A Smart Transducer Interface Based on an Interface File System

Wilfried Elmenreich and Wolfgang Haidinger
Institut für Technische Informatik
TU Vienna
Vienna, Austria
Fax: +49 (89) 24433-3633
{wil,wh}@vmars.tuwien.ac.at

ABSTRACT –

Smart transducers appear to become a key mechanism in fieldbus communications. However, efforts on the standardization of fieldbus systems are going on since for a few years without much success.

A recent call for a smart transducer interface proposal has been issued by the Object Management Group. In this paper, we discuss a proposed smart transducer interface that has been proposed in response to this request.

The key feature of the interface is the concept of an Interface File System (IFS) that contains all relevant transducer data. This IFS allows different views of a system, namely a real-time service view, a diagnostic and management view, and a configuration and planning view. The interface concept encompasses a communication model for transparent time-triggered communication.

Implementation experiences presented in this paper show that the IFS reduces the complexity of fieldbus applications and supports software reuse in other fieldbus nodes.

KEY WORDS: *Smart Transducer Interface, Interface File System, Architectures, Embedded Systems.*

1 Introduction

Sensors and actuators enable the interaction of a computer system with its environment. These sensors and actuators, the collective term is *transducer*, are usually placed spatially apart and need to be connected to the main processor. Earlier transducer systems used point-to-point wiring for these connections, but with the advent of embedded microcontrollers a comfortable solution with a transducer bus system is preferable.

In 1982 Wen H. Ko and Clifford D. Fung introduced the term "intelligent transducer" [Ko82]. An intelligent or *smart* transducer is the integration of an analog or digital sensor or actuator element and a local microcontroller that contains the interface circuitry, a processor, memory, and a network controller in a single unit. The smart sensor transforms the raw sensor signal to a standardized digital representation, checks and calibrates the signal, and transmits this digital signal via a standardized communication protocol to its users [Kop01a].

The design of the network interface of the smart transducer is of great importance. Transducers can come in a great variety with different capabilities from different vendors. A smart transducer interface must thus be very generic to support all present and future types of transducers. However, it must provide some standard functionalities to transmit data in a temporal deterministic manner in a standard data format, provide means for fault tolerance, and enable a smooth integration into a transducer network and its application.

Furthermore, such a smart transducer interface shall conform to a world-wide standard. A standard for a real-time communication network has been long sought, but efforts to find one agreed standard have been hampered by vendors, which were reluctant to support such a single common standard in fear of losing some of their competitive advantages [Pin95]. Hence, several different fieldbus solutions have been developed and promoted. Some of these existing solutions have been combined to form standards. In 1994, the two large fieldbus groups ISP (Interoperable Systems Project supported by Fisher-Rosemount, Siemens, Yokogawa, and others) and the WorldFIP (supported by Honeywell, Bailey, and others) joined to form the Fieldbus Foundation (FF). It is the stated objective of the FF to develop a single interoperable fieldbus standard in cooperation with the International Electrotechnical Commission (IEC) and the Instrumentation Society of America (ISA).

The IEC worked out the IEC 61158 standard. It is based on eight existing fieldbus solutions. However, the IEC fieldbus draft standard was not ratified at the final approval vote, following a set of controversies [Nou99]. The IEC 61158 has the great disadvantage that it still keeps a diversity of eight different

solutions.

The ISA, which developed the SP50 standard and IEC committees met jointly to make the development of an international standard possible. ISA SP50 was the same committee that introduced the 4-20 mA standard back in the 1970s.

Meanwhile, other standards for smart transducers were developed. The IEEE 1451.2 [Con00] standard deals with the specification of interfaces for smart transducers. An idea proposed by this standard is the specification of electronic data sheets to describe the hardware interface and communication protocols of the smart transducer interface model [Ecc98].

The Object Management Group (OMG), an international standardization body for the Common Object Request Broker Architecture (CORBA), called for a proposal on a smart transducer interface to a CORBA system in December 2000 [OMG00]. In response to this request an architecture that provides a gateway between a CORBA Object Request Broker (ORB) and a smart transducer subsystem has been submitted jointly by three companies under the support of the Vienna University of Technology [Obj01]. The interface to the smart transducers uses the concept of a file system to fulfil the requirements for smart transducer interfaces discussed above.

It is the objective of this paper to describe and discuss the above mentioned concept of an interface file system for a smart transducer interface.

The remainder of the paper is organized as follows: Section 2 examines the abstract properties and requirements of a smart transducer interface. Section 3 explains the concept of the Interface File System as proposed in [Obj01]. Section 4 describes the implementation of the Interface File System in a smart transducer fieldbus. Section 5 presents some implementation experiences. The paper is concluded in Section 6.

2 Abstract Interface Properties

An interface is a common boundary between two subsystems. An information exchange across an interface is only possible if the engaged subsystems share a common background of concepts and a common coding system. In the context of a distributed control system, the smallest area of concern is a cluster, consisting of a set of sensors, actuators, and processing nodes connected by a communication medium.

In the abstract, the purpose of a smart transducer interface is the timely exchange of "observations" between two subsystems. In the following Sections we will investigate the encoding, the properties of an observation, methods for communication flow control and possible interface types.

Common Code Spaces

Communication is only possible, if the interfacing subsystems share the concepts and representation of the data items in the interface. To exchange observations across a smart sensor interface, agreement on three code spaces must be provided [Kop01a]:

The Name Space is necessary for the meaning of transmitted values.

The Time Space serves for the definition of an instant of an event among the communicating nodes.

The Value Domain provides encoding schemes for the transmitted values.

One key task in the development of a generic smart transducer model is concerned with the specification of these code spaces and the meaning of the referenced elements.

Observations

In a real-time system, "observations" of real-time entities need to be communicated between the engaged subsystems across the provided interfaces. A real-time entity is a state variable of interest that has a name and a value at a particular instant. An observation [Kop97a] is thus an atomic triple:

$$<\textit{name, observation instant, value}>,$$

where *name* is an element of the common name space of real-time entities, the *observation instant* is a point in the time space and *value* is an element of the chosen domain of values. An observation thus states that the referenced real-time entity possessed the stated value at the indicated instant.

If the value of an observation does not change over the time interval of interest, we call this observation a timeless observation. Timeless observations can be represented by value pairs, consisting only of an entity name and the associated value.

Flow Control

Communication between subsystems exchanges information in two distinct domains, the *time domain* and the *value domain*. In the value domain the message data is transmitted, while in the time domain *control information* is exchanged [Krü97]. Control information allows the generating subsystem to influence the *temporal control flow* [Kop97a] of the other subsystem.

Commonly, a communication between two subsystems is either controlled by the sender's request (*push* style) or by the receiver's request (*pull* style) [Alc00].

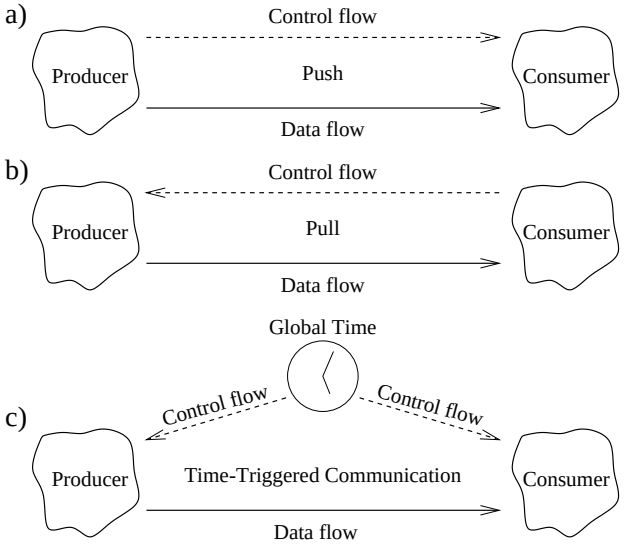

Figure 1: Push-, Pull-, and Time-Triggered Communication

For explanation, let us assume two components need to exchange data over a network. Further, without restrictions to generality, we assume the message data to be transmitted from a *producer* to a *consumer*.

In order to transfer data between two components, they must agree on the mechanism to use and the direction of the transfer.

Figure 1 a) shows the *push* method. The producer is empowered to generate and send its message at any time. Flow control relies on the producer. This method is very comfortable for the push supplier (producer), but the receiving push consumer has to be watchful for incoming data messages at any time, which may result in high resource costs and difficult scheduling. Popular "push" mechanisms are: messages, interrupts, or writing to a file [Del99]. The push style communication is the basic mechanism of event-triggered systems.

In the *pull* model depicted in Figure 1 b) the flow control is on the consumer. Whenever the consumer wants to access the message information, the producer has to respond on request. This facilitates the task for the pull consumer, but the pull supplier (producer) has now to be watchful for incoming data requests. Popular "pull" mechanisms are reading a file, polling, state messages, or shared variables [Del99]. The pull style communication is the basic mechanism of client-server systems.

Figure 1 c) depicts a *time-triggered* communication model where the flow control is achieved by a predefined instants of a global time.

Figure 2 depicts a communication model where the flow control of producer and consumer is decoupled by a *temporal firewall* [Kop97b]. This model uses a combination of all three flow control techniques depicted

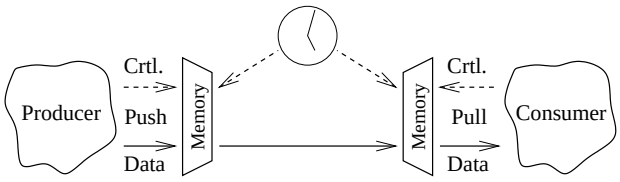

Figure 2: Decoupled Flow Control

in Figure 1.

Each component possesses a memory object that acts as a data source and sink for communication activities. Components that want to submit data are able to write the data into this memory using a *producer's push* interface. The transmission of data between message data is handled by a time-triggered communication model. After transmission the consumer component accesses the data using a *consumer's pull* interface.

The values in the memory are state messages that keep their content until they are updated and overwritten [Kop01b]. The critical ends of the push and the pull communication rely on the memory elements. However, because these elements are usually passive components, the negative effects of the push and the pull communication do not affect the system performance.

To avoid interference between concurrent read and write operations on the memory element, the task of the communication system is done by a time-triggered protocol like TTP/A [Kop00b] or TTP/C [Kop99b]. Since in the time-triggered architecture all nodes have knowledge about transmission schedules and access to a global time base, the instant when the protocol updates a value in the memory element is known to all components.

Interfaces for Smart Transducers

The smart sensor technology offers a number of advantages from the points of view of technology, cost, and complexity management [Kop99a]:

- Electrically weak non-linear sensor signals can be conditioned, calibrated and transformed into digital form on a single silicon die without any noise pickup from long external signal transmission lines [Die98].
- The smart sensor contains a well-specified digital communication interface to a sensor bus, offering "plug-and-play" capability if the sensor contains a reference to its documentation in form of an electronic data sheet as it is proposed in the IEEE 1451.2 Standard [Ecc98].
- It is possible to monitor the local operation of the sensing element via the network and thus simplify

the diagnosis at the system level.

The internal complexity of the smart-sensor hardware and software and internal failure modes can be hidden from the user by well-designed fully specified smart sensor interfaces that provide just those services that the user is interested in.

In order to support complexity management and composability, it is useful to specify distinct interfaces for functional different services [Kop00a]. As depicted in Figure 3, a smart transducer node can be accessed via three different interfaces:

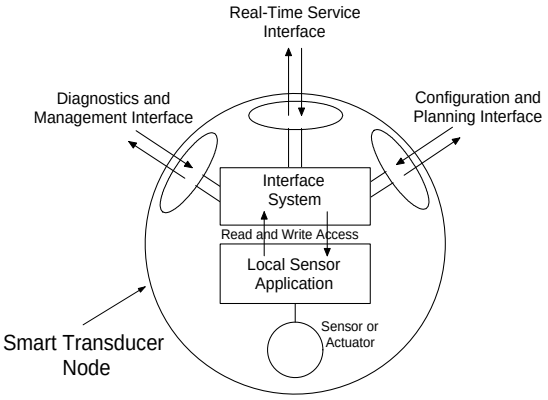

Figure 3: Three Interfaces to a Smart Transducer Node

Real-Time Service (RS) Interface: This interface provides the timely real-time services to the component during the operation of the system.

Diagnostic and Maintenance (DM) Interface: This interface opens a communication channel to the internals of a component. It is used to set parameters and to retrieve information about the internals of a component, e. g., for the purpose of fault diagnosis. The DM interface is available during system operation without disturbing the real-time service. Usually, the DM interface is not time-critical.

Configuration and Planning (CP) Interface: This interface is necessary to access configuration properties of a node. During the integration phase this interface is used to generate the "glue" between the nearly autonomous components. The CP interface is not time-critical.

Although the three interfaces represent different services it is possible to implement them using the same concepts, e. g. route the communication over the same bus cable. Another common concept of the three interfaces is the interface system depicted in Figure 3. The next section will explain a unique addressing scheme for this interface system.

3 The Interface File System

The Interface File System (IFS) [Kop01a] provides the name space for a smart transducer interface. It is a unique addressing scheme for memory containing the following data:

Transducer data: Transducer data means sensor measurements and/or set values for actuators. Sometimes it is also useful to keep a history on former measurements for sensor validation or state estimation. These data is usually frequently transmitted and accessible via the RS interface.

Configuration data: These data can comprise settings for communication modes, transducer operation, and fault-tolerance actions. It is accessed via the CP interface.

Self-describing information: These data contain either locally stored information about the node's type, capabilities, etc. or a reference to a transducer electronic data sheet somewhere outside the fieldbus network, e. g. on the vendors web page. Self-describing information is accessed via the CP interface.

Internal state information: For maintenance and diagnostic purposes it is often comfortable to have a look into local variables which are not transmitted over the bus during real-time service operation. So this internal state information is mapped into the node's file system and accessible via the CP and DM interface.

The values of a transducer network that are mapped into the IFS are organized in a static file structure. The address space is organized hierarchically representing the network structure:

Cluster name: The cluster name is an 8 bit integer value that selects a cluster which is a network of fully interconnected nodes. Native communication (without routing) among nodes is only possible within the same cluster.

Node Alias: The node alias or logical name selects a particular node. Node aliases can have values from 0...255, but some values have a special meaning, e. g. alias 0 addresses all nodes of a cluster in a broadcast manner.

File Name: The file name addresses a certain file within a node. A file name consists of a 6-bit identifier. Some file names have a special meaning in all nodes. Each file has a statically assigned number of records. The file information can be located in ROM or RAM memory or even generated at runtime. Each node has a local file map

	Timeslot 1	Timeslot 2	Timeslot 3
Node A	Send byte 1 of file 10 record 2		Receive and store in byte 1 of file 12 record 3
Node B	Receive and store in byte 1 of file 12 record 1	make local calculation based on byte 1 of file 12 record 1 and store result in byte 2 of file 14 record 1	Send byte 2 of file 14 record 1
Node C			Receive and store in byte 3 of file 19 record 6

Table 1: Example for the Semantics of a Communication Schedule (all IFS addresses are local)

containing the file lengths. The first record of each file is called the header record and contains information about the file.

Record Number: The record number addresses the record within the selected file. Each record contains 4 data bytes. The record number is an 8-bit identifier, but addressing a non-existing record of a file yields an error. Therefore the length of each file is stored read-only in the first record of every file.

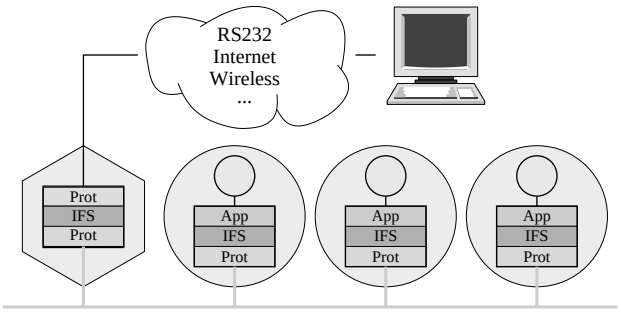

Figure 4: Physical Network Topology

Figure 4 depicts the topology of a smart transducer cluster. All transducer nodes are built as smart transducers and contain a physical sensor or actuator, a microcontroller, and a network interface. The local application that accesses the sensor or actuator uses the IFS as a data source and sink. The protocol will perform a periodical time-triggered communication to copy data from the IFS to the fieldbus and write received data into the IFS. Thus, the IFS acts as an interface that decouples the local transducer application from the communication task. The leftmost node in the figure depicts a gateway node that connects the fieldbus to a different network. The gateway node contains also a protocol and IFS part, but instead of the local application it contains a different protocol part that establishes a connection to another system via a serial, internet, or wireless communication.

It is the task of the protocol to keep consistency among the local copies of the IFS data elements. A predefined communication schedule defines time, origin, and destination of each protocol communication. A possible communication schedule is depicted in Table 1.

The programmer's view of the network can be simplified by abstracting over the protocol communication. Thus, any application "sees" just the IFS in the logical network structure depicted in Figure 5.

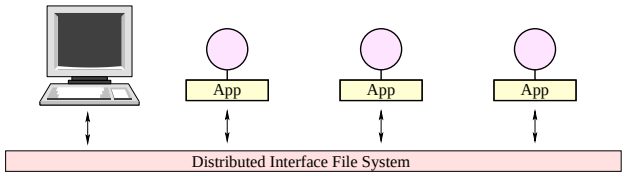

Figure 5: Logical Network Structure

4 An IFS Case Study: TTP/A

TTP/A is a time-triggered master/slave communication protocol for fieldbus applications that uses a time division multiple access (TDMA) bus arbitration scheme [Kop00b]. It is possible to address up to 250 nodes on a bus. One single node is the active master. This master provides the time base for the slave nodes. The communication is organized into rounds. Bus access conflicts are avoided by a strict TDMA schedule for each round. A round consists of several slots. A slot is a unit for transmission of one byte of data. Data bytes are transmitted in a standard UART format. Each communication round is started by the master with a so-called fireworks byte. The fireworks byte defines the type of round.

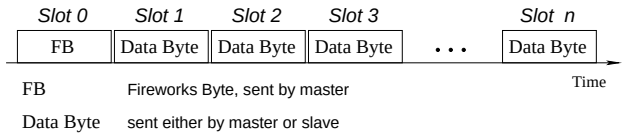

Figure 6: A TTP/A Round

A TTP/A round (see Figure 6) consists of a configuration dependent number of slots and an assigned

sender node for each slot. The configuration of a round is defined in the RODL (ROund Descriptor List). The RODL defines which node transmits in a certain slot, the semantics of each individual slot, and the receiving nodes of a slot. RODLs must be configured in the slave nodes prior to the execution of the corresponding multipartner round.

A master/slave round is a special round with a fixed layout that establishes a connection between the master and a particular slave for reading/writing monitoring or configuration data, e. g., the RODL information. In a master/slave round, the master addresses a data record in the IFS format and specifies an action like reading, writing or executing on that record.

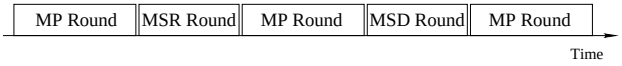

Figure 7: Recommended TTP/A Schedule

The master/slave rounds establish the DM and CP interface to the transducer nodes. Master/slave rounds are intended to be scheduled periodically between multipartner rounds as depicted in Figure 7.

The TTP/A protocol uses the Interface File System as global name space for all relevant data of a node like communication schedules, calibration data, and I/O properties. All nodes contain several files with a number of records that can contain information for automatic configuration, similar to the idea of Transducer Electronic Data Sheets in the IEEE 1451.2 standard [Ecc98].

The time space in TTP/A is supported by a global time base shared by all nodes of a cluster. It is also possible to synchronize the clocks of nodes in multiple clusters to a given precision [Pau01]. TTP/A supports three time formats for a representation of an observation instant. A simple time format is defined by a combination of a round epoch counter, the slot number of the current slot, and the bit number, if necessary. If an instant has to be defined more exact, TTP/A offers an external and an internal time format. The external time format uses 64 bit and describes the number of ticks since the start of GPS time (January 6, 1980). The external time format has a horizon of 2^{40} seconds, i. e. more than 30 000 years and will not wrap in the foreseeable future. One tick has a granularity of 2^{-24} seconds, i. e. about 60 ns. The internal time format contains the lower 32 bit of the external time format.

The value space in TTP/A is realized by a well-defined representation of data values. Besides the measurement value, TTP/A supports the transmission of error codes and confidence values with the value. Confidence values support the implementation of sensor fusion algorithms [Elm01a].

5 Implementation Experiences

We implemented the Interface File System together with the TTP/A protocol in smart transducer systems for a mobile robot car [Pet01, Sch01] and a robot arm with a medical arm prosthesis [Obe01].

The protocol was implemented on various hardware platforms, like Atmel 8-Bit RISC [Pet00], Microchip PIC [Obe01], and Motorola MC68376. The overhead for the file system implementation was about 600 Byte flash ROM code space (a reference implementation on an ATMEL AT90S2313 showed that the stand-alone TTP/A protocol needs about 1,2 kBytes of ROM, while the integration of the filesystem resulted in 1,8 kBytes). Despite this extra memory requirement complexity of the control application program was reduced.

The IFS was used as a universal interface – between configuration/maintenance tools and the TTP/A network – and among local node applications.

The structure given by the filesystem eased also the reuse of code pieces in other TTP/A nodes – even if they were running on a different protocol implementation. A recent work on the integration of plug and play functionality [Elm01b] also showed, that the concept of the filesystem supports the integration of new functionality to the protocol.

6 Conclusion

We presented the concept of an Interface File System (IFS) for smart transducers. This IFS allows different views of a system, namely a real-time service view, a diagnostic and management view, and a configuration and planning view. The interface concept encompasses a communication model for transparent time-triggered communication.

The IFS establishes the name space for the smart transducer interface. Together with a standard representation of observation instants and a common interpretation of values, a common code space is created that enables communication among smart transducers. We have implemented the IFS within a time-triggered fieldbus protocol (TTP/A) network.

Implementation experiences have shown that despite the afford for the implementation of the IFS, fieldbus applications can be set up with less complexity compared to ad-hoc approaches without a global addressing structure. Furthermore using the IFS addressing scheme eases the reuse of application software in other fieldbus nodes.

The described Interface File System and the time-triggered communication are part of an OMG standard proposal. It is most likely that the presented work becomes a world-wide fieldbus standard.

Acknowledgments

This work was supported in part by the Austrian Ministry of Science, project TTSB and by the European IST project DSoS under contract No IST-1999-11585.

References

[Alc00] Alcatel Corp., Fujitsu Ltd., IBM, NEC Corp., NTT Corp., and IONA Tech. Management of Event Domains. *OMG TC Document telecom/2000-01-01*, Jan. 2000. Available at http://www.omg.org.

[Con00] P. Conway, D. Heffernan, B. O'Mara, P. Burton, and T. Miao. IEEE 1451.2: An Interpretation and Example Interpretation. *Proceedings of the Instrumentation and Measurement Technology Conference*, pages 535–540, 2000.

[Del99] R. Deline. *Resolving Packaging Mismatch.* PhD Thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, June 1999.

[Die98] P. Dierauer and B. Woolever. Understanding Smart Devices. *Industrial Computing*, pages 47–50, 1998.

[Ecc98] L.H. Eccles. A Brief Description of IEEE P1451.2. *Sensors Expo*, May 1998.

[Elm01a] W. Elmenreich and S. Pitzek. Using Sensor Fusion in a Time-Triggered Network. In *Proceedings of the 27th Annual Conference of the IEEE Industrial Electronics Society, Denver, Colorado*, volume 1, Nov.-Dec. 2001.

[Elm01b] W. Elmenreich, W.Haidinger, P. Peti, and L. Schneider. New Node Integration for TTP/A Networks. Technical Report 5, Technische Universität Wien, Institut für Technische Informatik, 2001.

[Ko82] W.H. Ko and C.D. Fung. VLSI and Intelligent Transducers. *Sensors and Actuators*, (2):239–250, 1982.

[Kop97a] H. Kopetz. *Real-Time Systems, Design Principles for Distributed Embedded Applications.* Kluwer Academic Publishers, Boston, Dordrecht, London, 1997.

[Kop97b] H. Kopetz and R. Nossal. Temporal firewalls in large distributed real-time systems. *Proceedings of the 6th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS '97)*, 1997.

[Kop99a] H. Kopetz. Do Current Technology Trends Enforce a Paradigm Shift in the Industrial Automation Market? *Closing Keynote at the 7th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 99), Barcelona, Spain*, October 1999.

[Kop99b] H. Kopetz. *Specification of the TTP/C Protocol.* TTTech, Schönbrunner Straße 7, A-1040 Vienna, July 1999. Available at http://www.ttpforum.org.

[Kop00a] H. Kopetz. Software Engineering for Real-Time: A Roadmap. *IEEE Software Engineering Conference, Limmerick, Ireland*, 2000.

[Kop00b] H. Kopetz et al. Specification of the TTP/A Protocol. Technical report, Technische Universität Wien, Institut für Technische Informatik, March 2000. Available at http://www.ttpforum.org.

[Kop01a] H. Kopetz, M. Holzmann, and W. Elmenreich. A Universal Smart Transducer Interface: TTP/A. *International Journal of Computer System Science & Engineering*, 16(2), March 2001.

[Kop01b] H. Kopetz, M. Paulitsch, C. Jones, M.-O. Killijian, E. Marsden, N. Moffat, D. Powell, B. Randell, A. Romanovsky, and R. Stroud. Revised Version of DSoS Conceptual Model. Project Deliverable for DSoS (Dependable Systems of Systems), Research Report 35/2001, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, 2001.

[Krü97] A. Krüger. *Interface Design for Time-Triggered Real-Time System Architectures.* PhD Thesis, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, April 1997.

[Nou99] P. Noury. WorldFIP, IEC 61158 and the Internet: A New Look at Fieldbuses, 1999. Available at http://www.worldfip.org/noury02.html.

[Obe01] R. Obermaisser. Design and Implementation of a Distributed Smart Transducer Network. Master's Thesis, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, 2001.

[Obj01] Objective Interface Systems, TTTech Computertechnik, and VERTEL Corporation. Smart Transducers Interface. *OMG TC Document orbos/2001-06-03*, July 2001. Supported by Technische Universität Wien. Available at http://www.omg.org.

[OMG00] Object Management Group OMG. Smart Transducers Interface Request for Proposal. *OMG TC Document orbos/2000-12-13*, Dec. 2000. Available at http://www.omg.org.

[Pau01] Michael Paulitsch. Accuracy in Multi-Cluster Systems. Research Report 21/2001, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, 2001.

[Pet00] P. Peti and L. Schneider. Implementation of the TTP/A Slave Protocol on the Atmel ATmega103 MCU. Technical Report 28, Technische Universität Wien, Institut für Technische Informatik, August 2000.

[Pet01] P. Peti. Monitoring and Configuration of a TTP/A Cluster in an Autonomous Mobile Robot. Master's Thesis, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, 2001.

[Pin95] J.J. Pinto. A Neutral Instrumentation Vendor's Perspective. *ISA Proceedings '94 and Intech July '95*, July 1995.

[Sch01] L. Schneider. Real Time Robot Navigation with a Smart Transducer Network. Master's Thesis, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, 2001.