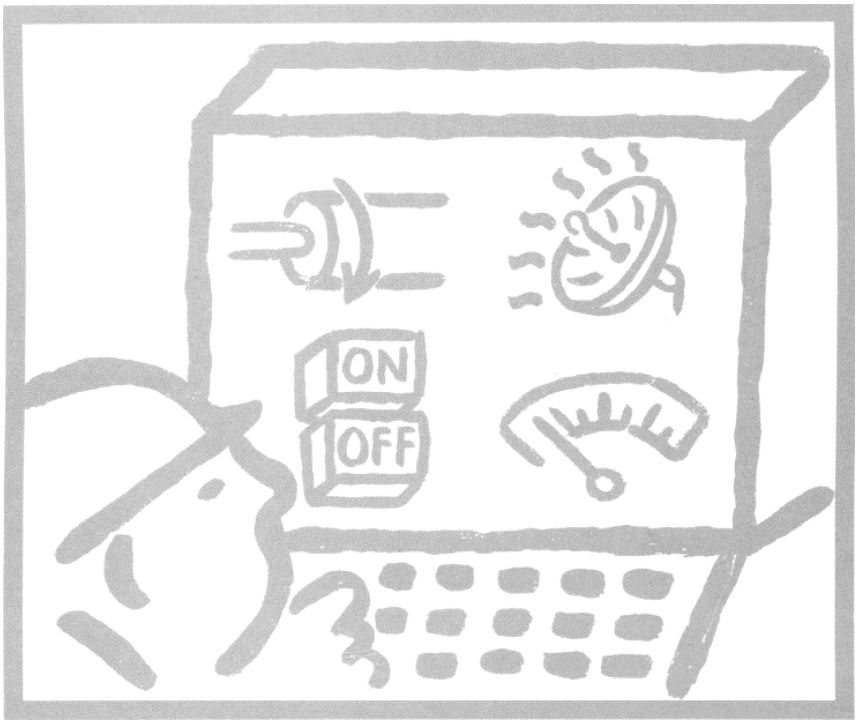
ASIC-200 Version 5.0

integrated industrial control software



Getting Started

Pro-face XYCOM

Revision	Description	Date
D	Name change, correct where applicable with document	4/07

Getting Started: 137586(D)

Published by: Pro-face
750 North Maple Road
Saline, MI 48176

ASCI-200 release 5.0 documents include:

Getting Started	137586
User Guide	139837
Language Reference	139183
HMI Guide	139168

Note: the current revisions of each of these documents should be used.

Note: Features available on your system depend on product version and installed options (toolkits).

Contents

Overview

ASIC-200 Software as a PLC Replacement

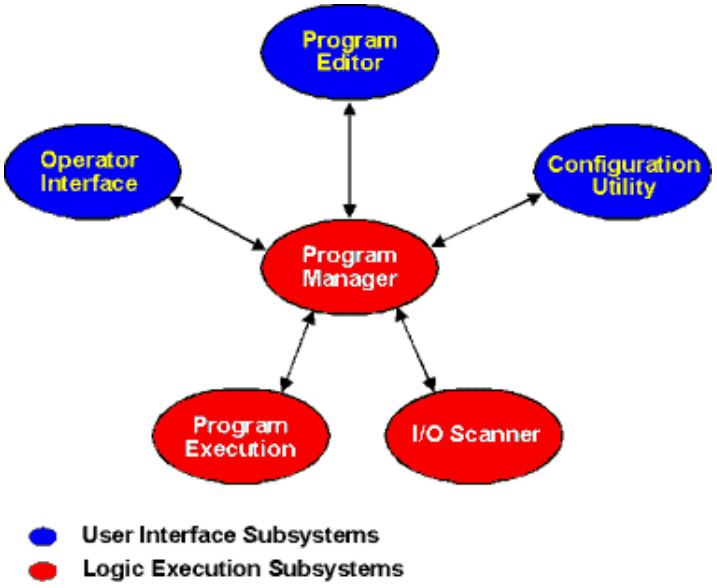
The ASIC-200 software

- Does not require a PLC device to do logic execution
- Performs logic and motion command execution in the PC without the need for a PLC
- Can use PC based scanner cards to control PLC style I/O racks, modules and other devices

Note: Features available in your system depend on version (ASIC-100, ASIC-200, or ASIC-200 with Motion) and options.

Overview of the Software

ASIC-200 software consists of the following subsystems:



Subsystem	Function
Configuration Utility	Defines the I/O structure and assign tag names to I/O points and I/O ports. For applications with motion control, the configuration utility lets you define motion parameters.
Operator Interface	Design and operate the operator interface environment. The Operator Interface is a series of screens, messages, or windows that are presented to the operator to control and monitor a machine or process.
Program Editor	Creates and runs Relay Ladder Logic, Sequential Function Chart (SFC+), Structured Text, and Instruction List programs.
Program Manager	Prepares programs for execution and manages the external interface to Program Execution and the I/O Scanner.
Program Execution	Executes the Relay Ladder Logic, Sequential Function Chart (SFC+), Structured Text, and Instruction List programs. Program Execution runs at Windows NT's real-time process priority.
I/O Scanner	Scans the physical I/O devices and makes the information available to Program Execution. The I/O Scanner runs at the real-time process priority. The I/O Scanner retrieves all inputs from and transmits all outputs to the physical I/O interface.

IEC 1131 Overview

IEC 1131-3 is an international standard from the International Electrotechnical Commission. IEC 1131-3 specifies the syntax and semantics of a unified suite of programming languages for programmable controllers. The IEC-1131-3 languages consist of textual and graphical languages. These languages can be used together in an integrated programming environment.

For information about ASIC-200 software enhancements to the IEC 1131-3 standard, refer to the RLL and SFC sections of the **User Guide**.

Textual Languages	
Instruction List	Instruction List format is similar to an assembly language.
Structured Text	Structured Text is best suited for complex algorithms, string and file operations, and manipulation of data structures best done in a procedural (text based) language. It is convenient for those who have experience with structured BASIC, Pascal, C or other high-level programming languages.

Graphical Languages	
Ladder Diagram (Relay Ladder Logic)	Ladder diagram is commonly used on programmable controllers to construct discrete logic programs. Ladder diagrams are designed to resemble the electrical diagram for an equivalent electrical relay logic circuit. The ladder diagram contains two vertical power rails. The left power rail is assumed to be an electrical current source and is energized whenever the program is running. The power rail on the right is assumed to be an electrical current sink. The two power rails are connected by horizontal lines called rungs (like the rungs of a ladder) on which the logical instructions are placed.
Sequential Function Chart	Sequential Function Chart is best suited for machine sequencing control and for control applications that have multiple operating modes, such as manual mode and automatic mode. A Sequential Function Chart represents an application program as a series of sequential steps. Steps are connected by links and control is passed between steps via transitions as the program executes. Control logic can be placed in the step or within an associated action attached to the step. The control logic executes when the step becomes active. Control passes to the next step when the transition is cleared.
Function Block Diagram	Function Block Diagram is best suited for analog signal processing and any control process or algorithm that runs on a continuous basis. Functions and function blocks appear in the FBD language as graphical blocks labeled with their function name and inputs and outputs. Function Block Diagrams consist of a network of these graphical functions and function blocks connected by signals. This language is not supported by ASIC-200.

Software Authorization

Software Authorization Description

ASIC-200 software can be run in demo mode or authorized mode. The software can be authorized by purchasing it with a hardware key or a software key.

Demo Mode

If in demo mode, the software runs in a two-hour demo mode. You can use the software as many times as you want, but only for two hours at a time.

Hardware Key

If you have a hardware key, install the hardware key on the PC's parallel port. The authorization information is read from the key when the ASIC software is started. The hardware key can be moved from one PC to another. There are no other authorization options if you have a hardware key.

Software Key

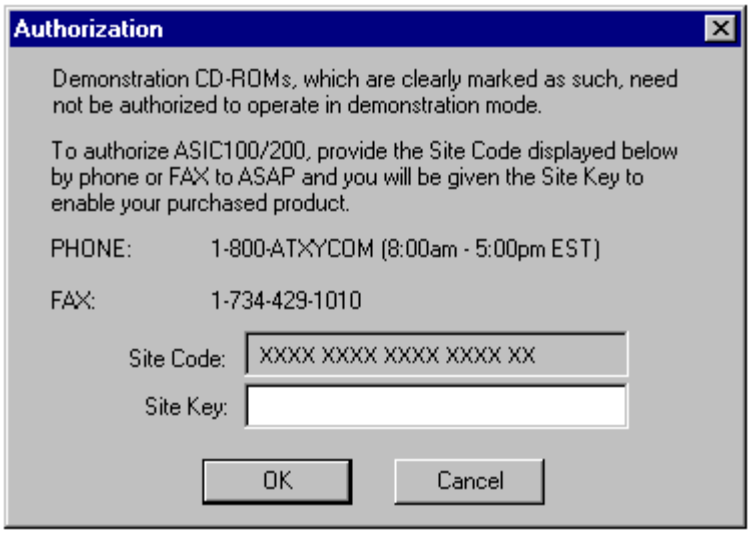
If you will be using a software key, you must provide ASAP with your Site Code; you will then be given a Site Key to authorize your software license.

Authorize

To authorize your software license:

1. Do one of the following:
   - Select *Authorize* from the dialog box that appears when you start the Program Editor from the ASAP Applications menu.
   - If you have started the Program Editor in Demo mode, select *Authorize* from the *License* menu.

The *Authorization* dialog box appears:



2. Contact the number provided and have your *Site Code* available.
3. When provided with your *Site Key*, enter it into the dialog box and click *OK*.

When you start the ASIC software, it will be authorized.

Upgrade

When you are ready to increase the functionality of ASIC-200 the *Upgrade Authorization* option from the *License* menu can be used to generate a new *Site Code*. This option is only enabled after the software has been authorized.

Temporary Authorization

Temporary authorization permits you to temporarily provide full development authorization on a run-time only system. Each run-time only system comes with 30 minutes of free temporary authorization; additional time can be purchased. Temporary authorization is valid for run-time only systems.

Selecting temporary authorization from the *License* menu displays the *Temporary Authorization* buttons and the amount of temporary authorization time remaining:

OK - turns on temporary authorization and starts the temporary authorization timer (providing there is time remaining).

Cancel - returns to run-time only.

Add More Minutes - allows you to purchase additional temporary authorization time for this system. Call the phone number given in the dialog box that appears and provide your code from the *Code* field. You will be given a key in return that must be entered in the *Key* field to add the additional temporary authorization time.

Emergency Authorization

Emergency authorization provides full authorization for all features for 4 days. After 4 days, ASIC-200 will return to demo mode. Once it has been selected, a warning indicates that this is a one-time authorization until the hard drive has been formatted. If you elect to continue, a signature is written to the hard drive.

To enable emergency authorization, choose *Emergency Authorization* from the *License* menu.

License Transfer

A software license installed as described in ***Authorize*** can be transferred to another PC. Assume you want to transfer license from machine **A** to machine **B** and the software is currently loaded on machine **A**. You start from the destination machine and do a *Register Transfer* operation. Next you go to the source machine and do a *Transfer Out* operation. Then you go back to the destination machine and do a *Transfer In* operation. These operations are performed from the ASIC-200 *License* menu.

The steps are detailed below:

1. First load ASIC-200 on machine **B**. On machine **B** start ASIC-200 in Demo mode.
2. Insert a diskette in machine **B** and from the *License* menu select the *Register Transfer* item.
3. When the *Register Transfer* is complete, remove the diskette from machine **B** and insert it in machine **A**.
4. At machine **A** perform a *Transfer Out* operation by selecting this item from the *License* menu.
5. Now you can transfer the license to machine **B** by inserting the diskette in machine **B** and performing a *Transfer In* operation by selecting this item from the *License* menu.

Note: The above can be performed on the network also.

Quick Start

Checklist

Use this as a checklist for getting started:

1. Install all PC hardware and I/O cards.
2. Start the Windows NT operating system.
3. Install ASIC-200 software and I/O drivers. Use a hardware key or your software key to authorize the program license.
4. Use the Program Editor to create a new project to hold the application program and configuration files. Refer to **Projects** in the *User Guide.*
5. Use the Configuration utility to configure the system hardware and define I/O symbol names for the application programs. Refer to **I/O Configuration** in the *User Guide.*
6. Use the Program Editors to construct the application program. Refer to the **Application Programs** and the programming sections in the *User Guide.*
7. Use the Symbol Editor to add global and local program variables as needed. Refer to **Symbol Configuration** in the *User Guide.*
8. Use the Operator Interface Screen Editor (located under the Tools menu) to construct the Human-Machine Interface screens. Alternatively, you can use a third-party HMI. Refer to the *HMI Guide.*
9. Use the Operator Interface and Program Editor to test the application programs.
10. You can start your application program from the Program Editor *Execute* menu. If you haven't started the runtime system, you will be prompted to do so; or you can manually start the runtime system from the *Execute* menu as well. Refer to **Running Application Programs** and **Program Operation** in the *User Guide.*
11. You can also perform online editing operations, refer to **On-Line Editing** in the *User Guide.*
12. After starting the program, you can use the Watch Window (*Watch/Force Variables* on the *View* menu) to observe or set program symbols. Refer to

Running Application Programs in the *User Guide*. You can also display symbol values graphically using the Trace feature from the *Tools* menu. Refer to **Trace** in the *User Guide*.

13. If you are transferring the application program to a runtime node, refer to **Transferring Project to Runtime** in the *User Guide*.
14. If desired, configure Windows NT to automatically start the ASIC-200 software and application programs on power up.

Running ASIC-200 Software

To start the Operator Interface

For a control system that includes an HMI, and you wish to run or edit the operator screens, do one of the following:

- Locate the *ASAP Applications* menu from the Windows *Start* menu and choose *Operator Interface*.
- If the Program Editor is open, choose *Operator Interface* from the Program Editor *Tools* menu.

The Operator Interface starts in activation mode. The last operator interface file open for the current project is used to define the operator interface. The start screen defined in the operator interface file is the first operator interface screen displayed.

If the runtime subsystems are not running when the Operator Interface is started, a prompt appears providing an opportunity to do so.

To start the Program Editor

If you wish to edit a program or run a program from the Program Editor, do one of the following:

- Locate the *ASAP Applications* menu from the Windows *Start* menu and choose *Program Editor*.
- If the Operator Interface Editor is open, choose *Program Editor* from the Operator Interface Editor *Tools* menu.

To start the Runtime Subsystems

Do one of the following:

- Locate the *ASAP Applications* menu from the Windows *Start* menu and choose *Runtime Engine*.
- If the Operator Interface or Program Editor is open, choose *Startup Runtime Subsystems* from the Operator Interface or Program Editor *Execute* menu.

When the Runtime subsystems are active the Runtime icon appears on the Task Bar or Tray at the bottom of the screen.

If the Runtime icon appears on the Tray, the mouse must be used to shutdown the Runtime Subsystems. If the Runtime application also appears on the Task Bar, it can also be shut down using the keyboard. To make the Runtime icon appear in the Task Bar, access *System Options* from the Program Editor *Tools* menu and check the option in the *Display Properties* tab.

To shut down ASIC-200 Runtime Subsystems

Do one of the following

- Right mouse click on the Runtime icon on the Tray and select *Shutdown Runtime.*
- If the Runtime application appears on the Task Bar, open the Runtime application and choose *Shutdown Runtime* from the *Shutdown* menu.

Runtime Subsystems

The Program Manager, Program Execution, and I/O Scanner subsystems are visually represented by the Runtime icon. The Event Log has its own icon.

Subsystem	Description
Program Manager	Prepares programs for execution and manages the external interface to Program Execution and the I/O Scanner.
Program Execution	Has real-time process priority, meaning CPU time is given to Program Execution before all normal applications, including mouse updates and disk access.
I/O Scanner	Has real-time process priority, meaning CPU time is given to I/O Scanner before all normal applications, including mouse updates and disk access. When the I/O Scanner starts up, it searches for motion and/or I/O cards. If no cards are detected in the PC backplane, a message box is displayed. Push the Simulate button to ignore the missing hardware and run the ASIC-200 software in simulation mode. In simulation mode, the inputs and outputs are resident in a memory table, but the values are not read from or written to the I/O devices. User simulation software can write to the input memory map and read from the output memory map to simulate I/O in the environment.

Subsystem	Description
Event Log	Stores time stamped system events, messages and errors. The Event Log is started and stopped automatically by the runtime subsystems. The Event Log must be running for messages to the Output Window to function.

Program Editor

The Program Editor lets you organize your work on a project by project basis. You can create and manage any number of projects from within the Program Editor.

At any one time only one project can be open. This is called the active project. Each project has an active configuration associated with that project. When a project is opened, the active configuration for that project is activated. When a new project is opened all programs are cancelled.

For more information, refer to the **User Guide**.

Access Levels

Description

Access levels and access codes are used to control access to application programs, operator interface screens, and configuration data. There are five access levels, with a different access code for each. The following table describes the privileges of each level. The lowest access level is 0; each successively higher access level has the privileges of the preceding levels.

Access Level	Default Code	Privileges
0	<Cancel>	Use operator interface screens, including selecting and running programs and using jogging and homing axes.
1	1234	Open, view, and monitor programs and symbols from the Program Editor. Perform licensing functions.
2	2345	Same as Access Level 1. (Can be used as an additional access level within the HMI for button operation.)
3	3456	Perform project functions. Edit system configurations. Edit programs. Edit operator interface screens.
4	4567	Change passwords for access levels 1 to 4 (from the Operator Interface).

Note: If you have the Program Editor open, changing the access level within the Operator Interface does not change the access level of the Program Editor.

Entering an Access Code

To enter an access code

1. Choose *Enter Password* from the *Access* menu. A keypad appears.
2. Enter the numbers on the keypad that represent your access code and click *OK*.

To set the access level to 0

- Click *Cancel* on the access keypad.

Assigning a Code to an Access Level

Access codes can be assigned to an access level from the Operator Interface provided that the current access level is level 4. There is one access code for each level.

To assign an access level code

1. Choose *Enter Password* from the *Access* menu. A keypad appears.
2. Click ***** (the asterisk key) four times. The message *Enter Access Level to Change.* appears.
3. Enter the number of the access level you want to change and click *OK*.
4. Enter the new four digit number. The message: *Enter new password again.* appears.
5. Enter the new four digit password again and click *OK*. If you verify the new password correctly, the password is changed.

Notes:

1. Click *Cancel* on the keypad to set the access level to level 0.
2. When attaching control functions to operator controls (in the operator interface edit mode) an access level can be specified to control the use of the operator control.

Menu Descriptions

Menu Descriptions

Available menu items are dependent on the access level entered. The highest access levels (3 and 4) have all menu items available.

File Menu

File Menu		
Item	**Button**	**Description**
New Editor		Creates a new program file.
Open Editor		Opens an existing program file.
Close	-	Closes all the windows associated with the active program file.
Save		Saves the active program file.
Save As	-	Saves the active program or configuration under a different name or version.
Save All	-	Saves all open program files and related project information.
Print...		Prints a program.
Print Preview	-	Displays a preview of the printed page. You can use this to adjust the placement of elements so that they do not span page boundaries.
Print XRef...	-	Prints program variables and where and how often they are used in the program. This option appears only when a program file is open.
Print Setup	-	Displays the *Print and Title Block Setup* dialog box. You can edit title and description for the page, page scaling, and change the printer and printing options.

File Menu		
Item	**Button**	**Description**
Reset Editor GUI	-	Resets tool bars to default positions.
New Config	-	Opens the Configuration Utility with a new I/O configuration file.
Open Config	-	Opens an existing I/O configuration file in the Configuration Utility.
Save Config	-	Saves the active configuration.
Save Config As	-	Saves the active configuration with a new name or path.
Import CSV to Config	-	Imports a comma separated configuration file.
Export Config to CSV	-	Exports the active configuration to a comma separated file that can be accessed by a text editor or spreadsheet.
Recently Used File List	-	List of the last four files accessed by the Program Editor.
Exit	-	Closes the ASIC-200 software. If the unsaved programs are open, you are prompted to save them.

Edit Menu - ALL

Edit Menu - All		
Item	**Button**	**Description**
Undo		Undoes the last action.
Redo		Redoes the previously undone action.
Cut		Cuts the selected object and places it on the clipboard.
Copy		Copies the selected object and places it on the clipboard.
Paste		Pastes the contents of the clipboard.
Delete	-	Deletes the selected object.
Select All	-	Selects all text in an ST or IL document.

Edit Menu - RLL

Edit Menu - RLL		
Item	**Button**	**Description**
Edit Element...	-	Opens the dialog box for the selected program element.
New Element	-	Insert an SFC or RLL element at the selected point in the program.
New Function Block	-	Inserts a function block element at the selected point in the program. Available only when an RLL program is open for edit.
Find...		Finds the specified text.
Find Next		Finds the next occurrence of the specified text.
Find Prev.		Finds the previous occurrence of the specified text.
Replace...	-	Replaces the specified text with new text.
Go To…	-	Jumps to the specified rung number. Available only when an RLL program is open for edit.
Trace...		Searches for the next occurrence of a output coil with the same symbol name as a selected contact. Available only when an RLL program is open for edit.
Insert Mode		Inserts new elements before the selected point.
Append Mode		Appends new elements after the selected point.

Edit Menu - SFC

Edit Menu - SFC		
Item	**Button**	**Description**
Edit Element...	-	Opens the dialog box for the selected program element.
New SFC Element	-	Inserts a new SFC element.
Step Properties	-	Edits an SFC step.
Find...		Finds the specified text.
Find Next		Finds the next occurrence of the specified text.
Find Prev.		Finds the previous occurrence of the specified text.
Replace...	-	Replaces the specified text with new text.

Edit Menu - SFC		
Item	**Button**	**Description**
Insert Mode		Inserts new elements before the selected point.
Append Mode		Appends new elements after the selected point.
Boolean Transitions	-	When editing an SFC, sets the default transition for all open SFC programs to the Boolean type, instead of the RLL type. Does not change existing transitions. Available only when an SFC program is open for edit.
Lock Algorithms	-	When editing an SFC, lets you add password protection to step algorithms. Available only when an SFC program is open for edit.

Edit Menu - Structured Text

Edit Menu - Structured Text		
Item	**Button**	**Description**
Insert ST Statement	-	Inserts a Structured Text statement.
Insert ST Function Calls	-	Inserts a Structured Text function.
Find...		Finds the specified text.
Find Next		Finds the next occurrence of the specified text.
Replace...	-	Replaces the specified text with new text.

Edit Menu - Instruction List

Edit Menu - Instruction List		
Item	**Button**	**Description**
Insert IL Statement	-	Inserts an Instruction List statement.
Insert IL Function Calls	-	Inserts an Instruction List function.
Find...		Finds the specified text.
Find Next		Finds the next occurrence of the specified text.
Replace...	-	Replaces the specified text with new text.

View Menu - All

View Menu - All		
Item	**Button**	**Description**
Toolbar	-	Displays editor commands as icons.
Status Bar	-	Displays the editor window status bar.
Single Axis Status	-	Displays all status information for a single axis.
Multi Axis Status	-	Displays specified status information for all axes
Axis Plot	-	Displays a plot of the status information for an axis.
Watch/Force Variables		At runtime, this window displays variables and their current values. You can specify (force) new values for them to help in debugging your program. Available only when a program is open for edit.
I/O Faults	-	Displays the I/O Fault window.
Output Window	-	Messages that occur at runtime appear in this window.
Program Status	-	Opens the Program Status window.
Show Parse Errors	-	Displays the Parse Results window.
Show Symbol Enumerations		Turns on the symbol enumerations that have been selected in the System Options dialog box.

View Menu - RLL

View Menu - RLL		
Item	**Button**	**Description**
Zoom	-	Various magnification of displayed program.
Scale to Fit Window		Resizes the program so that it fits into the currently displayed window. Available only when a program is open for edit.
Contact/Coil Bar	-	When editing an RLL program, the Contact/Coil bar displays the program elements that can be used in the program. Available only when an RLL program is open for edit.
Function Block Palette	-	When editing an RLL program, the Function Block Palette displays the function blocks that can be used in the program. Available only when an RLL program is open for edit.

View Menu - RLL		
Item	**Button**	**Description**
RLL Warnings	-	Turns notification of RLL warnings on or off. Warnings occur during a program parse.
All Steps	-	When editing an SFC program, this option lets you specify that steps be displayed showing their names, descriptions, associated icons, or their program code. Available only when an SFC program is open for edit.
Document Formatting	-	Refer to the following five items:
Program Comments		Displays or hides any program comments that you enter into the program. Available only when a program is open for edit.
Symbol Location		Displays the rack/slot/point information for I/O symbols.
FB Details		Turns On/Off the display of symbol names and real time data in RLL function blocks. Available only when an RLL program is open for edit.
Rung Wrapping	-	Wraps rungs to the number of pages defined in Print Setup. When the rung contents approach the page boundary of the last horizontal page (number of pages wide setting), the rung returns to the left edge of the first page and continues. Some things cannot be wrapped such as contacts or coils with very long symbol names, and branches.
Auto Page Break	-	Automatically breaks pages to avoid splitting elements on a rung horizontally. Prevents printing the top half of a rung's contents on one page and the bottom half on the next.
Print Boundaries	-	Displays faint lines at the left and lower edges of SFC and RLL diagrams to show the printed page boundaries.

View Menu - SFC

View Menu - SFC		
Item	**Button**	**Description**
Zoom	-	Various magnification of displayed program.
Scale to Fit Window		Resizes the program so that it fits into the currently displayed window. Available only when a program is open for edit.

View Menu - SFC		
Item	**Button**	**Description**
SFC Bar	-	When editing an SFC program, the SFC toolbar displays the program elements that can be used in the program. Available only when an SFC program is open for edit.
Application Icon Palette		When editing an SFC program, the Application Icon Palette displays the Icons that can be used in the program. Available only when an SFC program is open for edit.
Program Comments		Displays or hides any program comments that you enter into the program. Available only when a program is open for edit.
RLL Warnings	-	Turns notification of RLL warnings on or off. Warnings occur during a program parse.
All Steps	-	When editing an SFC program, this option lets you specify that steps be displayed showing their names, descriptions, associated icons, or their program code. Available only when an SFC program is open for edit.
Print Boundaries	-	Displays faint lines at the left and lower edges of SFC and RLL diagrams to show the printed page boundaries.

View Menu - Structured Text

View Menu - Structured Text		
Item	**Button**	**Description**
Accessory Bar		Displays the Structured Text or Instruction List accessory bar that automatically inserts program statements.

View Menu - Instruction List

View Menu - Instruction List		
Item	**Button**	**Description**
Accessory Bar		Displays the Structured Text or Instruction List accessory bar that automatically inserts program statements.

Project Menu

Project Menu	
Item	**Description**
New	Creates a new project for storing related application programs and configurations.
Open	Opens an existing project.
Copy	Copies an existing project (including all application programs and configuration files) to a new name.
Rename	Renames an existing project. Has no effect on the application programs and configuration files in the project.
Activate Config.	Activates the current configuration file. The current configuration file is displayed on the status bar at the bottom of the screen.

Execute Menu

Execute Menu	
Item	**Description**
Parse	Compiles the program
Run	Compiles then executes the program.
Run with Debug	Executes the program until it reaches a Structured Text BREAK statement and then stops executing and puts the program in a "Break" status. BREAK statements are ignored if a RUN command is given. Available only when an SFC program is open for edit.
Single Step	Executes one command at a time. Available only when an SFC program is open for edit.
Run with Restart	Marks the program to begin executing every time the runtime is started. A program that has been Aborted or is Faulted will no longer be marked to run with restart.
Stop	Stops executing program where it was stopped. When you restart the program, it begins executing at the point it left off. Available only when an SFC program is open for edit.
Abort	Stops executing program. When you restart the program, it starts executing at the beginning of the program.
Reset Estop and clear I/O Faults	Resets the I/O drivers and reestablishes communication with the I/O hardware.

Execute Menu	
Item	**Description**
Startup Runtime Subsystems	Startup the runtime subsystem programs.

Icon Menu

Icon Menu	
Item	**Description**
New App Icon	Accesses the Application Icon Step Library allowing you to create predefined Steps in the library.
Edit App Icon	Edits predefined Steps in the Application Icon library.
Delete App Icon	Deletes predefined Steps in the Application library.

Tools Menu

Tools Menu		
Item	**Button**	**Description**
Operator Interface	-	Starts the integrated Operator Interface
Symbol Manager		Accesses the Symbol Manager allowing you to create and edit variables.
Action Manager	-	Accesses the Action Manager allowing you to rename and delete SFC actions.
RLL Transition Manager	-	Accesses the Transition Manager allowing you to rename and delete SFC, RLL transitions.
Export Symbols	-	Exports all Global Symbols to a *.SNF file. Available only with the CIMPLICITY HMI interface.
System Options	-	Opens a dialog box that allows you the change the Heap Size and the memory variable Data Table size, display, and symbol enumeration preferences.
Trace Logging	-	Starts the Trace Logging utility that is used in conjunction with Trace Graphing. Permits you to log symbol values.
Trace Graphing	-	Graphs symbol values that have been logged using Trace Logging.

License Menu

License Menu	
Item	**Description**
Authorize	This option is used to create a software "License" for the ASIC-200 Software. Call or Fax your "Site Code" in to ASAP Inc. to receive a "Site Key" to authorize your software. This option is only activated when the software has been started in demo mode.
Upgrade Authorization	When you are ready to increase the functionality of ASIC-200 the Upgrade Authorization option can be used to generate a new "Site Code". This option is only enabled after the software has been authorized.
Register Transfer	Use this option to begin the process of transferring a software license from one computer to another. Place a blank floppy disk in a unauthorized computer and select "Register Transfer" to write a unique signature file for the computer. This option is only active when the software has been started in demo mode.
Transfer Out	Transfer Out is the second step in the process of transferring a software license from one computer to another. After completing the Register Transfer process on the unauthorized computer, place the floppy disk in the authorized computer and select the Transfer Out option. The software license will be removed from the computer and placed on the floppy disk. This software license can only be transferred to the computer that wrote the original signature file to the floppy. This option is only active on an authorized computer.
Transfer In	Transfer In is the last step in the process of transferring a software license from one computer to another. Place the floppy disk in the computer that has had the software license transferred. Select Transfer In to move the software license to the computer. A software license can only be transferred to the computer that wrote the original signature file to the floppy.

License Menu	
Item	**Description**
Temporary Authorization	Temporary authorization permits you to temporarily provide full development authorization on a run-time only system. Each run-time only system comes with 30 minutes of free temporary authorization; additional time can be purchased. Valid for run-time only systems. Selecting temporary authorization displays the *Temporary Authorization* buttons and the amount of temporary authorization time remaining: *OK* - turns on temporary authorization and starts the temporary authorization timer (providing there is time remaining). *Cancel* - returns to run-time only. *Add More Minutes* - allows you to purchase additional temporary authorization time for this system. Call the phone number given in the dialog box that appears and provide your code from the *Code* field. You will be given a key in return that must be entered in the *Key* field to add the additional temporary authorization time.
Reset Original Authorization	Turns off temporary authorization and stops the temporary authorization timer. Valid for run-time only systems.
Emergency Authorization	Provides full authorization for all features for 4 days. After 4 days, ASIC-200 will return to demo mode. Once it has been selected, the user is warned that this is a one-time authorization until the hard drive has been formatted. If they select to continue, a signature is written to the hard drive.

Status Bar

The Status Bar is located at the bottom of the Program Editor window. The status bar shows the name of the current configuration and the status of:

- NUM lock
- SCROLL lock
- CAPS lock
- Keyboard keys
- Insert and append modes

Tutorial

Use the tutorial to practice editor features.

1. Open Program Editor Help.
2. Click Options and select Keep Help on Top.
3. Select *On Top*. This leaves the help window visible while you are using the Program Editor.
4. Size and move the help window so you can work in the Program Editor.

Note: A help window with a scroll bar indicates the help topic is larger than the help window. Use the scroll bar to see all the information, or try expanding your window.

Creating a Sample Program

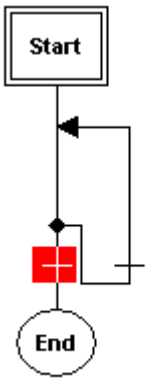
1. Start Program Editor.
2. Click the *Project* menu and select Open. The Open Project From dialog box appears.
3. Click the project called *SAMPLE* and click *OK*.
4. Click *Project* and select *Activate Config*. The Select Global Configuration File dialog box appears.
5. Select the file called *PAINTER.CFG* and click *Open*. This file contains I/O symbols you can use to create the program.
6. Click File and select New. The New dialog box appears.
7. Select SFC+M form and click OK. A window opens that shows the starting and end points of the new program.

Creating the Program Structure

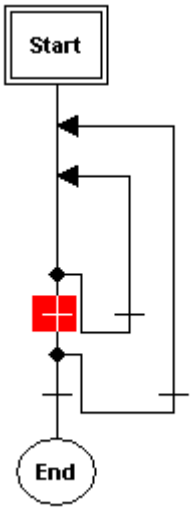
Press the Boolean transition mode button on the editor tool bar to select the Boolean transition mode. The button should be displayed depressed to reflect the active Boolean transition mode. 

1. Select the loop tool from the SFC Tool Bar. 

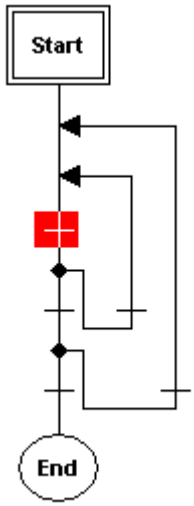
2. Move the cursor over the control path between the Start step and the End step. Press the left mouse button to drop a control loop element into the SFC program.



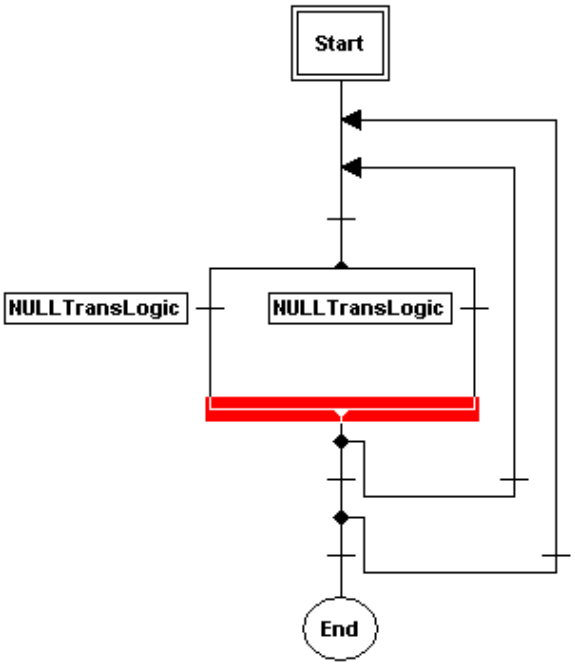
3. Move the cursor over the control path in the center of the loop. Press the left mouse button to drop a nested control loop element into the SFC program.



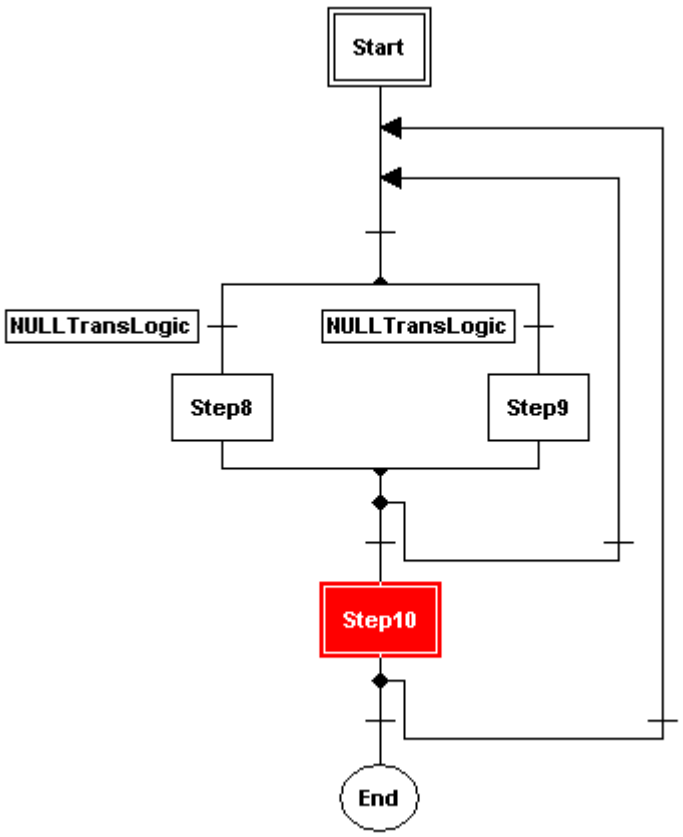
4. Select the transition tool from the SFC Tool Bar. 
5. Move the cursor over the control path in the center of the inner loop. Press the left mouse button to drop a transition element into the SFC program.



6. Press the Boolean transition mode button on the editor tool bar to deselect the Boolean transition mode. The button should be displayed non-depressed to reflect the inactive Boolean transition mode. 
7. Select the select diverge tool from the SFC Tool Bar. 
8. Move the cursor over the control path in the center of the inner loop, below the transition element. Press the left mouse button to drop a select diverge element into the SFC program (with RLL transitions).



9. Select the step tool from the SFC Tool Bar. 
10. Move the cursor over the left control path of the select diverge, below the transition. Press the left mouse button to drop a step element into the SFC program. Move the cursor over the right control path of the select diverge, below the transition. Press the left mouse button to drop a step element into the SFC program. Move the cursor over the control path in the outer loop, below the bottom of the inner loop. Press the left mouse button to drop a step element into the SFC program.



About the Boolean and RLL Transition Modes

When the RLL transition mode is active, all transitions added to the SFC will take the form of RLL transitions. When the Boolean transition mode is active, all transitions added to the SFC will take the form of Boolean transitions.

To toggle the state between RLL transition mode to Boolean transition mode, click *Edit* and select *Boolean Transitions* menu, or click the Boolean transition button on the SFC Editor Tool Bar.

When the Boolean transition mode is active the edit menu displays a check mark next to the Boolean Transition command and, the SFC edit tool bar displays the Boolean transition button as depressed.

Filling in Step Details

1. Select the selector tool from the SFC Tool Bar. 
2. Double click on the top leftmost step box (step8) to open the step edit dialog box.
3. Double click on the word "step8" to select the identifier in the Step Name edit box and change the name of the step to PaintCar. Select the Display Motion / Process command type. Select RS-274D as the Command Type. Select the Command List edit box and type in the desired RS-274D commands (these commands could also be structured text commands). For the sample program, type in the following commands:

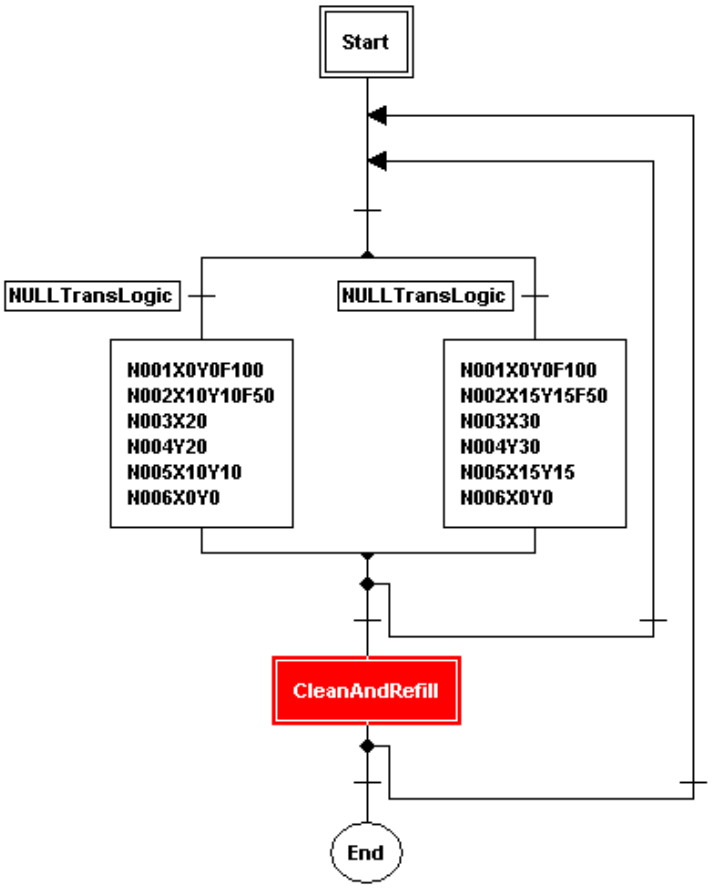
   N001G01X0Y0F100
   N002X10Y10F50
   N003X20
   N004Y20
   N005X10Y10
   N006X0Y0

4. Press the OK button to close the step edit dialog box and accept the changes.
5. Double click on the top rightmost step box (step9) to open the step edit dialog box.
6. Select the Step Name edit box and change the name of the step to PaintTruck. Select the Display Motion / Process command mode. Select RS-274D as the Command Type. Select the Command List edit box and type in the desired RS-274D commands. For the sample program, type in the following commands:

   N001G01X0Y0F100
   N002X15Y15F50
   N003X30
   N004Y30
   N005X15Y15
   N006X0Y0

7. Press the OK button to close the step edit dialog box and accept the changes.
8. Double click on the bottom step box (step10) to open the step edit dialog box.
9. Select the Step Name edit box and change the name of the step to CleanAndRefill. Press the OK button to close the step edit dialog box and accept the changes.

After filling in step details, the SFC program should look like this:

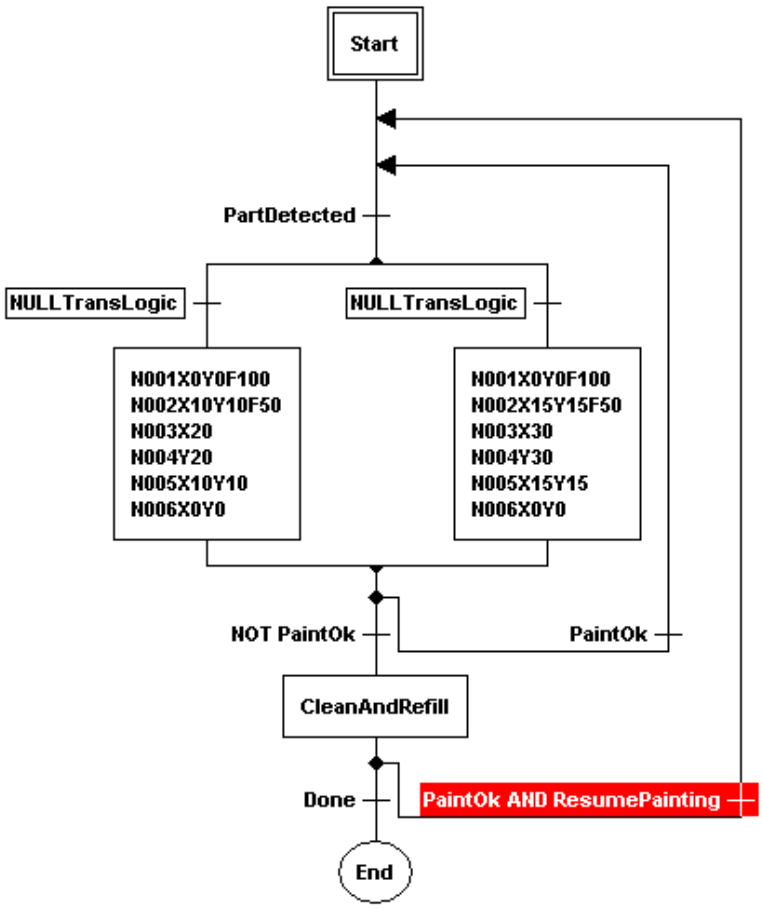


Filling in Boolean Transition Details

1. Select the selector tool from the SFC Tool Bar. 
2. Double click on the topmost transition to open the edit transition logic dialog box.
3. Using the scroll bar, scroll the Symbol List box down until the PartDetected symbol is visible. Double click on the PartDetected symbol to use it in the Transition Logic edit box. Press the OK button to close the edit transition logic dialog box and accept the changes.
4. Double click on the leftmost transition at the bottom of the inner loop to open the edit transition logic dialog box.

5. Press the NOT button to add the NOT operator to the Transition Logic edit box. Scroll the Symbol List box down until the PaintOk symbol is visible. Double click on the PaintOk symbol to use it in the Transition Logic edit box. Press the OK button to close the edit transition logic dialog box and accept the changes.
6. Double click on the rightmost transition at the bottom of the inner loop to open the edit transition logic dialog box.
7. Scroll the Symbol List box down until the PaintOk symbol is visible. Double click on the PaintOk symbol to use it in the Transition Logic edit box. Press the OK button to close the edit transition logic dialog box and accept the changes.
8. Double click on the leftmost transition at the bottom of the outer loop to open the edit transition logic dialog box.
9. Push the Symbol Manager button to define a new symbol.
10. Select the Select Symbol edit box and type in the name of the new variable. For the sample program type in the name "Done". In the Type box select the type BOOL. Enter any desired symbol comment in the Comment edit box and a full description of the purpose and use of the symbol in the Description edit box. Press the Add Symbol button to create the new symbol. The Done symbol will now appear in the Symbol List. Press the OK button to accept the new variable definition and close the Symbol Manager.
11. The new Done symbol will now appear in the Symbol List box of the edit transition logic dialog box. Double click on the Done symbol to use it in the transition logic edit box. Press the OK button to close the edit transition logic dialog box and accept the changes.
12. Double click on the rightmost transition at the bottom of the outer loop to open the edit transition logic dialog box.
13. Scroll the Symbol List box down until the PaintOk symbol is visible. Double click on the PaintOk symbol to use it in the Transition Logic edit box. Press the AND button to add the AND operator to the Transition Logic edit box. Scroll the Symbol List box down until the ResumePainting symbol is visible. Double click on the ResumePainting symbol to use it in the Transition Logic edit box. Press the OK button to close the edit transition logic dialog box and accept the changes.

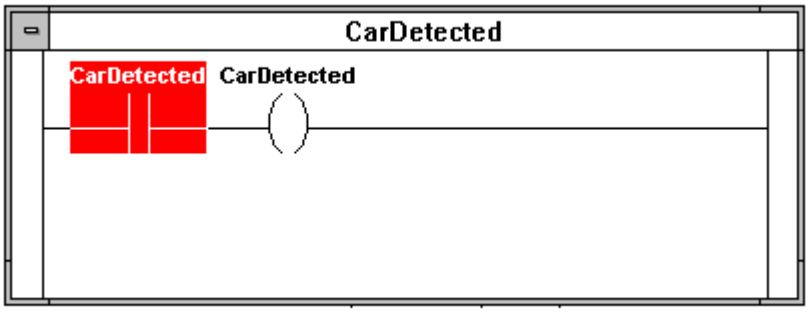
    After filling in Boolean transition details, the SFC program should look like the following figure.



Filling in RLL Transition Details

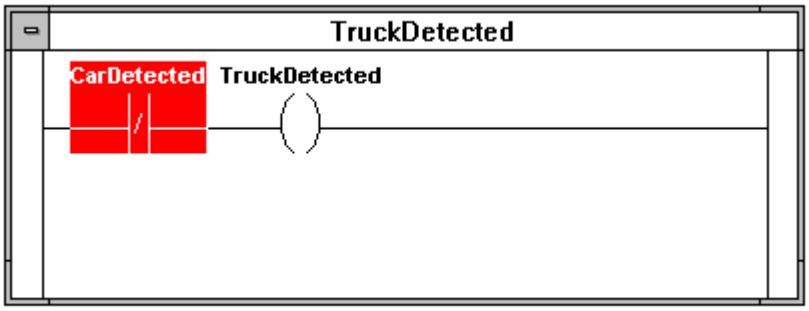
1. Select the selector tool from the SFC Tool Bar. 
2. Double click on the transition on the left control path of the select diverge. A message box will indicate that the RLL transition has not been defined.
3. Press the OK button and the Select RLL Transition dialog box appears.
4. Type in the name of the new RLL transition. For the sample program type in the name "CarDetected" and press the OK button to accept the name for the new RLL transition. A window opens to the right of the transition that contains the single rung RLL program for the RLL transition logic. The SFC Tool Bar is replaced with the RLL Tool Bar.

5. Click anywhere on the SFC program to replace the RLL Tool Bar with the SFC Tool Bar (editing the SFC). Click in the RLL window to replace the SFC Tool Bar with the RLL Tool Bar (editing the RLL). Select the contact tool from the RLL Tool Bar. 
6. Move the cursor over the RLL window, over the rung and to the left of the CarDetected coil. Press the left mouse button to drop a contact onto the RLL rung. The Edit Contact dialog box opens.
7. Select the Normally Open Contact as the Contact Type. Select the CarDetected symbol from the symbol list. Press the OK button to accept the contact definition and close the Edit Contact dialog box. The RLL program should now look like this:



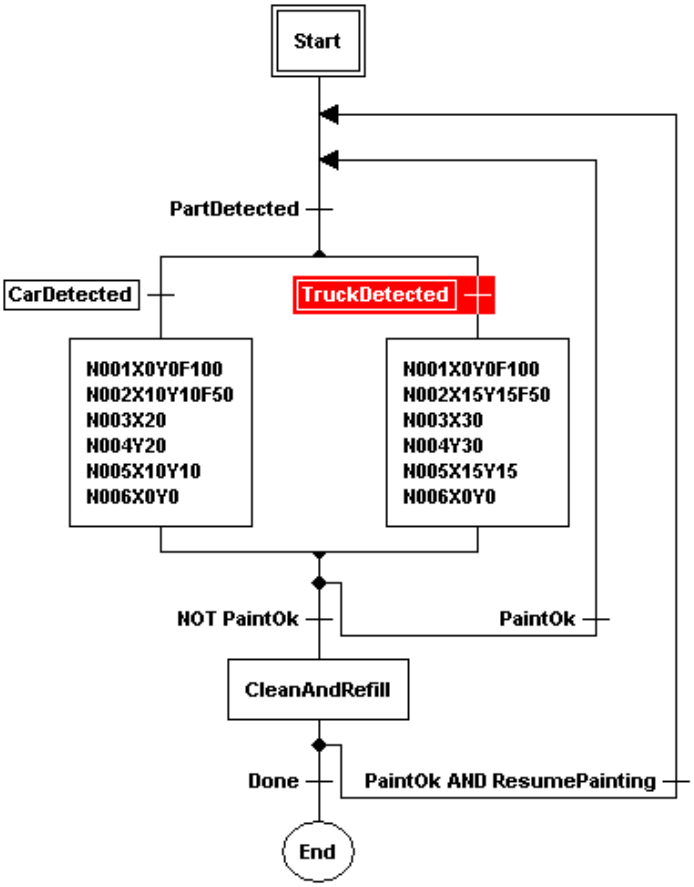
8. Double click on the control menu box on the top left corner of the RLL window to close the RLL window. 
9. Double click on the transition on the right control path of the select diverge. A message box indicates that the RLL transition has not been defined. Press the OK button and the Select RLL Transition dialog box appears.
10. Type in the name of the new RLL transition. For the sample program type in the name "TruckDetected." Press the OK button to accept the name for the new RLL transition. A window opens to the right of the transition that contains the single rung RLL program for the RLL transition logic. The SFC Tool Bar is replaced with the RLL Tool Bar. Select the contact tool from the RLL Tool Bar.
11. Move the cursor over the RLL window, over the rung and to the left of the TruckDetected coil. Press the left mouse button to drop a contact onto the RLL rung. The Edit Contact dialog box is opened. 

12. Select the Normally Closed Contact as the Contact Type. Select the CarDetected symbol from the symbol list. Press the OK button to accept the contact definition and close the Edit Contact dialog box. The RLL program should now look like this:



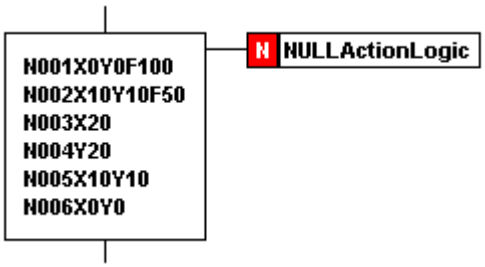
13. Double click on the control menu box on the top left corner of the RLL window to close the RLL window. 

    After filling in RLL transition details, the SFC program should look like the following figure.



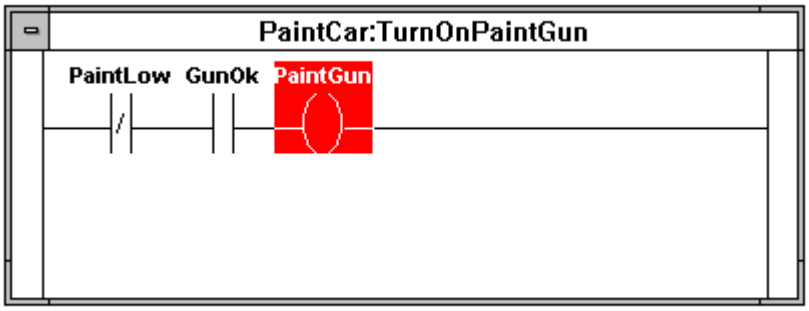
Adding Actions to the SFC

1. Select the action tool from the SFC Tool Bar  .
2. Move the cursor over the top leftmost step. Press the left mouse button to drop an action into the SFC program attached to that step. The step now looks like the following figure.



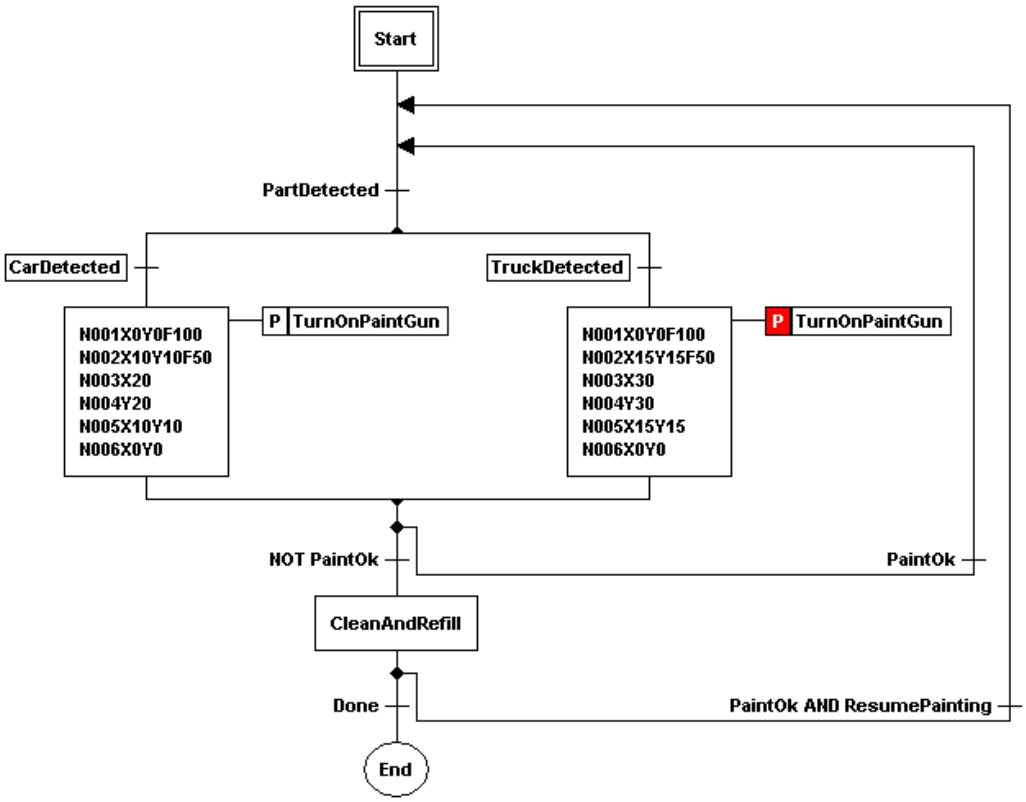
3. Select the selector tool from the SFC Tool Bar  .
4. Double click on the top action name to edit the action. Since this action does not exist a message box is displayed. Press the OK button to open the Edit Action Association dialog box.
5. Select the Action Name edit box. Type in the name for the new action. For the sample program, type in the name "TurnOnPaintGun." Press the OK button to create the new RLL program for the action. A window opens to the right of the action qualifier that contains the RLL program for that action. The SFC Tool Bar is replaced with the RLL Tool Bar.
6. Click anywhere on the SFC program to replace the RLL Tool Bar with the SFC Tool Bar (editing the SFC). Click in the RLL window to replace the SFC Tool Bar with the RLL Tool Bar (editing the RLL). Select the contact tool from the RLL Tool Bar. 
7. Move the cursor over the RLL window, anywhere over the rung. Press the left mouse button to drop a contact onto the RLL rung. The Edit Contact dialog box is opened. Select the Normally Closed Contact as the Contact Type. Scroll the symbol list box down until the PaintLow symbol is visible. Select the PaintLow symbol from the symbol list. Press the OK button to accept the contact definition and close the Edit Contact dialog box.
8. Move the cursor over the rung to the right of the PaintLow contact. Press the left mouse button to drop another contact onto the RLL rung. The Edit Contact dialog box is opened.
9. Select the Normally Open Contact as the Contact Type. Select the Contact Symbol edit box. Type in the symbol name for this contact. In the sample program type in the name "GunOk." Press the OK button to accept the contact definition and close the Edit Contact dialog box. Since the symbol GunOk is a new symbol, the Symbol Manager opens with the new symbol definition prepared as a variable of type BOOL. Press the OK button to close the Symbol Manager. Since the "GunOk" symbol has not yet been added, a message box prompts you to save the symbol. Press the OK button to create the new symbol.

10. Select the coil tool from the RLL Tool Bar  .
11. Move the cursor over the rung to the right of the GunOk contact. Press the left mouse button to drop a coil onto the RLL rung. The Edit Coil dialog box is opened. Select Output Coil as the Coil Type. Scroll the symbol list down until the PaintGun symbol is visible. Select the PaintGun symbol from the symbol list box. Press the OK button to accept the coil definition and close the Edit Coil dialog box.



12. Double click on the control menu box on the top left corner of the RLL window to close the RLL window. 
13. Select the selector tool from the SFC Tool Bar. Double Click on the action qualifier in the small box with an "N" attached to the Action labeled TurnOnPaintGun to open the Edit Action Association dialog box. Select the Pulsed (P) qualifier from the Standard SFC Qualifier list and press the OK button to close the dialog box. Notice that the action qualifier symbol changed from "N" (non-stored) to "P" (pulsed). 
14. Move the cursor over the action. Press and hold the Ctrl key on the keyboard. Press and hold the left mouse button while dragging the mouse to copy the action. Move the cursor over the top rightmost step and release the left mouse button. The same action is now be attached to both steps in the select diverge.

    After adding actions the SFC program should look like the following figure.


Start
PartDetected
CarDetected
TruckDetected
N001X0Y0F100
N002X10Y10F50
N003X20
N004Y20
N005X10Y10
N006X0Y0
P TurnOnPaintGun
N001X0Y0F100
N002X15Y15F50
N003X30
N004Y30
N005X15Y15
N006X0Y0
P TurnOnPaintGun
NOT PaintOk
PaintOk
CleanAndRefill
Done
PaintOk AND ResumePainting
End

Answers to Common Questions

Does the ASIC-200 Software Require a PLC?

- ASIC-200 software does not require a PLC device to do logic execution.
- ASIC-200 software performs logic and motion command execution in the PC without the need for a PLC.
- ASIC-200 software can use PC based scanner cards to control PLC style I/O racks, modules and other devices.

Does the ASIC-200 Software Perform Closed Loop Motion Control?

ASIC-200 software does not execute the low level servo closed loop motion control algorithms because they must be executed very fast (typically less than 1 millisecond). These servo control loops are executed on the motion control I/O cards.

ASIC-200 software does execute the motion control program and generates motion commands to be sent to the motion controller cards.

Does the ASIC-200 Software Perform Parity Checking?

Personal computers can be purchased and configured to provide parity checking on all system memory.

The ASIC-200 software does not do parity checking on application programs, but it does do background checksum calculations on all running application programs. If an error is detected, the application program is stopped, faulted and the operator is allowed to determine what should be done to recover.

What Happens on a Parity Error?

If a parity error is detected in the system, Windows NT will stop and display an error screen with a blue background indicating a Data Bus Error and the location where the error was detected. The system must be rebooted in order to continue.

The ASIC-200 system makes use of Watchdog timers and the Safe State I/O features available on some I/O systems in order to detect loss of ASIC-200

controller activity, and to place the Outputs into a predetermined safe condition. For more information contact ASAP Inc.

What Happens on Power Loss?

The personal computer and the I/O system will go to the power loss state. When power is reapplied, the I/O system will assume the state that it normally assumes or has been configured to assume when power and control is lost. When nonvolatile or battery backed memory is available in the system, ASIC-200 can restart the I/O and application programs from the point of interruption. To restart motion programs, absolute position feedback devices must be provided so that re-homing of the axes is not required.

Can the ASIC-200 Software be Automatically Started?

Windows NT can be configured to start the ASIC-200 software and the application program automatically upon boot up.

What about Uninterruptible Power Supply?

Windows NT can be configured to work with an Uninterruptible Power Supply (UPS) that can supply power to the system when the AC line is lost. Windows NT is notified with an interrupt when the AC line is lost and can be made to take appropriate actions in response. ASIC-200 can also use power failure and low battery signals received from the UPS to perform user programmed actions.

What About Fast I/O with Motion?

ASIC-200 software normally synchronizes I/O and motion by using status reports provided by the motion card. This synchronization can take from one to two I/O scans to occur.

Motion Cards

Most motion cards support:

- Interrupts for homing and fault conditions that can be used to start or stop motion or trigger I/O activity almost immediately
- Dedicated inputs that can capture the position of the axes in less than 1 microsecond from the activation of the trigger input.
- Dedicated I/O images that can be sent to the motion card along with the motion commands.

  This I/O image is transmitted when the specified motion command is started. This can be used to change the state of the dedicated outputs when the axes move through a specified position.

Some motion cards support the execution of logic programs directly on the motion card based on dedicated inputs and outputs. This feature can be used

for more complex applications that require very high speed logic execution along with motion.

How do you Program Motion?

With ASIC-200 software you can program motion in three ways:

- Single Axis Motion or asynchronous motion can be programmed using RLL.
- Single Axis and Coordinated Motion can be programmed in SFC diagrams in step boxes using IEC-1131-3 Structured Text (ST) statements.
- Single Axis and Coordinated Motion can be programmed in SFC diagrams in step boxes using RS-274D (CNC) statements.

Glossary of Terms

.CFG

A file with the extension of CFG is a data file that contains information about the system hardware configuration parameters and symbolic information that refers to physical components and the I/O signals.

Action

Named collection of operations associated with one or more Steps in an SFC.

Action Qualifier

Graphical programming element in an SFC associated with each action block that controls execution of the action logic relative to the period during which the associated Step is active.

Active File

Program file that is contained in the top Program Editor window with it's title bar highlighted. Commands that are executed from the menus or by clicking on buttons on the tool bars are performed on the active file.

ANY

Generic data type that can represent any of the supported data types.

ANY_BIT

Generic data type that can represent these data types: DWORD, WORD, BYTE, BOOL, including an individual bit within these data types.

ANY_INT

Generic data type that can represent the INT, UINT or DINT data type.

ANY_NUM

Generic data type that can represent these data types: ANY_REAL and ANY_INT.

ANY_REAL

Generic type name that can represent the REAL data type.

Application icon

An application icon is a step template with an icon attached. Whenever an application icon is placed in a SFC program, a new step is created and the information in the application icon is copied into that step.

Board

See I/O Card

BOOL

Member of the ANY_BIT group of data types. BOOL data types are valid in any instruction or function block that accepts an ANY, ANY_BIT, or BOOL data type. A BOOL is one bit in length and can have one of two values: TRUE (1, or on) or FALSE (0, or off).

Boolean

Logical element or expression that evaluates to either TRUE or FALSE.

Boolean transition

Type of transition in an SFC represented by an expression consisting of symbols and operators that evaluate to a single Boolean result. If the Boolean result is TRUE, then the SFC transition condition is satisfied.

BYTE

Member of the ANY_BIT group of data types. BYTE data types are valid in any instruction or function block that accepts an ANY, ANY_BIT, or BYTE data type. A BYTE is an unsigned integer data type that is composed of one or more of the digits (0-9) and cannot contain a decimal point. A BYTE is 8 bits in length and has a range of 0 to 255.

Card

See I/O Card

Character Position

Position of a character within a string. The first character in a string starts at position 1.

Coil

RLL graphical programming element that represents a Boolean output symbol.

Configuration File

A Configuration File contains data and parameters describing the systems hardware configuration. This file also contains symbolic identifiers that are used in the application program to refer to the physical hardware devices or I/O signals.

Connector

Group of I/O ports usually routed to one physical connector on a board.

Contact

RLL graphical programming element that represents a Boolean input symbol.

Contact tool

The contact tool is a button on the RLL tool bar that is used to insert an input contact into an RLL program.

Control Loop Element

See Loop.

DATE

Member of the ANY_DATE group of data types. DATE data types are valid in any instruction or function block that accepts an ANY, ANY_DATE, or DATE data type.

DDE

See: Dynamic Data Exchange

DINT

Member of the ANY_NUM group of data types. DINT data types are valid in any instruction or function block that accepts an ANY, ANY_NUM, ANY_INT, or DINT data type. The DINT is a signed integer data type that is composed of one or more of the digits (0-9) and cannot contain a decimal point. A DINT is 32 bits in length and has a range of -2147483648 to +2147483647.

DWORD

Member of the ANY_BIT group of data types. DWORD data types are valid in any instruction or function block that accepts an ANY, ANY_BIT, or DWORD data type. A DWORD is an unsigned integer data type that is composed of one or more of the digits (0-9) and cannot contain a decimal point. A DWORD is 32 bits in length and has a range of 0-4,294,967,295.

Dynamic Data Exchange

DDE is the passage of data between applications, accomplished without user involvement or monitoring. In the Windows environment, DDE is achieved through a set of message types, recommended procedures (protocols) for processing these message types, and some newly defined data types. By following the protocols, applications that were written independently of each other can pass data between themselves without involvement on the part of the user.

FCB

File Control Block. An internal data item that is used to control the File I/O Operations in Structured Text and RLL programs.

First Scan

First Scan Logic is used to execute specific commands one time during the first logic scan of a program. An example may be initializing certain symbols to specific values.

Function

Predefined algorithm that carries out a single operation, such as Square Root, Rotate Left, Rewind File, etc.

Function Block

Predefined algorithm that carries out a single operation, extending the program's capabilities.

HMI

HMI stands for "Human/Machine Interface," and refers to the body of functionality that lets the operator interact with the machine or process controller.

I/O

I/O stands for inputs and outputs to and from the controller.

I/O Card

Plugs into one of the expansion slots of the system unit and connects to the peripheral I/O racks and modules. Also called card or board.

I/O Scanner

I/O scanner is a task within the system that is responsible for retrieving all inputs from and transmitting all outputs to the I/O interface. The I/O scanner updates the internal I/O data tables that are used by the other subsystems.

IEC 1131-3

IEC 1131-3 is an international standard from the International Electrotechnical Commission defining programming languages for Programmable Controllers.

Instruction List

Instruction List is a textual language defined by IEC 1131-3.

INT

Member of the ANY_NUM group of data types. INT data types are valid in any instruction or function block that accepts an ANY, ANY_NUM, ANY_INT, or INT data type. The INT is a signed integer data type that is composed of one or more of the digits (0-9) and cannot contain a decimal point. In an enhancement to the IEC1131-3 specification, the INT is 32 bits in length and has a range of -2,147,483,648 to +2,147,483,647.

Integer

The integer data type, or INT is composed of 32 bits, signed and can represent the values -2,147,483,648 to +2,147,483,648.

Label

Graphical programming element in an SFC or RLL program that identifies where program flow is to resume from its corresponding jump.

Ladder Diagram

Ladder Diagram is graphical programming language defined by IEC 1131 using Relay Ladder Logic concepts.

Loop

Graphical programming element in an SFC diagram that either directs all program execution to continue in the downward direction or to loop back to

some position in the control path above. The Loop Element contains two transition conditions: one directs program flow to continue in the downward direction, and the other directs program flow to loop back. Multiple Loop Elements can be nested within each other, but they can not cross each other and can not enter Select Diverges or Parallel Diverges.

Loop Tool

The Loop Tool is a button on the SFC Edit Toolbar that inserts a control element into an SFC diagram.

Macro Step

Named graphical element in an SFC that represents the inclusion of another entire SFC as a single Step. The included SFC begins execution at its Start Step when the Macro Step that calls it becomes active. Execution in the Macro Step is completed when the included SFC reaches its End Step. Macro Steps can be used to control the complexity or provide a high level view of a larger SFC by breaking the SFC into a collection of smaller and simpler SFCs. Actions can be attached to Macro Steps just as they can be for regular Steps.

Operator Interface

The operator interface is a series of computer screens, messages or windows that are presented to the operator to control and monitor a machine or process.

Parallel Diverge

SFC graphical programming element that splits a control path into two or more parallel paths. When program execution reaches the beginning of a Parallel Divergence, all the subsequent control paths become active in parallel. These control paths continue to be active until all the control paths within a Parallel Diverge reach the point of convergence. At this point, all the paths within the Parallel Divergence are deactivated and the control path below the convergence point will become active.

PID

PID stands for Proportional-Integral-Derivative Controller. The PID is a closed loop process control block that uses the difference between the process set point and a feedback value (process value) from the process being controlled as well as the PID loop gains to generate a control output value that is intended to cause the process value to approach the set point value.

Pointer

Indirect addressing is a common paradigm in programming languages. When using indirect addressing, the symbolic name refers to the location where the data value is stored. These indirect symbols are commonly called pointer symbols. To get the actual data value using indirect addressing, the symbolic name of the pointer symbol is used to obtain the location of the data value, then the location of the data value is used to get the actual data value (an indirect operation).

If pVar1 is a pointer symbol, then in the following assignment, pVar1 is assigned the location of the X data value.

```
pVar1 := & X;
```

If pVar1 is a pointer symbol, then in the following assignment, Y is assigned the value contained in Var1, since pVar1 contains the location of Var1.

```
Y := * pVar1;
```

If pVar1 is a pointer symbol, then in the following assignment,Var1 is assigned the value contained in Y.

```
* pVar1 := Y;
```

Port

I/O port usually consisting of eight I/O bits. A port may be accessed as an integer within programs.

PRGCB

The Program Control Block allows SFC programs to create and control the execution of other application programs.

Program Editor

The Program Editor is a utility used to edit application programs. It consists of editors for SFC and RLL programs.

Project

Organizes or groups the application programs and configuration files for an application in a separate subdirectory.

REAL

Member of the ANY_NUM group of data types. REAL data types are valid in any instruction or function block that accepts an ANY, ANY_NUM, ANY_REAL, or REAL data type. A REAL number data type is composed of one or more of the digits (0-9), is signed, and contains a decimal point. The

range for REAL numbers is: -3.402823 E38 to -1.401298 E-45 (negative), and +1.401298E-45 to +3.402823 E38 (positive).

Relay Ladder Logic

RLL is the graphical programming language used for describing application program logic based on an electrical relay contact and coil analogy.

RLL

RLL stands for Relay Ladder Logic. A graphical programming language using electrical relay contact and coil analogy.

RLL Transition

Transition in an SFC that is programmed using Relay Ladder Logic. The RLL transition consists of a single RLL rung with an output coil that has the same name as the RLL transition. When this output coil has power flow, the SFC transition condition is satisfied.

Rung

RLL graphical programming element that represents an RLL function.

Runtime Engine

Module responsible for scheduling and executing the program logic associated with the project's source code, e.g., SFC, RLL, etc. This module also performs as a server to DDE clients.

Select Diverge

SFC graphical programming element that splits a single control path into two or more paths. The control path selected for execution is determined by the transition conditions that are located at the beginning of each of the new control paths.

Sequential Function Chart

SFC is the graphical programming language for diagramming sequential logic using Steps, Transitions and Actions.

SFC

SFC stands for Sequential Function Chart which is used as a graphical programming language for diagramming sequential logic using steps, transitions, and actions. The IEC 1131 has defined a specific implementation of SFC for use with PLCs.

SFC Transition Coil

SFC graphical programming element that can be associated with a Step or a Macro Step and provide program flow control. Associated with a Step, the SFC transition coil can cancel the Step and direct program flow to another Step. Associated with a Macro Step, the SFC transition coil can cancel the child SFC and direct program flow to another Step in the parent SFC.

SFC+

SFC+ is an enhanced version of the IEC 1131 Sequential Function Chart standard programming language used by the ASIC-100 software.

Simultaneous Diverge

Simultaneous diverge is a SFC graphical programming elements that splits a control path into two or more parallel control paths. When program execution reaches the beginning of a simultaneous divergence all subsequent control paths will become active in parallel. These control paths will continue to be active until all control paths with in the simultaneous divergence reach a point of convergence at which time the paths will be deactivated and the control path below the convergence will become active.

Slot

Slot refers to a space in an I/O rack or a computer in which a card is placed.

Step

Named graphical element in an SFC that represents a state or span of time in the program execution during which the actions and functions associated with the Step are performed.

Step Name

The Step Name is an identifier that refers to a step in a Sequential Function Chart. The Step name can be changed by double clicking on the Step box to open the Edit Step dialog box, then double clicking the identifier in the Step Name edit box and entering a new identifier.

STRING

Member of the ANY group of data types. STRING data types are valid in any instruction or function block that accepts an ANY or STRING data type. The format for a STRING data type consists of a string of up to 250 ASCII characters in single quotation marks.

Structured Text

Structured Text is a textual programming language defined by IEC 1131.

Symbol

A Symbol is an internal memory location that contains information. The data type defines the content of the information (i.e. real, integer, string etc.). The Symbol Manager is used to define a symbol and to assign it a symbolic name and data type.

TIME

TIME is a member of the ANY group of data types. TIME data types are valid in any instruction or function block that accepts an ANY or TIME data type. The format of the TIME data type consist of a T# or t# followed by a sequence of one or more numbers and time unit specifiers. Examples:

T#1D2h = 1 day and 2 hours
t#26H = 26 hours
t#5m45s = 5 minutes and 45 seconds
t#26S200MS = 26 seconds and 200 milliseconds
T#900ms = 900 milliseconds

Time of Day

The Time of Day data type, or TOD, is used to represent a specific time of day. IEC-1131-3 uses the format TOD#HH:MM:SS.ms to designate that the following characters will be a time of day. Ex. TOD#23:59:59.999

TMR

The Timer data type, or TMR is used to implement a timer using structured text.

TOD

Member of the ANY_DATE group of data types. TOD (Time of Day) data types are valid in any instruction or function block that accepts an ANY, ANY_DATE, or TOD data type.

Transition Condition

A transition condition is a logical expression associated with an SFC transition element resulting in a single Boolean result. The Boolean result is used to determine when activation is passed from the active step to the step following the transition.

Transition Logic

Transition logic can be represented by a single Boolean expression or single Relay Ladder Logic rung.

Transition Mode

The Transition Mode Button on the SFC toolbar determines what type of transition is inserted when using the Transition Insertion Tool. If the Transition Mode button is not depressed an RLL transition is inserted. If the Transition Mode button is depressed a Boolean transition is inserted.

UINT

Member of the ANY_INT group of data types. UINT data types are valid in any instruction or function block that accepts an ANY, ANY_INT, or UINT data type. A UINT is an unsigned integer data type that is composed of one or more of the digits (0-9) and cannot contain a decimal point. A UINT is 16 bits in length and has a range of 0 to 65535.

WORD

Member of the ANY_BIT group of data types. WORD data types are valid in any instruction or function block that accepts an ANY, ANY_BIT, or WORD data type. A WORD is an unsigned integer data type that is composed of one or more of the digits (0-9) and cannot contain a decimal point. A WORD is 16 bits in length and has a range of 0 to 65535.

Index

137586(D)

Xycom Automation, LLC
750 North Maple Road
Saline, MI 48176
http://www.profaceamerica.com

Phone: 734-429-4971
Fax: 734-429-1010