

Universally applicable

Using MOST as general purpose communication protocol



The challenges of today's head unit development are to allow a fast and flexible data exchange to create an end user experience of a rich and comfortable multimedia and infotainment environment. Therewith expanded future use cases, that allow a wireless connection of nomad devices with the system, will arise. A universal communication protocol is needed for these use cases to match all requirements derived from the new scenarios. The answer for the realization of all these tasks is the infotainment bus system MOST.

By Tobias Follner

where each CPU is acting as a self-contained, independent MOST (sub) system that grants pure MOST protocol based connectivity for all locally attached MOST objects (Function-Blocks and Shadows) and, upon closing the IPC-link, also to those objects executed on neighbor CPUs. All of the integrated applications have to be supported as parallel instances of MOST channels to allow device internal MOST communication, hidden from the external interlink with one or more MOST rings.

MOST builds the perfect solution to realize all those use cases and includes all mechanisms suitable to perform remote procedure calls for automotive applications (like MS-DCOM and CORBA are doing in LAN or WAN networking). It defines a comprehensive set of data types and mechanisms to exchange information signal based and provides an expandable and customizable set of already defined application interfaces and message sequences for operation.

MOST protocols are fully OS independent and portable. A so called Virtual MOST Mode allows the integration of nomad devices via Bluetooth or WLAN next to physically connected ECUs. An abstract overview clarifies the architecture of such a system (figure 2).

MOST is more than a powerful physical network, as the MOST specification, like MS-DCOM or CORBA includes all necessary data-types as well as mechanisms for exchanging plain data and for realization of remote procedure calls in a distributed computing environment.

This article shall bring a broad understanding about the requirements needed to realize such scenarios, the protocol itself and an overview of the architecture.

Various architectures with two or even more embedded CPUs as well as multi-core CPU scenarios mostly operated by different OS and connected via different interlinks have to be realized to provide state-of-the-art infotainment systems. That requires a

scalable architecture, supporting the connection of 1, 2 ... n CPUs directly or wirelessly coupled behind one physical MOST node using any speed grade (MOST25, MOST50, MOST150). Possible use cases can be seen in figure 1.

Support of proprietary and standard protocols required

For all applications integrated in a system inhomogeneous OS scenarios are supported where e.g. a Linux process is connecting on eye-level to OSEK tasks or a Windows application thread for data exchange and for triggering remote procedure calls. Furthermore for upcoming use cases the ability is needed to support different solutions for a serial Inter Processor Connection

(IPC), realized on different physical layers, based on standard (e.g. TCP/IP) or proprietary protocols.

A future MOST system should also contain modes

Server supports parallel MOST instances

The FunctionBlocks and Shadows in this overview are locally connected via a server in terms of a virtual MOST mode that forms a bigger MOST cluster. Servers are supporting parallel instances of MOST, pure internal ones powered by a virtual MOST mode and connections with an external

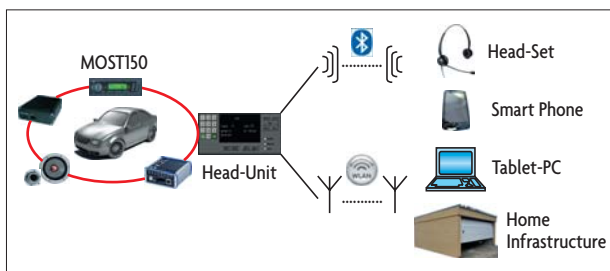


Figure 1. Connecting a car network to different infotainment devices.

MOST ring which is established via INIC.

In comparison MOST builds the basis for a dynamic attaching and detaching of new elements (FBs or Shadows) where multiple MOST channels can be operated either in real or in virtual mode in parallel. Unnecessary traffic can be avoided by the dispatching of MOST messages inside the cluster based on MOST function ID level.

Another possible future use case can be to apply MOST systems in home infrastructures to be able to operate electronic devices from one central point or with a remote control (e.g. reload the infotainment database of your car remotely while staying in your living room).

K2L meets the challenge for a smooth communication between all integrated ECUs with the Automotive Communication System (ACS) to make workflows easier for OEMs as well as suppliers.

K2L ACS includes a framework for embedded application development and a tool chain increasing the productivity of application programmers. It provides a comprehensive set of APIs for easing application development inside a rich C++ class library. The architecture allows a seamless distribution of applications on multiple CPUs sharing one physical

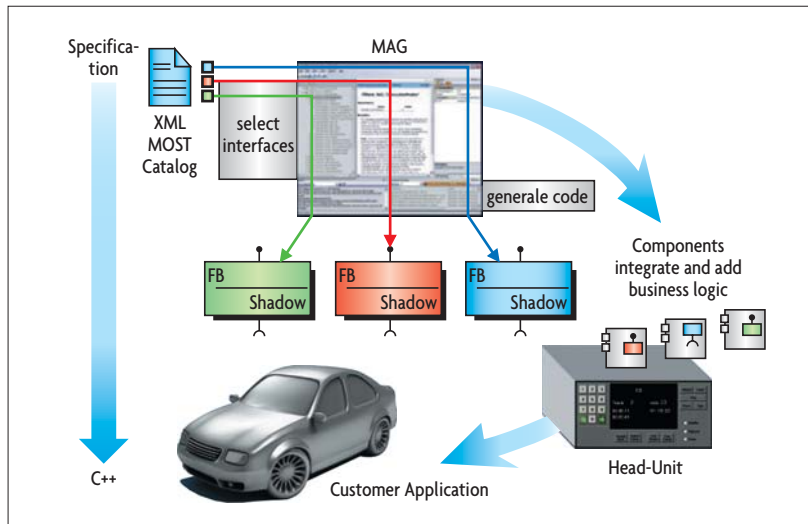


Figure 3. Work flow using K2L's Automotive Communication System.

MOST node. ACS allows developers to concentrate on programming behavioral aspects and business logic without climbing a steep learning curve on MOST by generation of MOST interface logic and data types rather than retyping MOST function catalogs. Specific customized feature requests on application level can be implemented if they are not part of the specification.

ACS supports multiple instances of physical or virtual MOST networks where own implementations on almost any appropriate layer (e.g. UART, SPI, USB, Ethernet) are applicable. The Server is designed to interact seamlessly with other K2L

products such as the K2L MOST Driver or the K2L Universal Gateway. The workflow can be seen in figure 3. The user that applies MOST technology for a modern application development benefits by getting more solid code in less time without re-typing of interfaces for internal or external MOST applications. Interfaces and sequences (MOST function catalog and sequence charts) can be defined clearly in machine readable form. No manual coding is needed for the generation of interfaces and stereotypic patterns like notification. All data types for encoding and decoding can be accessed on abstract and symbolic signal or function names. The implementation of business logic is done by deriving own applications with object-oriented patterns from generated stub classes. Through this, round trip engineering problems on specification updates can be avoided. The MOST protocol allows the provision of an identical programming model and API, independent of applied OS and CPU platforms.

sj

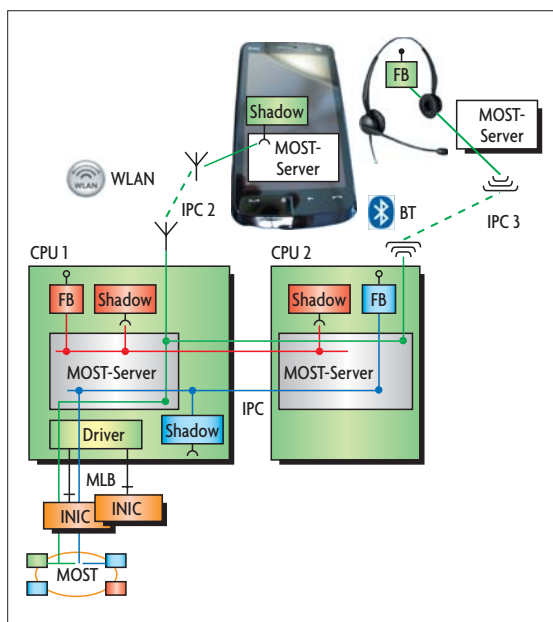


Figure 2. Virtual MOST mode: ECUs can be connected to mobile devices via Bluetooth or WLAN.



Tobias Follner
 hold a bachelors degree in sales engineering from Karlsruhe University and he is preparing for his masters degree in business psychology during an extra-occupational program at the University in Erding (Germany). Since 2008 he is working in the sales and marketing department of K2L GmbH.