Electrical and Computer Engineering Department

Introduction to Codesys

First steps...

Version 10.1 (2021/02/16)

1. Introduction

Codesys is a commercial soft PLC solution with wide acceptance in the marketplace. It provides a development environment for IEC 61131-3 programs, as well as runtime execution environments for many execution devices (from embedded platforms like the Raspberry Pi, or Wago IPCs, to personal computers). These runtime environments work as virtual machines that execute the compiled IEC 61131-3 programs, and program execution may be monitored by the development environment.

Codesys is already installed on all PCs of the The Industrial Automation lab at FEUP/DEEC. We suggest you use the exact same version of Codesys on your laptop so as to ease the porting of programs between computers. To download the version of Codesys installed on the Industrial Automation lab, please visit this lab's ftp server:

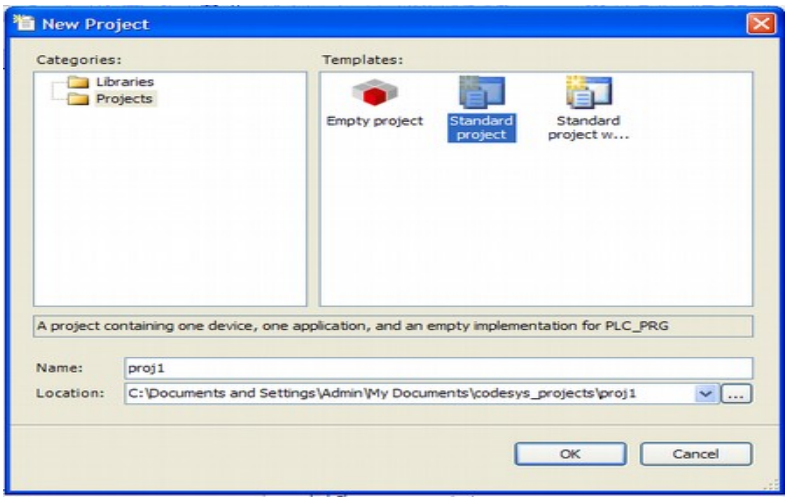
ftp://labs-automacao.fe.up.pt/I005/Demos%20Software/Codesys/V3.5/

Note: In order to access this server you must be connected to the internal FEUP network. Connect to FEUPnet via VPN if necessary.

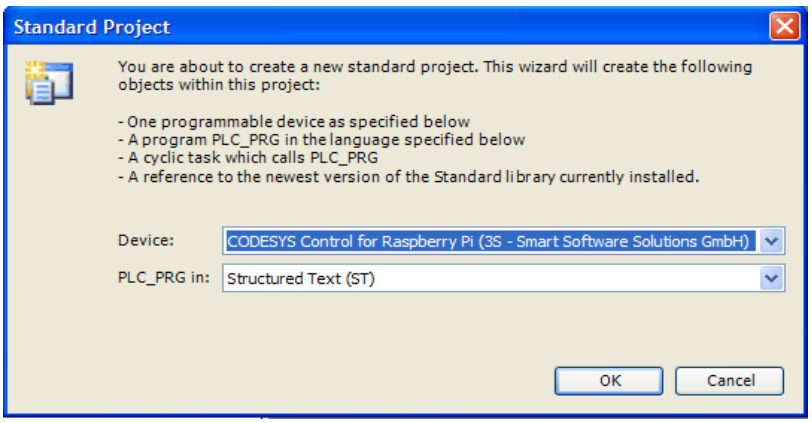
2. Configuring the First Project

Run Codesys v3 engineering environment, and create a new project using the "standard project" template. This will create a project with one default IEC 61131-3 Program (a POU – Program Organization Unit), as well as a configuration with a single task that runs this program. Don't worry, you can change all this later if required.

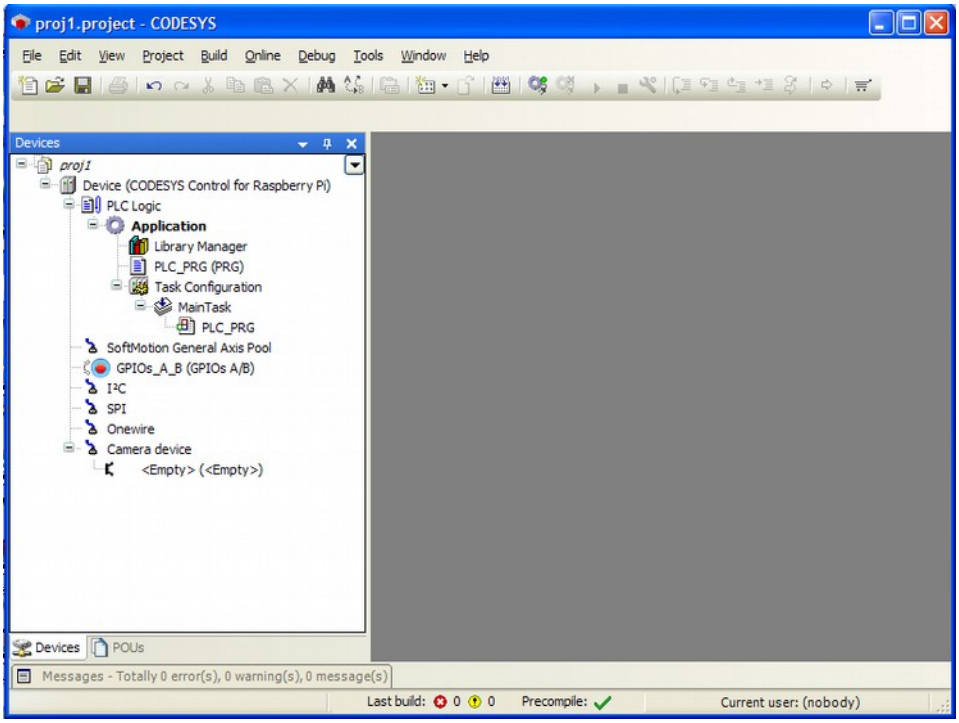
You should also preferably create a directory for storing this project, as Codesys will eventually create several files. Using a specific directory will help keep your files organized.



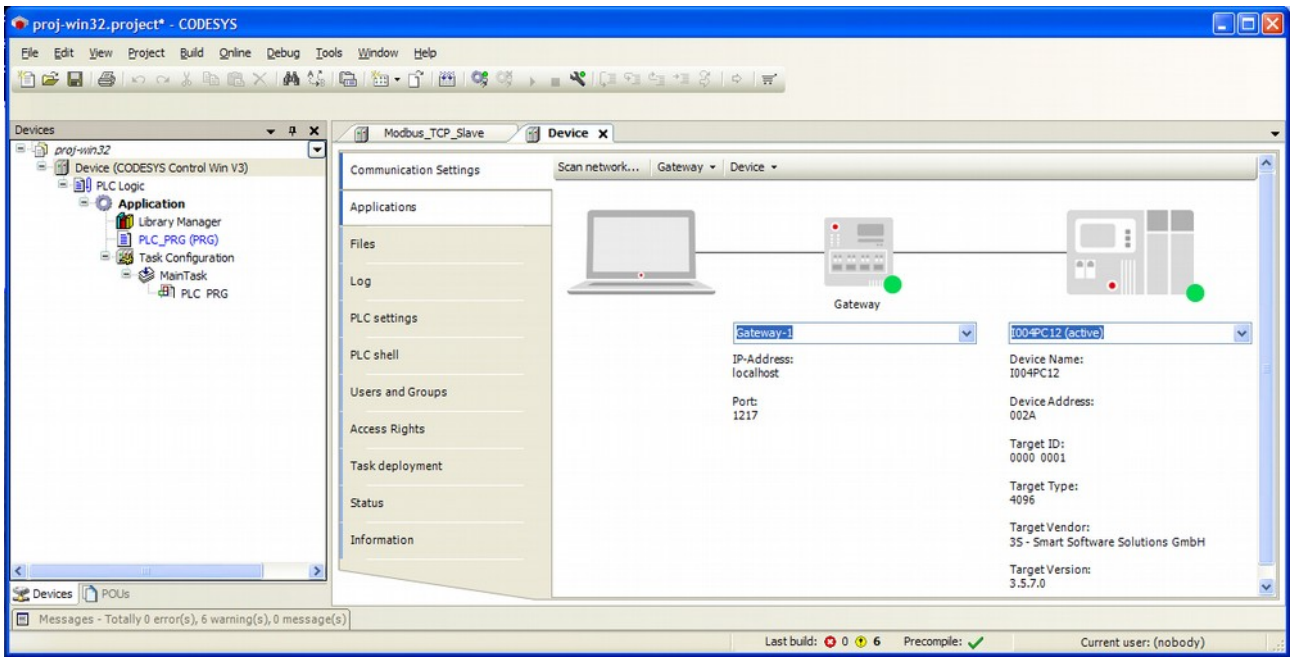
during project creation you will be asked to choose the computing device (computer) that will be running the PLC programs, as well as the programming language for the Program (an IEC 61131-3 POU) that is going to be automatically created. For the device, you should choose either the "Control for Raspberry Pi" or the "Control Win v3 x64", as both these devices support the Modbus communication protocol that we will require later on to communicate with the manufacturing plant. The "Control Win V3 x64" will execute on a standard windows PC (including the computer where the Codesys development environment is running), whereas the Raspberry Pi version will require a Rasberry Pi running the Raspbian Linux distribution, and where the Codesys Runtime has been installed.



You will now see the newly created project. The project browser on the left gives you an overview of the whole project. By double clicking on (some of) these entries you will be shown an editor on the right hand side, in which you will be able to program or configure the respective entry.



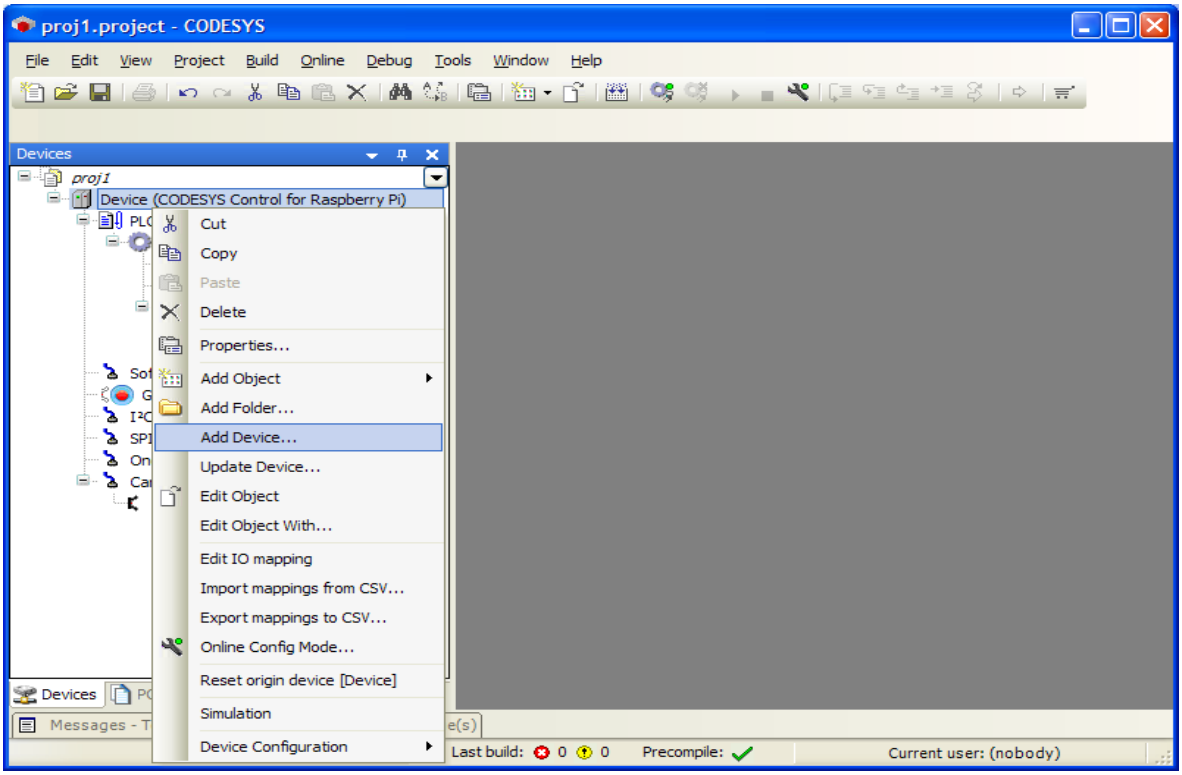
You can now compile your program, and execute this as yet empty program on a runtime.
To compile, use the menu option “Build->Build” or the shortcut key F11.
To execute the compiled program you will first need to start up the runtime virtual machine. If you are using the Raspberry Pi (with raspbian and codesys already installed), just connect it to the same IP network as the computer running the Codesys IDE. If you are using the “Control win” runtime, you may need to start it up manually on your computer. Goto the the 'start' button and execute the “CODESYS Control Win v3 x64” program (“Start -> 3s CODESYS -> CODESYS Control Win v3 -> CODESYS Control Win v3 x64”).
You can now configure how the Codesys IDE will communicate with the runtime virtual machine. Double-click the device on the project explorer window, and configure the ‘Communication settings’. If using the Raspberry Pi, over ‘device name’ place the IP address of the Raspberry Pi. If using the “Control Win v3” runtime, place here the localhost IP address (127.0.0.1). Don’t forget to activate the new IP address by pressing ‘enter’.
The Codesys IDE should now confirm that it found the device in question.



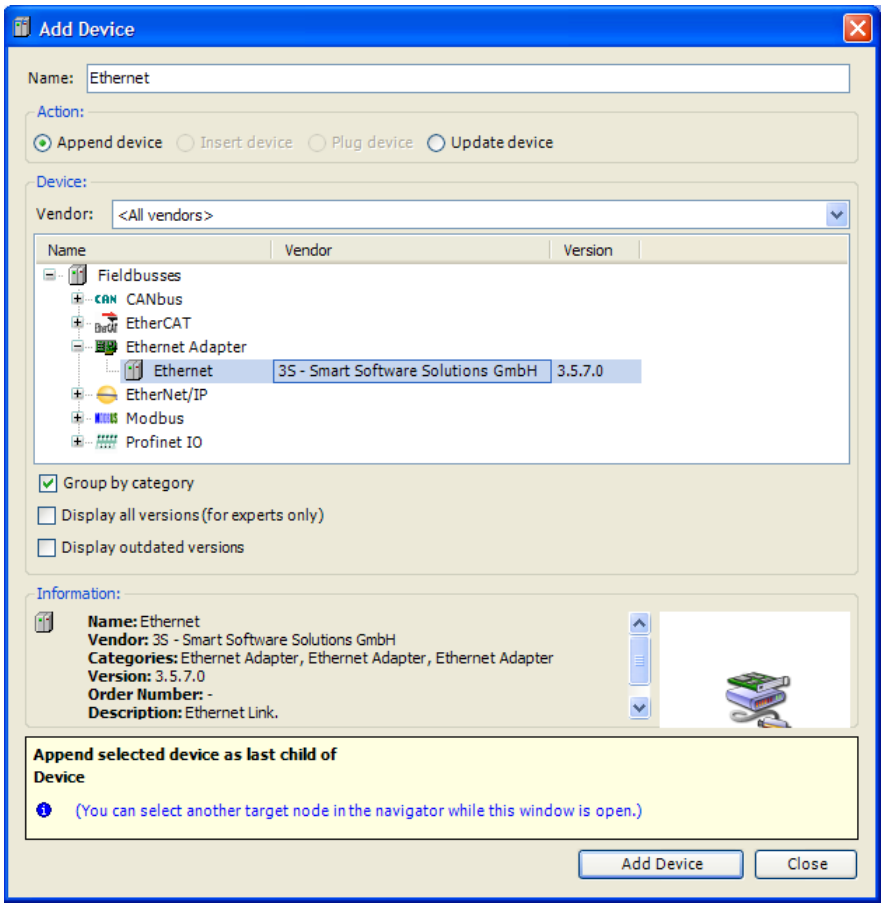
Connect to the device using the menu option “Online -> Login”.
Download the program (if not yet downloaded) using “Online -> Multiple download”.
Start the program execution using “Debug -> start”.

3. Configuring a Modbus/TCP Master

We will begin by configuring the communication to the plant floor, using the Modbus/TCP communication protocol. Right-click on the Device, and choose ‘Add Device’.



You will be shown a window where you may choose the device to add. Start off by adding an Ethernet Adapter.

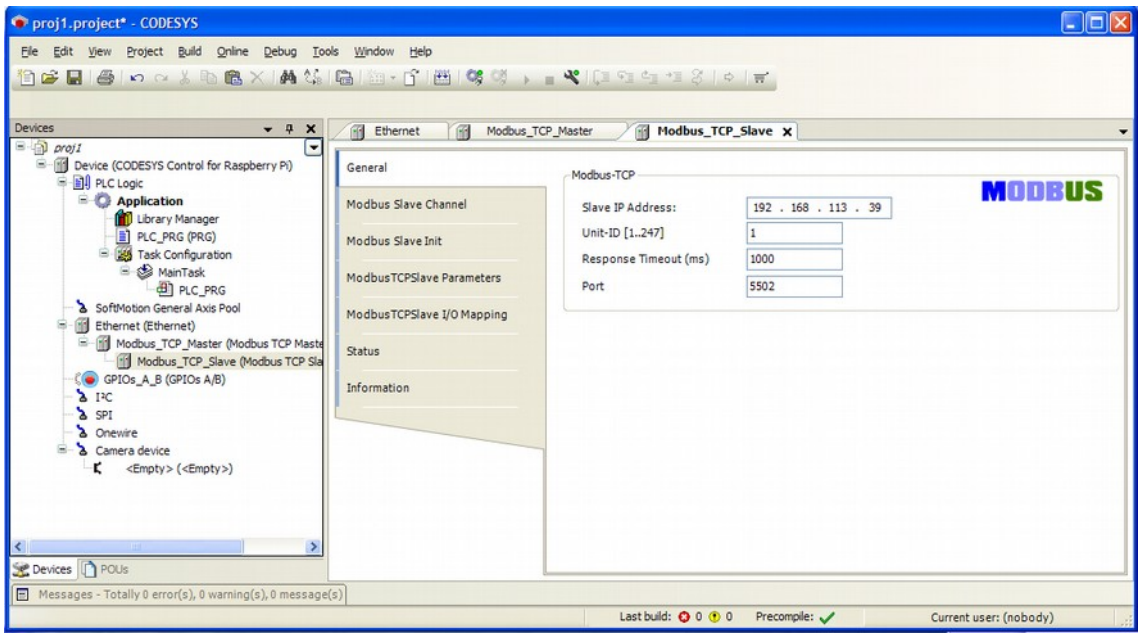


On the Project browser (‘Devices’) window on the left, select the newly created Ethernet Adapter, and add a ‘Modbus TCPMaster’. Once again, select the created ‘Modbus TCPMaster’ in the ‘Devices’ list, and add a ‘Modbus TCPSlave’.

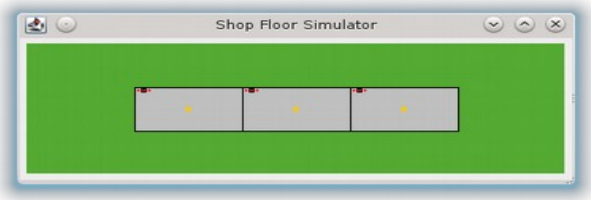
You should now have an Ethernet Adapter containing a Modbus TCPMaster, which in turn contains a Modbus TCPSlave. By double-clicking on the Modbus TCPSlave, its configuration window will be shown on the right.

Configure here the IP address of the Modbus Slave, as well as the port and unit ID.

- If you are running the plant floor simulator (Java program) on your computer, you should place here the IP address of your computer, the port 5502 (default value used by the plant floor simulator), and the unit ID 1. (Note: if you are using the “Control Win v3 x64” runtime running on the same PC as the plant floor simulator, you **must** use the localhost IP: 127.0.0.1. Do not use the PC's IP address. This is due to a limitation/bug of the plant floor simulator.)
- If you are using a Remote Terminal Unit (RTU), e.x. SixnetIO slave, or Schneider STB 'islands', then use the IP address of the RTU, and port 502 (default port for Modbus/TCP).



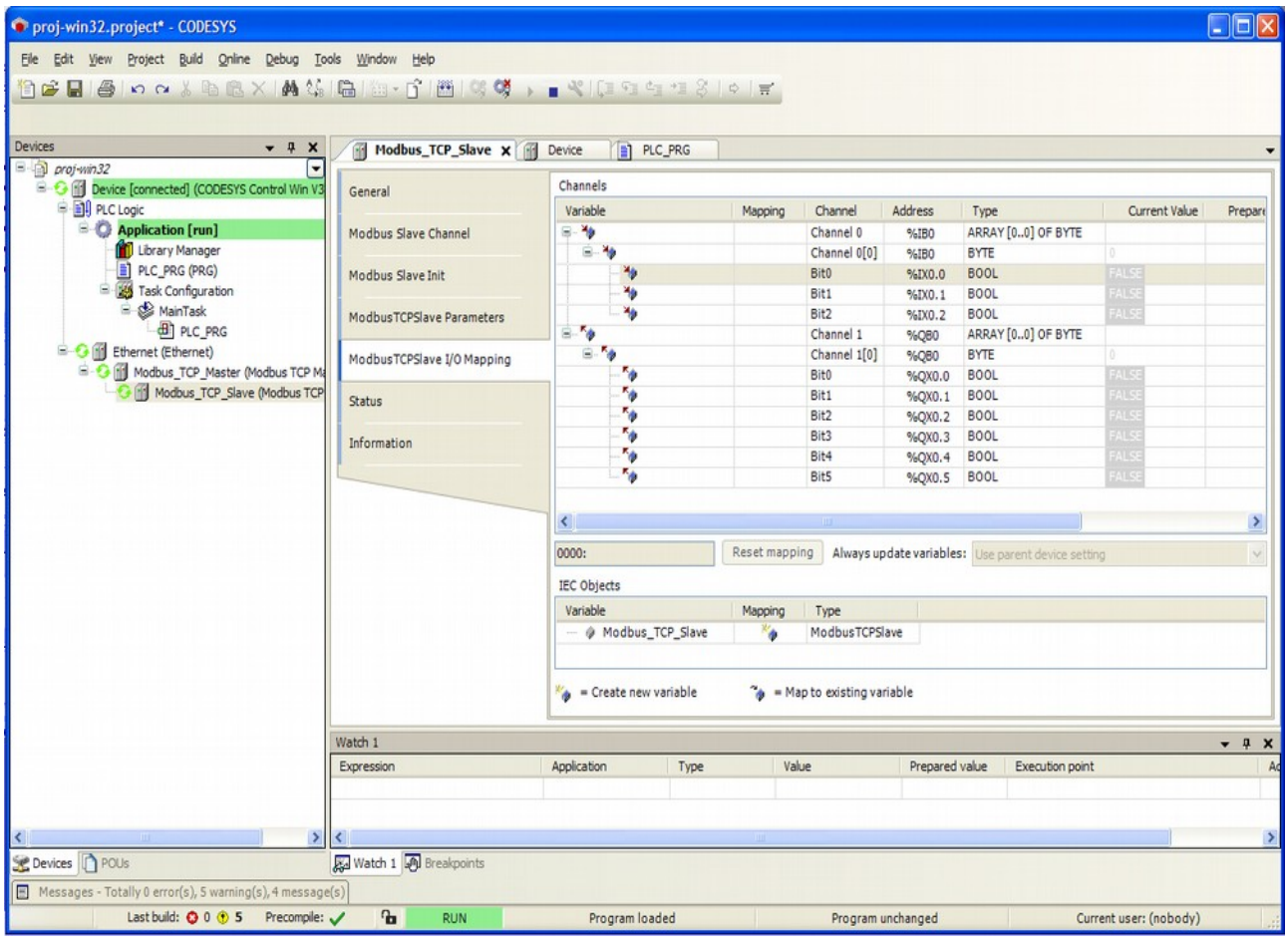
You will now need to configure which Modbus functions should be used when communicating with this slave. This is done by adding a “Modbus Slave Channel” for each required function.
Add one channel using the “Read Discrete Inputs (Function Code 2)”, with an offset of 0, and length 3. This will be used to read the 3 sensors on the 3 conveyors of plant floor used for this initial guide.
Add one more channel using the “Write Multiple Coils (Function Code 15)”, with an offset of 0, and length 6. This will be used to write to the 6 actuators on the 3 conveyors of plant floor.
Compile, download and run this new program. Launch the plant floor simulator.



	Move Left	Move Right	Sensor
T1	Coil 0	Coil 1	Input 0
T2	Coil 2	Coil 3	Input 1
T3	Coil 4	Coil 5	Input 2

(NOTE: when connecting to a Schneider STB, please use the “Read Holding Registers” and “Write Multiple Registers” instead of “Read Discrete Inputs” and “Write Multiple Coils”)

You should now be able to list the status of the sensors and actuators on the Modbus slave (i.e. the plant floor simulator). Double-click on the ‘Modbus Slave’ device in the project explorer window. Open the “I/O mapping” tab.



Note that the ‘status’ of all sensors and actuators are a pale gray. This means that this status is not up-to-date. This is because although the Modbus functions are configured to read the sensors and write to the actuators, Codesys realizes that none of these values are actually being used by any program, so it does not in fact run the communication with the Modbus slave.
You may however see that the syntax to access these inputs and outputs in your program is %IX0.0 to %IX0.2, and %QX0.0 to %QX0.5.

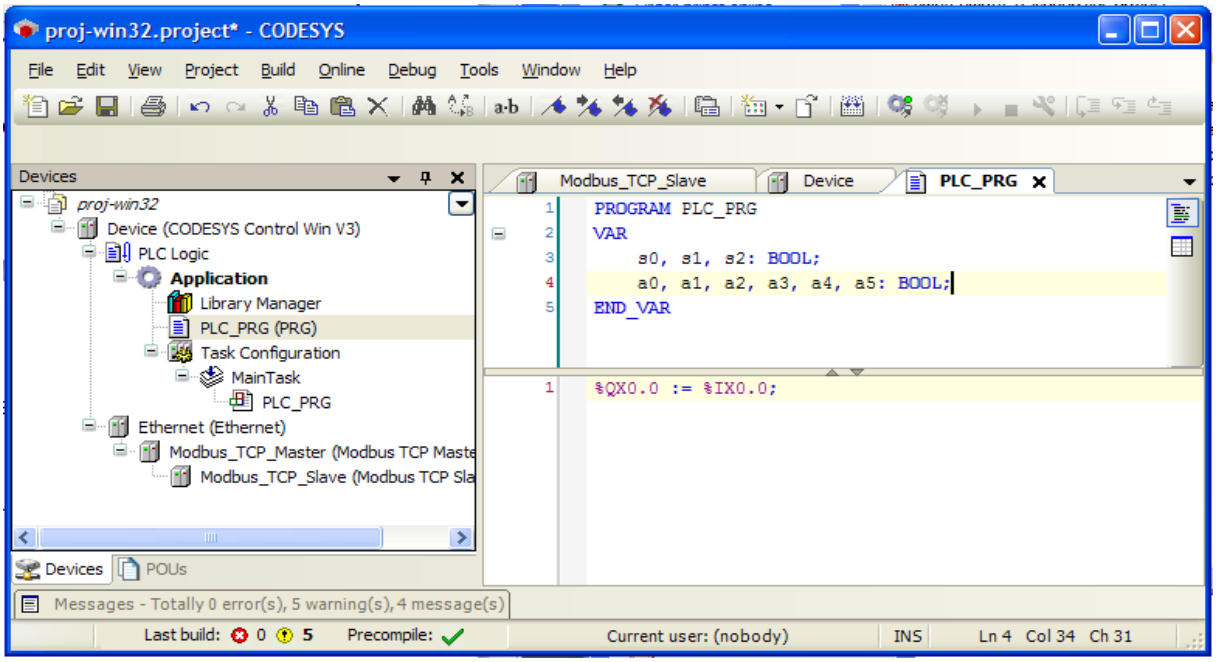
4. Writing a Simple Program

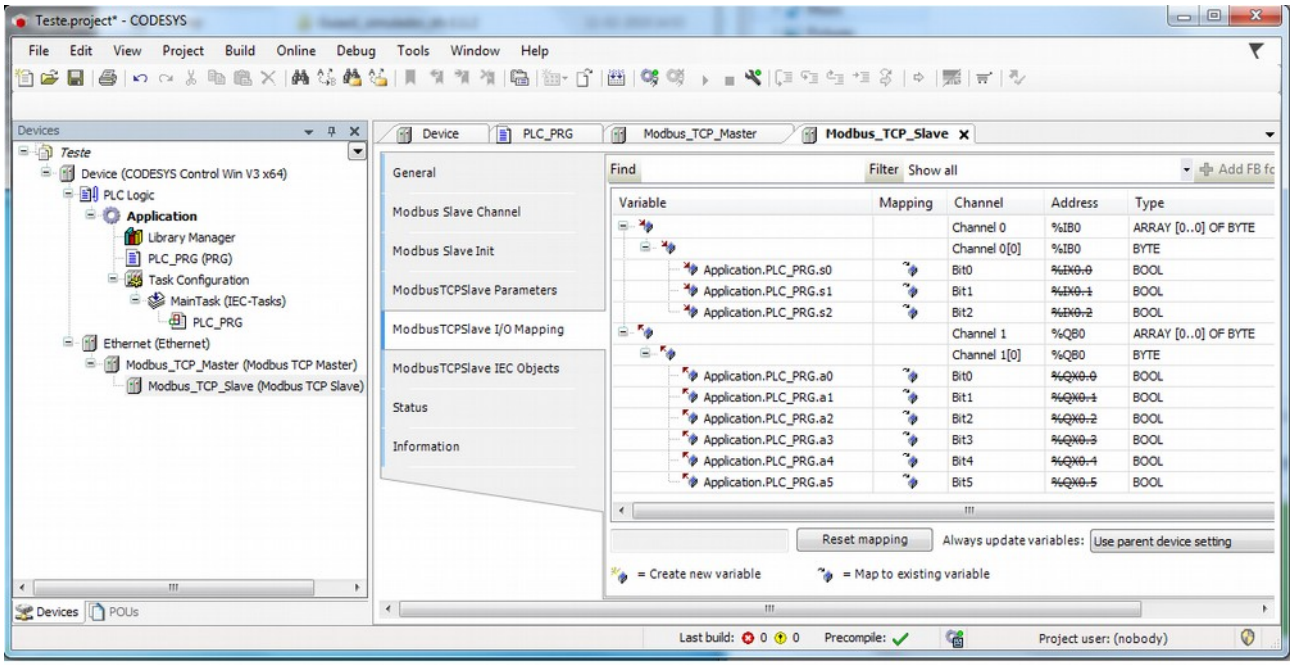
Go offline from the runtime device.

Edit the program so that the status of the first sensor is copied to the first actuator:

```
%QX0.0 := %IX0.0;
```

If you wish, you may take the opportunity to declare some Boolean variables.





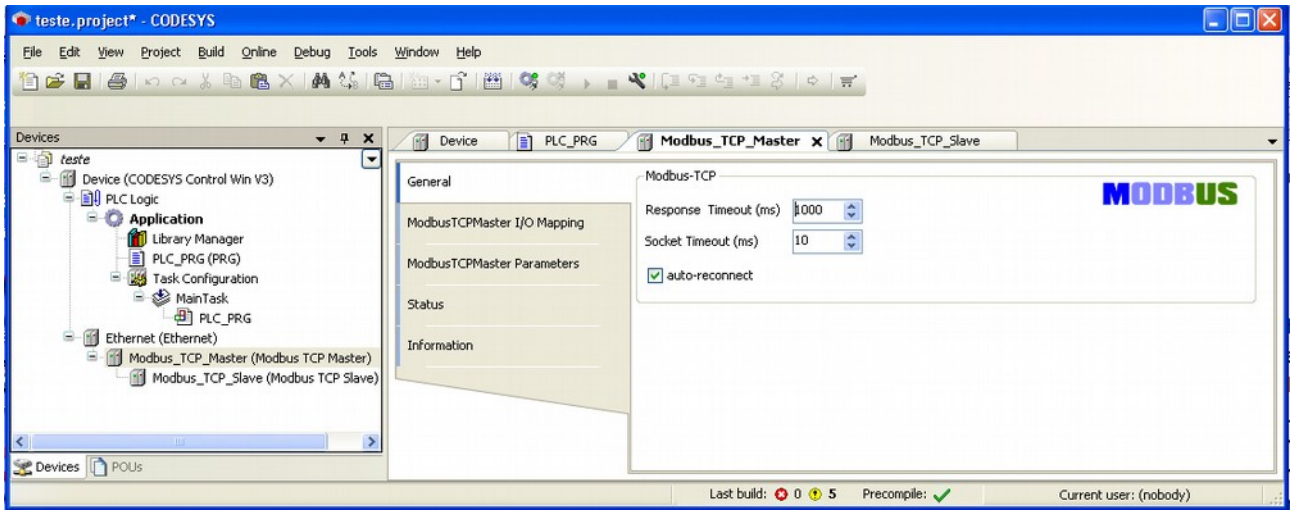
With the Boolean variables created, you can now map these variables to a specific input or output. This is done in the Modbus Slave device property editor.
NOTE: To insert the “Application.PLC_PRG.xxx” mappings, double click that cell (i.e. the cel under the 'variable' column, not the 'mapping' column).

Once the variables have been mapped, then you are free to change your program to use the variables instead:

```
a0 := s0;
```

Compile, download, and run both versions of this program. Test its execution by placing a workpiece on the left conveyor.

You will notice that for the PLC to correctly communicate with the plant floor simulator, you will need to launch the plant floor simulator **before** downloading and launching the PLC program. This is because, by default, the PLC will only try to connect to the Modbus Slave (i.e. the plant floor simulator) after a cold boot.
To change this behaviour and have the PLC program continuously try to reconnect to the Modbus Slave after a communication failure, please tick the option "**auto-reconnect**" on the Modbus Master tab.



5. Writing Complex Programs

You are now in a position to write your first control program You may do this by creating other POUs (example, a FB), and programming these in any of the 5 IEC 61131-3 programming languages.

Exercise:

Write a simple program that will transport a workpiece from the conveyor on the left to the conveyor on the right, and back again. Let this run in an infinite loop with the workpiece going back and forth. Do this using ST, and again using SFC.