Real-Time Models with Open Source Software Tools

Manfred Lohöfener [1]

[1] *University of Applied Sciences Merseburg, Faculty of Engineering and Natural Sciences, Merseburg, Germany*

Key words: REM 2009, Mechatronics, Open Source, Tools, Linux, Real-Time Patches.

I Introduction

Usually we associate mechatronics with new solutions in mechanical engineering. Many innovations are possible only with electronics and software. Especially in automobile industry we see a very rapid development based on mechatronics.

Many of our students write their final thesis in automobile research and development. Some of them are concerned with sensor problems, others with control algorithms and also with actuator tasks. In almost all cases we observe an important role of modeling of the systems to be developed, or modeling itself is the main task of the thesis.

So we often see confirmed the rightness of the mechatronics' definition by the Technical Committee Mechatronics (VDI/VDE GMA 4.15)[1]: "*Mechatronics is an interdisciplinary design method, solving mainly mechanical directed product tasks by the synergetic spatial and functional integration of mechanic, electric and information processing subsystems.*" In many of our students' theses the VDI guideline 2206 Design Methodology for Mechatronic Systems[2] is referenced for the organization of research and development tasks with the aid of V-models.

II Operating Systems

Now we will talk about the role of real-time operating systems. On the first view this is not particularly interesting for "mechatronicians". Usual desktop computers have their own proprietary operating system as a de facto standard, suitable not only for writing letters and browsing the Internet, but also to solve technical development tasks. We are able to develop models which communicate with the real world via input and output interfaces in real-time, usually well-known as software-in-the-loop SIL or hardware-in-the-loop HIL. If the power of the solution is insufficient, we should demand more powerful processors from Intel with more cores or GHz!

On the other hand we have PLC and embedded control systems already fitted with an operating system by its manufacturers. Only if we are concerned with the development of well tailored control systems with microcontrollers we ask for an own runtime system which could be delivered from several companies like The MathWorks, National Instruments, dSPACE, Wind River Systems, KEIL etc.

III Linux with RT Patches

A completely different method was proposed in 2005 at a meeting of Linux kernel developers. At that time, Linux was not only a server operating system but also a well suited embedded operating system with soft real-time characteristics on numerous DVD players, DSL routers and others. The proposal was the reconstruction of the Linux kernel to solve also hard real-time HRT control tasks. R. Schwebel wrote: "*In 2005, Linux kernel developers Ingo Molnar and Thomas Gleixner revealed two advances to the kernel community: the 'real-time preemption' and the 'high resolution timer' patches. The idea behind their work is simple. Why should we add a dual kernel approach below the kernel when we can fix the reasons why Linux was not HRT up to now? If that approach worked, we had a Linux kernel which could run all kinds of 'normal' userspace POSIX code under hard real-time conditions.*"[3]

The idea to develop Linux to a hard real-time operating system was so strange, that Linus Torvalds, the "father" of Linux, said: "*Controlling a laser with Linux is crazy, but everyone in this room is crazy in his own way. So if you want to use Linux to control an industrial welding laser, I have no problem with your using PREEMPT_RT.*"[4]

And the project was driven by the resulting fascinating possibilities. Surprisingly, the finance sector supported the project to perform finance transactions guaranteed in real-time[5]. On the other hand, audio engineers need such an operating system with very low latencies for audio recording, and they contributed to the development, too.[6]

Now we continue with the traditional conservative mechanical engineering. Here is a demand for open solutions to overcome the dependence on manu-

facturers of automation devices and certain software manufacturers. In 2005, the Open Source Automation Development Lab OSADL[7] was founded to set on the use of free and open source software FOSS in industry. The first goal is almost reached: The RT patches for Linux were developed in a quality to be accepted as a standard part of the so-called mainline Linux kernel in the near future. Maybe it will be accepted with the next kernel version 2.6.32? Next tasks of OSADL are the development of certain drivers and a real-time driver framework, and a contribution to the certification of safety critical Linux-based products.

IV Real-Time Models

Now back to the question, why should we be concerned with that Linux development. Of course we know we need real-time operating systems and real-time models in the sense of software-in-the-loop SIL or hardware-in-the-loop HIL, compare Table 1.

		Environment	
		Model	**Real**
Object to be developed	**Model**	Model-in-the-Loop **MiL**	Software-in-the-Loop **SIL**
	Real	Hardware-in-the-Loop **HIL**	**Final Test**

Table 1: Real-time models with HIL and SIL

There are promising developments in free and open source software FOSS for the use of desktop computers with real-time operating systems. Some examples are given with the use of Scilab/Scicos or MATLAB/Simulink with RTAI Linux or Xenomai and RTAI-Lab[8], [9], [10]. An example is shown in the figures 1 and 2. These investigations are made with the RTAI Knoppix live CD with the kernel 2.6.17 and RTAI version 3.4(11).

Obviously they obtained good results for aerospace industries. But until now there is no broad use of this technology. One reason is the rapid development of the Linux kernel but the development of RTAI modules remains on an early state of the kernel. An overview over kernel versions with release date and the appropriate real-time modules from RTAI and RT patches is showed in table 2.

We see the current RT patches and the two years old RTAI modules. In the near future all RT patches are in the mainline kernel and therefore they are in a very advantageous position compared with RTAI[12]. By the way it must be mentioned that we have another real-time Linux way with the Xenomai project[13], and this project is close to the mainline kernel.

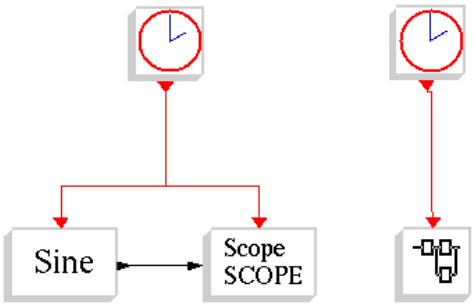

Figure 1: Simple Scicos model and its super block for compiling it into a real-time executable

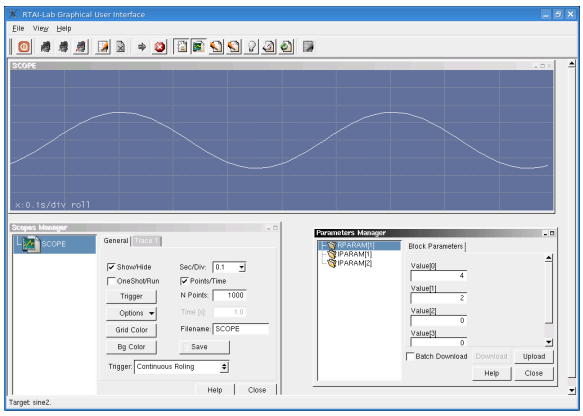

Figure 2: RTAI-Lab with model from Figure 1

kernel.org[14]		**RTAI**[15]	**RT patches**[16]
2.6.14	28/10/05	3.4	
2.6.17	18/06/06	3.5	2.6.17-rt3
2.6.19	29/11/06	3.7	
2.6.23	09/10/07	3.7.1	2.6.23.1-rt11
2.6.24	24/01/08		2.6.24.7-rt27
2.6.26	13/07/08		2.6.26.8-rt16
2.6.29	23/03/09		2.6.29.6-rt23
2.6.31	t.b.d.		2.6.31-rt(?)

Table 2: Available RT modules for Linux kernel versions

Now we have a fascinating aspect: If we use Linux, we have to grapple with only one operating system, no matter whether on desktop computer or embedded control system etc. We are able to work at all platforms in real-time without fundamental changes in the software.

V Empirical Findings

Let's have a look at main characteristics. First empirical investigations were made with different live CDs from OSADL with RT patches[17] on a P4 notebook. So we reached the results for kernel versions 2.6.17 and 2.6.23.

The experiments with the kernel versions 2.6.26 and 2.6.29 were done with a fresh installed kernel on openSUSE 11.1 using the how-to[18]. The necessary steps to install a real-time kernel are described in appendix 1.

To test the real-time behavior, a tool `cyclictest` is started, which starts new processes and measures the latency between the initialization and the first entrance in this task. A second task `hackbench` should be started to get a heavy load on the system. The pbm figures can be found in the directory `/tmp/`.

At first we will have a look on a P4 system running openSUSE 11.1 out of the box without a special real-time kernel. The maximum latency in this case is not bad with about 1.7 ms, but far away from a latency with a few µs, see figure 3.

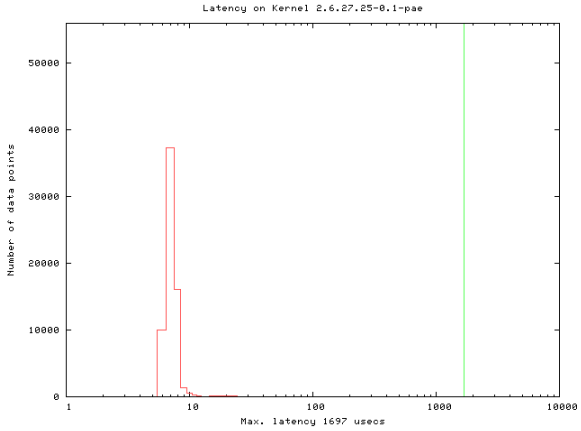

Figure 3: Latency plot with a standard kernel 2.6.27 (no real-time), maximum latency 1697 µs

Another plot shows the result with a two years old kernel 2.6.17 with RT patches. Here we got a very good result with a maximum latency of 14 µs, see figure 4.

A better shape of the latency we obtained with the RT patched kernel 2.6.23.1, with a maximum latency of 27 µs, see figure 5.

In the next example we tested the RT patches with the kernel version 2.6.26.8. The result and the latency of 34 µs are given in figure 6.

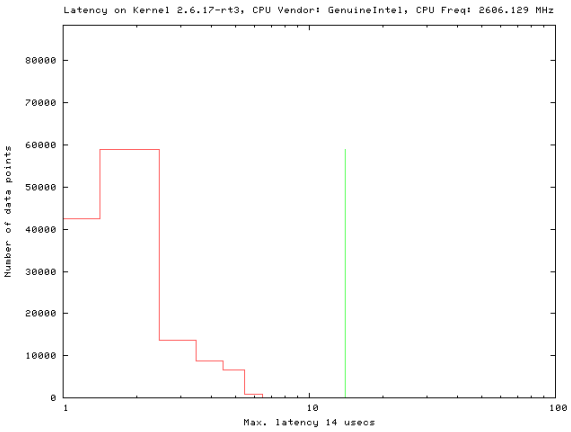

Figure 4: RT patched kernel 2.6.17 maximum latency 14 µs

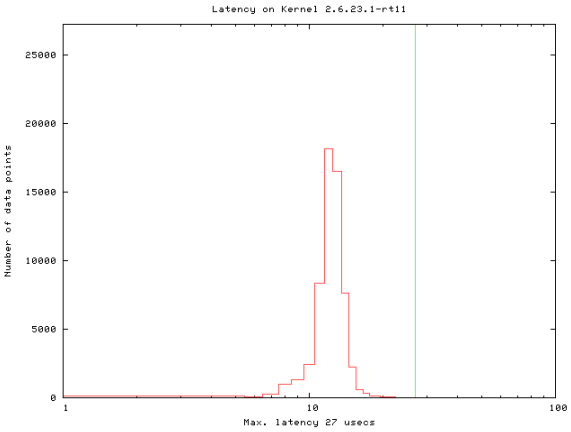

Figure 5: RT patched kernel 2.6.23.1 maximum latency 27 µs

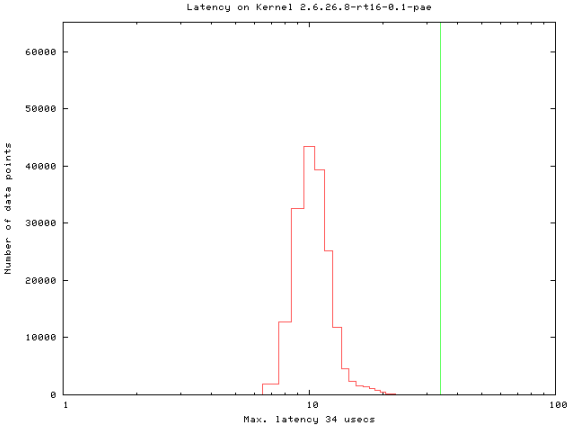

Figure 6: RT patched kernel 2.6.26.8 maximum latency 34 µs

Then we ran the latest stable kernel release 2.6.29.6 with RT patches. The result is given in figure 7, the maximum latency is 59 µs.

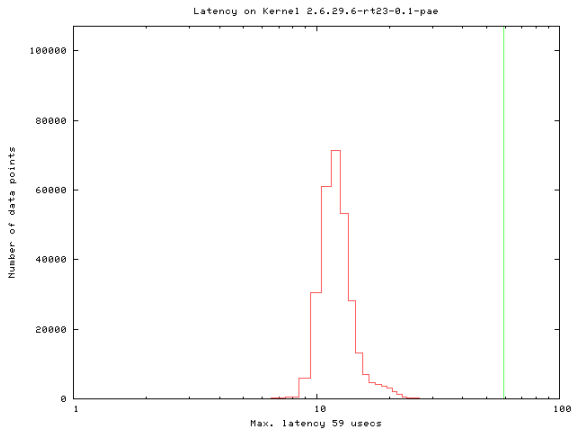

Figure 7: RT patched kernel 2.6.29.6 maximum latency 59 µs

It's indeed surprising, that the maximum latency increases with the kernel version, like summarized in diagram 1. Until now, we don't know any explanation for this unexpected behavior yet.

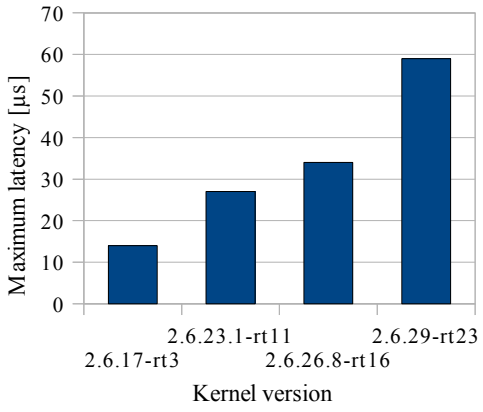

Diagram 1: Kernel versions and maximum latency

All tests are executed only in software without access to hardware components. This is sufficient for first investigations but in real cases we should add latencies of integrated circuits and other hardware components, and of interrupts. Of course, such test have to be done on the physical target control devices to include all influences from hardware and environment.

VI Interfacing the Physical Environment

So far a real-time driver framework for digital and analog input and output interfaces is under development but only with this drivers we can connect our software with industrial data acquisition devices. After this framework is available, we can continue testing but we have no final release yet.

An important step is the collaboration[19] of OSADL and the EPSG[20] (Ethernet POWERLINK Standardization Group). One aim is the integration of the open POWERLINK standard into real-time Linux. So we can expect solutions, in which our personal computer can be used with already available industrial POWERLINK input and output devices as a convenient HIL or SIL test system with more power, compared with present solutions.

VII Conclusions

After an explanation of the necessity of real-time operating systems for model-based design of mechatronic systems, we discussed first results from tests of the reached latency with several RT patched Linux kernel versions.

We expect great advantages after the availability of the real-time driver framework for connections of the computer to the physical environment. More informations can be collected from the OSADL link list[21].

References

[1] VDI/VDE Society for Measurement and Automatic Control, Technical Committee 4.15 Mechatronics. http://www.vdi.de/6481.0.html?&no_cache=1&L=1, http://www.vdi.de/3682.0.html (08/15/09)

[2] VDI, Verein Deutscher Ingenieure (Ed.). *Design Methodology for Mechatronic Systems*. VDI guideline 2206, Beuth Verlag, Berlin, 2004.

[3] Robert Schwebel: *Next-generation hard real-time on POSIX-based Linux*. ECE, July 2006, p. 15f http://www.pengutronix.de/oselas/realtime/Pengutronix-POSIX-Hard-Realtime-ECE-200606.pdf (08/15/09)

[4] Theodore Ts'o, Darren Hart (Ed.): *Real-Time Linux Wiki*. http://rt.wiki.kernel.org/index.php/Main_Page (08/15/09)

[5] Red Hat, Inc.: *MRG Realtime*. http://www.redhat.com/mrg/realtime/ (08/15/09)

[6] Canonical Ltd.: *ubuntu studio*. http://ubuntustudio.org/ (08/15/09)

[7] OSADL Open source Automation Development Lab: *Open Source Software for Automation and Other Industries*. http://www.osadl.org/ (08/15/09)

[8] Roberto Bucher, Simone Mannori, Thomas Netter: *RTAI-Lab tutorial: Scilab, Comedi, and real-time control*. February 28, 2008, https://www.rtai.org/RTAILAB/RTAI-Lab-tutorial.pdf (08/15/09)

[9] Cosateq GmbH & Co.KG, Wangen, DE: *Perform real-time HIL simulation and rapid prototyping using standard PC hardware*. http://www.scale-rt.com/ (08/15/09)

[10] Fraunhofer-Gesellschaft: *Open Real-Time Linux Projekt*. LiveDVD V0.2, http://www.open-realtime-linux.de/ (08/15/09)

[11] Gianluca Palli's Home Page: *Knoppix based RTAI LiveCD*. http://www-lar.deis.unibo.it/people/gpalli/files/rtai_knoppix.iso (08/15/09)

[12] Jonas Mitschang: *Harte Echtzeit unter Linux. Fallstudie RTAI vs. RT-Preempt*. FhG, IESE-Report Nr. 058.07/D, Version 1.0, 20. März 2007, RTLOpen-Projekt, FKZ 01 IS C14, RTLOpen-Report 010/D, Version 1.0, 20.03.2007, http://mitschang.net/download/IESE-Report%2058.pdf (08/15/09) (only in German)

[13] Xenomai: *Real-Time Framework for Linux*. http://www.xenomai.org/ (08/15/09)

[14] The Linux Kernel Organization: *The Linux Kernel Archives*. http://www.kernel.org/pub/linux/kernel/v2.6/ (08/15/09)

[15] Politecnico di Milano - Dipartimento di Ingegneria Aerospaziale: *RTAI - the RealTime Application Interface for Linux from DIAPM*. https://www.rtai.org/RTAI/ (08/15/09)

[16] Red Hat, Inc.: *Red Hat Enterprise MRG Realtime*. http://rt.et.redhat.com/download/ (08/15/09)

[17] OSADL Open Source Automation Development Lab: *OSADL Project: Realtime Kernel Live CD*. http://debian.tu-bs.de/project/tb10alj/knoppix.iso, http://debian.tu-bs.de/project/tb10alj/osadl-knoppix.iso (08/15/09)

[18] OSADL Open Source Automation Development Lab: *HOWTO: Realtime-Preempt Kernel. Enable real-time capabilities of the mainline kernel*. http://www.osadl.org/Realtime-Preempt-Kernel.kernel-rt.0.html (08/15/09)

[19] Bernecker + Rainer Industrie Elektronik: *Open Cooperation. EPSG and OSADL agree on close cooperation*. B&R automation, News, 05/14/08 http://www.br-automation.com/cps/rde/xchg/br-automation_com/hs.xsl/cookies_allowed.htm?caller=news_9877_ENG_HTML.htm (08/15/09)

[20] EPSG Ethernet POWERLINK Standardization Group: *Ethernet POWERLINK*. http://www.ethernet-powerlink.org/ (08/15/09)

[21] OSADL Open Source Automation Development Lab: *OSADL on the Internet*. http://www.osadl.org/On-the-Internet.osadl-on-the-internet.0.html (08/15/09)

Appendix 1: Install and Test a Real-Time Kernel on openSUSE 11.1 with KDE

Preparation
Install kernel sources, compiler etc.
	`[Kickoff Application Launcher] → Applications → System → Administrator Settings YaST (root password)`
	`Software → Software Management`
	`Filter → Patterns → Development → [√] Linux Kernel Development → [Alt]+[A]`

Install ketchup
Start konsole and change to root
Please notice dash after `su`
Attention: Only one line after `wget`
Set attributes and make backup
	`[Alt]+[F2] konsole`
	`su - (root password)`
	`wget -O /usr/local/bin/ketchup http://people.redhat.-com/srostedt/rt/tools/ketchup-0.9.8-rt3`
	`chmod +x /usr/local/bin/ketchup`
	`mkdir /backup`
	`mkdir /backup/boot`
	`cp -R /boot/* /backup/boot`

Prepare sources
Do in root console
Use the latest kernel configuration
Don't forget the dot at the end of `cp` …!
	`mkdir /usr/src/tmp`
	`cd /usr/src/tmp`
	`ketchup -r -G 2.6.29.6-rt23`
	`cd /usr/src/linux-2.6.29.6-rt23`
	`cp /proc/config.gz .`
	`gzip -d config.gz`
	`mv config .config`

Configure the kernel
Do in root console on desktop (also possible: `make menuconfig`)
Set options with search function
Don't change other options
Save configuration and close xconfig
	`make xconfig`
	`[Ctrl]+[F]` **`preempt`** `[Search]`
	`[√] Complete Preemption (Real-Time)`
	`tracer` `[Search]`
	`[√] Interrupts-off Latency Tracer`
	`[√] Preemption-off Latency Tracer`
	`[√] Kernel Function Tracer`
	`[√] Scheduling Latency Tracer`
	`[√] Trace various events in the kernel`
	`[Esc] → [Ctrl]+[S] → [Ctrl]+[Q]`

Install the kernel
Start `make -j[number]` with double of number of processor cores
	`make -j4`
	`make modules_install`
	`make install`

Restart computer Mark the added option with the new kernel in GRUB	`[√] 2.6.29.6-rt23`

Install test tools
Start root console
Install hackbench
Got to `/root/` directory
Load test tool sources
Compile test tools
	`[Alt]+[F2] konsole`
	`su - (root password)`
	`zypper install hackbench`
	`cd /root`
	`KERNELGIT=git://git.kernel.org/pub/scm/linux/kernel/git`
	`git clone $KERNELGIT/clrkwllms/rt-tests.git`
	`cd /root/rt-tests`
	`make`

Start cyclictest in root console	`/root/rt-tests/cyclictest -a -t -n -p99`
Cause heavy load in second root console	`while true; do hackbench 25; done`
Stop programs in consoles	`[Ctrl]+[C]`
Show resulting plots on the desktop	`konqueror /tmp/*pbm`