Usage of AUTOSAR Diagnostic Modules in a MOST Electronic Control Unit

Paul Hoser

BMW Car IT GmbH, Petuelring 116, 80809 Munich, Germany

Abstract: The AUTOSAR architecture is specified to provide drivers and services needed by automotive applications. Today's AUTOSAR architecture is primarily designed to fulfill the needs of applications from the body, chassis and power-train domain. Unfortunately the AUTOSAR architecture lacks some features high-end infotainment Electronic Control Units (ECUs) require. Among these is also the support for MOST. Thus, building a MOST-based high-end infotainment ECU with an AUTOSAR architecture is not an option. Nevertheless it is highly desirable to use some of the modules from the AUTOSAR architecture for building high-end MOST ECUs. This saves a lot of development, testing and integration time.
Good examples for such AUTOSAR modules are all the modules realizing the diagnosis functionality. In order to use these modules on a MOST-based high-end infotainment ECU an AUTOSAR environment has to be emulated on the ECU. Within that emulated AUTOSAR environment AUTOSAR's Diagnostic Communication Manager (DCM), DEM Diagnostic Event Manager (DEM), Runtime Environment (RTE) and a Central Diagnostic Software-Component can be executed. This brings standardized diagnosis functionality from the AUTOSAR architecture to a MOST ECU.
This approach can be taken even further by extending the AUTOSAR emulation with an additional module which translates MOST Function Block calls to AUTOSAR Virtual Function Bus (VFB) calls and vice versa. This enables AUTOSAR Software-Components deployed on the RTE to communicate via the MOST ring with MOST Function Blocks (FBlocks). This allows the integration of AUTOSAR Software-Components into the MOST-based ECU.

Keywords: AUTOSAR, MOST, Diagnosis, DCM, DEM

1 Motivation

For every vehicle there is a set of functionality specified which has to be provided by every ECU. Among those common functionalities there are such functions as handling remote request for updating the ECU's software, configuring the ECU and returning status data for the diagnosis of the ECU. Specification documents exist which describe these common functionality. Usually this is done in plain text form using human language.
This informal way of specifying software functionality has proven to be very error prone. The chances that the implementation from two developers of the same specification will behave in the same way are not very high. This makes the integration of the ECUs into the vehicle a very difficult and time consuming task. Numerous iterations are needed until the ECUs provided by different vendors behave in the same way. Vehicle manufacturers have reacted to this situation by providing their ECU suppliers with an implementation of a basic ECU architecture. This basic ECU architecture implements all the demanded common functionalities of the ECU. This makes the integration of the ECUs easier for all involved parties.

2 AUTOSAR

In the last years the vehicle manufacturers, suppliers and tool vendors have taken the idea of a common basic ECU architecture a step further. They joined forces in the AUTOSAR partnership. The partnership's goal is the specification of one common basic ECU architecture which is to be used by all vehicle manufacturers. The specification of the AUTOSAR architecture has matured over the last years. Numerous vehicle manufacturers have already begun with the migration of their basic ECU architecture implementation towards an AUTOSAR compliant implementation.
Today's architecture specified by AUTOSAR provides drivers for automotive specific technologies and services needed by automotive applications. Unfortunately, it does not support all the features required by high-end infotainment ECUs.
Infotainment applications therefore are often built on general purpose operating systems like WinCE, Linux or QNX which fulfill their needs. The drawback of using general purpose operating systems is that they lack automotive specific services.
From an integration point of view it would be desirable to reuse existing AUTOSAR implementations of these basic automotive services within high-end infotainment ECUs.

3 Architecture of the AUTOSAR Diagnosis

In this paper the AUTOSAR diagnosis functionality shall be used as an example to describe an

approach for reusing existing AUTOSAR modules and components in a high-end infotainment ECU.
The AUTOSAR architecture distinguishes between Basic Software (BSW) Modules and Software Components (SWCs).
The BSW Modules encapsulate the infrastructural functionalities and services of an ECU. The interfaces and behavior of the BSW Modules are well specified by the AUTOSAR partnership.
Application functionality is to be implemented as a SWC. SWCs exceptionally communicate with each other and with BSW Modules via the RTE. The RTE is an implementation of AUTOSAR's middleware concept which is called VFB.

Within the AUTOSAR architecture as specified in release 2.0 the diagnosis functionality of an ECU is realized by two BSW Modules and one SWC:

- **Diagnostic Communication Manager (BSW)**
  The DCM handles the communication with external diagnosis tools. It implements the necessary timing management and message segmentation as specified for the KWP2000 and UDS protocol. Some important diagnostic services (e.g. ReadDTCInformation) are directly handled by the DCM.

- **Diagnostic Event Manager (BSW)**
  The DEM is responsible for managing error entries in the ECU. This includes storing them persistently.

- **Central Diagnostic Software Component (SWC)**
  The Central Diagnostic SWC's responsibility is to implement the processing of vehicle manufacturer specific diagnosis requests and dispatch the requests to other SWCs.

4 Integration of BSW Modules

AUTOSAR BSW Modules were designed to run within the AUTOSAR architecture. They therefore depend on the availability of services provided by other BSW Modules. The system surrounding the BSW Modules has to pretend to be an AUTOSAR architecture by providing these services. The system has to emulate the required services of the missing BSW Modules. In the approach presented in this paper emulators are implemented for the missing BSW Modules which provide exactly the required services. This means that the emulators provide the specified interfaces and behavior of the required services to the integrated BSW Modules. The services are implemented with the means of the underlying operating system. Figure 1 shows the architecture of this approach.

5 DCM Integration

For the integration of the DCM services provided by the ComManager, the BSW Scheduler and the PDU Router have to be emulated. In this section the required services are described and a rough idea is given how these could be emulated.

Within the AUTOSAR architecture the ComManager is responsible for activating the communication BSW Modules. This is done as soon as the lower communication infrastructure of the AUTOSAR architecture is initialized. The DCM is a communication BSW Module and therefore expects to be initialized by the ComManager. The ComManager Emulator has to be hooked up with the `SystemCommunicationInit()` callback function of the MOST Supervisor Layer II. As soon as the emulator receives the callback from the MOST Supervisor Layer II it can initialize the DCM by calling it's the expected callback function as specified in the AUTOSAR specifications.

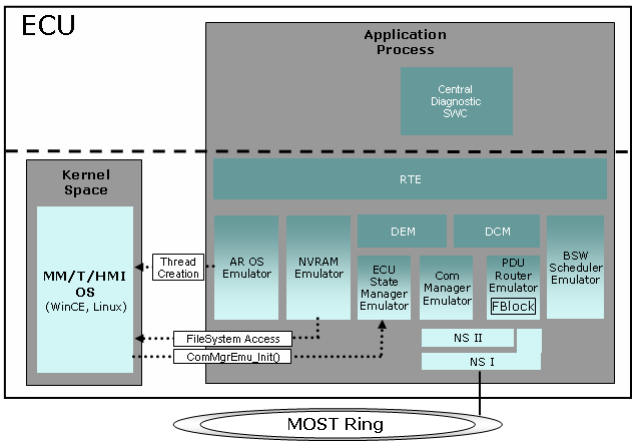

Figure 1 - Integration of AUTOSAR Diagnosis

The DCM expects its main function for processing diagnostic requests to be triggered periodically by the BSW Scheduler. The BSW Scheduler Emulator has to implement the periodical triggering of the DCM's main processing function with the means of the underlying operating system. For an operating system which is compliant to the POSIX standard it could spawn a thread which calls the DCM's main processing function. After executing the main processing function the thread would wait for a signal from a periodic timer which is armed by the BSW Scheduler Emulator.
The periodic triggering of the DCM is a problematic requirement. The main processing function has to be triggered with a high frequency because besides processing the diagnostic request it also calculates the timeouts. Only by triggering the main processing function very frequently the jitter for the timeout can be kept at a minimum. The high trigger frequency implies an increase in the consumption of Central

Processing Unit (CPU) time. This is especially undesirable as most of the time no diagnostic request has to be processed anyway. Within a high-end infotainment ECU the wasted CPU time might even be higher than in an AUTOSAR ECU as process switching might be necessary if multiple processes are executed in parallel.
Another feature of the DCM can be exploited to reduce the CPU time being wasted. The DCM is specified to inform the ComManager every time it has received a diagnostic request or it has ended processing a diagnostic request. The ComManager Emulator can be implemented to forward these callbacks from the DCM to the BSW Scheduler Emulator. The BSW Scheduler Emulator can then arm and disarm the timer based on these callbacks. By this implementation the periodical triggering of the DCM main processing function is reduced to the period of time needed for actually processing diagnostic requests.

The PDU Router provides an interface for receiving and sending messages on an abstract level. The DCM uses this interface for receiving and sending its diagnostic messages. The main responsibility of the PDU Router within the AUTOSAR architecture is to dispatch incoming and outgoing messages to the BSW Modules which are in charge for processing them. Within the MOST Netservices for incoming messages the Command Interpreter takes care of that. An implementation of the PDU Router Emulator can exploit this by providing a diagnosis FBlock. The Command Interpreter simply is configured to forward incoming diagnostic requests to that diagnosis FBlock. For outgoing messages the emulator will use the Application Message Service (AMS) provided by the Netservices Layer I.
In both cases – incoming and outgoing messages – the PDU Router Emulator acts as an adapter. It converts the MOST messages into the message format required by the DCM and vice versa.

As mentioned in the previous section the DCM does not forward all diagnosis requests to the Central Diagnosis SWC. Some important diagnosis requests are being directly processed by the DCM. For those requests concerning the monitoring and management of the ECU's error entries the DCM communicates with the DEM. The DEM does not have to be emulated as it also is being integrated.

6 DEM Integration

The DEM itself depends on the services of the ECU State Manager and the NVRAM Manager. In this section the required services of these two BSW Modules are described and a rough idea is given how these could be emulated.

The initialization of the DEM is not triggered by the ComManager as it does not depend on the communication infrastructure to work correctly. It is initially triggered by the ECU State Manager.
The DEM expects to be initialized in a two-phased way. It provides two callback functions. The first callback function initializes the DEM's ability to accept error entries from other BSW Modules and from SWCs. At that time the DEM will not yet store the error entries persistently. This functionality of the DEM is initialized by the second callback function. It is to be called as soon as the persistent memory management (NVRAM Manager) is initialized.
When integrated into a high-end infotainment ECU the DEM is not part of the operating system. It is run as part of an application process.
At the point in time when the DEM is started the operating system including the file system is already up and running. A two-phased initialization approach is therefore not necessary. The emulation of the ECU State Manager simply boils down to calling the two callback functions immediately when the process running the DEM is started.
The DEM requires the services of the NVRAM Manager for writing/reading error descriptions to/from persistent memory. The interface of the NVRAM Manager works with block identifiers to determine where to write/read data to/from the persistent memory. The NVRAM Manager Emulator needs to map the block based interface to a file system. A pragmatic mapping approach is to simply create a file for each block.

7 Central Diagnostic SWC Integration

Besides the listed BSW Modules the DCM also depends on the presence of the Central Diagnostic SWC. As mentioned before diagnostic requests specific to a vehicle manufacturer are not handled by the DCM. They are delegated to the Central Diagnostic SWC.
The AUTOSAR architecture defines that SWCs and BSW Modules do not communicate directly with each other. They exchange data via the RTE. To enable the communication between the DCM and the Central Diagnostic SWC the integration of the RTE into the high-end infotainment ECU is required.
The RTE depends on the services of the AUTOSAR OS. Hence an emulation of the AUTOSAR OS is required. This means that calls to AUTOSAR OS services from the RTE have to be mapped to the appropriate OS services of the underlying operating system. For a POSIX compliant operating system this means mapping the AUTOSAR OS services to threads, timers, mutexes and conditions.

The integration of the RTE not only allows the communication between the DCM and the Central Diagnostic SWC but also enables a further use case. As SWCs only depend on the RTE the presence of

an RTE enables the deployment of SWCs in general. Unfortunately AUTOSAR does not provide a stack for MOST. This implies that the deployed SWCs can not communicate with SWCs on other ECUs (remote communication).
By providing an additional module which will be called a MOST Proxy the previous described limitation can be dropped. The MOST Proxy proxies the interfaces of all the external SWCs with which the deployed SWCs communicate. Instead of communicating with their actual communication partners, located outside the ECU, the SWCs communicate with the MOST Proxy (see Figure 2). This way the RTE only has to handle communication between locally present components. From the RTE's point of view no MOST access is required. The MOST proxy takes care of converting the AUTOSAR VFB calls (Sender/Receiver, Client/Server) to MOST Function Block calls (Set, Get, Status, …) and vice versa.
With the MOST Proxy in place AUTOSAR SWCs deployed on the MOST ECU are able to communicate over the MOST ring with any other SWC or FBlock.

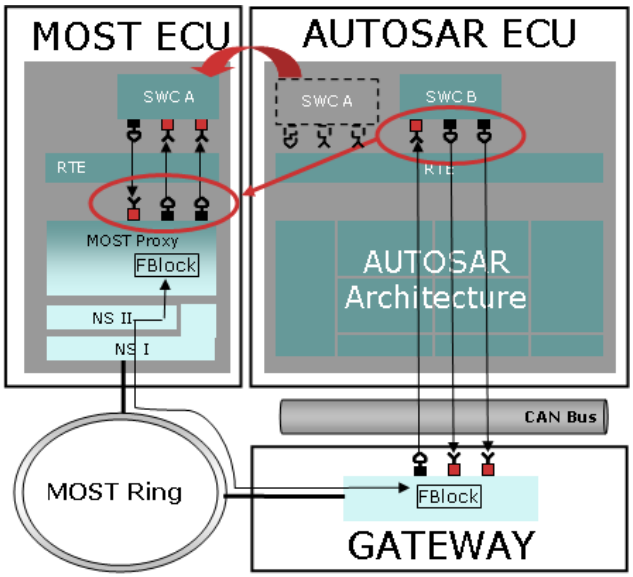

Figure 2 – Communication via MOST Proxy

8 Conclusion

Developers of MOST ECUs can benefit from reusing selected existing implementations of AUTOSAR BSW Modules and SWCs. As exemplarily shown for AUTOSAR's diagnosis functionality AUTOSAR BSW Modules and SWCs can be ported and executed on a high-end infotainment ECU. This is achieved by partially emulating the AUTOSAR architecture.
Still the described integration of the AUTOSAR modules is rather loose. A stronger coupling between the MOST stack and the ported AUTOSAR diagnostic modules would be desirable. Errors occurring within the MOST stack could also be passed to the DEM. This would make them accessible to external diagnostic tools in the same way as the other errors.

9 References

[1] AUTOSAR development partnership: "*AUTOSAR: Technical Overview*", http://www.autosar.org/download/r2/AUTOSAR_TechnicalOverview.pdf, 2006

[2] AUTOSAR development partnership: "*AUTOSAR: Specification of Diagnostic Communication Manager*", http://www.autosar.org/download/r2/AUTOSAR_SWS_DCM.pdf, 2006

[3] AUTOSAR development partnership: "*AUTOSAR: Specification of Diagnostic Event Manager*", http://www.autosar.org/download/r2/AUTOSAR_SWS_DEM.pdf, 2006

[4] AUTOSAR development partnership: "*AUTOSAR: Specification of Communication Manager*", http://www.autosar.org/download/r2/AUTOSAR_SWS_ComManager.pdf, 2006

[5] AUTOSAR development partnership: "*AUTOSAR: Specification of PDU Router*", http://www.autosar.org/download/r2/AUTOSAR_SWS_PDU_Router.pdf, 2006

[6] AUTOSAR development partnership: "*AUTOSAR: Specification of ECU State Manager*", http://www.autosar.org/download/r2/AUTOSAR_SWS_ECU_StateManager.pdf, 2006

[7] AUTOSAR development partnership: "*AUTOSAR: Specification of BSW Scheduler*", http://www.autosar.org/download/r2/AUTOSAR_SWS_BSW_Scheduler.pdf, 2006

[8] AUTOSAR development partnership: "*AUTOSAR: Specification of Operating System*", http://www.autosar.org/download/r2/AUTOSAR_SWS_OS.pdf, 2006

[9] AUTOSAR development partnership: "*AUTOSAR: Specification of RTE Software*", http://www.autosar.org/download/r2/AUTOSAR_SWS_RTE.pdf, 2006

10 Glossary

ECU: Electronic Control Unit
BSW: Basic Software
SWC: Software Component
DCM: Diagnosis Communication Manager
DEM: Diagnosis Error Manager
AMS: Application Message Service
CPU: Central Processing Unit
FBlock: Function Block