Development and implementation of real-time data acquisition systems for fusion devices with Open Source software

M. Zilker[a,*], K. Behler[a] , T. Bluhm[b], P. Heimann[a], Ch. Hennig[b], H. Kroiss[a], G. Kühner[b], H. Laqua[b], M. Lewerentz[b], J. Maier[a], G. Neu[a], G. Raupp[a], M. Reich[a], H. Riemann[b], J. Schacht[b], A. Spring[b], W. Treutterer[a], A. Werner[b], T. Zehetbauer[a]

[a] *Max-Planck-Institut für Plasmaphysik, EURATOM Association, Boltzmannstraße 2, D-85748 Garching, Germany*

[b] *Max-Planck-Institut für Plasmaphysik, EURATOM Association, Teilinstitut Greifswald, Wendelsteinstraße 1, D-17491 Greifswald, Germany*

[*] *Corresponding author; Tel.: +49-89-32991930; Fax: +49-89-32992044; E-mail address: manfred.zilker@ipp.mpg.de*

Abstract

To improve the plasma position and shape control system in ASDEX Upgrade, it is constantly expanded by the integration of data acquisition systems of important and interesting diagnostics. Usually the main responsibility of data acquisition systems is to collect data and subsequently put it into a data base archive system from where it is later analysed.

The obligation now is also to simultaneously process the acquired data with an appropriate algorithm and send the pre-processed data in real-time to the plasma control system during a discharge. To achieve this requirements the used hardware must provide enough processing power and the operating system has to meet some real-time constraints.

To avoid the burden of using proprietary real-time operating systems the trend is to use Open Source variants mainly based on Linux. Some of these solutions also allow us doing real-time capable communication using standard Ethernet hardware.

By way of an example the implementation process of a prototype of a real-time data acquisition system based on a multi-core processor and Xenomai is demonstrated. Other possible solutions like Realtime Linux and their differences to Xenomai which we propose as the most sophisticated real-time framework for Linux are discussed in this paper.

Keywords: data acquisition, control, Linux, real-time, Xenomai, Rtai, RTnet

1. Introduction

The demand of introducing real-time enabled data acquisition systems at ASDEX Upgrade emerged from the closer interaction between the data acquisition domain and the Discharge Control System (DCS) which promises better shape control of the plasma during a discharge. Several connected data acquisition- and control systems can later compose a multi-platform real-time network [1]. No less important are such systems for W7-X where the data must be analysed and stored on the fly and continuous operation is a fundamental design principle [2]. The motivation for this work was to evaluate, if non-commercial, Open Source based, real-time systems can provide a reliable and long-term solution in this area. In the specific example of the Motional Stark Effect (MSE) diagnostic data acquisition system at ASDEX Upgrade this possibility should be examined. Its old implementation was based on single i386 processor hardware in combination with a Linux 2.4 system. All of the time-critical parts were implemented in loadable kernel modules co-operating together, which satisfied the real-time constraints but showed a lot of drawbacks in terms of maintainability, extensibility and user acceptance. The necessary redesign should have the option that most of the software can be written and tested as user space programs and only the absolutely necessary parts should run in kernel mode. Both parts should interact through a well-defined interface. This approach can afford a simpler overall software design of the application and facilitates testing and debugging greatly. It should be a hard real-time capable Linux extension supporting state of the art hardware and the performance of multi-core based systems should be fully exploited.
For a better understanding, first the basics of a hard real-time capable operating system are presented. Then we give a more in-depth introduction to the Xenomai real-time extension which turned out to be the best solution to meet our demands. The implementation of a prototype data acquisition system using all the previously described tools is presented in the following paragraph. We do not study extensively on Linux based real-time systems like [3] but make some latency and task-switching tests to get an idea of this important numbers for our application. Finally we give a conclusion and an outlook for the future work.

2. Linux and real-time

Traditional Linux is optimised for throughput and server processing and never was intended to support real-time applications from scratch. As Linux gained more acceptance in the desktop area the first real-time demands were expressed mainly for audio and video applications. Soft real-time is now already a common feature in standard Linux but the achievable response times are not deterministic. Recently two approaches gained acceptance both in the Linux community and in industry to endow standard Linux with the properties of a hard real-time operating system:
The first variant dating back to 1998 is the so called co-kernel approach where applications run on a small real-time enabled microkernel co-operating with the Linux kernel.
This functionality is added to the standard kernel by applying a small patch and loading some kernel modules at runtime. The mostly used actual representatives are RTAI [4] and Xenomai [5].

Both have the same roots and the capability of providing hard real-time in user space. Although RTAI can deliver the lowest possible numbers for interrupt latency and task-switching times, which is a big advantage for very time-critical real-time applications [6], the Xenomai approach is more concentrated on extensibility (RTOS skins, RT_PREEMPT support), portability, stability and user friendliness. More detailed information about this topic can be found in the Xenomai FAQ [5].
The second approach aims to make the standard kernel more responsive by introducing additional pre-emption points for safe scheduling; a real-time scheduler and reducing the overall interrupt latency. This results in significant modifications on the kernel code which is very hard to check and verify. Most of the commercially available Realtime Linux products are based on this principle like MontaVista Linux and LynxOS. The improvements of the Open Source version are done by the Realtime Linux Project [7] and migrate stepwise into new releases of the standard kernel as of Linux 2.6.18. This strategy continuously makes progress and in recent editions the task-switch latency could be reduced to the 50 µs range. A big advantage of this concept is that the software developer can revert to the normal GNU/Linux and POSIX API without any restrictions.
RTAI and Xenomai support the Real-Time Driver Model (RTDM) already in its entire form, which provides a unified interface of real-time user space applications and the hard real-time system. It is an approach to unify the interfaces for developing device drivers and associated applications under real-time. A RTDM compatible driver developed for one extension can be run on the other extensions as well. This is a big advantage over commercially available solutions where only proprietary driver APIs are supported. It is also expected that the final RTDM version for Realtime Linux will be available in the near future.
Real-time communication is accomplished by RTnet [8], a real-time capable network protocol stack with which supports standard Ethernet hardware and offers deterministic versions of UDP/IP, ICMP and ARP which guarantee minimal bounded submission and receive times for data packets to and from a network. RTnet is completely supported by Xenomai and RTAI.
Although Realtime Linux was also considered as a candidate for our purposes at the beginning of the evaluation it was later discarded because it is still under heavy development and testing and finally because no real-time version UDP/IP was available. It is also a better strategy being independent from the development cycle of the mainline Linux and rely on the lightweight Xenomai microkernel only for the real-time services. The combination of both, Xenomai and the final version of Realtime Linux integrated fully in the mainline kernel could offer the best of both worlds. This led us to propose Xenomai as the appropriate Linux real-time solution for our demands.

3. Xenomai

In 2005 Xenomai emerged from the RTAI project and hence shares a lot of common concepts and code. However the Xenomai project being independent since this time, invested a lot of effort to continually improve the product for the user. A block diagram of the Xenomai subsystem co-operating with the Linux kernel is shown in Fig. 1. Hard real-time applications executed in user space are the preferred method but it is still possible to use kernel space modules only when better time response is absolutely necessary. In the upcoming version 3 of Xenomai the kernel modules concept is discontinued and the only interface to the kernel will be over RTDM. Thus a hard real-time user task can call any service of the native Linux domain where it looses the hard real-time property but when the service is finished it resumes hard real-time execution. The migration between the primary (hard RT) and secondary domain (soft RT) is completely transparent to the user and the time costs depend only on the granularity of the Linux kernel which can be quite small when the Linux kernel itself gets better real-time characteristics. A Xenomai task entering the secondary domain does not loose its usually high priority as the Linux kernel as a whole inherits the priority of the real-time task. This is an important difference to RTAI/LXRT where the migrated real-time task always gets the lowest priority from the scheduler.

Xenomai provides the following real-time interfaces for hard real-time user space applications:

A POSIX 1003.1b skin which is a conformant implementation of the universal real-time API .

The RTDM (Real-Time Driver Model) interface provides a framework for writing real-time device drivers which can be accessed by a POSIX interface. It follows the clean separation between hardware interface and application program and can guarantee deterministic behaviour.

A native API skin with semantics close to the traditional RTOS APIs offers the most efficient use of the real-time kernel services.

4. The real-time prototype

The MSE data acquisition system described here is controlled completely by the Discharge Control System (DCS) with a lot of intercommunication. Other systems are more loosely coupled and have almost no interaction during a discharge with the DCS. To guarantee the desired real-time behaviour of the whole application running under Xenomai it was necessary to have real-time enabled device drivers for the involved I/O hardware. The RTnet drivers for RT communication are available by default. Therefore it was only necessary to make a port for the used Datel ADC board and the in-house developed TDC board. For both the driver source code was available and the work to create a RTDM compatible real-time version turned out to be less complicated and error-prone than expected. The RTDM drivers consist of a real-time part for the time-critical actions and a non-real-time part for initialization and cleanup functions. Below is a small code excerpt of the *rtdm_device structure* where the RTDM drivers operations are defined :

```
ops: {
        open_rt:        NULL
        open_nrt:       tdc_close,

        close_rt:       NULL
        close_nrt:      tdc_close,

        ioctl_rt:       tdc_ioctl_rt,
        ioctl_nrt:      tdc_ioctl_rt,
        read_rt:        tdc_read_rt,
        read_nrt:       NULL,
        ...
    }
```

At this stage most of the code for the non-real-time parts could be directly taken from the original Linux version with only minor changes. The other real-time functions have to be re-implemented. A crucial part is the function *rtdm_irq_request (irqnr,..)* It registers the interrupt line *irqnr* of a I/O board at the Xenomai microkernel where it is serviced at the first opportunity when a real-time handler becomes available. Otherwise the interrupt is propagated to the standard Linux kernel. Properly written RTDM device drivers can be used in the Xenomai real-time domain as well as in the Linux domain depending on the domain (primary or secondary) the program is executed.

Within the MSE data acquisition system the whole discharge time is divided into data frames, each of 1 ms duration. A frame consists of 16 digitized analogue input signals for 190 points on the time axis. The ADC board is clocked with a frequency of 190 KHz which leads to one DMA interrupt exactly every millisecond. The device driver handles the interrupt and sets up the next DMA. The ADC sample clock and start trigger are supplied by the Time to Digital Converter (TDC) [9] an experiment wide synchronized timing unit with 20 ns resolution. The acquired data frame then is transferred into the user space and subsequently processed by an user-defined algorithm which must be performed in less than 1 ms. After the data acquisition starts the initial latency is slightly smaller than 2 ms for the first available result. The following frames are in a strictly 1 ms interval because of the overlapping (parallel) action of the data sampling and real-time algorithm. An illustration of the timing is given in Fig. 2.

In the case of MSE a real-time algorithm generates the “measured field angle” information for each frame tagged with a global 64-bit timestamp. This result is sent immediately to the DCS via UDP/IP. For a plasma discharge all acquired raw data for a 16-channel ADC board is stored in an array of approximately 60 MBytes. When the discharge has finished the acquired raw data and online computed meta data are stored as a Level-1 shot file in the Andrew File System (AFS) tree of the diagnostics project from where it can later be analysed by special software tools.

The new software architecture consists of four main parts:

1. A main program for setup and cleanup of the tasks and the final shot file writing.
2. One or two RT data acquisition task (depending on the number of ADC boards).
3. One RT communication task.
4. RTDM drivers for I/O hardware.

The whole application runs in user space and the 3 tasks share a common memory. To make sure that the real-time characteristics of the data acquisition task(s) are not disturbed at any time the propagation of the results is put into a separate RT communication task which is notified by a fast inter-task event mechanism when a packet is ready to be sent to the DCS. Fig. 3 shows the software architecture of the real-time prototype and its execution spaces.

The communication with the DCS of ASDEX Upgrade is realized with UDP/IP. The current prototype only showed the feasibility of this method. To relieve all the connected data acquisition systems from implementing the communication by themselves and keeping track of continuous protocol changes in the Discharge Control System the rtdiag library was developed by ASDEX Upgrade. It encapsulates the protocol and the messages sent to and received from the discharge control computer and provides an application interface for the user. This library is available for Solaris and VxWorks at the moment and in its final test phase. One of the next tasks is to port this library to Linux and integrate it into the Xenomai/RTnet environment.

5. Performance tests

The crucial parts of any hard real-time capable OS are mainly interrupt latency and task-switching times. We measured these numbers for Xenomai. As test platform served an industrial main board equipped with an Intel ® Core2Duo E6400 processor with 2.4 GHz and the 945G chipset. The interrupt latency was measured with an external source as follows: the square-wave output signal from a frequency generator is connected to the ACK pin of the printer port which is configured to generate an interrupt on every leading edge. In return a special developed printer port device driver reads the TSC register of the CPU to record the actual time and toggles an output pin signalling that the interrupt was received. The latencies were observed with this method and a Sample Oscilloscope. Fig. 4 shows an example of the latency measurement: the external source (lower signal) initiates an interrupt on the leading edge and the device driver toggles the upper signal. Before the interrupt service routine terminates, the user task is notified which in turn generates a second pulse (upper signal) when it is scheduled. The horizontal axis is 5 µs/div.

Xenomai shows a interrupt latency in the range of 5 – 16 µs without any system load. Increasing the load to ˜75% has only a minor negative impact on this numbers. The task-switching time varies between 2 µs and 14 µs. The context-switch benchmark shows times between 48 and 55 µs independent of the system load. The real-time algorithm to produce the “measured field angle” needs an average time of 504 µs for each frame.

6. Summary and Outlook

The development process of the real-time data acquisition showed that even projects completely based on Open Source software can produce high-performing results. It has been demonstrated that a hard real-time application could be developed, debugged and deployed completely in user space except for the device driver which was the only part implemented as a RTDM compatible kernel module. The Real-Time Driver Model can become a standard for driver development in the Realtime Linux community which would greatly facilitate the exchange and distribution of such device drivers. With Comedi over RTDM the first step is already done into the right direction. The execution speed overhead compared to completely kernel based implementations is negligible on modern processors. We propose Xenomai as most suitable real-time framework for Linux because of its flexibility, smart architecture – in particular the seamless co-operation with native Linux services - the various skins it offers to the application developer, and finally the interesting roadmap it presents for the future.

References

[1] K. Behler, H. Blank, H. Eixenberger, A. Lohs, K. Lüddecke, R. Merkel, G. Raupp, G. Schramm, W. Treutterer , M. Zilker , ASDEX Upgrade Team real-time diagnostics at ASDEX Upgrade - architecture and operation, Fusion Engineering and Design 83 (2008) 304-311.

[2] P. Heimann, S. Heinzel, Ch. Hennig, H, Kühntopf, H, Kroiss, G. Kühner, J. Maier, J. Reetz, M. Zilker, Status report on the development of the data acquisition system of Wendelstein 7-X, Fusion Engineering and Design, 71 (2004) 219-224.

[3] Barbalace, A.; Luchetta, A.; Manduchi, G.; Moro, M.; Soppelsa, A.; Taliercio, C. Performance Comparison of VxWorks, Linux, RTAI, and Xenomai in a hard real-time application, IEEE Transactions on Nuclear Science 55 (1) (2008) 435 – 439.

[4] RTAI Home Page [Online] www.rtai.org [Online].

[5] Xenomai Home Page [Online] www.xenomai.org [Online].

[6] André Neto et al, "Linux real-time framework for fusion devices" Fusion Engineering and Design 84 (2009) 1408-1411.

[7] Realtime Linux Home Page, www.osadl.org/Realtime-Linux.projects-realtime-linux.0.html [Online]

[8] RTnet Home Page, www.rtnet.org [Online]

[9] G. Raupp, R. Cole, K. Behler, M. Fitzek, P. Heimann, A. Lohs, K. Lüddecke, G. Neu, J. Schacht, W. Treutterer, D. Zasche, Th. Zehetbauer, M. Zilker, A "Universal Time" system for ASDEX Upgrade, Fusion Engineering and Design, 66-68 (2003) 947.