Targeting the Scicos Codegenerator for embedded applications

Ing. Roberto Bucher

Scuola Universitaria Professionale della Svizzera Italiana (SUPSI)

Paris - 1.7.2009
## Objectives

<table>
<thead>
<tr>
<th>(Prices in Euro)</th>
<th>Commercial Control design environment</th>
<th>Proposed solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core SW</td>
<td>1950.-</td>
<td>0.-</td>
</tr>
<tr>
<td>Control SW</td>
<td>1000.-</td>
<td>0.-</td>
</tr>
<tr>
<td>Graphical GUI</td>
<td>3000.-</td>
<td>0.-</td>
</tr>
<tr>
<td>Code generator</td>
<td>7500.-</td>
<td>0.-</td>
</tr>
<tr>
<td>RT target</td>
<td>3000.-</td>
<td>0.-</td>
</tr>
</tbody>
</table>
Objectives
Outline

1. Rapid Controller Prototyping
2. Linux RTAI
3. Scilab/Scicos and Linux RTAI
4. Implementation
5. Examples
6. Conclusions
Control Design Loop

specification

verification

implementation

simulation

modelling
identification

control
design

specification
Rapid Controller Prototyping - Requirements

- 2 main components
  1. An OS with hard real-time features
  2. A Computer Aided Control System Design environment including a code generator

→ Linux RTAI + RTAI-Lab + Scilab/Scicos
The Linux RTAI project

- Hard real-time extension to the Linux OS
- Based on the ADEOS pipeline (with some improvements!)
- Free Open Source Software (FOSS)
- Implementation of hard real-time controllers using general purpose hardware
- Same PC for the controller design, the hard real-time controller task and the soft real-time monitor task
- Hard real-time in kernel and user space
- Distributed control through the net_rpc module
Linux RTAI - Features

- Latency: $< 10 \mu s$ depending on the HW
- Typical sampling frequencies: $1 \ldots 10 kHz$
The GUI application - xrtailab
The GUI application - xrtailab

PC with HRT task and xrtailab
The GUI application - xrtailab
Systems controlled with Linux RTAI

Cycab (INRIA)

Biped Robot (INRIA)

http://www.inrialpes.fr/sed
National Radio Astronomy Observatory controlled with Linux RTAI - West Virginia

http://www.gb.nrao.edu/43m/
Add-ons for Scilab/Scicos

The link between Scilab/Scicos and Linux RTAI is implemented through:

- A code generator → RTAICodeGen_.sci.
- A `rtmain.c` (main file for the real-time task) specific for the Scilab/Scicos environment.
- An external "template Makefile"
- A new scicos palette `RTAI-Lib.cosf`.
- Macros for the new blocks (Scicos interface functions).
- A RTAI library specific for the new Scicos blocks (Scicos implementation functions).
The code generator used in RTAI is a modification of the standard Scicos code generator

- Only stand-alone generation with specific main procedure (rtmain.c)
- Sensors and actuators are integrated in the Scicos block diagram as specific custom peripheral blocks
- Each block can be identified by a name which is then referred in RTAI-Lab
The code generator

**Code generation**

```
Design

SCOPE
comedi0 CH-0
COMEDI A/D
COMEDI D/A
SCOPE

Sine

COMEDI D/A
comi0 CH-0

COMEDI A/D
comi0 CH-0

Scope
SCOPE

Sup

RT
AICodegen.

Co
Code

Compiling
and
rtmain.c
libsciblk.a
Scilab

lib

Ha

rd real-time stand-alone executable

RT
exec
```
Rapid Controller Prototyping

Linux RTAI

Scicos

Implementation

Examples

Conclusions

The code generator

Code generation

Design

Superblock
The code generator

Code generation

Design

Superblock

Code

RTAICodegen.sci
Rapid Controller Prototyping

Linux RTAI

Scicos

Implementation

Examples

Conclusions

The code generator

**Code generation**

- **Design**
- **Superblock**
- **Code**
  - rtmain.c
  - libsciblk.a
  - Scilab lib

Compiling and Link
Code generation

- Design
- Superblock
- Code
  - rtmain.c
  - libsciblk.a
  - Scilab lib
- Hard real-time stand-alone executable
- RT exec
The code generator

Code generation - Details

Superblock
The code generator

Code generation - Details

Superblock

rtai.gen
The code generator

Code generation - Details

- rtai.gen
  - Superblock
  - standalone.cmd
  - rtai.mak
The code generator

Code generation - Details

rtai.gen

Superblock

standalone.cmd

rtai.mak

model.c

model_Cblocks.c

Makefile
Target specific files - The “gen” file

- rtai.mak
- standalone.cmd
Target specific files - The "cmd" file

```plaintext
[CCode,FCode]=gen_blocks();
[Code,Code_common]=make_standalone42();
files=write_code(Code,CCode,FCode,Code_common);
Makename=rt_gen_make(rdnom,files,archname);
ok=compile_standalone();
```
Target specific files - The “mak” file

```
all: ../../../%MODEL%
...
CC = gcc
CC_OPTIONS = -O -DNDEBUG -Dlinux -DNARROWPROTO -D_GNU_SOURCE

MODEL = ../../../%MODEL%
OBJSSTAN = rtmain.o common.o ../../../%MODEL%%.o ../../../OBJ%

SCILIBS = $(SCIDIR)/libs/scicos.a $(SCIDIR)/libs/poly.a $(SCIDIR)/libs/calelm.a
OTHERLIBS =
ULIBRARY = $(RTAIDIR)/lib/libsciblk.a $(RTAIDIR)/lib/liblxrt.a

CFLAGS = $(CC_OPTIONS) -O2 -I$(SCIDIR)/routines -I$(SCIDIR)/routines/scicos $(C_FLAGS) -DMODEL=$(MODEL) -DMODELN=$(MODEL).c

rtmain.c: $(RTAIDIR)/share/rtai/scicos/rtmain.c $(MODEL).c
cp $< .

../../../%MODEL%%: $(OBJSSTAN) $(ULIBRARY)
gcc -static -o @ $(OBJSSTAN) $(SCILIBS) $(ULIBRARY) -lpthread $(CME) @echo "### Created executable: $(MODEL) ###"
```
The new Scicos palette

Sine
Scope
Step
Square
Led
extdata
test.dat
SENSOR
SENS
ACTUATOR
ACT
Mbx Send Ovw
MBX1
Mbx Rcv no blk
MBX1
Mbx Send if
MBX1
Mbx rcv blk
MBX1
COMEDI A/D
comedi0 CH–0
COMEDI D/A
comedi0 CH–0
COMEDI DI
comedi0 CH–0
COMEDI DO
comedi0 CH–0
SEM wait
SEM1
SEM signal
SEM1
C RTAI
Block
Specific blocks for RTAI-Lab

- Scope
- Meter
- LED
Specific blocks for RTAI-Lab

Scope
SCOPE

Meter
METER

LED
LED
Specific blocks for RTAI-Lab

- **Scope** (SCOPE)
- **Meter** (METER)
- **LED** (LED)
Specific blocks for RTAI-Lab

Scope
SCOPE

Meter
METER

LED
LED
Additional palettes

- Epos canopen sync
- Epos Analog 0x08 1
- INIT ENC 1s 0
- Epos encoder 0x08
- Epos motX 0x08
- signal means out.txt
- Epos motor 1 0x08
- Switch 0 1
- Pport_out 2 0x378
- Pport_in 15 0x378
The main file *rtmain.c*

3 threads:
- Main thread *rtMain*
- Hard RT thread *rt_BaseRate*
- Communication thread with the GUI client application *rt_HostInterface*
The \textit{rt\_BaseRate} thread

\begin{itemize}
\item\texttt{NAME(MODEL,\_init)();}
\item\texttt{WaitTimingEvent}
\item\texttt{NAME(MODEL,\_isr)(double t);}
\item End?
\item Y \quad \text{SOFT real-time}
\item N \quad \text{HARD real-time}
\item Y \quad \text{SOFT real-time}
\item\texttt{NAME(MODEL,\_end)();}
\end{itemize}
Adding new blocks

Adding a new block

Each new block needs

- An interface function (.sci)
- An implementation function (.c)
The Inverted Pendulum
The inverted pendulum

The Inverted Pendulum

Particular of the RF encoder
The Inverted Pendulum

The inverted pendulum
The Inverted Pendulum

- **Particular of the RF encoder**
- **M1**
- **i1**
- **φ1**
- **RF receiver**
- **Driver**
- **M1**
- **Encoder + RF sender**
- **PC with HRT controller**
- **CAN BUS**
- **SCOPE**
- **Safety block**
- **Real Plant**
- **LQR_controller**
- **Input_ref**
- **Swingup_controller**
- **Sine**
- **0.2**
- **1**
- **Switch**
- **ABS**

**The inverted pendulum**

**The Inverted Pendulum**
The Inverted Pendulum

- **RF encoder**
- **Driver**
- **CAN BUS**
- **PC with HRT controller**
- **Encoder + RF sender**

**Reduction of Order Observer**

- **Klqr**
- **y**
- **f**
- **u**

**Scicos scheme**

- **SCOPE**
- **Safety block**
- **Real Plant**
- **LQR controller**
- **Input ref**
- **Swingup controller**
- **Sine**
- **0.2**
- **1**
- **Switch**
- **ABS**
The Inverted Pendulum

The Scicos scheme

Switch

Safety block

Real Plant

LQR_controller

Sine

Swingup_controller

Scope

SCOPE

Input_ref

ABS

Switch

1
0.2

Safety block

Real Plant

LQR_controller

Sine

Swingup_controller

Scope

SCOPE

Input_ref
Ball on Beam
Ball on Beam
Ball on Beam

The Scicos scheme

- Extdata
- siginp.dat
- \(-K\)
- Scope
- SCOPE
- Ball on beam
- Controller + Osservatore
- 1
- MUX
- C + O Direct
- y = Cx + Du
- x_+ = Ax + Bu
- Sum
- C + O Feedback
- y = Cx + Du
- x_+ = Ax + Bu
- 3
- 3
- Saturazione
- Expression
- Mathemati...
- 0x02
- encoder
- Epos
- 0x02
- motor
- I
- Epos
- 0
- 0x02
- Epos Analog
- X0
- 1s
- INIT ENC
- PHI0
- 1s
- INIT ENC
- den(z)
- num(z)
- sync
canopen
Epos

Rapid Controller Prototyping

Linux RTAI

Scicos Implementation

Examples

Conclusions
Ball on Beam

The anti-windup controller

\[ x_{+} = Ax + Bu \]
\[ y = Cx + Du \]

Mux

\[ C + O \text{ Direct} \]

\[ \sum \]

Sum

\[ C + O \text{ Feedback} \]

Saturazione

1
Ball on Beam

The CANopen blocks
Videos

Sample Videos
Conclusions

- Open source!!
- Complete and stable.
- Able to solve complex control problems in a simple way.
- It runs parallel to a MATLAB/Simulink/RTW environment in my student laboratory
Questions?

Thank You

- www.rtai.org
- www.dti.supsi.ch/~bucher/scilab.html
- www.dti.supsi.ch/~smt/labO4.html