SYSTECS

Entwicklung von eingebetteter Software in der Automobilindustrie mit AUTOSAR

Dr. Thomas Zurawka, SYSTECS, Februar 2015

Introduction

- The **Vehicle –** A **driving low cost Computer network**
  - … more than 40 Mio. new vehicles with 1.000 Mio. ECU per year
  - … ECU **Software** for 8…32-Bit Fixed-Point Processor

Electronic
Control
Unit

Source for ECU: Robert Bosch GmbH

Introduction

- The **Vehicle –** A **driving low cost Computer network** contains
  - … heterogeneous Components
    - Gasoline & Electrical Engines, Wireless Cell Phones, …

  must be delivered
  - … with 0 Defects
  - … for 5.000 Euro – 150.000 Euro
  - … within 3 years Development Time

  must be developed
  - … together with **many**, highly integrated **Suppliers**

  must be maintained
  - … for 30 years

Source for Throttle: Robert Bosch GmbH

Trends in the Automotive Industry

- The Vehicle - A driving **low cost** Computer network

8 Bit … 32 Bit Controllers

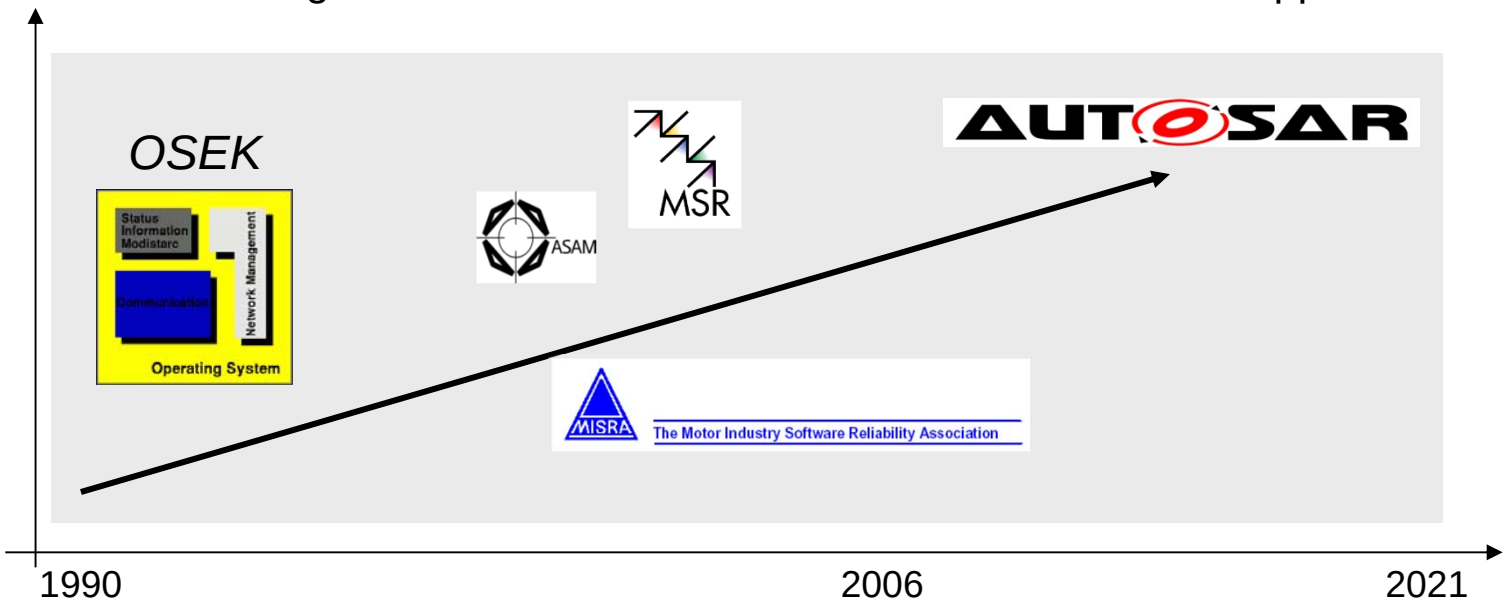
5kByte – 2MByte Bytes Flash

Why is there so much pressure on Controller size and Memory consumption ?

➔ *1MByte Flash costs 10 Euro !*

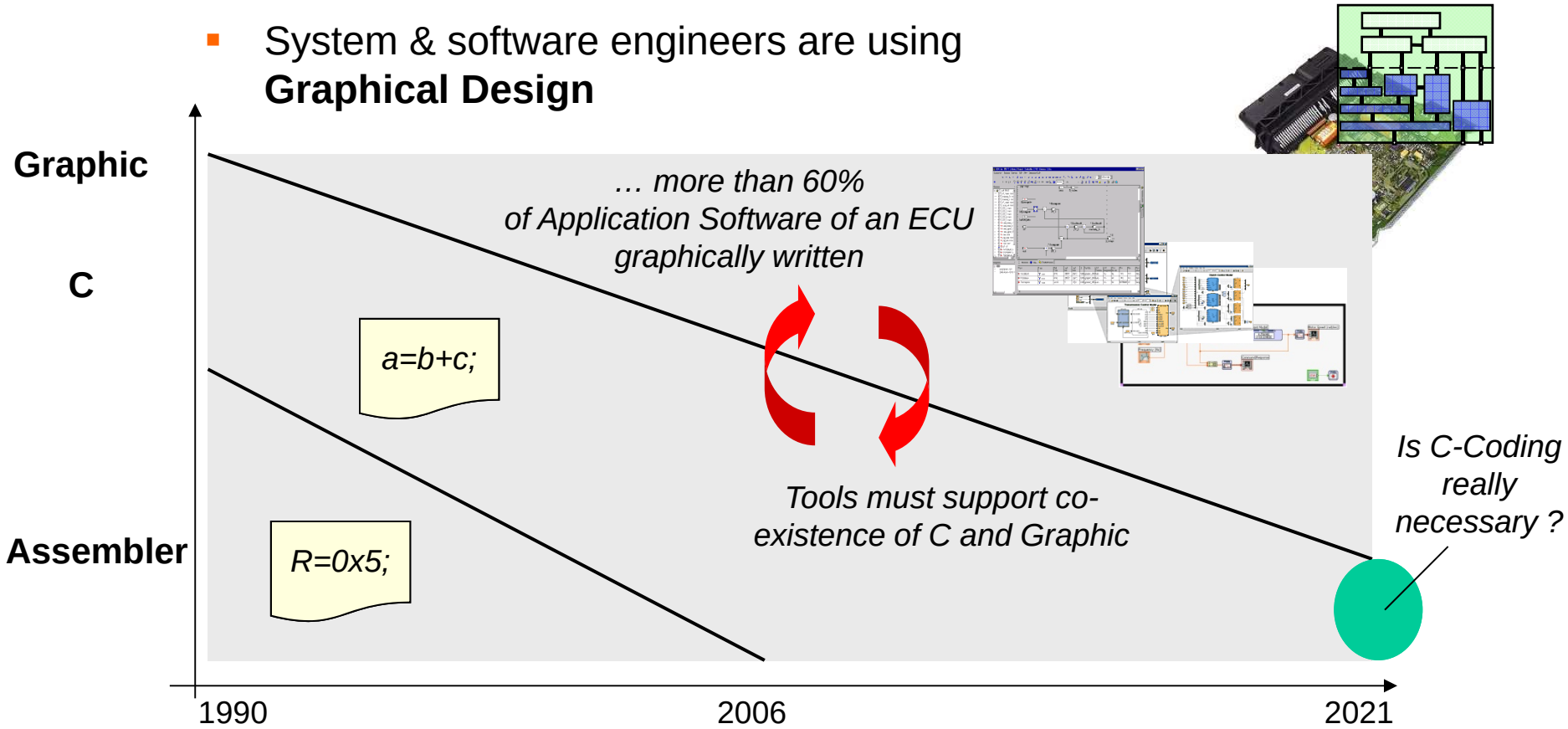
Trends in the Automotive Industry

- The Vehicle - A driving low cost **Computer network**
  - **Standardization** of ECU Software Architecture and Data due to problems with
    - ECU networks and
    - Exchange of Data **between Tools** and between OEM & suppliers



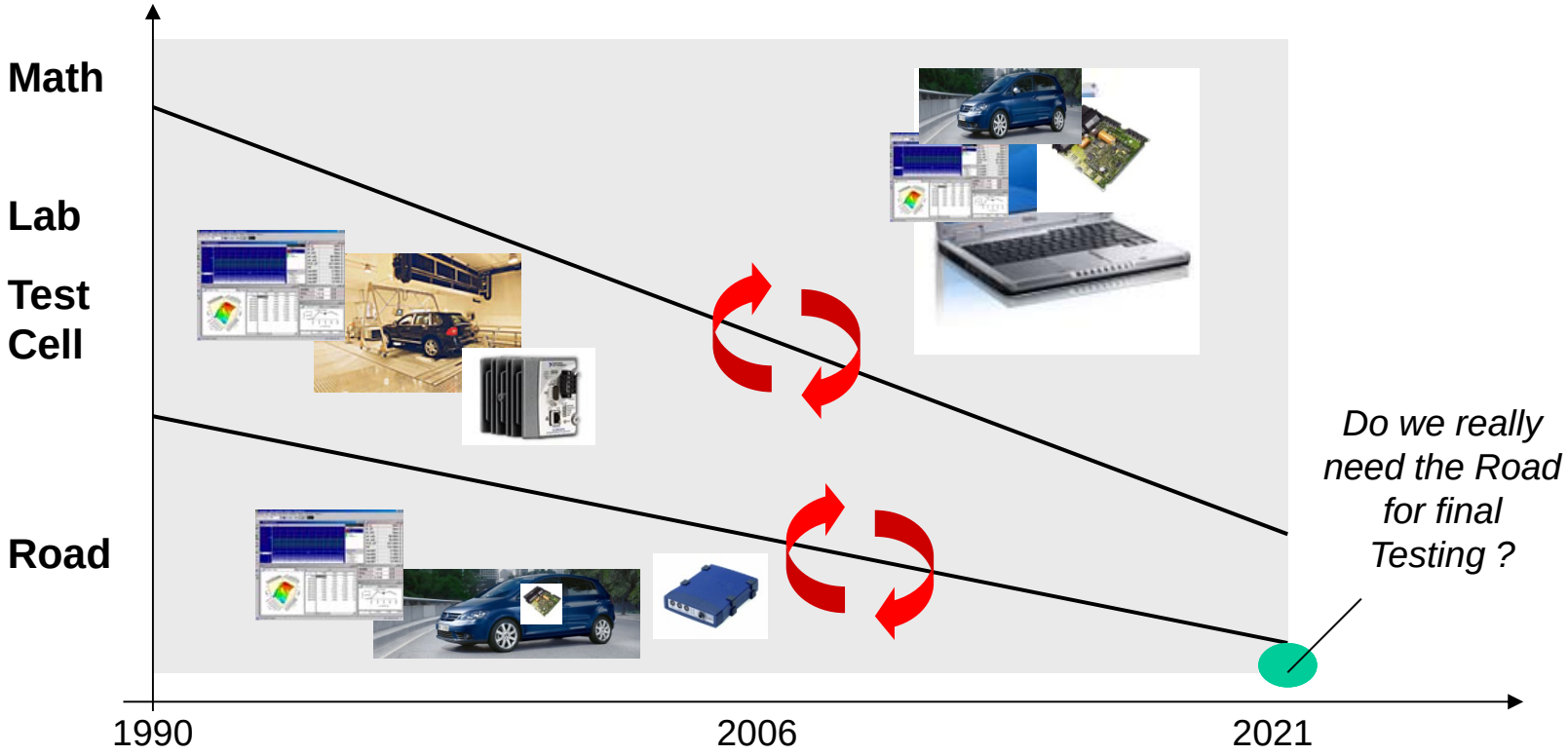
Trends in the Automotive Industry

- **The Vehicle** - A driving low cost **Computer** network
  - System & software engineers are using **Graphical Design**



Trends in the Automotive Industry

- The Vehicle - A **driving** low cost **Computer** network
  - Road -> Test Cell -> Lab -> Math



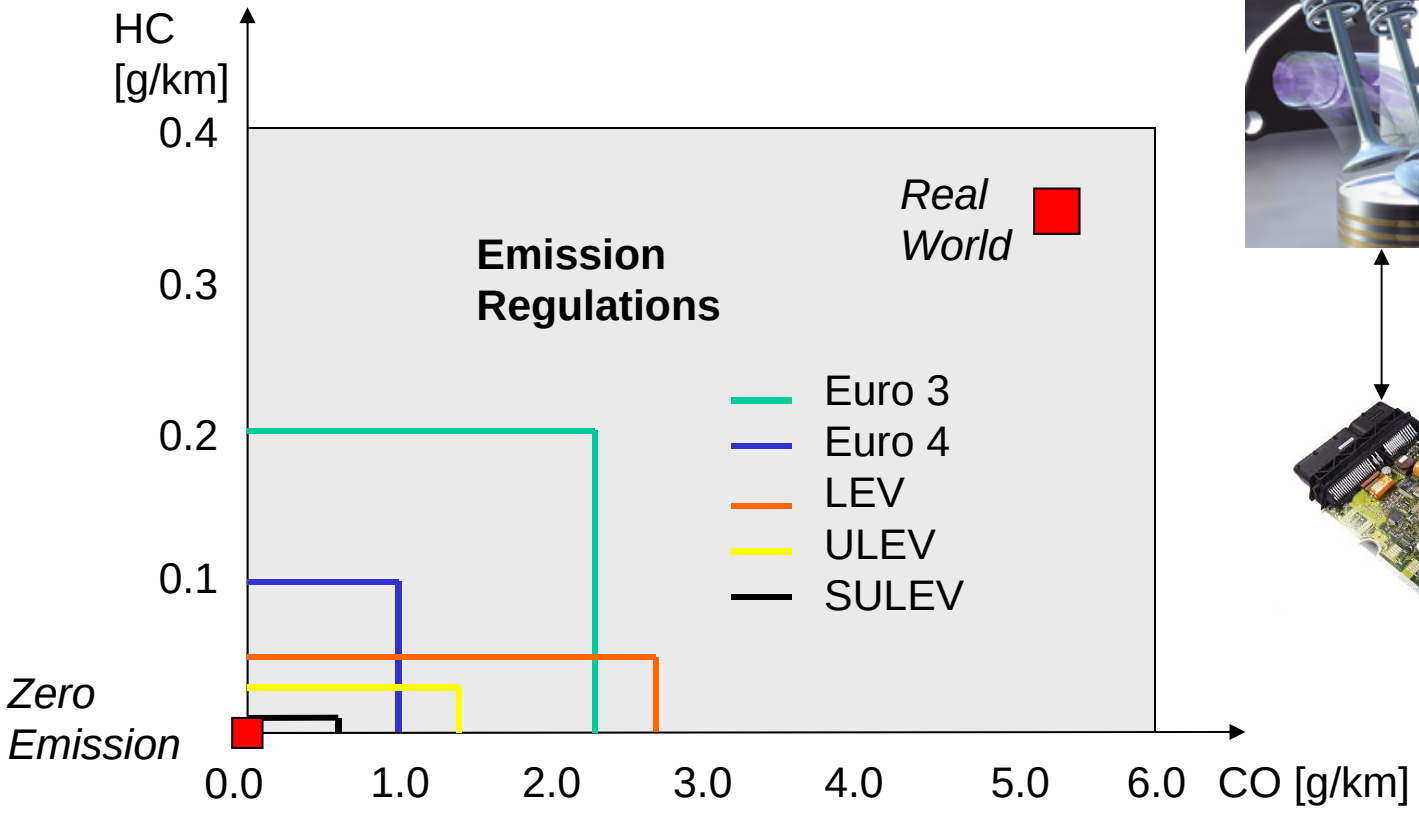
Trends in the Automotive Industry

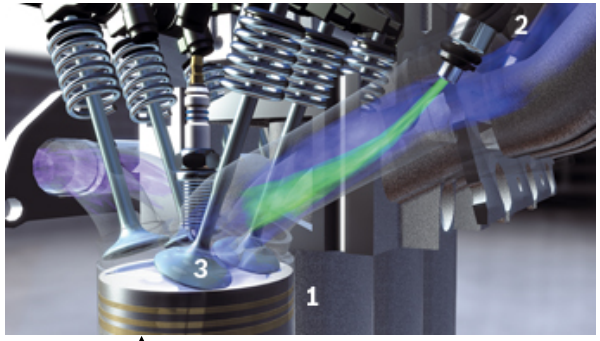
- **Worldwide** distributed Development Teams
  - e.g. **1000 Employees** & **100 Projects** per year for **one** ECU generation

Do we already have the Tools which support this use case ?

Trends in the Automotive Industry

- **Increasing complexity**
  - An Example: Legal Requirements





Much more Software needed …

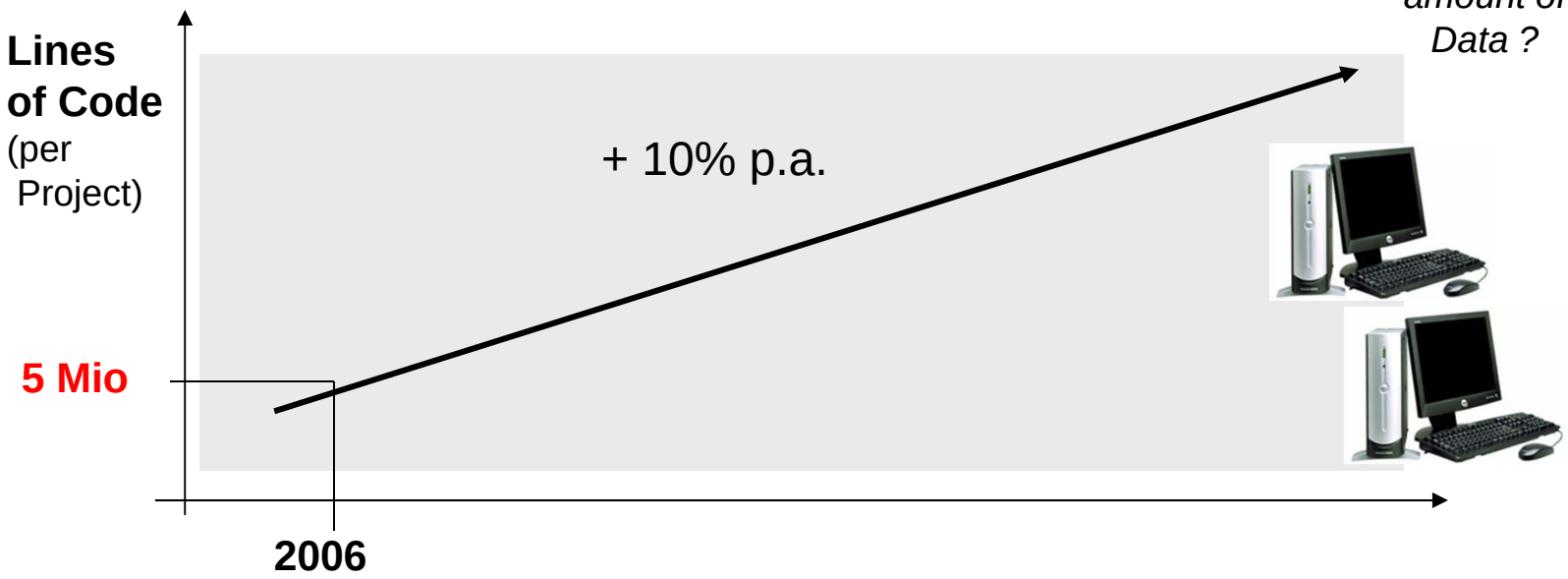
and … other System Concepts, e.g. Hybrids, Fuel Cell, …

Source: Robert Bosch GmbH

Trends in the Automotive Industry

- **Increasing Complexity** due to
  - Customer Requirements, Legal Requirements
  - more Variants **and** more complex Controllers
    - 50 Registers -> 1000 Registers

Which Tools can manage this huge amount of Data ?



Trends in the Automotive Industry

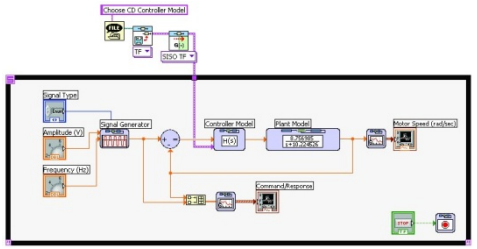
- **How** does the Automotive Industry master these challenges ?

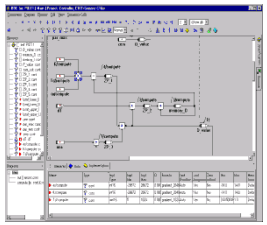
  - **CMMI** or SPICE complaint Processes

Is CMMI 3,4 or 5 the right final level ?

  - Tricky development Methods and …

    … Graphical Design & Test with **Tools, Tools, Tools, …**

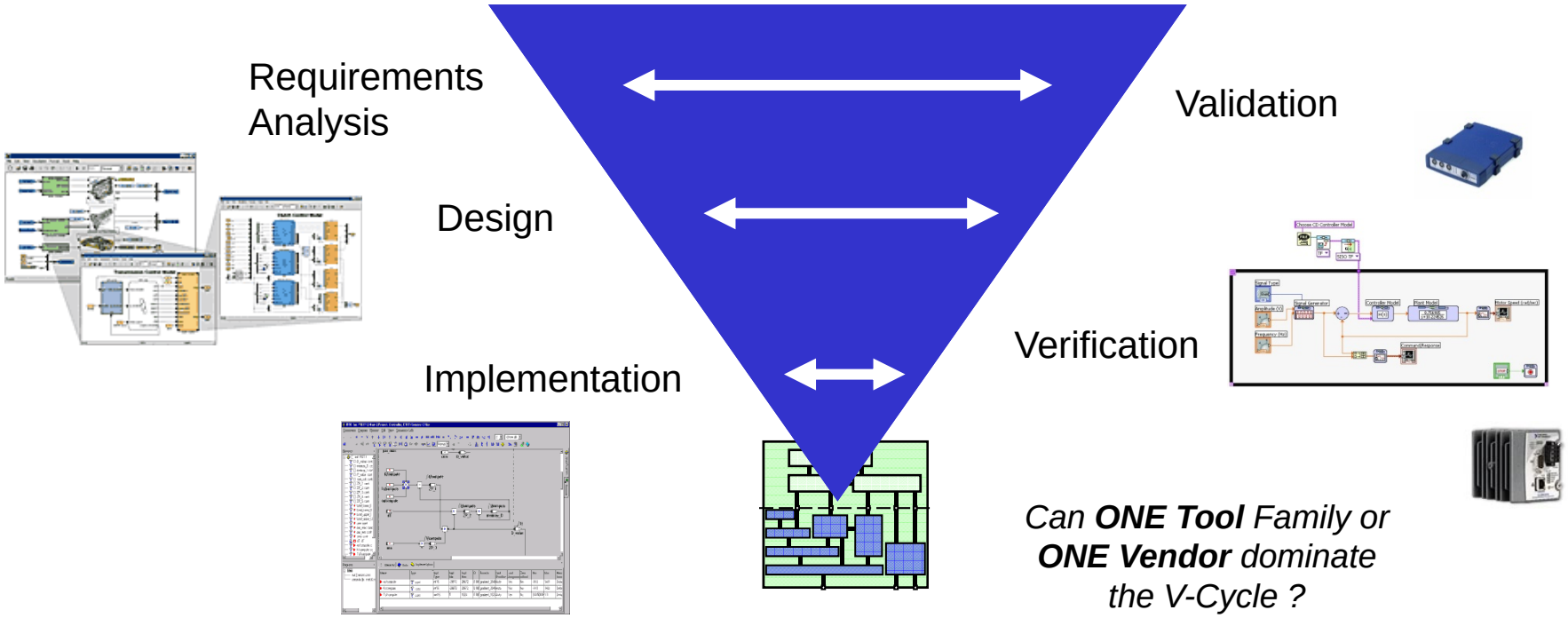




Trends in the Automotive Industry

- **CAE- Tools** for the Development of **Software** for Vehicles
  - V-Cycle for ECU Software

***Many Tools** from **different Vendors** are in use !*



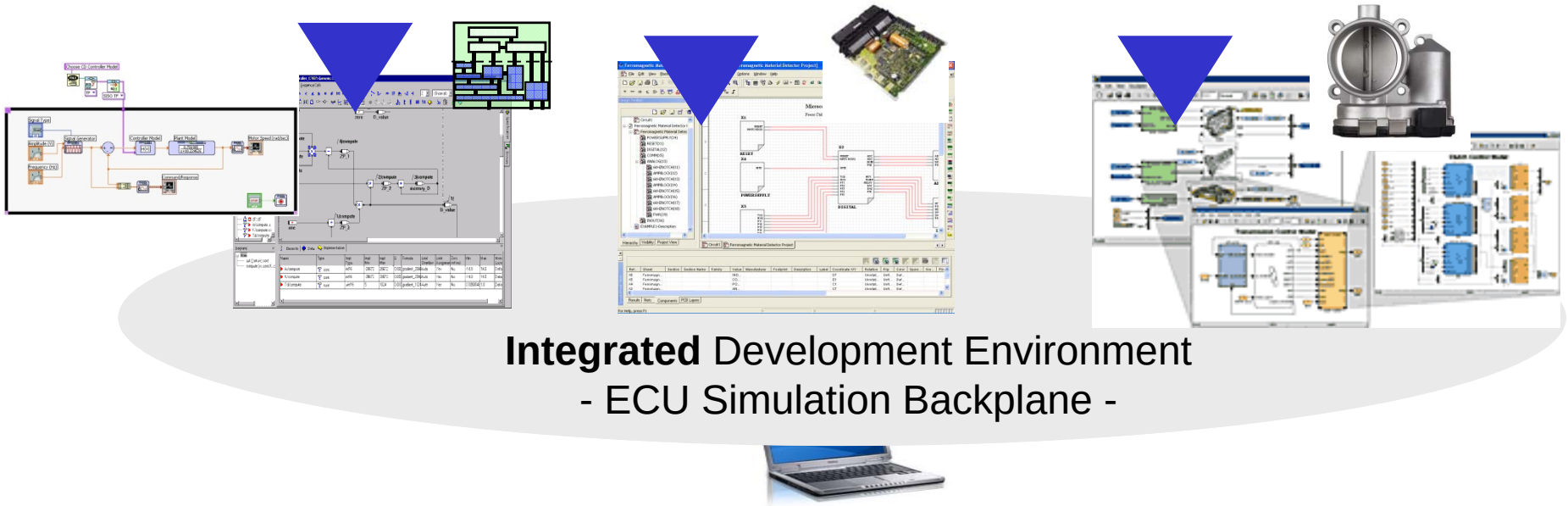
*Can **ONE Tool** Family or **ONE Vendor** dominate the V-Cycle ?*

Initiatives in the Automotive Industry

- **The Virtual Development Initiative**

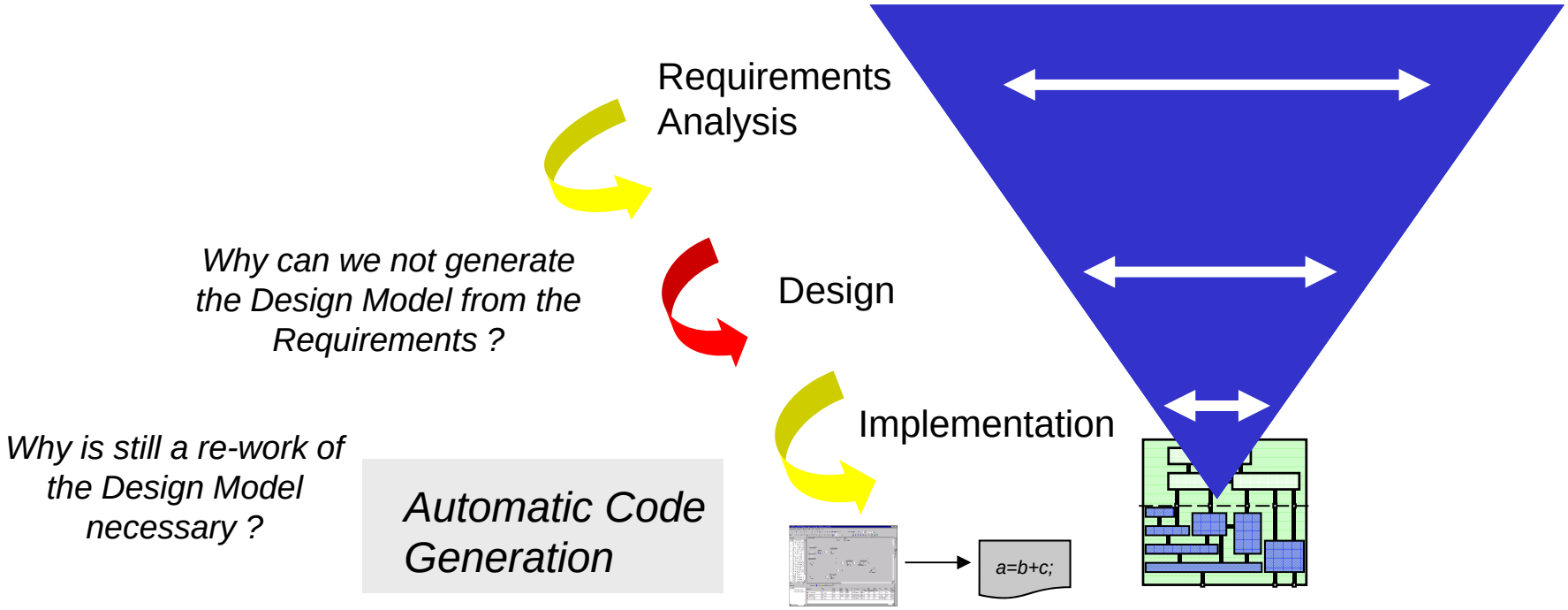
*Which Tool Vendor supports **Co-Design** of ECU Software, ECU Hardware and Vehicle Components first ?*

ECU Software ECU Hardware Throttle, …



Initiatives in the Automotive Industry

- **The Automation Initiative**



Initiatives in the Automotive Industry

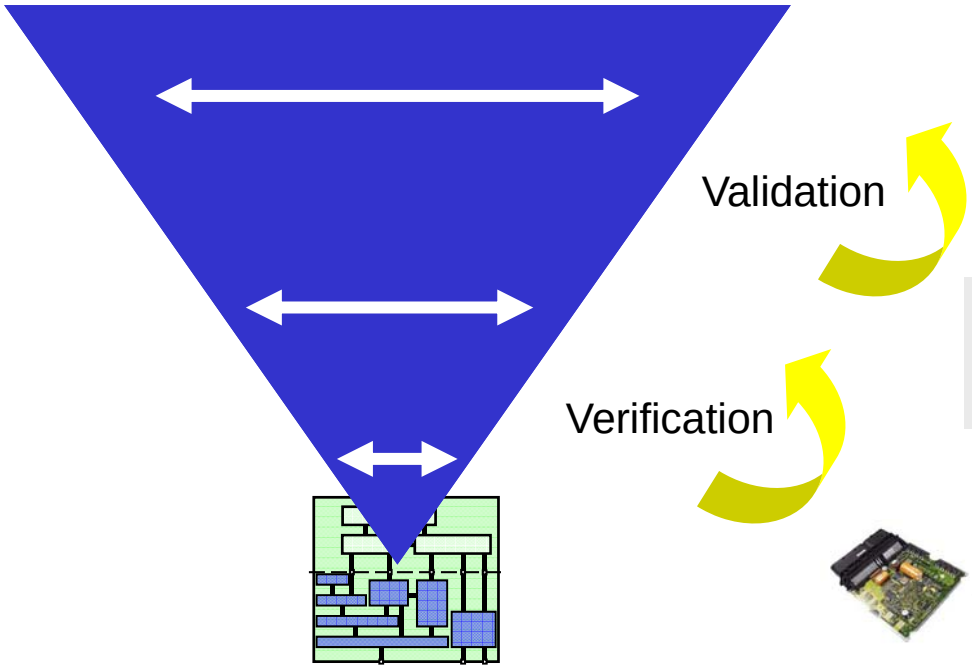
- **The Automation Initiative**

With what language shall I code my Test Scripts ?

Can I share them with others ?

How can I manage them ?

Is manual coding really necessary ?



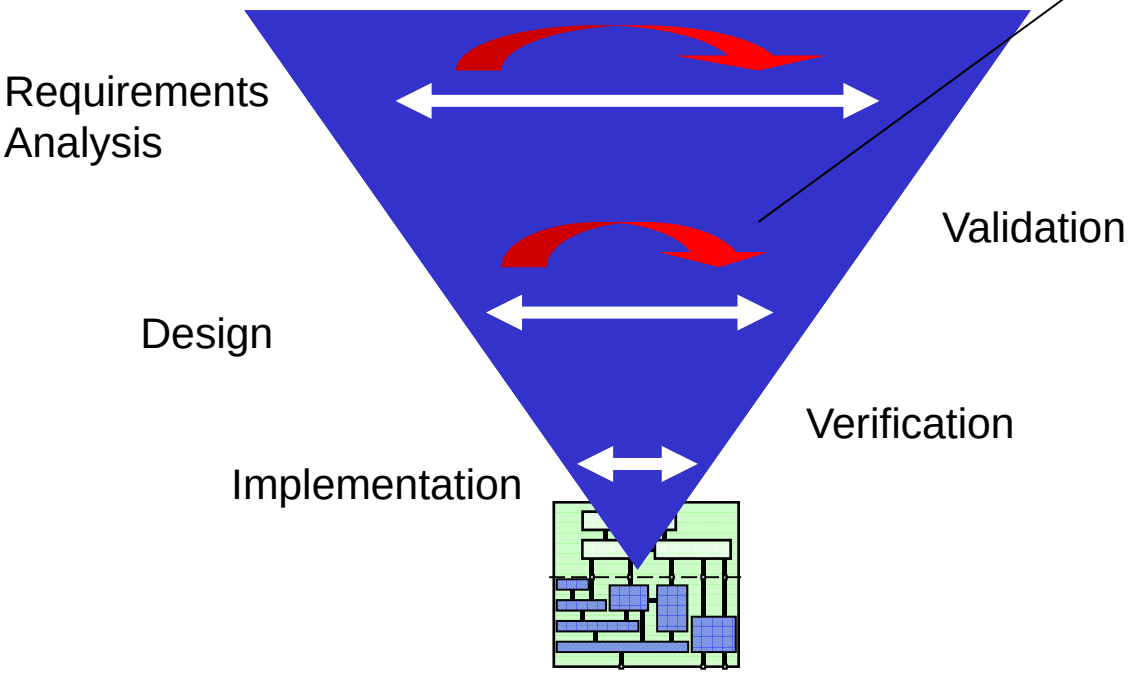
HIL Tests are very important for release of high quality Software !, ***but ...***

Automatic Tests - HIL -

Initiatives in the Automotive Industry

- **The Automation Initiative**

Automatic
Test Script Generation



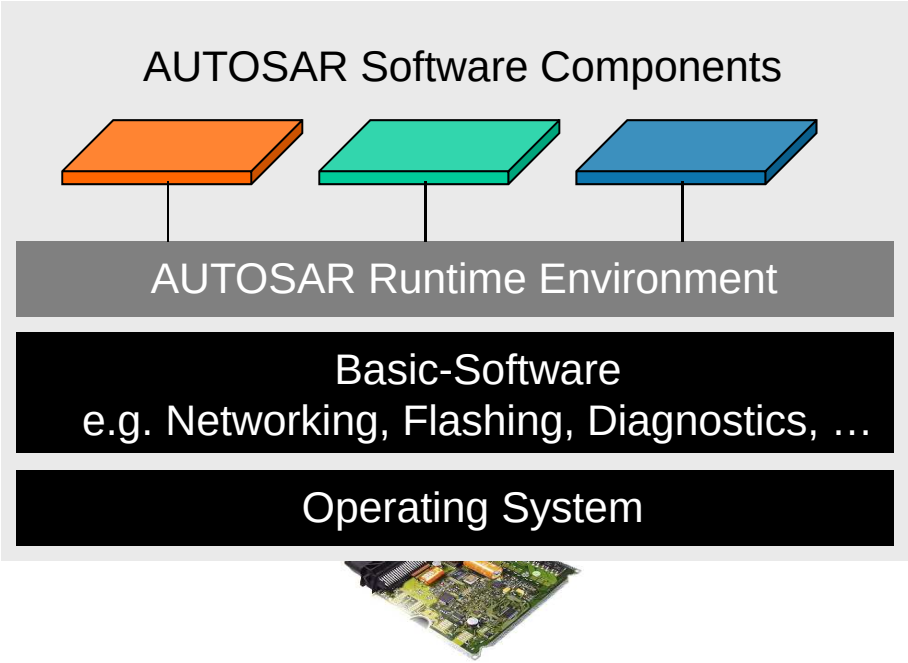
*Huge Efficiency increase possible, **but…***

still no Technology available !

Initiatives in the Automotive Industry

- **The Standardization Initiative**

AUTOSAR
ECU Software Architecture

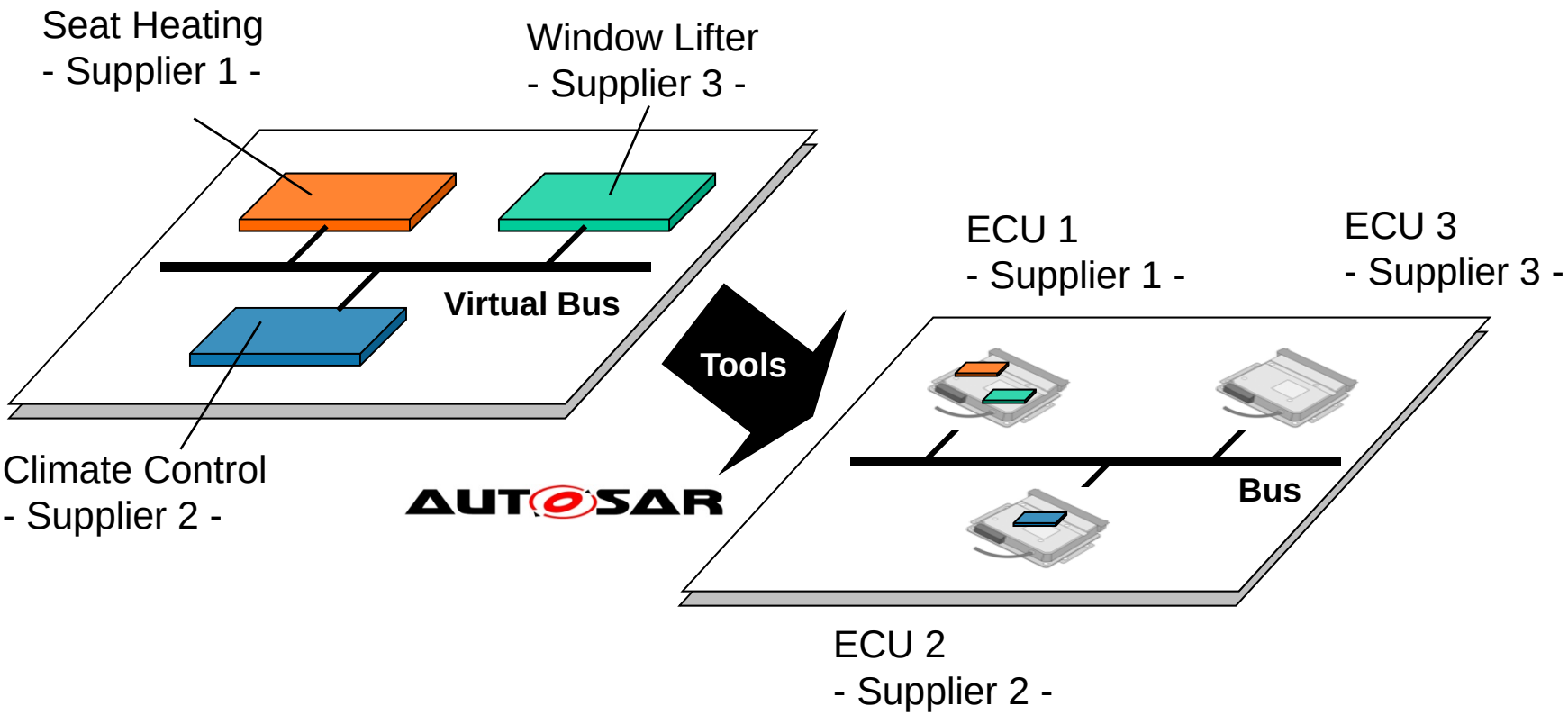


AUTOSAR

- *Some of the Objectives*
  - *Standardization of Basic Software*
  - *Transferability of Software Components throughout network*
  - *Integration of Software Components from multiple suppliers*

Initiatives in the Automotive Industry

- **The Standardization Initiative**



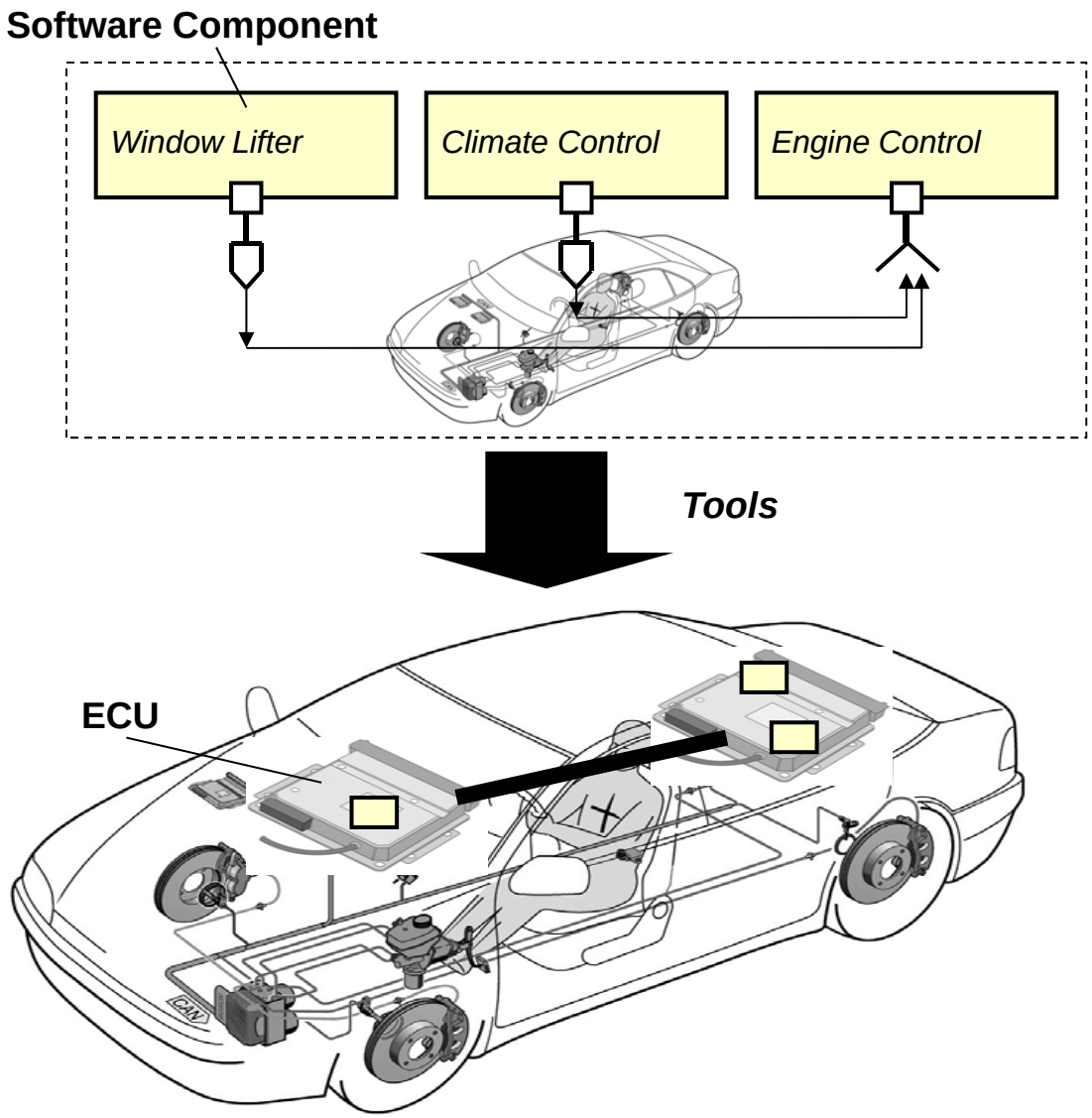
SYSTECS

Part 1: AUTOSAR – in a Nutshell

Concepts – Software Component, Virtual Functional Bus,
ECU Software Architecture

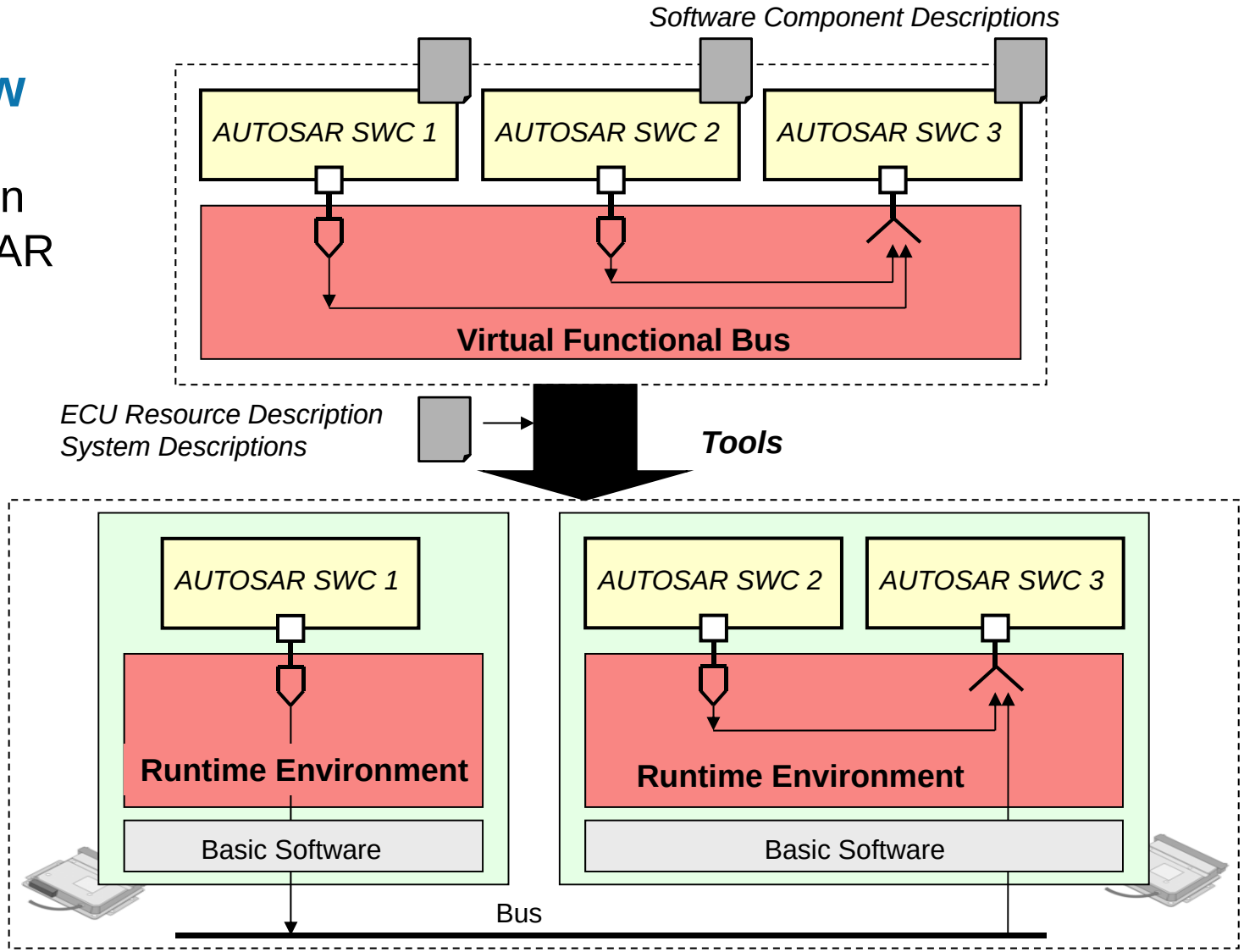
Overview

- This is
  the basic **Idea**

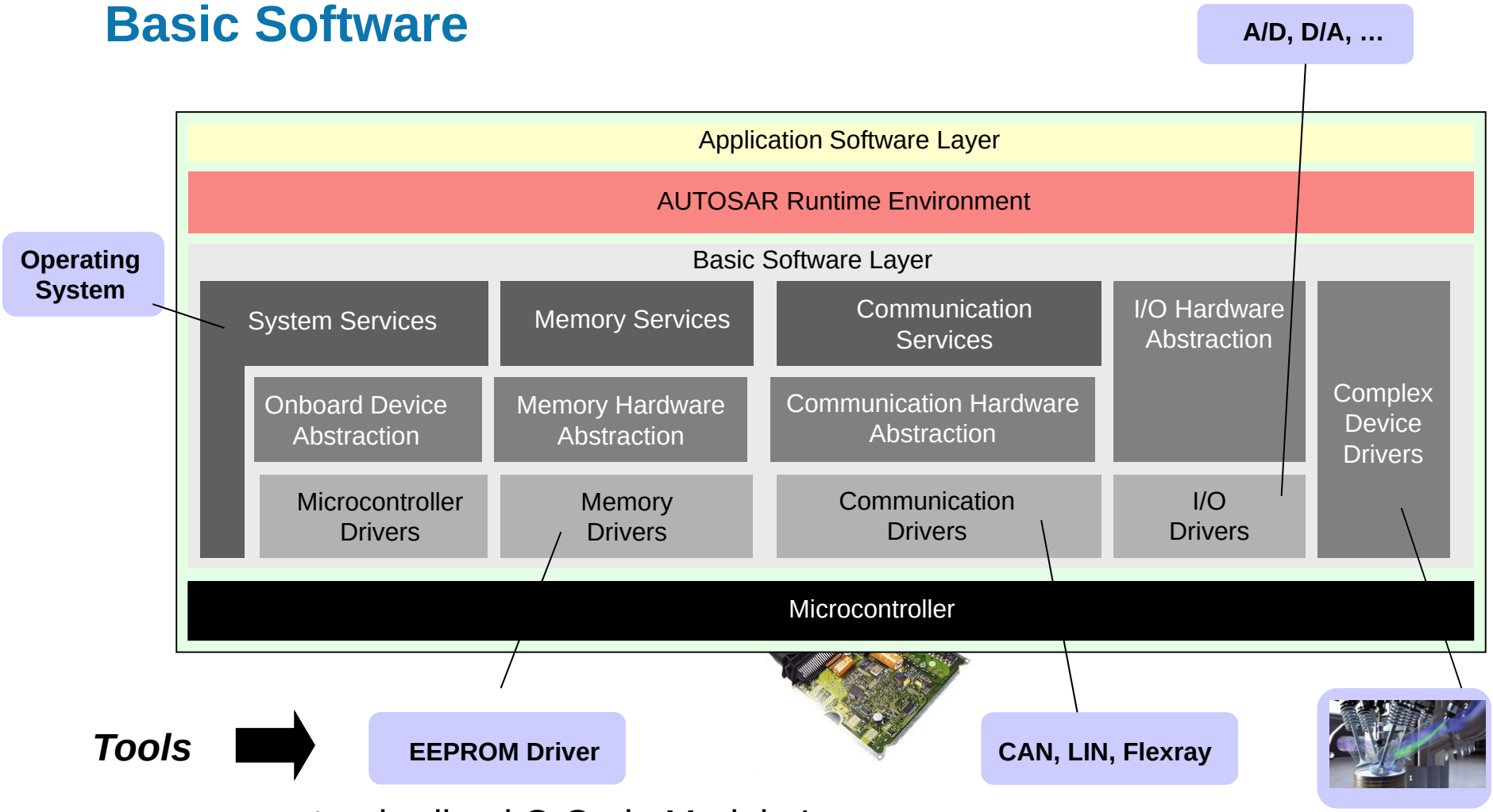


Overview

- … now in AUTOSAR Terms



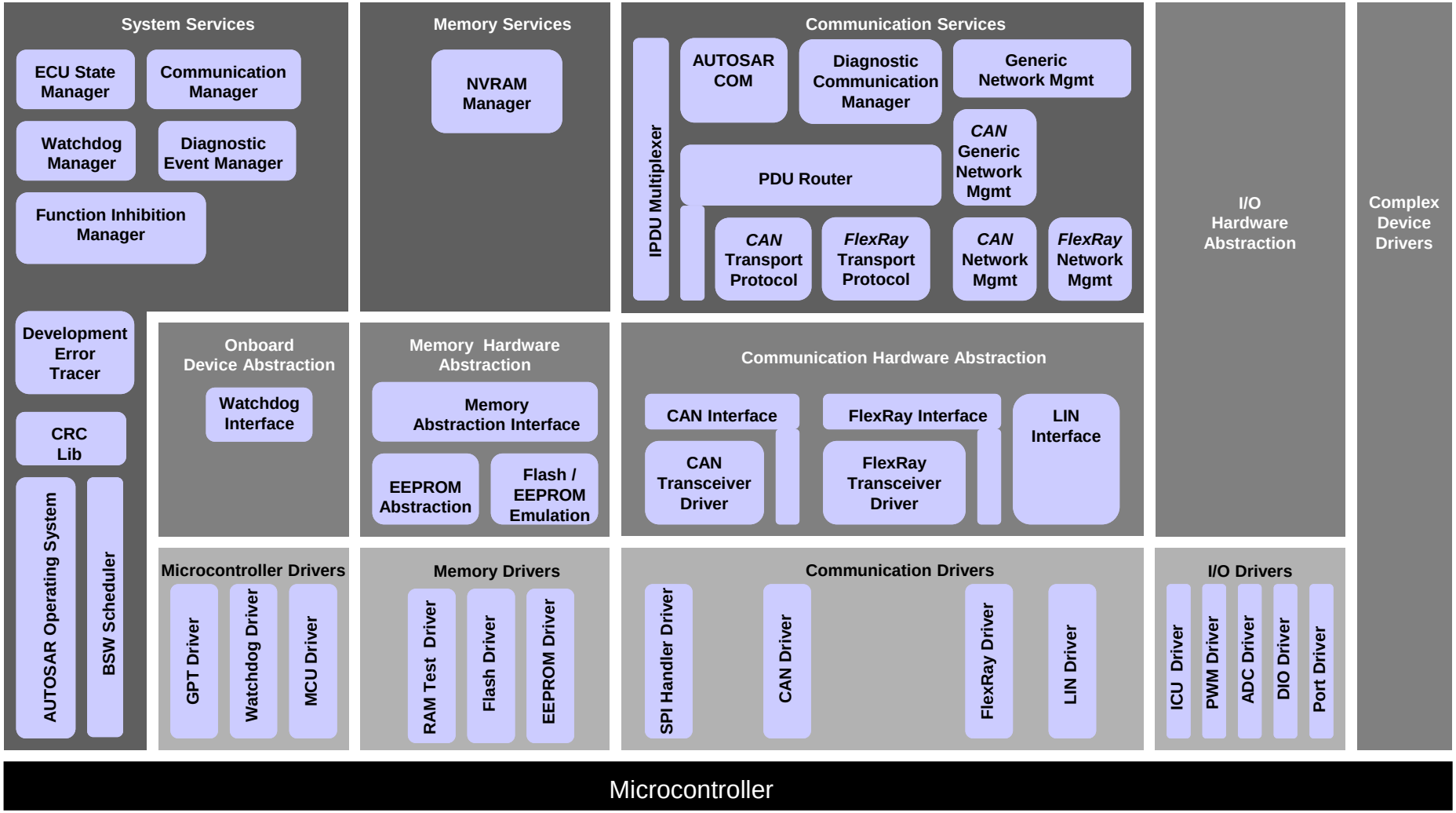
Basic Software



Tools ➡

standardized C-Code Module !

Basic Software – The complete Picture (roughly 50 Modules)



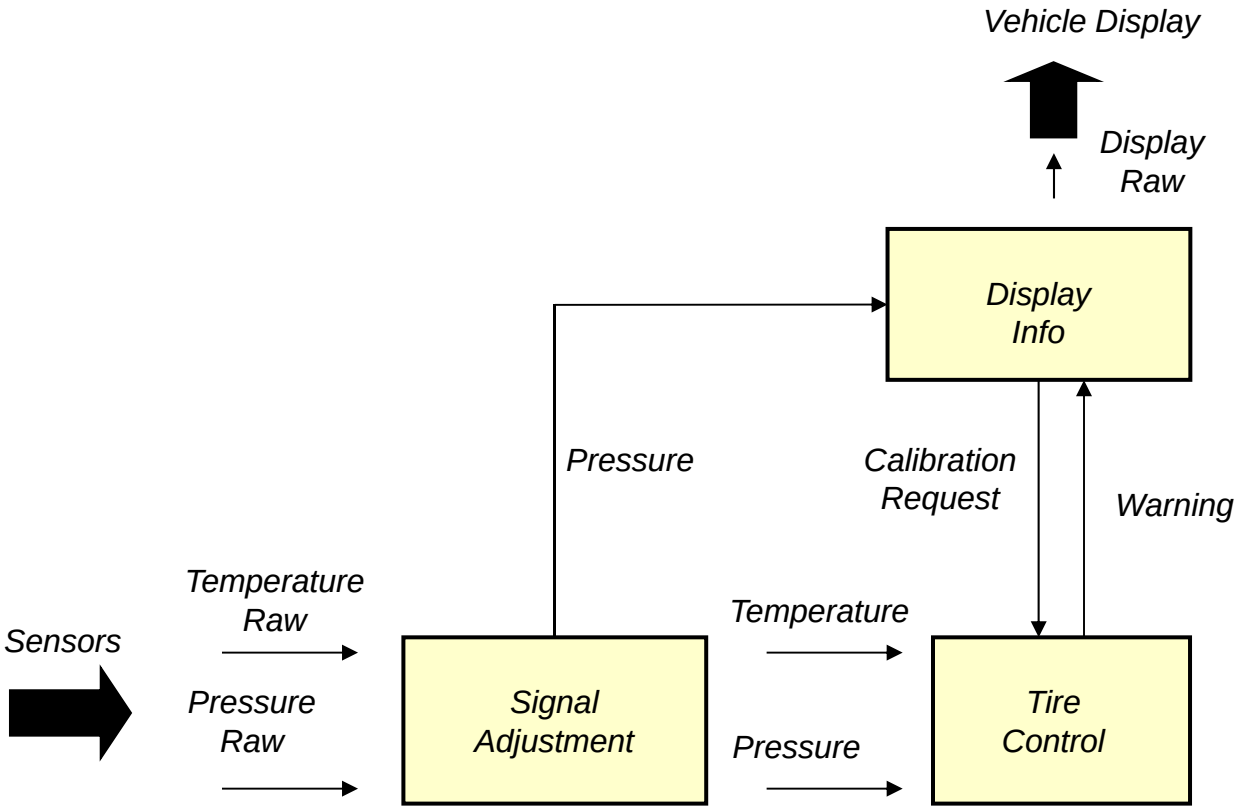
SYSTECS

Part 2: AUTOSAR - Example

Introduction

An Example

- **Measuring and Displaying the Tire Pressure**



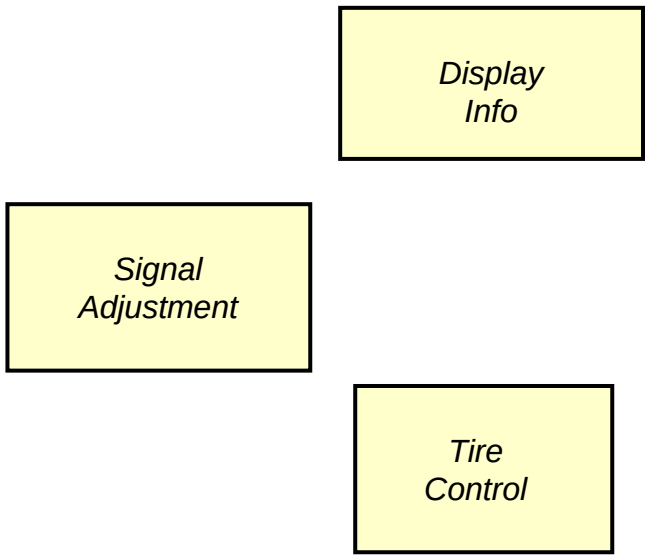
SYSTECS

Part 2: AUTOSAR - Example

Step 1: Edit the Software Components

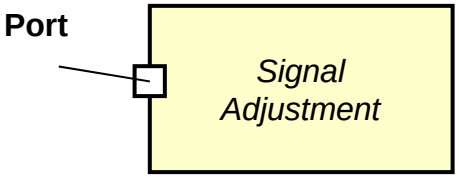
Step 1: The Software Components itself

- We define 3 **Atomic Software Components**
  - *“Signal Adjustment”*
  - *“Tire Control”*
  - *“Display Info”*

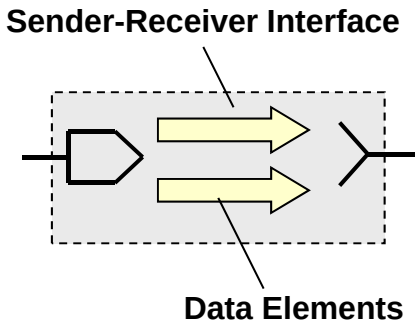


Step 1: The Ports of the Software Components

- The Software Components have to communicate, e.g. to exchange information
  - They need **Ports** to do this



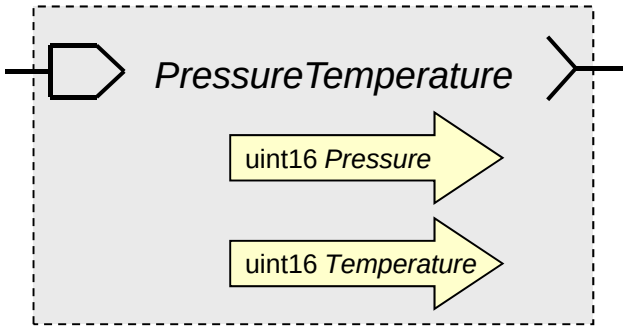
- Ports will be assigned either to
  - **Sender-Receiver Interfaces** or
  - **Client-Server Interfaces**
    - Not used in this Example !

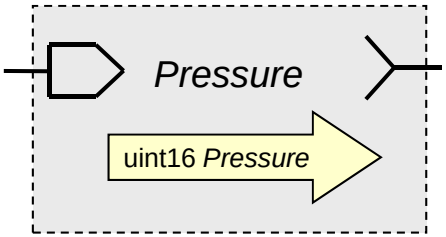


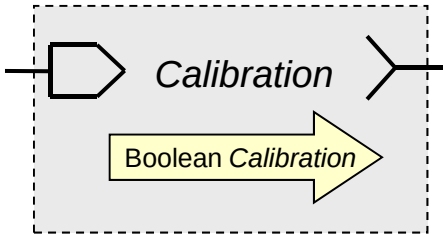
- We choose Sender-Receiver Interfaces to transport our Data

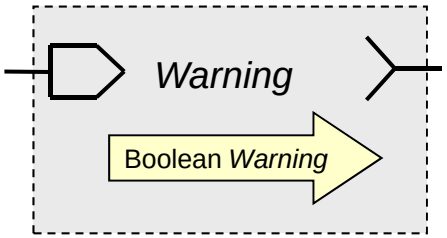
Step 1: The Ports of the Software Components

- We define the **Sender-Receiver Interfaces**
  - *PressureTemperature*
  - *Pressure*
  - *Warning*
  - *Calibration*



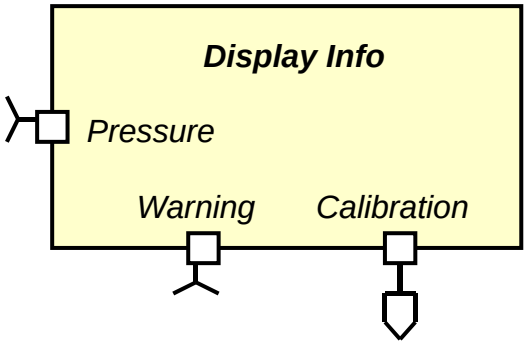


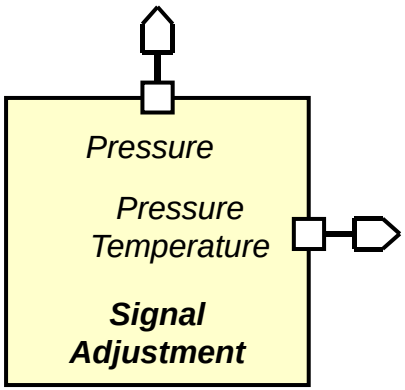


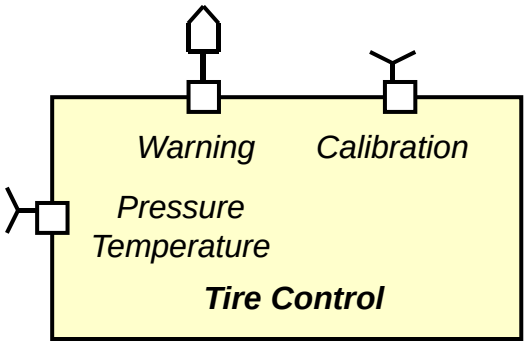


Step 1: The Ports of the Software Components

- We **assign** now the Sender-Receiver Interfaces to the Software Components
  - Ports become P-Ports or R-Ports
    - P-Ports provide information
    - R-Ports receive information







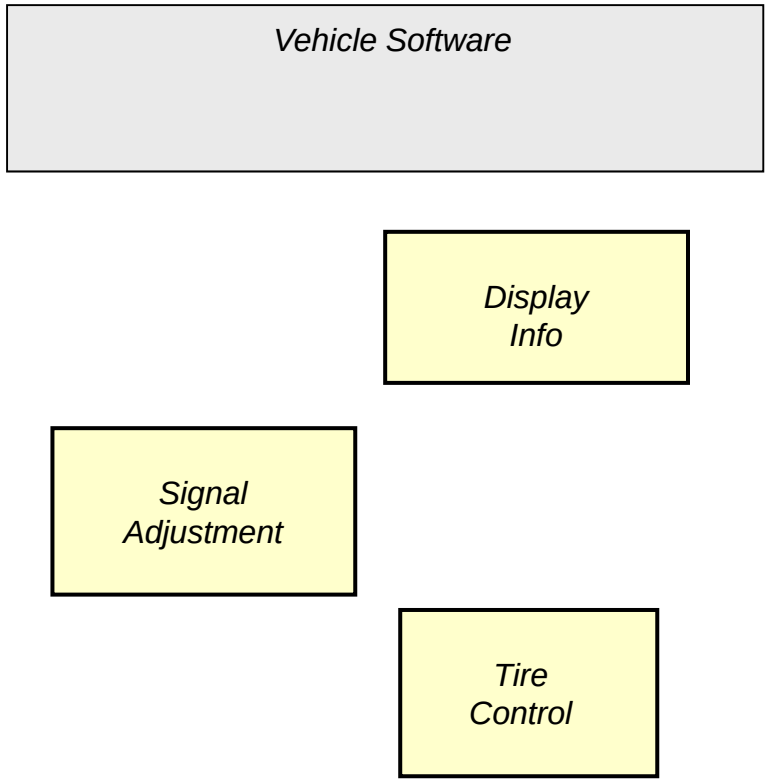
SYSTECS

Part 2: AUTOSAR - Example

Step 2: Edit the Software Component Network

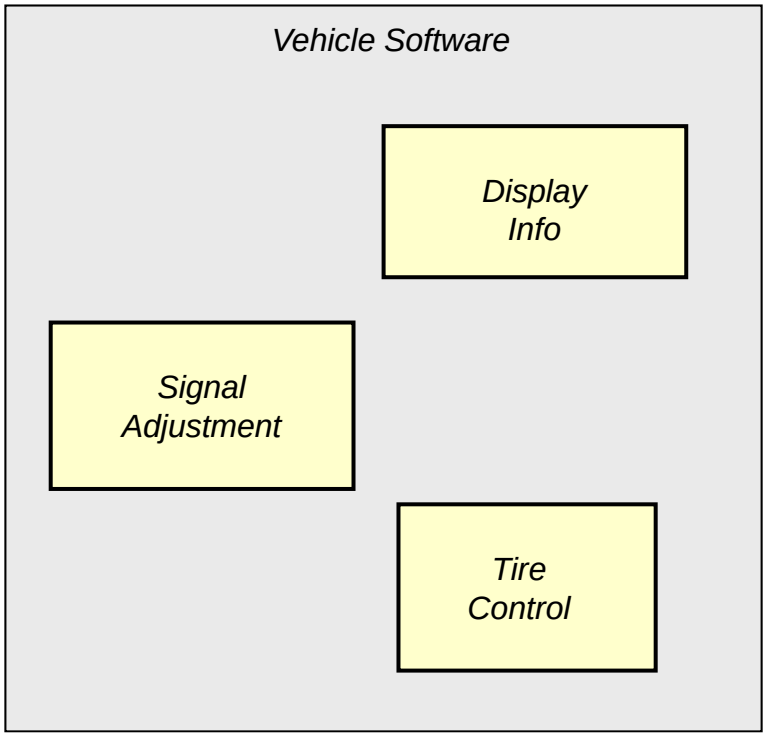
Step 2: The Software Component Network

- We define
  - A **Composition** "*Vehicle Software*"



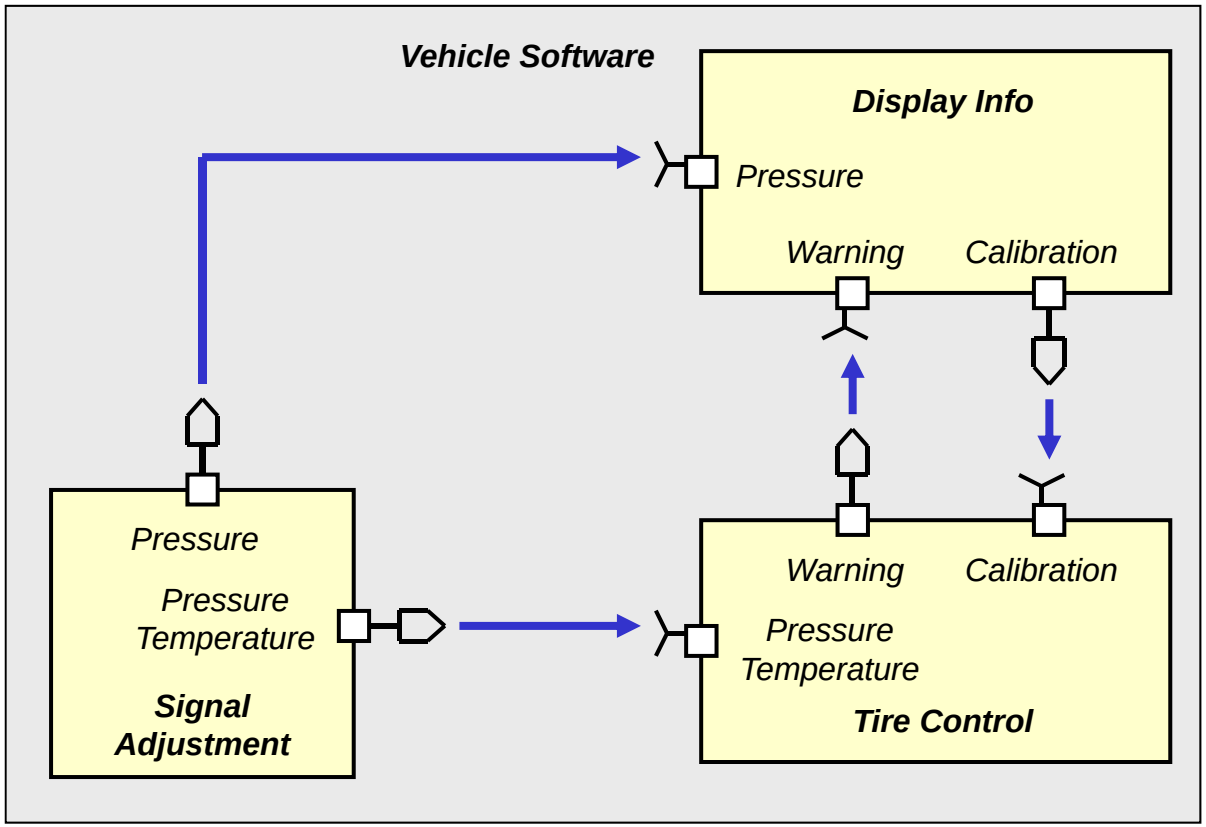
Step 2: The Software Component Network

- We assign the Software Components to the Composition

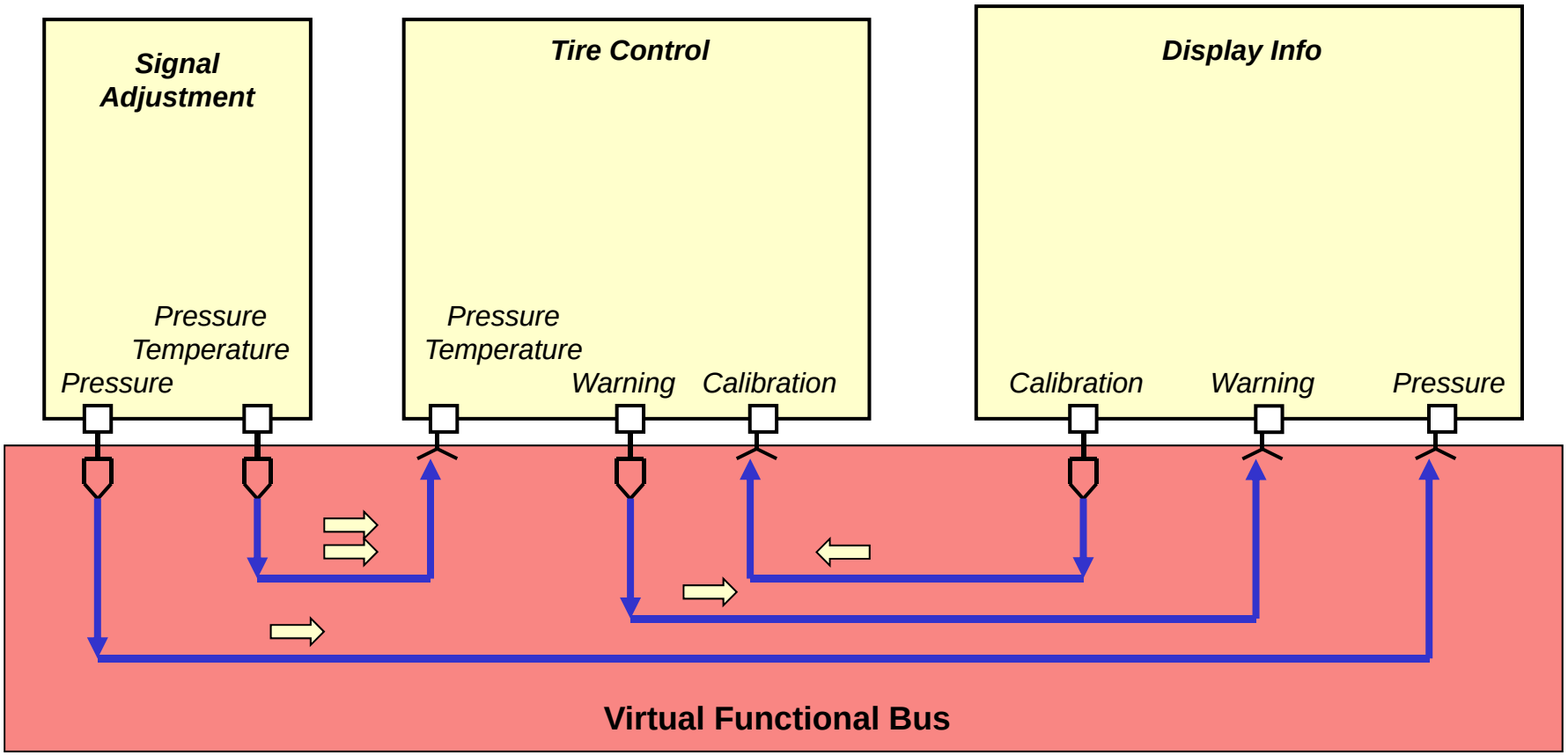


Step 2: The Software Component Network

- We take the Composition "*Vehicle Software*" and **connect** the Software Components



Step 2: The Software Component Network



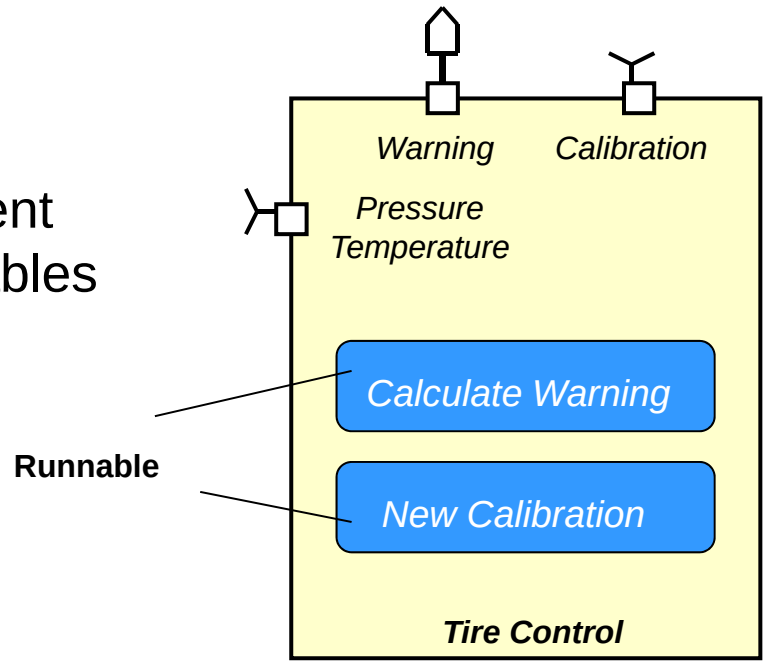
SYSTECS

Part 2: AUTOSAR - Example

Step 3-1: Edit Runnables & Events for the Software Components

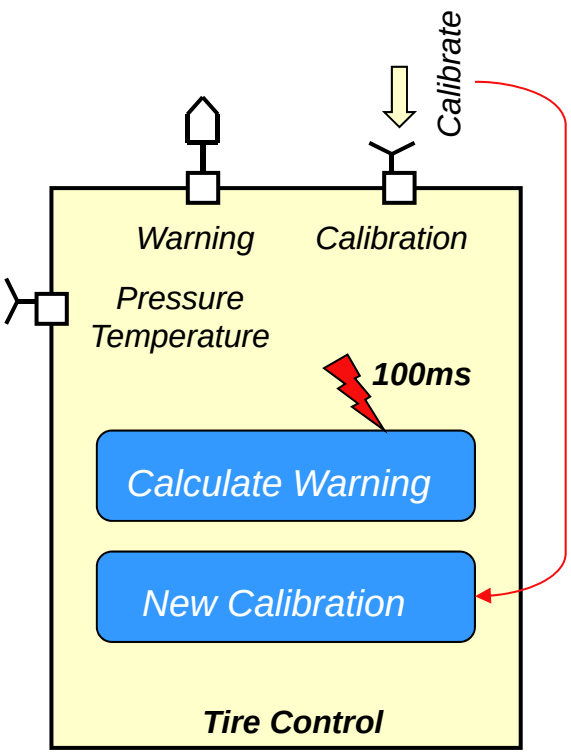
Step 3: Runnables & Events

- Where is the Code to fill the Software Components with Life ?
- This is done with **Runnables** !
- For any Atomic Software Component you can define one or more Runnables

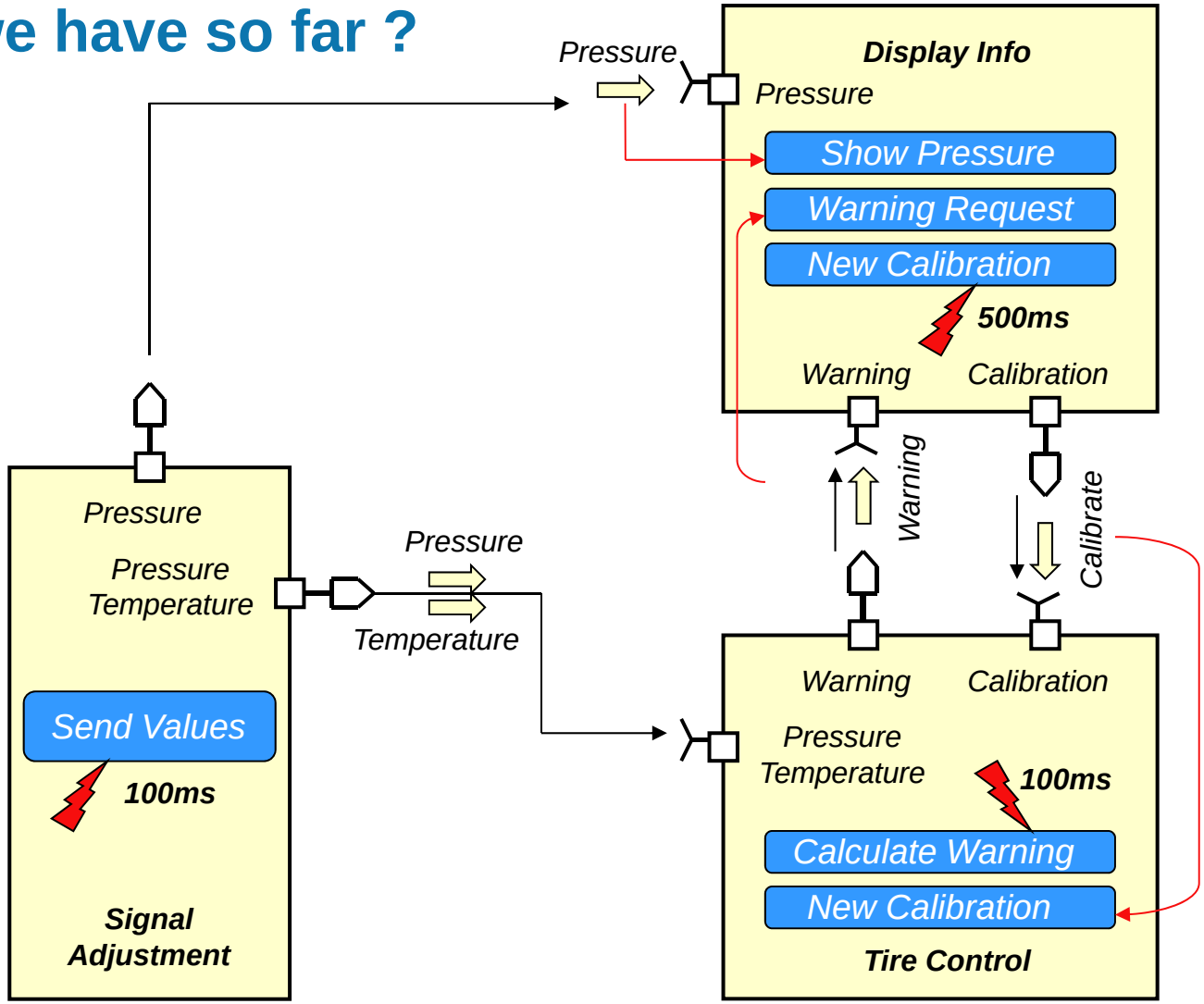


Step 3: Runnables & Events

- Runnables can be executed in the following ways
  - **Periodically triggered** with a defined Period
  - On **Data Reception**
    - with the Reception of a certain Data Element
    - The Data Element that should trigger the Runnable must be specified



What do we have so far ?



SYSTECS

Part 2: AUTOSAR - Example

Step 3-2: Edit the Code for the Software Components

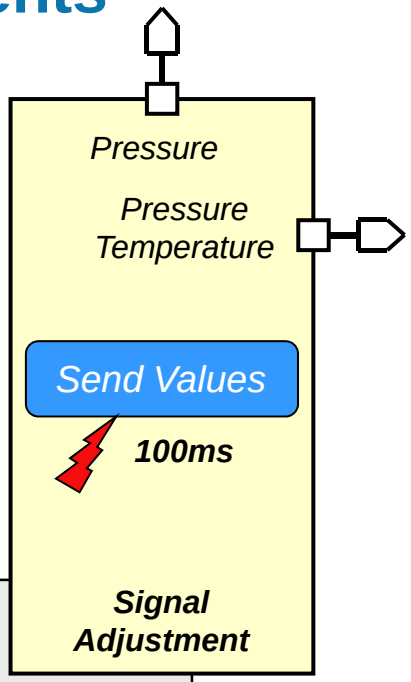
Step 3: The Code for the Software Components

- We still miss the real Code !

- Based on the Information on the previous slide, **AUTOSAR Code-Generators** may generate Templates for each Software Component
  - Header Files (fully automatically generated)
  - C/C++ Files (only generation of Templates)

Step 3: The Code for the Software Components

- Here we are:

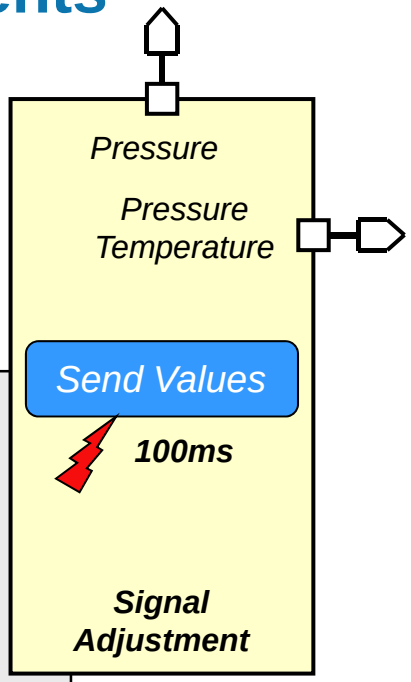


Signal_Adjustment .h

```
/* Header File for Tire Control */

/* Output Interfaces */
RTE_WRITE_Pressure_Pressure                    (IN uint16 Pressure);
RTE_WRITE_PressureTemperature_Pressure         (IN unit16 Pressure);
RTE_WRITE_PressureTemperature_Temperature      (IN unit16 Temperature);
```

Step 3: The Code for the Software Components

- Here we are:

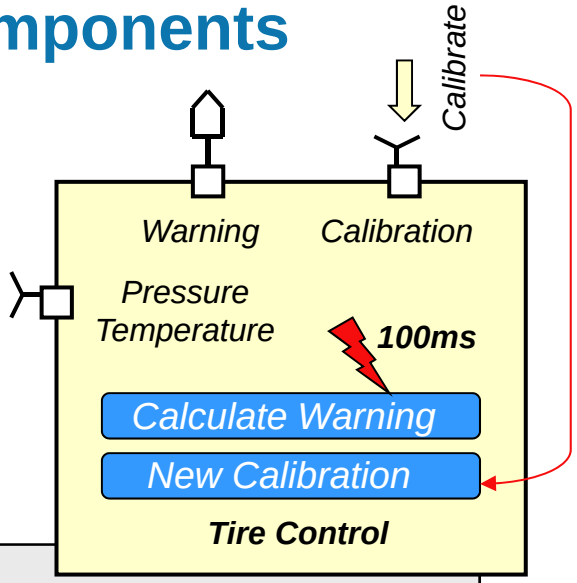


Signal_Adjustment .c

```
/* Header File for Tire Control */
#include signal_adjustment.h

/* Runnable Entities */
Void Send_Values (void);
{
uint16 sensor_temperature;
uint16 sensor_pressure;

Sensor_temperature= 20;  /* That's trick, since we have no sensor software component */
Sensor_pressure = 3      /* That's trick, since we have no sensor software component */

RTE_WRITE_PressureTemperature_Temperature     (sensor_temperature);
RTE_WRITE_PressureTemperature_Pressure        (sensor_pressure);
RTE_WRITE_Pressure_Pressure                   (sensor_pressure);
}
```

Step 3: The Code for the Software Components

- Here we are:

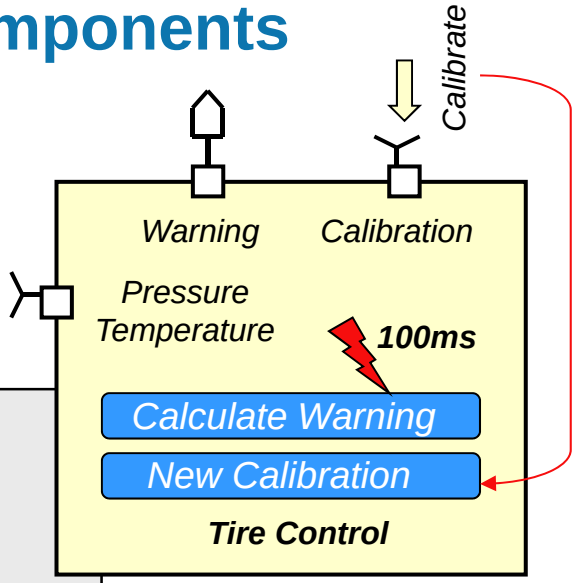


Tire_Control .h

```
/* Header File for Tire Control */

/* Input Interfaces */
RTE_READ_Calibration_Calibrate                    (OUT boolean* Calibrate);
RTE_READ_PressureTemperature_Pressure             (OUT unit16* Pressure);
RTE_READ_PressureTemperature_Temperature          (OUT unit16* Temperature);

/* Output Interfaces */
RTE_WRITE_Warning_Warning                         (IN boolean Warning);
```

Step 3: The Code for the Software Components

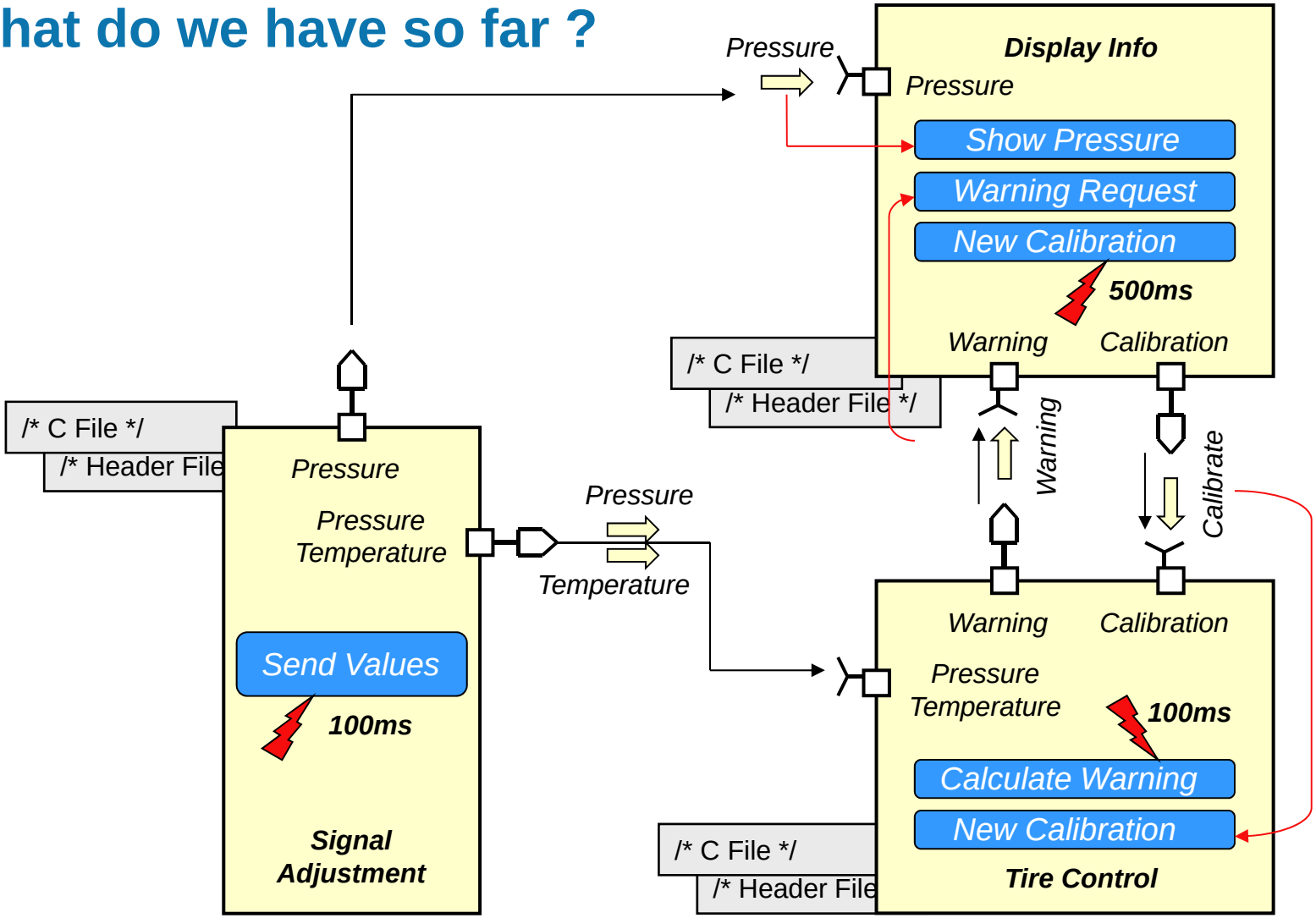
- Here we are:



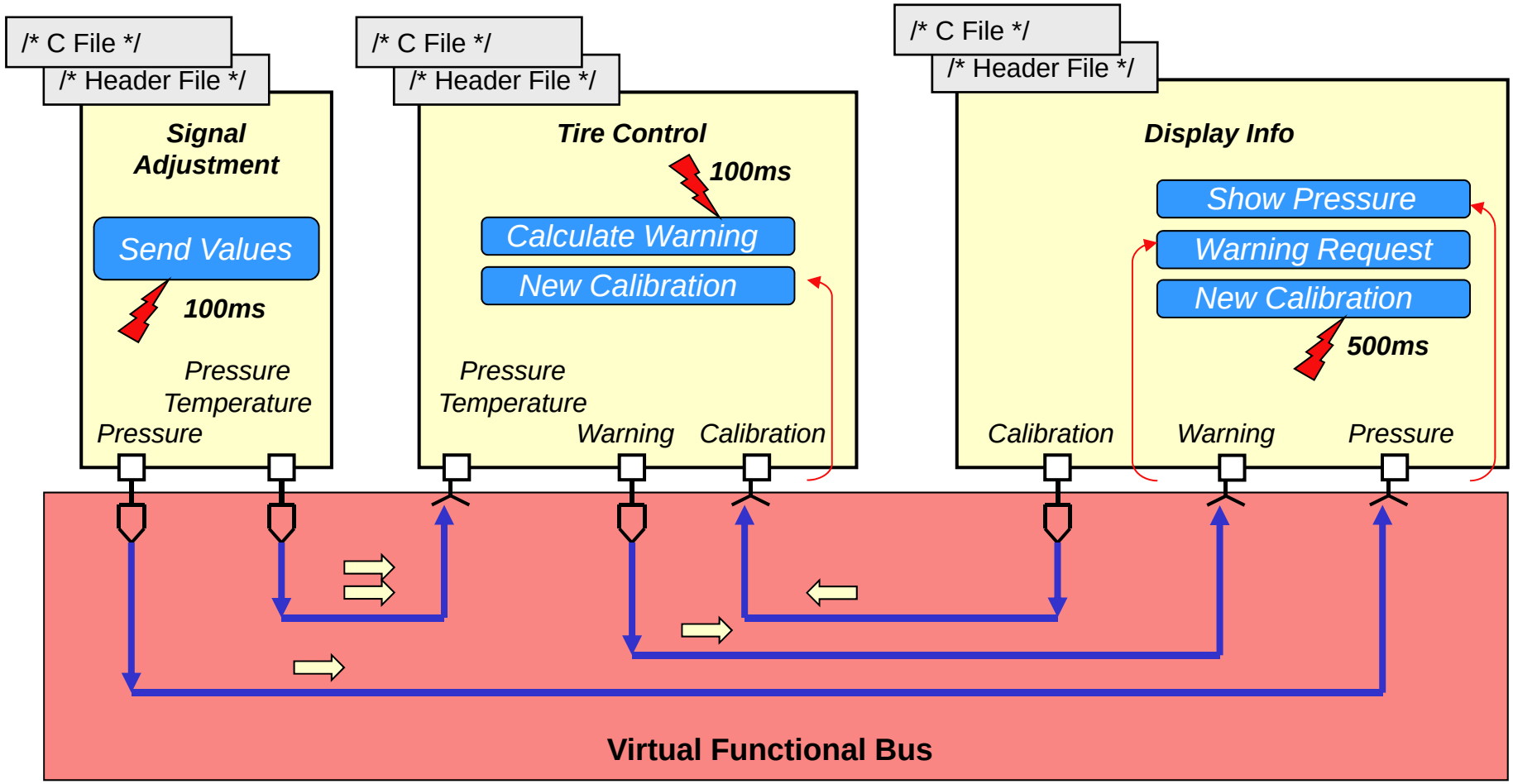
Tire Control .c

```
/* Header File for Tire Control */
#include tire_control.h

/* Runnable Entities */
Void Calculate_Warning (void);
{
uint16 local_temperature;
uint16 local_pressure;

RTE_READ_PressureTemperature_Temperature        (& local_temperature);
RTE_READ_PressureTemperature_Pressure           (& local_pressure);

If (local_pressure < 3)
{
RTE_WRITE_Warning_Warning                       (1);
}
}
```

What do we have so far ?



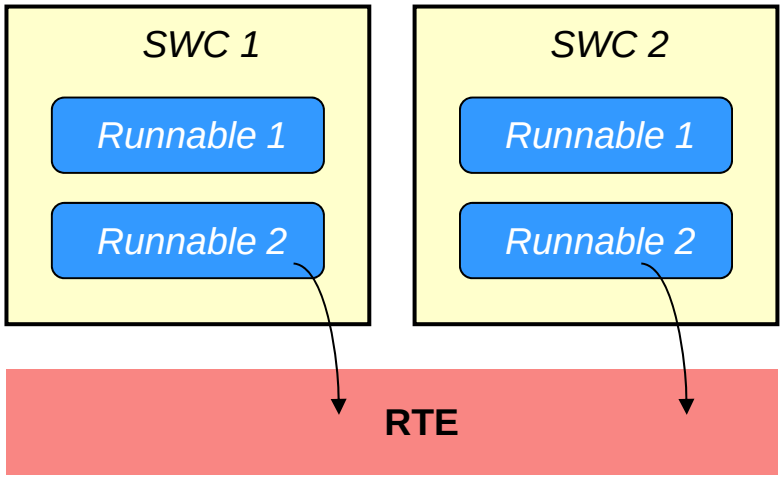
Step 3: Summary



Implementation of a Software Component

Principles

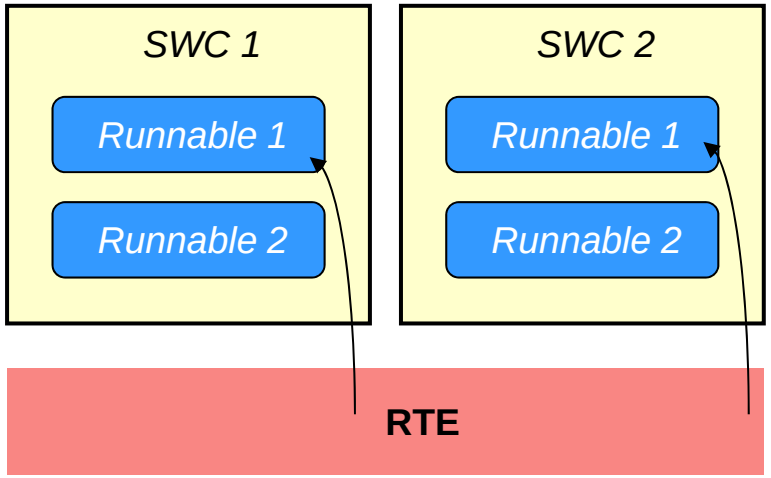
- **Interaction Mechanisms**
  - SWCs call RTE functions through a well-defined API



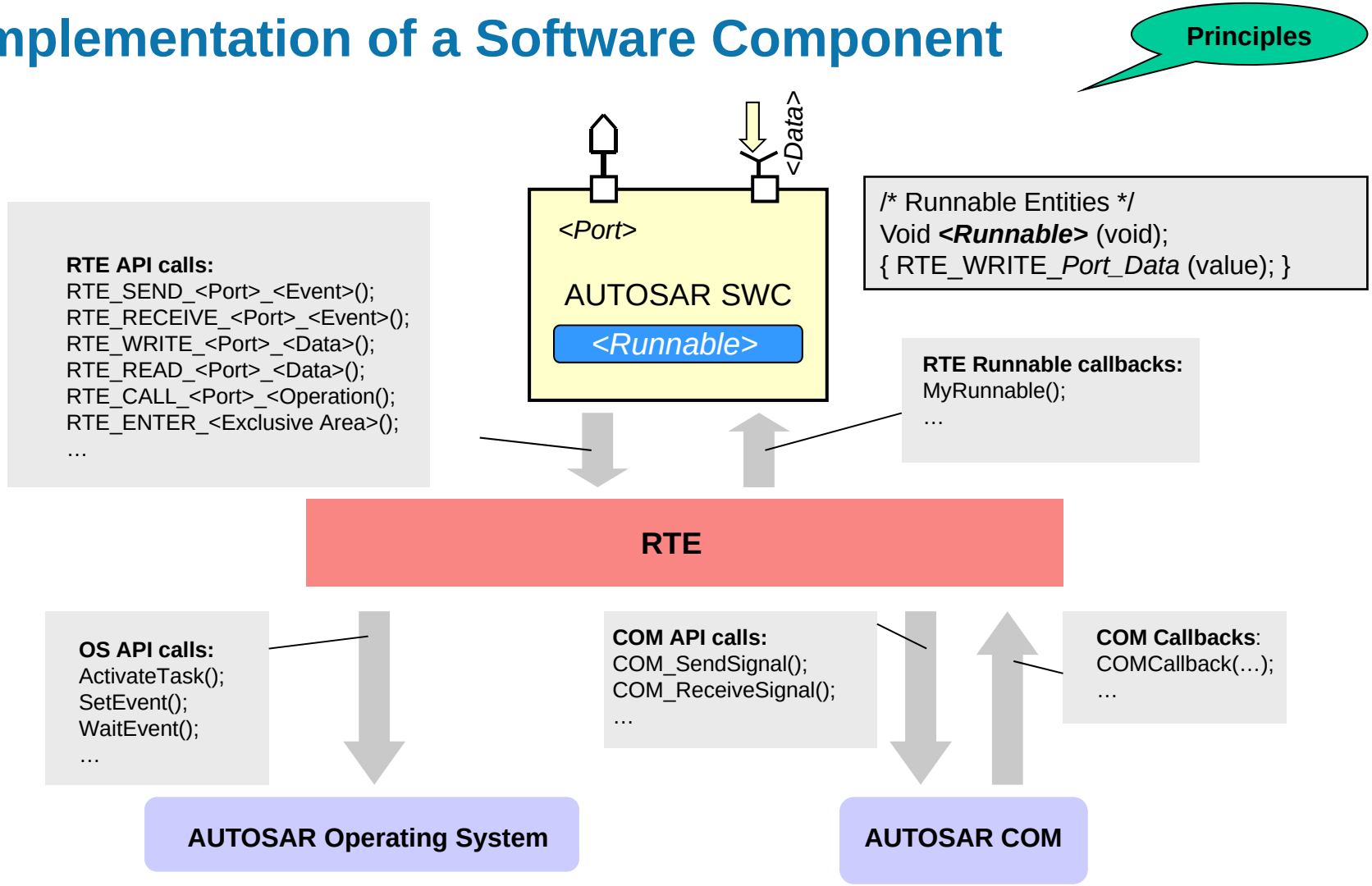
Implementation of a Software Component

Principles

- **Interaction Mechanisms**
  - The **RTE calls SWCs** by invoking user defined *Runnables*
  - This is simply a Call-back from the RTE to user code

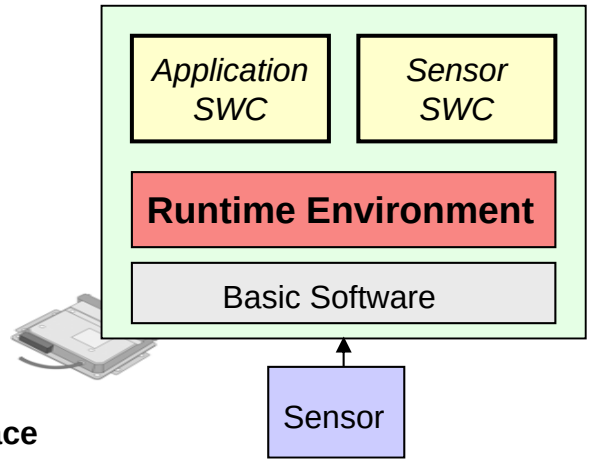


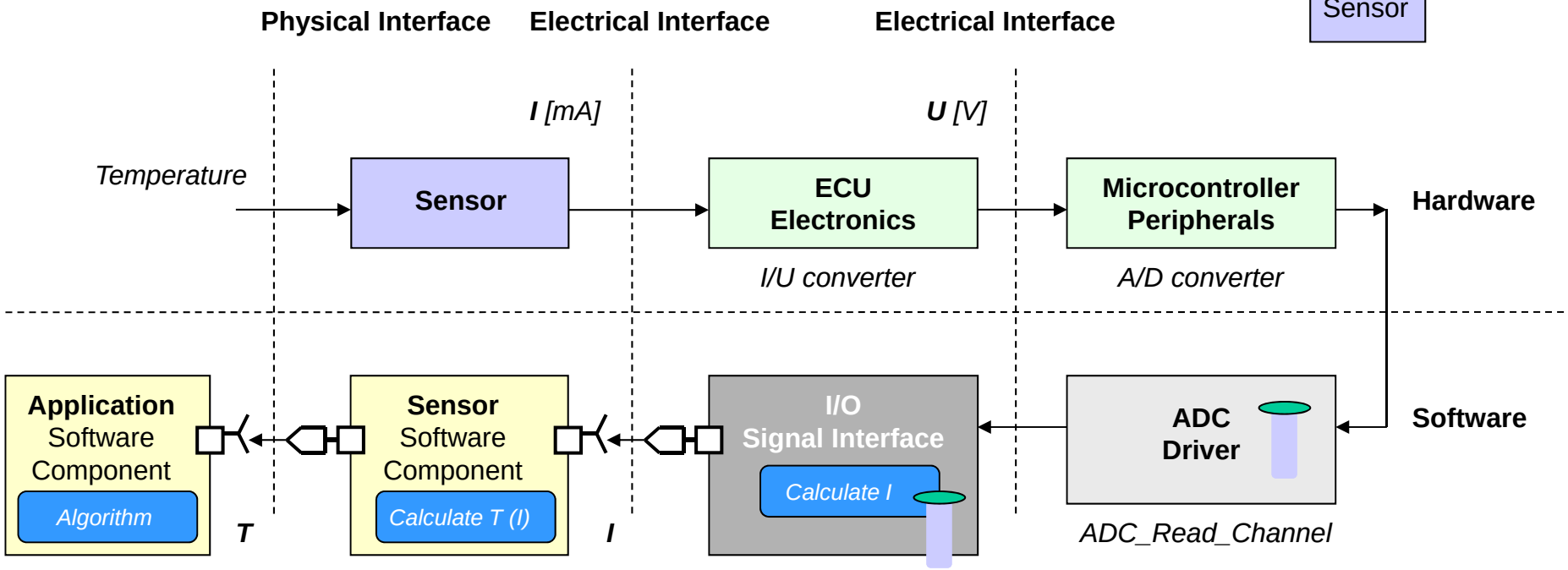
Implementation of a Software Component



Implementation of a SW Component

- **Example: Sensor** Software Component





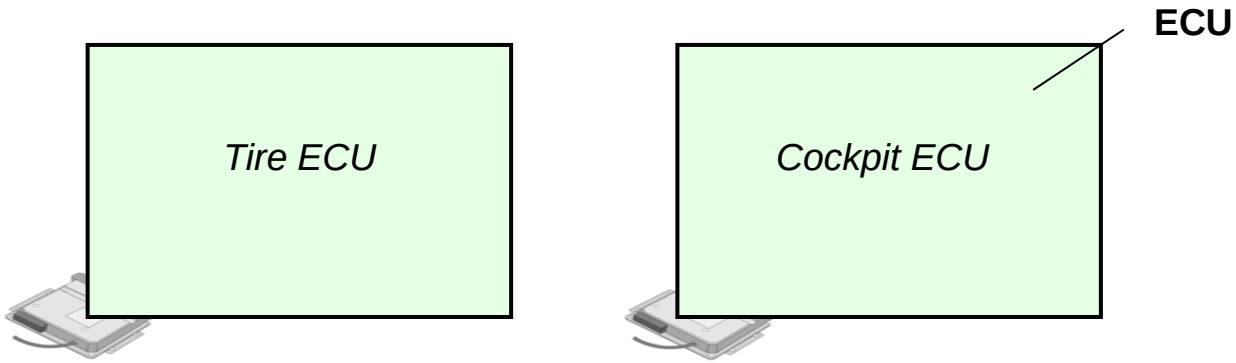
SYSTECS

Part 2: AUTOSAR - Example

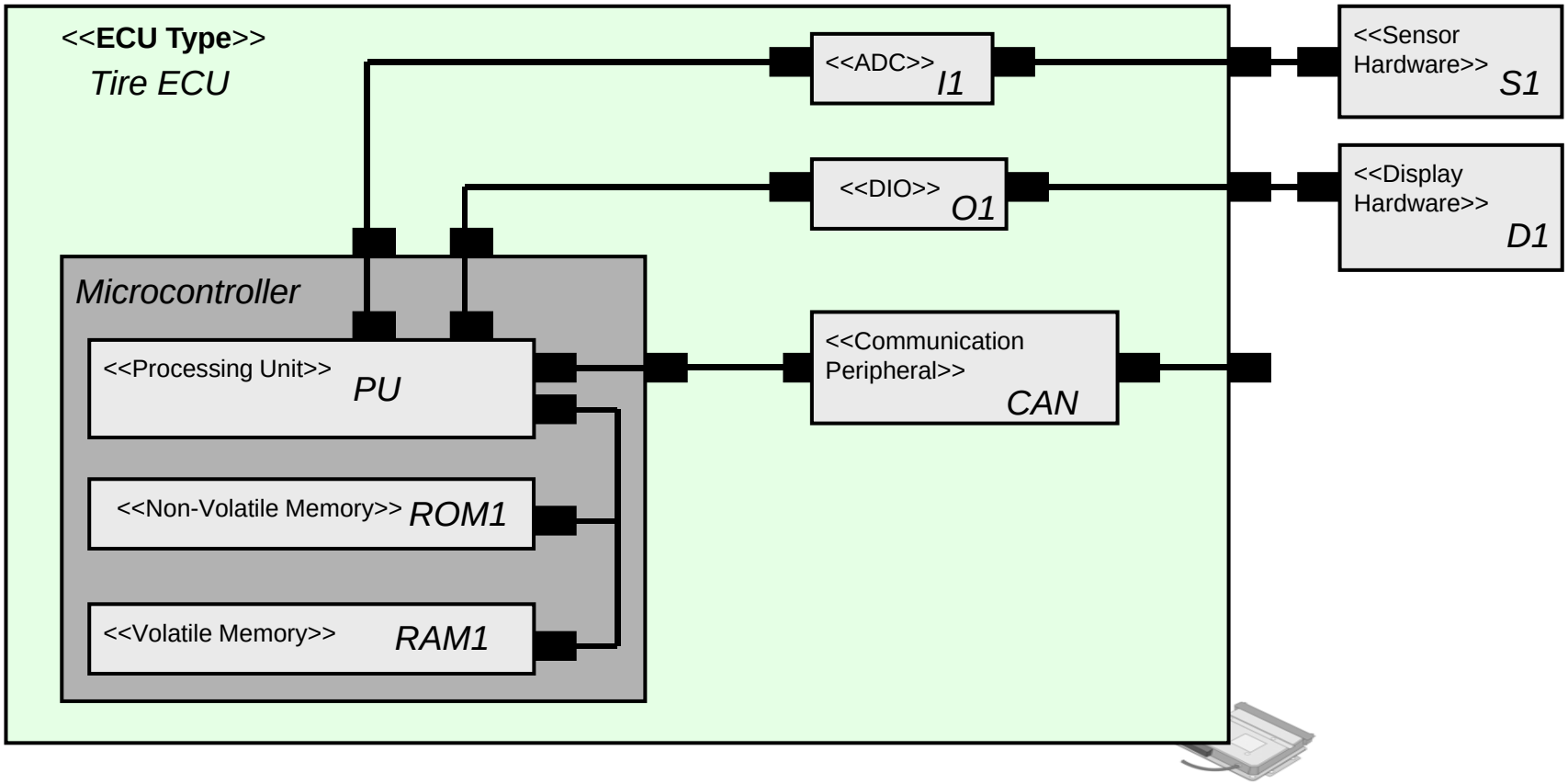
Step 4-1: Edit the ECUs

Step 4: Definition of the ECUs

- For this Example we need **2 ECUs**:
  - *Tire ECU*
  - *Cockpit ECU*



Step 4: Definition of the ECUs



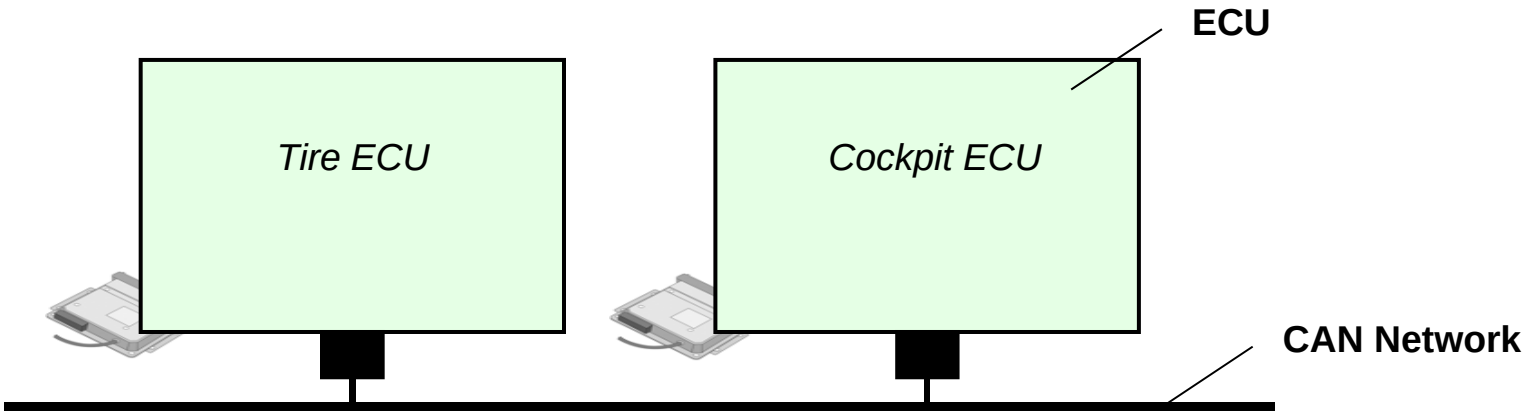
SYSTECS

Part 2: AUTOSAR - Example

Step 4-2: Edit the ECU Network

Step 4: Definition of a Network of ECUs

- Additionally we have to define a **Network** and we have to assign the 2 ECUs to it
  - We use a CAN Network, called *High Speed CAN*



High Speed CAN

Bus Type:	CAN
CAN Addressing Mode:	Extended
Communication Speed:	500.000 Bits/s
Protocol Name:	-
Protocol Version:	2.0b

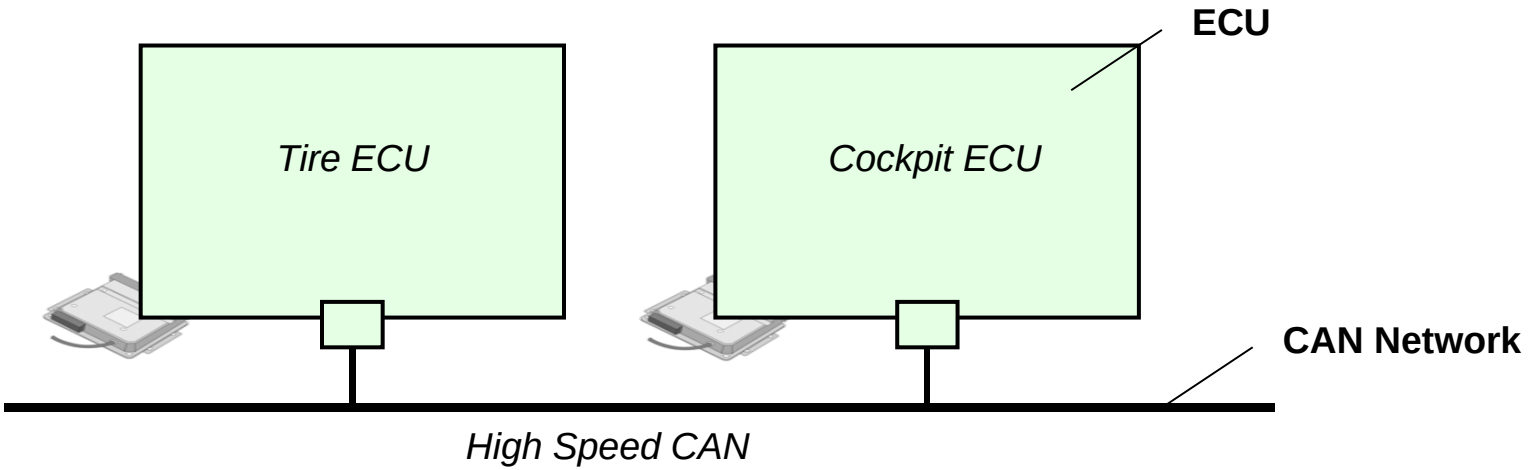
SYSTECS

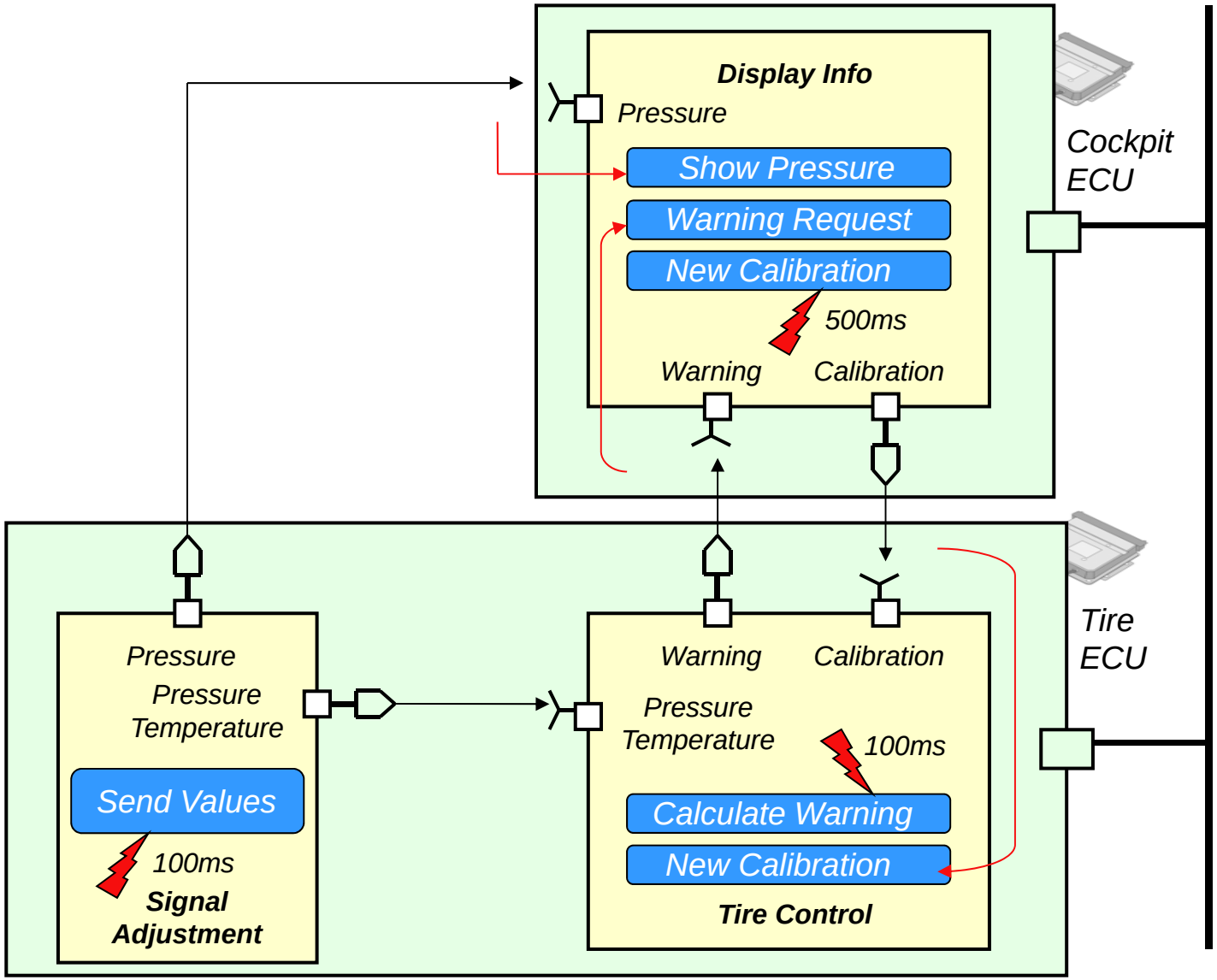
Part 2: AUTOSAR - Example

Step 5: Mapping Software Components to ECUs

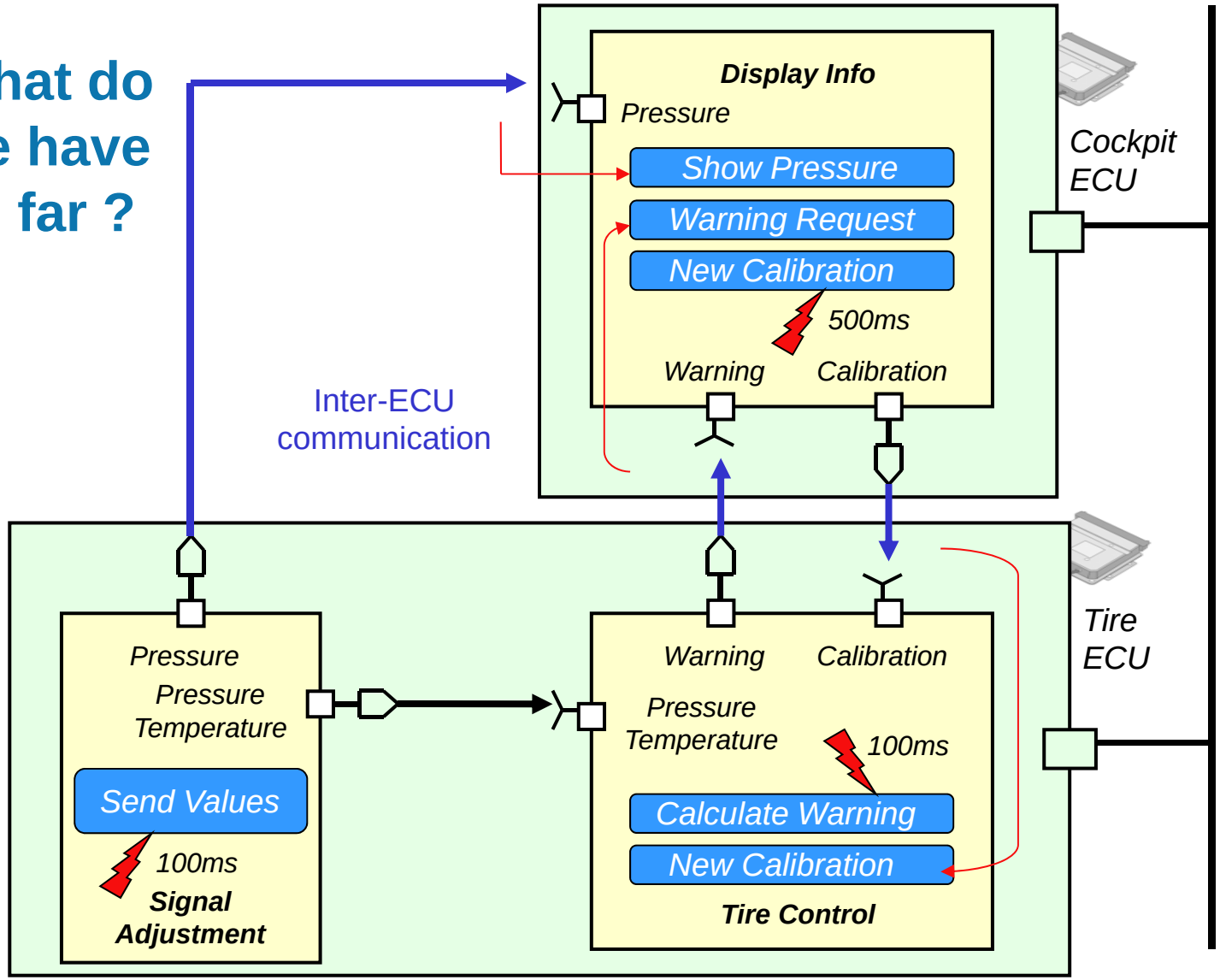
Step 5: Mapping Software Components to ECUs

- Now we can **map** one or more Software Components to one ECU

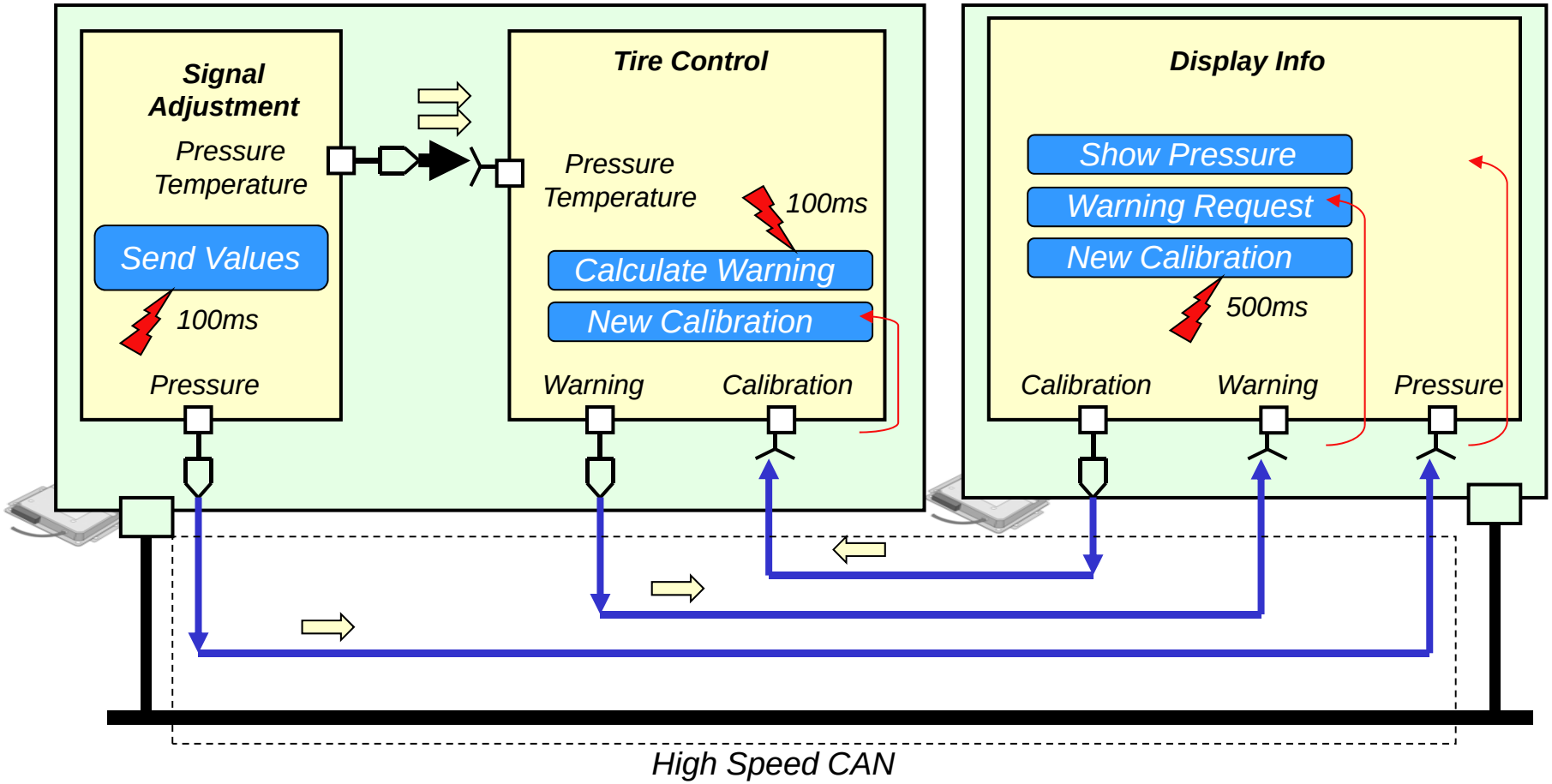



Display Info
Pressure
Show Pressure
Warning Request
New Calibration
500ms
Warning
Calibration
Cockpit ECU
Pressure
Pressure
Temperature
Send Values
100ms
Signal Adjustment
Warning
Calibration
Pressure
Temperature
100ms
Calculate Warning
New Calibration
Tire Control
Tire ECU

What do we have so far ?



Step 5: Summary (Just a redraw of the previous figure …)



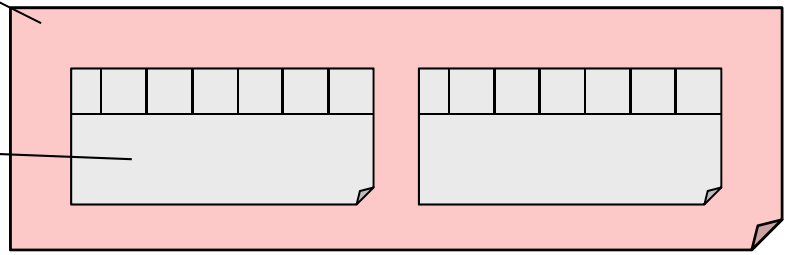
SYSTECS

Part 2: AUTOSAR - Example

Step 6: Edit Frames

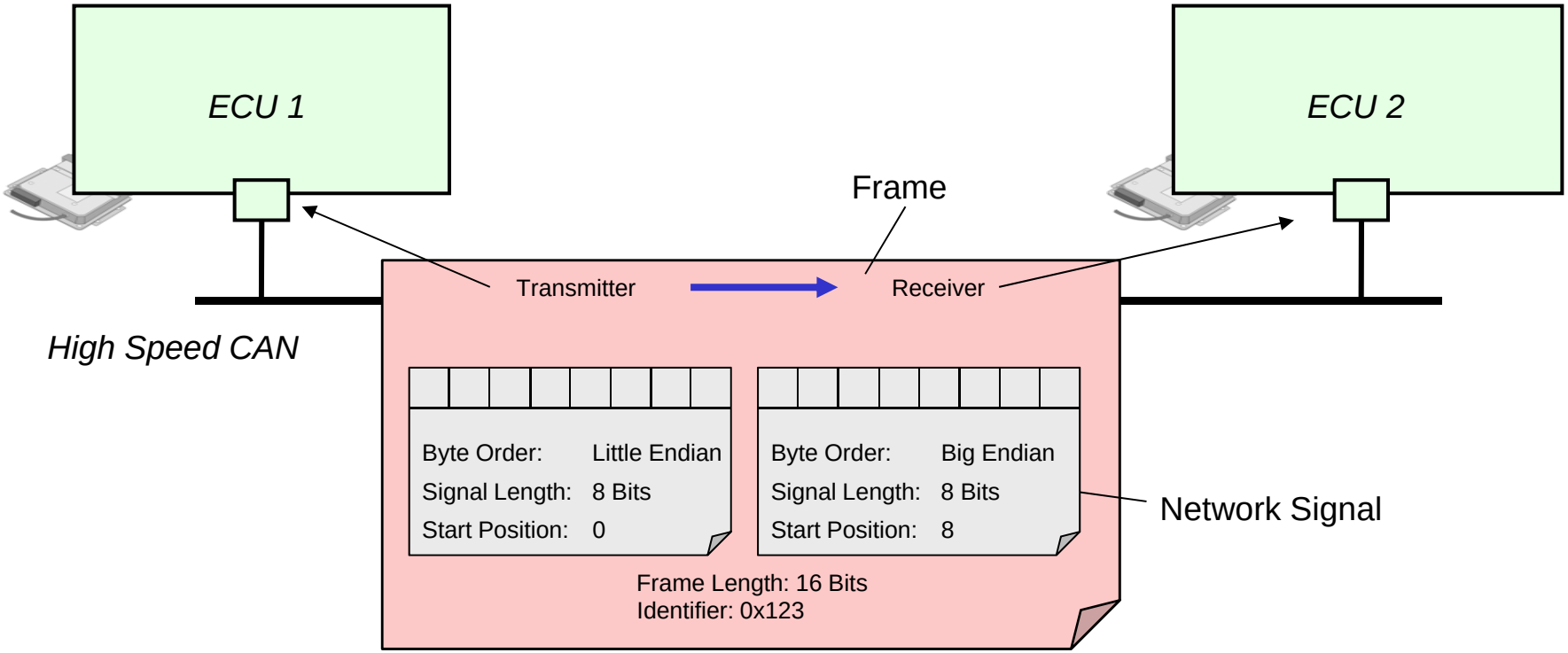
Step 6: Frames for Inter-ECU communication

- Before any inter-ECU communication, e.g. external Connections can be configured we need to define the basic **Frames**
  - It contains a number of **Network Signals**
    - Define the structure of a Frame to be sent across the Network
    - But **not** the content of the Frame



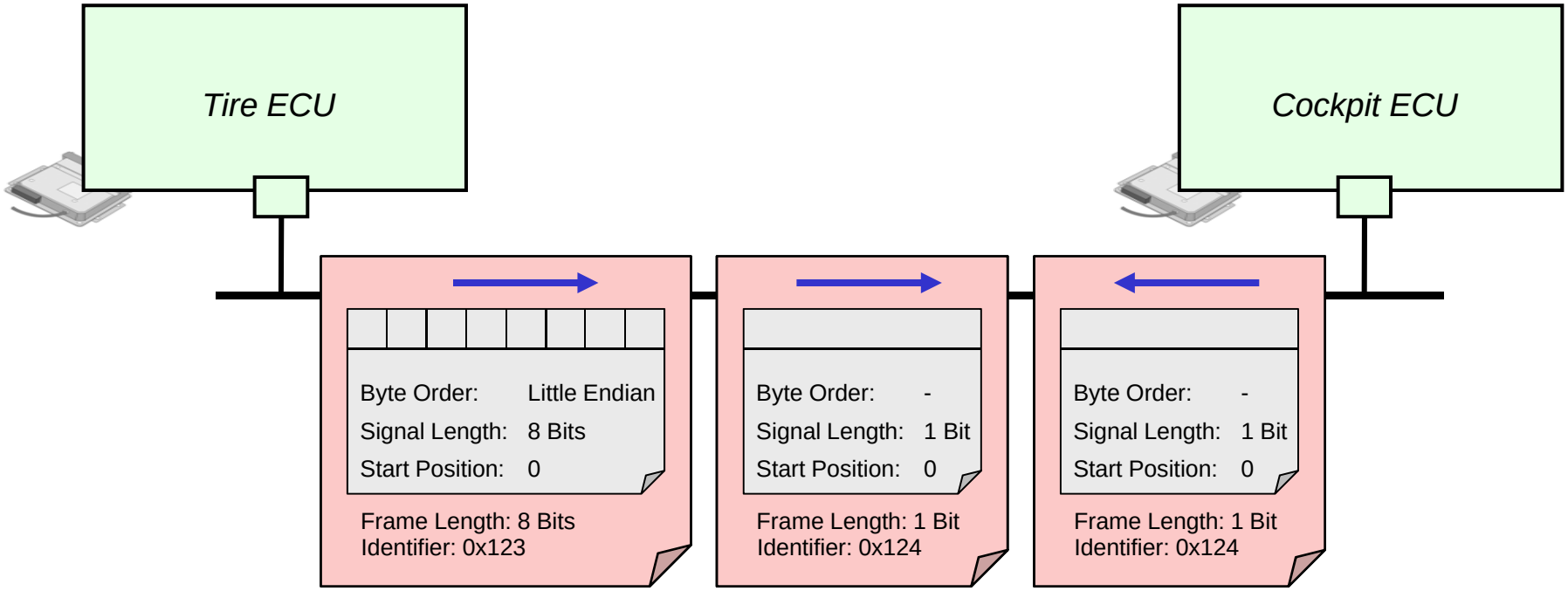
Step 6: Frames for Inter-ECU communication

- Definition of a Frame
  - General Approach



Step 6: Frames for Inter-ECU communication

- We have 3 external connections
  - *Pressure, Warning, Calibration*
- So we define for each one a Frame (Communication Matrix)

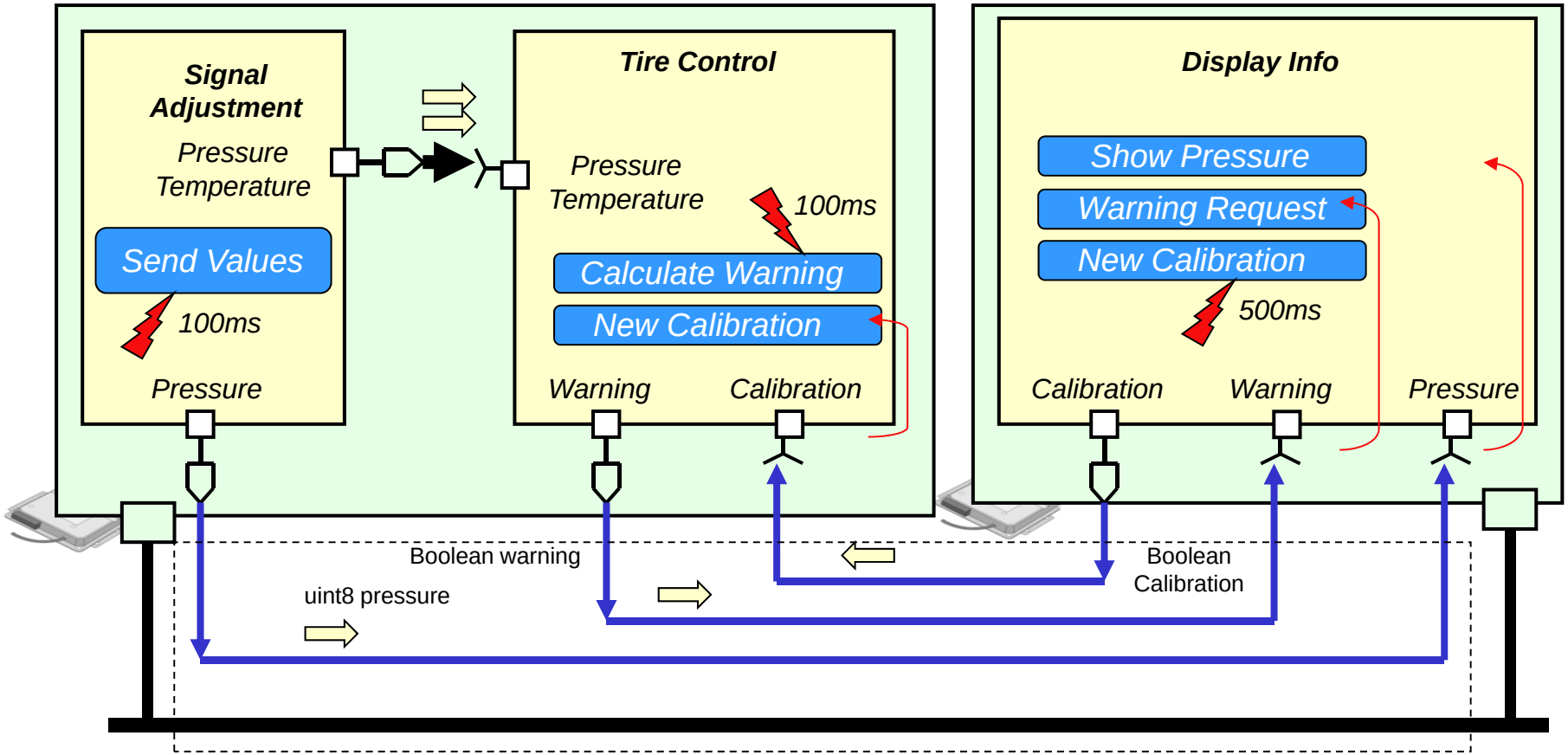


SYSTECS

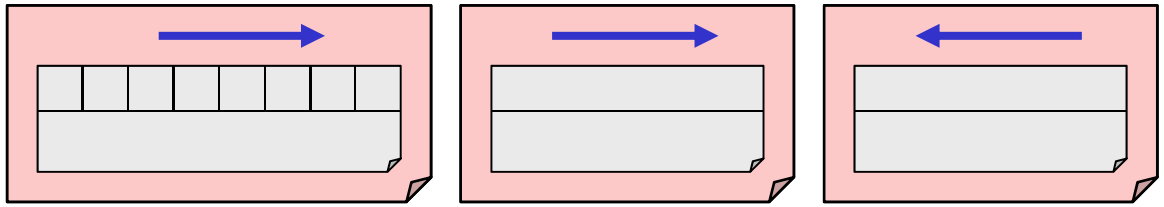
Part 2: AUTOSAR - Example

Step 7: Map Data for Inter-ECU Communication

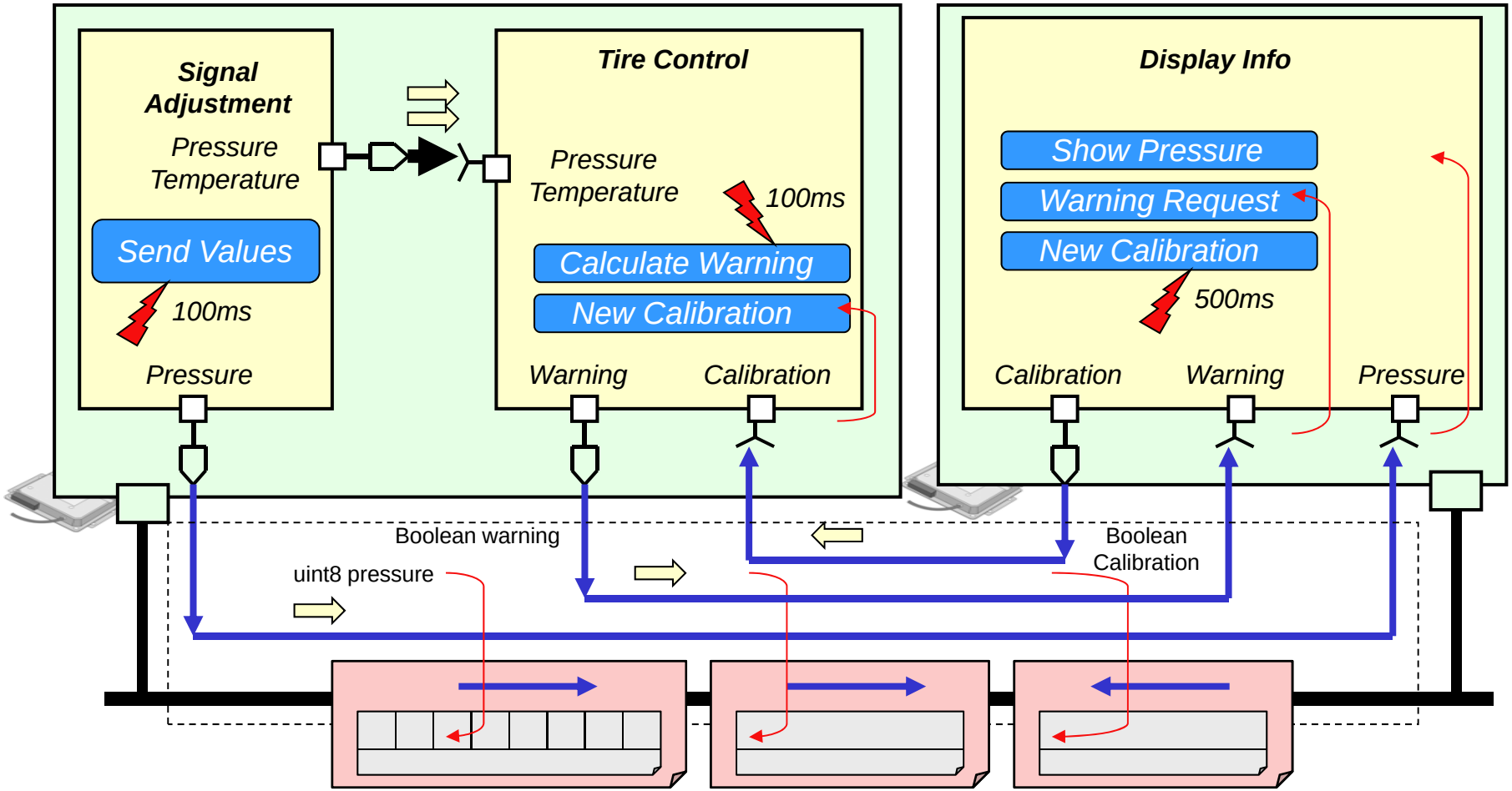
That was the Result of Step 5



That was the Result of Step 6



That's the Result of this Step 7



SYSTECS

Part 2: AUTOSAR - Example

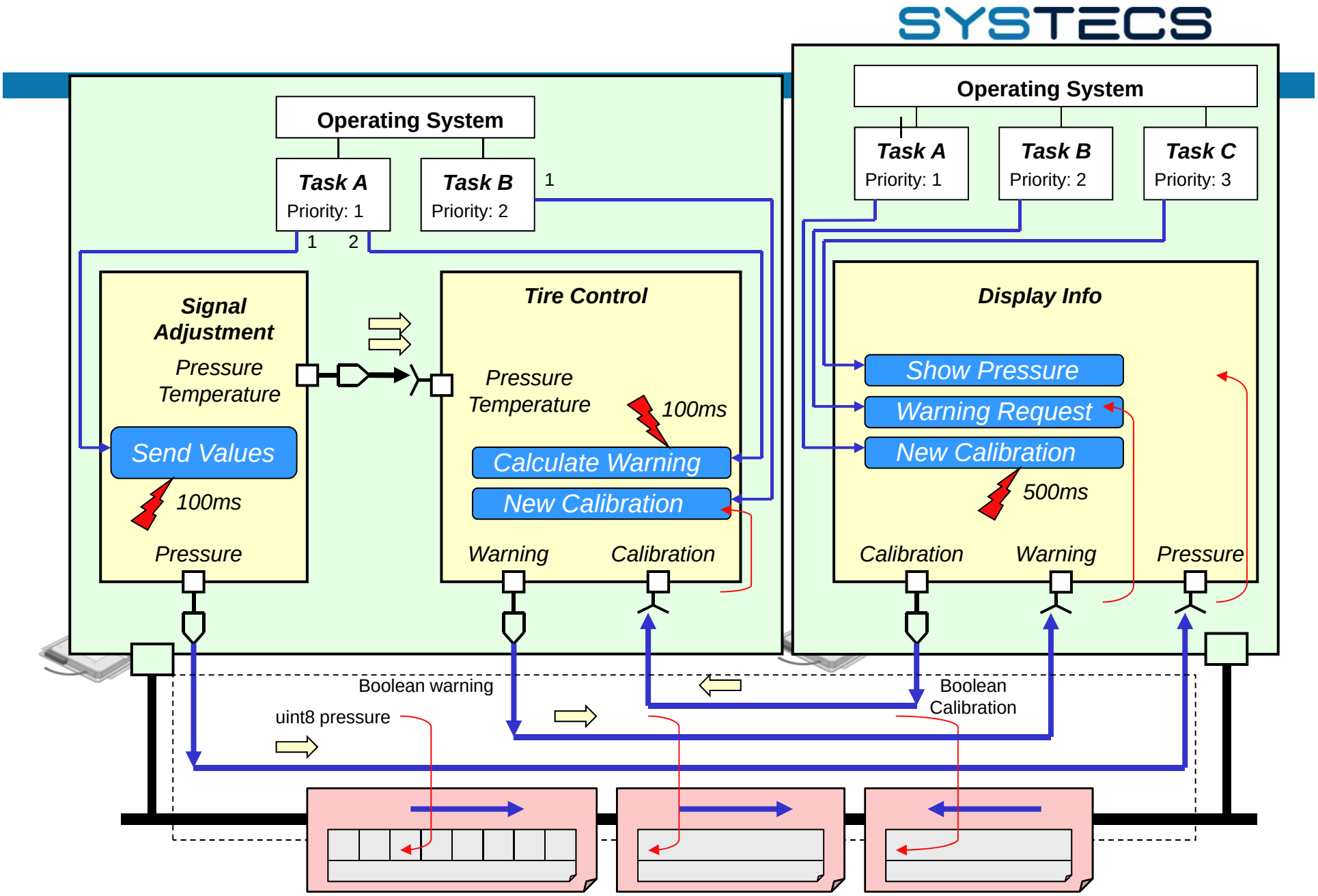
Step 8: Configuring the ECUs - Map Runnables to Tasks

Step 8: Task Mapping for Runnables

- The Runnables are triggered by Data Reception or Cyclic

- This is realized by an **Operating System** with
  - **Tasks**
  - Events
  - Alarms
  - …

- We have to decide **which Runnable** is activated **by which Task** !


SYSTECS
Operating System
Task A
Priority: 1
Task B
Priority: 2
1
1
2
Signal Adjustment
Pressure
Temperature
Send Values
100ms
Pressure
Tire Control
Pressure
Temperature
100ms
Calculate Warning
New Calibration
Warning
Calibration
Operating System
Task A
Priority: 1
Task B
Priority: 2
Task C
Priority: 3
Display Info
Show Pressure
Warning Request
New Calibration
500ms
Calibration
Warning
Pressure
Boolean warning
uint8 pressure
Boolean
Calibration

SYSTECS

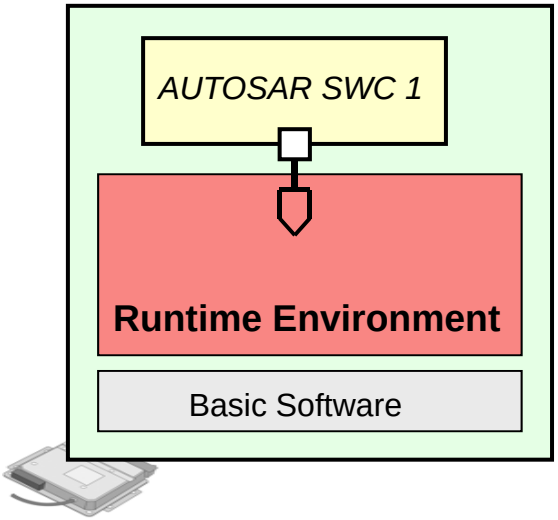
Part 2: AUTOSAR - Example

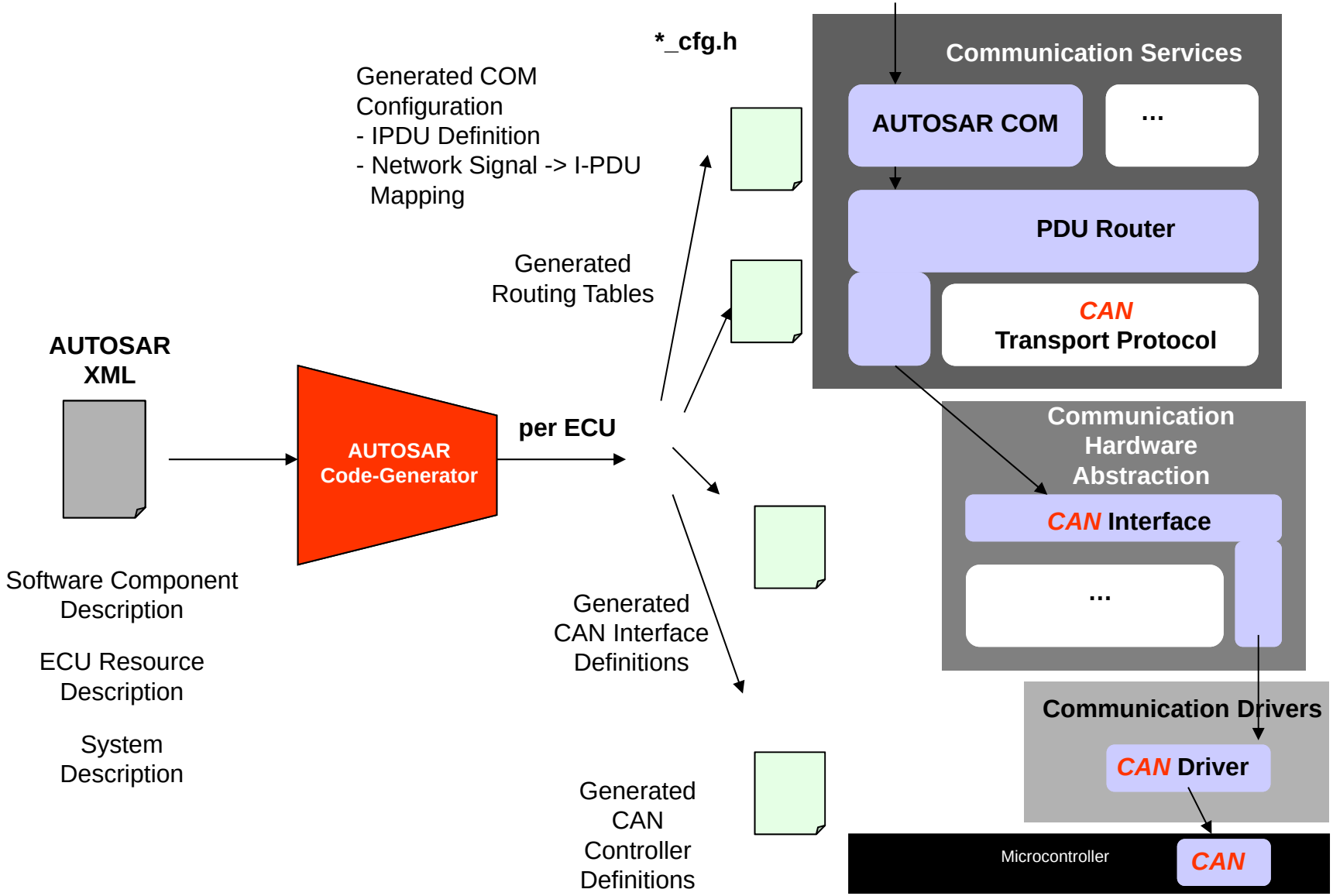
Step 9: Configuring the ECUs - Basic Software

Step 9: Configuring the Basic Software

- This step is necessary to setup a **real** ECU

- We have to configure the **Basic Software**
  - COM
  - PDU Router
  - CAN Interface
  - CAN Driver
  - … !




*_cfg.h
Generated COM Configuration
- IPDU Definition
- Network Signal -> I-PDU Mapping
Generated Routing Tables
AUTOSAR XML
AUTOSAR Code-Generator
per ECU
Software Component Description
ECU Resource Description
System Description
Generated CAN Interface Definitions
Generated CAN Controller Definitions
Communication Services
AUTOSAR COM
...
PDU Router
CAN Transport Protocol
Communication Hardware Abstraction
CAN Interface
...
Communication Drivers
CAN Driver
Microcontroller
CAN

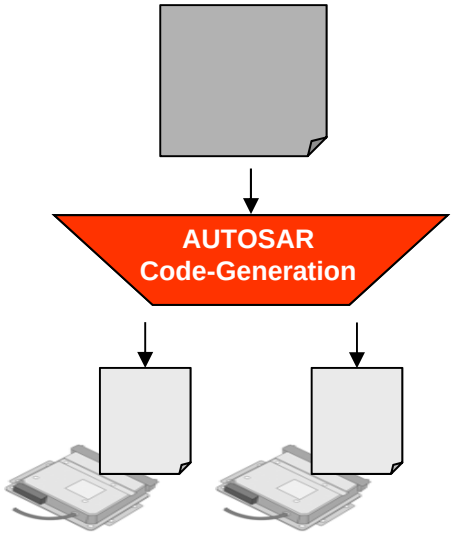
SYSTECS

Part 2: AUTOSAR - Example

Step 10: Code-Generation for ECUs

Step 10: Code-Generation for each ECU

- The AUTOSAR System for a Vehicle is stored as a **XML Description**
- The Configuration of the Basic Software for each ECU is stored as well as a **XML Description**
- **Code-Generators** generate now the complete Code for each ECU
- **Compiler & Linker** are generating the final Binaries !



SYSTECS

Part 4: Final Discussion

Thank you for listening

SYSTECS Informationssysteme GmbH
Bahnhofstraße 11
D 70771 Leinfelden-Echterdingen

Phone +49- 711- 722 312 - 10
Fax +49- 711- 722 312 - 8

www.systecs.com

info@systecs.com