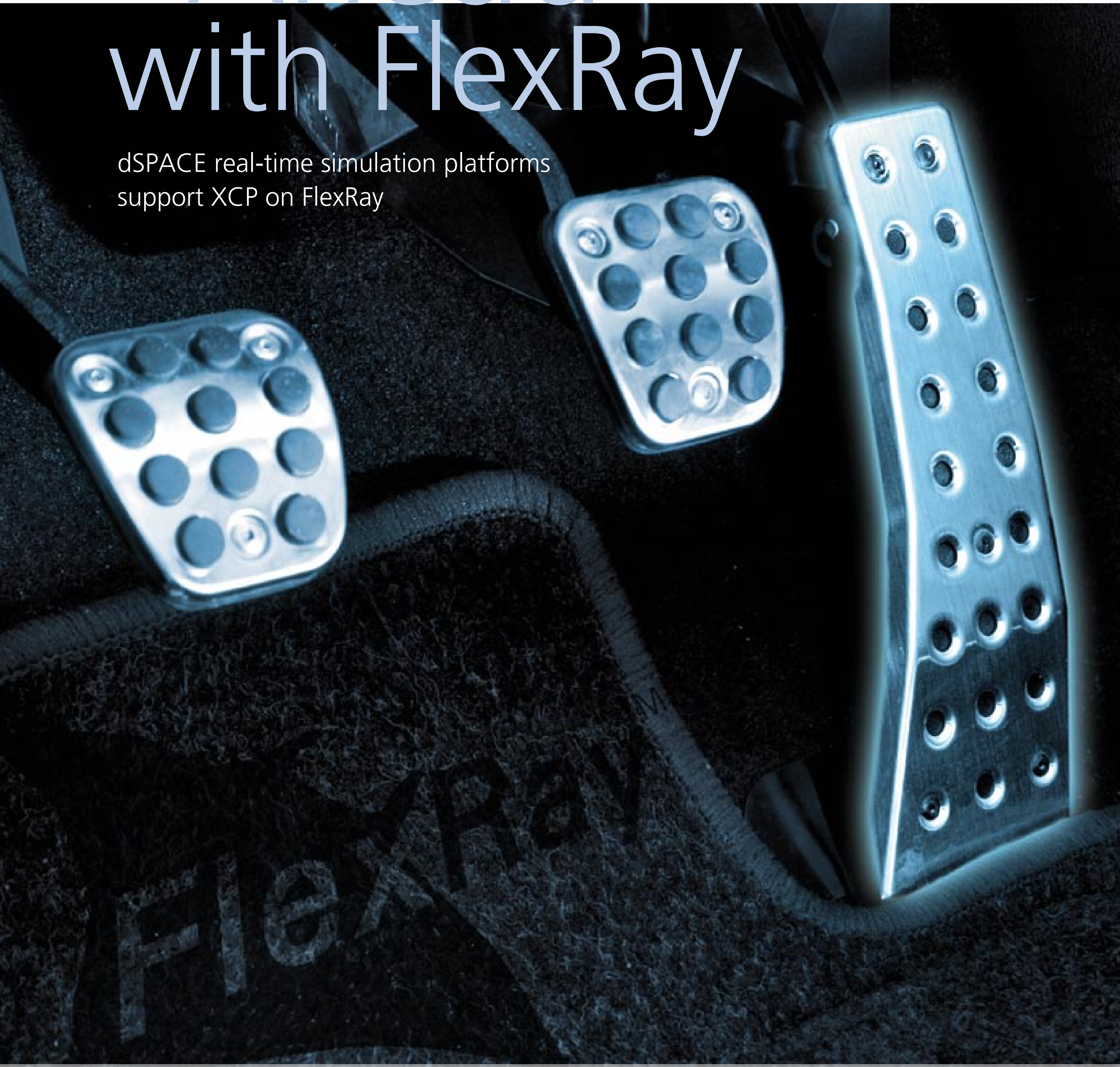
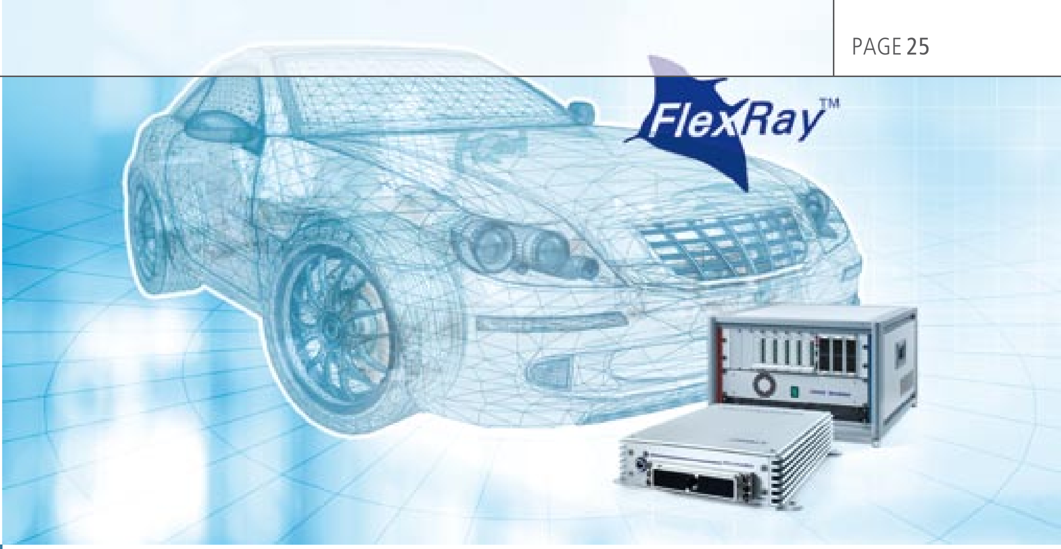
Full Speed Ahead with FlexRay

dSPACE real-time simulation platforms support XCP on FlexRay





ECU communication via FlexRay is becoming more and more essential. For the development and validation of automotive control systems, internal ECU variables have to be accessed via standardized protocols. dSPACE's real-time simulation platforms support the powerful XCP on FlexRay protocol. Typical application cases are function bypassing and ECU testing via hardware-in-the-loop simulation.

Requirements for Modern Bus Systems

When CAN-based bus architectures were developed for use in vehicles, the data traffic on automotive bus systems was only a small fraction of its present volume. LIN was developed as an especially inexpensive bus system, and with its low data transmission rate, it is good for application cases with small data volumes and comparatively long transmission times. CAN and LIN are increasingly coming up against their limits in modern vehicles. The FlexRay protocol is a viable way around this obstacle: It is faster (currently 10 Mbit/s; CAN provides a maximum of 1 Mbit/s, LIN a maximum of 20 Kbit/s) and also allows deterministic data transmission. 'Deterministic' means that time slots are executed in precisely defined sequences. Only one specific node is allowed to transmit in each slot. This ensures that important messages (to the steering system, the brakes, etc.) are executed within defined periods of time. Moreover, precise scheduling of data traffic also makes it easy to combine subsystems.

FlexRay in Production Development

More and more production projects are using the FlexRay bus. The execution of individual tasks in the ECUs can be synchronized via FlexRay. With deterministic data transmission via dedicated FlexRay slots, ECUs that were developed and tested by different suppliers can be networked to form an overall system (figure 1) without mutual effects. Calibrating and testing these ECUs is possible via the standard XCP interface – the universal measurement and calibration protocol published by ASAM. XCP is not only indepen-

dent of the physical data transport layer, it also takes the communication interfaces of today and tomorrow into account. As the FlexRay bus is installed in ever more vehicles, XCP on FlexRay is becoming more important.

RTI Bypass Blockset: Now with XCP on FlexRay

To equip ECU developers for meeting the challenges of future vehicle bus systems, dSPACE now supports communication between dSPACE real-time platforms and ECUs via XCP on FlexRay. In this scenario, the dSPACE system acts as the XCP master, while the ECUs are XCP slaves. This is made possible, for example, by the new XCP on FlexRay interface of dSPACE's RTI Bypass Blockset, a Simulink® blockset for dialog-based configuration of bypass interfaces and real-time ECU access. For bus configuration, the FlexRay interface of the RTI Bypass Blockset builds on the dSPACE FlexRay Configuration Tool. The FIBEX network description file is loaded to the dSPACE FlexRay Configuration Tool, and the XCP on FlexRay parts are configured in addition to the rest of the FlexRay communication. The result is an RTI FlexRay Blockset Library whose blocks can be added to the simulation model. The measurement and stimulus data needed for exchange with a specific ECU is configured in the second step, using the RTI Bypass Blockset. This is done by first reading the ASAP2 file containing the available ECU variables. Then RTI blocks are used to define measurement and stimulus access to the ECU (figure 2). With the RTI Bypass Blockset, up to four ECUs can be addressed in parallel via one FlexRay bus.

Flexible Configuration in the RTI Bypass Blockset

To define the XCP slots which are to be used with each ECU, the RTI Bypass Blockset provides dedicated configuration dialogs for matching the ECU-specific FlexRay communication options (FlexRay buffer) of each ECU to the FlexRay slots actually in use. Users can define the transmission bandwidth for communication with an individual ECU, and the points in time for data transmission within the FlexRay cycle any way they like (figure 3, figure 4). The XCP slots used for a specific data transmission can be either specified manually or assigned automatically by the RTI Bypass Blockset. If the precise positions of the XCP slots in the communication cycle in relation to the bypass hooks within the ECU tasks are known, it is possible to perform data capture or stimulation at defined points in time via XCP on FlexRay. If the positions of the XCP slots in relation to the bypass hooks are not known, automatic assignment can be used. Upcoming data packets are then transmitted with the next available XCP slot. Whichever mechanism is chosen, if XCP slots are used for more than one purpose

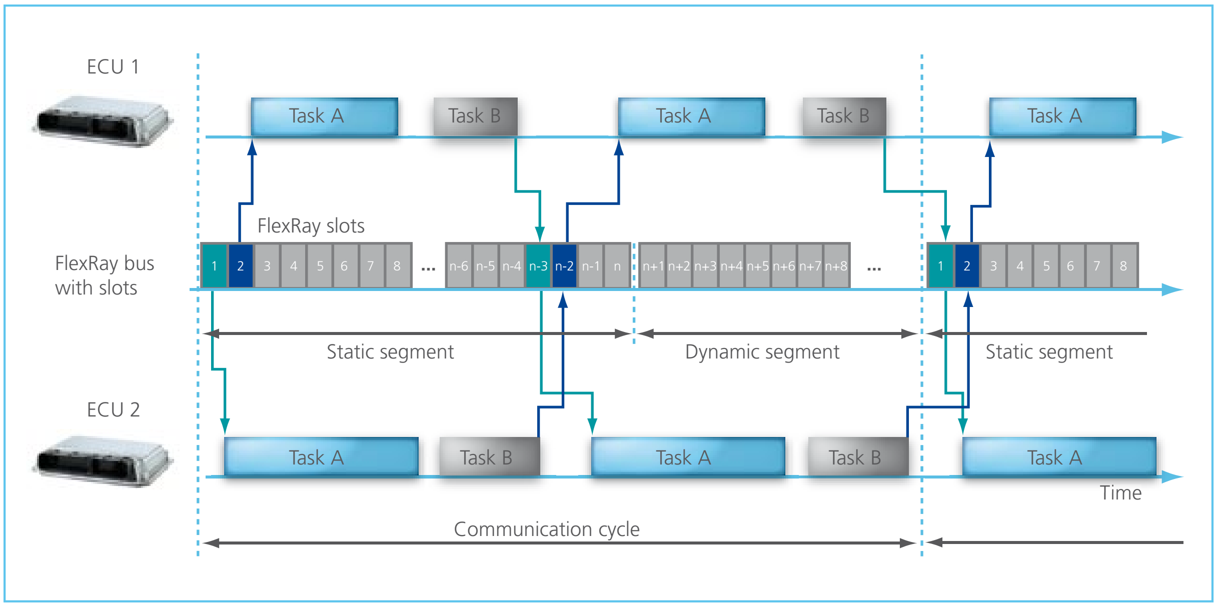
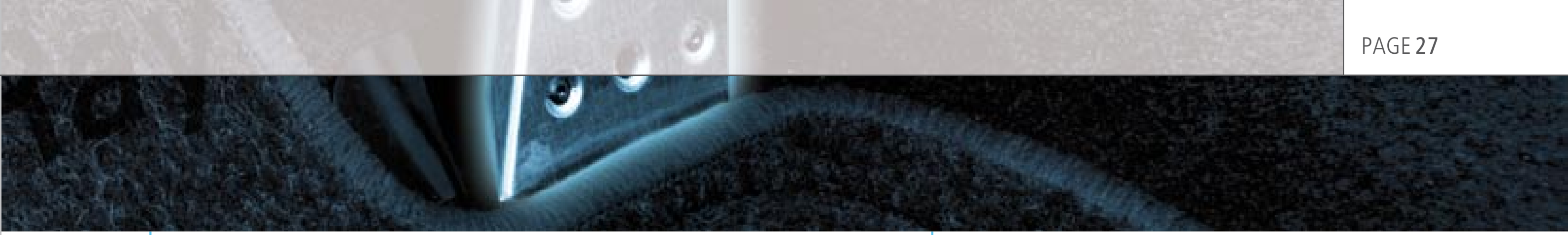


Figure 1: Synchronizing ECU tasks via assigned FlexRay slots. The overall system can be put together without the individual ECUs affecting one another.



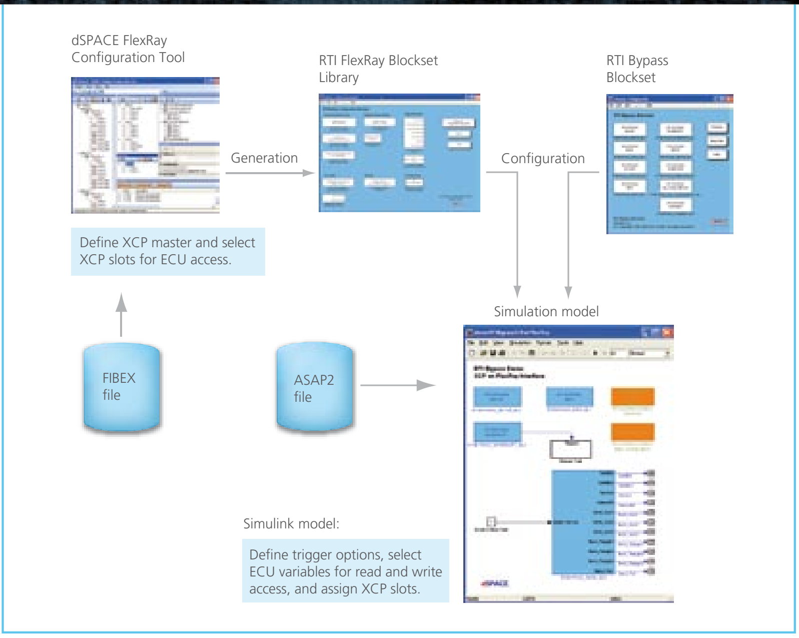

dSPACE FlexRay Configuration Tool
RTI FlexRay Blockset Library
RTI Bypass Blockset
Generation
Configuration
Define XCP master and select XCP slots for ECU access.
Simulation model
FIBEX file
ASAP2 file
Simulink model:
Define trigger options, select ECU variables for read and write access, and assign XCP slots.

Figure 2: Configuration process with the dSPACE FlexRay Configuration Tool and the RTI Bypass Blockset.

(e.g., with one XCP slot communicating alternately with ECU 1 and ECU 2), the sequence of data to be transmitted by the simulation system can be prioritized. There are several ways to synchronize the real-time platforms with the ECU. One is to run calculations on the real-time platform synchronously to specific points in time within the FlexRay cycle, in which case execution is time-driven. Another way is to run calculations immediately after the associated data has been transmitted.

Application Case 1: Bypassing via XCP on FlexRay

In the field of function development, the RTI Bypass Blockset makes it possible to develop and test new ECU functions on a real-time simulation platform, with data exchange and synchronization to the ECU being performed via the existing FlexRay interface. To make data exchange possible, all the necessary data is transmitted to the development system via XCP service calls or bypass hooks, at defined positions in the ECU code. The development system then calculates the new function (bypass task) and returns the output data to the ECU, where it is placed in the ECU's memory with an XCP hook. This sequence can be carried out within one FlexRay cycle (function bypass without sample step delay). Because of the tight time correlation between task executions and FlexRay communication, bypass data is typically exchanged via the static part of the FlexRay

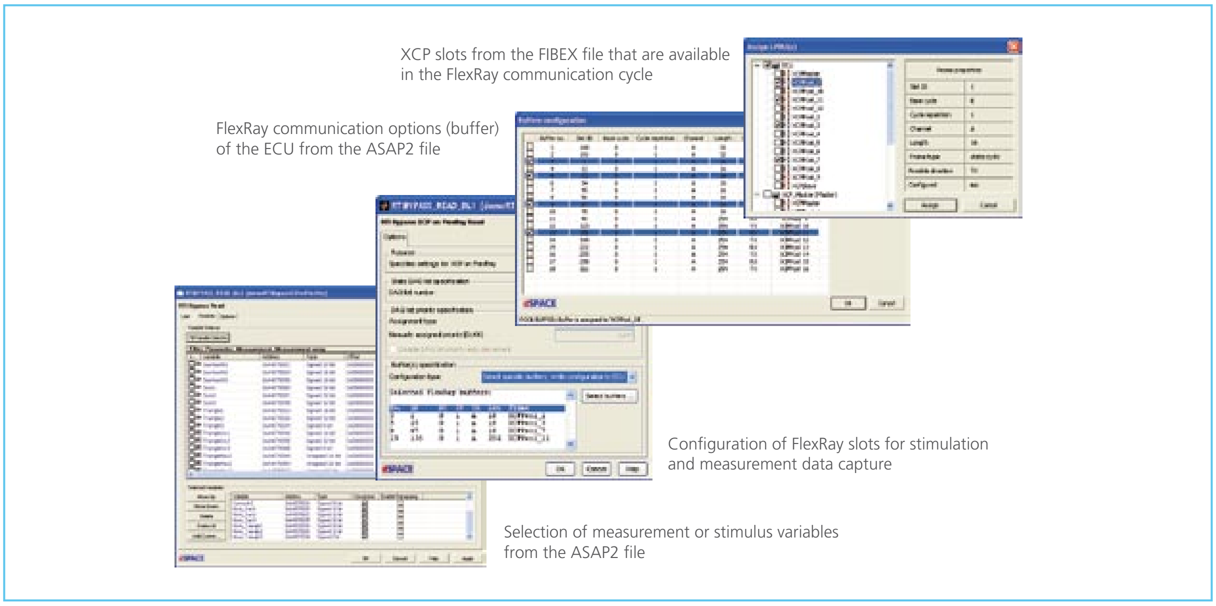

XCP slots from the FIBEX file that are available in the FlexRay communication cycle
FlexRay communication options (buffer) of the ECU from the ASAP2 file
Configuration of FlexRay slots for stimulation and measurement data capture
Selection of measurement or stimulus variables from the ASAP2 file

Figure 3: The RTI Bypass Blockset provides dedicated dialogs for manual configuration of the FlexRay slots used for XCP. As an alternative, the slots can also be assigned automatically.

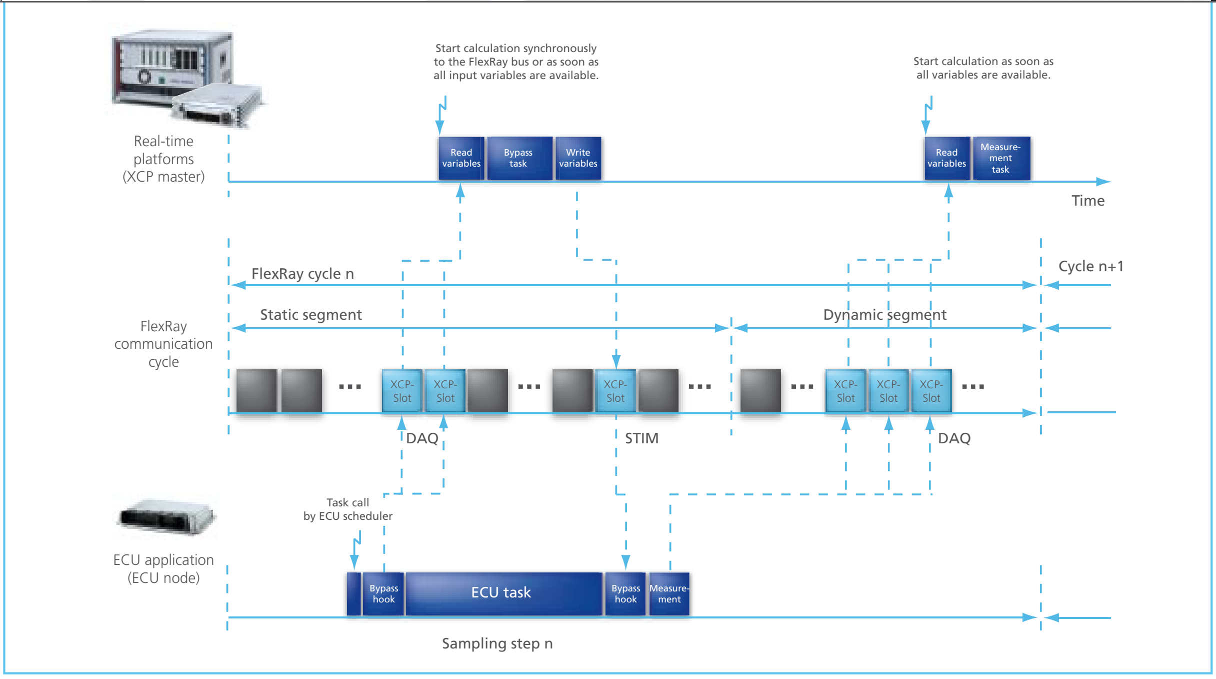

Real-time platforms (XCP master)
Start calculation synchronously to the FlexRay bus or as soon as all input variables are available.
Start calculation as soon as all variables are available.
Read variables
Bypass task
Write variables
Read variables
Measurement task
Time
FlexRay cycle n
Cycle n+1
Static segment
Dynamic segment
FlexRay communication cycle
XCP-Slot
DAQ
STIM
DAQ
Task call by ECU scheduler
ECU application (ECU node)
Bypass hook
ECU task
Bypass hook
Measurement
Sampling step n

Figure 4: FlexRay communication sequence for a function bypass without sample step delay, and for a measurement data capture.

cycle. In applications with sample step delay, the ECU uses the bypass function outputs from the previous sampling step for its calculations. In scenarios like this, FlexRay communication can be performed in the dynamic segment of the FlexRay cycle.

Application Case 2: Data Capture in an HIL Scenario

Another typical XCP on FlexRay application for the RTI Bypass Blockset is monitoring the ECU's internal variables in a hardware-in-the-loop (HIL) scenario. The XCP service calls that are already available in the ECU at the end of each task are usually used for this. XCP communication is performed via the dynamic segment of the FlexRay cycle (figure 4). Thus, using the RTI Bypass Blocksets allows both simulation and ECU data to be recorded on precisely the same time basis. ■

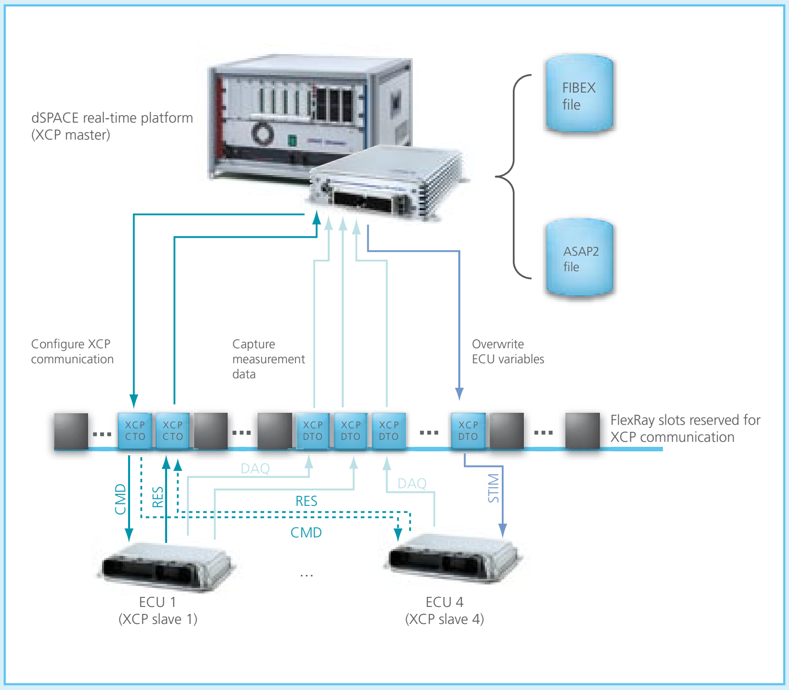


Figure 5: XCP on FlexRay communication: command transfer objects (CTO) are used for command-response sequences (CMD, RES), data transfer objects (DTO) for measurement and stimuli data (DAQ, STIM). dSPACE real-time platforms support parallel access on up to four XCP slaves per FlexRay bus.

XCP on FlexRay Technology

XCP: The Universal Measurement and Calibration Protocol

When an ECU is developed and validated, additional data has to be exchanged with it dynamically in parallel to the statically predefined bus communication. Typical appli-

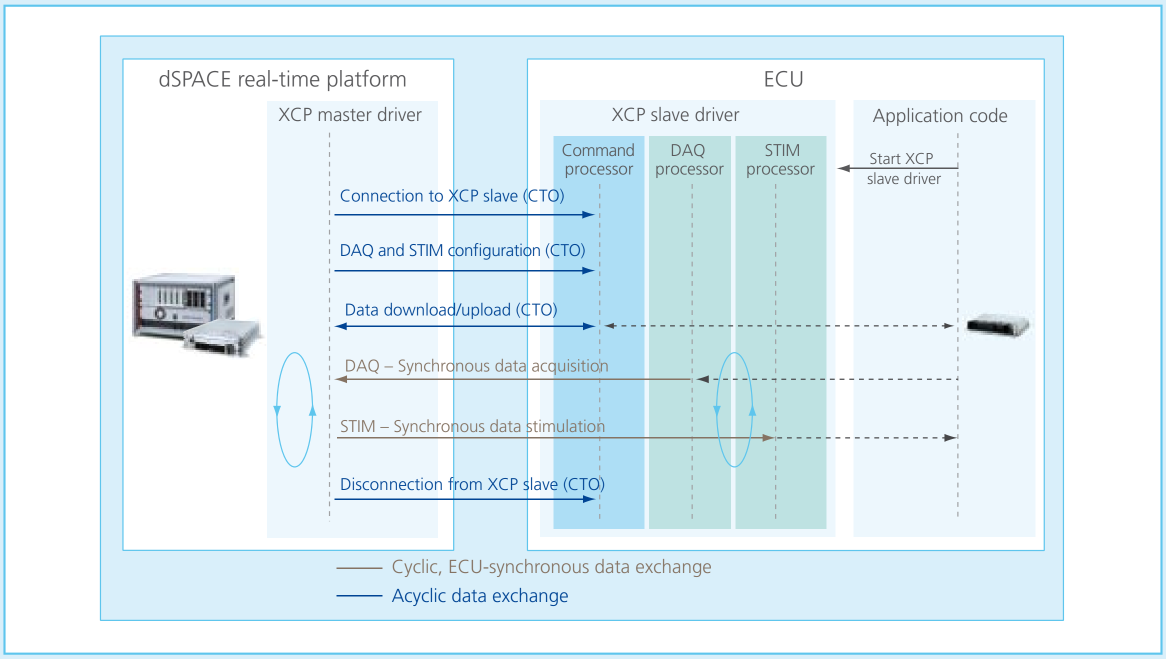

dSPACE real-time platform
XCP master driver
ECU
XCP slave driver
Application code
Command processor
DAQ processor
STIM processor
Start XCP slave driver
Connection to XCP slave (CTO)
DAQ and STIM configuration (CTO)
Data download/upload (CTO)
DAQ – Synchronous data acquisition
STIM – Synchronous data stimulation
Disconnection from XCP slave (CTO)
Cyclic, ECU-synchronous data exchange
Acyclic data exchange

Figure 6: XCP communication sequences. The XCP slave driver consists of three components: a command processor, DAQ processor, and STIM processor.

cations are adjusting control parameters in the ECU software (ECU calibration), ECU-synchronous data acquisition (DAQ) and cyclic overwriting of variables in the ECU RAM (STIM). The XCP protocol is tailor-made for these applications. It is based on a master-slave concept in which either a PC or the real-time simulation platform plays the part of the XCP master, and the ECUs act as the XCP slaves. The XCP driver software is usually a component in the production software inside the ECU. Starting from the XCP master, the XCP communication is configured by means of command-response sequences via XCP command transfer objects (CTO) during system run time. Noncyclic data exchange (download/upload), such as for calibration, is also performed via XCP command packets. When configuration has been completed, ECU-synchronous transmission of measurement and stimulus data is performed cyclically via dedicated XCP data transfer objects (DTO). The recipient does not have to respond to these, so they provide greater bandwidth (figure 6).

Fast and Flexible: XCP on FlexRay

One challenge with XCP on FlexRay is to integrate the typical XCP event-driven communication, e.g. as it appears with CAN, into the deterministic communication of the FlexRay bus. This is done by means of slots in the FlexRay communication cycle which are exclusively reserved for XCP communication and defined as XCP slots in the FIBEX network description file. Because the CTO packets contain slave-related node addresses, XCP communication with all ECUs via FlexRay basically requires only two single XCP slots for exchanging command-response sequences (CMD, RES). If more bandwidth is required, especially with regard to the cyclic transmission of measurement and stimulus data (DAQ, STIM), additional XCP slots can be added to the communication. XCP slots can basically be located in both the static and the dynamic segment of the FlexRay communication sequence. The majority of XCP slots are usually located in the dynamic segment, so that no bandwidth has to be allocated for measurement tasks statically. The static segment is usually reserved for deterministic communication. Assigning XCP slots in the FlexRay cycle means that the total bandwidth provided for XCP and the XCP communication timing both have to be taken into account when the bus is being planned. With some applications, up to around 30% of the FlexRay bus's possible bandwidth is made available for XCP communication.

Parallel Access to Several ECUs

For parallel access to several ECUs in a FlexRay network, arbitrary subsets of the XCP slots can be used for each ECU's communication. The result is dynamic partitioning of the total available bandwidth. The XCP slave configuration of each ECU is placed in the ASAP2 description file. It includes information on the ECU's internal FlexRay buffers that can be used for exchanging CTO and DTO packets, as either statically predefined or dynamically configured during run time (figure 5).