



# Trends in Automotive Software Engineering

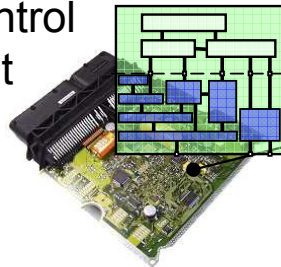
University of Braunschweig – February 2007, 5<sup>th</sup>

Dr. Thomas Zurawka, SYSTECS Informationssysteme GmbH

## 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

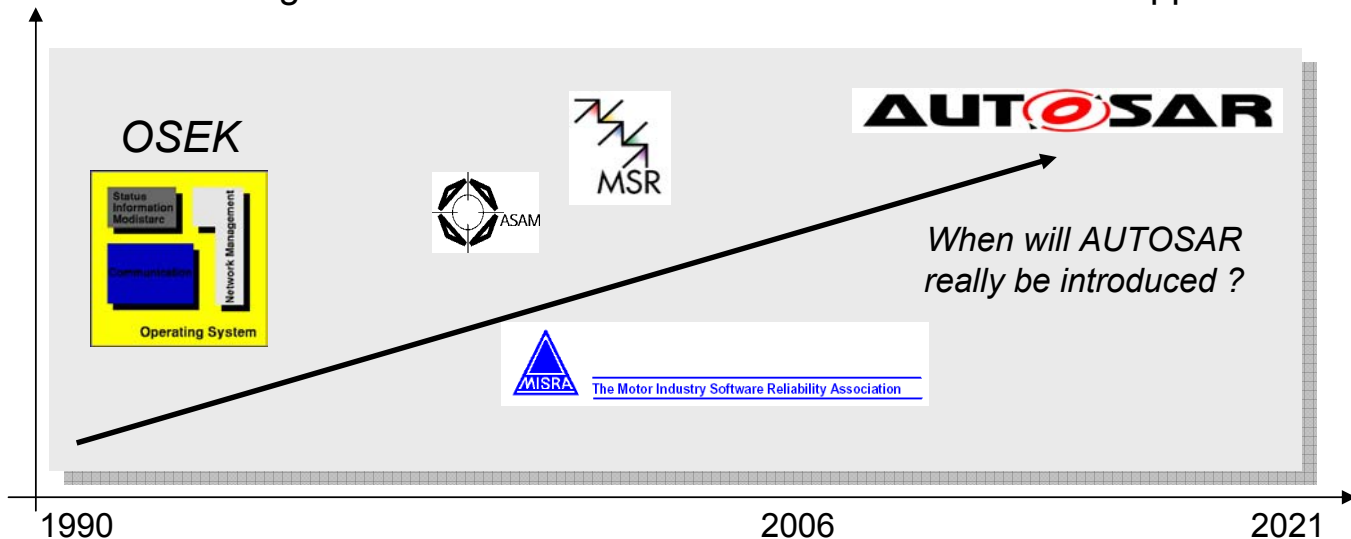
5kByte – 2MByte Bytes Flash

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

**→** *1MByte Flash costs 10 Euro !*

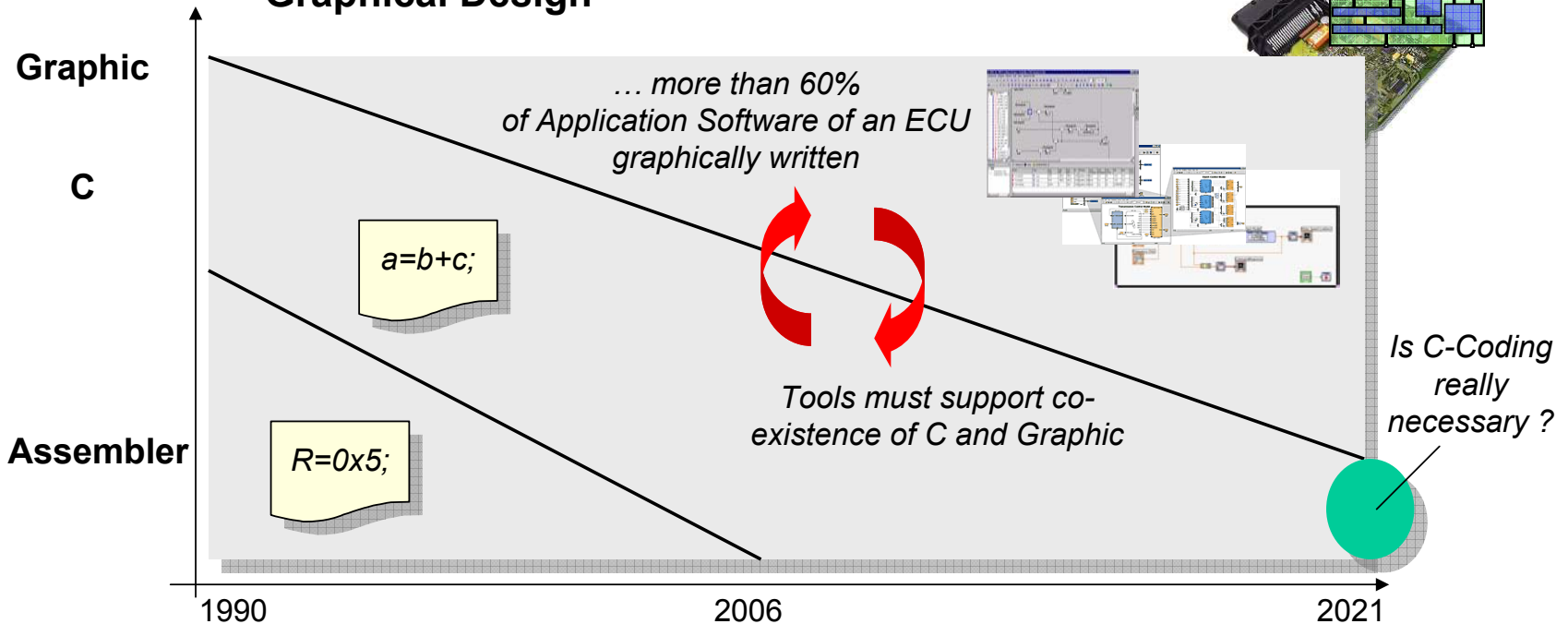
## Trends in the Automotive Industry

- 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



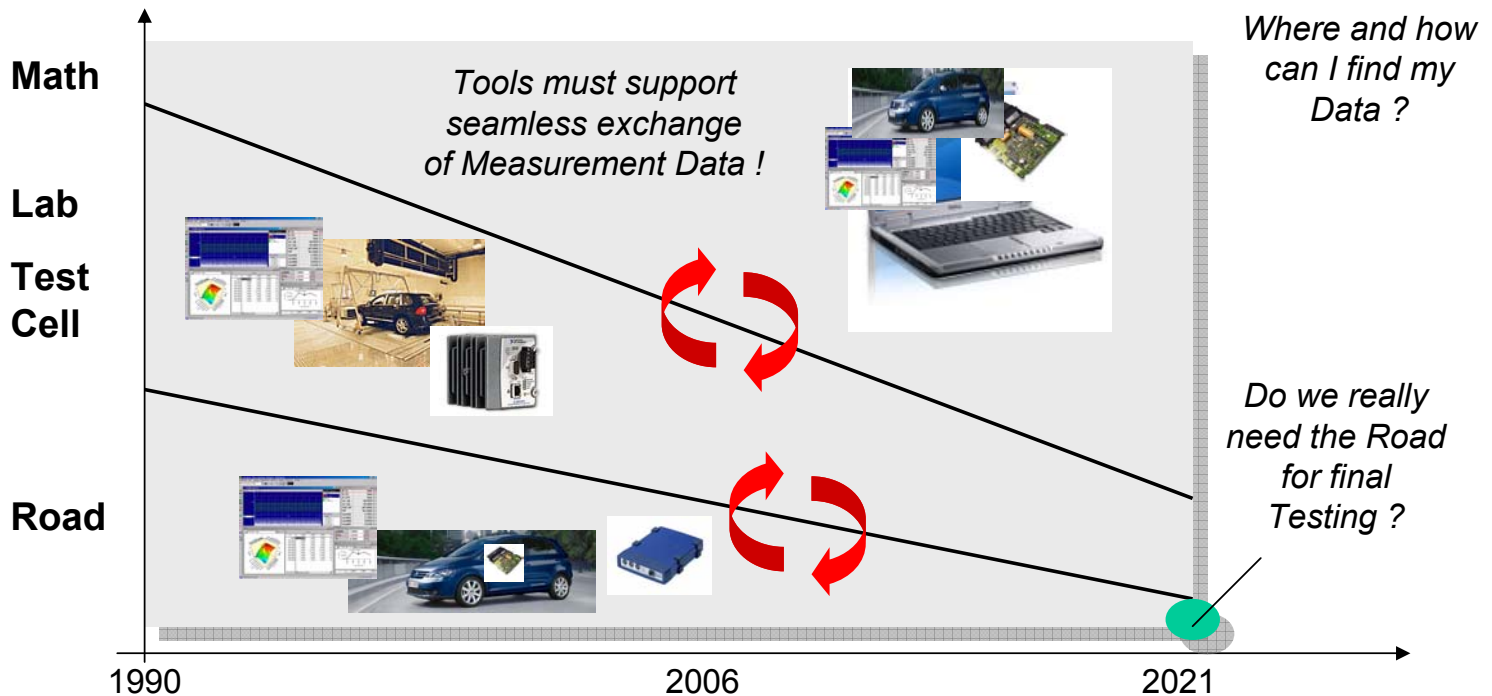
# Trends in the Automotive Industry

- 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

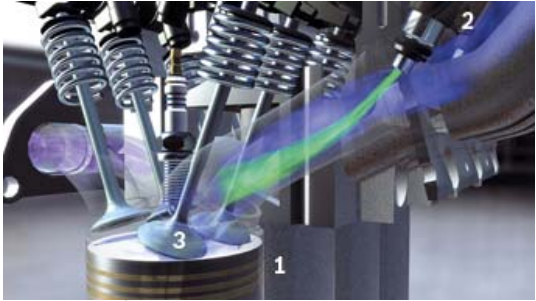
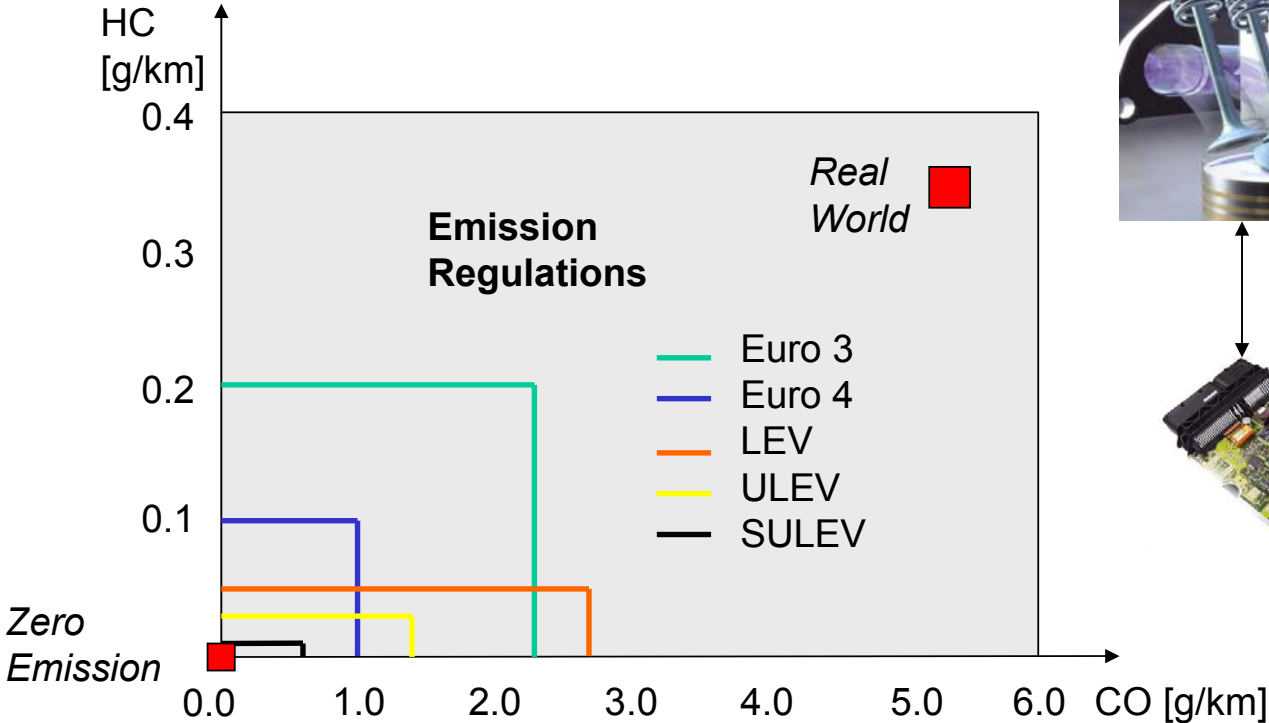
- **Worldwide** distributed Development Teams
  - e.g. **1000 Employees & 100 Projects** per year for **one** ECU generation



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

# Trends in the Automotive Industry

- Increasing complexity
  - An Example: Legal Requirements

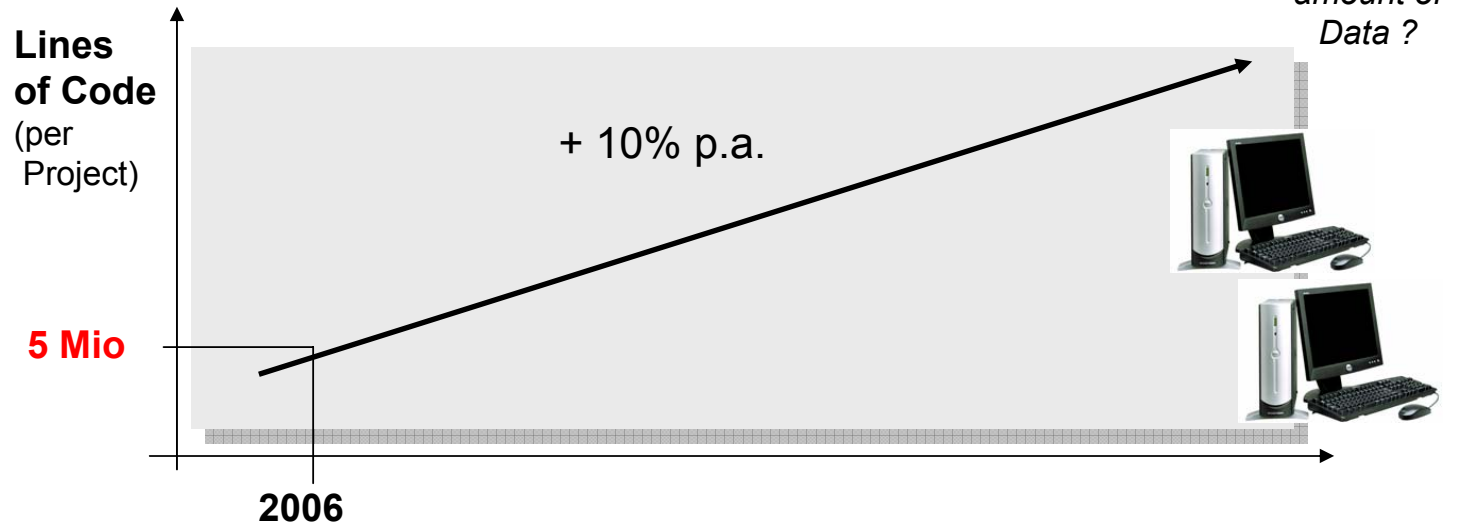


Much more Software needed ...  
and ... other System Concepts, e.g. Hybrids, Fuel Cell, ...

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



## Trends in the Automotive Industry

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

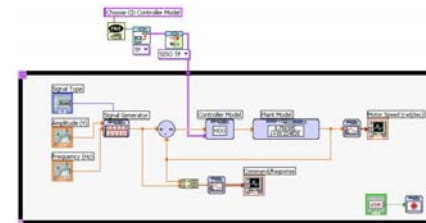
- **CMMI** or SPICE complaint Processes



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

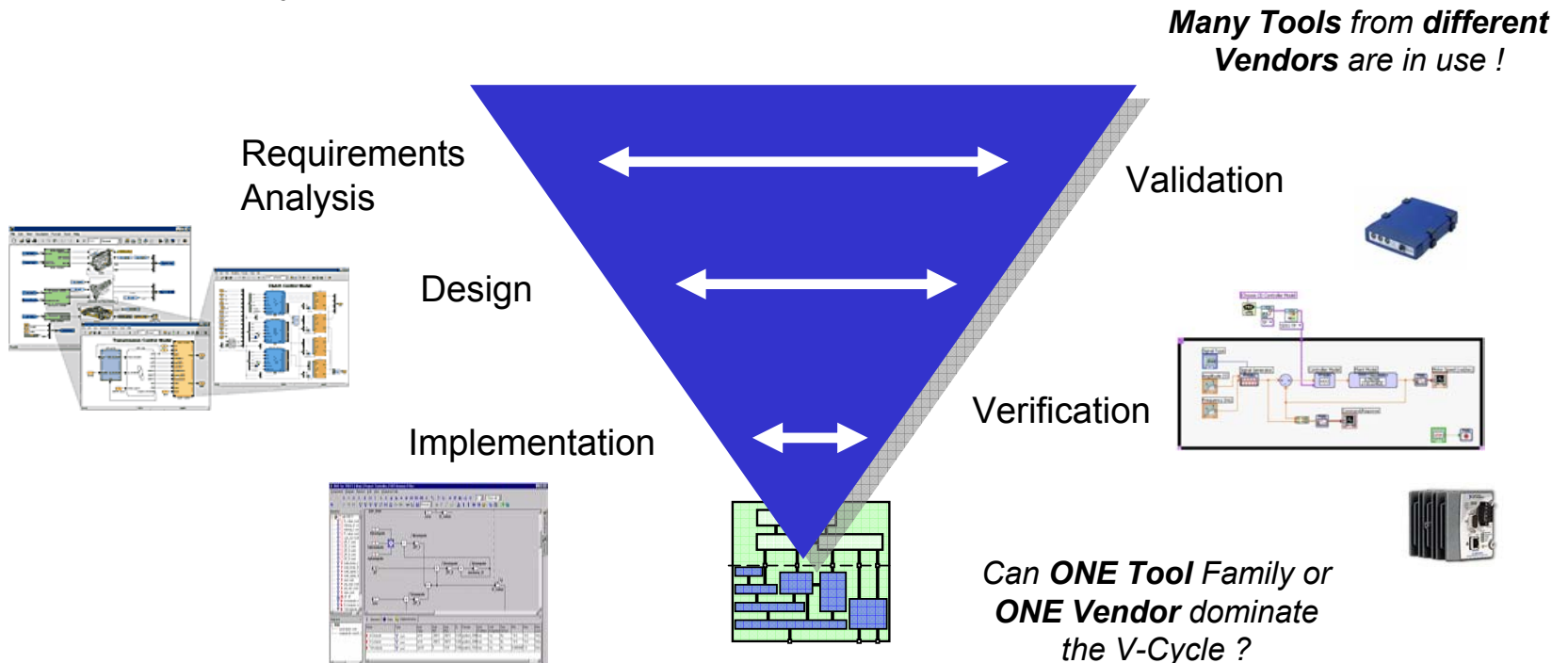
- Tricky development Methods and ...

... 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



## Trends in the Automotive Industry

- **What** are additional **Initiatives** in the Automotive Industry in order to master these challenges ?

## Initiatives in the Automotive Industry

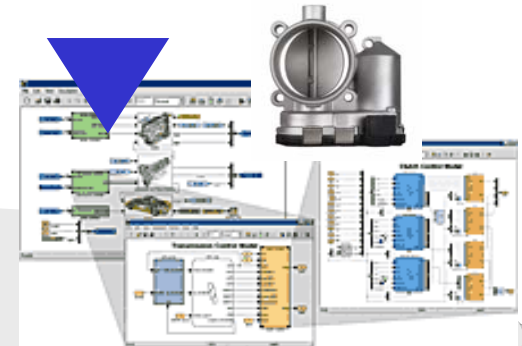
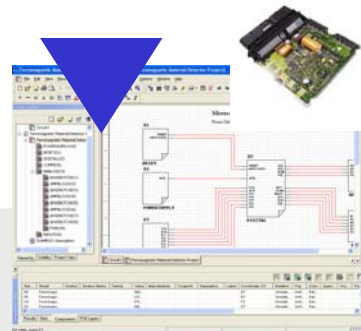
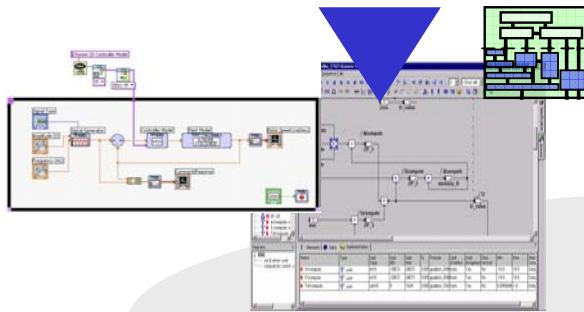
- **The Virtual Development Initiative**

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

ECU Software

ECU Hardware

Throttle, ...

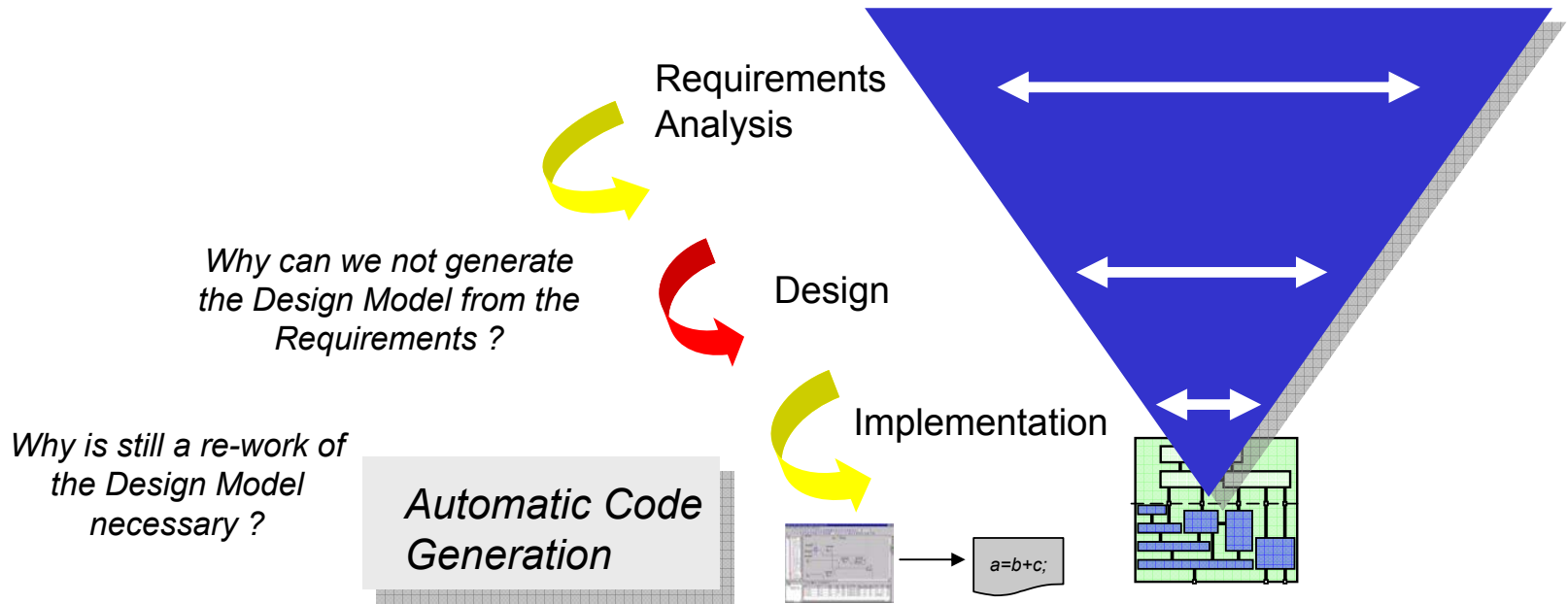


**Integrated Development Environment**  
- ECU Simulation Backplane -



# 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 ?*

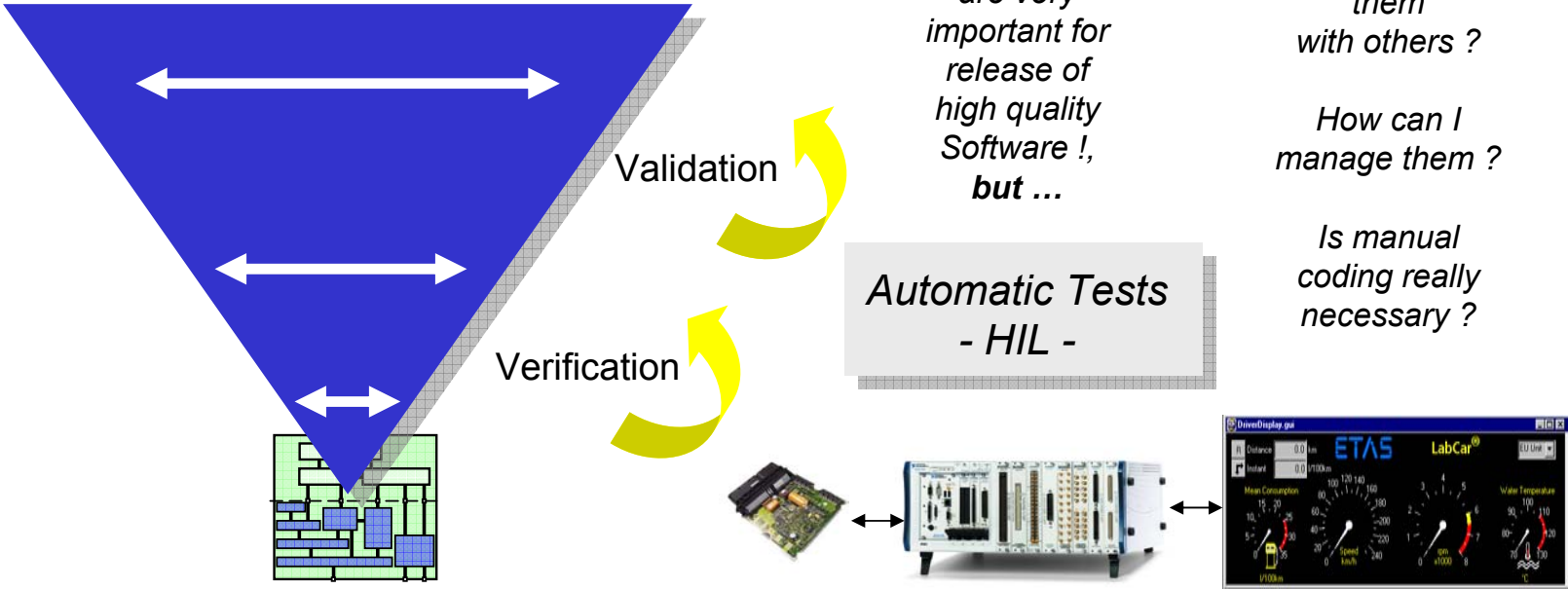
*Can I share them with others ?*

*How can I manage them ?*

*Is manual coding really necessary ?*

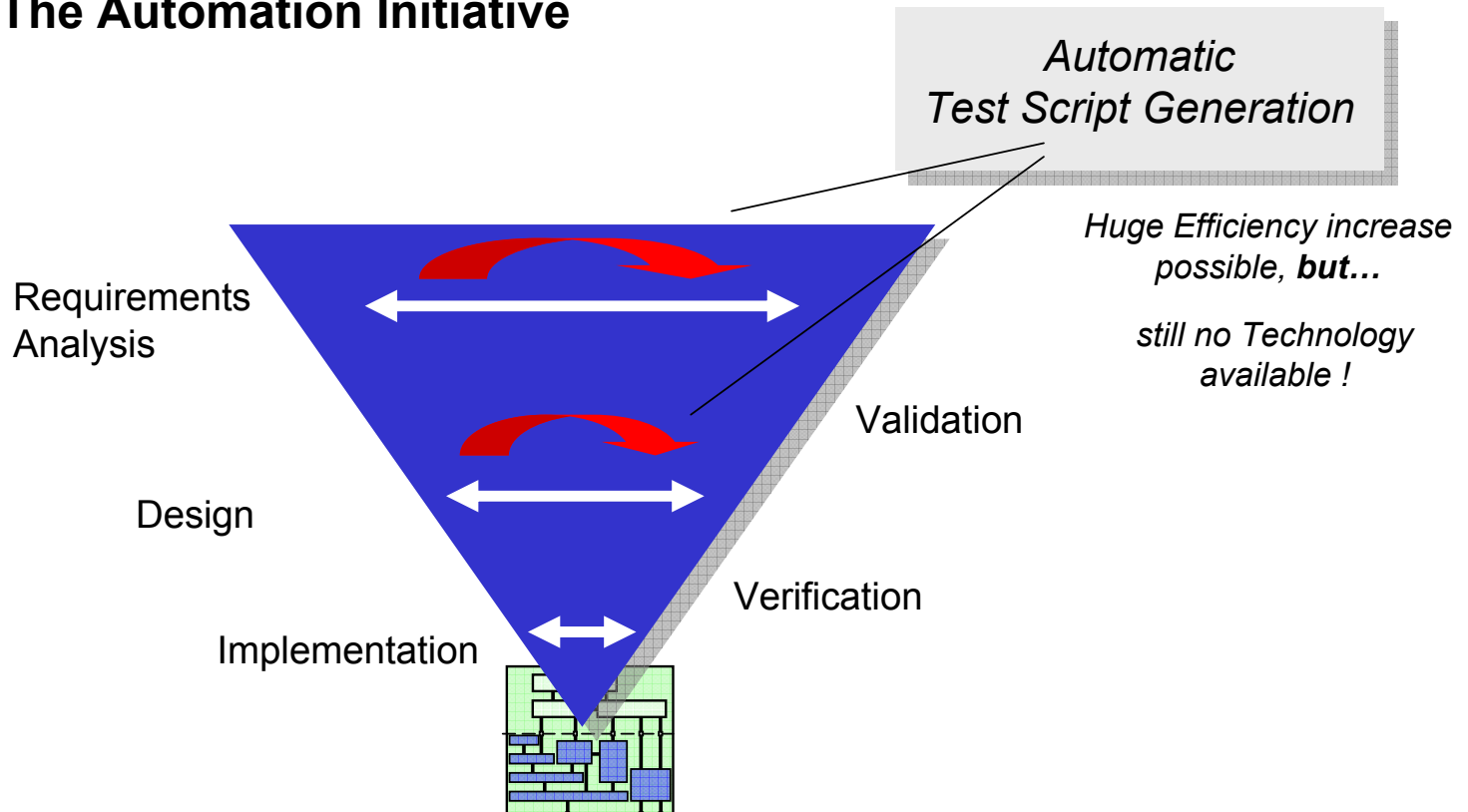
*HIL Tests are very important for release of high quality Software !, but ...*

**Automatic Tests - HIL -**



## Initiatives in the Automotive Industry

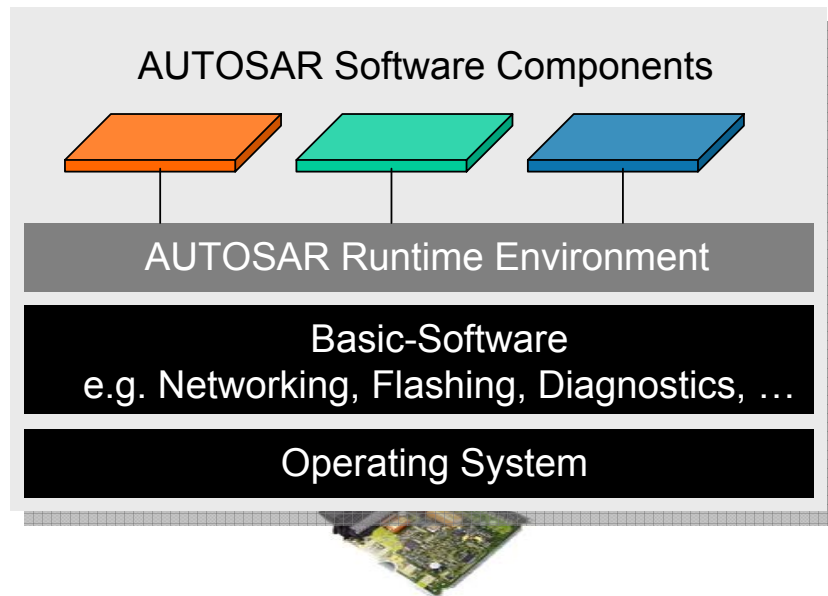
- **The Automation Initiative**



## Initiatives in the Automotive Industry

- **The Standardization Initiative**

**AUTOSAR**  
ECU Software Architecture

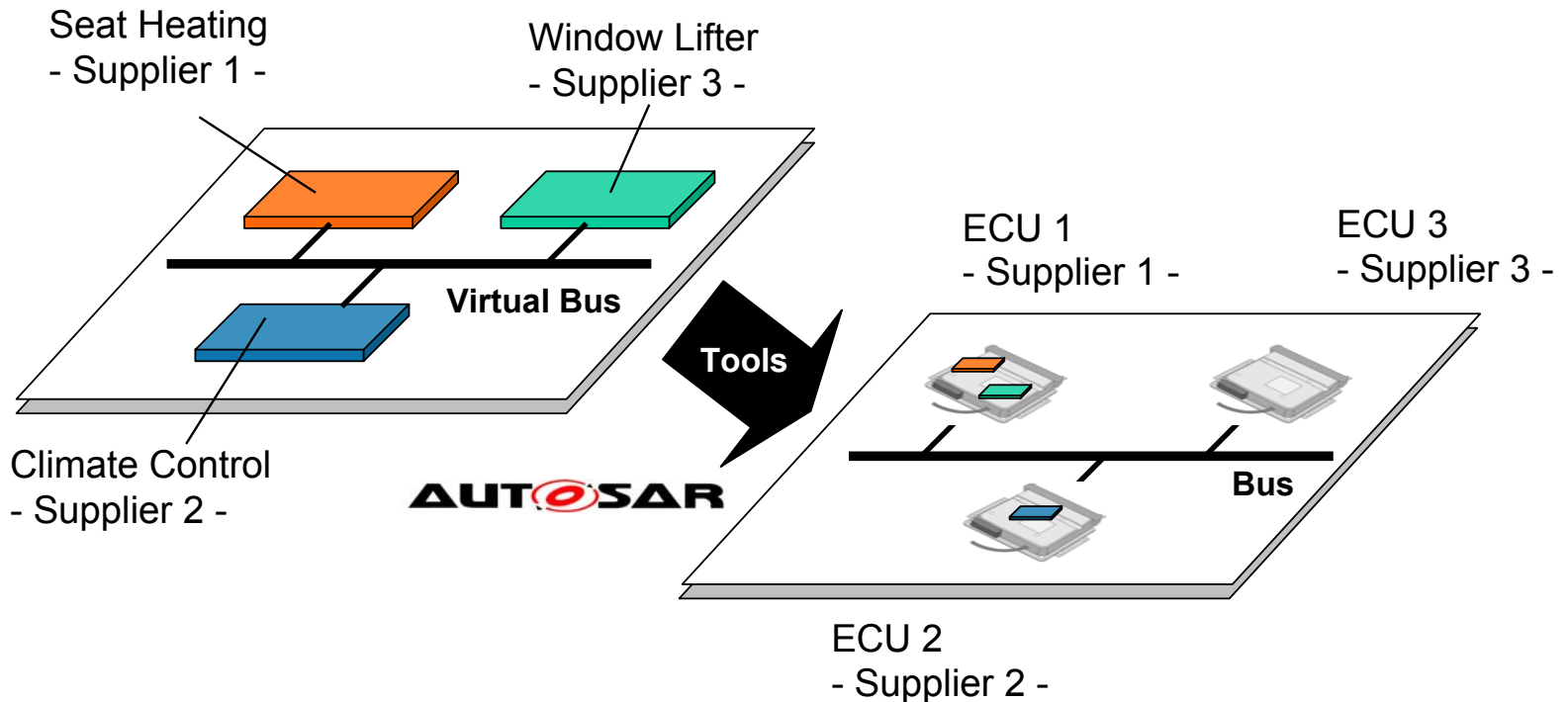


**AUTOSAR**

- *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



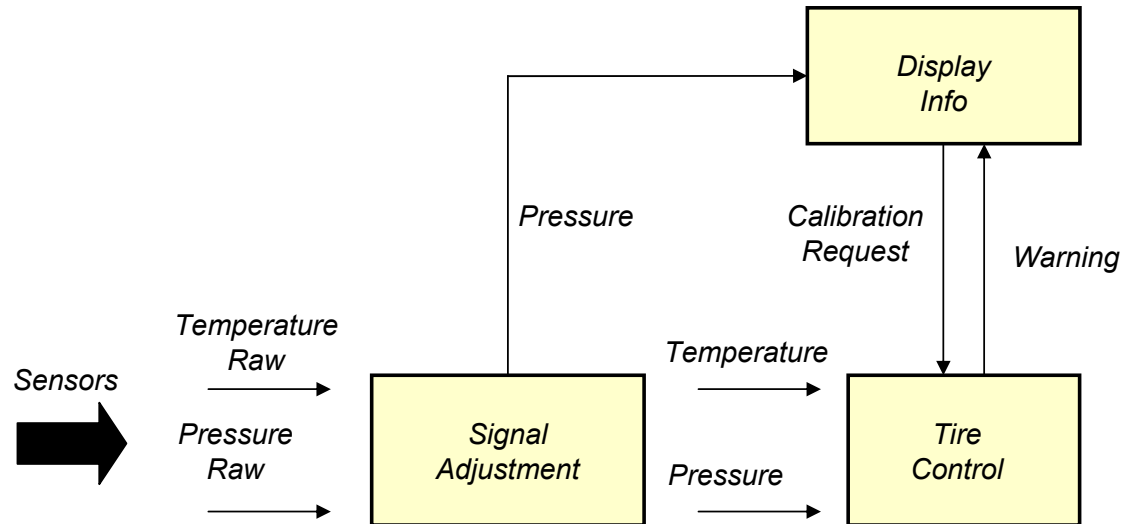


## Part 2: AUTOSAR - Example

Introduction

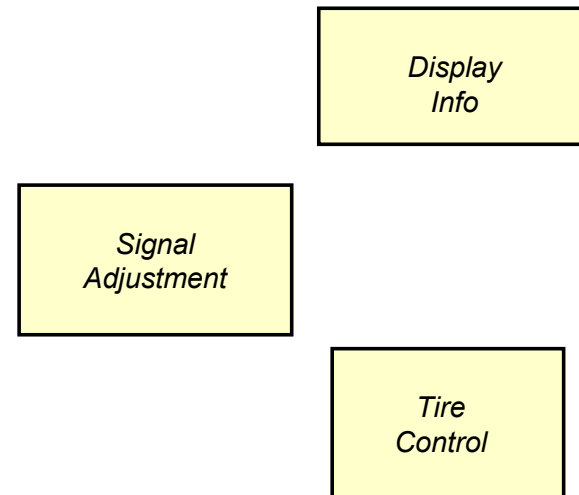
## An Example

- Measuring and Displaying the Tire Pressure
  - Overview



## Step 1: The Software Components itself

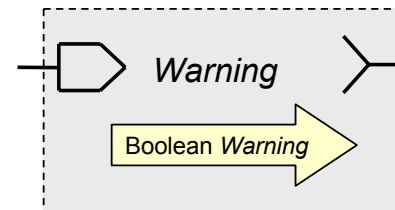
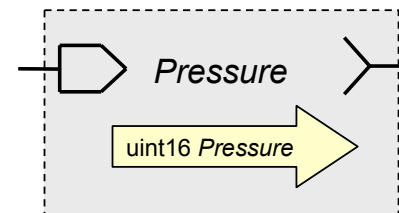
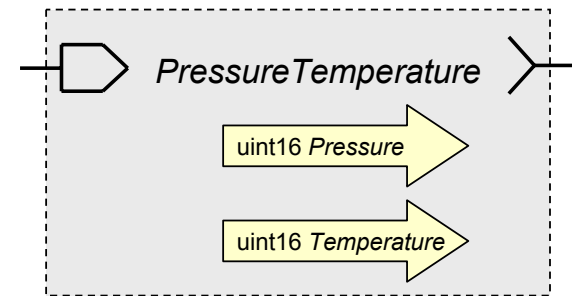
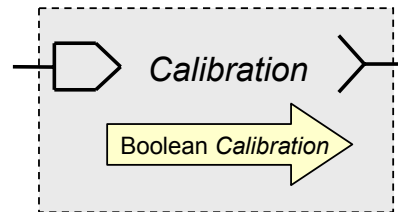
- We define 3 **Atomic Software Components**
  - *“Signal Adjustment”*
  - *“Tire Control”*
  - *“Display Info”*



## Step 1: The Ports of the Software Components

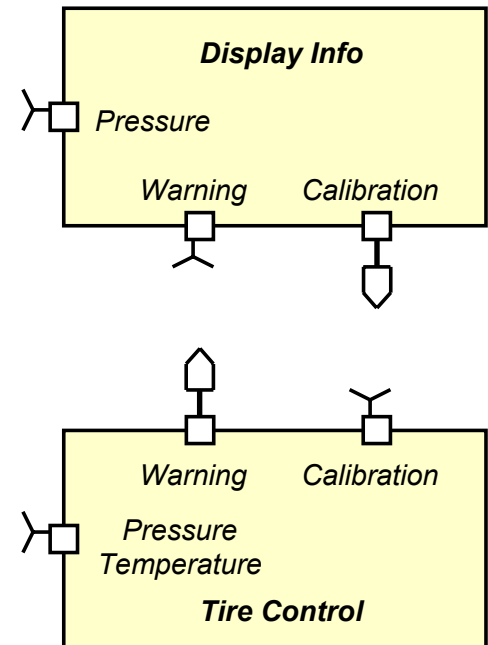
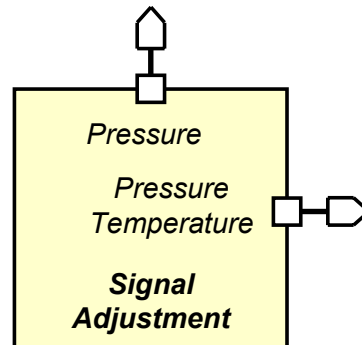
- We define the **Sender-Receiver Interfaces**

- PressureTemperature*
- Pressure*
- Warning*
- Calibration*



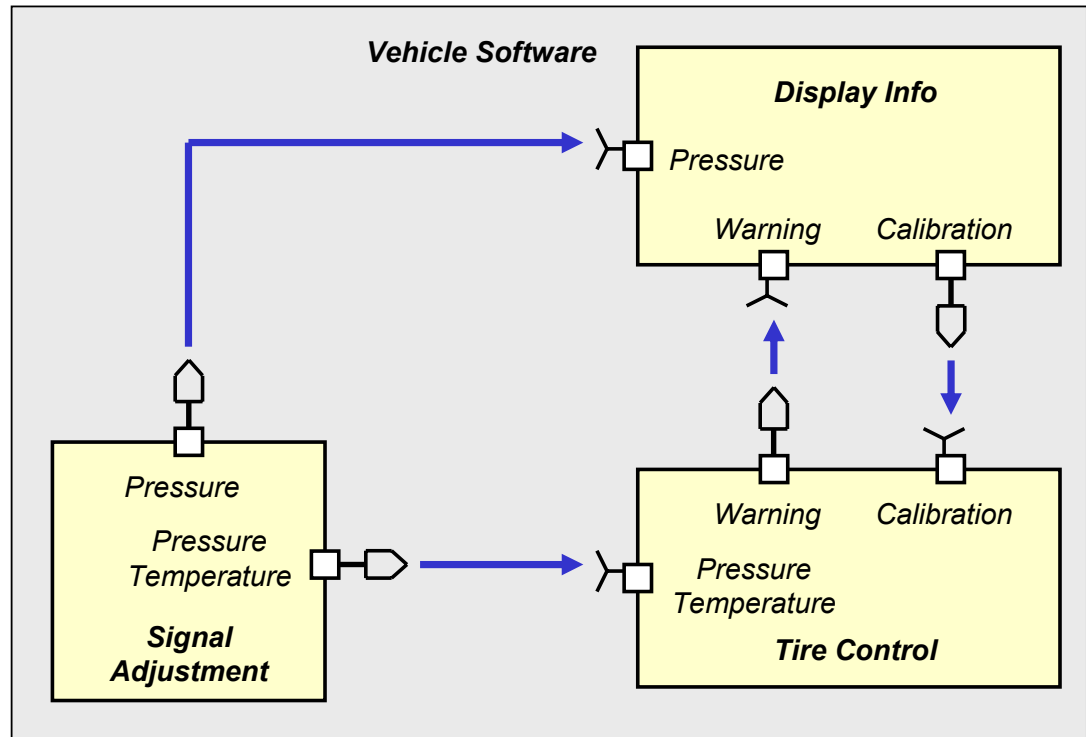
## Step 1: The Ports of the Software Components

- 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

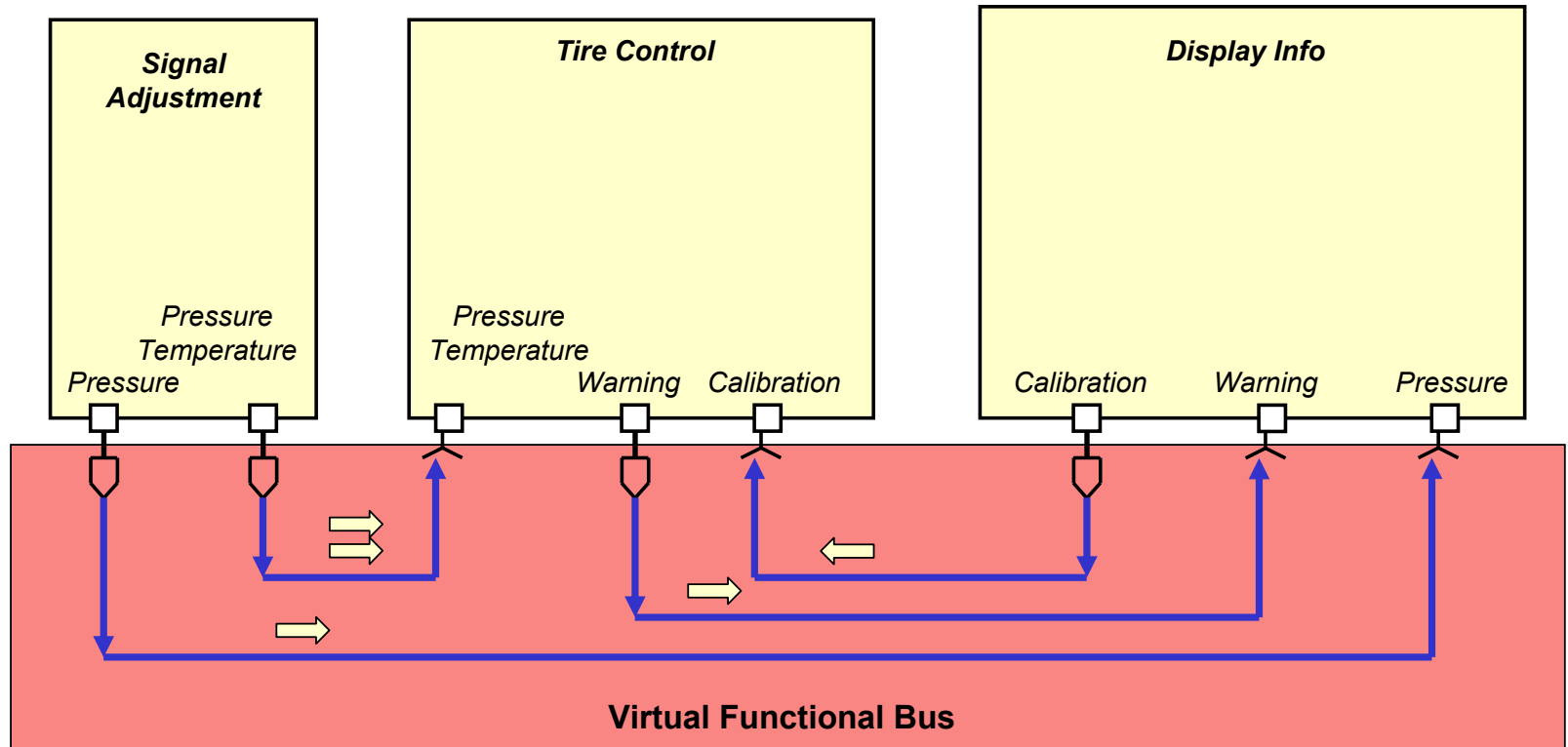


## Step 2: The Software Component Network

- We take the Composition “*Vehicle Software*” and **connect** the Software Components

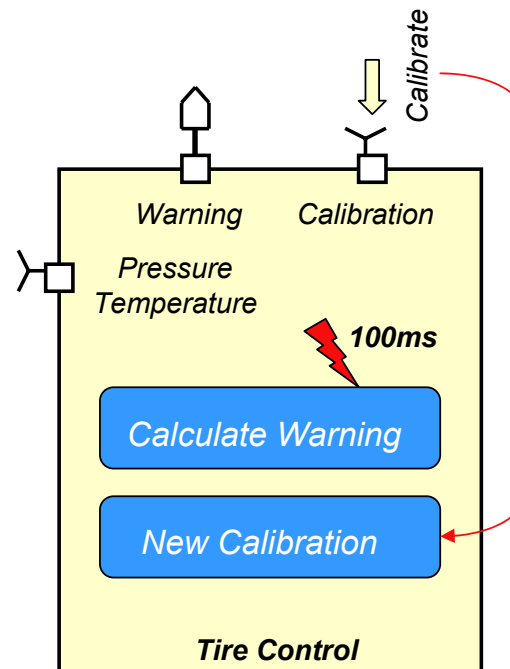


## Step 2: The Software Component Network

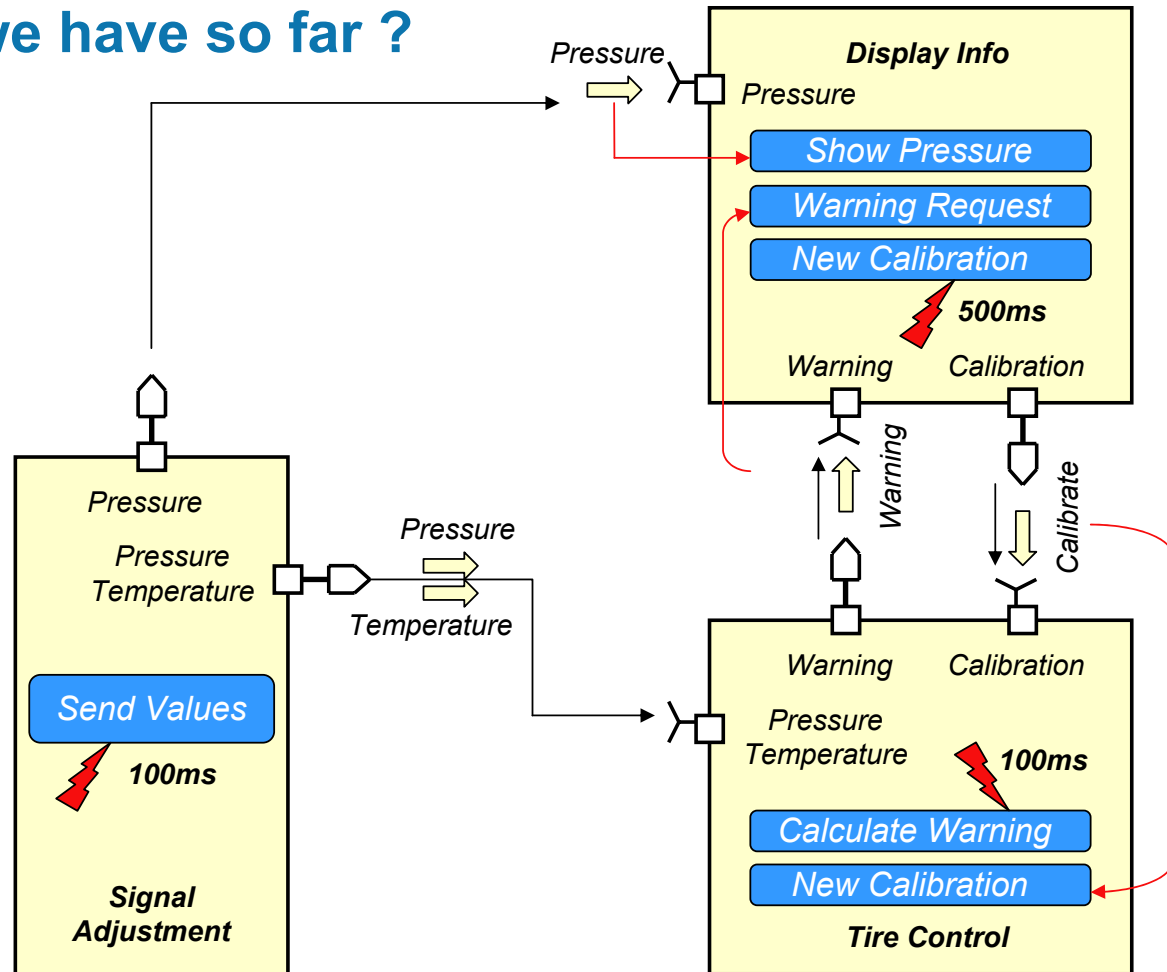


## Step 3: The Code for the Software Components

- 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

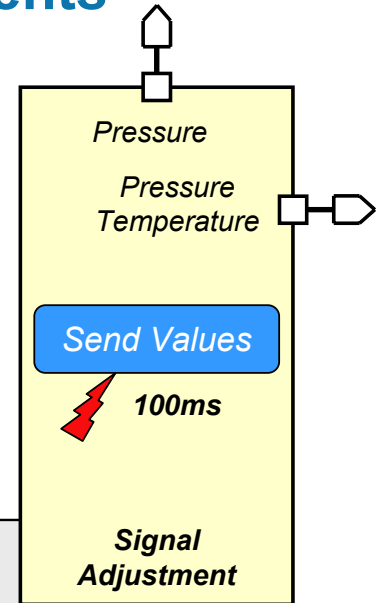


# What do we have so far ?



## 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 RTE_Instance self, IN uint16 Pressure);
RTE_WRITE_PressureTemperature_Pressure (IN RTE_Instance self, IN unit16 Pressure);
RTE_WRITE_PressureTemperature_Temperature (IN RTE_Instance self, IN unit16 Temperature);
```

## Step 3: The Code for the Software Components

- Here we are:

### Signal\_Adjustment.c

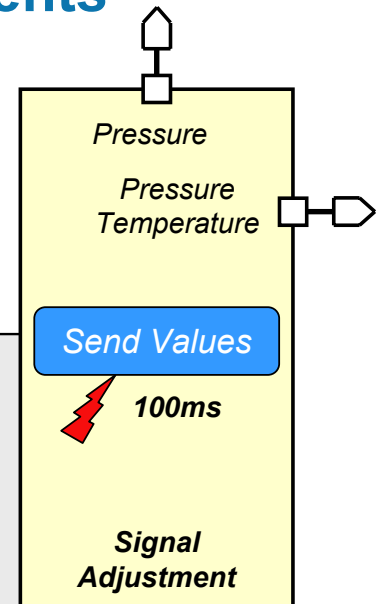
```

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

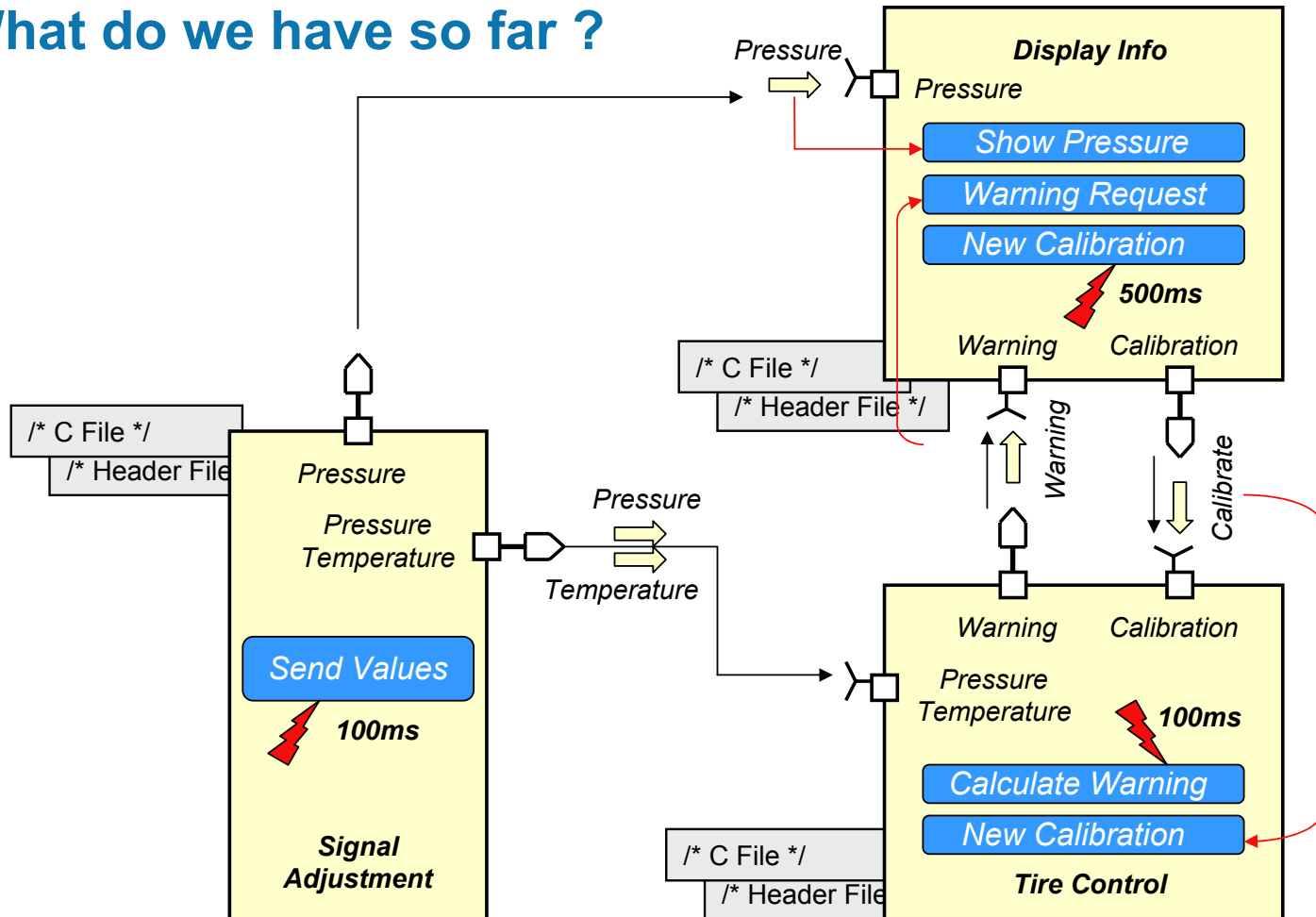
/* Runnable Entities */
Void Send_Values (RTE_Instance self);
{
uint16 sensor_temperature;
uint16 sensor_pressure;

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 */

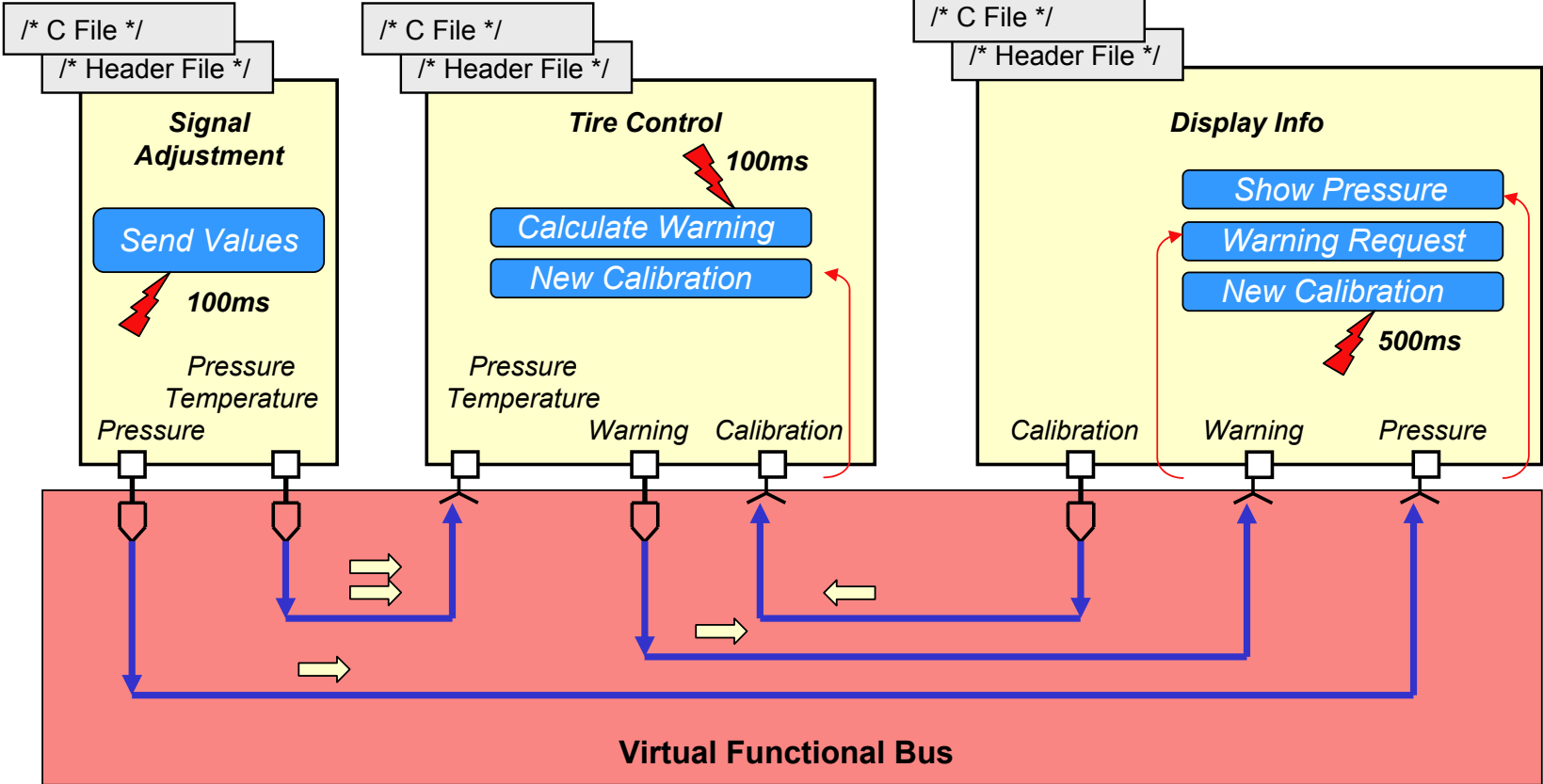
RTE_WRITE_PressureTemperature_Temperature (self, sensor_temperature);
RTE_WRITE_PressureTemperature_Pressure   (self, sensor_pressure);
RTE_WRITE_Pressure_Pressure               (self, sensor_pressure);
}
    
```



# What do we have so far ?

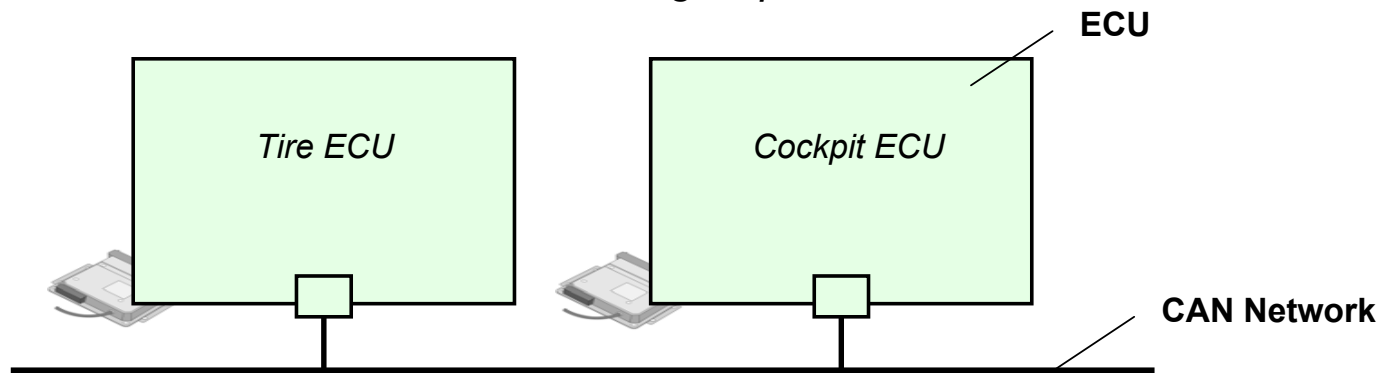


# Step 3: Summary



## Step 4: Definition of a Network of ECUs

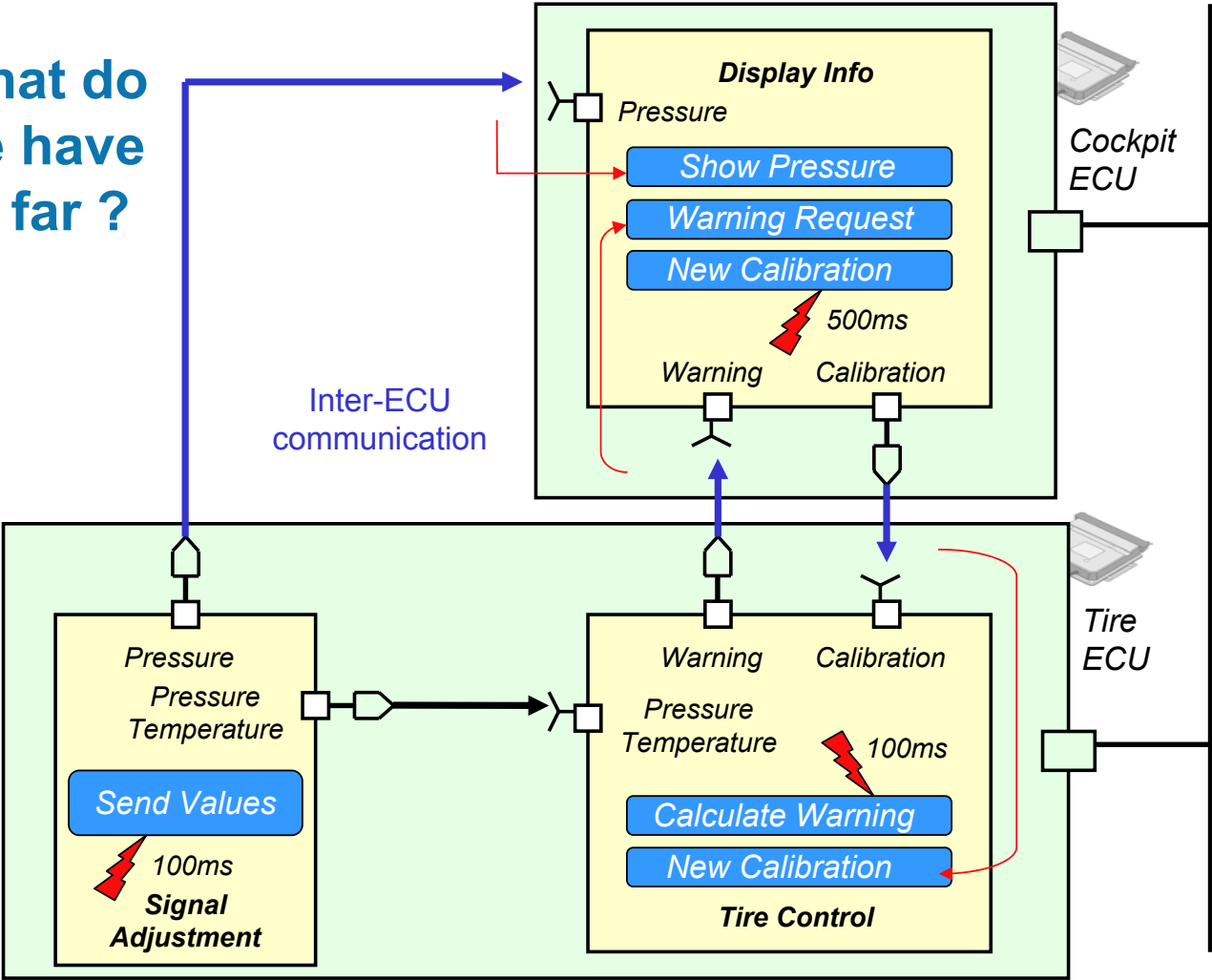
- Additionally we have to define a **Network** and we have to assign the 2 ECUs to it
  - We use a CAN Network, called *High Speed CAN*



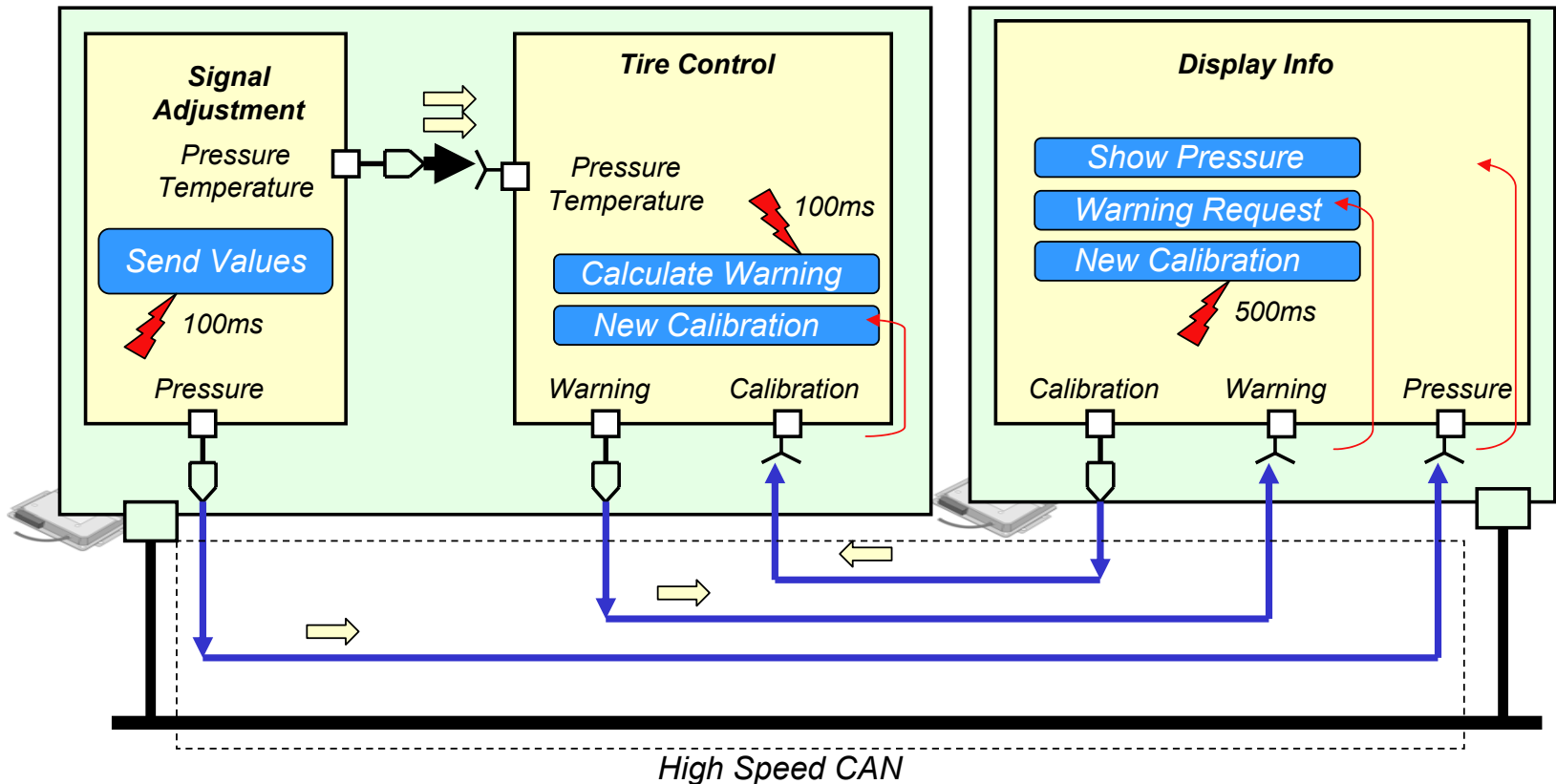
*High Speed CAN*

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

What do we have so far ?

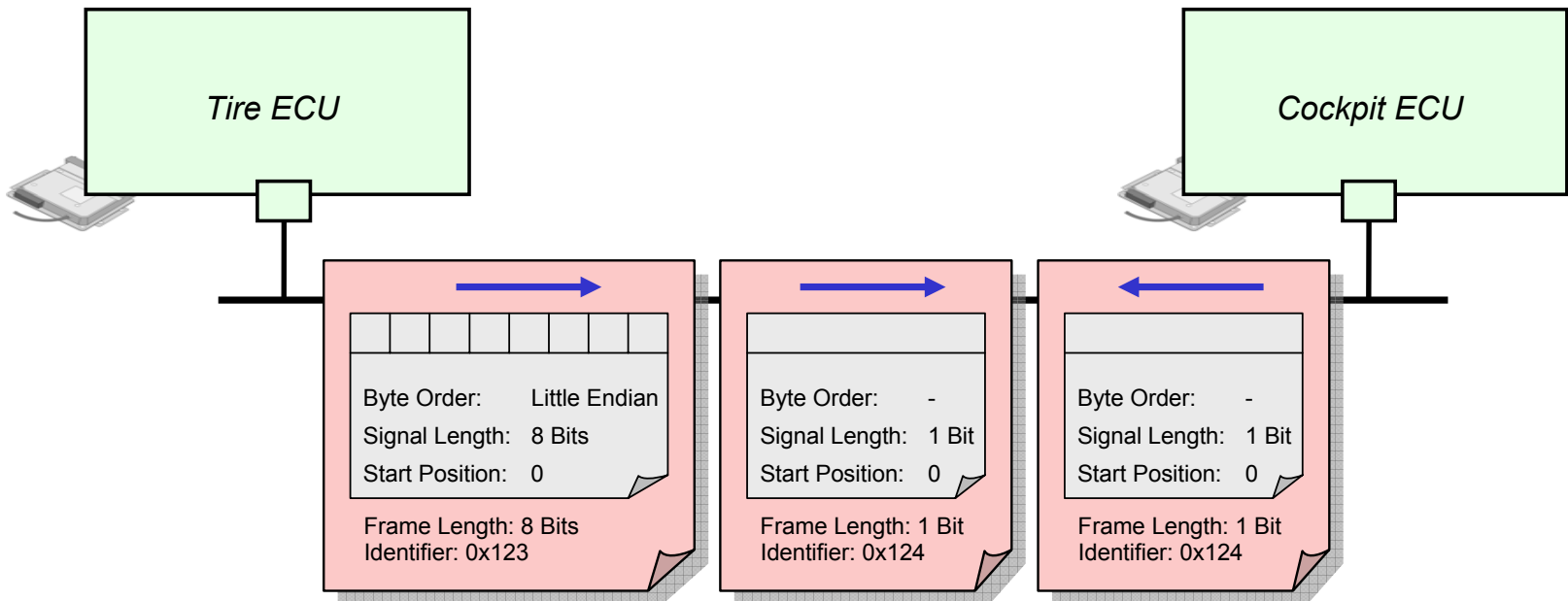


## Step 5: Summary (Just a redraw of the previous figure ...)

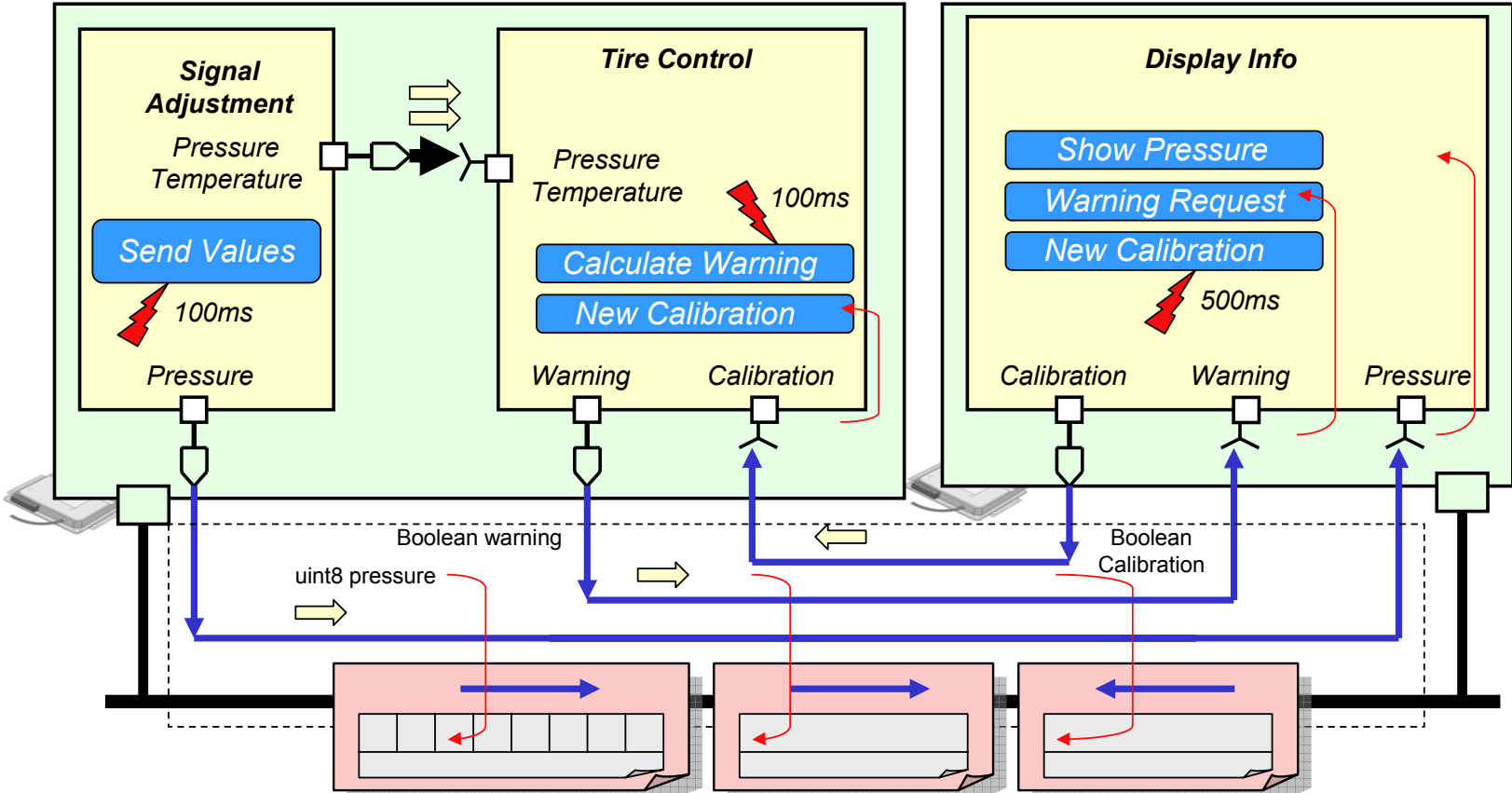


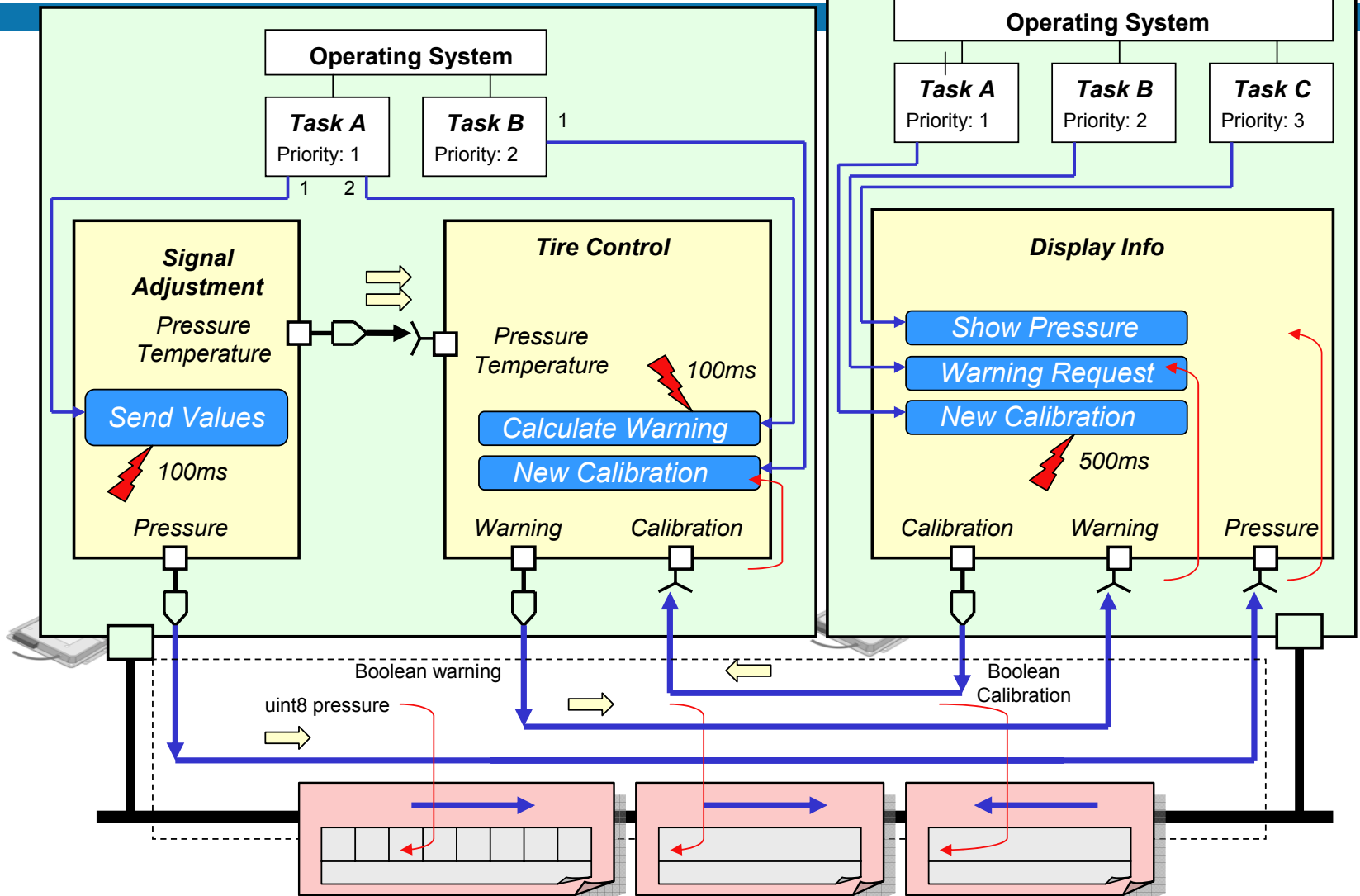
## Step 6: Frames for Inter-ECU communication

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



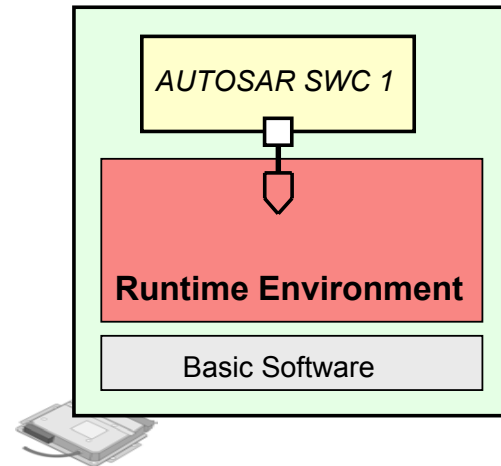
# That's the Result of this Step 7





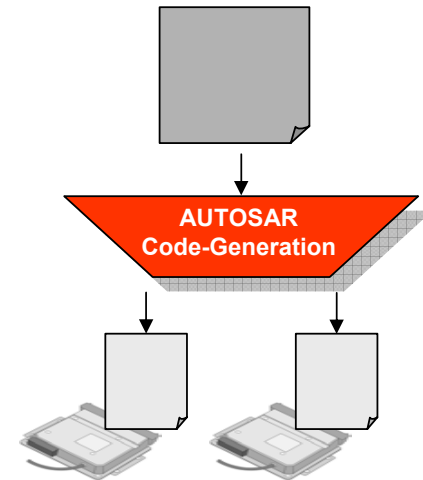
## Step 9: Configuring the Basic Software

- This step is necessary to setup a **real** ECU
- We have to configure the Basic Software
  - CAN Drivers, ...
- This is typically done **once** a time !
- And a huge effort
  - But not of interest for the understanding of the example



## 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 !



## Step 10: Summary

- You can now **download the ECU Software** for the complete Vehicle
- We are done !

## What can you expect from SYSTECS ?

- **SYSTECS** is providing
  - Software **Engineering Services**
  - Innovative **Frameworks** for CAE-Tools
    - Architecture enforcement
    - Re-usable automotive Software Components
  
- **Consulting, Seminars & Workshops** for use of best-in-class *Processes, Methods, CAE-Tools & Software Architectures*
  - SYSTECS is Associated Member of
  
- **INCODIO®** - Integration & Simulation of C-Code



## Thank you for your Attention

### **SYSTEMCS Informationssysteme GmbH**

Kernerstr. 4

D 70771 Leinfelden-Echterdingen

Phone +49- 711- 16082 - 10

Fax +49- 711- 16082 - 8

[www.systemcs.com](http://www.systemcs.com)

[sales@systemcs.com](mailto:sales@systemcs.com)