HAVEit

Highly automated vehicles for intelligent transport

7th Framework programme
ICT-2007.6.1
ICT for intelligent vehicles and mobility services
Grant agreement no.: 212154

The future of driving.

Deliverable D21.3

Generic platform core demonstrator available in lab

Version number	Version 1.0
Dissemination level	CO / public version
Lead contractor	Continental Automotive GmbH
Due date	31.01.2010
Date of preparation	27.01.2010

Authors

Name	Company
Michael Gutknecht	USTUTT
Sergej Bahnmüller	USTUTT
Philipp Luithardt	USTUTT
Sergiy Prutyanyy	USTUTT

Project Managers

Alfred Hoess
Continental Automotive GmbH
Siemensstrasse 12
93055 Regensburg, Germany
Phone +49 941 790-5786
Telefax +49 941 790 99 5786
E-mail: Alfred.Hoess-EXT@continental-corporation.com

Holger Zeng
Continental Automotive GmbH
Siemensstrasse 12
93055 Regensburg, Germany
Phone +49 941 790 92330
Fax. +49 941 790 99 92330
E-mail: holger.zeng@continental-corporation.com

Project Co-ordinator

Reiner Hoeger
Continental Automotive GmbH
Siemensstrasse 12
93055 Regensburg, Germany
Phone +49 941 790 3673
Fax +49 941 790 99 3673
E-mail reiner.hoeger@continental-corporation.com

Executive summary

This document summarizes the generic platform core demonstrator which is now available in the USTUTT lab. Therefore a short introduction is given, explaining the meaning of the generic platform core in the overall project context. Finally, a short outlook is given explaining the role of the generic platform core described within this deliverable according to the final vehicle platforms.

Background

The main objective of the HAVEit project deals with the improvement of safety in road traffic. Therefore, beside others, one measure was defined to achieve this goal: the development of the next generation assistance and safety functions (ADAS) which provide up to highly automated driving in order to support the driver in exhaustive tasks or in order to avoid accidents. While today's assistance systems generally affect the overall driving task just by quite slight interventions into the driver's commands (e.g. ESP, ABS, …) or by informing the driver about the environment around the vehicle (e.g. lane-change assistant, parking-assistant, …), the new ADAS, which shall be developed within the HAVEit-project, are intended to go much further beyond that. In the highest automation level, these new ADAS shall initiate own distinctive acceleration, steering or braking commands in order to control the overall vehicle. In consequence, these new ADAS require full access to the very basic safety functionalities of the vehicle: steering and braking.

Although a lot of technology found its way into today's vehicles, the very basic safety-related functionalities like braking or steering are still realised in a very traditional manner, i.e. mechanically. This is due to the low failure probability of mechanical components as well as its favourable failure behaviour (e.g. rarely sudden loss of functionality). These mechanical structures ("M architectures") now need to be broke open in a manner, allowing the electronically realised ADAS-algorithms to interact with them.

One suitable way to do so will surely be the entire replacement of the today's mechanical control-chain by a respective electromechanical one. Nevertheless, such a solution will only be acceptable by the society if at least a safety level can be achieved, which is comparable to today's mechanical solutions. Unfortunately, concerning safety, electrical (simplex) components pose a characteristic less favourable than mechanical components: not only the failure probability of an electrical component is much higher, but also the failure behaviour itself is less favourable (e.g. sudden loss of the entire functionality). Therefore, it is not sufficient to solely replace mechanical components by electromechanical ones. Redundancy needs to be applied.

The usage of redundant components provides additional challenges for the system designer, because though there are surplus components, there still needs to be non-ambiguous, fault free chains of effect. In consequence, besides the realisation of the pure driving / assistance functionality the following issues have to be taken into consideration:

- How can faulty components dependably be detected, identified and isolated ('detection and passivation concepts')?
- How can the functionality be kept up as long as possible ('degradation and reconfiguration concepts')?
- How can consistency and integrity of the signal flow be ensured ('communication')?

As one can see, the realisation of a full electrical access to the vehicle is not only challenging in what the vehicle topology is concerned, but also in the high amount of management tasks, required to ensure a proper execution.

Within the predecessor project SPARC it could be shown that the implementation of a redundant 2E platform is a suitable solution for providing the full electrical access, required by new ADAS. In the SPARC project, a safe platform core was realised, which not only enabled the safe execution of the ADAS applications but also provided the management of the redundant components. One of the advantages of the SPARC approach was the detachment of safety (redundancy) and functionality (ADAS function). Within the platform core, the overall redundancy management provided a virtual simplex world to the application (simplex minded development), which simplified the ADAS development significantly.

In HAVEit, the SPARC approach shall be picked up and enhanced by the following innovation: SPARC was driven by the development of the mentioned platform. Its realisation covered a lot of individual tasks and solutions. Most of the required software was written manually. In HAVEit, we intend to examine the SPARC solutions in order to identify the very basic functionalities and create reusable, configurable modules. In parallel, a configuration process shall be developed which finally provides the particular configuration of the dedicated modules. Therewith, a systematic meta tool chain will be developed, all in all allowing an “easy” realisation of safety-critical, fail-tolerant 2E-platforms.

Platform Concept

Within the HAVEit project, two vehicles will be equipped with a 2E platform core in order to provide a full electrical access to the safety critical functionalities “steering” and “braking”. First vehicle will be the “Extended Joint System Demonstrator” of WP4100, which will be equipped with a steer-by-wire functionality (for further detail see [4]). This means interruption of the original mechanical steer column and introduction of according electrical components in order to allow the required access. Besides two electrical paths (“2E architecture”), a mechanical path will be kept as backup in order to allow for example transfer of the demonstrator vehicle without appliance of the steer-by-wire functionality. The second vehicle will be the truck of WP4200. Here, the entire mechanical chain of effects of a common brake system will be replaced by electrical / electromechanical components. This even includes the original mechanical brake actuators – in order to achieve brake-by-wire functionality (for further detail see [3]).

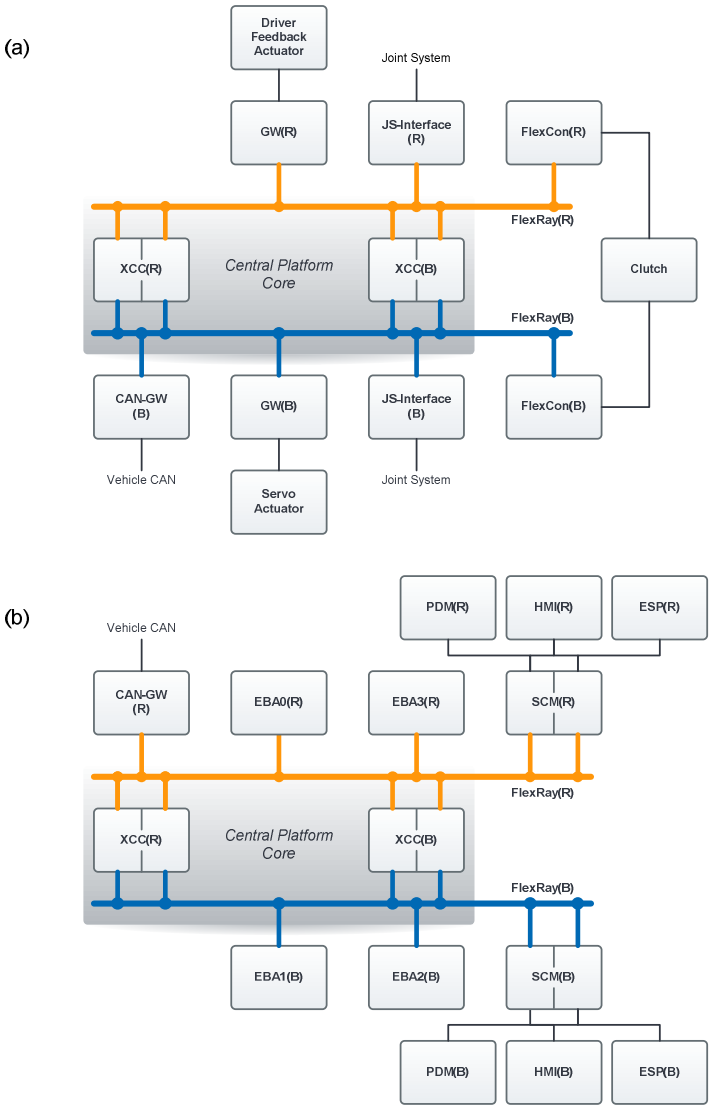

Figure 1: 2E architectures of the vehicles of WP4100 (a) and WP4200 (b)

Figure 1 depicts the simplified topologies of the vehicles mentioned above. It becomes obvious that both topologies are built up in a quite similar way. On one hand, they consist of components like sensors, actuators, or interfaces which are dependant on the particular system functionality (braking or steering) which is intended to be realised in the according vehicle. On the other hand, each topology contains an identical core, the so called “central platform core” (CPC). The CPC consists of multiple redundant XCC-ECUs (for further detail see [1]) as well as the FlexRay buses, providing the entire platform communication.

As mentioned above, the platform core shall not only enable the plain execution of system applications but shall also provide the management tasks required for controlling the overall platform behaviour. These management tasks are realised in terms of software modules, which are generally structured as follows:

- **System specific layer:**
  Layer providing the system specific functionality as well as the system specific management tasks.
- **Platform(-core) specific layer:**
  Layer providing the management of the platform core.
- **Integrity and communication layer:**
  Layer providing the commonly used tools for failure detection and classification as well as the overall communication.
- **Operating system:**
  Layer providing the basic operating system functionality.

Each of the mentioned layers contains one or multiple software modules, providing the particular functionality (for further details see [2]). These are:

- **System specific layer**
  - *Application (Appli)*: software module providing the plain system functionality (steering, braking, …)
  - *Application Management (AppMa)*: software module providing the management of all functionality dependent components as well as management of the application itself (i.e. execution level, operation mode, or determination of performance level).
- **Platform(-core) specific layer**
  - *Platform Management (PlaMa):* software module providing management of the modules and communication within the platform core.
- **Integrity and communication layer**
  - *Module-Input-Output (MIO)*: software module providing module input integrity, mainly by detecting faulty input data concerning the aggregates.
  - *Message Router (MsgRouter)*: software module which performs the transformation “message-frame to signal” and the other way round.
  - *Failure Management (FailMa)*: software module which performs the classification of failures.
  - *Consolidation Module (CoMo)*: software module which performs the consolidation of data between the two lanes of a XCC.
- **Operating system**
  - Database
  - Jobtables/Scheduling
  - Driverhandler
  - FlexRay configuration

According to the particular layer, the software modules can be assigned to the attributes “required for system functionality”, “required for platform core” and “required for both” (see figure 2).

As a consequence, in order to provide the generic platform core demonstrator, the following software modules are intended to be available:

- Platform Management
- Module-Input-Output
- Message Router
- Failure Management
- Consolidation Module
- Database
- Jobtable
- Driverhandler
- FlexRay configuration.

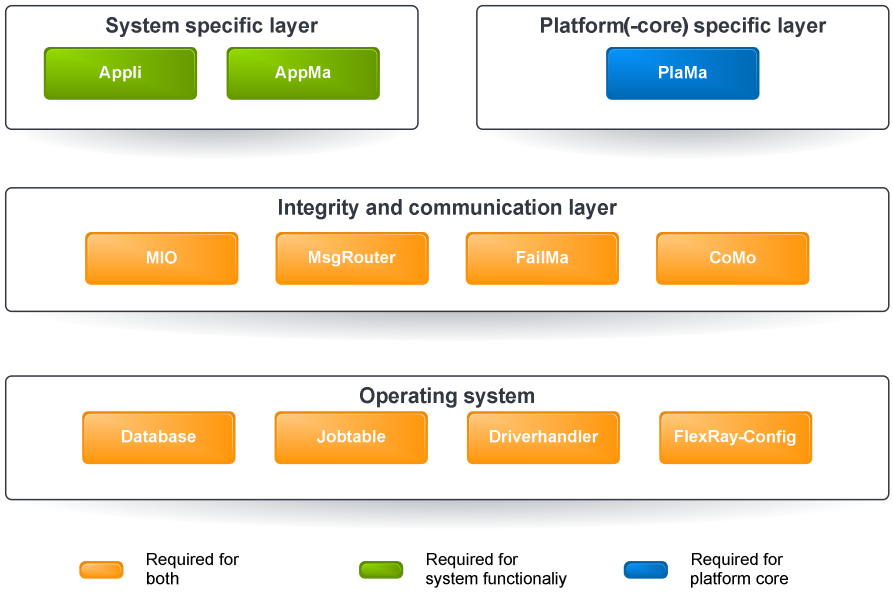

Figure 2: Software modules

In summary, the central platform core can be depicted as shown in figure 3.

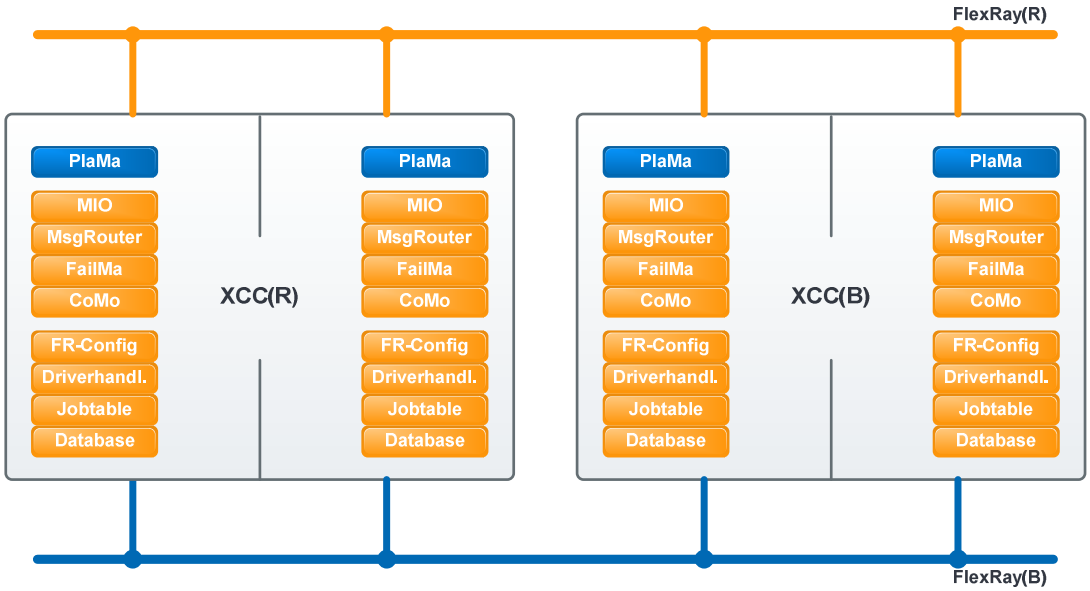

Figure 3: Central Platform Core (HW and SW)

Conclusions

In addition to the applied hardware components, our first implementation of the meta-tool could be verified to deliver useful results concerning configuration of the generic core.

The generic platform core is a very important milestone on our way towards configurable platform cores and already establishes a basis for the upcoming vehicle specific configuration of the platform cores dedicated to WP4100 and WP4200. Nevertheless, there is still some work to be done until the finalisation of the software and configuration package. Surely, the meta tool which provides the configuration data needs to be continuously adapted to upcoming needs during configuration of the vehicle-specific data. In parallel, the online configurable software bodies need to be adapted (if required).

Further, the vehicle specific information needs to be collected and inserted into the meta-tool to derive the vehicle specific configuration. Therefore it is necessary to check whether the existing (configurable) software module bodies are able to provide all the functionalities required by the particular system or if the software module bodies need to get enlarged. Equally, it will take again some effort to include all the vehicle specific signals into the platform, because of the different redundancy management mechanisms which work with these signals. Here, according to the degree of redundancy, the signal type, and the degradation level combination, some unpredictable combinations could occur which would require further adaption of the meta-tool as well as the software module bodies.

Finally, to complete the development process the vehicle-specific bare platforms within WP4100 and WP4200 need to be verified. Only this step guarantees that the preliminary defined redundancy management, platform management and the application perform as specified and the required safety levels can be achieved.

References

[1] D21.1 “CSC & XCC HW-development complete“, HAVEit deliverable, 2009

[2] D21.2 “Software and Configuration Process Concept available”, HAVEit deliverable, 2009

[3] D23.1 “Brake by wire for challenge 4.2”, HAVEit deliverable, 2010

[4] D23.2 “Steer by wire for challenge 4.1”, HAVEit deliverable, 2010