Usable Simulink Embedded Coder Target for Linux

Michal Sojka, Pavel Písa

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Control Engineering

16th Real-Time Linux Workshop
October 12, 2014
Düsseldorf, Germany
Motivation

▶ Not everybody is a programmer
▶ Control engineers can design control algorithms but cannot program them in C
▶ Solution: Matlab/Simulink + automatic code generation
▶ Control applications = real-time applications
  while (1) { read input; calculate control; write output }
▶ The generated code needs to run on RTOS, e.g. GNU/Linux
Motivation

- Not everybody is a programmer
- Control engineers can design control algorithms but cannot program them in C
- They use tools like Simulink to simulate their algorithms
- They still want to run their algorithms on real hardware, not just simulations

Solution: Matlab/Simulink + automatic code generation

Control applications = real-time applications

while (1) { read input; calculate control; write output }

The generated code needs to run on RTOS, e.g. GNU/Linux
Motivation

- Not everybody is a programmer
- Control engineers can design control algorithms but cannot program them in C
- They use tools like Simulink to simulate their algorithms
- They still want to run their algorithms on real hardware, not just simulations
- Solution: Matlab/Simulink + automatic code generation
Motivation

- Not everybody is a programmer
- Control engineers can design control algorithms but cannot program them in C
- They use tools like Simulink to simulate their algorithms
- They still want to run their algorithms on real hardware, not just simulations
- Solution: Matlab/Simulink + automatic code generation
- Control applications = real-time applications
  - while (1) { read input; calculate control; write output }
Motivation

- Not everybody is a programmer
- Control engineers can design control algorithms but cannot program them in C
- They use tools like Simulink to simulate their algorithms
- They still want to run their algorithms on real hardware, not just simulations
- Solution: Matlab/Simulink + automatic code generation
- Control applications = real-time applications
  - while (1) { read input; calculate control; write output }
- The generated code needs to run on RTOS, e.g. GNU/Linux
What is Simulink?

- Widely used tool for design and simulation of **dynamic systems**
- Commercial, **non-free** program – a part of Matlab
- Developed by MathWorks
- In its core is a **solver of differential equations**
- Algorithm is drawn as a **data-flow graph** (= Simulink model)
- Simulink can **simulate** it
- Simulink can **generate C code** of the model
Why people use it?
Example:

```c
double in1, out1;
in1 = adc_read(1);
out1 = sin(t) + 3 * in1;
dac_write(1, out1);
```
Code generation in Simulink

Example:

```c
double in1, out1;
in1 = adc_read(1);
out1 = sin(t) + 3 * in1;
dac_write(1, out1);
```

Simulink Coder & Embedded Coder

Code generation target
  ▶ Template for “main.c”
  ▶ Template for Makefile
  ▶ I/O blocks
Can Simulink generate code that runs on Linux?
Can Simulink generate code that runs on Linux?

Yes, but...
Problems of Simulink code generation targets for Linux

- Very good support for popular embedded boards (Raspberry Pi, BeagleBone, ...)

- Can only be installed on Windows hosts
- Other Linux targets are fortunately supported via "IDE Link"
- Uses Eclipse IDE to compile and run the code
- Slow!
- Generated code uses POSIX timers
- POSIX timers do not guarantee real-time properties (latencies) even under preempt_rt
Problems of Simulink code generation targets for Linux

- Very good support for popular embedded boards (Raspberry Pi, BeagleBone, ...)
- Can only be installed on Windows hosts 😞
Problems of Simulink code generation targets for Linux

- Very good support for popular embedded boards (Raspberry Pi, BeagleBone, ...)
- Can only be installed on Windows hosts 😞
- Other Linux targets are fortunately supported via “IDE Link”
Problems of Simulink code generation targets for Linux

- Very good support for popular embedded boards (Raspberry Pi, BeagleBone, ...)
- Can only be installed on Windows hosts 😞
- Other Linux targets are fortunately supported via “IDE Link”
  - Uses Eclipse IDE to compile and run the code 😞
- Slow! 😞
Problems of Simulink code generation targets for Linux

- Very good support for popular embedded boards (Raspberry Pi, BeagleBone, ...)
- Can only be installed on Windows hosts 😞
- Other Linux targets are fortunately supported via “IDE Link”
  - Uses Eclipse IDE to compile and run the code 😞
  - Slow! 😞
- Generated code uses POSIX timers
  - POSIX timers do not guarantee real-time properties (latencies) even under preempt_gettime 😞
POSIX timers under preempt_rt Linux

- POSIX timers use **signals** to notify user space about their expiration
- Signal delivery path uses **sleeping locks** under preempt_rt
- Deferred to **softirq**
- Softirqs are:
  - executed in the context of arbitrary task or ksoftirqd
  - executed with random (e.g. **non-real-time**) priority
Is there better code generation target?

Yes

ERT_LINUX
ERT_LINUX code generation target

- Goal: Minimalist target which reuses as much Simulink code as possible (not that easy)
- No Eclipse needed 😊
- Uses clock_nanosleep() for timing 😊
- External mode
- Works both natively and with cross-compiling

http://lintarget.sf.net
ERT_LINUX I/O support

- MF624: PCI data acquisition card
  - Digital IO, Analog IO, PWM out
  - Counters (can be used for PWM input!)
  - Incremental encoder input
  - Based on UIO driver ⇒ no syscall overhead
ERT_LINUX I/O support

- MF624: PCI data acquisition card
  - Digital IO, Analog IO, PWM out
  - Counters (can be used for PWM input!)
  - Incremental encoder input
  - Based on UIO driver ⇒ no syscall overhead

- CAN bus transmit/receive

- Some support for Raspberry Pi
ERT_LINUX I/O support

- MF624: PCI data acquisition card
  - Digital IO, Analog IO, PWM out
  - Counters (can be used for PWM input!)
  - Incremental encoder input
  - Based on UIO driver $\Rightarrow$ no syscall overhead

- CAN bus transmit/receive
- Some support for Raspberry Pi
- No Comedi yet
Applications/demos
PMSM motor simulation

Hardware-in-the-loop (HIL) simulation

▶ Analysis of how does security influences safety
▶ PMSM motor simulation @ 20 kHz
▶ Hand-written motor control algorithm runs on real hardware
▶ Message authentication on CANbus + brute force attacks
Robot with parallel kinematic structure

- 4 DC motors, 4 incremental encoders, other I/Os
- Presented at Embedded world 2014
- Sampling period 1 ms but complex computations
- More reliable than previously used Windows target
Motor control on Raspberry Pi

- Final task for students of our real-time course:
  - Create software motor controller on an embedded PowerPC board with VxWorks
  - Can the same be implemented on Raspberry Pi with Linux?
Usable Simulink Embedded Coder Target for Linux

Recent News

Native Linux Real-Time and I/O Cards Support

Current version of the Linux ERT target is optimized to use the proper Linux Real-Time library. MATLAB Simulink running on GNU/Linux on the selected platform is intended to support Linux ERT targets as well.
Hardware

- As simple as possible
- Four NOR gates (SN74HCT02)
- H-bridge (L6203)
Incremental encoder (IRC) processing

- The most demanding part
- IRC signal: 0 – 20 kHz
- Every edge generates an interrupt (up to 80 k interrupts/sec)
- Kernel driver to calculate the position

Sustainable IRQ frequency up to \( \approx 32 \text{kHz} \)

Further improvement: Use of ARM FIQ
Incremental encoder (IRC) processing

- The most demanding part
- IRC signal: 0 – 20 kHz
- Every edge generates an interrupt (up to 80 k interrupts/sec)
- Kernel driver to calculate the position
- Sustainable IRQ frequency up to ≈ 32 kHz
- Further improvement: Use of ARM FIQ
IRC processing details

Position calculation works better if derived from the order of IRQs than from the signal values read in the handler.

FIFO run queue preserves order!
Demo: Raspberry Pi motor controller

- **RESET**: Double
- **Position Request**: Double
- **Trajectory**: Double
- **Pos**: Double
- **Opt. PSD Controller**: Double
- **Subsys PSD**: Double
- **Anti Windup**: Double
- **Active Output Range**: Double
- **Manual PWM Control**: Double
- **Manual PWM**: Double
- **PWM-display**: 0.05976
- **PWMwDirOutp**: Double
- **PWMwDir**: Double
Demo: Raspberry Pi & CAN
Conclusion

- ERT_LINUX target works
- Used in several applications
- Does not suffer from serious problems as original MathWorks code generation targets.
- Limited support for I/Os – Blocks for Comedi are on our todo list
- It might be possible to use our target with other I/O blocks (e.g. from MathWorks)
Conclusion

- ERT_LINUX target works
- Used in several applications
- Does not suffer from serious problems as original MathWorks code generation targets.
- Limited support for I/Os – Blocks for Comedi are on our todo list
- It might be possible to use our target with other I/O blocks (e.g. from MathWorks)

Thank you!