sigmadue Control Unit CU-02

Quick Guide

Quick Guide
Q.G. Sigmadue CU-02-1b/11.07
Cod. J30 - 478 - 1ACQGS2E

Ascon Tecnologic srl

Headquarters:	via Indipendenza 56, 27029 Vigevano (PV)	***Milan office:***	Via Falzarego 9/11, 20021 Baranzate (MI)
Phone	+39 02 333 371	***Fax***	+39 02 350 4243
`www.ascontecnologic.com`		`sales@ascontecnologic.com`	

INDEX

Prerequisites

The products described in this manual should be installed, operated and maintained only by qualified application programmers and software engineers who are almost familiar with EN 61131-3 concepts of PLC programming, automation safety topics and applicable national standards.

This quick guide gives the first principles of use for the ASCON sigma**PAC** system. To start programming, you need this guide, a PC, the OpenPCS programming suite and at least the sigma**due** CPU module.

There can be three possibilities about your hardware and software:

1. Use the sigma**PAC** demo box. In this case you have all the things you need. Just install OpenPCS on your PC and connect it to the demo box.
2. Use sigma**due** CPU alone. In this case you should provide:
   A power supply of adequate characteristics;
   An Ethernet CAT 5 cross cable
   (e.g. ASCON part #: AP-S2/CABLECUPROG);
   The service port serial cable
   (e.g. ASCON part #: AP-S2/CANBLECUCONF).
3. Use sigma**due** CPU with some I/O modules; in this case you should provide:
   A power supply of adequate characteristics and power;
   An Ethernet CAT 5 cross cable
   (e.g. ASCON part #: AP-S2/CABLECUPROG);
   The service port serial cable
   (e.g. ASCON part #: AP-S2/CANBLECUCONF);
   A CANopen cable for each connected I/O module
   (e.g. ASCON part #: AP-S2/LOCAL-BUS152);
   A CANopen terminating device
   (e.g. ASCON part #: AP-S2/TERM-CAN);
   A CANopen network configurator (for advanced users).

Using this manual

Specifications within the text of this manual are given in the International System of Units (SI), with non SI equivalents in parentheses.

Fully Capitalized words within the text indicate markings found on the equipment.

Words **in bold** style within the text indicate markings found in the Configuration Tools.

Warnings, Cautions and Notes are used to emphasize critical instructions:

DANGER!

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

WARNING

Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.

Caution

Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury, or property damage.

Note: Highlights important information about an operating procedure or the equipment.

Current Documentation on the Internet

Make sure you are always working with the latest version of this document.

Make sure you are always working with the latest version of this document.

Ascon Tecnologic Srl reserves the right to make changes to its products in the name of technological advancement. New manual revisions, when published, and can be found online at:

http://www.ascon.it

Chapter 1
Hardware Installation

1-1 Mechanical installation

The sigma**due** CPU and the I/O modules are installed on standard DIN rails.
In a normal cabinet layout the first slot on the left is usually reserved to the CPU. This comes from the fact that the CPU has just one CANopen outlet, on the right end. Up to 127 I/O modules can be connected in chain to each CPU. This value is the theoretical limit, Ascon spa reccomends to never exceed the number of 32 units.

1-1-1 Installing and Removing modules

A complete description on how the modules can be mounted on or removed from a DIN Rail can be found in the "*CU-02 Installation Manual*" **[6]**.

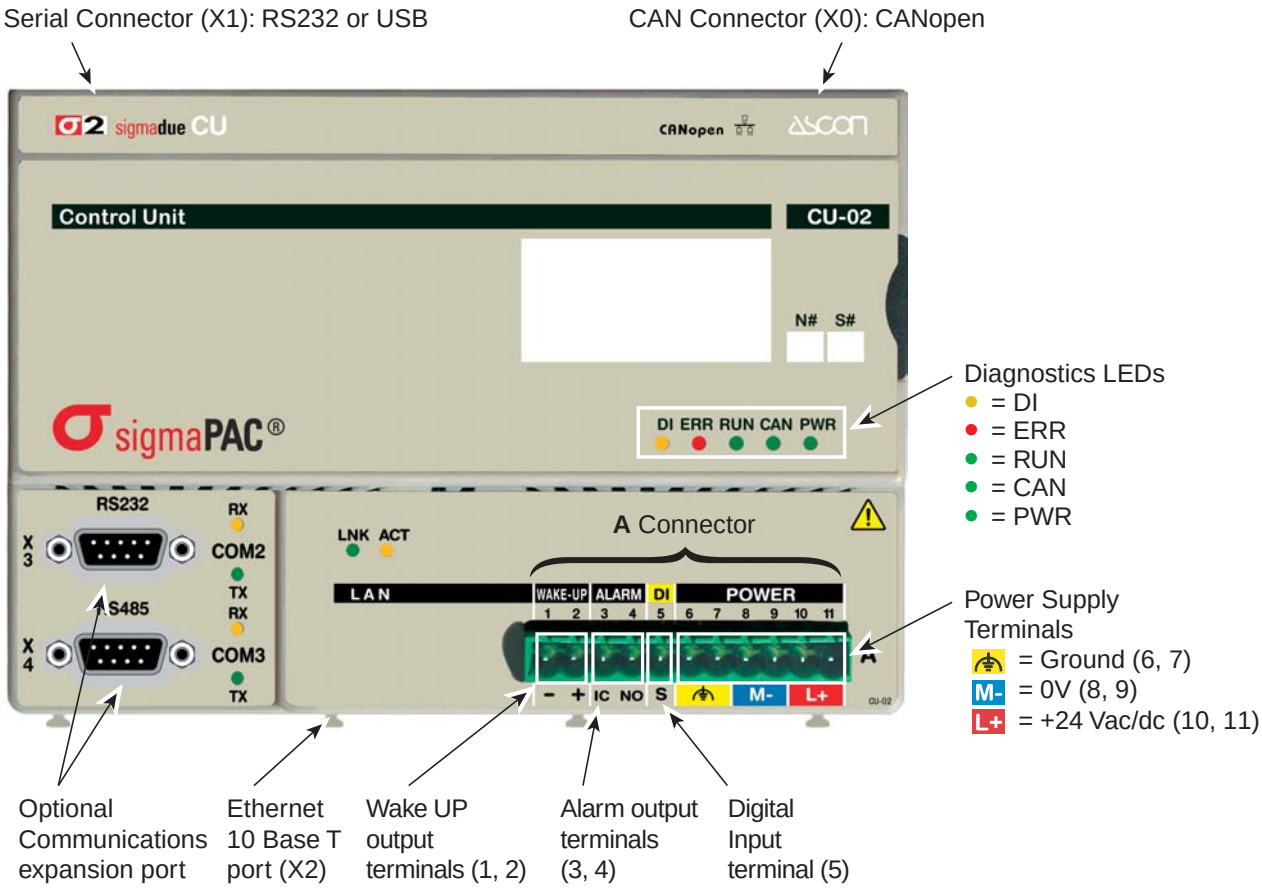


Figure 1.1 -CPU I/O and Communication Ports

For pin-outs and electrical characteristics see the Installation Manual.

1-2 Electrical installation

Referring to "*Figure 1.1 -CPU I/O and Communication Ports*"

1-2-1 Connect the communication cables

CANopen remote I/O Port

For CANopen I/O modules connection (**X0**). I/O modules are connected with the included cables in a daisy chain fashion. The RJ45 type connectors have the pinout:

Pin	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Signal	CANH	CANL	CAN-GND	NC	NC	CAN-SHLD	CAN-GND	CAN-V+

Serial Port

RS232 service port (X1)
The connector X1 on the CPU module is an RJ45 type, with the following pinout:

Pin	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Signal	NC	NC	NC	GND	RX	TX	NC	NC

USB service port (X1)
When installed , the connector X1 on the Contro Unit is a B type USB standard connector. The pinout of this cable is meaningless as the connection is standard.

Ethernet 10baseT

For OpenPCS development station (**X2**). The connector on the CPU module is an RJ45 type, with the following pinout:

Pin	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Signal	TX+	TX-	RX+	NC	NC	RX-	NC	NC

1-2-2 Connector "A" connections

Power supply

Use the 6 poles on the right of the "**A**" connector and respect the polarity. Each of these terminals is doubled in order to allow the user to power, using an additional terminal block, other devices or sensors.

Pin	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Name	-	+	IC	NO	S	FE	FE	M-	M-	L+	L+
Function	WAKE UP		ALARM		DI	F. EARTH		POWER SUPPLY			
Signal						FE	FE	**0V**	**0V**	**+24V**	**+24V**

Power Supply

Auxiliary ports

The 5 poles on the left of the "**A**" connector are auxiliary ports.

Pin	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Name	-	+	IC	NO	S	FE	FE	M-	M-	L+	L+
Function	WAKE UP		ALARM		DI	F. EARTH		POWER SUPPLY			
Signal	COM	OUT	COM	OUT	INPUT	FE	FE	**0V**	**0V**	**+24V**	**+24V**

Auxiliary ports

Wake up Software activated Digital Output. 24Vdc, 0.2A high side power switch, terminals 1 and 2;

Alarm Relay type digital output. SPST NO 24V, 1A, terminals 3 and 4;

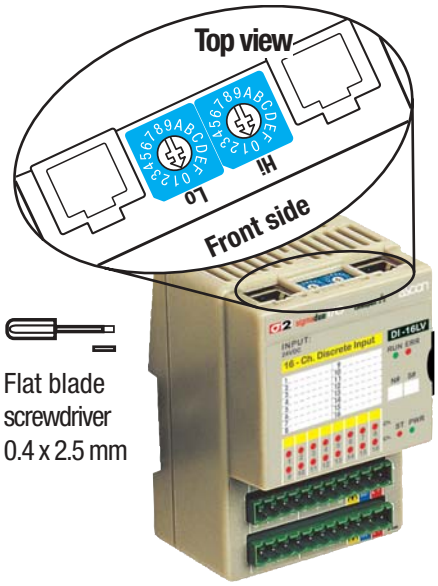
DI 24Vdc digital Input, terminal 6. The return path can be linked to terminal M- (terminal 8 or 9) or to Wake-Up terminal (teminal 1).

Chapter 2
Node ID and Baud Rate Configuration

In the case the User wish to install I/O modules, the first to do is Node ID and Baud Rate settings.
All I/O modules come with default values for Node ID and speed.
It goes without saying that the Node ID must be different for each node and the speed must be the same. Therefore, for the purposes of this Quick Guide, the User may want to set IDs only and keep the speed at default value: 500kbps.
Two rotary switches (**HI** and **LO** hexadecimal switches) are used to set the module's Bit Rate and CAN Node ID.

LO switch	**Bit Rate** kbps	**Bus Lenght m**
1	20	2500
2	50	1000
3	100	500
4	125	500
5	250	250
6	500	100
7	800	50
8	1000	25

	HI switch	**LO switch**	**Valid Node ID**
Value	0	1	01h (1d)
	0	2	02h (2d)
	↓	↓	↓
	7	F	7Fh (127d)



Note that Bit Rate is used as synonymous of Baud Rate.
The Bus Lengths are given for reference only.

The default values are:
- Bit Rate = 500kbps,
- Node ID = 127d

The complete procedure is as follows:

1. Turn the Power OFF
2. Set the HI switch to "F"

Note: Select the desired Bit Rate value by setting the HI switch following the table (e.g. "8" for 1 Mbps)

3. Turn the Power ON
4. Shift the HI switch to "E" (all the module service LEDs should flash)

5. Turn the Power OFF. Now configure Node ID
6. Set the HI and LO switches to the desired valid Node ID, following the table
7. Turn the Power ON

To configure the Node ID only, just start from step 6.

Note: The default Node ID for the CPU module is **32d**.

Chapter 3
OpenPCS Programming Suite Installation

The OpenPCS programming suite from Infoteam is provided on CD-ROM.
It is also available online at: www.infoteam.de.

3-1 Installing OpenPCS

3-1-1 Hardware and Software Requirements

OpenPCS requires a PC with at least:

- Pentium II, 1GHz;
- 512 MB RAM;
- 16 GB of free disk space;
- CD-ROM and 1024*768 resolution;
- Windows 2003 Server, Windows XP SPII or Windows Vista 32bit.

3-1-2 Installation

OpenPCS is provided on CD-ROM. The CD auto-starts a screen where you can select the software you want to install. If auto-start is not activated or does not work, please start the last distributed OpenPCS programming tool version (e.g. `OpenPCS_Ver_631e.exe` file) available in `X:\SETUP\` folder ("**X**": is the letter assigned to the CD-ROM drive in your PC).
At the end of the installation, you will be asked if you want to install hardware drivers. If drivers were provided with your PLC, enter the path to the hardware driver, otherwise click "**Exit**". If drivers were received for your PLC, a license key for OpenPCS was also included. See Licence Editor for how to insert a licence key. If you did not receive a hardware driver nor a licence key, OpenPCS is still fully functional, but restricted to 'SIMULATION'.

Note: Installations to substitute drives are not supported by Windows XP.

3-1-3 Starting OpenPCS

With Windows started choose:
Start → Programs → infoteam OpenPCS 2008 → infoteam OpenPCS 2008
this will open the Framework.

Chapter 4
ASCON target .cab file Installation

4-1 Configuring OpenPCS

In order to work with the Ascon CPU target, you must install in OpenPCS a **cab** file. The file **Ascon_sigmadue_Lxx_Hyy_zzzz.cab** contains all the files describing Ascon sigma**due** Hardware, drivers, examples and utilities (**xx**, **yy** and **zzzz** are digits to identify the software version).
In the OpenPCS "*Extras*" menu, select "*tools – Driver install…*". "*Select*" the desired cabinet (e.g. Ascon_sigmadue_L13_H7_2009.cab), then "*Install*".

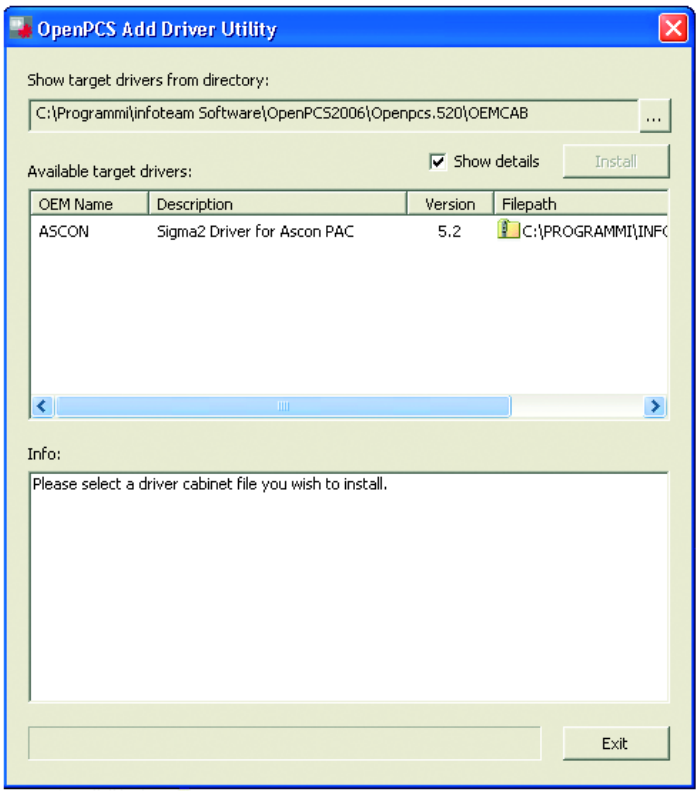


Figure 4.1 - OpenPCS OEM Driver Installation

Chapter 5
PC Ethernet port configuration

In order to communicate with the ASCON CPU you have to set the IP address and subnet mask of your PC.
To do this, go to the
Start → Control Panel → Network Connection → LAN.
Right-click it with your mouse to show the context menu, and select “Properties”.
In the “**General**” sheet select “**Internet Protocol**” and chose “**Properties**”.
In the “**General**” sheet now you can set:

IP address	`192.168.5.xx`	xx: all except 11
subnet mask	`255.255.255.0`	

Chapter 6
OpenPCS Set-up

6-1 OpenPCS Setup

To connect the OpenPCS development system to the Ascon target, a new connection must be defined.
Select "*Connections...*" item in the "*PLC*" menu. In the window of *OpenPCS Connection Setup* select "*New*".
Now in the window "*Edit connection*" it is possible to set the new connection. In the field "*Name*" you can name the new connection.
By pushing the "*Select*" button you can pick the driver that manages the communication with the target: for Ascon CPU is TCP52.

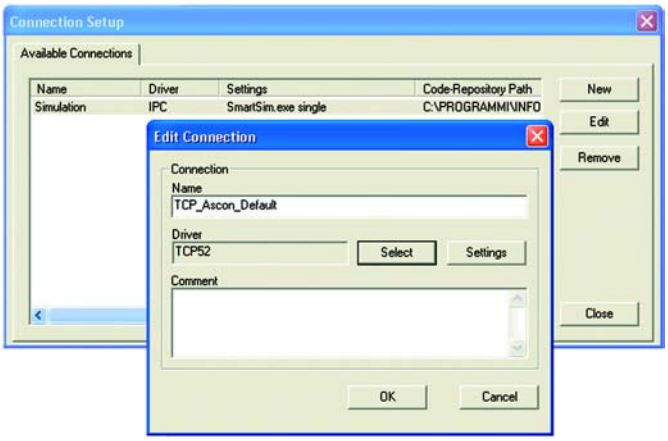


Figure 6.1 - OpenPCS Connection Setup

By clicking the "*Settings*" button you can set set the communication parameters.

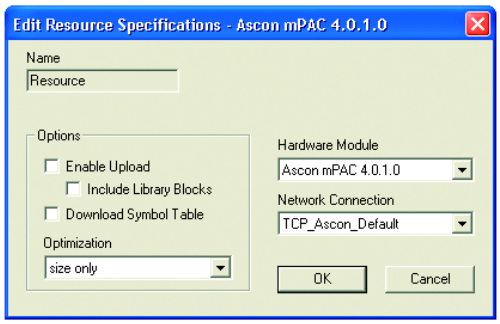


Figure 6.2 - TCP Settings

The Port number and IP address must be the same as those configured at the initial CPU configuration session. See the Ethernet setup menu, items 7 and 2.
OpenPCS environment is now ready to communicate with the Ascon target.
The project must be set up in order to use the CPU.

Select the "*Resource Properties*" item in the PLC menu, select "*Ascon…*" in the "*Hardware Module*" field, then select the newly created TCP connection in the "*Network Connection*" field.

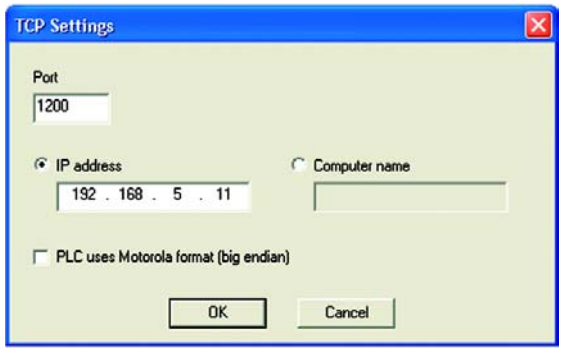


Figure 6.3 - OpenPCS resource Specifications

The code "*Optimization*" menu allows for three choices of compilation: "*Normal*" and "*Speed only*" refers to the NCC: Native Code Compilation, while "*Size only*" refers to the standard code.
Please note that the use of NCC does not permit the user to insert break points in debugging projects.

Setup Communication Timeout

There are several conditions that could make it necessary to set the Ethernet Port communication timeout to a value higher than the default value. This timeout checks the dialogue between OpenPCS and the target CPU. When dealing with large programs, it may be necessary to set a longer driver timeout. The default value of 20000ms can be increased by using the following register key:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\infoteam Software GmbH\
OpenPCS\6.x.x\Online\TcpDriverTimeout_ms]
```

Value = "20000" means a timeout of 20 seconds.

Chapter 7
Getting started: the first project

To introduce you to OpenPCS, we will use a simple example shown below. The rest of this chapter will then implement the solution with OpenPCS.

Problem: A blinker shall blink when a button `button` is pressed with an interval of 2 seconds. If the button is released, the blinker should immediately turn off.

Solution: might look like this:

```
PROGRAM ❶ blinker_st ❷
VAR ❸
button  ❹    AT %I0.0 :   BOOL; (*input ❺*)
blinker      AT %Q0.0 :   BOOL; (*output*)
timer            :         TON; (*timer functionblock*)
END_VAR
(* call 2s-timer ❺*)
       timer(in:= button,pt:=T#2s); ❻
       (* if 2s are over...  *)
       if (timer.q) then
             (* ...  toggle blinker *)
             blinker := not blinker;


             (* and reset timer! *)
             timer (in:= false);
       end_if;

       blinker := blinker AND button;
END_PROGRAM ❶
```

❶ The program starts with the keyword PROGRAM, and ends with the other keyword END_PROGRAM. When working with OpenPCS, you will not type in these keywords, but rather the editor will create them automatically for you.

❷ OpenPCS will prompt you for the name of the program when you create a new program.

❸ In contrast to traditional PLC programming languages, IEC61131 requires that you declare all variables that you use.

❹ This line declares a variable of name ‘button’ of data type ‘BOOL’, to be mapped to hard ware address ‘%I0.0’, i.e. this variable shall denote the lowest bit of the first input byte.

❺ Almost everywhere in IEC61131 you can use comments to describe your programs.

❻ The instruction part of the program starts with a call of a functionblock. As you will notice, most instructions are assignments and function(block) calls.

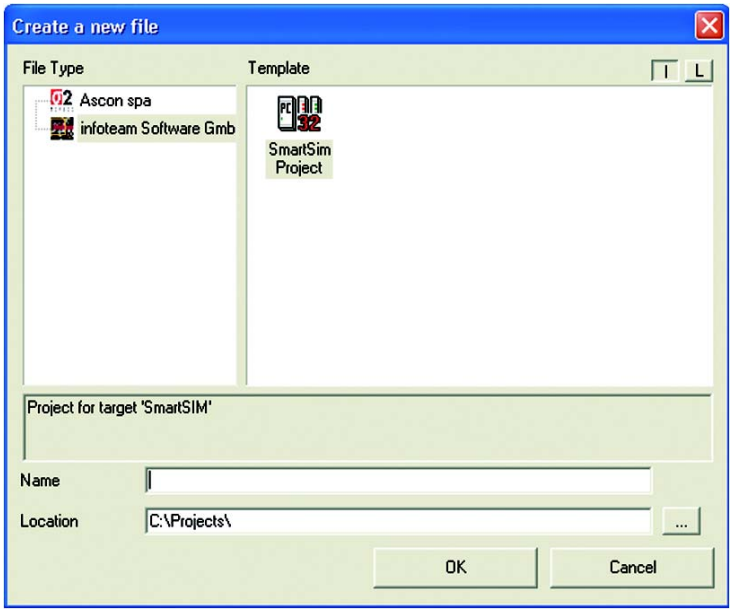
In this sample program, we have a functionblock **`TON`** (output is true after expiration of time), control strucure **`IF`** (code of this block is only executed if its expression is satisfied), assignments **`:=`** (result of the right is assigned to variable on the left) and the operator **`AND`** (and-connect operand to current result).

Please note: which hardware addresses are valid is strongly dependent on the PLC you are using.

7-1 Creating a New project

For our first OpenPCS code, we will set up a new project.

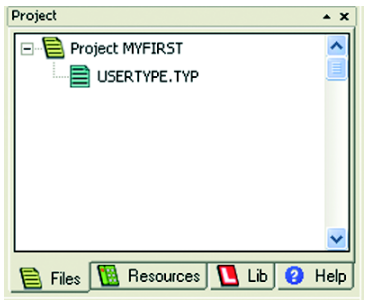
Start OpenPCS and select ***File*** → ***Project*** → ***New***. A dialog box will prompt for the name and location of the new project.



Enter a name of your choice, e.g. “MyFirst”.

Note: The name of an OpenPCS project should not contain blank (space) characters or special characters. Plus, for easy updates, it is recommended that you store your application separate from OpenPCS. To give an example, **`C:\PROJECTS`** is a good location to store your projects.

Now, the Browser contains the new project.

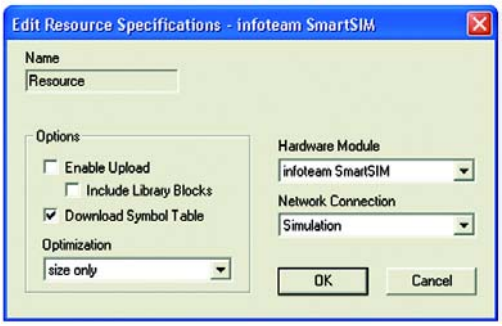


There are different views on your project:

1. The **Files-Pane** shows all files of the project;
2. The **Resource-Pane** displays the current configuration, with all defined Resources and their tasks;
3. The **Library-Pane** contains all installed libraries;
4. The **Help-Pane** shows the help topics.

OpenPCS has already created one (empty) file to contain your type definitions named “**usertype.typ**” and a default resource, named “**resource**”.

Typically, the default resource will need to be configured properly for your controller. We are not using any controller here, so the resource is quite ok, but to demonstrate breakpoints later we will need to set optimisation low enough to allow that. Find the “**resource**” entry in the Resource-Pane, right-click it with your mouse to show the context menu, and select “**Properties**”:



Under “**Optimization**”, “**Size only**” should be selected by default. If you use a sample project from your PLC manufacturer, other optimization settings can be set.

WARNING

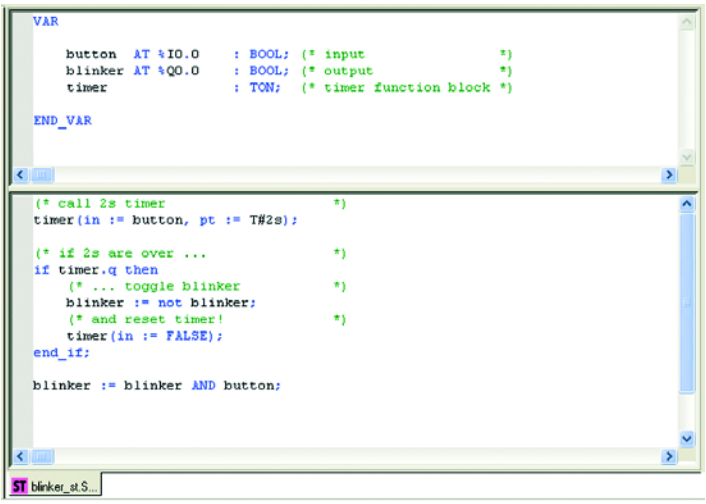
For using breakpoints, optimization must be set to “**Size only**”!

7-2 Writing Code

To create a new program, choose “**File → New**”. A dialog-box will appear, where you must choose the programming language, a filename and the location where the file will be stored. As you can see there are plenty of programming languages that can be chosen, but we will only use ST in this introduction.

Enter “**blinker_st**” as the name and ST as language and press “**OK**”. Do not change the file location. Now you are asked if you want to link the new program to the active resource. Click “**Yes**” and a new task, named “**blinker_st**” will appear in the resource-pane under the active resource.

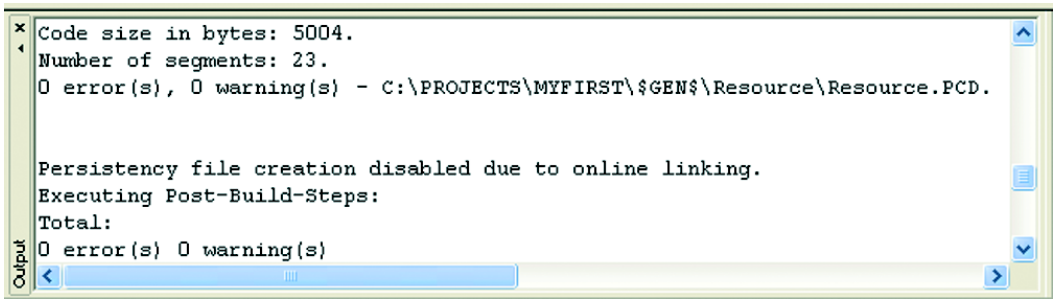
The Editor-Pane will open, displaying two different windows: At the top is the declaration window for your first program, at the bottom is the instruction window of your program. Enter the sample program like shown below:



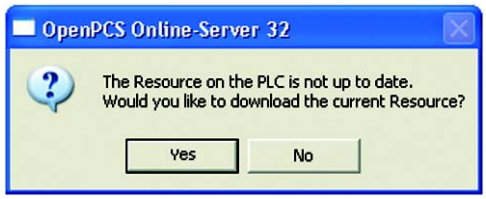
Press ***File → CheckSyntax*** to invoke a syntax check. In the diagnostic output window, you should read "**0 errors, 0 warnings**". If not, carefully check what you have entered.

7-3 Executing Code

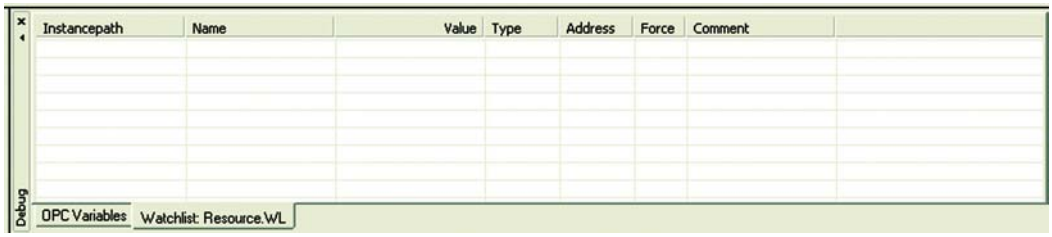
To execute your small application, we need to compile it and transfer the code to the controller first. To build the code for the controller select ***PLC → Build Active Resource*** from the menu bar. In the output window, you will see the compilation proceed. The end of the output should look similar to the following:



After compilation finished successfully, your code needs to be transferred to your controller. Now select **PLC → Online** to Connect to the resource. OpenPCS will detetct, that your application needs to be downloaded and will prompt your permission to do so:



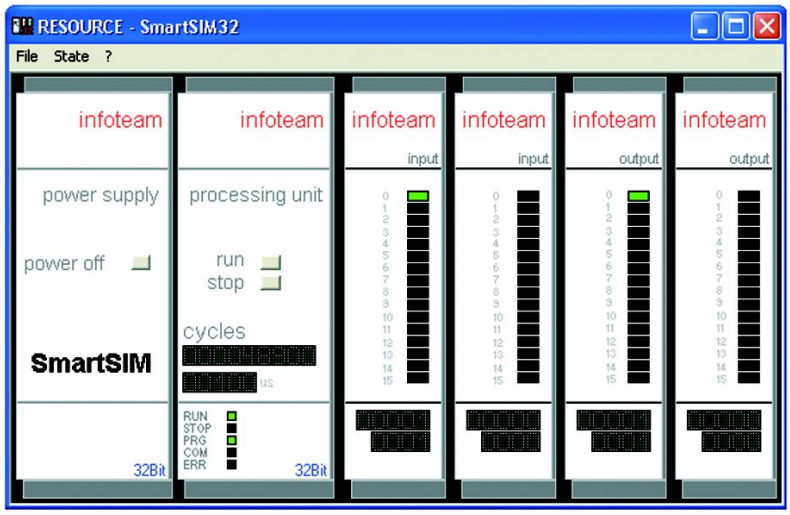
Accept that with "**yes**". You will see a progress bar while the code is being transferred, but for this small example it should be finished very quickly. When download has finished, you will see that OpenPCS automatically opened another of its tools, the "**Test and Commissioning**". This is proof that OpenPCS is online:



Note: In this introduction, we are not using a real hardware controller. Instead, we are using the "*Windows Simulation tool*" that comes with OpenPCS, named **SmartSIM**:

Use ***PLC → Coldstart*** (or press the red arrow in the toolbar) to start execution of your code.

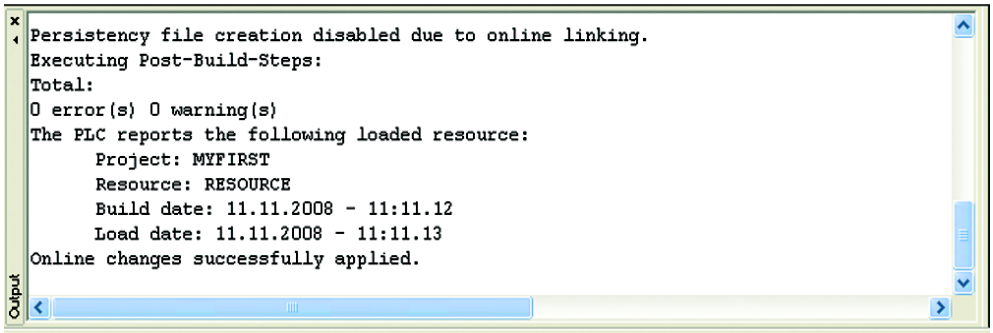
Go to SmartSIM and activate the first input ("**button**"). This should activate the first output ("**blinker**"). After an expiration of three seconds it should go inactive switching back after another three seconds. De-activate the first input ("**button**"), and the first output ("**blinker**") should be inactive.



7-4 Monitoring Code

Now that your application is running, go back to the Browser and find the "**Resource**" in your project. Click all the small plus signs to open the entire tree under the resource entry. This will reveal the "**instance tree**", showing all instances of programs and function blocks and all variables that you used in your program:

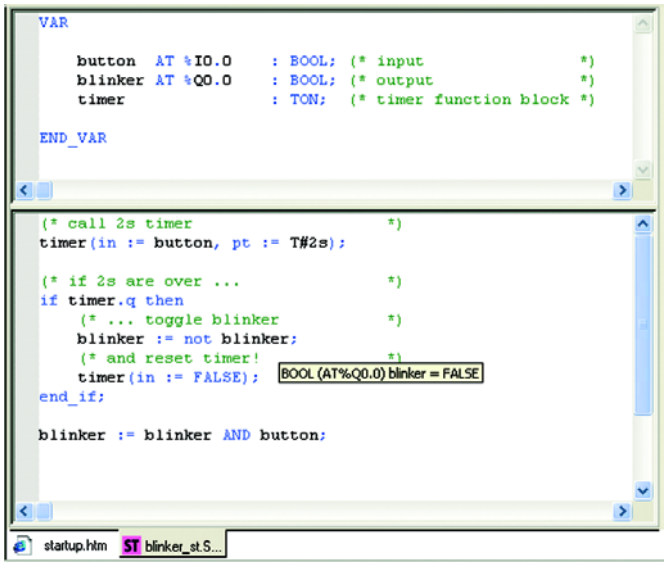
Double-click some of the variable entries (grey boxes with 0/1 shown), and see the corresponding variables added to the watch list in the Test&Commissioning:



Go back to SmartSIM and modify the inputs to see the effect in the watch list.

The ST-Editor will also be in monitor mode. You should see a different cursor once you move the mouse onto the ST-Editor.

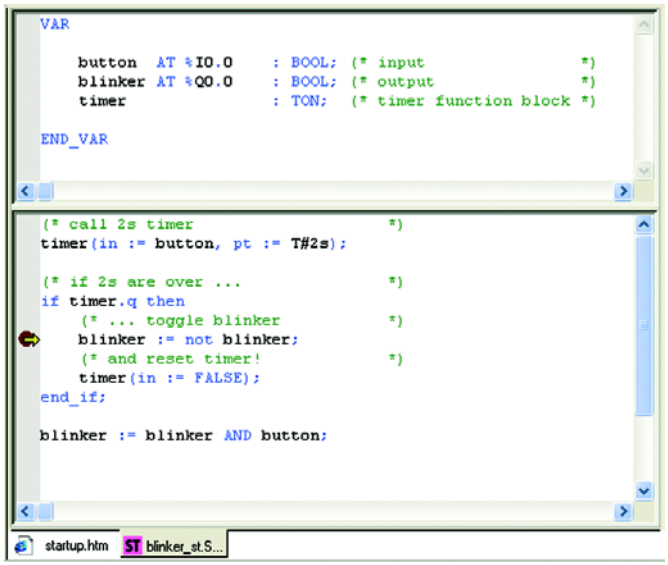
Move the mouse cursor to a variable in your code and after a short period, you will see a "**tooltip**" like display of online value display:



Move the mouse around and point at different variables to examine their values. If the variables are modified by the application, the display will be automatically refreshed.

If you need to analyse the logic of your code, value display alone may not be enough. Move the cursor to a line of your program that contains code, and single click the mouse. Now press **F9** to set a breakpoint to that line. You will see a red dot immediately, marking the breakpoint. Shortly after that, you will notice a yellow arrow, identifying the current instruction pointer.

OpenPCS will display "**Breakpoint reached**" in the output window.



You may still move the mouse cursor around to examine variable values while the controller is stopped at the breakpoint. Press **F10** to single-step through your code, or press **F5** to continue execution. Within a line that contains a breakpoint, press **F9** again to delete the breakpoint. OpenPCS supports "**Online Edit**", for further information see *Online Edit* in the user manual.

Note: If SmartSIM does not stop when you set a breakpoint, you probably did not set **optimisation settings** properly. Be sure your resource is configured for "**size only**".

7-5 Online Edit

Online Edit (or Online Change) is a feature whereby program changes are applied to the PLC without the need to restart it. The following Stepps need to be done in order to run Online Edit.

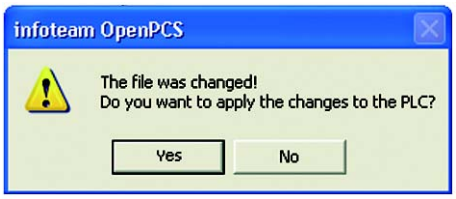
The program must be compiled and running on the PLC. The source is opened in an editor window.

The Editor can be switched from Monitor Mode (green colored symbol) to Edit Mode (red colored symbol) and back via ***PLC → Online/ Edit*** or the corresponding button of the toolbar

.

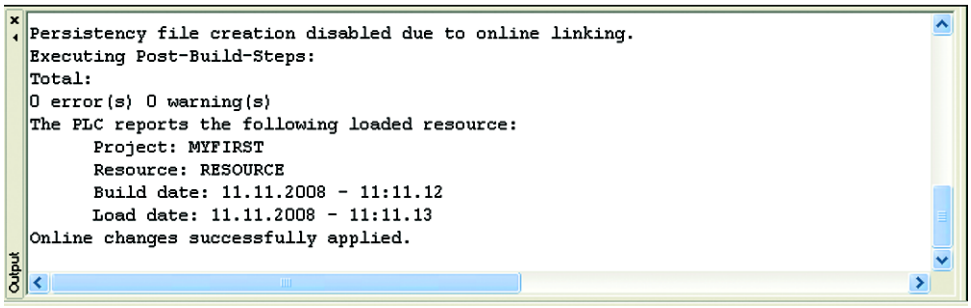
Implement the desired changes and close the Edit Mode via ***PLC → Online/ Edit*** again.

OpenPCS prints a dialogue to accept and download the changes:



If the changes are accepted, OpenPCS recompiles the necessary unit and downloads them to the PLC without stoppig the running cycle. The changes have bearing on the next cycle.

OpenPCS prompts a message in the output window, if the update is finished:



Chapter 8
Working with remote I/O Modules

8-1 Introduction

The Ascon sigma**due** system is based on the remote I/O modules of sigma**due** I/O series. The modules communicate with CU-02 CPU through the well known CANopen protocol. In order to establish the link with infoteam OpenPCS development system, a configuration session of the CANopen network should be performed.

Configurators support project managers in all project phases, such as planning, development, startup and service in conjunction with CANopen networks.

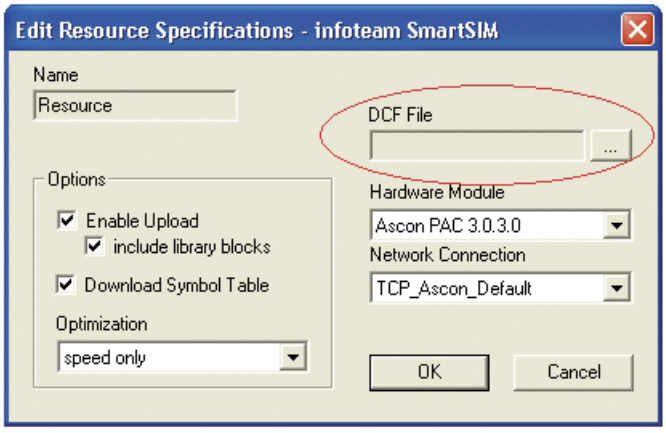
The CANopen Configurator or Configuration Tool is a special software tool that enables design and management of CANopen networks, interconnection of inputs and outputs on various devices, as well as configuration of network parameters. Furthermore, the CANopen Configurator is used to connect network variables of a PLC program to the corresponding inputs and outputs on the CANopen I/O module.

The CANopen Configurator is, right now, a separate software tool and not included in the delivery contents of the OpenPCS IEC61131 programming system.

8-2 Insert a DCF-file into OpenPCS

If your hardware supports CANOpen, you can insert a DCF-file into your OpenPCS project with

the dialog Edit resource :



8-3 Ascon I/O Function Blocks

An undemanding and yet less efficient method of managing the CANopen network is to do things manually and, having all the network in mind, edit a .DCF file, with the rules and formats stated by CiA in the document Cia DS306.

The task is made more friendly by using the Ascon I/O Function Blocks (FBs) directly from OpenPCS programming tool.

Ascon I/O FBs have two main purposes:

- Make accesses to I/O modules both for read / write field values and parameters / configuration data
- Mask the CANopen communication protocol structures and controls.

Ascon I/O FBs are based on the standard FBs included in the communication library, as described in:
CANopen Extension for IEC61131-3 – User manual – Edition March 2005 – Systec Electronic,
most of which are compliant with the:
CiA specification CiA Draft Standard 405 – CANopen Interface and Device Profile for IEC61131-3 Programmable Devices – version 2.0 .

The I/O accesses are made by CANopen Master agent, that is resident on the CPU. To make it easy, the CANopen master uses a series of basic Function Blocks, such as Send/Receive SDOs/PDOs, etc.

By the way, these FBs are also used by the more structured ASCON I/O FBs. Nevertheless the User may want to use the basic FBs.

The user can find it in the standard OpenPCS libraries of FBs.

We recommend however the use of ASCON I/O library of Function Blocks.

Appendix A
Reference documents

[1] OpenPCS 5.4.4 – User Manual.

[2] IEC 61131-3: Programming Industrial Automation Systems – Karl-Heinz John, Michael Tiegelkamp - Springer.

[3] CANopen Extension for IEC61131-3 – User manual – Edition March 2005 –
Systec Electronic.

[4] CiA DS 405 V2.0: CANopen Interface and Device Profile for IEC61131-3 Programmable Devices.

[5] CiA 301 DSP V4.1: CANopen application layer and communication profile

[6] CU-02 Installation manual (code: J30 - 658 - 1ACU-02 E).

[7] CU-02 User manual (code: J30 - 478 - 1ACU02 E).

[8] sigma**due** I/O modules Installation Manuals: DI-16LV, DI-32LV, DO-04RL, DO-04TX, DO-08RL, DO-16TS, DO-16TP, DO-32TS, AI-02UI, AI-08HL, AO-08HL, DM-08TS, DM-16TS.

[9] sigma**due** I/O modules User Manuals: DI-16LV, DI-32LV, DO- 04RL, DO-04TX, DO-08RL, DO-16TS, DO-16TP, DO-32TS, AI-02UI, AI- 08HL, AO-08HL, DM-08TS, DM-16TS.

[10] Ascon Firmware Function Block Library.

[11] IEC 61131-3 Function Block Library.

[12] I/O Function Block Library.