TTP/A: A Low-Cost, Highly Efficient, Time-Triggered Fieldbus Architecture

Ralf Schlatterbeck
TTTech Computertechnik AG

Wilfried Elmenreich
Technical University Vienna

ABSTRACT

TTP/A is the fieldbus protocol of the Time Triggered Architecture (TTA). It provides periodic transmission of real-time data and allows for on-line configuration, diagnostics, and maintenance by use of an interface file system. It is well integrated with the TTP/C protocol and is designed to meet the requirements of a low-cost sensor/actuator bus. TTP/A is a master-slave protocol where the master establishes a common time base within a TTP/A cluster. Since the master establishes the time base prior to the communication of slaves, the protocol can be implemented with low-cost on-chip RC oscillators for the slaves. Using a standard UART-based serial interface as physical layer, the slave TTP/A protocol can be implemented in very low cost Commercial Off-The-Shelf (COTS) hardware.

INTRODUCTION

The digital revolution is coming to the automobile. About 90 % of innovations in this area consist of or include electronic devices. The next step to an integrated electronics architecture in the car will create new challenges for the interconnection of components manufactured by different suppliers. The time triggered architecture (TTA) provides a systematic approach for an integrated electronics architecture. TTA is a family of time triggered communication protocols. TTP/A is a SAE class A protocol intended for non safety critical low-speed communications. Due to its easy implementation and low hardware requirements it can be implemented on a wide range of off-the-shelf microcontrollers. This paper is structured as follows. The section (Protocol Objectives) will give a short introduction to features supported by TTP/A. The following section (Protocol Overview) will give an overview of the protocol. The next section (Related Work) gives a comparison to another class A protocol and finally (Conclusion) summarizes the paper.

PROTOCOL OBJECTIVES

TTP/A is a bus interface for small universal transducers [2,3]. A smart transducer is an integration of one or more sensor/actuator elements with a micro-controller and a real-time network interface thus forming a mechatronic component. The smart transducer can support signal conditioning, calibration, and conversion to standard measurement units. Since all sensor/actuator specific details are hidden behind a standard network interface supporting plug and play capabilities, the complexity of sensor and actuator interfaces is reduced.

- **Three Interfaces of a Smart Transducer**

A smart transducer should provide three different interfaces to support different views of the transducer. The Real-Time Service interface (RS) supports the real-time view. It is time sensitive and used for periodic control information, e.g., sensor reading, actuator setting.

The Diagnostic and Maintenance interface (DM) is not time sensitive. It is used for diagnosis and calibration purposes and requires knowledge about the internals of a transducer. It is either used sporadically or can be scheduled periodically to monitor certain values during operation. For periodic operation a certain amount of bandwidth during operation is reserved for the DM interface.

The Configuration and Planning interface (CP) is used to install a node into a new configuration. It supports automatic identification of a new node on the bus and the configuration of this new node. TTP/A supports all of these interfaces. Both, the CP and the DM interface may even be used without disturbing the ongoing real-time traffic.

- **Standard UART, High Data Efficiency**

TTP/A builds upon a standard UART interface. This allows the use of COTS components without having to implement dedicated silicon. Furthermore by using a standard interface, the protocol is compatible with a large number of standard microcontrollers. This makes it possible to reach even the most demanding cost goals. Due to its simplicity the protocol could even be implemented by a statemachine.

To reduce wiring costs, TTP/A is designed to operate on a single wire physical layer according to the ISO 9141 standard. On this physical layer a transmission speed of up to 33.6kbit/s is supported. A theoretical limit of the ISO 9141 bus is 50kbit/s but a certain safety margin should be taken into account. For other applications, TTP/A supports other physical layers that permit higher transmission rates. Without dedicated support 1 Mbit/s will be an approximate limit.

To support a good response time while being able to use low bitrates on the transmission medium, TTP/A has been designed for high data efficiency, especially for short messages. Using the time triggered paradigm, a-priori knowledge of the participants, the time and order of transmissions leads to lower protocol overhead in the communication.

- **The Two Level Design Approach**

The TTP protocol family and related tools support a two-level design approach. On the cluster level, a system integrator specifies the cluster schedule. The various nodes may then be implemented by subcontractors.

In order to support a two-level design approach a protocol must support composability. A protocol is composable with respect to a specified property if the system integration does not invalidate this property when it was established at the subsystem level. TTP/A is composable regarding the properties timeliness and testability. This means system integrators are not faced with the challenging task of finding out why the system does not work, although all subsystems work according to their specifications.

Composability is achieved for TTP/A because subsystems do not interfere with the temporal properties of other subsystems. By using the composability property, proper operation of the whole system can be inferred from the proper operation of all subsystems.

- **Small Error Detection Latency**

TTP/A supports the design of fail safe body electronics. To support fail safe mechanisms on a single wire bus, each device connected to the bus must have a safe state. An example of a safe state for a car window is that it immediately stops moving once it detects an error condition. By providing a small latency for error detection, each device knows within a short period of time when to go to the safe state.

- **Clock Synchronisation, Low Jitter**

Clock synchronisation is achieved by central master synchronisation where the master has a precise oscillator. Slaves are synchronized to the master and may have a frequency deviation within 50 % of the nominal frequency and a drift rate of up to 10 %. Because a slave has no knowledge of the deviation of its local clock, it has to be synchronised by the master initially. Further synchronisation during a round is achieved by knowledge at the slaves about which bytes are transmitted by the master with its precise clock.

For implementing control loops via TTP/A it has to support a low variability of the point in time when reading sensors and writing actuator values. This variability in I/O activities is called jitter. TTP/A supports low jitter in the range of 0.1 % to 1 % of the duration of a transmission [1] depending on how often resynchronisation of slaves is performed (i.e., when and how often slots are transmitted by the master).

PROTOCOL OVERVIEW

TTP/A is a master-slave protocol. There are up to 255 slaves on a bus and a single active master. The master provides the time base for the slave nodes. The communication is organized into rounds. A round consists of several slots. A slot is a unit of transmission for one byte of data.

A communication round is started by the master with a so-called fireworks byte [3]. The fireworks byte defines the type of round. Each round consists of a configuration dependent number of slots and an assigned sender node for each slot. See Figure [1].

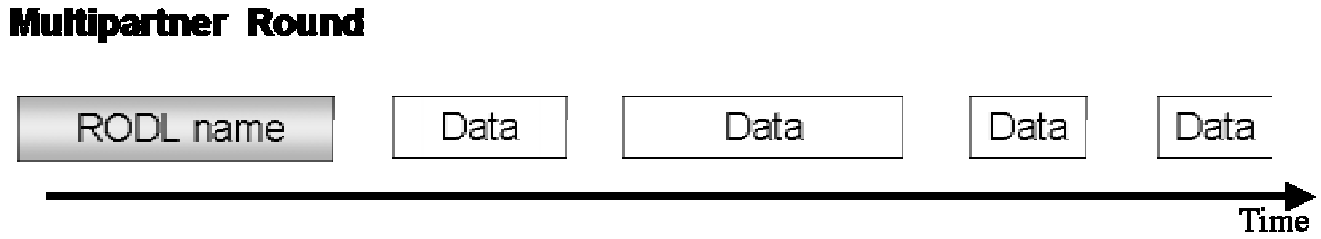

Figure 1: Multipartner Round in TTP/A

- **Interface File System**

All relevant data of a TTP/A node like round definitions, application specific parameters and I/O properties are organized into a structure called interface file system (IFS). The IFS is structured as a record-oriented format where every record is separately addressable. Optionally there is a horizontal and vertical checksum for a file supporting error detection and error correction.

There are four types of files in the IFS. All hardware facilities are addressed by I/O files in the interface file system of a transducer. Command files serve to start actions on the node, documentation files are read-only and contain the documentation of the node. Finally RODL files (ROund Descriptor List) contain the round definitions for the RS communication. RODL file entries define actions on files in the IFS for certain slots of a round. Each round has its own RODL file on each node. Files can be read, written, or executed via actions in RODL files.

- **Two Types of Rounds**

There are two types of rounds, multipartner rounds and master/slave rounds (see Figure 2). The multipartner round provides the RS interface while the master/slave round provides the DM and CP services.

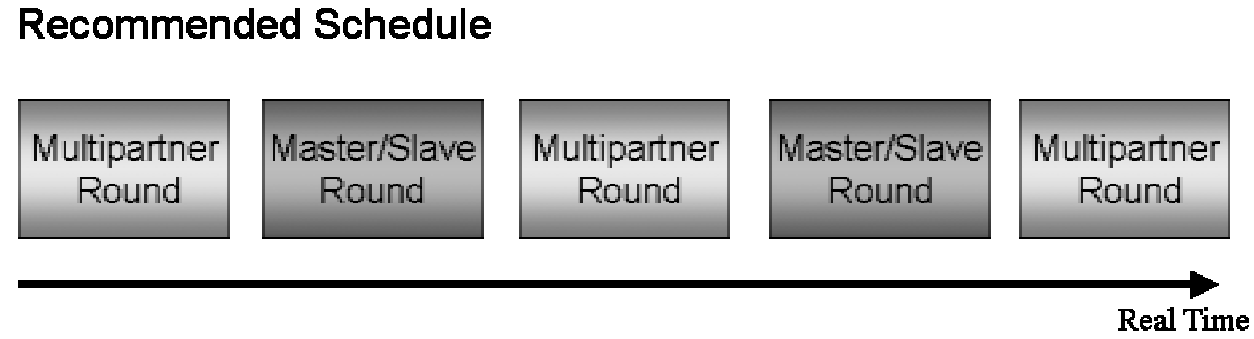

Figure 2: Recommended TTP/A Schedule

A master/slave round has a fixed layout and is used for detecting new slave nodes, configuration of slave nodes, sleep mode request and manipulation of RODL files. The action and the address in the IFS of a master/slave round is encoded in three parameter bytes sent after the fireworks byte.

Multipartner rounds execute a RODL file in parallel on each node participating in the communication. For every slot during a multipartner round each node knows its action to perform through the currently executed RODL. The RODL defines which node transmits a certain slot, the semantics of each individual slot, and the receiving nodes of a slot.

To support both, the RT interface and the DM and CP interfaces in parallel, it is recommended, that multipartner rounds and master/slave rounds are interleaved (see Figure 2).

RELATED WORK

There is another class-A bus for low-cost automotive networks, the Local Interconnect Network (LIN) [4]. It uses the same physical layer as TTP/A and is also a master-slave protocol. It has about the same requirements concerning slave node implementations as TTP/A, in particular it also places low requirements on the oscillators of slave nodes.

LIN has a communication scheme where each of up to 64 different messages are requested by the master with a one-byte identifier consisting of a six-bit address field and a two-bit checksum. Each message is uniquely assigned to a single slave or the master in a particular bus configuration. A message has a fixed length encoded in the identifier of the message with two, four, or eight data bytes and an additional checksum.

Although LIN addresses the smart sensor market like TTP/A, it focuses on easy implementation for real-time sensor communication. This results in some differences in protocol performance summarized in Table 1. An in-depth comparison of the protocols is given in an earlier paper [1]. Differences are in the areas of protocol overhead, data protection and protocol features. Due to a higher protocol overhead of LIN compared to TTP/A, TTP/A provides better response times and higher data efficiency. Due to high tolerance of slave timing in LIN the message jitter is very high compared to TTP/A. The high tolerance of slave timing eases the protocol implementation of LIN slaves but introduces difficult timing behavior at system level.

The data protection in LIN and TTP/A is different. While LIN does not have a parity bit for each transmitted byte, TTP/A does. The initial command in LIN is protected by a two-bit checksum, while the TTP/A fireworks byte is protected by a five-bit checksum. TTP/A offers additional protection mechanisms depending on the protection requirements of certain messages while LIN does not have such mechanisms.

LIN omits mechanisms for online configuration. This is possible because all message identifiers are assigned before the nodes are connected to the LIN network. However, this facilitation of implementation does not come without a drawback: a LIN designer has to face the problem that only 64 message identifiers are available. Being able to configure slaves for different environments is a required property if smart sensors should be used in different contexts, e.g., in different cars with different bus topologies. In order to provide inter-operability of smart transducer manufacturers and applications the identifiers would have to be standardized. Thus a LIN designer has to carefully keep track of the 64 available messages and the assignment of messages to sensors. Assignment of the same message identifier for different scenarios results in a high complexity of managing sensor reuse. Furthermore, if some identical sensors are connected to one bus, e.g. the same type of temperature sensor is used twice, it is necessary to configure the message identifiers of the nodes before they are connected to the network. This extra task increases cost and requires a persistent configuration memory for the slaves. TTP/A can solve this problem with a persistent configuration memory in the master which re-configures slaves via the configuration interface in case a slave loses its configuration (e.g., due to a power failure).

Protocol property	LIN	TTP/A
Protection of Command	2bit	5bit
Jitter (in % of round)	40 %	1 %
Support for on-line Diagnosis and Maintenance	no	yes
Implementation Complexity	very low	low
Design for Reuse (Plug-and-Play)	difficult	easy

Table 1: comparison of protocol features of LIN and TTP/A

CONCLUSION

TTP/A is a protocol that is suitable for interconnection of smart transducers, intelligent sensors and actuators. It is composable in the time domain and with regard to testability and thus can support a two level design framework. It provides short error detection latency because of a-priori knowledge of the communication pattern which facilitates the detection of lost messages or a missing node. Due to its use of standard UARTs with minimal requirements regarding the quality of oscillator and hardware resources it can be implemented on a wide range of microcontrollers. There is no special need for protocol-specific hardware for supporting clock synchronization or other protocol functions. In comparison to LIN, TTP/A offers a more efficient communication, a more predictable timing and flexible online configuration for the cost of an increased effort for slave implementations.

ACKNOWLEDGMENTS

We wish to thank Jürgen Bauer from Audi for helpful discussions regarding LIN. Furthermore we thank the anonymous reviewers for helpful comments on earlier versions of this paper.

REFERENCES

[1] H. Kopetz, W. Elmenreich, and C. Mack. A comparison of LIN and TTP/A. In Proceedings of the 3rd IEEE International Workshop on Factory Communication Systems, Porto, Portugal, 2000. http://www.vmars.tuwien.ac.at/papers/extern/Paper2000_4.html

[2] H. Kopetz, M. Holzmann, and W. Elmenreich. A universal smart transducer interface: TTP/A. In 3rd IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC 2000), 2000.

[3] Hermann Kopetz. TTP/A The fireworks protocol. In SAE International Congress and Exposition, Detroit, Michigan, February 1995.

[4] Hans-Christian von der Wense (Ed.). LIN specification package. Revision 1.1, LIN Consortium, April 2000.

CONTACT

Ralf Schlatterbeck
TTTech Computertechnik AG
schlatterbeck@tttech.com

Wilfried Elmenreich
Technical University Vienna
wil@vmars.tuwien.ac.at