Application of RT-Preempt Linux and Sercos III for Real-time Simulation

Michael Abel\textsuperscript{a,b}, Luis Contreras\textsuperscript{a}, Prof. Peter Klemm\textsuperscript{a,b}
\textsuperscript{a}Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW), Seidenstr. 36, 70174 Stuttgart, Germany
University of Stuttgart
\textsuperscript{b}Graduate School of Excellence advanced Manufacturing Engineering (GSaME), Nobelstr. 12, 70569 Stuttgart, Germany
University of Stuttgart
\{Michael.Abel, Peter.Klemm\}@isw.uni-stuttgart.de, Luis.Contreras@ipmc.de

Abstract

This paper presents the application of RT-Preempt Linux in a virtual commissioning scenario. In this scenario, a proprietary Programmable Logic Controller (PLC) is connected to a real-time simulation model. The model is located on a separate Linux personal computer which simulates, for example, the hardware of a production machine. Furthermore, the controller and the simulation computer are connected through the sercos III automation bus. The simulation computer uses a sercos III PCI card as communication hardware in combination with a user space IO (UIO) driver. This allows the execution of the simulation model and the sercos III driver as real-time processes on the simulation computer. The sercos III driver was adapted in order to imitate the bus-interface of a custom sercos III bus-coupler and to provide easy integration into the PLC engineering system. Moreover, variables in the PLC can be coupled to input and output values of the simulation model. With this virtual commissioning method, it is possible to reduce the time to market of a machine, since writing and testing the PLC code for the controller can be done in parallel to the construction of the hardware.

1 Introduction

Due to the constant pressure of the market, manufacturers have to bring new production machines to the market regularly. To reduce the time to market of their machines manufacturers have to find methods to decrease the overall development time. Especially at companies where machines are built for special purposes, every machine is unique and even needs software which has to be developed particularly for one machine. The overall development time of a machine can be decreased dramatically by parallelisation of development tasks. Since most of the development tasks depend on each other, dependencies have to be taken into account. Software components can be written by means of the system specification. Parts of the software can even be tested by software tests before the hardware of the machine is finished. But testing and bug-fixing of functions which depend on the availability of the machine can only be done when the hardware is available. The same problem comes up at the commissioning of the machine, since the whole machine needs to be finished until commissioning can begin. To solve this problem, a virtual commissioning of the machine can be performed. This can be achieved by applying a virtual machine model which simulates the mechatronic parts of the machine. Since production machines are usually controlled by real-time systems, a real-time model for hardware in the loop simulation is well suited for this purpose [1]. Hardware in the loop simulation means that a simulation model calculates the mechatronic behaviour of a machine while its functions are controlled by real control programs.
2 Theory & State of the Art

This chapter gives a short introduction into the technologies and software systems which are used within this project.

2.1 RT-Preempt Linux and User Space IO Drivers

Linux with real-time kernel preemption (RT-Preempt) is an enhancement to the Linux kernel. The aim of this patch is to enable real time capabilities in the Linux kernel. The RT-Preempt patch allows user space programs to run in real-time [2], [3].

The User Space IO (UIO) driver model enables drivers to run in the user space of a Linux system [4]. UIO drivers are a convenient method to implement drivers for non-standard and rarely used hardware which does not fit into the regular kernel subsystems. The memory of a device is mapped into addresses which are accessible from user space memory segments. To handle interrupts, a user-space thread can be applied. In addition, a small interrupt handler within the kernel space is necessary to wake the thread. With this functionality it is possible to write drivers for special purpose devices without the need to handle complex in-kernel structures. UIO-drivers are often used to handle networking devices for field-buses on systems which are running on RT-Preempt Linux.

2.2 Serial Real-Time Communication System (sercos) III

The automation bus sercos III is an Ethernet based field-bus system which can be used in a wide range of automation applications. Sercos III is standardised by the association sercos International e.V. [5]. In the following the term sercos is used as abbreviation to sercos III. Sercos is based on standard Ethernet and uses Ethernet frames to communicate on the bus. A sercos network consists of a bus master and several slave devices (Figure 1).

![Figure 1: Sercos III ring with master and slave devices.](image)

Each device is equipped with two Ethernet ports. The preferred bus topology is a ring structure, since a ring provides more redundancy than a star topology. Apart from this, a line topology with one or two lines (i.e. broken ring) can be used as well. Sercos uses a sophisticated device model which classifies every bus component into different classes of functionality. According to the device model it is possible to distinguish between servo-drives, IO-devices and other automation hardware.

Furthermore, a parameter model was introduced to describe functional interfaces of field-bus devices. Every device has a set of sercos parameters which characterise the interface of the device. Parameters can be accessed by unique identification numbers (IDN). Furthermore, a parameter contains a description of the parameter as string, several attributes and the data of the parameter with a with fixed or variable length.

Sercos uses a start-up phase with five different communication phases (CP) which are usually called CP0 to CP4. When the communication phase has passed the early stages and reaches CP4, real-time communication is active, devices and connections are set up adequately and real-time data can be transmitted. Furthermore, sercos devices can be described in the sercos Device Description Markup Language (SDDML) which is based on the Extended Markup Language (XML).

2.3 Passive sercos III PCI Card

Custom and PC based sercos slaves can be built by equipping PCs for example with sercos III PCI networking cards from the company Automata [6]. The card contains standard Ethernet communication hardware and a FPGA in order to connect it to the PCI bus. To bring the card to operation, a proprietary driver is necessary. This Sercos Slave Driver (SSLV) is written OS independently and contains a hardware abstraction layer which can be ported to other operating systems easily. The card is named "passive" because a driver, which executes the sercos networking stack, and a real-time operating system are necessary to use the card.

This project utilises a port of the SSLV to RT-Preempt Linux. The SSLV is running as UIO-Driver within the user space. To support the user space part of the driver there is also a small kernel module called uio_sercos3 in the mainline kernel. Figure 2 shows a rough overview of the SSLV.
2.4 MLP VEP and IndraWorks

In this project a proprietary Programmable Logic Controller (PLC) produced by the company Bosch Rexroth [8] is deployed. The MLP VEP is a PC based PLC which is equipped with several sercos ports and acts as sercos master device on the bus system. Furthermore, it can be programmed and configured with the engineering tool IndraWorks. It is able to execute PLC programs written in the five languages specified in IEC 61131-3 [9] and has additional Motion Logic Control (MLC) functionality.

2.5 Virtuos

Virtuos [10],[1] is a simulation software which enables the execution of mechatronic and other models in real-time. Virtuos consists of three different software parts: Virtuos-M, Virtuos-V and Virtuos-S which provide different functionality to the user. Virtuos-M and Virtuos-V are used for modelling and visualisation of simulation models. Virtuos-S is used as simulation solver which can compute simulation models in real-time. Within this project a Virtuos-S variant is deployed which can be executed on a Linux system. To synchronise Virtuos with other programs semaphores are a convenient method. They can trigger the beginning of a simulation step or inform that an simulation step has finished. Other programs can use a Virtuos library that provides access to the input and output ports of the simulation model.

3 Problem Definition

To perform virtual commissioning of a production machine, a real-time simulation model of the hardware is necessary. This virtual machine model can usually be executed by Virtuos on a PC which also executes a PLC or an other type of controller software. Indeed, the project specification demanded the application of a MLP VEP PLC, a proprietary controller which is not able to execute a Virtuos model. Moreover, it provides no standard interfaces to connect it to a virtual machine model. To solve these problems, a new method is desired to connect the PLC to the model. Since the PLC is a proprietary device which needs to be programmed by proprietary software there are no simple methods to extend it by custom real-time tasks.

The solution to this problem is to move the simulation model to a PC and let it communicate with the PLC. Since the communication between PLC and simulation model needs to be run in real-time, a field-bus can be used to connect the PLC to the simulation (Figure 3).

4 Approach

Sercos uses a device model which can provide different types of automation devices. But no device has an interface which resembles the complexity of
Virtuos simulation model. A simulation model produces and consumes a high amount of data in every simulation cycle. In this project it is sufficient to provide exchange of floating point values and integer values, since the simulation consists of a mechatronic model. The interface between the PLC and the simulation was defined as an amount of integer and floating point values. To be able to integrate the simulation model into the bus system the interface of the model was enhanced to resemble the interface of a (very large) bus-coupler (see figure 4).

A bus-coupler is a standard field-bus device that usually couples various kinds of electrical input and output signals to the field-bus. With this solution it is possible to "hide" the interface of the simulation model behind the interface of a large bus-coupler. This also enables easy integration of the simulation model into the PLC program, since the interface of the simulation model looks like a bus-coupler. Due to the application of the real bus system to connect to the simulation model, the timing which comes to use later is also applied.

## 5 System Design

This chapter introduces the design of the simulation system. Figure 5 shows the overall system structure.

The PLC Controller is located on the left hand side of the figure. A separate computer with IndraWorks is needed to develop and compile programs for the PLC. The configuration of the field-bus system is done from this system as well. On the right hand side the simulation computer system with an RT-Preempt patched Linux kernel is shown. This system is also equipped with a passive sercos PCI communication card. PLC and simulation PC are connected via sercos. For debugging purposes an Ethernet wiretap can be inserted, as shown in the figure. Inside the simulation PC the SSLV and the simulation model are executed. Since the system is (beside the RT-kernel) a standard Linux PC, additional software can be used for debugging purposes as well. The SSLV is executed to support sercos communication with the PCI card. Moreover, it is equipped with IPC interfaces to communicate with the simulation model. Besides of that, the SSLV can record debugging information in real-time into a FIFO buffer. This information can be easily read by third party programs or saved for later analysis.

### 5.1 Communication Concept

The concept for distributing data within the system is shown in figure 6.

Simulation data is transferred in data packets. A data packet consists of a certain amount of integer values and a certain amount of floating point values. To support the simulation with enough data, packets with 64 32-bit integer values and 32 64-bit floating point values are used. All the values are composed together to data packets of 512 bytes in size. Data packets are composed in the PLC and are put into Ethernet frames which are sent via the field-bus to the simulation PC. In the PC, packets are copied to the address space of the SSLV. The Virtuos-IO (VIO) thread synchronises the execution of the simulation, decomposes the packet into data types and copies them to a memory mapped address space which is shared with the simulation. The data transfer back from the simulation to the PLC works in a similar manner.
5.2 Controlling the Simulation from the PLC

Within the programming system of the PLC, IO ports of devices can be mapped to variables. Variables can be connected to either input or output ports. Afterwards, IO-operations can be done by setting bit-masks in the PLC program. In addition, field-bus devices can be added to the IndraWorks project from a device database. The database can be extended by device descriptions. For sercos devices this can be achieved by using files in the SDDML-Language. Since the bus-coupler which is used in this project is not a standard off-the-shelf bus-coupler, a SDDML file was written which describes a very large bus-coupler. The file was added to the device database of IndraWorks to be able to use it within the PLC program.

Moreover, a data structure was created which combines all the IO-data that has to be send or received in one communication cycle. The structure contains a certain amount of 32 bit integer and 64 bit floating point variables (See listing 1).

```
TYPE io_type:
  STRUCT
    reals: ARRAY [0..31] OF LREAL;
    integers: ARRAY [0..63] OF DINT;
  END_STRUCT
END_TYPE

Listing 1: Specification of a data packet in the PLC
```

Since the size of bytes in the structure is equal to the size of bytes of IO-data in the bus-coupler it is possible to connect the complete structure to the IO-configuration at once. To support input and output data, two structures were added to the input and to the output of the device. Consequently, a regular PLC program can be used to perform calculation input and output operations.

5.3 Enhancing the sercos Slave Driver (SSLV)

The SSLV provides two different tasks. On the one hand, it connects to the field-bus system to exchange data with the PLC System. On the other hand it is used to connect to the simulation and exchange data with the simulation. Since both tasks have critical timing behaviour they are executed separately in two threads within the SSLV. To move data from one thread to another they write into global data structures. One thread is responsible for covering sercos communication and the other thread handles the data exchange with Virtuos. As a first step, the sercos interface of the IDN database of the SSLV was enhanced to emulate the bus-interface of a standard off the shelf bus-coupler with just 16 bits of IO-data. This is an error prone process since there is no description which IDNs are retrieved and evaluated by the PLC during the start-up phases. The fields for cyclic real-time data were extended afterwards to the size of 512 bytes as specified in the custom SDDML file of the device. In the final configuration 512 bytes of data are transferred from the PLC to the simulation and the same from the simulation to the PLC in every communication cycle. Data composition and decomposition is also done by the communication thread. Listing 2 shows the specification of a data packet in C source code:

```
typedef struct {
  double doubles[32];
  int  ints[64];
} io_type;

Listing 2: Specification of a data packet in the SSLV
```

Luckily, the compiler for the PLC code and the GNU-C compiler use the same method of storing data. To decompose the data packet back into structures of variables a pointer to a byte array can be used. The pointer has to be casted into a pointer of type io_type and vice versa. To connect the SSLV to the running simulation and to synchronise a separate Virtuos-IO (VIO) thread is used. The VIO thread has the responsibility to exchange data with the running Virtuos simulation and to trigger simulation steps from outside. For purposes of synchronisation two semaphores are deployed. Figure 7 shows the execution model as simple Gantt-diagram (without the running communication thread).

```
FIGURE 7: Execution model of the simulation and the VIO-thread
```

The execution model of both processes is very similar to those of a traditional producer-consumer model. Two semaphores are deployed: The "start"
semaphore has the purpose to signal the beginning of a simulation step, the "end" semaphore signals the end of a simulation step. The VIO thread is started at $T_N$. When the VIO thread has completed its data transfer to the simulation, the simulation is started. The simulation executes one simulation cycle and signals the end of the cycle to the VIO-thread. Since the exact simulation time varies from application to application, the VIO-thread does not start data transfer immediately but sleeps until the next $T_{N+1}$ to be in time with the other parts of the system.

6 Conclusion

This paper presents how RT-Preempt Linux can be used for real-time simulation and the virtual commissioning of production machines. A proprietary PLC is connected to a simulation PC which executes the real-time simulation model. The automation bus sercos III is used to transfer data in a deterministic manner between PLC and simulation PC. To adapt the simulation model to the field-bus, its interface is hidden behind the interface of a bus-coupler device. For this purpose a sercos III PCI networking card is utilised. The driver of the card is enhanced to emulate the interface of a bus-coupler and to transfer data between the bus and the simulation model. The simulation model is executed by the simulation software Virtuos on the simulation PC. With this setup, PLC programs for controlling production machines, which need run on their (proprietary) and unmodified target hardware can be tested by means of simulated mechatronic hardware. Accordingly, the time to market of a production machine can be reduced by parallelisation of development tasks. As testing of programs which control or depend on mechanical hardware can be tested without the real hardware to be available.

7 Future Work

At current, only one field-bus device can be emulated by the PCI card. This is why a small hardware abstraction layer in the PLC code is necessary to switch from simulated to real hardware. In a follow-up project, the emulation of more than one sercos III devices will be possible. Consequently, the simulation may cover more than one field-bus devices at one. As a result it will, be feasible to switch between simulated hardware and real hardware without the need for any changes in the PLC.

8 Acknowledgement

The authors would like to thank the German Research Foundation (DFG) for financial support of the projects within supporting the Graduate School of Excellence advanced Manufacturing Engineering (GSaME) at the University of Stuttgart.

References


