USING SCILAB/SCICOS WITH RTAI-LAB

Roberto Bucher
University of Applied Sciences of Southern Switzerland
Galleria 2, Manno 6928, C H
roberto.bucher@supsi.ch

Abstract

This paper describes how to implement a controller for the Rtai-Lab environment using Scilab/Scicos. In the second part of this paper the generation of user specific IO blocks is presented.

1 Introduction

Computer Aided Control System Design (CACSD) subsumes a broad variety of computational tools and environments for control system design, real time simulation, with and without hardware in the loop, making the best use of high desktop computer power, graphical capabilities and ease of interaction with low hardware cost. Integrated CACSD software environments allow an iteractive control system design process to be automated with respect to multi-objective performances evaluation and multi-parameter synthesis tuning. Visual decision support provides the engineer with the clues for interactively directing an automated search process to achieve a well balanced design under many conflicting objectives and constraints. Local/remote on line data down/upload makes it possible a seamless interaction with the control system, in order to supervise its operation and to adapt it to changing operational needs.

2 A simple example

2.1 Scheme

In the following, a simple example will be analized in both environment, Matlab/Simulink and Scilab/Scicos. The system is represented by a transfer function

$$Gs(s) = \frac{20}{s^2 + 4s}$$

with unity feedback,
In both cases the system has been implemented as discrete-time transfer function

$$Gz(z) = 10^{-6}\frac{9.987z + 9.973}{z^2 - 1.996z + 0.996}$$

with a sampling time of $1ms$. Different signals are sent to scopes, meters, and leds.
The model is saved with the name "test".

2.2 Implementation under Scilab

First of all we have to design the system using scicos. Figure **??** represents the Scicos scheme of the example.

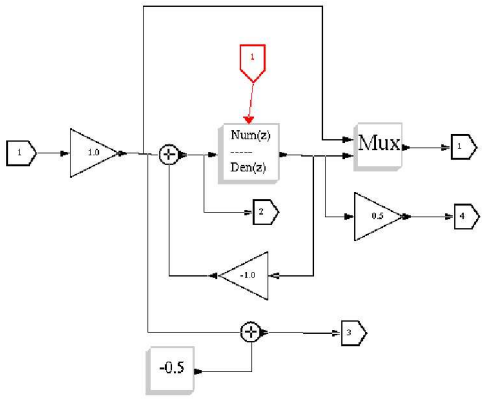

FIGURE 1: *Scicos scheme*

In order to generate the code this scheme must be transformed into a "Super Block" which can be used to generate the code. The menu "Diagramm" "Region to Super Block" allows to select a part of the scheme and to put it into a "Super Block". We can access to the Super Block scheme simply by clicking on it. A good idea is to open the "Super Block" and to rename it ("Diagram" "Rename") to "test". This will be the name of the generated model and of the directory where the generated files are stored.
At this moment is very important to save the scicos scheme!
Now we can simply choose the menu "Object" "Code Generation" to generate the source code.
The result is a compiled function into the scheme (Figure **??**).

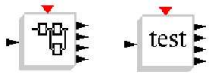

FIGURE 2: *Super block before and after the compilation*

Exit scicos and scilab now without saving the scheme, otherwise you can't no more modify it!

The next steps are to be executed outside of the Scilab environment. First, the user must declare the input and the 4 outputs of the system using a configuration file. The file "config" needed by the example is:

```
0.001
rtai_scope out 1 2 IO 0 0 0 0 0
square inp 1 0 0 1.0 10.0 5.0 0.0 0.0
rtai_scope out 2 1 U 0 0 0 0 0
rtai_led   out 3 1 LED 0 0 0 0 0
rtai_meter out 4 1 METER 0 0 0 0 0
end
```

The first line of the config file give the sampling time of the system. The following lines describe the I/O blocks.
Using the utility "gen_io" the file "test_void_io.c" generated by scicos will be modified into the file "test_io.c", which includes now the calls to the I/O functions.
The last step is given by the compilation of the "test_standalone" executable file, which is performed with the following command:

```
make -f test_Makefile test_standalone
```

3 I/O Blocks

3.1 Basics

I/O blocks are implemented in a library (libsciblk.a). There are 3 kind of I/O blocks:

1. Blocks which can be connected to an input or an output port (ex. rtai_comedi_data)
2. Blocks which can be connected only to an input port (ex. step)
3. Blocks which can be connectes only to an output port (ex. rtai_scope)

For each block in the config file the following (block dependent) items have been defined:

- the type (example: rtai_scope)
- input (inp) or output (out) port (where the block is connected)
- the port number to which the block is connected
- an identifier, a channel number or a maximal number of channels/signals for this block
- a name
- 5 parameters

3.2 Available blocks

The following IO Blocks have been implemented:

sinus creates a sinus input signal

square creates a square inpit signal

step creates a step input signal

rtai_scope sends signal(s) to the Rtai-Lab scope widget

rtai_led sends signal(s) to the Rtai-Lab leds widget

rtai_meter sends signal to the Rtai-Lab meter widget

mem allows to connect an output port to an input port of the scicos modul

rtai_comedi_data allows to connect COMEDI drivers for analog signals to the scicos modul (input or output)

rtai_comedi_dio allows to connect COMEDI drivers for digital signals to the scicos modul (input or output)

extdata gets the data from a file and creates a periodic input signal

The tables 1 . . . 11 describe the different meanings of the parameters for each block.

The following modules are not contained in the official RTAI 3.x release:

pcan allows to connect the scicos modul to a Maxon driver through a bus can using a Peaks epp-dongle

cioquad4 build an input signal getting data from a Computer Boards CIOQUAD4 digital encoder card

The tables 12 . . . 14 show the parameters of IO Block not contained in the official RTAI 3.x release.

io	inp
port	port Nr.
ch	not used
name	not used
p1	Amplitude
p2	Frequency
p3	Phase
p4	Bias
p5	Delay

Table 1: Parameters for the IO block: ***sinus***

io	inp
port	port Nr.
ch	not used
name	not used
p1	Amplitude
p2	Period
p3	Impulse width
p4	Bias
p5	Delay

Table 2: Parameters for the IO block: ***square***

4 Implementation of new user blocks

4.1 Implementing the code for a IO device

Different tools facilitate the implementation of new blocks. In order to implement new drivers some skeleton are provided:

- "template.c"
- "devtmpl.h"

Using the command

```
gen_dev <model>
```

a file "<model>.c" will be created. This file contains all the needed functions to implement the driver as "input" and "output" driver. The utility "gen_dev" fills the file "devices.h" too.
The file "devstruct.h" contains the description of the structure used to store the block specific datas.

```
typedef struct devStr{
  int nch;
  char IOName[20];
  char sName[20];
  double dParam[5];
  int i1;
  void * ptr;
}devStr;
```

A structure for input (inpDevStr) and a structure for output (outDevStr) are provided in "rtmain.c". Basically, the channel information can be stored under the field "nch", the name under "sName" and the 5 parameters p1...p5 into the field "dParam". The field IOName is used to store the name of the IO Block needed by the online parameter modification.
The function interfaces in the generated file reflect the call implemented in <model_io.c> after calling "gen_io". Now the user can simply implement the code for the IO block.

io	inp
port	port Nr.
ch	not used
name	not used
p1	Amplitude
p2	Delay
p3	not used
p4	not used
p5	not used

Table 3: Parameters for the IO block: ***step***

io	out
port	port Nr.
ch	number of signals
name	Scope name
p1	not used
p2	not used
p3	not used
p4	not used
p5	not used

Table 4: Parameters for the IO block: ***rtai_scope***

4.2 Create the new library file

In order to generate the new library file with the new implemented IO Block, the user must modify the "Makefile.am" file, adding <model>.c to the list of the files to be compiled (libsciblk_a_SOURCES). The user have to launch "automake rtai-lab/scilab/devices/Makefile" from the RTAI root directory, followed by "./config.status". Now he can run "make" and "make install' in the scilab/devices directory. The libsciblk.a file will be created and copied in the library directory.

io	out
port	port Nr.
ch	number of leds (1 . . . 16)
name	Led name
p1	not used
p2	not used
p3	not used
p4	not used
p5	not used

Table 5: Parameters for the IO block: ***rtai_led***

io	out
port	port Nr.
ch	not used
name	Meter name
p1	not used
p2	not used
p3	not used
p4	not used
p5	not used

Table 6: Parameters for the IO block: ***rtai_meter***

io	inp or out
port	port Nr.
ch	id (0,1,2,...)
name	not used
p1	not used
p2	not used
p3	not used
p4	not used
p5	not used

Table 7: Parameters for the IO block: ***mem***

io	inp or out
port	port Nr.
ch	ch or number of signals (device dependent)
name	"comediX"
p1	range (device dependent)
p2	aref (device dependent)
p3	not used
p4	not used
p5	not used

Table 8: Parameters for the IO block: ***rtai_comedi_data***

io	inp
port	port Nr.
ch	ch or number of signals (device dependent)
name	"comediX"
p1	not used
p2	not used
p3	not used
p4	not used
p5	not used

Table 9: Parameters for the IO block: ***rtai_comedi_dio***

io	out
port	port Nr.
ch	ch or number of signals (device dependent)
name	"comediX"
p1	threshold
p2	not used
p3	not used
p4	not used
p5	not used

Table 10: Parameters for the IO block: ***rtai_comedi_dio***

io	inp
port	port Nr.
ch	number of values
name	file name
p1	not used
p2	not used
p3	not used
p4	not used
p5	not used

Table 11: Parameters for the IO block: ***extdata***