Virtual Open Systems

A Safe Graphics Rendering Solution for Consolidated Operating Systems

contact@virtualopensystems.com

(*) The work was supported by the *NGPasS* project (grant agreement No. 761557)

Introduction

- Motivation and background
- VOSYSmonitor
- Safe split-display architecture
- Automotive architecture and application
- Experimental results
- Conclusion

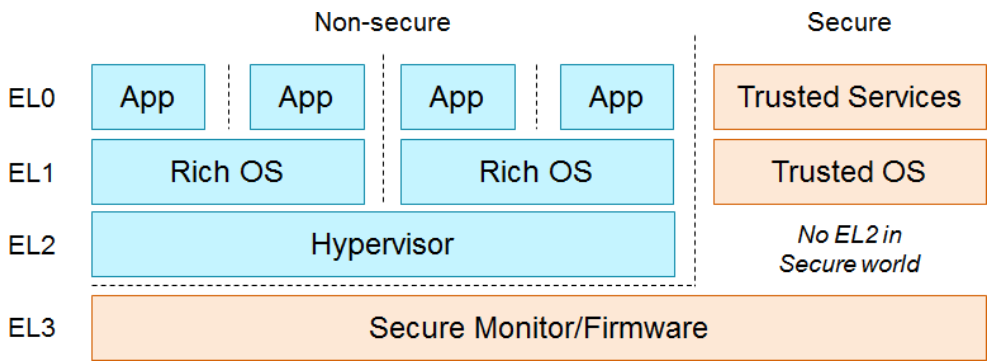
Motivation

- Consolidation of safety critical systems with general purposes rich environments
  - Safety instrument cluster and In-Vehicle-Infotainment (IVI) system
- Sharing of physical *Screens* between OSes with different level of criticality
- To guarantee graphics rendering of safety related applications
- To reboot GPOS without impacting the Safety critical OS

Background – ARM TrustZone

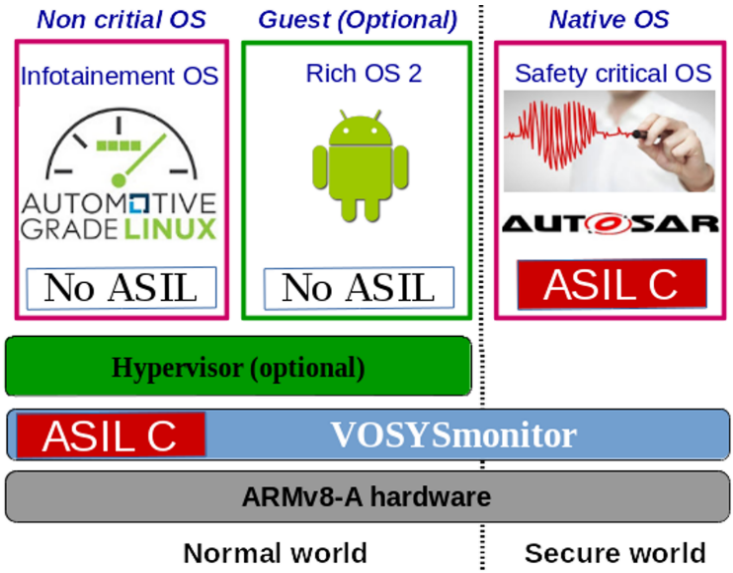
- Separation of “Worlds”
  - Secure – RTOS
  - Normal - GPOS
- 4 exception levels
- Status propagation over the AXI bus (AxPROT)
- Memory and IRQ isolation
- Worlds interaction via the ‘smc’ instruction

ARM TRUSTZONE
System Security



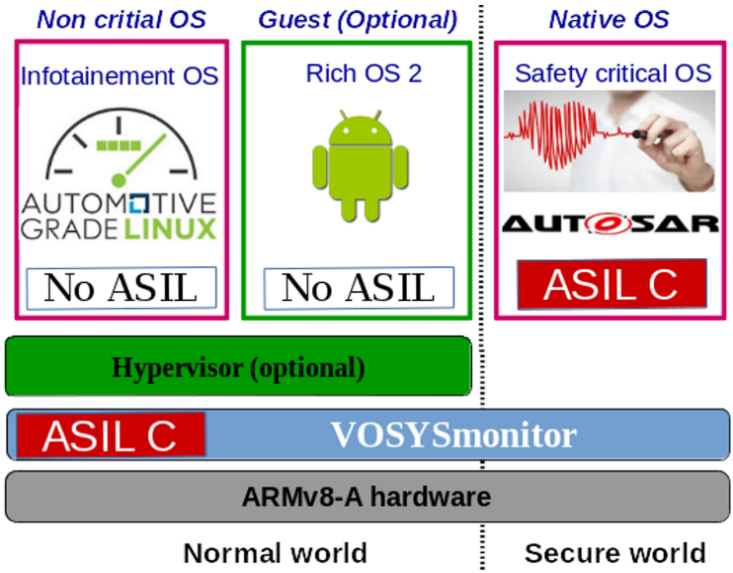
Background – VOSYSmonitor

- System partitioner firmware running in EL3
- ISO 26262 certifiable
- ARMv7 and ARMv8 support
- Enables the co-execution of the two operation systems e.g.,
  - FreeRTOS on Secure world
  - Linux on Non-Secure world



Background – VOSYSmonitor

- Leverages TrustZone capabilities provided by the SoC
- Provides real time guarantees to the RTOS
- Provides system isolation (memory / IRQs and devices)
- Allows GPOS warm reboot
- Power management
- Safety monitoring and management on hardware failure (ISO 26262)



Background – VOSYSmonitor

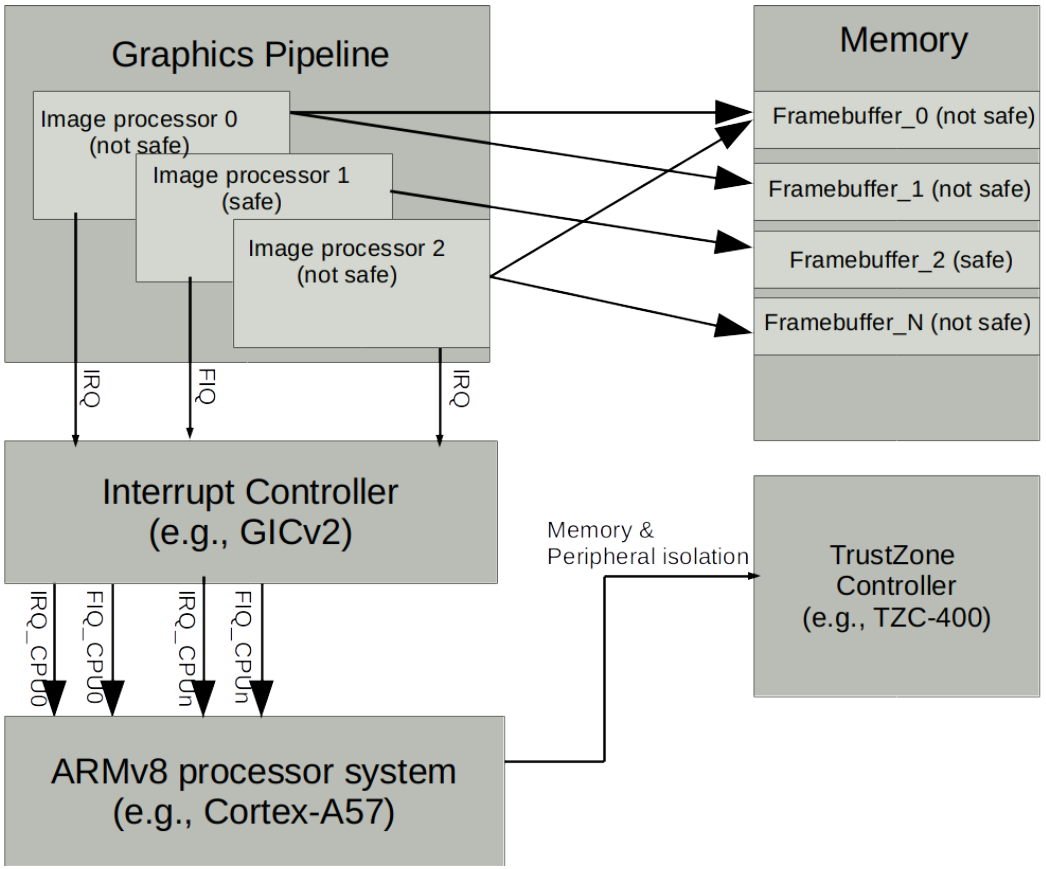
- System partitioner firmware running in EL3
- ISO 26262 certifiable
- ARMv7 and ARMv8 support
- Enables the co-execution of the two operation systems e.g.,
  - FreeRTOS on Secure world
  - Linux on Non-Secure world
- Leverages TrustZone capabilities provided by the SoC
- Provides real time guarangees to the RTOS
- Provides system isolation (memory / IRQs and devices)

Safe split-display architecture

- Consolidation of graphics from multiple OS to a *Screen*
- Guarantees to content rendered from the RTOS
- Isolation of the graphics pipeline to the RTOS

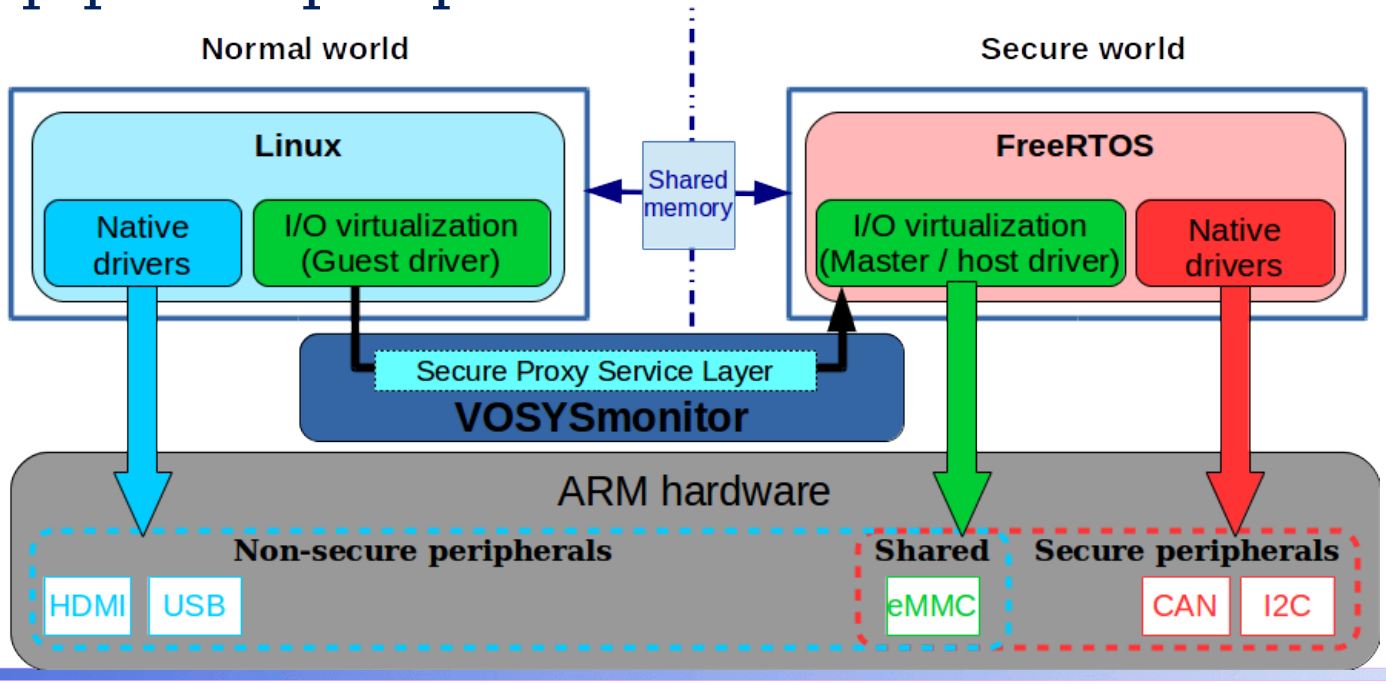
Safe split-display architecture

- The graphics pipeline is controlled by the RTOS
  - Device peripherals
  - Frame-buffers
  - Device interrupts
- GPOS can acquire device after verification from the RTOS
  - RPC style communication via SMCs



Safe split-display - RPC

- The Secure Proxy service
- Allows the GPOS to interact with the RTOS
- Used from Linux drivers to forward read/write requests to graphics pipeline peripherals



Safe split-display – Isolation

- Hardware isolation is provided by the ARM TrustZone
- Booting procedure
  - VOSYSmonitor starts the Secure OS - RTOS
  - The RTOS uses a secure service in order to isolate the peripherals of the graphics pipeline
  - Memory for the Secure frame-buffer (plane) is reserved
  - Devices' IRQs are reserved to target the Secure OS
  - The RTOS initializes the graphics pipeline (drivers)
  - VOSYSmonitor start the Non-Secure OS - Linux

Safe split-display – Isolation

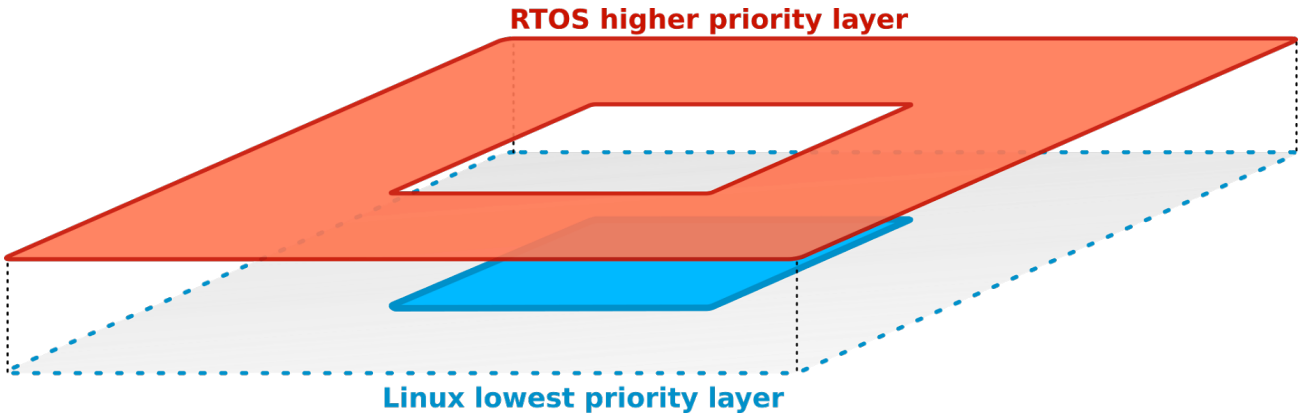
- Isolation of components related to the *Screen*
  - The display on the *Screen* does not depends only on the graphics pipeline
  - Clock generation for the peripherals may also corrupt the *Screen*'s output display, also
  - Peripheral software resets
  - Pin multiplexer controllers
- These drivers are fully controlled by the RTOS
- Non-secure OS access the devices using RPC to the RTOS

Safe split-display – Isolation

- Peripheral's registers that are affect the composition are protected by the RTOS
  - Read/Writes to sensitive registers are ignored
  - e.g., disable the Clock for a Display Unit controller
- Linux drivers co-operate with FreeRTOS drivers in order to setup the Linux part of the graphics pipeline
- Interrupts for Linux are managed by FreeRTOS
  - FreeRTOS notifies Linux for the generated IRQs
  - Linux receives IRQs by using Inter-Processor-Interrupts

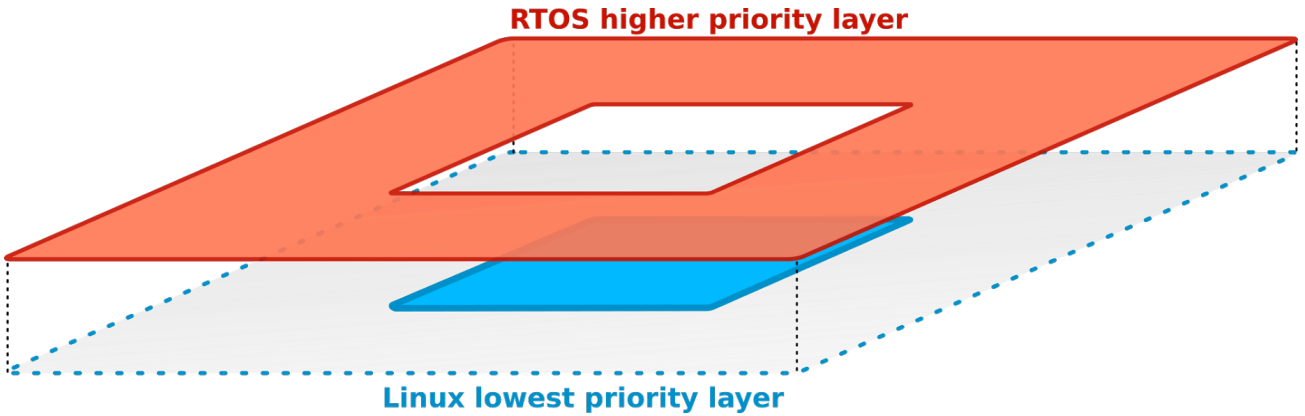
Safe split-display – Image composition

- The composition of multiple planes (frame-buffers) leverages SoC's features of the graphics pipeline
- Modern SoC can support multiple planes
- An image (frame) is composed from multiple sources
- The RTOS manages the position and size of all planes



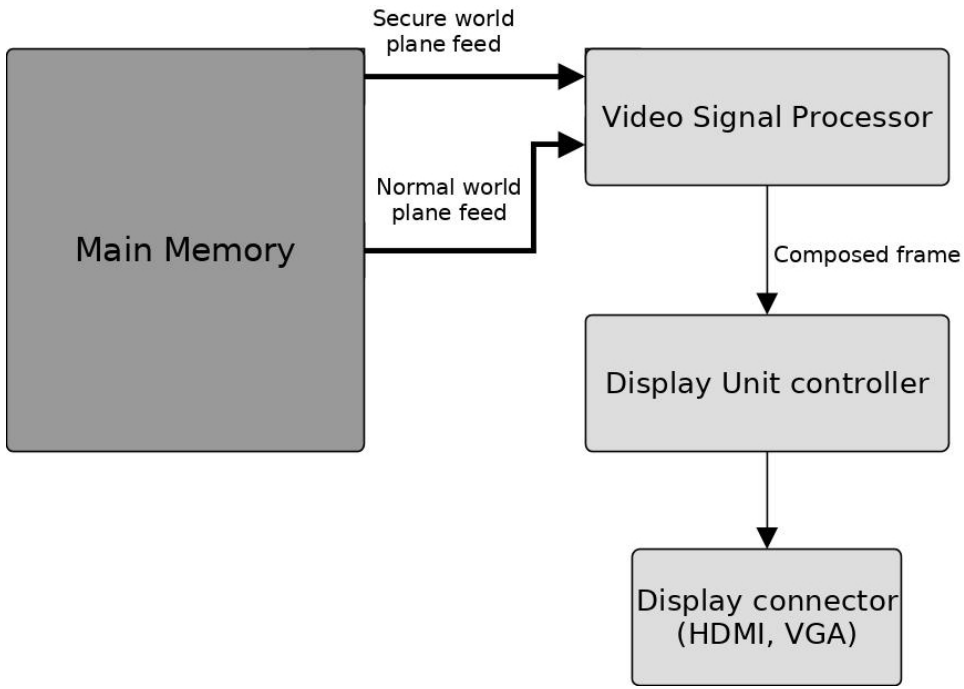
Safe split-display – Image composition

- The RTOS manage the display timings / resolution etc.
  - Changes from the GPOS are prohibited
- On a software crash of the GPOS the graphics pipeline rendering and the *Screen* is not affected



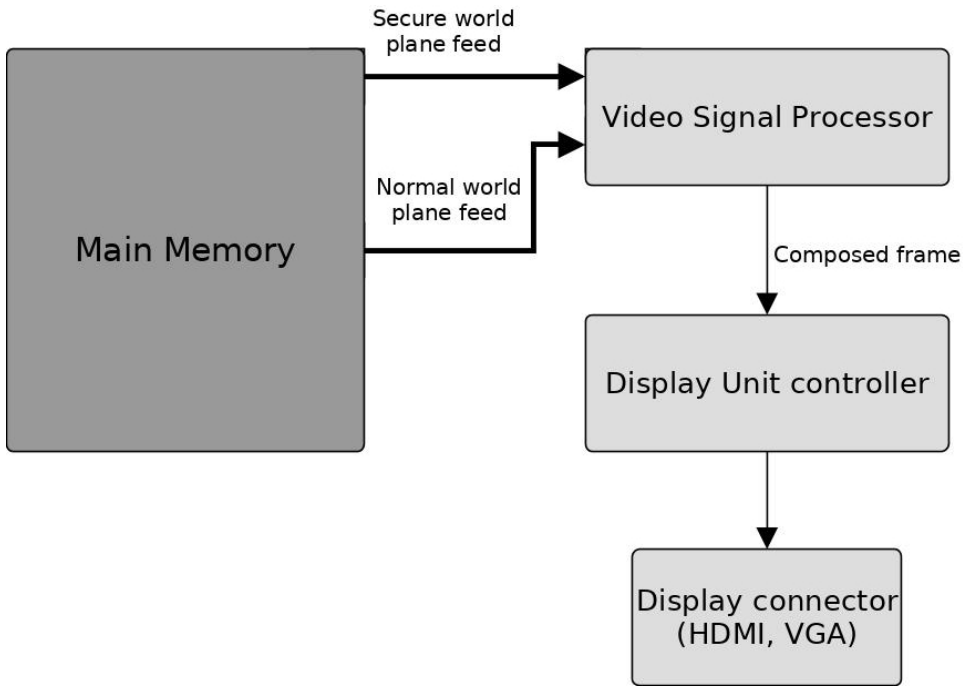
Safe split-display – Graphics pipeline

- Example pipeline for a Renesas R-Car Gen3 SoC
- The graphics pipeline consists of the following components
  - Video Signal Processor
  - Display Unit controller
  - Display Encoder (optional)



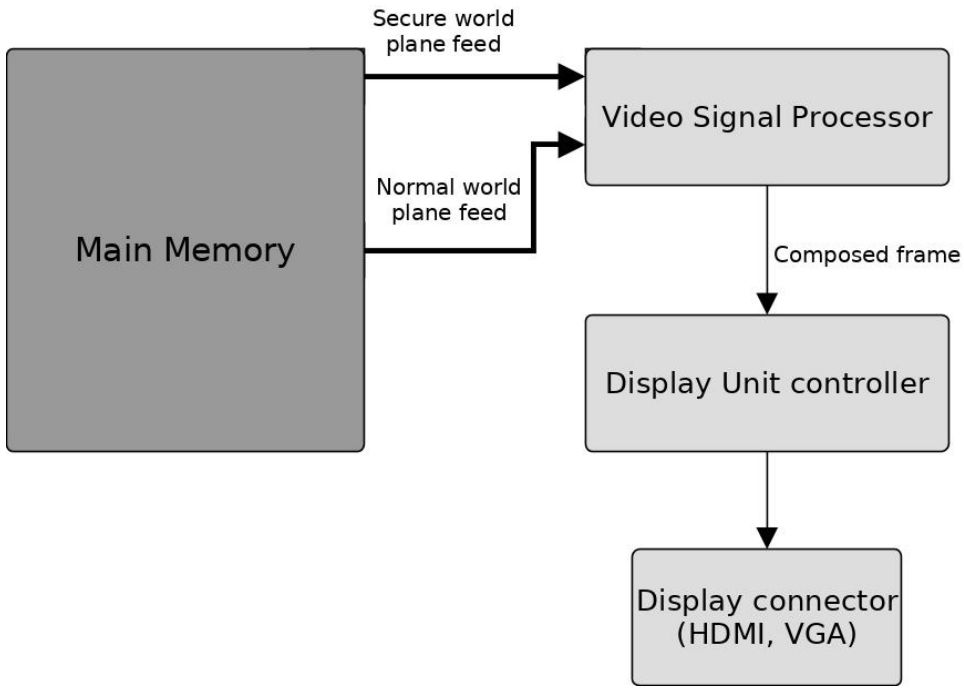
Safe split-display – Graphics pipeline

- Video Signal Processor
  - Fetches the planes from the main memory
  - Composes the frame from the two planes based on the hardware configuration
  - Forwards the frame to the Display Unit controller



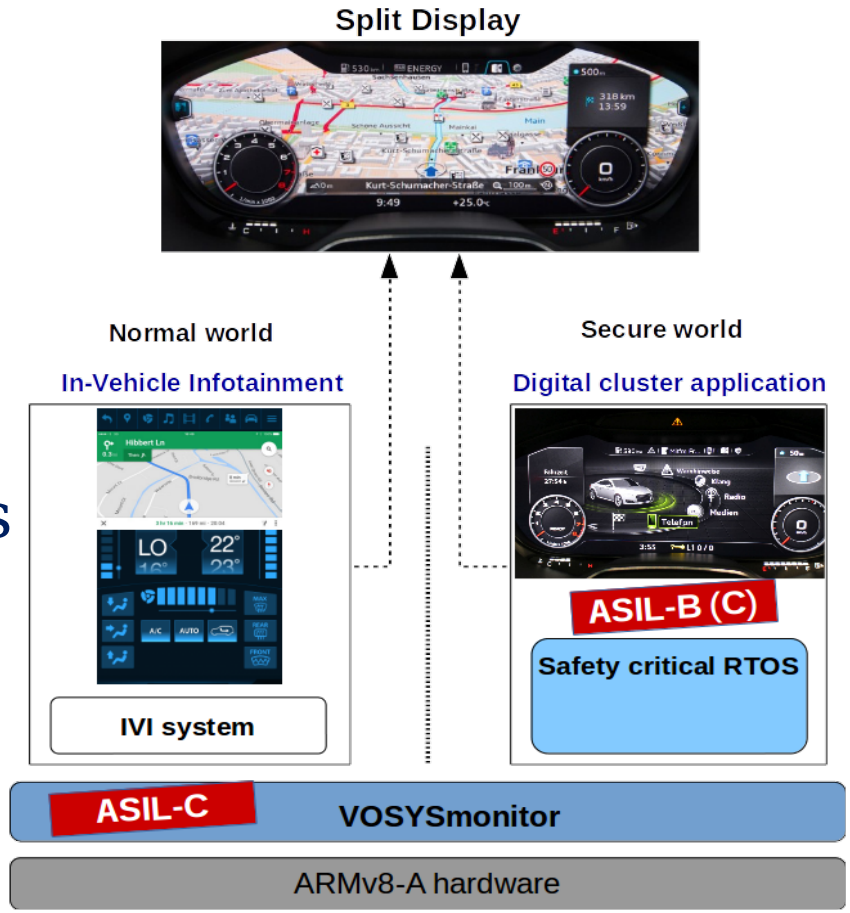
Safe split-display – Graphics pipeline

- Display Unit controller
  - Receives a frame from the Video Signal Processor
  - Controls the *Screen*'s timings / resolution
  - Outputs direct to VGA
  - Or to an encoder
    - e.g., HDMI / LVDS



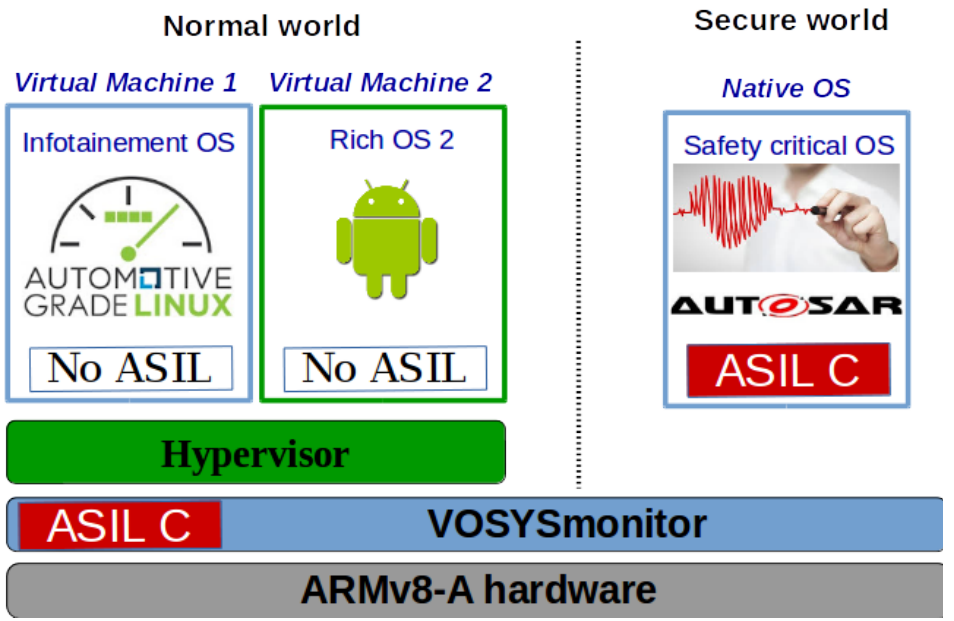
Safe split-display – Rendering

- Normal world
  - Can use graphics acceleration provided by the GPU
  - APIs such as OpenGL ES
- Secure world
  - CPU rendering for critical parts
  - Text messages rendering
  - Warning icons control



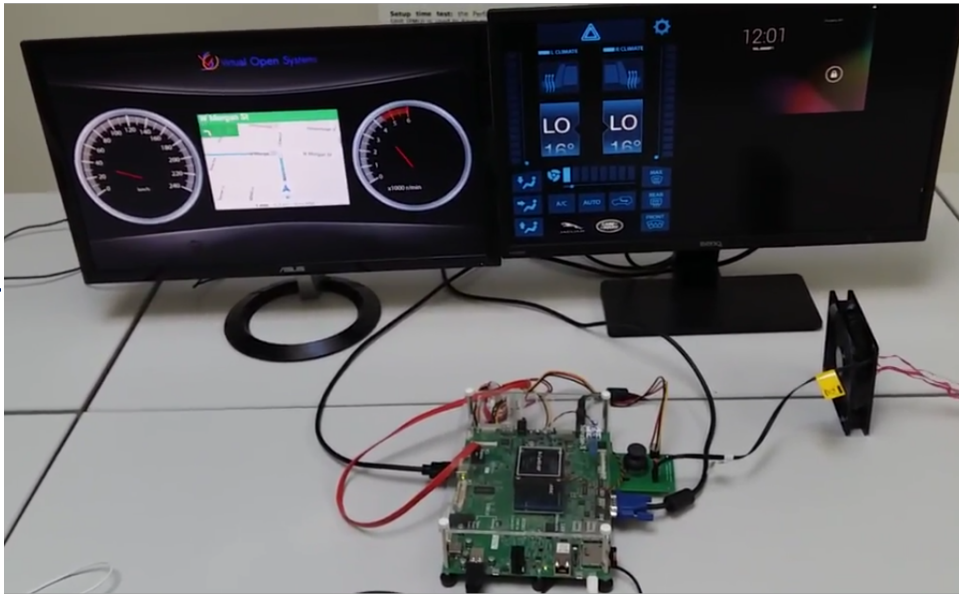
Evaluation - Automotive architecture

- Consists of a rich graphical automotive application
- Certified firmware level
  - VOSYSmonitor
- A certified RTOS
- Virtualized GPOS's
  - Linux and Android
  - No need for certification in normal world



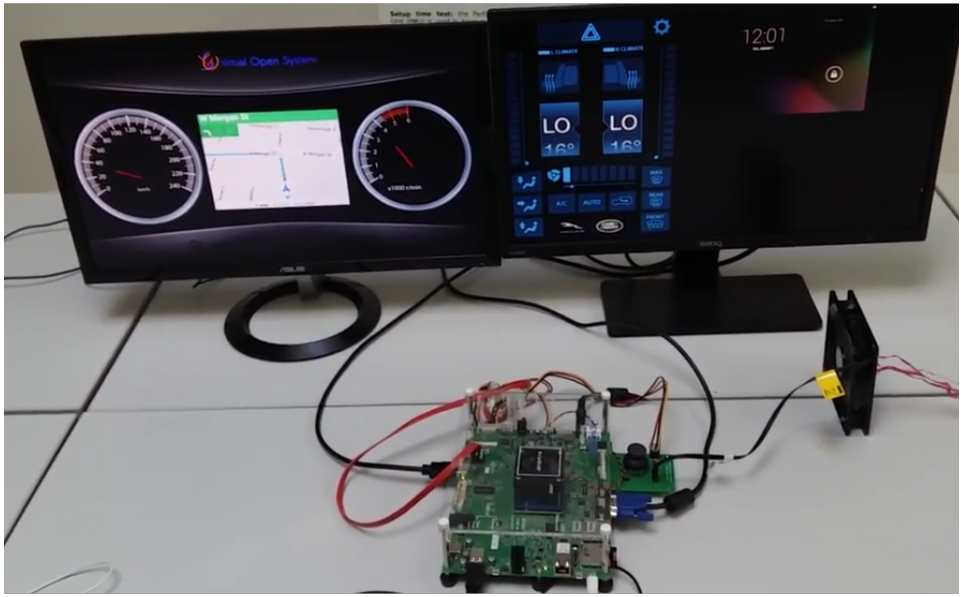
Evaluation - Automotive application

- Using an Renesas R-Car H3
- One *Screen* used by Linux and FreeRTOS (VGA)
  - Cluster, warning icons and text messages rendered from FreeRTOS
  - Navigation maps rendered from Linux
- The second *Screen* used only from Linux (HDMI)
  - HVAC panel and Android VM

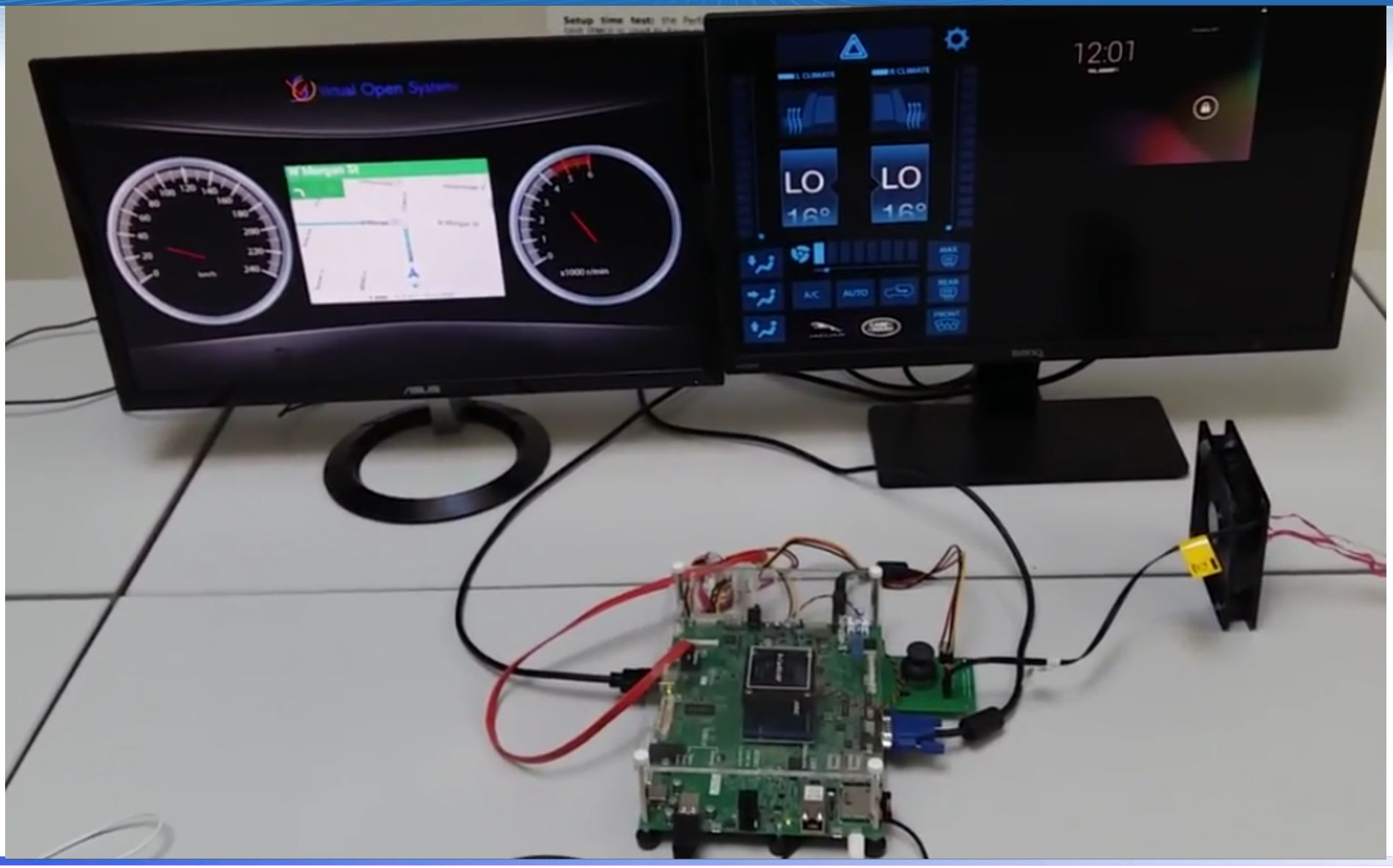


Evaluation - Automotive application

- Planes composition on VGA
- Plane of FreeRTOS:
  - Highest priority
  - Full HD (1920x1080)
- Plane of Linux:
  - Low priority
  - Resolution of 640x480 positioned at the middle of the screen



Evaluation - Automotive application

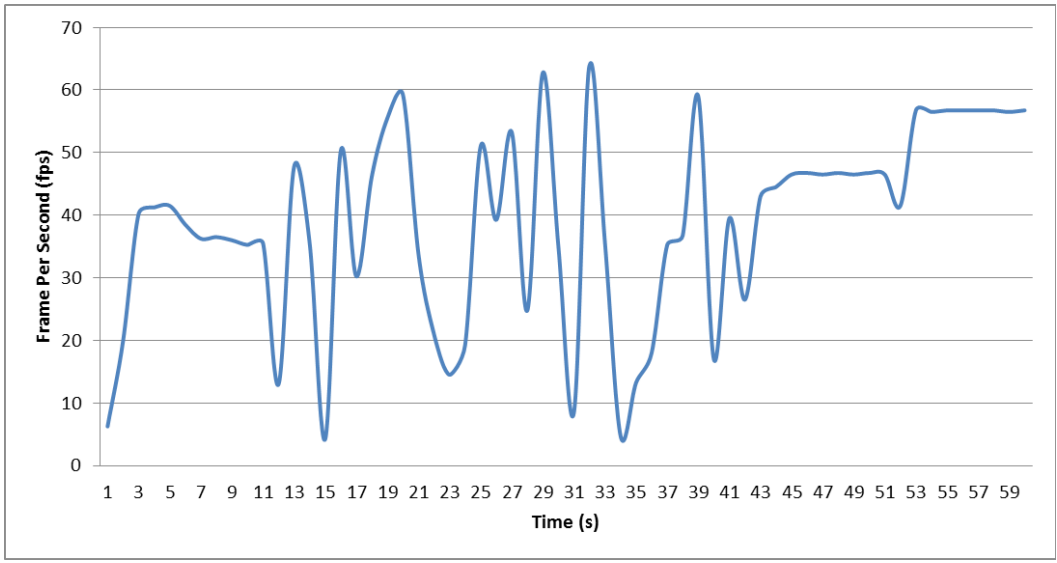


Evaluation

- We measure the rendered frames per second on an idle system
- Then we simulate a kernel crash on Linux
- The graphics pipeline is not affected on a software crash of the Linux system
- On a Linux crash VOSYSmonitor reboots Linux

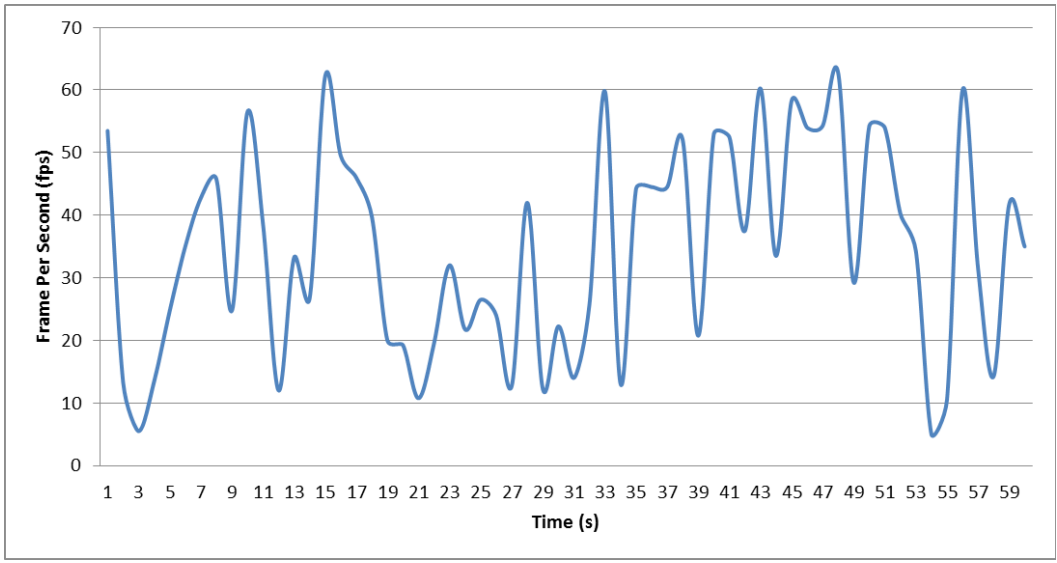
Evaluation – Idle system

- Higher frame rates are produced when there is less animation on the RTOS gauges
- Even on a low rendering frame rate the *Screen* is updated at a standard frame rate defined in the Display Unit driver



Evaluation – GPOS kernel crash

- The crash is simulated at the 10th second
- The RTOS produces the animation frames without impact from the Linux reboot
- Text messages and warning icons are not impacted by the crash



Conclusion

- Safe split-display
  - Architecture for security and safety aware display sharing
  - Drivers co-operation architecture using RPC
  - Separation of graphics pipeline components
  - Strict isolation for critical information rendering
    - Even on an non recoverable crash of the GPOS
- A video demo is available at Virtual Open Systems page:
  - http://www.virtualopensystems.com/en/solutions/demos/vosysmonitor-als2017/

Virtual Open Systems