

# Getting Started With SoMachine

## Self Study Manual

SoMachine Ver 4.1.1

## DISCLAIMER

Schneider Electric Inc. 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 Inc. 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 Inc. 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 Inc. 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.

© 2014 Schneider Electric Inc.. All rights reserved.

The contents of this manual are proprietary to Schneider Electric Inc. and all rights, including copyright, are reserved by Schneider Electric Inc.. No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric Inc..

## SoMachine Training Manual

### INTRODUCTION AND LEGAL NOTICE

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

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

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

### TRADEMARKS

Schneider Electric Inc. 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.

SoMachine, Magelis, Vijeo Designer and Lexium are trademarks owned by Schneider Electric or its affiliated companies. All other trademarks are the property of their respective owners.

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

Third-party's trademark(s) used in the manual] is a (are) trademark(s) / registered trademark(s) of / owned by Third-party legal name (and applicable 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

Bruce Howlett

# 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.**

## Before the Course Begins (cont.)

### 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<sup>1</sup>.
- 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.**

<sup>1</sup> 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 tutorial you will be able to:

- Create a new SoMachine project
  - Create a simple Programmable Organization Unit
  - Add a POU to the MAST task
  - Download a project to the software simulator(Simulation)
  - Run a project in the software simulator
  - Force values into registers to observe the POU's operation
- 

## Target Audience

The SoMachine V4.0 Tutorial is designed for:

- Technical support Level 1-2-3
- Product application Engineers
- Application Engineers
- Automation Experts
- Solution Experts
- Product Managers
- OEM's, Integrators
- Customers
- SE Sales

# Course Overview (cont.)

**Course Objectives**

The training course will take two days to complete. The following program outlines the topics that will be covered.

Day	Topics
1	<ul style="list-style-type: none"> <li>➤ SoMachine at a Glance</li> <li>➤ System Requirements</li> <li>➤ SoMachine Interface</li> <li>➤ Opening Projects</li> <li>➤ System Options</li> <li>➤ New Project Creation</li> <li>➤ New Project Assistants</li> <li>➤ Templates</li> <li>➤ Archiving</li> </ul>
2	<ul style="list-style-type: none"> <li>➤ Creating a project</li> <li>➤ Adding Devices</li> <li>➤ The Logic Builder</li> <li>➤ The Device Tree</li> <li>➤ The Tools Tree</li> <li>➤ The Application Tree</li> <li>➤ Catalogs</li> <li>➤ Gateways</li> <li>➤ Tasks</li> <li>➤ Controller Program Execution</li> <li>➤ POU Creation</li> <li>➤ Firmware updating</li> <li>➤ CoDeSys Programming Languages</li> <li>➤ Global Variables</li> <li>➤ Program Simulator</li> <li>➤ Forcing Variables</li> </ul>

# 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:

➤ Topic A .....	1-2
➤ Topic B .....	1-3
➤ Topic C .....	1-5

---

## 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 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 *SoMachine 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

---

<b>CHAPTER 1: INTRODUCTION TO SoMACHINE &amp; THE WORKSTATION.....</b>	<b>1-1</b>
Overview - SoMachine at a Glance.....	1-1
System Requirements.....	1-2
Features of SoMachine.....	1-3
The SoMachine Interface .....	1-6
SoMachine Central Screen.....	1-7
Recent Projects Option.....	1-8
New Project Option.....	1-9
Open a Project Option.....	1-10
Exercise - Browse for an Existing Project .....	1-11
Properties Screen.....	1-13
System Options .....	1-14
<b>CHAPTER 2: PROJECT MANAGEMENT.....</b>	<b>2-1</b>
Overview .....	2-1
New Project Creation .....	2-2
New Project Option.....	2-4
New Project Assistant .....	2-5
Exercise - Starting a New Project Using the Assistant .....	2-7
New Project Using a Template.....	2-8
Exercise - Starting a Project from a Template .....	2-9
New Empty Project .....	2-10
Archive Projects .....	2-13
Exercise - Restore an Archived Project.....	2-16
<b>CHAPTER 3: SoMACHINE CENTRAL FUNCTIONS.....</b>	<b>3-1</b>
Overview - Creating a Project.....	3-1
The Workflow Screen .....	3-2
Device Selection.....	3-3
Exercise - Add a Device to an Empty Project.....	3-5
<b>CHAPTER 4: NEW PROJECT CREATION.....</b>	<b>4-1</b>
Overview .....	4-1
The Logic Builder .....	4-3
The Device Tree .....	4-4
The Tools Tree .....	4-5
The Application Tree .....	4-7
Work Area.....	4-9
Catalogs.....	4-10
Embedded Functions .....	4-11
Communication .....	4-15
Tasks.....	4-16
Controller Program Execution .....	4-17

POU Program Creation .....	4-18
Exercise - Create a POU.....	4-21
Initial USB Communications Configuration.....	4-24
Connecting to the Controller .....	4-26
Updating a Controller's Firmware .....	4-30
Task Configuration.....	4-35
Exercise - Configure a Task .....	4-37
PLC Simulator.....	4-39
Exercise - Using Simulation Mode .....	4-41
CoDeSys Program Languages.....	4-45
Exercise - Program a FBD POU .....	4-46
Exercise - Convert IL to LD.....	4-52
Exercise - Program a CFC POU.....	4-55
Watchdog Mechanisms .....	4-56
Structuring an Application .....	4-57
The POU Function .....	4-58
Sample Project.....	4-62
Global Variables.....	4-65

# Chapter 1: Introduction to SoMachine & the Workstation

## Overview - SoMachine at a Glance

### Introduction

---

SoMachine is a professional, efficient and open OEM software solution that develops, configures and commissions the entire machine in a single environment including logic, motor control, HMI and related network automation functions.

---

### Chapter Objectives

By the end of this chapter, the student will be able to:

- Describe the computer requirements to install SoMachine software
  - Describe SoMachine's basic features
  - Describe SoMachine's GUI and Home Screen
  - Configure basic system parameters
- 

### This Chapter Covers These Topics:

- System Requirements ..... 1-2
- Features of SoMachine ..... 1-3
- The SoMachine Interface ..... 1-6
- SoMachine Central Screen ..... 1-7
- Recent Projects Option ..... 1-8
- New Project Option ..... 1-9
- Open a Project Option ..... 1-10
- Exercise - Browse for an Existing Project ..... 1-11
- Properties Screen ..... 1-13
- System Options ..... 1-14
- Exercise - Examine the User Interface ..... 1-17

## System Requirements

The computer equipment may need to be upgraded to run Version 4. as the minimum hardware requirements have changed:

Description	Minimum Specification	Recommended
Processor	Pentium IV, 1.8 GHz, Pentium M, 1.0 GHz or equivalent	Intel Core (tm) I7-2.7 GHz
Random Access Memory (RAM)	2 GB	3 GB
Free Hard Drive Space	<b>5 GB</b> including the memory space for the software installation, temporary space for execution and space for saving applications	10 GB
Drive	DVD Reader	
Display	Resolution: 1280 x 1024 pixel	Resolution: 1680 x 1050 pixel
Peripherals	Mouse or compatible pointing device USB interface	
Web Access	Web registration requires Internet access.	

### Software Requirements

SoMachine Component	Software Requirements
Operating system	<p>The SoMachine software supports the following operating systems:</p> <ul style="list-style-type: none"> <li>➤ Microsoft Windows XP Professional edition Service Pack 3</li> <li>➤ Microsoft Windows 7 Professional Edition - 32 bits</li> <li>➤ Microsoft Windows 7 Professional Edition - 64 bits</li> </ul>
Software requirements	<ul style="list-style-type: none"> <li>➤ SoMachine literature contains PDF-formatted documents that require the installation of the Adobe Reader. This reader is not part of the SoMachine installation but can be downloaded from <a href="http://www.adobe.com/go/getreader">http://www.adobe.com/go/getreader</a>.</li> <li>➤ Microsoft .NET Framework 4.0</li> </ul>

# Features of SoMachine

## Standard Languages

---

SoMachine includes, as standard, 6 IEC (International Electrotechnical Commission) languages which are compliant with IEC 61131-3. Depending on requirements, the application may use any mixture of these different languages.

- Function Block Diagram (FBD)
- Sequential Functional Chart (SFC)
- Structured Text (ST)
- Instruction List (IL)
- Ladder (LD)
- Continuous Function Chart (CFC)
- Continuous Function Chart (CFC) - page oriented

---

Note - In addition to the CFC standard editor CODESYS provides the so-called CFC editor pagination. Besides the tools and commands of the standard CFC editor this editor allows to arrange the elements on any number of different pages.

---

## Controller Programming Services

- 
- User Created Function Blocks (FBs)
  - User Created Functions
  - Data Unit Type (DUTs)
  - On-line changes
  - Watch windows
  - Graphical monitoring of variables (trace)
  - Breakpoints, step-by-step execution
  - Simulation
  - Visualization for application and machine set-up

## HMI Based Services

- 
- Graphics libraries containing more than 4000 2D and 3D objects.
  - Simple drawing objects (points, line, rectangles, ellipses, etc ...)
  - Preconfigured objects (button, switch, bar graph, etc ...)
  - Recipes (32 groups of 256 recipes with max. 1024 ingredients)
  - Action tables
  - Alarms
  - Printing
  - Java scripts
  - Multimedia file support: .wav, .png, .jpg, .emf and .bmp
  - Variable trending

## Features of SoMachine (cont.)

---

### Motion Services

- Embedded devices configuration and commissioning
- CAM profile editor
- Sample application trace
- Motion and drive function blocks libraries for inverters, servos and steppers
- Visualization screens

## Features of SoMachine (cont.)

---

### Global Services

- User access and profile
  - Project documentation printing
  - Project comparison (control)
  - Variable sharing based on publish/subscribe mechanism
  - Library version management
- 

### Integrated Fieldbus Configurators

- Master:
    - CANopen
    - CANmotion
    - Modbus Serial Line
    - AS-interface
  - Connectivity:
    - Profibus-DP
    - Ethernet IP
    - Modbus TCP
- 

### Application Libraries

- General:
  - PLCopen function blocks for Motion control
- Segment Solutions:
  - Packaging function blocks
  - Conveying function blocks
  - Hoisting function blocks
  - Booster Pumping Application Pumping function blocks

# The SoMachine Interface

## How to Start SoMachine

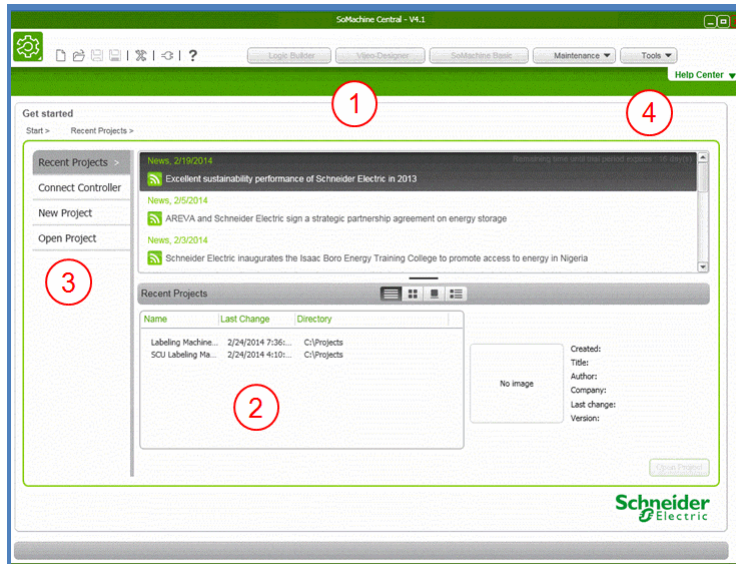
### ➤ To start SoMachine:

Select the SoMachine V4.0 item from the Windows start menu:

**Start » Programs » Schneider Electric » SoMachine Software » V 4.0 » SoMachine V4.0**

or

Double-click the **SoMachine V4.0** icon on the desktop.



The **SoMachine Central Screen** appears. Some of the key areas are:

Number	Description
1	SoMachine Central, home page
2	List of any previous projects
3	Available Operations... What do you want to do?
4	Access to help center

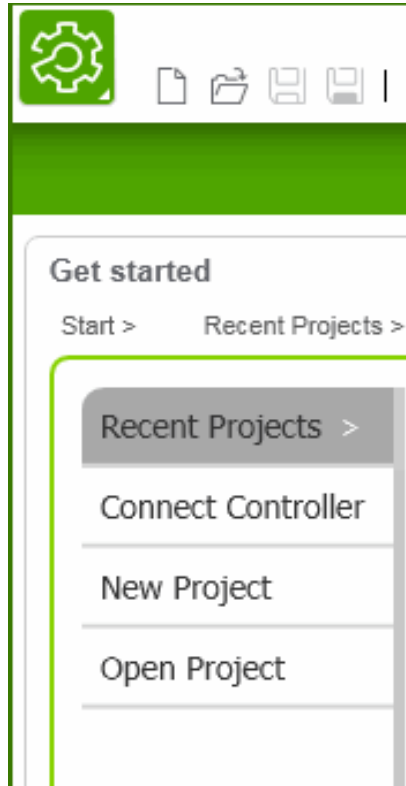


# SoMachine Central Screen

## Visual Graphical User Interface

SoMachine has evolved from a monolith to a suite-like software package that is both intuitive and highly visual. All project tools can be used in the same manner, providing a clear and easy to use user interface. With no mixture of old and new styles, the user is insured of a consistent, and easy to learn, approach to application development.

## SoMachine Central Tasks

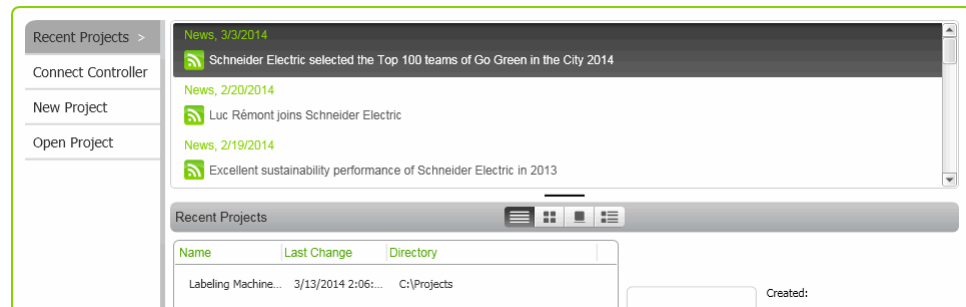


Tasks available from the SoMachine Central screen are:

Name	Description
Recent Projects	Displays recently opened projects
Connect Controller	Connect to a controller
New Project	Start a new project
Open Project	Open an existing project

# Recent Projects Option

## Open Recent Projects

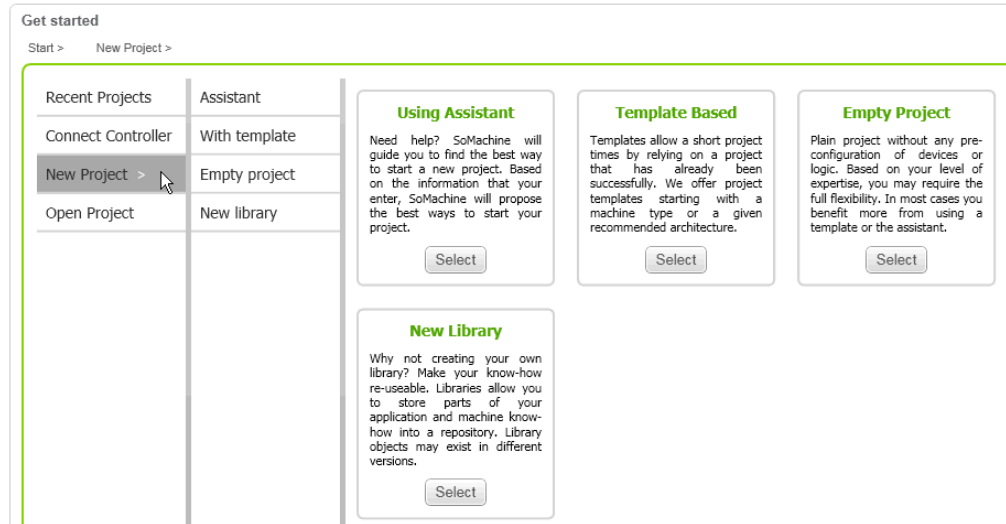


When the **Recent Projects** option is selected, SoMachine displays a list of projects from the defined project program open/save location. Refer to **System Options** page for information on how to modify this location

# New Project Option

## Start a New Project

New projects are started by first selecting New Project, then selecting the method that you wish to use to start a project



The available for starting a new project are:

Method	Description
Assistant	A wizard for helping to select hardware
With Template	TVDA templates for common applications
Empty Project	User defined project, nothing is provided
New Library	Start a new user defined, custom library

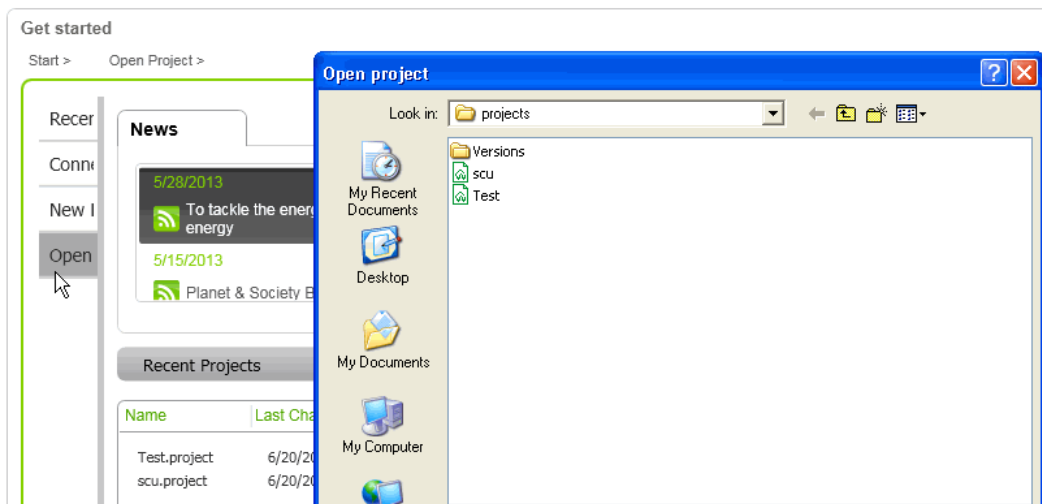
# Open a Project Option

## Open an Existing Project

The Open an Existing Project option allows you to browse the computer for existing projects, starting with the default project path.

## How to Open an Existing Project

➤ To Open an existing project:



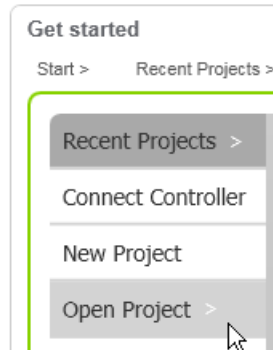
Select the Open Existing Project option from SoMachine Central window. The computer will start browsing with the path specified as the default program path. If the project exists elsewhere, you can easily browse any location accessible by the computer.

## Exercise - Browse for an Existing Project

---

### 1 Open an Example Project.

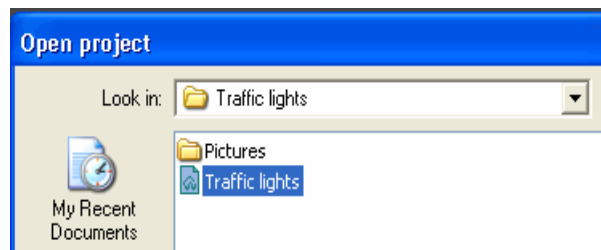
- i. Launch SoMachine.
- ii. When the **SoMachine Central** screen appears click the **Open Project** item.



- iii. Following the path below, browse for the file named Traffic Lights.

**C:\Documents and Settings\All Users\Documents\SoMachine Software\V4.1\Learning Center\Examples\Generic\Traffic Lights**

- iv. Open the Traffic Lights project stored there. If SoMachine prompts you to update the example project, select the update option and wait for the project to finish updating



## Exercise - Browse for an Existing Project (cont.)

- v. When the project has finished updating and opened SoMachine will display the workflow model for the Traffic Lights project



- vi. Select the **Properties Tab** to display the projects properties. How many POUs does this project have? \_\_\_\_\_.

The screenshot shows the 'Properties' tab for the 'Traffic\_lights.project'. It displays various project details:

- Filename: Trafficlights
- File path: C:\Documents and Settings\CM-MPA\Local Settings\Temp\Central\TemporaryProject6daccase-7...
- Last changed: 2/28/2014 10:48:30 AM
- Title: Traffic lights
- Author: Machine Solutions
- Company: Schneider Electric
- Version: 4.1.4.0

There is a 'Disclaimer Of Warranty' section with a warning message. Below this is a 'Custom Information' table:

Name	Information	Type	Re...
CustomField01	true	Text	
CustomField01	CustomField01	Text	
CustomField01	Example	Text	
CustomField02	CustomField02	Text	

On the right, there is a 'Statistics' section showing a list of project components:

- 5 Action
- 1 Application
- 1 Device
- 1 GlobalImagePool
- 1 GlobalTextList
- 2 Library Manager
- 1 PLC Logic
- 2 POU
- 2 Program call
- 1 Project Information
- 1 Target/Visualization

The Schneider Electric logo is visible in the bottom right corner.

- 2 Leave the Project open and return to the Home screen.

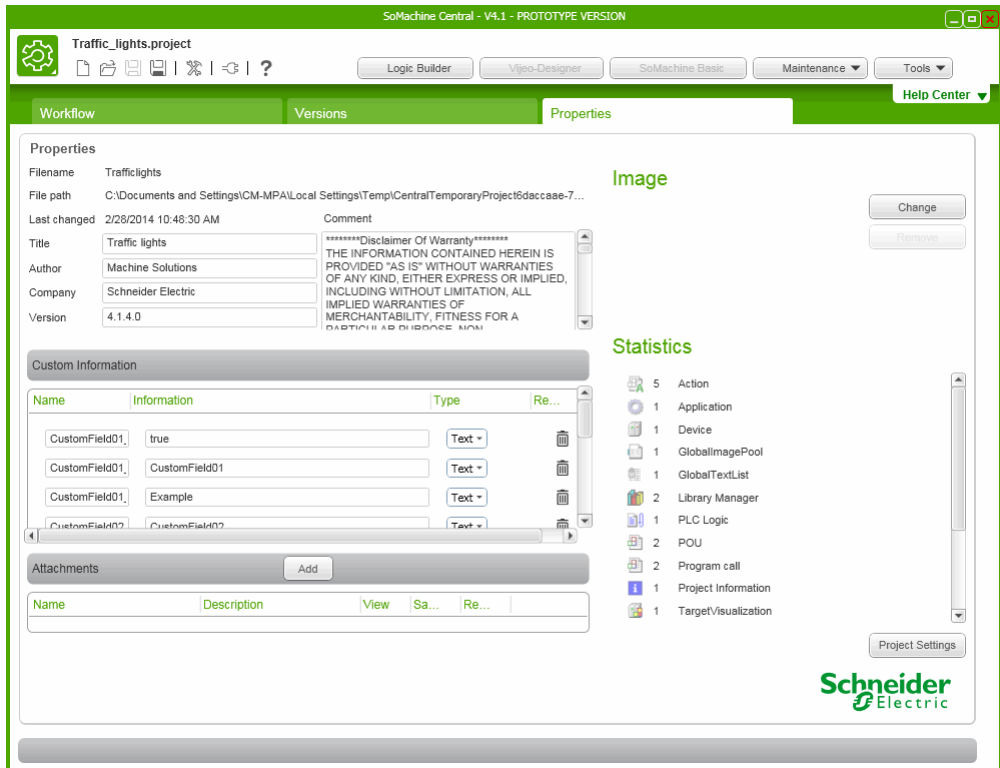


# Properties Screen

## Additional Project Information

The Properties screen allows users to enter additional project information. The textual and graphical information entered here is optional. As this information is always displayed in the information pane for the project selected in the work area, it helps to identify the individual projects, avoiding the need to open them.

This screen is only displayed after a project has been opened.



Other information that you can add to a project includes:

Properties	Description
Title	Optional project Title
Author	Optional author(s) names
Company	optional company name
Version	Versioning is an important piece of information to add
Comments	any comment(s) you wish to add

# System Options

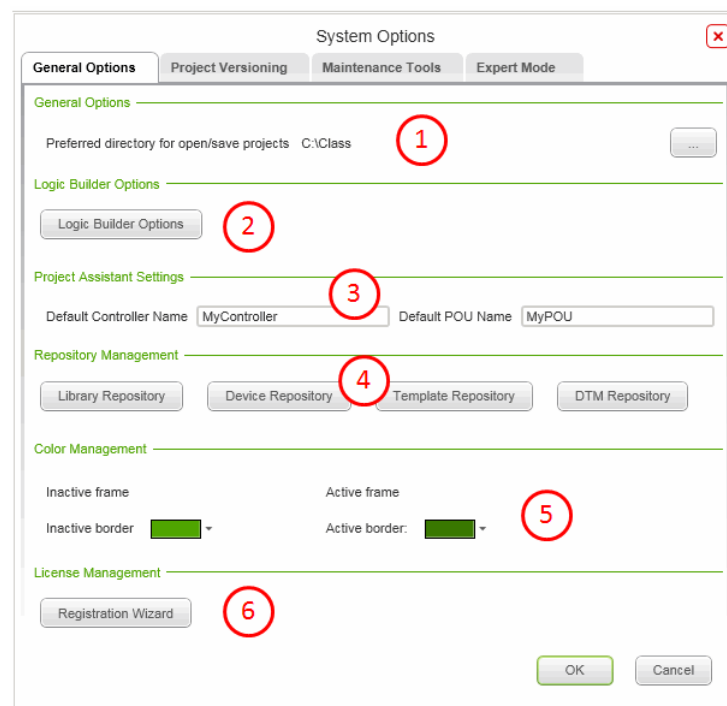
## General Options

After an initial installation of the SoMachine software, it's a good idea to validate or modify the system options such that the system operates the way desired.

System options are set by opening the Modify System Options screen



The programmable system options are as shown

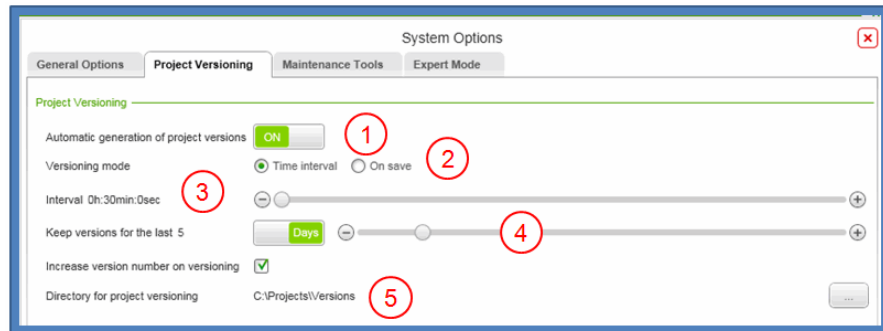


Number	Description
1	Select the preferred location to save projects
2	<b>Logic Builder options.</b> Define how the editor looks and works. Can also be set from the <b>Tools Menu</b> inside the Logic Builder
3	Specify the default name for the project's controller
4	<b>Repository Management</b> - when a project is open, use these buttons to add non-standard libraries, Devices (i.e., Controllers, HMI's drives etc.), Templates and DTM's to the project
5	<b>Color Management</b> -define interface colors
6	<b>License Management</b> - Launch the registration wizard tool



## Modify System Options (cont.)

### Project Versioning



Number	Description
1	Activate/Deactivate project versioning (ON is recommended )
2	Set versions based on a time interval or when you save
3	If time based, indicate the amount of time
4	Set the number of days to keep a version
5	Set the path for versions

## Modify System Options (cont.)

### Maintenance Tools

System Options [X]

General Options | Project Versioning | **Maintenance Tools** | Expert Mode

External Maintenance Tools

Name	Path	Remove

Add external maintenance tool

Number	Description
1	Add any external maintenance tool the project requires

### Expert Mode

System Options [X]

General Options | Project Versioning | Maintenance Tools | **Expert Mode**

Expert Mode

Enable Expert Mode

Select tool to start on opening project

- Logic Builder
- Vijeo-Designer

Number	Description
1	Set the Editor to be launched when the project opens

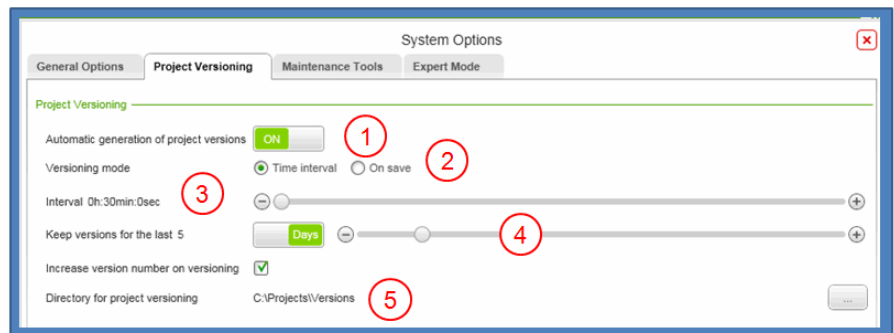
# Exercise - Examine the User Interface

## 1 Create a new project based on one of the existing examples.

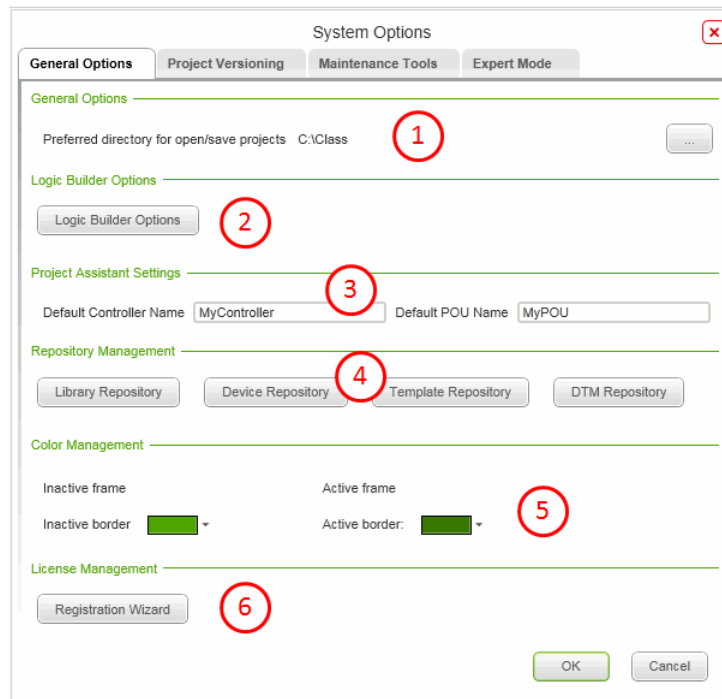
- i. Use the **Windows Explorer** and check to see if a folder named **C:\Class** exists. If it doesn't, **create it (C:\Class)**. Also check for a folder named **C:\Projects**. If this does not exist, create it now
- ii. Click the **Modify the system options** icon from the SoMachine Central window. Change Option 5 in the first image and option 1 in the second to match match images below



- iii. Change items **5** in your application to match the image below.

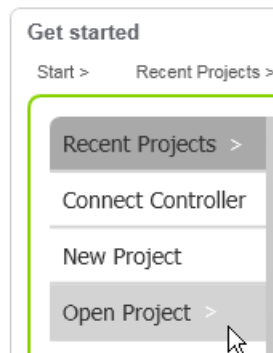


Change item **1** in your application to match the image below. Close the **System Options** window when finished and save.



## Exercise - Examine the User Interface (cont.)

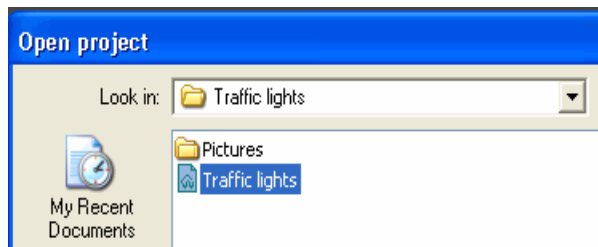
- iv. Return to the SoMachine Central screen click the **Open Project** item.



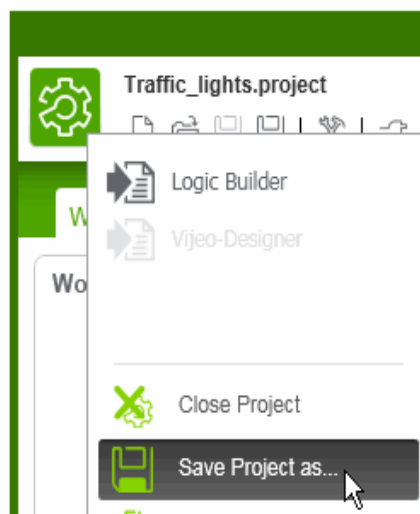
- v. Following the path below, browse for the file named Traffic Lights.

**C:\Documents and Settings\All Users\Documents\SoMachine Software\V4.1\Learning Center\Examples\Generic\Traffic Lights**

- vi. Open the Traffic Lights project stored there. If SoMachine prompts you to update the example application, select the update option and wait for the project to finish updating



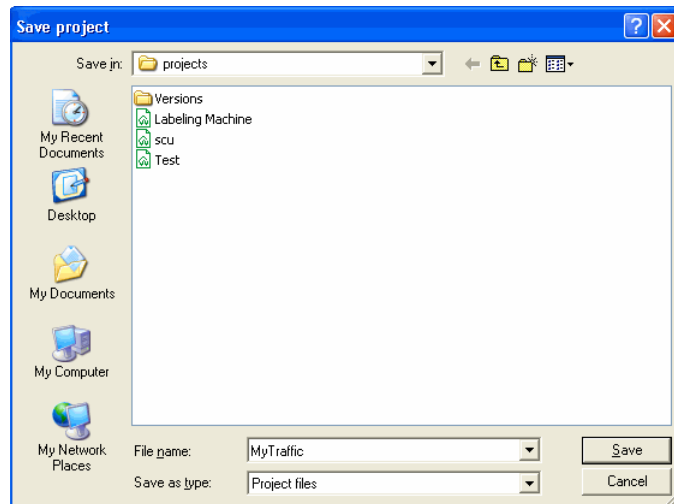
- vii. Click on the "Open Main Menu Icon from the SoMachine Central screen and select the Save Project as" option



## Exercise - Examine the User Interface (cont.)

---

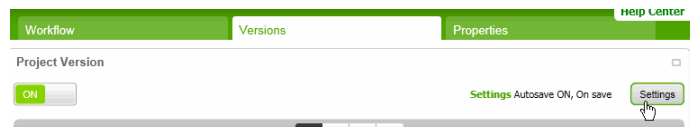
- viii. Save the application as "**MyTraffic**" as shown. Since you changed the default project path in step 3, this version of the project should be automatically directed there. t



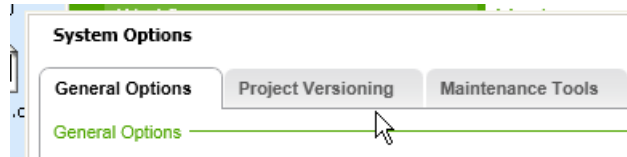
## Exercise - Examine the User Interface (cont.)

### 2 Examine the interface using an example project.

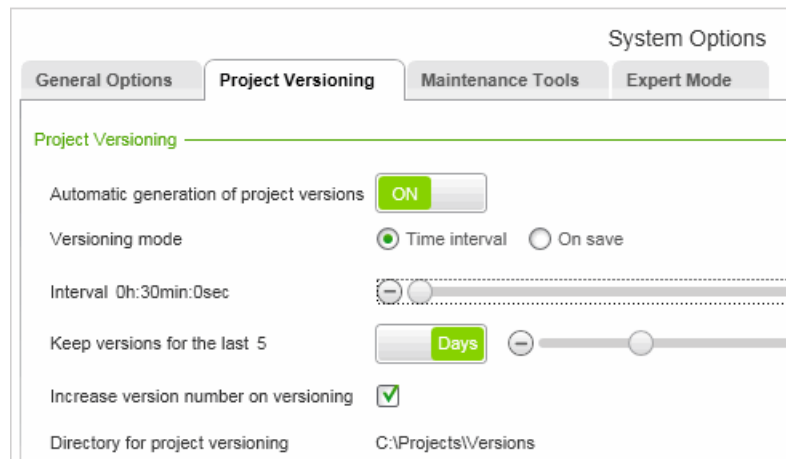
- i. Navigate to the Versions tab then click the Settings button.



- ii. When System Options appears, select Project Versioning

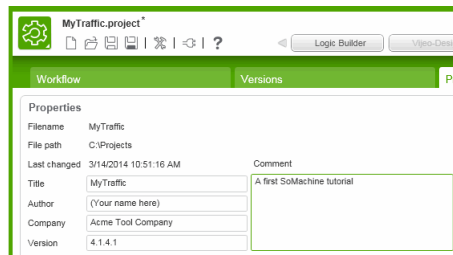


- iii. Set **project versioning** as shown. In this case, the project will automatically create a version about every 30 minutes and will keep 5 days worth of versions. Close this panel when finished.



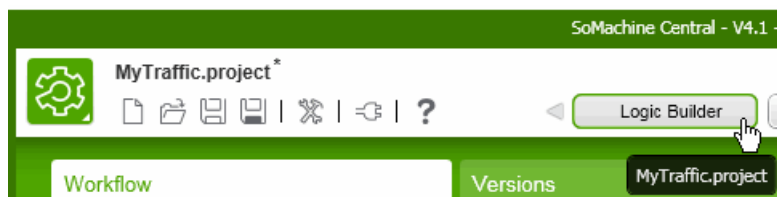
## Exercise - Examine the User Interface (cont.)

- iv. **Select the Properties** tab. Add some information to the project under the Title, Author and Company Name area (left side). Close this panel when finished

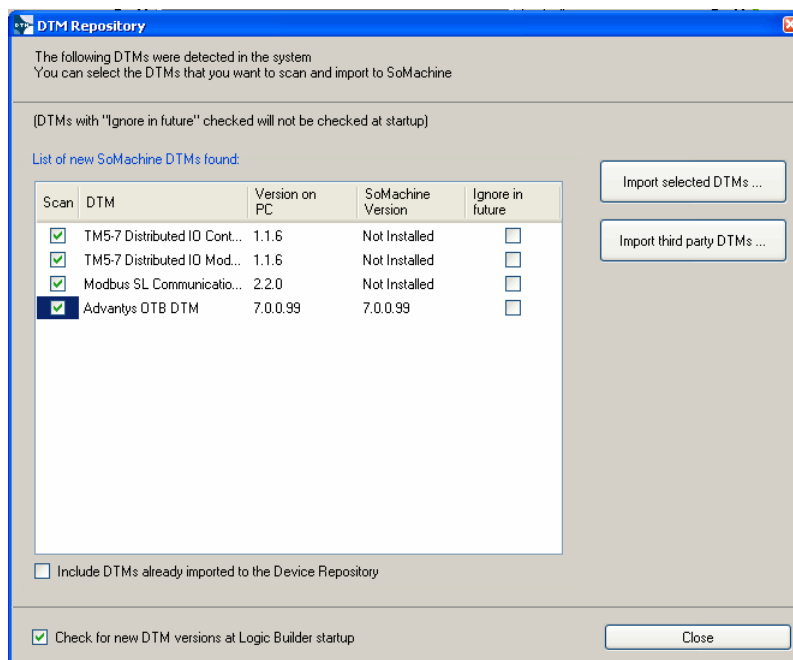


### 3 Open the Logic Builder

- i. Click on the Logic Builder button on the SoMachine Central window to open the Logic Builder

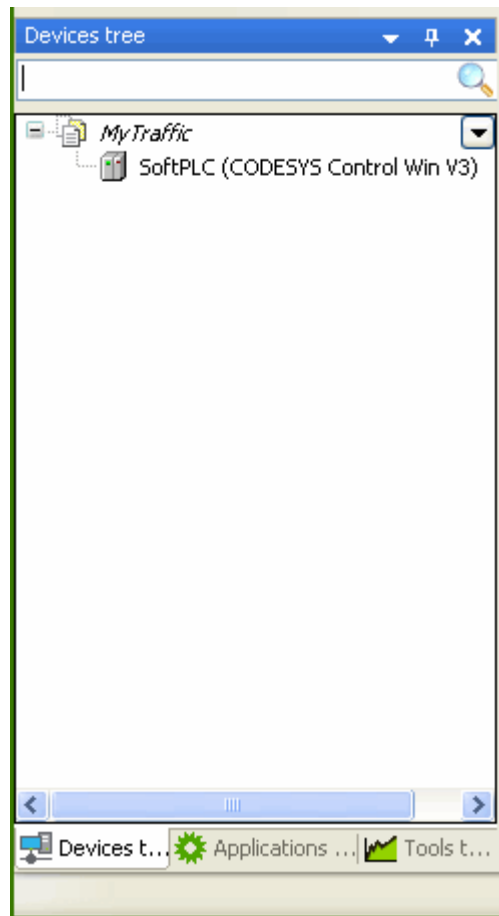


- ii. The software opens the Logic Builder. Ignore any requests to add DTMs to this project. Simply close that window if it appears. This feature can be disabled permanently by removing the check located in the lower left of the image below.

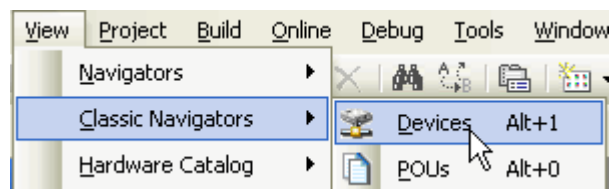


## Exercise - Examine the User Interface (cont)

- iii. When the Logic Builder opens, notice the Devices Tree on the left side of the window. In version 4.0, the default appearance has the Devices Tree split into 3 tabs (bottom), simplifying each tab. If no Bring each tab to the foreground in turn and observe what each displays



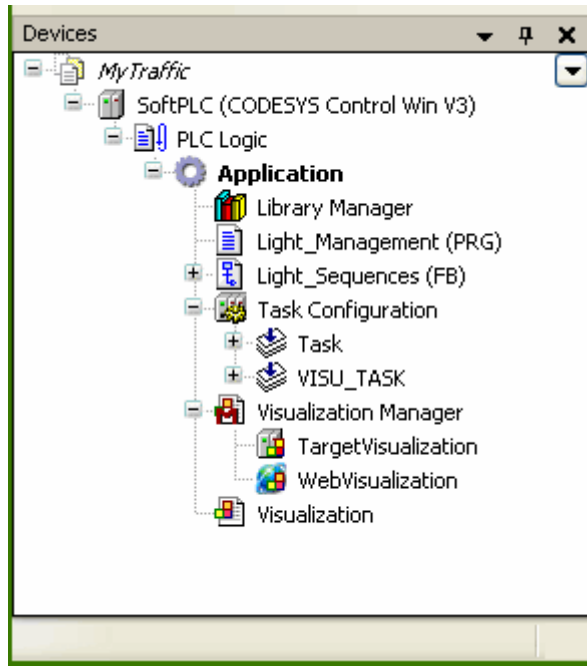
- iv. Open the View menu and select Classic Navigators followed by Devices





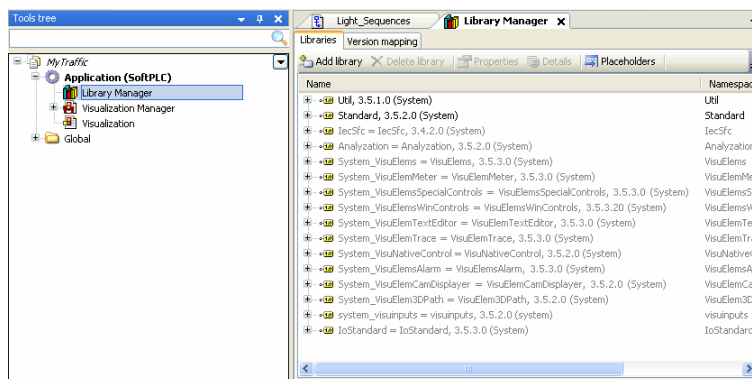
## Exercise - Examine the User Interface (cont)

- v. Notice that the Devices Tree is now a single tree with all devices displayed (no bottom tabs). This was the way the SoMachine Devices tree appeared prior to V 4.0 and it is still supported. The choice is yours as to how you like the Devices Tree to be displayed. **Close** the Classic Devices Tree.



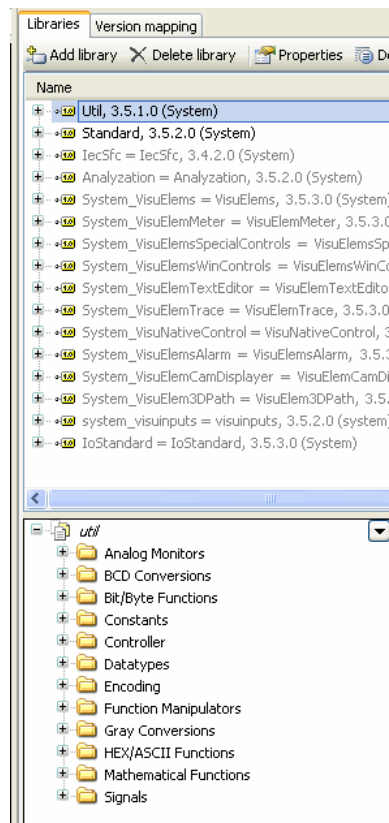
### 4 Navigate the Devices Pane

- i. From the View » Navigators menu, open the Tools Tree and open the Library Manager by double clicking on it. You will see the libraries associated with this project in the work area. SoMachine dynamically adds libraries to a project as you add hardware to the project. In addition you can add optional libraries to the project by clicking on the Add Library icon.



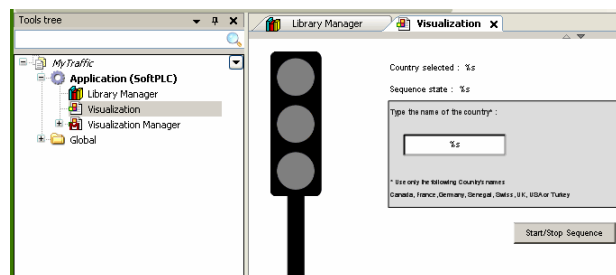
## Exercise - Examine the User Interface (cont)

- ii. Select the Util Library and observe what is in it in the lower frame. You can always see the contents of a library in this manner



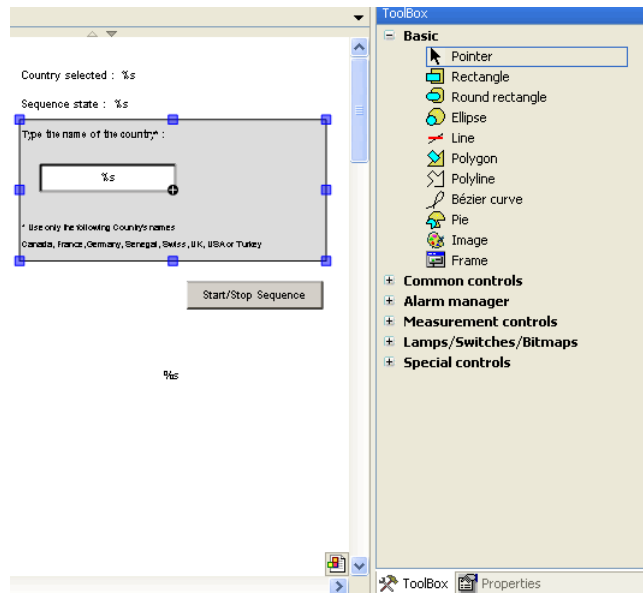
## 5 Examine the objects in the Example project.

- i. From the Tools Tree tab, Double click the **Visualization** object.



## Exercise - Examine the User Interface (cont)

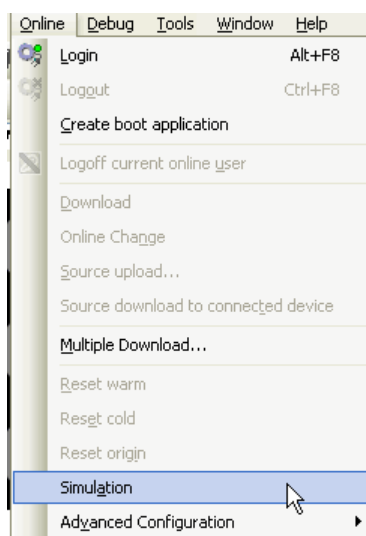
- ii. The toolbars are context sensitive. When the **Visualization** is the focus window, a graphic editor toolbar and properties tabs appear in the toolbox frame to the right of the visualization. Visualizations may be either user created or sometimes are part of a library that has been added to a project. If the visualization is part of a library, many are designed to work with specific function blocks that are also part of that library.



- iii. Select different items from the visualization and observe how the edit changes. Take a look at the tools in the toolbox that are available to build custom, user created visualizations.

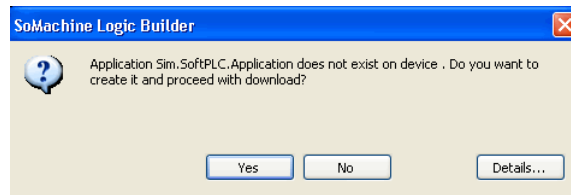
## 6 Run the project.

- i. **Select Online » Simulation** from the **Online** menu. The project is placed in **Simulation Mode**.

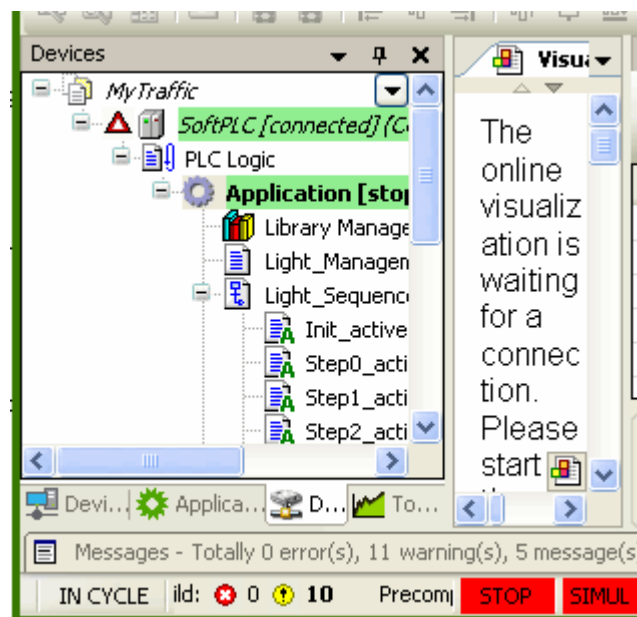


## Exercise - Examine the User Interface (cont)

- ii. Select **Online » Login** to login to the Simulator.
- iii. Since there is no program in the simulator, the following message appears. **Click Yes.**



- iv. Open the Classic Device Tree again. The operator interface shows that the program is connected by displaying a **green background**. **Notice that in simulation mode, there is a red indicator at the bottom, right of the screen as well**



- v. **Select Online » Start** to start the controller or simply click the right arrow from the toolbar as shown below.

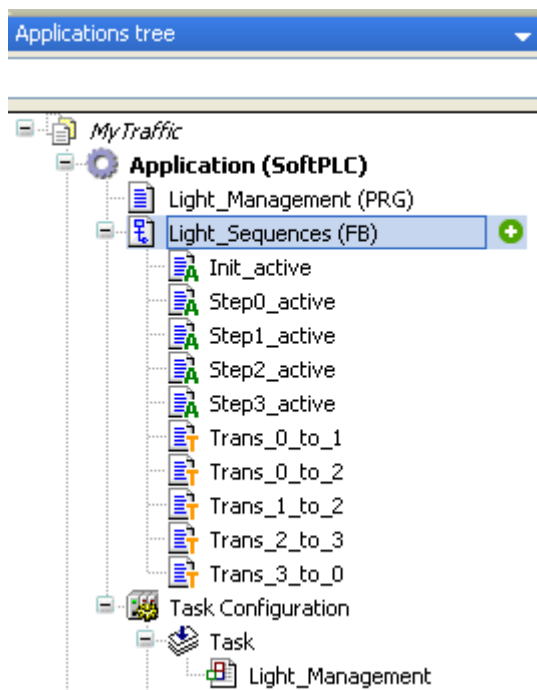


## Exercise - Examine the User Interface (cont)

- vi. **Open** the visualization and **Type** the name of one of the *supported countries* into the input field. **Press ENTER**.



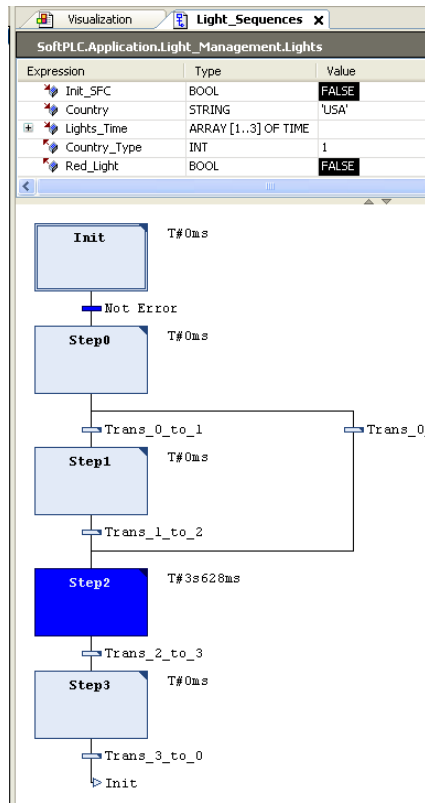
- vii. **Click** the **Start/Stop Sequence** button.
- viii. **Double click** the **Light Sequences (FB)** POU in the Devices tree to *open the POU*. **Observe** the values changing.



## Exercise - Examine the User Interface (cont)

---

- ix. Watch the SFC structure run as the program goes through its cycle.



The various steps in the SFC structure will change color when active. The top part of the screen shows variables and their status. Notice the country that you entered in the Country variable. Examine the user interface further as time permits

---

### 7 Shutdown the runtime.

- Select **Online » Stop**.
- Select **Online » Logout**.
- Select **Online » Simulation** to disable **Simulation Mode**.

---

### 8 Return to the Home screen (SoMachine Central).

---



# Chapter 2: Project Management

## Overview

---

### Introduction

SoMachine allows the user to perform fundamental tasks such as creating, exporting and importing projects. There are also other project management tools such as archiving which create a highly compressed version of the project.

The implemented project management principle allows users to browse the existing projects quickly obtaining the relevant information without the need to open them before selection.

The user can create a new project, starting from several means: using Tested Validated and Documented Architectures, using the provided examples, using an existing project or from scratch. There is quick access to the most recently used projects.

### Chapter Objectives

---

By the end of this chapter, you will be able to:

- Describe various ways of starting a new project
  - Browse for an existing project
  - Create a project archive
  - Restore a project archive
- 

### This Chapter Covers These Topics:

- New Project Creation.....2-2
- New Project Option .....2-4
- New Project Assistant.....2-5
- Exercise - Starting a New Project Using the Assistant.....2-7
- New Project Using a Template .....2-8
- Exercise - Starting a Project from a Template .....2-9
- New Empty Project.....2-10
- Exercise - Create a New Empty Project.....2-11
- Archive Projects.....2-13
- Exercise - Archive Projects.....2-14
- Exercise - Restore an Archived Project .....2-16

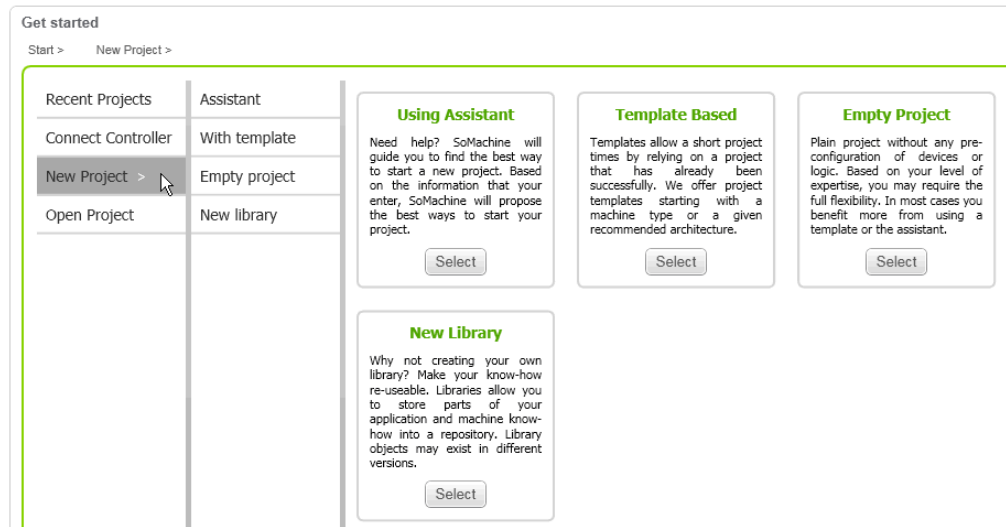
# New Project Creation

## First Step in the Configuration

The first step when configuring SoMachine is to create a new project. The project is where all information is stored.

SoMachine facilitates project creation by providing different ways of starting a new project.

The available methods for starting projects is shown below:



New projects are started by first selecting New Project, then selecting the method that you wish to use to start a project. The options are:

Method	Description
Assistant	A wizard for helping to select hardware
With Template	TVDA templates for common applications
Empty Project	User defined project, nothing is provided
New Library	Start a new user defined, custom library



## New Project Creation (cont.)

---

### Configure Projects

SoMachine provides tools that assist users in creating new projects quickly and easily.

It provides for project startup:

- A variety of preferred implementations. A dedicated **TVDA Finder** tool (Tested Validated Documented Architectures) assists users in selecting the preferred implementation that best suits individual projects.
- A variety of application projects for conveying, hoisting, and packaging that provide basic configurations for these applications.
- Some examples that provide basic projects for making yourself familiar with SoMachine.

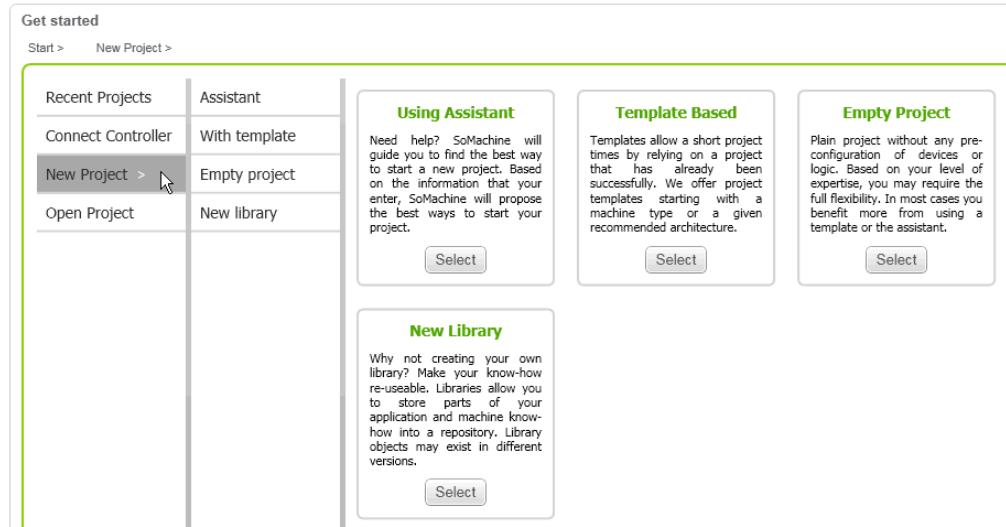
Once the project is created, SoMachine provides extensive possibilities to add textual and graphical information to each project file. This additional information enables users to distinguish projects avoiding the need to open them when they have to select the suitable project out of those that are available on the computer.

For easy configuration of the project, SoMachine provides a graphical configuration editor that allows users to add and configure the requested devices easily.

# New Project Option

## Start a New Project

New projects are started by first selecting New Project, then selecting the method that you wish to use to start a project



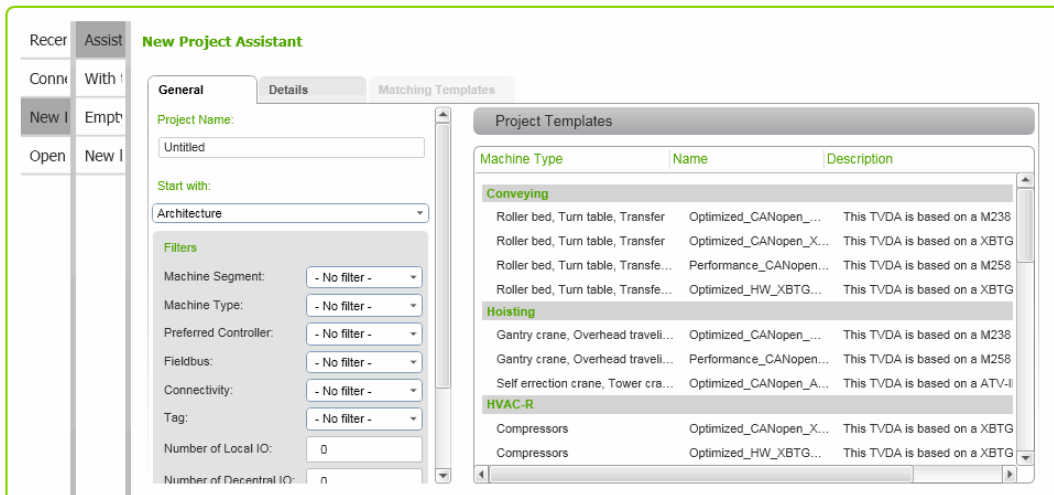
The available for starting a new project are:

Method	Description
Assistant	A wizard for helping to select hardware
With Template	TVDA templates for common applications
Empty Project	User defined project, nothing is provided
New Library	Start a new user defined, custom library

# New Project Assistant

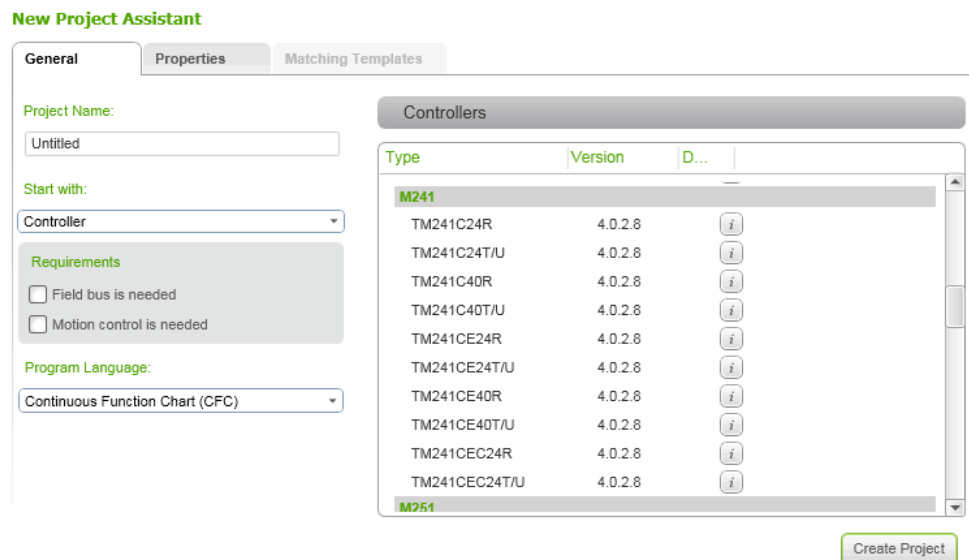
## New Project Assistant

The New Project Assistant helps with the selection of a hardware platform and project template by applying user configured filters. It is basically a type of Wizard. What options you select at the Architecture level, impacts the controllers displayed at the Controller level.



## Controller Filter

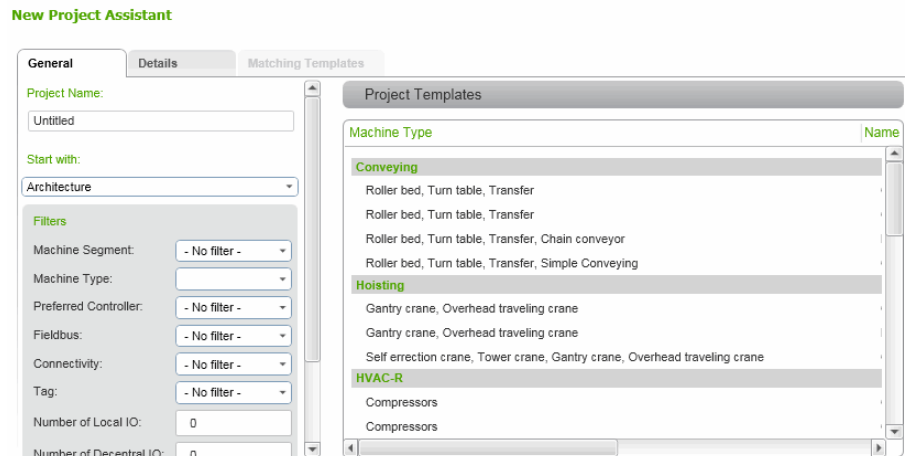
One of the primary filters is the Architecture/Machine type/Controller filter. Various controllers are shown when that option is selected. Under the Matching Templates tab, a list of available templates are displayed.



## New Project Assistant (cont.)

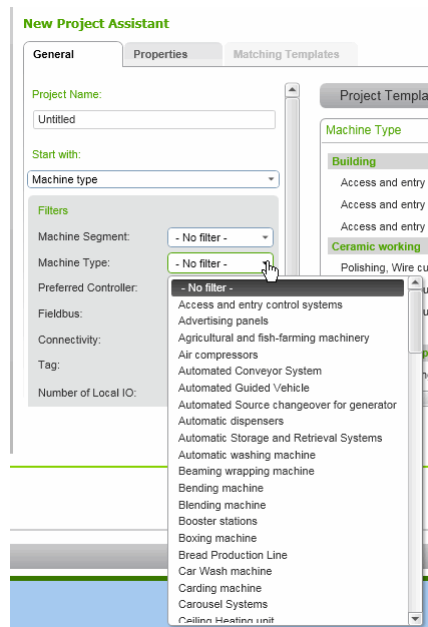
### Architecture Filter

The Architecture filter provides several templates that may be used to create a new project. If no particular **Machine Segment** is selected, all available templates for all available platforms are displayed.



### Machine Type Filter

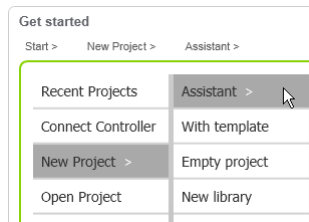
The Machine Type Filter offers various TVDA's based on the filters you select. TVDA's with platform and Drive options are available



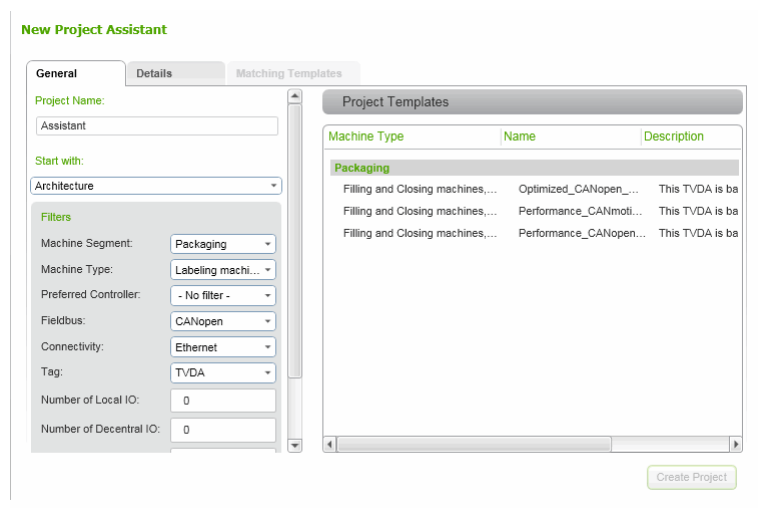
# Exercise - Starting a New Project Using the Assistant

## 1 Start a new Project

- i. Start a new using the Assistant



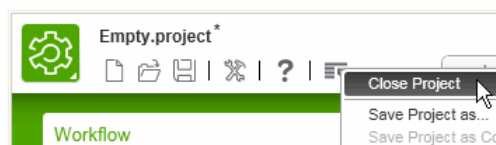
- ii. Name your project Assistant and select the filters shown below. A list of possible controllers that are left after filtering is displayed on the right. Select one of the available controllers and click The Create Project button.



- iii. Experiment with other options as time allows
- iv. Save and close your Assistant project

## 2 Return to the Home screen.

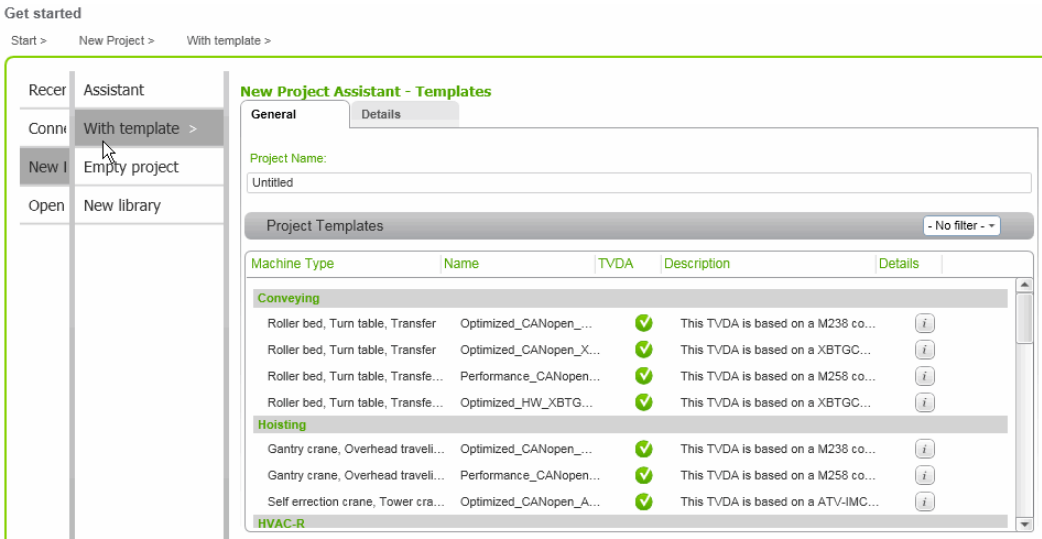
- i. Return to the the Home Screen (SoMachine Central) and close and save the project



# New Project Using a Template

## Using Templates

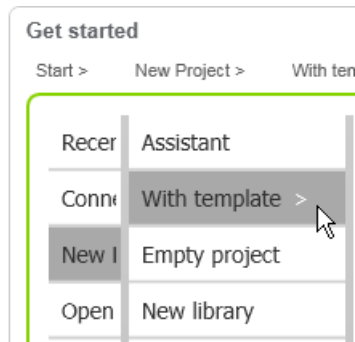
Starting a new project via a template can greatly shorten a projects development time. Templates are all tested and validated, partial solutions. Schneider Electric Inc. provides templates for various architectures or applications. A filter to help you select the correct one.



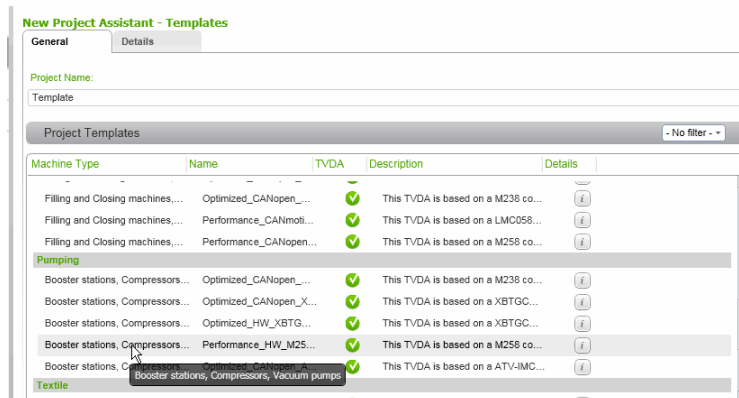
# Exercise - Starting a Project from a Template

## 1 Start a New Project using a Template

- i. Select **New Project** then **WithTemplate** from the SoMachine Central screen



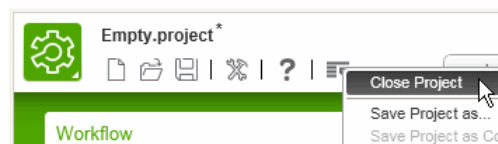
- ii. Browse the available Templates. Experiment with applying Filters (button at the right). See how the available hardware changes as Different filters



- iii. Select one of the templates and click the **Create Project** button.

## 2 Return to the Home screen.

- i. Return to the the Home Screen (SoMachine Central) and close and save the project



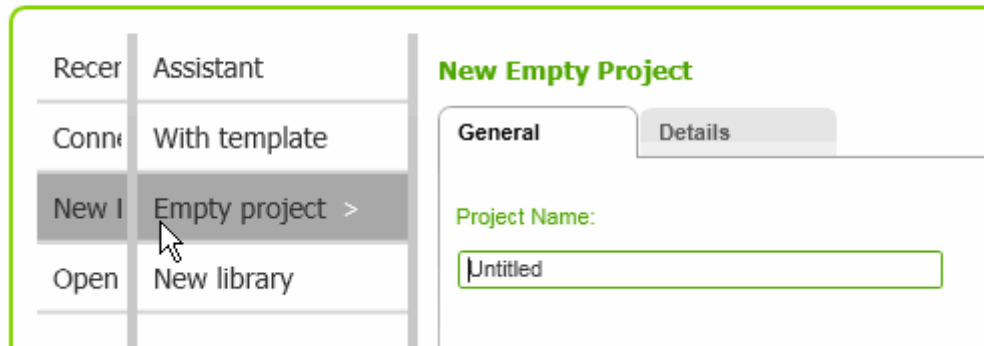
# New Empty Project

## Start with a New Empty Project

Starting a new project as an empty project means that there is no preconfiguration of devices or logic provided. This option provides you with the most flexibility but you must add everything to the project yourself.

### Get started

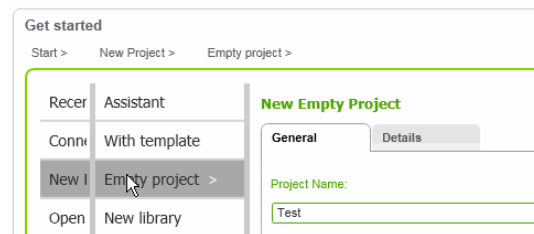
Start > New Project > Empty project >



## How to Create an Empty Project

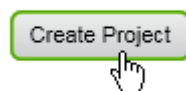
### ➤ To create an empty project:

Click the **empty project** menu item



Type the project name in the Project Name Field

Click **Create Project** to create. The project is created in the project directory that you specified earlier..

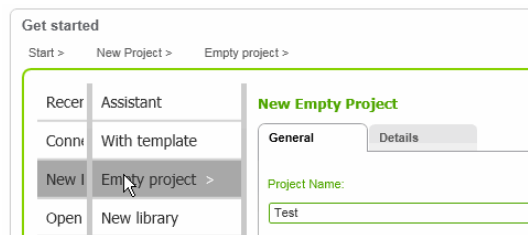




# Exercise - Create a New Empty Project

## 1 Create a new Empty Project.

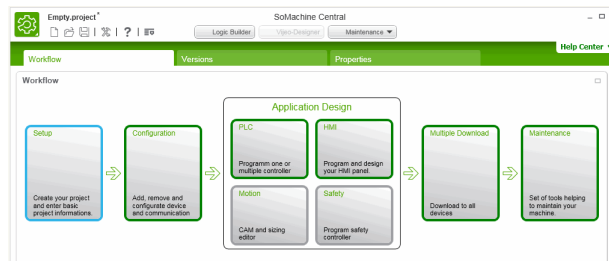
- i. Click the **New Project** item in the left pane.
- ii. Click the **Empty Project** item from the expanded menu.



- iii. Name your project "**empty**" (Test is shown above). Projects have the extension **".project"** so the actual name is **empty.project**.



- iv. When the project is created SoMachine will display the project workflow screen.

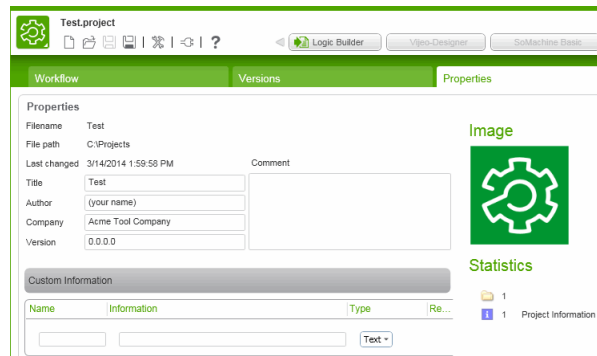


## Exercise - Create a New Empty Project (cont.)

---

### 2 Examine the project settings.

- i. Click the **Properties** tab. Add your name and the information shown.



- ii. Save and Close the project



---

### 3 Return to the Home screen.

---



# Archive Projects

## Save the Compiled Project

A project **Archive** is a very compressed version of the project. The compression rate can result in a file that is more than one fiftieth the size of the **project** file. The project file alone is much smaller than the original file but if libraries and other objects are included in the archive, the **resulting archive file is actually larger than the project file alone**. Archiving is similar to exporting except that it saves the compiled application. The consequences of this is that the user cannot archive a project until a successful compile can be completed. Until this is possible it is necessary to export.

## Archive Properties

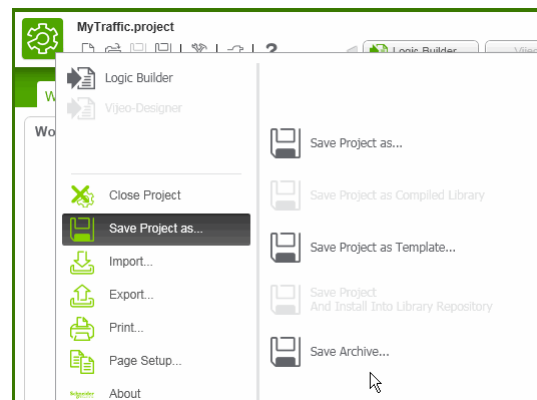
A project Archive is saved with the extension **.ProjectArchive**. The project archive file contains:

- The PLC binary code
- The Upload information
- Visualizations, device descriptions
- The Operator Screens

## How to Create a Project Archive

- **To create an archive:**

Select **Save Archive** from the Main Menu on SoMachine Central. The Archive will be created on the program path.



The system prompts you to indicate the options that you wish to include in the Archive. More items selected results in a larger archive

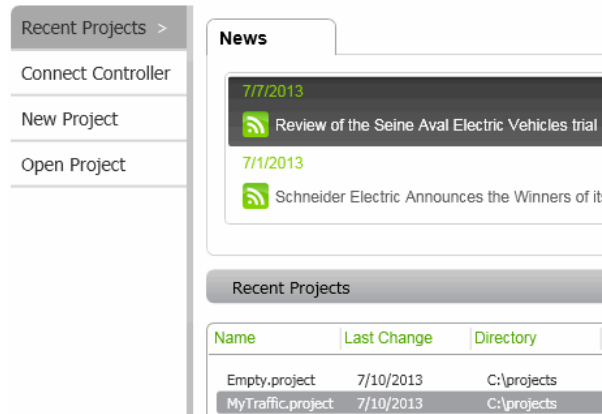


**Note** - Archiving is possible only if the project is **compiled**.

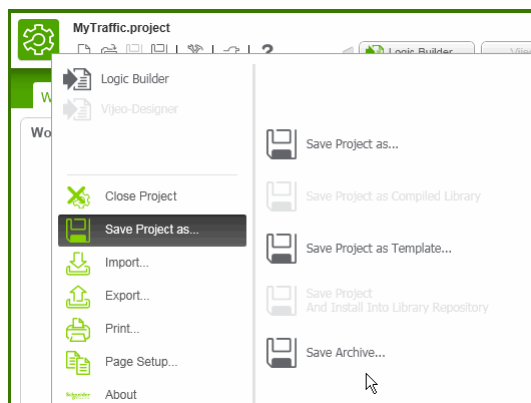
# Exercise - Archive Projects

## 1 Archive the MyTraffic project.

- i. Open your **MyTraffic** project from Recent Projects. Since you ran this project in the simulator, it is already compiled.



- ii. From the SoMachine Central, **Main Menu**, Select **Save Archive**.



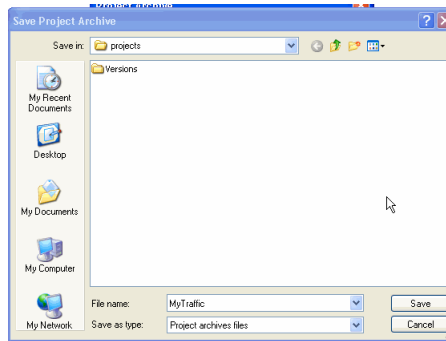
- iii. Save the file as **MyTraffic.projectarchive**.
- iv. Select the options that you wish to have included in your archive. Normally you would not include Referenced Libraries **unless** the project contains a custom library




## Exercise - Archive Projects (cont.)

---

- v. Save the Archive as **MyTraffic**. It will have a different extension from the project. Click the Save button when ready



- vi. Open Windows Explorer and compare the size of the two files **MyTraffic.project** and **MyTraffic.projectarchive**.

	MyTraffic	679 KB	SoMachine Suite.project data file
	scu	283 KB	SoMachine Suite.project data file
	Test	1,220 KB	SoMachine Suite.project data file
	MyTraffic	1,767 KB	SoMachine Suite.projectarchive data file

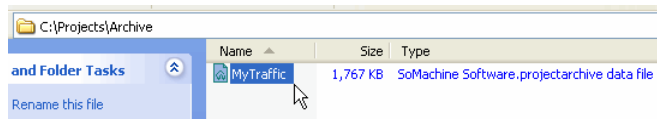
- vii. Which file is larger? \_\_\_\_\_
- viii. Why? \_\_\_\_\_
- 



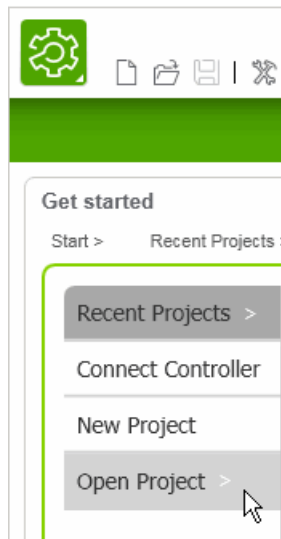
# Exercise - Restore an Archived Project

## 1 Create a Temporary Directory

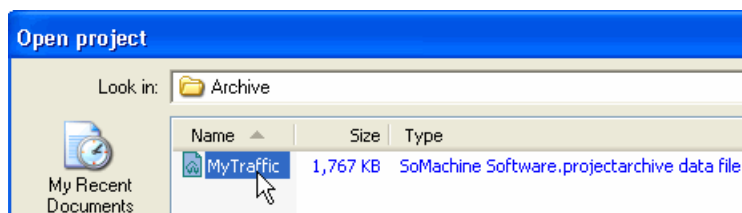
- i. Open your **C:\Projects** folder and create a subdirectory named **C:\Projects\Archive**
- ii. **Move the MyTraffic archive file there**



- iii. Return to **SoMachine Central** and close the **MyTraffic** project (if open)

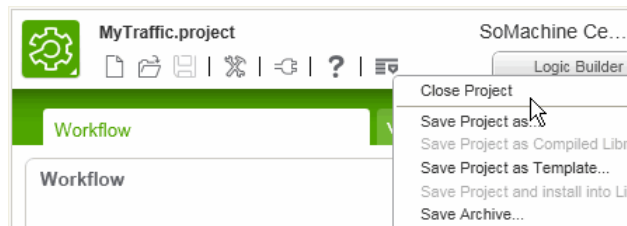


- iv. To open the archive, select the **Open Project** option

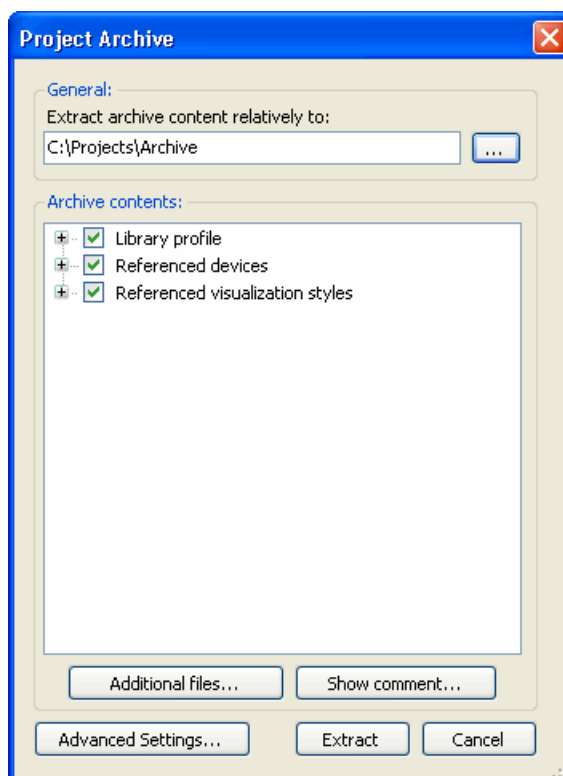


## Exercise - Restore an Archived Project (cont.)

- v. Navigate to the archive folder that you just created (**C:\Projects\Archive**) and double click on the **project archive**



- vi. SoMachine prompts you to confirm which parts of the project you wish to restore and where to restore it. Don't change anything and restore this project into the **C:\Projects\Archive** folder. Click **Extract** when ready.



## 2 Testing

- i. When the project has finished restoring, open the **Logic Builder.**, then **login to the Simulator**, **download** and **verify** the restored project works







# Chapter 3: SoMachine Central Functions

## Overview - Creating a Project

---

### Introduction

The Workflow screen is only displayed after a SoMachine project has been opened. It consists of a block diagram that shows all areas of the opened project. The actual project is further configured by opening a specific area. The editors consist of a graphical configuration editor that provides functions to perform the entire hardware and network configuration of the machine. The configuration settings performed here will also be available in the controller and Vijeo Designer Program screen

---

### Chapter Objectives

By the end of this chapter, the student will be able to:

- Describe the basics of a CANOpen network
  - Create a new project
  - Select hardware for a project
  - Configure project versioning
  - Configure a CANOpen bus
  - Create a custom SDO
  - Configure a controller's Ethernet port
  - Download a project to the controller via Ethernet
  - Create a program POU
  - Create a custom function block POU
  - Validate function block's operation
- 

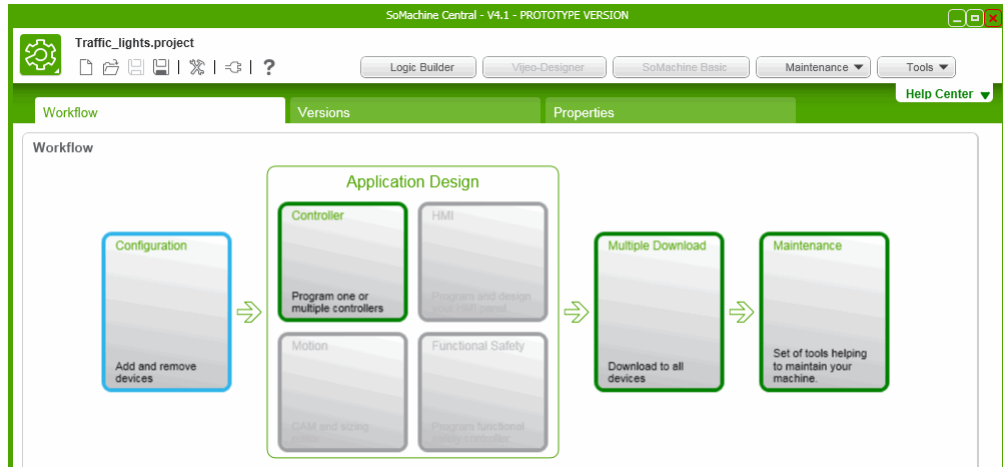
### This Chapter Covers These Topics:

- The Workflow Screen .....3-2
- Device Selection .....3-3
- Exercise - Add a Device to an Empty Project .....3-5

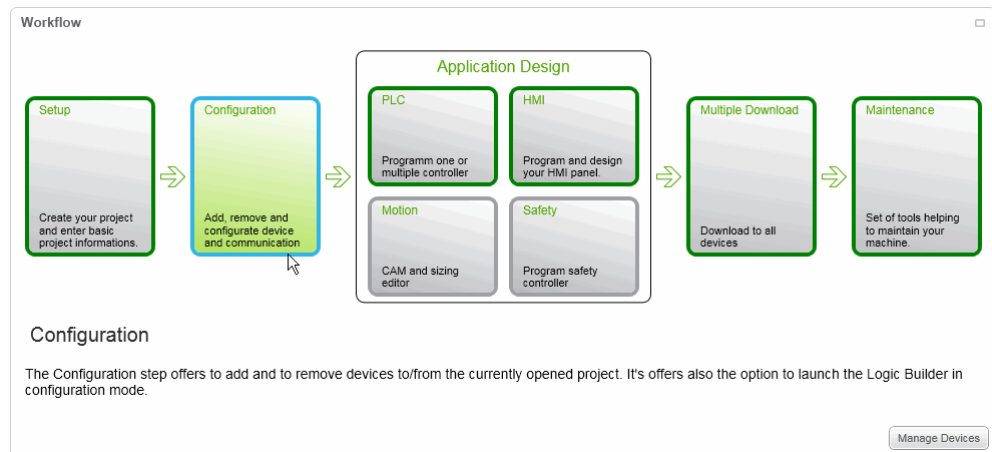
# The Workflow Screen

## The Workflow Screen Interface

Upon opening an existing project or starting a new project, the first screen seen is the Workflow screen. The Workflow screen provides a graphic representation of the entire opened project. Key areas that must be configured include the Configuration area and the Logic Builder.



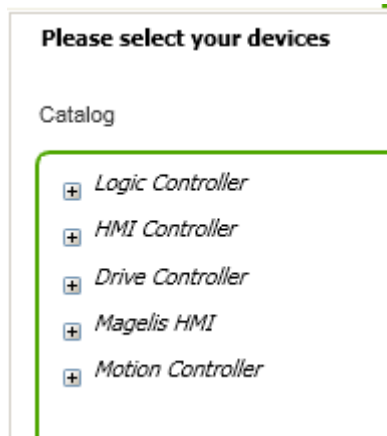
Controller selection is done from the Configuration block and is typically the next step in developing a project. If you select the Configuration block, it turns green. Click Manage devices to open the device catalog



# Device Selection

## The Device Catalog

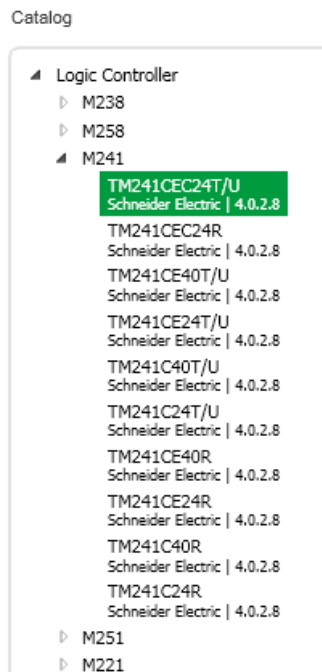
When the device catalog is open, the list of current devices that may be added to this project is displayed.



Available Devices are filed in different categories. The current categories are:

Category	Description
Logic Controller	M238,M258,M241, M221 controllers
HMI Controller	XBTTCC, HMISCU, XBTGT or GK models, with control
Drive Controller	ATV-IMC
Magelis Controller	Magelis models (no control) or XBTGC, HMISCU models
Motion Controller	LMC058 Motion controller

The M241 category expanded shows the various models.



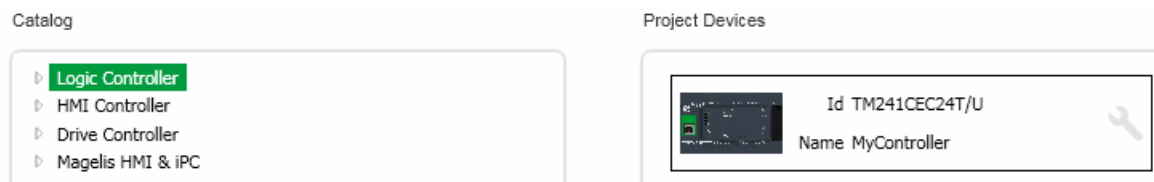
## The Device Catalog (Cont.)

---

### Device Added

#### ➤ How to Select a controller

To select a controller, select the controller from the expanded category list and use the right arrow to add the device to the Project Devices panel.



Once you have selected device(s), Somachine adds required, supporting libraries to the project. A project may have multiple devices included in it.

---

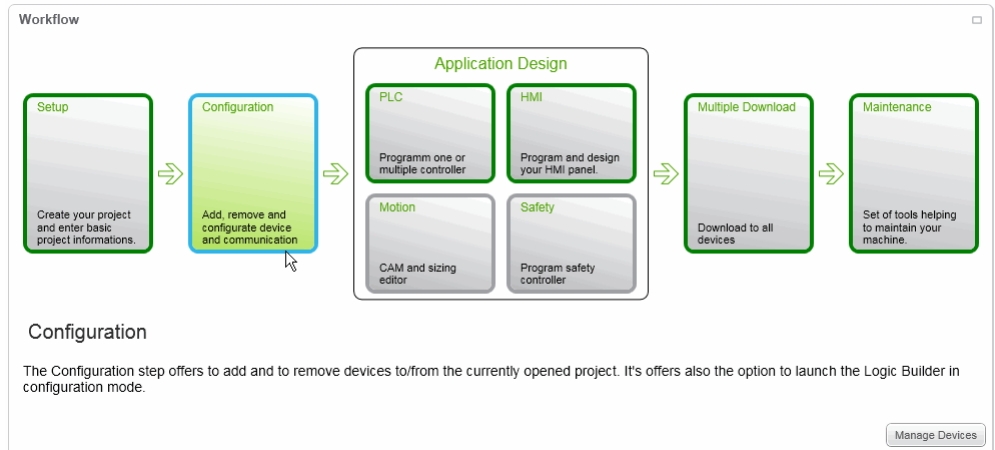
If you plan on having a Magelis HIM interface with a controller, it is strongly recommended that you include it in the same project with the controller. Doing so makes communication between the HMI and controller far easier to implement and provides more options

---

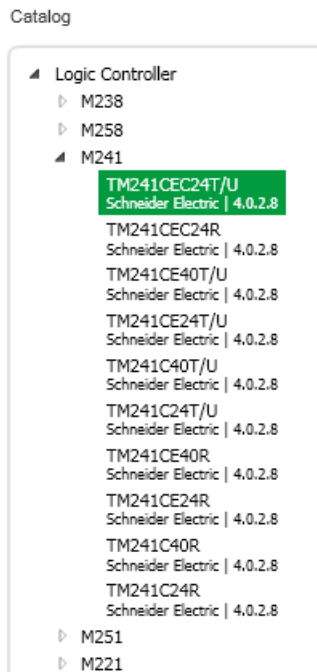
# Exercise - Add a Device to an Empty Project

## 1 Add a Device From the Workflow Screen

- i. Return to the **SoMachine Central** screen and open the **Empty.project** created earlier.
- ii. Select the Configuration block and click **Manage Devices**



- iii. Expand the logic controller category from devices catalog. Select the **TM241CEC24T** controller to the project



## Exercise - Add a Device to an Empty Project (cont.)

---

- iv. The **TM241CEC24T** is added to the project



---

### 2 Save the Project and Return to the Home (SoMachine Central) screen

---



# Chapter 4: New Project Creation

## Overview

### Introduction

---

SoMachine is able to use the six IEC standard languages. These are **Sequential Function Chart (SFC)** language and four inter-operable programming languages: **Ladder Diagram (LD)**, **Function Block Diagram (FBD)**, **Structured Text (ST)**, **Instruction List (IL)** and **Continuous Function Chart (CFC)**.

SFC is used mainly to control highly sequential processes, like repetitive machine operations. This is properly referred to as a state machine. It can also be used as a supervisory control mechanism.

---

### Chapter Objectives

By the end of this chapter, you will be able to:

- Understand types of tasks and their operation
  - Be able to use the simulation mode
  - Describe the six IEC languages supported by SoMachine
  - Create a POU Program
  - Create a POU function
  - Create a POU Function Block
  - Create a gateway
  - Create a POU program, download it to the simulator and run the application
- 

### This Chapter Covers These Topics:

- The Logic Builder .....4-3
- The Device Tree.....4-4
- The Tools Tree.....4-5
- The Application Tree .....4-7
- Work Area.....4-9
- Catalogs .....4-10
- Embedded Functions.....4-11
- Communication.....4-15
- Tasks .....4-16
- Controller Program Execution .....4-17
- POU Program Creation .....4-18
- Exercise - Create a POU .....4-21
- Initial USB Communications Configuration .....4-24
- Connecting to the Controller.....4-26

- Updating a Controller's Firmware.....4-30
- Task Configuration .....4-35
- Exercise - Configure a Task.....4-37
- PLC Simulator .....4-39
- Exercise - Using Simulation Mode.....4-41
- CoDeSys Program Languages .....4-45
- Exercise - Program a FBD POU .....4-46
- CoDeSys Program Languages (cont.).....4-51
- Exercise - Convert IL to LD .....4-52
- CoDeSys Program Languages (cont.).....4-53
- Exercise - Program a CFC POU .....4-55
- Watchdog Mechanisms.....4-56
- Structuring an Application.....4-57
- The POU Function .....4-58
- Exercise - POU Function .....4-59
- Sample Project .....4-62
- Exercise - Start the Escalator Project.....4-63
- Global Variables .....4-65
- Exercise - Add Global Variables .....4-66
- Exercise - Add a POU to the Escalator Project .....4-67



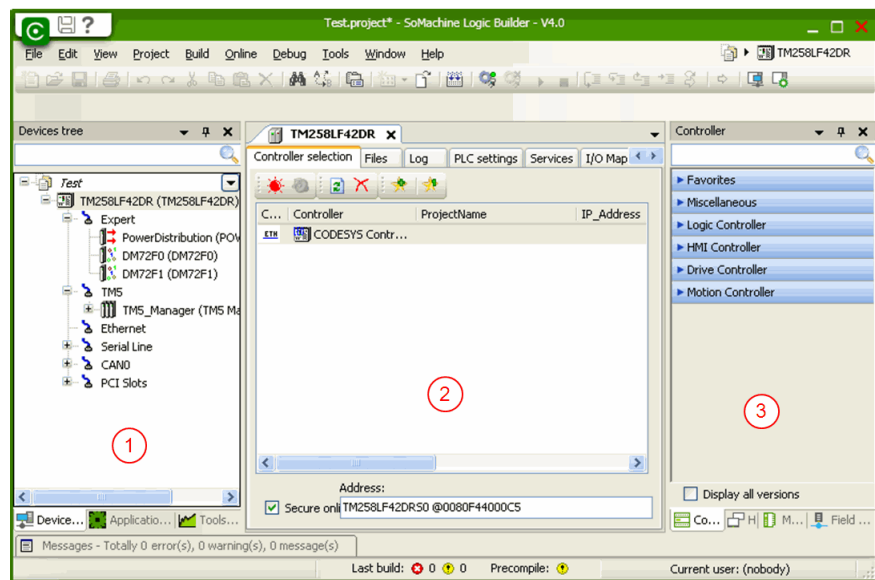
# The Logic Builder

## The Logic Builder



Open the **Logic Builder** to create the applications programming and complete detailed configuration. Access to the Logic Builder is from a button on the SoMachine Central screen labeled Logic Builder.

## Programming Screen

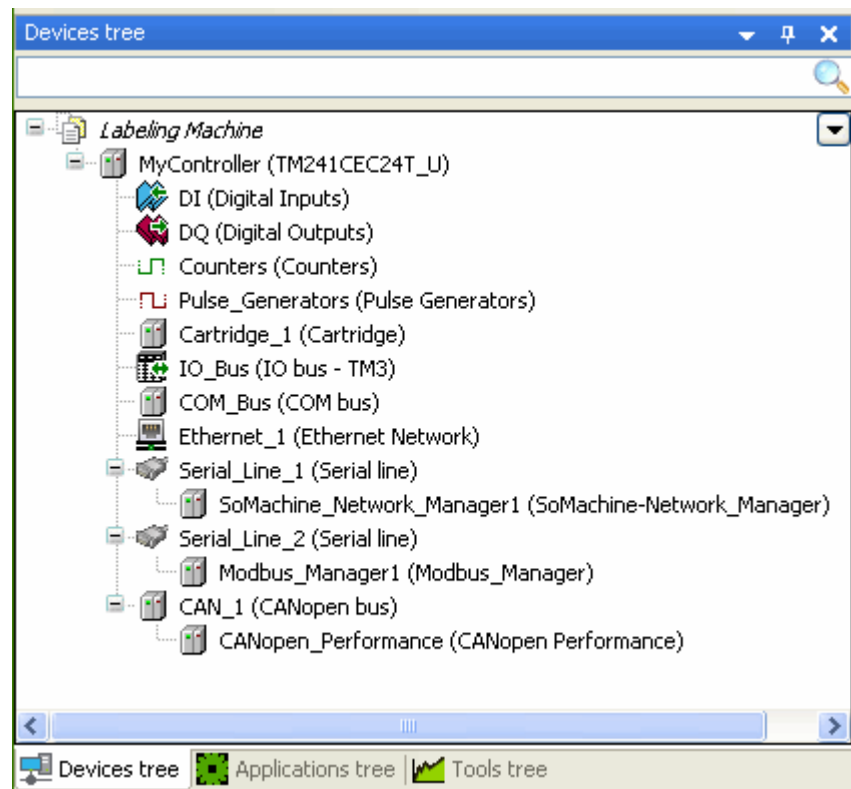


The Logic Builder GUI has been redesigned from earlier versions to simplify application development. There is now a clear separation between logic and configuration (1). Key areas of the logic builder are:

Number	Description
1	Device, Application and Tool Trees
2	Work Area - function varies depending on where you are in the software
3	Catalogs - options here change depending where you are within the software. In this image, hardware options that may be added to a project are displayed. If you were inside a POU, variables and functions/function blocks would be displayed.

# The Device Tree

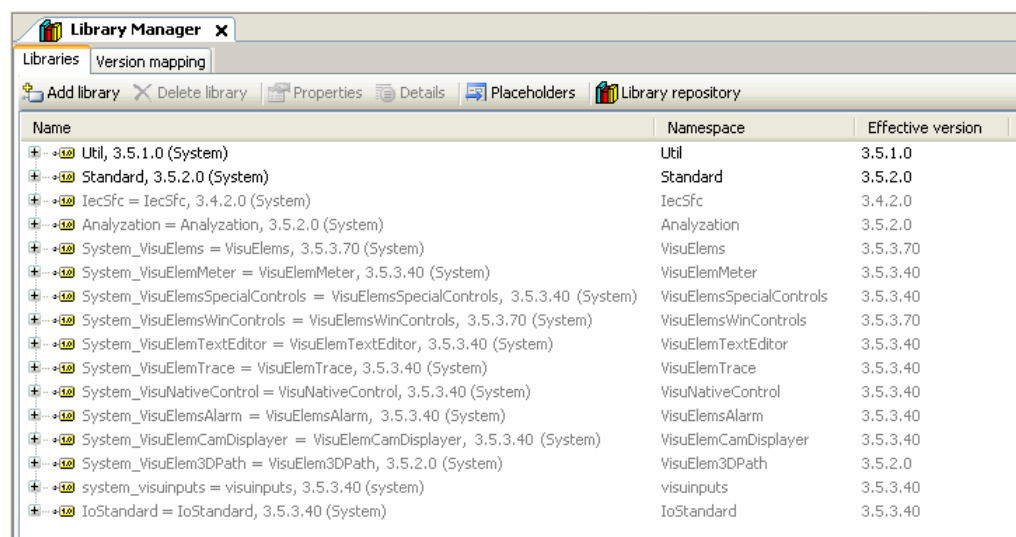
## Device Tree Details



The Device Tree is used to configure the hardware of the current controller. In the example shown, a TM241CEDC24T\_U controller is displayed. This controller has Ethernet, two-serial lines and a CANopen Master port that must be configured if they are to be used. I/O Variable mapping as well as HSC Counter(s) and Pulse Generator(s) can also be configured.

# The Tools Tree

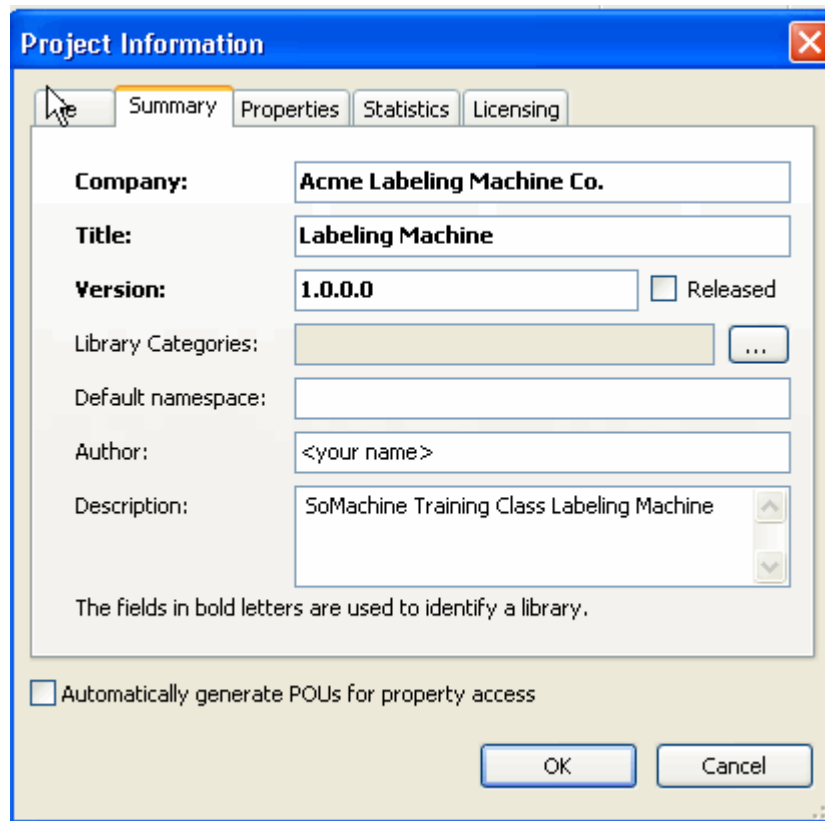
## Library Manager



The Library Manager allows you to see what libraries are in the current project as well as add optional libraries to the current project.

## Project Information

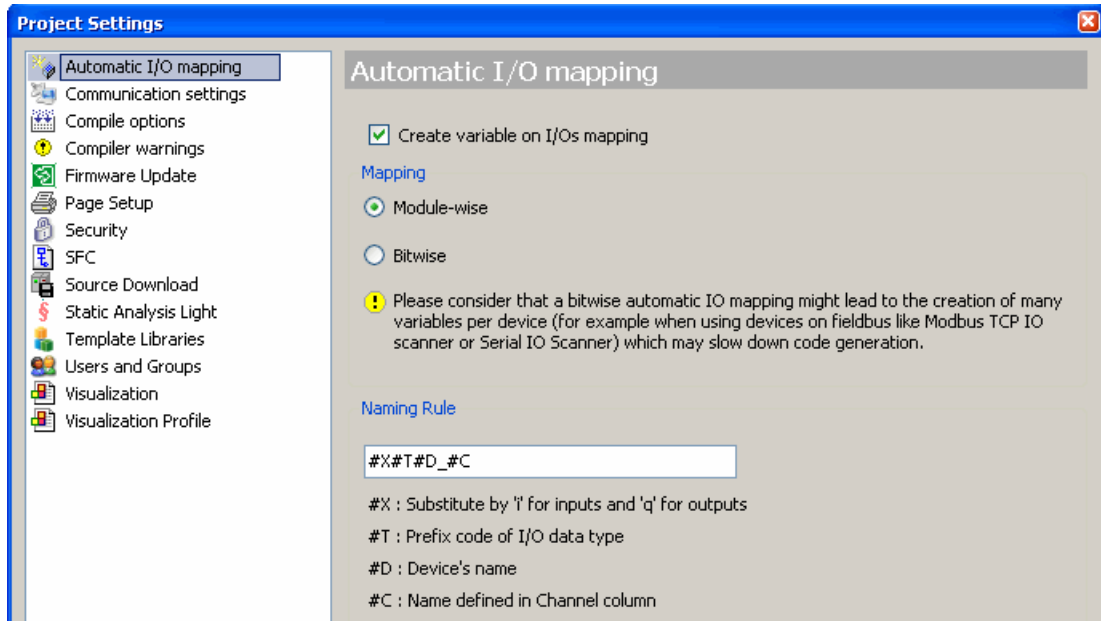
Project Information (Project » Project Information) displays or edits the information entered when the project was first created.



## Project Information (cont)

### Project Settings

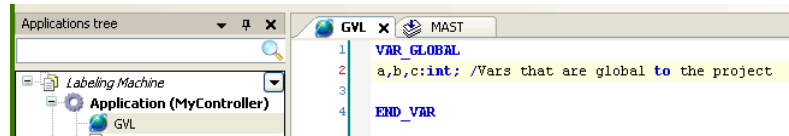
Project Settings (Project » Project Settings) allows you to customize the way the editor operates with the current project.



# The Application Tree

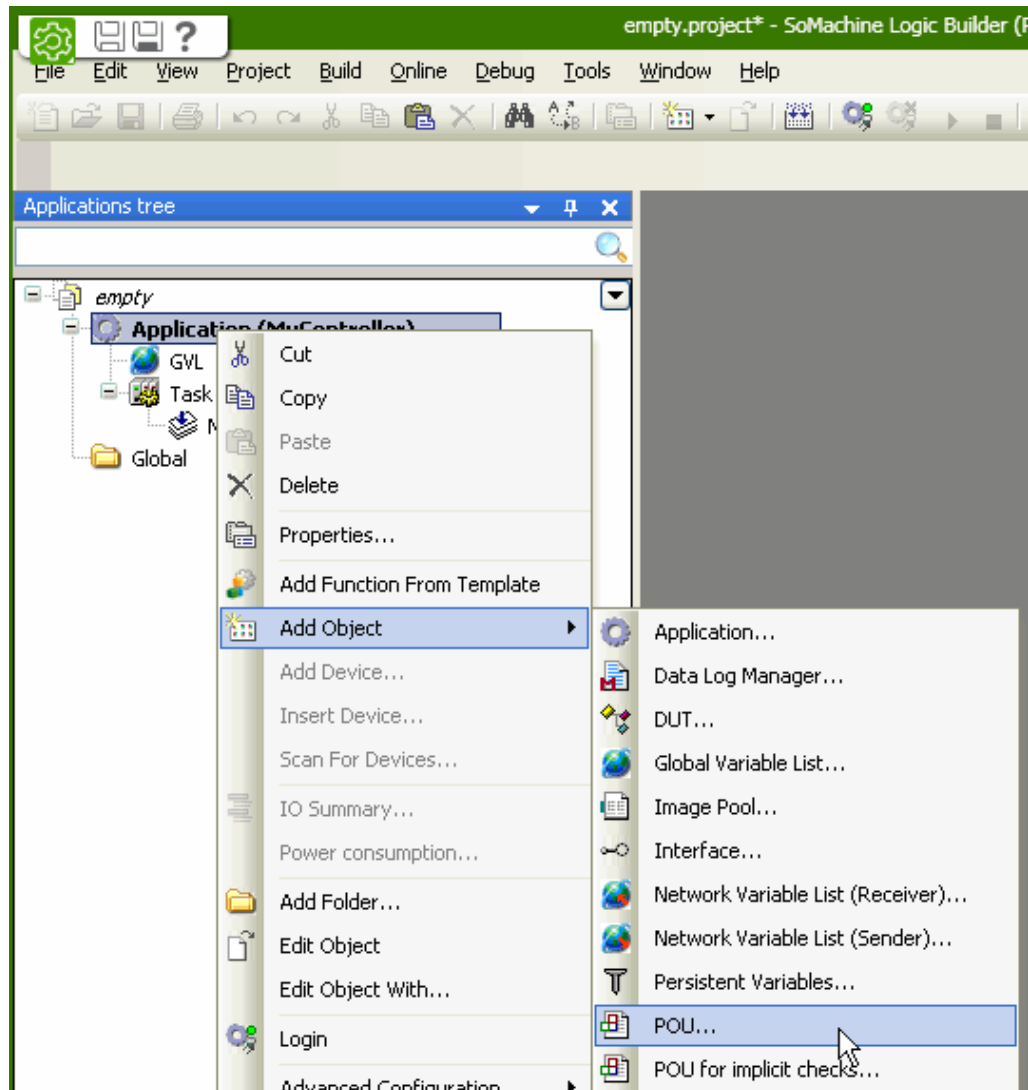
## GVL

Variables added to the GVL list are global to all POUs in the project. This means that the variables value is available in any POU and any POU can read/write these variables. In addition, these variables may be mapped to controller I/O points if desired.



## The Program POU

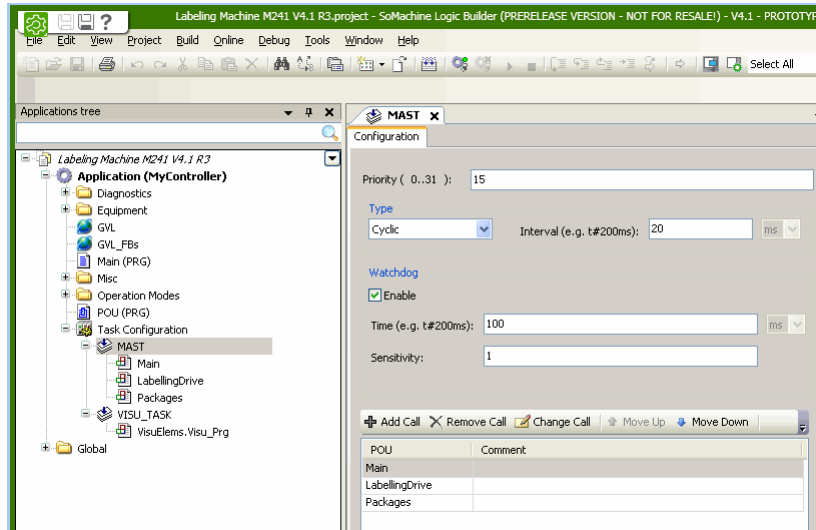
A Program POU (Program Organizational Unit) is created by selecting Add Object followed by POU from the Application Tree. A typical application consists of many POUs, each one providing a part of the overall solution.



## The Application Tree (cont)

### MAST Task

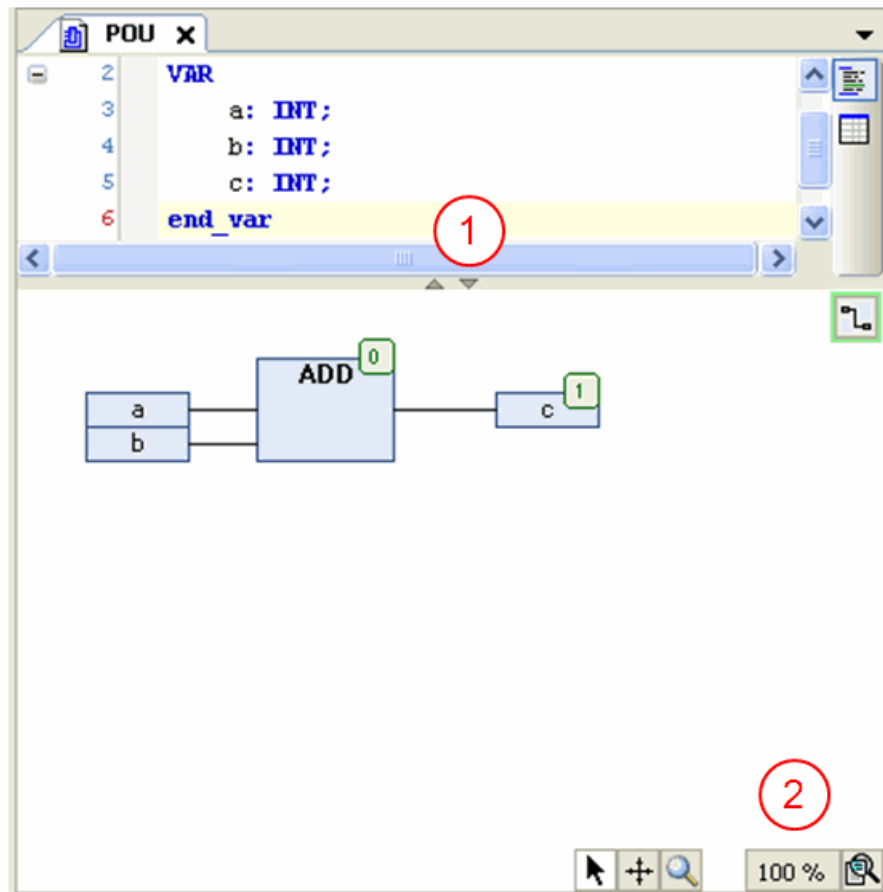
The **MAST Task** exists in all applications. It is a cyclic task that calls an application's POU(s) and controls their execution. The programmer must first create the POU(s) then add them to the MAST task. A POU is NOT automatically added to the MAST task upon creation. POU(s) are executed in the order they appear in the MAST Task. The Default cycle time for the MAST task is 15 ms.



# Work Area

## The Work Area

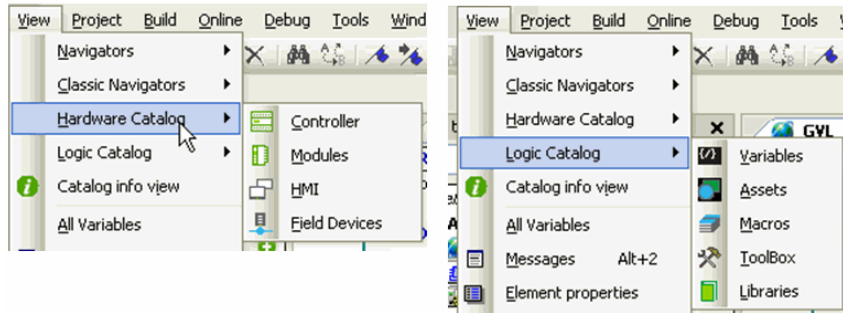
The work area appears as shown when creating a POU. The exact content of the work area depends on the operation being performed.



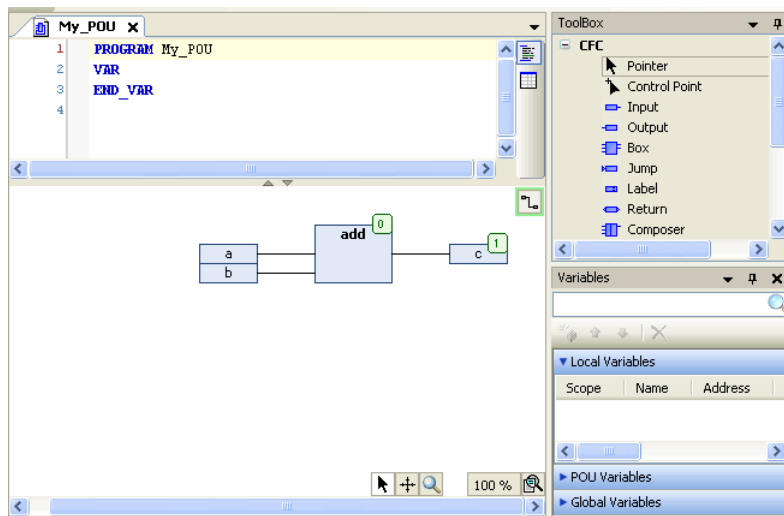
# Catalogs

## Catalog Introduction

One of the enhancements of the SoMachine 4.1.2 software is the use of Catalogs. A Catalog is a way of organizing items that are used to construct an application. Catalogs can be added to the project as needed from the View Menu. The image below shows the Hardware and Logic Catalog available options

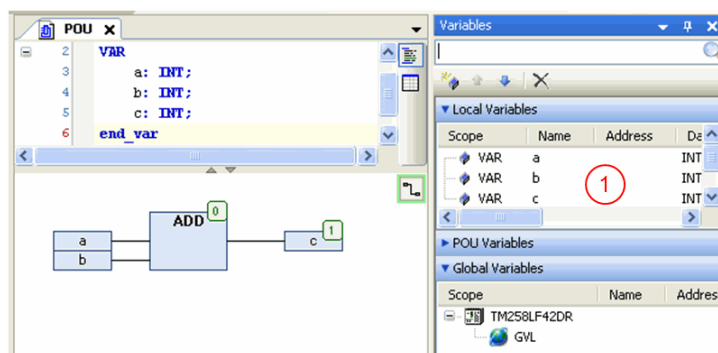


The next image shows the **Variable** and **Toolbox** catalogs added to a project



## Catalog - Variables

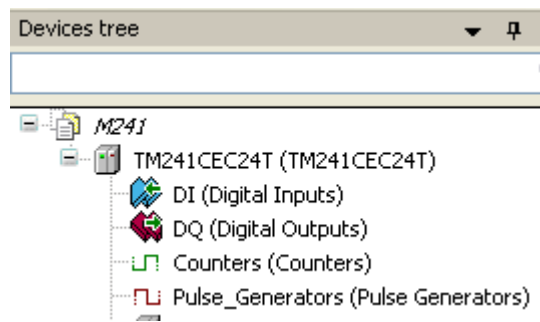
Notice that local variables may be seen in the traditional area (over the ADD block) or in the variable catalog (1)





# Embedded Functions

## Embedded Functions M241



The M241 has a number of embedded functions available for user configuration. Digital inputs or outputs can be configured for simple ON/OFF operation or they can have advanced functionality such as as high speed counters, Latches, or event generation.

Variables may be mapped directly to IO points a couple of different ways. The names shown in the figure below were automatically generated by the system when the project was first started. These variables follow the Schneider Electric recommended naming convention refereed to as the **Hungarian Notation** (see the Help Screens for more information). .

The screenshot shows the 'I/O Configuration' window for the 'DI' function. The 'Channels' section contains a table with the following data:

Variable	Mapping	Channel	Address	Type
Inputs				
		IW0	%IW0	WORD
ixDI_I0		I0	%IX0.0	BOOL
ixDI_I1		I1	%IX0.1	BOOL
ixDI_I2		I2	%IX0.2	BOOL
ixDI_I3		I3	%IX0.3	BOOL
ixDI_I4		I4	%IX0.4	BOOL
ixDI_I5		I5	%IX0.5	BOOL
ixDI_I6		I6	%IX0.6	BOOL

Auto I/O mapping may be controlled from the **Project»Project Settings»Automatic I/O Mappings Menu**. Automatic mapping can be set to be generated by word or by bit but not both.

## Embedded Functions M241 (cont.)

### ➤ How to map variables to I/O Points

To map variable(s) to an IO point you can:

- Accept the system generated defaults (above)
- Click on default variable name and type your own name in
- Click on default variable name and browse for a variable that has already been created in the GVL (Global Variable List).

---

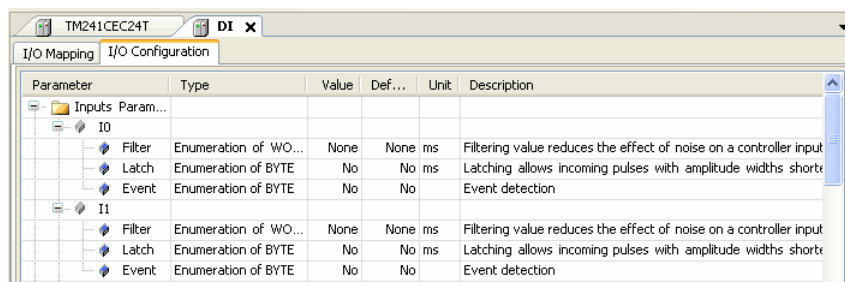
**Note** - in each case, the variable associated with an IO point is global. Care must be taken not to have a variable declared as both local to a POU and global. This is a **programming error** and the system will always use the local declaration.

---

### ➤ How to assign an advanced function to an IO point

To assign an advanced function to an IO point, click in the Value field to the right of the function and select the operation desired. A brief description of the functions available are

- **Filter** - assigns a 1, 4 or 12ms filter to an input point. Filters reduce the effect of noise on an input point. In counting applications, a mechanical switch may "chatter" when activated. This results in multiple counts when a switch closes. Filters, will eliminate this such that only a single count is recorded
- **Latch** - In high speed applications, an input point may go ON or OFF in less than a scan. This can result in the system never seeing a point transition. Assigning a latch to a point instructs the system to "remember" that an input transition has taken place
- **Event** - Input point(s) configured as an "event" create interrupt processing. This works in conjunction with a configured event task. Event processing can be on the leading, trailing or both edges of the signal on the input point. When the signal on the configured input point occurs, the controller stops it's normal program execution and processes the associated event task, thus creating an interrupt.



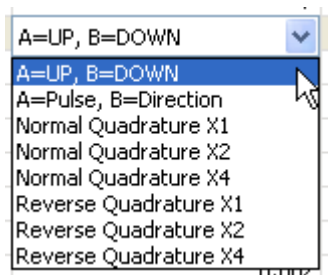
Parameter	Type	Value	Def...	Unit	Description
Inputs Param...					
I0					
Filter	Enumeration of WO...	None	None	ms	Filtering value reduces the effect of noise on a controller input
Latch	Enumeration of BYTE	No	No	ms	Latching allows incoming pulses with amplitude widths shorter
Event	Enumeration of BYTE	No	No		Event detection
I1					
Filter	Enumeration of WO...	None	None	ms	Filtering value reduces the effect of noise on a controller input
Latch	Enumeration of BYTE	No	No	ms	Latching allows incoming pulses with amplitude widths shorter
Event	Enumeration of BYTE	No	No		Event detection

## Embedded Functions M241 (cont.)

---

**Counters** - The M241 offers a variety of different counters. A brief description of counters is provided below:

- **HSC Simple** - there are two modes for simple counters; one-shot and modulo loop. These counters basically start at a setpoint (modulo) and count down to zero. A modulo loop counter resets itself for the next counting cycle on the pulse after it has reached zero. a one-shot counter requires a user provided signal to reset for the next counting cycle
- **HSC Main Single Phase** - Main counters are more complex and there are more configurable options. This counter has three sub modes. One-shot, modulo loop and event counting. A brief description is as follows; one-shot counters count in the same manner as a simple one-shot counter but also have the possibility of reflex actions at programmed thresholds. Modulo loop counters also count the same as their simple counterpart and have reflex actions available. Event counting counts how many times an input (event) occurs during a user programmed period of time
- **HSC Main Dual Phase** - There are two sub modes in this counter category - modulo loop or free large. The main difference here is that you can use a variety of different input signals for the counting action. Multiple channels control the way the input signals are used. Encoder inputs can also be used. The figure below shows the configurable input options

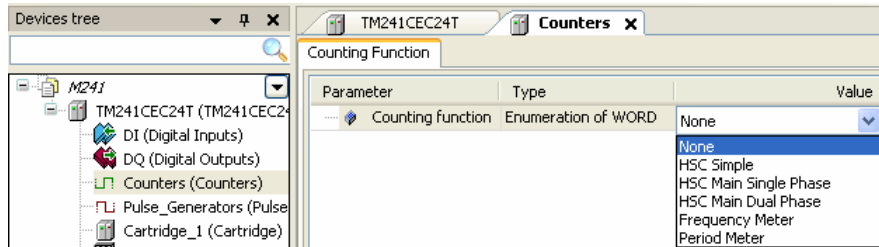


- **Frequency Meter** - displays the frequency of a signal applied to the input
- **Period Meter** - Use the Period meter type to:
  - determine the duration of an event
  - determine the time between 2 events
  - set and measure the execution time for a processThe Period meter can be used in 2 ways:
  - Edge to opposite: Allows the measure of the duration of an event.
  - Edge to edge: Allows the measure of the length of time between 2 events.

## Embedded Functions M241 (cont.)

### ➤ How to configure a High Speed Counter

To configure a counter you must select Counters, from the Devices tree, select the type of counter desired, select the sub mode of the selected counter, configure the options desired.



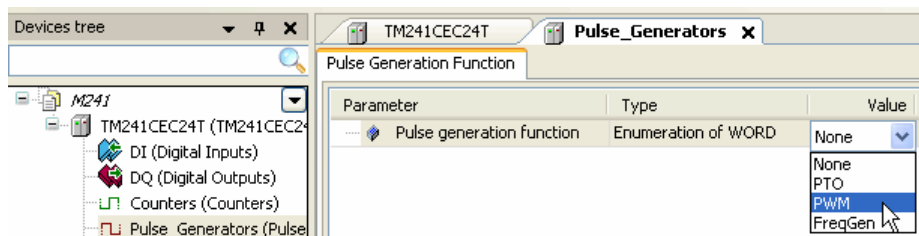
Note - this is a brief introduction on counters. More information is available in the help screens or in the full training course

**Pulse generators** - The Pulse Generator function is used to create PTO, PWM or FreqGen outputs.

- FreqGen option generates a square wave signal on dedicated output channels with a fixed duty cycle (50%).
  - Frequency is configurable from 1 Hz to 100 kHz with a 1 Hz step
- The PTO, PWM, and Frequency (Pulse) Generator functions use the same dedicated outputs. Only one out of these 3 functions can be used on the same channel. Using different functions on channel 0 and channel 1 is allowed.

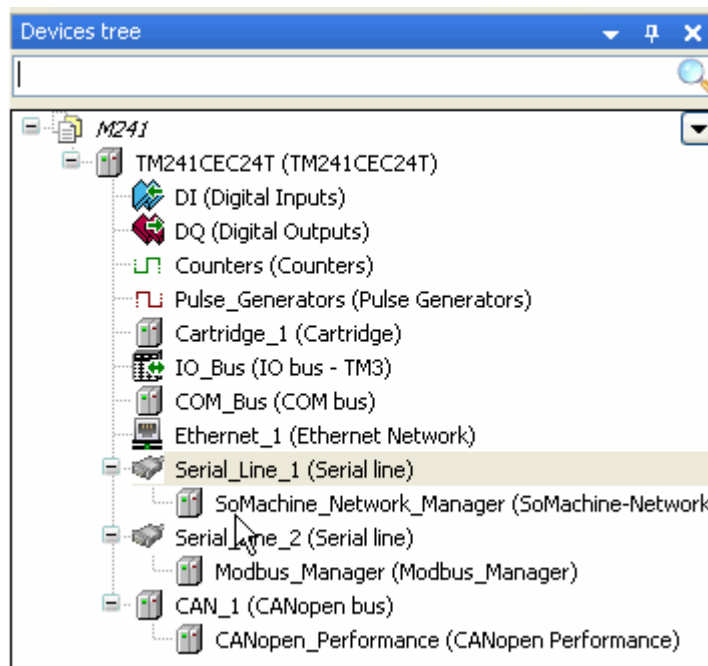
### ➤ How to configure a Pulse Generator

Select Pulse Generators from the device tree. Select the mode of operation. Configure the mode selected



# Communication

## Communications



The M241 has two configurable serial ports

By default, devices are preconfigured. For example, the **TM241CEC24T** device is preconfigured as follows:

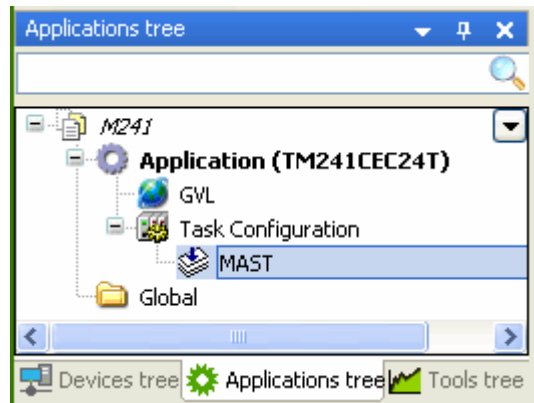
- Serial Line 1: SoMachine Network Manager
  - Commonly serial communications to a Magelis HMI
- Serial Line 2: Modbus manager
  - RS232/485, RTU/ASCII
  - Communications to any Modbus devices
- **CANbus**
  - Connect to remote CANopen devices
  - OTB modules, drives or any external CAN device

# Tasks

## Task Basics

A **Task** is an organizational object that consists of a collection of POU(s) (Program organizational Unit) that controls their execution. Tasks may be executed cyclically (MAST task) or periodically (a user selected time). Other types of execution are also possible

**Task Configuration** allows the definition of one or several tasks to control the execution of an application program.



The SoMachine allows the configuration of up to **seven** tasks with the restrictions listed below.

There are **Five** types of tasks:

- **Cyclic (3 max, MAST is one)** - executed on a user defined time schedule... e.g., every 50 ms
- **Event (2 Max)** - executed on transition (L> H, H>L or change) on event tag
- **Freewheeling (1 Max)** – starts with program and cycles, no specific time
- **External Event (4 Max)** – (not in menu) executed when designated “system event” is TRUE. Example - embedded input = ON or OFF or Both

---

### See Also:

For further information about **Maximum Number of Tasks** for each platform, see *SoMachine Help* - and search for *Maximum Number of Tasks*.

---

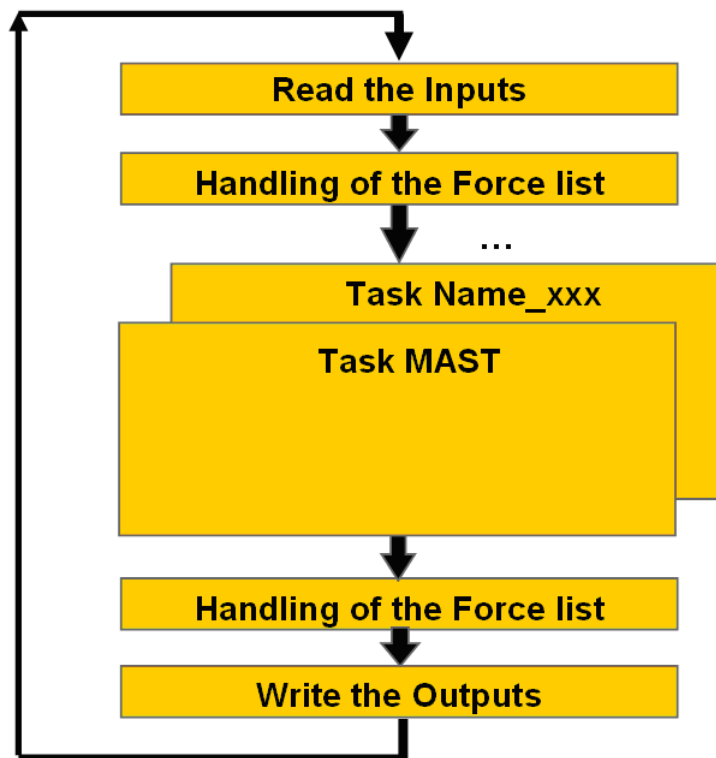
## Task Triggering

**Tasks** may be triggered by:

- A time (cyclic, freewheeling)
- An internal or external event
  - The rising edge of a global project variable
  - An interrupt event of the controller
- The combination of priority and condition determine the order that the tasks will be executed

# Controller Program Execution

Program Cycle



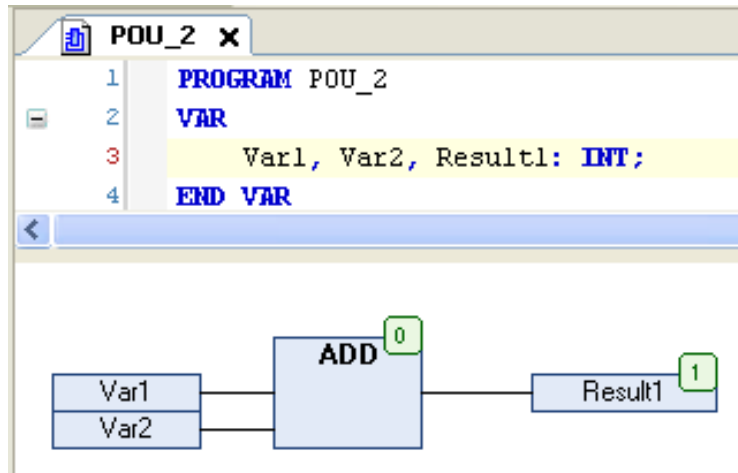
The diagram above describes a PLC Program cycle.

- **Local I/O** - processed by the tasks which use them (Task MAST, Task Name-xxx,...).
- **Expansion Module I/O** - processed by the MAST task only, not by other tasks if they exist

# POU Program Creation

## Program Organization Units (POU)

The **Devices** window allows users to add **POUs** (Program Organization Units) to the application. A **POU** is an object in SoMachine where programming code is written.



The different types of POU are:

POU Type	Description
<b>Program POU</b>	Returns one or several values during operation. All values are retained from the last time the program was run until the next. It can be called by another POU.
<b>Function Block POU</b>	Provides one or more values during the processing of a program. As opposed to a function, the values of the output variables and the necessary internal variables shall persist from one execution of the function block to the next. So invocation of a function block with the same arguments (input parameters) need not always yield the same output values.
<b>Function POU</b>	Yields exactly one data element (which can consist of several elements, such as fields or structures) when it is processed. The call in textual languages can occur as an operator in expressions.

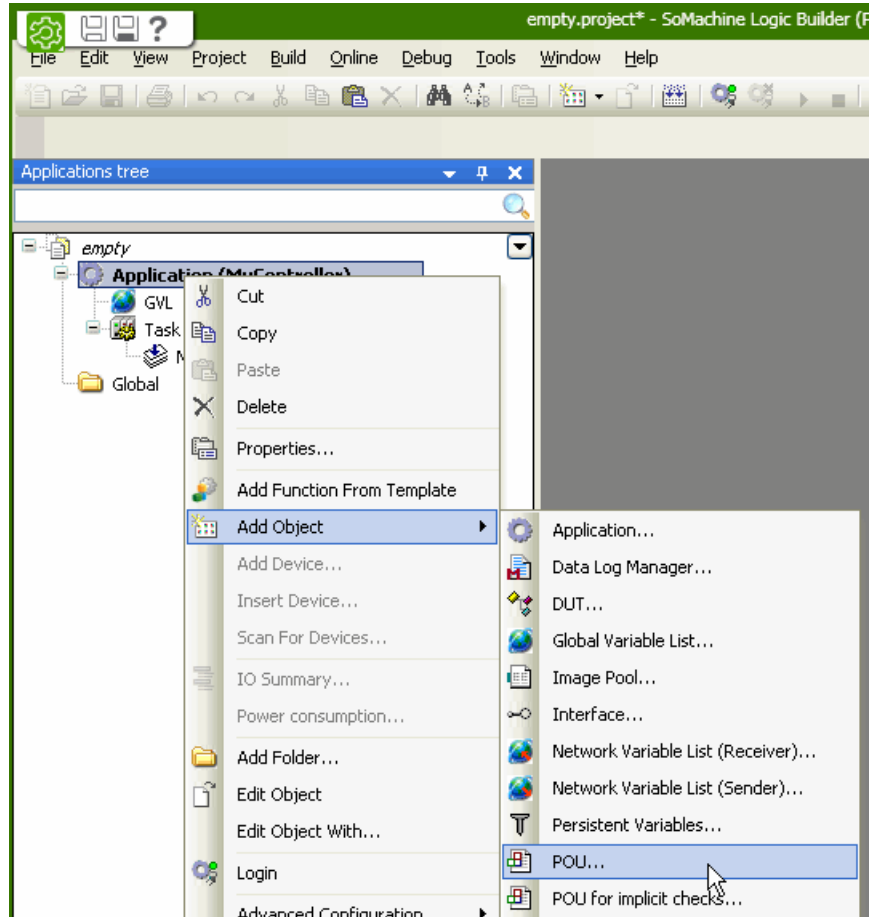


## POU Program Creation (cont.)

### How to Create a POU

- To create a POU:

Right-click **Application** and select **Add Object » POU** from the menu.



## POU Program Creation (cont.)

### How to Create a POU (cont.)

Enter a **Name** and an **Implementation Language**.

**Add POU** [X]

Create a new POU (Program Organization Unit)

Name:  
POU\_LD

Type:

- Program**
- Function Block**
  - Extends: [ ] ...
  - Implements: [ ] ...
- Function**
  - Return type: [ ] ...

Method implementation language:  
Structured Text (ST)

Implementation language:  
Ladder Logic Diagram (LD)

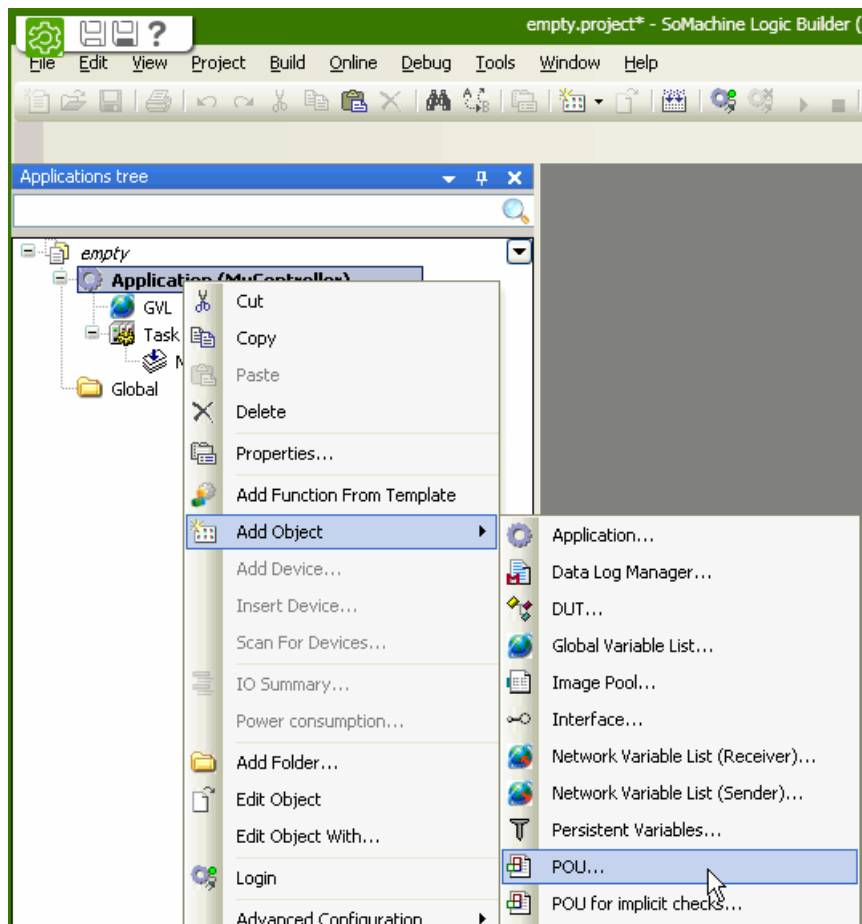
Open Cancel

All POU's are created in this manner. When finished, they must be scheduled to run by assigning them to a **Task**. POU's can also be "called" directly from another POU making the called POU in effect, a subroutine

## Exercise - Create a POU

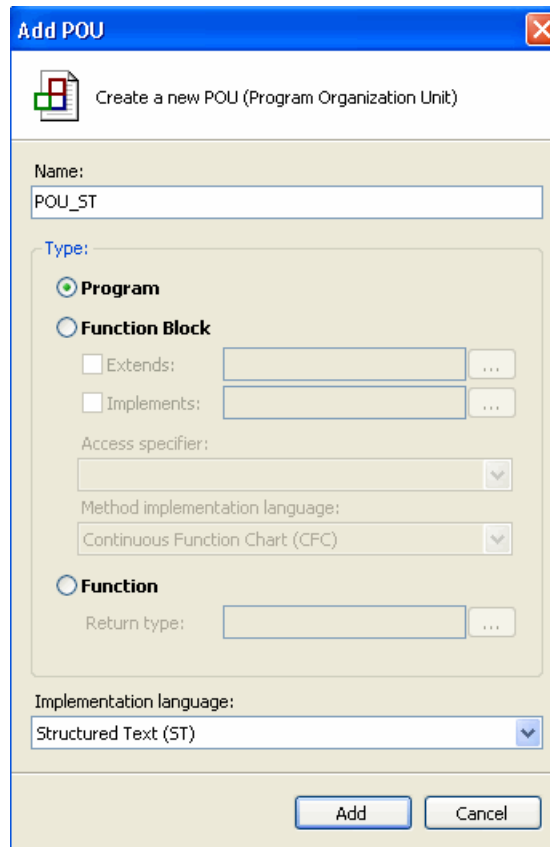
### 1 Use the Program screen to create a POU.

- i. Return to the **Home Screen** and open the **Empty.project** created in **Exercise - Create a New Empty Project**
- ii. Open the **Logic Builder**.
- iii. Right click the **Application** node and select **Add Object » POU** from the menu.

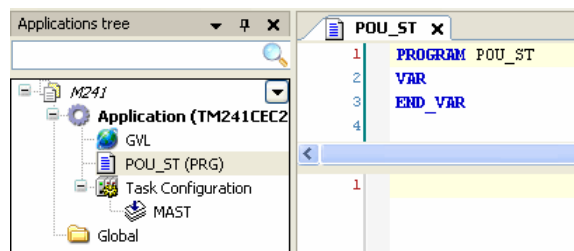


## Exercise - Create a POU (cont.)

- iv. When the **Add POU** dialog opens type the name **POU\_ST** in the **Name:** field then select **Structured Text (ST)** in the **Implementation Language:** field. Click the **Open** button when complete.



- v. The **Devices** pane will display the new **POU** and the **Work** area will be open ready for the user to enter the new program.



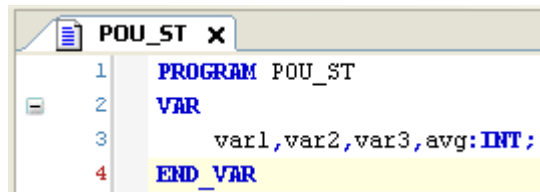
- vi. Click the **Save Program**  button on the main toolbar to save the project.

## Exercise - Create a POU (cont.)


---

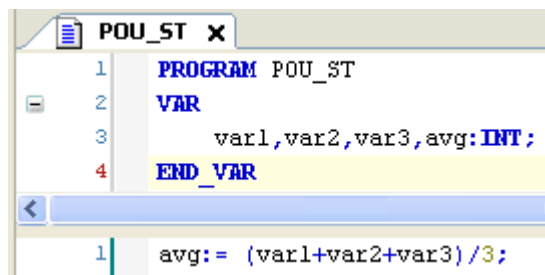
### 2 Create a program to calculate the average of three variables.

- i. Return to the **Devices** pane and double click the **POU\_ST** item to open the programming editor.
- ii. Create four **INT** variables shown between the **VAR** and **END\_VAR** (use the down arrow to expand the local var area). These are local variables and are only usable in this POU. Be careful with the syntax.



```
POU_ST x
1 PROGRAM POU_ST
2 VAR
3     var1,var2,var3,avg:INT;
4 END_VAR
```

- iii. Add the code shown in the lower window. Click the **Save Project**  button on the main toolbar to save the POU.



```
POU_ST x
1 PROGRAM POU_ST
2 VAR
3     var1,var2,var3,avg:INT;
4 END_VAR

1 avg:= (var1+var2+var3)/3;
```

---

### 3 Save the project.

---



# Initial USB Communications Configuration

---

## Installing the USB Driver

The first time a controller is connected to the configuring PC, the USB communications driver must be installed before communications can take place between the PC and the controller. This is a one time operation but must be done correctly.

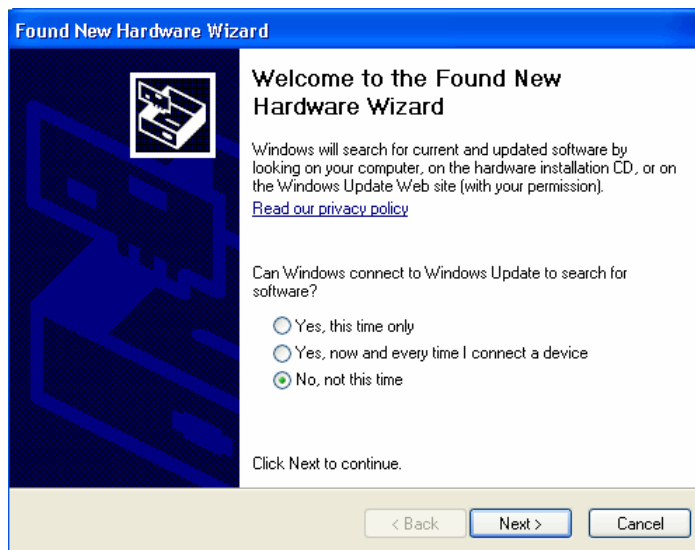
## How to Install the USB Driver

### ➤ To Install a USB Communications Driver:

The PC detects and identifies the new hardware. In this example, it's a TM241 Controller



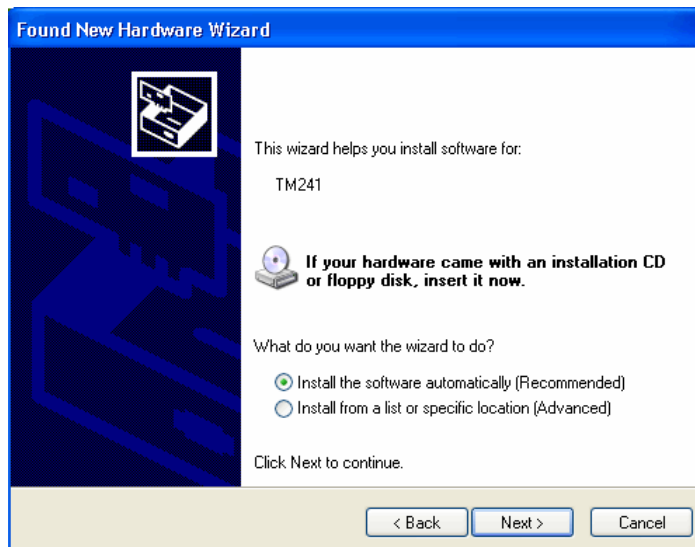
Select the No, Not this time option



## How to Install the USB Driver (cont.)

---

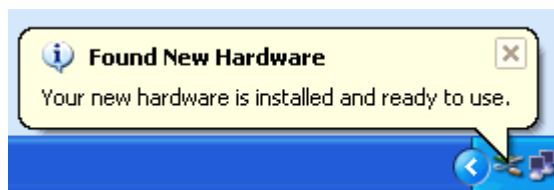
Instruct the PC to install the drivers automatically



The PC starts installing the driver



Upon successful installation, the PC informs you that it is ready to use.



You can now use the USB programming cable to download applications to the controller

# Connecting to the Controller

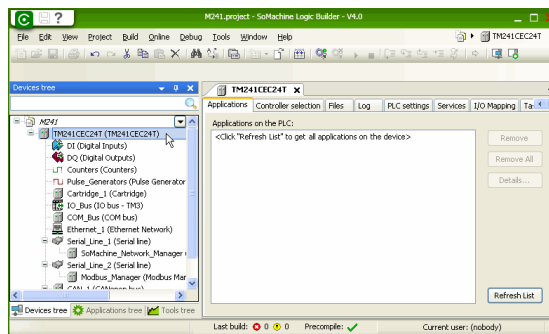
The connection from the PC to the controller is done using the SoMachine Gateway.

## How to Add a Gateway

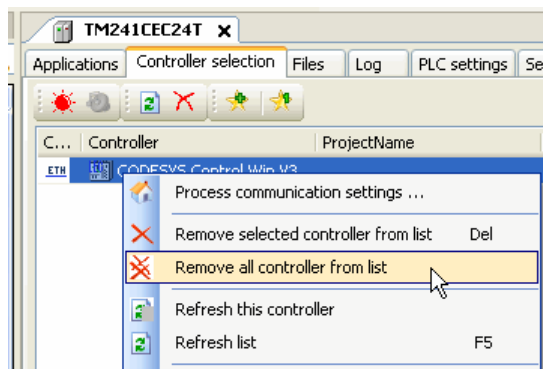
A Gateway is the communications setup for transferring an application to a controller. Gateways can be via a USB cable or Ethernet. This section shows how to add a USB gateway. An Ethernet gateway is added at the end

### ➤ To add a Gateway:

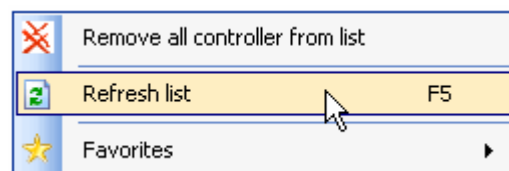
Double click on the controller from the Devices Tree. This must be done from the devices tree. The controller tab launches (right pane)



Open the Controller selection tab. right click on anything there and select Remove all controllers from the list. This is not really necessary but for the sake of explaining how the gateway setup is done, it will be done in this example.



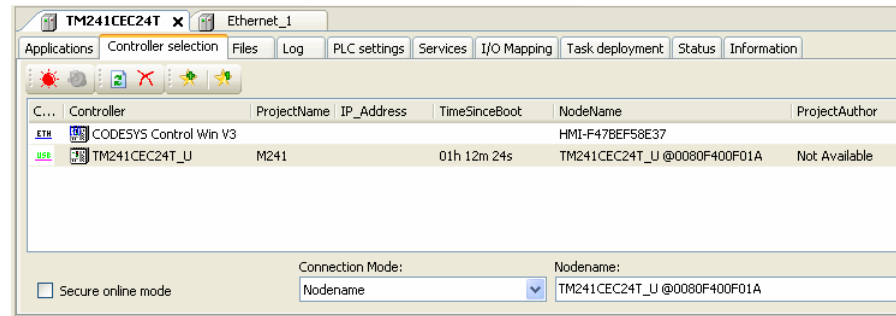
Insure that the USB cable is plugged into both the controller and the PC and right click in the now open area and select Refresh list. This causes the software to scan for any controllers. There is also a Refresh list icon in the Controller Selection toolbar that can be used.



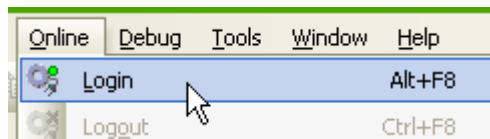


## How to Add a Gateway (cont.)

The software finds the controller on the USB cable and displays it. Insure that the Connection Mode at the bottom of the screen is set to Node Name as shown



Select **Login** from the Online Menu

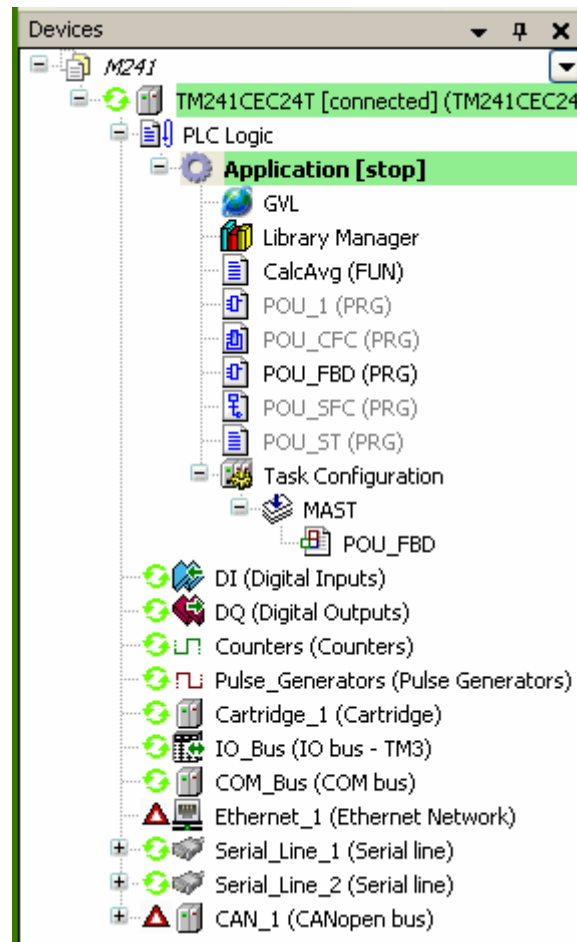


If your gateway configuration is correct, you will see the next screen. Click <Alt +F> as the message indicates



## How to Add a Gateway (cont.)

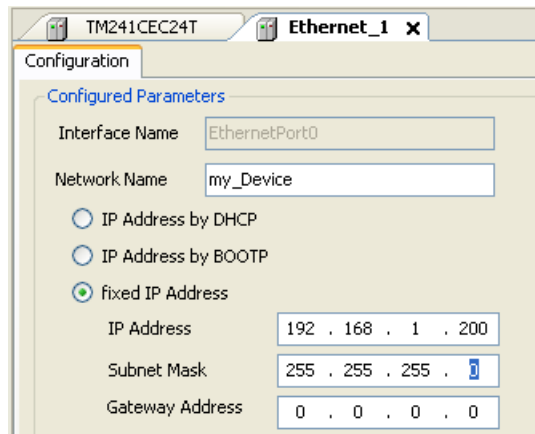
SoMachine will download your application and connect. When connected, you will see green, (healthy) icons in the Devices Tree



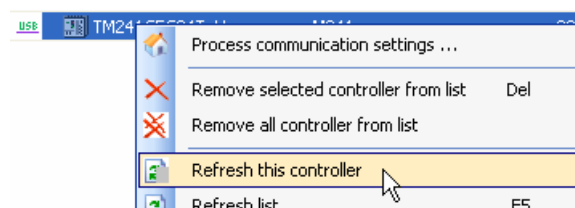
## How to Add a Gateway (cont.)

### ➤ How to Create an Ethernet Gateway

Double click on the **Ethernet** tree item in the Devices menu and add the IP address for the controller. Log back in. SoMachine informs you that an online change download cannot be performed. This is always true when you change the controllers configuration. Perform the full download. The controllers IP address is downloaded to the controller



Logout, connect an Ethernet cable to the controller and plug into Ethernet network being used. Then right click on the controller in the Controller Selection tab and select Refresh this controller



SoMachine sees the Ethernet connection and adds the IP address to the controller

C...	Controller	ProjectName	IP_Address	TimeSinceBoot	NodeName
ETH	CODESYS Control Win V3				HMI-F47BEF58E37
USB	TM241CEC24T_U	M241	192.168.1.200	00h 08m 44s	TM241CEC24T_U@0080F400F01A

Unplug the USB cable and again login. Since only Ethernet is possible, the connection is made via Ethernet.

**Note** - Since the communication setting above was set to be by "**Node Name**" There are now two possible methods of communicating with this controller. Either USB or Ethernet can be used. If multiple controllers are present in the controller selection tab, double click on the one that is to be the active connection. It turns bold when selected.

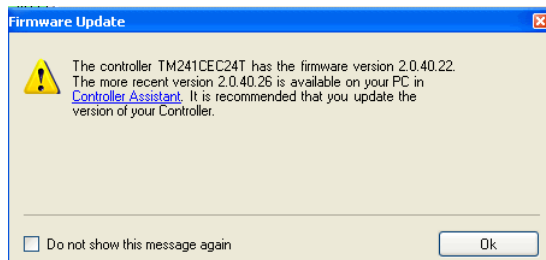
# Updating a Controller's Firmware

## How to Reflash a Controller

If you connect to a controller that has an older version of firmware, SoMachine informs you that a newer version is available. This usually happens if it is a new controller, fresh out of the box or if you have updated the SoMachine software installation.

### ➤ To Reflash a Controller

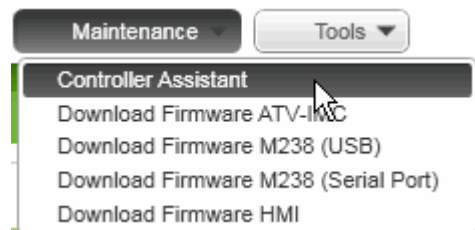
A similar message to the one below is displayed when connecting to a controller with an older version of firmware installed than the version available in the programming PC. The updating process is done via the **Controller Assistant** (link in message in blue)



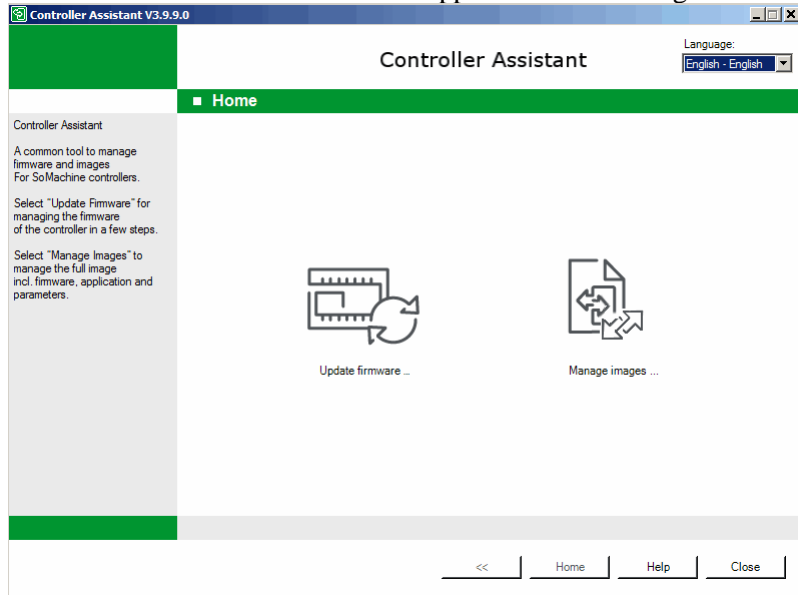
Click on the link in the message above to launch the **Controller Assistant**. This is also available from SoMachine Central.



or



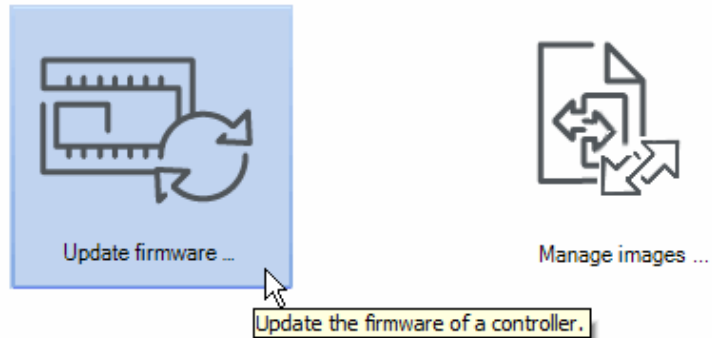
The Controller Assistant Screen appears. There are a couple of options available. This tutorial covers the most direct approach to reflashing a controller.



## How to Reflash a Controller (cont.)

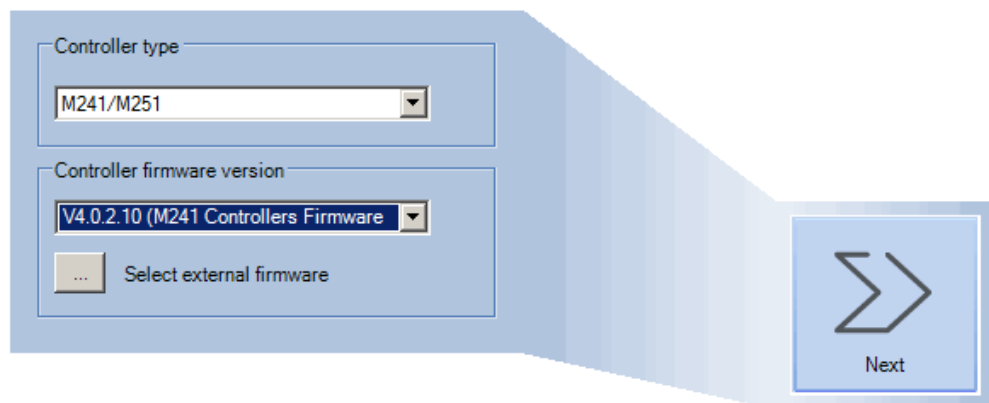
---

Select the **Update firmware** option. This is the path for downloading a new firmware directly to a controller using either an Ethernet or USB connection to the



controller. t

Browse for the controller model and firmware version.

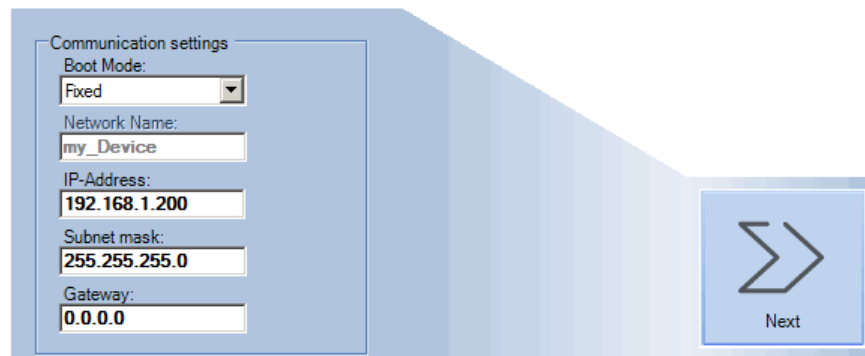


## How to Reflash a Controller (cont.)

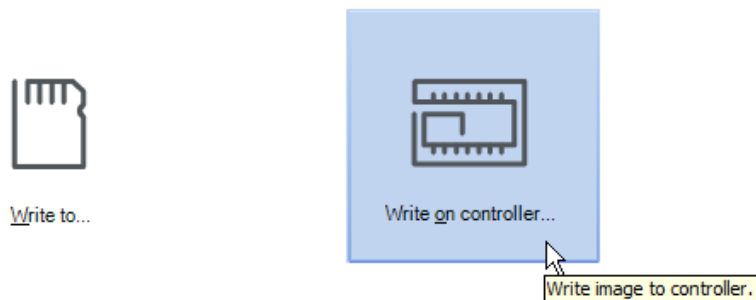
---

Enter an Ethernet address for the controller. This will create a **Post Configuration** file on the controller which you may want to delete later if it conflicts with your applications IP address. Entering an IP address here gives the controller an initial IP address of your choosing. If this is the first time anything has been downloaded to the controller, the controllers Ethernet address is based on the controllers MAC address and will be visible in a few steps.

Click on **Next** to generate the firmware file. This file will exist on the PC and then be transferred to the controller in a few steps.



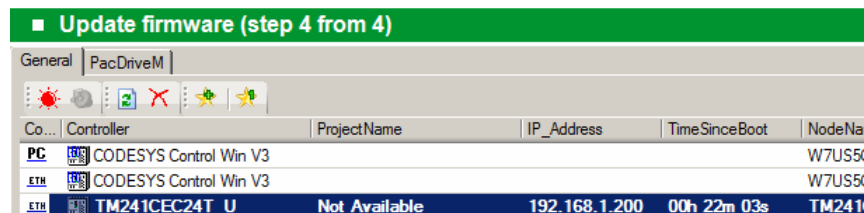
When the screen below appears, select **Write on controller** to initiate the firmware transfer. The other option will write the firmware on a SD memory card with a script that initiates a firmware update upon a power cycle.



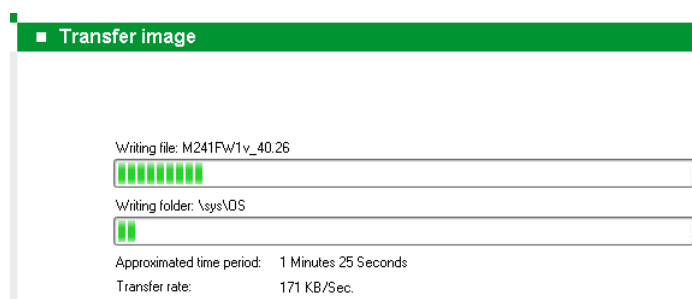
## How to Reflash a Controller (cont.)

---

SoMachine opens the controller selection window. You can download the firmware via USB or Ethernet if its available. If you wish to use Ethernet (faster), select the controller (shown) . Click the **Connect** button. If a USB cable were connected from the PC to the controller, USB would also be an option. If this is an initial update, the controller will have a default IP address based on its MAC address. You can still use this tool to download the firmware with your post configuration file (sets your IP address).



After confirming that you wish to transfer (**Alt + F**) The image starts to transfer to the controller



When the image has completed transferring, the system will indicate the transfer was a success. This process clears out any existing programming in the controller. Power cycle the controller before continuing. Your post configuration IP address should now be in effect and you should be able to ping it.

---

**Note** - do not interrupt this transfer (reflashing) of the controller or you may lose the ability to communicate with it.

---



# Task Configuration

## How to Configure a Task

### ➤ To Configure a Task:

A **POU\_ST** is added to **MAST** Function.

**Priority** - Define the priority of the current task (0 is the highest priority, 31 is the lowest priority).

**Type** - Define the type of task

- **Cyclic** - execute cyclically according to the period defined.
- **Event** - start on rising edge of the variable associated to trigger the event.
- **External Event** - start on rising edge of the trigger input for the event.
- **Freewheeling** - execute at start program. At the end of a cycle run, the task is automatically restarted in a continuous loop, after a delay that is 30% - proportional to the duration of the last task cycle. There is no cycle time defined but T#: 1...1000 ms.

**Watchdog** - Enable the watchdog, enter the watchdog time and the sensitivity. The **Sensitivity** field defines the number of times a watchdog overrun can occur before a watchdog event is generated.

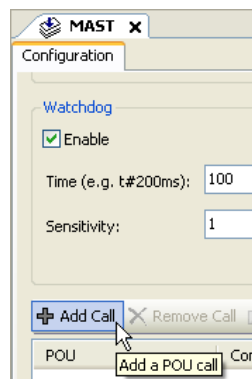
**POU** - Add previously created POU program(s) in the task and fix the execution order of the POU in online mode. Like a segment scheduler in other Schneider Electric PLCs.

## How to Add a POU to a Task

### ➤ To add a POU to a Task:

Double click the **Task** in the **Devices** pane to open the **Configuration** window in the **Work** area.

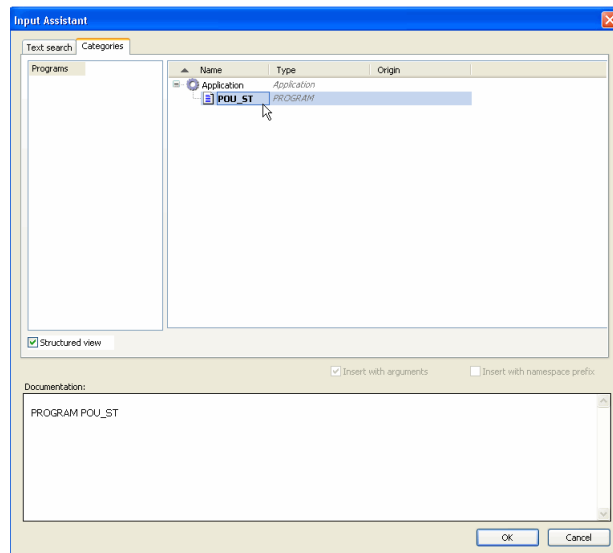
Click **Add POU** in the **POUs** section of the **Configuration** tab.



## Task Configuration (cont)

---

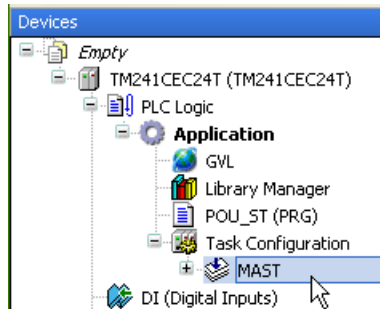
➤ Select the POU.



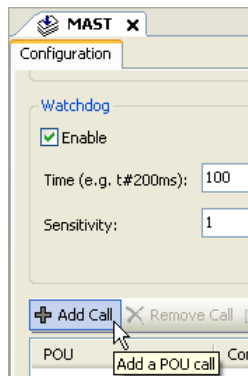
# Exercise - Configure a Task

## 1 Add a POU to the MAST Task.

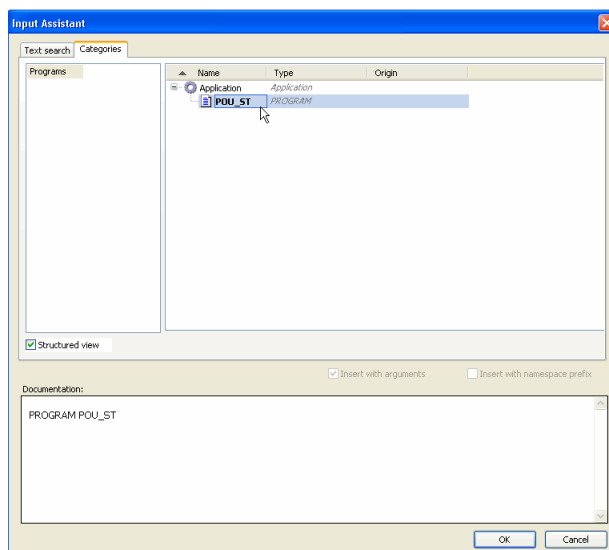
- i. Double click the **MAST** item in the **Devices** pane to open the **MAST Configuration** window in the **Work** area.



- ii. Click **Add Call** in the **POUs** section of the **Configuration** tab. This will open the **Input Assistant**.



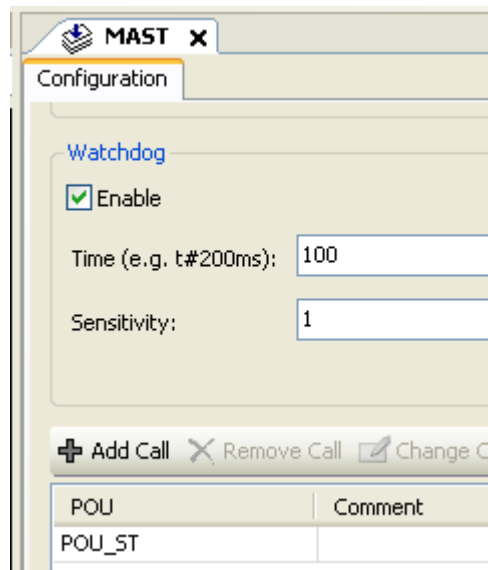
- iii. Expand the branches of **Application** and select the **POU\_ST**. Click **OK**.



## Exercise - Configure a Task (cont.)

---

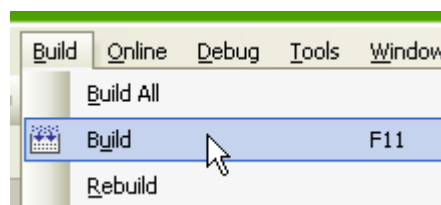
- iv. The **POUs** section of the **MAST Configuration** tab will display the **POU\_ST** item.



- v. Check the lower center of the SoMachine window under the **Messages** section to see that the message **Precompile OK** has appeared. If the Messages window is not visible, open it from the View window. If everything is OK, a green check mark appears (below). If there is an error or warning, open the Message window to view the details.



- vi. Select **Build » Build** from the menu. Build is another name for compiling the application into executable machine code.



---

## 2 Save the project.

---



# PLC Simulator

## Offline PLC Simulator

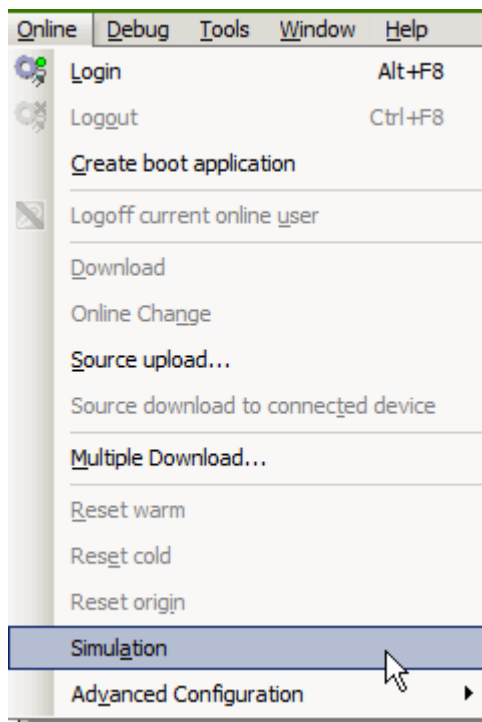
SoMachine provides an **Offline PLC Simulator**. Applications may be downloaded to the simulator and run offline for:

- Program development
- Program debugging

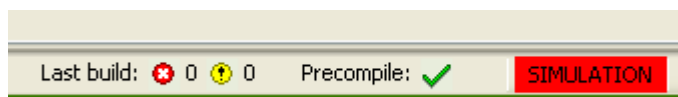
## How to Use the Simulator

- **To start the simulator:**

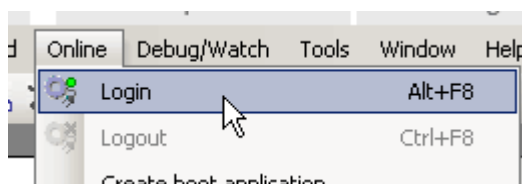
Select **Online » Simulation <PLC Name>** from the main menu to place SoMachine into simulation mode.



When Somachine is in Simulation mode, a red box at the bottom of the window indicates it



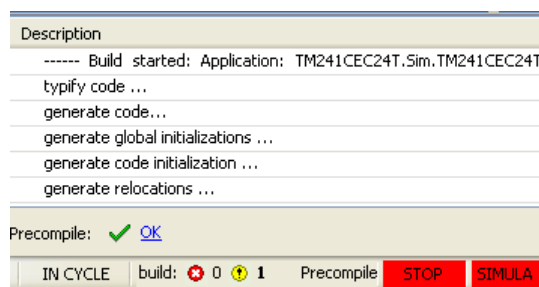
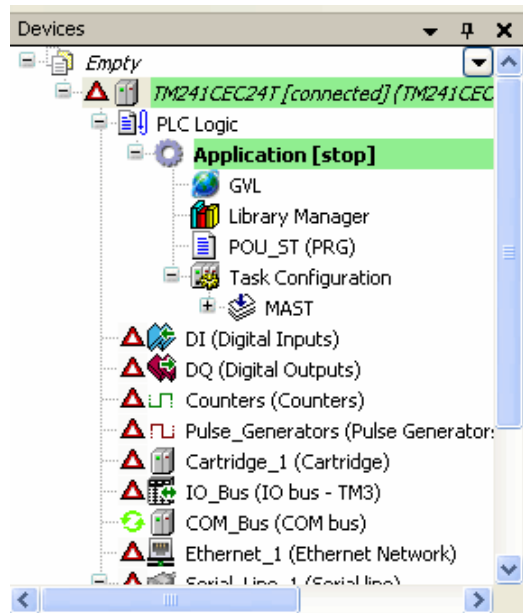
Select **Online » Login** to log in to the **Simulator**. Transfer the application.



## PLC Simulator (cont.)

### How to Use the Simulator (cont.)

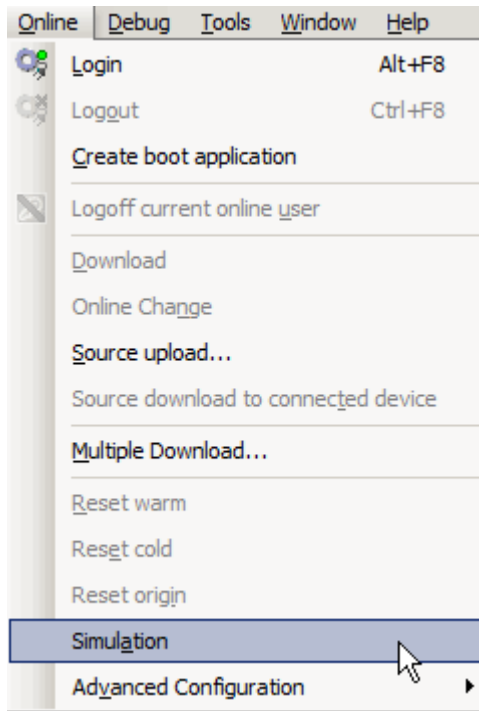
The Simulation mode is indicated in the software by a red rectangle reading **SIMULATION** being displayed in the information line of the dialog box and by the controller in the **Devices** window being displayed in italics. Also the controller and the Application area in the Device tree assume a green background. The Triangle warning icons on the various hardware components of the controller is normal for Simulation mode. This is because in Simulation mode, there is no communications with these objects so the software flags them as having an error.



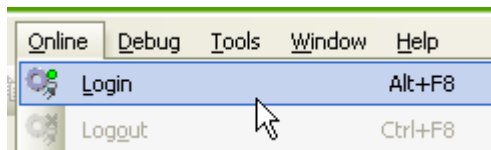
## Exercise - Using Simulation Mode

### 1 Connect the project to the Simulator.

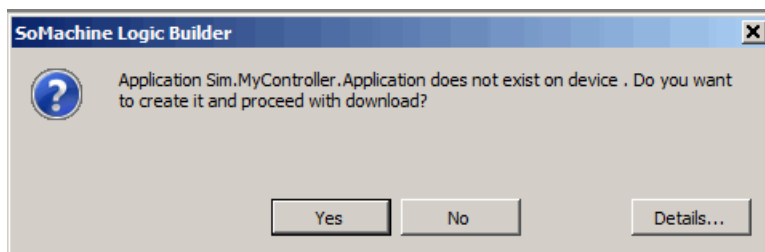
- i. Select **Online » Simulation** from the main menu to place SoMachine into simulation mode.



Select **Online » Login** to log in to the **Simulator**.

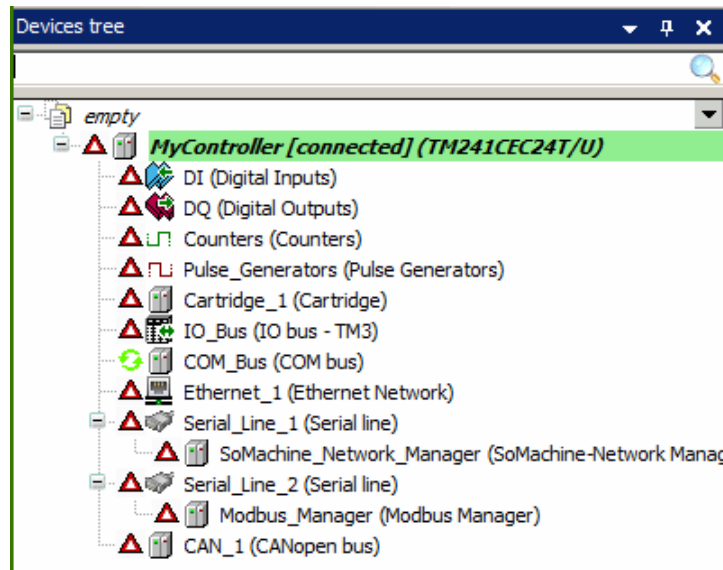


- ii. Since there is no program in the simulator, the following messages will appear. Click **Yes**.for both

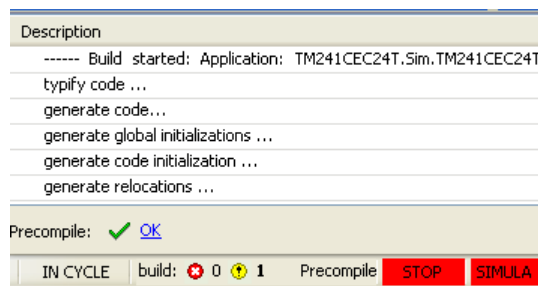


## Exercise - Start the Simulator (cont.)

- iii. The operator interface shows that the program is connected. An M241 is shown in this example.

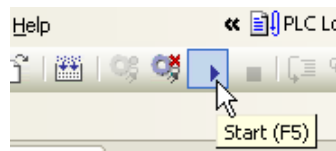


Also shown is that the simulator is currently in STOP mode



## 2 Test the running project

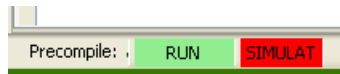
- i. Select **Online » Start** to start the Controller or select the **Start Icon** from the toolbar





## Exercise - Start the Simulator (cont.)

The software indicates the controller is running



- ii. Open the POU and observe its operation. Notice that all the variables currently have a value of zero.

This status only appears if the POU is being executed. An unscheduled POU displays no status at all. This usually means it was not added to the MAST task.

Expression	Type	Value
var1	INT	0
var2	INT	0
var3	INT	0
avg	INT	0

```
1 avg 0 := (var1 0 + var2 0 + var3 0) / 3; RETURN
```

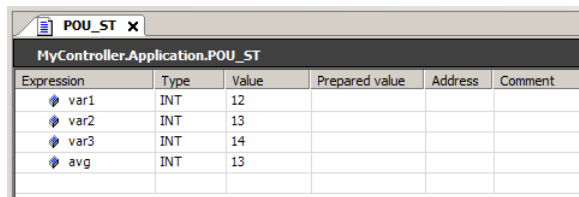
- iii. Expand the local variable reference area (down arrow) and enter some values into the **Prepared value** field for **var1**, **var2**, **var3** as shown. The Prepared Value field is a queue for the variables. The values are not currently applied to the variables.

Expression	Type	Value	Prepared value	Address	Comment
var1	INT	0	12		
var2	INT	0	13		
var3	INT	0	14		
avg	INT	0			

## Exercise - Start the Simulator (cont.)

---

- iv. Press **<CTRL> + F7** on the keyboard or **Online » Write Values** from the menu to inject the prepared values into the variables.

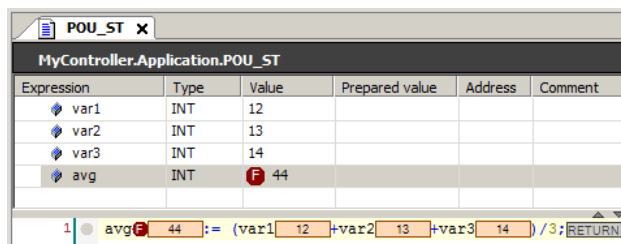


The screenshot shows a window titled 'POU\_ST x' with a sub-header 'MyController.Application.POU\_ST'. It contains a table with the following data:

Expression	Type	Value	Prepared value	Address	Comment
var1	INT	12			
var2	INT	13			
var3	INT	14			
avg	INT	13			

The values are written into the variables and the average is calculated.

- v. Add a value into the prepared value field for the avg variable and press **<F7>**. Notice the difference. This is a **Force** instead of a write and the letter **"F"** is displayed.



The screenshot shows the same window as above, but the 'avg' row is highlighted. The 'Prepared value' field for 'avg' now contains '44' with a red 'F' icon next to it. Below the table, a code editor shows the following expression: `1 avg 44 := (var1 12 + var2 13 + var3 14) / 3; RETURN`

The prepared value is forced into the avg variable. Forcing takes a variable offline and injects your value into it. The expression result is overwritten. This can be done on any variable type

- vi. With the avg variable selected as shown above, press **ALT + F7** to release the force and put the avg variable back online, under the control of logic

---

### 3 Log out of the application and save the project

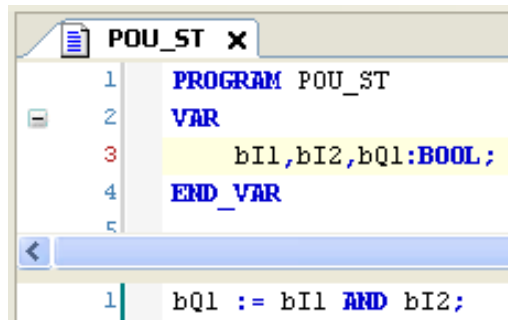
---



# CoDeSys Program Languages

## POU ST

**ST (Structured text)** is a textual language similar to other higher level computer programming languages. It's strengths are in data manipulation, relational expressions conditional execution of code, calculations. ST is one of the recommended languages for applications.



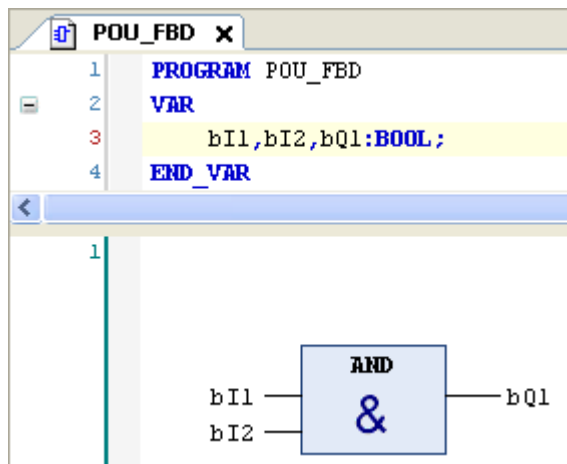
```
1 PROGRAM POU_ST
2 VAR
3   bI1,bI2,bQ1:BOOL;
4 END_VAR
5
6 1 | bQ1 := bI1 AND bI2;
```



**Note:** Multiple variable declarations of same data type is possible as shown above

## POU FBD

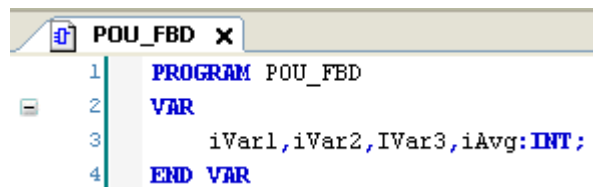
**FBD (Function block diagram)** is a network oriented, graphical language. The components are placed in **Networks** (i.e., Row) so there is no free placement. This language is easily identified by the Row numbers to the left and the direct placement of variables on the pins of a block. At runtime, code is solved left to right, by rows.



## Exercise - Program a FBD POU

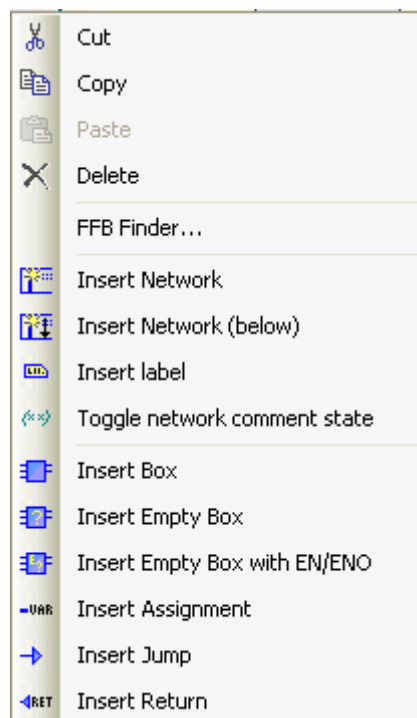
### 1 Create a program using FBD language to calculate the average of three variables.

- i. Return to the **Devices** pane and add a new **POU** program in the **Function Block Diagram (FBD)** language. Name the new POU **POU\_FBD**.
- ii. Create the same four **INT** variables that were used in Exercise - Create a POU.



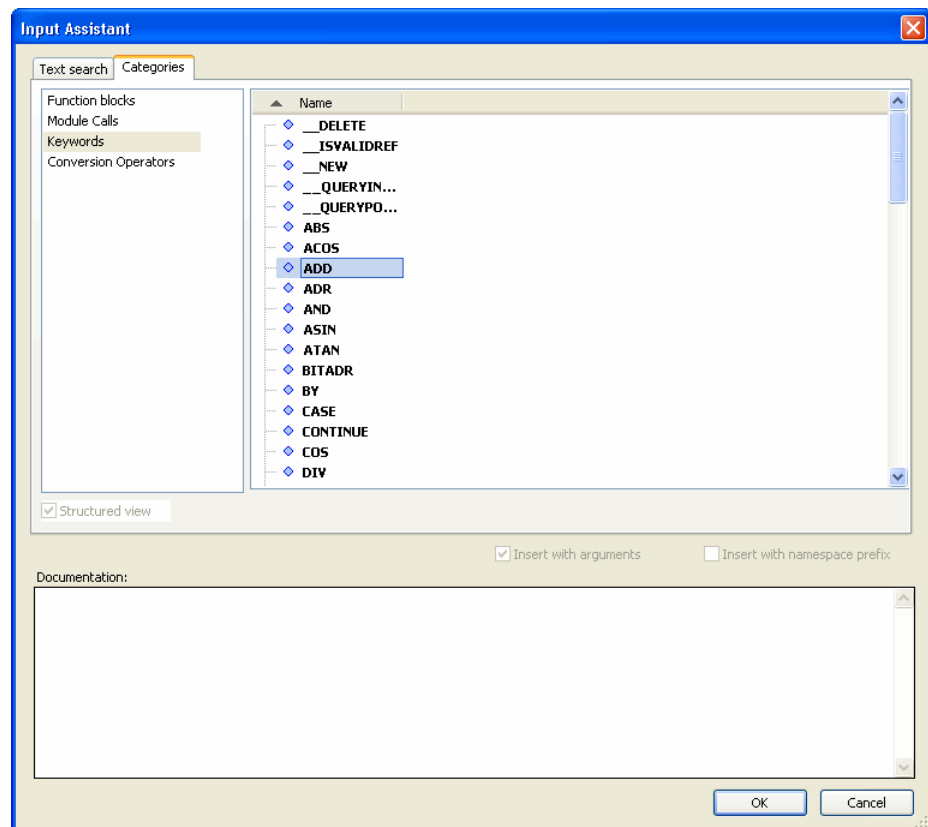
```
1 PROGRAM POU_FBD
2 VAR
3   iVar1,iVar2,IVar3,iAvg:INT;
4 END_VAR
```

- iii. Right click to the left of **Network 1** in the lower pane of the **Work** area and select **Insert Box** from the menu.

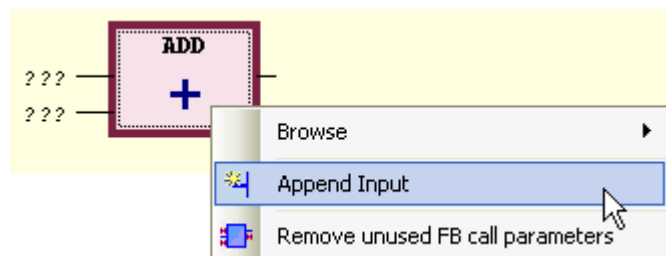


## Exercise - Program a FBD POU (cont.)

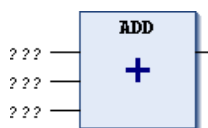
- iv. When the **Input Assistant** opens select **Keywords** from the **Categories:** pane then select **ADD** from the **Items:** pane. Click **OK**.




- v. The **ADD** operator will be inserted into the editor. Right click the **ADD** block and select **Append Input** from the menu.

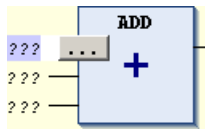


A third input will be added to the **ADD** operator.

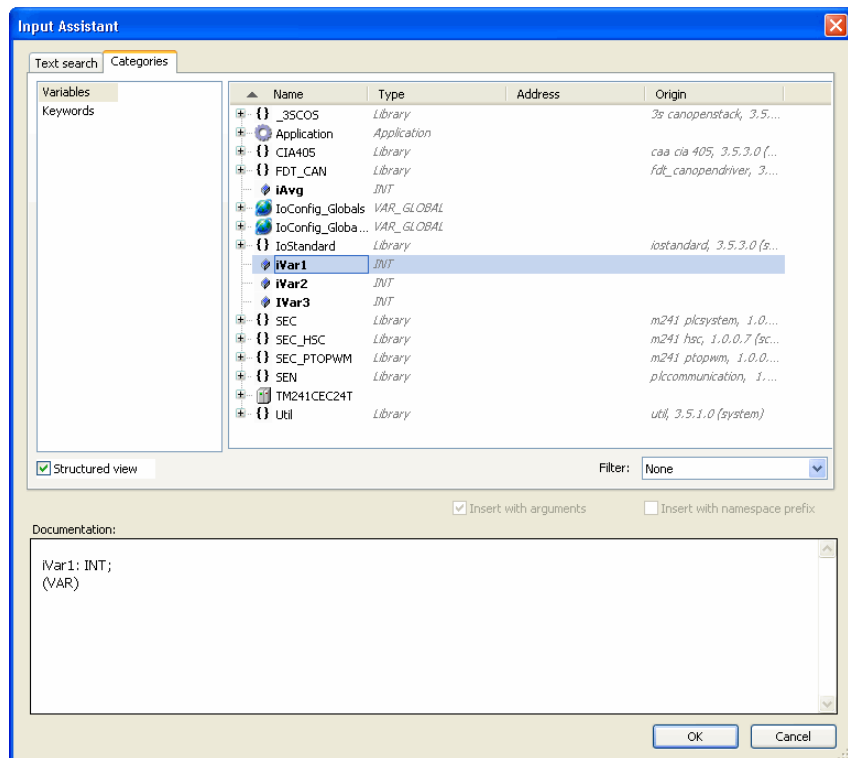


## Exercise - Program a FBD POU (cont.)

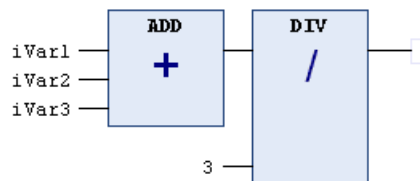
- vi. Click to select the first **Input** then click the **Ellipsis**  button.



- vii. This will open the **Input Assistant**. With the variables option selected, select **iVar1** then click **OK**. In the same manner add variables **iVar2**. Click on the third input and start typing **iVar3**. Notice the system opens a window with variables spelled like **iVxxxx**. Select **iVar3** from the presented choices.

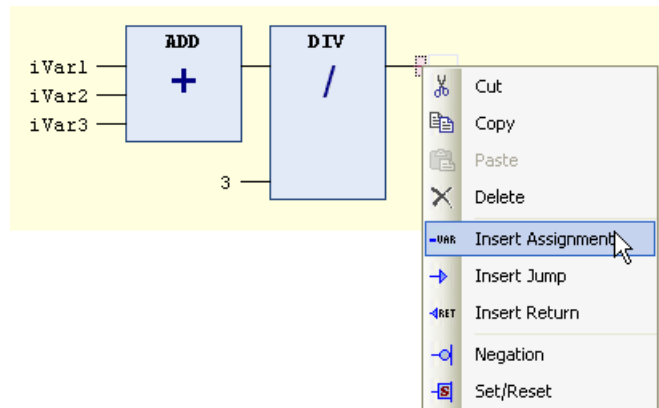


- viii. **Right click** the **Output** pin and select **Insert Box** to open the **Input Assistant**. Select **Keywords** from the **Categories:** pane then select **DIV** from the **Items:** pane. Click **OK**. Add **3** to the lower input pin (denominator).

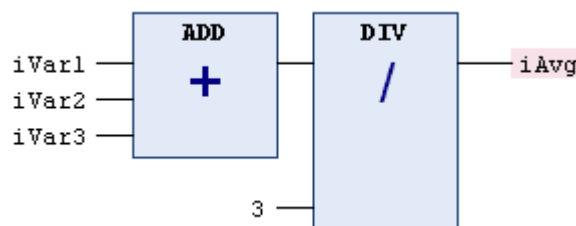


## Exercise - Program a FBD POU (cont.)

- ix. Right click the **Output** pin of the **DIV** block and select **Insert Assignment**.



- x. Use the **Input Assistant** to add the **avg** variable. The completed program will look like this:

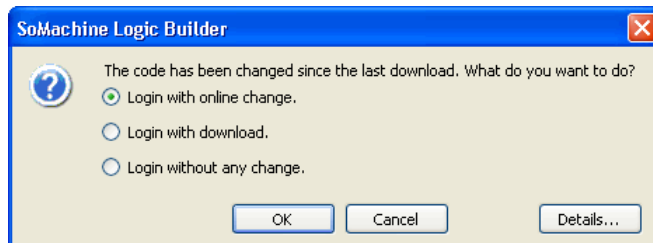



## Exercise - Program a FBD POU (cont.)

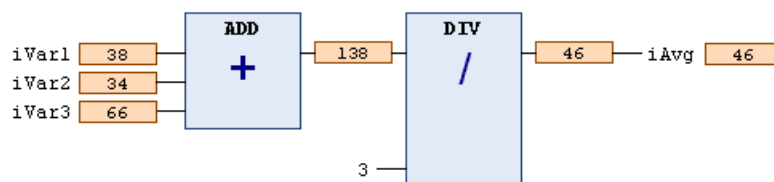
---

### 2 Test the program.

- i. Add the new **POU** to the **MAST** task.
- ii. Select **Online » Login** from the main menu. Since the code has changed SoMachine will prompt for the type of login. Select **Login with online change** then click **OK**.



- iii. Click the **Start Application**  button to start the program.
- iv. Enter values into the variables in the same manner as before. Check the display to see whether the program is functioning correctly.



- v. Select **Online » Logout** to log out of the program.

---

### 3 Log out of the application and save the project.

---

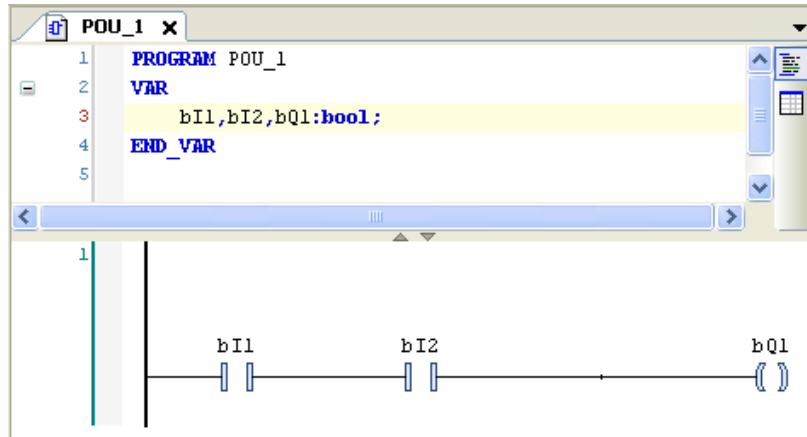




## CoDeSys Program Languages (cont.)

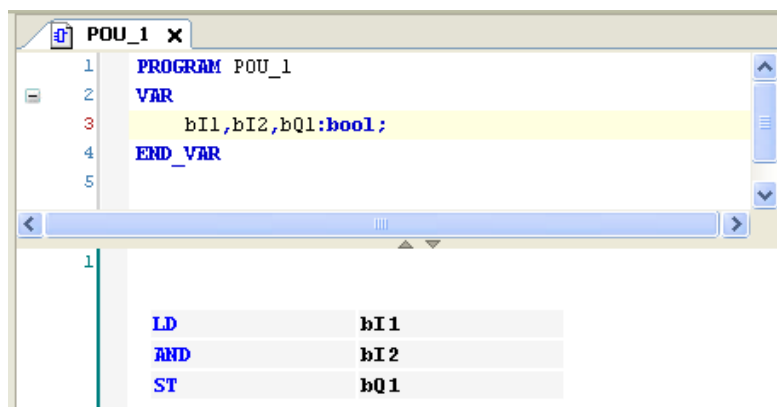
### POU LD

**LD (Ladder Diagram)** enables the programmer to virtually combine relay contacts and coils. The strengths of LD are in complex switching applications. This is a recommended programming language.



### POU IL

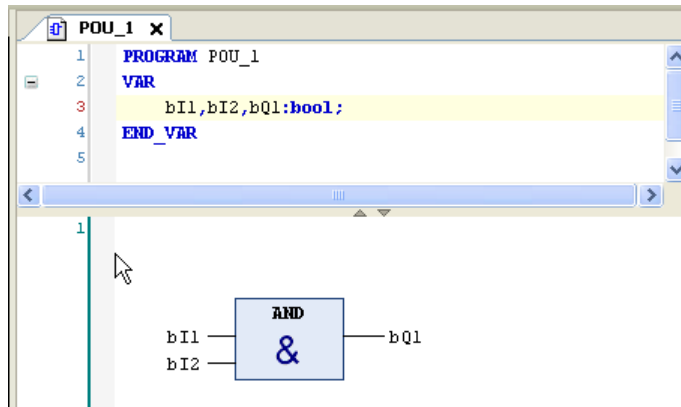
**IL (Instruction list)** is an Assembler like programming language. This language is obsolete for new applications and is available mostly to support existing applications. In general, this is not used for new application development.



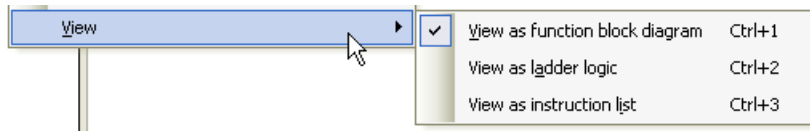
# Exercise - Convert IL to LD

## 1 Convert a FBD POU to Ladder Diagram (LD) language.

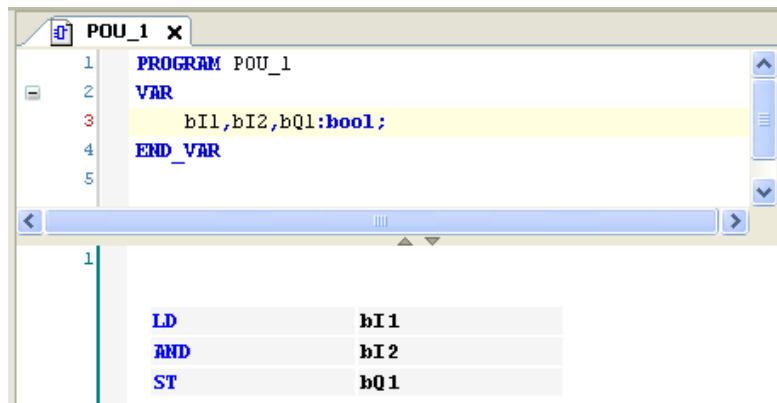
- i. Start a new program POU in the FBD language. Name it anything you like
- ii. Create the program shown below



- iii. Select **FBD/LD/IL » View » View as ladder logic** to change the POU from FBD language to Ladder Diagram language. LD and FBD look very similar where there are only blocks involved, the changes are minor.



- iv. Select **FBD/LD/IL » View » View as instruction list** to change the POU to Instruction List language.



### Note:

FBD, LD and IL formats can be converted from one format to another. SFC, ST and CFC cannot be converted

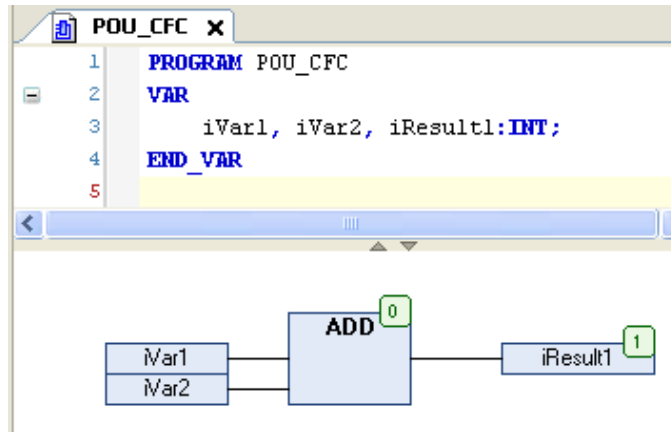
## 2 Log out of the application and save the project.



## CoDeSys Program Languages (cont.)

### POU CFC

CFC (Continuous Function Chart) is similar to FBD. Unlike FBD it allows the free placement of components. Since components may be placed anywhere, the order of execution is determined by the order components are added to a POU and the order is indicated by the little boxes to the right side of the object. If the order is not correct, it can be edited by the user. CFC is one of the recommended languages for applications. Its strengths are free component placement, easy control over execution order, graphic boxes for complex functions.

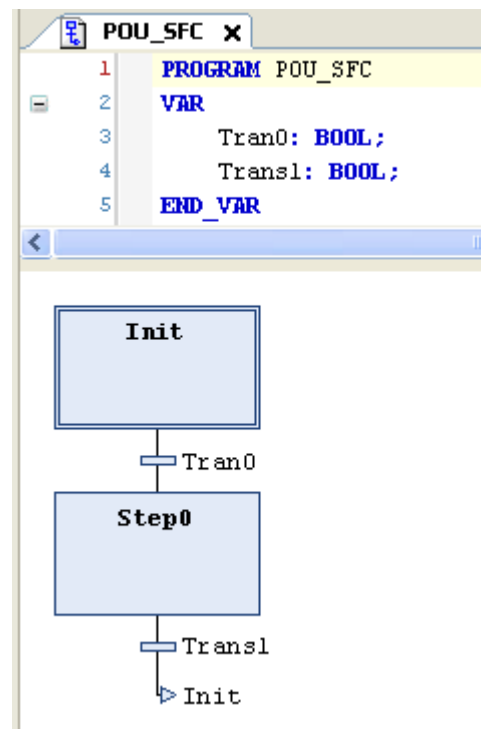


**Note-** Local variables are declared in the "VAR" section, just above the ADD block or in the Local Var section of the catalog.

## CoDeSys Program Languages (cont.)

### POU SFC

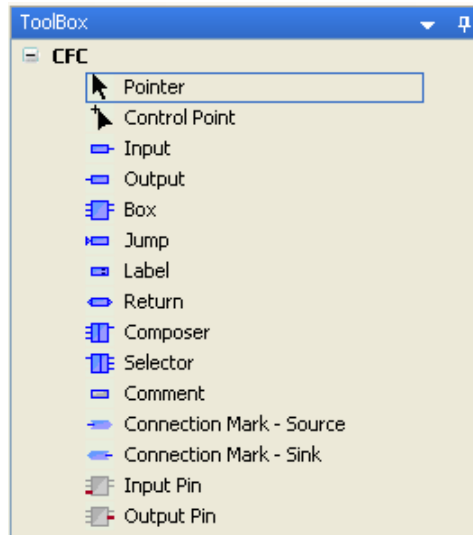
**SFC (Sequential function chart)** is used to program sequential processes. This language consists of steps (large boxes) and transitions. It is like a flow chart with complex code inside the steps. An active step stays active until the downstream transition comes TRUE. The active step stops processing and the next step starts processing. SFC is useful in very sequential applications, e.g., an application that always goes through the exact same sequence. The strengths of this language are many. It's easy to see exactly where an application is in its sequence by looking at the SFC structure. Less interlocking coding is needed because inactive steps are not even solved. Troubleshooting is isolated to the active step(s) and monitoring the transition conditions



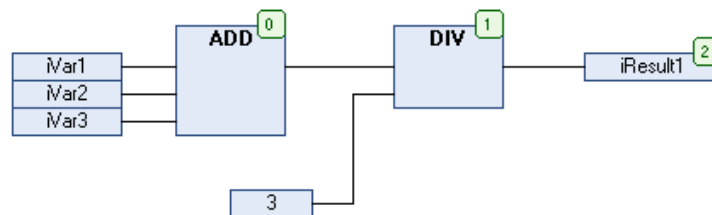
## Exercise - Program a CFC POU

### 1 Create a program using CFC language to calculate the average of three variables.

- i. Create a new POU program named **POU\_CFC** using the **CFC** language. Create the same program to calculate the average of three variables.
- ii. The CFC language allows free placement of the components. The **Inputs**, **Outputs** and **Boxes** must be added to the work area by using the CFC Toolbox located to the right of the **Work** area.



- iii. Create this POU by following the program below. To add a new pin on the ADD block, drag & drop the **input pin** component. Ask the instructor if additional help is needed.



- iv. **Save**, add to **MAST task**, **Login**, **download** and **test** this POU in the Simulator.

### 2 Log out of the application and save the project.



# Watchdog Mechanisms

## M238 Watchdog Operation

A **Watchdog** is a function that responds to specific software or hardware resource overloads that may cause the system to stop responding.

There are two types of **Watchdog** in SoMachine:

- **Application (configured) watchdog** - Each task cycle can be monitored by a watchdog timer (a maximum duration of the task cycle). This helps debugging certain application conditions (such as infinite loops, etc.) and provides a maximal duration for refreshing outputs.

A watchdog can be defined for each task.

- **System Watchdog** - System Overload is reached when all the user tasks use more than 80% of system resources. This is computed on a 1 second window, each second.

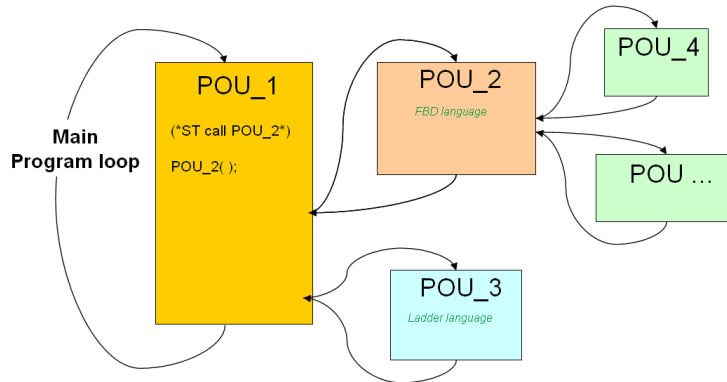
This system overload mechanism cannot be disabled, so that system tasks can be executed properly.

Task	Status	IEC-Cycle Count	Cycle Count	Last Cycle Time (µs)	Average Cycle Time (µs)	Max. Cycle Time (µs)
MAST	Valid	21051	24716	14	15	457

- In **Online** mode, task processing can be monitored by double clicking on the **Task Configuration** from the **Applications Tree**.
  - Monitoring the task shows how long the task is taking to execute
  - Monitoring the actual task execution time helps determine the correct WDT setting

# Structuring an Application

## POU Program Structure



A POU can call other POUs (nesting)

- No limits to number of calls
- Could affect Watchdog limits if not taken into account

Program structuring is possible

- Conditional execution may be added
  - Watchdog issues are possible
- Called POUs can be in any language

POU calling in the application program

	POU Function	POU Function Block	POU Program
<b>Example</b>	Function Fun1:INT 3 Inputs (INT): A, B, C	Function_Block FunBlck1 3 Inputs (INT): A, B, C 2 Outputs (INT): D, E Instance1: FunBlck1	Program Prgr1 3 Inputs (INT): A, B, C 2 Outputs (INT): D, E
<b>List</b>	LD 5 Fun1 3,2 ST Result	CAL Instance1(A:=5, B:=3, C:=2) ... LD Instance1.D ST Result1 LD Instance1.E ST Result2	CAL Prgr1(a := 5, b := 3, c := 2) ... LD Prgr1.D ST Result1 LD Prgr1.E ST Result2
<b>Structured text</b>	Result:=Fun1(5,3,2); or Result:=Fun1(A:=5,B:=3,C:=2);	Instance1(A:=5, B:=3, C:=2, D => Result1, E => Result2); or Result1:=Instance1.D; ...	Prgr1(A := 5, B := 3, C := 2, D => Result1, E => Result2); or Result1:= Prgr1.D; ...
<b>Ladder or FBD or CFC</b>			

# The POU Function

## A POU that Returns a Result

**POU Function** – A POU that returns a result

- Normally used when you need to repeat the same calculation with different vars. Number crunching
- Returns the result of the calculation to the calling POU
- Arguments are passed to the function, returns a single result

## Function Creation

**Example -**

```
1  FUNCTION CalcAvg :INT  (*function name, return data type*)
2  VAR_INPUT              (*Inputs to the function*)
3      var1:INT;          (*var name, Data type*)
4      var2:INT;
5      var3:INT;
6  END_VAR                (*end of input variable declarations*)
7  VAR
8  END_VAR                (*local var declaration area*)
9
10
11 CalcAvg := (var1 + var2 + var3) /3;
12 (*function, function name also contains result of function*)
```

Function **CalcAvg** is created in ST language.

## Calling Function

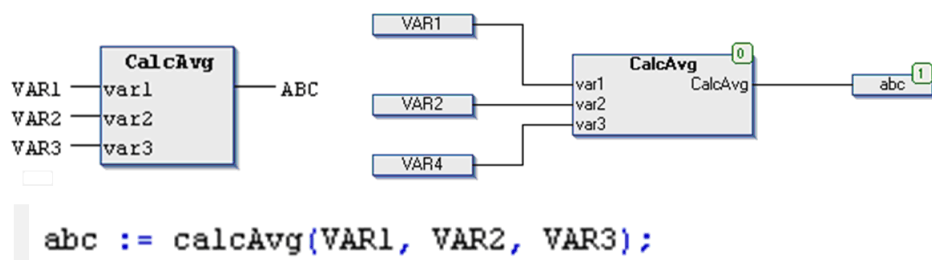
Function **CalcAvg** called from ST POU and POU running

```
abc := CalcAvg(10, 20, 15);
abc 15 := CalcAvg(10, 20, 15);
```

Values **10, 20, 15** are passed to function CalcAvg. A result of 15 is returned.

## POU Function in Three Languages

The same POU Function is shown in three different programming languages

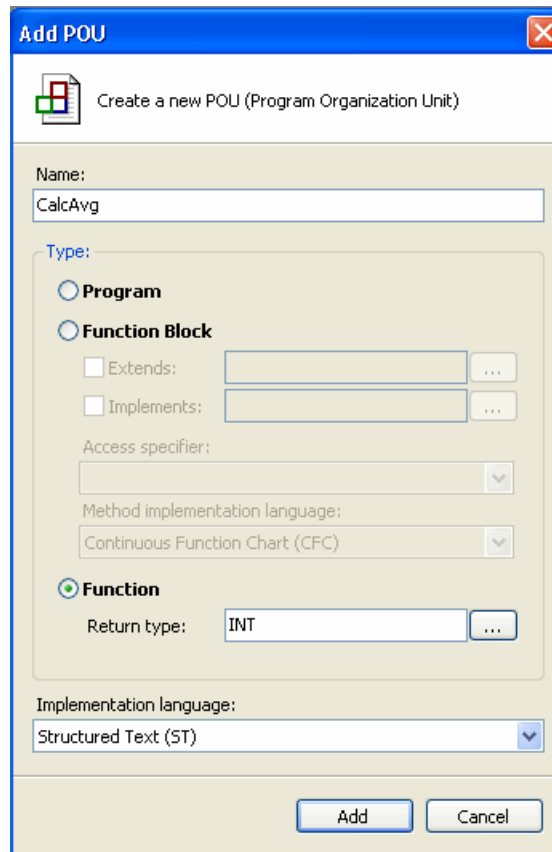




## Exercise - POU Function

### 1 Create a POU Function and add it to a POU Program.

- i. Right click the **Application** node and select **Add Object » POU**.
- ii. Name the function **CalcAvg**. Select **Structured Text** as the creation language for this function and return data type of INT



The screenshot shows the 'Add POU' dialog box with the following configuration:

- Name:** CalcAvg
- Type:** Function (selected)
- Extends:** (empty)
- Implements:** (empty)
- Access specifier:** (empty)
- Method implementation language:** Continuous Function Chart (CFC)
- Return type:** INT
- Implementation language:** Structured Text (ST)

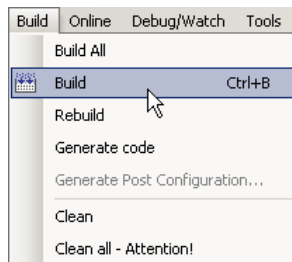
Buttons: Add, Cancel

## Exercise - POU Function (cont.)

- iii. Create the function as shown. The function returns an **Integer** and has three input variables. The comments do not need to be added to the function. Save the function.

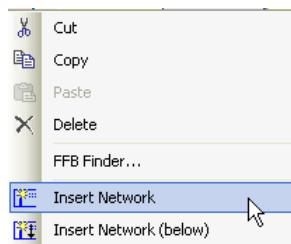
```
1 FUNCTION CalcAvg :INT (*function name, return data type*)
2 VAR_INPUT (*Inputs to the function*)
3   var1:INT; (*var name, Data type*)
4   var2:INT;
5   var3:INT;
6 END_VAR (*end of input variable declarations*)
7 VAR
8 END_VAR (*local var declaration area*)
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

- iv. **Save and Build** the application. Fix any errors that appear.

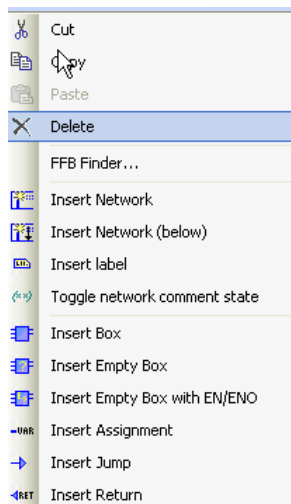


## 2 The next steps will call the function from a new program POU. Create a new POU program call the function

- i. Open the **POU\_FBD** program. Right click the **left margin**, and select **Insert Network** from the menu.

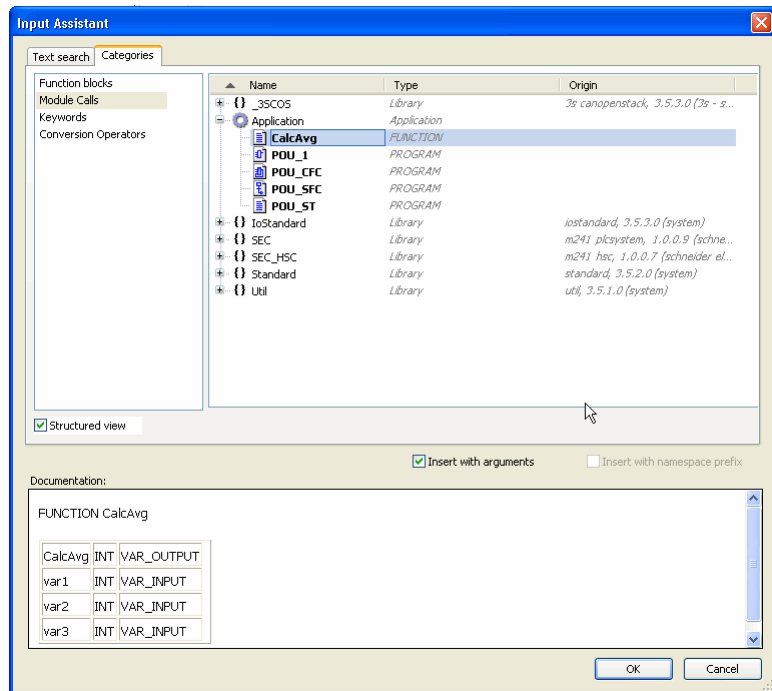


- ii. Right click the left margin and add a box to the network that was just created.

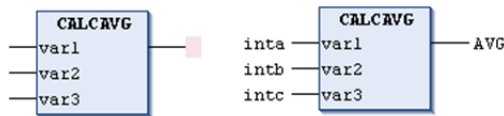


## Exercise - POU Function (cont.)

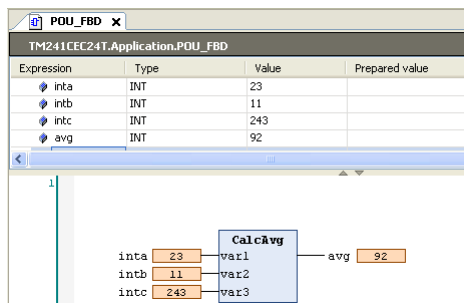
- iii. When the **Input Assistant** opens select the item **Module Calls** in the **Categories** pane. Expand the **Application** branch and select the function **CalcAvg**. Click **OK**.



- iv. Exercise - POU Function (cont.)The **CalcAvg** function is added. **Add** the **vars** shown



- v. **Save, login** and **download** this application to your controller. Test the operation by entering values into the **Prepared Value** fields and using the **CNTRL + F7** keys to enter values into the Value field.

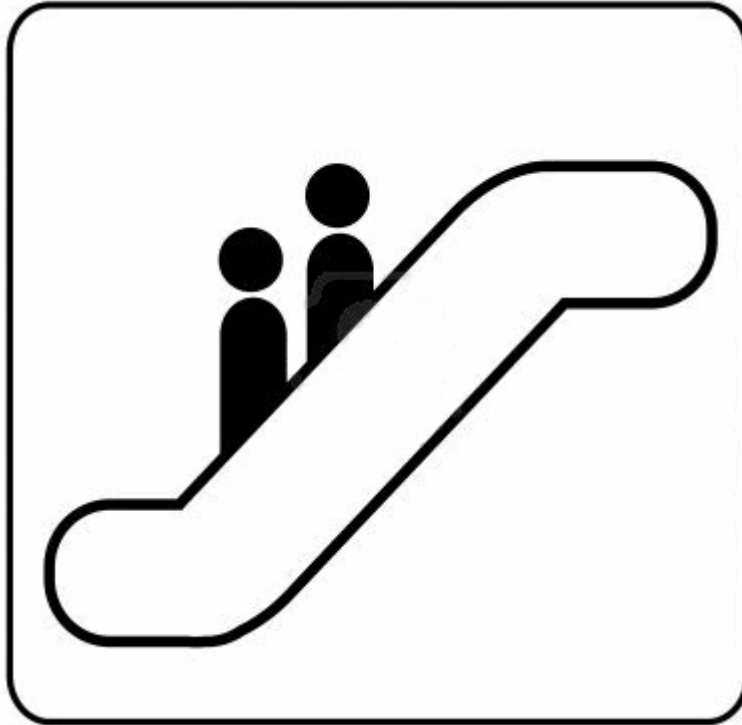


## Sample Project

---

### Escalator Project

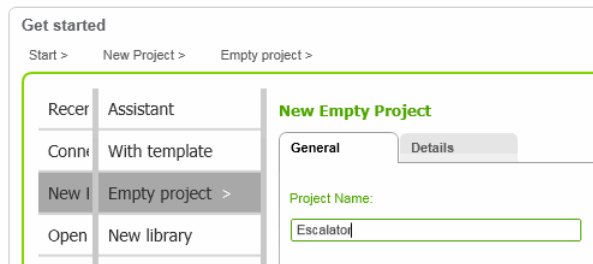
The Escalator project is a simple project that is designed to start and run an escalator. This project will be used to consolidate all of the features of SoMachine that have been demonstrated in this tutorial.



# Exercise - Start the Escalator Project

## 1 Create a new Standard Project.

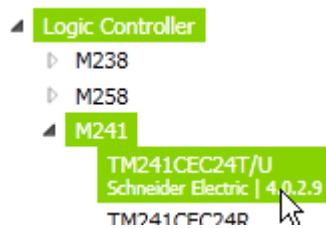
- i. At the **Home** screen click the **New Project** option.
- ii. Click the **Empty Project**. Name the project **Escalator** as shown below. Click **Create Project**.



- iii. Select the **Configuration** Box then click on **Manage Devices**

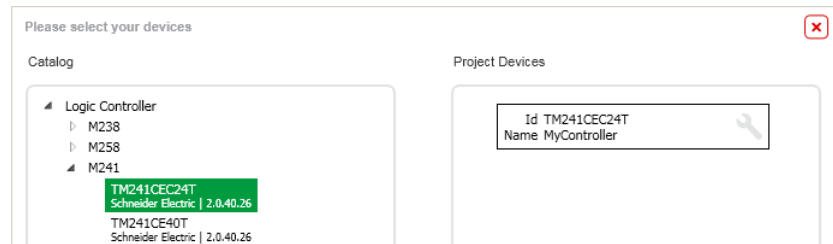


- iv. Select the **M241 Logic Controller** for the project. It really doesn't matter which controller is selected since you will be using the simulation mode to test



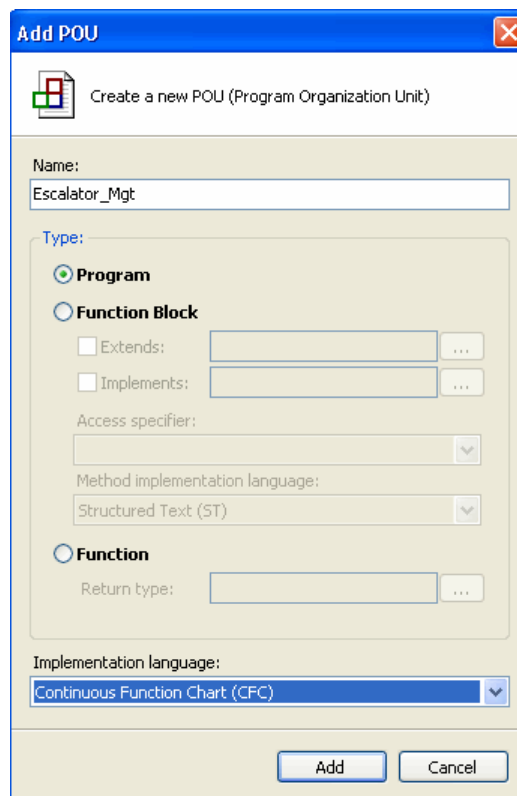
## Exercise - Start the Escalator Project (cont.)

- v. Use the right arrow to finish the selection process. Your project appears as shown.



### 2 Create a CFC POU

- i. Open the Logic Builder and from the **Applications Tree**, create a **CFC POU** named **Escalator\_Mgt**



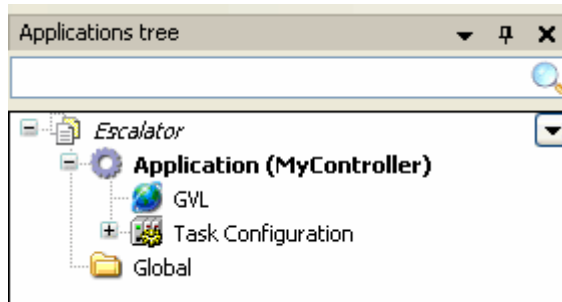
- ii. Save the **Escalator** project.



# Global Variables

## Global Variable List (GVL)

A **Global Variable List (GVL)** is a list of variables that are available to all parts of the application. The fact that all POU's as well as other sections of the application, have access to these variables is what makes them global. A maximum of three GVL lists may be created per application



### Note:

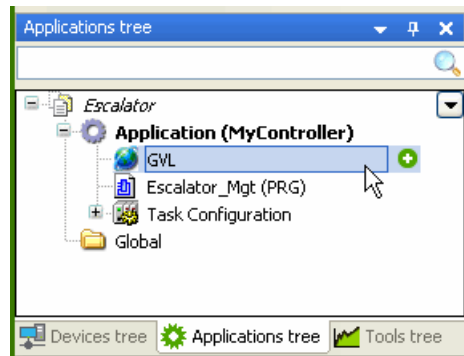
**Global Variables** are declared in the same manner as local variables but are located in a GVL file.

```
GVL x
1  VAR_GLOBAL
2  // Operating Mode
3  bInit:BOOL;
4  bRun:BOOL;
5  bStop:BOOL;
6  bManual:BOOL
7
8  // Reference Speed
9  iSpeedRef:INT;
10
11 // Operator Interface
12 bPB_Run:BOOL;
13 bPB_Stop:BOOL;
14 bMotor_Run:BOOL;
15 bMotor_Stop:BOOL;
16 bMotor_Fault:BOOL;
17 dStatus:DINT;
18 iTab2_4[1..4,1..4] OF INT:=1,2,3,4,5,6,7,8]; //array 2 dimensions
19 END_VAR
```

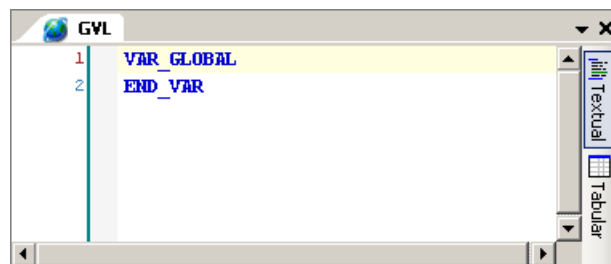
# Exercise - Add Global Variables

## 1 Add Global Variables to the Escalator project.

- i. Open the **Program** screen to view the **Applications Tree** pane.



- ii. Double click the **GVL** item in the **Devices** tree. The **GVL Tab** will open in the right pane.



- iii. Add these variables to the project.

### **VAR\_GLOBAL**

```
// these global variables will be physical inputs and outputs of the PLC
LI_DOWN_TO_UP: BOOL; // operator switch to indicate which direction the escalator should run
LI_EMER_STOP_1: BOOL; // emergency stop located at the end of the escalator
LI_EMER_STOP_2: BOOL; // emergency stop located at the other end of the escalator
LI_ENABLE: BOOL; // enabling switch to switch on the escalator
LI_MAINTENANCE: BOOL; // maintenance switch to switch the escalator in maintenance mode.
// maintenance mode allows change of direction and normal speed.
DO_RUN_FORWARD: BOOL; // physical output of the PLC linked to the RUN forward command of the Drive
DO_RUN_REVERSE: BOOL; // physical output of the PLC linked to the RUN reverse command of the Drive
AO_SPEEDREF: INT; // physical analog output of the PLC linked to the speed reference command of the Drive
AI_POTENTIOMETER: INT; // physical analog input of the PLC linked to the operator potentiometer
//to set the appropriate speed of the escalator
```

### **END\_VAR**

## 2 Save the project.





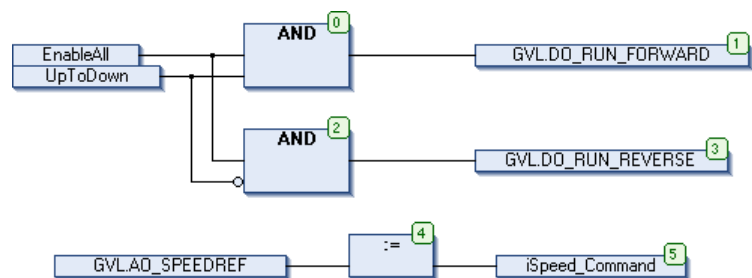
# Exercise - Add a POU to the Escalator Project

## 1 Add a POU to the project.

- i. The **Escalator Project** already contains a blank POU named **Escalator\_Mgt (PRG)**. Double click the POU to open for editing.
- ii. Add these **Local Variables** to the top pane of the POU.

```
PROGRAM Escalator_Mgt
VAR
EnableAll: BOOL; // activate the escalator
UpToDown: BOOL; // current direction
iSpeed_Command: INT; // current selected speed
END_VAR
```

- iii. Add the CFC Diagram to the lower pane.



- iv. Notice that the **AND** function block has a little circle on the lower left pin.



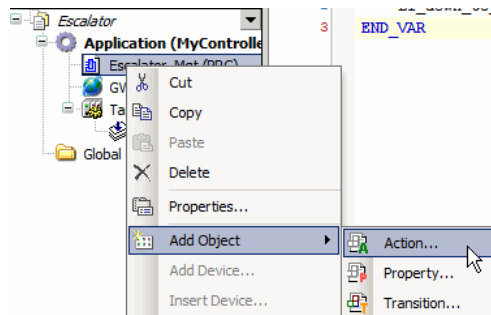
This is a **Negated** pin. To negate the pin right click the pin and select **Negate** from the menu.

- v. Add the **Escalator\_Mgt** POU to the **MAST** task.
- vi. **Save** and **build** the project.
- vii. Run the project in **Simulation mode** to test the logic.

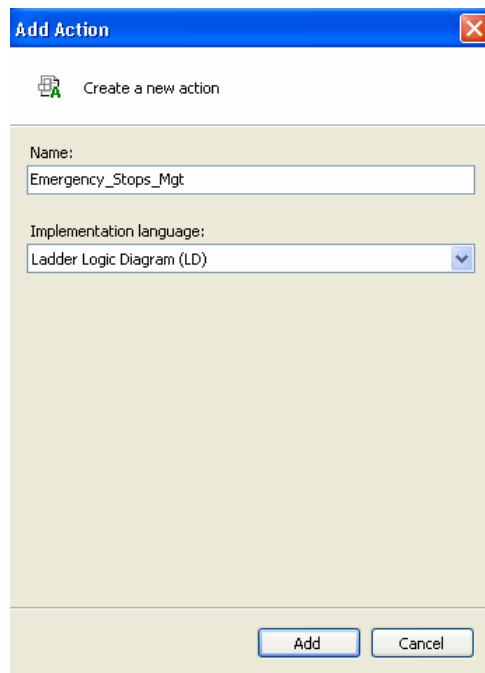
## Exercise - The Escalator Project (cont.)

### 2 Add an Action (sub program) using Ladder Logic Diagram for the Emergency Management.

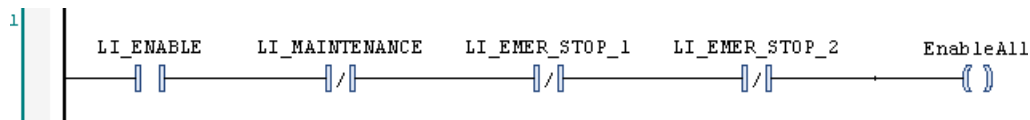
- i. Right click the **Escalator\_Mgt** POU and select **Add Object » Action...** from the menu.



- ii. When the **Add Action** dialog appears name the Action **Emergency\_Stops\_Mgt** and select **Ladder Logic Diagram** as the Implementation language. Click **Open**.



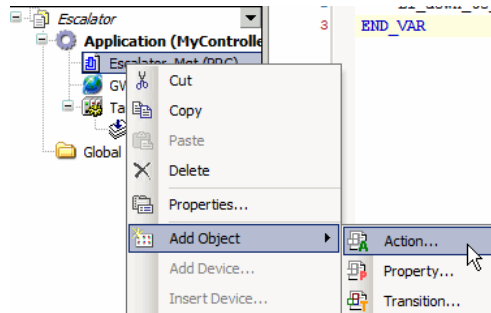
- iii. Add this ladder logic to the Action.



## Exercise - The Escalator Project (cont.)

### 3 Add an Action using Structured text for the Maintenance.

- i. Right click the **Escalator\_Mgt** POU and select **Add Object » Action...** from the menu.

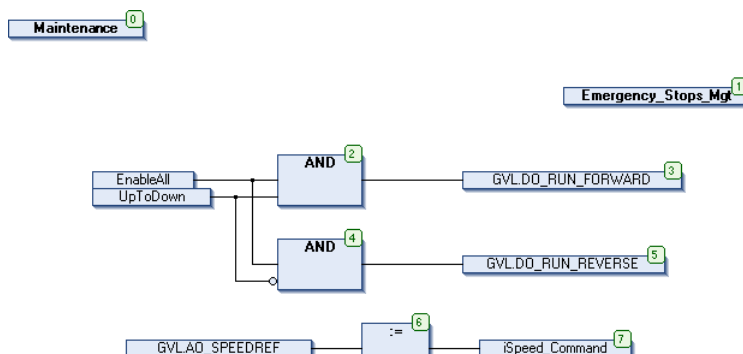


- ii. When the **Add Action** dialog appears name the Action **Maintenance** and select **Structured Text** as the Implementation language. Click **Open**.
- iii. Add this code to the action.

```
IF (LI_MAINTENANCE = TRUE) THEN
  IF (EnableAll = FALSE) THEN
    iSpeed_Command := AI_POTENTIOMETER;
    IF (LI_DOWN_TO_UP = TRUE) THEN
      UpToDown := FALSE;
    ELSE
      UpToDown := TRUE;
    END_IF
  END_IF
END_IF
END_IF
```

### 4 Add the Actions to the POU.

- i. Add boxes to the program to add the two Actions to the program.



- ii. Log in to the program and test using the simulator. Force a value into the **EnableAll** variable then into the **UpToDown** variable. Observe the outputs to see that the escalator is running the correct direction. Modify the **GVLAD\_SPEEDREF** and observe the results. Logout when finished. This completes the Getting Started with SoMachine Self Study



