Scilab/Scicos code generator for FLEX

From model to simulation to hardware in one click!

version: 0.50
January 18, 2011

About Evidence S.r.l.

Evidence is a spin-off company of the ReTiS Lab of the Scuola Superiore S. Anna, Pisa, Italy. We are experts in the domain of embedded and real-time systems with a deep knowledge of the design and specification of embedded SW. We provide embedded firmware based on the Erika Enterprise RTOS for small single and multicore microcontrollers, custom Embedded Linux distributions for dedicated hardware, and a configuration and schedulability analysis tool named RT-Druid.

Evidence Contact Info

Address:
Evidence Srl,
Via Carducci 64/A
Località Ghezzano
56010 S.Giuliano Terme
Pisa - Italy
Tel: +39 050 991 1122, +39 050 991 1224
Fax: +39 050 991 0812, +39 050 991 0855

For more information on Evidence Products, please send an e-mail to the following address: info@evidence.eu.com. Other informations about the Evidence product line can be found at the Evidence web site at: http://www.evidence.eu.com.

About Embedded Solutions S.r.l.

Embedded Solutions is an Italian company specialized in hardware and software design. Embedded Solutions develops electronic devices based on microprocessor and microcontroller architectures, and it offers the possibility of having not only the hardware and software design of a product, but also the prototyping of multilayer boards with SMT and PTH technologies. The application fields covered by Embedded solutions starts from small applications with 8 bit microcontrollers, up to complex applications involving 32 bit general purpose microprocessors.

Embedded Solutions Contact Info

Address:
Embedded Solutions Srl
Via Piave, 14
20020 COGLIATE (MI), Italy
Tel: +39 02 96460455
Fax: +39 02 96468648

For more information on Embedded Solutions Products, please send an e-mail to the following address: info@es-online.it. Other informations about the Embedded Solutions product line can be found at the Embedded Solutions web site at: http://www.es-online.it.

Contents

1 Acknowledgements

Evidence would like to acknowledge as the first thing in this document the work of Simone Mannori (INRIA, FR), Roberto Bucher (SUPSI Lugano, CH), Mauro Marinoni, Nicola Serreli, Tullio Facchinetti, and Gianluca Franchino. These people really made a huge effort to let this toolchain work on the FLEX board, spending nights and weekends to propose you a working system which is fairly easy to use.

This document is just a first step toward a definitive documentation, so please excuse us for any imprecision or error in this document, and please report us any problem to let us correct them as soon as possible.

The Evidence Srl Team.

2 Introduction

This page contains the information related to the Scilab/Scicos code generator for the FLEX board.

The main idea is to *develop a single-click digital control automatic code generation tool for* FLEX.

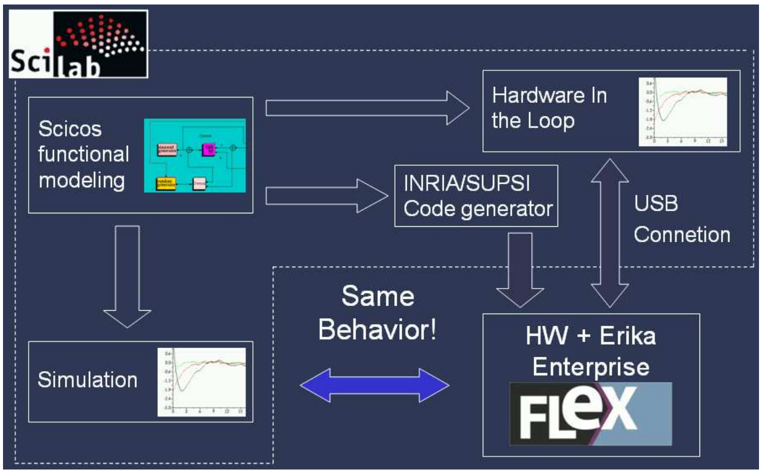

Figure 2.1: Scilab/Scicos development flow for ERIKA Enterprise and FLEX.

The envisioned design flow, depicted in Figure 2.1, is composed by the following steps:

1. Design of a control system in Scicos;
2. Simulation and tuning of the control system in Scicos;
3. Single-click code generation for ERIKA Enterprise for FLEX;
4. Automatic flashing of the FLEX board;
5. Integration in the Scicos Hardware In the Loop (HIL) support using the FLEX USB/wireless connection.

To use The Scilab/Scicos code generator you need at least the following hardware and software:

- A FLEX Board;
- Erika Enterprise (this document has been written by using Erika Enterprise GPL version 1.4.3);

- Microchip MPLAB IDE, and the Microchip C30 compiler (available from the Microchip web site);
- A Microchip ICD2 or any device which can be used to program the FLEX board.

Note: The need for a programmer will be removed soon, because the FLEX Full version will host a USB programmer on board.

3 Notes for Windows XP and Windows Vista users

If you are using Windows, and especially if you are using Windows Vista, please look carefully at the following warnings:

Warning: Do NOT install the Evidence package in a name containing spaces. `c:/Evidence/Evidence` works.

Warning: Do NOT install the Scilab package in a name containing spaces. `c:/Evidence/scilab-4.1.2` works.

Warning: If using Vista, be aware that directories like `c:/Programmi`, `c:/Users/Documenti` are not REAL directories but are aliases. DO NOT USE THEM. Put your RT-Druid workspace under `c:/Users/yourusername/workspace`.

4 Install steps

To install the Scilab/Scicos code generator toolchain, please follow the steps described in the following paragraphs:

1. If you have installed a previous version of Erika Enterprise, please save any modification done to the Evidence install directory, which is typically stored in the directory `c:/Evidence/Evidence`;

2. If you have installed a previous version of Erika Enterprise, please uninstall it by pressing on the `Uninstall` menu item in the in the `Evidence` menu. Then, please remove by hand everything which has been left on your `Evidence` directory.

3. Download Erika Enterprise 1.4.3 from the Evidence web site at the following URL: `http://erika.tuxfamily.org/scilabscicos.html`.

4. Install Erika Enterprise 1.4.3 as explained in the documentation available on the Evidence web site, and in particular in the Erika Enterprise Tutorial for dsPIC (R) DSC. Use `c:/Evidence/Evidence` as install directory.

5. Please follow exactly the instructions at the previous point. In particular, please remind to install the Microchip MPLABIDE and the C30 Compiler before installing Erika Enterprise.

6. Download the `scicos_pack_v6.zip` from the Evidence web site.

7. Unzip the file on the desktop. A set of directories are created: `scicos_examples`, `scilab`, and `user`. You need to copy these directories in various positions on your hard disk.

8. Please check that the file `c:/Evidence/Evidence/bin/rtd_config.properties` contains meaningful settings for your installation. Possible settings are explained in the comments in the properties file. In particular, you should specify the correct location of the Microchip C30 and ASM30 tools. The following is an example for an italian installation:

   ```
   preference_pic30__path_for_asm_compiler = <same line>
     C:\\Programmi\\Microchip\\MPLAB ASM30 Suite
   preference_pic30__path_for_gcc_compiler = <same line>
     C:\\Programmi\\Microchip\\MPLAB C30
   ```

   Please note to use a double backslash and not a single backslash!

Warning: If you are using Windows Vista, put the REAL directory here. For example, in Italian Vista installations `c:/Programmi` is an alias to `c:/Program Files`. Please use `c:/Program Files` and not `c:/Programmi`.

9. Download and install Scilab 4.1.2 binary version for Windows from the web site http://www.scilab.org. Please use an install directory **which does not contain spaces**: `C:/Evidence/scilab-4.1.2` is ok.

10. Erase the `contrib` directory inside your Scilab installation, and replace it with the `scilab/contrib` directory you just unzipped. This step copies the code generator and the palettes for FLEX inside the Scicos install directory.

11. Copy the content of the `user` directory inside your `c:/Documents and Settings/username` directory.

Warning: For Windows Vista users: read `c:/Users/username`

This step will add a `.scilab` file inside the Scilab environment directory. The new `.scilab` is used to display the palettes of the code generator for FLEX.

12. Copy the `scicos_examples` directory in a useful place, e.g. `c:/`.

13. Your Scilab/Scicos installation is now ready to produce code.

5 Your first Scicos application

This Chapter will guide you to the creation, compilation and execution of a first simple Scicos example on a FLEX board. The example created in this tutorial can be found in the directory `scicos_examples/led_sin`.

If you are looking for a prebuilt example, go directly to Section 5.2.

5.1 Creating the Scicos example files

1. Please start Scicos 4.1.2 from the `Start` menu. The Scilab window appears.
2. Type "`scicos();`", as showed in Figure 5.1, and press Enter.

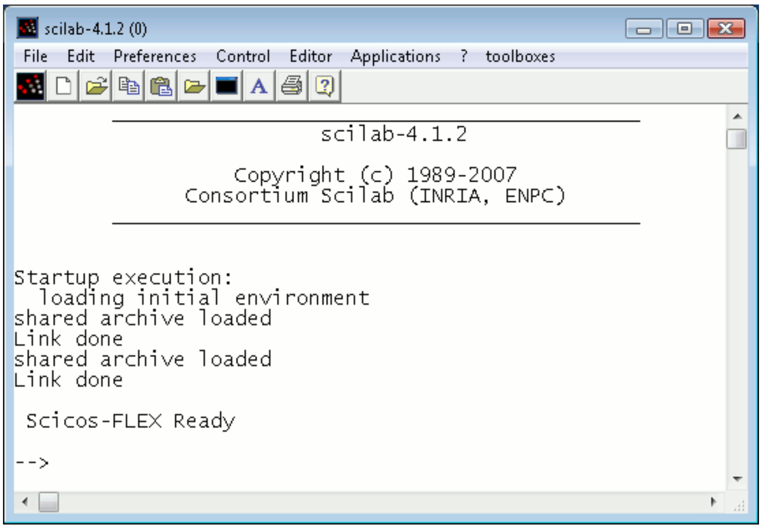

Figure 5.1: The Scilab splash screen. Type `scicos();` to start Scicos.

3. The Scicos windows appears, as showed in Figure 5.2.
4. Select `Palettes` from the `Edit` menu, as showed in Figure 5.3.
5. A little list appear in place of the menu. Select `FLEX-Sinks`, as showed in Figure 5.4.
6. A windows appears, with some sink blocks specific for the FLEX board (see Figure 5.5) (a complete list of the available blocks is available at the end of this document).

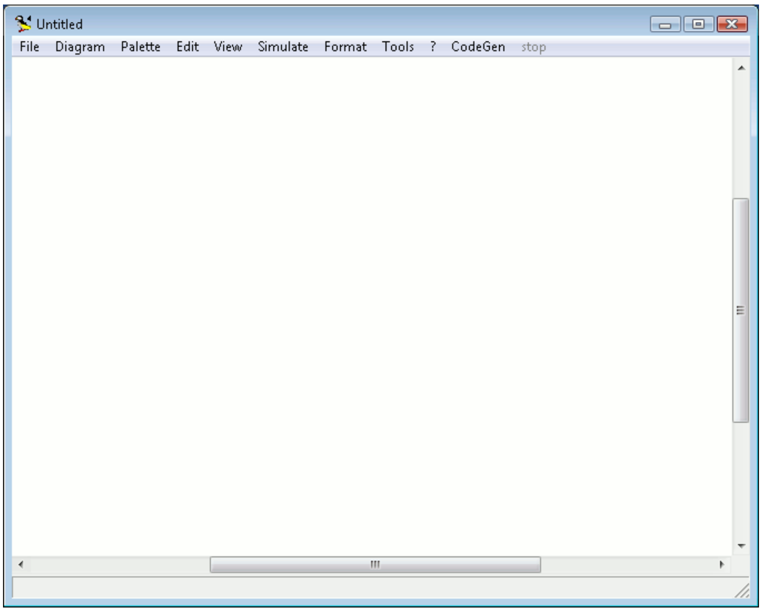

Figure 5.2: The Scicos splash screen.

7. Single click on the `FLEX-LED` block. The window selection moves to the Scicos window. The mouse now becomes a white rectangle of the dimension of the `LED` block. Single click somewhere in the white part of the window. A `LED` block is dropped in the diagram, like in Figure 5.6.

Note: If you need to move a block, go over it with the mouse, press `m`, then move the block and click on the new position!

Note: If you need to delete a block or a line, go over it with the mouse, then press `d`!

Note: If some garbage appears on the diagram windos, don't panic! Just press `r`!

8. Open the MCHP16-Sources palette, and repeat the same with the `Sine` block, placing it on the left of the `LED` block, as in Figure 5.7.

9. Link the black triangle of the `Sine` block to the black triangle of the `LED` block. To do that, press `l`, then single click on the triangle of the `Sine` block (the *source*), then click again on the triangle of the `LED` block (the *sink*). See Figure 5.8.

10. From the `MCHP16-Sources` Palette, which can be found il the palette list (see Figure 5.4. Choose the red clock, and put it on the diagram as shown in Figure 5.9.

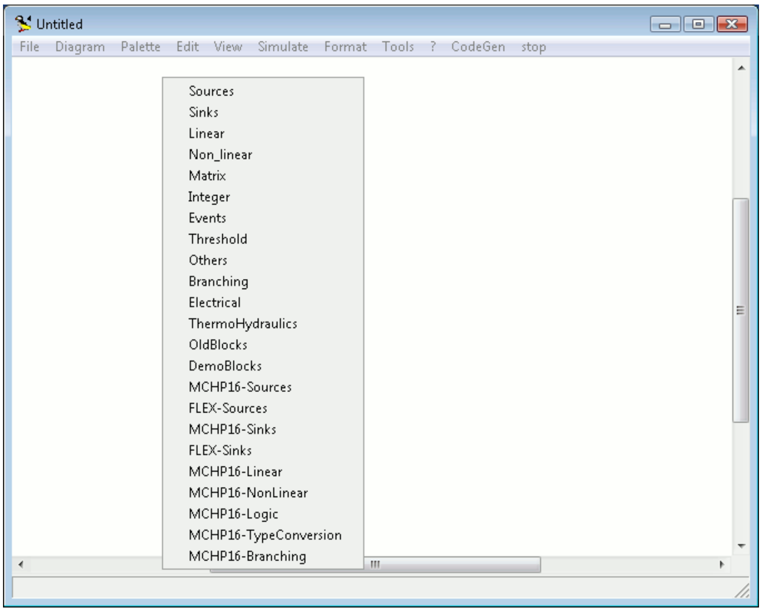

Figure 5.3: The `Edit` menu. Select `Palettes`

11. Now connect the clock signal to the two blocks. To do that, single click on the red triangle of the `clock` block, then single click below it, then single click over the `Sine` block, then click on the red triangle of the `Sine` block. After that, single click on the line below the `clock` block, then over the `LED` block, then on the red triangle of the `LED` block. The result is shown in Figure 5.10

12. Single click on the `Clock` block. Its properties window appears. Leave them untouched, and press `OK`. You can do the same on the `Sine` block. Figure 5.11 and 5.12 show these windows.

13. The code generator can produce code which only comes from a special block named *Super Block*. For this reason, we need to create a Super Block enclosing the `Sine` and the `LED` blocks. To do that, select the `Region to Super Block` menu item from the `Diagram` menu (see Figure 5.13). Then, draw a selection which includes the `Sine`, the `LED`, and the red lines in a way that only *one* red line exits the selection, as shown in Figure 5.14.

14. As a result, a Super Block is created (see Figure 5.15), which contains the `Sine` and `LED` blocks. To see these blocks, just single click on the Super Block, and another window will appear (see Figure 5.16). Please note that this window is very similar to the previous one except that the `clock` object is substituted by a placeholder signed with the number 1.

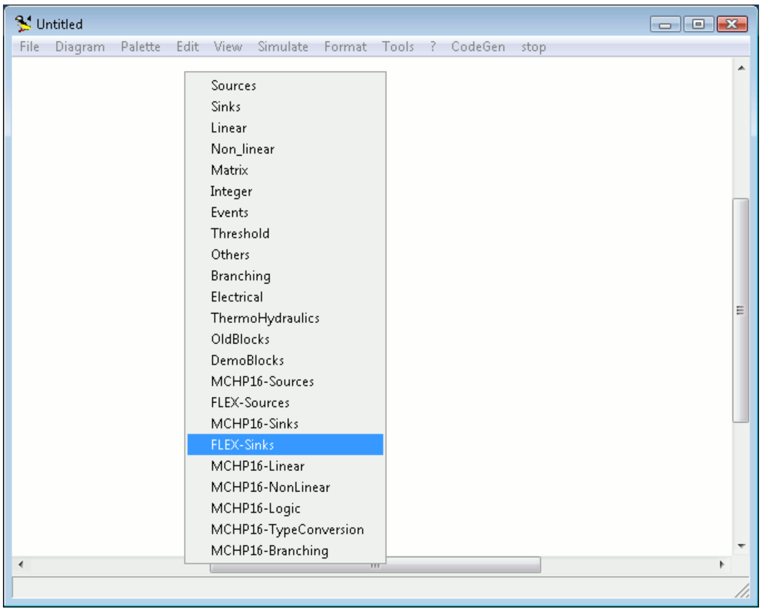

Figure 5.4: The Palette list.

Note: The Diagram containing the Super Block is disabled when the Super Block diagram is displayed. Only one window can be enabled at a time in Scicos. The limitation will be removed in the next version of Scicos.

15. It is now time to save the two diagrams. From the `File` menu, choose `Save as`. Save the diagram containing the Super Block as `led_sin.cos`.

5.2 Generating dsPIC code from a Scicos Diagram

It is now time to generate the code for the example we just created.

Note: A copy of the file created in the previous steps is included inside the `scicos_examples/led_sin` directory. To open it, double click on the `scicos_examples/led_sin/led_sin.cos` file.

1. Select `EmbCodeGen` from the `CodeGen` menu (see Figure 5.17).
2. A window appear, like the one in Figure 5.18. You can specify the directory where all the files will be created by modifying the `Created files Path` textbox. Please leave the other options unchanged.
3. Press `Ok`. As a result, a set of files are generated in the output directory.

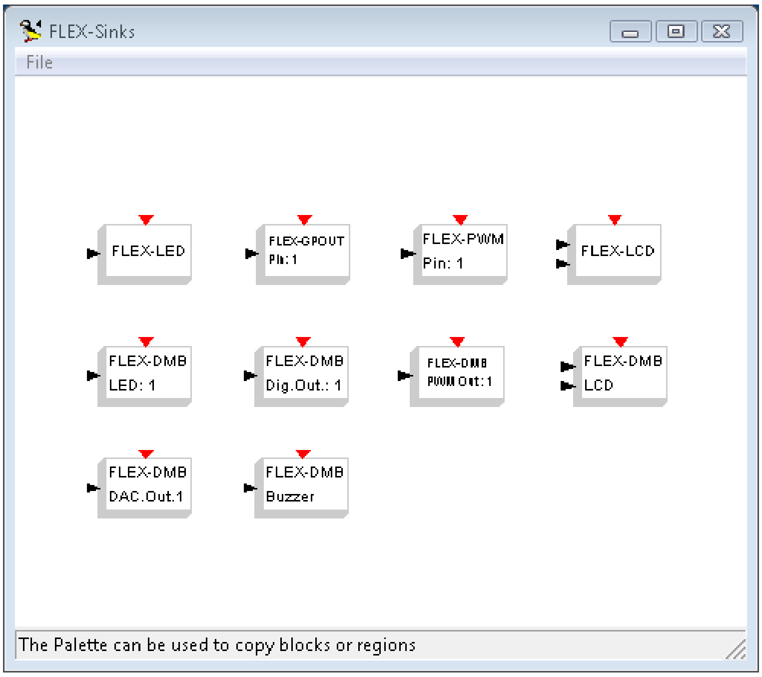

Figure 5.5: The dsPIC Palette.

4. Then, Scicos automatically opens a console window, running in it the following commands:
   - the RT-Druid template generator to instantiate the Scicos template application;
   - the RT-Druid standalone code generator to produce the Erika Enterprise configuration files from the generated OIL file;
   - the `make` application to compile the code.
5. The result of the code generation process is depicted in Figure 5.19. The executable file is named `pic30.elf` and it is located inside the `Debug` directory as usual for all the Erika Enterprise applications.

Warning: If the error depicted in Figure 5.20 appears, it is likely that you have an expired license of the Microchip C30 compiler Student Edition.

Try putting to `true` the following two lines inside the `c:/<programfiles>/Evidence/bin/rtd_config.properties` file, to select the Evidence compiler recompiled from the sources.

```
preference_pic30__use_evidence_compiler_4_deps = true
preference_pic30__use_evidence_compiler_4_compile = true
```

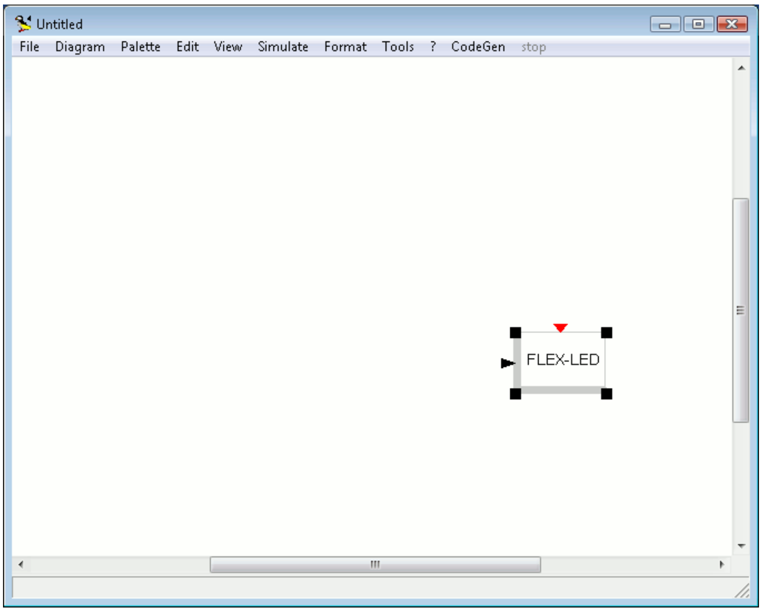

Figure 5.6: The `LED` block is dropped in the design window.

6. You can now program your application on your FLEX board. To do that, you need to open MPLABIDE as you usually do to program other Erika Enterprise applications. Please refer to the Erika Enterprise tutorial for dsPIC (R) DSC for more information.

Warning: If the ELF file fails to import on the MPLABIDE, try to use the C30 compiler recompiled from the Microchip sources by setting the following variables in the file `c:/<programfiles>/Evidence/bin/rtd_config.properties`:

```
preference_pic30__use_evidence_compiler_4_deps = true
preference_pic30__use_evidence_compiler_4_compile = true
```

7. Running the code on your FLEX board has the following behavior: the system led on the board flashes with a period of 20 seconds, and a duty cycle of around 6 seconds over 20. The explanation is the following:
   - The system works like a synchronous control system, with a sampling frequency of $0.1s$ (see Figure 5.11).
   - The `Sine` block output is a sinus with a frequency of 0.05, which correspond to a period of $20s$ (see Figure 5.12).
   - The `LED` block is directly linked to the system led, and is programmed to put on the system led when its input is greater than 0.5.

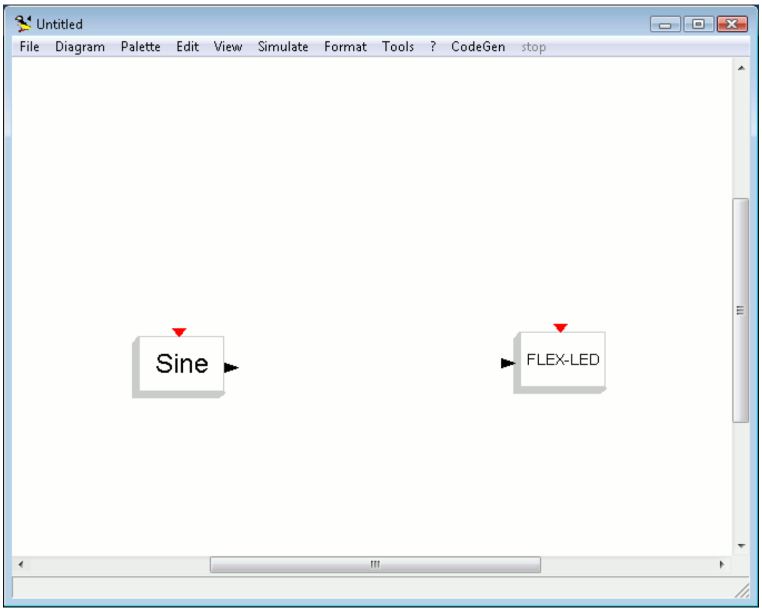

Figure 5.7: Place the `Sine` block to the left of the `LED` block.

- Looking at Figure 5.21, it is clear that the sinus has a value greater than 0.5 for around a third of its period. Given that, the system led is on for around 6 seconds over 20.

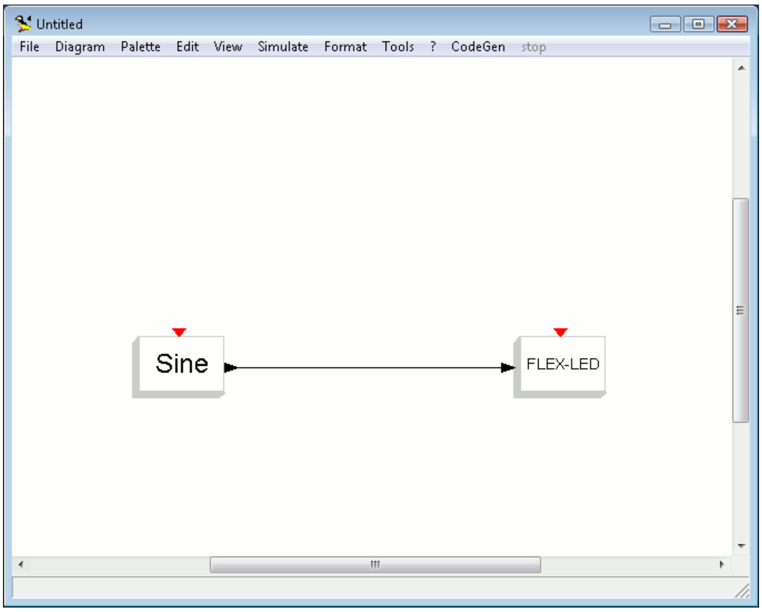

Figure 5.8: `Sine` and `LED` are now linked.

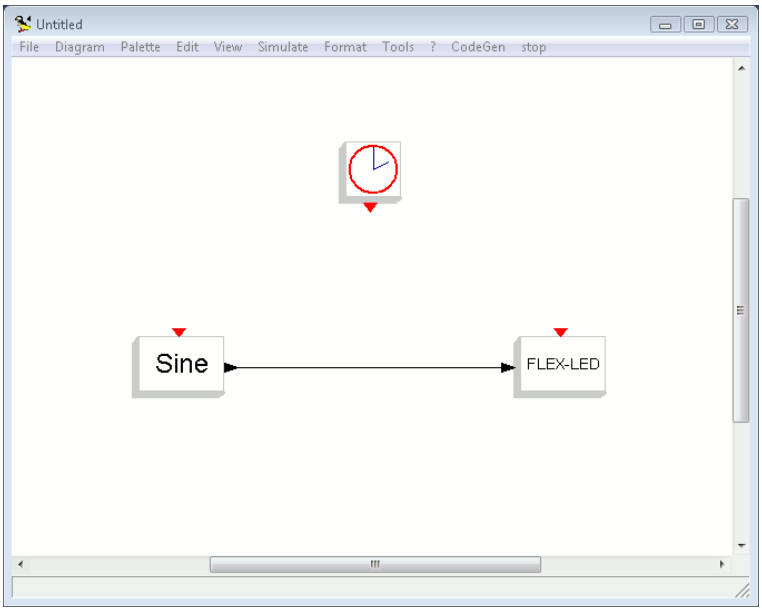

Figure 5.9: Put the `Clock` block over the `Sine` and `LED` blocks.

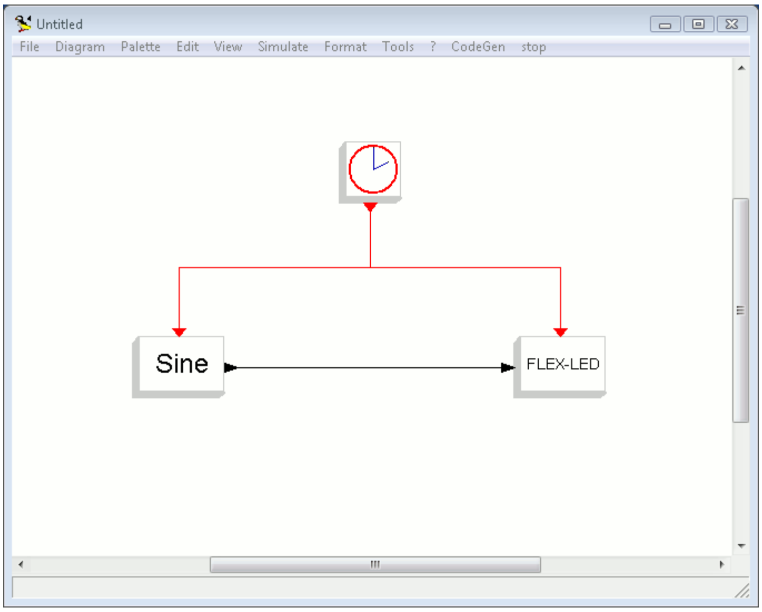

Figure 5.10: The `Clock` block is connected to the `Sine` and `LED` blocks.

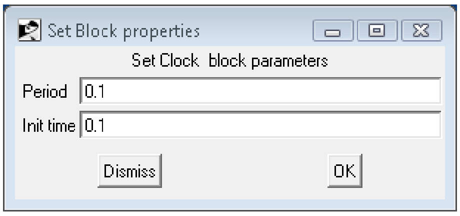

Figure 5.11: The `Clock` block properties.

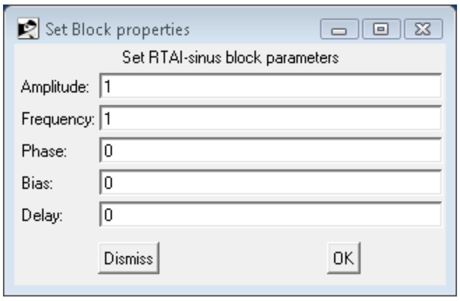

Figure 5.12: The `Sine` block properties.

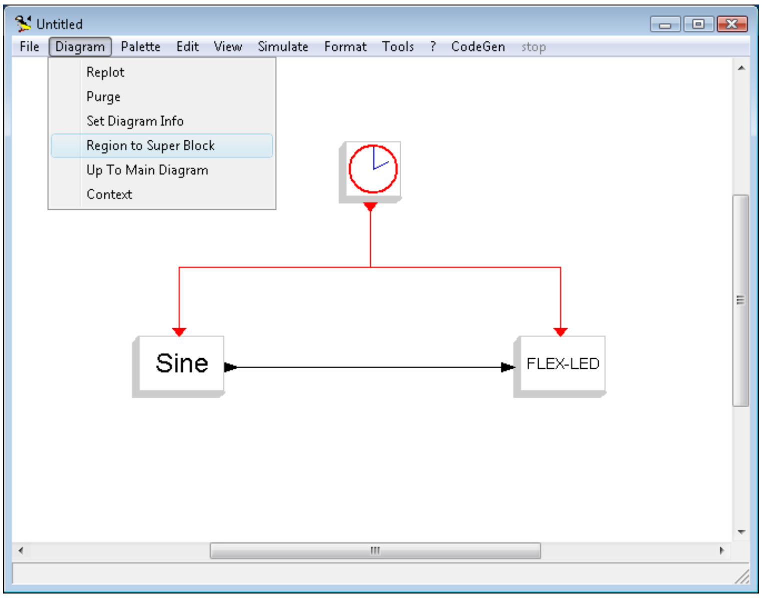

Figure 5.13: The Region to Super Block menu item.

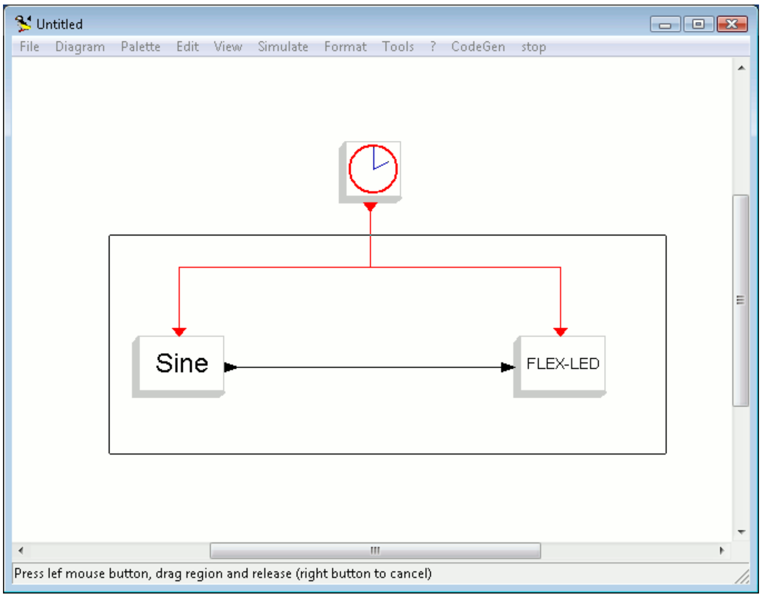

Figure 5.14: The selection made to create a Super Block.

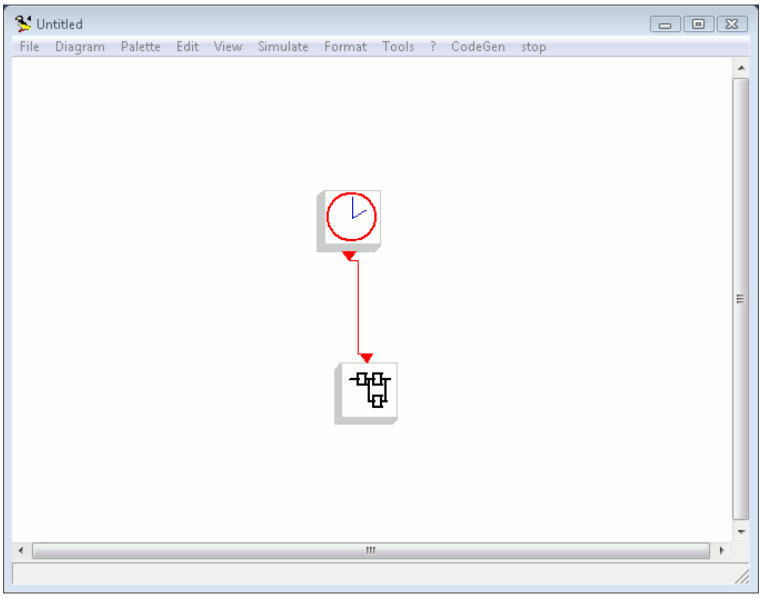

Figure 5.15: The Super Block.

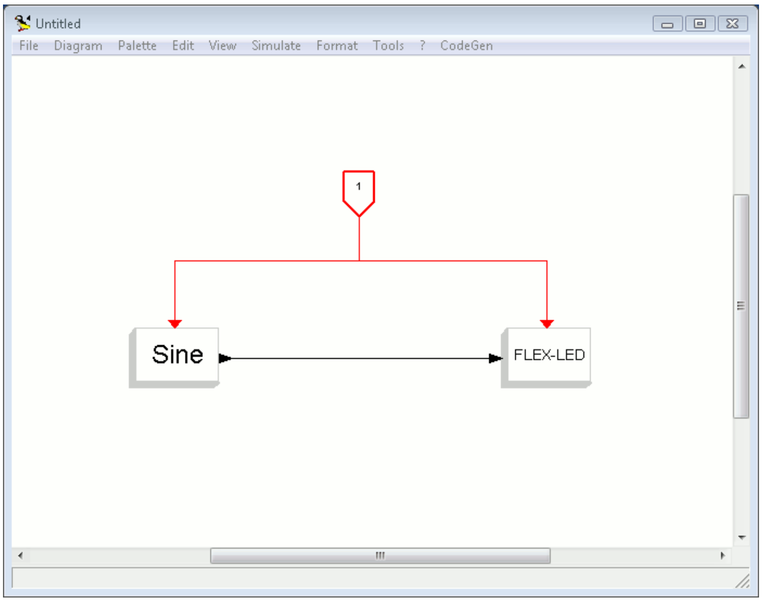

Figure 5.16: The contents of the Super Block.

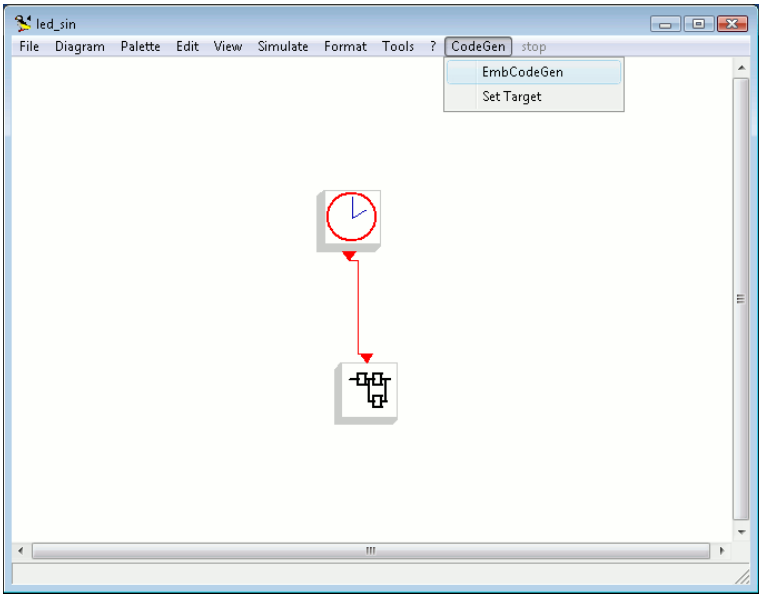

Figure 5.17: The `CodeGen` menu.

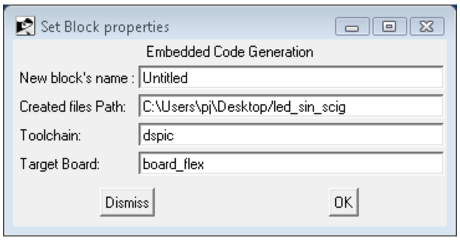

Figure 5.18: The `EmbCodeGen` dialog box.

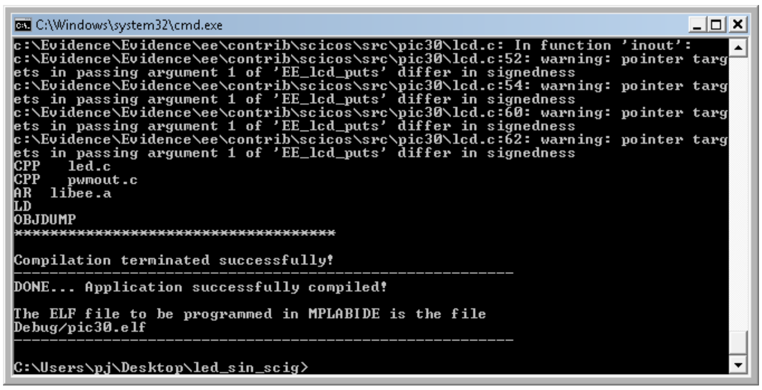

Figure 5.19: The compilation console.

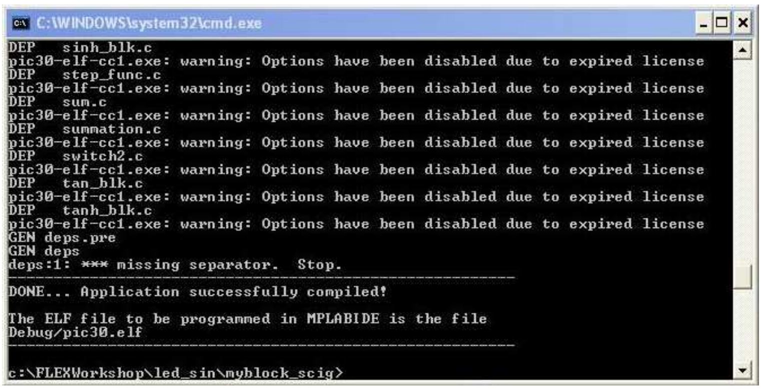

Figure 5.20: License problem when compiling.

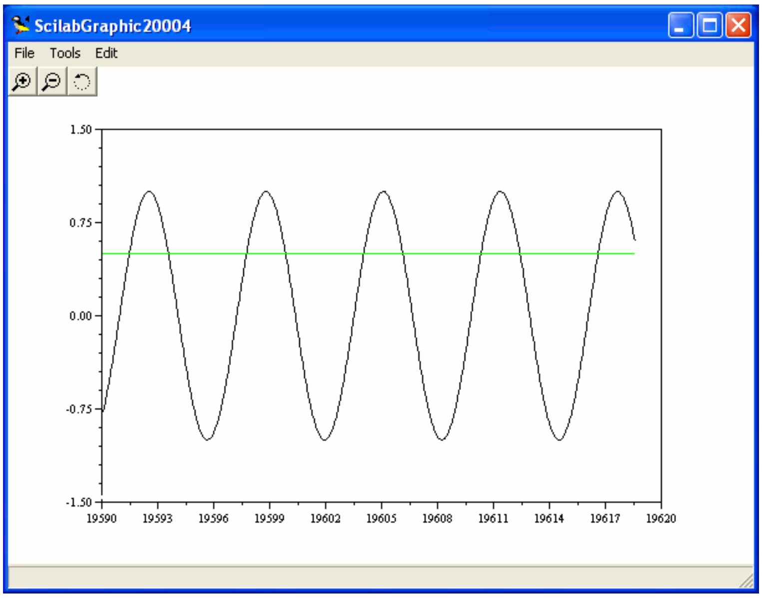

Figure 5.21: A graphic of a Sine and of a constant value 0.5.

6 Internals of the genaretd code

6.1 Templates and customization of the generated application

The default application wich is generated by the Scicos embedded code generator for dsPIC (R) DSC generates a basic application which uses Erika Enterprise with the FP kernel, a periodic task and an Alarm triggered by a timer interrupt to activate it.

In general, it is likely that advanced users would like to customize the application which is generated by the code generator, to add other activities to be executed concurrently with the code generated from the Scicos design. Examples of this activities could be for example background activities for reporting, supervision, display, debug, and so on.

Implementing such variations is very easy, because the application scheleton used by the code generator is contained inside a RT-Druid template. In particular, the default template is the `pic30_empty_scicos` template stored inside the `examples/pic30/pic30_scicos` directory under the Erika Enterprise install tree. The user can add a new template using the following steps:

1. Copy the `examples/pic30/pic30_scicos` directory in another location under the `examples` directory;
2. Change the `ID` of the template by modifying the `template.xml` file contained inside the directory. The `ID` is specified in the second line of the XML file as follows:

   ```
   <evidence_example version="1" ID="pic30_empty_scicos">
   ```
3. Change the files included in the new template. If you need to add a new file, please remember to add it in the corresponding list in the `template.xml` file.

Finally, specify the new template when generating the code in the `Template` textbox in Figure 5.18.

6.2 Assumptions of the default template

The code generated by the Scilab/Scicos code generator for FLEX uses the template named `pic30_empty_scicos`, and has the following symplifing assumptions:

1. There is a single sampling time T_s in the system;
2. T_s is forced to 1 ms;

3. Every sampling time specified by the user under the Scicos design will be rounded to a multiple of a millisecond;
4. An Erika Enterprise counter is linked to the a periodic timer;
5. The periodic timer used in the dsPIC hardware is set to raise an interrupt every 1 ms;
6. An Erika Enterprise alarm is attached to the counter, to periodically activate a task;
7. The task body just calls the routines generated by the Scicos code generator. Which executes the functions you specified in the design;
8. The PWM object has a fixed period of 1 ms. This means that if the sampling period is a multiple of T_s, then the PWM will repeat the same duty cycle until the PWN value is changed;
9. The A/D converter always works "on demand", meaning it always executes the following steps:
   - selects a channel;
   - starts the conversion;
   - waits for the end of the conversion (typically max $10\mu sec$)
   - it converts the result in a value from $0.0V$ and $3.3V$
10. To speedup the compilation process, the default configuration does not produce the dependency files and the `.src` file from every `.c` file.

6.3 Palette descriptions

The are more than 100 Scicos blocks available for the FLEX boards. A previous version of this manual included a screenshot and a short description for each one. Since the number of blocks and its descriptions are an ongoing work, we moved all this Section inside the Evidence Wiki Page under the Community Section of the Evidence website.

7 History

Version	Comment
0.10	Initial revision.
0.20	Updated document including the informations for the Scicos v2 Pack, Scilab 4.1.1, and including a basic step-by-step tutorial.
0.21	Added template selection. Updated infos for Erika Enterprise 1.4.1. Corrected typos.
0.22	Added some notes on the installation procedures. Updated infos for Scicos Pack v4. Corrected typos.
0.30	Support for Scilab 4.1.2 and the Demo Daughter Board.
0.40	Updated screenshots. Scilab pack v6.
0.50	Removed Scicos block descriptions (moved to evidence Wiki).