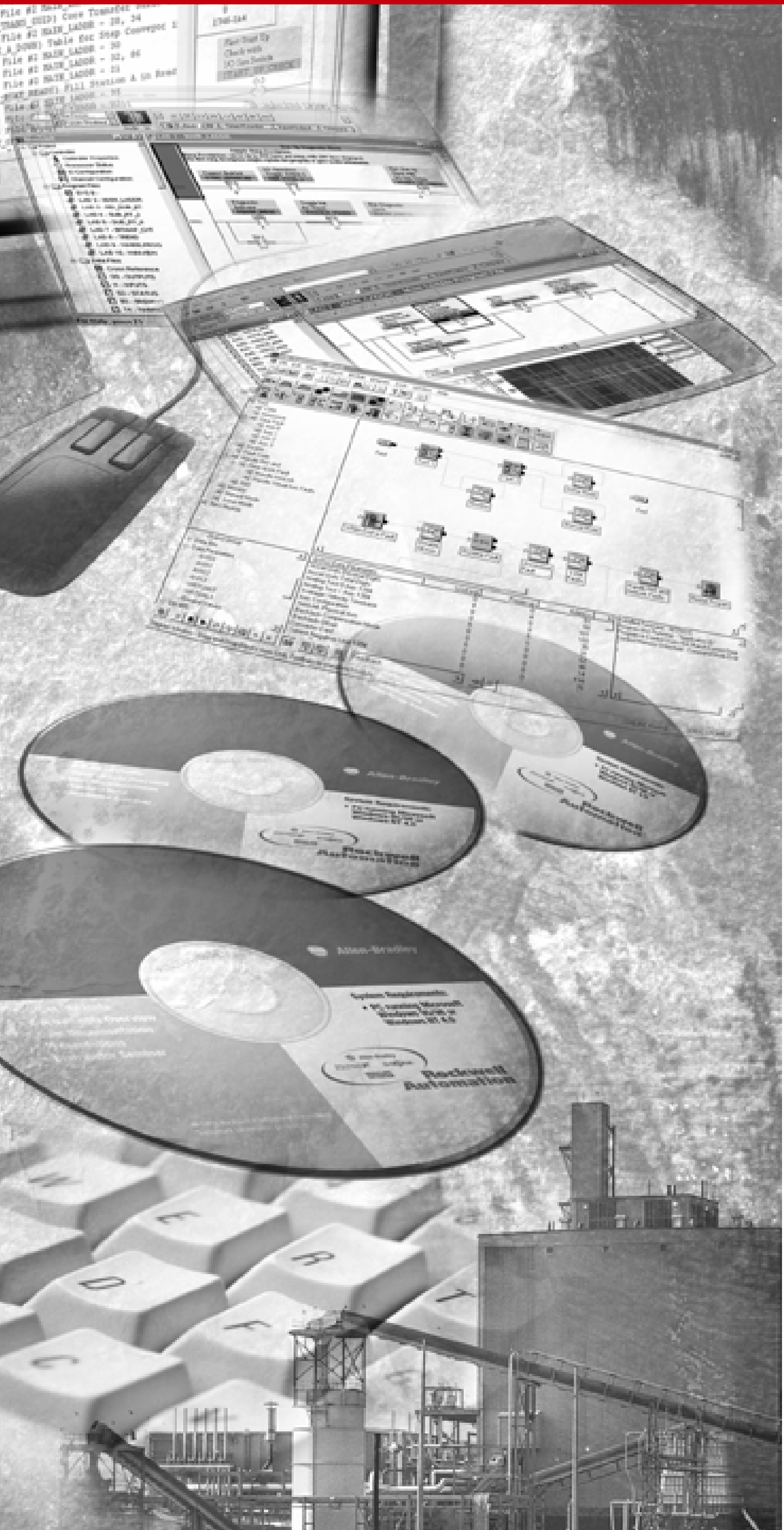


Allen-Bradley

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

1756 ControlLogix Controllers
1769 CompactLogix Controllers
1789 SoftLogix Controllers
1794 FlexLogix Controllers
PowerFlex 700S DriveLogix Controllers

Reference Manual

Rockwell Automation

Important User Information

Because of the variety of uses for the products described in this publication, those responsible for the application and use of this control equipment must satisfy themselves that all necessary steps have been taken to assure that each application and use meets all performance and safety requirements, including any applicable laws, regulations, codes and standards.

The illustrations, charts, sample programs and layout examples shown in this guide are intended solely for purposes of example. Since there are many variables and requirements associated with any particular installation, Allen-Bradley does not assume responsibility or liability (to include intellectual property liability) for actual use based upon the examples shown in this publication.

Allen-Bradley publication SGI-1.1, *Safety Guidelines for the Application, Installation and Maintenance of Solid-State Control* (available from your local Allen-Bradley office), describes some important differences between solid-state equipment and electromechanical devices that should be taken into consideration when applying products such as those described in this publication.

Throughout this manual we use notes to make you aware of safety considerations:

ATTENTION	Identifies information about practices or circumstances that can lead to personal injury or death, property damage or economic loss

Attention statements help you to:

- identify a hazard
- avoid a hazard
- recognize the consequences

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

Summary of Changes

This document describes how to use the version of the translation tool that is included with RSLogix 5000 programming software, version 10.0.

Changes to this document include:

- When you select the source file to convert, the dialog box now includes a drop down menu where you select the type of Logix controller for the converted file.
- While RSLogix 5000 software now supports ASCII instructions and a string data type, the translation tool does not convert ASCII instructions to the Logix equivalents. You still have to rework any converted ASCII instructions. See page 3-15 for more information on how to rework PLC-5/SLC 500 ST data table files.
- These changes were made to the translation tool:

Issue:	**Resolution:**
Empty strings and/or non-supported symbols in address comment statements do not convert.	No comment is generated for an empty string in an address comment. Non-supported symbols in address comments (such as / : - \ #] [< > + = . ~ @ $ ^ & * | ; ? ( )) are converted to underscore (_) symbols. The tool replaces consecutive underscore symbols with one underscore symbol.
IEEE format infinity or not-a-number values do not convert. Infinity appears as 1.#INF0000e 000. Not-a-number appears as 1.#NAN0000e 000.	Infinity constants are translated to "1.$" Not-a-number constants are translated to "1.#QNAN"

The generated .L5K file does not have an N: in front of each rung statement.	The translation tool now puts N: in front of all rung statements. The N: prefix identifies the rung as Normal. For example
	PROGRAM Continuous (Main:=mcpMain)
	    ROUTINE mcpMain ()
	        N: JSR(_2_LADDER, 0)
	    END_ROUTINE
	
	    ROUTINE _2_LADDER ()
	        N: XIC(B3[0].0) OTE(B3[0].1);
	        N: XIC(B3[0].0) OTE(B3[0].2);
	    END_ROUTINE
	END_PROGRAM

If there is no PROJECT statement in the PLC-5 or SLC file, the translation tool puts a blank project name in the controller statement. The generated .L5K file does not import correctly with a blank name.	If there is a PROJECT statement in the import file, then the stated name is used as the project name in the CONTROLLER statement. If there is no PROJECT statement in the import file, then the translation tool uses the name of the generated .L5K file as the project name.

Notes:

Chapter 1
Converting a PLC-5 or SLC 500 Program into a Logix Project
Introduction 1-1
Comparing PLC-5 and SLC 500 architecture to Logix architecture 1-2
Exporting a PLC-5 or SLC 500 Program. 1-4
Using RSLogix5 or RSLogix500 programming software . . 1-5
Using 6200 series programming software 1-7
Using A.I. series programming software 1-8
Converting a PLC-5 or SLC 500 Program. 1-9
View the Conversion Results 1-10
Viewing the log file 1-12
Import the ASCII Text File into a Logix Project 1-13
Rework PCE Instructions 1-14
Locating PCE instructions 1-15
Rework UNK Instructions 1-16
Configure the Controller and Chassis 1-16
Map I/O 1-17
Complete MSG Configuration 1-18
Other Considerations. 1-19

Chapter 2
Converting Program Structure
Introduction 2-1
Creating a Continuous Task. 2-2
Converting Selectable Timed Interrupts (STIs) 2-3
Converting Input Interrupts (DIIs/PIIs). 2-3

Chapter 3
Converting Data
Introduction 3-1
How PLC-5 and SLC 500 import/export files identify data table values 3-2
How Logix import/export files identify file types. 3-2
Converting Input (I) and Output (O) Data 3-3
Converting the Status (S) File Type 3-4
Converting the Binary (B) File Type. 3-5
Converting the Timer (T) File Type 3-5
Timer conversion rules. 3-7
Converting the Counter (C) File Type 3-7
Converting the Control (R) File Type 3-8
Converting the Integer (N) File Type 3-9
Converting the Floating Point (F) File Type 3-10
Converting the ASCII (A) File Type 3-10
Converting the Decimal (D) File Type 3-11
Converting the Block-Transfer (BT) File Type. 3-11
Block-transfer conversion rules. 3-12

Converting the M0 and M1 File Types 3-12
Converting the Message (MG) File Type. 3-13
Message conversion rules. 3-14
Converting the PID (PD) File Type 3-14
Converting the ASCII String (ST) File Type. 3-15
Converting the ControlNet (CT) File Type 3-16
Converting Constant Values. 3-17
Converting Indirect Addresses 3-17
Converting indirect addressing on the file number. 3-18
Converting Indexed Addresses. 3-20
Converting indexed addresses controlled by the processor status word S:24. 3-20
Converting indexed addresses that specify data in files (Logix arrays) 3-20
Converting Symbols 3-21
Converting Address Comments 3-22

Chapter 4
Converting Instructions
Introduction 4-1
Conversion Rules 4-1
Instruction List 4-2
Converting CAR instructions 4-16
Converting FOR/NXT/BRK instructions 4-16

Appendix A
Conversion Messages
Introduction A-1
Status Messages. A-1
Information Messages A-2
Question Messages A-3

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

Introduction

RSLogix 5000 programming software includes a translation tool that 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 translation tool that comes with release 8.0 of RSLogix 5000 programming software.

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

.

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

The translation tool 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 between rules of precedence, indexed addressing, I/O addressing, etc. The log file captures these differences.

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

The entire conversion process involves:

Conversion step:	**See page:**
Export the PLC-5 or SLC 500 program to an ASCII text file	1-4
Use the translation tool to convert the logic	1-9
View the conversion results	1-10
Import the ASCII text file into a Logix project	1-13
Rework PCE instructions	1-14
Rework UNK instructions	1-16
Configure the controller and chassis	1-16
Map I/O	1-17
Complete MSG configuration	1-18
Check other considerations	1-19

Comparing PLC-5 and SLC 500 architecture to Logix architecture

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

Some of the most significant differences in architecture are:

Architectural issue:	**Comparison:**
CPU	The PLC-5 and SLC 500 processor is based on 16-bit operations. Logix controllers use 32-bit operations. The translation tool 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 translation tool converts the legacy program types into appropriate Logix tasks. The PLC-5 and SLC 500 processors use an S data file to store processor status. A Logix controller stores data differently. Instead of accessing different locations within a file, you use Get System Value (GSV) and Set System Value (SSV) instructions to specify the status information you want. This is a significant difference that will require rework once the converted logic is imported into a Logix controller. The PLC-5 and SLC 500 processors also use bits in S:0 for the arithmetic status flags. For example, S:0/03 stores sign status. A Logix controller uses keywords to reference these flags. For example, instead of referencing a bit address to monitor a sign operation, you use the keyword S:N.
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 translation tool converts the PLC-5 and SLC 500 data table files into Logix arrays.
timers	The PLC-5 and SLC 500 timers 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 translation tools converts the legacy timers as they best fit into the Logix architecture. Converted timers 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 MSG instructions. The Logix controllers support MSG instructions. The translation tool converts the legacy BTR, BTW, and MSG instructions into Logix MSG instructions. Any CIO instructions are not converted. Once you import the converted logic, you must configure the MSG instructions so that they work properly and rework any CIO instructions.

The rest of this manual describes the specifics of how these architectural issues are converted.

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

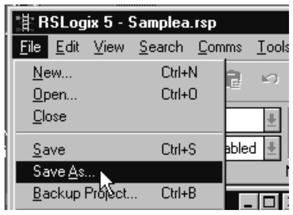
How you export the program to an ASCII text file depends on the programming software you use.

If you use:	See page:
RSLogix5 or RSLogix500 programming software	1-5
6200 series programming software	1-7
A.I. series programming software	1-8

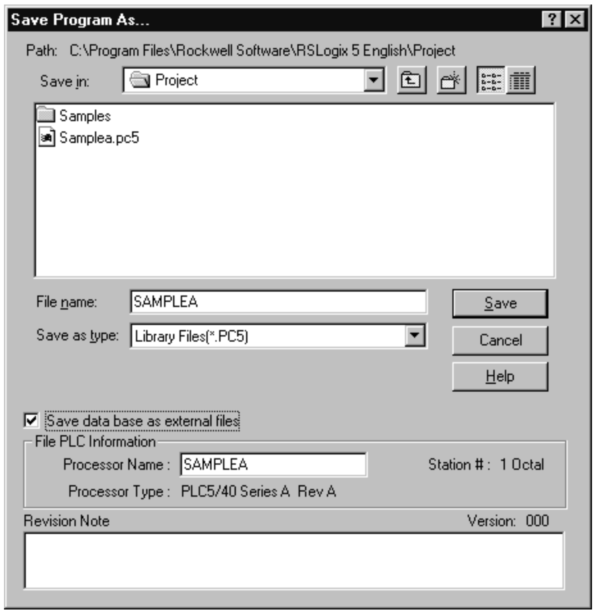
Using RSLogix5 or RSLogix500 programming software

Create a .PC5 or .SLC file for the program file:

1. **Select File → Save As.**



2. **Select the program to export.**



Path: C:\Program Files\Rockwell Software\RSLogix 5 English\Project

File name: SAMPLEA

Save as type: Library Files(*.PC5)

☑ Save data base as external files

File PLC Information

Processor Name : SAMPLEA Station # : 1 Octal

Processor Type : PLC5/40 Series A Rev A

Revision Note Version: 000

Select the program to export.

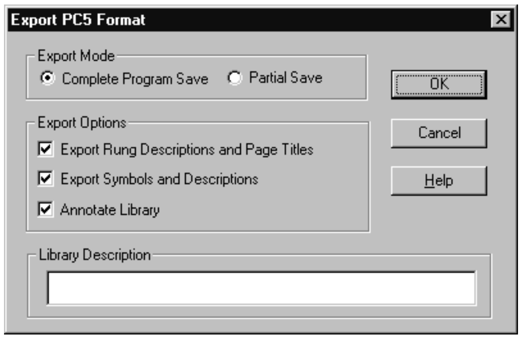
By default, the software points to the \Project folder for the destination. You can enter a different destination directory.

You must select the Library Files (.PC5 or .SLC) format.

Select this option so that comments and symbols are included in the export.

Click Save.

3. **Select export options.**



Export Mode

(●) Complete Program Save ( ) Partial Save

Export Options

☑ Export Rung Descriptions and Page Titles

☑ Export Symbols and Descriptions

☑ Annotate Library

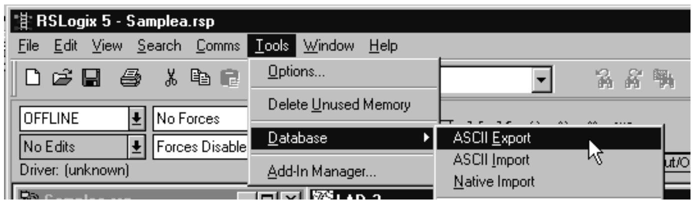
Select Complete Program Save.

Select all these options.

Click OK.

Create a .TXT file for comments and symbols:

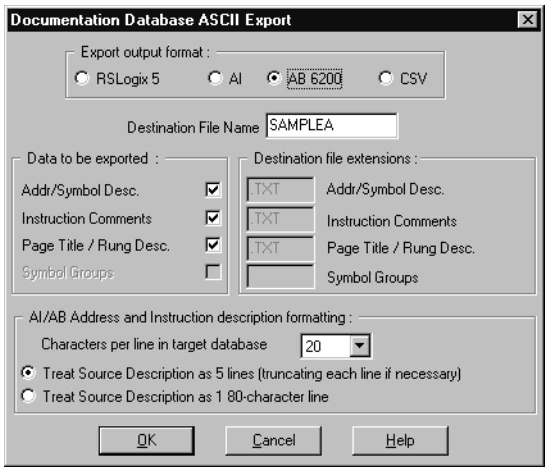
1. **Select Tools → Database → ASCII Export.**



2. **Select where to export comments and symbols.**

Select the AB 6200 format.

Click OK.


Documentation Database ASCII Export

Export output format : ( ) RSLogix 5 ( ) AI [✓ AB 6200] ( ) CSV

Destination File Name SAMPLEA

Data to be exported :		Destination file extensions :	
Addr/Symbol Desc.	✓	.TXT	Addr/Symbol Desc.
Instruction Comments	✓	.TXT	Instruction Comments
Page Title / Rung Desc.	✓	.TXT	Page Title / Rung Desc.
Symbol Groups			Symbol Groups

AI/AB Address and Instruction description formatting :

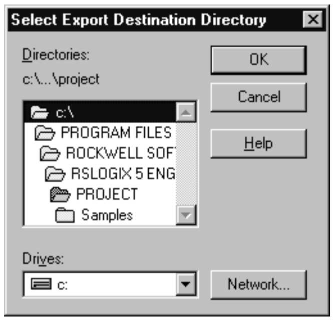
Characters per line in target database 20

[✓ Treat Source Description as 5 lines (truncating each line if necessary)]

( ) Treat Source Description as 1 80-character line

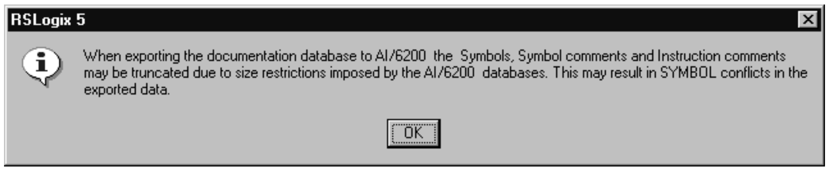
3. **Select the directory where the .PC5 or .SLC file is.**

Click OK.



4. **Accept the warning about comments and symbols.**

Click OK.


RSLogix 5

When exporting the documentation database to AI/6200 the Symbols, Symbol comments and Instruction comments may be truncated due to size restrictions imposed by the AI/6200 databases. This may result in SYMBOL conflicts in the exported data.

RSLogix5 programming software stores PLC-5 programs using .RSP file extensions. RSLogix500 programming software stores SLC 500 programs using .RSS file extensions.

Using 6200 series programming software

To export a program and its symbols using 6200 series programming software:

1. Place the program files in \IPDS\ARCH\PLC5
2. Start 6200 series programming software.
3. Select **F7**:File Utils → **F7**:Export → **F1**:Processor Memory File Only.
4. Cursor to the program to export.
5. Select **F3**:Select Source → **F1**:Begin Operation.
6. When the export process is complete, press any key to continue.
7. Rename the log file in \IPDS\ARCH\PLC5 because the next step will overwrite the file.
8. Select **F7**:File Utils → **F7**:Export → **F3**:Comments and Symbols.
9. Use the cursors to select the program to export.
10. Select **F3**:Select Source → **F1**:Begin Operation.
11. When the export process is complete, press any key to continue.
12. Copy or move the .PC5/.SLC and .TXT files to where the Logix translation tool will find them.

6200 software uses these file extensions for program files: .AC$, .AF5, .B0$, .B1$, .D1$, .IX$, .LX$, .OP$, .P1$, and .PC$.

Using A.I. series programming software

To export a program and its symbols using A.I. series programming software:

1. Start A.I.5 series programming software.

2. Select **F1**:Select Program/PLC-5 Address.

3. Cursor to the program to export and press **Enter**.

4. Select **F5**:Utility Options → **F1**:Rebuild Damaged Data Base → **F1**:Rebuild current program → **F1**:Yes – Force rebuilding of Index files.

5. When the rebuild process is complete, press any key to continue.

6. Select **F4**:Export data base → **F4**:6200 ASCII.

7. Enter a name without an extension for the exported database file.

8. When the database export is complete, press any key to continue.

9. Press **Esc** to return to the main menu.

10. Select **F2**:Offline Programming → **F3**:Edit → **F2**:Block → **F1**:Block Start → **F2**:Copy Block (it does not matter what is actually selected)→ **F8**:Save Block.

11. Select **F1**:ASCII and enter a name without an extension for the exported program.

12. Select **F2**:No Rung Descriptions. The translation tool uses comments from the .TXT file, not the .PC5/.SLC file.

13. Select **F3**:Entire Program → **F4**:No Annotation → **F5**: No→ **F7**:Export. You do not need the “short description.”

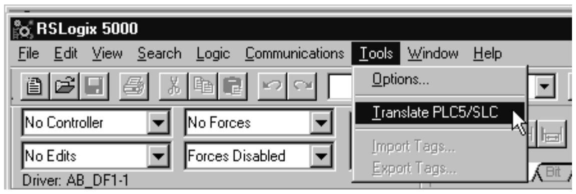
A.I.5 software uses these file extensions for program files: .ADR, .CEI, .CET, .CFG, .DSC, .IO2, .IO4, .PRF, .RCK, .RPD, .RPI, .SYM, .X5, .XRF, and .XRI.

Converting a PLC-5 or SLC 500 Program

After you have the ASCII text file of the PLC-5 or SLC 500 program file, you can convert the logic to its Logix equivalent.

Use RSLogix5000 programming software:

1. **Select Tools → Translate PLC5/SLC.**



2. **Select the text file to convert.**

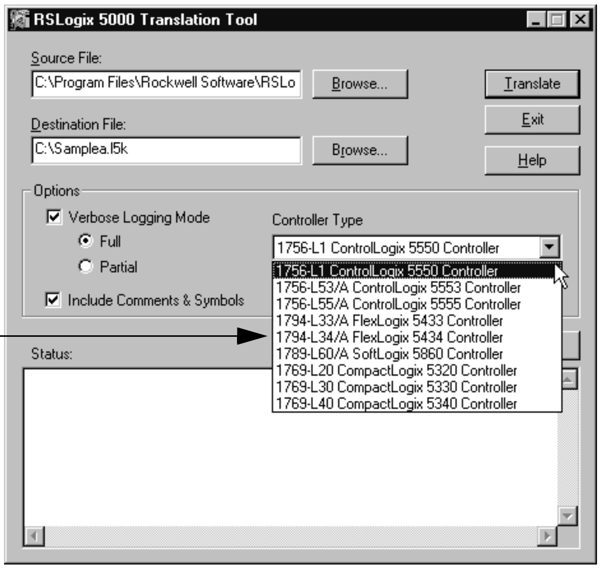
Select the file to convert.
It must have a .PC5 or .SLC extension

By default, the software points to the \RSLogix5000\Project folder for the destination. You can enter a different destination directory.

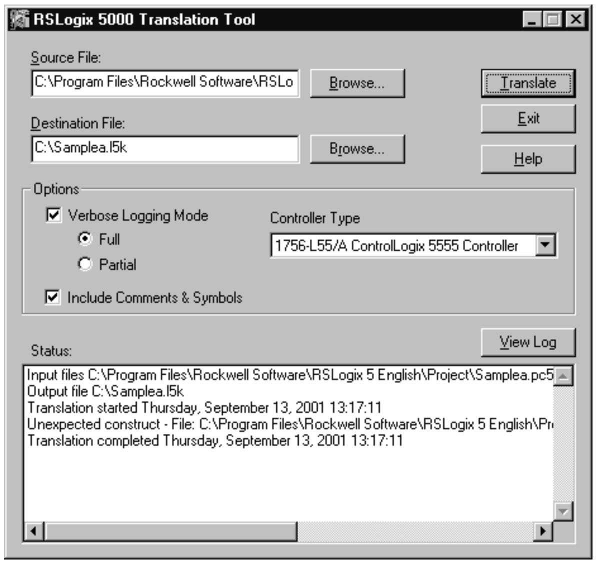
Note: Both the .PC5/.SLC and .TXT files must be in the same directory for the conversion to work.

Select conversion options.

Select the controller type.


RSLogix 5000 Translation Tool
Source File:
C:\Program Files\Rockwell Software\RSLo
Browse...
Translate
Exit
Destination File:
C:\Samplea.l5k
Browse...
Help
Options
Verbose Logging Mode
Full
Partial
Include Comments & Symbols
Controller Type
1756-L1 ControlLogix 5550 Controller
1756-L1 ControlLogix 5550 Controller
1756-L53/A ControlLogix 5553 Controller
1756-L55/A ControlLogix 5555 Controller
1794-L33/A FlexLogix 5433 Controller
1794-L34/A FlexLogix 5434 Controller
1789-L60/A SoftLogix 5860 Controller
1769-L20 CompactLogix 5320 Controller
1769-L30 CompactLogix 5330 Controller
1769-L40 CompactLogix 5340 Controller
Status:

Click Translate.



```
Input files C:\Program Files\Rockwell Software\RSLogix 5 English\Project\Samplea.pc5
Output file C:\Samplea.l5k
Translation started Thursday, September 13, 2001 13:17:11
Unexpected construct - File: C:\Program Files\Rockwell Software\RSLogix 5 English\Pr
Translation completed Thursday, September 13, 2001 13:17:11
```

This box displays the status of the conversion process.

Click View Log to see the log file.

You can select from these conversion options:

Option:	Description:
Verbose logging mode	Check this option to write all messages of all categories to the log file. Otherwise, only a subset of the status messages and all of the question messages will be written to the log file. Select Full to cause all messages of all categories to be written to the log file, in addition to extra, descriptive text. Select Partial to cause all messages of all categories to be written to the log file, without the extra, descriptive text.
Include comments and symbols	Check this option to specify whether documentation is to be included in the conversion. By default, this option is enabled, causing the ASCII comment file .TXT to be processed along with the processor program file. **Important:** The .TXT file must exist for the conversion process to operate when this option is selected. **Important:** Both the .PC5/.SLC and .TXT files must be in the same directory.

View the Conversion Results

The translation tool creates a complete import/export file (.L5K extension) that you can then import into a Logix project. For more information on the contents of a Logix import/export file, see the *Logix5000 Controller Import/Export Reference Manual*, publication 1756-RM084.

After the conversion process, the import/export file follows this format:

```
CONTROLLER <Controller Name>

TAG
      (* All tags, aliases, and associated descriptions are
      placed here. *)
END_TAG

PROGRAM Continuous ( MAIN := mcpMain )

      ROUTINE mcpMain
            JSR <Routine Name>;
            %% More JSR calls could appear dependent upon
            processor type.
      END_ROUTINE

      %% A routine is created for each ladder program that
      executes.

      ROUTINE <Routine Name>
            (* A translated legacy ladder program *)
      END_ROUTINE

END_PROGRAM
```

```
PROGRAM Sti ( MAIN := <Routine Name> )

        ROUTINE <Routine Name>
                (* A translated legacy ladder program *)
        END ROUTINE

        %% A routine is created for each ladder program that
        executes.
END PROGRAM

TASK Continuous ( MODE := CONTINUOUS, WATCHDOG := 500 )
        Continuous;
END_TASK

TASK Sti ( MODE := PERIODIC, RATE := <Rate>, WATCHDOG := 500 )
        Sti;
END_TASK

END_CONTROLLER
```

The components of the converted import/export file are:

Component:	Description:
CONTROLLER	The conversion process creates one CONTROLLER structure. The controller name is based on the PROJECT statement in the PLC-5 import/export file. If the controller name is the same as any other instruction or keyword in the PLC-5 import/export file, the conversion process appends _DUP to the controller name.
TAG	The conversion process creates one, controller-scoped TAG structure. All tags and aliases are placed in this global TAG structure
PROGRAM Continuous	The conversion process creates one PROGRAM named Continuos. This program holds all the routines.
ROUTINE mcpMAIN	The conversion process creates one ROUTINE names mcpMAIN. This routine contains the JSR instructions for one or more ROUTINES that are considered the main routines. The main routines are determined from PLC-5 processor status data that identifies the main control programs.
ROUTINE	The conversion process creates a ROUTINE for each PLC-5 program file. It is possible that JSR calls or processor status information may specify that the same ROUTINE is required by multiple PROGRAMS. If this occurs, the conversion process creates duplicate ROUTINES, one for each PROGRAM that needs the ROUTINE.
PROGRAM Sti	The conversion creates this program for the STI logic, if is exists, for the PLC-5 processor.
TASK Continuous	The conversion process creates one TASK to specify how to execute the programs. This TASK is always continuous and references the Continuous PROGRAM.
TASK Sti	The conversion creates this task to execute the STI logic. This is a periodic task that references the STI PROGRAM.

Viewing the log file

Each conversion process generates an ASCII-based log file. This log file provides a summary of the conversion process, and contains formatted messages describing the actions and steps taken during the conversion process. The number and type of messages depend on the options you selected for the conversion process. The messages are written to the log file in the order in which their related translation actions occur.

IMPORTANT The log file identifies areas that should be examined for potential problems.

You can open the log file from within the translation tool by pressing the View Log button. You can also use any standard windows text editor to open the log file. The name of the log file is the same as the name of the output file, but with a .LOG extension.

The line numbers referenced in the log file correspond to the line numbers in the exported PLC-5 or SLC 500 program file. Having an editor that displays line numbers is helpful if you have to refer back to the exported PLC-5 or SLC 500 program file.

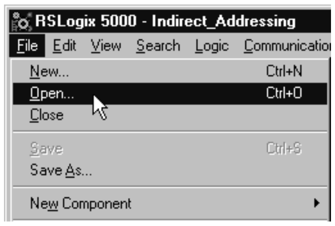
For more information about the messages that can appear in the log file, see appendix A.

Import the ASCII Text File into a Logix Project

The output file from the conversion process is a Logix import/export file with a .L5K extension. Import this file into a Logix project using RSLogix5000 programming software.

Use RSLogix5000 programming software:

1. **Select File → Open.**



2. **Select the text file.**

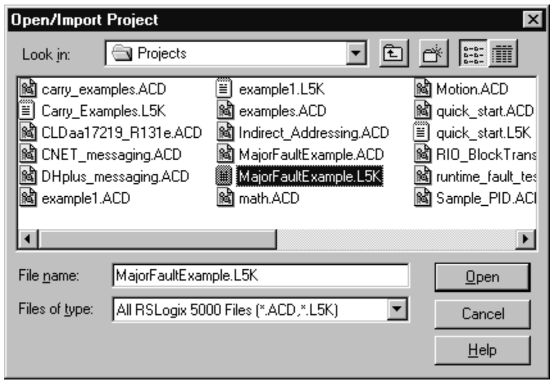
The text file should have a .L5K extension.

Select the file to import.

By default, the software points to the \RSLogix5000\Project folder. You can change the default via Tool → Options.

Specify the name for the file to import.

Click Open.


Open/Import Project

Look in: Projects

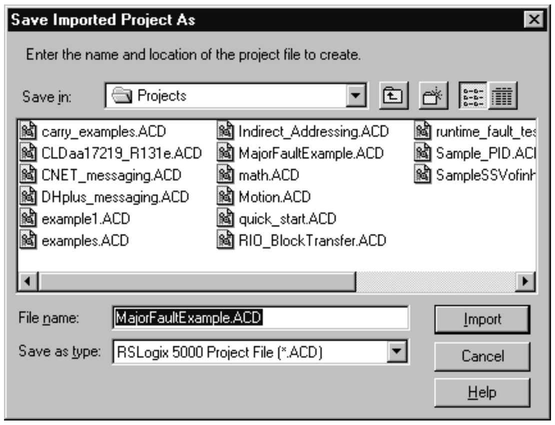
File name: MajorFaultExample.L5K

Files of type: All RSLogix 5000 Files (*.ACD,*.L5K)

3. **Specify the name and location of the project**

Specify the project location.

Specify the project name.


Save Imported Project As

Enter the name and location of the project file to create.

Save in: Projects

File name: MajorFaultExample.ACD

Save as type: RSLogix 5000 Project File (*.ACD)

Click Import.

Rework PCE Instructions

The conversion process inserts a PCE (Possible Conversion Error) instruction to identify possible errors. The PCE instruction follows this format (in the ASCII text file):

```
PCE(<Message>, <PCETag>)
```

Where:

Parameter:	Description:
`Message`	identifies the type of error or warning that occurred. See appendix A for a list of possible conversion messages.
`PCETag`	identifies the error Each conversion error receives a unique PCETag The output import/export file and the log file both have the PCE instruction. You can search on either file using the PCETag to find the related information.

For example:

A rung in the converted import/export file would look like:

```
PCE( “3000”, “pce00001” ),  OTE( B3[0].0 );
```

The corresponding rung in the log file would look like:

```
pce00001
QUES:3000  356:1024  MyProg:MyFirstRoutine:10
Output File reference is not valid
```

Where:

Value:	Corresponds to:
`3000`	PCETag 3000
`356`	line 356 in the original, ASCII PLC-5 or SLC 500 file
`1024`	line 1024 in the converted, ASCII Logix file
`MyProg`	program in the imported Logix project
`MyFirstRoutine`	routine in the imported Logix project
`10`	rung number in the imported Logix project

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

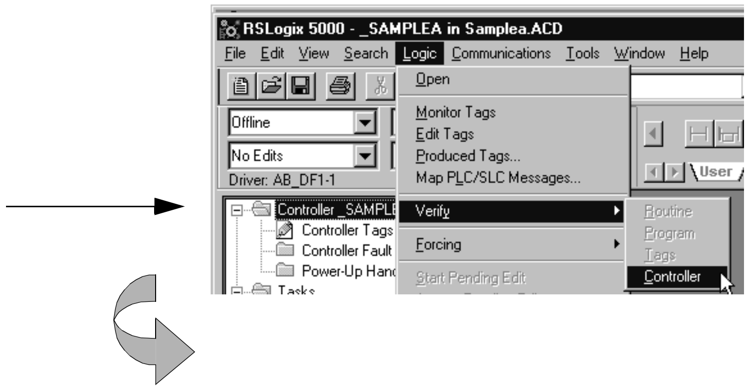
PCE instructions can highlight these possible errors:

A PCE instruction can mean:	How to correct the error:
Instruction cannot be converted	Delete the PCE instruction. Rewrite the logic to achieve the desired functionality.
Status word S:24	In a PLC-5 processor, this status word contains the index offset for indexed addressing. This word does not exist in a Logix controller. The translation tool inserts a PCE instruction for each occurrence of S:24. For example, in a COP instruction, there will be two PCE instructions, one for the source and one for the destination. Be sure that you account for how S:24 is used and then delete the PCE instruction. Similarly, all file instructions will have a "+S24" added to the source and destination words. Again, account for how S:24 is used in the instruction and then delete the "+S24."
Battery low	Delete the PCE instruction. Use a GSV instruction to get this status information.
Math overflow	Delete the PCE instruction. Use the S:V keyword in a bit instruction.

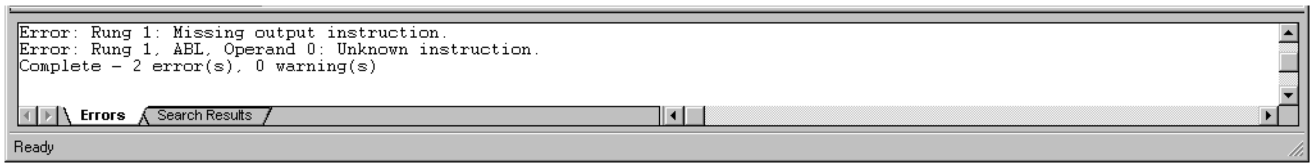
Locating PCE instructions

You can locate all of the PCE instructions by verifying the logic.

1. **Select Logic → Verify.**



The bottom of the screen displays the results:



```
Error: Rung 1: Missing output instruction.
Error: Rung 1, ABL, Operand 0: Unknown instruction.
Complete - 2 error(s), 0 warning(s)
```

Double-click on an error to go directly to the rung.

Rework UNK Instructions

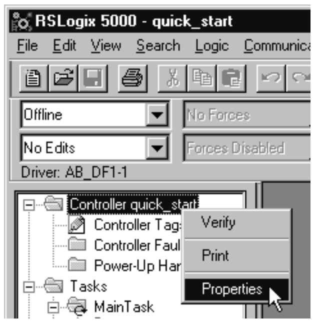
The translation tool converts some PLC-5 and SLC 500 instructions that have no equivalent in the Logix architecture. Once you import these instructions into a Logix project, they appear as UNK instructions. You must delete each UNK instructions and replace it with the appropriate, corrected logic.

You can also verify the logic to locate UNK instructions, as shown above for locating PCE instructions.

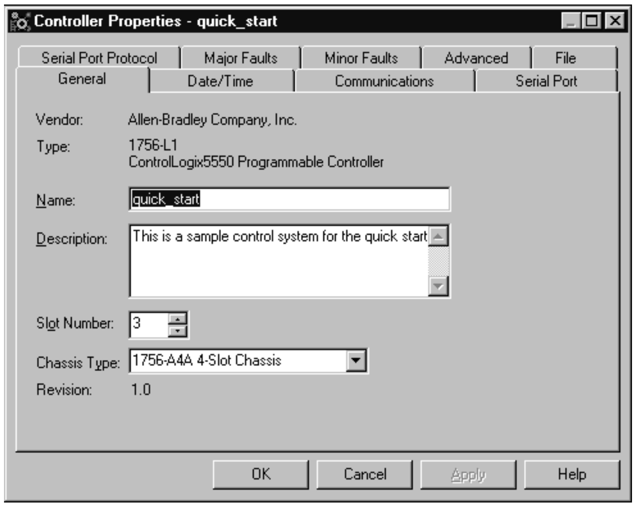
Configure the Controller and Chassis

Use the Controller Properties dialog to assign the chassis size and slot number of the controller.

1. **Place the cursor over the Controller folder.**
2. **Click the right mouse button and select Properties.**



3. **Configure the controller.**



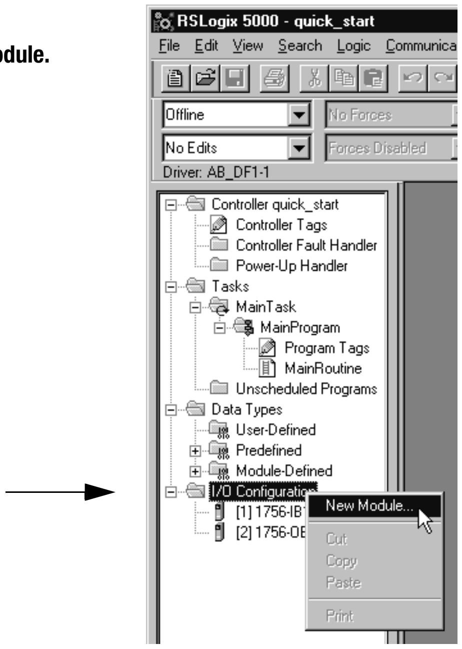
Vendor: Allen-Bradley Company, Inc.
Type: 1756-L1
ControlLogix5550 Programmable Controller
Name: quick_start
Description: This is a sample control system for the quick start
Slot Number: 3
Chassis Type: 1756-A4A 4-Slot Chassis
Revision: 1.0

Specify the slot number of the controller.

Specify the chassis size.

Then use the Controller Organizer to specify the I/O modules and other devices for the controller.

1. **Select I/O Configuration.**
2. **Click the right mouse button and select New Module.**



Map I/O

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

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

The tags created for physical I/O (e.g. I.2) are empty at the end of the conversion process. You must use the programming software 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 COP instructions to map the Local:1.Data word into the I2 tag created by the conversion process.

A 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 which 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 COP 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:

COP

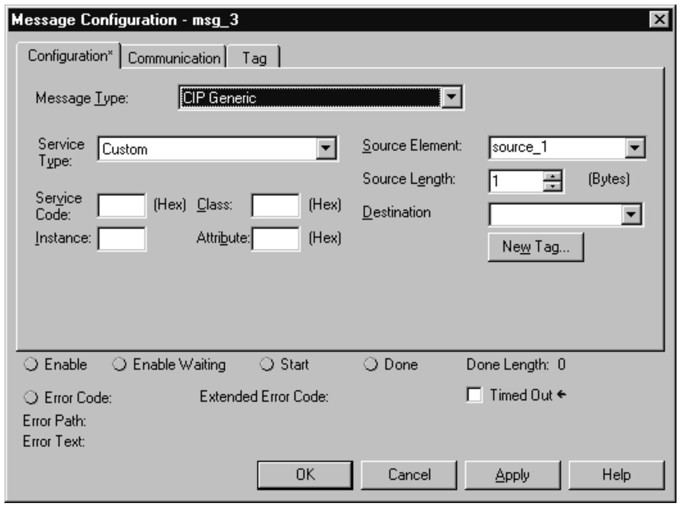
Source	Local:1:I.Data[0]
Destination	I[0]
Length	8

If you use a 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.

For more information about how the translation tool converts the PLC-5 or SLC 500 data table, see the next chapter.

Complete MSG Configuration

The translation tool only partially converts MSG instructions. You must use the RSLogix 5000 programming software to configure each MSG instruction by completing the information on the Communications tab.


Message Configuration - msg_3

Configuration* | Communication | Tag

Message Type: CIP Generic

Service Type: Custom

Service Code: (Hex) Class: (Hex)

Instance: Attribute: (Hex)

Source Element: source_1

Source Length: 1 (Bytes)

Destination

New Tag...

Enable | Enable Waiting | Start | Done | Done Length: 0

Error Code: Extended Error Code: | Timed Out

Error Path:

Error Text:

OK | Cancel | Apply | Help

For more information about configuring MSG instructions, see the *Logix5000 Instruction Set Reference Manual*, publication 1756-RM003. This manual is available in PDF format on the RSLogix 5000 programming software CDROM.

Other Considerations

The following are additional issues to keep in mind:

- The time base for timer instructions is fixed at 1 msec for a Logix controller. The conversion process scales PLC-5 and SLC 500 timer presets and accumulators accordingly. For example, a PLC-5 timer 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.

- RSLogix 5000 programming software does not support programmable input interrupts (DIIs/PIIs). A DII/PII program in converted as a program in the continuous task.

- 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.

  Aliases take up memory in a Logix controller so you might want to delete those aliases you do not plan to use. Use the RSLogix 5000 programming software to delete aliases after you import the project.

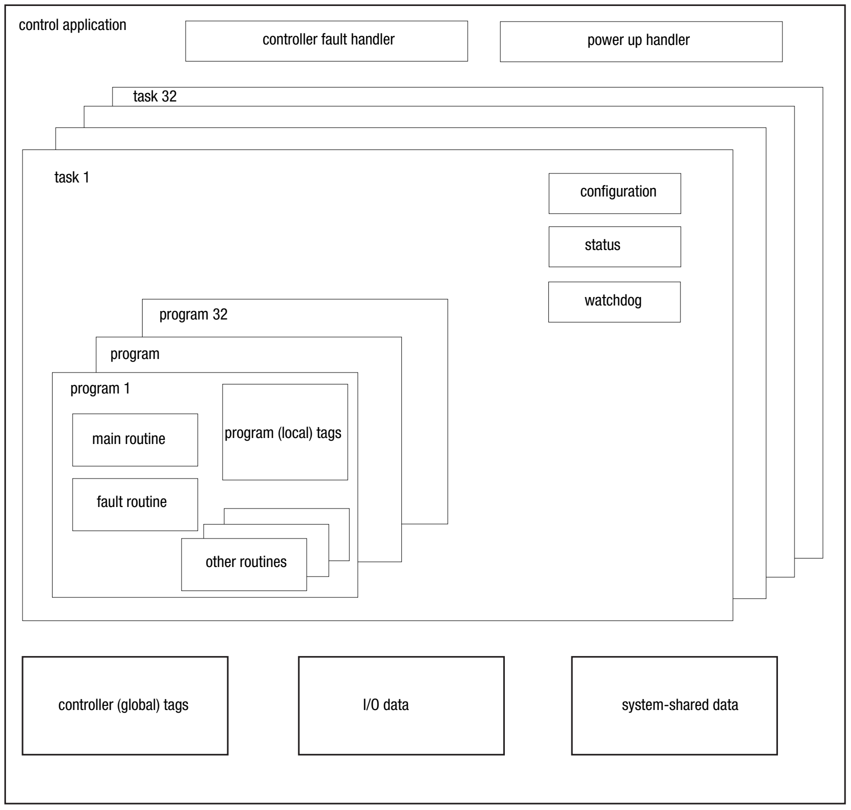
Notes:

Converting Program Structure

Introduction

A Logix controller uses a different execution model than either the PLC-5 processor or the SLC 500 processor. A Logix controller uses:

- tasks to configure controller execution
- programs to group data and logic
- routines to encapsulate executable code written in a single programming language



40012

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

A task provides scheduling and priority information for a set of one or more programs that execute based on specific criteria. You can configure tasks as either continuous or periodic.

Task Type:	Number Supported by a Logix Controller:
continuous	1
periodic	31 if there is a continuous task 32 if there is no continuous task

A task can have as many as 32 separate programs, each with its own executable 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 grouped. Programs can only appear once in the controller organizer and cannot be shared by multiple tasks. The scheduled programs within a task execute to completion from first to last.

A routine is a set of logic instructions in a single programming language, such as ladder logic. Routines provide the executable code for the project in a controller. A routine is similar to a program file or subroutine in a PLC or SLC processor.

Each program has a main routine. This is the first routine to execute when the controller triggers the associated task and calls the associated program. Use logic, such as the JSR instruction, to call other routines.

You can also specify an optional program fault routine. The controller executes this routine if it encounters an instruction-execution fault within any of the routines in the associated program.

As the translation tool converts the PLC-5 or SLC 500 logic, consider these program structures:

Conversion step:	See page:
Creating a continuous task	2-2
Converting STIs	2-3
Converting DIIs and PIIs	2-3

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 translation tool automatically creates one continuous task named Continuous with a default watchdog setting of 500 msec. This task is the main task of the converted project.

Within this continuous task, the translation tool creates a main routine named mcpMain, which lists one or more routines. This main routine contains JSR instructions to each of the other routines.

- In PLC-5 processors, the first master control program (MCP) becomes the main routine. The translation tool uses the PLC-5 status file to determine which is the first MCP.
- In SLC 500 processors, ladder program 2, which is the main ladder program, becomes the main routine.

Each remaining routine within a program is considered a subroutine. Subroutines are local in scope to the program (i.e., they are accessible by the main routine and other subroutines within the current program only). Because of this, it is possible for ladder programs to appear as the same subroutine in multiple programs.

The conversion process does not define a fault routine for a program. You have to define the fault routine yourself, if needed.

Converting 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 translation tool converts this program file into the main routine of a periodic task named Sti.

The translation tool retrieves the STI interval from the processor status file. If necessary, the translation tool 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 translation tool 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 translation tool converts it to a routine in the Continuous task. You must edit the Logix logic to call the converted routine.

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

Notes:

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).

PLC-5 or SLC file type:	Logix array type:	Radix:	Remarks:	See page:
O	INT	BINARY		3-3
I	INT	BINARY		3-3
S	INT	HEX		3-4
B	DINT	BINARY	The 16-bit value is copied into the 32-bit location and sign-extended.	3-5
T	TIMER			3-5
C	COUNTER		A PCE instruction is generated when overflow (.OV) and underflow (.UN) bit fields are encountered.	3-7
R	CONTROL			3-8
N	DINT	DECIMAL	The 16-bit value is copied into the 32-bit location and sign-extended.	3-9
F	REAL			3-10
A	INT	HEX		3-10
D	DINT	HEX	The 16-bit value is copied into the 32-bit location and zero-filled.	3-11
BT	MESSAGE			3-11
M0	INT			3-12
M1	INT			3-12
MG	MESSAGE			3-13
PD	PID			3-14
ST	na	na	This file type is not converted; the data table is ignored.	3-15
CT	na	na	This file type is not supported; the data table is ignored.	3-16

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

The PLC-5 and SLC 500 import/export files use DATA statements to identify file types:

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

Where:

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: N7:0 10 N7:1 11 N7:2 12

How Logix import/export 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	INT[3] := {1,2,3}	an integer array with three elements
T4 with 2 elements	TIMER[2] := {{1,2,3}, {4,5,6}}	an array of two timer structures; each timer 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:

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_07
	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_07
	O:010	O[10]	O_010
	I:021/05	I[21].05	I_21_Bit05
	O:035/15	O[35].15	O_35_Bit015

Converting the Status (S) File Type

The conversion process creates a single-dimension, INT array for the S file. The tag name is S. 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 S:127 0X0000 0X0000	tag S type INT[128] (Radix := Hex) := {16#0000, ... }

Here are 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

IMPORTANT Each S address generates a PCE instruction.

There are special considerations for some data in the status file:

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 the Binary (B) File Type

A B file is translated 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 and BSR 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, ... }

Here are some 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. Timers 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 TIMER 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 TIMER structure, which consists of three, 32-bit DINT words. Here is a comparison of the PLC-5/SLC 500 timer and the Logix timer:

Word:	**PLC-5/SLC 500 bits:**	**Logix bits:**	**Mnemonic:**	**Description:**
0	15	31	EN	enable
0	14	30	TT	timer 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 TIMER[2] := {16#E0000000, 1000, 123000} The .PRE and .ACC values were converted from a 1 second time base.

Here are 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

IMPORTANT Each address that references a .PRE or .ACC value generates a PCE instruction. The time base from a PLC-5 or SLC 500 instruction might change when converted to a Logix instruction, so it's important to examine all direct references to these parameters to make sure the logic still executes properly.

Timer 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 timer is used for converting the preset and accumulator values each time that timer appears. If the timer appears multiple times, but with different time bases, the conversion process places a PCE instruction on every occurrence of that timer.
- Each logic reference to a .PRE or .ACC value is replaced with 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. Here is 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
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.

Here are some 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. Here 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
0	8	24	FD	found
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.

Here are some 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 Integer (N) File Type

The conversion process creates a single-dimension, INT array for the N file. The tag name is N*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 N7:99 153 227	tag N7 type INT[100] (Radix := Decimal) := {153, 227, ... }

Here are some 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, ... }

Here is 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}

Here are 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}

Here is an example D address and its Logix equivalents:

Original Address:	Converted Address:
D10:0	D10[0]

Converting the Block-Transfer (BT) File Type

The conversion process creates an individual MESSAGE structure for each element in the BT 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).

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

For example, in the ASCII text file:

This DATA statement:	Converts to:
DATA BT9:1	tag A_MSG_TEMPBUF2 type INT[1] (Radix:=HEX, Description:=”Temporary Buffer Tag created during conversion”) := [0] tag MG9_1 type MESSAGE (DF1DHFLAG :=0, ProduceCount := 0 LocalTag := A_MSG_TEMPBUF2, RequestedLength := 10, MessageType := Block Transfer Read);

The conversion process creates a temporary buffer into or out of which the actual Logix MSG instruction operates. Then the conversion process uses an FAL instruction to copy the data to/from the actual Local tag.

Here are some BT addresses and their Logix equivalents:

Original Address:	Converted Address:
BT11:5	MG11_5
BT11:5.RLEN	MG11_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 DF1DHFlag is always set to 0.
- 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, ...]

Here are 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. However, only the local message information is converted, which consists of the message type, the message itself, and the message length. After the conversion, you must use the programming software to configure the message.

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). Here is a comparison of the PLC-5/SLC 500 MG structure and the Logix 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 MSG instruction:	Converts to:
MG9:0 PLC-5 MSG message typePLC-2 unprotected read local data table addressN7:0 size in elements1 port1A target address10 target node2 local	tag A_MSG_TEMPBUF1 type INT[1] (Radix:=HEX, Description:="Temporary Buffer Tag created during conversion") := [0] tag MG9_0 type MESSAGE (DF1DHFLAG :=0, LocalTag := A_MSG_TEMPBUF1, RequestedLength := 1, MessageType := PLC2 Unprotected Read);

The conversion process creates a temporary buffer into or out of which the actual Logix MSG instruction operates. Then the conversion process uses an FAL instruction to copy the data to/from the actual Local tag. Here are the lines in an .L5K file that are generated by a simple PLC-5 MSG instruction, using the above declarations:

```
XIO(MG9_0.EN) RES(R_MSG_CTL1) FAL(R_MSG_CTL1, 10, 0, ALL,
A_MSG_TEMPBUF1[0 + R_MSG_CTL1.POS], N7[10 +
R_MSG_CTL1.POS]) MSG(MG9_0);
```

Here is the line from the .PC5 file that generated the above:

```
SOR XIO MG9:0/EN MSG MG9:0 EOR
```

Here are 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 DF1DHFlag is always set to 0.

Converting the PID (PD) File Type

A PD file is converted to a PID type tag. Any PID instruction that uses an N control file is not fully converted. In this case, the N file is converted along with the PID instruction, but the instruction will fail during program verification. You have to use the programming software to configure the control information.

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	tag PD10
256 0 0 0 0 0	type PID10[11].1 := {536870912, 0, 0, 0, 0, 0, 0,
0 0 0 0 0 0	0, 0, 0, 0, 0, 0.1, 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 15 10 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	
...	

Here are 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

Converting the ASCII String (ST) File Type

The ASCII data type is supported in a Logix controller, but the conversion process does not convert any ST files. The conversion process replaces any instructions that reference an ST data type with a PCE instruction. You will have to rework the PCE instructions into the appropriate Logix ASCII instructions.

You can create Logix tags to represent the ST data tables using the predefine STRING data structure in a Logix project. For example, if you have a ST10 data table file with 100 elements, create ST10[100] of type STRING in the converted Logix project. Then, convert any data from the ST data table as:

Reference:	PLC-5 / SLC 500 Data Table Address:	Logix Tag:
full string (used by string instructions)	ST10:0	ST10[0]
string length field	ST10:1.LEN	ST10[1].Len
string data field	ST10:2.DATA[5]	ST[10[2].DATA[5]

When you rework the converted logic, these PLC-5 and SLC 500 instructions correspond to these Logix instructions:

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
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 ControlNet data type is not supported in a Logix controller. The conversion process does not convert any CT files. The conversion process replaces any instructions that reference an CT data type with a PCE instruction.

Converting Constant Values

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

For example:

This 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
BCD	&D0049	16#0031	convert to hex constant
Octal	&O61	8#61	replace &O 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 conversion tool supports indirect addresses, except when the indirection is an array specification. Indirect array specifications are converted to aliases.

For example:

Type:	PLC-5/SLC 500 example:	Conversion:	Conversion rule:
File number	N[N7:0]:5	na	The conversion tool cannot convert an indirect file number. A PCE instruction is generated.
Word or element number	N12:[N7:0]	N12[N7_0]	N7:0 translates to array tag N7[0]. Alias N7_0 replaces the indirect address.
	N12:[T4:1.PRE]	N12[T4_1_PRE]	T4:1.PRE translates 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 5 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 translated 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.

Converting indexed 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 and 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]

Converting indexed addresses that specify data in files (Logix arrays)

Indexed addresses are also used with the file instructions to operate on files of data. These instruction 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)

Converting Symbols

The conversion process converts a symbol to a description.

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

```
SYM <address_reference> <literal>
```

Where:

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.

For example:

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 : TIMER[2]	SYM T4:0 Ducky SYM T4:1 2ndDuck	T4 : TIMER[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.

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 translation tool converts individual instructions. Areas to consider are identified where appropriate.

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

Conversion Rules

When converting instructions, the conversion tool follows these rules:

- PLC-5 and SLC 500 parameters use 16 bits. They will be extended to 32 bits for Logix parameters.
- Constants are converted to binary format.
- 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.
- Any constant that represents a serial port is always converted to 0, the Logix serial port.
- If data type mixing occurs between integer and real types, log a type conversion message to the conversion log file and insert a PCE instruction in the output import/export file.
- Log messages are inserted for all converted expressions to encourage the user to verify that operator precedence was correctly converted.

Instruction List

The following table lists the PLC-5 and SLC 500 instructions and includes comments where appropriate to identify conversion issues.

Instruction:	Name:	Processor:	Parameter:	Considerations, if any:
ABL	ASCII Test Buffer for Line	PLC-5 SLC 500	Channel	The translation tool does not convert ASCII instructions. When you import the converted file, the instruction appears as a PCE instruction and you must rework the logic. Channel is set to zero.
			Control	
			Characters	
ABS	Absolute Value	SLC 500	Source	The translation tool does not convert ASCII instructions. When you import the converted file, the instruction appears as a PCE instruction and you must rework the logic.
			Destination	
ACB	ASCII Number of Characters in Buffer	PLC-5	Channel	The conversion tool does not convert ASCII instructions. When you import the converted file, the instruction appears as a PCE instruction and you must rework the logic. Channel is set to zero.
			Control	
			Characters	
ACI	ASCII String to Integer	PLC-5 SLC 500	Source	The conversion tool does not convert ASCII instructions. When you import the converted file, the instruction appears as a PCE instruction and you must rework the logic.
			Destination	
ACL	ASCII Clear Buffer	SLC 500	Channel	The conversion tool does not convert ASCII instructions. When you import the converted file, the instruction appears as a PCE instruction and you must rework the logic. Channel is set to zero.
			Transmit Buffer	
			Receive Buffer	
ACN	ASCII String Concatenate	PLC-5 SLC 500	Source A	The conversion tool does not convert ASCII instructions. When you import the converted file, the instruction appears as a PCE instruction and you must rework the logic.
			Source B	
			Destination	
ACS	Arc Cosine	PLC-5 SLC 500	Source A	
			Destination	
ACT	SFC Action	PLC-5	na	Ignores as part of the SFC section.
ADD	Add	PLC-5 SLC 500	Source A	
			Source B	
			Destination	

Instruction:	Name:	Processor:	Parameter:	Considerations, if any:
AEX	ASCII String Extract	PLC-5 SLC 500	Source	The conversion tool does not convert ASCII instructions. When you import the converted file, the instruction appears as a PCE instruction and you must rework the logic.
			Index	
			Number	
			Destination	
AFI	Always False	PLC-5	na	
AGA	AGA Flow	PLC-5	na	There is no Logix equivalent. Logs a message to the log file and generates a PCE instruction.
AHL	ASCII Set/Reset Handshake Lines	PLC-5 SLC 500	Channel	The conversion tool does not convert ASCII instructions. When you import the converted file, the instruction appears as a PCE instruction and you must rework the logic. Channel is set to zero.
			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	The conversion tool does not convert ASCII instructions. When you import the converted file, the instruction appears as a PCE instruction and you must rework the logic.
			Destination	
AND	Logical AND	PLC-5 SLC 500	Source A	
			Source B	
			Destination	
ARD	ASCII Read Characters	PLC-5 SLC 500	Channel	The conversion tool does not convert ASCII instructions. When you import the converted file, the instruction appears as a PCE instruction and you must rework the logic.Channel is set to zero.
			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	The conversion tool does not convert ASCII instructions. When you import the converted file, the instruction appears as a PCE instruction and you must rework the logic. Channel is set to zero.
			Destination	Does not convert S:24 for indexing. Uses .POS value from Control.
			Control	
			String Length	
			Characters Read	

Instruction:	Name:	Processor:	Parameter:	Considerations, if any:
ASC	ASCII String Search	PLC-5 SLC 500	Source	The conversion tool does not convert ASCII instructions. When you import the converted file, the instruction appears as a PCE instruction and you must rework the logic.
			Index	
			Search	
			Result	
ASN	Arc Sine	PLC-5 SLC 500	Source	
			Destination	
ASR	ASCII String Compare	PLC-5 SLC 500	Source A	The conversion tool does not convert ASCII instructions. When you import the converted file, the instruction appears as a PCE instruction and you must rework the logic.
			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	The conversion tool does not convert ASCII instructions. When you import the converted file, the instruction appears as a PCE instruction and you must rework the logic. Channel is set to zero.
			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	The conversion tool does not convert ASCII instructions. When you import the converted file, the instruction appears as a PCE instruction and you must rework the logic. Channel is set to zero.
			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	

Instruction:	Name:	Processor:	Parameter:	Considerations, if any:
BSL	Bit Shift Left	PLC-5 SLC 500	File	Does not convert S:24 for indexing. Logs message to log file.
			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 to log file.
			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.
			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	Logs message to log file and generates PCE instruction.

Instruction:	Name:	Processor:	Parameter:	Considerations, if any:
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 to log file.
			Destination	Does not convert S:24 for indexing.
			Length	
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	Converts, but RSLogix 5000 software does not support this instruction. When you import the converted file, the instruction appears as a UNK instruction and you must rework the logic.
			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.
			Reference	Does not convert S:24 for indexing.
			Result	Does not convert S:24 for indexing.
			Compare Control	
			Length	
			Position	
			Result Control	
			Length	
			Position	
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 log file and generates a PCE instruction.
DIV	Divide	PLC-5 SLC 500	Source A	
			Source B	
			Destination	

Instruction:	Name:	Processor:	Parameter:	Considerations, if any:
DTR	Data Transition	PLC-5	Source	
			Mask	
			Reference	
ENC	Encode 1 of 16 to 4	SLC 500	Source	Converts, but RSLogix 5000 software does not support this instruction. When you import the converted file, the instruction appears as a UNK instruction and you must rework the logic.
			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	Converts, but RSLogix 5000 software does not support this instruction. When you import the converted file, the instruction appears as a UNK instruction and you must rework the logic.
ERO	Error on Output Instruction	PLC-5	na	Converts, but RSLogix 5000 software does not support this instruction. When you import the converted file, the instruction appears as a UNK instruction and you must rework the logic.
ESI	End of SFC Simultaneous Branch	PLC-5	na	Ignores as part of SFC section.
FAL	File Arithmetic	PLC-5	Control	
			Length	
			Position	
			Mode	
			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.

Instruction:	Name:	Processor:	Parameter:	Considerations, if any:
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	Does not convert the label number. You must modify the converted FOR instruction. See page 4-16.
			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.

Instruction:	Name:	Processor:	Parameter:	Considerations, if any:
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	
HSC	High Speed Counter	SLC 500	Counter	Converts, but RSLogix 5000 software does not support this instruction. When you import the converted file, the instruction appears as a UNK instruction and you must rework the logic.
			Preset	
HSD	HSC Interrupt Disable	SLC 500	Type	Converts, but RSLogix 5000 software does not support this instruction. When you import the converted file, the instruction appears as a UNK instruction and you must rework the logic.
			Counter	
			Preset	
			Accum	
HSE	HSC Interrupt Enable	SLC 500	Counter	Converts, but RSLogix 5000 software does not support this instruction. When you import the converted file, the instruction appears as a UNK instruction and you must rework the logic.
HSL	HSC Load	SLC 500	Counter	Converts, but RSLogix 5000 software does not support this instruction. When you import the converted file, the instruction appears as a UNK instruction and you must rework the logic.
			Source	
			Length	
IDI	Immediate Data Input	PLC-5	Data File Offset	There is no Logix equivalent. Logs a message to the log file and generates a PCE instruction.
			Length	
			Destination	
IDO	Immediate Data Output	PLC-5	Data File Offset	There is no Logix equivalent. Logs a message to the log file and generates a PCE instruction.
			Length	
			Destination	
IID	I/O Interrupt Disable	SLC 500	Slots	There is no Logix equivalent. Logs a message to the log file and generates a PCE instruction.
IIE	I/O Interrupt Enable	SLC 500	Slots	There is no Logix equivalent. Logs a message to the log file and generates a PCE instruction.
IIM	Immediate Input with Mask	SLC 500	Slot	There is no Logix equivalent. Logs a message to the log file and generates a PCE instruction.
			Mask	
			Length	
IIN	Immediate Input	PLC-5	RRG	
INT	I/O Interrupt	SLC 500	na	There is no Logix equivalent. Logs a message to the log file and generates a PCE instruction.
IOT	Immediate Output	PLC-5	RRG	

Instruction:	Name:	Processor:	Parameter:	Considerations, if any:
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. See page 4-16.
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 a message to the log file and generates a PCE instruction.

Instruction:	Name:	Processor:	Parameter:	Considerations, if any:
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	
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 page 4-16.
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	

Instruction:	Name:	Processor:	Parameter:	Considerations, if any:
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	Converts, but RSLogix 5000 software does not support this instruction. When you import the converted file, the instruction appears as a UNK instruction and you must rework the logic.
			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	Converts, but RSLogix 5000 software does not support this instruction. When you import the converted file, the instruction appears as a UNK instruction and you must rework the logic.
			Channel 1	
RES	Reset	PLC-5 SLC 500	File Reference	
RET	Return	PLC-5 SLC 500	Return Parameters	
RPI	Reset Pending Interrupt	SLC 500	Slots	Converts, but RSLogix 5000 software does not support this instruction. When you import the converted file, the instruction appears as a UNK instruction and you must rework the logic.
RTO	Retentive Timer On	PLC-5 SLC 500	Timer	
			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	Converts, but RSLogix 5000 software does not support this instruction. When you import the converted file, the instruction appears as a UNK instruction and you must rework the logic.
			Rate	
			Offset	
			Destination	

Instruction:	Name:	Processor:	Parameter:	Considerations, if any:
SCP	Scale with Parameters	SLC 500	Input	Converts, but RSLogix 5000 software does not support this instruction. When you import the converted file, the instruction appears as a UNK instruction and you must rework the logic.
			Input Minimum	
			Input Maximum	
			Scaled Minimum	
			Scaled Maximum	
			Scaled Output	
SDS	Smart Directed Sequencer	PLC-5	na	There is no Logix equivalent. Logs a message to the log file and generates a PCE instruction.
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 a message to the log file and generates a 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.
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	

Instruction:	Name:	Processor:	Parameter:	Considerations, if any:
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 a message to the log file and generates a PCE instruction.
STE	Selectable Timed Interrupt Enable	SLC 500	na	There is no Logix equivalent. Logs a message to the log file and generates a 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 a message to the log file and generates a PCE instruction.
			Time	
SUB	Subtract	PLC-5 SLC 500	Source A	
			Source B	
			Destination	
SUS	Suspend	SLC 500	Suspend ID	Converts, but RSLogix 5000 software does not support this instruction. When you import the converted file, the instruction appears as a UNK instruction and you must rework the logic.
SVC	Service Communications	SLC 500	Channel 0	Converts, but RSLogix 5000 software does not support this instruction. When you import the converted file, the instruction appears as a UNK instruction and you must rework the logic.
			Channel 1	
SWP	Swap	SLC 500	Source	Converts, but RSLogix 5000 software does not support this instruction. When you import the converted file, the instruction appears as a UNK instruction and you must rework the logic.
			Length	

Instruction:	Name:	Processor:	Parameter:	Considerations, if any:
TAN	Tangent	PLC-5 SLC 500	Source	
			Destination	
TND	Temporary End	PLC-5 SLC 500	na	
TOD	To BCD	PLC-5 SLC 500	Source	
			Destination	
TOF	Timer Off Delay	PLC-5 SLC 500	Timer	
			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	Timer On Delay	PLC-5 SLC 500	Timer	
			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	
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 instructions

The conversion tool does not convert CAR instructions. A PCE instruction is inserted in the output import/export file for each CAR instruction encountered. The CAR instructions include AGA (AGA flow), SDS (smart directed sequencer), and DFA (diagnostic fault annunciator) instructions.

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.

Conversion Messages

Introduction

The conversion process generates a log file that provides information as to how the conversion progressed. The conversion process generates:

Conversion message:	See page:
status message	A-1
information message	A-2
question message	A-3

Status Messages

The status messages record major events during the conversion process. Status messages are always written to the log file, regardless of the logging mode.

Status messages use this format:

`STAT:<code> <text>`

Where:

Parameter:	Description:
code	identifies the status message
text	describes the event

Code:	Text:	When logged:
200	Input files <*file_name*>.	Before any conversion activity has taken place.
201	Output file <*file_name*>.	Before any conversion activity has taken place.
202	Conversion started <*date and time*>.	Before any conversion activity has taken place.
203	Conversion completed <*date and time*>.	When all conversion activity is complete.
204	Unsupported file; terminating.	Before any conversion activity has taken place.
205	Edit control instruction encountered; terminating.	When an SDZ, SIZ, or SRZ instruction is encountered.
206	Failed to open <*file_name*>.	At the time that the legacy .TXT file is opened.
207	Failed to read file <*file_name*>.	At the time that the legacy .TXT file is read.
208	Failed to write to file <*file_name*>.	At the time that the Logix import/export file is created.
299	<*general_status*>.	Any other status messages not covered in the above list.

Information Messages

The information messages record details of the conversion process. These messages are only written to the log file if you select the verbose logging mode.

Information messages use this format:

```
INFO:<code>[<input_line>:<output_line>][<program>[:<routine>[:<rung>]]] <text>
```

Where:

Parameter:	**Description:**
code	identifies the information message
input_line	line number in the original, ASCII PLC-5 or SLC 500 file
output_line	line number in the converted, ASCII Logix file
program	program in the imported Logix project
routine	routine in the imported Logix project
rung	rung number in the imported Logix project
text	describes the message

Code:	**Text:**	**When logged:**
1	The IOA statement was ignored.	Each time an IOA statement is skipped in the legacy .TXT file.
2	The IOC statement was ignored.	Each time an IOC statement is skipped in the legacy .TXT file.
3	The IOS statement was ignored.	Each time an IOS statement is skipped in the legacy .TXT file.
4	The FCI statement was ignored.	Each time an FCI statement is skipped in the legacy .TXT file.
5	The FCN statement was ignored.	Each time an FCN statement is skipped in the legacy .TXT file.
6	The RACK statement was ignored.	Each time a RACK statement in skipped in the legacy .TXT file.
7	The SLOT statement was ignored.	Each time a SLOT statement is skipped in the legacy .TXT file.
8	The SEQUENTIAL FUNCTION CHART section was ignored.	Each time a SFC section is skipped in the legacy .TXT file.
9	The STRUCTURED TEXT section was ignored.	Each time a ST section is skipped in the legacy .TXT file.
10	The FORCE statement was ignored.	Each time a FORCE statement is skipped in the legacy .TXT file.
11	The INPUT FILTERS statement was ignored.	Each time a INPUT FILTERS statement is skipped in the legacy .TXT file.
12	The MULTI POINT statement was ignored.	Each time a MULTI POINT statement is skipped in the legacy .TXT file.
13	The PLC2 Compatibility statement was ignored.	Each time a PLC2 statement is skipped in the legacy .TXT file.
14	THE PLC5 Compatibility statement was ignored.	Each time a PLC5 statement is skipped in the legacy .TXT file.

Code:	Text:	When logged:
15	The CHANNEL CONFIGURATION statement was ignored.	Each time a CHANNEL CONFIGURATION statement is skipped in the legacy .TXT file.
16	The CONFIG statement was ignored.	Each time a CONFIG statement is skipped in the legacy .TXT file.
17	This DATA type is not supported; the data table was ignored.	Each time a DATA statement for an unsupported file type (ST, CT, or SC) is encountered.
18	The PROGRAM HEADER section was processed.	Each time the START statement is processed.
19	The SLOT statement was processed.	Each time the SLOT statement is processed.
20	The PROJECT section was processed.	Each time the PROJECT statement is processed.
21	The DATA section was processed.	Each time the DATA statement is processed.
22	The LADDER section was processed.	Each time the LADDER statement is processed.
23	The Source and Destination file type of the COP instruction do not match.	Each time a COP instruction with differing Source and Destination types is encountered.
24	A RAC instruction was encountered.	Each time an RAC instruction is encountered.
25	There are note sufficient initialization values in the data table.	Each time a DATA statement is encountered with initialization values that do no match the specified dimension.
26	The data table's dimension was increased to accommodate the initialization values encountered.	Each time a DATA statement is encountered with initialization values that exceed the specified dimension.
27	An address comment that was associated with a file was ignored.	Each time an address comment associated with a file is encountered.
28	The BT or MG data type has insufficient information, and was therefore ignored.	Each time a BT or MG type that is missing necessary attributes is encountered.
29	The ASCII type message was ignored.	Each time an ASCII type message is encountered.
30	The IO statement was ignored.	Each time an IO statement is encountered in the legacy .TXT file.
99	Conversion cancelled <*date and time*>.	Each time the user cancels the conversion by pressing the CANCEL button.

Question Messages

The question messages highlight items that might require further attention from the user. Question messages are always written to the log file, regardless of the logging mode.

Question messages use this format:

```
QUES:<code>[[file_type]<input_line>:<output_line>][<program>[:<routine>[:<rung>]]] <text>
```

Where:

Parameter:	Description:
code	identifies the information message
input_line	line number in the original, ASCII PLC-5 or SLC 500 file
output_line	line number in the converted, ASCII Logix file
program	program in the imported Logix project
routine	routine in the imported Logix project
rung	rung number in the imported Logix project
text	describes the message

Code:	Text:	When logged:
100	The address references an unsupported type. It was not converted.	Each time an address reference of type STRING, ControlNet, or SFC status is encountered.
101	The address references a counter's Update Accum (.UA) bit field. It was not converted.	Each time a reference to a counter's .UA field is encountered.
102	The address references a counter's Overflow (.OV) or Underflow (.UN) field. The conversion needs to be validated.	Each time a reference to a counter's .OV or .UN field is encountered.
103	Warning: The S file reference is not equivalent to the status file.	Each time a reference to the S file is encountered.
104	The instruction is not supported.	Each time an instruction not supported in RSLogix 5000 software is encountered.
105	The Selectable Timed Interrupt Disable (STD) instruction needs to be validated.	Each time an STD instruction is encountered.
106	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 sixth parameter of the converted FAL instruction might have an operator precedence error.	Each time an FAL instruction that has an expression is encountered.
108	The address reference might have an incorrect index. The conversion needs to be validated.	Each time an index into an array can not be determined.
109	The instruction has been converted, but it needs to be validated.	Each time a BTR, BTW, or MSG instruction is converted.
110	The address references a timer's accumulator (.ACC) field. The conversion needs to be validated.	Each time a reference to a timer's .ACC field is encountered.
111	The address references a timer's preset (.PRE) field. The conversion needs to be validated.	Each time a reference to a timer's .PRE field is encountered.

Numerics

6200 series programming software 1-7

A

A files 3-10
A.I. series programming software 1-8
address comments 3-22
addresses
- constant values 3-17
- indexed 3-20
- indirect 3-17

aliases
- address comments 3-22
- symbols 3-21

arrays 3-20
ASCII files 3-10

B

B files 3-5
binary files 3-5
block-transfers
- files 3-11
- instructions 4-5

BT files 3-11

C

C files 3-7
CAR instructions 4-16
chassis 1-16
completing MSG instructions 1-18
configuring controller and chassis 1-16
considerations 1-19
constant values 3-17
continuous tasks 2-2
control files 3-8
controller 1-16
ControlNet files 3-16
converting
- A files 3-10
- address comments 3-22
- B files 3-5
- block-transfer instructions 4-5
- BT files 3-11
- C files 3-7
- CAR instructions 4-16
- completing MSG instructions 1-18

converting, continued
- configure controller and chassis 1-16
- considerations 1-19
- constant values 3-17
- continuous tasks 2-2
- CT files 3-16
- D files 3-11
- data 3-1
- DIIs 2-3
- exporting PLC-5 or SLC 500 program 1-4
- F files 3-10
- files 3-20
- FOR/NXT/BRK instructions 4-16
- I files 3-3
- import converted file 1-13
- indexed addresses 3-20
- indirect addresses 3-17
- information messages A-2
- instructions 4-1
- log files 1-12
- M0/M1 files 3-12
- mapping I/O 1-17
- messages A-1
- MG files 3-13
- MSG instructions 4-11
- N files 3-9
- O files 3-3
- options 1-10
- overview 1-1
- PCE instructions 1-14
- PD files 3-14
- PIIs 2-3
- PLC-5 or SLC 500 program 1-9
- program structure 2-1
- question messages A-3
- R files 3-8
- results 1-10
- S files 3-4, 3-20
- ST files 3-15
- status messages A-1
- STIs 2-3
- symbols 3-21
- T files 3-5
- UNK instructions 1-16
- verifying 1-15

counter files 3-7
CT files 3-16

D

D files 3-11
data 3-1
decimal files 3-11
deleting
  PCE instructions 1-14
  UNK instructions 1-16
DIIs 2-3

E

execution model 2-1
exporting
  PLC-5 or SLC 500 program 1-4
  using 6200 software 1-7
  using A.I. software 1-8
  using RSLogix software 1-5

F

F files 3-10
file data 3-20
file formats
  6200 software 1-7
  A.I. software 1-8
  L5K 1-1
  PC5 1-1
  RSP 1-7
  RSS 1-7
  TXT 1-4
file types
  A (ASCII) 3-10
  B (binary) 3-5
  BT (block-transfer) 3-11
  C (counter) 3-7
  CT (ControlNet) 3-16
  D (decimal) 3-11
  F (floating point) 3-10
  I (input) 3-3
  M0/M1 (specialty) 3-12
  MG (message) 3-13
  N (integer) 3-9
  O (output) 3-3
  PD (PID) 3-14
  R (control) 3-8
  S (status) 3-4, 3-20
  ST (string) 3-15
  T (timer) 3-5
floating point files 3-10
FOR/NXT/BRK instructions 4-16

I

I files 3-3
importing 1-13
indexed addresses 3-20
indirect addresses 3-17
information messages A-2
input files 3-3
input interrupts
  See DIIs/PIIs 2-3
instructions 4-1
integer files 3-9

L

L5K extension 1-1
log files 1-12, A-1

M

M0/M1 files 3-12
mapping I/O 1-17
message files 3-13
MG files 3-13
MSG instructions 1-18, 4-11

N

N files 3-9

O

O files 3-3
output files 3-3

P

PC5 extension 1-1
PCE instructions 1-14
PD files 3-14
PID files 3-14
PIIs 2-3
program structure 2-2
  DIIs 2-3
  execution model 2-1
  overview 2-1
  PIIs 2-3
  programs 2-2
  routines 2-2
  STIs 2-3
  task 2-2
programs 2-2

Q

question messages A-3

R

R files 3-8
results
 file structure 1-10
 log files 1-12, A-1
routines 2-2
RSLogix programming software 1-5
RSP extension 1-7
RSS extension 1-7

S

S files 3-4, 3-20
selectable timed interrupts
 See STIs 2-3
ST files 3-15
status files 3-4, 3-20
status messages A-1
STIs 2-3
string files 3-15
symbols 3-21

T

T files 3-5
tasks 2-2
timer files 3-5
translating
 See converting 1-1
TXT extension 1-4

U

UNK instructions 1-16

V

verifying 1-15

Notes:

How Are We Doing?

Your comments on our technical publications will help us serve you better in the future.
Thank you for taking the time to provide us feedback.

You can complete this form and mail it back to us, visit us online at www.ab.com/manuals, or email us at RADocumentComments@ra.rockwell.com

Pub. Title/Type Converting PLC-5 or SLC 500 Logix to Logix5000 Logix Reference Manual

Cat. No. Logix-based controllers Pub. No. 1756-RM085B-EN-P Pub. Date November 2001 Part No. 957555-71

Please complete the sections below. Where applicable, rank the feature (1=needs improvement, 2=satisfactory, and 3=outstanding).

Overall Usefulness	1 2 3	How can we make this publication more useful for you?
Completeness (all necessary information is provided)	1 2 3	Can we add more information to help you? ☐ procedure/step ☐ illustration ☐ feature ☐ example ☐ guideline ☐ other ☐ explanation ☐ definition
Technical Accuracy (all provided information is correct)	1 2 3	Can we be more accurate? ☐ text ☐ illustration
Clarity (all provided information is easy to understand)	1 2 3	How can we make things clearer?
Other Comments		You can add additional comments on the back of this form.

Your Name ____________ Location/Phone ____________

Your Title/Function ____________ Would you like us to contact you regarding your comments?

___No, there is no need to contact me

___Yes, please call me

___Yes, please email me at ____________

___Yes, please contact me via ____________

Return this form to: Allen-Bradley Marketing Communications, 1 Allen-Bradley Dr., Mayfield Hts., OH 44124-9705

Phone: 440-646-3176 Fax: 440-646-3525 Email: RADocumentComments@ra.rockwell.com

PLEASE FASTEN HERE (DO NOT STAPLE)

Other Comments

PLEASE FOLD HERE

PLEASE REMOVE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 18235 CLEVELAND OH

POSTAGE WILL BE PAID BY THE ADDRESSEE

Allen-Bradley
RELIANCE ELECTRIC
DODGE
ROCKWELL SOFTWARE
Rockwell Automation

1 ALLEN-BRADLEY DR
MAYFIELD HEIGHTS OH 44124-9705

Reach us now at www.rockwellautomation.com

Wherever you need us, Rockwell Automation brings together leading brands in industrial automation including Allen-Bradley controls, Reliance Electric power transmission products, Dodge mechanical power transmission components, and Rockwell Software. Rockwell Automation's unique, flexible approach to helping customers achieve a competitive advantage is supported by thousands of authorized partners, distributors and system integrators around the world.

Americas Headquarters, 1201 South Second Street, Milwaukee, WI 53204, USA, Tel: (1) 414 382-2000, Fax: (1) 414 382-4444
European Headquarters SA/NV, avenue Herrmann Debroux, 46, 1160 Brussels, Belgium, Tel: (32) 2 663 06 00, Fax: (32) 2 663 06 40
Asia Pacific Headquarters, 27/F Citicorp Centre, 18 Whitfield Road, Causeway Bay, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication 1756-RM085B-EN-P - November 2001 PN 957555-71
Supersedes Publication 1756-6.8.5 - April 1999