It is Time for FlexRay

In parallel to the definition of the FlexRay protocol, supporting development solutions have been built. They rely on proven concepts for the design and test of in-vehicle systems and also address the demands of a time-triggered protocol. This contribution by dSPACE gives an introduction to these solutions and describes how they can be used to develop advanced vehicle control functions for the FlexRay protocol.

The FlexRay Consortium [1] has finalized the FlexRay Communications System Specifications Version 2.0 in summer 2004. Long before these specifications were made available to the general public, the FlexRay protocol had established itself as a de facto industry standard for a time-triggered, in-vehicle communication system. A lot of effort has been invested in defining and validating the protocol and in developing appropriate communication hardware. The stage has now been reached where more attention can be paid to developing the actual applications. With its large bandwidth, deterministic communication behavior, and fault tolerance mechanism, FlexRay is ideally suited to the role of a central backbone in future ECU architectures. Standardized communication services and easier reuse of existing ECU functions promise to offer cost savings in the long term.

The introduction of a time-triggered bus system such as FlexRay is doubtless a major technological advance and needs to be successfully addressed in all phases of the application development. This situation makes it all the more important to build on mature and proven products, while at the same time integrating support for FlexRay features at an early stage. Suitable development tools are needed to handle the complexity of networked applications. Being one of the first development partners in the FlexRay Consortium, dSPACE added FlexRay support to its product range very early on. As a partner to the automotive industry with years of experience in simulation technology for developing and testing ECU applications, dSPACE focuses on the features of FlexRay that are relevant to simulation under real-time conditions.

Automotive hardware platforms for FlexRay

For years dSPACE has been offering powerful and proven hardware platforms for function prototyping and ECU testing. These platforms have been extended and can be equipped with FlexRay communication controllers which are implemented as IP modules from DECOMSYS [3]. The use of IP modules has proven successful in handling the dynamics in the definition phase of the FlexRay protocol. Figure 1
shows two prototyping systems from dSPACE – the MicroAutoBox and the AutoBox [2]. Both of them are available with interfaces for LIN, CAN and FlexRay.

The platforms are a continuation of proven hardware concepts into the realm of time-triggered bus systems. Thus, it is not only possible to build pure FlexRay systems, but also to implement applications in which the time-triggered bus system has to be integrated into conventional in-vehicle control technology and in which typical automotive sensors and actuators have to be operated. This includes providing options for connecting lambda and engine speed probes, processing crankshaft-synchronous signals, and generating output signals for ignition and injection systems. Interfaces for incremental encoders and for controlling electric motors by means of PWM signals (particularly necessary in x-by-wire applications) are also needed.

Model based design of FlexRay systems

For design and test of control functions a model based development approach using MATLAB/Simulink® is well established. This approach can also be applied to applications using FlexRay as the communication system. However, FlexRay's underlying time-triggered approach requires additional planning steps as compared to the familiar procedure.

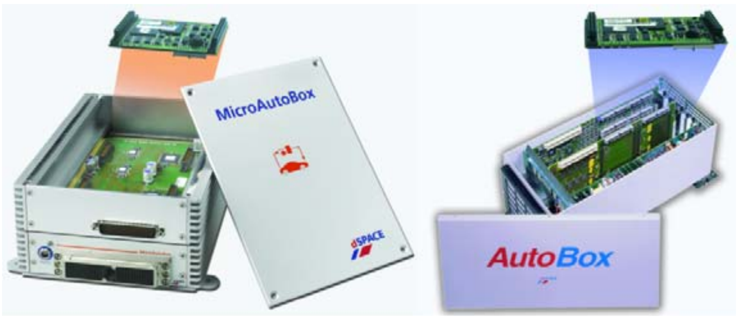
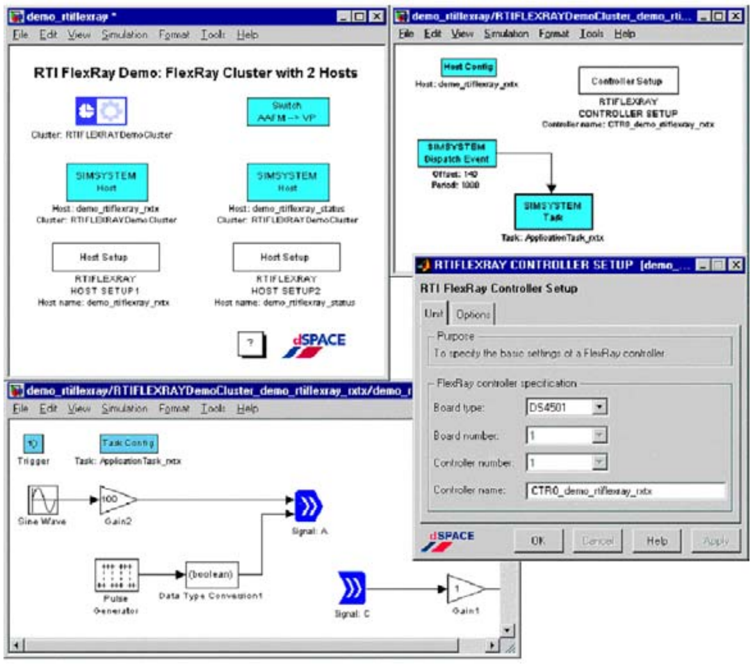


Figure 1: MicroAutoBox and AutoBox – automotive hardware platforms from dSPACE with interfaces for LIN, CAN and FlexRay.

The model based development process must be enriched by tools which generate a communication matrix in the time domain, i.e. which schedule message transmissions according to the communication relationships and execution patterns inherent in the FlexRay application. Such tools are available from DECOMSYS [3] and they have been integrated into a tool combination which first became available as products in spring 2003 and which has been updated regularly since then. The solution is called the dSPACE RTI FlexRay Blockset [2], and allows simulation hardware from dSPACE to be integrated as nodes in a FlexRay network.

The DECOMSYS tools in the tool combination are primarily intended for planning communication, generating code for the fault-tolerant communication layer (FTCom) according to OSEK/VDX [4], and configuring the FlexRay controller. The RTI FlexRay Blockset from dSPACE is used for generating the application code automatically and for linking all the components needed to execute a FlexRay application on a dSPACE system. Execution, and simulation itself, are performed with the aid of the dSPACE real-time kernel. Existing products such as ControlDesk can be used for executing and visualizing an experiment in the familiar way.

Figure 2 shows an example of how the integrated tool combination is used under MATLAB/Simulink. The overall system (FlexRay cluster) in this example consists of two nodes (hosts), on which two communicating applications (tasks) are located. The Simulink windows display the models for the overall system, for the parts of an individual node, and for a task to be implemented on that node. The hosts and tasks are modeled by means of blocks from the SIMSYSTEM blockset from DECOMSYS. The Host Setup block from the RTI FlexRay Blockset is used to specify that a host is realized as a dSPACE system during simulation. The dSPACE hardware, including the FlexRay controller that is used, is configured via the Controller Setup block. The actual application model is embedded in a task subsystem. It also contains the blocks that model communication via the FlexRay bus.


demo_rtiflexray *
File Edit View Simulation Format Tools Help
RTI FlexRay Demo: FlexRay Cluster with 2 Hosts
Cluster: RTIFLEXRAYDemoCluster
Switch AAFM --> VP
SIMSYSTEM Host
Host: demo_rtiflexray_rxtx
Cluster: RTIFLEXRAYDemoCluster
SIMSYSTEM Host
Host: demo_rtiflexray_status
Cluster: RTIFLEXRAYDemoCluster
Host Setup
RTIFLEXRAY HOST SETUP1
Host name: demo_rtiflexray_rxtx
Host Setup
RTIFLEXRAY HOST SETUP2
Host name: demo_rtiflexray_status
?
dSPACE
demo_rtiflexray/RTIFLEXRAYDemoCluster_demo_rti...
Host Config
Host: demo_rtiflexray_rxtx
Controller Setup
RTIFLEXRAY CONTROLLER SETUP
Controller name: CTR0_demo_rtiflexray_rxtx
SIMSYSTEM Dispatch Event
Offset: 140
Period: 1000
SIMSYSTEM Task
Task: ApplicationTask_rxtx
demo_rtiflexray/RTIFLEXRAYDemoCluster_demo_rtiflexray_rxtx/demo_r
Trigger
Task Config
Task: ApplicationTask_rxtx
Sine Wave
100
Gain2
Signal: A
Pulse Generator
(boolean)
Data Type Conversion1
Signal: C
1
Gain1
RTIFLEXRAY CONTROLLER SETUP [demo_...
RTI FlexRay Controller Setup
Unit
Options
OK
Cancel
Help
Apply

Purpose
To specify the basic settings of a FlexRay controller

FlexRay controller specification
Board type: DS4501
Board number: 1
Controller number: 1
Controller name: CTR0_demo_rtiflexray_rxtx

Figure 2: MATLAB/Simulink model of a sample FlexRay application.

In the Simulink model a FlexRay task is annotated with additional temporal attributes, indicating the execution cycle of the task and its trigger time as an offset to the cycle start. Thus, the sequence of task executions is statically defined at the design stage. During run time the tasks are activated according to these pre-defined schedules. In relation to the planned sequence of application tasks the communication matrix is designed in the time domain. Each message is assigned an exclusive transmission slot within the communication cycle. The result is a static, deterministic schedule for messages and tasks that is determined offline.

Real-time simulation of FlexRay systems

When real sensors, actuators or communication controllers are involved, a FlexRay application needs to be simulated in real-time, both for rapid control prototyping and for hardware-in-the-loop simulation. A simulation service on the hardware platform must activate the predefined sequences for task execution and message transmission. The services necessary for this are defined in a time-triggered extension of the OSEK/VDX specification called OSEKtime. An implementation of OSEKtime in conjunction with the fault-tolerant communication layer (FTCom) provides a reliable execution basis for a FlexRay system. The infrastructure must synchronize the local time of the application simulation to the global time on the FlexRay bus. The local time is the basis for guaranteeing the correct timing of task execution. Error situations can also be detected and responded to. Finally, idling behavior is possible, i.e., the tasks can be executed without being synchronized to a FlexRay system. This makes it possible to measure the execution times of the tasks, for example. Resynchronization to the bus directly from idling behavior is possible. The necessary components of OSEKtime have been added to the existing real-time kernel of dSPACE systems. This allows time-triggered and angle-based tasks to be executed in the same system, with time-triggered tasks having higher priority to guarantee deterministic behavior.

Conclusion and Outlook

The solutions presented here are currently demonstrating their capabilities in various projects by companies in the FlexRay Consortium, for example, in the field of chassis development. The applications range from advance development projects right through to projects at the transition to production. Thus, there is already a demand for FlexRay tools with long-term viability, for example, for designing hardware-in-the-loop simulators that will validate future FlexRay based production projects. The paper demonstrates that the design of FlexRay systems can now begin, as proven tools have been adapted to the new challenges and will continue to be developed in future.

References

[1] FlexRay Consortium. http://www.flexray.com.

[2] Real-Time Interface FlexRay Blockset. dSPACE GmbH, Paderborn. http://www.dspace.de/goto?RTIFlexray.

[3] DECOMSYS GmbH. http://www.decomsys.com.

[4] OSEK/VDX Consortium. http://www.osek-vdx.org

……………………………………………………………………

Dipl.-Inform. Joachim Stroop is Product Manager for System and Function Design Tools with responsibility for FlexRay products at dSPACE GmbH.

Dipl.-Ing. Ralf Stolpe is Group Leader for Multiprocessor and Distributed Systems at dSPACE GmbH. He is responsible for the development of FlexRay tools.

……………………………………………………………………