

ACROSS

**ACROSS
ARTEMIS Cross-Domain
Architecture**

***Specification of Industrial Control Application
Demonstrator WP6
D 6.6***

Project Acronym	ACROSS	Grant Agreement Number	ARTEMIS-2009-1-100208	
Document Version	1.0	Date	2010-09-29	Deliverable No. 6.6
Contact Person	Majid Ghameshlu	Organisation	Siemens AG Österreich	
Phone	+43 664 8011737613	E-Mail	Majid.ghameshlu@siemens.com	

Change History

Version No.	Date	Change	Author(s)
01	2010-07-26	Initial version	Majid Ghameshlu
02	2010-08-09	Contributions from DICE & fortiss	Brandstätter, Huang
03	2010-08-16	Further updates from SAGÖ, DICE	Brandstätter, Ghameshlu Hinterstoisser
04	2010-08-23	Further updates from SAGÖ, DICE	Brandstätter, Ghameshlu Hinterstoisser
05	2010-08-30	Further updates from SAGÖ	Ghameshlu Hinterstoisser
06	2010-09-06	Further updates from SAGÖ, Verimag, Input for ACROSS Executive Board review	Ghameshlu, Griesmayer, Hinterstoisser
07	2010-09-13	Further updates from SAGÖ	Ghameshlu, Hinterstoisser
1.0	2010-09-29	Final Version	Ghameshlu,

Contributors

Name	Partner	Part Affected	Date
Siegfried Brandstätter	DICE	all	continuous
Jia Huang	fortiss	all	continuous
Majid Ghameshlu	SAGÖ	all	continuous
Thomas Hinterstoisser	SAGÖ	all	continuous
Andreas Griesmayer	Verimag	all	continuous

Table of Contents

Change History	2
Contributors	2
1 About this Document	5
1.1 Role of the Deliverable.....	5
1.2 Relationship to Other ACROSS Deliverables	5
1.3 Structure of this document.....	5
1.4 Glossary / Definitions	5
2 Industrial Control Application Demonstrator	7
2.1 Demonstrator Overview	7
2.2 Demonstrator Hardware Description	9
2.2.1 Mechanical Description.....	9
2.2.2 Electrical Description.....	10
2.3 Demonstrator Software Description.....	19
2.3.1 Application SW Workflow	19
2.3.2 Application SW	19
2.3.3 IO CPU SW	22
2.4 General Model	24
2.4.1 Behavioral Model	25
2.4.2 Combined Model and Applications	27
3 Wireless Communication Application Demonstrator	28
3.1 Demonstrator Overview	28
3.2 Demonstrator Hardware Description	31
3.2.1 Mechanical Description.....	31
3.2.2 Electrical Description.....	31
3.3 Demonstrator Software Description.....	33
3.3.1 Application SW	33
3.3.2 Interaction with core, generic and domain specific services	36
A) Annex: Refined Requirements	38
B) Annex: Platform Requirements	45
C) Annex: Bibliography	46

Table of Figures

Figure 1: typical PLC structure	7
Figure 2: WP6 Industrial Control Demonstrator	9
Figure 3: WP6 Industrial Control Demonstrator: mechanical structure	10
Figure 4: WP6 Industrial Control Demonstrator: option A	12
Figure 5: WP6 Industrial Control Demonstrator: option B	13
Figure 6: WP6 Industrial Control Demonstrator: option C	14
Figure 7: architecture of Industrial Control Demonstrator based on the STRATIX IV GX 230 Development Board.....	15
Figure 8: Block Diagram of the HSMC Prototype Board	16
Figure 9: 5 Volt power supply and level shifter on HSMC Prototype Board.....	17
Figure 10: Application connector of HMS Anybus Profibus Master Card.....	18
Figure 11: Application connector Pinout of HMS Anybus Profibus Master Card	18
Figure 12: Demonstrator Software Workflow	19
Figure 13: Flow chart of application software	22
Figure 14: Interaction between Application Software and Services.....	22
Figure 15: Flow chart of IO CPU application	23
Figure 16: Interaction between IO Software and Services	24
Figure 17: Physical interface of the slide controller	25
Figure 18: State machine of the Slide Controller	26
Figure 19: Library Models	27
Figure 20: Combined Model	28
Figure 21: Hardware block diagram of a wireless communication system.	29
Figure 22: Basic setup of the wireless communication application demonstrator.	29
Figure 23: MPSoC for wireless communication application demonstrator (Version A).	30
Figure 24: MPSoC for wireless communication application demonstrator (Version B).	30
Figure 25: Detailed setup of the wireless communication application demonstrator.....	31
Figure 26: Interconnection over a DigRF V1.12 digital interface.	32
Figure 27: Interconnection over a DigRF V1.12 digital interface including additional signals.	33
Figure 28: Allocation of CPUs within the ACROSS MPSoC.	33
Figure 29: Time flow of single test cycle.	34
Figure 30: Flow chart of typical transmit test scenario.	35
Figure 31: Interaction between components involved in a test scenario.	36
Figure 32: Interaction between components involved in the pre- and post-processing.	37

1 About this Document

1.1 Role of the Deliverable

This deliverable specifies and defines the Industrial Control application demonstrator based on the requirements identified in the task T 6.1. An appropriate application will be selected to demonstrate the outstanding solutions for requirements out of the field of safety, interoperability and effectiveness.

1.2 Relationship to Other ACROSS Deliverables

The inputs for this deliverable are:

- Requirements identified in T 6.1 for the Industrial Control domain

This document is the base for the design of the industrial control application demonstrator.

1.3 Structure of this document

Subsequent to this introductory section the two application demonstrators “industrial control” and “wireless communication” are specified. Both demonstrators are specified in an unit manner and contain:

- Demonstrator overview
- Demonstrator hardware description
 - Mechanical
 - Electrical
 - Input interfaces (e.g. sensor)
 - Output interfaces (e.g. actuator, signaling)
 - Debug interface
- Demonstrator software description
 - Application SW
 - Interaction with core, generic and domain specific services

Annex A contains the refined version of requirements defined in deliverable D 6.1.

1.4 Glossary / Definitions

The following definition is taken from the book GENESYS, the output document of the GENESYS project. The GENESYS project is the precursor of the ACROSS project.

Platform

A platform is the hardware/software foundation for the execution of applications. The platform comprises generic services for the development of applications, which are denoted as platform services.

API	Application Programming Interface
DSS	Domain Specific Services
ECC	Error Correction Code
ET	Event Triggered
FIFO	First In First Out
GUI	Graphical User Interface
IP	Intelligent Property
MPSoC	Multi Process System on Chip
POSIX	Portable Operating System Interface
PLC	Programmable Logic Controller
Si-Area	Silicon Area
SW	Software
TTNoC	Time-Triggered Network on Chip
WP	Work Package

2 Industrial Control Application Demonstrator

2.1 Demonstrator Overview

The Industrial Control Application Demonstrator should highlight the deployment capability of the ACROSS MPSoC platform for industrial PLC applications. Figure 1 shows the simplified structure of a cyber physical system with PLC application running on the Control Unit. PLC's are used nowadays for e.g. process control, motion control. Industrial fieldbuses like Profibus or Profinet are often used for the communication between sensors/actuators and control unit. Several standard (IEC 61131-3) based programming languages like Step7 of Siemens build the frontend for the programming of the PLC system. They offer text based languages (like "Structured Text" or "Instruction List") and/or graphical based languages (like "Ladder Diagram" or "Functional Block Diagram") to program the PLC application. Also the "Sequential Function chart" language has means to organize programs for sequential and parallel control processing.

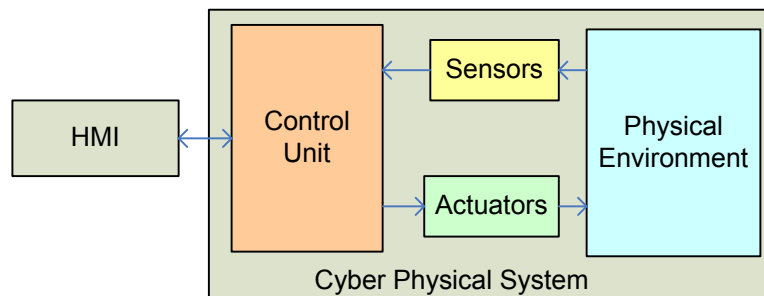


Figure 1: typical PLC structure

The WP6 demonstrator consist of several Festo MPS station with the following functionalities:

- The **Robot** station can transport workpieces that are fed via a slide and place them in an assembly retainer. The sensor in the gripper enables the robot to differentiate workpieces by colour (black/non-black). The sensor in the assembly retainer monitors the orientation of the workpiece. From the assembly retainer the robot sorts the workpieces into various magazines or passes them on to the downstream station. Combination with the assembly station facilitates the assembly of workpieces.
- The **Testing** station detects the various properties of the workpieces inserted into it. It differentiates workpieces with the aid of an optical and a capacitive sensor. A retro-reflective sensor monitors whether the operating space is free before the workpiece is raised via a linear cylinder. An analogue sensor measures the height of the workpiece. A linear cylinder guides correct workpieces via the upper air slide to the neighboring station. Faulty workpieces are rejected via the lower air slide
- In the **Processing** station, workpieces are tested and processed on a rotary indexing table. This station only uses electrical drives. The rotary indexing table is driven by a DC motor. The table is positioned by a relay circuit, with the position of the table being detected by an inductive sensor. On the rotary indexing table, the workpieces are tested and drilled in two parallel processes. A solenoid probe with an inductive sensor checks that the workpieces are inserted in the correct position.

During drilling, the workpiece is clamped by a solenoid actuator. Finished workpieces are passed on via the electrical sorting gate.

- The **Handling** station is equipped with a flexible two-axis handling device. Workpieces inserted into the holder are detected by an optical diffuse sensor. The handling device picks up the workpieces from there with the aid of a pneumatic gripper. The gripper is equipped with an optical sensor which differentiates between “black” and “non-black” workpieces. The workpieces can be placed on different slides according to this criterion. Other sorting criteria can be defined if the station is combined with other stations. Workpieces can also be transferred to a downstream station.
- The **Pick&Place** station is equipped with a two-axis Pick&Place module. Workpiece housings placed on the conveyor are detected by an optical diffuse sensor. The workpiece is transported to the pneumatic separator on the conveyor and detected by a second diffuse sensor. The Pick&Place module picks up a workpiece insert from the slide and places it on the workpiece housing. The complete workpiece (housing and insert) is released by the separator and transported to the end of the conveyor. A light barrier detects the workpiece at the end of the conveyor.
- The **Storing** station places workpieces in and takes workpieces out of storage. The station is equipped with three storage levels, with a level each for six red, six silver and six black workpieces. The workpieces are gripped using a pneumatic gripper. The linear movement is executed using a linear cylinder. The rotary movement is performed by an electrical servo drive with integrated controller. The stroke movement is executed using an electrical linear axis with separate controller.
- The **Sorting** station sorts workpieces onto three slides. Workpieces placed on the start of the conveyor are detected by a diffuse sensor. Sensors upstream of the stopper detect the workpiece features (black, red, metal). Sorting gates actuated by short-stroke cylinders via a deflector allow sorting of workpieces onto the appropriate slides. A retro reflective sensor monitors the level of the slides. The station is equipped with a CP valve terminal with Profibus-DP interface (DP slave). A PLC with Profibus-DP master functionality is required for operation.

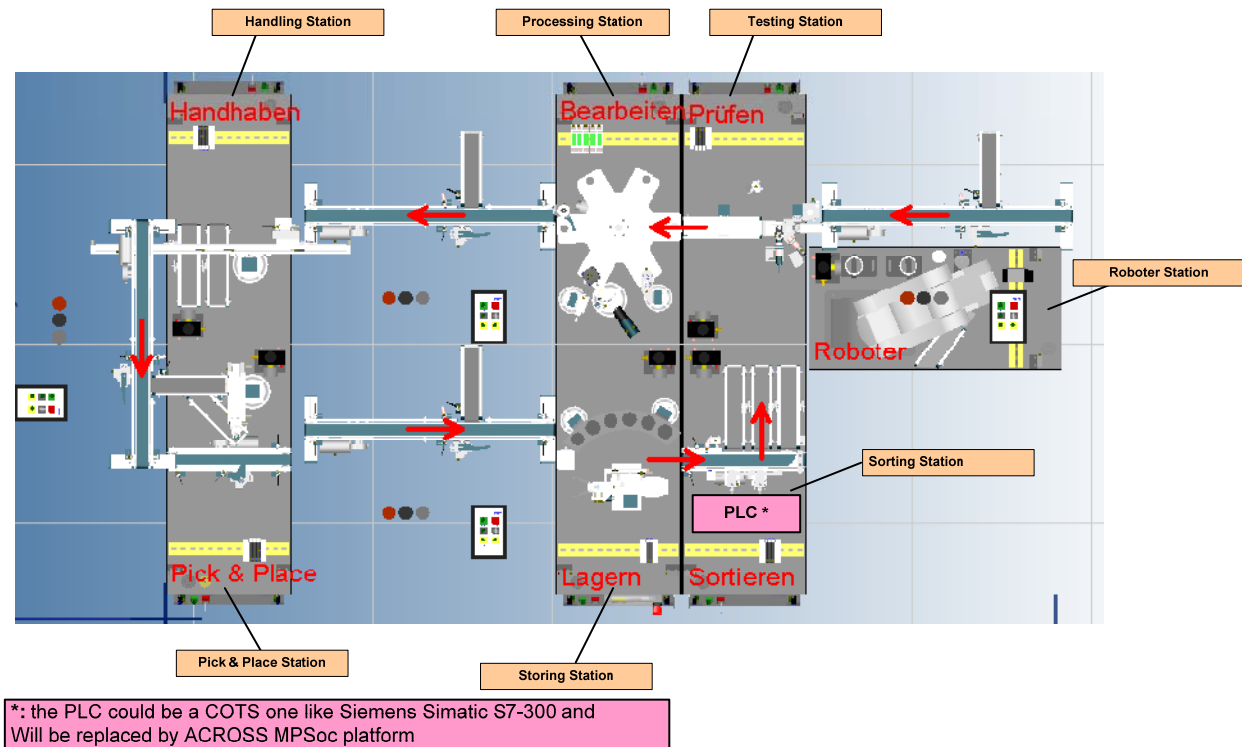


Figure 2: WP6 Industrial Control Demonstrator

The Figure 2 shows the arrangement of these stations. To show the capabilities of the ACROSS MPSoC platform the PLC of the **Sorting** station (COTS platform like Siemens Simatic S7-300) will be replaced by the ACROSS platform.

2.2 Demonstrator Hardware Description

2.2.1 Mechanical Description

The demonstrator consists of the following parts (as shown on Figure 3)

1. MPSoC platform with on board Stratix FPGA containing the MPSoC architecture
2. HSMC prototype board connected to MPSoC platform by HSMC connector
3. Profibus master card connected to HSMC prototype board by the application connector. It is a 2mm low profile strip header that can be mounted on either side of the card.
4. profibus cable between master card and the Sorting station of the Festo MPS equipment.

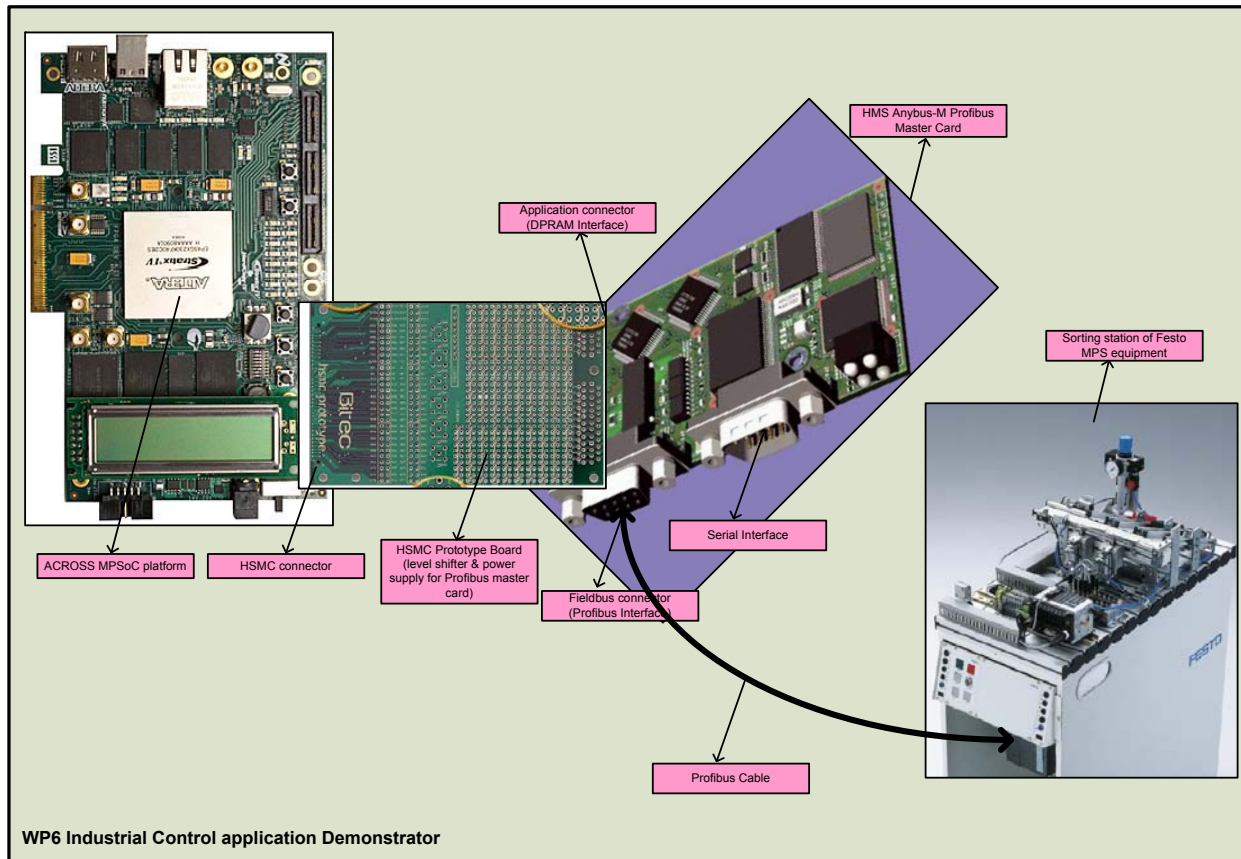


Figure 3: WP6 Industrial Control Demonstrator: mechanical structure

2.2.2 Electrical Description

2.2.2.1 Input interfaces (e.g. sensor)

Following sensors are inputs to the Profibus Master Card:

- Workpiece available
- Workpiece is metallic
- Workpiece is not black
- Slide is full
- Switch 1 is retracted
- Switch 1 is extended
- Switch 2 is retracted
- Switch 2 is extended

One of the above sensors or a new “to be defined” sensor will be used as input for the Plausibility Safety Service (DSS1_IC).

The input to the control lamp (System Shut Down) is read back to the MPSoC platform and analyzed by the “Switch Off Channel Service” (DSS3_IC), if not having the intended value the safety output is turned on by the service.

2.2.2.2 Output interfaces (e.g. actuator, signaling)

Following sensors are inputs to the Profibus Master Card:

- Extend Switch 1
- Extend Switch 2
- Retract Stopper
- Signal station as occupied
- Turn on the Conveyor Motor

A control lamp will be turned on by the “Switch Off Channel Service” (DSS3_IC) to indicate “System Shut Down”. (General Purpose Output)

A safety output controlled by the “Switch Off Channel Service” (DSS3_IC) will be turned on if the message “System Shut Down” is not transferred safely to the control lamp. (General Purpose Output)

2.2.2.3 Debug interface

The embedded application and IO CPU are debugged through the JTAG interface of the used FPGA platform of the ACROSS MPSoC. This is the standard debug interface of the platform.

The Debug interface shall also offer means to manipulate the input data to the Incoming Voting Service of the application CPUs to verify the TMR redundant concept. If not possible DIP Switches shall be used.

A DIP Switch is used to invert the signal to the control lamp “System Shut Down” to verify the “Switch Off Channel Service”.

2.2.2.4 Functional Description

Following figures show the three concepts analyzed for the WP6 Industrial Control Application demonstrator:

- Option A:** three redundant sensor systems act as inputs for the three redundant IO CPUs. The data are transferred over the TTNOC to the three application CPUs with separate Incoming Voter. The calculated actuator control data are sent via the IO CPU1 to the Festo MPS station. The Plausibility safety service running on all application CPUs checks the plausibility of the input data (e.g. safe position) and switch the system to a safe state (e.g. System Shut Down).

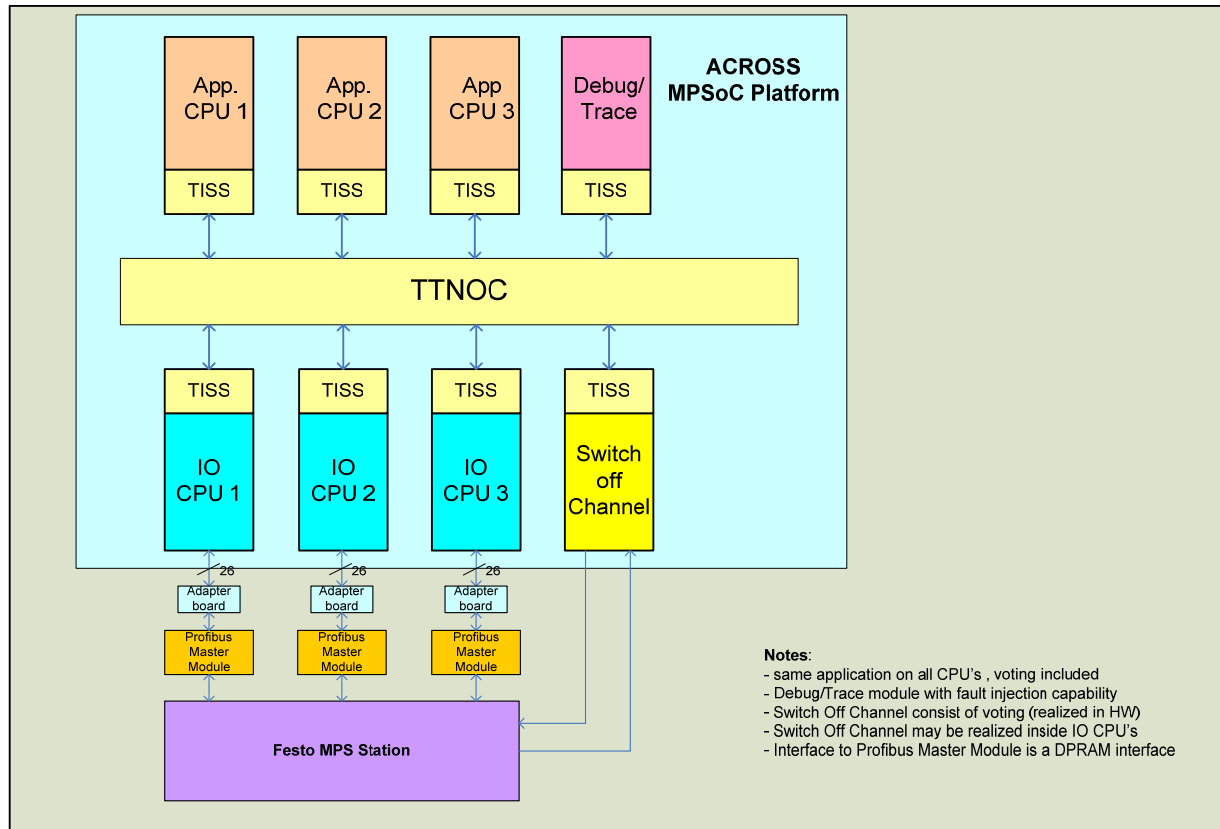


Figure 4: WP6 Industrial Control Demonstrator: option A

- **Option B:** the same as **Option A**, but there is only one sensor system, the sensor input data are fed into ACROSS MPSoC platform via separated physical channels (FPGA pins).

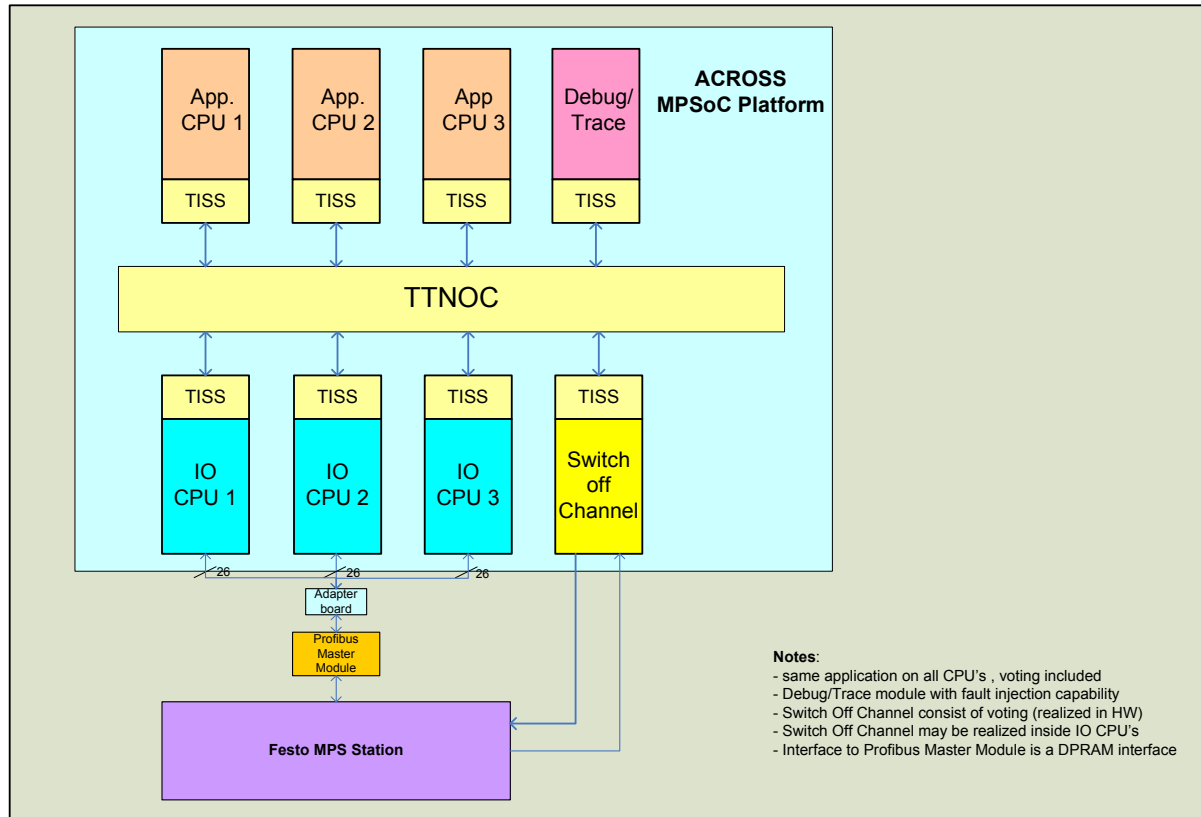


Figure 5: WP6 Industrial Control Demonstrator: option B

- Option C:** the same as **Option B**, but using only one IO CPU and showing the “System Shut Down” feature on the Adaptor Board to simplify the demonstrator and reach the goals regarding schedule and costs. The “Switch Off Channel” may be realized inside IO CPU using a domain specific service. If the IO CPU is available as a full soft macro (e.g. in VHDL) an online BIST could be realized to show an alternative solution to option A and B regarding redundant IO CPU’s.

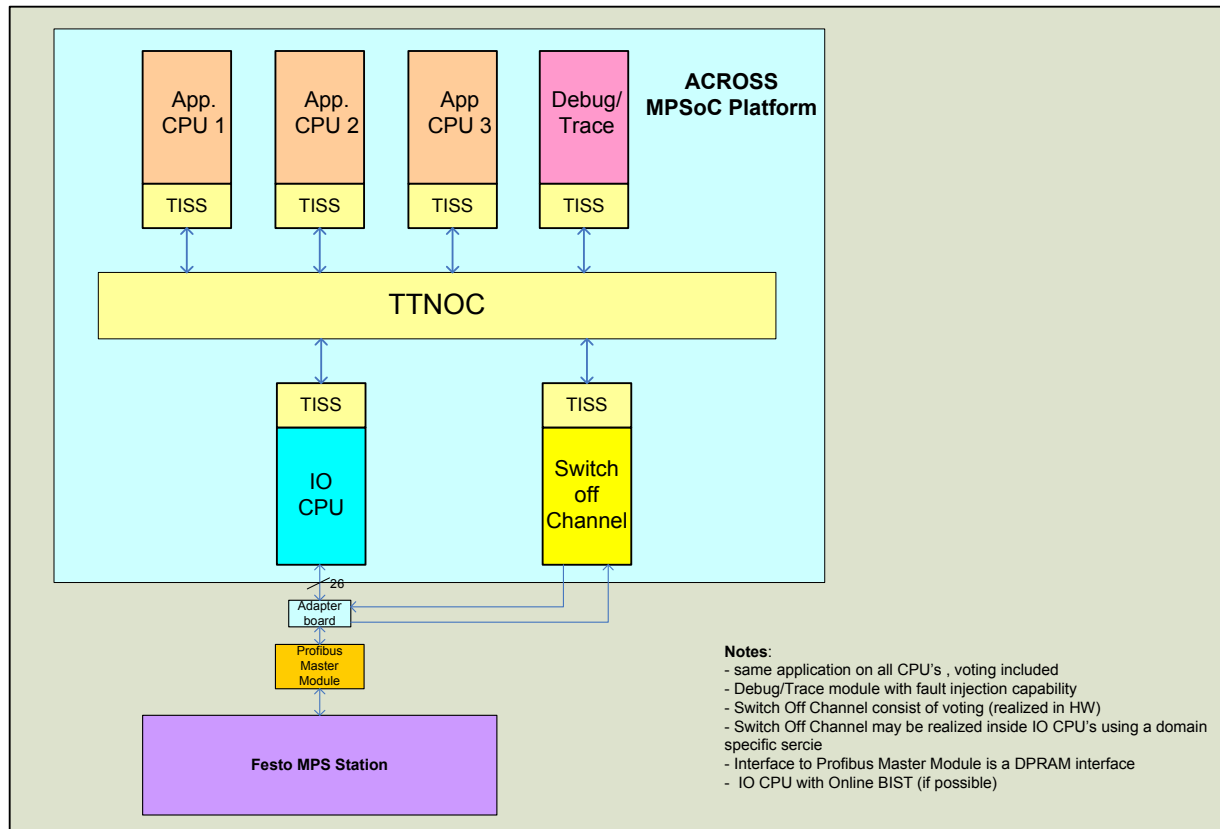


Figure 6: WP6 Industrial Control Demonstrator: option C

The main goal of the Industrial Control Application demonstrator is to show the capabilities of the ACROSS MPSoC platform replacing the Control Unit of the SPS based systems. To meet the project boundary conditions the demonstrator part containing the sensors and the communication means (e.g. Profibus Master Card, IO CPU) is considered as a safe one and therefore the **Option C** is selected for this application demonstrator. Since the FPGA platform does not offer the possibility of having safety compliant silicon areas with e.g. separated power supply all connection of the Switch Off Channel are implemented inside the same silicon area.

Also the small circuit for the demonstration of System Shut Down Feature is placed on this board. Separate DIP switches support a failure injection by using the GPIO ports, which directly influence the Profibus functionality.

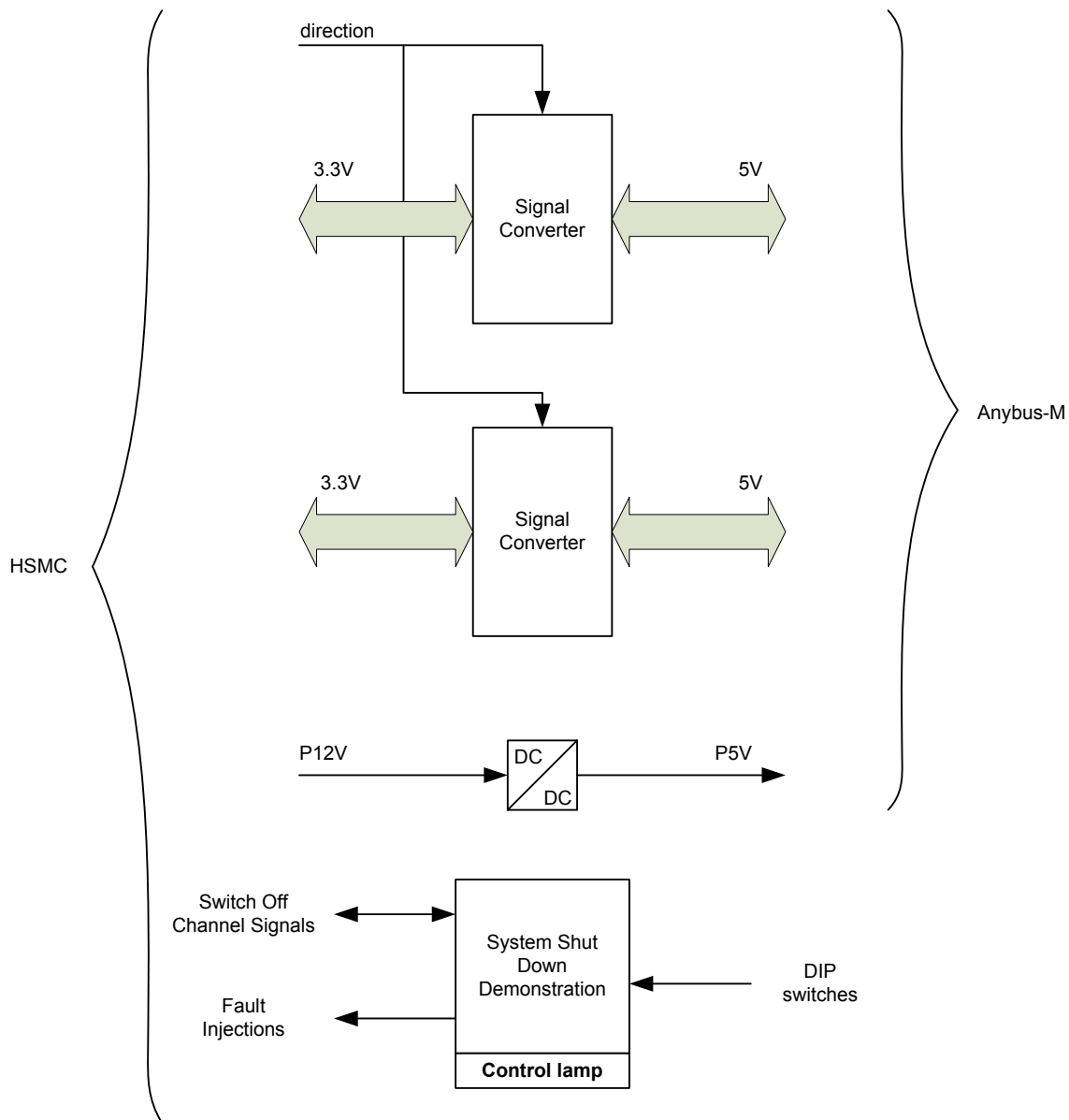


Figure 8: Block Diagram of the HSMC Prototype Board

For the signal conversion bi-directional components are used, which converts the 3.3V CMOS standard to the 5V TTL standard. The signal direction is controlled by the r/w signal. An additional DC/DC converter provide the +5V (derived from the +12V from the HSMC connector) for the TTL signal standard and the for the power supply for the Anybus-M board.

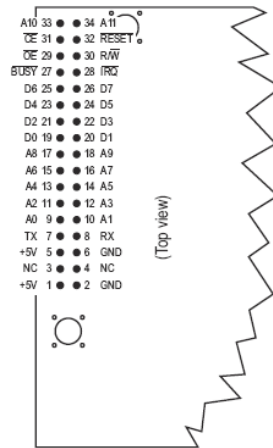


Figure 10: Application connector of HMS Anybus Profibus Master Card

Note: All signals are TTL level unless otherwise stated.

Pin	Name	Description	Direction	Note
1	+5V	VCC	Input	Power Supply, bus interface. See D-1 "Power Supply Requirements".
2	GND	Ground	-	
3, 4	NC	Isolation distance	-	(Not connected)
5	+5V	VCC	Input	Power Supply, module electronics. See D-1 "Power Supply Requirements".
6	GND	Ground	-	
7	TxD	Transmit Data ^a	Output	Asynchronous serial interface transmit ^a
8	RxD	Receive Data ^a	Input	Asynchronous serial interface receive ^a
9 - 12	A ₀ - A ₃	Address Inputs	Input	Address lines 0 ... 3
13	A ₄	Address Inputs ^a	Input	Address line 4 ^a
14 - 18	A ₅ - A ₉	Address Inputs	Input	Address lines 5 ... 9
19 - 26	D ₀ - D ₇	Data Input / Output	Bidirectional	Databus, bits 0 ... 7
27	BUSY	Busy Signal ^a	Output	Active low open collector output ^a
28	IRQ	Interrupt Request ^a	Output	Active low open collector output ^a
29	OE	Output Enable	Input	Active low input
30	R/W	Read/Write	Input	Active low input
31	CE	Chip Enable ^a	Input	Active low input ^a
32	RESET	Reset	Input	Active low input. Internally pulled up with 35 - 75kΩ.
33	A ₁₀	Address Input ^a	Input	Address line 10 ^a
34	A ₁₁	Address Input ^b	Input	Address line 11 ^b

a. Internally pulled up with 10kΩ.

b. This signal is used on some Anybus modules to accommodate a larger dual port memory. On those modules, this pin is internally pulled up with 10kΩ. For more information, see A-1 "Extended Memory Mode (4K DPRAM)". Note that the use of this pin is optional. If not used, it must be left unconnected or pulled to VCC.

Figure 11: Application connector Pinout of HMS Anybus Profibus Master Card

2.3 Demonstrator Software Description

2.3.1 Application SW Workflow

The application software used for WP6 demonstrator will be developed using a workflow presented in Figure 12.

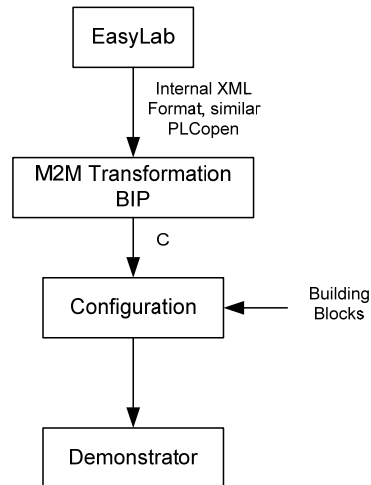


Figure 12: Demonstrator Software Workflow

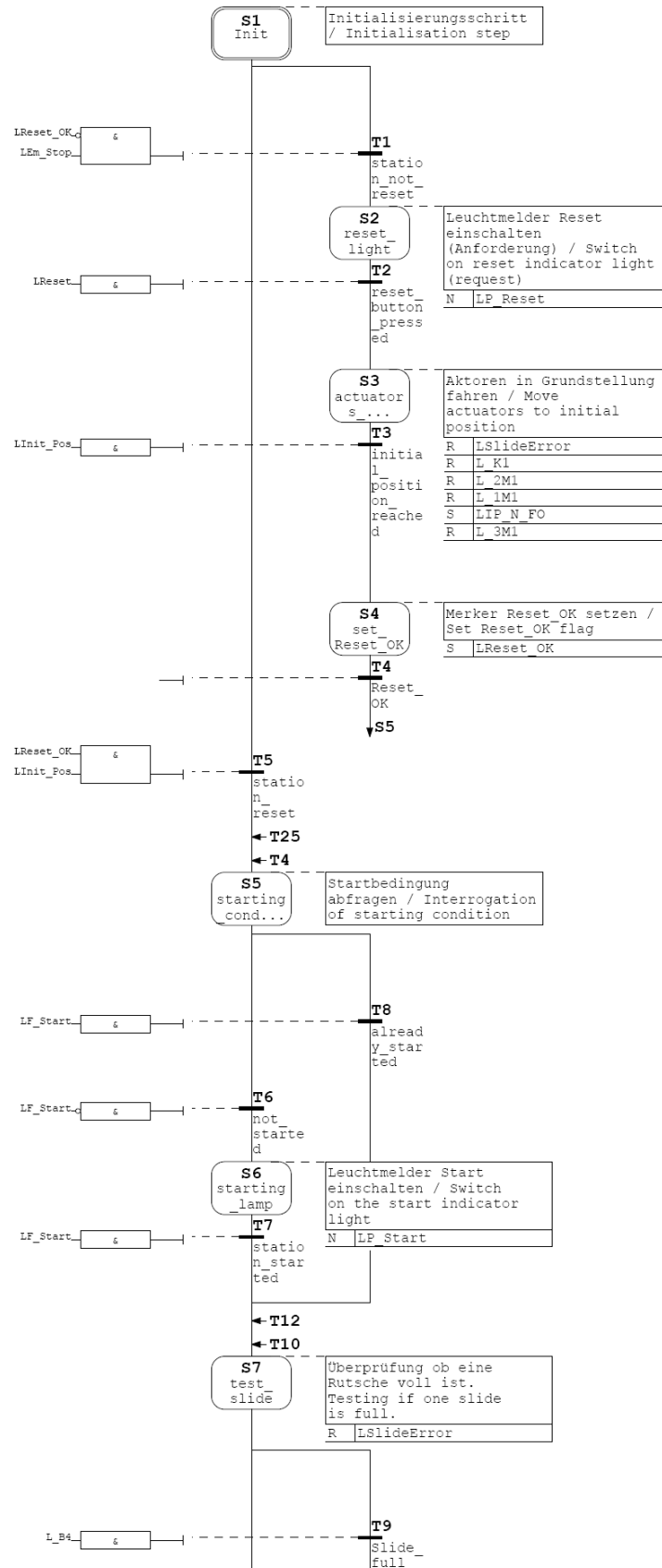
On the top level, the application software will be designed using the domain-specific tool *EasyLab* from fortiss. *EasyLab* is a model-based development tool that supports a subset of standard models defined in IEC-61131 norm. The front end of *EasyLab* features a graphical user interface (GUI) to allow software design using the SFC and FBD languages specified in IEC-61131 standard. The backend of the tool can produce C code for several target platforms.

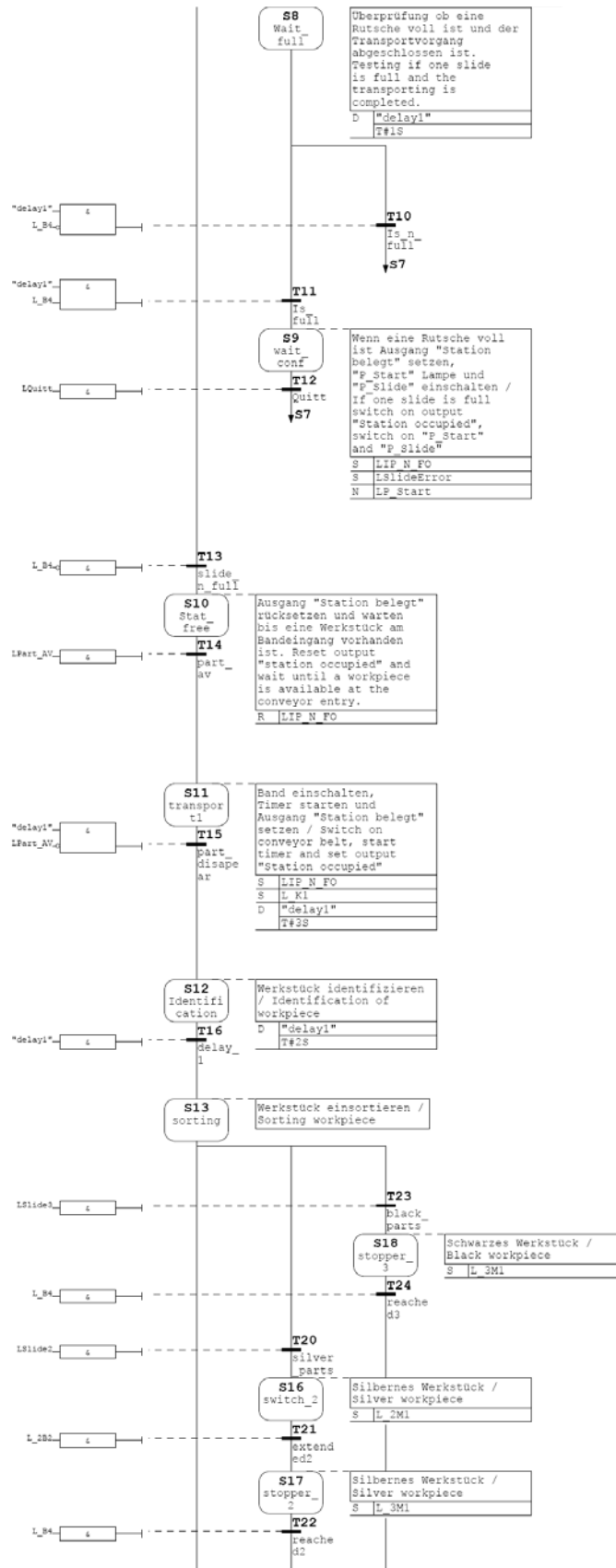
EasyLab is designed to be an extensible framework. In particular, the functional blocks used in software development and the code generation backend can be extended. Within the scope of the ACROSS project, *EasyLab* can be extended if necessary to provide adequate features to model the application of WP6 demonstrator.

Using the model-to-model transformation developed in WP3, the *EasyLab* model can be transformed into the generic model, more precisely, a BIP representation. Special care is put on the transformation to guarantee that important properties are preserved. Using this methodology, existing tools developed for BIP can be reused to formally analyze/verify the model, e.g. to guarantee deadlock freedom.

The verified model will then be used to generate an executable that implements the demonstrator functionality. The generated code will be encapsulated use the POSIX API provided by the middleware. The processes are then distributed to individual cores according to the mapping table specified in the configuration file. The configuration of used services, e.g. core-to-core communication will also be generated.

2.3.2 Application SW





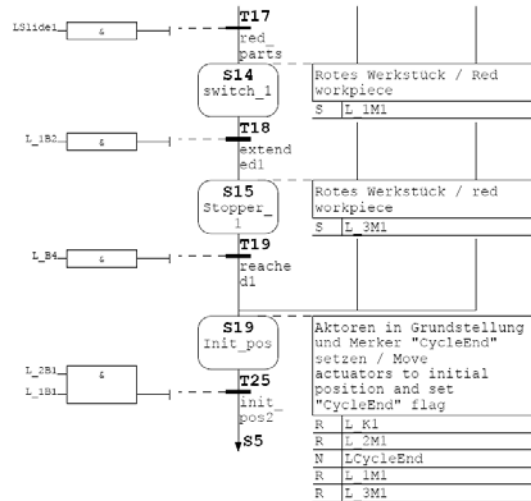


Figure 13: Flow chart of application software

2.3.2.1 Interaction with core, generic and domain specific services

For the application software the core service for TISS communication (“Sporadic Messaging Service” or “Periodic Messaging Service”), the generic service for the incoming data voting (OPT5) and the domain specific service for safety plausibility check (DSS1_IC) are used. All these services can be accessed by the application control software via Application Programming Interfaces (APIs) and running on the same Application CPU. Figure 14 demonstrates the interaction between the device, the application and services.

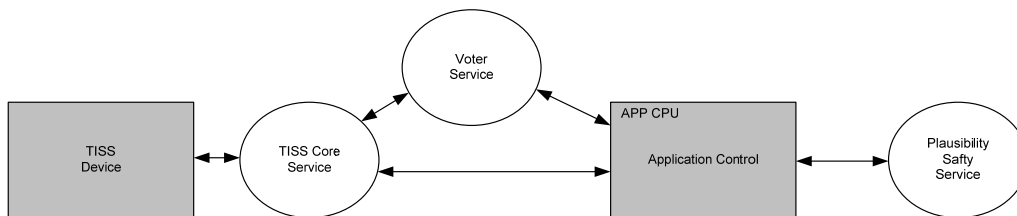


Figure 14: Interaction between Application Software and Services

2.3.3 IO CPU SW

For the communication between a TTNOC (TISS interface) and Profibus module a separate IO CPU is used. After Power On or reset the application software initializes the following components:

- **Setup Interrupt service routine:** Configuration for the interrupt service routines for the Anybus module and the TISS interface. After initialization all interrupts are cleared.

- **Timer Setup:** The timer is used for different transfer timeouts.
- **Anybus configuration:** The initialization sequence determines how the Anybus module will operate on the network, and certain basic operational parameters such as memory layout etc.
- **Anybus database download:** The Anybus module needs a bus database in order to know which slaves to establish connections to, how they shall be configured and how much data to exchange with them. The Profibus network topology is loaded via the mailbox interface.
- **TISS/Voter configuration:** This task configures the TISS interface for the TTNOC communication and set the voting parameters.

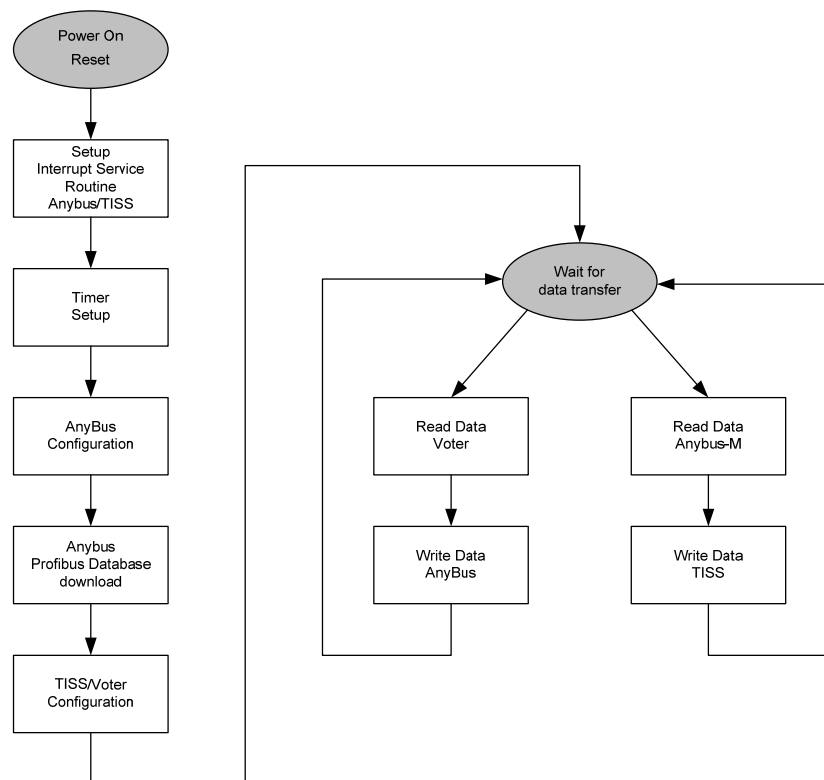


Figure 15: Flow chart of IO CPU application

After Initialization the IO application is ready to transfer data from the Anybus module to the TISS interface and reverse. The handshake mechanism is realized via interrupt controlling. An additional voting service for the incoming data from the TTNOC side handles the decision for the redundant information. The outgoing data to TTNOC are broadcasted to all application CPUs.

2.3.3.1 Interaction with core, generic and domain specific services

Following domain specific services for the Anybus-M module communication are necessary:

- Profibus Interface Service (DSS5_IC)
- Profibus Transfer Service (DSS6_IC)
- Profibus Configuration Service (DSS7_IC)

Two additional domain services provide a safety read operation (DSS2_IC) and a channel switch off operation (DSS3_IC). For TISS communication (“Sporadic Messaging Service” or “Periodic Messaging Service”) and data voting (OPT5) the core/generic services are used. All these services can be accessed by the IO transfer application via Application Programming Interfaces (APIs) and running on the same IO CPU. Figure 16 demonstrates the interaction between the devices, the IO application and services.

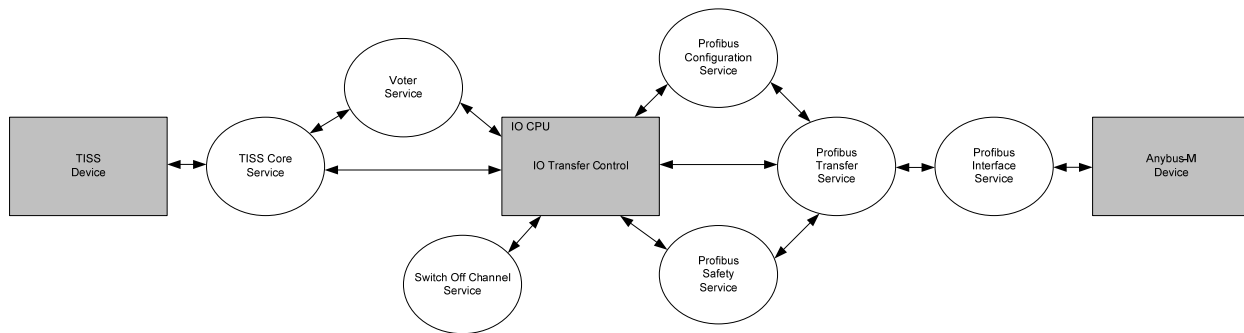


Figure 16: Interaction between IO Software and Services

2.4 General Model

The ACROSS Development Process will define the development of an application from first abstract sketches to detailed integrated models that can be verified, simulated and deployed. The general development process is described in more detail in deliverable D3.2. This section will provide an overview of the development process and its application to the industrial automation domain.

Key to the development process is the use of a library of predefined ACROSS services, which is available to the application developer. These components serve as *building blocks* for new applications and can be interfaced from a domain specific language (DSL). Applications are developed in DSL and translated to the general model, which captures its behavior in an ACROSS tool language, as well as the interfaces and connections with service components. This general model is the starting point for model-to-model transformation to generate the inputs for later analysis and deployment steps. In the following, the behavior of the system will be captured in BIP, which is one of the supported tool languages of the ACROSS development platform. In the future, the task of translating industrial control applications from EasyLab to BIP will be supported by translation tools – for this document, that task is performed manually.

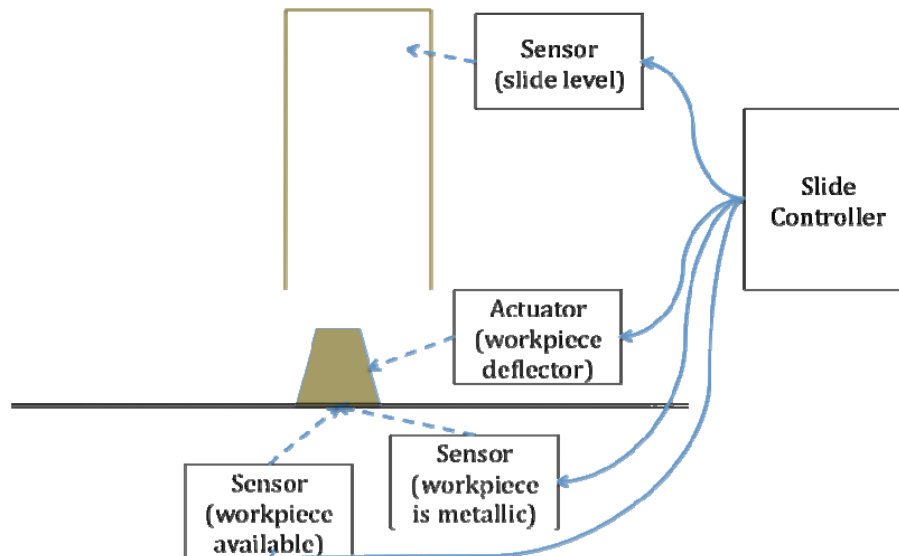


Figure 17: Physical interface of the slide controller

2.4.1 Behavioral Model

In the final development process, the general model of the software will be generated using translation tools. To demonstrate the approach and give some more behavioral specification of the demonstration application, this step is performed manually and presented in the following. The presented models are sketches of the expected results and therefore subject of change. We use BIP to give the behavior of the **sorting** station, which was introduced in the previous chapters.

BIP is a state machine based modeling language for component based systems. A component is a processing entity that is concerned with fulfilling a certain task. It maintains a state that is updated on certain events and communicates with other components during interactions. Applied to the example at hand, the sorting station is a component that interacts with the sensors (work pieces available, switch

retracted etc.) and the actuators (extend switch, etc.) respectively. E.g., an input from the sensor that a work piece is available will change the state of the component such that in turn the action to start the conveyor belt motor will become available. Depending of the level of abstraction, the models are more or less detailed. BIP supports hierarchical systems, which means that the behavior of the sorting station can be composed from smaller modules that allow to model, e.g., conveyor belt and storage slides separately. The same holds true for the sensor inputs and actuator outputs, which can be modeled in great detail to reflect initialization steps and fault handling mechanisms. In this document we assume a very simple model of the sorting station that uses storage slide controllers to check if a work piece is available and should be sorted into a slide and abstract sensors and actuators. Figure1 shows the physical interface of the storage slide controller. It has inputs to indicate if a work piece is available and if it is of the desired type, and an actuator that can push the work piece to the respective slide.

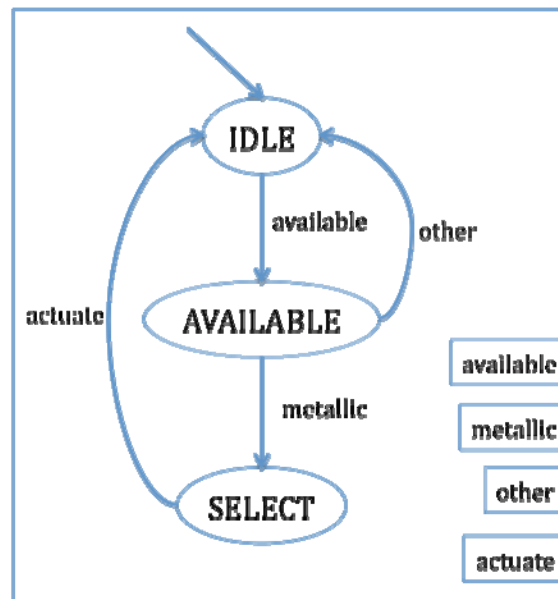


Figure 18: State machine of the Slide Controller

The corresponding state machine is given in Figure 2. The controller starts in IDLE state. When a work piece becomes available it switches to the state AVAILABLE and checks the kind of work piece. For accepted work pieces, the state is switched to SELECT, otherwise the work piece is ignored and the state is set back to IDLE. In state SELECT, the actuator is activated and moves the work piece onto the slide. The controller moves back to IDLE to await the next work piece.

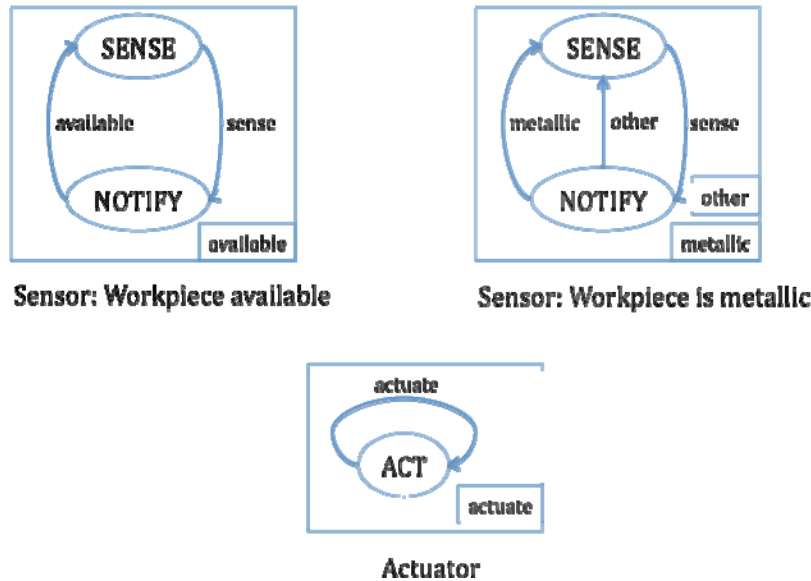


Figure 19: Library Models

The corresponding general models for the Sensors and Actuators are given in Figure 3. General models for sensors and actuators are supposed to be maintained in the building block library and don't have to be modeled for every new application.

To show how to connect to ACROSS components from the building block library, we use the "Switch Off Channel Service" (DSS3_IC), which ensures a safe shutdown in case of a failure by the "Plausibility Safety Service" (DSS1_IC).

The "Switch Off Channel Service" DSS3_IC is a domain specific ACROSS service for the industrial control domain and is defined in deliverable D6.2. The main difference to standard channels is the capability of reading back an acknowledge signal to check if the intended event has taken place. In case the acknowledge is not raised, a second signal is raised by DSS3_IC. Intuitively, DSS3_IC can be used to make sure a machine is turned off by first raising a signal that requests clean shutdown and, if after some time the machine is still running, raising a second signal that cuts the machine from the main power source.

2.4.2 Combined Model and Applications

The application programmer can access ACROSS components using their interfaces (or stub components respectively). When the application is translated to BIP, these stubs are replaced with the corresponding models from the building blocks library. The resulting model consists of generated components from the application, library components from the building blocks, and connectors between the sub-components.

The exact configuration of the model depends on the intended use. E.g., automated verification usually requires a "complete" model that does not interact with the environment but contains a model of the anticipated environment instead. For such uses, "input" components like sensors will be modeled such that they allow arbitrary inputs to take place. For simulation on the other hand, some of the inputs might be fixed, while others are to be provided by the user. For deployment, the sensor models have to be connected to the actual hardware and provide the real data. This can be done by using BIP models that carry the required C code for connecting to the respective services of the underlying operating

system. Note that the generation of the model itself in each case is performed the same way and differs only in the model that is loaded from the building block library.

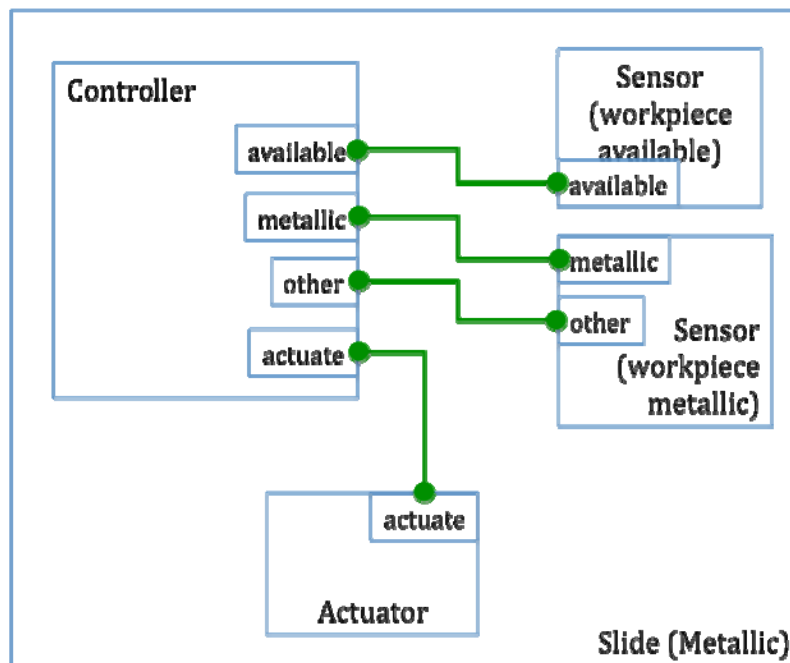


Figure 20: Combined Model

The combination of the different models for the controller can be seen in Figure 4, where the respective events from the state machines are connected. The connector (green lines) in the figure represent the behavior that, e.g., a transition in Controller with the label “available” can only take place if the connected Sensor indicates that the work piece is available. In this level of abstraction, the actual behavior within the components is insignificant and can be replaced easily by updated or more adequate models. One example of this is to replace the model of the Actuator by one that actually uses system calls to send the message to start the corresponding switch. Augmented with such method calls, the model can directly be used for code generation and deployment.

3 Wireless Communication Application Demonstrator

3.1 Demonstrator Overview

The wireless communication application demonstrator in WP6 shows the wireless communication capability of the ACROSS platform. It is important to mention that the demonstrator does not implement the full protocol stack for a particular wireless communication standard (e.g. GSM, UMTS, LTE), since it would go beyond the scope of this project. In fact, the demonstrator evaluates the low level communication between the ACROSS platform and an RF (Radio Frequency) transceiver device.

A simplified hardware block diagram of a wireless communication system can be seen in

Figure 21. The multiplexer or switch in front of the antenna is used to separate the received and the transmitted signals in the frequency or the time domain. The LNA (Low Noise Amplifier) and the PA (Power Amplifier) are required to amplify these signals. Basically, the RF transceiver is responsible for mixing the received signal down to the baseband frequency and mixing the transmitted signal up to the desired band. Modern RF transceiver devices also convert the signals from the analog to the digital domain and vice versa. Therefore, the interface between the baseband device and the RF transceiver device is digital. The baseband device implements the DSP (Digital Signal Processing) for the receive and the transmit paths as well as the controlling of the RF transceiver device. When used in a mobile phone, the baseband device typically includes other tasks like audio coding, keypad control and display control.

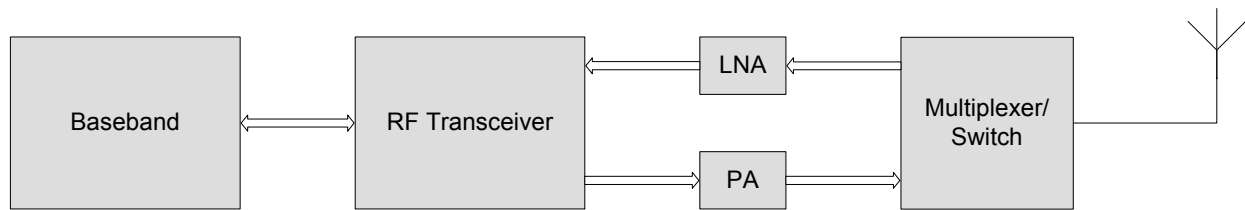


Figure 21: Hardware block diagram of a wireless communication system.

As stated above, the demonstrator will evaluate the low level communication between the ACROSS platform and the RF transceiver device. This is of special interest when implementing baseband devices on basis of the ACROSS platform. Running a baseband service on an ACROSS platform includes several basic requirements regarding the bandwidth and latency of the core services (TT-NoC, TISS, TRM). These requirements are defined by the implemented wireless communication standard. The wireless communication application demonstrator implemented in WP6 evaluates the basic requirements for the GSM (Global System for Mobile Communications) wireless communication standard.

Figure 22 illustrates the basic setup of the wireless communication application demonstrator. It consists of the ACROSS MPSoC, the RF transceiver device and the measurement equipment. The interface between the RF transceiver device and the ACROSS MPSoC complies with the DigRF V1.12 digital interface specification, therefore this interface has to be provided by the ACROSS MPSoC platform. The measurement equipment is needed to verify an errorless communication between the ACROSS MPSoC and the RF transceiver device.

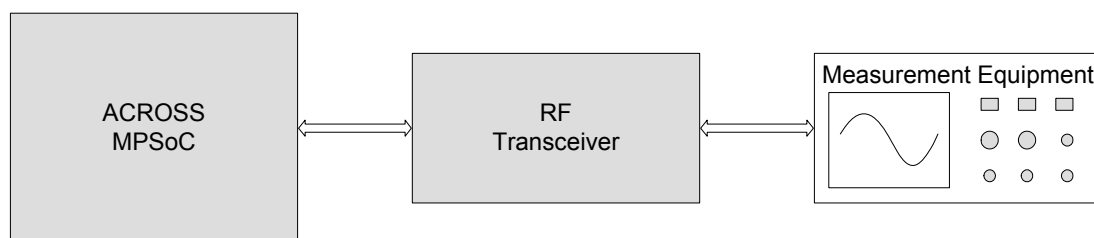


Figure 22: Basic setup of the wireless communication application demonstrator.

Figure 23 demonstrates the ACROSS MPSoC (Version A) which is used for the wireless communication application demonstrator in WP6. It implements three general purpose CPUs, the UART interface, the debug and trace interface, and the DigRF interface. The ACROSS MPSoC (Version B) illustrated in Figure 24 differs in the implementation of the UART interface. In this alternative realization the interface is directly connected to the CPU1 component and therefore, no additional communication between the UART interface and the CPU1 component over the TT-NoC is required.

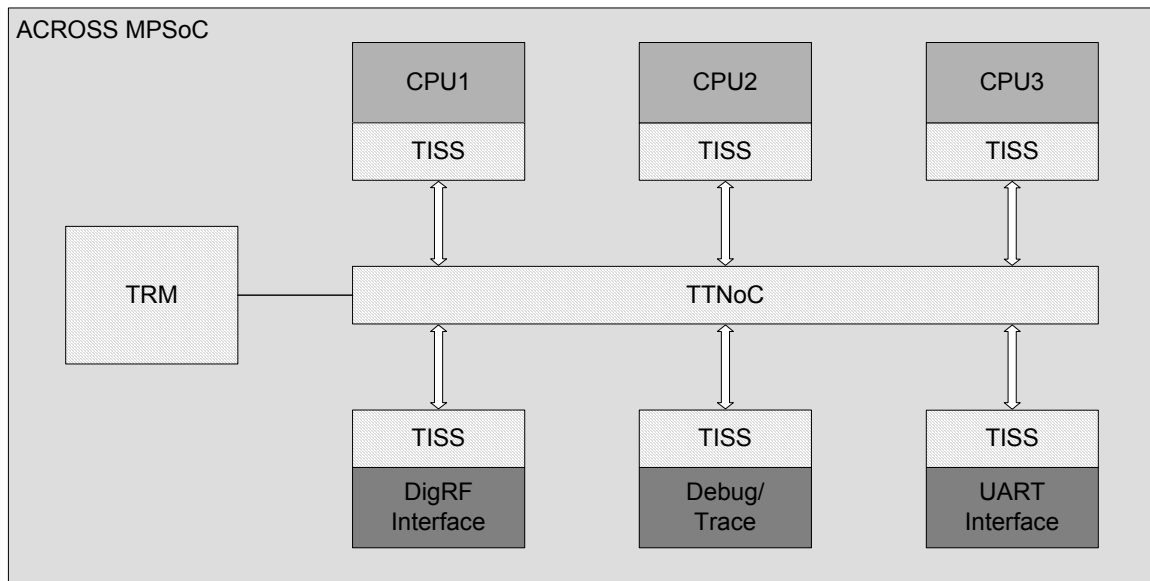


Figure 23: MPSoC for wireless communication application demonstrator (Version A).

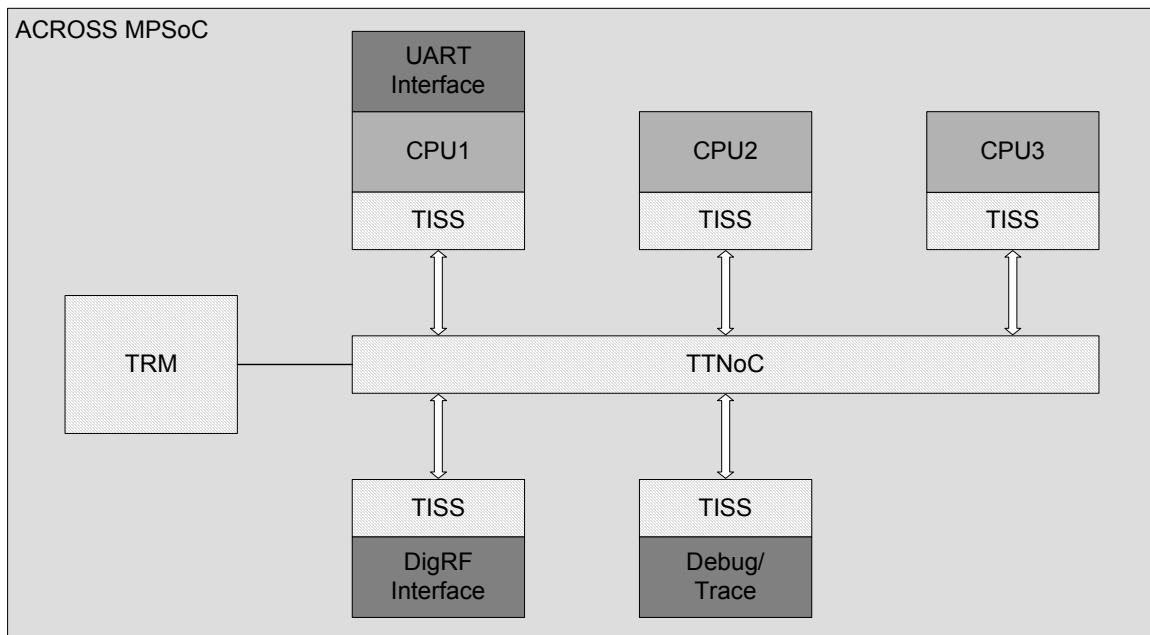


Figure 24: MPSoC for wireless communication application demonstrator (Version B).

3.2 Demonstrator Hardware Description

3.2.1 Mechanical Description

Figure 25 shows the detailed setup of the wireless communication application demonstrator in WP6. It consists of a PC, the FPGA board implementing the ACROSS MPSoC, the RF transceiver interface and evaluation boards, and the measurement equipment. The evaluation board hosts the RF transceiver device and is mounted on top of the interface board. The interface board basically includes the power supply for the RF transceiver and level shifters for the digital interface to the ACROSS MPSoC. The communication between the ACROSS MPSoC and the RF transceiver device can be tested using two different modes, the loop-back mode and the operational mode. When using the operational mode, the measurement equipment is needed to analyze the transmitted signal and to generate the signal which is received by the RF transceiver device. The loop-back mode, if supported, forces the RF transceiver device to directly send back the data which has been received over the DigRF V1.12 interface. The PC is used as a Human Machine Interface (HMI) to control different scenarios which evaluate the low level communication between the ACROSS platform and the RF transceiver device, and to analyze the resulting data.

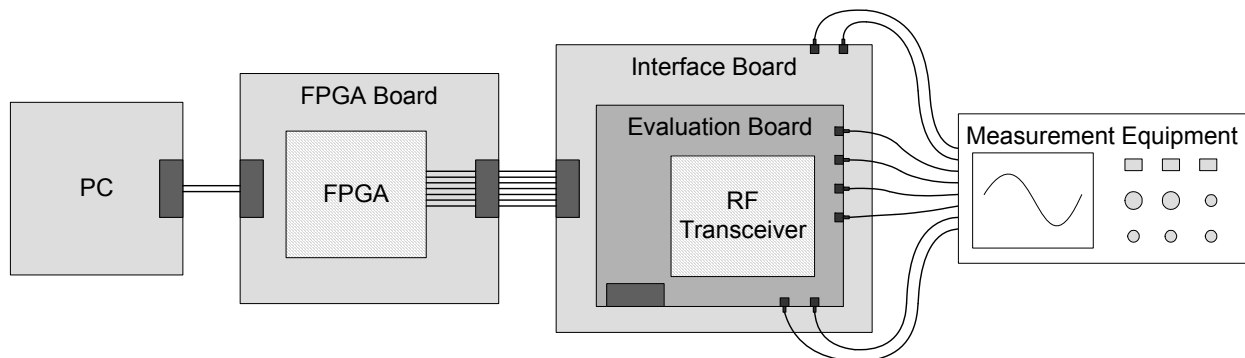


Figure 25: Detailed setup of the wireless communication application demonstrator.

An existing connector of the FPGA board, which at least provides 13 pins, is used for the mechanical interface between the FPGA board and the RF transceiver interface board. The type of the connector depends on the chosen FPGA board. 12 pins of this connector are routed to GPIOs of the FPGA and one pin is routed to a clock input of the FPGA.

The interface board and the evaluation board of the RF transceiver device are connected over a 60-pin SAMTEC connector.

For connecting the interface board and the evaluation board of the RF transceiver device with the measurement equipment SMA connectors are used.

The PC is connected to the FPGA board using a standard RS232 connector.

3.2.2 Electrical Description

3.2.2.1 DigRF V1.12 Digital Interface

Figure 26 illustrates the interconnection between a master and a slave device over a DigRF V1.12 digital interface. The interface includes a multiplexed serial digital interface for receive and transmit data, a

serial digital interface for control and the signals needed for system clock distribution. An additional strobe signal is provided to enable the master device to trigger events exactly within the slave device.

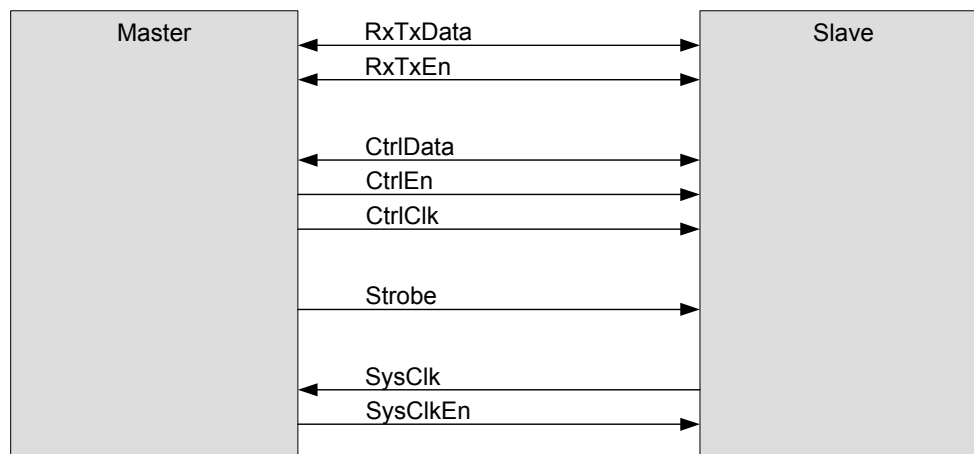


Figure 26: Interconnection over a DigRF V1.12 digital interface.

The interface between the interface board of the RF transceiver and the ACROSS MPSoC platform complies with the DigRF V1.12 digital interface specification but additional signals are required to split the bi-directional paths of the interface and to control the level shifters on the interface board. The level shifters are required to translate the different voltage levels between the FPGA board and the RF transceiver. Since parts of DigRF V1.12 digital interface are bi-directional, the coupling direction has to be controlled by the interface implementation of the ACROSS MPSoC platform.

The interconnection between the FPGA board and the RF transceiver interface board over a DigRF V1.12 digital interface including the additional signals can be seen in Figure 27. As stated above, the bi-directional paths are split and additional signals are added to control the direction of the data and control path.

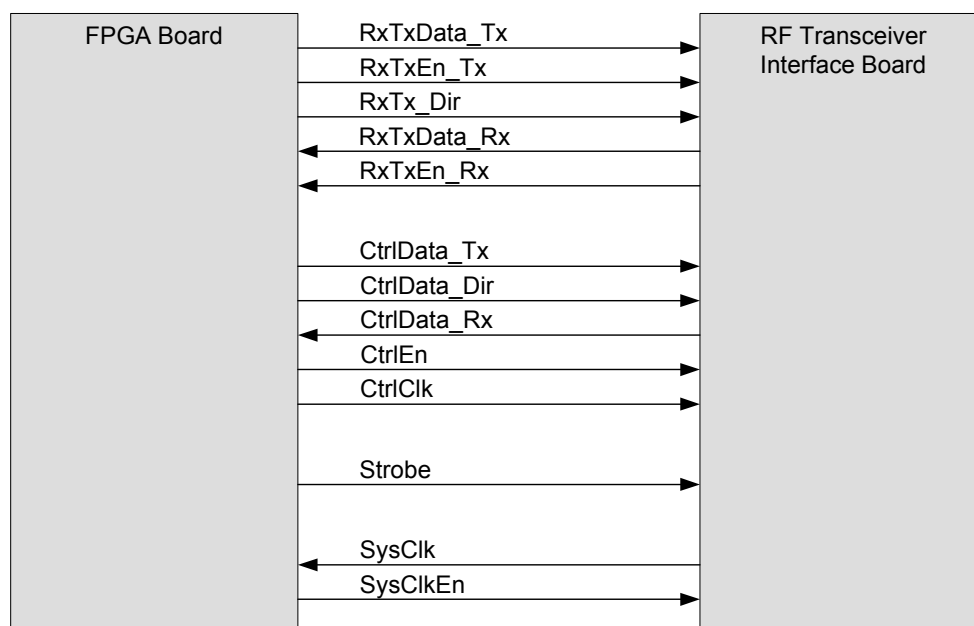


Figure 27: Interconnection over a DigRF V1.12 digital interface including additional signals.

3.2.2.2 UART interface

The wireless communication application demonstrator in WP6 evaluates the low level communication between the ACROSS platform and an RF transceiver device by executing several test scenarios. These scenarios require external input and of course can generate data which has to be analyzed by the verification engineer.

For this reason the demonstrator implements a UART interface for communicating with a HMI.

3.2.2.3 Debug / Trace interface

In case of an unwanted behavior in any test scenario, there has to be the opportunity to debug and trace the ACROSS platform.

Since the platform implements a standard debug and trace interface, this interface is used by the wireless communication application demonstrator for analyzing faulty scenarios.

3.3 Demonstrator Software Description

3.3.1 Application SW

The software implemented for the wireless communication application demonstrator in WP6 is used to evaluate the low level communication between the ACROSS platform and an RF transceiver device. Since this communication heavily depends on the performance of the TT-NoC, the tasks of the application software are assigned to different CPUs within the ACROSS MPSoC. Three main tasks are required to evaluate the communication:

- controlling the RF transceiver device,
- transmitting data and
- receiving data.
-

Figure 28 shows how these tasks are distributed to the CPUs of the ACROSS MPSoC. The component CPU1 is used to control the RF transceiver device, the component CPU2 provides the transmit data and the component CPU3 receives data.

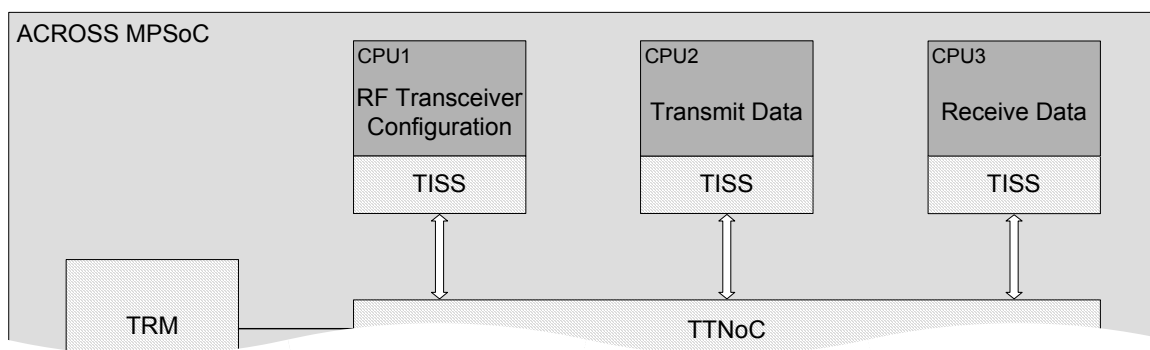


Figure 28: Allocation of CPUs within the ACROSS MPSoC.

The component CPU1 is not only responsible for controlling the RF transceiver device but also for controlling the component CPU2. The reason therefore is that the data stream, which is sent to the RF transceiver device, is timely coupled to the configuration of the device. The stream data is directly stored within the component CPU2 and can be configured prior to the execution of the test scenario. The received data stream is stored within the component CPU3 and can be read out after the execution of the test scenario.

Furthermore, the component CPU1 is used to communicate with the HMI. This communication is essential to setup up the test scenario, to pre-configure the transmit data stream and to read out the received data stream. The time flow of a single test cycle is illustrated in Figure 29.

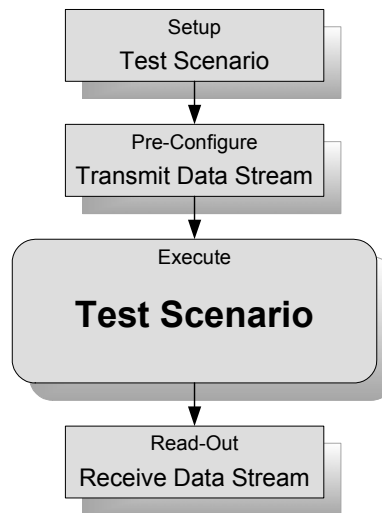


Figure 29: Time flow of single test cycle.

Figure 30 shows a flow chart of a typical transmit test scenario. In this scenario the RF transceiver device is configured to transmit three data shots, one GMSK (Gaussian Minimum Shift Keying) modulated slot and two 8-PSK (8-Phase Shift Keying) modulated slots. Furthermore, the chart illustrates the interaction between the CPU1 component (RF Transceiver Control) and the CPU2 component (Transmit Data). Before each shot is ramped up, the CPU1 component sends a message to the CPU2 component to enable the transmission of slot data.

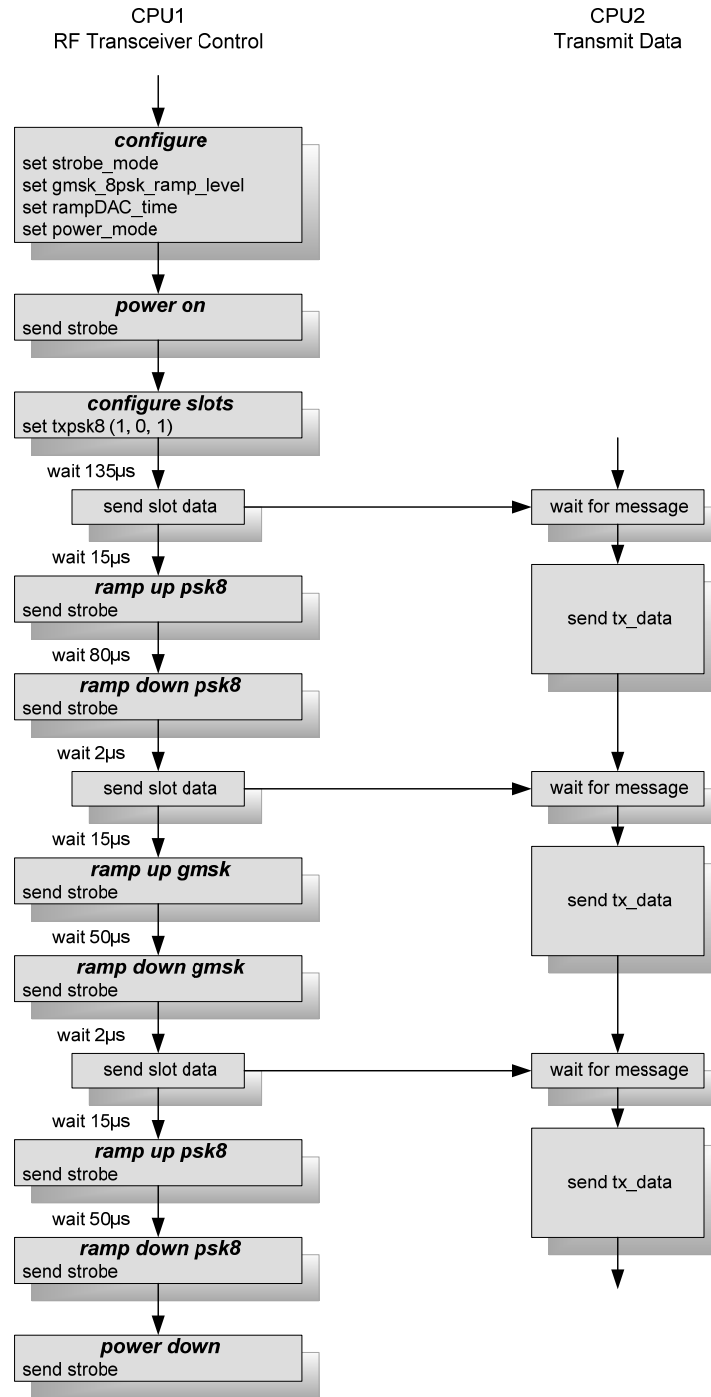


Figure 30: Flow chart of typical transmit test scenario.

3.3.2 Interaction with core, generic and domain specific services

Basically, three domain specific services are needed for the low level communication between the ACROSS platform and an RF transceiver device:

- the RF Transceiver Device Configuration Service,
- the RF Transceiver Device Data Transfer Service and
- the RF Transceiver Device Timing Accurate System Trigger Service.

All these services depend on the DigRF Interface Service implement in WP1. In addition to these domain specific services, a synchronization service is needed to enable the CPU1 component to trigger the transmission of slot data executed by the CPU2 component. All these services can be accessed by the test scenario application via Application Programming Interfaces (APIs).

Figure 30 demonstrates the interaction between the components involved in a test scenario by using the appropriate domain specific services.

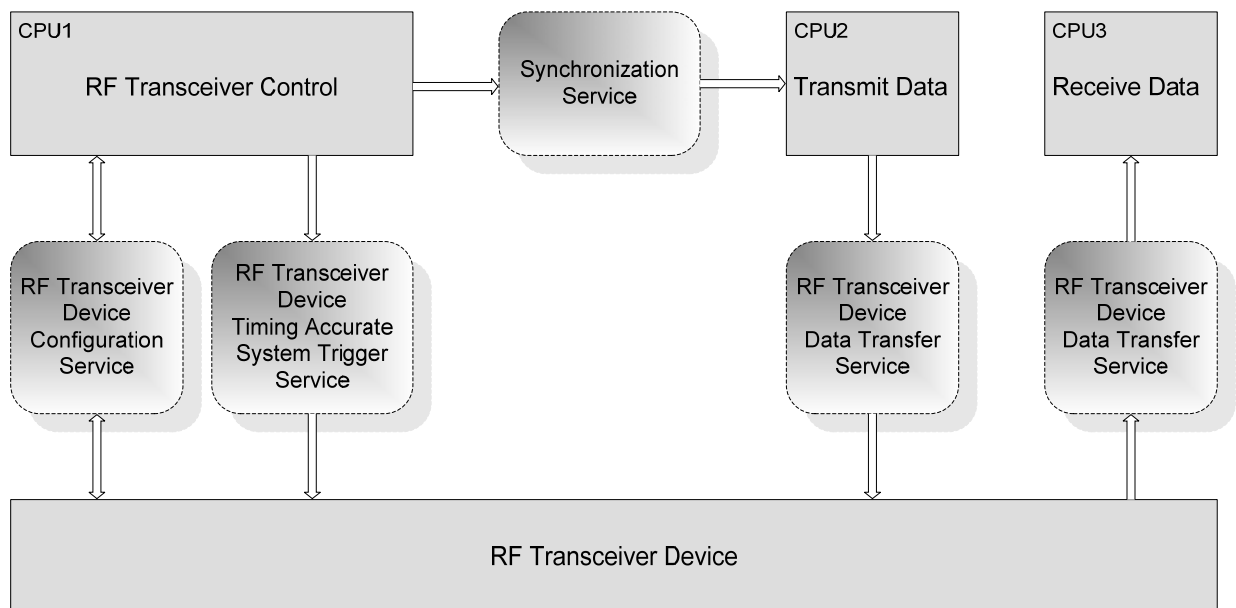


Figure 31: Interaction between components involved in a test scenario.

Furthermore, the demonstrator software implements services which are needed to configure and control the test scenario. These include:

- the Human Machine Interface Service,
- the Transmit Data Configuration Service and
- the Receive Data Read-Out Service.

These services are used prior and subsequent to the execution of the test scenario to configure and start the test scenario, to configure the transmit data stream and to read out the receive data stream. Figure 32 illustrates the interaction between the components involved in this pre- and post-processing by using the specified services.

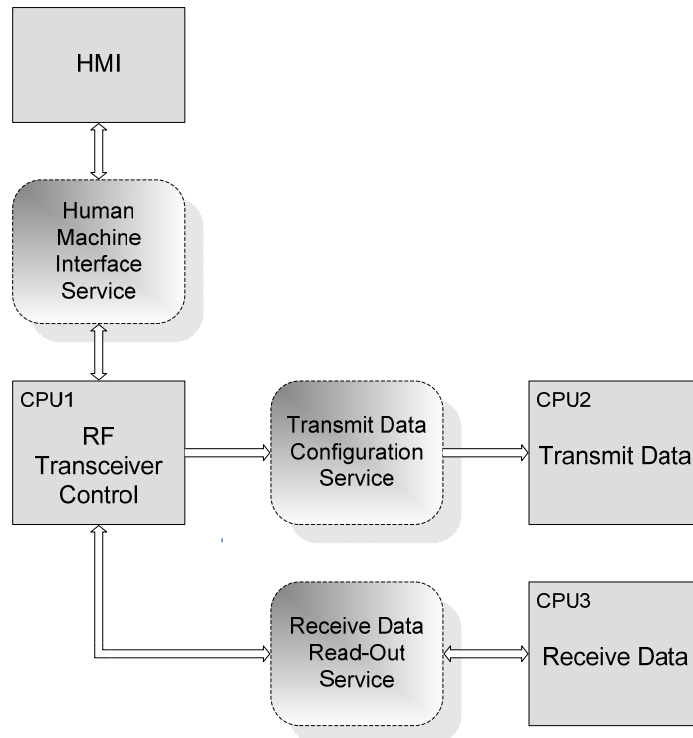


Figure 32: Interaction between components involved in the pre- and post-processing.

A) Annex: Refined Requirements

Name of the Requirement	Incorporated in WP	ID of Requirement after Incorporation in Resp. WP
Safety Properties	WP3	R 3.3.14
Fault-Tolerance	WP3	R 3.3.14
ACROSS: Safety Certification	WP3	R 3.3.14
Operating System Services: POSIX API	WP2	R 2.4.1

R 6.7.1 (WP6 Domain Specific Model GUI)

WP6 should provide a frontend for IEC 61131.

Significance: Mandatory

Rationale: WP6 should provide a GUI supported frontend for standard models used in industrial control domain (IEC 61131).

Originator: fortiss

Responsibility: fortiss

Requirement Status: accepted

Means for validation: The GUI shall be used for SW development on the WP6 demonstrator.

Requirement Source:

R 6.7.2 (WP6 Domain Specific Model Languages)

The frontend for IEC 61131 should support at least the languages SFC (Sequential Function Chart) and FBD (Function Block Diagram).

Significance: Mandatory

Rationale: These languages are the most used ones within industrial control domain

Originator: fortiss

Responsibility: fortiss

Requirement Status: accepted

Means for validation: These languages shall be used for SW development on the WP6 demonstrator.

Requirement Source:

R 6.7.3 (WP6 Domain Specific Model Transformation)

WP6 should implement transformation of domain specific models to generic models.

Significance: Mandatory

Rationale: The verification, analysis and code generation are done based on these generic

models (e.g. BIP)
 Originator: fortiss
 Responsibility: fortiss
 Requirement Status: accepted
 Means for validation: During case studies in WP6 demonstrator
 Requirement Source:

R 6.7.4 (IEC 61131 Debugging and Visualization)

The frontend for IEC 61131 should provide means for debugging and visualization of the application.

Significance: optional
 Rationale: This enables an efficient development of the applications.
 Originator: fortiss
 Responsibility: fortiss
 Requirement Status: accepted
 Means for validation: analysis during assessment (T6.11) and case study (T6.10)
 Requirement Source:

R 6.7.5 (Validate Fault-Tolerance by WP6 Demonstrator Application)

At least one use case in WP6 demonstrator should demonstrate Fault Tolerance by TMR.

Significance: high
 Rationale: Verification of the Fault Tolerance requirement should be done in the real world applications.
 Originator: fortiss
 Responsibility: SAGÖ, fortiss
 Requirement Status: accepted
 Means for validation: analysis during assessment (T6.11), case study (T6.10)
 Requirement Source:

R 6.7.6 (Interface to Demonstrator)

WP1 and WP2 should also provide services to interface the components used in the demonstrator of WP6

Significance: Mandatory
 Rationale: Festo MPS will be used as the WP6 demonstrator. WP1 and WP2 should also provide services to interface the Festo components.
 Originator: fortiss
 Responsibility: SAGÖ & fortiss
 Requirement Status: accepted
 Means for validation: analysis during specification (D1.2, D2.2), assessment (T6.11), case study (T6.10)
 Requirement Source:

R 6.7.7 (Validation of Requirements by Demonstrator)

The demonstrator should provide means to validate the requirements of WP 6 with respect to WP 1-3.

Significance: Mandatory
 Rationale: Based on the demonstrator, it should be possible to validate the correctness of

requirements.
Originator: fortiss
Responsibility: SAGÖ & fortiss
Requirement Status: accepted
Means for validation: All mentioned requirements must be reflected in at least one use case of the demonstrator
Requirement Source:

R 6.7.8 (Demonstrator: Safety Transmission)

Some physical values like position, speed, etc. should be measured by redundant sensors (e.g. one accurate and one coarse sensor) and should be transferred to the application via safe channels (e.g. physically separated). The values of the coarse sensor are used by domain specific safety services for plausibility checks.

Significance: optional
Rationale: This is necessary for the certification process of SIL-compliant applications.
Originator: SAGÖ
Responsibility: SAGÖ & fortiss
Requirement Status: accepted
Means for validation: analysis during assessment (T6.11), case study (T6.10)
Requirement Source:

R 6.7.9 (Demonstrator: Safety Services)

Safety relevant services (according IEC61508) like safely limited speed, safe stop, safe position and safe torque should be implemented. Safely limited speed for example ensures that the motion controlled robot arm must not exceed a speed threshold at all

Significance: high
Rationale: The implemented services are supporting the certification process of SIL-compliant applications.
Originator: SAGÖ
Responsibility: SAGÖ & fortiss
Requirement Status: accepted
Means for validation: analysis during assessment (T6.11), case study (T6.10)
Requirement Source:

R 6.7.10 (Demonstrator: Switch Off Channel)

A building block should provide safe resources to switch the peripherals, which are part of "Functional Safety" to a safe state. This channel is managed by the demonstrator application.

Significance: high
Rationale: There must be a safe channel to the components which should be switched to a safe state. This channel acts like a software emergency button and it can improve the system safety.
Originator: SAGÖ
Responsibility: SAGÖ & fortiss
Requirement Status: accepted
Means for validation: analysis during assessment (T6.11), case study (T6.10)
Requirement Source:

R 6.7.11 (ACROSS Benchmarking)

It shall be possible to replace the ACROSS platform with a commercial PLC platform.

Significance: optional
Rationale: Replacing the ACROSS platform with a commercial platform shall enable to benchmark the ACROSS platform.
Originator: SAGÖ
Responsibility: SAGÖ & fortiss
Requirement Status: accepted
Means for validation: analysis during assessment (T6.11), case study (T6.10)
Requirement Source:

R 6.7.12 (Demonstrator Application Interface: POSIX API)

Demonstrator applications should communicate with operating system services via POSIX API.

Significance: Mandatory
Rationale: POSIX conformance is essential for real-time embedded software development
Originator: SAGÖ
Responsibility: SAGÖ & fortiss
Requirement Status: accepted
Means for validation: analysis during assessment (T6.11), case study (T6.10)
Requirement Source:

R 6.7.13 (WP6 Fieldbus Communication Interface)

Communication interfaces in industrial control domain (e.g. Profibus, Profinet, sensor interfaces and actuator interfaces) should be supported.

Significance: high
Rationale: Profibus and Profinet are the most used industrial control communication interfaces. Sensor interfaces and actuator interfaces will be necessary during development of WP6 demonstrator.
Originator: SAGÖ
Responsibility: SAGÖ & fortiss
Requirement Status: accepted
Means for validation: analysis during assessment (T6.11), case study (T6.10)
Requirement Source:

R 6.7.14 (Domain Specific Services: Synchronisation Services)

Coordination and synchronisation of several parallel running applications should be supported.

Significance: optional
Rationale: Accurate synchronisation is essential for the development of applications with multiple sensors and actuators regarding functional and safety requirements.
Originator: SAGÖ
Responsibility: SAGÖ & fortiss
Requirement Status: accepted
Means for validation: analysis during assessment (T6.11), case study (T6.10)
Requirement Source:

R 6.7.15 (Domain Specific Services: Regular Industrial Services)

Within WP6 a set of commonly used industrial services (e.g. PID controller) should be provided to the application developer.

Significance: optional
Rationale: This enables reuse of components and increases the development productivity.
Originator: SAGÖ
Responsibility: SAGÖ & fortiss
Requirement Status: accepted
Means for validation: analysis during assessment (T6.11), case study (T6.10)
Requirement Source:

R 6.7.16 (Demonstrator Application)

An industrial Automation PLC application should be realized. This application should focus on the domain specific services based on IEC 61131 for programmable logic controls. It should demonstrate the performance enhancements by using ACROSS approach.

Significance: Mandatory
Rationale: PLC industrial automation is one of the largest segments of the industrial segment.
Originator: SAGÖ
Responsibility: SAGÖ & fortiss
Requirement Status: accepted
Means for validation: analysis during assessment (T6.11), case study (T6.10)
Requirement Source:

R 6.7.17 (Fault injection: TMR)

The demonstrator should provide means to inject faults to corrupt the data channel to the TMR voter.

Significance: mandatory
Rationale: the TMR voting result and the correct system behaviour should be validated.
Originator: SAGÖ
Responsibility: fortiss
Requirement Status: accepted
Means for validation: analysis during assessment (T6.11), case study (T6.10)
Requirement Source:

R 6.7.18 (Fault injection: DDR device)

The demonstrator should provide means to inject multi bit faults of data written to the DDR device.

Significance: high
Rationale: The behaviour of the Memory Scan Service in case of multi bit faults should be validated.
Originator: SAGÖ
Responsibility: SAGÖ
Requirement Status: Originally accepted, afterwards changed to rejected, since the current MPSoC platform architecture does not contain any more a shared memory resource with EDC
Means for validation: analysis during assessment (T6.11), case study (T6.10)
Requirement Source:

R 6.7.19 (Domain Specific Services: RF Transceiver Device Timing Accurate System Trigger Service)

Domain specific service for timing accurate trigger of RFIC functionality.

Significance: Mandatory
Rationale: GSM slot timing requirements have to be met.
Originator: DICE
Responsibility: DICE
Requirement Status: accepted
Means for validation: analysis during assessment (T6.11), case study (T6.10)
Requirement Source:

R 6.7.20 (Domain Specific Services: RF Transceiver Device Configuration Service)

Domain specific service for register access on different communication channels to/from the RF transceiver device.

Significance: Mandatory
Rationale: Prove successful configuration of RX/TX functionality of the RF transceiver device.
Originator: DICE
Responsibility: DICE
Requirement Status: accepted
Means for validation: analysis during assessment (T6.11), case study (T6.10)
Requirement Source:

R 6.7.23 (Domain Specific Services: RF Transceiver Device Data Transfer Service)

Domain specific service for GSM data transfer on different communication channels to/from the RF transceiver device.

Significance: Mandatory
Rationale: This enables the slot data transfer to/from the RF transceiver device.
Originator: DICE
Responsibility: DICE
Requirement Status: accepted
Means for validation: analysis during assessment (T6.11), case study (T6.10)
Requirement Source:

R 6.7.21 (Demonstrator Application Interface: RF Transceiver Device Access API)

Domain specific application API for accessing registers, time-accurate triggering and control of RX/TX data streams.

Significance: Mandatory
Rationale: Provide RF transceiver device API functions for higher application layers.
Originator: DICE
Responsibility: DICE
Requirement Status: accepted
Means for validation: analysis during assessment (T6.11), case study (T6.10)
Requirement Source:

R 6.7.22 (Demonstrator: Loopback Data Transfer to/from RF Transceiver Device)

Enable TX to RX stream loopback to prove successful data transfer to/from RF transceiver device.

Significance: Optional
Rationale: Test data transfer independent of RF part.

Originator: DICE
Responsibility: DICE
Requirement Status: accepted
Means for validation: analysis during assessment (T6.11), case study (T6.10)
Requirement Source:

R 6.7.23 (Demonstrator: Regular Data Transfer to/from RF Transceiver Device)

Check for a feasible TX spectrum on the RF transceiver device outputs to prove successful TX data transfer. Check for a feasible data stream in the RX output buffer (separate micro component).

Significance: Mandatory
Rationale: Test data transfer including RF part.
Originator: DICE
Responsibility: DICE
Requirement Status: accepted
Means for validation: analysis during assessment (T6.11), case study (T6.10)
Requirement Source:

R 6.7.24 (Demonstrator: Bandwidth Requirements)

Prove RF transceiver device bandwidth requirements are met by TTNoC.

Significance: Mandatory
Rationale: GSM data & RF transceiver device control bandwidth requirements have to be met.
Originator: DICE
Responsibility: DICE
Requirement Status: accepted
Means for validation: analysis during assessment (T6.11), case study (T6.10)
Requirement Source:

B) Annex: Platform Requirements

This section references the requirements that have been contributed by this WP to the platform work packages WP1, WP2 and WP3. These requirements are included in the respective deliverables (D1.1, D2.2 or D3.1).

Name of the Requirement	Incorporated in WP	ID of Requirement after Incorporation in Resp. WP
Composability regarding to HW	WP1	R 1.1.13
LIF: Minimalism	WP1	R 1.1.8
LIF: Robustness	WP1	R 1.1.14
LIF: Operational Specification	WP1	R 1.1.15
Multi Vendor Environnement	WP1	R 1.1.16
Hiding internal Services	WP1	R 1.1.17
Heterogeneity	WP1	R 1.1.9
Different Levels of Reliability	WP1	R 1.2.11
Guarantees for ET Message Transfer	WP1	R 1.2.10
Naming	WP1	R 1.2.12
Control Loop	WP1	R 1.2.1
Global Time Service	WP1	R 1.2.2
Real-time Message Transfer	WP1	R 1.2.3
Deterministic and Static Allocation	WP1	R 1.3.7
Debugging	WP1	R 1.4.3
No Probe Effect	WP1	R 1.4.4
Frequency Islands	WP1	R 1.5.3
Memory Controller configuration: Windowing Logic	WP3	R 3.2.8
Memory Controller configuration: Debug Support	WP3	R 3.2.9
Memory Controller configuration: FIFO Sizes	WP3	R 3.2.10

Name of the Requirement	Incorporated in WP	ID of Requirement after Incorporation in Resp. WP
TTNOC: Safe Message Transfer	WP1	R 1.9.4
Verification of Timing Properties	WP3	R 3.3.13
Correct System Synthesis	WP3	R 3.2.4

C) Annex: Bibliography

- [1] Fieldbus Appendix Anybus-M Profibus DP-V Rev. 1.31 (85-0344-ABM-DP-V1_1_31.pdf)
- [2] Parallel Interface Design Guide Anybus-S Slave & Master Doc.Id. SCM-1200-015 Rev. 2.06 (Anybus-M-2737-ABS_M_Parallel_DG_2_06_SCM-1200-015.pdf)
- [3] Festo MPS Station Sorting: www.festo.com