52. IWK

Internationales Wissenschaftliches Kolloquium
International Scientific Colloquium

PROCCEDINGS | **10 - 13 September 2007**

FACULTY OF COMPUTER SCIENCE AND AUTOMATION

COMPUTER SCIENCE MEETS AUTOMATION

VOLUME II

- Session 6 - Environmental Systems: Management and Optimisation
- Session 7 - New Methods and Technologies for Medicine and Biology
- Session 8 - Embedded System Design and Application
- Session 9 - Image Processing, Image Analysis and Computer Vision
- Session 10 - Mobile Communications
- Session 11 - Education in Computer Science and Automation

TECHNISCHE UNIVERSITÄT
ILMENAU

Bibliografische Information der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der deutschen
Nationalbiografie; detaillierte bibliografische Daten sind im Internet über
http://dnb.ddb.de abrufbar.

ISBN 978-3-939473-17-6

Impressum

Herausgeber: Der Rektor der Technischen Universität Ilmenau
Univ.-Prof. Dr. rer. nat. habil. Peter Scharff

Redaktion: Referat Marketing und Studentische Angelegenheiten
Kongressorganisation
Andrea Schneider
Tel.: +49 3677 69-2520
Fax: +49 3677 69-1743
e-mail: kongressorganisation@tu-ilmenau.de

Redaktionsschluss: Juli 2007

Verlag:
Technische Universität Ilmenau/Universitätsbibliothek
Universitätsverlag Ilmenau
Postfach 10 05 65
98684 Ilmenau
www.tu-ilmenau.de/universitaetsverlag

Herstellung und Auslieferung: Verlagshaus Monsenstein und Vannerdat OHG
Am Hawerkamp 31
48155 Münster
www.mv-verlag.de

Layout Cover: www.cey-x.de

Bezugsmöglichkeiten: Universitätsbibliothek der TU Ilmenau
Tel.: +49 3677 69-4615
Fax: +49 3677 69-4602

Preface

Dear Participants,

Confronted with the ever-increasing complexity of technical processes and the growing demands on their efficiency, security and flexibility, the scientific world needs to establish new methods of engineering design and new methods of systems operation. The factors likely to affect the design of the smart systems of the future will doubtless include the following:

- As computational costs decrease, it will be possible to apply more complex algorithms, even in real time. These algorithms will take into account system nonlinearities or provide online optimisation of the system's performance.
- New fields of application will be addressed. Interest is now being expressed, beyond that in "classical" technical systems and processes, in environmental systems or medical and bioengineering applications.
- The boundaries between software and hardware design are being eroded. New design methods will include co-design of software and hardware and even of sensor and actuator components.
- Automation will not only replace human operators but will assist, support and supervise humans so that their work is safe and even more effective.
- Networked systems or swarms will be crucial, requiring improvement of the communication within them and study of how their behaviour can be made globally consistent.
- The issues of security and safety, not only during the operation of systems but also in the course of their design, will continue to increase in importance.

The title "Computer Science meets Automation", borne by the 52nd International Scientific Colloquium (IWK) at the Technische Universität Ilmenau, Germany, expresses the desire of scientists and engineers to rise to these challenges, cooperating closely on innovative methods in the two disciplines of computer science and automation.

The IWK has a long tradition going back as far as 1953. In the years before 1989, a major function of the colloquium was to bring together scientists from both sides of the Iron Curtain. Naturally, bonds were also deepened between the countries from the East. Today, the objective of the colloquium is still to bring researchers together. They come from the eastern and western member states of the European Union, and, indeed, from all over the world. All who wish to share their ideas on the points where "Computer Science meets Automation" are addressed by this colloquium at the Technische Universität Ilmenau.

All the University's Faculties have joined forces to ensure that nothing is left out. Control engineering, information science, cybernetics, communication technology and systems engineering – for all of these and their applications (ranging from biological systems to heavy engineering), the issues are being covered.

Together with all the organizers I should like to thank you for your contributions to the conference, ensuring, as they do, a most interesting colloquium programme of an interdisciplinary nature.
I am looking forward to an inspiring colloquium. It promises to be a fine platform for you to present your research, to address new concepts and to meet colleagues in Ilmenau.

Professor Peter Scharff
Rector, TU Ilmenau

Professor Christoph Ament
Head of Organisation

Table of Contents

CONTENTS

Page

6 Environmental Systems: Management and Optimisation

7 New Methods and Technologies for Medicine and Biology

8 Embedded System Design and Application

9 Image Processing, Image Analysis and Computer Vision

10 Mobile Communications

11 Education in Computer Science and Automation

52nd Internationales Wissenschaftliches Kolloquium
Technische Universität Ilmenau
10 – 13 September 2007

A. Pretschner / J. Alder / Ch. Meissner

A contribution to the design of embedded control systems

1. Introduction

The area of automated control engineering increasingly tends towards distributed autonomous systems. Thus rises the requirement for control modules that can be implemented in a simple and cost-effective way by means of ordinary PC technology. An obvious case is reliable Linux-based systems. Such systems, also in form of embedded systems, represent an alternative to the conventional control systems. This development is accelerated by the rapid growth of the hard and software components. The introduction of the International Electronical Commission 61131-3 (international standard of modern software programming of programmable logic controllers) has likewise a positive influence on the use of off-the-shelf PC technology in a market that has been dominated by proprietary industrial technology. Even the IEC 61499 provides a framework and architecture for describing the functionality in distributed control systems. The engineering tool [3, 5] can be used to fill out these standards with executable software. With the developed software tool [6] the software design process of automatic controls can be accomplished in particular for users of embedded systems error free. The Tool- Chain from the process behavior to the process visualization, developed by the example of the ARM9 processors, illustrates the typical engineering process for control-system applications.

2 Preceding work

For the automated design of control programs several approaches exist. We choose an automata-theoretic approach as the basis of our design tool. The structure of the program package is governed by the IEC 61131 standard. Access to variables is implemented by a data base, declaration of classes is implemented by function blocks, and their instantiation and integration into the program are also implemented. The system requirements are described and converted into an appropriate software tool. Transfer to practice is done using conventional control applications. The intensified employment of distributed automation systems led to the use and to the design of control systems based on PC and/or microcontroller. The necessary information of the description of the software packages are referred on the home site of the project. Among them are both a detailed on-line assistance, and the system manual as pdf file [4]. The executable files of the project are freely available.

3 Function block design

The PLC is provided as automatically generated code by the design tool SPaS [3,4]. The graphic editor enables to create process flow charts. Each process flow chart syntactically corresponds to a function block. The function block design corresponds to the design of the IEC 61131-3 function block design

The behavior of the function block is defined in terms of the algorithms and state information. In extension to the IEC 61499 the algorithms are expressed as state machines and the Execution Control Chart is coupled with the state machine. The *WITH* qualifier will be omitted with assumption of a sequential behavior of the scheduler

All function blocks have to be programmed completely by graphics. The project structure in the design tool is governed by the IEC 61131-3 structure guidelines, i.e., there are project nodes for the resource, the programs, as well as for the needed instances. The function blocks can be linked by simply connecting event / data flow connections between block input and output variables. A special graphical net-editor does not exists at time yet. The development is under way. Linking the function block instances by variables you have to use a text editor manually. The execution order of the function blocks is determined by the user choosing the appropriate level in the project tree. The IEC 61131-3 software model does consider configuration that have multiple resources. To communicate between them you can use global variables or communication function blocks (IEC 61131-5)

From the graphic representation, syntactically correct code (in this case C and C++) is produced. This code may be integrated as an independent task. Here also the binding to the I/O of the controller takes place. The design is transparent and permits the user to map to equivalent source code which later has to be compiled to binary code.

For the start-up of the control a back representation of the graph can be created. It can visualize at run time the current state of the automaton.

4 Extension to distributed systems

Applying to distributed systems you have to take care about the execution order. Unfortunately, such mechanisms are not defined in the IEC61131-3. The main restriction using the IEC 61131-3 software model is described as following [22]:

- Applications in the IEC 61131-3 model are not distributable over multiple resources.
- The function block execution order is not always clearly defined.

The function blocks expressed in IEC 61131-3 language can be encapsulated as IEC 61499 function blocks as a first approach.

The application has to be separated into different resources necessarily at this modeling stage. Further you have to bring in communication function blocks to the modell. In this application a interface function block was designed. It is an composite function block containing appropriate Publisher / Subscriber function blocks and / or IO_Writer / IO_Read function blocks. This modeling manner is often sufficient for ordinary PLC's, which are driven sequentially by their operating system. The execution order of the function blocks depends on the calling level in the main thread of the operation system. The execution control of the composite function blocks may be deferred to the mechanisms described by the IEC 61499.

The execution order of the function blocks and the validity of the sampled data is not clearly defined for more advanced applications realized with microcontrollers. You have to provide a execution control chart (ECC) to the composite function block at each resource to achieve a well defined coordination of the net behavior. Service interface function blocks may be necessarily to communicate to the devices.

The deployment of the software modules on distributed resources is possible. The use of the flow Charts of " SPaS" in connection with the function block model of the IEC 61499 allow the administration of several resources in one project. Communication between the resources is not the subject of the draft process.

5 Communication device

Once the program is translated to *C* or *C++* the communication to the inputs/ outputs of the process has to be established. This is done via a common bus system: *CAN*. Using the standard CANopen application interface, the system provides the master slave-behavior between the master control device (this) and the I/O devices. To obtain a modular system, we have chosen the USB-interface as breakout to the periphery. We developed several layers, which handle bus protocols and software interfaces to provide an easy high abstracted interface of I/O access (see figure 1). Even you can see the software module deployment between the kernel and the user space of Linux.

The first two layers present theirself as linux kernel modules. First *CANu* connects to the *CAN* controller device and distributes *CAN* messages and second *CANopenDrv* implements the *CANopen* protocol. *CANu* receives incoming *CAN* messages and delivers them to specific applications, which are registered on this module. With this approach, the applications, which are user space programs or other kernel modules, share one ressource. Internal *CANu* follows completely the idea of *CAN* to broadcast messages, because a message received by a *CAN* controller is delivered to every on this device registered application and the other way around. The advantage of this is, that the limited access of one device is now overridden and an abstracted *CAN* Interface is provided. *CANopen Drv* is a virtually *CANopen* Master/Sla ve conforming the CiA 301 DS V4. 0.2 specification. It's protocol defines broadcast and client-server communication objects by partitioning the available range of *CAN* identifiers. It processes received messages and handles network events and states. According to this the I/O data is provided through the internal object dictionary, which can be accessed by the control application. The control application itself runs in userspace and communicates with the *CANopen* driver over syscalls. That means, all important time critical *CANopen* protocol functions are running in kernelspace with very low latency. To improve the handling, an additionally developed C library is provided as a third layer, which makes it easier to use the *CANopen* driver. On top of this another C++ library is implemented to encapsulate the functionality one more time. Based on this, it is possible to derive own classes to be used in the control applications *ioplc* interface. The control program is developed as a plugin module of the Lintouch server application. If the control system application is divided in parts over the communication network, each intelligent network node should correspond to these hard and software requirements.

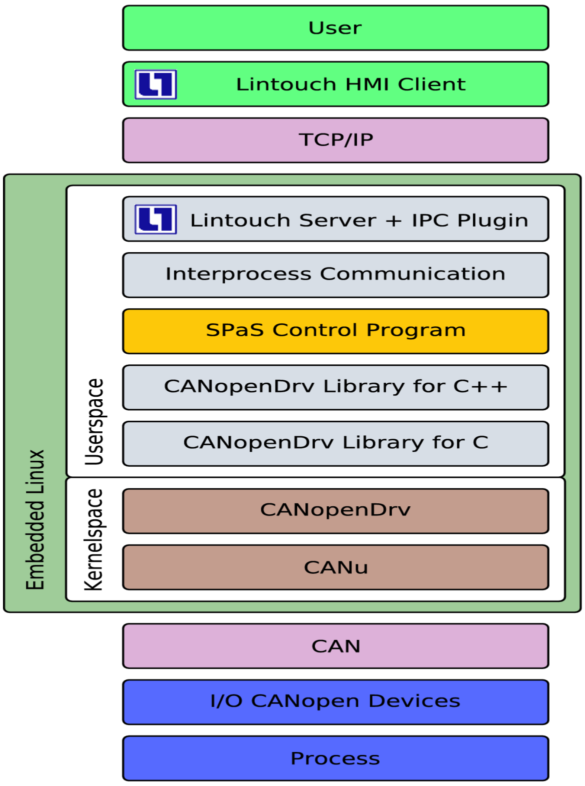

Figure 1: Software modules and communication layers

6 Conclusion

As you can see the aim to provide an easy way to access the data of I/O devices in the *CANopen* network is fulfilled. In order to that it is also possible to exchange *CAN* controllers and I/O devices, without changing the control applications code. The next step is the communication between the control program and the user. This is done by the open source human machine interface (HMI) *Lintouch*. It contains a TCP/IP server on the control system that collects and distributes process variable values. A user is now able to connect from a plugged ethernet network to this server. That lintouch client downloads a previous generated lintouch project file from the server, which provides an interface that displays the current process state and allows the user to interact with the process. This interface could run on a panel or something else. To gather the current process variables the lintouch server uses plugins, which are building the bridge between the server and the process. In this case we developed our own, which is able to receive and transmit data from the *SPaS* control program. Our decision leads to interprocess (OS process) communication. This means, that we use the *System V IPC* of linux, especially the shared memory capabilities, to distribute data to the server and the other way around. Because *SPaS* uses data vectors as data input, output and memory, we followed this approach and map these vectors to shared memory. The plugin now reads and converts the data to lintouch conform formats. The interaction with the process only happens via the memory vector, so that the direct feedthrough setting of output variables is not possible.

Because of the high abstraction of the I/O access and process visualization, it is imaginable easy to exchange the used control program, which also can be generated very easy by the *SPaS* software.

References:

[1] IEC 61131-3. Programmable controllers - part 3: Programming languages. Technical report, DIN Deutsches Institut für Normung e.V., 2004.

[2] Robert Lewis. Modelling control systems using IEC 61499. IEE Engineering Series 59. IEEE, The Institution of Electrical Engineers, 2001.

[3] A. Pretschner. Softwareprojektierungsumgebung für automatische Steuerungen SPAS. In International Congress SPS/IPC/Drives Nürnberg 2000, pages S. 99–103, 2000.

[4] A. Pretschner. Spas - online - dokumentation. Technical report, University of Applied Sciences Institut for Prozessinformation Technology and Controls Systems, 2005.

[5] VDI/VDE 3681. Einordnung und Bewertung von Beschreibungsmitteln aus der Automatisierungstechnik. Technical report, VDI/VDE-Handbuch Regelungstechnik, 2004.

[6] Alder,J.;Pretschner,A.:Prozess-Steuerungen- Projektierung und Inbetriebnahme mit dem Softwaretool SpaS, Springerverlag 2007, ISBN 978-3-540-71083-7

Authors:
Prof. Dr. Andreas Pretschner
Dr.-Ing. Jochen Alder
HTWK Leipzig, FBEIT, PSF 30 11 66.
04251, Leipzig
Phone:+49 341 3076 1135
Fax:+49 341 3076-1243
E-mail:andreas@pretschner.com