Embedded multi-core systems for mixed criticality applications in dynamic and changeable real-time environments

Project Acronym:

EMC²

Grant agreement no: 621429

Deliverable no. and title	**D3.4 – Dynamic Run-time Environments, Implementation, Prototyping**	
Work package	WP3	Dynamic runtime environments and services
Task / Use Case	-	Mechanisms
Subtasks involved	T3.2 – T3.7	
Lead contractor	Infineon Technologies AG Dr. Werner Weber, mailto:werner.weber@infineon.com	
Deliverable responsible	Technical University Braunschweig Adam Kostrzewa, kostrzewa@ida.ing.tu-bs.de	
Version number	v1.0	
Date	30/07/2016	
Status	Final	
Dissemination level		

Copyright: EMC² Project Consortium, 2016

Authors

Partici-pant no.	Part. short name	Author name	Chapter(s)
01A	IFAG	Harald Zweck	Chapter 2
01V	TUBS	Adam Kostrzewa	Publishable Executive Summary, Introduction, Chapter 6, Conclusions
04B	Freescale	Michal Princ	Chapter 7
07A	CEA	Paul Dubrulle	Chapter 4
07J	TCS	Aurelien Berhault	Chapter 3
17C	NXPNL	Gerardo Daalderop/CEA	Chapter 5

Please note: only the contributions by task leaders are listed above. However, we would like to thank all other WP3 partners for their valuable inputs, especially in the form of requirements appended to this deliverable.

Document History

Version	Date	Author name	Reason
0.1	01/06/2016	Adam Kostrzewa	Initial Template
0.2	12/07/2016	Adam Kostrzewa	Partial integration of task inputs
0.3	20/07/2016	Adam Kostrzewa	Executive summary, intro and conclusions
0.4	29/07/2016	Adam Kostrzewa	Final integration of inputs from tasks
1.0	30/07/2016	Alfred Hoess	Final editing and formatting, deliverable submission

Publishable Executive Summary

The broad objective of WP3 is to enhance mechanisms and architectures for run-time environments with dynamic control. This is done in order to support mixed-critical systems, security techniques, safety and real-time properties. The document presents the results of the work done in WP3 until M28. The first phase of work in WP3 concluded in the D3.3 and concentrated on the analysis of opportunities for dynamic platform control and the development of mechanisms for run-time adaption comprising monitoring and flow control, re-configuration, and scheduling.

The main input to this document originates from deliverable D3.3 and concentrates on: (1) summary and further refinement of solutions/platforms/architectures proposed in D3.3; (2) implementation of solutions/platforms/architectures proposed in the D3.3; (3) analytic and experimental evaluation of the proposed solutions against the project requirements and goals of WP3.

In D3.4, reported results originate from Tasks T3.2 - T3.7 where evaluation and implementation was performed jointly with Living Labs (WP7-12) targeting high-level and technical-level requirements gathered in the scope of the work in T3.1 and WP3 (see deliverable D3.1). Therefore, the content of this document is established relying on: (1) inputs from T3.1 and WP1 (requirements), (2) inputs from Living Labs (WP7-WP12), and (3) inputs from all other tasks belonging to WP3 (T3.2-T3.7) relying on experience from previous work and projects.

In the following part of D3.4, you may find the summary of the inputs from users grouped and structured thematically for easier comprehension. The inputs for each task (T3.2-T3.7) are constructed as follows: (1) the introduction with a brief summary of goals and assumptions of the task, (2) overview of the introduced mechanisms, solutions and methods from D3.3, (3) description of the actions conducted by partners for implementation and evaluation of their work in D3.4, (4) planned deployment and usage by partners, including transfer to another technical WPs and Living Labs, (5) most important (for the particular) requirements of the WP3 task which were targeted during implementation and evaluation of the mechanisms.

The detailed description of work done by partners is attached in the appendixes of this document. The input for each of the partners consists of two sections: the short summary of the achievements and work; technical annex containing all inputs in form of technical reports and publications.

The summaries contain the following information: (1) an overview of the solutions / platforms / architectures developed by partner in the scope of the WP3, (2) description of WP3 partners' contributions to D3.4 e.g implementations of mechanisms and their evaluation (3) information about the toolchain, deployment environment and cooperation with other partners/WPs/LLs and finally the high- and low-level requirements targeted and covered by the work of a particular partner.

In WP3, an iterative work approach is assumed. Therefore, the presented results constitute an initial input for the next deliverable D3.6. The initial implementations should be later supported with results from reference platforms and fully adjusted to provide support for the considered hardware architectures. Therefore, inputs presented in D3.4 will be improved, aligned and extended in D3.6 (PM36).

The gathered inputs and presented outcomes constitute the basis for fulfilling the objectives of WP3 and shall support Living Labs in identification of the technologies and relevant partners fulfilling their needs.

Table of contents

1. Introduction

WP3 investigates mechanisms and architectures for run-time environments (RTE) that are able to support mixed-critical systems, security techniques, safety and real-time properties. Technologies include mechanisms for virtualization, hypervision and monitoring, which need to be adapted to handle increasing application system dynamics with no loss in safety and minimal loss in performance.

1.1 Objective and scope of the document

WP3 is split in seven thematic tasks which can be considered as technology clusters grouping the developed mechanisms. Each of the tasks contains two subtasks. The first one addresses basic cornerstones to the topic entailed in the task. The second one enhances the investigation further and involves an implementation development, and the evaluation. This document concentrates on the achievements done during the work on the second subtask until M28.

The main goals were:

- Architectural overview and further refinement of the solutions and methods proposed in D3.3;
- Implementation of the developed mechanisms for safe and dynamic run-time adaption (Design & Theory);
- Evaluation of the proposed solutions addressing the goals of the WP3 and requirements resulting from T3.1 and WP1.

In D3.4, reported results originate from Tasks T3.2 - T3.7 where evaluation and implementation were done jointly with Living Labs (WP7-12) targeting high-level and technical-level requirements gathered in the scope of the work in T3.1 and WP3 (see deliverable D3.1). Therefore, the content of this document is established relying on: (1) inputs from T3.1 and WP1 (requirements), (2) inputs from Living Labs (WP7-WP12), and (3) inputs from all other tasks belonging to WP3 (T3.2-T3.7) relying on experience from previous work and projects. All the mechanisms are implemented and evaluated ensuring consistency with the expectations of the Living Labs covering these domains. This is confirmed with the technology transfer charts and tables that present the cooperation between the WP3 and LLs. Consequently, most of the mechanisms propose incremental improvements of existing state-of-the-art solutions targeting LLs, whenever possible.

In WP3, an iterative work approach is assumed, as shown in Figure 1. Therefore, the presented results constitute an initial input for the next deliverable D3.6. The initial implementations should be later supported with results from reference platforms and fully adjusted to provide support for the considered hardware architectures (including WP4). Therefore, inputs presented in D3.4 will be improved, aligned and extended in D3.6 (PM36).

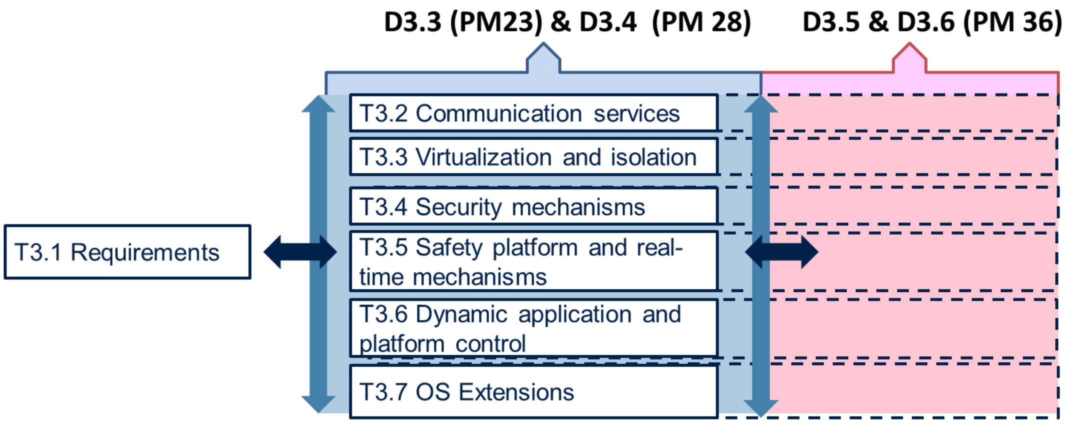

D3.3 (PM23) & D3.4 (PM 28)
D3.5 & D3.6 (PM 36)
T3.1 Requirements
T3.2 Communication services
T3.3 Virtualization and isolation
T3.4 Security mechanisms
T3.5 Safety platform and real-time mechanisms
T3.6 Dynamic application and platform control
T3.7 OS Extensions

Figure 1: Structure of the technical inputs in WP3

The gathered inputs and presented outcomes constitute the basis for fulfilling the objectives of WP3 and shall support Living Labs in identification of the technologies and relevant partners fulfilling their needs.

1.1.1 Technology transfer to LLs and other WPs

A strong link to the Living Labs is one of the most important objectives of WP3. The links that have been established are depicted in Figure 2. The evaluation of the evolvement of WP3 in LLs is presented in two tables showing work package coverage as a ratio of: WP3 partners' participation in a selected WP to the overall number of WP participants, and WP3 partners' effort in a selected WP to the overall WP effort. This high coverage of the LLs in WP3 resulted in the broad spectrum of the developed mechanisms (more than 30) targeting many different domains (more than 10 use-cases). In order to improve the transfer of the technologies and information between tasks, partners and living labs, there was a cooperation matrix established.

Coverage	Description	WP1	LL1 (WP7)	LL2 (WP8)	LL3 (WP9)	LL4 (WP10)	LL5 (WP11)	LL6 (WP12)
Summary of WP3 Participation Coverage	WP3 Partners in	23	16	8	6	3	4	5
	All WP Partners	36	37	12	10	9	12	23
	Ratio	64%	43%	67%	60%	33%	33%	22%
Summary of WP3 Effort Coverage	WP3 Partners' Efforts in	217	596.1	253	123.1	187	104	156
	Total WP Effort	379.2	1731.6	334	479.1	311	610	868
	Ratio	57%	34%	76%	26%	60%	17%	18%

Due to its size, the complete table is only presented in Appendix **Fehler! Verweisquelle konnte nicht gefunden werden.**. The table contains the list of connections between tasks / mechanisms and LLs along with the evaluation of progress (in %) and the scope of the implementation. The following four scopes were considered:

- Scope 1: Complete implementation into use cases in the project
- Scope 2: Partial implementation into use case demonstrator(s)
- Scope 3: Will be transferred to LL (WP6-12) but not implemented into demonstrator
- Scope 4: Topic for future applications; technology transfer subsequent to EMC^2

The data used for the evaluation was taken from the EMC^2 Effort Planning document [2].

Additionally, the cooperation with WP1 and WP4 followed, resulting in face to face meetings and workshops. Cooperation with WP1 concentrates on the development of the general methodology for the development of the embedded systems. Cooperation with WP4 concentrates on evaluation of proposed mechanisms with respect to the architectures considered and developed in WP4.

1.1.2 Iterative Approach

The development of the mechanisms follows an iterative approach that allows extension and refinement of the assumptions and scopes proposed in the previous deliverable D3.3 or even introducing new solutions basing on the implementation-based experimental evaluation. This is a two-staged process:

- Partners will apply the results of this initial mechanism development to implement their solutions in the existing environments and platforms.
- Key functions and assumptions will then be validated using prototyping and/or formal analysis.

The process allows smooth incorporation of the results in the living labs and other transfer to the industrial applications. The evaluation of results from D3.3 is done in this document and the further refinement of inputs will follow in D3.5 and D3.6 (M36).

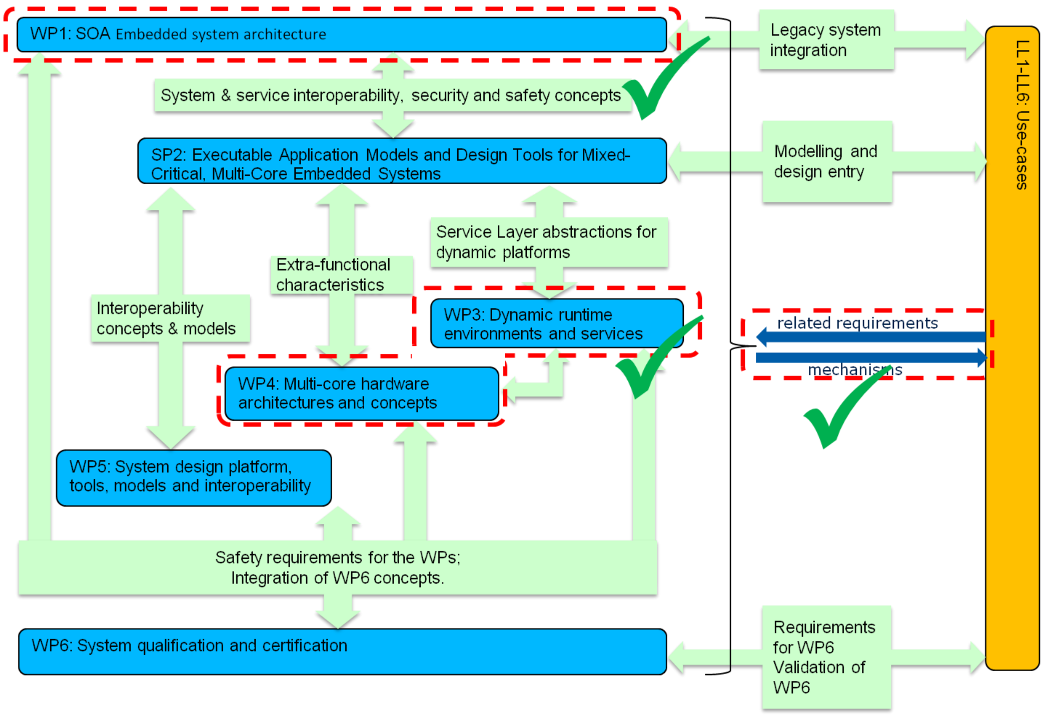

WP1: SOA Embedded system architecture
System & service interoperability, security and safety concepts
SP2: Executable Application Models and Design Tools for Mixed-Critical, Multi-Core Embedded Systems
Service Layer abstractions for dynamic platforms
Extra-functional characteristics
Interoperability concepts & models
WP3: Dynamic runtime environments and services
WP4: Multi-core hardware architectures and concepts
WP5: System design platform, tools, models and interoperability
Safety requirements for the WPs; Integration of WP6 concepts.
WP6: System qualification and certification
Legacy system integration
Modelling and design entry
related requirements
mechanisms
Requirements for WP6 Validation of WP6
LL1-LL6: Use-cases

Figure 2: Connections between WP3 and other WPs in EMC2

1.2 Structure of the deliverable report

The presented report is structured similarly to the D3.3, as it extends the reported earlier mechanisms and solutions with results from implementation and evaluation. The main deliverable's document is divided into the three main parts: overview of the work done in each of the tasks (technology clusters) presented in Chapters 2-7, general conclusions and summary presented in Chapter 9 and finally the section including all appendixes.

The appendix section starts with the cockpit charts (Appendix **Fehler! Verweisquelle konnte nicht gefunden werden.**) presenting in a graphical manner the progress of technology transfer to LLs on the example of selected mechanisms / solutions from each of the tasks due to the high number of mechanisms (more than 30) and extensive converge of LLs. The full cooperation matrix presenting all registered mechanisms and progress of their transfer to particular use cases in LLs (WP7-W12) follows in Appendix **Fehler! Verweisquelle konnte nicht gefunden werden.**.

Appendixes **Fehler! Verweisquelle konnte nicht gefunden werden. - Fehler! Verweisquelle konnte nicht gefunden werden.** present the detailed information about inputs from all partners divided into: Summary of work done by the partner partners (for details see Section 1.2.2) as well as the technical annexes enclosing results delivered by individual partners in form of technical reports, publications and reports.

1.2.1 Structure of the inputs from Tasks

The subsequent Chapters 2 to 7 correspond to inputs from each task (T3.2–T3.7) and adopt an uniform and similar structure. They are divided into five main sections: introduction, overview of used platforms, and description of the proposed mechanisms, planned deployment and usage by partners and evaluation of mechanisms against EMC2 goals. In the following we present the description of sections content.

Introduction. This section summarizes the goals of the particular task in general as well as presents the necessary actions which were due to 28PM.

Overview of introduced mechanisms
The section presents the overview of the most prominent technologies which were developed in the particular task until PM28. Due to the high number of proposed mechanisms (more than 30 were registered) the description is structured depending on the task: (1) T3.2 based on the communication layer, (2) target implementation domain / use-case as in T3.3-T3.6, (3) depending on the mechanisms origin (task) in T3.7 which encompasses the OS support functions for mechanisms developed in other tasks in WP3. The descriptions include: the problem which mechanism is targeting (and/or platform), as well as the principles of work.

Implementation, Prototyping
This section includes the description of the **most significant** results from implementation and prototyping. Inputs for this document are obtained from translation of the mechanisms (results of D3.3) into the RTE functions and protocols. This section should consider the most prominent results (per cluster): (1) Short description of the design of RTE functions and protocols – (per cluster), (2) short description of the implementation/prototyping setup, (3) if possible (i.e. data are already available) the experimental validation – quantitative summary of the obtained results, for instance average performance or the worst-case effectiveness.

Planned deployment and usage by partners
The input to this section encompasses the information about the deployment of most prominent mechanisms developed in the task and collaboration with Living Labs. It includes the description of demonstrators developed until M28, summary of the cooperation with other partners and tasks within WP3 as well as Living Labs (WP7-WP12) and other Technical Work Packages of the EMC2 project (WP1-WP6). Finally it presents an overview of used toolchains and platforms used by partners looking for similarities and possibilities of the technology transfer between different domains.

Evaluation of mechanisms against EMC2 goals
The summary is concluded with a comparison of the activities and work conducted in the task's scope with the goals of the EMC2 project with particular focus on WP3. This is done to enforce the coherence of inputs and work, which is especially important in case of WP3 (36 partners).

The comparison covers the High Level Requirements (see deliverable D3.1) as well as the Low Level (Technical) Requirements. The document concentrates on requirements which were especially important for partners in their work on the particular technology cluster. The code of requirements references the list of requirements gathered in the scope of the work in T3.1 and WP3 and presented in deliverable report D3.1. The requirements were established relying on: (1) inputs from WP1, (2) inputs from Living Labs (WP7-WP12), and (3) inputs from all other tasks belonging to WP3 (T3.2-T3.7) relying on experience from previous work and projects.

1.2.2 Structure of the inputs from Partners

The descriptions of work provided by individual partners can be found in Appendixes **Fehler! Verweisquelle konnte nicht gefunden werden. - Fehler! Verweisquelle konnte nicht gefunden werden.** and constitute the main input to summaries presenting the overview of tasks. Their main purpose is to present a brief overview of activities conducted by each EMC2 participant until M28 in the second subtask of all tasks. They should also describe the content of the technical annex allowing easy access to the included technical reports and publications. In the following we present the description of the content of particular document's sections.

Overview of mechanism
This section presents the mechanism (or list of mechanisms) developed by partners in the scope of the work in the particular task. Input (per mechanism) is structured as follows

- Name of the mechanism / Title of the contribution;
- Improvements over the state-of-the-art solutions presented in the overview section;

- Preferably links to the publications and technical reports included in the Technical Annex.

RTE functions and protocols derived from mechanisms
This section presents the implementation and, whenever already available, evaluation of mechanisms developed by partners in the scope of the work in the particular task. The main focus of D3.4 is on the translations of the mechanisms (results of D3.3) into the RTE functions and protocols. Input (per mechanism) is structured as follows:

- Short description of the design of RTE functions and protocols
- Short description of the implementation/prototyping
- Experimental Validation – if already available summary of the obtained results, for instance average performance or worst-case effectiveness

Note, that this section includes only an overview with summary/abstract of contributions. All details are provided in the technical annexes (as in case of D3.3) i.e. technical reports, experimental results, publications etc.

Planned deployment and usage by partners
The input to this section is divided into a threefold contribution: (1) information about the deployment including, whenever possible, the overview of the platforms and toolchains, (2) information about available or foreseen demonstrators, (3) summary of the cooperation with other partners and tasks within WP3 as well as Living Labs (WP7-WP12) and other Technical Work Packages of the EMC^2 project (WP1-WP6).

Evaluation of mechanisms against EMC^2 goals
The summary is concluded with an evaluation of the proposed mechanism/approach with the goals of the EMC2 project with the particular focus on WP3.

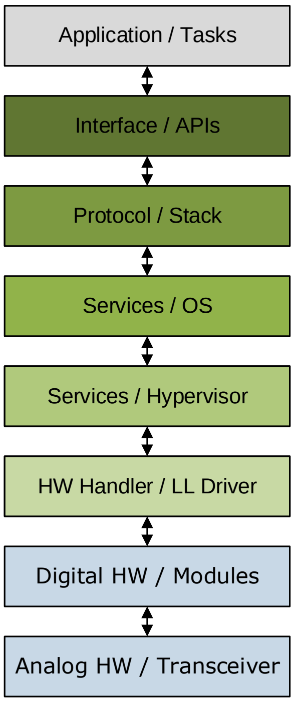
The evaluation covers the High Level Requirements (see deliverable D3.1) as well as the Low Level (Technical) Requirements. The document references the list of requirements gathered in the scope of the work in T3.1 and WP3 and presented in deliverable report D3.1. The requirements were established relying on: (1) inputs from WP1, (2) inputs from Living Labs (WP7-WP12), and (3) inputs from all other tasks belonging to WP3 (T3.2-T3.7) relying on experience from previous work and projects.

Finally the technical annex contains all materials and inputs referenced by partners in the summaries as their work and input to the WP3.

2. T3.2 Communication services

2.1 Introduction

In deliverable D3.4, the mechanisms developed in D3.3 were implemented. The solutions created in T3.2 cover a range of challenges, starting from the point of view of the communication stack, moving to the level of a typical Hypervisor functionality, further up to the Middleware and finally to the communication APIs. For more information see the diagram below.



Independently of the stack level, the following mechanisms were implemented or applied:

- Synchronization of tasks and communication;
- Secure exchange of mixed-criticality data;
- Mapping of security constraints to available computing and communication units;
- Measures to decouple low level structures from the application level;
- Introduction of rule sets for decoupling low level and application level;
- Systematic analysis and rules for deployment of low level features;
- Introduction of determinism to standard messaging API(s);
- Harmonization of wireless protocols;
- Implementation of communication protection rules into HW.

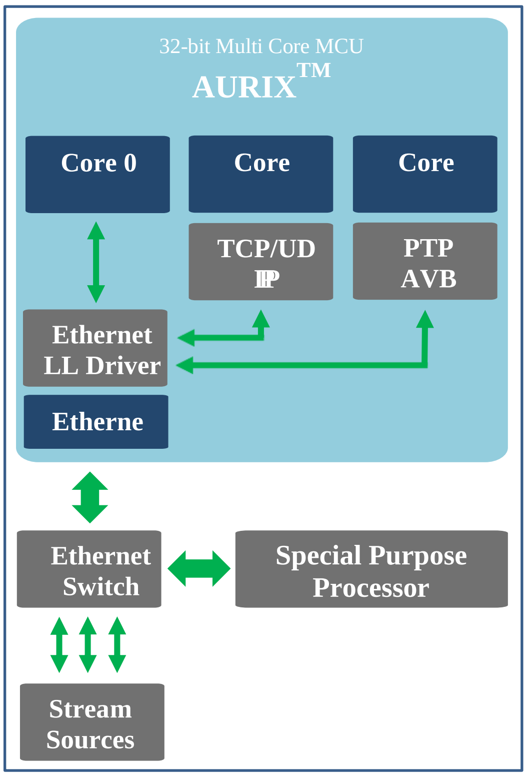
2.2 Implementation, Prototyping

2.2.1 Infineon

Infineon used its multi-core controller to demonstrate the implementation of protection rules for communication units into the HW of a controller. The following solutions were integrated:

- Isolation at application level;
- Separation of message flows;
- Isolation at stack level to enable deterministic behavior.

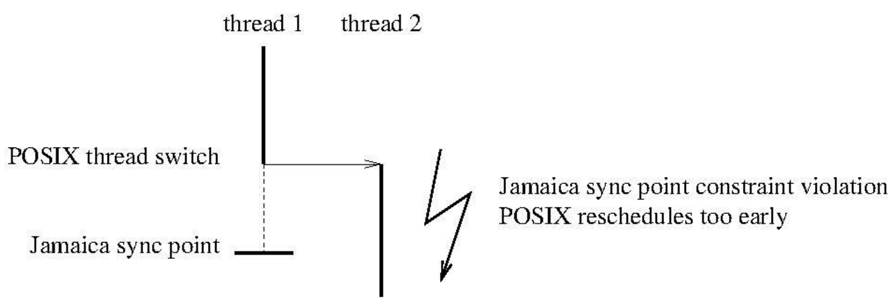
As shown below the incoming and outgoing data streams are separated according to defined rules and forwarded to the specific processing unit.

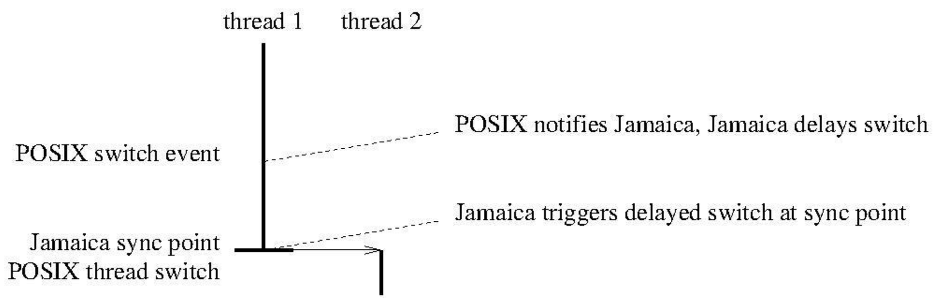


The necessary separation operations are executed in HW, which gives them robustness and protection against unintended changes like SW failures. The solution was already demonstrated at its preliminary version status at the EMC^2 meeting in 2015.

2.2.2 SYSGO / AICAS

SYSGO implemented a method to synchronize two real-time capable SW systems. One is SYSGO's PikeOS operating system, the other AICAS Java scheduler called "Jamaica". In the beginning the two entities could not synchronize at a common point in time. The work in T3.2 brought a solution for this issue by extending the system to use a common scheduling table. See below.


thread 1
thread 2
POSIX thread switch
Jamaica sync point
Jamaica sync point constraint violation
POSIX reschedules too early


thread 1
thread 2
POSIX switch event
POSIX notifies Jamaica, Jamaica delays switch
Jamaica triggers delayed switch at sync point
Jamaica sync point
POSIX thread switch

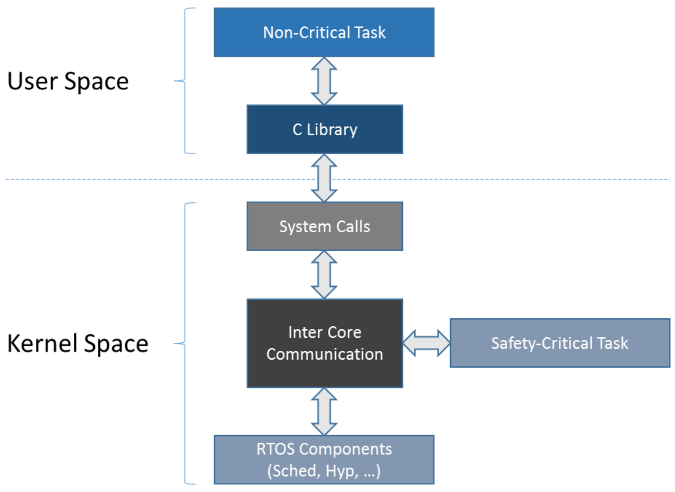
The technology was offered to several Living Labs and is expected to be integrated into several demonstrators.

References

Ref. to HL Requirements	Requirement ID	Short Description
HL-WP03-006	Predictable Resource Sharing	The Java-RT environment for supporting CPU time resource sharing is implemented in JamaicaVM on top of PikeOS's POSIX environment. This task will improve the support for predictable CPU time sharing by giving better support to implement the Java-RT standard on top of PikeOS.

2.2.3 CSOFT

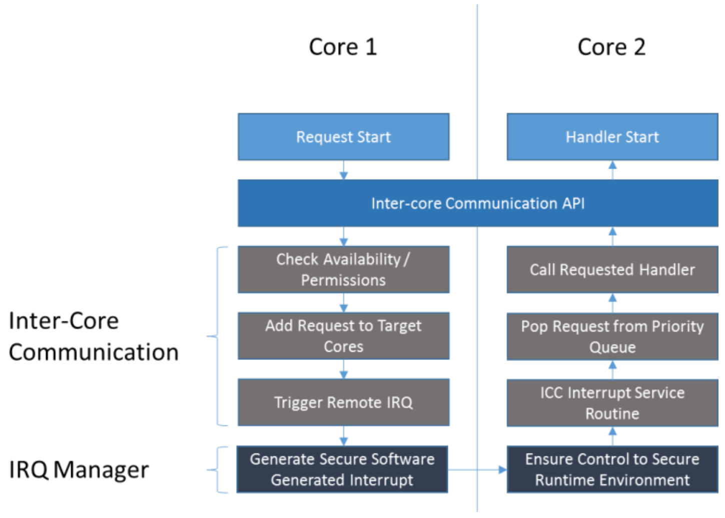
CSOFT develops a communication infrastructure for mixed-criticality data exchange. The communication components can reside at distinct levels of safety-critical requirements. A secure exchange of data is supported by this implementation.



Some of the mechanisms which are used are:

- Hypervisor services;
- Scheduler for synchronization between cores;
- Inter-process communication.

An example is shown below.


Core 1
Core 2
Request Start
Handler Start
Inter-core Communication API
Check Availability / Permissions
Call Requested Handler
Add Request to Target Cores
Pop Request from Priority Queue
Trigger Remote IRQ
ICC Interrupt Service Routine
Generate Secure Software Generated Interrupt
Ensure Control to Secure Runtime Environment
Inter-Core Communication
IRQ Manager

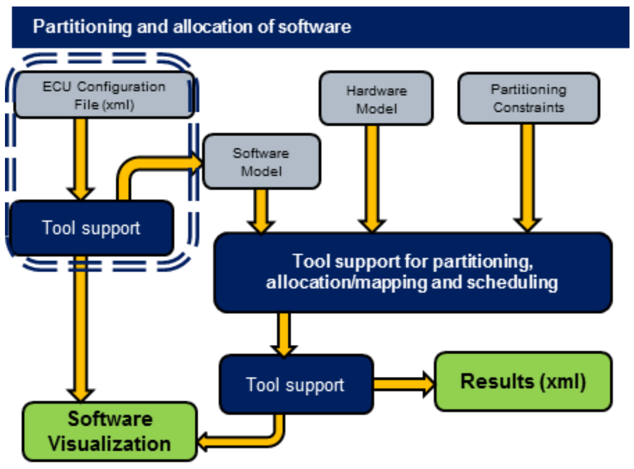
The system will be used in the T7.5 demonstrator of Living Labs (see D7.12) and in WP3 together with the partners CISTER and INESC-ID.

References

Ref. to HL Requirements	Requirement ID	Short Description
HL-REQ-WP07-044		Provides the necessary infrastructure for secure data communication

2.2.4 **VOLVO**

VOLVO developed a method, including the related tool, to map functionality to multicore control units. Based on a HW and SW model plus configuration information, an allocation is proposed by the tool, which meets a set of given constraints. See the figure below.


Partitioning and allocation of software
ECU Configuration File (xml)
Hardware Model
Partitioning Constraints
Software Model
Tool support
Tool support for partitioning, allocation/mapping and scheduling
Tool support
Results (xml)
Software Visualization

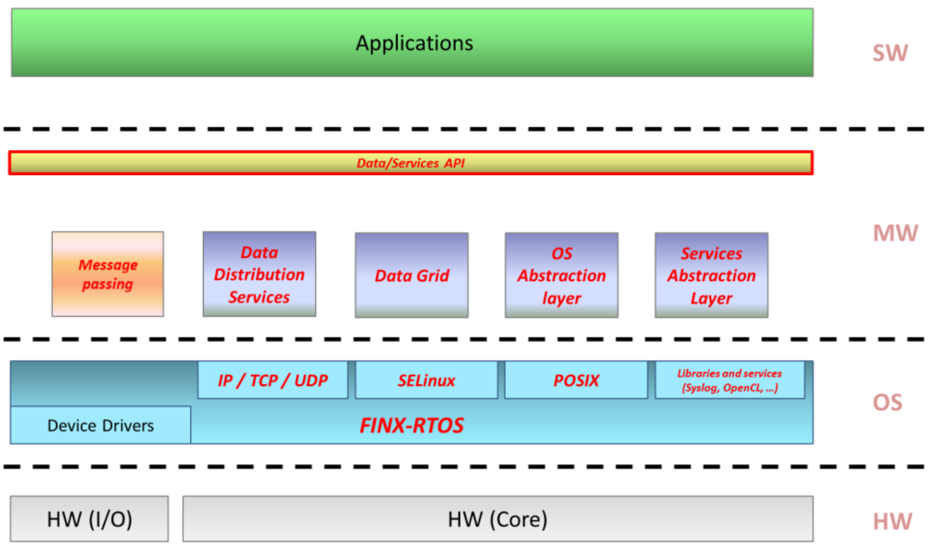
The related constraint information which is predefined is e.g. bandwidth, variance and core allocation. It separates components of different safety integrity levels. The implementation is based on AutoSAR RTE and a three core controller system. The work is a cooperation between DTU and Volvo. The tool is part of the Volvo use case in WP7 T7.6.

References

Ref. to HL Requirements	Requirement ID	Short Description
HL-WP07-025		Software partitioning and allocation: Methods and tools for partitioning and allocation of software across the multicore ECUs at the E/E architecture level and across the processor cores within a particular ECU in order to maximize performance and resource utilization shall be developed.
HL-WP07-27		Mixed criticality support: Separation, i.e. freedom from interferences, among applications with different criticality levels (as defined in ISO 26262) shall be ensured in one and the same ECU.

2.2.5 **MBDA**

MBDA developed a middleware, which provides separation of applications and fencing them into their own domains. The middleware supports services for communication, services and resources.



The middleware decouples application SW from physical implementations and provides a standard interface. So it enables the reuse of SW components. It solves critical relations to HW functionality by offering a data/services API and hiding the low level interfaces.

2.2.6 **ISEP**

ISEP worked on methods and implementations, for which they use information from the underlying HW and regulations on how to access it. The result improves the analysis and makes more options available to schedule mixed-criticality behavior.

The target is a unified approach to improve management of contention avoidance of resource utilization and separation of mixed-criticality resources. It defines rules like dropping low criticality tasks and to use the resources for high criticality tasks. It re-uses resources which would go else into an idle status. An analysis testbed will be created to enable comparison of the new developed setup with existing systems. The results will be shared with CSoft in WP7.

2.2.7 **Freescale**

FSL works on an improved Remote Processor Messaging (RPM) concept. The result will be available as open source SW in the Open Asymmetric Multi Processing Framework (OpenAMP).

RPM is a standard messaging system for inter-processor communication, available in upstream Linux OS. RPM processes data in the interrupt context, which creates undefined jitter, due to task preemption, possible random execution time and random run time lengths. The new system is aware of the RTOS requirements, avoids processing of code in interrupt context.
It provides the following improvements:

- No data processing in the interrupt context;
- Blocking receive API;
- Zero-copy receive API;
- Receive with timeout provided by RTOS;
- Compatibility with Linux upstream kept.

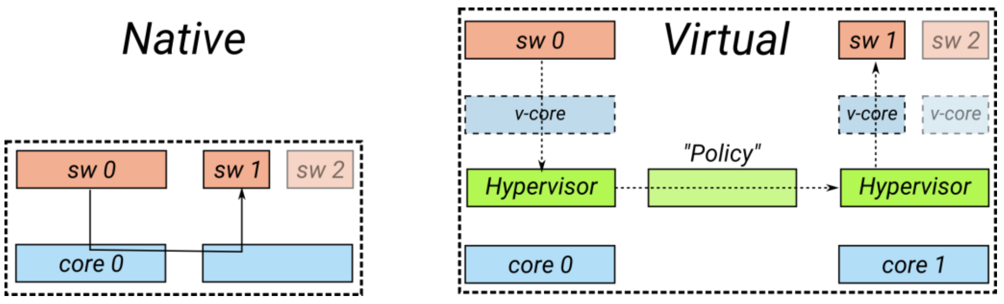
For better matching the requirements of embedded systems, the memory requirement was decreased, and a better Linux RPM module implemented. The new RPM is available as open source project. It will be provided in form of a RPM library. There will be cooperation with partners in WP10.

References

Reference to HL Requirements	Requirement ID	Short Description
HL-REQ-WP1-003 HL-REQ-WP1-005 HL-REQ-WP03-007 HL-REQ-WP03-008 HL-REQ-WP4-001 HL-REQ-WP07-004 HL-REQ-WP07-032 HL-REQ-WP07-034 HL-REQ-WP07-058	04B-001	The concept of portable multi-core communication interface.
	04B-002	Versioning of the multi-core communication subsystem.
	04B-004	Performance & footprint of the multi-core communication subsystem.
	04B-005	Abstraction of the multi-core communication subsystem.

2.2.8 **SICS**

SICS develops a secure and performing communication service provided by a "thin" hypervisor. The communication service is separated into several layers to minimize the impact on the computing system.



SICS is experimenting with an implementation that divides the communication service into multiple layers in order to minimize its impact on the trusted computing base.
The layer's setup summary:

- Hypervisor layer provides minimal mechanisms for shared memory and signaling; These mechanisms are protocol-independent;
- The OS implements the required protocols as kernel drivers;
- The OS provides communication services to its applications.

The system is implemented on an ARM 32-bit system, and uses some virtualization technologies from T3.3 and security mechanisms from T3.4. Additionally, there is cooperation with partners in WP1 and WP3.

References

Reference to HL Requirements	Requirement ID	Short Description
HL-REQ-WP03-003	16E-001	Secure communication across partitions
HL-REQ-WP03-003	16E-002	Runtime monitoring of guest configuration
HL-REQ-WP03-006	07J-005	ARM based platform

2.2.9 **AMBAR**

AMBAR implements a multi-connectivity gateway and sensor nodes combining several wireless technologies, which are Wi-Fi, Bluetooth LE, ZigBee, 6LowPan, 3G.

The implementations require a solution for multi-protocol integration, due to the different wireless technologies, low power consumption, due to battery operation, and low resource allocation, due to the target systems being embedded devices.

In the gateway the following blocks are available: Wi-Fi / Bluetooth, ZigBee / 6LowPan, 3G. Both, the required SW stacks and according HW elements were integrated. The challenge in this task was to make the different elements work together flawlessly. The gateway functions were tested regarding interoperability while using simultaneous running wireless links. The ZigBee nodes are optimized and prepared for T10.3 and T11.3 scenarios. The integration of the other protocols is still ongoing.

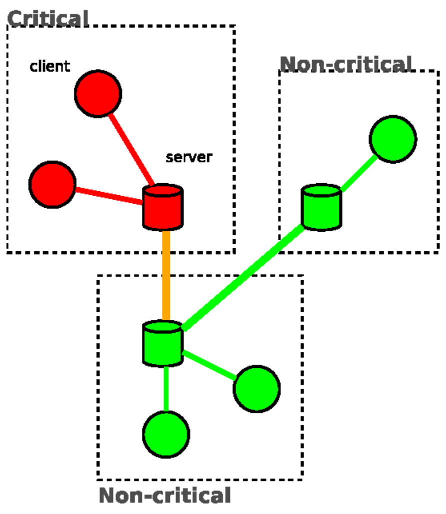
The gateway will be provided for test purposes to T10.3. The nodes will be provided to T10.2 and T11.3 and according technical support to T11.2.

References

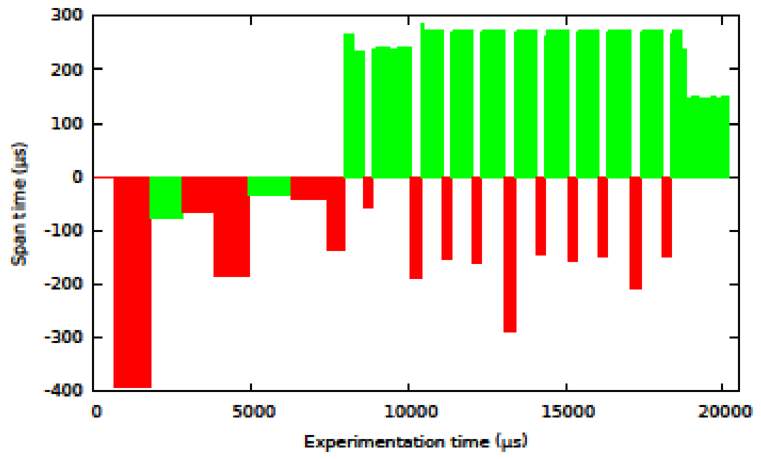
Reference to HL Requirements	Requirement ID	Short Description
HL-REQ-WP03-002	15R-001	Provide different fully functional well identified communication interfaces for the supported wireless channels (Wi-Fi,BLE,ZigBee,6LowPan,3G)
HL-REQ-WP03-008	15R-002	Provide the integration of the HW/SW elements to implement the wireless communication services of the Gateway.
HL-REQ-WP03-010	15R-003	Optimize both Gateway and nodes in relation to their power consumption devoted to provide the communication services.

2.2.10 **CEA**

CEA carries research in communication technologies using shared memory structures. The focus is on the organization of shared memory, on creating a data consistence protocol and a message passing interface. The topology is based on the elements servers and clients. These virtual elements are mapped using a defined process to physical elements. Additionally, critical and non-critical roles are defined and applied.



The research project focuses on criticality and worst-case execution times. The implementation of a test system is based on a data consistency model, which uses separate time slots, which can be opened and closed for accesses, so separating data accesses. The slot operation is non-blocking, and it can be limited by a deadline. The system was tested with several patterns. Results, see below.



For each call, a bar represents the difference between the deadline and the actual primitive return time. Positive values mean the deadline is met, negative values mean the deadline is missed. Green bars indicate that the primitive succeeded (all protocol messages have been sent / received and write access is granted), while the red bars mean the protocol did not perform to the end (and write access is not granted).

Summarized, in this system, in the context of mixed criticality, shared data can be accessed by critical and non-critical clients.

2.3 Planned deployment and usage by partners

Table of deployment and work package partners

WP03 T3.2 Partner	Cooperation Partner	Cooperation WP(s)	Comment
AMBAR		WP10 T10.2, T10.3; WP11 T11.2, T11.3;	Gateway design will be provided to WP10, WP11
CSOFT	CISTER, INESC-ID	WP03 T3.2, T3.3, T3.5, T3.7; WP7 T7.5	
Freescale		WP03 T3.7; WP10	
Infineon			Infineon will provide its own demonstrator
ISEP (CISTER)	CSOFT	WP7	
SICS	Ericsson	WP3 T3.3, T3.4; WP1, WP3	
SYSGO sro	AICAS		
VOLVO	DTU	WP7 T7.6	

3. T3.3 Virtualization and isolation

3.1 Introduction

This task studies the virtualization and isolation mechanisms to ensure real-time and safety properties. In addition to ensuring that the virtualized systems are strictly isolated and have a fair access to the resources they need for their correct execution, these mechanisms shall provide additional key features, such as real-time constraints enforcement, and safe mixed-criticality domains virtualization.

Several prototypes of the RTE/hypervisor implement a subset of the functionalities specified in T3.4, T3.5, T3.6 and T3.7. A second objective is an investigation of virtualization of shared SoC resources under real-time constraints.

3.2 Proposed mechanisms

The mechanisms proposed focus on several aspects of virtualization: security for isolation, security for communication, management of communications to guarantee real-time constraints in mixed critical systems, low power operation, and respect of safety standards such as DO-178, Common Criteria EAL4+. The hardware components are also investigated and mechanisms are proposed to establish restricted access to hardware resources. Hence, applications could be executed in a trusted environment.

Regarding the safe and secure parallel execution of VMs/applications on multicore processors, several hypervisor mechanisms are proposed by HIB, THALES, CEA and CSOFT partners. CSOFT uses hardware virtualization capabilities available on the target hardware to implement a software container allowing the concurrent execution of a second runtime environment (RTE) with fine grain access privileges. As a critical component to the correct partitioning of safety-critical and non-critical software, it requires that all internal functions are isolated from external access. For that reason, the hypervisor has two main inner components. The API, i.e. a C library to be used by real time tasks from the RTOS runtime environment, and a set of small patches to the non-critical Linux kernel. THALES, TAF partner will use SYSGO PikeOS capabilities for avionics computers. More precisely, they have studied the impact of potential interferences between different cores at the cache level as well as at interconnect and memory levels.

The potential impact of such interferences on performance will make it possible to define deployment policies on multicore processors to ensure that the performance of applications will not be impacted by interferences due to shared resources. In the frame of Fin.X (MBDA RT Linux distribution) certification and system certifiability studies, MBDA team faces Common Criteria EAL4+ and DO-178B requirement. Isolation is often put in place to prevent unwanted interference and KVM (Kernel Virtual Machine) is the adopted solution CC compliant and, using Qemu as user-space hosted hypervisor and emulator a full-virtualization is achieved. As a valid alternative, the use of a Linux container (LXC) was investigated, something in the middle between a powerful chroot and a full-fledged virtual machine. Moreover, CEA partner uses the S-MMU to insure a strong isolation between the Virtual Machines and the Hypervisor itself. Memory isolation can be warranted if an only if, the processor mode in which the hypervisor is executing does not allow it to access other memory areas than its own. As a result, the memory isolation requires only the secure module to be trusted.

Several SW control mechanisms for virtualized communication channels will be proposed in this task. In order to improve inter-core communications abstraction, Freescale is going to develop a concept of cross-core Remote Procedure Call that could be used in embedded applications as a base enablement for virtualization of services provided and consumed by different cores in multi-core and multi-processor applications. The Embedded Remote Procedure Call (eRPC) library has been designed. It implements a transparent function call interface to remote services (running on a different core of a multicore platform). TUBS introduces a Mechanism: Flexible TDM-Based Resource Management in On-Chip Networks.

TUBS proposes an alternative approach for providing efficient service guarantees in NoCs for mixed-critical real-time systems. It exploits the concept of virtualization of shared resources under the presence of real-time constraints for controlling accesses from processing nodes to the NoC. TUBS introduces a global access layer (the network hypervisor), working transparently for running applications and deciding about the order in which transmissions may proceed through the NoC

The scheduling for virtualised systems is studied in order to support several OS services. ISEP's main task in T3.3 is the research of real-time scheduling mechanism that contributed to the development of a real-time scheduler and resource sharing protocol for a safety critical operating system targeting mixed-critical automotive applications. The mechanisms developed focused on hierarchical scheduling. Hierarchical scheduling encapsulates different applications or tasks in different partitions (or servers), each with a given execution budget allowed for execution on the processor. Hierarchical scheduling is a good candidate for mixed-criticality systems as it ensures time isolation between applications mapped in different partitions. SICS is investigating certain effects of virtualization (or similar technologies) in multi-core embedded systems. SICS has implemented an experimental hypervisor / separation kernel for multi-core embedded systems which takes advantage of certain virtualization capabilities provided by the CPU. These capabilities allow the hypervisor to execute different operating systems on different cores (as both single-core or multi-core) in isolation while handling multi-core operations such as interrupt routing in a transparent manner. HI Iberia has continued the work on porting Android Applications to an EMC2 environment for the management of criticality and real-time parameters. The results presented are preliminary and expected to be concluded during Y3. A technical annex is available detailing the results.

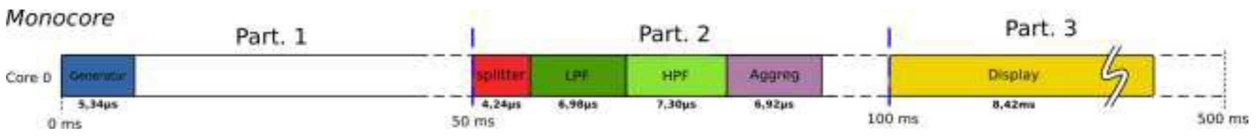
The power management mechanisms proposed for mixed-critical systems under the presence of real-time constraints will take benefits of available temporal and space isolation methods on RTOS in order to ease the certification process. According to the VMs behaviour at run-time, the energy consumption will not be always constant for the whole system. In the scope of EMC², the mechanisms will combine the Dynamic Voltage and Frequency Scaling (DVFS) technology available in most of modern processors and the temporal and space isolation. TCS and SYSGO has studied, evaluated and implemented the mechanism on the Freescale i.MX6 quad-core ARM board. One of the particularities of this chip is the common PLL (ARM_PLL) clocking the ARM cores and the common voltage (VDD_ARM and VDD_SOC). This implies that all cores have to share the same voltage/frequency set at any time. Under the presence of real-time constraints the proposed mechanism should manage this feature.

3.3 Implementation, Prototyping

The tool METrICS (see D3.4_technical_annex_THALES_07H.pdf):

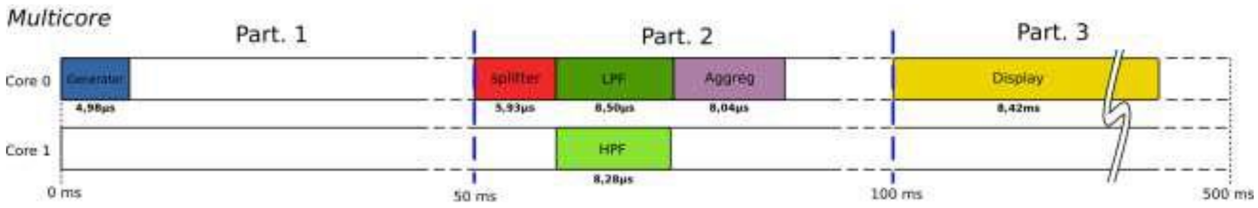
THALES, TAF and SYSGO PikeOS have collaborated to develop a tool in order measure performance. The tool is called METrICS (Measure Environment for TIme-Critical Systems). The implementation of METrICS has been developed for PikeOS 4.0, and the target hardware is the Freescale/NXP T2080. It is, however, easily portable on other targets of the QorIQ family. It has been executed with a synthetic APEX application.

In the first case, all the applications are mapped on only one core. This is the baseline, equivalent to a legacy application running on a single-core processor, with a static sequential scheduling.



This figure shows the average execution time of each process, to scale, illustrating the sequential scheduling.

A second deployment uses two cores, with parallel execution of the filter computation processes. The rest of the processes still run on core 0.



We observe an overall reduction of execution time of partition 2, but not as much as expected. Indeed the individual execution time of all the processes has increased. We observe that some of the hardware performance counters distributions have a good correlation with the execution time, hinting at a possible cause of variability.

service	Provided by	Frequency	overhead
GET_TIME()	APEX		9736600 ns
p4_get_time()	PikeOS		240ns
TimeBase	CPU	37,496 MHz	17 cycles (9.4ns)
Alternate TimeBase	CPU	1,8 GHz	17 cycles (9.4ns)

Figure 3: T2080 processor contains several timer devices

The measurement environment is being used to gather precise behavior information on real avionics applications, for various deployment policies.

<u>The power management implementation (see D34D32_Partner_Input_TCS_07J_TechAnnex.pdf):</u>

In addition to respecting real-time constraints for critical system, TCS and Sysgo have implememted a first version of the power manager system: the driver DVFS and the control manager. The control manager has to be the less intrusive as possible (avoiding timing overheads), and be able to be integrated with existing virtual machines with minimal software changes.

Configuration	**Total power for 100 iterations(mWh)**
With DVFS	1.35
Without DVFS	1.05

Those results are not enough, they are however perfectly logical in their configuration. The driver version used can effectively change the frequency of the ARM core and the internal regulator. In order to make a better test and see the advantages of DVFS a new version of the driver is being developed by SYSGO for the year 3. It will be able to directly control the external regulator at runtime of the Board.

<u>The flexible TDM-Based Resource Management in On-Chip Networks (see D34_T33_TUBS_Technical_Annex.pdf):</u>

Time-division multiplexing (TDM) is the commonly used and well established solution to the problem of sharing re- sources in real-time Networks-on-Chip (NoCs). This work introduces an efficient method for sharing safely resources in networks-on-chip. Flexible TDM is performed using an additional control layer built on top of the existing NoC architecture. The major improvement is based on the RM's knowledge about the current state of the system i.e. global arbitration. The RM adjusts the length of the TDM-cycle by skipping the slots of non-active senders. This introduces dynamics which can drastically improve average latencies of transmissions whenever the system in not fully loaded.

TUBS works on hardware implementation on the IDAMC Many-Core platform. The hardware implementation is developed in VHDL with the Gaisler Leon3 processors and GRLIB IP library. Toolchain: Modelsim/Xilinx ISE and Virtex-6-LX760 Xilinx FPGA board. Moreover, we cooperate together with SYSGO AG on software implementation supporting ARINC-653 compliant operating systems. Results will flow to the WP8.2 (Living Lab Avionic).

3.4 Planned deployment and usage by partners

Most of the implementation efforts of the proposed mechanisms is in progress in the WP7 automotive domain Living Labs. CSOFT partner is directly involved in WP7. Avionics related domain mechanisms are naturally transferred to WP8 Living Labs without implementation, or partial implementation within EMC2. HIB has currently working with WP12.2: video surveillance for critical infrastructure and WP7.2: Highly automated highway.

Also, implementation of virtualization and isolation mechanisms will be done on top of existing hypervisors or RTOS. Partners providing hypervisor or RTOS like SYSGO or partners who are currently using those hypervisors like CEA, TCS, THALES, MBDA, HIB will then take benefit of developed extensions.

Moreover, Freescale eRPC implementation will be available as open source

3.5 Evaluation of solutions against EMC^2 goals

EMC2 Requirement	WP03 T3.2 implementations	Comment
HL-REQ-WP03-006	Android application virtualization capabilities. The hypervisor shall support the running of Android applications (API level 19).	Partner(s): HIB
HL-REQ-WP07-035	Provide secure data communication for resource access	Partner(s): CSOFT
HL-REQ-WP07-039	Provides API to request safety-critical resource	Partner(s): CSOFT
HL-REQ-WP07-041	Provides guest OS isolation	Partner(s): CSOFT
HL-REQ-WP07-042 HL-REQ-WP07-043	Provides resource management interface	Partner(s): CSOFT
HL-REQ-WP07-046	Provides core affinity	Partner(s): CSOFT
HL-REQ-WP1-005 HL-REQ-WP1-006 HL-REQ-WP4-004 HL-REQ-WP07-004 HL-REQ-WP07-034 HL-REQ-WP07-061 HL-REQ-WP08-009	The concept of cross-core Remote Procedure Call targeted at embedded systems -eRPC. eRPC client and server. eRPC modularity. eRPC generator of client and server stub routines. eRPC support for multiple services. eRPC support for segmented/chained messages. eRPC support for any data types. eRPC support for initial client-server hand-shaking	Partner(s): Freescale
HL-REQ-WP03-009 HL-REQ-WP04-001 HL-REQ-WP08-007 HL-REQ-WP07-025 HL-REQ-WP07-027 HL-REQ-WP07-036 HL-REQ-WP07-037 HL-REQ-WP07-038 HL-REQ-WP07-045	The exposed solutions fit transversally in all WPs where segregation/partitioning is needed.	Partner(s): MBDA
HL-REQ-WP03-003	Secure communication across partitions	Partner(s): SICS
HL-REQ-WP03-003	Runtime monitoring of guest configuration	Partner(s): SICS
HL-REQ-WP03-003	Initial secure configuration	Partner(s): SICS

HL-REQ-WP03-003	Firmware updates	Partner(s): SICS
HL-REQ-WP03-006	ARM based platform	Partner(s): SICS
HL-REQ-WP03-003 HL-REQ-WP03-009 HL-REQ-WP03-006 HL-REQ-WP03-008	On chip time and space partitioning	Partner(s): SICS
HL-REQ-WP03-001 HL-REQ-WP03-009	On-chip monitoring and dynamic configuration of virtualized resources	Partner(s): SICS
HL-REQ-WP07-008	Non-critical hardware access	Partner(s): SICS
HL-REQ-WP03-003 HL-REQ-WP03-009 HL-REQ-WP03-006 HL-REQ-WP03-008	On chip time and space partitioning	Partner(s): THALES
HL-REQ-WP03-003 HL-REQ-WP03-009	Time Composability	Partner(s): THALES
HL-REQ-WP03-003 HL-REQ-WP03-009	On-chip monitoring and dynamic configuration of virtualized resources	Partner(s): THALES
HL-REQ-WP03-010 HL-REQ-WP03-006	Control over-provisioning	Partner(s): THALES
HL-REQ-WP03-007 HL-REQ-WP03-008	SW Support for interconnect's resource sharing mechanisms	Partner(s): TUBS
HL-REQ-WP03-006	Interference caused by concurrent requestors must be bounded.	Partner(s): TUBS
HL-REQ-WP03-004 HL-REQ-WP03-005	Identification of concurrent accesses to interconnect	Partner(s): TUBS
HL-REQ-WP03-006	Error contention for accesses to shared resources	Partner(s): TUBS
HL-REQ-WP03-010	Efficient sharing of memory	Partner(s): TUBS
HL-REQ-WP03-007	Temporal synchronization of accesses to memory	Partner(s): TUBS
HL-REQ-WP03-00	With the hypervisor PikeOS, it is partial completed. It cannot be fully OS agnostics cause it is para-virtualization.	Partner(s): TCS
HL-REQ-WP03-008	Changed. A debug/trace probe has been set in order to be not intrusive. No code is used to trace.	Partner(s): TCS
HL-REQ-WP03-003	A first version developed used virtual machines with 3 different levels of security.	Partner(s): TCS
HL-REQ-WP03-003	The critical application reached it QoS	Partner(s): TCS
HL-REQ-WP03-001	iMX6 board chosen has thes functionalities	Partner(s): TCS
HL-REQ-WP03-010	The chosen application should have timing constraints to benefit from a reliable power management strategy	Partner(s): TCS
HL-REQ-WP1-003		Partner(s): CEA

4. T3.4 Security mechanisms and services

4.1 Introduction

The goal of task T3.4 is to study mechanisms and services of the RTE/Hypervisor to ensure security properties including static and dynamic security policies and services, encryption, authentication and access control.
During this task, the partners considered the following security properties:

- Secure dynamic reconfiguration of FPGA devices,
- The security features provided by separation kernel or the lightweight hypervisor mechanisms traditionally used in powerful systems,
- The VMs privacy protection from software attacks,
- A model-driven system engineering framework for embedded automotive systems that enables the configurations of basic software components (BSW) and the generation of a runtime environment layer (RTE).

4.2 Proposed mechanisms

INESC-ID investigated the **secure dynamic reconfiguration of FPGA devices** and the design of efficient cryptographic engines towards improved security-oriented modules. For hardware based systems, when used in safety critical applications, each running component must be isolated from the remaining components, including the reconfiguration process when the components are loaded. This also includes the need for a clear and well defined isolation and validation of the reconfiguration process. Towards this, a secure reconfiguration mechanism was developed for the FPGA providing a transparent and easy to use interface.

The dynamic partial reconfiguration functionality of FPGAs can be attacked, particularly when the FPGA is remotely located or the configuration bitstreams are sent through insecure networks. The existing FPGA technologies provide some built-in security mechanisms; however, these are often inadequate. The existing solutions still impose a significant impact on the reconfiguration process and on the available resources.

In this work, a solution to improve the security of dynamic partial reconfiguration of FPGAs was developed. This was achieved without significantly affecting the reconfiguration performance. This functionality is achieved by changing the encryption key of the remotely received bitstream by a randomly generated key, unique for each configuration, when storing them in the external unsecured memory.

SICS investigated security mechanisms that can be provided by a **separation kernel or lightweight hypervisor**, alongside other additional services that can be provided using the same security mechanisms. These mechanisms and technologies have in past been used on more powerful systems (e.g. server or desktop) but their use in embedded systems has been more limited with the notable exception of some smartphones.

CEA proposed a mechanism, namely **Blind Hypervision to protect Virtual Machines** that intends to guarantee both confidentiality and integrity of code and data of virtual machines (VM) from software attacks from other VMs and from the hypervisor itself. The untrusted hypervisor manages the VM without being able to access their code and data by relying on two trusted components:

- The Secure Memory Management Unit enables to securely partition and isolate memory areas dedicated to the hypervisor and each Virtual Machine,
- The Trusted Loader (with cryptographic functions) enables the hypervisor to load/update/migrate Virtual Machines without accessing their code and data in clear.

AVL has developed **a model-driven system engineering framework add-on (BSW and RTE generator)**, which enables the configurations of basic software components and the generation of a runtime environment layer (RTE; interface between application and basic software) for embedded automotive system, consistent with preexisting constraints and system descriptions. With the aim in mind to develop a tool bridge to seamlessly transfer artifacts from system development level to software development level to enable better traceability for safety critical development. This enables the seamless description of automotive software and software module configurations, from system level requirements to software implementation and therefore ensures both consistency and correctness for the configuration.
Additionally research on appropriate systematic approaches to support the identification of trust boundaries and attack vectors for the safety- and cybersecurity-relates aspects of complex automotive systems have been investigated. In the course of this investigation, a method to identify attack vectors on complex systems via signal interfaces has been developed focusing on a central development artifact of the ISO 26262 functional safety development process, the hardware-software interface (HSI).
Therefore, extensions for the HSI document to support the cyber-security engineering process have been made and automatization of this artifact for cyber-security engineering has been implemented.

4.3 Implementation, Prototyping

Secure dynamic reconfiguration of FPGA devices:
To evaluate the proposed secure dynamic reconfiguration system, experimental results were obtained for the resulting prototype on a Xilinx Virtex-7 FPGA device XC7VX485T-2. To better analyze the impact of the proposed solution, one base reconfigurable system was implemented without security, presenting the core architecture supporting the partial reconfiguration process. The resources used by the base reconfigurable system consist of the MicroBlaze processor, the buses, the DMA controller, the memory controller, the communication controller, and the configuration module.

The obtained post Place&Route results for the base system require 14463 Slice registers, 14608 Slice LUTs, 48 BRAMs, 3 DSP Slices, and an ICAP instance operating at 100MHz. Since the transfer of data is performed via DMA, a maximum throughput of 3.2Gbps (32-bit words at 100MHz) can be achieved, which is the limit of the ICAP interface. The obtained results for the complete system with the proposed security co-processor requires 7758 Slices and 78 BRAMs, that is, the proposed security system requires an additional 774 Slices and 30 BRAMs. These results allow concluding that the proposed security co-processor imposes an overall resource increase of 1.02% regarding the base reconfiguration solution. Performance-wise, the kernel is able to operate at maximum frequency of 196MHz. At this frequency, the kernel achieves throughput of 2.5Gbps, 22% lower than the maximum reconfiguration throughput of the ICAP interface. This limit is imposed by the used AES cores. If faster cryptographic implementations are considered, lower reconfiguration times can be achieved. Note that the cryptographic kernel and DMA use a different clock, being able to operate at a different frequency than the base system.

Separation kernel or hypervisor:
SICS are experimenting with security services in an MPSoC environment where software components at different security co-exist. In particular, they are considering the interface to the security services and how security functionality can be affected by other software executing on the same devices. Using the experimental hypervisor / separation kernel from T3.3 and its existing interfaces as a simple framework for security services, a sample security services has been implemented which was assumed to be

1. critical to system security
2. timing critical
3. Affecting operation at all security levels
4. Affecting operation on some or all cores

The next step will be to add runtime monitoring and life-cycle management services.

Blind Hypervision to protect Virtual Machines:
A first proof of concept of the mechanism has been implemented relying on the ARM TrustZone hardware feature. A Secure Kernel implementing in software the Secure MMU security function runs in the Secure World, protected from attacks from the VMs and the hypervisor running on the Normal World.

The Trusted Loader has not been completely implemented yet (no cryptographic functions). VMs binaries are statically embedded in the Secure Kernel binary in clear text and securely (no possible access from hypervisor and VMs) loaded to their dedicated memory by the Secure Kernel when requested by the hypervisor.

A demonstration with a lightweight hypervisor blindly scheduling 2 bare metal (i.e. without guest OS) VMs (a malicious one and a target victim) validated the implemented security properties. The small size of the Trusting Computing Base (the Secure Kernel size is 2400 LoC) makes its formal verification feasible at a relatively low cost. See technical annex for more details about the implementation and the demonstration.

RTE generation and BSW configuration tool-extension for embedded automotive systems:
The underlying concept of the approach is to have a consistent information repository as a central source of information, to store all information of all engineering disciplines (among others safety and security) involved in embedded automotive system development in a structured manner. The concept focuses on allowing different engineers to do their job in their own specific way, but providing traces and dependency analysis of features concerning the overall system, e.g. safety, security, or dependability.

The tool approach introduced in this work provides a framework for the visualization of ASW and BSW interface configuration and automated generation of these interfacing .c and .h files. Furthermore, the available hardware- software interfacing (HSI) information can be used to generate basic software (BSW) component configurations and the HSI information import functionality can also handle HSI spreadsheet templates to ensure more versatility of the tool.
The work consists of the following parts:

- An AUTOSAR aligned UML modeling framework: Enhancement of an UML profile for the definition of AUTOSAR specific artifacts, more precisely, for the definition of the components interfaces (based on the virtual function bus abstraction layer).
- BSW and HW module modeling framework: Enhancement of an UML profile to describe BSW components and HW components. To ensure consistency of the specification and implementation for the entire control system.
- RTE generator: Enables the generation of interface files (.c and .h) between application-specific and hardware-specific software functions.
- Basic software configuration generator: Generates BSW configurations according to the specifications within the HSI definition.

The benefits of the approach for development of automotive embedded systems have been demonstrated based on an automotive battery management system (BMS).The use-case consists of 10 ASW modules and 7 BSW modules with 19 interface definitions between ASW and BSW and makes use of the 3 fundamental low-level HW functions (digital input/output, analog input/outputs, and PWM outputs). The definition of the 19 HW/SW interfaces with 10 parameters for each SW signal and 13 parameters for each HW pin sums up to 437 parameter configurations within the HSI spreadsheet template or in the MDB tool, which can be used to generate ASW/BSW interfaces and BSW configurations. This results in 8 additional auto-generated interfacing files with 481 lines of code (LoC) source and 288 LoC configurations.

The related publication can be found in Proceedings of the 8th European Congress of Embedded Real Time Software and Systems http://www.erts2016.org/uploads/pdf/erts_2016_Proceedings.pdf [1]

[1] Georg Macher, Rene Obendrauf, Eric Armengaud, Eugen Brenner, and Christian Kreiner. RTE Generation and BSW Configuration Tool-Extension for Embedded Automotive Systems. In 8th European Congress Embedded Real Time Software and Systems Proceedings, 2016.

4.4 Planned deployment and usage by partners

Secure dynamic reconfiguration of FPGA devices:
INESC-ID worked on a hardware implementation on the Xilinx Virtex-7 FPGA device XC7VX485T-2. The hardware implementation is realized in VHDL using the Xilinx ISE 14.7 and EDK 14.7 tools.
There has been collaboration with CSoft.

Separation kernel or hypervisor:
The implementation initially targeted a 64-bit multi-core ARM system primarily designed for mobile devices and high-end embedded systems. SICS are currently porting this to a more accessible 32-bit ARM multi-core system after request by other EMC2 partners. The new target is a COTS embedded system based on the ARMv7-A architecture in a Broadcom BCM2836 system-on-chip. The toolchain used for development in the GNU compiler collection (GCC) for ARM. The security services will use virtualization technology from T3.3. Some cooperation is planned together with Ericsson in WP1 and WP3.

Blind Hypervision to protect Virtual Machines:
The demonstration has been implemented on the Freescale SABRE SDB development board containing an ARM cortex A9 iMX6Q processor. The mechanism has been proposed to be integrated (Technology Transfer Level scope 3) in the automotive use case 7.5 “Infotainment and eCall Application Multi-Critical Application”.

RTE generation and BSW configuration tool-extension for embedded automotive systems:
The work is part of the AVL use case T7.3 in WP7 and has been developed / evaluated in cooperation with project partner ViF.

4.5 Evaluation of solutions against EMC2 goals

Secure dynamic reconfiguration of FPGA devices:

Reference to HL Requirements	Requirement ID	Short Description
HL-REQ-WP03-001		Identification
HL-REQ-WP03-02E-001	NXP-001	Configuration Interface
HL-REQ-WP03-009		Analysis to ensure space- and time-partitioning of accesses to shared resources
HL-REQ-WP07-008		SW architectures - safety mechanisms
HL-REQ-WP07-022		ISO 26262 safety case (before integration/validation) automatic generation

Separation kernel or hypervisor:

Reference to HL Requirements	Requirement ID	Short Description
HL-REQ-WP03-003	16E-001	Secure communication across partitions
HL-REQ-WP03-003	16E-002	Runtime monitoring of guest configuration

Blind Hypervision to protect Virtual Machines:

Reference to HL Requirements	Requirement ID	Short Description
HL-REQ-WP03-002 HL-REQ-WP03-003	16E-002	Runtime monitoring of guest configuration
HL-REQ-WP1-003	07A-001	RTOS/VM-MONITOR shall not be able to access module/task/VM code/data
HL-REQ-WP1-003	07A-002	A secured MPU should be implemented either in software/hardware

	11M-008	Security management
HL-REQ-WP03-001		Identification
HL-REQ-WP03-004		Concurrent tasks identification
HL-REQ-WP03-005		Shared resource identification
HL-REQ-WP03-008		HW/SW co-design
HL-REQ-WP03-009		Analysis to ensure space- and time-partitioning of accesses to shared resources

RTE generation and BSW configuration tool-extension for embedded automotive systems:

Reference to HL Requirements	Requirement ID	Short Description
HL-REQ-WP03-008		Methods supporting the interconnection of hardware and software mechanisms.
HL-REQ-WP07-005		Availability of a set of harmonized tools that allow to easily integrate the design and development work flow, from system to component/code level.
HL-REQ-WP07-007		Industrial tools such as e.g., Enterprise Architect, MKS, Matlab Simulink shall be integrated into a tool chain supporting seamless model-based systems / SW / safety engineering based on a semi-formal system description (e.g., EAST-ADL, SysML)
HL-REQ-WP07-009		The safety case information shall be automatically compiled based on existing information from product development.

5. T3.5 Safety platform and real-time mechanisms

5.1 Description of the task

The application cases have safety requirements including resource availability (processing time, memory, bandwidth, etc.), predictability and fault tolerance. To provide guarantees that these requirements will be met, it is responsibility of the RTE/Hypervisor to integrate safety oriented features (at software or hardware level) and provide interfaces encouraging the application of safety methodologies when designing mixed-critical software. The parallelism of multi/many-core environments add a lot of complexity to the problem of proving that the guarantees are provided in any case.

The main objective of T3.5 is to specify and design safety-oriented features of the RTE/Hypervisor, such that the safety requirements of the applications are met, taking into account the cohabitation of different tasks with different safety requirements on parallel/distributed architectures. The interference between the different task groups shall be harmless regarding safety properties, while ensuring the best possible utilization of the available hardware resources. The results of the task will be proposed to standardization bodies, such as AUTOSAR.

5.2 Overview of technologies in the task

This section provides an overview of the technologies developed in the task, and regroups them into categories that relate to one or more high-level requirements.

The first category involves the design of RTE/Hypervisor mechanisms to have fault tolerant design. The two main concerns are on scheduling and resource allocation and usage. From the scheduling point of view, the technologies designed in this task focus on enforcing task isolation (HL-REQ-WP3-003), providing guarantees and bounds on the delay to access an execution resource for a critical task (HL-REQ-WP3-006), and providing a consistent time base to rely upon in the scheduling of tasks (HL-REQ-WP3-011). These scheduling mechanisms all try to achieve this while maximizing the use of the computational power of the architectures (HL-REQ-WP03-010). From the resource allocation and usage point of view, the technologies designed in this task first focus on providing safe access to the shared resources: resource allocation policies and synchronization techniques are designed to prevent concurrent accesses while ensuring that there is no deadlock, even in case of a preemptive scheduling (HL-REQ-WP3-006/007).

The second category involves fault detection and mitigation mechanisms in the RTE/Hypervisor. Though the techniques described above reduce significantly the probability a fault will occur, it is still mandatory to supervise the execution and check that it is going as planned. New architectural patterns designed in this task allow a safer and less intrusive monitoring of the platforms (HL-REQ-WP3-008/009), along with the appropriate feedback to the application for fault mitigation, through reconfiguration for example (HL-REQ-WP03-001).

The third category is a set of configuration and verification tools to verify feasibility and that the technologies of the previous categories actually provide the safety guarantees that they claim, and also takes into account the standardization of the interfaces to these technologies. These mechanisms rely on analysis tools and methodologies that can identify and inspect the tasks, their communications, their real-time constraints and their interactions (HL-REQ-WP3-002/003/004/005).

Finally, the fourth category involves the design of safe communication layers, to allow a good responsiveness level in the dynamic and changeable environments of connected devices. The technologies designed in the task focus on monitoring network congestion and using alternate channels in case of an overload (HL-REQ-WP03-001), and providing a consistent time in a distributed environment (HL-REQ-WP3-011).

5.3 RTE functions and protocols

5.3.1 Fault tolerant design

The extension of single-core OS EB Tresos to multi-core by Elektrobit (01L), with the constraint to retain both its classification under level D of the Automotive Safety Integrity Level (ASIL-D), and its compliance with the interfaces specified by the AUTomotive Open System ARchitecture (AUTOSAR) initiative, includes the following functions: 1) non-blocking interface to trigger actions on other cores, increasing isolation of a fault on the target core; 2) message passing interface with local memory allocation to leverage the memory architecture of current automotive multi-cores. The development has passed the verification criteria according to part 6 of ISO26262, and the analysis w.r.t the ASIL and safety are on-going. This work is done in cooperation with Infineon, Denso, BMW, AVL and VIF, and the OS will be part of the demonstrator in LL1, T7.1 and T7.3.

The resource safety principles and dynamic scheduling algorithm introduced by CEA (07A) are implemented in a hypervisor prototype on the many-core chip MPPA256, including the following functions: 1) an access control service in constant space and time to enforce isolation; 2) a service call interface with resource lending to prevent denials of service; 3) minimal memory management service relying for safe and dynamic task creation/memory allocation; 4) dynamic mixed-critical asymmetric scheduler with bounded latencies for access to CPU by critical tasks (measured latencies: 3.2µs if the target core is idle, or 4.0µs if the target core is used by a less critical task). The overhead incurred by these functions is on-going, as well as WCET analysis to provide the bound on CPU access latencies in case of preemption. The prototype and the preliminary results were presented to partners from LL1 and LL2, in particular Denso with whom future collaboration was suggested.

The protocol for access control to NoCs introduced by TUBS (01V) is implemented in an extension of the IDAMC architecture. The arbitration protocol is implemented using an overlay to the NoC to exchange access requests/grants between a client attached to each node and resource managers. The overhead for 8 senders resulted in 12% surface overhead for the sender's network interface. The worst-case latencies outperform a state of the art approach by up to 80%. Full or partial implementation into the LL Avionic is on-going.

The certifiable shared memory for avionics proposed by SILKAN (07G) is implemented in a ARRIA10 FPGA, and includes the following functions: 1) data access with independence between application and communication, data integrity and temporal/spatial data consistency; 2) seamless routing of data between remote processes; 3) synchronization and time stamping functions with accuracy up to 200ns. As the technology is a hardware IP, it is planned to be fitted into a Freescale or NXP component based in CPLD technology. The demonstration aimed is the UC1 of LL2 Avionic.

The co-processor proposed by EADS (01K) and CAS (01H) to execute two applications of different Development Assurance Level (DAL) level on a single dual-core processor is implemented by a Xilinx MicroBlaze attached to a Freescale/NXP P5020 to provide a function to monitor events in a critical application in cohabitation with another of lesser criticality, compare these events with a reference obtained for the critical application alone, and take corrective measures in case of a drift. This work is directly integrated to the demonstrator in WP8.1, with a strong cooperation between EADS and CAS, as well as with partners TUBS, TUKL and ISEP.

5.3.2 Fault detection and mitigation

The Fault Tolerant Multi-core Library (FaToMLib) proposed by Freescale (04B) is prototyped on FSL/NXP multi-core platforms, and includes algorithms to detect Loss of Integrity and Data Errors: 1) Time Monitored Comparator (TMC) to detect high-level LOI; 2) Time Multi-Watchdog Processor (TMWDP) for LoI detection; 3) Heart Beat Interactive Consistency (HBIC) for high-level LoI detection.

TMC has already been filed for a patent, and HBIC is likely to be filed too. The implementation of the algorithms is done by a combination of hardware and software.

5.3.3 Configuration and verification

The real-time analysis technique proposed by NXPNL (17C) provides two functions: 1) temporal analysis for task graphs with cyclic dependencies, taking into account task interferences; 2) simulator for exact verification of the analysis results. These functions were evaluated using a WLAN 802.11p application executed on a software defined radio, and the evaluation was compared to alternative techniques. There is a cooperation with INRIA on the temporal dataflow analysis techniques.

The architecture design phase safety analysis to verify safety properties and global efficiency proposed by CSOFT (13B) provides the following functions: 1) systematic identification of faults or failures that could lead to the violation of safety goals or safety requirements; 2) evaluation of the consequences of each identified fault. The analysis is implemented using a Qualitative Failure Modes and Effects Analysis (FMEA) and a Qualitative Hardware-Software Interaction Analysis (HSIA) at the software architecture level. The target platform is a new real time operating system developed in cooperation with T3.2, T3.3, T3.4, T3.5 and T3.7 with partners CISTER and INESC-ID.

The CompSOC platform proposed by TUE (17E) allows for the synthesis of hierarchical architecture configuration**s**, with the following functions: 1) component-based predictable performance analysis; 2) composable and predictable dynamic loading of components; 3) online monitoring and debugging. The platform relies on Xilinx MicroBlaze processors, and allows synthesis of heterogeneous tile-based many-core architectures. A collaboration with NXP NL in T3.5 provided an application scenario, and a cooperation with Technolution is planned to integrate the CompSOC platform with their demonstrator in T7.1.
The response time analysis developed by ISEP (13A) relies on the analysis of the variability in memory demand from a job due to cache-persistence of some memory blocks. This allows to check schedulability of fixed-priority preemptive systems with an average improvement of 10% compared to state of the art approaches. This work is expected to contribute to the joint work with CSOFT for the LL1.

The tool for assignment and allocation of AUTOSAR runnables proposed by DTU (05B) and Volvo (16F) is implemented in C# using the .NET Framework, and maps an application according to constraints provided by the designer (communication bandwidth, core utilization…). Based on a simulated annealing meta-heuristic, the tool produces a result satisfying 3 out of 4 end to end signal constraints in 8 minutes, for an application of 50 components/75 runnables to an architecture with 3 cores. The work on this tool is part of the Volvo use case in T7.6.

The microcontroller firmware proposed by IFAG (01A) to comply to ISO26262 and allow fast and bug-free development is implemented into the lead 2nd Gen. AURIX product, and includes the following functions: 1) CRC checks, ECC error detection, and register read-back/compare to increase safety; 2) hardware/firmware interaction; 3) partial reprogramming of the FW features to ease the development of derivative products.

The measurement environment proposed by Thales (07H) is implemented on a NXP QorIQ multi-core and Sysgo PikeOS 4.0, using the improved ARINC653 (APEX) personality, and includes functions to: 1) quantify and qualify the interference between applications in different deployment configurations; 2) compare the variability in application behavior in the different deployment configurations; 3) select deployment configurations with lowest impact on determinism. The analysis of the measurements is ongoing, in collaboration between Thales, TAV, Sysgo AG and Sysgo SRO.

As a part of the system modelling tool proposed by AVL (02A), a code generation tool provides interface files between application specific and hardware specific software functions, whose configuration can be updated and validated by the modelling tool. This work eliminates the need for manual information reworking, enforcing reproducibility and traceability argumentation. This work is the result of a cooperation with partner ViF and is part of the AVL use case T7.3.

5.3.4 Safe communication

The platform for Intelligent Transport Systems (ITS) proposed by NXPNL (17C) is implemented on NXP MK5 802.11p platform and provides the following functions: 1) a load-balancing framework to adapt communication to channel load and avoid collisions; 2) a simulation framework to perform Decentralized Congestion Control (DCC) study. The simulation framework is used to compare the performance of the Data Rate based DCC algorithm used in function 1 against other DCC implementations. A demonstrator of an ITS network is available based on the simulation framework. The work is a result of collaborations with Technolution (WP7), TNO (WP2, WP7) and in the Use-case Cooperative ADAS (T7.2).

The implementation of a synchronization protocol compliant with IEEE 802.1AS by NXPNL (17C) is implemented in the NXP SJA1105 switch and relies on the publicly available Linux PTP implementation; it allows for a time synchronization across nodes of an in-vehicle network. The first evaluation of the implementation showed a time synchronization with an accuracy within the required limits for typical automotive networks. Further evaluation is in progress using realistic in-vehicle applications. A cooperation with TUE is on-going to integrate the time synchronization protocol and the CompSOC platform.

5.4 Evaluation of the functions against EMC2 goals

As mentioned in Section 5.2, the technologies developed in T3.5 answer several High-Level Requirements:

- enforcing task isolation (HL-REQ-WP3-003)
- providing guarantees and bounds on the delay to access an execution resource for a critical task (HL-REQ-WP3-006)
- providing a consistent time base to rely upon in the scheduling of tasks (HL-REQ-WP3-011)
- maximizing the use of the computational power of the architectures (HL-REQ-WP03-010)
- preventing concurrent accesses while ensuring that there is no deadlock, even in case of a preemptive scheduling (HL-REQ-WP3-006/007)
- allowing safer and less intrusive monitoring of the platforms (HL-REQ-WP3-008/009)
- providing appropriate feedback to the application for fault mitigation, through reconfiguration for example (HL-REQ-WP03-001)
- identifying and inspecting the tasks, their communications, their real-time constraints and their interactions (HL-REQ-WP3-002/003/004/005)
- monitoring network congestion and using alternate channels in case of an overload (HL-REQ-WP03-001)
- providing a consistent time in a distributed environment (HL-REQ-WP3-011)

6. T3.6 Dynamic application and platform control

6.1 Introduction

The majority of currently used safety critical systems avoid the dynamics in platform control to improve the determinism of the design. However, allowing and exploiting dynamic system behavior can significantly improve performance and resource utilization of the architecture. The main goal of the task T3.6 is to support the dynamics and dynamic change as long as the required freedom from interference is enforced, i.e. the safety of applications and of the whole system is guaranteed. Consequently, partners working in the task develop mechanisms which allow work conserving behavior of the systems, therefore its higher utilization, increased safety, security and performance.

The first phase of work in T3.6 was driven by the subtask "ST3.6.1 Dynamic platform control investigation and development of mechanisms for run-time platform adaptation" and concluded in the D3.3. It concentrated on: (1) analysis of opportunities for dynamic platform control and (2) the development of mechanisms for run-time adaption comprising monitoring and flow control, re-configuration, and scheduling.

The main input to this deliverable originates from D3.3 and was conducted in the scope of the second subtask "ST3.6.2 Derivation of RTE functions and protocols from mechanisms, investigation and validation" which translates the results of the ST3.6.1 to the RTE functions and protocols. The main challenges arise from the safety requirements which demands that the proposed solutions must not impact performance and safety guarantees of the existing RTEs used in the respective living labs. Therefore the efficiency of the extensions was investigated also with respect to the validation of the worst-case guarantees.

6.2 Proposed mechanisms

The mechanisms designed by partners were grouped depending on the design objectives achieved through the introduced dynamics into three clusters: performance (C1), fault-tolerance (C2), security (C3). In the following we present the short summary of the introduced mechanisms.

6.2.1 Cluster 1 (C1): Performance

TTTech investigated dynamic (re)configuration mechanisms for deterministic networks. For the work in Task 3.6, concepts of incremental scheduling based on modifications in the communication network were developed. The management software executed on the communication devices (switches and end systems) is extended to match the new features of the more dynamic but deterministic applications. This is done based on the existing configuration tools for TTEthernet, which are extended to integrate the new scheduling algorithms as well as to provide required new features and parameters used in the dynamic system configuration process.

TUBS introduced approach for providing efficiently service guarantees in NoCs for mixed-critical real-time systems. The mechanism combines the global scheduling for the end to end guarantees, with the local arbitration performed in routers. TUBS introduced scheduling modules, called resource managers (RMs), with which applications have to negotiate their accesses to interconnect. Synchronization is achieved using control messages and a dedicated protocol. RMs conduct a global, priority based scheduling and grant transmissions access to the NoC for a predefined amount of time. This allows exclusive access to the NoC, hence reducing blocking and decreasing the size of necessary buffers in routers. Moreover, it supports sharing of the same VC between transmissions with different criticality levels while preserving service guarantees.

CETRAC execution platform designed by SILKAN The CETRAC execution platform allows the connection of existing devices via a Real-Time Ethernet or Arinc664 Part7 (or other) connection as well as the connection of an "End System". Extensions are possible to provide connectivity to networks such as the Arinc825, the ARINC 429 or the Mil-Std-1553. This architecture is based on a Software Hosting Structure (SHS). It provides a distributed database and numerous system-level services to which are connected: (i) the physical elements necessary to connect to the surrounding environment; (ii) the Software Hosting Structure (SHS), which supports the functions required to manage applications and models; (iii) the data storage system.

In the scope of task T3.6 Thales researched “Dynamic Deterministic Platform control Software (DPS)” in the context of avionic safety critical software. The goal of Deterministic Platform Software is to ensure a deterministic time behavior of safety critical applications while running on top of non-deterministic multi-core COTS.

6.2.2 Cluster 2 (C2): Fault-Tolerance

TUW works on ontology-based runtime reconfiguration [1], that is, a novel semantic technique to dynamically recover or substitute a failed service of a distributed real-time system. The method is designed to perform temporal predictable reconfiguration that is necessary for real-time systems. The technique can run on platforms based on service-oriented architectures and platforms that enable reconfiguration of its structure during runtime, i.e., communication between services is adaptive.

Secure System Monitoring and Fault Injection is developed by CSOFT proposed infrastructure that allows each safety-critical component to register dynamically specific variables and callbacks through a C API. Their values are then observed and modified by an online monitoring and test applications to evaluate the correct state of each component and overall system security. Allowing the change of these variables, during runtime, allows to test the safety containment measures by forcing specific faults that otherwise wouldn’t be possible under normal testing.

6.2.3 Cluster 3 (C3): Security

INESC-ID is developing an infrastructure for online monitoring with built-in fault injection capabilities. The mechanism improves the security of the dynamic, partial FPGA-reconfiguration process. The main goals are (i) a low footprint on the available reconfiguration logic and (ii) a minimum impact on the reconfiguration process itself. Security wise, the proposed solution considers the largest set of security confidentiality, integrity, authentication, and freshness of the bitstreams during storage on the unsecured external memory and during the reconfiguration process itself.

6.3 Implementation, Prototyping

6.3.1 Cluster 1 (C1): Performance

TTTech designed a flexible simulation environment to analyze and evaluate distributed scheduling algorithms based on the principles of industrial control networks. The simulation environment supports a set of prototypical incremental scheduling algorithms (CSI) including 1) simple CSI - this algorithm tries to divide the total message interval into equal sub-intervals per link. The algorithm tries to allocate a transmission window on each link within its assigned sub-interval and fails if no free interval is found. 2) proportional CSI – similar to simple CSI, but the sub-interval division is proportional to the current link utilization. 3) iterative CSI – a sequential approach taking the end of the allocated transmission window in its current link as the beginning of the search interval for the next link. 4) Proportional Random CSI – Similar to proportional CSI, but instead of searching for the first free interval, divides the search space in various parts and randomly picks one. And 5) random Iterative CSI – Same principal as Iterative CSI, but extended with the random search function as mentioned in Proportional Random CSI.

The evaluation of the TUBS solution for real-time NoCs was done through comparison with NoC architectures using priority-based arbitration done locally in the routers. According to TUBS approach, the RM safely delays the execution of transmissions with real-time (RTT) requirements in order to improve the performance of best-effort applications. For each RTT is computed a budget called slack, which defines the maximum delay an RTT can experience without endangering its deadline. Later, the RM monitors the slack budgets of currently ongoing RTTs and dynamically adjusts the priority of best effort transfers accordingly. The worst-case latencies (i.e. temporal budgets) were evaluated using the formal worst-case analysis implemented in the pyCPA framework. Simulations were carried out with OMNeT++ event-based simulation framework and HNOCS library. The applications with real-time constrains were modeled after the MPEG-4 video decoder application. For best effort we use activation traces of applications from the CHSTONE benchmark. The results have confirmed initial assumptions that by selectively prioritizing best-effort traffic over critical one (using slack budgets) resulted in significant improvement of the system utilization (up to 80%).

SILKAN divided their development and evaluation efforts into three stages. The first one is the CETRAC technology which is described in T3.5 and deployed into FPGA. The second one is the application generation tool, called ControlBuild (provided by Dassault Systems) tuned for hybrid simulation purposes to include natively the software housing structures. The last level to consider is the middleware (called here “SW housing structure”) which discharges the user application from all the communication tasks and use all the powerful and the services provided by the communication technology CETRAC.

In the scope of the T3.6.1 Thales identified a set of DPS solutions acting at different levels: Operating System, Hypervisor, Scheduling, inside partitions, and enforcing either a control strategy or a regulation strategy. In the scope of the T3.6.1 these DPS solutions were evaluated for avionics domain, including DPS with dynamic, adaptive or reactive behavior. The evaluation included 1) extraction of major stakes from standards 2) deduction of the first-order evaluation criteria 3) definition of assessment levels for each criteria 4) ranking of assessment criteria with regard to their importance in the avionic domain 5) and finally evaluation of each DPS with regards to the above defined assessments.

6.3.2 **Cluster 2 (C2): Fault-Tolerance**

Ontology-based runtime reconfiguration (ORR) exploits implicit redundancy to provide fault-tolerance to the system. For the evaluation, TUM has implemented ORR in C++ on top of the Robot Operating System (ROS), a SoA-based platform. The application can be distributed on several components or nodes implementing some services, e.g., ORR. Further ROS provides a communication framework suitable for dynamic reconfiguration. In particular, nodes exchange messages via topics, which is also known as publish/subscribe. Subscribing nodes do not care from which nodes the messages of interest come from, i.e., a node is just interested in the content (also known as data-centric approach of communication). Therefore, a failed service can be shut down and substituted by another service, simply publishing to the same topic (in contrast to connection-oriented communication: the sender and receiver of the messages need no change). The first prototype was fully implemented on ROS. However, a second demonstrator shows that ORR can also be integrated into existing systems with an appropriate communication gateway.

DNDE evaluates using Advanced Driver Assistance Systems a safe and dynamic update procedure. The target is to make safe partial software updates during the vehicle’s lifetime, i.e. ensuring a correct function of the new and the previously installed applications using the EMC² RTE. DNDE implements a Green Light Optimized Speed Advisory (GLOSA) application for demonstration purposes, which is the focus of the work conducted in this work package. The capability-based approach proposed by CEA provides partly appropriate concepts for implementing the aspired safe and dynamic RTE. However, the approach spans the system as a whole, i.e. a considerably high amount of changes in the kernel are needed for the integration. Further hardware support is needed for an efficient implementation.

6.3.3 Cluster 3 (C3): Security

For evaluation of their solution in T3.6, INESC-ID is currently developing the memory manager and the resource manager with particular focus on the RTE/ Hypervisor for the online monitoring with built-in fault injection being developed by CSOFT for the T7.5 use case demonstrator. The infrastructure allows a component to register specific variables and callbacks through a C API. These values are then observed and modified by an online monitoring mechanism to set and assure the correct state of each component of the system.

6.4 Planned deployment and usage by partners

A strong ling to Living Labs was a major objective of T3.6. The majority of mechanisms have been transferred to Automotive and Avionic Living Labs – WP7 and WP8 respectively.

In WP7 TUW deployed the reconfiguration mechanism in the usecase WP7.3 demonstrator coordinated by AVL which implements parts of an electric vehicle based on the microcontroller platform AURIX from Infineon in cooperation with AVL, BUT, TNO and VIF.
Based on the results from WP1 regarding SoA, WP7.3 concentrates on demonstrator running ORR on ROS for providing fault-tolerance to data on the CAN network of an electric vehicle (for more details, see D7.8). GCC and ROS tools have been used to compile, debug, simulate and demonstrate the implementation. Moreover, it was proven that ORR can be integrated also in non-SoA systems as independent platform.

Results from CSOFT flows into the usecase WP7.5 while being developed in conjunction with other WP3 tasks, namely: T3.2, T3.3, T3.5 and T3.7 along with EMC2 partners: CISTER and INESC-ID. DENSO targets the usecase WP7.1. The initial evaluation was done on iMX6 platform from Freescale with Embedded Linux from Yocto project and communication middleware SOME/IP from BMW with RPM package manager and in parallel on AURIX TC277 from Infineon with AUTOSAR implementation AutoCore from Elektrobit.

WP8 Avionic applications were targeted by TUBS. The results from the initial implementation of the RM will flow into the WP8.2. usecase. TUBS is porting an application provided by Airbus to IDAMC, a NoC-based many-core platform with mixed critical capabilities while employing mechanisms developed in Tasks T3.5 and T3.6 from WP3 to bound the interference that the Airbus application suffers (from other applications in the system) when accessing the NoC. Experiments were done with in VHDL with the Gaisler Leon3 processors and GRLIB IP library. Toolchain: Modelsim/Xilinx ISE and Virtex-6-LX760 Xilinx FPGA board. The same usecase (WP8.2) is targeted by Silkan and their Cetrac platform developed in cooperation with the task T3.5.

WP11 TTTech is a main partner in the usecase WP11.2. The aim of the demonstrator from Task 11.2 is to demonstrate and evaluate the results from WP3 in a realistic and practical setting. Partners AIT Austrian Institute of Technology (Smart Vision System) and Denmark Technical University (Mixed-criticality Networking) are involved in the demonstrator, whereby both partners have set up and configured a system containing an (extended) TTEthernet network system. DTU mainly focus on the integration of the proposed technology in the configuration tool chain, making (re)configuration mechanisms available for system developers. AIT, on the other hand, demonstrates a novel approach for monitoring people on airports and passenger data by interconnecting cameras, server and clients in a deterministic communication network.

Finally, Silkan designed a hardware implementation on the Xilinx Virtex-7 FPGA device XC7VX485T-2. The hardware implementation is realized in VHDL using the Xilinx ISE 14.7 and EDK 14.7 tools. This module will be a core component of the xLuna critical system being developed by the CSoft partner. This implementation is designed in layers and following the avionic standard VISTAS for hybrid simulation which meets the WP8.2 usecase's goals.

6.5 Evaluation of solutions against EMC^2 goals

The majority of partners concentrated on the requirements originating from WP3. The most frequently used were requirements for runtime adaptation (e.g. HL-REQ-WP03-001, HL-REQ-WP03-002 and HL-REQ-WP03-003, HL-REQ-WP03-006) which require identification of the involved system components and applications which are running concurrently in the system TTTech, TUW, TUBS, DENSO. Of special importance are requirements enforcing safe and predictable sharing of resources (memory, interconnect) e.g. resource pre-reservation, AUTOSAR OS Application multi-core allocation, dynamic resource allocation based on criticality, dynamic loading, creation and execution of tasks at run-time.
This included also 01V-002 (SW Support for interconnect's resource sharing mechanisms),

INESC-ID considered additionally HL-REQ-WP10-001, HL-REQ-WP10-010, taking into account the dynamic binding of drive functions allowing for the needed flexibility in order to adjust to new application conditions. CSOFT addresses the high level requirement HL-REQ-WP07-045 by providing the necessary technology to perform online monitoring and fault injection on the target platform, as well a list of 10 technology requirements specified in D3.1 from ID 13B-40 to 13B-49. Cluster C2 targeted fault-tolerance requirements e.g. HL-REQ-WP03-006 (Error contention for accesses to shared resources) and 1V-004 failures on time base / triggering of external watchdog

The main challenges in D3.4 arise from the safety requirements which demand that the proposed solutions must not impact performance and safety guarantees of the existing RTEs used in the respective living labs. Consequently, partners referred to: HL-REQ-WP03-009 (designed mechanisms should support analysis to ensure space- and time-partitioning of multiple simultaneously-executing software components with special support for shared resources), HL-REQ-WP03-003 (Parts, e.g., services or communication, of the system should be spatially and temporally encapsulated.), HL-REQ-WP03-006 (Mechanisms should support bounding of the interference resulting from accesses to different shared resources (e.g. memory, interconnect) to enable tight analysis).

7. T3.7 OS support functions

7.1 Introduction

The primary goal of the T3.7 task is to ensure that new Real Time Extensions (RTE) defined in other WP3 tasks (T3.2-T3.6) are supported in Operating Systems used for RT applications (Real Time Operating Systems – RTOS). It deals with implementations of RTE extensions for existing RTOSes (improvements, modifications, new features) as well as the implementation of a new RTOS designed for Multi-core Mixed-Criticality applications.

7.2 Proposed mechanisms

This section provides the overview of solutions (clusters) that have been elaborated by T3.7 partners and firstly introduced in the deliverable D3.3, chapter 7.3:

Class 1 (C1): Entirely new RTOS design and development for Multi-core Mixed-Criticality
- Multi-core Mixed-Criticality RTOS Interface

Class 2 (C2): Existing RTOS enhancements
- Lightweight tracing capability for multicore PikeOS-based applications
- Deterministic garbage collection into PikeOS

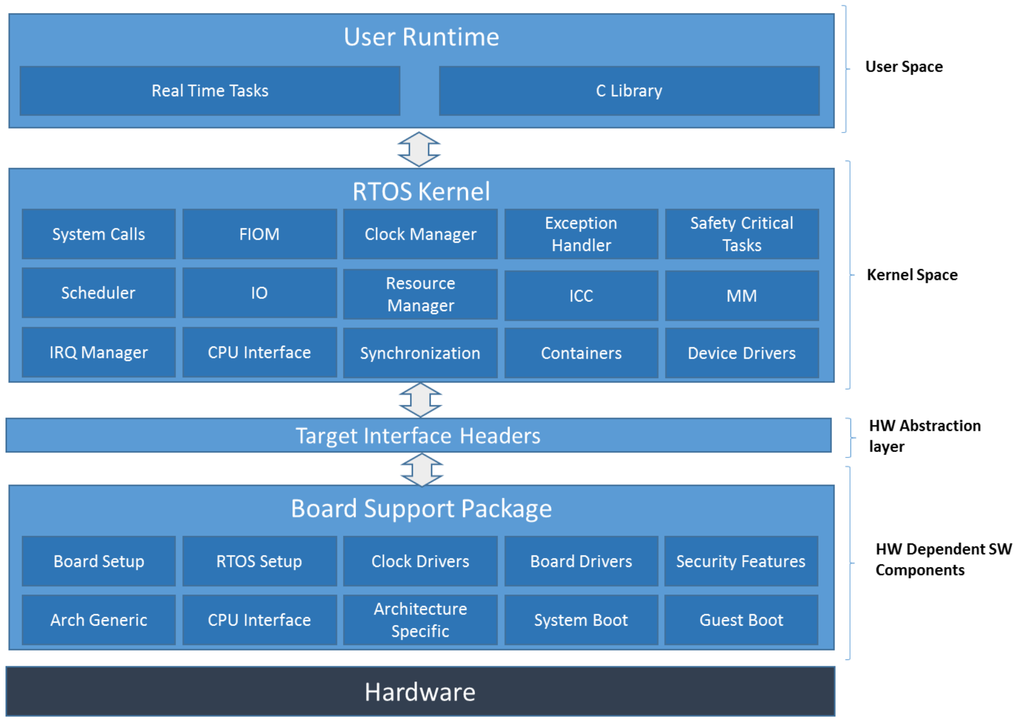
Class 3 (C3): Inter-Core communication support for existing RTOSes
- Inter-core communication library for Tresos Safety OS
- Remote Processor Messaging (RPMsg)

7.3 Implementation, Prototyping

This section discusses results from implementation and prototyping of selected individual mechanisms that are of primary focus of T3.7.

7.3.1 Class 1 (C1): Entirely new RTOS design and development for Multi-core Mixed-Criticality

This class is elaborated by **CSOFT** whose primary focus is to develop the ***operating system API for managing multi-core mixed criticality tasks*** as basis for the T7.5 use case demonstrator real time operating system. The following figure depicts the high level architecture of this RTOS:



This infrastructure allows multiple tasks to run on a multi-core mixed-criticality real time operating system by isolating two types of tasks. The first type runs in kernel space with direct access to kernel internals, as such, it has the same safety-critical requirements as the RTOS itself. The second type runs in user space through a virtualized memory space and which access to the RTOS is made only through system calls. Being space and time partitioned it’s flexible to distinct safety-critical requirements.

There are three main components to the T3.7 currently being developed.

One is the RTOS interface that provides functionality abstraction from the target hardware. For example, this interface must contemplate:

- Atomic primitives
- Container primitives
- Synchronization primitives
- Mathematical primitives
- Real time clock access
- Hardware timers
- IRQ management
- Device management and interfacing

Second is the RTOS interface with kernel space tasks. It must support:

- Multi-core task deployment, management and scheduling
- Timers
- Synchronization
- Resource sharing

Third is the RTOS interface with user space tasks. This must allow for:

- System calls and virtual memory addressing (through MMU) for space isolation
- Portable C library for application development

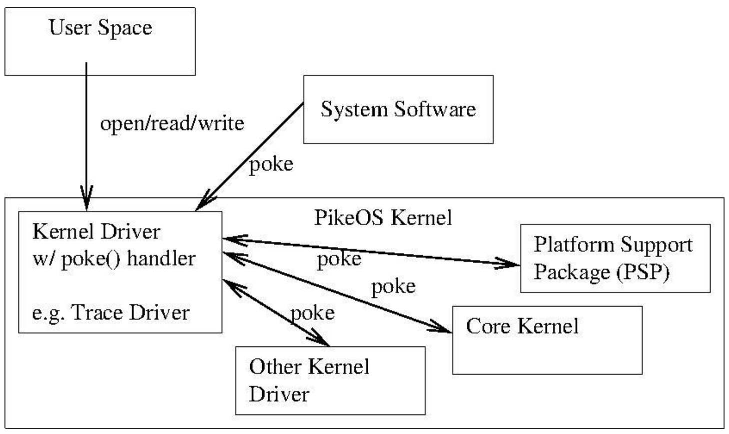
For more details about the design, first prototype and preliminary evaluation results, see the D7.12 deliverable document. The whole technology is in the integration phase now as several partners are involved and participating on.

7.3.2 Class 2 (C2): Existing RTOS enhancements

One of the important mechanism of this cluster is the ***Lightweight tracing capability for multicore PikeOS-based applications***, driven by **SYSGO.** It deals with a mechanism that will allow tracing events in Symmetric multiprocessing (SMP) systems running the PikeOS and that will be possible to run even in the certified production code. The main challenges are to ensure a large overhead on the system is not imposed when debugging is disabled and also to ensure traced events in the system are logged efficiently even when coming from multiple cores at the same time.

When comparing the PikeOS and the Linux tracing architecture, the Linux kernel can dynamically patch its code segment to insert specific trace points into a production kernel. Consequently, the production kernel is practically free of any overhead when tracing is disabled. This technique, however, would not work with PikeOS, because SYSGO's microkernel is certified, and in such a system, code patching techniques are completely undesirable. Even the presence of disabled subroutines that could patch the code poses a great danger of making certification more expensive or even infeasible. For this reason, SYSGO looks into a very lightweight way to check conditions at run-time. The idea is to have a very simple `'if(codition) do_trace()'` instrumentation in the code. After much evaluation, SYSGO finally settled for a new low-level generic function call called 'poke', which allows all parts of the kernel and all system partitions (including the PSSW) to directly, and very quickly enter a kernel driver that installs a 'poke' handler. The lookup of the handler is done statically, so that the actual invocation of 'poke' in the kernel is very fast, with almost no overhead if no handling driver is present. This way, the poke invocations can be left in the standard kernel code without much impact. The mechanism was found to also be usable for another task SYSGO is working on in EMC2, namely the EDF scheduling. The 'poke' mechanism is used here to have a very fast plug-in mechanism for the scheduler so that it is not impacted much if no special scheduling driver is present, but still allow a kernel driver to plug in generically in to the PikeOS scheduling.

The 'poke' system was first published in Q1-2016 with the release of PikeOS 4.1. The following figure shows how the 'poke' mechanism allows very lightweight driver entries from kernel and system software. For communicating with user space (e.g. to tunnel out trace events), the same driver can implement the normal I/O calls for open/read/write.



7.3.3 Class 3 (C3): Inter-Core communication support for existing RTOSes

There are two mechanisms evolved in this class, both ensuring the inter-core communication. While the inter-core communication for Tresos Safety OS is the proprietary one and dedicated just for this particular OS, the Remote Processor Messaging (RPMsg) concept is generic and allows inter-core communication between cores running different operating systems.

The RPMsg protocol defines a standardized binary interface used to communicate between multiple cores in a heterogeneous multicore system. There is the RPMsg implementation in the upstream Linux as well as another open source implementation being part of the Open Asymmetric Multi Processing (OpenAMP) framework (https://github.com/OpenAMP/open-amp). This implementation of the OpenAMP is intended to be portable to different environments (OSes) and target platforms, however there are several limitations of this solution when running in an RTOS environment:

- Received data processing in the interrupt context
- Blocking receive API is missing
- Zero-copy support is missing
- Memory footprint not suitable for multicore MCU parts with limited memory resources

To overcome this limitation of the RPMsg implementation of OpenAMP the ***RTOS aware extension*** has been designed and implemented that allows processing of received data outside the interrupt context and keeping the binary protocol compatible with Linux. The RTOS extension has been already proposed to the OpenAMP project community. The first NXP release that includes the RPMsg port for i.MX6SoloX can be found here: http://www.nxp.com/products/microcontrollers-and-processors/arm-processors/i.mx-applications-processors-based-on-arm-cores/i.mx-6-processors/i.mx6qp/i.mx-6-series-software-and-development-tool-resources:IMX6_SW#bsp

The following RTOS environment porting layers have been created for the RPMsg:

- FreeRTOS (primary)
- Freescale MQX (secondary)

Except of the RTOS aware extension the ***RPMsg-Lite implementation is being prepared*** by Freescale/NXP to offer a code size reduction, API simplification, improved modularity and still keeping protocol compatibility with the Linux implementation. Small MCU-based systems often do not implement dynamic memory allocation. The creation of static API in RPMsg-Lite enables another reduction of resource usage. Not only the dynamic allocation adds another 5kB of code size, but also the communication is slower and less deterministic. Following table shows some rough comparison data between the currently available RPMsg implementation of the OpenAMP and new RPMsg-Lite implementation.

Component / Configuration	Flash [B]	RAM [B]
OpenAMP RPMsg / Release (reference)	5547	456 + dynamic
RPMsg-Lite / Dynamic API, Release	3462	56 + dynamic
Relative Difference [%]	~62.4%	~12.3%
RPMsg-Lite / Static API (no malloc), Release	2926	352
Relative Difference [%]	~52.7%	~77.2%

Until recently, the RPMsg protocol was not documented and its only definition was given by the Linux Kernel and the OpenAMP implementations. This changed with the new initiative driven by the Multicore Association and several involved companies incl. Freescale/NXP which is a standardization of the protocol, allowing multiple different implementations to coexist and still be compatible mutually.

7.4 Planned deployment and usage by partners

The deployment of T3.7 outputs and the cooperation with other EMC2 partners can be demonstrated by links to following Living Labs (LL):

WP7 (Automotive applications): The entirely new multi-core mixed-criticality real time operating system will be presented in a WP7 task T7.5 where it will manage all interfaces between tasks and RTOS resources. Tresos Safety OS updated by IPC is planned to be part of the T7.1 demonstrator.

WP8 (Avionics applications), WP9 (Space applications): The AICAS application framework will be available for demos in these areas as well as for Automotive and IoT. The framework is being ported to run on SYSGO's PikeOS, so this light weight virtualization can also be combined with heavy-weight OS virtualization for criticality separation to ease certification for multi-critical systems as per DO-178C.

WP10 (Industrial manufacturing and logistics): Freescale/NXP plans to offer the developed multi-core sw enablement consisting of IPC for RTOSes and the eRPC implementation that uses the RPMsg / RPMsg-Lite as the transport layer to several Living Labs. The primary target is the incorporation into the T10.1 demonstrator for electric motor control. The goal is also to make the RPMsg RTOS aware extension publically available to all partners and the broader community. Because it deals with a generic component it could be demonstrated on any dual- or multi-core platform running bare-metal or an RTOS application. There is also a plan to create the demonstrator "Embedded Remote Procedure Calls (eRPC) over the Remote Processor Messaging (RPMsg) library".

SYSGO will propose the light weight tracing technique for multicore PikeOS-based applications to different Living Labs in order to get feedback and to raise the interest for the technique. The 'poke' system was first published in Q1-2016 with the release of PikeOS 4.1.

7.5 Evaluation of solutions against EMC2 goals

The T3.7 task addresses requirements coming from WP3 but also from WP1, WP4 and WP7,8,10 (LL). The design and the implementation of the RPMsg corresponds to the HL-REQ-WP1-005 which demands for the reusability across different application domains, then it corresponds to the HL-REQ- WP3-007 which requires synchronized access from different applications to shared resources and it also addresses the HL-REQ-WP3-008 requirement for the interconnection of hardware and software mechanisms. From the perspective of the automotive application the RPMsg fulfills requirements for the effective portability (HL-REQ-WP07-004), support for different operating systems (HL-REQ-WP07-032) and uniform function interface (HL-REQ-WP07-034) and the use of available and open standards (HL-REQ-WP07-058).

The development of the multi-core mixed-criticality real time operating system addresses the HL-REQ-WP07-033 (provide a mixed-criticality task management), HL-REQ-WP07-034 (provide a multi-core task management) and HL-REQ-WP07-037 (provide multi-core mixed-criticality task management and the necessary interfaces for their interaction with the RTOS kernel) requirements by providing the necessary interfaces for creating, managing and scheduling of mixed-criticality tasks into a multi-core hardware platform.

The requirement for Mixed Criticality Support (HL-REQ-WP07-027) is fulfilled by the design of the inter-core communication library for Tresos Safety OS. The OS supports mixed criticality software by separating the entities spatially and on separate cores. It provides defined and safe communication channels between the entities on the same and on different cores via the IOC module. This mechanism also covers the HL-REQ-WP07-044 requirement (Real Time Data Communication for safety critical and safety relevant applications) by providing priority based scheduling algorithms.

Lightweight tracing aims to provide a separated module that does not interfere with operation in a production system in order to reduce costs by making the development system and the production system as similar as possible and thus addressing the HL-REQ-WP03-003 requirement for encapsulation.

8. Conclusions

This deliverable summarizes results of the work that was performed within WP3 until M28. The main goal was to utilize dynamic control during runtime in order to support mixed-critical systems as well as security techniques, safety and real-time properties in existing safety critical architectures/RTEs. The preceding chapters provide an overview of inputs gathered from partners in scope of ttasks T3.2 – T3.7, together almost ~80 individual contributions. Mechanisms (more than 30) were divided into six technology clusters (tasks). The work in each cluster was organized as follows: (1) summary and further refinement of solutions/platforms/architectures proposed in the D3.3; (2) implementation of solutions/platforms/architectures proposed in the D3.3; (3) analytic and experimental evaluation of the proposed solutions against the project requirements and goals of the WP3.

The inputs for each task (T3.2-T3.7) are structured with a summary of the most important features of the introduced mechanisms. The summary is followed by an overview of the implementation and evaluation steps conducted until the PM 28. Finally, the planned deployment and usage by partners is described along with most important requirements fulfilled by the mechanisms.

The detailed description of work done by partners is attached at the appendixes (9.4 - 9.9) of this document. The input for each of the partners consists of two sections: the short summary of the achievements and work and the technical annex containing all inputs in form of technical reports and publications. Moreover, in the Appendix 9.2 cockpit charts are included followed by the full cooperation matrix in Appendix 9.3 presenting all registered mechanisms and progress of their transfer to particular use cases in LLs (WP7-W12).

In WP3, an iterative work approach is assumed, as depicted in Figure 1. Therefore, the presented results constitute an initial input for the next deliverable D3.6. The initial implementations should be later supported with results from reference platforms and fully adjusted to provide support for the considered hardware architectures (including WP4). Therefore, inputs presented in D3.4 will be improved, aligned and extended in D3.6 (PM36).

The gathered inputs and presented outcomes constitute the basis for fulfilling the objectives of WP3 and shall support Living Labs in identifying the technologies and relevant partners fulfilling their needs.