A GENERIC TOOL FOR SYSTEMATIC TESTS IN EMBEDDED AUTOMOTIVE COMMUNICATION SYSTEMS

Roman Pallierer
Dependable Computer Systems
DECOMSYS GmbH
Stumpergasse 48/28, A-1060 Vienna
pallierer@decomsys.com

Martin Horauer, Martin Zauner
University of Applied Sciences
Technikum Wien
Höchstädtplatz 5, 1200 Vienna, Austria
{horauer, zauner}@technikum-wien.at

Andreas Steininger, Eric Armengaud, Florian Rothensteiner
Vienna University of Technology
Embedded Computing Systems E182-2
Treitlstr. 3/2, 1040 Vienna, Austria
{steininger, armengaud, rothensteiner}@ecs.tuwien.ac.at

Abstract. ***S***ystematic ***T***est of ***E***mbedded ***A***utomotive ***C***ommunication ***S***ystems is the focus of the so-called STEACS project[1] - a partnership between academia and industry. The main emphasis in respect of automotive communication systems is laid on the communication protocol FlexRay. The goal is to achieve a systematic test approach and to evaulate the feasibility by corresponding prototype implementations.

This paper presents one of the results of the project, a generic tool basis supporting monitoring and diagnosis methods for FlexRay based systems, with interfaces to other protocols, such as CAN, and digitial and analog input signals.

1 Introduction and Motivation

While well-proven and highly optimized mechanical systems in today's cars hardly leave any room for further improvements, the introduction of electronic control systems has revolutionized the automotive industry. In fact, electronics has established itself as the most important innovation driver in the automotive domain. Electronic control units (ECUs) not only constitute a cheap and lightweight replacement for their mechanical counterparts but even facilitate new functionality. One of these new options is to connect individual control systems by means of communication networks, thus they facilitate sharing resources and – most importantly – exchanging information. The access to global information provides the local control systems with a more comprehensive view of the environment and allows them to coordinate their activities. This finally forms the key to unprecedented "intelligent" functions, e.g. to use brake-assistance for steering. The introduction of "by-wire" systems – which essentially means replacing mechanical and hydraulic components by electronic components connected by wires without the provision of a mechanical fallback – is therefore viewed as a substantial innovation step for the automotive industry [LH02].

At the same time these new approaches create a lot of new challenges. Considering that modern cars already contain up to 70 ECUs and thus represent highly complex distributed architectures, one of these challenges is certainly to ensure the dependability required for the safety critical applications under the given tight cost constraints. Car manufacturers have soon identified the network architecture as the crucial point in this context. To that end, an industrial consortium is establishing the communication protocol FlexRay [F04] that provides the flexibility to operate the communication system in a time- and/or event-triggered mode, respectively.

[1] The STEACS-project (http://embsys.technikum-wien.at/steacs.html) received support from the Austrian "FIT-IT[embedded systems" initiative, funded by BMVIT and is managed by the FFG (FFF) under grant 807146.

Despite all the wealth of properties to provide a reliable communication testing of such complex distributed architectures remains an issue. Even though we can rely on proven traditional test approaches for ensuring the functional integrity of all involved components (in isolation), their proper interoperation remains to be verified – with a large number of product variants and configurations. In particular, the test efforts are known to rise more than linearly with system complexity. For electronic systems in automobiles, however, system complexity increases with every new generation, hence, a solution to this problem is vital for the future.

The project STEACS (Systematic Test of Embedded Automotive Communication Systems) was established under the leadership of the industrial company DECOMSYS – a development member of the FlexRay consortium – and two academic partners, namely the University of Applied Sciences Technikum Wien and the Vienna University of Technology. The common mission of this project is to address the above challenges and develop a systematic approach that enables a test of the fully assembled complex system while still facilitating some kind of decomposition, such that complexity remains manageable. Clearly this approach shall be cost efficient and applicable during various phases within the lifecycle of an electronic system.

2 Requirements and Objectives

It seems evident that the test requirements are strongly dependent on the considered phase within the product lifecycle. During prototype development, for instance, the central aim is to prove the absence of design errors, assess the robustness of a given configuration or to diagnose observed phenomena. Access to any desired location is usually provided and the test engineer has an in-depth knowledge of the system details. For a maintenance test in a garage, on the other hand, accessibility is limited and a simple go/no-go statement is desired since the test is performed by a non-expert with the aim to find defects (and assuming a correct design). In order to be applicable in all phases of the life cycle our test approach must be generic. In particular it should come along with limited access and provide detailed diagnostic information that may be condensed to a go/no-go statement if desired.

Four means are described in [AL+04] to attain dependability and security within dependable systems. Fault prevention and fault tolerance are usually deployed during system development respectively to prevent implementation faults and to improve future system's robustness. Fault removal aims at finding and correcting faults within the system and can be applied during product development (removal of implementation faults) or field operation (maintenance). Finally, the goal of fault forecasting is to evaluate the system's robustness with respect to its future field operation.

Our current focus in the development of our concept is fault removal during the design phase, since this is where tool support is most urgently needed at the moment. Together with our automotive partners we have identified four main scenarios in which a diagnosis tool would be required:

1. **Node level testing:** A single node is defective before system operation actually starts. This case is not within the focus of our project, since efficient tests methods such as BIST are already available for chips or printed circuit boards.
2. **System start-up:** A set of successfully tested single nodes are integrated into a prototype cluster which does not start then. Here a tool is required to provide diagnostic information. Alternatively, if the prototype cluster does start up, robustness tests are desired to guarantee that the given configuration will also start up properly in mass production.
3. **System operation:** A system is successfully put into operation but then suddenly fails. Again diagnostic information is required here. If no failure is encountered during prototype operation, robustness tests shall support the projection of this result to the mass production.

4. **System fault tolerance:** A tool is required to systematically subject the system to all anticipated fault scenarios and thus test the fault-tolerance services.

To illustrate these points let us take a closer look at scenario 3: Here a typical example would be that an error detection mechanism on one node triggers, and as a consequence the communication controller switches the node to silent. This means it prevents the node from sending further messages – which are, however, expected by the other nodes. Another usual problem to cope with is the physical line break of one channel. Some messages might still be successfully transmitted due to channel redundancy but the overall system dependability is decreased. A third use case for our tool would be to acquire a view of the global system state by monitoring the signals actually exchanged. This is an important aid for application development.

Considering the above mentioned role of communication as the most important enabler for distributed functionality, and in order to answer the current (and possible future) needs of our partners, the focus of our project is set on diagnosis and testing methods for the communication subsystem. It should be noticed, however, that the accurate information obtained at the data link layer can easily be processed to increase observability at higher layers (e.g. at the FTCOM [OV01] or application layer). The aim of the project can thus be detailed as (i) elaborate a strategy for automated diagnosis in the communication subsystem, (ii) develop methods for validation and verification of protocol parameters, (iii) elaborate a fault injection methodology for testing of robustness and fault-tolerance mechanisms upon system integration and maintenance, (iv) define a basis for protocol conformance testing and finally as a matter of course (v) develop a demonstrator and experimentally evaluate the concepts.

3 The STEACS approach

The quality of a test result (in terms of test speed and coverage) attainable with given efforts strongly depends on the testability of the system under test. According to [BMS87] two key aspects for testability are *controllability* and *observability*. Controllability is an indicator of how easy it is to bring the system or node under test to a given state. Observability is a measure of how easy it is to observe certain activities of the system or node under test. For the given architecture this means that we must be able to bring the system under test to all relevant states and observe all relevant details by means of our tester node.

Testing and diagnosis of distributed systems are confronted to three main problems, which are (i) geographic diversity: Each node is a discrete physical entity distant to the other one. This results in the difficulty to observe and control the whole system (i.e. every node). The second problem is (ii) the large system state space: Each module can usually be modeled by a state machine, and consequently the system state is the product of each parallel state machine. As a result, testing all states of the system is an elaborate operation; moreover, synchronization between the nodes (state machines) is an additional problem compared to single systems with only one global state machine. Finally, (iii) the highly layered architecture leads to a very large fault range (from low level such as EMC to high level such as malicious faults) that the usual fault injection methods can not handle.

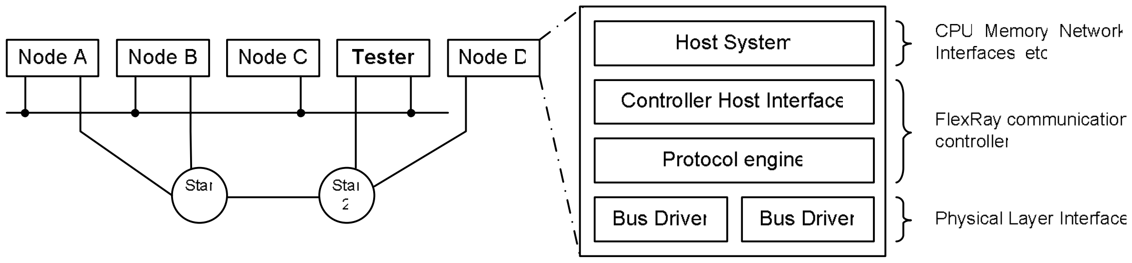

Figure 1: System Setup Example

The STEACS approach is to connect a dedicated tester – the BUSDOCTOR – to the communication medium of the distributed system, see Fig. 1. Since this is usually a single dedicated node, one can afford a quite sophisticated design. This setup presents three main advantages: (i) the intrusiveness of the system is kept low because no node is modified, (ii) the centralized and single tester can reach every node, which solves the geographic diversity problem, and (iii) this approach provides a very large coverage. Indeed, the tester node is directly connected to the communication medium, and both correct bus traffic and possible disturbances (e.g. glitches) or errors can be monitored with a very high accuracy.

In this context, a main contribution of the STEACS approach is the decomposition of the complex system by means of a layer structure. While this decomposition facilitates a comparable complexity reduction as the usual (geographical) divide and conquer approach, the implied vertical structuring suits more naturally to the nature of the distributed system (with single services being distributed over multiple ECUs, e.g.). The fine grained layer model we have developed for this purpose (see [AS+04, PH+04]) enables us to project information collected on a low layer to any desired higher layer. Therefore, this generic approach enables the diagnosis (and testing) of both low level mechanisms such as e.g. glitches on the bus and high level mechanisms such as e.g. correct signal exchange within the application.

4 The STEACS implementation

The implementation of the test approach encompasses two main hardware components: a tester ECU, called BUSDOCTOR, and a remote PC. The BUSDOCTOR performs the actual monitoring and replay functionality, see [HR+04], whereas the remote PC provides the graphical representation, cf. Fig. 2. In order to speed-up the development, the BUSDOCTOR hardware is based on the COTS component termed Node<ARM> from DECOMSYS.

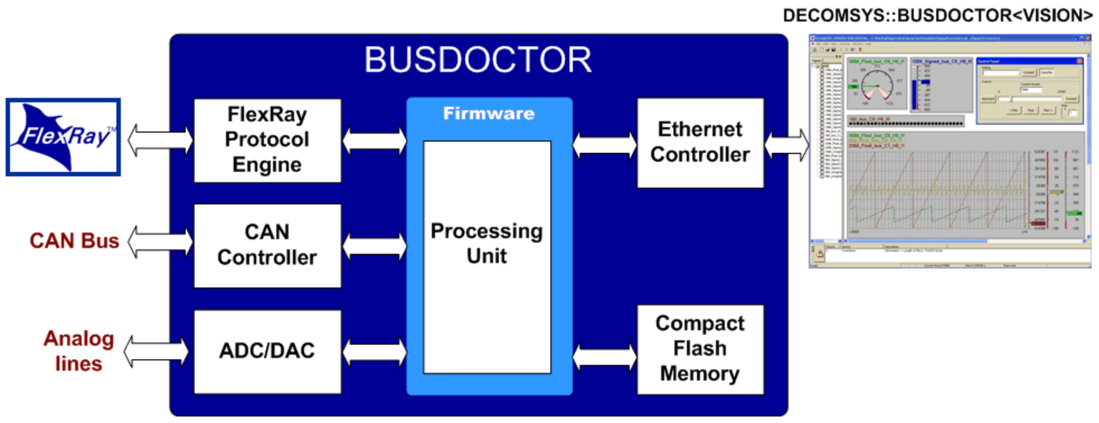

Figure 2: The BUSDOCTOR System Architecture

The BUSDOCTOR hosts the following implemented modules and units:

Hardware (Altera Excalibur FPGA)

- Two large FIFOs are used as replacement of the controller-to-interface of a COTS FlexRay controller, cf. Fig. 1, to transfer the monitoring/replay data between the FlexRay/CAN controller and the embedded ARM processor.
- A timebase unit with a resolution of 25 ns provides the basis for timestamping of events.
- A cluster synchronization unit serves to synchronize the global time of the FlexRay system towards the local timebase used for timestamping.
- Several monitoring and replay modules at different layers of abstraction enable recording and insertion of data from/to the network traffic on either bus system.

Firmware (RTAI Linux)

- Control and configuration modules for the monitoring/replay hardware and the bus controllers.
- A high-speed compact flash interface to transfer data to/from large flash disks.
- A Fast-Ethernet interface to transfer data to/from the remote PC via the TCP or UDP transport protocols.
- Trigger and filter modules to record/inject only specific information.

Remote Software

- A BUSDOCTOR DLL handles data transfers between the embedded BUSDOCTOR units on one hand and the visualization software and the PC filesystem on the other hand.
- The visualization software provides various different ways to present the relevant data to the user. This visualization software integrates seamless to the software toolchain of DECOMSYS for configuration and control of automotive systems.

4.1 Logging Functionality

The BUSDOCTOR is capable of simultaneous logging both FlexRay channels, the CAN bus and the analog/digital lines. By providing each sample with a timestamp (i.e. a common reference point in time) it is possible to see all measurements in the same timeline, hence investigating the total system response.

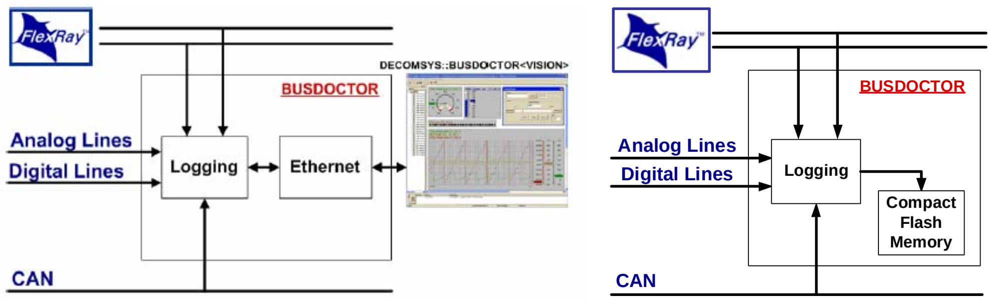

Figure 3: Online and Offline Logging

When logging communication or analog/digital lines the BUSDOCTOR can be configured in two different operation modes, either in online or in offline mode. Online mode aims primarily at lab tests whereas the offline mode is applicable for field tests as well.

When there is a lot of communication activity it often becomes necessary, depending of course on the mode of operation, to limit the amount of data being collected, or to accept that some logged data might get lost because of bandwidth limitations or limited memory size. Limiting the amount of recorded data is done in the BUSDOCTOR by using a Filter, and when to actual start/stop recording is done using a Trigger.

4.2 Filter

The Filter functionality allows defining specific rules to determine which data is being logged. It is possible to specify different filters for the data going to the PC and the data going to the CompactFlash Memory.

ID	Repetition	Base Cycle	Channel
1-2047	1	0	-
1	1	0	A
2	2	0	B
3	4	1	AB
10-20	1	0	AB

Table 1: Example of an array with Filter rules

Filter rules consist of an array containing slot ID, communication cycle and channel. The slot ID identifies the corresponding FlexRay TDMA slot, where data is transmitted. FlexRay supports up to 64 communication cycles. Within the filter rule, a communication cycle is specified by two values, base and repetition. Repetition defines the number of cycles until the next cycle to investigate; only possible values are in step of 2 (1, 2, 4, 8, …, 64). Base defines which cycle to start with; possible values are from 0 to Repetition -1.

4.3 Trigger

The trigger functionality allows specifying rules to control the start and stop of the data logging. The trigger rules may consist of several logical equations including, e.g., the following sources

- FlexRay header information (ID, CycleCounter, SyncBit, SUP Bit, Length)
- FlexRay Statusinformation (Errorflags)
- Data content

The Trigger is applied to one FlexRay frame at a time, and if a trigger equation results to true the logging will either be activated or deactivated.

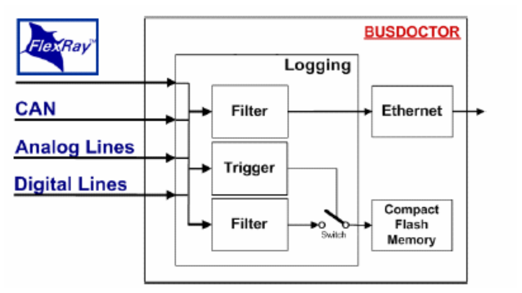

Figure 4: Combined Trigger and Filter Functionality

4.4 Replay Functionality

The replay functionality allows sending data back earlier recorded FlexRay logs onto the communication bus.

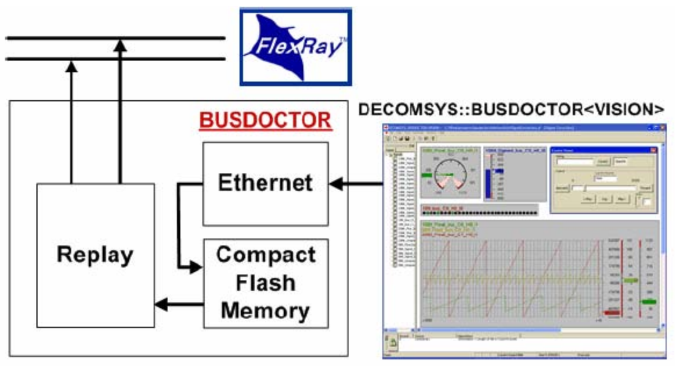

Figure 5: Replay Functionality

There are two replay modes supported:

- In asynchronous mode the BUSDOCTOR emulates the majority of nodes participating in the clock synchronization. Therefore a functional and timely replay is possible since all other nodes synchronize to the BUSDOCTOR.
- In synchronous mode the BUSDOCTOR synchronizes itself to the schedule present on the network. Hence, replay can only be performed in the correct functional way, i.e.

recorded FlexRay data can be sent back onto the bus with the right ID and in the right cycle only.

5 Applications of the tool

With simultaneous logging of FlexRay and CAN communication together with analog/digital input signals, connected for example to external sensors, the BUSDOCTOR can on occurrence of external events record a total communication system response. The recorded communication response can succinctly be analyzed using the software visualization tool, where specific problem areas can be identified. Within the test scenarios described in Chapter 2 typical application areas are therefore:

System startup - Debugging of FlexRay systems: For example in an prototype environment the BUSDOCTOR could give valuable insights into an actual bus communication, hence, verify the design and make it easier to identify problems.

System Operation - Analyzing of mixed bussystems: In a system where multiple bus architectures are used (e.g. FlexRay and CAN being interconnected over gateways) the BUSDOCTOR provides with its simultaneous logging functionality valuable communication timing information.

System fault tolerance – Integration Testing: While integrating independent modules into a larger system, e.g. in a car where many independent modules communicate over a common bus, the BUSDOCTOR can record all important bus communication for later analysis during system integration test drives.

To support these applications the BUSDOCTOR can operate in several different modes of operation:

- In the **Full Operation Mode** the BUSDOCTOR is on one side connected to the bus systems and the analog/digital sensors and via the Ethernet interface to the remote PC, see Fig. 3 (left). Recorded bus traffic is directly fed to the visualization software and vice versa for replay or fault injection. This kind of operation is often used in a prototype development environment, where different kinds of system designs are being designed and tested. In particular, the application-engineer wants to test the basic functionality of a system; here the BUSDOCTOR is a valuable monitoring tool for verifying the design. Furthermore, when many third party modules are being integrated into a larger system interoperation issues can be a major obstacle; here the BUSDOCTOR is a powerful tool for identifying which modules do not behave as specified.
- In **Standalone Mode** the BUSDOCTOR operates to/from the flash disk and must be preconfigured prior to its operation. This kind of operation is especially useful for field tests following the prototype phase. To monitor everything and provide documentation for system wide correct behavior a BUSDOCTOR is taken along during test drives, cf. Fig. 3 (right). When everything is working fine the BUSDOCTOR has nothing to record, however, sometimes under real world conditions systems may temporarily fail, e.g. due to EMC disturbances. Following the test drive the BUSDOCTOR can be coupled with a remote PC and the recordings can be read-out and analyzed by a test-engineer.

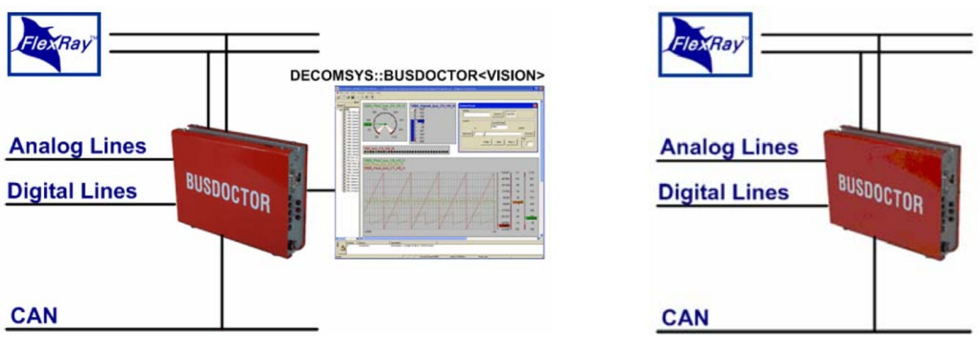

Figure 6: Full Operation (left) and Standalone (right) Modes

6 Conclusion and Outlook

So far the STEACS project has made two main contributions. First, the establishment of the scientific basis for systematic testing by introducing a layer-based test methodology targeted for time-driven protocols such as FlexRay. The framework has been published at international academic and industrial conferences. Second, prototype implementations of the generic tool have been performed laying the basis for professional monitoring and diagnosis products such as the DECOMSYS::BUSDOCTOR.

A wide range of testing activities is currently ongoing which use the presented generic tool. At automotive customers, the tool is already used for monitoring FlexRay and CAN messages simultaneously. Within the FlexRay consortium, the presented tool is used to perform extended fault injection tests of the FlexRay communication protocol. Further planned activities include the support of conformance and interoperability tests of the FlexRay protocol, as well as the support of fault injection and diagnosis projects at automotive customers.

References

[AL+04] A. Avizienis, J.-C. Laprie, B. Randell, C. Landwehr, *Basic concepts and taxonomy of dependable and secure computing*. IEEE Transactions on Dependable and Secure Computing, Vol. 1, Issue 1, pp. 11 – 33, Jan.-March 2004.

[AS+04] E. Armengaud, A. Steininger, M. Horauer, R. Pallierer, *A Layer Model for the Systematic Test of Time-Triggered Automotive Communication Systems*. 5th IEEE International Workshop on Factory Communication Systems, pp. 275 – 283, 2004.

[BMS87] P. Bardell, W. McAnney and J. Savir, *Built-in Test for VLSI, Pseudorandom techniques*. John Wiley & Sons, New York 1987.

[F04] -, *FlexRay Communications Systems - Protcol Specification Version 2.0*. FlexRay Consortium, 2004. http://www.flexray.com.

[HR+04] M. Horauer, F. Rothensteiner, M. Zauner, E. Armengaud, A. Steininger, H. Friedl, R. Pallierer, *An FPGA based SoC Design for Testing Embedded Automotive Communication Systems employing the FlexRay Protocol.* Proceedings of the Austrochip Conference, pp. 119 – 123, 2004.

[LH02] G. Leen and D. Hefferman, *In-Vehicle Networks, Expanding Automotive Electronic Systems*. IEEE Transaction on Computers, pages 88-93, January 2002.

[OV01] -, *OSEK/VDX Fault-Tolerant Communication Specification 1.0*. 2001. http://www.osek-vdx.org.

[PH+04] R. Pallierer, M. Horauer and A. Steininger, *Monitoring and Fault-Injection of X-by-Wire Communication Networks*. Design & Elektronik Tagungsunterlagen zum Entwicklerforum "Drahtlose und drahtgebundene Netzwerke 2004", Munich - Germany, July 2004.