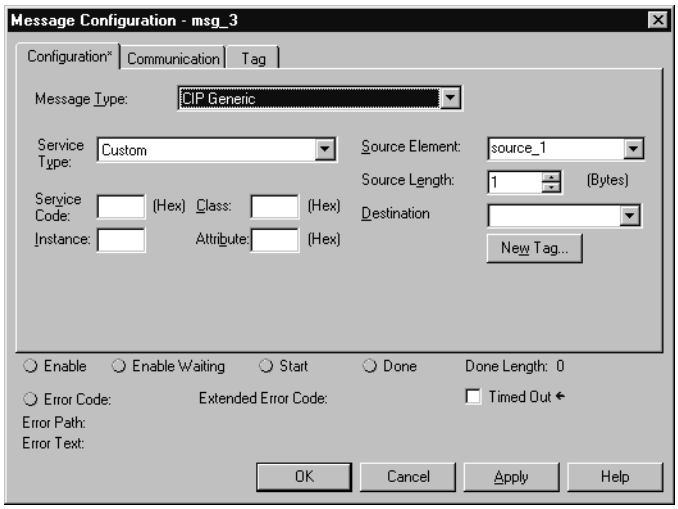
    Local:1.C (configuration information)
    Local:1.Data (fault and input data)
  - Use a BTD, MOV, or CPS instructions to map the Local:1.Data word into the I2 tag created by the conversion process.
  - An MOV instruction moves one element at a time. A BTD instruction moves a group of bits, which lets you account for the offset in the starting bit that occurs when you map an INT data type to a DINT data type. If consecutive I/O groups map to consecutive elements in an array, a CPS instruction is more efficient.

    For example, if I:000 through I:007 map to Local:1:I.Data[0] through Local:1:I.Data[7], use:
    CPS
    SourceLocal:1:I.Data[0]
    Destination:I[0]
    Length:8
  - If you use an MOV instruction, do not mix data types. If you mix data types, the conversion from one data type to another manipulates the sign bit, which means you cannot be sure that the high-order bit is set properly.

See Chapter 2 Converting Program Structure on page 25 for more information about how the Project Migrator converts the PLC-5 or SLC 500 data table.

Completing the MSG Configuration

The Project Migrator only partially converts MSG instructions. Use the Logix Designer application to configure each MSG instruction by completing the information on the **Communication** tab.



IMPORTANT For more information about configuring MSG instructions, see the Logix 5000 Instruction Set Reference Manual, publication 1756-RM003.
This manual is available in PDF format in the Rockwell Automation Lit Library.

Other Considerations

The following are additional issues to keep in mind:

- The time base for instructions is fixed at 1 msec for a Logix controller. The conversion process scales PLC-5 and SLC 500 presets and accumulators accordingly. For example, a PLC-5 with a time base of 0.01 sec and a preset of 20 is converted to a time base of 1 msec and a preset of 200.
- Instruction comments are not converted.
- A Logix controller is a 32-bit based controller. This means that most of the Logix instructions use 32-bit words, as opposed to the 16-bit words in PLC-5 processors. This might mean that instructions that use masks might work differently after the conversion.
- The conversion process creates alias tags for address comments. These aliases are then used in place of the converted tags.

  Alias tags utilize additional memory in a Logix controller, so you may want to delete those alias tags that you do not plan to use. Use the Logix Designer application to delete aliases after you import the project.

Converting Program Structure

Introduction

A Logix 5000 controller uses a different execution model than either the PLC-5 processor or the SLC 500 processor. The Logix 5000 controller operating system is a preemptive multitasking system that is IEC 61131-3 compliant and uses:

- Tasks
- Programs
- Routines

This chapter provides a short description of the Logix 5000 controller to help explain the migration results.

Dividing Logic into Tasks, Programs, and Routines

The tasks, programs, and routines work together as follows:

- **Tasks:** Tasks are used to configure controller execution. A task provides scheduling and priority information for a set of one or more programs. You can configure tasks as either continuous, periodic, or event tasks.
- **Programs:** Programs are used to group data and logic. A task contains programs, each with its own routines and program-scoped tags. Once a task is triggered (activated), all the programs assigned to the task execute in the order in which they are listed in the **Controller Organizer**.

  Programs are useful for projects developed by multiple programmers. During development, the code in one program that makes use of program-scoped tags can be duplicated in a second program, which minimizes the possibility of tag-name collisions.
- **Routines**: Routines are used to encapsulate executable code written in a single programming language.

  Routines contain the executable code. Each program has a main routine that is the first routine to execute within a program. You can use logic, such as the Jump to Subroutine (JSR) instruction, to call other routines. You can also specify an optional program fault routine.

IMPORTANT Currently, the Project Migrator converts only ladder instructions. SFC and structured text files are not converted.

As the Project Migrator converts the PLC-5 or SLC 500 logic, consider the program structures in the table below.

Conversion step	Page
Creating a Continuous Task on page 26	24 on page 26

Conversion step	Page
Creating Event Tasks on page 26	24 on page 26
Creating Periodic Tasks for Selectable Timed Interrupts (STIs) on page 26	24 on page 26
Converting Input Interrupts (DIIs/PIIs) on page 27	25 on page 27
Creating a Status File on page 27	25 on page 27

IMPORTANT For more information on Logix 5000 Controllers, see the Logix 5000 Controllers Design Considerations Reference Manual, publication 1756-RM094F-EN-P.

Creating a Continuous Task

A Logix controller supports one continuous task that operates in a self-triggered mode. It restarts itself after each completion. The continuous task operates as the lowest priority task in the controller (one priority level lower than the lowest periodic task). This means that all periodic tasks will interrupt the continuous task.

The Project Migrator automatically creates one continuous task named MainTask with a default watchdog setting of 500 msec. It contains a single program named MainProgram and uses a main routine named MainRoutine.

The Project Migrator creates a continuous task, but it uses the EVENT instruction to better simulate the PLC-5/SLC 500 behavior.

Creating Event Tasks

The Project Migrator also creates Event tasks for each program file configured in the PLC-5 Main Control Program (MCP).

To call each Event task, the Project Migrator creates EVENT instructions within the continuous task. It uses the PLC-5 status file to determine which is the first MCP and orders them accordingly, in the MainRoutine.

The SLC 500 processors do not contain an MCP, so ladder program 2, which is the main ladder program, becomes the main routine.

Creating Periodic Tasks for Selectable Timed Interrupts (STIs)

Processor status word 31 contains the number of the ladder program, if any, that is designated for use as a selectable timed interrupt (STI). The Project Migrator creates a Periodic task and converts this program file named _filenumber_STI into its main routine.

The Project Migrator retrieves the STI interval from the processor status file. If necessary, the Project Migrator converts the interval to a 1 msec time base. After the conversion, you will have to edit the task properties to specify its priority.

Processor status bit S:2/1 allows enabling and disabling of the STI. A Logix controller does not support this. The Project Migrator generates a PCE instruction if it encounters any references to S:2/1.

Converting Input Interrupts (DIIs/PIIs)

A Logix controller does not support input interrupts (DIIs or PIIs). If the PLC-5 processor has a PII or the SLC 500 processor has a DII, the Project Migrator converts it to a subroutine in the Continuous task. You must edit the Logix 5000 logic to call the converted routine.

Processor status word 46 identifies the program file to be used as a DII or PII. The Project Migrator generates a PCE instruction and places it in the converted DII/PII routine.

Creating a Status File

Within the continuous task, the Project Migrator automatically creates a subroutine named StatusFile. This StatusFile contains GSV instructions to retrieve the following controller information.

- The controller local date and time in human readable format
- Fault information about the controller provided by the FAULTLOG object
- Status for the Battery, bad or missing
- The physical hardware of the controller identified by the CONTROLLERDEVICE object
- Status for Mode switch in REMOTE
- Status for Forces enabled and present

There are special considerations for some data in the status file as shown in the table that follows.

This status data:	Is handled this way:
MCP status data	The PLC-5 processor can support from 1-16 main control programs. Each MCP uses 3 words of status data. Status words 80-127 contain this information.
STI status data	The Enhanced PLC-5 processor can also support a selectable timed interrupt. The processor status file contains the interrupt time interval and the number of the program file to execute. Status word 31 contains the program file number; status word 30 contains the interrupt time interval.
DII/PII status data	The PLC-5 and SLC 500 processors support an input interrupt. Status word 46 contains the number of the program file to execute. A Logix controller does not support this feature. If the import/export file contains PII status data, the PII program file is converted and placed as a routine in the Continuous program. The conversion process also places a PCE instruction in the converted routine to identify that the routine was used for a PII.
Indexed addressing	Status word 24 contains the current address index used for indexed addressing. A Logix controller does not use this index value. During the conversion, the process creates a tag for S24: S24 INT (Radix:=Decimal) := <value>

Converting Data

Introduction

A Logix controller is based on a 32-bit architecture, as opposed to the 16-bit architecture of PLC-5 and SLC 500 processors. To provide seamless conversion and the best possible performance, many data table values are converted to 32-bit values (DINT values). This chapter provides detailed information about converting various file types. The table that follows shows the file conversions "at a glance" and where in the chapter you can find the conversion detail.

PLC-5 or SLC file type	Logix array type	Radix	Comments	Page
O	INT	BINARY		28 on page 31
I	INT	BINARY		28 on page 31
S	INT	HEX	A PCE instruction is generated for each S address.	29 on page 32
B	DINT	BINARY	The 16-bit value is copied into the 32-bit location and sign-extended.	31 on page 33
T			Each address that references a PRE or ACC value generates a PCE instruction.	32 on page 34
C	COUNTER		A PCE instruction is generated when overflow (.OV) and underflow (.UN) bit fields are encountered.	33 on page 35
R	CONTROL			34 on page 36
R to Serial Port Control				35 on page 37
N	DINT	DECIMAL	The 16-bit value is copied into the 32-bit location and sign-extended.	35 on page 37
F	REAL			36 on page 37
A	INT	HEX		36 on page 38
D	DINT	HEX	The 16-bit value is copied into the 32-bit location and zero-filled.	36 on page 38
BT	MESSAGE			37 on page 38
MO	INT			38 on page 40
M1	INT			38 on page 40
MG	MESSAGE			39 on page 40

PLC-5 or SLC file type	Logix array type	Radix	Comments	Page
PD	PID			40 on page 42
SC			This is a SFC status type.	41 on page 42
ST	STRING		The RSLogix 5000 structure contains 1 16-bit word (INT) and 82 8-bit words (SINT).	42 on page 43
CT	MESSAGE			44 on page 45

How PLC-5 and SLC 500 files identify data table values

The PLC-5 and SLC 500 import/export files use DATA statements to identify file types, as shown in the example below.

```
DATA <file_reference>:<last_element_number>
<data_value>
```

The table that follows describes the fields in the example above:

This field:	Specifies the:
file_reference	file type For example, N identifies an integer file type.
last_element_number	size of the file The conversion process uses this value to determine the number of elements to place in the array used for this file. For example, DATA N7:9 means that file number 7 is an integer file with 10 elements.
data_value	contents of the file For example: DATA N7:2 10 11 12 shows that file number 7 is an integer file with three elements. The values of these elements are: N7:010 N7:111 N7:212

How Logix files identify file types

The Logix import/export file uses tag declarations to initialize values. For example:

This data table file and elements:	Could convert to:	Specifies:
F8 with 1 element	REAL := 3.25	a single, real value
N7 with 3 elements	DINT[3] ={42, -56, 1090}	an integer array with three elements
T4 with 2 elements	[2] := {{16#c0000000, 1000, 910}, {16#c0000000, 3000, 2550}	an array of two structures; each structure has three members

Converting Input (I) and Output (O) Data

The conversion process for I/O data tables tries to follow the layout of the input and output image tables in the PLC-5 and SLC 500 processor. To do this, the conversion process creates one, single-dimension array for I data and one, single-dimension array for O data. The size of the input and output image tables in the PLC-5 or SLC 500 processor determines the size of these converted arrays.

The conversion process creates single-dimension, INT arrays for I and O files. The tags names are I and O, respectively. The number of elements in the converted array is the same as the number of elements in the original data table file.

For example, in the ASCII text file:

This DATA statement:	Converts to:
DATA O:177 0X0000 0X0000	tag O type INT[128] (Radix := Binary) := {16#0000, ... }
DATA I:037 0X0000 0X0000	tag I type INT[32] (Radix := Binary) := {16#0000, ... }

The PLC-5 processor, SLC 500 processor, and Logix controllers use different addressing schemes for I/O data. For example:

Controller	I/O Addressing
PLC-5 processor	Base 8 (octal)
SLC 500 processor	Base 10 (decimal)
Logix controller	Base 10 (decimal)

To preserve the original address, the conversion process creates alias tags based on the physical address. For example:

Controller	Original Address	Converted Address	Alias Tag Name
PLC-5 processor	I:007	I[7]	I_007
	O:010	O[8]	O_010
	I:021/05	I[17].05	I_021_Bit05
	O:035/15	O[29].13	O_035_Bit015
SLC 500 processor	I:007	I[7]	I_007
	O:010	O[10]	O_010
	I:021/05	I[21].05	I_021_Bit05
	O:035/15	O[35].15	O_035_Bit015

Converting the Status (S) File Type

PC5 file migration

Status files are handled differently during the conversion depending on whether it is a PC5 or SLC file that is being migrated.

- The RSLogix tag name is S.
- RSLogix tag dimension is one more than the dimension specified after the colon in the Legacy DATA statement.
- Initial values follow the constant conversion rules.

The number of elements in the converted array is the same as the number of elements in the original data table. For example, in the ASCII text file:

PC5 DATA statement:	Converts to:
DATA S:127 0X0000 0X0000	S: INT[164] (Radix := Hex) := {16#0000, ... };

The table that follows shows some examples of S addresses and their Logix equivalents.

Original Address	Converted Address
S:3	S[3]
S:1/15	S[1].15
S:24	S24

There are special considerations for some data in the status file as shown in the table that follows:

Status data:	How handled:
MCP status data	The PLC-5 processor can support from 1-16 main control programs. Each MCP uses 3 words of status data. Status words 80-127 contain this information.
STI status data	The Enhanced PLC-5 processor can also support a selectable timed interrupt. The processor status file contains the interrupt time interval and the number of the program file to execute. Status word 31 contains the program file number; status word 30 contains the interrupt time interval
DII/PII status data	The PLC-5 and SLC 500 processors support an input interrupt. Status word 46 contains the number of the program file to execute. A Logix controller does not support this feature. If the import/export file contains PII status data, the PII program file is converted and placed as a routine in the Continuous program. The conversion process also places a PCE instruction in the converted routine to identify that the routine was used for a PII.
Indexed addressing	Status word 24 contains the current address index used for indexed addressing. A Logix controller does not use this index value. During the conversion, the process creates a tag for S24: S24 INT (Radix:=Decimal) := <*value*>

SLC file migration

- The RSLogix tag name is S.
- RSLogix tag dimension is based off the number of initial values present.
- Initial values follow the constant conversion rules.

- If legacy logic references the file type (S) with the number following, the number will be removed during the migration.

The number of elements in the converted array is the same as the number of elements in the original data table file. For example, in the ASCII text file:

SLC DATA statement:	Converts to:
DATA S:0 0X0000 0X0000 ... ... DATA S2:0 0X0000 0X0000	S: INT[128](Radix := Hex) := { 16#0000, ... }; S: INT[128](Radix := Hex) := { 16#0000, ... };

Tags created through GSV during conversion

- Status and Forcestatus are new INT tags to retrieve Status and Force enabled values through GSV created during conversion.
- DateTime is a DINT[7] array to retrieve the Date/Time values through GSV during conversion.
- MinorFaults is a DINT to retrieve the fault values through GSV created during conversion.

See Creating a Status File on page 27 in chapter 2 to understand how the Project Migrator creates status files and uses GSV instructions.

Converting the Binary (B) File Type

A B file is migrated by converting 16-bit values into 32-bit values by filling the upper 16 bits with zeros. This method of conversion lets instructions that manipulate B files work correctly, except for BSL, BSR, and BTD instructions. You have to rework these instructions because shifting bits that would have moved into another 16-bit word might only shift into the upper (or lower) 16 bits of the same 32-bit word in the Logix architecture.

The conversion process creates a single-dimension, DINT array for the B file. The tag name is B*x* (where *x* is the PLC-5 or SLC 500 data table file number). The number of elements in the converted array is the same as the number of elements in the original data table file.

For example, in the ASCII text file:

This DATA statement:	Converts to:
DATA B3:15 153 227	tag B3 type DINT[16](Radix := Binary) := {153, 227, ... }

The table that follows shows examples of B addresses and their Logix equivalents:

Original Address	Converted Address
B3.4/1	B3[4].1
B3/65	B3[4].1

Converting the Timer (T) File Type

Timers in the PLC-5 and SLC 500 processors consist of a 16-bit preset value, a 16-bit accumulator value, and a time base of 1 sec or 10 msec. s in a Logix controller consist of a 32-bit preset value, a 32-bit accumulator values, and a 1 msec time base.

The conversion process creates a single dimension array of structures for the T file. The tag name is T*x* (where *x* is the PLC-5 or SLC 500 data table file number). The number of elements in the converted array is the same as the number of elements in the original data table file. Each element in the array is a structure, which consists of three, 32-bit DINT words. The table that follows shows a comparison of the PLC-5/SLC 500 bits and the Logix bits:

Word	PLC-5/SLC 500 bits	Logix bits	Mnemonic	Description
0	15	31	EN	enable
0	14	30	TT	timing
0	13	29	DN	done
0	na	28	FS	first scan (SFC use)
0	na	27	LS	last scan (SFC use)
0	na	26	OV	overflow
0	na	25	ER	error
1	na	na	PRE	preset value
2	na	na	ACC	accumulator value

For example, in the ASCII text file:

This DATA statement:	Converts to:
DATA T4:1 0xE000 1 123	tag T4 type [2] := {16#E0000000, 1000, 123000} The .PRE and .ACC values were converted from a 1 second time base.

The table that follows shows some T addresses and their Logix equivalents:

Original Address	Converted Address
T4:1	T4[1]
T4:1/15 T4:1/EN T4:1.0/EN	T4[1].EN
T4:1.1 T4:1.PRE	T4[1].PRE
T4:1.2 T4:1.ACC	T4[1].ACC

Conversion rules

- The PRE and ACC values are converted to equivalents for a 1 msec time base.
- The first time base encountered for an individual is used for converting the preset and accumulator values each time that appears.
- Each logic reference to a PRE or ACC value generates a PCE instruction.

Converting the Counter (C) File Type

The conversion process creates a single dimension array of COUNTER structures for the C file. The tag name is C*x* (where *x* is the PLC-5 or SLC 500 data table file number). The number of elements in the converted array is the same as the number of elements in the original data table file. Each element in the array is a COUNTER structure, which consists of three, 32-bit DINT words. The following table shows a comparison of the PLC-5/SLC 500 counter and the Logix counter:

Word	PLC-5/SLC 500 bits	Logix bits	Mnemonic	Description
0	15	31	CU	count up
0	14	30	CD	count down
0	13	29	DN	done
0	12	28	OV	overflow
0	11	27	UN	underflow
0	10	26	UA	update accum(SLC only)
1	na	na	PRE	preset value
2	na	na	ACC	accumulator value

For example, in the ASCII text file:

This DATA statement:	Converts to:
DATA C5:4 0xF800 500 0 ...	tag C5 type COUNTER[5] := {{16#F8000000, 500, 0 }, ... }

- The PRE and ACC values do not receive any special manipulation during the conversion.
- PCE messages are generated along with OV or UN values.

The table that follows shows C addresses and their Logix equivalents:

Original Address	Converted Address
C5:2	C5[2]
C5:2/15 C5:2/CU C5:2.0/CU	C5[2].CU
C5:2.1 C5:2.PRE	C5[2].PRE

C5:2.2 C5:2.ACC	C5[2].ACC

Converting the Control (R) File Type

The conversion process creates a single dimension array of CONTROL structures for the R file. The tag name is R*x* (where *x* is the PLC-5 or SLC 500 data table file number). The number of elements in the converted array is the same as the number of elements in the original data table file. Each element in the array is a CONTROL structure, which consists of three, 32-bit DINT words. The table that follows is a comparison of the PLC-5/SLC 500 control structure and the Logix control structure:

Word	PLC-5/SLC 500 bits	Logix bits	Mnemonic	Description
0	15	31	EN	enable
0	14	30	EU	queue
0	13	29	DN	done
0	12	28	EM	empty
0	11	27	ER	error
0	10	26	UL	unload
0	9	25	IN	inhibit
1	NA	NA	LEN	length
2	NA	NA	POS	position

For example, in the ASCII text file:

This DATA statement:	Converts to:
DATA R6:19 0xFFF00 0 0 ...	tag R6 type CONTROL[20] := {{16#FF000000, 0,0 }, ... }

The LEN and POS values do not receive any special manipulation during the conversion.

The table that follows shows R addresses and their Logix equivalents:

Original Address	Converted Address
R6:3	R6[3]
R6:3/15 R6:3/EN R6:3.0/EN	R6[3].EN
R6:3.1 R6:3.LEN	R6[3].LEN

Converting the Control (R) File Type to Serial Port Control

The SERIAL_PORT_CONTROL is a structure similar to Control R. R types are converted to SERIAL PORT CONTROL tags only if the R file type is used in a serial port instruction.

During the conversion process, the Control R file type from the PLC-5/SLC is copied to both a CONTROL tag array and a SERIAL_PORT_CONTROL tag array in Logix Designer.

Once all of the R data has been migrated to the SERIAL_PORT_CONTROL type, you can remove the R data equivalent.

If an instruction that requires an R file type (or SERIAL_PORT_CONTROL type post conversion) uses an N file type instead, the N file type will be treated as an R file type and converted. Treating an N file type as an R file type requires 3 N elements.

Converting the Integer (N) File Type

The conversion process creates a single-dimension, INT array for the N file. The tag name is Nx (where x is the PLC-5 or SLC 500 data table file number). The number of elements in the converted array is the same as the number of elements in the original data table file.

For example, in the ASCII text file:

DATA statement	Converts to:
DATA N7:99 153 227	tag N7 type INT[100](Radix := Decimal) := {153, 227, ... }

The table that follows shows N addresses and their Logix equivalents:

Original Address	Converted Address
N7:0	N7[0]
N7:1/2	N7[1].2

Converting the Floating Point (F) File Type

The conversion process creates a single-dimension, REAL array for the F file. The tag name is F*x* (where *x* is the PLC-5 or SLC 500 data table file number). The number of elements in the converted array is the same as the number of elements in the original data table file.

For example, in the ASCII text file:

This DATA statement:	Converts to:
DATA F8:6 1.23 4.56	tag F8 type REAL[7] := {1.23, 4.56, ... }

The table that follows shows an example F address and its Logix equivalent:

Original Address	Converted Address
F8:3	F8[3]

Converting the ASCII (A) File Type

The conversion process creates a single-dimension, INT array for the A file. The tag name is A*x* (where *x* is the PLC-5 or SLC 500 data table file number). The number of elements in the converted array is the same as the number of elements in the original data table file.

For example, in the ASCII text file:

This DATA statement:	Converts to:
DATA A9:1 24930 25444	tag A9 type INT[2] := {24930, 25444}

The table that follows shows some A addresses and their Logix equivalents:

Original Address	Converted Address
A9:4	A9[4]
A9:5/6	A9[5].6

Converting the Decimal (D) File Type

The conversion process creates a single-dimension, INT array for the D file. The tag name is D*x* (where *x* is the PLC-5 or SLC 500 data table file number). The number of elements in the converted array is the same as the number of elements in the original data table file.

For example, in the ASCII text file:

This DATA statement:	Converts to:
DATA D10:2 256 512 768	tag D10 type INT[3] := {256, 512, 768}

The table that follows shows an example D address and its Logix equivalents:

Original Address	Converted Address
D10:0	D10[0]

Converting the Block-Transfer (BT) File Type

The BT file type appears only in 6200 Legacy files (PC5).

The conversion process creates an individual MESSAGE structure for each element in the BT file (not an array of structures), because MESSAGE tags cannot be array elements. The tag name is BTx (where x is the PLC-5 or SLC 500 data table file number).

The initial values appearing in the Legacy DATA statement are first partitioned into sets of 6 individual elements.

The mapping from BT type to Logix Designer MESSAGE type is shown in the table that follows:

Word	PLC-5/SLC 500 bits	Logix bits	Mnemonic	Logix Designer Mnemonic	Description
0	15	31	EN	EN	enable
0	14	30	EU	EU	queue
0	13	29	DN	DN	done
0	12	28	EM	EM	empty
0	11	27	ER	ER	error
0	10	26	UL	UL	unload
0	9	25	IN	IN	inhibit
0	8	24	FD	FD	found
0	7	na	RW	na	
1	na	na	RLEN	REQ_LEN	length
2	na	na	DLEN	DN_LEN	position
3	na	na	FILE	RemoteIndex	
4	na	na	ELEM	RemoteElement	
5	na	na	RGS	na	rack, group, slot

Only the local message information is converted, which consists of the message type, the message itself, and the message length. After the conversion, use the programming software to configure the message.

For example, in the ASCII text file:

This DATA statement:	Converts to:
DATA BT9:1	BT11_007 : MESSAGE (MessageType := Block Transfer Write, RequestedLength := 21, LocalElement := N9[162], CacheConnections := TRUE);

The table that follows shows some BT addresses and their Logix equivalents:

Original Address	Converted Address
BT11:5	BT11_5
BT11:5.RLEN	BT11_5.RLEN

Block-transfer conversion rules

- The MessageType is set to either Block Transfer Read or Block Transfer Write, depending on the PLC-5 block-transfer instruction.
- The LocalTag is set to the tag specified by the PLC-5 block-transfer instruction.

Converting the M0 and M1 File Types

The conversion process creates one single-dimension, INT array for the M0*x* and M1*x* (where *x* is the PLC-5 or SLC 500 data table file number). The number of elements in the converted array is the same as the number of elements in the original data table file.

For example, in the ASCII text file:

This SLOT statement:	Converts to:
SLOT 4 1747-SN SCAN_IN 32 SCAN_OUT 32 ISR 0 M0_SIZE 3300 M1_SIZE 3300 G_FILE 8	tag M0_4 type INT[3300] () := [0, 0, ...] tag M1_4 type INT[3300] () := [0, 0, ...]

The table that follows shows some M0/M1 addresses and their Logix equivalents:

Original Address	Converted Address
M0:0/1	M0_0[1]
M1:1/1	M1_1[1]

Converting the Message (MG) File Type

An MG file is converted to a MESSAGE type tag. The MG file type appears only in 6200 Legacy files (PC5).

The conversion process creates an individual MESSAGE structure for each element in the MG file (not an array of structures). MESSAGE tags cannot be array elements. The tag name is MG*x* (where *x* is the PLC-5 or SLC 500 data table file number). The table below shows a comparison of the PLC-5/SLC 500 MG structure and the Logix Designer MESSAGE structure:

Message type	Logix message type
TYPEDREAD	PLC5 Typed Read
TYPEDWRITE	PLC5 Typed Write
PLC3_WORDRANGEREAD	PLC3 Word Range Read
PLC3_WORDRANGEWRITE	PLC3 Word Range Write
PLC2_UNPROTECTEDREAD	PLC2 Unprotected Read
PLC2_UNPROTECTEDWRITE	PLC2 Unprotected Write
SLC_TYPEDREAD	SLC Typed Read
SLC_TYPEDWRITE	SLC Typed Write

For example, in the ASCII text file:

This DATA statement:	Converts to:
MG9:0 PLC-5 MSG message typePLC-2 unprotected read local data table addressN7:0 size in elements1 port1A targetaddress10 target node2 local	MG94_019 : MESSAGE (MessageType := PLC5 Typed Write, RequestedLength := 2, LocalElement := CT10[17], RemoteElement := N10:17, CacheConnections := TRUE);

The initial values appearing in the Legacy DATA statement are first partitioned to into sets of 56 individual elements.

The mapping from MG type to Logix Designer message type is shown below:

Word	Legacy Bit #	RSLogix 5000 Bit #	Legacy Mnemonic	RSLogix 5000 Mnemonic	Description
0	15	31	EN	EN	Enable
0	14	30	ST	ST	
0	13	29	DN	DN	Done
0	12	28	ER	ER	Error
0	11	27	CO	CO	
0	10	26	EW	EW	
0	9	25	NR	NR	
0	8	24	TO	TO	
1	N/A	N/A	ERR	ERR	Error value
2	N/A	N/A	RLEN	REQ_LEN	Length
3	N/A	N/A	DLEN	DN_LEN	Position

The table that follows shows some MG addresses and their Logix equivalents.

Original Address	Converted Address
MG9:5	MG9_5
MG9:5.ERR	MG9_5.ERR

Message conversion rules

- The MessageType is set to the appropriate type, depending on the message instruction.
- The LocalTag attribute of the MESSAGE structure is computed as follows:
  - The file number is extracted from the most significant byte of the 16th element of the set of initial values for an element.
  - The word offset is extracted from the least significant byte of the 16th element of the set of initial values.
  - The file number is also used to determine what the file type is based on usage.
  - The generated tag is the value of the LocalTag attribute.

- After the conversion, you need to provide the communication path of the message.

Converting the PID (PD) File Type

A PD file is converted to a PID type tag.

The conversion process creates a single dimension array of PID structures for the PD file. The tag name is PD*x* (where *x* is the PLC-5 or SLC 500 data table file number). The number of elements in the converted array is the same as the number of elements in the original data table file. Each element in the array is a PID structure.

For example, in the ASCII text file:

This DATA statement:	Converts to:
DATA PD10:10 256 0 0 0 0 0 0 0 0 0 0 0 0 0.1 0 0 0 0 0 0 0 0 0 0 0 0 15 10 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...	tag PD10 type PID10[11].1 := {536870912, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.1, 0 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]} ...

The following table lists shows some PD addresses and their Logix equivalents:

Original Address	Converted Address
PD10:1	PD10[1]
PD10:1/15 PD10:1/EN PD10:1.0/15	PD10[1].EN
PD10:1.2	PD10[1].SP

Although the PID instruction has been migrated, the PID instruction has many parameters that do not convert directly to the Logix Designer application. The migration must be verified.

Converting SFC Status (SC) Type

For the SC type, a UDT is created that mimics the file type structure of an SC so the data is not lost. Look for the PCE instructions that are created for all SC-related statements, address references, and instructions. The table that follows shows the file comparisons:

Word	Legacy Bit #	Logix Designer UDT Bit#	Mnemonic	Description
0	0	0	SA	
0	1	1	FS	First Scan (SFC use)
0	2	2	LS	Last Scan (SFC use)
0	3	3	OV	Overflow
0	4	4	ER	Error
0	5	5	DN	Done

Word	Legacy Bit #	Logix Designer UDT Bit#	Mnemonic	Description
1	NA	NA	BASE	
2	NA	NA	PRE	
3	NA	NA	TIM	

This DATA statement:	Converts to:
DATA SC10:0 0X003F 0 0 ...	SC10 : SC_UDT[1] := { {16#0000003F, 0, 0, ...}, ... };

Converting the ASCII String (ST) File Type

ASCII string files are handled differently during the conversion depending on whether it is a PC5 or SLC file that is being migrated. The size of each structure type is equivalent. However, there are some data type differences. The tables that follow compare the ASCII string structure with the Logix Designer string structure.

Legacy ASCII String Structure		
Legacy ASCII string structures are made up of 42 16-bit words		
Word	Mnemonic	Description
0	LEN	This element of the structure contains the length of the string
1-41	N/A	These 41 words contain the string data. Two ASCII bytes are stored in each word.

Logix Designer String Structure		
The Logix Designer structure contains 1 16-bit word (INT) and 82 8-bit words (SINT)		
Mnemonic	Type	Description
LEN	INT	This is the length of the string
STR	SINT [82]	Each SINT contains a single ASCII character.

The data type differences are described below.

PC5 file translation

For this file format:

- The strings' data values remain as strings.
- The LEN is determined when the Logix Designer tag is initialized.

For example, in the ASCII text file:

This DATA statement:	Converts to:
DATA ST15::1 ...	ST15 : STRING[2] := { {5, {72,101,108,108,111,0,...0}}, {5, {84,104,101,114,101,0,...0}} }; **Note**: No empty spaces are allowed in the initialization of structures. Therefore the STR element of the tag is shown with zeros padding its contents.

SLC file migration

For this file format:

- The ASCII strings are broken apart. In other words, a data statement is created for each ASCII string, not each ASCII string file.
- The Project Migrator then creates a single dimension tag.
- These data statements do not display the <# of Elements> after the <File Reference>. In this case, this integer represents an actual element number. The data values contain integers. The first value encountered is the byte length in decimal format. The remaining values are shown as 41 16- bit words in hexadecimal format.

For example, in the ASCII text file:

This DATA statement:	Converts to:
DATA ...ST15:000 5 0X4865 0X6C6C 0X6F00 0X0000 0X0000 ... DATA ...ST15:001 5 0X5468 0X6572 0X6500 0X0000 0X0000 0X0000	ST15 : STRING[2] := { {5, {72,101,108,108,111,0,...0}}, {5, {84,104,101,114,101,0,...0}} }; **Note**: No empty spaces are allowed in the initialization of structures. Therefore, the STR element of the tag is shown with zeros padding its contents.

The following table summarizes the ladder instructions specifically related to strings.

Description	PLC-5 Instruction	SLC 500 Instruction	Logix Instruction
string to integer conversion	ACI	ACI	STOD
integer to string conversion	AIC	AIC	DTOS
string to real conversion	na	na	STOR
real to string conversion	na	na	RTOS
string compare for equal	ASR	ASR	EQU
string compare for not equal	na	na	NEQ
string compare for greater than	na	na	GRT
string compare for greater than or equal	na	na	GEQ
string compare for less than	na	na	LES
string compare for less than or equal	na	na	LEQ
append on string to another	ACN	ACN	CONCAT
move characters from one string to another	AEX	AEX	MID
search one string for a matching string	ASC	ASC	FIND
delete characters from a string	na	na	DELETE

Description	PLC-5 Instruction	SLC 500 Instruction	Logix Instruction
insert a string into another string	na	na	INSERT
convert a string to all uppercase letters	na	na	UPPER
convert a string to all lowercase letters	na	na	LOWER

Converting the ControlNet (CT) File Type

The CT type appears only in the PC5 files. The initial values appearing in the Legacy Data statement are first partitioned into sets of 22 individual elements. The table below shows the file comparisons.

Word	Legacy Bit #	RSLogix 5000 Bit #	Legacy Mnemonic	RSLogix 5000 Mnemonic	Description
0	15	31	TO	TO	
0	14	30	EN	EN	enable
0	13	29	ST	ST	
0	12	28	DN	DN	done
0	11	27	ER	ER	error
0	10	26	CO	CO	
0	9	25	EW	EW	
1	na	na	ERR	ERR	error value
2	na	na	RLEN	REQ_LEN	length
3	na	na	DLEN	DN_LEN	position
4	na	na	FILE	RemoteIndex	
5	na	na	ELEM	RemoteElement	

For each partition of the CT array, a new RSLogix MESSAGE structure is created. This structure's name is formed by concatenating the Legacy filename, and the Legacy element index separated by an underscore.

The MessageType and LocalTag attributes of the MESSAGE structure are set later when a CIO instruction that uses this CT element as the fourth operand is encountered. Then, the fifth operand is used to set the LocalTag.

Converting Constant Values

The conversion process maintains constants. The format of converted constants varies slightly to conform to Logix format requirements.

For example:

Constant type	PLC-5/SLC 500 example	Conversion	Conversion rule
Integer	&N49 -49	49 -49	remove &N, if present copy remainder of constant
Binary	&B00110001	2#00110001	replace &B with 2# copy remainder of constant
ASCII	&A1 &Amx	16#0031 16#6D78	convert to hex constant
Hex	&H0031 0x0032 0X0033	16#0031 16#0032 16#0033	replace &H, 0x, or 0X with 16# copy remainder of constant

Constant type	PLC-5/SLC 500 example	Conversion	Conversion rule
BCD	&D0049	16#0031	convert to hex constant
Octal	&061	8#61	replace &0 with 8# copy remainder of constant
Float	-12.34E-12 3.45	-12.34E-12 3.45	this syntax is completely compatible copy the constant as is

Converting Indirect Addresses

Indirect addressing is when a part of an address is replaced with a reference to another address. The PLC-5 and SLC 500 processors can use an address reference to define these address parts:

- file number
- word or element number
- bit number (only for B type addresses)

The Project Migrator supports indirect addresses, except when the indirection is an array specification. Indirect array specifications are converted to aliases, as shown in the example that follows.

Type	PLC-5/SLC 500 example	Conversion	Conversion rule
File number	N[N7:0]:5	na	The Project Migrator cannot convert an indirect file number. A PCE instruction is generated.
Word or element number	N12:[N7:0]	N12[N7_0]	N7:0 converts to array tag N7[0]. Alias N7_0 replaces the indirect address.
	N12:[T4:1.PRE]	N12[T4_1_PRE]	T4:1.PRE converts to array tag T4[1].PRE. Alias T4_1_PRE replaces the indirect address.
Bit number	B3/[N7:0]	B3[N7_0 / 16].[N7_0 AND 15]	The conversion process must convert to the correct word and bit within that word. Alias N7_0 replace the indirect address.

Converting indirect addressing on the file number

Indirect addressing on the file number can actually be implemented after the conversion process if the original data table files are consecutive. For example, a PLC-5 processor has five program files with heat treating "recipes" in them.

Element	Description
0	Recipe number
1	Heat segment 1: time in minutes
2	Heat segment 1: temperature in F °
3	Heat segment 2: time in minutes
4	Heat segment 2: temperature in F°
5	Room temperature cooling time in minutes

In the ASCII text file:

DATA N10:5
0, 5, 350, 15, 200, 60

DATA N11:5
1, 10, 400, 25, 300, 15

DATA N12:5
2, 5, 500, 20, 350, 90

DATA N13:5
3, 50, 300, 120, 150, 90

DATA N14:5
4, 10, 700, 30, 500, 240

These data files convert to:

N10 : DINT[6] (Radix:=Decimal):=[0, 5, 350, 15, 200, 60];
N11 : DINT[6] (Radix:=Decimal):=[1, 10, 400, 25, 300, 15];
N12 : DINT[6] (Radix:=Decimal):=[2, 5, 500, 20, 350, 90];
N13 : DINT[6] (Radix:=Decimal):=[3, 50, 300, 120, 150, 90];
N14 : DINT[6] (Radix:=Decimal):=[4, 10, 700, 30, 500, 240];

Use a text editor to modify these integer files into a two-dimensional array:

RECIPES: DINT[6, 6] (Radix:=Decimal):=[0, 5, 350, 15, 200, 60,
1, 10, 400, 25, 300, 15,
2, 5, 500, 20, 350, 90,
3, 50, 300, 120, 150, 90,
4, 10, 700, 30, 500, 240];

Assume that there is an indirect address reference to N[N7:0]:0 to read the recipe number. In the converted project, use RECIPES[N7_0, 0], where N7_0 is the converted form of N7:0. You have to modify the bounds checking because the original file numbers ranged from 10 to 14, but the first index in the two-dimensional array ranges from 0 to 4.

Converting Indexed Addresses

Indexed addresses in the PLC-5 and SLC 500 processors are when a # character precedes the address.

Addresses controlled by the processor status word S:24

The processor status word S:24 contains the current index value to add to an address reference. The conversion process adds the value of S:24 to an indexed values it converts and places a PCE instruction in the output import/export file.

For example:

This address:	Converts to:
#N7:2	N7[2 + S24]

Addresses that specify data in files (Logix arrays)

Indexed addresses are also used with the file instructions to operate on files of data. These instructions use a CONTROL structure to determine the index value -- the current position within the file.

A Logix controller stores data in arrays, rather than files. Indexed addresses for PLC-5 and SLC 500 file instructions are converted to array tags, without adding the value of status word S:24.

For example:

This instruction:	Converts to:
AVE #N10:0 N11:0 R6:0 6 0	AVE(N10[0], 0, N11[0], R6[0], 6, 0)

Alias Creation Rules

The Project Migrator tool creates Logix Designer alias declarations following specific rules.

- Aliases are literals assigned to specific tag references. These literals are then used in place of the associated tag reference.
- The Project Migrator creates alias declarations based upon the content of the legacy documentation import/export file.
- Aliases are also created when the file number, word offset, or bit offset of an address is indirect.
- Aliases may be created when you choose to have the Project Migrator create aliases during the migration process.
- Alias declarations are always associated with a tag declaration. If a tag declaration created by the Project Migrator has an associated radix, then any aliases based in that tag must be assigned the same radix.

Converting Symbols

The conversion process converts a symbol to a description. The Project Migrator gives you the option to have the system create alias tags for symbols.

No aliases created (default)

The Project Migrator converts symbols *without* aliases being created, as follows:

The PLC-5 and SLC 500 import/export file uses SYM statements to identify symbols:

SYM <*address_reference*> <*literal*>

The following table describes the fields in the example above.

This field:	Specifies the:
address_reference	address The conversion process creates a tag to correspond to the actual address.
literal	symbol text The conversion process converts the symbol text to a description.

The PLC-5 and SLC 500 processors support some symbol formats that are not supported in a Logix controller. In these cases, the conversion process modifies the symbol text.

The table below shows how the conversion process modifies the symbol text.

Logix tag:	SYM statement:	Modified tag:
N7 : INT[9] (Radix := Decimal)	SYM N7:2 Kitty	N7 : INT[9] (Radix := Decimal, Comment[2]:="Kitty")
B3 : INT[5] (Radix := Binary)	SYM B3:4/5 Puppy	B3 : INT[5] (Radix := Binary, Comment[4].5:="Puppy")
T4 : [2]	SYM T4:0 Ducky SYM T4:1 2ndDuck	T4 : [2] (Comment[0]:="Ducky", Comment[1]:="_2ndDuck")
na	SYM N[N7:0]:0 Pig	This address format is not supported in the conversion process. No tag is created.

If an address reference has both a symbol and an address comment, the conversion process concatenates the symbol to the end of the address comment.

Aliases created

If you choose to have the Project Migrator create aliases, the migration process is the same, but a Logix Designer alias is generated with the SYM "name" as the (alias) tag name and the <address reference> is the alias reference.

Generating alias tags uses up memory in the Logix 5000 processor.

The following table shows the difference between the symbol conversion options.

Associated Tag	Symbol Statement	Symbol as Tag Comment	Symbol as Alias
N7 : INT[9] (Radix := Decimal);	SYM N7:2 Kitty	N7 : INT[9] (Radix := Decimal, Comment[2]="Kitty");	Kitty OF N7[2]
B3 : INT[5] (Radix := Binary);	SYM B3:4/5 Puppy	B3 : INT[5] (Radix := Binary, Comment[5]="Puppy");	Puppy OF B3[4].5
T4 : [2];	SYM T4:0 Ducky SYM T4:1 2dnDuck	T4 : [2] (Comment[0]="Ducky", Comment[1] = "_2ndDuck";	Ducky OF T4[0] _2ndDuck OF T4[1]
N/A	SYM N[N7:0]:0 Piglet	N/A	No alias will be created. Unsupported address format

Converting Address Comments

The conversion process converts address comments to descriptions.

The PLC-5 and SLC 500 import/export file uses AC statements to identify address comments:

AC [*formatting_keyword*] <*address_reference*> <"comment_text">Where:

This field:	Specifies the:
formatting_keyword	format of the comment text. The PLC-5 and SLC 500 processors support formatting commands for comment text. The conversion process ignores these formatting keywords.
address_reference	address The conversion process creates a tag to correspond to the actual address.
literal	comment text The conversion process converts the comment text to a description.

For example:

Logix tag:	AC statement:	Modified tag:
N7 : INT[9] (Radix := Decimal)	AC N7:2 Kitty	N7 : INT[9] (Radix := Decimal, Comment[2]:="Kitty")
B3 : INT[5] (Radix := Binary)	AC B3:4/5 Puppy	B3 : INT[5] (Radix := Binary, Comment[4].5:="Puppy")

If an address reference has both a symbol and an address comment, the conversion process concatenates the symbol to the end of the address comment.

Converting Instructions

Introduction

This chapter explains how the Project Migrator converts individual instructions.

Conversion Rules Review

When converting instructions, the Project Migrator follows these rules:

- Instructions that are not supported by Logix 5000 controllers are converted with all their parameters intact. A PCE (Programming Conversion Error) is generated to highlight the error.
- PLC-5 and SLC 500 parameters use 16 bits. They are extended to 32 bits for Logix parameters.
- All references to S:0/0, S:0/1, S:0/2, and S:0/3 are replaced with the Logix keywords S:C, S:V, S:Z, and S:N, respectively.
- Each reference to the OV and UN bits of a COUNTER file type results in a PCE instruction.
- Each logic reference to a PRE or ACC value generates a PCE instruction.
- Any constant that represents a serial port is always converted to 0, the Logix serial port.
- Directly modifying the ladder logic text of the PC5/SLC file before importing can cause a syntax error. The Project Migrator shows the error and where to find it. It then gives the option to correct the error and import the file again. Syntax errors should not occur if the program is exported directly from the PLC-5/SLC application.

Instruction List

The following table lists the PLC-5 and SLC 500 instructions alphabetically. It also includes comments to identify conversion issues:

Instruction	Name	Processor	Parameter	Considerations
ABL	ASCII Test Buffer for Line	PLC-5 SLC 500	Channel	Channel is set to zero. Generates a serial port control tag.
			Control	
			Characters	
ABS	Absolute Value	SLC 500	Source	
			Destination	
ACB	ASCII Number of Characters in Buffer	PLC-5	Channel	Channel is set to zero. Generates a serial port control tag.
			Control	
			Characters	
ACI	ASCII String to Integer	PLC-5 SLC 500	Source	
			Destination	

Instruction	Name	Processor	Parameter	Considerations
ACL	ASCII Clear Buffer	SLC 500	Channel	Channel is set to zero. Generates a serial port control tag.
			Transmit Buffer	
			Receive Buffer	
ACN	ASCII String Concatenate	PLC-5 SLC 500	Source A	
			Source B	
			Destination	
ACS	Arc Cosine	PLC-5 SLC 500	Source A	
			Destination	
ADD	Add	PLC-5 SLC 500	Source A	
			Source B	
			Destination	
AEX	ASCII String Extract	PLC-5 SLC 500	Source	
			Index	
			Number	
			Destination	
AFI	Always False	PLC-5	na	
AHL	ASCII Set/Reset Handshake Lines	PLC-5 SLC 500	Channel	Channel is set to zero. Generates a serial port control tag.
			AND Mask	Does not convert S:24 for indexing. Uses .POS value from Control.
			OR Mask	Does not convert S:24 for indexing. Uses .POS value from Control.
			Control	
			Channel Status	
AIC	ASCII Integer to String	PLC-5 SLC 500	Source	
			Destination	
AND	Logical AND	PLC-5 SLC 500	Source A	
			Source B	
			Destination	
ARD	ASCII Read Characters	PLC-5 SLC 500	Channel	Channel is set to zero. Generates a serial port control tag.
			Destination	Does not convert S:24 for indexing. Uses .POS value from Control.
			Control	
			String Length	
			Characters Read	
ARL	ASCII Read Line	PLC-5 SLC 500	Channel	Channel is set to zero. Generates a serial port control tag.
			Destination	Does not convert S:24 for indexing. Uses .POS value from Control.
			Control	
			String Length	
			Characters Read	
ASC	ASCII String Search	PLC-5 SLC 500	Source	
			Index	
			Search	
			Result	
ASN	Arc Sine	PLC-5	Source	

Instruction	Name	Processor	Parameter	Considerations
		SLC 500	Destination	
ASR	ASCII String Compare	PLC-5 SLC 500	Source A	
			Source B	
ATN	Arc Tangent	PLC-5 SLC 500	Source	
			Destination	
AVE	Average	PLC-5	File	Does not convert S:24 for indexing.
			Destination	Inserts 0 for dimension to vary.
			Control File	
			Length	
			Position	
AWA	ASCII Write with Append	PLC-5 SLC 500	Channel	Channel is set to zero. Generates a serial port control tag.
			Source	Does not convert S:24 for indexing. Uses .POS value from Control.
			Control	
			String Length	
			Characters Sent	
AWT	ASCII Write	PLC-5 SC 500	Channel	Channel is set to zero. Generates a serial port control tag.
			Source	Does not convert S:24 for indexing. Uses .POS value from Control.
			Control	
			String Length	
			Characters Sent	
BND	Branch End	PLC-5 SLC 500	na	Converts to right bracket (]).
BRK	BRK	PLC-5	na	
BSL	Bit Shift Left	PLC-5 SLC 500	File	Does not convert S:24 for indexing. Logs message directly in the rung along with the PCE instruction.
			Control File	
			Bit Address	
			Length	If the length is greater than 1, ensure the correct bit numbers are being operated on by using ONS and BTD instructions in parallel branches.
BSR	Bit Shift Right	PLC-5 SLC 500	File	Do not use S:24 for indexing. Logs message directly in the rung along with the PCE instruction.
			Control File	
			Bit Address	
			Length	If the length is greater than 1, ensure the correct bit numbers are being operated on by using ONS and BTD instructions in parallel branches.
BST	Branch Start	PLC-5 SLC 500	na	Converts to left bracket ([).
BTD	Bit Distribute	PLC-5	Source	
			Source Bit	
			Destination	
			Destination Bit	
			Length	
BTR	Block-Transfer Read	PLC-5	Rack	Ignores rack parameter. Converts instruction to MSG instruction and generates a PCE instruction.

Instruction	Name	Processor	Parameter	Considerations
			Group	Ignores group parameter.
			Module	Ignores module parameter.
			Control Block	
			Data File	Uses this data file to set the LocalTag attribute. Add RES and FAL instructions to make adjustments for the 16-bit to 32-bit conversion.
			Length	Ignores the length parameter.
			Continuous	Ignores the continuous parameter.
BTW	Block-Transfer Write	PLC-5	Rack	Ignores rack parameter. Converts instruction to MSG instruction and generates a PCE instruction.
			Group	Ignores group parameter.
			Module	Ignores module parameter.
			Control Block	
			Data File	Uses this data file to set the LocalTag attribute. Add RES and FAL instructions to make adjustments for the 16-bit to 32-bit conversion.
			Length	Ignores the length parameter.
			Continuous	Ignores the continuous parameter.
CIO	ControlNet I/O Transfer	PLC-5	Control Block	Converts to a MSG instruction and generates a PCE instruction.
CIR	Custom Input Routine	PLC-5	na	There is no Logix equivalent. Generates a PCE instruction. AGA3, AGA7 and API routines use this instruction. See Converting CAR routines on page 62.
CLR	Clear	PLC-5 SLC 500	Destination	
CMP	Compare	PLC-5	Expression	Check the converted expression for correct precedence order.
COP	Copy	PLC-5 SLC 500	Source	Does not convert S:24 for indexing. If source and destination types differ, logs message directly in the rung along with the PCE instruction.
			Destination	Does not convert S:24 for indexing.
			Length	
COR	Custom Output Routine	PLC-5	na	There is no Logix equivalent. A PCE instruction is generated. AGA3, AGA7 and API routines use this instruction.
COS	Cosine	PLC-5 SLC 500	Source	
			Destination	
CPT	Compute	PLC-5 SLC 500	Destination	
			Expression	Check the converted expression for correct precedence order.
CTD	Count Down	PLC-5 SLC 500	Counter	
			Preset	
			Accum	
CTU	Count Up	PLC-5 SLC 500	Counter	
			Preset	
			Accum	
DCD	Decode 4 to 1 of 16	SLC 500	Source	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
			Destination	
DDT	Diagnostic Detect	PLC-5	Source	Does not convert S:24 for indexing. Follow the DDT instruction with MOV and FAL instruction on parallel branches to ensure the correct bits are being operated on.

Instruction	Name	Processor	Parameter	Considerations
			Reference	Does not convert S:24 for indexing.
			Result	Does not convert S:24 for indexing.
			Compare Control	
			Length	
			Position	
			Result Control	
			Length	
			Position	
DDV	Double Divide	SLC 500	na	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
DEG	Degree	PLC-5 SLC 500	Source	
			Destination	
DFA	Diagnostic Fault Annunciator	PLC-5	na	There is no Logix equivalent. Logs a message to the message directly in the rung along with the PCE instruction. The DDMC routine uses this instruction to provide diagnostic and automatic messaging capabilities to an HMI. See Converting CAR routines on page 62.
DIV	Divide	PLC-5 SLC 500	Source A	
			Source B	
			Destination	
DTR	Data Transition	PLC-5	Source	
			Mask	
			Reference	
ENC	Encode 1 of 16 to 4	SLC 500	Source	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
			Destination	
EOC	End of SFC Compression	PLC-5	na	Ignores as part of an SFC section.
EOR	End of Rung	PLC-5 SLC 500	na	No action is taken.
EOT	End of Transition	PLC-5	na	Ignores as part of an SFC section.
ESE	End of SFC Section	PLC-5	na	Ignores as part of an SFC section.
EOP	End of SFC Program	PLC-5	na	Ignores as part of an SFC section.
EQU	Equal to	PLC-5 SLC 500	Source A	
			Source B	
ERI	Error on Input Instruction	PLC-5	na	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
ERO	Error on Output Instruction	PLC-5	na	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
ESI	End of SFC Simultaneous Branch	PLC-5	na	Ignores as part of SFC section.
FAL	File Arithmetic	PLC-5	Control	
			Length	
			Position	
			Mode	

Instruction	Name	Processor	Parameter	Considerations
			Destination	Uses the .POS value for indexing, not S:24.
			Expression	Uses the .POS value for indexing, not S:24. Check converted expression for correct precedence order.
FBC	File Bit Compare	PLC-5	Source	Does not convert S:24 for indexing. Follow the DDT instruction with MOV and FAL instruction on parallel branches to ensure the correct bits are being operated on.
			Reference	Does not convert S:24 for indexing.
			Result	Does not convert S:24 for indexing.
			Compare Control	
			Length	
			Position	
			Result Control	
			Length	
			Position	
FFL	FIFO Load	PLC-5 SLC 500	Source	
			FIFO	Does not convert S:24 for indexing.
			Control File	
			Length	
			Position	
FFU	FIFO Unload	PLC-5 SLC 500	FIFO	Does not convert S:24 for indexing.
			Destination	
			Control File	
			Length	
			Position	
FLL	File Fill	PLC-5 SLC 500	Source	
			Destination	
			Length	Does not convert S:24 for indexing.
FOR	For Loop	PLC-5	Label	Converts label "n" to "label_n" because a Logix label cannot be a number. See Converting FOR/NXT/BRK instructions on page 63.
			Index	
			Initial Value	
			Terminal Value	
			Step Size	
FRD	From BCD	PLC-5 SLC 500	Source	
			Destination	
FSC	File Search and Compare	PLC-5	Control	
			Length	
			Position	
			Mode	
			Expression	Uses the .POS value for indexing, not S:24. Check converted expression for correct precedence order.
GEQ	Greater Than or Equal to	PLC-5 SLC 500	Source A	
			Source B	
GRT	Greater Than	PLC-5 SLC 500	Source A	
			Source B	

Instruction	Name	Processor	Parameter	Considerations
HSC	High Speed Counter	SLC 500	Counter	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
			Preset	
HSD	HSC Interrupt Disable	SLC 500	Type	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
			Counter	
			Preset	
			Accum	
HSE	HSC Interrupt Enable	SLC 500	Counter	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
HSL	HSC Load	SLC 500	Counter	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
			Source	
			Length	
IDI	Immediate Data Input	PLC-5	Data File Offset	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
			Length	
			Destination	
IDO	Immediate Data Output	PLC-5	Data File Offset	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
			Length	
			Destination	
IID	I/O Interrupt Disable	SLC 500	Slots	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
IIE	I/O Interrupt Enable	SLC 500	Slots	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
IIM	Immediate Input with Mask	SLC 500	Slot	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
			Mask	
			Length	
IIN	Immediate Input	PLC-5	RRG	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
INT	I/O Interrupt	SLC 500	na	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
INV	Invert	PLC-5	na	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
IOM	Immediate Output with Mask	SLC 500	na	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
IOT	Immediate Output	PLC-5	RRG	
JMP	Jump	PLC-5 SLC 500	Label	Converts label "n" to "label_n" because a Logix label cannot be a number.
JSR	Jump to Subroutine	PLC-5 SLC 500	Ladder Program	Converts to a routine name.
			Input Parameters	
			Return Parameters	
LAB	Label	PLC-5	na	Ignores as part of SFC section.
LBL	LBL	PLC-5 SLC 500	Label	Converts label "n" to "label_n" because a Logix label cannot be a number. You must modify the converted FOR instruction.

Instruction	Name	Processor	Parameter	Considerations
LEQ	Less Than or Equal to	PLC-5 SLC 500	Source A	
			Source B	
LES	Less Than	PLC-5 SLC 500	Source A	
			Source B	
LFL	LIFO Load	PLC-5 SLC 500	Source	
			LIFO	Does not convert S:24 for indexing.
			Control File	
			Length	
			Position	
LFU	LIFO Unload	PLC-5 SLC 500	LIFO	Does not convert S:24 for indexing.
			Destination	
			Control File	
			Length	
			Position	
LIM	Limit	PLC-5 SLC 500	Low Limit	
			Test	
			High Limit	
LN	Natural Log	PLC-5 SLC 500	Source	
			Destination	
LOG	Log to the Base 10	PLC-5 SLC 500	Source	
			Destination	
MCR	Master Control Relay	PLC-5 SLC 500	na	
MEQ	Mask Compare Equal to	PLC-5 SLC 500	Source Operand	
			Source Mask	
			Compare Operand	
MOD	Modulo Divide	PLC-5 SLC 500	na	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
MOV	Move	PLC-5 SLC 500	Source	
			Destination	
MSG	Message	PLC-5 SLC 500	Type	Logs message and generates a PCE instruction. Add RES and FAL instructions to make adjustments for the 16-bit to 32-bit conversion. You must configure MSG communication parameters.
MUL	Multiply	PLC-5 SLC 500	Source A	
			Source B	
			Destination	
MVM	Move with Mask	PLC-5 SLC 500	Source Operand	
			Source Mask	
			Destination	
NEG	Negate	PLC-5 SLC 500	Source	
			Destination	
NEQ	Not Equal to	PLC-5 SLC 500	Source A	
			Source B	

Instruction	Name	Processor	Parameter	Considerations
NOP	No Operation	PLC-5	na	
NOT	Logical NOT	PLC-5 SLC 500	Source	
			Destination	
NSE	SFC Next Selection Branch	PLC-5	na	Ignores as part of SFC section.
NSI	SFC Next Simultaneous Branch	PLC-5	na	Ignores as part of SFC section.
NXB	Next Branch	PLC-5 SLC 500	na	Converts to a comma (,).
NXT	Next	PLC-5	Label	Does not convert the label number. You must modify the converted FOR instruction. See Converting FOR/NXT/BRK instructions on page 63.
ONS	One Shot	PLC-5	Source Bit	
OR	Logical OR	PLC-5 SLC 500	Source A	
			Source B	
			Destination	
OSF	One Shot Falling	PLC-5	Storage Bit	
			Output Bit	Combines output bit and output word.
			Output Word	
OSR	One Shot Rising	PLC-5 SLC 500	Storage Bit	If SLC 500 instruction, converts to an ONS instruction.
			Output Bit	Combines output bit and output word.
			Output Word	
OTE	Output Energize	PLC-5 SLC 500	Destination Bit	
OTL	Output Latch	PLC-5 SLC 500	Destination Bit	
OTU	Output Unlatch	PLC-5 SLC 500	Destination Bit	
PID	PID	PLC-5 SLC 500	Control Block	Verify the converted PID configuration parameters.
			PV Value	
			Tieback Value	
			CV Value	
RAC	HSC Reset Accumulator	SLC 500	Counter	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
			Source	
RAD	Degrees to Radians	PLC-5 SLC 500	Source	
			Destination	
REF	SFC Reference	PLC-5	na	Ignores as part of SFC section.
REF	I/O Refresh	SLC 500	Channel 0	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
			Channel 1	
RES	Reset	PLC-5 SLC 500	File Reference	
RET	Return	PLC-5 SLC 500	Return Parameters	

Instruction	Name	Processor	Parameter	Considerations
RHC	Read High Speed Clock	SLC 500	na	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
RMP	Ramp	SLC 500	na	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
RPC	Read Program Checksum	SLC 500	na	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
RPI	Reset Pending Interrupt	SLC 500	Slots	Converts, but Logix Designer application does not support this instruction.
RTO	Retentive On	PLC-5 SLC 500		
			Time Base	Converts time base to 1 millisecond.
			Preset	Replaces with "?." You must modify the converted RTO instruction.
			Accum	Replaces with "?." You must modify the converted RTO instruction.
SBR	Subroutine	PLC-5 SLC 500	Input Parameters	
SCL	Scale	SLC 500	Source	Logix Designer does not support this instruction; however, it is converted to a CPT instruction.
			Rate	
			Offset	
			Destination	
SCP	Scale with Parameters	SLC 500	Input	Logix Designer does not support this instruction; however, it is converted to a CPT instruction.
			Input Minimum	
			Input Maximum	
			Scaled Minimum	
			Scaled Maximum	
			Scaled Output	
SDS	Smart Directed Sequencer	PLC-5	na	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction. The SDSC routine adds control capability by using the SDS instruction to provide state machine control for sections of the machine. See Converting CAR routines on page 62.
SEL	SFC Selection Branch	PLC-5	na	Ignores as part of SFC section.
SFR	SFC Reset	PLC-5	File Number	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
			Restart at Step	
SIM	SFC Simultaneous Branch	PLC-5	na	Ignores as part of SFC section.
SIN	Sine	PLC-5 SLC 500	Source	
			Destination	
SOC	SFC Start of Compression	PLC-5	na	Ignores as part of SFC section.
SOP	SFC Start of Program	PLC-5	na	Ignores as part of SFC section.
SOR	Start of Rung	PLC-5 SLC 500	na	Starts output on a new line.
SQC	Sequencer Compare	PLC-5	na	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.

Instruction	Name	Processor	Parameter	Considerations
SQI	Sequencer Input	PLC-5 SLC 500	File	
			Mask	
			Source	
			Control File	Does not convert S:24 for indexing.
			Length	
			Position	
SQL	Sequencer Load	PLC-5 SLC 500	File	
			Source	
			Control File	Does not convert S:24 for indexing.
			Length	
			Position	
SQO	Sequencer Output	PLC-5 SLC 500	File	Does not convert S:24 for indexing.
			Destination Mask	
			Destination	
			Control File	
			Length	
			Position	
SQR	Square Root	PLC-5 SLC 500	Source	
			Destination	
SRT	Sort	PLC-5	Sort File	Does not convert S:24 for indexing.
			Control File	Inserts 0 for dimension to vary.
			Length	
			Position	
STD	Standard Deviation	PLC-5	File	
			Destination	Inserts 0 for dimension to vary.
			Control File	
			Length	
			Position	
STD	Selectable Timed Interrupt Disable	SLC 500	na	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
STE	Selectable Timed Interrupt Enable	SLC 500	na	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
STP	SFC Step	PLC-5	na	Ignored as part of SFC section.
STS	Selectable Timed Interrupt Start	SLC 500	File	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
			Time	
SUB	Subtract	PLC-5 SLC 500	Source A	
			Source B	
			Destination	
SUS	Suspend	SLC 500	Suspend ID	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
SVC	Service Communications	SLC 500	Channel 0	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
			Channel 1	
SWP	Swap	SLC 500	Source	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.

Instruction	Name	Processor	Parameter	Considerations
			Length	
TAN	Tangent	PLC-5 SLC 500	Source	
			Destination	
TDF	Compute Time Difference	SLC 500	na	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction.
TND	Temporary End	PLC-5 SLC 500	na	
TOD	To BCD	PLC-5 SLC 500	Source	
			Destination	
TOF	Off Delay	PLC-5 SLC 500		
			Time Base	Converts time base to 1 millisecond.
			Preset	Replaces with "?." You must modify the converted RTO instruction.
			Accum	Replaces with "?." You must modify the converted RTO instruction.
TON	On Delay	PLC-5 SLC 500		
			Time Base	Converts time base to 1 millisecond.
			Preset	Replaces with "?." You must modify the converted RTO instruction.
			Accum	Replaces with "?." You must modify the converted RTO instruction.
TRC	SFC Transition	PLC-5	na	Ignores as part of SFC section.
UID	User Interrupt Disable	PLC-5	na	
UIE	User Interrupt Enable	PLC-5	na	
UIF	User Interrrupt Flush	SLC 500	na	There is no Logix equivalent. Logs message directly in the rung along with the PCE instruction. Also, the Project Migrator does not support Micrologix.
XIC	Examine On	PLC-5 SLC 500	Source Bit	
XIO	Examine Off	PLC-5 SLC 500	Source Bit	
XOR	Exclusive OR	PLC-5 SLC 500	Source A	
			Source B	
			Destination	
XPY	X to the Power of Y	PLC-5 SLC 500	Source A	
			Source B	
			Destination	

Converting CAR routines

The Project Migrator does not convert CAR routines. A PCE instruction is generated for each CAR related instructions encountered. The CAR routines are as follows:

- AGA3, AGA7 and API - Use the CIR and COR instructions
- DDMC - Uses the DFA instruction
- SDSC - Uses the SDS instruction

Converting FOR/NXT/BRK instructions

The structure of FOR/NXT/BRK statements has changed in the Logix architecture. In the PLC-5 processor, the FOR and NXT instruction enclosed a section of code that was to be iterated multiple times, while the BRK instruction allowed a way to break out of the repeating code. In the RSLogix architecture, the FOR instruction calls a given routine a specific number of times, so a NXT instruction is not needed. The BRK instruction works in a similar fashion as in the PLC-5 processor.

Because this architecture change is significant, you will probably have to consider restructuring your logic.

Programming Conversion Errors (PCE) Messages

Introduction

Below is a list of all of the messages that are generated with a PCE instruction. The text is appended to the rung comments that have the PCE instruction. The message text begins with asterisks (*) and the words **Generated by Translation Tool**, and ends with asterisks.

PCE Messages

The table that follows lists the message identifiers, descriptions, and when they are logged:

ID	Text	When logged
101	The address references a counter's Update Accum (UA) bit field. This is not supported in the Logix Designer application.	Each time a reference to a counter's UA field is encountered (SLC only).
102	The address references a counter's Overflow(OV) or Underflow(UN) field. This has been converted but the conversion needs to be validated.	Each time a reference to a counter's OV or UN field is encountered.
103	**Warning**: Status files do not exist in Logix Designer software. GSV instructions are used in Logix Designer software to obtain controller information where applicable. This conversion must be validated.	Each time a reference to the S file is encountered.
105	The address references an indirect file number. It was not converted.	Each time an address reference with an indirect file number is encountered.
107	The address reference may have an incorrect index. The conversion needs to be validated.	Each time suitable index into the array could not be determined.
108	The BTR, BTW or MSG instruction has been converted. However, the conversion needs to be validated. These instructions have many parameters that cannot be directly converted and require review.	Each time a BTR, BTW or MSG instruction is converted.
109	PLC-5 and SLC s use 0.01 second and 1 second timebases. Logix Designer software uses a 0.001 second time base. The address references a counter's Accumulator (ACC) field. The conversion needs to be validated.	Each time a reference to a counter's ACC field was encountered.
110	PLC-5 and SLC s use 0.01 second and 1 second timebases. Logix Designer software uses a 0.001 second time base. The address references a counter's Preset (PRE) field. The conversion needs to be validated.	Each time a reference to a counter's PRE field was encountered.
113	Follow the <FBC or DDT> instruction with MOV and FAL instruction on parallel branches to ensure the correct bits are being operated on.	Each FBC and DDT instruction.
114	Although the PID instruction has been converted, the PID instruction has many parameters that do not convert directly to Logix Designer software. The conversion must be verified.	Each time a PID instruction is converted.
115	16-bit parameters have been extended to 32-bit. Ensure bit manipulation is correct.	Each time BSL, BSR, BTD instruction is converted.

ID	Text	When logged
116	The structure of FOR/NXT/BRK statements has changed in the Logix architecture. In the PLC-5 processor, the FOR and NXT instruction enclosed a section of code that was to be iterated multiple times, while the BRK instruction allowed a way to break out of the repeating code. In the RSLogix architecture, the FOR instruction calls a given routine a specific number of times, so a NXT instruction is not needed. The BRK instruction works in a similar fashion as in the PLC-5 processor. Because this architecture change is significant, you will probably have to consider restructuring your logic.	Each time FOR/NXT/BRK instructions are encountered.
117	AGA instruction not supported.	Each time a AGA instruction is found.
119	CIR/COR not supported.	Each time a CIR or CIO instruction is found.
120	Source and destination types differ.	When source and destination types differ in a COP instruction.
121	DFA instruction not supported	Each time a DFA instruction is found.
122	ERI/ERO instruction not supported.	Each time a ERI or ERO instruction is found.
123	IDI/IDO instruction not supported.	Each time a IDI or IDO instruction is found.
124	IIN/IOT instruction not supported.	Each time a IIN or IOT instruction is found.
128	SFC routines aren't migrated.	Each time a SFR or EOT instruction is found.
129	Online edit instructions are not supported.	Each time a SDS, SIZ or SRZ instruction is found.
130	User Interrupt instructions not supported.	Each time a UID, UIE or UIF instruction is found.
131	DDV instruction not supported.	Each time a DDV instruction is found.
132	High Speed Counter instructions not supported.	Each time a HSC/HSD/HSE/ SL or RHC/RAC/TDF instruction is found.
133	I/O Interrupt Enable/Disable instructions not supported.	Each time a IID or IIE instruction is found.
134	IIM/IOM instruction not supported.	Each time a IIM or IOM instruction is found.
135	INT instruction not supported.	Each time a INT instruction is found.
136	REF instruction not supported.	Each time a REF instruction (in SLC) is found.
137	RPI instruction not supported.	Each time a RPI instruction is found.
138	Selectable Timed Interrupt instructions not supported.	Each time a STD/STE or STS instruction is found.
139	SUS instruction not supported.	Each time a SUS instruction is found.
141	RMP instruction not supported.	Each time a RMP instruction is found.
142	RPC instruction not supported.	Each time a RPC instruction is found.
143	SVC instruction not supported.	Each time a SVC instruction is found.
144	SWP instruction not supported.	Each time a SWP instruction is found.
145	SQC instruction not supported.	Each time a SQC instruction is found.
146	INV instruction not supported.	Each time a INV instruction is found.
147	DCD/ENC instruction not supported.	Each time a DCD or ENC instruction is found.
148	The CEM, DEM, or EEM instruction has been converted. However, the conversion needs to be validated. These instructions have many parameters that cannot be directly converted and require review.	Each time a CEM, DEM or EEM instruction is found.
149	Modbus messaging is not supported in Logix Designer software.	If MSG instruction is configured for Modbus.
150	MSG instruction and associated MESSAGE tag need to be manually verified.	Each time a MSG instruction is found.
151	**Warning**: Status files do not exist in Logix Designer software. However this status file value is handled through the StatusFile routine.	S file type indexes that can be directly converted to functionality in Logix Designer software.
152	Logix Designer software has a different fault handling mechanism than the PLC-5/SLC. This fault routine will not be called.	Start of identified legacy processor fault routine.
153	This PII/DII routine is not used by Logix Designer software.	Start of identified legacy processor PII/DII routine.

Index

S

T

U

V

Rockwell Automation support

Use these resources to access support information.

Technical Support Center	Find help with how-to videos, FAQs, chat, user forums, and product notification updates.	rok.auto/support
Knowledgebase	Access Knowledgebase articles.	rok.auto/knowledgebase
Local Technical Support Phone Numbers	Locate the telephone number for your country.	rok.auto/phonesupport
Literature Library	Find installation instructions, manuals, brochures, and technical data publications.	rok.auto/literature
Product Compatibility and Download Center (PCDC)	Get help determining how products interact, check features and capabilities, and find associated firmware.	rok.auto/pcdc

Documentation feedback

Your comments help us serve your documentation needs better. If you have any suggestions on how to improve our content, complete the form at rok.auto/docfeedback.

Waste Electrical and Electronic Equipment (WEEE)

At the end of life, this equipment should be collected separately from any unsorted municipal waste.

Rockwell Automation maintains current product environmental information on its website at rok.auto/pec.

Rockwell Otomayson Ticaret A.Ş. Kar Plaza İş Merkezi E Blok Kat:6 34752, İçerenkÖy, İstanbul, Tel: +90 (216) 5698400 EEE YÖnetmeliğine Uygundur

Connect with us.

rockwellautomation.com — expanding **human possibility**™

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 NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, 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 1756-RM085E-EN-P - September 2020
Supersedes Publication 1756-RM085D-EN-P - November 2018