VCC COMPLETE HIL RIGS MEETING OUR NEXT GENERATION CORE BASED SERVICE ORIENTED ARCHITECTURE
Agenda

- Short background
- Complete HIL
- Electrical Architectures
- Complete HIL - new SPA2 challenges
Engine example – Real vehicle setup

~ 100 ECUs
Engine example – Component HIL setup

- Rest-bus simulation
- Plant Model
- Sensors & Actuators stimulation
HIL SOLUTION EXAMPLE: Active Safety
Complete HIL rig setup

3-4 full-size dSPACE Scalexio / PHS simulators

~ 100 ECUs

Rest-bus simulation

Plant Model

Sensors & Actuators stimulation
Example of a *Complete HIL rig* – Volvo XC60

- **Powertrain Domain**
- **Body Domain**
- **HMI & Infotainment**
- **HOST PC**
- **Chassis Domain**
Complete HIL and test focus areas

On vehicle complete system integration level:
- *Stimulate* all ECUs – “I’m in a real vehicle”

Recommended test focus areas:
- Distributed functionality
- Exploratory testing
- Dynamical test cases
- Non-functional system tests

*Manual and automated* test possibilities
Project milestone examples in Complete HIL

- First Engine start
- First Collision Avoidance
- First Autonomous Vehicle Drive (Uber collaboration)
Evolution of the electrical architectures at Volvo

- S80 1998 (20 ECUs)
- XC90 2002 (38 ECUs)
- V40 2003 (49 ECUs)
- S80 2006 (68 ECUs)
- V60 PHEV 2012 (78 ECUs)
- XC90 2015 (108 ECUs)

Over past 20 years: from 20 -> 100+ ECUs
SPA Electrical Architecture

- Domain oriented
- FlexRay Backbone
- ~10 CAN networks
- ~20 LIN networks
- Signal- and AUTOSAR oriented
- 100+ ECUs
Challenge: No longer easy to replace a complete ECU with a model
From *Signal* oriented paradigm...

**Signal A to E:**
- CAN Frame ECUA_1
- Update rate: 20 ms
...

ECU A

Signal A
Signal B
Signal C
Signal D
Signal E

ECU C
ECU E

ECU A

Signal X
Signal Y
Signal Z

ECU E
From *Signal* oriented to *Service* oriented paradigm (continued)

**Signal oriented paradigm**

- CAN frame
  - wheelSpeeds
  - Device: Brake Module
  - Device: Brake Module
  - Device: Door Module

- Ethernet frame
  - wheelSpeeds
  - Local SDB ver. 1.0
  - Local SDB ver. 2.1
  - VIU Front

- CAN frame
  - wheelSpeeds
  - Device: Brake Module
  - Device: Brake Module
  - Device: Door Module

**Service oriented paradigm**

- VCU
  - Application: VehicleState
    - getVehicleSpeed(x, freq)
    - getVehicleRollingBackwards(x, onChange)
  - Application: LockManager
    - Service: setLockStatus(x)
    - Service: getLockStatus()

- Device proxy: Brake Module
  - getVehicleSpeeds()
  - getWheelSpeeds()

- Device proxy: Door Module
  - Service: getGpsData()
  - Service: getWheelSpeeds()

- Device proxy: X
  - Service: getWheelSpeeds()

- Device proxy: Y
  - Service: getWheelSpeeds()

**New system integration points!**

**Previous system integration points**

- Capability
  - A: Read Wheel Speeds
  - B: Apply Brake Force
  - C: Lock / Unlock Door
  - D: Open / Close Window

Note: Conceptual sketch!
From *Signal* oriented to *Service* oriented paradigm (continued)

**DoorDevice.h**:

```cpp
Class DoorDevice{
    Public:
        Void setLockStatus(int8 lockCmd);  // lockCmd = 1 => lock the door; lockCmd = 2 => unlock the door
        Bool getLockStatus();  // Returns True if door is locked
};
```

**LockManager.cpp (part of substitute)**:

```cpp
#include "DoorDevice.h"
#include "ScalexioAPI.h"

main() {
    public int8 doorCmd = 1;
    DoorDevice driverDoor* = ServiceFramework.findService("DriverDoorDevice");
    doorCmd = ScalexioService.getValue("driverDoorCmd");  // driverDoorCmd symbolic name to find correct
                                                          // model value from Scalexio processor's memory
    driverDoor.setLockStatus(doorCmd);
}
```

**Note**: Conceptual sketch!
To summarize things...

VCC Complete HIL rigs

Electrical architectures, leading us into SPA2

Signal -> Service oriented paradigm
Thank you for your attention!