SoMachine Basic

Training Manual

SoMachine Basic Version 1.1

Schneider Electric

DISCLAIMER

Schneider Electric makes no representations or warranties with respect to this manual and, to the maximum extent permitted by law, expressly limits its liability for breach of any warranty that may be implied to the replacement of this manual with another. Furthermore, Schneider Electric reserves the right to revise this publication at any time without incurring an obligation to notify any person of the revision.

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information that is contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

SoMachine Basic Training Manual

INTRODUCTION AND LEGAL NOTICE

Your purchase of this official SoMachine Basic Training Manual entitles you to undertake the SoMachine Basic training course.

Satisfactory completion of the course evaluation is mandatory for you to obtain a Schneider Electric certificate of completion of the training course.

Schneider Electric will not accept any liability for action taken in reliance on this training manual.

TRADEMARKS

Schneider Electric has made every effort to supply trademark information about company names, products and services mentioned in this manual. Trademarks shown below were derived from various sources.

Microsoft Windows, Windows XP, Windows Vista, Windows 7, Windows 8, Microsoft Office and Microsoft Excel are either registered trademarks or trademarks of Microsoft® Corporation in the United States and/or other countries.

General Notice:

Some product names used in this manual are used for identification purposes only and may be trademarks of their respective companies.

Validity Note

The present documentation is intended for qualified technical personnel responsible for the implementation, operation and maintenance of the products described. It contains information necessary for the proper use of the products.

About Us

Members of Schneider Electric's team of Instructional Designers have tertiary qualifications in Education, Educational Course Development and are also experienced Instructors. Currently, the team is supporting a range of Schneider Electric courses in multiple languages and multiple software environments.

Authors

Richard Irons

March 2014 Edition for SoMachine Basic Version 1.1
Manual Release 1.1

Safety Information

Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a Danger or Warning safety label indicates that an electrical hazard exists, which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety alert messages that follow this symbol to avoid possible injury or death.

⚠ DANGER

DANGER indicates an imminently hazardous situation which, if not avoided, **will result in** death or serious injury.

⚠ WARNING

WARNING indicates a potentially hazardous situation which, if not avoided, **can result in** death or serious injury.

⚠ CAUTION

CAUTION indicates a potentially hazardous situation which, if not avoided, **can result in** minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

Safety Information (cont.)

Important Information (cont.)

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and the installation, and has received safety training to recognize and avoid the hazards involved.

Before the Course Begins

Scope of this Training Manual

This training manual is a supplement to the authorised training. In order to make proper use of the software students should also refer to the documentation that has been provided with the product such as the Help Files, User Guides or Knowledge Base.

The graphics displaying screen captures were taken using the Windows® XP operating system using Classic mode display properties. If students are running a different version of Windows then screen images may differ slightly from those shown in the training manual.

Some screen captures may have been taken from beta versions of the software and may vary slightly from release screen captures.

Product Related Information

⚠ DANGER

HAZARD OF ELECTRIC SHOCK, EXPLOSION OR ARC FLASH

- Disconnect all power from all equipment including connected devices prior to removing any covers or doors, or installing or removing any accessories, hardware, cables, or wires except under the specific conditions specified in the appropriate hardware guide for this equipment.
- Always use a properly rated voltage sensing device to confirm the power is off where and when indicated.
- Replace and secure all covers, accessories, hardware, cables, and wires and confirm that a proper ground connection exists before applying power to the unit.
- Use only the specified voltage when operating this equipment and any associated products.

Failure to follow these instructions will result in death, serious injury, or equipment damage.

Product Related Information (cont.)

⚠ WARNING

LOSS OF CONTROL

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines[1].
- Each implementation of this equipment must be individually and thoroughly tested for a proper operation before being placed into service.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

[1] For additional information, refer to NEMA ICS 1.1 (latest edition), "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control" and to NEMA ICS 7.1 (latest edition), "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Before the Course Begins (cont.)

User Responsibilities

The products specified in this document have been tested under actual service conditions. Of course, your specific application requirements may be different from those assumed for this and any related examples described herein. In that case, you will have to adapt the information provided in this and other related documents to your particular needs. To do so, you will need to consult the specific product documentation of the hardware and/or software components that you may add or substitute for any examples specified in this training documentation. Pay particular attention and conform to all safety information, different electrical requirements and normative standards that would apply to your adaptation.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

The use and application of the information contained herein require expertise in the design and programming of automated control systems. Only the user or integrator can be aware of all the conditions and factors present during installation and setup, operation, and maintenance of the machine or process, and can therefore determine the automation and associated equipment and the related safeties and interlocks which can be effectively and properly used. When selecting automation and control equipment, and any other related equipment or software, for a particular application, the user or integrator must also consider any applicable local, regional or national standards and/or regulations.

Before the Course Begins (cont.)

User Responsibilities (cont.)

Some of the major software functions and/or hardware components used in the examples described in this training document cannot be substituted without significantly compromising the performance of your application. Further, any such substitutions or alterations may completely invalidate any proposed architectures, descriptions, examples, instructions, wiring diagrams and/or compatibilities between the various hardware components and software functions specified herein and in related documentation. You must be aware of the consequences of any modifications, additions or substitutions. A residual risk, as defined by EN/ISO 12100-1, Article 5, will remain if:

- it is necessary to modify the recommended logic and if the added or modified components are not properly integrated in the control circuit.
- you do not follow the required standards applicable to the operation of the machine, or if the adjustments to and the maintenance of the machine are not properly made (it is essential to strictly follow the prescribed machine maintenance schedule).
- the devices connected to any safety outputs do not have mechanically-linked contacts.

⚠ CAUTION

EQUIPMENT INCOMPATIBILITY

Read and thoroughly understand all device and software documentation before attempting any component substitutions or other changes related to the application examples provided in this document.

Failure to follow these instructions can result in injury or equipment damage.

Before the Course Begins (cont.)

Start-up and Test

When applying this training and before using electrical control and automation equipment after design and installation, the application and associated functional safety system must be subjected to a start-up test by qualified personnel to verify correct operation of the equipment. It is important that arrangements for such testing be made and that enough time is allowed to perform complete and satisfactory testing.

⚠ CAUTION

EQUIPMENT OPERATION HAZARD

- Verify that all installation and set up procedures have been completed.
- Before operational tests are performed, remove all blocks or other temporary holding means used for shipment from all component devices.
- Remove tools, meters and debris from equipment.

Failure to follow these instructions can result in injury or equipment damage.

Verify that the completed system, including the functional safety system, is free from all short circuits and grounds, except those grounds installed according to local regulations. If high-potential voltage testing is necessary, follow the recommendations in equipment documentation to help prevent injury or equipment damage.

Before the Course Begins (cont.)

Operation and Adjustments

Regardless of the care exercised in the design and manufacture of equipment or in the selection and ratings of components, there are hazards that can be encountered if such equipment is improperly installed and operated.

In some applications, such as packaging machinery, additional operator protection such as point-of-operation guarding must be provided. This is necessary if the hands and other parts of the body are free to enter pinch points or other hazardous areas where serious injury can occur. Software products alone cannot protect an operator from injury. For this reason, the software cannot be substituted for or take the place of point-of-operation protection.

⚠ WARNING

UNGUARDED MACHINERY CAN CAUSE SERIOUS INJURY

- Do not use this software and related automation equipment on equipment which does not have point-of-operation protection.
- Do not reach into machinery during operation.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Ensure that appropriate safeties and mechanical/electrical interlocks related to point-of-operation protection have been installed and are operational before placing the equipment into service. All interlocks and safeties related to point-of-operation protection must be coordinated with the related automation equipment and software programming.

 **Note:**

Coordination of safeties and mechanical/electrical interlocks for point-of-operation protection is outside the scope of the examples and implementations suggested in this training documentation.

It is sometimes possible to adjust the equipment incorrectly and may produce unsatisfactory or unsafe operation. Always use the manufacturer instructions as a guide to functional adjustments. Personnel who have access to these adjustments must be familiar with the equipment manufacturer instructions and the machinery used with the electrical equipment.

Course Overview

Course Objectives

By the completion of this training course the student will be able to:

- Understand the SoMachine Basic software
- Use SoMachine Basic to program the M221 range of Logic Controllers
- Create an application
- Program an application
- Faultfind an application
- Document an application

Target Audience

The SoMachine Basic training course is aimed at new users of SoMachine Basic and the M221 Logic Controller.

Prerequisite Knowledge

It is expected that readers of this document will know how to operate a computer and have a general understanding of control techniques but no previous knowledge of SoMachine Basic is required.

Course Overview (cont.)

Support

If support or additional information about any concepts or products in the course is required, students should ask the Instructor who will either address the question or obtain additional technical assistance as required.

Conventions Used in this Manual

Objectives

These are the skills to be achieved by the end of each chapter. An overview providing a brief synopsis of the topic begins each section. Often, examples are given to illustrate the conceptual overview.

Example -

The configuration environment consists of several toolbars, browser windows and programming editors. This chapter introduces the user to the configuration environment using an example project with pre-defined elements.

This Chapter Covers These Topics:

Exercises

After a concept is explained students will be given exercises that practice the skills just learned. These exercises begin by explaining the general concept of each exercise and then step-by-step procedures are listed to guide students through each exercise.

Example -

Paste an object from a library onto a test page called **Utility**.

1 Run the Milk_Upgrade project then trigger and view some alarms.

i. Use the following template settings:

User Input

Whenever information is to be typed into a field or dialog box it will be written in this font:

```
KETTLE_TEMP/25
```

Note that some exercises will show a fragment of information already typed into a SoMachine Basic screen and then ask students to add extra lines of configuration. In this instance, the previously entered material will be given to the student as light grey italic text.

KETTLE_TEMP/25

```
OVEN_TEMP/5
```

Conventions Used in this Manual (cont.)

Hints & Tips

This heading will provide students with useful or helpful information that will make configuring the project easier.

Example -

Hints & Tips:

To go to the next field, use the mouse cursor or press the **TAB** key.

Note

A note will refer to a feature which may not be obvious at first glance but something that should always be kept in mind.

Example -

Note:

Any events named **GLOBAL** are enabled automatically when events are enabled.

Menus and Menu Options

Text separated by the double arrow symbol "**»**" indicates that students are to select a menu.

Example -

File » New…

Open a menu "**File**" then select the menu option "**New…**"

Horizontal and Vertical Tabs

Text written this way indicates the **Horizontal** then the **(Vertical)** tab is to be selected.

Example -

Appearance (General)

Conventions Used in this Manual (cont.)

See Also

Text written in this way indicates further references about the current topic.

Example -

See Also:

For further information about **Templates,** see S*oMachine Basic Help - Using Page Templates*.

Further Training

This heading describes topics that are covered in more advanced courses.

Example -

Further Training:

Trend Table Maths is a topic in the **Customisation and Design Course**.

Table of Contents

Table of Contents (cont.)

Chapter 1: Introduction

Overview

Introduction

SoMachine Basic was written specifically for programming the M221 Logic Controller and is based on the full version of the SoMachine software.

This chapter gives an overview of the SoMachine Basic software and the M221 Logic Controller.

This Chapter Covers These Topics:

What is SoMachine Basic?

Software

SoMachine Basic is software created to program the Machine Solutions M221 Logic controller.

The SoMachine programming software is used for programming Machine Solutions Logic Controllers and has evolved over several years. The M221 Logic Controller is a new, low-cost unit aimed at simple machine applications; it can very easily be used in place of a Twido PLC. SoMachine is an expensive programming solution for this so SoMachine Basic was produced as a cut-down version of SoMachine. It does not have the functionality of the full SoMachine software but this free-issue software can be used for the M221 controller.

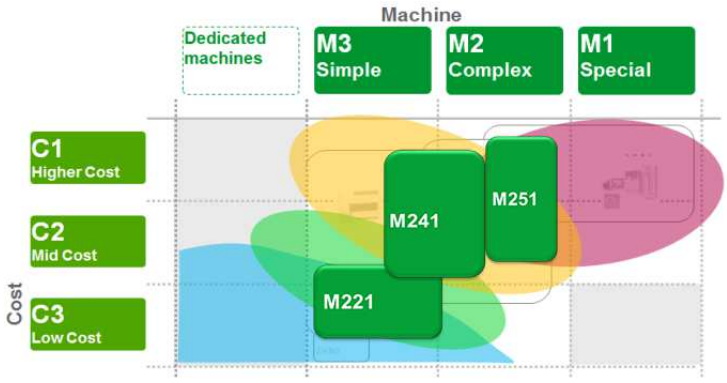
Note:

SoMachine Basic can only be used to program the M221 Logic Controller. Any other Logic Controller will require the full SoMachine Software.

Machine Solutions Hardware

M2xx Logic Controllers

The M2xx logic controllers form the basis of Machine Solutions control systems. There are 3 logic controllers; the M221, M241 and M251. Selection will depend on the required complexity of the application and cost. Generally, as the complexity increases so does the cost.



SoMachine Basic is only able to program the M221 logic controller. The full version of the SoMachine programming software must be used to program the M241 and M251.

The M221 Logic Controller

M221 Logic Controllers

The M221 Logic Controller is a small, low-cost controller for simple machine applications. AM221 Logic Controllers are powered by either 24VDC or 100-240VAV and have a modular design allowing the addition of further analog, digital and specialised I/O.

They are fitted with one or two RJ10 connector serial ports and a USB port for application transfer.

A SD card slot is located under the central cover to allow data transfers and firmware update via SD card.

Information and diagnostic LEDs are also fitted on the front panel.

Common Components

There are several common options or components that are present on all M221 Logic Controllers.

- Serial Port for data transfer
- USB connector for programming
- 2 independent 0-10V Analog Inputs
- SD Card slot for data transfer
- LED Information display
- On-board I/O
- High speed counter inputs
- I/O Expansion

The internal workings of all M221 Logic Controller are also basically the same with just a few different components for the options. This means that the performance will be the same across the entire M221 product range.

The M221 Logic Controller (cont.)

Optional Components

There are a number of options for the M221 Logic Controller.

- A range of input and output combinations
- 0.1A transistor outputs or 2A relay outputs
- Ethernet connection for programming and data transfer
- Removable terminal block or HE10 connector
- Screw or spring connectors for the removable terminal blocks
- Pulse generator to provide pulsed or Pulse Width Modulation (PWM) control

Some are exclusive, for example the 16 input/16 output units can only be supplied with transistor outputs.

For a full list of available units and options, see C*hapter 2 - M221 Hardware*.

I/O Expansion

The M221 comes with some I/O built-in but this may not be enough to meet the needs of the application.

If more I/O is required, expansion modules can be added to the M221 Logic Controller to provide more I/O. These expansion modules can be either the existing TM2 modules or a new range of TM3 modules. Modules can be either digital or analog and provide additional inputs, outputs or a combination of both.

Compatibility with the TM2 range of I/O means that it is very easy to replace a Twido with a M221 Logic Controller.

SoMachine Basic Software Installation

System Requirements

To install SoMachine Basic the following computer specification is required:

- Windows 8/7/XP
- 32bit or 64bit
- Microsoft .NET version 4.0
- 1GB RAM
- 200MB Hard disk space
- Dual core processor

Installation

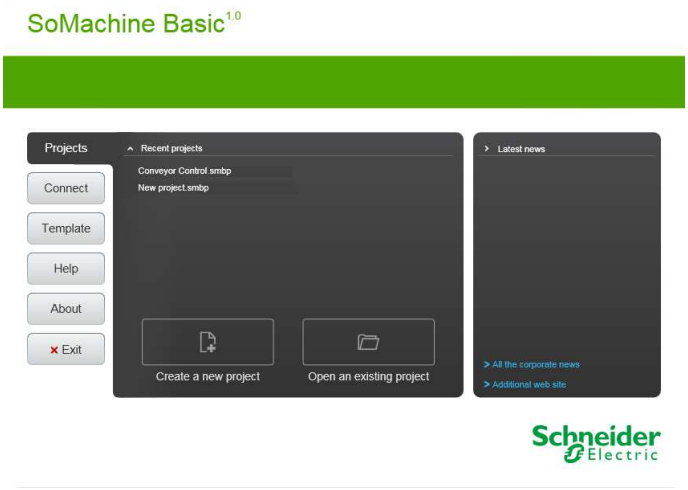
SoMachine Basic installation comes as an executable file. Simply run the executable to install the software. The installer for SoMachine Basic will uninstall any previous version of SoMachine Basic before the installation.

If SoMachine Basic is updated to a newer version, the firmware in the M221 controller must also be updated.

For further details, see the installation guide located on the CD and firmware update in Chapter 2 - Firmware Update.

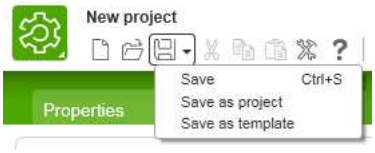
Managing Applications

When SoMachine Basic starts, the start page is displayed with the option to open an existing project or create a new one. Recent projects are also listed to allow easy access to these.



Click any listed project to open it or click the **Open an existing project** button to navigate to a folder and open a project. Click the **Create a new Project** button to create a new project.

Projects can be saved by clicking the **Save** button on the toolbar or saved as a new file by dropping down the menu and selecting **Save as project** from the menu.



Exercise - Managing Applications

Learning Outcomes

By the completion of this exercise you will:

- Create a new application
- Save and load an application

1 Start the SoMachine Basic software.

i. Either

Double-click the SoMachine Basic icon on the desktop.

or

Go to Start » All Programs » Schneider Electric » SoMachine Basic » SoMachine Basic

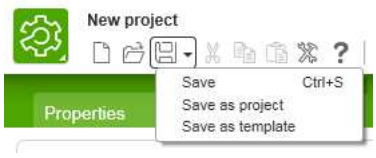
2 Create a new project.

i. From the startup page click the **Create a new project** button.

ii. The new project will be opened with a controller already configured.

3 Save the project

i. Drop down the menu next to the Save button on the toolbar.



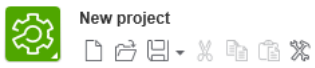
There are options to Save and Save as. Choose **Save** from the menu. The save can also be done using the standard key combination <Ctrl>S.

ii. Give the project an appropriate name. One that describes the application will be helpful when trying to find an application file later.

Exercise - Managing Applications (cont.)

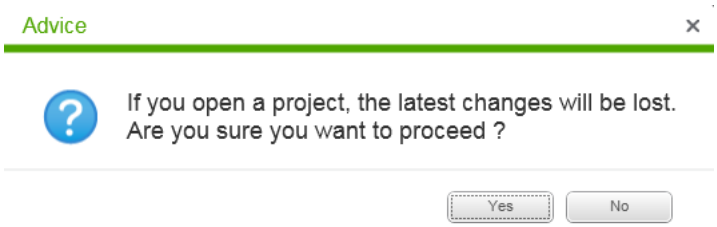
4 Load a project.

i. Click the SoMachine Basic icon in the top left hand corner to display the start page.



ii. The file just saved will be displayed. Click the file to open it.

If a message box appears stating that the latest changes will be lost and asking if you want to proceed click the **Yes** button to open the project.



The Configuration Environment

Application Configuration

The main application configuration area consists of four tabs. These are:

- Properties
- Configuration
- Programming
- Commissioning

These sections provide a logical structure and make the various configuration tools easy to find. Working from left to right creates a structured approach to creating the application.

Properties

The Properties section allows information to be recorded about the application including the programmer and where it is being used.

- The Front Page Section allows the creator of the application to enter contact information. This is useful to the customer as if the application needs modifying as it provides a first point of contact.
- The Company section allows information about the company to be entered. This information can be useful to an integrator to identify who the application was created for.
- The project information section gives information about the project itself. This is useful for both integrators and end users who have multiple projects for a site as it will identify the application and where it is used.

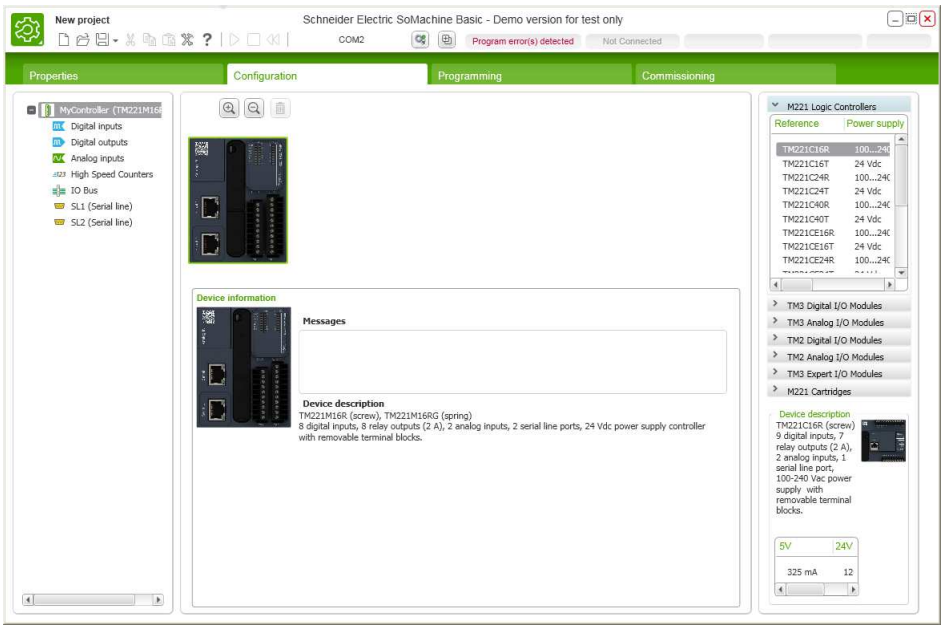
A photo can also be included which can be useful to show the area where the application is being used.

This section also contains areas where the project can be protected from viewing or changing. This can be useful for OEMs and Integrators who wish to protect their intellectual property.

The Configuration Environment (cont.)

Configuration

The configuration page is used to configure the hardware used in the project. This will include selecting the correct controller and additional I/O modules, and configuring the parameters for the hardware.



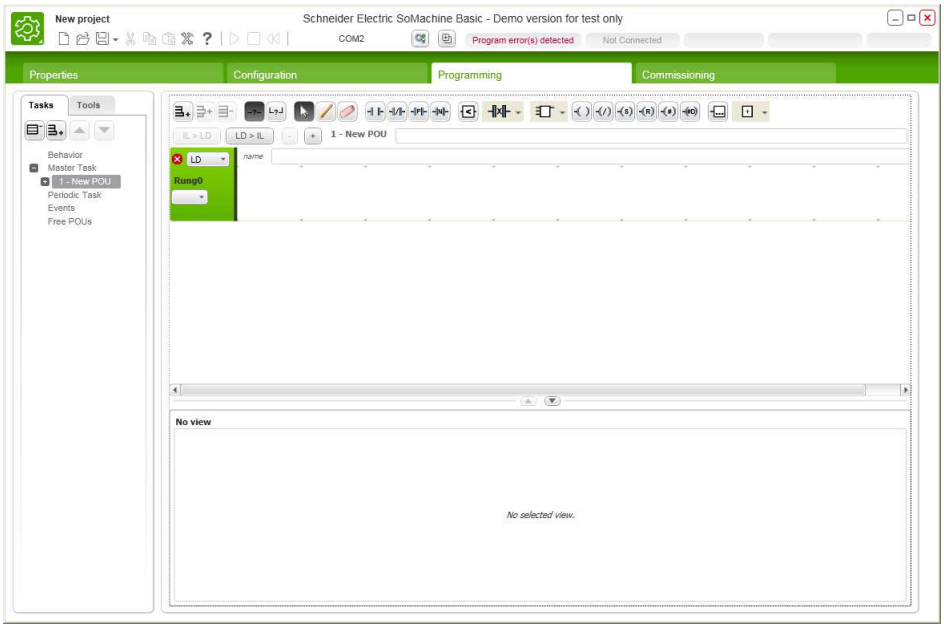
The SoMachine Basic software has drag-and-drop hardware configuration. Controllers and modules are listed in a catalog allowing the modules to be selected and dropped into the configuration area. Modules may be reordered by simply dragging them to the new position.

When the required modules have been added, they can be selected and a list of configurable sections is shown in the hardware tree allowing parameters to be configured.

The Configuration Environment (cont.)

Programming

The programming tab is where the control program is actually written. This can currently be done using either Instruction List or Ladder. SoMachine Basic provides a way to program in either of these languages and switch easily between the two. It also provides a way of separating sections of program into a logical structure to make the program easy to follow.



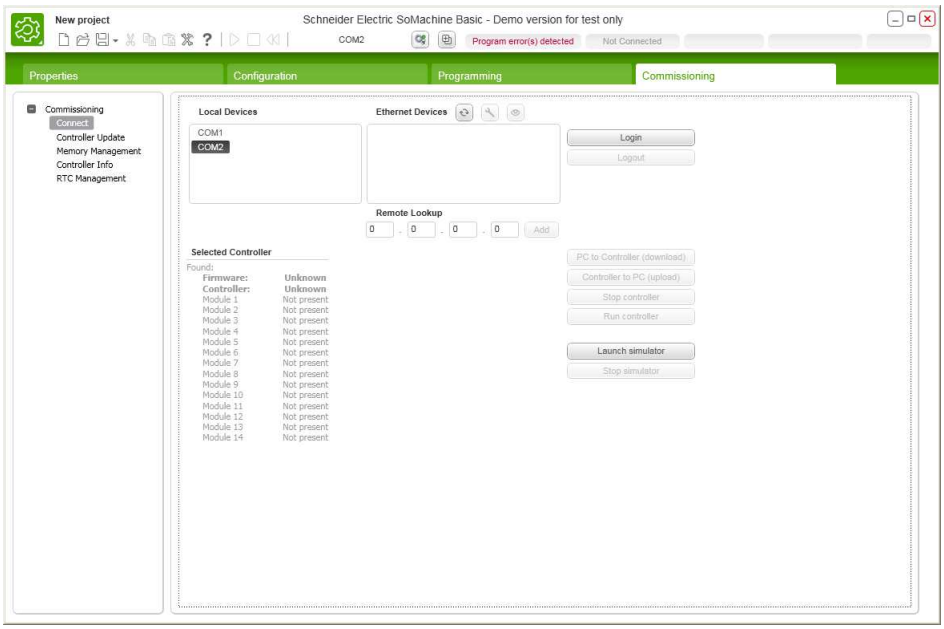
This section can also be used for debugging the application when SoMachine Basic is connected to the M221 Logic Controller.

The Programming section is covered in more detail in *Chapter 3 - Programming a Simple Application.*

The Configuration Environment (cont.)

Commissioning

When an application has been created, SoMachine basic can be connected to a M221 Logic Controller and the application sent to the controller. This connection can be made by serial, USB or Ethernet. Before the application is downloaded, the program is checked for programming errors to ensure that it is not run in the controller with an error present.



Various tools are then available to check the operation of the program and that the application is working correctly. These include in-line monitoring of values in the program, animation tables and the ability to change the values of objects in the program.

The Commissioning section is covered in more detail in the *Chapter 4 - Getting it all Working.*

The Configuration Environment (cont.)

Status Bar

The status bar shows information about the connection between SoMachine Basic and the M221 Logic Controller.

It shows whether there is a program error, connection status, controller status, scan time and the last error. It also shows whether the controller is running or stopped.

The information shown here can be useful for debugging applications.

Customising the Software

Accessing the Settings

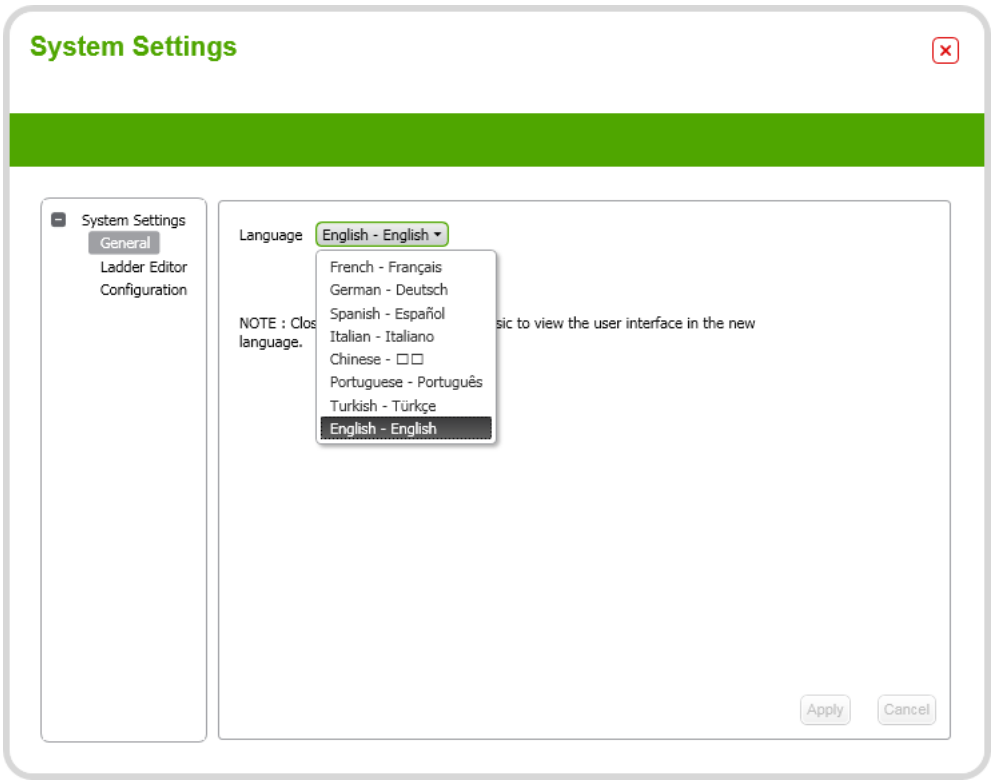
To access the settings, click the settings button on the SoMachine Toolbar.

There are three sections in the settings. These are:

- General
- Ladder Editor
- Configuration

General Settings

The General Settings allow the language of SoMachine Basic to be changed. Current options are English, French and Italian.



After the language has been changed, SoMachine Basic must be restarted.

Settings (cont.)

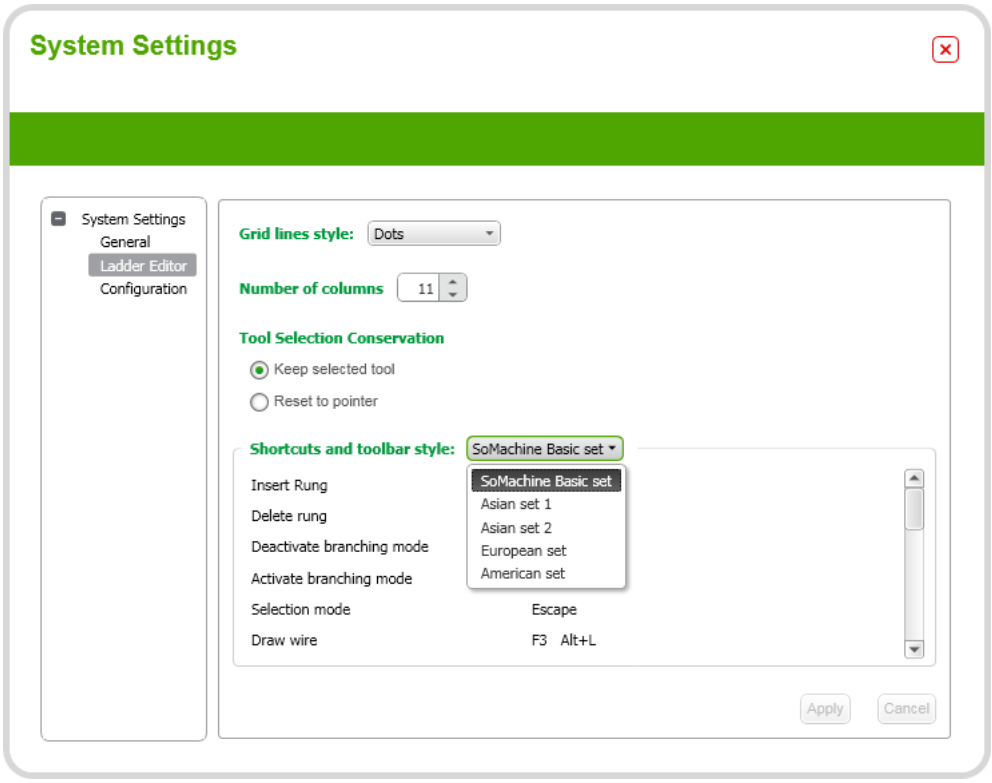
Ladder Editor Settings

The Ladder Editor Settings configure the behaviour of the ladder editor.

The ladder editor grid style can be changed to dots, dashed lines or solid lines

The number of columns in the ladder editor can be chosen by setting a value from 11 to 30.

The tool selection setting determines what happens in the ladder editor when an object has been placed on a rung. Keep selected tool will allow multiple objects of the same type to be placed without having to reselect the object from the toolbar. This is useful for placing multiple contacts. Reset to pointer will place one object then reset to the pointer so another action can be carried out.

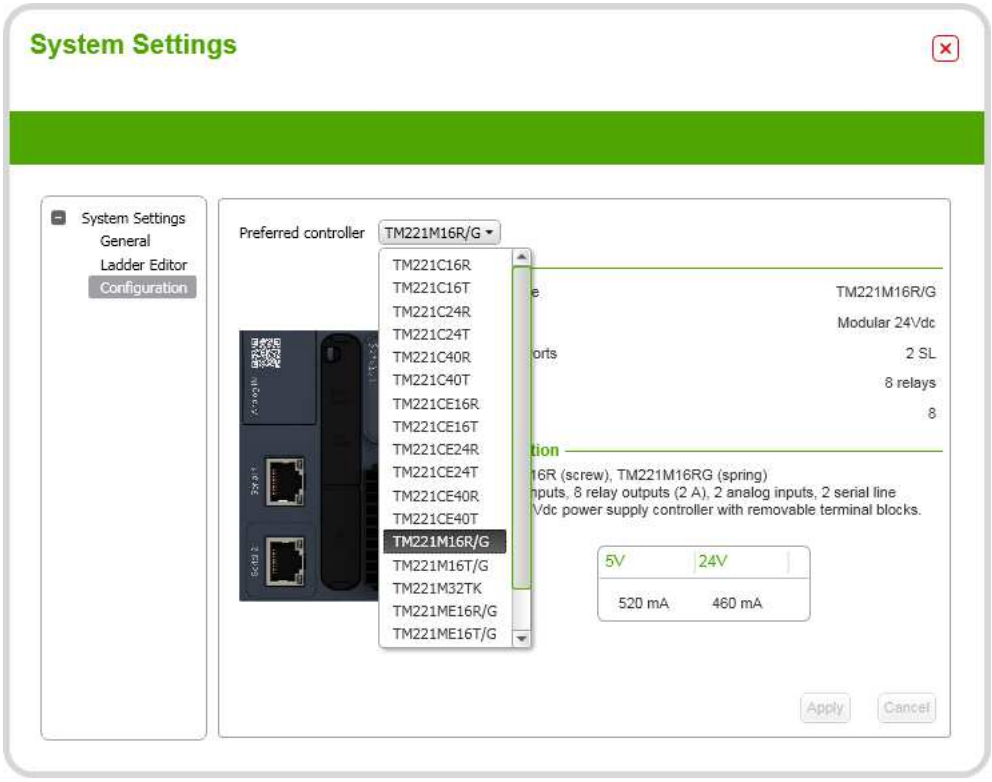


The shortcut and toolbar style can be changed to either the standard SoMachine Basic style or an Asian/European/American style. These alternative styles are similar to other programming software shortcuts and toolbars making programming in SoMachine Basic easier for engineers who are used to other programming tools.

Settings (cont.)

Configuration

The Configuration setting allows the default controller to be chosen. This controller will then be used whenever a new application is created.



This is useful for OEMs who are creating similar applications and will use the same controller most of the time.

Summary

Summary

This chapter covered the following topics:

- ***What is SoMachine Basic?*** (page 1-2)
- ***The M221 Logic Controller*** (page 1-4)
- ***The SoMachine Basic Software*** (page 1-7)
- ***The Configuration Environment*** (page 1-10)
- ***Customising the Software*** (page 1-15)

Questions

The following questions are to check understanding of the topics covered in this chapter:

- Name the four configuration sections for SoMachine Basic

- SoMachine Basic can be used to program which Machine Solutions Logic Controllers?

- What are the three ways to connect to the Logic Controller for programming and commissioning?

Chapter 2: M221 Hardware

Overview

Introduction

The M221 Logic Controller and its expansion modules is the hardware that is configured and programmed with SoMachine Basic. This chapter looks at this hardware as a foundation for creating an application.

This Chapter Covers These Topics:

The M221 Logic Controller

M221 Controllers

The M221 Logic Controller is a small controller containing on-board I/O. It has serial and USB connectors and an Ethernet option. A SD card slot allows data transfer and a Run/Stop switch is fitted for program control. The unit also has two 0-10V analog inputs.

"

Power supply for all M221 Brick Controllers is by 24VDC connected at the bottom of the unit. The standard M221 Logic Controller can be powered by either 24VDC or 100-240VAC.

Capabilities

The M221 Logic Controller has many features that differentiate it from other controllers on the market. These include:

- Small size - 70mm with 32 I/O
- 5K instructions per millisecond
- High speed counters and pulse output
- SD card for data transfer
- On-line modification
- Real Time Clock
- .csv export/import of object names and descriptions
- Program via USB or Ethernet (where fitted)
- Serial port for Modbus or ASCII communication
- Two built-in analog inputs

These features mean that the M221 is highly flexible for most small applications. It is also easily configurable using SoMachine Basic.

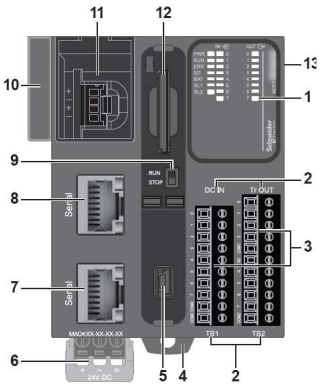
The M221 Logic Controller (cont.)

Features of the M221

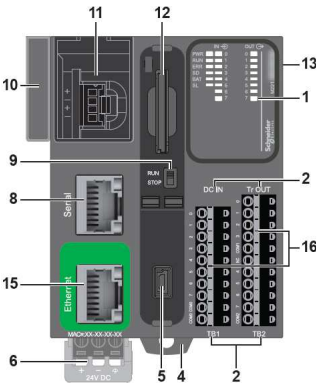
The appearance of the M221 Logic Controller will depend on the connector type and whether it has an Ethernet port. Three of the configurations are shown below.

Screw Connector

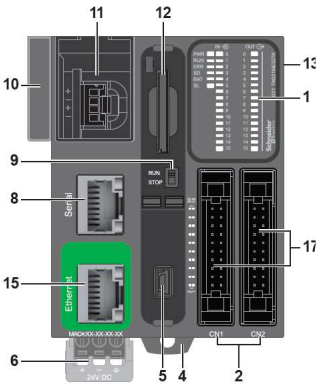
No Ethernet



Spring Connector



HE10 Connector



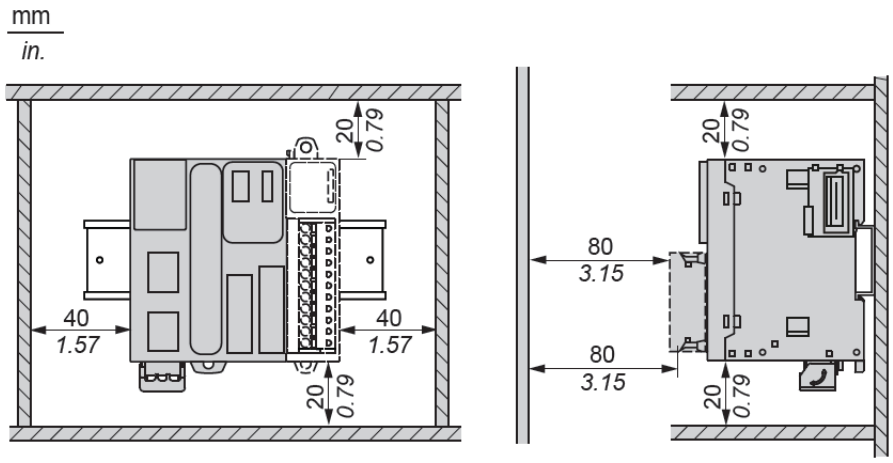
1	System LEDs	10	Analog input cover
2	I/O label information	11	2 analog inputs
3	I/O removable screw terminals	12	SD card port
4	Clip lock for 35mm DIN rail	13	TM3 bus connector
5	USB mini-B programming port	14	Plastic cover
6	24V DC power supply	15	Ethernet port
7	Serial line port 2	16	I/O removable spring terminals
8	Serial line port 1	17	I/O HE10 terminals
9	Run/Stop switch		

The pictures and table above shows the M221 Book Controller but the M221 Logic Controller shares the same basic features.

The M221 Logic Controller (cont.)

Panel Mounting

The M221 should be mounted on a horizontal 35mm DIN or top-hat rail with at least 20mm spacing above and below and at least 40mm spacing on either side. There should be at least 80mm spacing between the front of the M221 Logic controller and the panel door.



The plug-in connectors and removable I/O terminal blocks mean that the M221 can be removed and replaced easily without having to remove individual cables.

M221 Selection

Product Code

The product code defines the options for the M221 Logic Controller and the fitted options can be determined from the code.

- The first five characters of the part number (TM221) define that it is a M221 Logic Controller
- The next character defines whether the controller is a M221 logic controller (C) or a M221 book controller (M)
- If this is followed by the letter 'E' the controller is fitted with an Ethernet port
- The next two numbers define the total amount of digital I/O.
- The next character defines whether the outputs are (R)elay or (T)ransistor
- For book controllers a G on the end indicates spring I/O connectors instead of screw connectors
- For book controllers a K on the end indicates HE10 connectors which must be used with the high density 32 I/O units due to space limitations.

M221 Selection (cont.)

M221 Logic Controller Selection

There are several options for the M221 Logic Controller to allow for the requirements of the application. These requirements will determine the number of inputs and outputs, the type of power supply and whether an Ethernet connection is required.

Controller	Inputs	Outputs	Power	Ethernet
TM221C16R	9	7	100-240VAC	No
TM221C16T	9	7	24VDC	No
TM221C24R	14	10	100-240VAC	No
TM221C24T	14	10	24VDC	No
TM221C40R	24	16	100-240VAC	No
TM221C40T	24	16	24VDC	No
TM221CE16R	9	7	100-240VAC	Yes
TM221CE16T	9	7	24VDC	Yes
TM221CE24R	14	10	100-240VAC	Yes
TM221CE24T	14	10	24VDC	Yes
TM221CE40R	24	16	100-240VAC	Yes
TM221CE40T	24	16	24VDC	Yes

M221 Selection (cont.)

M221 Selection

There are several options for the M221 Book Controller to allow for the requirements of the application. These requirements will determine the number of inputs and outputs, the type of I/O connection and whether an Ethernet connection is required.

Controller	Inputs	Outputs	Connection	Ethernet
TM221M16R	8	8 Transistor	Screw	No
TM221M16RG	8	8 Transistor	Spring	No
TM221M16T	8	8 Relay	Screw	No
TM221M16TG	8	8 Relay	Spring	No
TM221M32TK	16	16 Transistor	HE10	No
TM221ME16R	8	8 Transistor	Screw	Yes
TM221ME16RG	8	8 Transistor	Spring	Yes
TM221ME16T	8	8 Relay	Screw	Yes
TM221ME16TG	8	8 Relay	Spring	Yes
TM221ME32TK	16	16 Transistor	HE10	Yes

M221 Connectivity

Connectivity

The M221 has several connection options which can be split into two types: I/O for process control and Data/Programming connections.

The I/O provided with the M221 will depend on the model and consist of:

- M221 logic controllers come with 9, 14 or 24 inputs
- M221 logic controllers come with 7,12 or 20 outputs
- M221 book controllers come with 8 inputs and 8 outputs, or 16 inputs and 16 outputs
- All M221 logic controllers come with removable screw terminal blocks
- M221 book controllers with 8 inputs and 8 outputs can be either screw or spring connector
- M221 book controllers with 16 inputs and 16 outputs use a HE10 connector
- Two independent 0-10V analog inputs
- Expansion bus for TM2 or TM3 I/O modules

The data connections are:

- USB port for programming
- SD card slot for data transfer
- Serial port for data transfer
- Optional Ethernet port for programming and data transfer

Special I/O

The M221 has I/O that can be used for special purposes.

The M221 Logic Controller has four fast inputs that can be used for high speed counters. These are configurable in the program. When these are being used for high speed pulse inputs the wiring should be shielded.

The first two outputs of an M221 with transistor outputs are fast outputs that can be used for pulse or PWM outputs. These can be used to control drives and other pulsed devices. When these are being used for high speed pulse outputs the wiring should be shielded.

M221 Logic controllers with relay outputs do not have high speed outputs but high speed counters can still be configured.

M221 Connectivity (cont.)

High Speed Counters

The M221 controller has two high speed counters that can be configured in five modes: Counter/down counter, Counter/down counter bi-phases, simple counter, simple down counter or frequency meter. These can be connected to the third and fourth input and configured to write to the third and fourth output of the controller.

Pulse Generator

The M221 controller has two pulse generators that can be configured to produce pulses or for Pulse Width Modulation. The timing has four preset values; 0.142ms, 0.57ms, 10ms or 1 second. These pulse generators when configured will write to the first and second digital outputs of the controller.

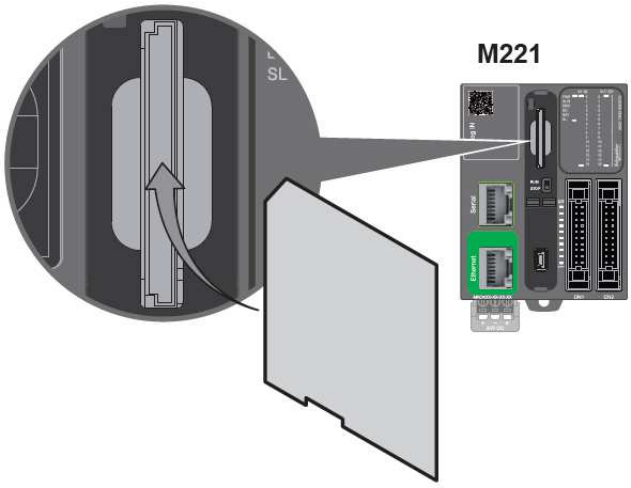
Ethernet Port

The Ethernet port is only available on the TM221ME logic controllers. It can be used for program upload/download as well as communicating to remote devices. The Ethernet port can communicate using Modbus TCP (salve) or Ethernet IP.

M221 Connectivity (cont.)

SD Card

The front of the M221 Logic Controller has a SD card slot. This allows a SD card to be used for data transfer and upgrading the firmware in the M221. The Schneider part number for the SD card is TMASD1



When the M221 powers up it will automatically check to see if a SD card is inserted.

Firmware Update

The files necessary for a firmware update are normally supplied in a zip file with each version of SoMachine Basic and should be used to update the M221 when a new version of SoMachine Basic is installed. Unzip the files and copy them to a SD card.

Ensure the power is off and the USB connection is removed before inserting the SD card. When the power is applied to the M221, it will check for a SD card and boot file. The boot file contains the command to copy the operating system files to the M221. These files will be copied and the M221 will perform a restart to load the new firmware.

Various LEDs will flash during the update process including the SD LED which may come on more than once. Leave the M221 until a steady state for the LEDs is seen for at least 15 seconds. The SD card can now be removed.

Note:

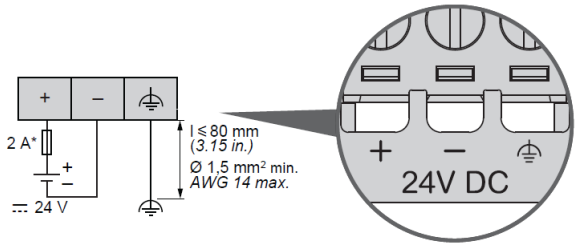
DO NOT REMOVE THE POWER WHILE THE FIRMWARE UPDATE IS TAKING PLACE.

If the power is removed, the firmware will be corrupted. The M221 will not be able to perform another firmware update and must be returned to Schneider Electric for repair.

M221 Wiring

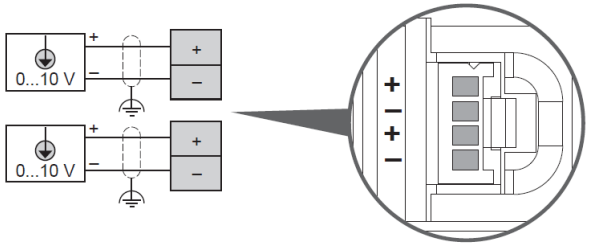
Power Supply

M221 logic controllers can be powered using 24VDC or 100-240VAC. All M221 book controllers are powered by 24V DC. The connector is on the bottom of the unit.



Analog Inputs

All M221 controllers have two independent 0-10V analog inputs. These are located under the cover at the top left of the unit (it has the QR code on it). The wiring is as shown.

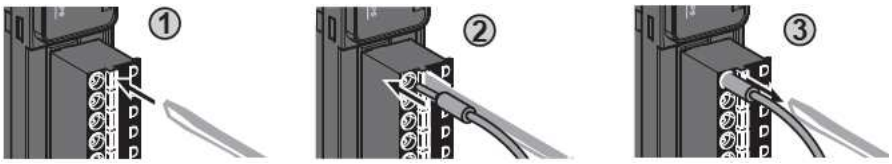


I/O Connectors

There are three types of connector for the M221 Logic controller I/O.

Screw terminals allow each wire to be fitted and a screw will hold the wire in place.

Spring terminals allow the wires to be more easily inserted or extracted.



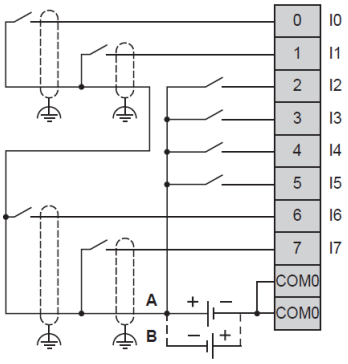
HE10 connectors are a standard connection for Schneider PLCs and controllers. They allow high density connection so are ideal for the 32 I/O version of the M221. This connector does require a special cable and different types are available - see the catalog for details of part numbers.

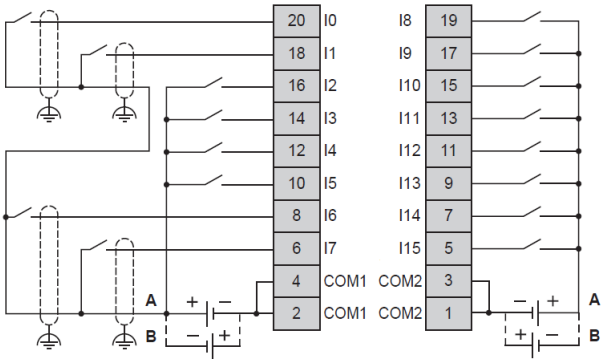
In all cases, the terminal block can be unplugged, allowing the wiring to be easily removed from the M221.

M221 Wiring (cont.)

Input Wiring

The wiring for the 8 and 16 input versions of the M221 Logic Controller are shown below.



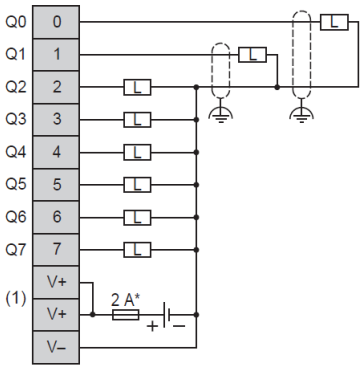


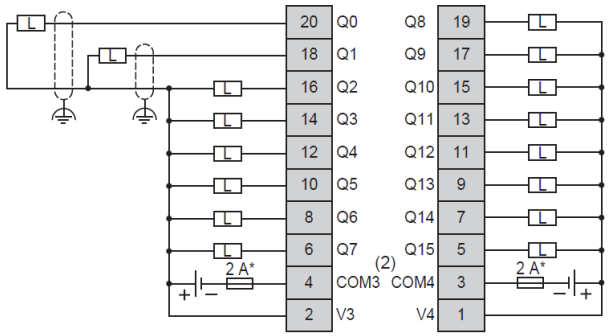
The first two inputs are shown shielded as they can be used for high speed pulse inputs so should be protected from noise. If these are used as normal outputs, they need not be shielded.

M221 Wiring (cont.)

Output Wiring

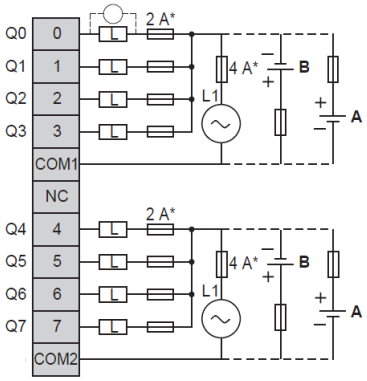
The wiring for the 8 and 16 output versions of the M221 Logic Controller are shown below.





The first two outputs are shown shielded as they can be used for high speed pulse outputs so should be protected from noise. If these are used as normal outputs, they need not be shielded.

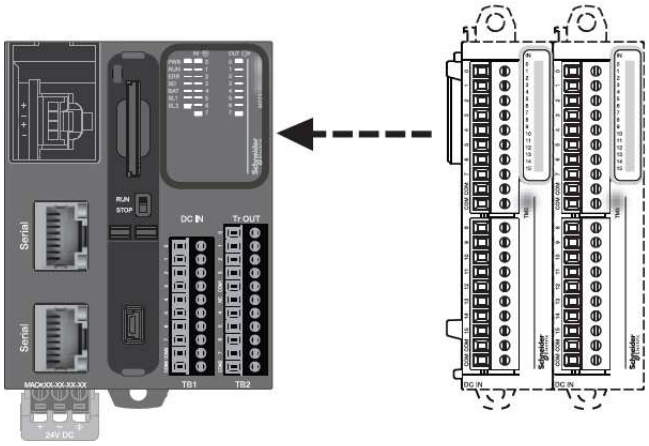
For the relay output version of the M221, the outputs need not be shielded as they cannot be used for high speed pulse output.



Expansion

I/O Modules

I/O modules can be added to the M221 controller to expand its capabilities.



There are two types of module that can be added.

TM2 modules are existing modules for Twido that can be used and allow the M221 to be compatible with Twido applications. This makes upgrading from the Twido to M221 simple especially when coupled with the Twido to M221 application conversion that is part of the SoMachine Basic software.

TM3 modules are a new range specifically designed for the M2xx range of Logic Controllers to take advantage of the extended I/O expansion bus. They offer new capabilities that are not available to the Twido.

Expansion (cont.)

TM2 Digital Modules

TM2 Digital modules offer a range of I/O configurations with 8, 16 or 32 inputs, 8, 16 or 32 outputs and a mixture of input and outputs. As with the M221, the outputs can be either transistor or relay.

They come with removable screw, spring contact or MIL connectors, the last of which is used for the high density 32 I/O modules.

These modules are compatible with the Twido controller.

TM2 Analog Modules

TM2 Analog modules offer a range of I/O configurations with 2, 4 or 8 inputs, 1 or 2 outputs and a mixture of input and outputs. The inputs are designed for a range of field devices including thermocouples, NTC probes and PT100/PT1000. The outputs are configurable for either 0-10V DC or 4-20mA.

All come with removable screw terminal blocks except the PT100/PT1000 modules which are fitted with a RJ11 connector.

These modules are compatible with the Twido controller.

TM3 Digital Modules

TM3 Digital modules offer a range of I/O configurations with 8, 16 or 32 inputs, 8, 16 or 32 outputs and a mixture of input and outputs. As with the M221, the outputs can be either transistor or relay.

They come with removable screw or HE10 connector which are either an option on the 16 I/O modules or required for the high density 32 I/O modules.

TM3 Digital Modules

TM3 Analog modules offer a range of analog inputs and outputs. They can be 2,4,or 8 input or 2 or 4 output. There is also a 4 input 2 output module for greater flexibility. Both inputs and outputs can be either voltage or current.

The TM3T modules also allow temperature input and can be 4 or 8 input. There is also a 2 input 1 output module that can be used for temperature inputs.

Expansion (cont.)

TM3 Expert Modules

There are two standard modules and four safety modules currently in the TM3 expert range.

The Tesys module allows connection to up to four Tesys motor starters and the transmitter receiver module allows the M221 I/O to be expanded even further by adding another 7 I/O modules. Only one transmitter receiver module can be used by a single M221.

The safety modules offer single function or dual function for CAT3 (SIL2) or CAT4 (SIL3).

Total I/O

Up to seven I/O modules can be added to the M221. These can be TM2 or TM3 modules or a mixture of both.

If one of the attached modules is a TM3 expansion module then another seven modules can be added. The M221 can control up to 144 I/O in total with a maximum of 42 relay outputs.

Cartridges

What are Cartridges?

Cartridges are a way of increasing the capabilities of the M221 Logic Controller By adding more I/O or connection options. They are also targeted at specific applications such as conveying, hoisting and packaging. They are attached to the front of the M221 Logic controller and provide options that are not available in either the TM2 or TM3 I/O.

Note:

The M221 Book Controller is unable to accept cartridge expansion and can only use TM2 or TM3 expansion modules.

Types of Cartridge

Cartridges allow additional analog I/O and communications options to be added to the M221 Logic Controller. The full list for step 2 is given below.

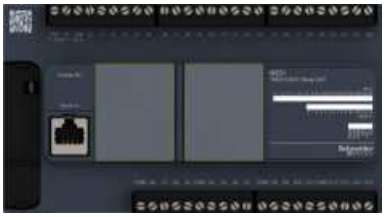
Cartridge	Function
TMCAI2	2 analog or current inputs
TMCAQ2C	2 current outputs
TMCAQ2V	2 voltage outputs
TMC2CONV01	Serial line for conveying applications
TMC2HOIS01	2 analog voltage or current inputs for hoisting load cell
TMC2PACK01	2 analog voltage or current inputs for packaging
TMC2SL1	1 serial line
TMC2TT2	2 analog temperature inputs

Cartridges (cont.)

Adding Cartridges

Physically, the cartridge is plugged on to the front of the M221 Logic Controller. A single cartridge can be added to the TM221C16X, TM221CE16X, TM221C24X and TM221CE24X controllers.

Two cartridges can be added to the TM221C40X or TM221CE40X controllers.



Configuring in SoMachine Basic

The cartridges are in the SoMachine Basic Library and can be added the same way as I/O modules.

Select the cartridge and drag it on to the M221 Logic Controller.

Summary

Summary

This chapter covered the following topics:

Questions

The following questions are to check understanding of the topics covered in this chapter:

- What type of power supply can be used to power the M221 Logic Controllers?

- What is the maximum I/O supported by the M221 Logic Controller?

- The HE10 connector is fitted to which of the M221 Logic Controllers?

Chapter 3: Programming a Simple Application

Overview

Introduction

Programming is the way of telling the M221 Logic controller how to function. It uses two of the standard IEC61131 programming languages; Ladder Logic and Instruction List. The programming environment also provides tools to help debug the program if it does not perform as expected.

This Chapter Covers These Topics:

Languages

IEC61131

IEC61131 is the international standard for programming PLCs. Its main purpose is to define standard data types and languages. There are many data types that will be discussed later, but there are only five programming languages:

- Ladder Diagram (LD)
- Function Block Diagram (FBD)
- Structured Text (ST)
- Instruction List (IL)
- Sequential Function Chart (SFC)

Of these languages, only Ladder Diagram and Instruction List are currently supported by SoMachine Basic although other languages are planned for future releases.

Ladder

Ladder is a popular programming language as it is similar to electrical diagrams and visually, it is very easy to see what the program is doing. With the addition of in-line status display, it is also very easy to debug.

It consists of a series of program lines or rungs, so called because they look like the rungs of a ladder. To the left of the rung are a set of inputs and conditions that must be solved. To the right of the rung is an object (or objects) defining what to do with the result.

A ladder rung may look something like the following:

```
%I0.0  %I0.1              %Q0.7

--]  [------]  [-----------------(  )--
```

Put simply, if input %I0.0 is on and input %I0.1 is on then output %Q0.7 will turn on. If either of the inputs is off then the output will also be off.

Ladder does have some limitations though. It cannot perform all functions due to limitations in the programming rules and it must be converted to Instruction List before it is processed by the logic controller.

Instruction List

Instruction List is similar to computer machine code and anyone familiar with that will see many similarities. It uses a working store to load and combine values. This store is then written to an output or memory location called an accumulator.

The commands:

```
0001|    LD %I0.0

0002|    AND %I0.1

0003|    ST %Q0.7
```

The three steps are:

Line 1 - Load the value of %I0.0 (0 or 1) into the accumulator

Line 2 - Perform a logical AND of the accumulator with the value of %I0.1 (0 or 1) and store the result back in the accumulator.

Line3 - Write the result (the contents of the accumulator) to output %Q0.7

This is similar to the way a calculator works. The first number is entered (or loaded) at step 1. A mathematical operator such as plus or minus is pressed and another number is entered at step 2; pressing the equals or another operator stores the result back on the display. The result is transferred from the calculator display to a piece of paper at step 3.

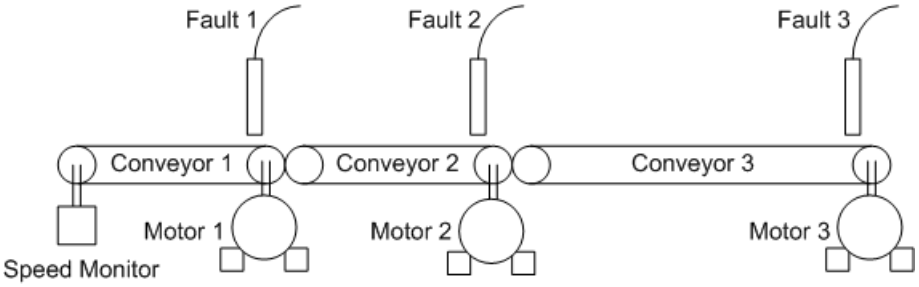
This can be written in ladder as:

```
%I0.0  %I0.1              %Q0.7

--]  [------]  [-----------------(  )--
```

The Conveyor Application

Conveyor Application

For the purposes of creating a simple application and program, a simple three-conveyor application will be used. Control will be provided for starting, stopping and fault monitoring.



The start control will simply start all three conveyors at the same time although for better control, timers could be used to feed product in a controlled way or distribute startup current for the motors.

Shutdown is normally timed to allow product to clear the conveyors before each is shutdown. This section will allow the demonstration of timer programming

Fault monitoring can determine whether there is a problem and take the appropriate action to prevent spillage or damage. A speed monitoring program will also describe the configuration of analog value manipulation and the use of the analog input.

For any application it is important to have a list of the addresses of all I/O and internal registers.

Addressing

Addressing Format

The M221 Logic Controller addressing uses the following format:

%<Type>[<Identifier>]<Location>

An address must always begin with the % character. This tells the Logic Controller that it is an address, not some other piece of information.

This is followed by one of five letters identifying the type of address:

I	The address is a physical input on either the controller or an expansion module.
K	The address is an internal memory location within the controller. The value is fixed and can NOT be changed by the program.
M	The address is an internal memory location within the controller. This value can be changed by the program.
Q	The address is a physical output on either the controller or an expansion module.
S	Internal system locations that are used to perform various functions and monitor the controller

The type may then be followed by a type identifier which can be one of the following:

(none)	The address contains a value that is a single bit having a value of either 0 or 1.
W	The address contains a value that is a word and has a value between 0 and 65535.
D	The address contains a value that is a double word and has a value between 0 and 4294967295.
F	The address contains a value that is in floating point format and has a value between 0 and 65535.

The numeric part of the address contains the location. This location can have one of two formats depending on the address type. The format nnnn is used for internal memory locations. The format x.y is used for inputs and outputs where the first part (x) identifies the module position in the rack. Slot zero is the left most slot and is the processor.

The second part (y) identifies the input or output number on that module.

Thus an address of %I1.2 refers to the third input on the first expansion module (Numbering starts at zero).

Exercise - Identifying Addresses

Learning Outcomes

By the completion of this exercise you will:

- Understand the addressing format used by SoMachine Basic

1 Explain the following addresses.

Address	Int/Ext	Location	Data Type
%I0.4			
%MW2			
%Q0.6			
%IW1			
%S6			
%MF10			
%KW2			
%SW100			

2 Create an address for the following.

i. The second digital input of the logic controller

ii. An internal memory location that can be used to store the value 17.4

iii. System word 8

iv. The third output of the second expansion module (assuming that it is a digital output)

v. The first input of the second digital output module

Addressing (cont.)

I/O List

The following I/O will be used in the application.

Inputs:

Address	Description
%I0.2	Conveyor 1 Fault Signal
%I0.3	Conveyor 1 Fault Signal
%I0.4	Conveyor 1 Fault Signal
%I0.6	Stop Button
%I0.7	Start Button

Outputs:

Address	Description
%Q0.2	Conveyor 1 Run Control
%Q0.3	Conveyor 2 Run Control
%Q0.4	Conveyor 3 Run Control
%Q0.6	Run Indicator
%Q0.7	Fault Indicator

Analog:

Address	Description
%IW0.0	Conveyor Speed

Addressing (cont.)

Register List

The following internal registers will be used in the application.

Address	Description
%M100	Run Relay
%M101	Conveyor 1 Stop Sequence
%M102	Conveyor 2 Stop Sequence
%M103	Conveyor 3 Stop Sequence
%M104	Not Used
%M105	Conveyor 1 Fault
%M106	Conveyor 2 Fault
%M107	Conveyor 3 Fault
%M108	Not Used
%M109	Stop Sequence Relay
%M110	Remote Start
%M119	Remote Stop
%M120	Variable Speed OK
%M121	Variable Speed High
%M122	Variable Speed Low

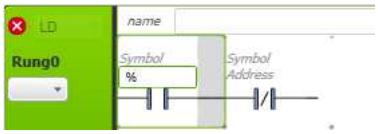
Address	Description
%MW1	Low Setpoint
%MW2	High Setpoint
%MW3	Scaled Speed

Addressing (cont.)

How to Assign Objects to I/O and Memory

When a program is created, the objects used in the program must be assigned to either an I/O or memory address. If this is not done, the program cannot be compiled and downloaded to the Logic Controller.

To assign an object to an I/O or memory address, first double-click the address above the object. An entry field will appear.



Enter the desired address into this entry field.

Note:

The % character is already in the entry box as this identifies the entry as an address (see previous section).

Exercise - Create a Basic Application

Learning Outcomes

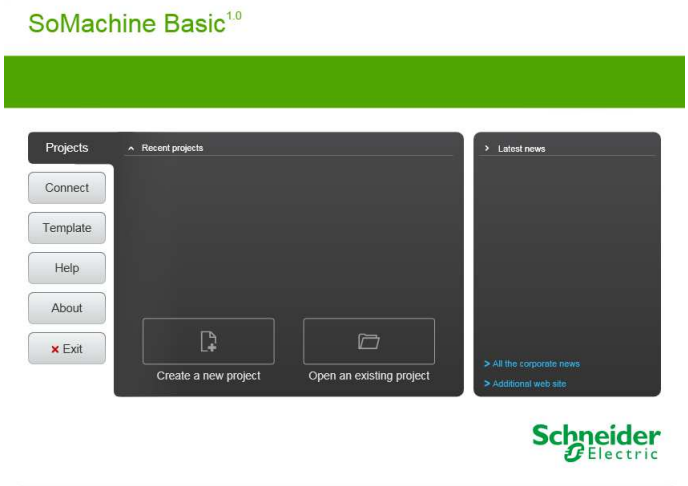
By the completion of this exercise you will:

- Be able to create a new application

Skip this exercise if you have already completed the exercise in the eLearning - this is a repeat of that exercise.

1 Create a new application.

i. Start the SoMachine Basic software using either by double-clicking the icon or the desktop or going to **Start » Programs » Schneider Electric » SoMachine Basic » SoMachine Basic.**

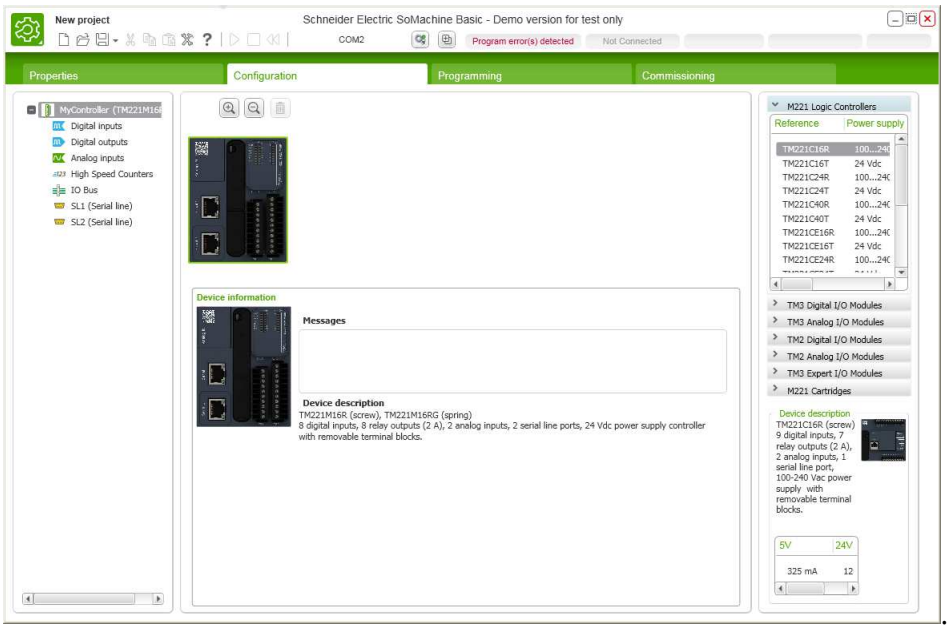


ii. On the start page click the **Create a new project** button.

Exercise - Create a Basic Application (cont.)

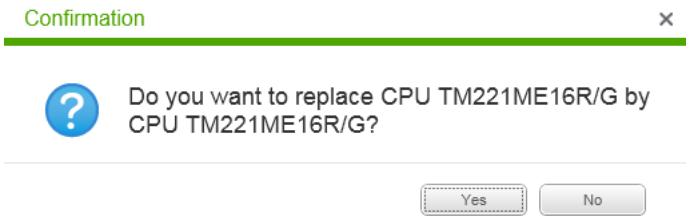
2 Assign a controller.

i. The project will open on the Configuration tab. If the configuration tab is not selected, click to select it.



ii. The right hand side of the screen shows the hardware catalog. In the Logic Controller section, select the logic controller that you have and drag-and-drop it to the centre window. A picture of the logic controller will be shown under the mouse cursor.

iii. If the controller is being changed, a message box will be displayed to confirm the change of controller.

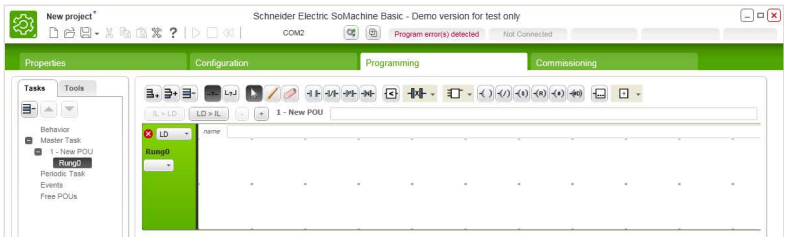


Click the **Yes** button to replace the controller.

Exercise - Create a Basic Application (cont.)

3 Create the first rung.

i. Click the Program tab to enter the programming section of the SoMachine Basic software.

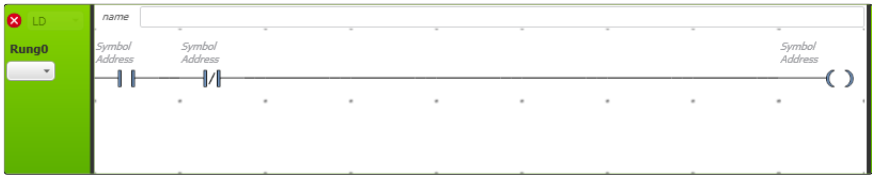


The first ladder rung will have been created ready for programming.

ii. Select the normally open contact from the toolbar and place it in the first row and column of the program.

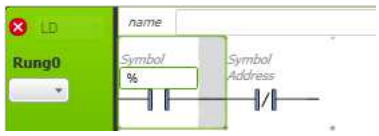
iii. Select the normally closed contact from the toolbar and place it in the first row second column.

iv. Select the coil from the toolbar and place it on the first row, final column. The program should now look like the following:



4 Assign Inputs and Outputs to the rung.

i. Double-click the word "Address" above the first contact.



ii. In the dialog box that opens, enter the address %I0.7. The % will already be in the dialog box.

The % indicates that it is an address in the Logic Controller, the I means that it will be a physical input and the 0.7 means that it will be the eighth input on the Logic Controller.

Exercise - Create a Basic Application (cont.)

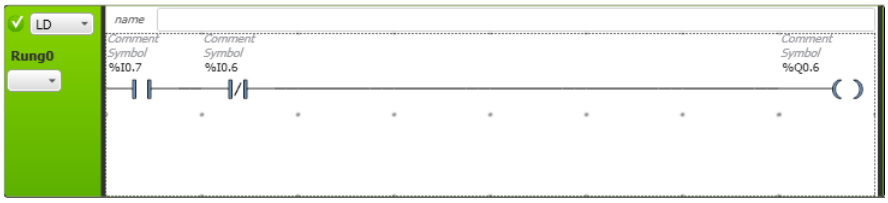
iii. Double-click the word "Address" above the second contact and enter the address %I0.6.

This will assign the contact to the sixth input of the Logic Controller.

iv. For the output coil, enter the address %Q0.6.

The Q signifies that this is an output and the 0.6 that this will be the seventh output on the Logic Controller.

The final rung should look similar to the following:



5 Save the application.

i. Click the save button on the toolbar.

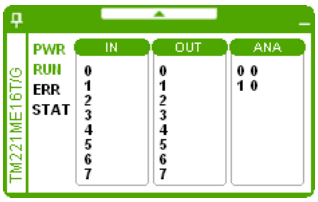
ii. Choose the location and enter "Conveyors" for the filename.

iii. Click the **Save** button to save the application.

The Simulator

Simulating Programs

The simulator is a piece of software supplied with SoMachine Basic that allows programs to be tested without using the controller.



The program is loaded into the controller and the simulated controller inputs to be operated and their effects examined. The outputs can be monitored and the internal program analysed.

Starting the Simulator

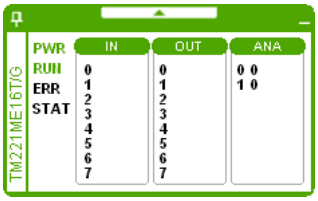
The simulator is started using the Launch Simulator button on the Commissioning tab.

This will start the simulator and load the current program. The program can then be tested using the simulator without the need to download to a physical controller.

The Simulator (cont.)

The Simulator Windows

The simulator window shows information about the simulator and allows control of the inputs for the application. The controller is shown on the left hand side along with, power, run, error and status indicators.

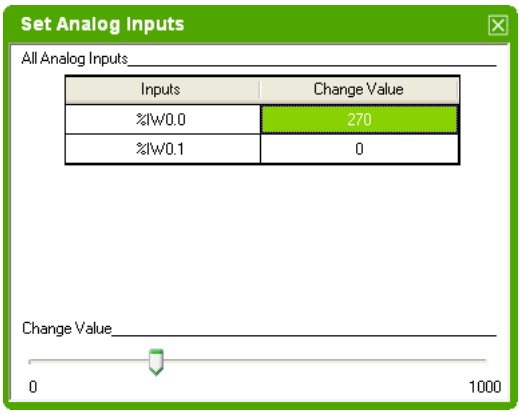


The PWR indicator will be green showing that the simulator is running. The RUN indicator will either be flashing to show that the application in the simulator is not running, or solid green to show that the application is running. The ERR and STAT LEDs are not currently used.

The IN column shows the state of the inputs for the controller. These will be numbered from zero up to the total number of inputs supported by the controller. These inputs can also be clicked to change the state of the input, toggling it on or off. When the input turns on it will be coloured green. When the input is off it will be coloured white.

The OUT column shows the state of the outputs for the controller. These will be numbered from zero up to the total number of outputs supported by the controller.

The ANA column shows the values of the analog inputs for the controller. Double-clicking this will open another window allowing the values of the analog inputs to be changed.



The vale can be changed by selecting the analog value and using the slider at the bottom of the window. If a more accurate input is required, the value shown in the change value column can be double-clicked and the value typed in.

Exercise - Test the Program

Learning Outcomes

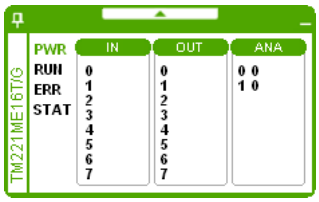
By the completion of this exercise you will: be able to:

- Start the simulator
- Operate the simulator to test a program

1 Start the Simulator.

i. Go to the commissioning tab in SoMachine Basic and click the **Start Simulator** button.

The simulator will be started and the program will be loaded into the simulator.



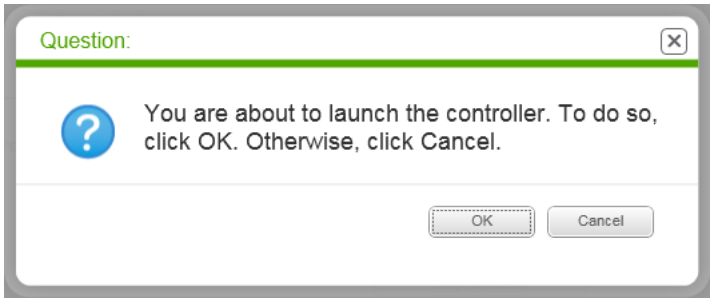
Do not operate anything in the time management window.

The **Run** indicator in the simulator will be flashing to show that the simulator is stopped.

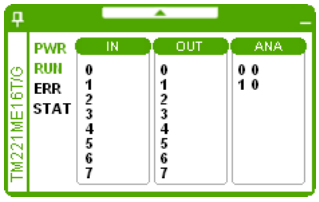
Exercise - Test the Program (cont.)

ii. Click the **Run Controller** button on the commissioning screen to run the program in the simulator.

A pop-up window will be shown asking to confirm the action.



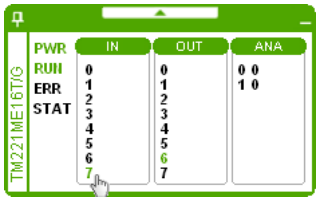
Click the OK button to confirm and the **Run** indicator will change to a solid green.



---

2 Test the program.

i. In the simulator window, click the number 7 representing the last input. The input will turn green to show that it is on.



Output 6 should also turn on. This is being controlled by the program. If input 7 is on and input 6 is off, then output 6 should turn on.

ii. If output 6 doesn't not turn on then check the program carefully for any mistakes.

Exercise - Test the Program (cont.)

3 Monitor the program.

i. Switch to the programming tab and observe the state of the program. The normally open contact will be coloured green and shown as true, indicating that input 7 is on. The output coil should also be coloured green to show the output being controlled by the program.

Operate the two inputs and observe the effect on the program.

4 Further testing.

i. Ensure that input 7 is off. If not, click input 7 to turn it off. Click input 6 to turn it on. There should be no effect.

ii. Click input 7 to turn it on. This time output 6 should not come on as it is being prevented by the normally closed contact for input 6 which is now true.

Note:

The True/False indication shows the state of the input, not the state of the coil or contact. The green colouring shows the state of the contact. This can be confusing for inverted contacts and coils but a green colour will always signify power or on.

5 Stop the simulator.

i. Go to the Commissioning tab and click the Stop Simulator button to stop the simulator.

Symbols

Using Symbols

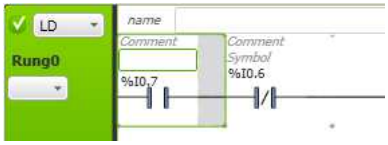
Symbols are a way of naming objects and making them easier to identify. If an object has an address of %I0.7 it is impossible to determine what that object does without referring to the documentation for the application. If the object is also given the symbol "Start_Button", anyone looking at the program will have a good idea what that object is supposed to do.

Choosing meaningful symbols can help to make the program self-documenting as they will describe what each object does. Usually a good name for the output object will also help to identify what the entire rung does.

Symbols (cont.)

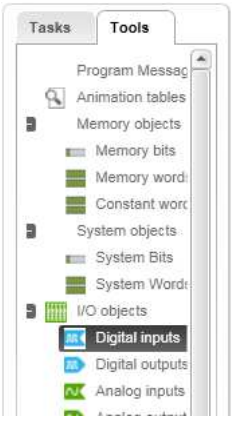
How to Add Symbols

In the ladder editor, double-click the word "Symbol" above an object. This will open a dialog box and the symbol can be entered.

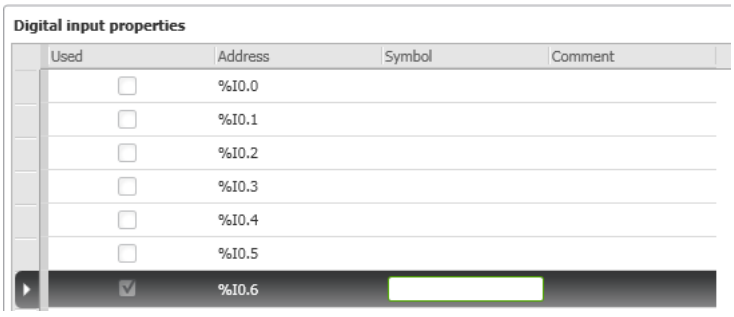


The symbol can contain letters numbers and the underscore character. Spaces are not allowed.

Alternatively, the object property box can be displayed at the bottom of the screen by selecting Tools in the Module Programming Tree and selecting the appropriate input or output type.



Double-clicking the Symbol column of the row containing the object will open a dialog box which will also allow the symbol to be entered.



When the symbol has been entered, the Apply button must be clicked to accept the changes. This allows multiple symbols to be entered before accepting the change.

A different method of entering multiple symbols will be explored in the next section.

Exercise - Adding Symbols to Objects

Learning Outcomes

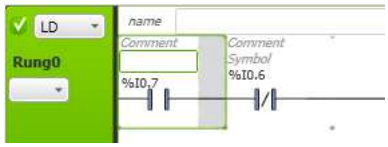
By the completion of this exercise you will:

- Be able to add symbols to the program to identify objects

Skip this exercise if you have already completed the exercise in the eLearning - this is a repeat of that exercise.

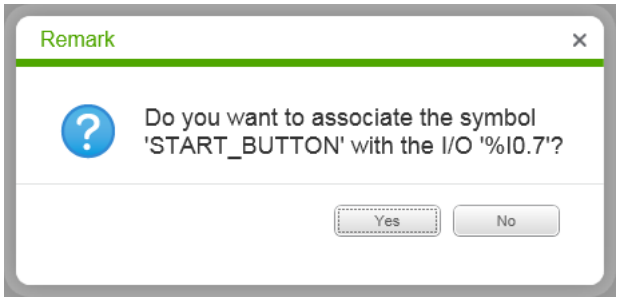
1 Add symbols to the four objects contained in the program.

i. In the ladder editor, double-click the word "Symbol" above the first contact.



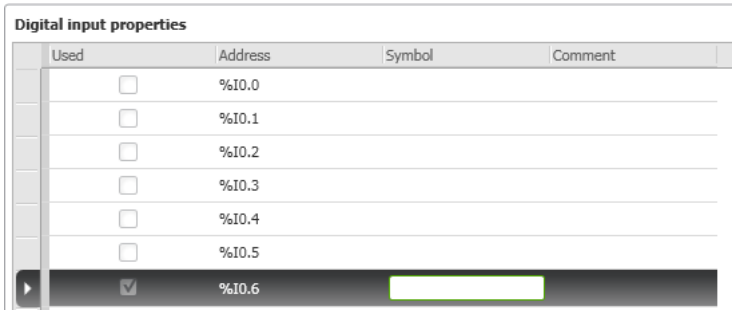
Enter the symbol `START_BUTTON` and press **Return**.

A message will appear asking if you want to associate the symbol "START_BUTTON" with the I/O %I0.7.



Click the **Yes** button to confirm.

ii. At the bottom of the screen, double-click the Symbol column of the row containing %I0.6.



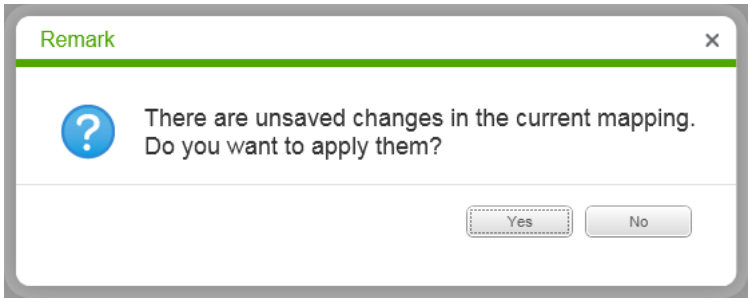
Enter the symbol `STOP_BUTTON`. Click the **Apply** button in the bottom right hand corner to accept the change.

Exercise - Adding Symbols to Objects (cont.)

iii. In the **Module Programming Tree**, select the **Tools** Tab. Under **I/O Objects** select **Digital outputs**.

iv. Double-click the Symbol column of the row containing %Q0.6 and enter the symbol "CONVEYORS_RUNNING". Do not click the **Apply** button.

v. In the Module Programming Tree select another section such as Analog Inputs. The following message will be displayed.



This ensures that any unsaved changes will not be lost if you forget to apply the changes and try to navigate away from the page.

vi. Click the **Yes** button to apply the changes the change.

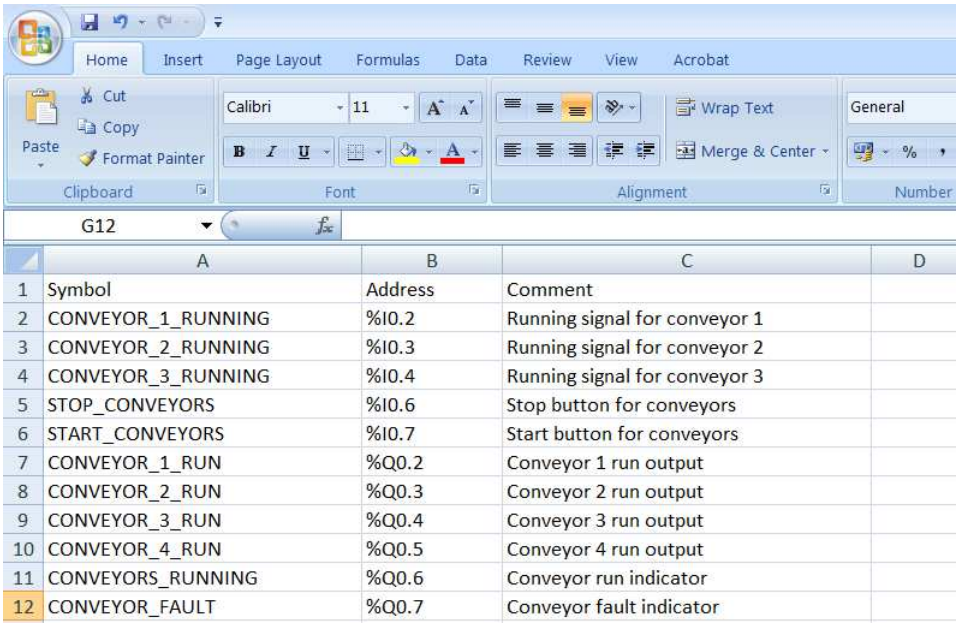
vii. Save the application.

Symbols (cont.)

Exporting and Importing

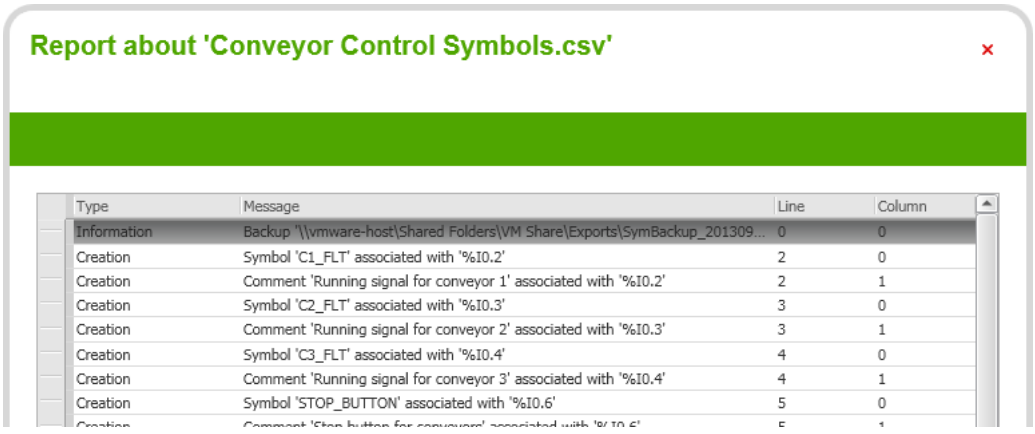
Exporting and importing provide an easy way of editing symbol names and descriptions outside SoMachine Basic.

The symbols can be exported to a .csv file; either comma separated or semi-colon separated. This file can then be opened in Microsoft Excel and edited, allowing all the features of Microsoft Excel to be used to edit the list.



	A	B	C	D
1	Symbol	Address	Comment	
2	CONVEYOR_1_RUNNING	%I0.2	Running signal for conveyor 1	
3	CONVEYOR_2_RUNNING	%I0.3	Running signal for conveyor 2	
4	CONVEYOR_3_RUNNING	%I0.4	Running signal for conveyor 3	
5	STOP_CONVEYORS	%I0.6	Stop button for conveyors	
6	START_CONVEYORS	%I0.7	Start button for conveyors	
7	CONVEYOR_1_RUN	%Q0.2	Conveyor 1 run output	
8	CONVEYOR_2_RUN	%Q0.3	Conveyor 2 run output	
9	CONVEYOR_3_RUN	%Q0.4	Conveyor 3 run output	
10	CONVEYOR_4_RUN	%Q0.5	Conveyor 4 run output	
11	CONVEYORS_RUNNING	%Q0.6	Conveyor run indicator	
12	CONVEYOR_FAULT	%Q0.7	Conveyor fault indicator	

When the editing is complete, the file can be saved to .csv format and imported back into SoMachine Basic. The Import will check whether the symbol already exists and if not, import the symbol into the application. If the symbol already exists then it will be ignored. The description however, will be replaced if it already exists. A report will show the actions carried out by the import process.

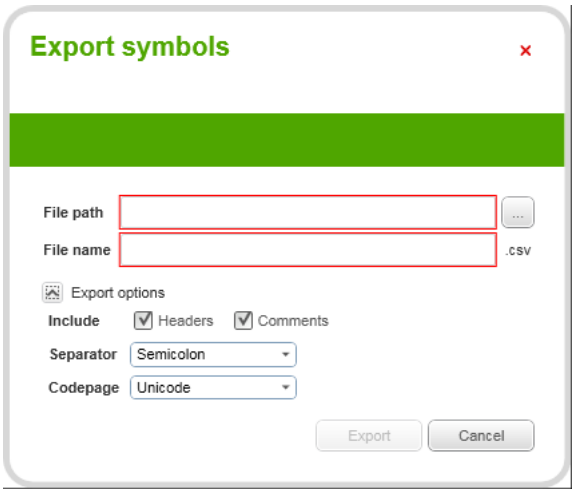


Type	Message	Line	Column
Information	Backup '\\vmware-host\Shared Folders\VM Share\Exports\SymBackup_201309...	0	0
Creation	Symbol 'C1_FLT' associated with '%I0.2'	2	0
Creation	Comment 'Running signal for conveyor 1' associated with '%I0.2'	2	1
Creation	Symbol 'C2_FLT' associated with '%I0.3'	3	0
Creation	Comment 'Running signal for conveyor 2' associated with '%I0.3'	3	1
Creation	Symbol 'C3_FLT' associated with '%I0.4'	4	0
Creation	Comment 'Running signal for conveyor 3' associated with '%I0.4'	4	1
Creation	Symbol 'STOP_BUTTON' associated with '%I0.6'	5	0

Symbols (cont.)

How to Export Symbols

To export a list of symbols, click the **Export** button. The following dialog box will appear: If the export options are not shown, click the down-arrow button to the left of Export options.

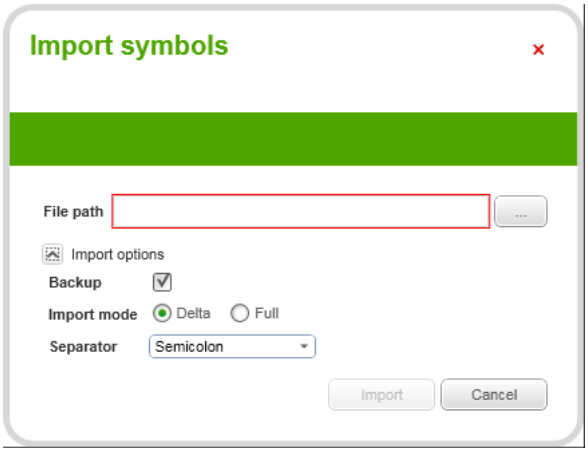


Enter the path and filename and choose the export options:

- Whether to include headers and comments
- Whether to use a semicolon or comma separator
- Whether to use Unicode or ASCII

How to Import Symbols

To import a list of symbols, click the **Import** button. The following dialog box will appear:



Enter the path and filename and choose the import options:

- Whether to create a backup first
- Whether to import only changes or import all items
- Whether to use a semicolon or comma separator
- Whether to use Unicode or ASCII

Exercise - Exporting and Importing Symbols

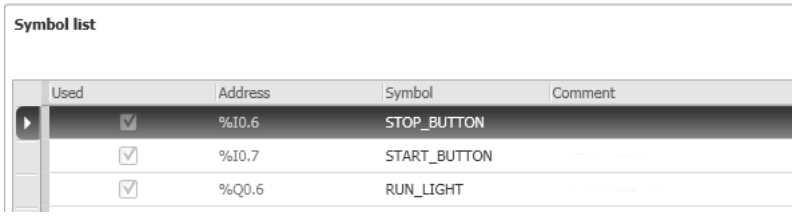
Learning Outcomes

By the completion of this exercise you will:

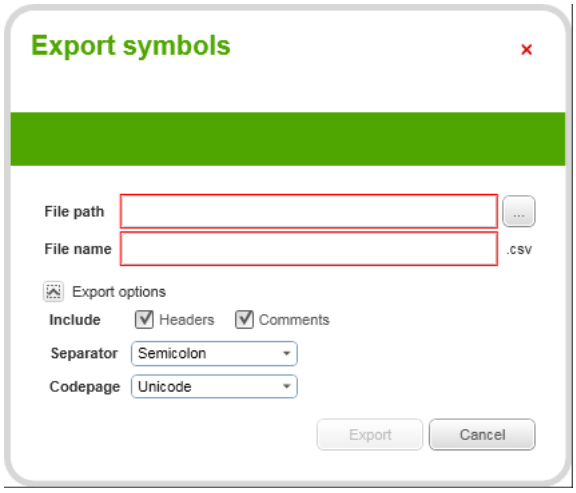
- Be able to export symbols to a .csv file
- Be able to import symbols from a .csv file

1 Export the symbols to a .csv file

i. In the **Module Programming Tree**, under **Software Objects** select **Symbols List.** A list of the currently configured symbols will be displayed.



ii. Click the **Export** button above the list of symbols. The following dialog box will be displayed. If the Export options are not displayed, there will be a Down Arrow button to the left of "Export options". Click this **down arrow button** to display them.



iii. Click the **ellipsis button** to the right of the File Path entry box and choose the desktop.

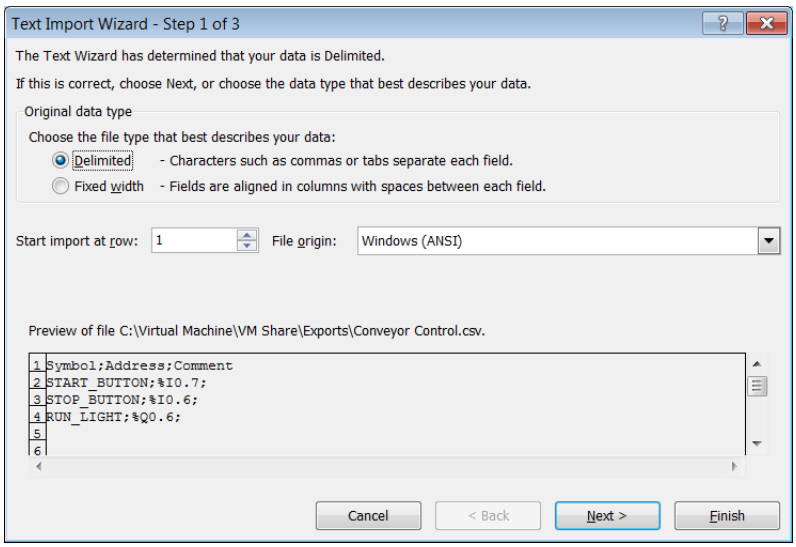
iv. Enter `Conveyor Control` for the filename.

v. Click the **Export** button to export the file.

Exercise - Exporting and Importing Symbols (cont.)

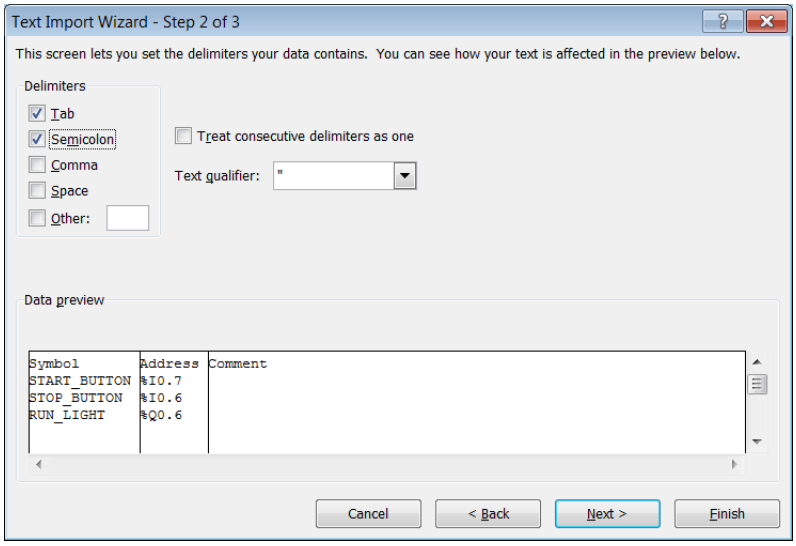
2 Edit the exported symbols file in Excel.

i. Start Excel and open the symbols file that is on the desktop. If the file was saved to a different location in the previous step, open it from there. The import wizard will be automatically displayed.



Click the **Next** button to go to page 2.

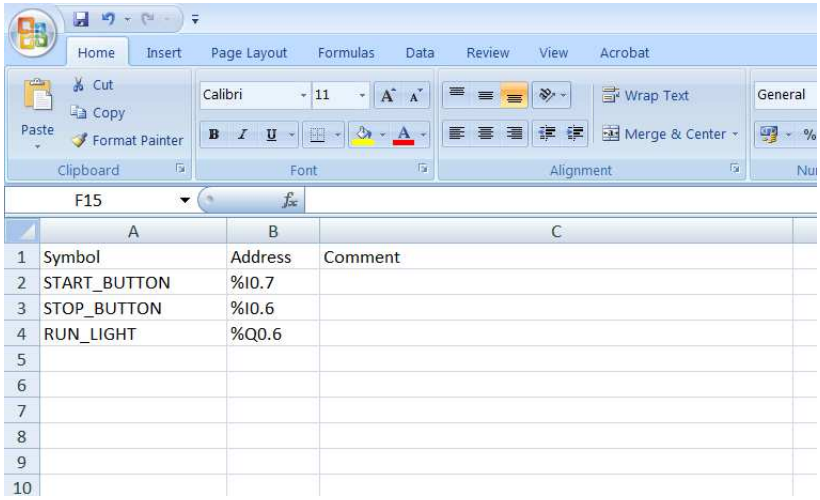
ii. On page 2 of the wizard, ensure that the **Semicolon** tickbox is selected.



Click the **Finish** button to complete the import process.

Exercise - Exporting and Importing Symbols (cont.)

iii. Excel will display the symbols in three columns of the spreadsheet; Symbol, Address and Comment



	A	B	C
1	Symbol	Address	Comment
2	START_BUTTON	%I0.7	
3	STOP_BUTTON	%I0.6	
4	RUN_LIGHT	%Q0.6	
5			
6			
7			
8			
9			
10			

iv. Edit the spreadsheet to contain the following entries:

	A	B	C
1	Symbol	Address	Comment
2	C1_FLT	%I0.2	Running signal for conveyor 1
3	C2_FLT	%I0.3	Running signal for conveyor 2
4	C3_FLT	%I0.4	Running signal for conveyor 3
5	STOP_BUTTON	%I0.6	Stop button for conveyors
6	START_BUTTON	%I0.7	Start button for conveyors
7	C1_SPD	%IW0.0	Conveyor 1 speed sensor
8	C1_RUN	%Q0.2	Conveyor 1 run output
9	C2_RUN	%Q0.3	Conveyor 2 run output
10	C3_RUN	%Q0.4	Conveyor 3 run output
11	RUN_LIGHT	%Q0.6	Conveyor run indicator
12	FLT_LIGHT	%Q0.7	Conveyor fault indicator
13	RUN_RELAY	%M100	Conveyor run relay
14	C1_STOPR	%M101	Conveyor 1 stop sequence
15	C2_STOPR	%M102	Conveyor 2 stop sequence
16	C3_STOPR	%M103	Conveyor 3 stop sequence
17	C1_FLTR	%M105	Conveyor 1 fault relay
18	C2_FLTR	%M106	Conveyor 2 fault relay
19	C3_FLTR	%M107	Conveyor 3 fault relay
20	STOP_RELAY	%M109	Stop sequence relay
21	VS_OK	%M120	Variable Speed OK
22	VS_HIGH	%M121	Variable Speed high
23	VS_LOW	%M122	Variable Speed low
24	VS_LO_SP	%MW1	Variable speed conveyor low setpoint
25	VS_HI_SP	%MW2	Variable speed conveyor high setpoint

Exercise - Exporting and Importing Symbols (cont.)

v. Save the spreadsheet as a .csv file called **Conveyor Control Symbols**.

If a message is displayed about compatible features, click the **Yes** button to save the file.

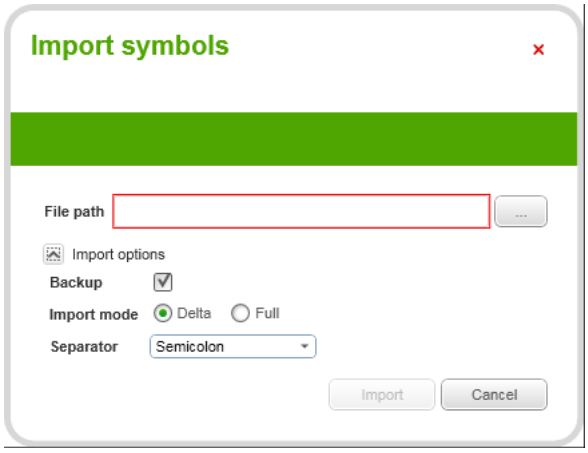
---

3 Import the .csv file into SoMachine Basic.

i. In the SoMachine Basic Symbols List view, click the **Import** Button.

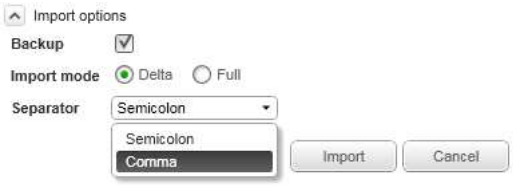
The project must be saved before the symbols file can be imported. If the following message is displayed, the project has not been saved.

Click the **OK** button, save the project and then click the **Import** button again.



ii. Click the **ellipsis** button to the right of the File Path and navigate to the desktop. Select the file called Conveyor Control Symbols.csv. If the file was saved to a different location in the previous step, open it from there.

iii. If the Import Options are not displayed, there will be a Down Arrow button to the left of "Import Options". Click this **down arrow button** to display them. Drop down the **Separator** selection box and choose **Comma**.

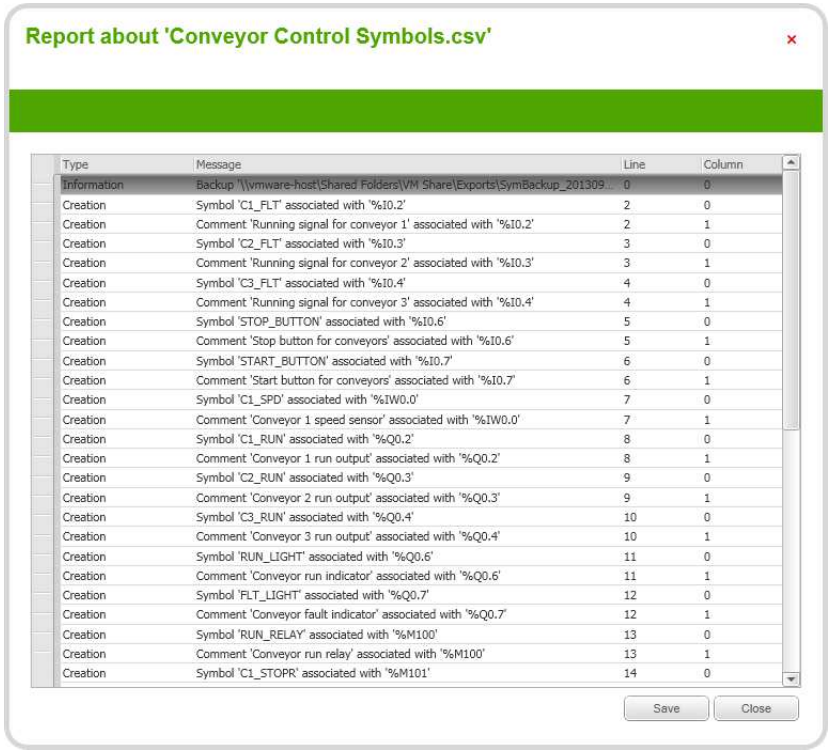


(Excel will use the comma separator by default unless a different separator has been chosen in the Excel options)

iv. Click the **Import** button to begin the import process.

Exercise - Exporting and Importing Symbols (cont.)

v. When the import has completed, a window will open showing the report for the import. Review the report for any errors.



Program Structure

POUs

Program Operation Units (POUs) are a way of organising code into sections, making it easier to find were certain control functions are located.

Although there is a Search function in SoMachine Basic, grouping sections of code in this way can help to see how rungs interact with each other.

For example, the conveyor application has three POUs

- Control
- Fault Handling
- Speed Monitoring

Thus if a programmer wanted to look at setpoints for the speed control, it makes sense for the program code handling that to be in the Speed Monitoring POU. If they wanted to find out why the running light wasn't coming on, they would look in the Control Section (although perhaps the first task would be to check the bulb).

Program Structure (cont.)

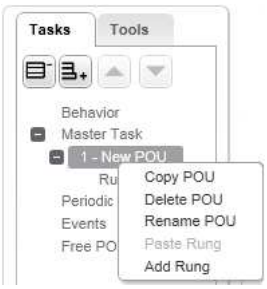
Adding POUs and Rungs

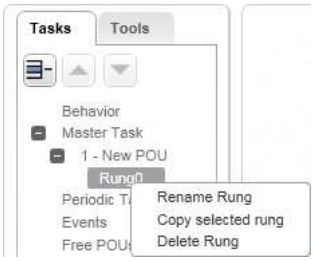
To create a new POU, right click the Master Task and from the menu select **Add POU**.

To create a new rung, right click the POU and from the menu select **Add Rung**.

Renaming POUs and Rungs

POUs and Rungs can also be renamed. Right click the POU or Rung and select **Rename POU** or **Rename Rung**.



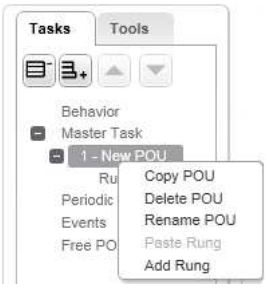


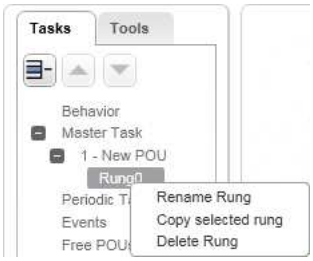
This renaming helps to self-document the program.

Program Structure (cont.)

Copying POUs and Rungs

POUs and Rungs can also be copied. Right click the POU or Rung to be copied and select **Copy POU** or **Copy selected rung**.





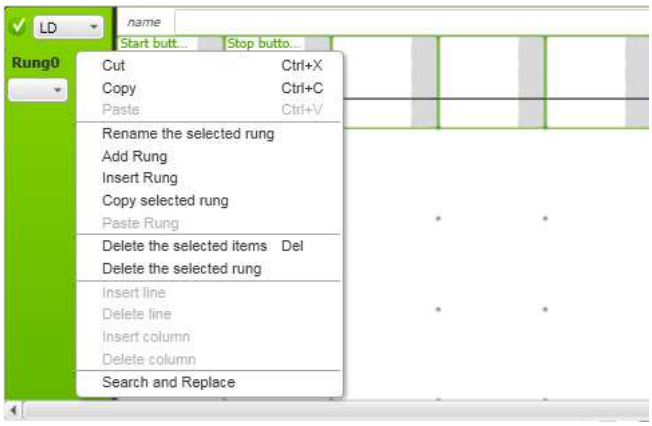
When a POU has been copied it can be pasted into the Master Task. Right-click the Master Task and select Paste POU from the menu.

When a Rung has been copied it can be pasted into any POU. Right-click the required POU and select Paste Rung from the menu.

Copying Objects in Rungs

Individual items within rungs can also be copied. Select the items to be copied. The Windows shortcut keys will allow multiple items to be selected - <Ctrl>click and <Shift>click. <Ctrl>a can also be used to select all items in the rung.

Right-click the green area to the left of the rung and select **Copy** from the menu.



Select the rung where the items are to placed, right-click the green area to the left of the rung and select **Paste** from the menu.

The options to add, rename, copy and paste rungs are also here along with options to insert a rung and edit lines, columns and items within the rung.

Exercise - Create a Program Structure

Learning Outcomes

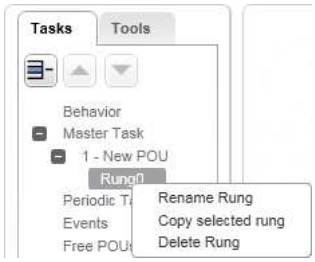
By the completion of this exercise you will:

- Create new ladder rungs and POUs
- Rename ladder rungs and POUs

Skip this exercise if you have already completed the exercise in the eLearning - this is a repeat of that exercise.

1 Create new rungs for the program.

i. Right-click the rung Rung_0 and from the menu select **Rename Rung**.



ii. Change the name of the rung to `Start Control`.

iii. Right-click the POU POU_0 and from the menu select **Add Rung**.

iv. Change the name of the rung to `Stop Control`.

Exercise - Create a Program Structure (cont.)

v. Add new rungs to the program and give them the following names:

- Conv 1
- Conv 2
- Conv 3
- Running
- Stop 1
- Stop 2
- Stop 3

i. Right-click the POU and rename it to `Control`.

2 Create new POUs for the program

i. Right-click the Master Task and from the menu select Add POU.

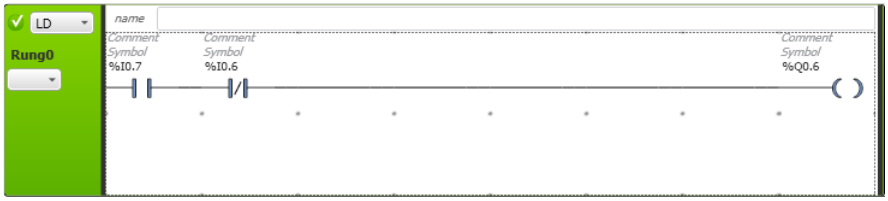
ii. Rename this POU `Fault Handling`.

iii. Add another POU and call it `Speed Monitoring`.

3 Save the application

Links

Links are often created automatically when objects are placed in a rung. The link is usually correctly placed but there will be times when the link must be changed.

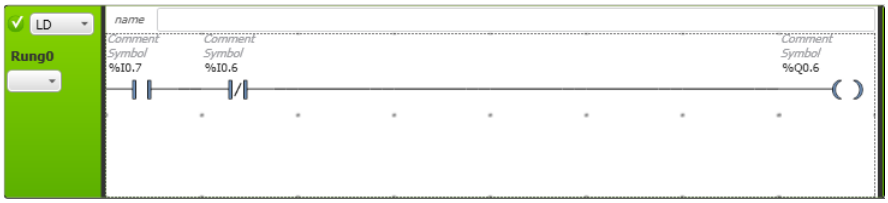


For example, when the output coil is drawn in the above rung, the link between the last contact and the coil is drawn automatically.

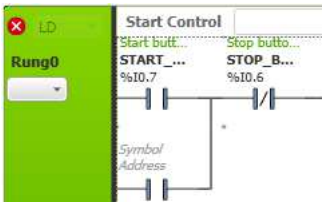
There are two modes when placing a contact, normal mode and branching mode. Normal mode will simply place the object in the rung; branching mode will place the object in the rung and attempt to draw the links to existing contacts.

Automatic Links

If Branching Mode is selected, any contact drawn on the screen adjacent to another contact already in the rung, will be linked to that existing contact.



For example, if a new contact is placed under the normally open contact in the above rung, the links will be drawn automatically.



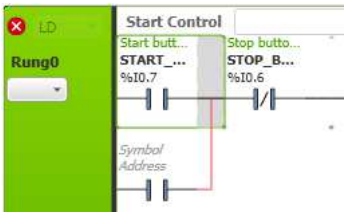
If the contact is not placed adjacent to another contact, the branches will have to be drawn manually. Links will also have to be redrawn manually if they are incorrectly placed by the automatic process.

Programming Rungs (cont.)

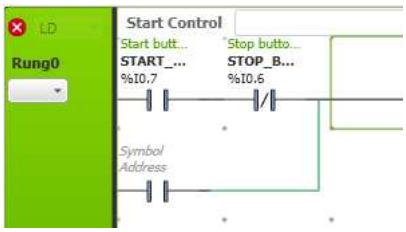
Manually Drawing Links

There are two tools to allow links to be drawn or erased. These are the draw line tool and the erase line tool.

The Erase Line tool will allow lines or links to be erased anywhere in the rung, except where another object has been placed. They are erased by dragging over the line or link to be erased. The line will appear red as it is being erased.



The Draw Line tool will allow lines or links to be drawn anywhere in the rung, except where another object has been placed. The line will appear green as it is being drawn.

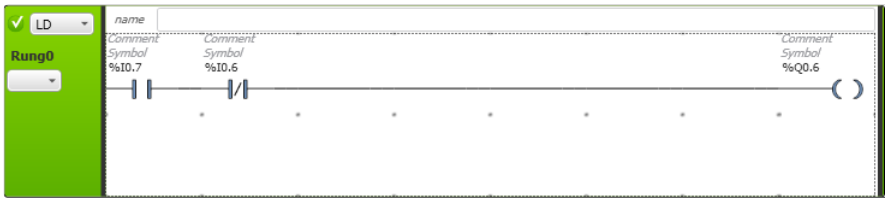


Programming Rungs (cont.)

Power Flow

A way of analysing ladder logic is to imagine power flowing through the contacts. If a contact is closed then power will flow through it and on to the next contact or coil.

In the following example, the normally closed contact will allow power to flow through but this does not answer the question of whether the output coil will turn on. Imagining power flow along the rung allows the rung to be analysed.

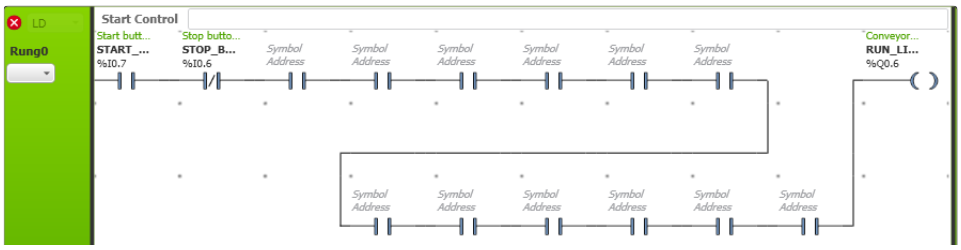


If the first contact is closed, ie input %I0.7 is true, then power will be applied to the left hand side of the normally closed contact. If the second contact is closed, ie input %I0.6 is false, then the power will be applied to the left hand side of the coil allowing it to turn on.

More complex rungs require more complex analysis but this concept of power flow and whether there is power at a certain point in the rung, remains the same. Thus any rung can be "solved" to see whether the output will turn on.

Reverse Power Flow

In the following example, there is a point in the middle of the rung where power flows from right to left instead of from left to right. This is known as Reverse Power Flow. The following shows an example of reverse power flow where power must flow back to the start of the second row of contacts.



Some programming packages allow reverse power flow when a rung is configured. However SoMachine Basic does not allow reverse power flow so care must be taken to avoid this when configuring a rung.

Program errors will be present if this is configured in SoMachine Basic.

Programming Rungs (cont.)

Start Control

The first section of the program will be responsible for Start Control. The start control will simply start all three conveyors at the same time.

In practice timers are often used to provide a staggered start to space out product and prevent current surges. Timed control of the conveyors will be performed as part of the stop control where it is much more important to clear product from the conveyors.

Exercise - Create the Start Control Program

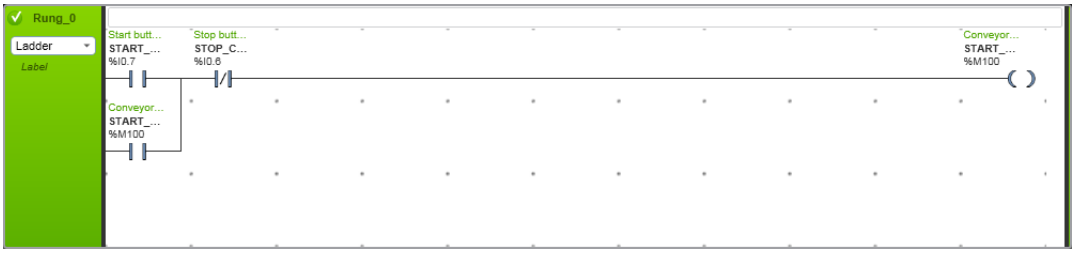
Learning Outcomes

By the completion of this exercise you will:

- Be able to use copy and paste to create program rungs

1 Modify the start control rung.

i. Click the descriptor for the coil and change the address from %Q0.6 to %M100. Similarly, change the descriptor for the latching contact to %M100

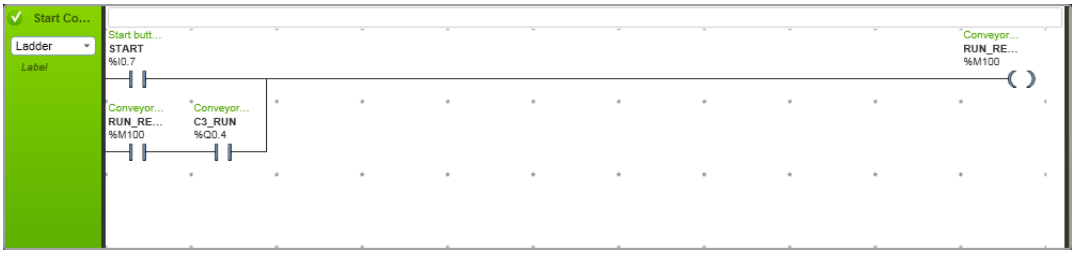


ii. Delete the Stop contact - this will be implemented in a different way as a controlled shutdown is required.

iii. Add two normally open contacts on the second row of the rung.

iv. Select the line drawing tool.

v. Click and hold at the right hand side of the contact at row 2 column 2 and drag the line up to join the top row. The completed rung should look like the following picture



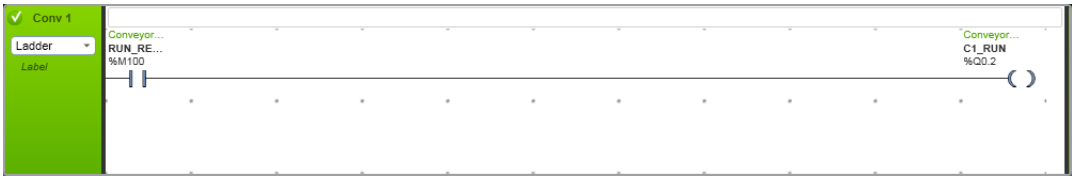
This will complete the change to the start control rung.

Exercise - Create the Start Control Program (cont.)

2 Create the code for conveyor 1.

i. In the rung "Conv 1" add a contact and a coil.

ii. Enter the address %M100 for the contact and %Q0.2 for the coil.



3 Create the code for conveyor 2.

i. Delete the existing **Conv 2** rung

ii. Select the **Conv 1** rung.

iii. Right-click the green area of the rung and select **Copy selected rung** from the menu.

iv. Right-click the empty area below the rung and select **Paste rung** from the menu.

v. Click the address above the coil and change it to %Q0.3.

4 Create the code for conveyor 3.

i. Select the contact and coil from the Conv 1 rung.

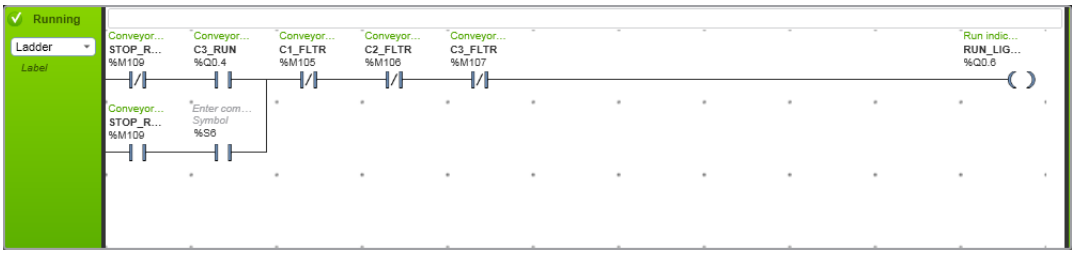
ii. Right-click the green area of the rung and select **Copy** from the menu.

iii. Right-click the green area on the left of the Conv 3 rung and select **Paste** from the menu.

iv. Change the address of the contact to %Q0.4.

5 Create the code for the running indicator lamp.

i. Program the "Running" rung with the following objects



6 Save the application.

Timers

Stop Control

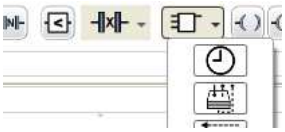
Stop control will perform a controlled stop of the conveyors.

Timers will be used to ensure product has travelled down the conveyor before each is stopped. This approach ensures that the conveyor stops and no product is left on the conveyor.

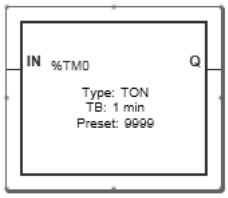
Each conveyor will have a separate timer which will accommodate different conveyor lengths and speeds.

How to Create a Timer

Clicking the **Functions** button on the Ladder Editor menu will open a drop-down menu of special functions. The first of these is the Timer function.



Select the Timer object and when the cursor is moved over the ladder rung a timer will be displayed underneath it.



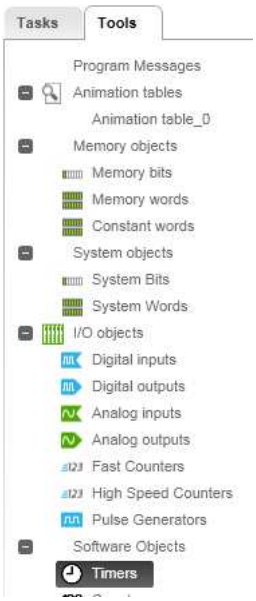
Place the timer at the desired location.

Timers (cont.)

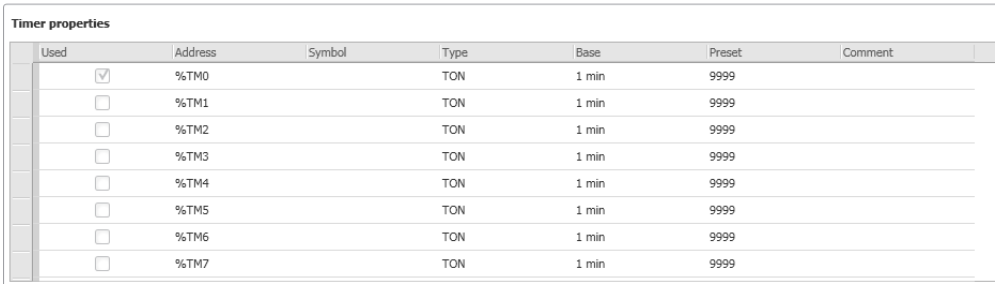
How to Configure Timer Parameters

Parameters must also be configured for the timer to set the type of timer and time value. This is done in the Timer Properties section.

Either double-click the timer function block in the ladder editor or go to the **Module Programming Tree,** select the **Tools** tab then under **Software Objects** select **Timers**.



The Timer Parameter configuration will be displayed at the bottom of the page.



Used	Address	Symbol	Type	Base	Preset	Comment
☑	%TM0		TON	1 min	9999	
☐	%TM1		TON	1 min	9999	
☐	%TM2		TON	1 min	9999	
☐	%TM3		TON	1 min	9999	
☐	%TM4		TON	1 min	9999	
☐	%TM5		TON	1 min	9999	
☐	%TM6		TON	1 min	9999	
☐	%TM7		TON	1 min	9999	

To change the type or base, double-click the column on the row containing the required timer. A drop-down box will appear allowing the type or timebase to be selected.

To change the symbol or preset, double-click the column on the row containing the required timer. A dialog box will open allowing the required information to be entered.

Exercise - Create the Stop Control Program

Learning Outcomes

By the completion of this exercise you will:

- Be able to create and configure timers in the program

1 Create the stop sequence ladder rung.

i. Add the following objects to the Stop Control rung:

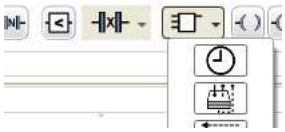
2 Create timer for the first conveyor.

i. In the Stop 1 ladder rung, place a normally open contact at the start of the rung. Click the Symbol text above the contact and enter `Stop`. A list of the previously entered symbols that contain "STOP" will appear.

ii. Select the "STOP_RELAY" symbol.

This is an alternative way of assigning an address to an object.

iii. In the functions menu bar, drop down the functions sub-menu and select the timer object.

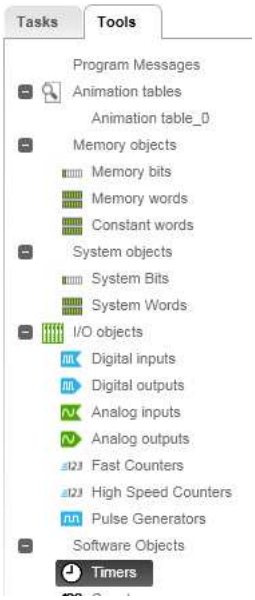


iv. Place the timer next to the contact

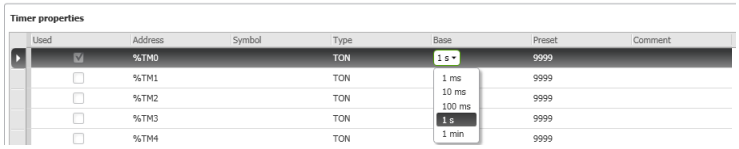
Exercise - Create the Stop Control Program (cont.)

3 Modify the parameters for the timer.

i. In the **Module Programming Tree**, select the **Tools** tab. Under **Software Objects**, select **Timers**.



ii. Double-click the **base** column on the row containing **%TM0**. A drop-down box will appear allowing the timebase to be selected. Choose the **1s** (1 second) timebase.



iii. Double-click the **Preset** column on the row containing **%TM0** and enter a preset value of `2`. This will set the timer to 2 seconds.

iv. Apply the changes.

4 Complete the rung.

i. Add a contact to the end of the rung and assign it to address **%M101**.

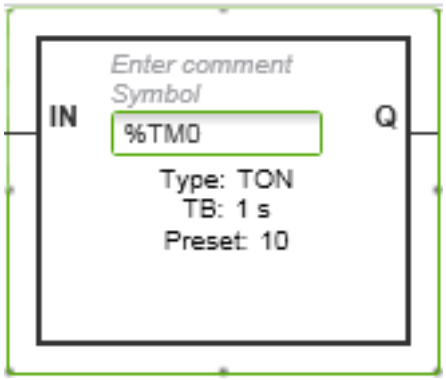
Exercise - Create the Stop Control Program (cont.)

5 Create the code to stop the other conveyors.

i. Copy the objects from the Stop 1 rung and paste them into the Stop 2 rung.

ii. Change the address of the coil to **%M102**.

iii. Double-click the timer address to change the timer number.



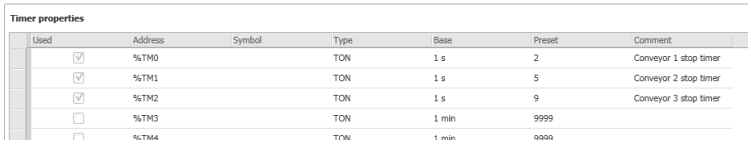
iv. Change the address of the timer to `%TM1`.

v. Paste the objects into the Stop 3 rung.

vi. Change the timer to **%TM3** and the output coil to address **%M103**.

6 Modify the parameters for the other timers.

i. Open the timer configuration and enter the following parameters for the timers.



ii. Accept the changes.

Exercise - Create the Stop Control Program (cont.)

7 Complete the stop control.

i. On the Conv 1 ladder rung add a normally closed contact and assign it to address **%M101**.

ii. On the Conv 1 ladder rung add a normally closed contact and assign it to address **%M102**.

iii. On the Conv 1 ladder rung add a normally closed contact and assign it to address **%M103**.

8 Save the application.

9 Download the application to the controller and test it.

i. When the start input, %I0.7, is operated. All the conveyor outputs will come on at the same time; %Q0.2, %Q0.3 and %Q0.4.

ii. When the Stop input is operated, the outputs will turn off in a timed sequence:

%Q0.2 will turn off after two seconds.

%Q0.3 will turn off after five seconds

%Q0.4 will turn off after nine seconds.

If the program does not work as expected, carefully check the programming for any mistakes, resolve them and try again.

Note:

If the program still does not work after checking, complete the rest of the exercises in this chapter. Diagnostic tools to help find programming problems are described in *Chapter 4 - Getting it all Working*.

Exception Handling

Exceptions

Exception handling is an important part of any application. If a problem condition can be identified and correctly handled it can prevent damage to equipment or loss of production.

Typical exception handling may be written for the following:

- Equipment being turned on but not running, such as a conveyor jam
- Motors not running at the correct speed
- Insufficient product
- No bottles or caps for a bottling machine
- Storage container overfill (if a high level probe fails)
- Product backup on conveyor

The last of these for example, may cause product to spill on the floor if there is not enough room on the conveyor and it is not stopped.

The following exercises will add code to handle the first two of these.

Note:

If any of these physical exceptions occur they are often referred to by engineers as faults. The fault is not part of the control system but with the process itself and if programmed correctly, the control system can handle the fault in a controlled way.

Exception Indication

As well as handling the exception, the program should also provide some kind of indication to the operator so they can take the appropriate action to resolve it. This can be in the form of indicator lights, horns and/or alarm displays. An output will be provided for lights/horns and the program must be written to operate these outputs under the appropriate circumstances.

Alarm displays are usually implemented using a graphic or text display and data connection to the controller. In most cases there must be some programming in the controller to accommodate this.

Exception Handling (cont.)

Conveyor Application

For this section of program, inputs %I0.2, %I0.3 and %I0.4 will simulate conveyor faults. A settling time will be provided for each conveyor and if the input is activated after this time then an exception will be generated.

The appropriate conveyor will be stopped along with all preceding conveyors to prevent product from being fed to the conveyor that has stopped.

The speed of a conveyor will also be monitored. This speed will be represented by an analog value on input %IW0.0. A low and high setpoint will be configured and if the speed of the conveyor is outside the range of the setpoints then an indication will be made to the operator. No other automation control is required.

A fault Indicator output will be provided on %Q0.7: It will be flashing for a conveyor speed error and permanently on for a conveyor fault.

Set and Reset Coils

Set and Reset coils

Set and Reset coils work slightly differently to normal coils.

When the preceding logic resolves to true a Set coil will turn the output on. The output will then stay on, even if the preceding logic changes state and resolves to false. The coil must be reset using a reset coil.

When the preceding logic resolves to true a Reset coil will turn the output off. If the preceding logic resolves to false, the Reset coil will not change the state of the output.

These usually operate independently in separate ladder rungs; the only link between them is that they operate on the same output.

Note:

If the same object appears later in the program as a normal coil, the logic of this rung will determine the state of the object. The Set and Reset will be ignored.

Exercise - Create an Exception Handling Program

Learning Outcomes

By the completion of this exercise you will:

- Use the Set and Reset coils.

Note:

The term fault is used to refer to a physical problem with the process hardware that is detected and controlled by the SoMachine Basic application.

1 Create new rungs for the exception handling.

i. Add five new rungs to the Fault Handling POU and give them the following names:

Conv 1 Flt

Conv 2 Flt

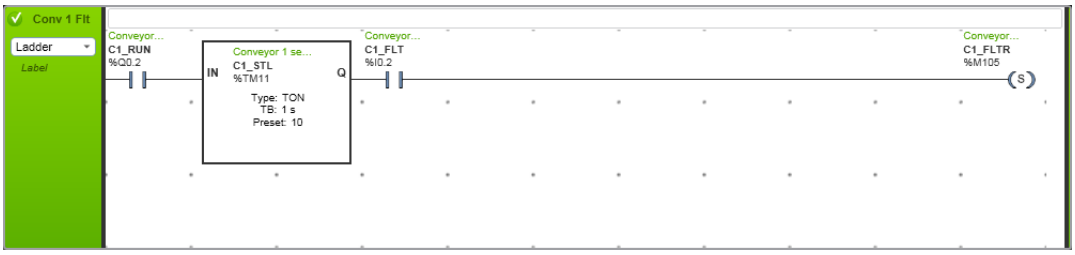
Conv 3 Flt

Reset

Fault Light

2 Program the settling timers.

i. Program the following objects for the settling timer and fault for conveyor 1.



Notice the use of the Set Coil not a normal coil. This will have to be placed using the **Set Coil** tool on the toolbar.

ii. Configure the following parameters for the settling timer.

iii. Configure the other two fault rungs and settling timers similarly.

Try to work out which contacts and coils are required and program them before turning the page.

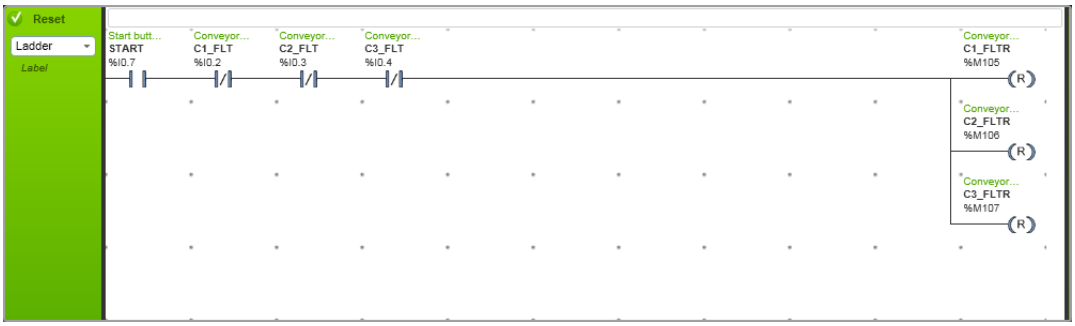
Exercise - Create an Exception Handling Program (cont.)

imer properties

Used	Address	Symbol	Type	Base	Preset	Comment
☑	%TM11	C1_STL	TON	1 s	10	Conveyor 1 settling time
☑	%TM12	C2_STL	TON	1 s	10	Conveyor 2 settling time
☑	%TM13	C3_STL	TON	1 s	10	Conveyor 3 settling time
☐	%TM14		TON	1 min	9999	

3 Add code to reset the faults.

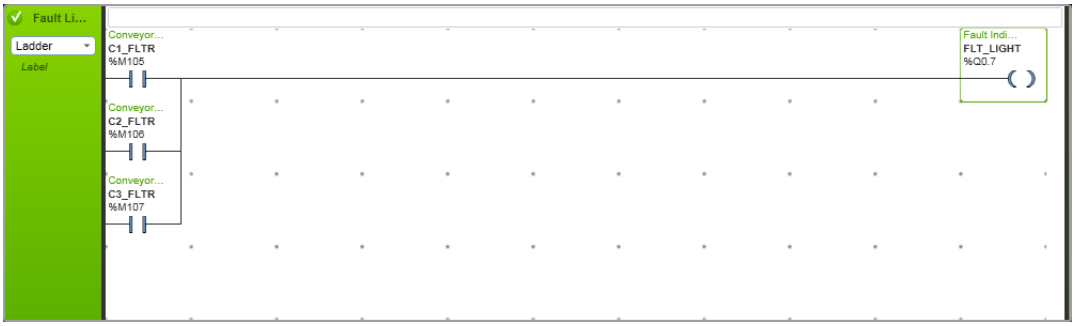
i. For each set coil, there must be a reset coil. These will be programmed into a single rung and will reset the circuits when all the faults have been removed and the start button is pressed.



Exercise - Create an Exception Handling Program (cont.)

4 Add code for the fault light.

i. Finally add the program for the fault light which will come on when there is a fault on any of the conveyors.



5 Save the application

The Operation Block

Setting Numeric Values

The setpoints for speed handling need to be set by the program at startup. SoMachine Basic provides a function for setting numeric values. It is called the Operation Block.

The Operation Block is used to perform a calculation or other operation and write the result to a register.

How to Create an Operation Block

The Operation block tool is located on the toolbar next to the Other Items button. It is a rectangle with three dots inside.

To create an Operation block select the tool and place the object on the right hand side of the rung.

Double-click the Operation Expression above the block to open an entry box and enter the required expression.

Note:

Operation blocks can only be placed on the right hand side of a ladder rung.

Conveyor Application

The speed of a conveyor will also be monitored. This speed will be represented by an analog value on input %IW0.0. A low and high setpoint will be configured and if the speed of the conveyor is outside the range of the setpoints then an indication will be made to the operator. No other automation control is required.

Exercise - Creating Setpoints

Learning Outcomes

By the completion of this exercise you will:

- Be able to use the Operation block

1 Create the ladder rung.

i. In the Speed Monitoring POU, create a rung and name it "Setpoints".

2 Create the programming for the rung.

i. Select the Operation Block tool and place the block at the right hand end of the rung. Place a second Operate block beneath it.

ii. Click the Operation Expression above the first block and an entry box will open.

Enter the expression `%MW1 := 625`

Notice that this is colon-equals, not just equals.

iii. Configure the second Operation block with the expression `%MW2 := 400`

Note:

Although this will perform the correct function and load the setpoints with the values, this will happen each time the logic is processed.

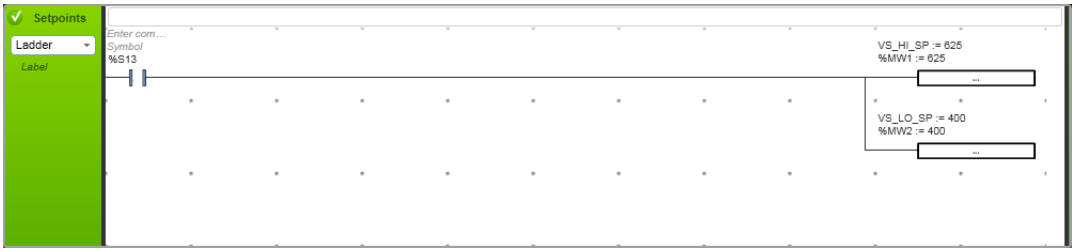
Sometimes the setpoints will need changing depending on the state of the process so it is better to load the setpoints once when the program starts up. The setpoints can then be changed without being overwritten by the program.

The system bit %S13 provides a single pulse when the program starts and remains off after that. This can be used for any initialisation in the program.

Exercise - Creating Setpoints (cont.)

3 Change the rung so that the values are loaded once when the program starts.

i. Place a normally open contact at the start of the rung and assign it to address %S13. The completed rung should look like the following.



4 Save the application.

The Comparison Block

Comparing Numeric Values

Sometimes it is necessary to compare values to see if a value is above or below another. For example, consider a storage silo weighing 200Kg and that silo can contain 100 Kg of product. If the silo is fitted with a load cell then the weight can be measured. If the weight exceeds 290 Kg then the storage silo can be considered full and the control system can ensure that no more product is put into that silo.

Comparing numeric values in SoMachine Basic is achieved using the Comparison block.

How to Create a Comparison Block

The Comparison Block tool is located on the toolbar next to the XOR and Functions. It is a rectangle with a left chevron (<) inside.

To create a Comparison Block select the tool and place the object on the rung in any position except the right hand side of the rung.

Double-click the Comparison Expression above the block to open an entry box and enter the required expression.

Conveyor Application

The speed of a conveyor is represented by an analog value on input %IW0.0 and will be compared with the low and high setpoints that were configured previously. If the speed of the conveyor is outside the range of the setpoints then an indication will be made to the operator. No other automation control is required.

Exercise - Comparing With the Setpoints

Learning Outcomes

By the completion of this exercise you will:

- Be able to use the Comparison block

1 Create the rungs for comparing the analog value.

i. Create two new rungs in the Speed Monitoring POU.

ii. Rename these rungs VS High and VS Low

2 Program the rung to compare with the high setpoint

i. Place a contact at the start of the rung and assign it to address **%M100**.

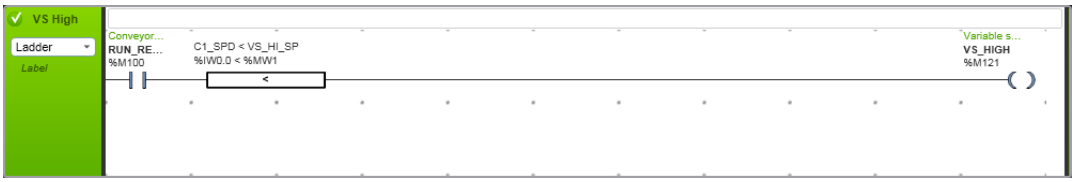
The comparison only needs to be done when the conveyor is running.

ii. Select the Comparison Block tool and place the block next to the contact.

iii. Click the Operation Expression above the Operation block and enter the expression **%IW0.0 > %MW1**.

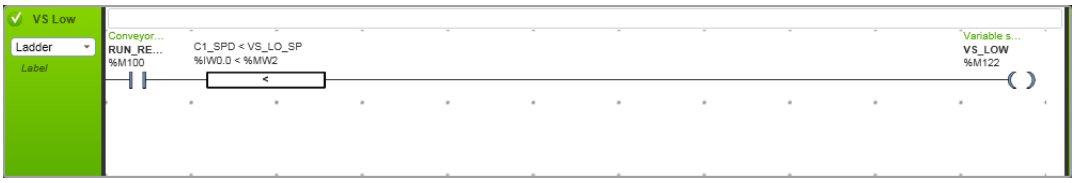
iv. Add a coil to the rung and assign it to address **%M121.**

The completed rung should look like this



3 Program the rung to compare the low setpoint.

i. Program the Low SP rung with the following objects and the expression **%IW0.0 < %MW2**



4 Save the application.

Linking the Program Parts

Completing the application

In this final exercise the programs in the separate POUs will be linked together by adding contacts to rungs that were previously created. This linking is typical of the way a program is developed with new functionality and is sometimes called hooking in to the existing application.

The speed errors will be amalgamated into a single contact and that contact will be used to make the fault light flash. The fault relays will also be used to stop the conveyors.

Exercise - Complete the Program

Learning Outcomes

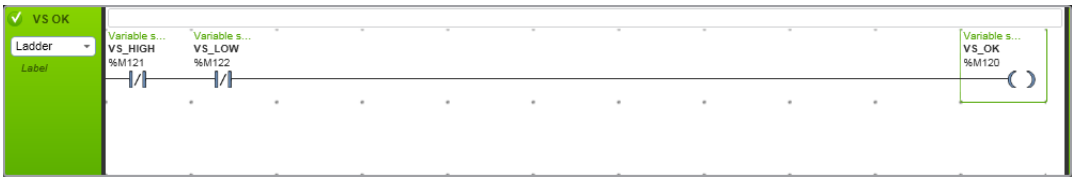
By the completion of this exercise you will:

- Complete the conveyor application

1 Add a rung to indicate that the variable speed is OK.

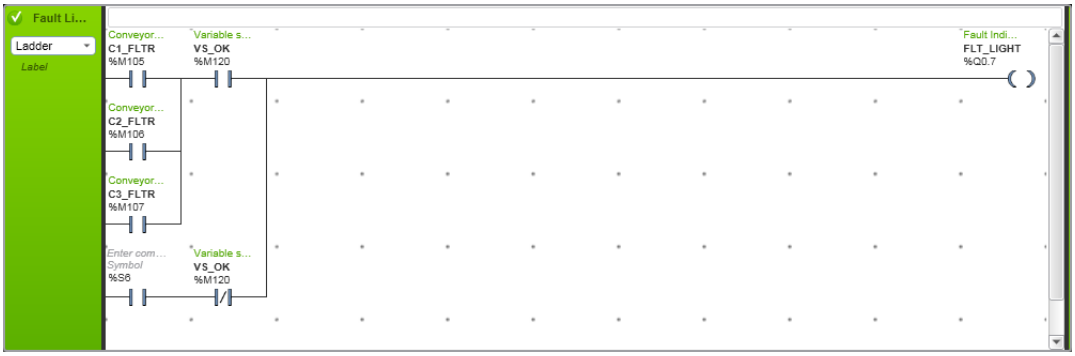
i. Create a new rung in the Speed Monitor POU and rename it VS OK

ii. Add the following objects.



2 Modify the fault light rung to make the light flash when the speed is not within the range of the setpoints.

i. Modify the Fault Light rung to the following



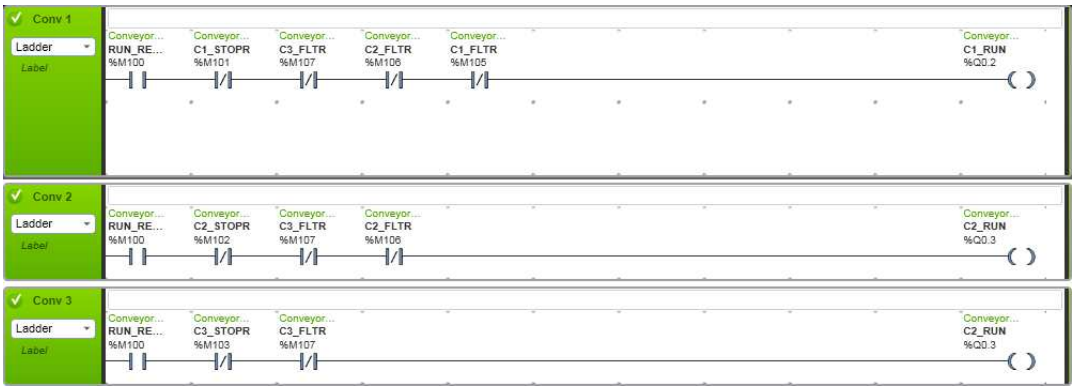
Note:

This contains a contact using the address %S6, This is a system bit which turns on and off with a 1 second interval. It is often used to create cyclic actions such as flashing a light.

Exercise - Complete the Program (cont.)

3 Modify the conveyor run rungs to make the conveyors stop when a fault occurs.

i. Modify the conveyor rungs to the following.



4 Save the application

Conclusion

Conclusion

This chapter covered the following topics:

- ***Languages*** (page 3-5)
- ***Addressing*** (page 3-5)
- ***Symbols*** (page 3-30)
- ***Program Structure*** (page 3-30)
- ***Programming Rungs*** (page 3-36)
- ***Timers*** (page 3-42)
- ***Fault Handling*** (page 3-48)
- ***Set and Reset Coils*** (page 3-50)
- ***The Operation Block*** (page 3-54)
- ***The Comparison Block*** (page 3-57)

Questions (Programming a Simple Application)

The following questions are to check understanding of the topics covered in this chapter:

- Which languages are supported by SoMachine Basic
- What type of object is %SW1 and can you write to this object?
- What is a quick way to create multiple symbols?
- How can an output be turned off when it has been turned on using a Set coil?
- Which block can be used by the program to write values into objects?

Chapter 4: Getting it all Working

Overview

Introduction

After creating the application and writing the program, the next step may be to find out why it does not work as intended. The issue may be due to the M221 Logic Controller, an incorrectly written program or the field wiring or devices.

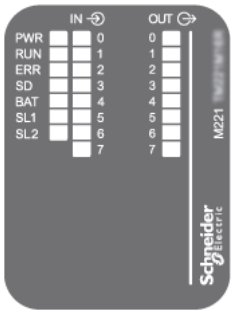
SoMachine Basic provides a number of tools to help with faultfinding the M221 Logic Controller and the Application.

This Chapter Covers These Topics:

Visual Clues

LEDs

The LEDs provide a visual indication of the status of the M221 Logic Controller. They can show when the program is not running, and combinations can sometimes give an indication why. They also give an indication of the state of various parts of the M221.



PWR LED

The first LED shows the state of the power supply to the M221 controller. If the PWR LED is off then the 24V power supply to the M221 is not working. The first thing to check would be the wiring and the power supply itself.

If the PWR LED is not on there is no point looking for any other problems as this must be resolved first.

During normal operation this LED should be permanently ON.

RUN LED

The next LED to look at is the RUN LED. If this is permanently on then the program is running and any fault is likely to be in the program itself.

If the RUN LED is blinking slowly (and the ERR LED is off) then the M221 is in Stop mode. This can be for several reasons but the first thing to check is the RUN/STOP switch on the front of the M221. This switch must be up for the program to run. If the switch is up then the SoMachine Basic software must be used to put the M221 controller in Run. Alternatively, changing the RUN/STOP switch down to the STOP position and then back up to RUN will set the controller in run mode.

If the RUN LED is off, there is no application in the M221 Logic Controller.

During normal operation this LED should be permanently ON.

LEDs (cont.)

ERR LED

The ERR LED can indicate various faults in the M221 Logic Controller.

If it is permanently on, this indicates an exception in the M221 which has caused it to stop. The program cannot be run and no communication with the M221 is possible. To recover, the firmware must be reloaded into the M221.

If the ERR LED is flashing, this indicates that the M221 has a valid application but there is an internal error. If the Run LED is on, then the program in RAM is not the same as the stored program. This typically occurs when an on-line modification has been made and not saved to the M221. If the Run LED is off, then this indicates an internal error. Limited communication is allowed to resolve the error but the program cannot be run.

A single flash when power is applied to the M221 indicates that there is no application loaded into the controller. Full communication with the controller is possible in order to download an application.

During normal operation this LED should be permanently OFF.

SD LED

The SD LED indicates when the M221 Logic Controller is accessing the SD card. This will usually be during a firmware upgrade or data transfer.

During normal operation this LED should be permanently OFF.

BAT LED

The BAT LED indicates the state of the battery. A slow flash indicates that the battery is low. When the LED is permanently on, this indicates that the battery is flat.

During normal operation this LED should be permanently OFF.

SL LED

The SL LED indicates when the M221 Logic Controller is communicating via the Serial Port. If the M221 has more than one serial port, there will be two LEDs labelled SL1 and SL2 giving indication for ports 1 and 2.

During normal operation the state of this LED will depend on the application.

Exercise - LED status

Learning Outcomes

By the completion of this exercise you will:

- See some of the LED indications for the M221 Logic Controller

1 If SoMachine Basic is not running, start it and open the Conveyor Control Application.

2 Make an online modification and observe the effect on the LEDs.

i. Connect to the M221 Logic Controller and ensure that the program in the controller is the same as the program in SoMachine Basic.

ii. Make a small change to a ladder rung. This can be any change such as changing a normally open contact to normally closed or changing the address of an object.

iii. Click off the rung and when the message appears asking if the change is to be permanent, click the No button.

iv. Observe the effect on the LEDs.

3 Remove the battery.

i. Power off the M221 controller and remove the battery from the unit.

ii. Apply power to the M221 Controller and observe the effect on the LEDs.

iii. Remove the power for at least 30 seconds, insert the battery, apply power to the M221 and observe the effect on the LEDs.

Looking at the Program

Displaying Values on a Rung

When SoMachine Basic is connected to the M221, the values of objects are displayed on rungs. If an output does not turn on when expected, the logic can be checked and the cause determined.

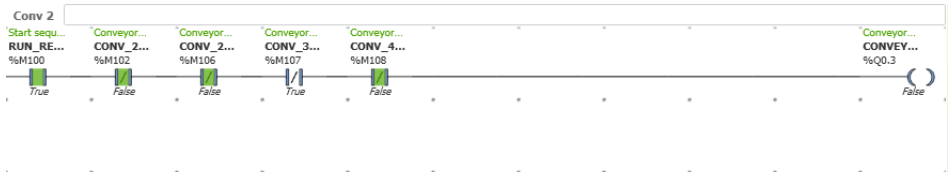
The colouring of the contacts will indicate power flow. Green colouring shows when a contact will allow power to flow and a contact that is not coloured will not allow power flow.

The state shown under each object also gives the state of the contact or coil, showing whether it is true or false.

Note:

The True/False indication shows the state of the input, not the state of the coil or contact. The green colouring shows the state of the contact. This can be confusing for inverted contacts and coils but a green colour will always signify power or on.

Analysing a Rung



In the above example the outputs controls conveyor 2 which is not running. Looking at the displayed values on the rung, %M100 which is the Run Relay is shown as true so the conveyor should be running. However, %M107 which is Conveyor 3 fault is also shown as true and no power will be allowed through this normally closed contact.

Even though there is no fault on conveyor 2, the fact that another conveyor has developed a fault will also stop this one. The program has been designed to behave this way. This method of analysing a rung can be helpful to determine why something is not happening as expected, especially when it is being prevented by something in another part of the program.

Exercise - Values in Rungs

Learning Outcomes

By the completion of this exercise you will:

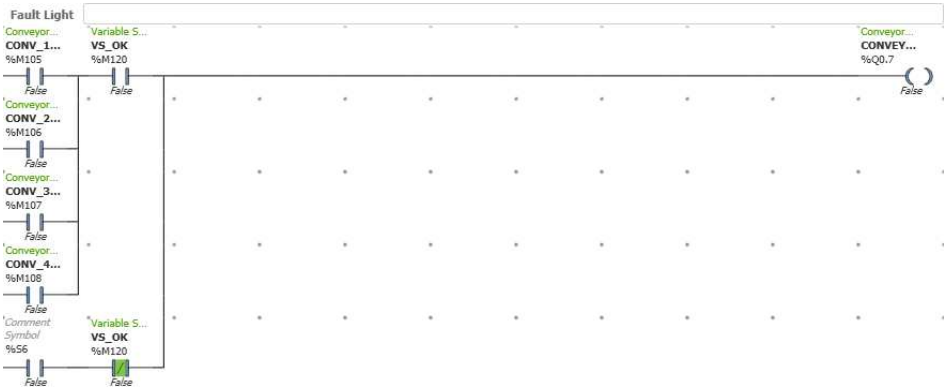
- Use the values in displayed in rungs to fault find the program

1 If SoMachine Basic is not running, start it and open the Conveyor Control Application.

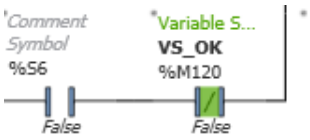
2 Investigate why the fault light is flashing.

i. Display the **Fault Light** rung located in the **Fault Handling** POU.

ii. Identify the part of the rung that is causing the fault light to flash.



This leg contains %S6 which is the system bit cycling with a 1 second interval. This is in series with **%M120** the normally closed contact is false which is causing the fault light to flash.

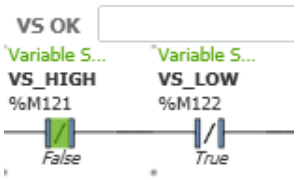


The symbol for this is **VS_OK**.

Exercise - Values in Rungs (cont.)

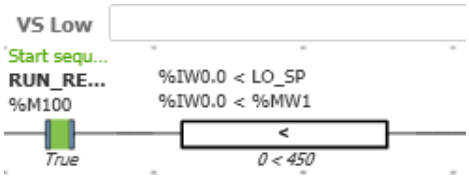
iii. Display the **VS OK** rung located in the **Speed Monitoring** POU.

The VS_OK object is turned on when the speed is not low and not high. Looking at the state of the two contacts, the VS_LOW contact is true so the low speed means that the speed is not OK.



iv. Display the **VS Low** rung located in the **Speed Monitoring** POU.

The Run Relay (%M100) is true so the error must be in the comparison. The variable speed low setpoint (VS_VO_SP) is 400 but the actual speed of conveyor 1 (C1_SPD) is equal to one. The comparison is successful and the low speed turns on.



The conveyor is either not running or there is no signal from the device monitoring the conveyor speed.

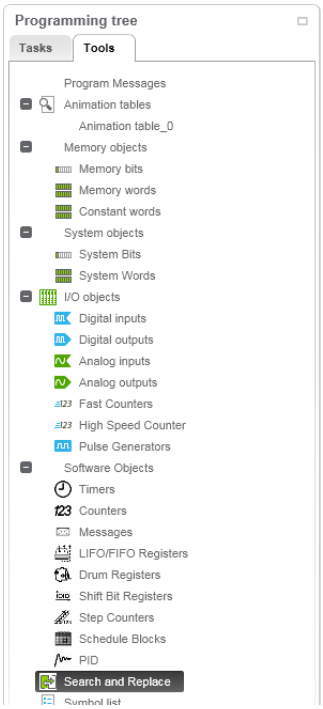
(In this case, the signal is not connected)

Searching for Things

Using the Search

The previous method of navigating around the project is possible if the rungs have been given meaningful names of the project is familiar. If not then the object must be searched for.

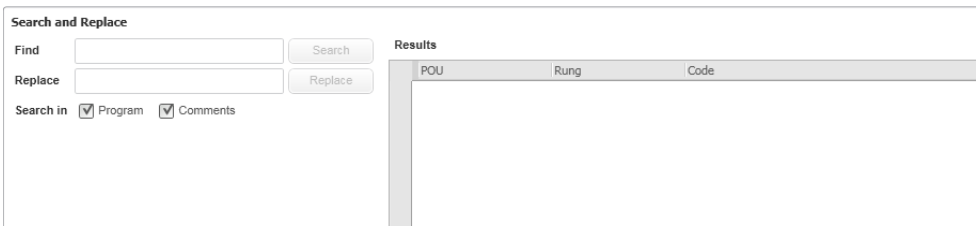
The search function in SoMachine Basic makes finding things easy. At the moment it only works on rung names or descriptions so when programming, make sure meaningful rung names and descriptions are entered. The search will not work for symbols.



To start a search, go to the **Programming** tab and select **Tools** in the **Programming Tree**. Near the bottom under **Software Objects**, select **Search and Replace.**

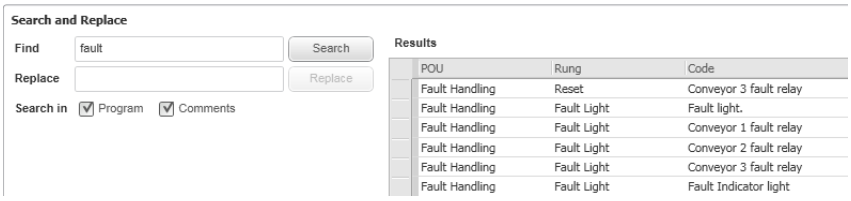
In the section that opens at the bottom of the screen, type the required search into the **Find** dialog box and click the **Search** button (Pressing the return key will not work). A list of matching descriptions will be displayed. The wildcard is not required as the list will include all descriptions that contain the matching string.

The search can only be used to search the code.



Using what has been Found

After the search has produced a list of matching items, these can be used to navigate around the application.

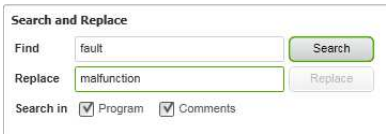


Clicking an item in the list will go to the rung where that object has been programmed and place the cursor on the object.

Replacing Things

Using Replace

When something has been found, the replace may be used to a piece of text in a description to something. For example, if all descriptions contained the word "fault" and this needs to be changed to "malfunction", enter the word malfunction into the replace dialog box and click the **Replace** button (once again, pressing the return key will not work).



This function can only be used for replacing ALL instances of a string.

Exercise - Searching for Things

Learning Outcomes

By the completion of this exercise you will:

- Search and replace in SoMachine Basic.

1 If SoMachine Basic is not running, start it and open the Conveyor Control Application.

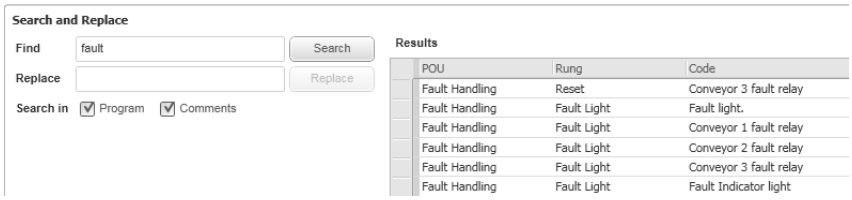
2 Search for objects associated with faults.

i. Click the **Programming** Tab.

ii. Click the **Tools** Tab in the **Programming Tree.**

iii. Select **Search and Replace** under **Software Object**s

iv. In the **Find** dialog box enter the word `fault` and click the **Search b**utton.



3 Navigate using the search.

i. Click one of the items in the search results and observe the effect on the ladder rungs displayed.

ii. Select the first item in the list.

iii. Select the second item in the list and observe the effect on the ladder rungs displayed. The rung should not change but the cursor will move to the next object.

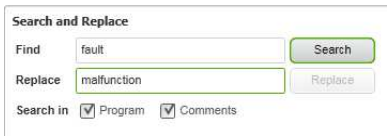
iv. Click the third item in the list.

v. Select other items to see other parts of the conveyor fault programming.

Exercise - Searching for Things (cont.)

4 Replacing items.

i. In the Find dialog box enter the word `malfunction` and click the **Replace** Button. Note that all instances of "fault" have been replaced with "malfunction".

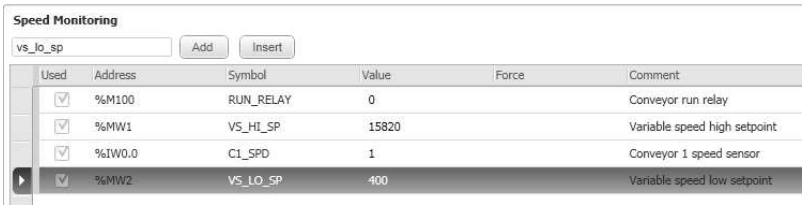


ii. Confirm this by clicking the **Search** button. SoMachine Basic will search for all occurrences of the word "fault" as this is still in the search box. The result will be the message "No search results available" as they have all been replaced.

Looking at Values

Animation Tables

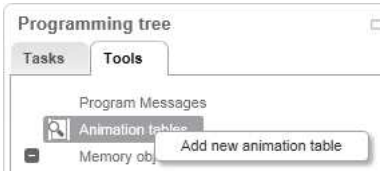
If the values of many objects need to be viewed at the same time then animation tables provide a way of doing this.



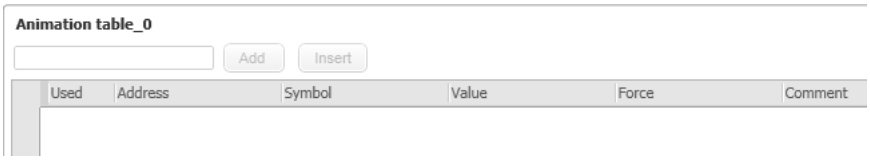
Animation tables allow multiple objects to be displayed in a single table along with their values. Multiple animation tables can be created and saved for different debugging sessions.

How to Create an Animation Table

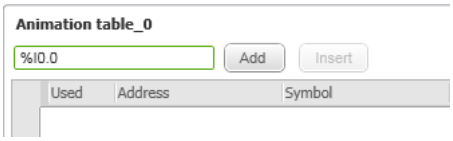
To create an animation table, go to the **Tools tab** of the **Programming tree.** Right-click **Animation tables** and select **Add new animation table** from the menu.



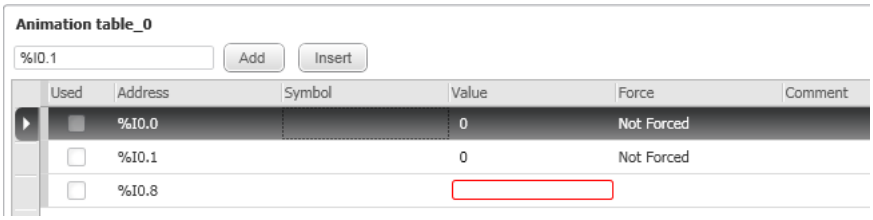
An empty animation table will be created where new objects can be added.



In the entry box, enter either the symbol or address of an object and click the **Add** button.



If the object is found, the value will be displayed if known. If the object is not found, a red box will be displayed indicating an error.



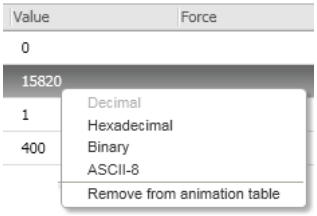
Looking at Vales (cont.)

Changing the Display Format

The format of values displayed in animation tables can be changed to help when displaying data in different addressing formats. The allowable display formats are:

Decimal	(65)
Hexadecimal	(0041)
Binary	(0000000001000001)
ASCII	(A)

Right-click a value displayed in an animation table and a list of display formats will be listed in the pop-up menu. Choose the appropriate format for the data displayed.



Exercise - Animation Tables

Learning Outcomes

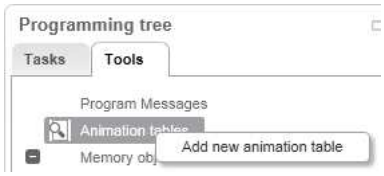
By the completion of this exercise you will:

- Create an animation table to show values of objects.

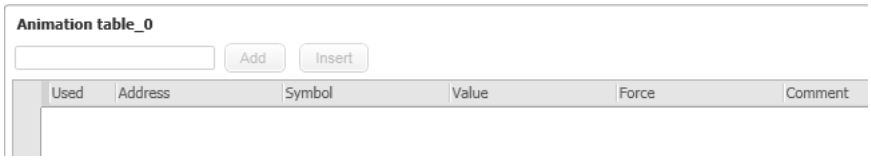
1 If SoMachine Basic is not running, start it and open the Conveyor Control Application.

2 Create an animation table.

i. Go to the **Tools tab** of the **Programming tree.** Right-click **Animation tables** and select **Add new animation table** from the menu.



ii. In the entry box enter the object `%M100`. Click the **Add** button.



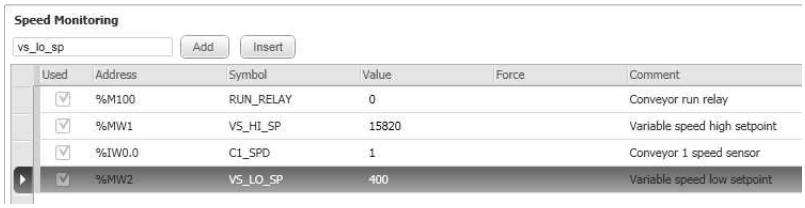
iii. Add the following objects to the animation table:

VS_HI_SP

C1_SPD

VS_LO_SP

This table will allow the speed check to be carried out very quickly. If the system is running, the actual speed should be between the low and high setpoints.



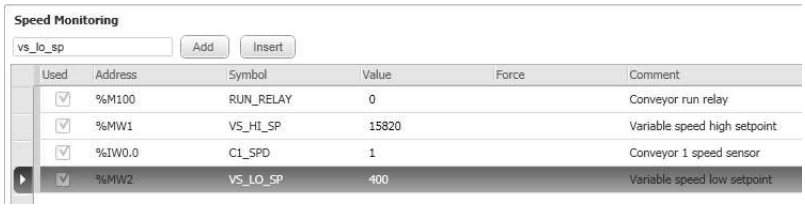
The order of the objects in the table has been carefully chosen so that when everything is running correctly, the analogue values decrease from top to bottom. If they do not, this indicates a problem.

Exercise - Animation Tables (cont.)

3 Rename the animation table to indicate its purpose.

i. In the Programming tree, right-click the animation table **Animation table_0** and select **Rename animation table** from the menu.

ii. Change the name to `Speed Monitoring`.



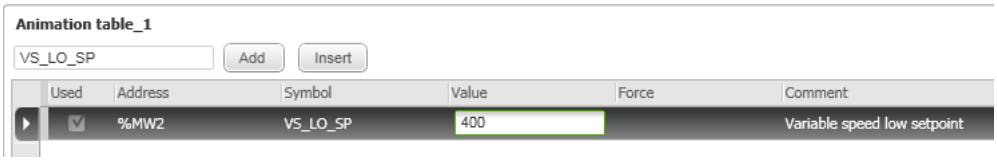
4 Save the application.

Changing Values

Modifying a Value

In some cases, values in animation tables can be changed as well as viewed.

Most word values can be changed using the SoMachine Basic animation tables. Click the value displayed and a dialog box will open allowing the value to be changed.



Note:

If the program writes a value to an object, the value of that object may still be changed using an animation table but the next time the program logic is solved, the value will be overwritten.

Fixing a Value

Sometimes it can be helpful to fix a value so that it does not change. This is particularly true for the logical sections of a program where states can change quickly making it difficult to see what is happening. This can also be useful for testing something that does not happen very often.

Setting a digital object such as an input, output or coil to a fixed state is known as forcing.

Forcing objects removes them from program control and can cause unexpected and uncontrolled operation of machinery or the process.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

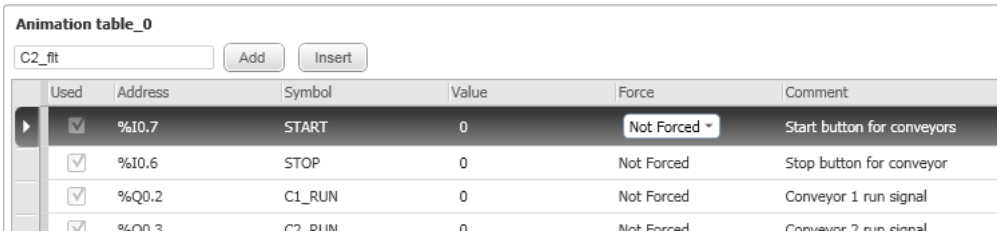
- Do not force objects while the system is running
- Do not force objects without knowing the effect

Failure to follow these instructions can result in death, serious injury, or equipment damage.

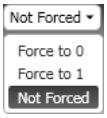
Forcing

How to Force an Object

Objects can only be forced from an animation so the first step is to create an animation table which contains the object(s) to be forced. The animation table can also contain other objects to be monitored to see the effect of the force.



Click the force column of the animation table and a drop-down box will appear, giving the choices Force to 0, Force to 1 or Not Forced. Select either Force to 0 or Force to 1 depending on the desired state.



How to Remove Forces

The Not Forced option in the animation table can be used to remove the force for a single object. Click the Force column in the animation table and select **Not Forced** from the menu.

If the object is not in an animation table or there are multiple objects to be unforced, these must be added to an animation table in order to unforce them.

When looking at a section of program, forced items can be identified by the letter 'F' displayed underneath the coil or contact, in front of the True/False state.

Exercise - Forcing

Learning Outcomes

By the completion of this exercise you will:

- Force a value

1 If SoMachine Basic is not running, start it and open the Conveyor Control Application.

2 Create an animation table

i. Create an animation table containing the following objects:

START_BUTTON

STOP_BUTTON

C1_RUN

C2_RUN

C3_RUN

C2_FLT

3 Using Force to control the application.

i. Click Not Forced in the Force column for the START_BUTTON symbol. This will select the object and allow it to be forced.

ii. Drop down the selection box and select **Force to 1** from the menu.

The process will start and the outputs controlling the conveyors will turn on (%I0.2, %I0.3 and %I0.4).

Exercise - Forcing (cont.)

iii. Drop down the selection box for the START_BUTTON object force and select **Not forced** from the menu.

The object will display the value 0 as the input is not on but the outputs will remain on as they have been latched by an internal contact.

iv. Select the STOP_BUTTON object, drop down the selection box and select **Force to 1** from the menu.

The shutdown sequence will operate until all the outputs are off.

v. Drop down the selection box and select **Not forced** from the menu.

4 Forcing an object on to test the output.

i. Drop down the selection box for the C1_RUN object force and select **Force to 1** from the menu.

The output for conveyor 1 (%I0.2) will come on.

ii. Drop down the selection box for the C2_RUN object force and select **Force to 1** from the menu.

The output for conveyor 2 (%I0.3) will come on.

iii. Drop down the selection box for the C3_RUN object force and select **Force to 1** from the menu.

The output for conveyor 3 (%I0.4) will come on.

 **Note:**

This process allows outputs field wiring and output devices to be tested without running the application.

5 Remove all forces from the program.

i. For each item that is forced, drop down the force dialog box in the animation table and choose the **Not Forced** item from the menu.

Exercise - Forcing (cont.)

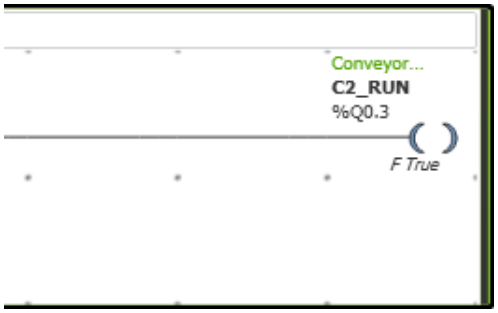
6 Forcing an object to a fixed state.

i. Force the **START_BUTTON** object on to start the process. The outputs controlling the conveyors will turn on (%I0.2, %I0.3 and %I0.4).

ii. Set the **START_BUTTON** object to **Not forced**.

iii. Force the **C2_RUN** object to 1. There will be no visible difference on the front of the M221 as the output is already on.

iv. Display the **Conv 2** rung in the Control **POU**.



Notice that the display shows the letter F in front of the value indicating that the output is forced.

v. Select the STOP_BUTTON object and force it to 1.

As before, the stop sequence will operate but one output will remain on. This will be the forced output %Q0.3.

vi. Start the process again in the same way, then force the C3_FLT object to 1.

This should stop the faulty conveyor (conveyor 3) and all conveyors feeding it. Note that the output for conveyor 2 stays on.

Note:

This shows how an object can be forced with no visible effect outside the programming software and leave equipment in a dangerous state unable to stop when required.

Summary

Summary

This chapter covered the following topics:

- ***LEDs*** (page 4-2)
- ***Displaying Values on a Rung*** (page 4-5)
- ***Search and Replace*** (page 4-8)
- ***Animation Tables*** (page 4-12)
- ***Changing Values*** (page 4-16)

Questions

The following questions are to check understanding of the topics covered in this chapter:

- Which LEDs should NOT be illuminated during normal operation?

- Is it possible to search for an object name?

- What options are there for changing a value in an animation table?

Chapter 5: Taking it Further

Overview

Introduction

The basic application that has been developed is suitable for simple machine or process application. However, the M221 has many capabilities that make it suitable for more complex application. This section looks at a few of those capabilities.

This Chapter Covers These Topics:

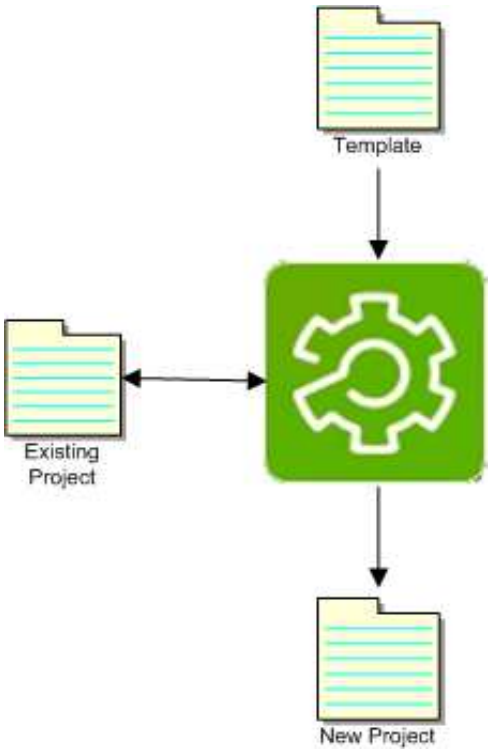
Templates

Using Templates

Templates are a way of creating a standard application that can be used as the basis for any new applications.

Taking the conveyor application that has been created, if an OEM or integrator creates many systems that have the same basic conveyor control system, this application can be saved as a template and used to create all new applications. This saves time recreating the basic application.

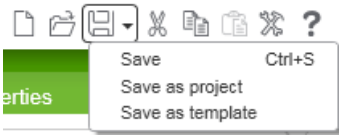
When an existing project is used to create a new project, the existing project may be accidentally overwritten when the new project is saved. The template is simply the project stored in a special format which can then be used to create new projects. When the new project is created from a template and saved, a new project must be created; the template cannot be overwritten.



Templates (cont.)

How to Create a Template

With a project loaded, select **Save As Template** from the **Save** drop-down menu.



This will save the project as a template. However, it will not be visible on the start page unless it is saved in the correct directory. Templates should be saved to the directory

\Program Files\Schneider Electric\SoMachine Basic\Examples

The template will then be visible from the start page.

Exercise - Templates

Learning Outcomes

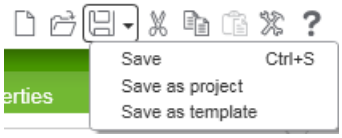
By the completion of this exercise you will:

- Create a SoMachine Basic Template

1 If SoMachine Basic is not running, start it and open the Conveyor Control Application.

2 Save the application as a template file.

i. Drop down the menu next to the save button and select **Save as Template** from the menu.



ii. Select the directory to save as:

\Program Files\Schneider Electric\SoMachine Basic\Examples

iii. Click the **Save** button to save the template.

3 Create a new application based on the template.

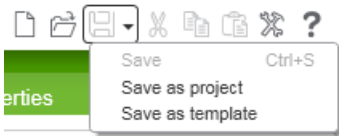
i. Click the SoMachine Basic icon to go to the start page.

Exercise - Templates (cont.)

ii. Select the **Templates** tab on the left hand side.

Select the Conveyor Control template and click the **Open Template** button.

iii. When the new application has been created, drop down the save selection box.



Note that the Save option is greyed out and the template cannot be overwritten.

Programming Languages

Instruction List

Instruction list is a low-level programming language for SoMachine Basic. It uses a single memory location called an accumulator to load and combine values before the result is stored in an object. It is not the easiest programming language but it is the most flexible as results can be achieved that are not possible in other programming languages. However, as SoMachine Basic has the ability to switch between Instruction List and other programming languages some restrictions are placed on this flexibility.

The following is an example of Instruction List program.

```
LD          %I0.7
OR          %Q0.2
AND NOT     %I0.6
STR         %Q0.2
```

When a SoMachine Basic application is downloaded to the M221 Logic Controller, the program is converted to Instruction List as this is what the controller actually processes. This can lead to some programming errors only being discovered when the application is downloaded.

Ladder

Ladder is a popular programming language as it is similar to electrical diagrams and visually, it is very easy to see what the program is doing. With the addition of in-line status display, it is also very easy to debug.

It consists of a series of program lines or rungs, so called because they look like the rungs of a ladder. To the left of the rung are a set of inputs and conditions that must be solved. To the right of the rung is an object (or objects) defining what to do with the result.

A ladder rung may look something like the following:

```
%I0.0  %I0.1              %Q0.7

--]  [------]  [-----------------(  )--
```

Ladder does have some limitations though. It cannot perform all functions due to limitations in the programming rules and it must be converted to Instruction List before it is processed by the logic controller.

Programming Languages (cont.)

Switching Between Languages

There are two ways to switch between languages in SoMachine Basic.

To change the language of a single rung, drop down the language selection box in the rung and choose the required language.

To change all the rungs at the same time, click either the **IL > LD** button or the **LD > IL** button.

Using the Ethernet Port

Types of Addressing

The IP address can be configured in one of three ways:

- Fixed IP
- DHCP
- BOOTP

For fixed IP addresses, the address is simply entered into the configuration section, along with the Subnet mask and Gateway if required. For DHCP or BOOTP there must be a DHCP or BOOTP server on the network to give the device its address.

Fixed IP

With Fixed IP addressing, the IP address and subnet mask are entered as part of the M221 application configuration. The gateway can also be configured if required. This means that the M221 has a known IP address but does require management of the IP address if the M221 is on anything but a small network.

Enter the required address details into the configuration form and click the **Apply** button.

DHCP

DHCP uses a DHCP server to manage IP addresses so is much easier to configure within the M221. It is simply a case of selecting the DHCP Radio button.

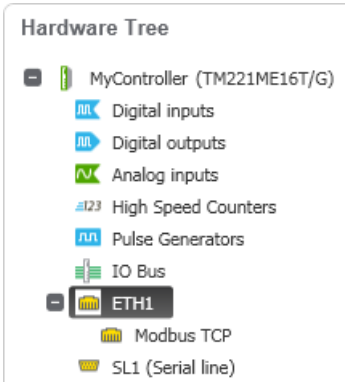
If there is just one M221 on the network it can be easily identified. Device discovery in the Connection tab will show the network address of the M221 which can be selected to allow the software to connect to the controller.

If there are more controllers, then the network addresses of all the controllers will be visible. Currently it is difficult to distinguish between controllers as they do not announce their device name. The current work-around is to use USB to ensure that the correct controller is being programmed. This will be addressed in a future release of the software.

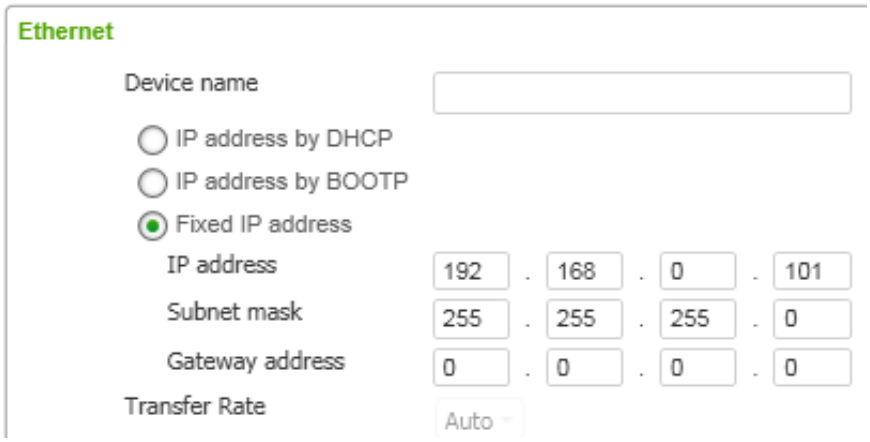
Using the Ethernet Port (cont.)

How to Set the IP Address

The IP address is configured as part of the hardware configuration on the configuration page. Select **ETH1** from the **Hardware Tree.**



The Ethernet configuration will be displayed.



Choose the required options and click the **Apply** button.

Exercise - Ethernet Connection

Learning Outcomes

By the completion of this exercise you will:

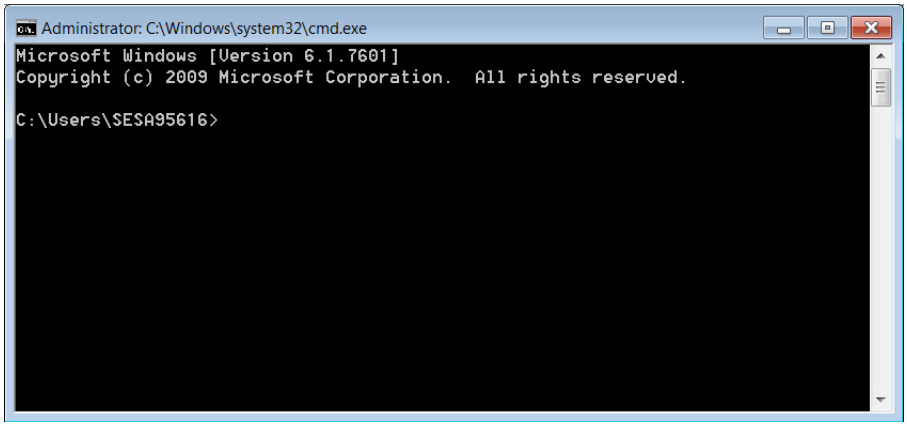
- Configure the Ethernet connection to download and commission an application

1 Find the IP address of the programming computer.

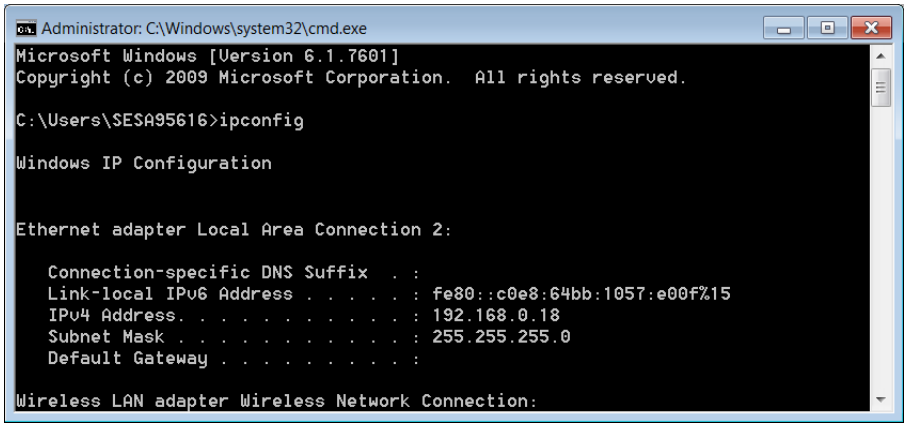
i. If using Windows XP, go to the Start menu select Run.

If using Windows Vista or 7, go to the start menu and click the search box.

ii. Type CMD and press enter.



iii. In the command prompt window enter the command ipconfig.



There may be more than one adapter each with its own configuration. Look for the one that is connected to the network containing the M221. This will most likely be the one labelled "Local Area Connection".

Exercise - Ethernet Connection (cont.)

iv. The two numbers required are the IP (or IPV4) address and the subnet mask. Write them in the first two rows of the table below.

Programming Computer IP Address	
Subnet Mask	
M221 IP Address	

v. The subnet mask for the M221 will be the same as the subnet mask for the programming computer.

vi. The IP address for the M221 will be in the same IP range as the programming computer. To determine this apply the following rules. Where the subnet mask is not equal to zero, then the IP number must be the same. Where the subnet mask is equal to zero, the IP number will be different and a free number on the network. An example is given below.

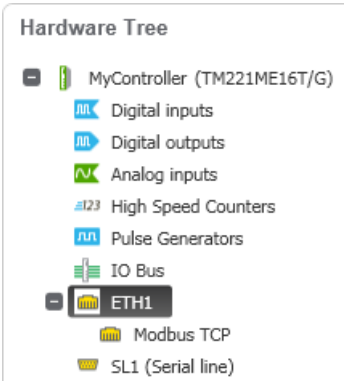
Programming Computer IP Address	192 . 168 . 0 . 4
Subnet Mask	255 . 255 . 255 . 0
M221 IP Address	192 . 168 . 0 . 101

vii. Enter your own IP address into the third row of the first table in step (iv).

2 Configure the Ethernet Port.

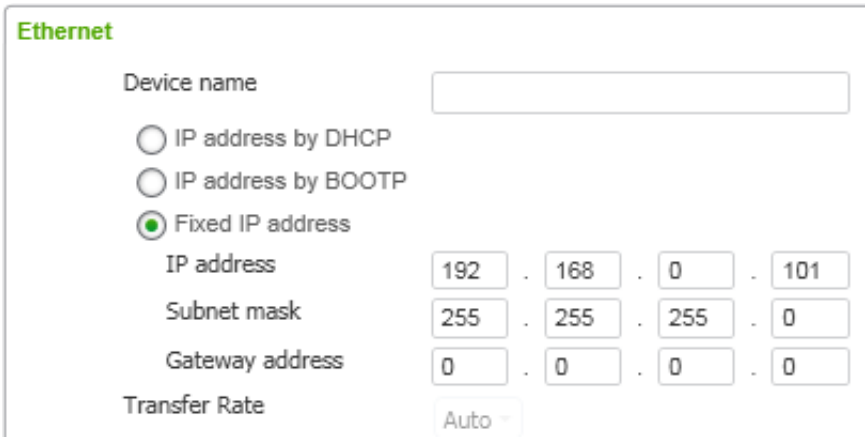
i. Use the USB cable to connect to the M221 Logic Controller.

ii. Go to the configuration tab and from the Program Tree select **ETH1.**



Exercise - Ethernet Connection (cont.)

iii. Select the **Fixed IP address** radio button and enter the M221 IP Address and Subnet Mask from the table in step 1(iv) on the previous page. The screenshot below shows the Ethernet address configured from the example table.



Note:

If you are on a large network do not continue with the exercise. The steps so far show how to configure the IP address but continuing may cause a duplicate IP address on the network.

iv. Click the **Apply** button.

Exercise - Ethernet Connection (cont.)

3 Download to the M221 controller.

i. Connect to the M221 Logic Controller via USB and download the application.

ii. When the download has completed run the controller.

iii. Disconnect the USB cable

4 Reconnect via Ethernet

i. Connection can be made in two ways. Either:

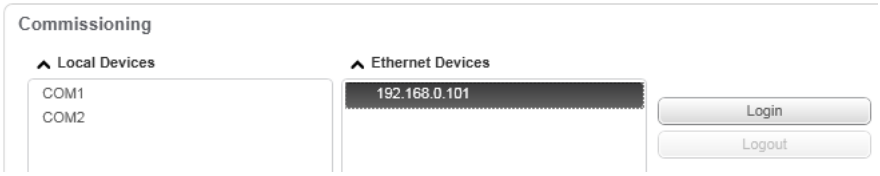
Connect an Ethernet cable to the Ethernet port on the front of the M221 Logic Controller and an existing switch on the network.

or

Connect a crossover cable to the Ethernet port on the front of the M221 Logic Controller and plug the other end directly into the Ethernet port of the programming computer.

ii. SoMachine Basic will recognise the Ethernet connection and display the Ethernet address in the Ethernet Devices column.

Select the IP address and click the **Login** button.



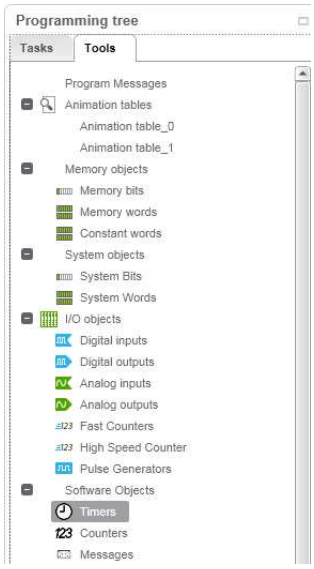
iii. Go to the **Programming** tab and observe the status of the objects in the program and animation tables to confirm that SoMachine Basic is communicating with the M221 Logic Controller.

Exercise - Ethernet Connection (cont.)

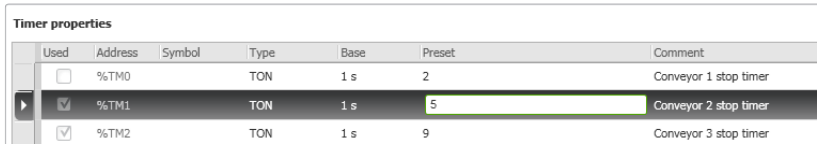
5 Program the M221 Logic Controller using the Ethernet connection.

i. Go to the **Commissioning** tab and logout from the controller.

ii. Go to the **Program** tab and select **Timers** from the **Program tree**



Change the time value for one of the conveyor stop timers.



iii. Go back to the **Commissioning** tab, make sure the Ethernet connection is still selected and login. Transfer the program as before.

Using the Serial Port

Serial Port

All M221 Logic Controllers are fitted with at least one serial port to allow serial communication to other devices. Those without an Ethernet port will have two serial ports fitted. The serial port uses a RJ45 connector so a special cable is required.

The serial port can be configured for RS232 or RS485 operation using either Modbus or ASCII protocols.

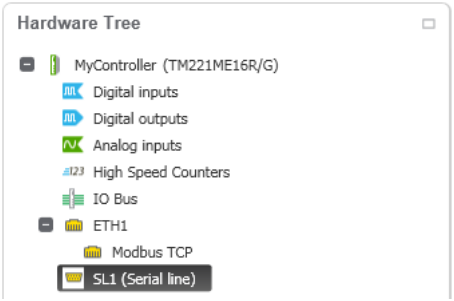
Setting the Parameters

There are several settings that can be configured on the serial port. These are divided into two categories, physical settings and protocol settings.

The physical settings allow the baud rate, parity and number of data and stop bits to be configured. RS232 or RS485 can also be selected here.

The protocol settings allow Modbus ASCII, Modbus RTU or plain ASCII to be selected and protocol specific settings configured for each.

To access the configuration settings, select **SL1 (Serial line)** from the **Hardware Tree** on the **Configuration** tab.



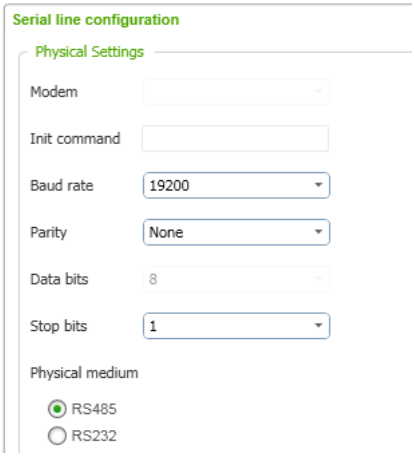
Note:

Where fitted, selecting **SL2 (Serial line)** will configure parameters for the second serial port.

Using the Serial Port (cont.)

Physical Settings

The physical settings allow the physical characteristics of the serial port to be configured. These are the Baud rate, Parity, Data bits and Stop bits. It also allows either RS232 or RS485 to be selected,

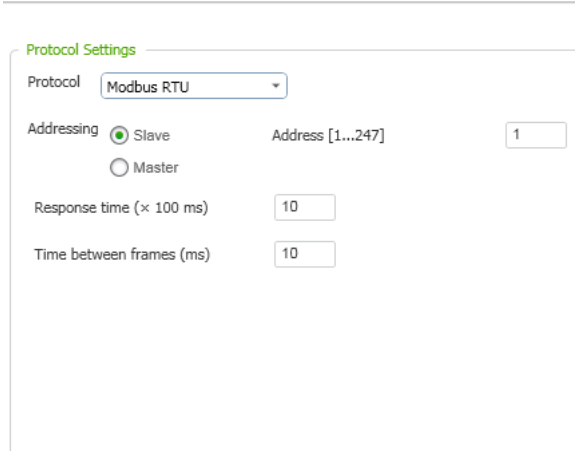


If the protocol is set to one of the Modbus options then the Data bits selection will be greyed as this must always be 7 data bits.

Choosing the correct settings will depend on the application. A lower Baud rate for example, will improve communications in a noisy environment but at the cost of slower communication speed. Whatever settings are selected they must match the settings of the device to which the M221 is communicating.

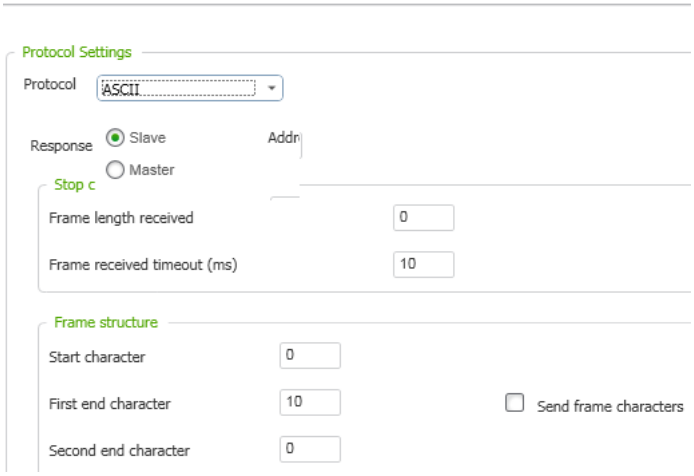
Protocol Settings

The options for protocol settings will also depend on whether Modbus or ASCII is selected. If either Modbus ASCII or Modbus RTU is selected then the following options will appear:



The response time and time between frames are used for fine-tuning the communications. The main setting is whether the M221 Logic Controller is the master on the network or a slave. If it is a slave then the address is also configurable.

If the ASCII protocol is selected the following protocol options are available:



These settings are for fine-tuning the network and mainly used where interference is causing message loss or corrupted messages.

Using the Serial Port (cont.)

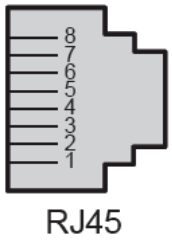
Wiring

The serial port uses a RJ45 connector so a special cable is required.

If communicating with a XBT display then the VW3A8306Rxx cable with a RJ45 connector on both ends is required.

If communicating with a computer or printer then the TCSMCN3M4M3S2 cable is required. This has a RJ45 connector on one end and a 9 pin socket on the other.

If a special cable is required then the connections are given in the table below:



Pin	RS232	RS485
1	RXD	Not Connected
2	TXD	Not Connected
3	RTS	Not Connected
4	Not Connected	D1 (A+)
5	Not Connected	D0 (B-)
6	CTS	Not Connected
7	Not Connected	Not Connected
8	0V Common	0V Common

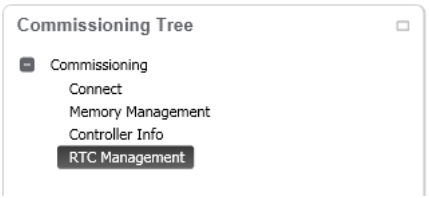
Note:

If making up a serial cable, all rules concerning RS232/RS485 distance and shielding must be followed to avoid errors or loss of communication. If in doubt, refer to a RS232 or RS485 wiring guide.

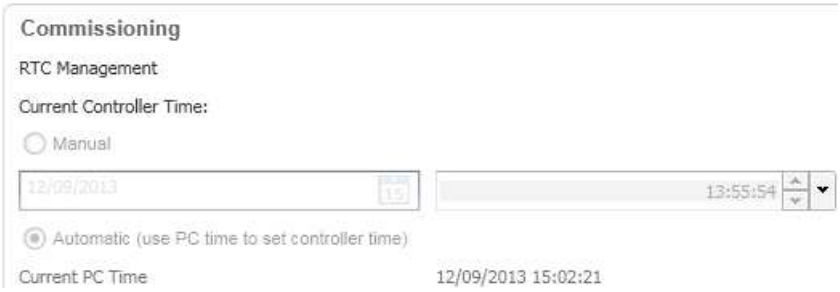
Real Time Clock

Setting the Clock

Settings are made in the RTC Management section which is in the **Commissioning** section of the **Commissioning Tree**.



There are two modes for setting the clock. The clock can either be set manually to a specific time and date or it can be synchronised to the computer time and date.



Using the Clock

The values for the Real Time Clock are accessible through five system registers which can be used to access the time and date.

Register	Content	Format
%SW49	Day of the week	xN where N=1 for Monday
%SW50	Seconds	00SS
%SW51	Hour and Minute	HHMM
%SW51	Month and Day	MMDD
%SW53	Century and Year	CCYY

These values can be used for time-stamping information and events or measuring event duration.

Comparison blocks may be used to trigger an event at a particular time or on a particular date. For example, comparing %SW51 with a value representing a particular time can trigger an event at that time.

The values are in BCD but by viewing the values in Hexadecimal format, the date and time can be seen.

Exercise - Using the Real Time Clock

Learning Outcomes

By the completion of this exercise you will:

- Be able to use the Real Time Clock in an application

1 Create a new application in SoMachine Basic

i. Start SoMachine Basic and create a new application.

ii. Assign the correct controller to match the hardware.

iii. Save the application and give it the name "RTC Example".

2 Create the program.

i. Go to the programming tab and rename the POU, calling it "RTC".

ii. Rename the rung calling it "Time".

iii. Check the current time and add 3-4 minutes. Write this time here.

_____

iv. Add a compare block to the rung and enter the expression %SW51 = 16#nnnn where nnnn is the time written above without any formatting characters. For example if the time now is 10:45, the expression would be `%SW51 = 16#1048`.

v. Add an output coil and assign the address **%Q0.0** so the first output will turn on when the time is reached.

3 Display the values in an animation table.

i. Create an animation table and add the status words for the Real Time Clock; **%SW49, %SW50, %SW51 and %SW52.**

ii. Click the value displayed in the table and a selection box will allow the format to be change

Choose Hexadecimal for the four values.

4 Connect to the controller and test the application.

5 Change the RTC and test the application again.

6 Reset the M221 Controller clock to the correct time.

Summary

Summary

This chapter covered the following topics:

- ***Templates*** (page 5-2)
- ***Programming Languages*** (page 5-6)
- ***Using the Ethernet Port*** (page 5-8)
- ***Configuring the Serial Port*** (page 5-15)
- ***Real Time Clock*** (page 5-19)

Questions

The following questions are to check understanding of the topics covered in this chapter:

- What file extension do template files have?

- What type of IP addressing is supported by SoMachine Basic/M221?

- Which serial port protocols are supported by SoMachine Basic/M221?

- What can the M221 Real Time Clock be used for?

Chapter 6: Documenting the Application

Overview

Introduction

Documenting the application is an important step to ensure the way the application works can be understood in the future. This applies to any engineers who are unfamiliar with the application and sometimes the engineer who wrote it if they are coming back to it after a long period of time.

SoMachine Basic provides many ways to document the application which will be explored in this chapter.

This Chapter Covers These Topics:

Object Documentation

Symbols

If the symbol names are chosen carefully, they will help to document the application. Consider the following rung of ladder:

```
%I0.0                %I0.1                                %Q0.7

--]  [----------------------]  [-------------------------------------(  )--
```

What does this rung actually do?

Without some other information it is impossible to work out the function of this rung. Now consider the following rung of ladder.

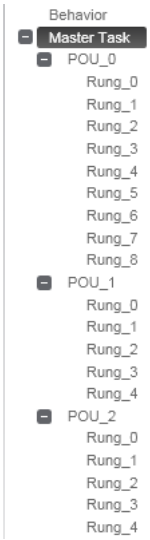
```
CONVEYOR             INTERLOCK                            CONVEYOR
START                                                     RUN SIGNAL

--]  [----------------------]  [-------------------------------------(  )--
```

It immediately becomes much more obvious what the rung is doing simply by reading the object labels. This self-documentation is used by many engineers as an aid to programming and to understanding the program in the future.

The program created in *Chapter 3 - Programming a Simple Application* uses many meaningful symbol names making the program easy to understand without any additional comments.

Naming POUs and Rungs

Similarly, choosing meaningful names for POUs and Rungs can help anyone understand what the program is doing and where to find parts of the program.



The names in the above image are not very helpful but the ones below are much more descriptive.

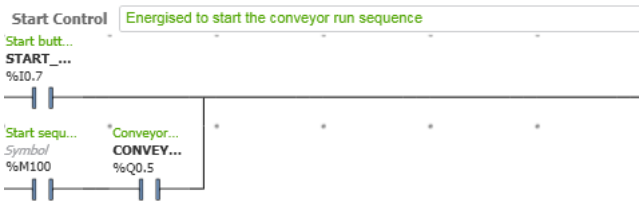
The program created in *Chapter 3 - Programming a Simple Application* uses meaningful POU and Rung names to make the program easy to navigate without additional cross referencing.

Comments

Comments

Comments can be added to each rung to describe the action of the rung.

While symbols can help to document the application, often the logic of the rung must be inspected to understand the function of the entire rung. A comment attached to the rung can immediately explain the purpose of that rung.



How to Add a Comment

At the top of each rung there is a grey box.

Click the grey box to add a comment.

Exercise - Add Comments to the Application

Learning Outcomes

By the completion of this exercise you will:

- Add comments to rungs of the application.

1 View the Conveyor Control Application

i. If SoMachine Basic is not running, start it.

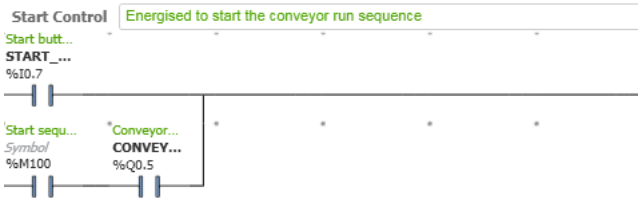
ii. Open the **Conveyor Control** application.

If in doubt, refer to the Exercise - Managing Applications which shows how to do this.

2 Add comments to the control POU.

i. Go to the programming tab and select the **Control** POU.

ii. Select the **Start Control** rung and click the grey box at the top of the rung. Enter the text `Energised to start the conveyor run sequence`.



iii. For the remaining rungs of the **Control** POU enter the following comments:

Conv 1	Conveyor 1 control
Conv 2	Conveyor 2 control
Conv 3	Conveyor 3 control
Running	Run indicator
Stop Control	Stop request - Initiate phased shutdown
Stop 1	Stop signal for conveyor 1
Stop 2	Stop signal for conveyor 2
Stop 3	Stop signal for conveyor 3

Exercise - Add Comments to the Application (cont.)

3 Optionally, for completeness enter the following comments for the Fault Handling and Speed Monitoring POUs:

Conv 1 Flt	Conveyor 1 fault
Conv 2 Flt	Conveyor 2 fault
Conv 3 Flt	Conveyor 3 fault
Reset	Reset conveyor faults when start button is pressed
Fault Light	Fault light
VS High	Compares with high setpoint - energised if speed is high
VS Low	Compares with low setpoint - energised if speed is low
VS OK	Speed OK
Setpoints	Initialise speed voltage setpoints on startup (voltage is x10 speed)

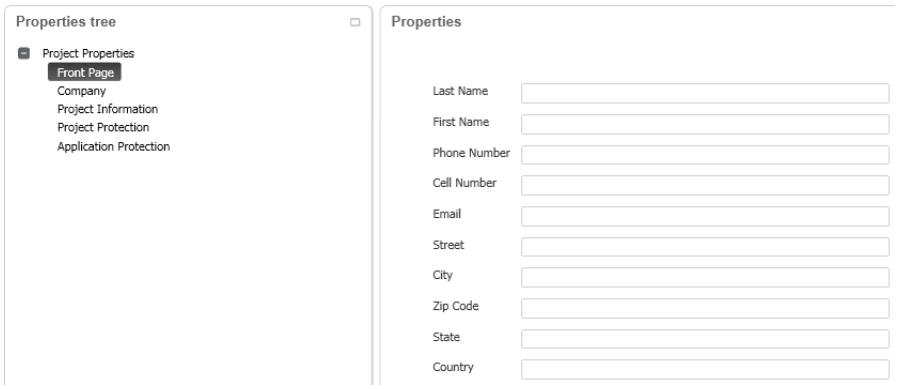
Application Information

Application

There is some information that will apply to the entire application. This includes who wrote it, where the application is being used and what the application does. The properties section in SoMachine Basic allows this information to be recorded as part of the application.

Front Page

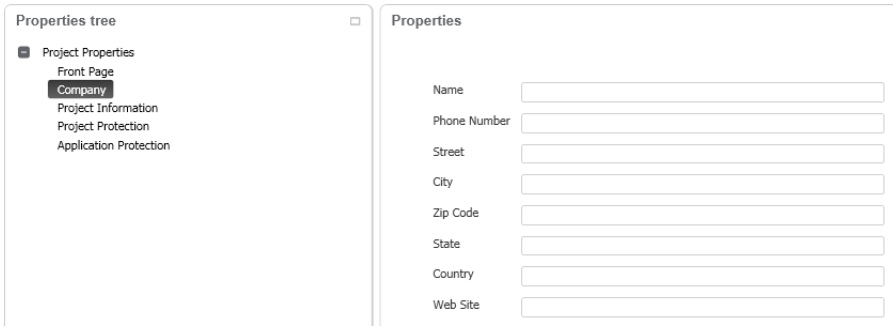
The Front Page is used to store information about the engineer and company that produced the application



This is of particular use to OEMs and integrators as it gives a point of contact if the application requires updating and the customer does not have the ability to do it themselves. It can also be used internally to identify the actual engineer who wrote the application and provide contact details.

Company

This can store information about the company using the application.



This is of particular use OEMs and integrators who have many customers as it documents where the application is being used.

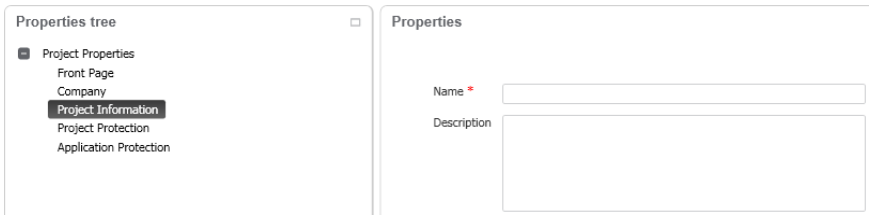
Application Information (cont.)

Company (cont.)

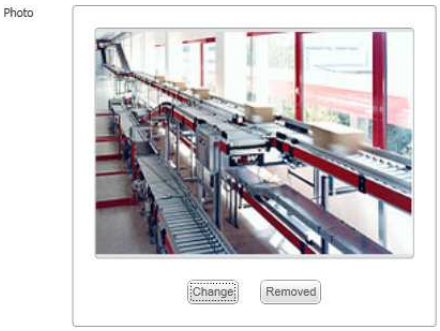
In the Company section there is also space for the company logo.

Project Information

The Project Information section is where information about the project itself can be stored.



This can be the name of the project, a description and a photo. The photo can be useful to show where the application is being used.



Exercise - Enter Application Information

Learning Outcomes

By the completion of this exercise you will:

- Use the Properties tab to enter basic information

1 View the Conveyor Control application.

i. If SoMachine Basic is not running, start it.

ii. Open the **Conveyor Control** application.

If in doubt, refer to the Exercise - Managing Applications which shows how to do this.

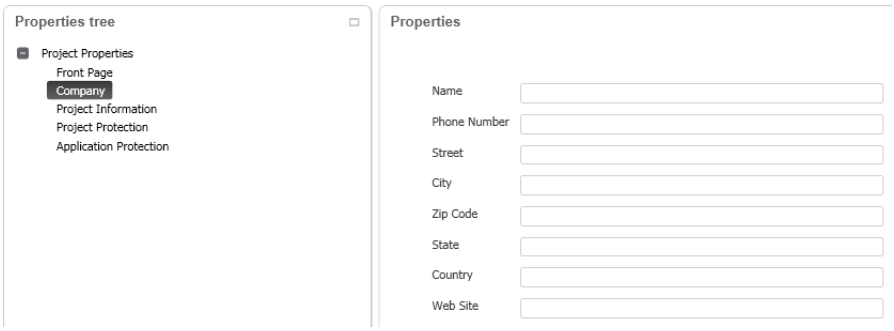
2 Enter the programmer information.

i. Go to the properties tab and ensure that **Front Page** is selected in the Properties tree.

ii. Enter your details into the form and click the **Apply** button.

3 Enter the company information.

i. Select **Company** from the Properties tree.



Exercise - Enter Application Information (cont.)

ii. Enter the information for the local Schneider Electric office and click the **Apply** button to apply the changes.

4 Change the company logo.

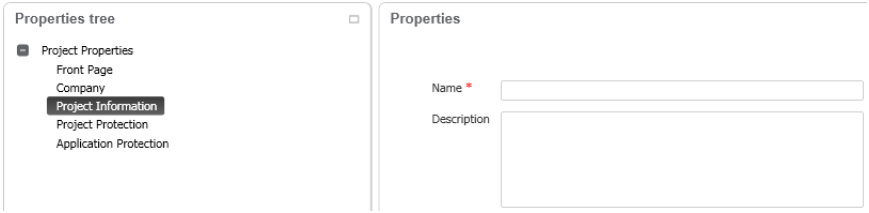
i. Under **Company Logo**, click the **Change** Button.

ii. There is a Schneider Electric logo on the DVD located in the same folder as this document. Navigate to the folder and select **Schneider Electric.jpg**. Click the Open button.

iii. Click the **Apply** button to apply the changes.

5 Enter the project information

i. Select **Project Information** from the Properties tree.



ii. Enter the name `Conveyor Control Application` into the **Name** field. Give the application an appropriate description. Click the **Apply** button to apply the changes.

Exercise - Enter Application Information (cont.)

6 Change the photo.

i. Under **Photo**, click the **Change** Button.

ii. There is a picture of a conveyor system on the DVD located in the same folder as this document. Navigate to the folder and select **Conveyor.png**. Click the **Open** button.

iii. Click the **Apply** button to apply the changes.

7 Save the application.

Summary

Summary

This chapter covered the following topics:

- ***Object Documentation*** (page 6-2)
- ***Section Documentation*** (page 6-3)
- ***Comments*** (page 6-4)
- ***Application Information*** (page 6-7)

Questions

The following questions are to check understanding of the topics covered in this chapter:

- How do meaningful POU and Rung names help?

- Where can comments be added to the program?

- What information can be stored in the Front Page?

Appendix A: Answers to Questions

Introduction

The following pages give suggested answers to the questions asked at various points in the document. These are mainly the end of chapter questions.

This Chapter Covers These Topics:

End of Chapter

- Name the four configuration sections for SoMachine Basic

  Properties, Configuration, Programming and Commissioning

- SoMachine Basic can be used to program which Machine Solutions logic controllers?

  [Only the] M221

- What are the two ways to connect to the Logic Controller for programming and commissioning?

  USB or Ethernet (where fitted to the logic controller)

End of Chapter

- What voltage power supply can be used to power the M221?

  24V DC

- What is the maximum I/O supported by the M221?

  144

- The HE10 connector is fitted to which of the M221 Logic Controllers?

  M221s with 32 I/O built in

  (TM221M32TK and TM221ME32TK)

Identifying Addresses

1 Explain the following addresses.

Address	Int/Ext	Location	Data Type
%I0.4	External	Logic Controller	Bit
%MW2	Internal	Variable Memory	Word
%Q0.6	External	Logic Controller	Bit
%IW1.2	External	First Expansion Module	Word
%S6	Internal	System	Bit
%MF10	Internal	Variable Memory	Float
%KW2	Internal	Constant memory	Word
%SW100	Internal	System	Word

2 Create an address for the following.

i. The second digital input of the logic controller

%I0.1

ii. An internal memory location that can be used to store the value 17.4

%MW1 (The location is chosen to suit the application)

iii. System word 8

%SW8

iv. The third output of the second expansion module (assuming that it is a digital output)

%Q1.2

v. The first input of the second digital output module

This will depend on exactly where the expansion module is located as it could be in any position following the first.

Programming a Simple Application (cont.)

End of Chapter

- Which languages are supported by SoMachine Basic

  Ladder and Instruction List.

- What type of object is %SW1 and can you write to this object?

  A System Word that cannot be written.

- What is a quick way to create multiple symbols?

  Export the symbol list and use Excel

- How can an output be turned off when it has been turned on using a Set coil?

  Normally using a Reset Coil (but it can also be turned off if the object is programmed as a normal coil later in the program)

- Which block can be used by the program to write values into objects?

  The Operate Block.

End of Chapter

- Which LEDs should NOT be illuminated during normal operation?

  ERR, SD and BAT.

- Is it possible to search for an object name?

  No.

- What options are there for changing a value in an animation table?

  Change the value of an object or forcing.

Taking it Further

End of Chapter

- What file extension do template files have?

  .smbe

- What type of IP addressing is supported by SoMachine Basic/M221?

  Static, DHCP and BootP

- Which serial port protocols are supported by SoMachine Basic/M221?

  Modbus ASCII, Modbus RTU and ASCII

- What can the M221 Real Time Clock be used for?

  To trigger events that need to happen at a certain time and/or on a certain day and for time-stamping events.

End of Chapter

- How do meaningful POU and Rung names help?

  They can self-document the application meaning that the purpose of a rung can be seen without having to refer to other documentation.

- Where can comments be added to the program?

  Comments can be added to each rung of the program.

- What information can be stored in the Front Page?

  Information about the engineer and company that produced the application.