BMW GROUP

THE COMPLEXITY OF AUTOMOTIVE SWITCH SOFTWARE: CHALLENGES AND APPROACHES TOWARDS A STANDARD

2024 - IEEE SA E&IP@ATD, DETROIT – BMW, STÉFANY CHOURAKORN

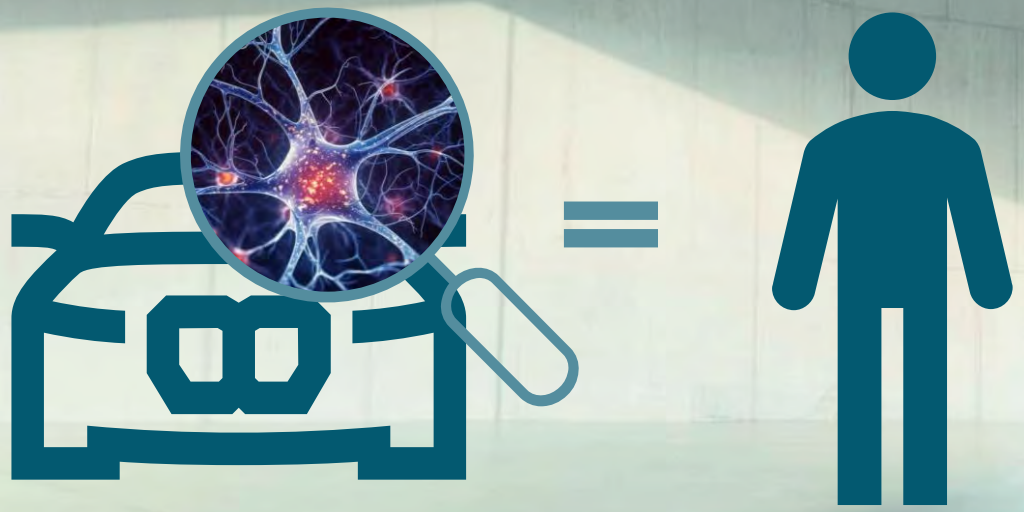
IN-VEHICLE NETWORK COMPLEXITY = HUMAN BODY. ANALOGY.

Coordination

Motor

Powertrain

Ethernet, LIN, CAN,...



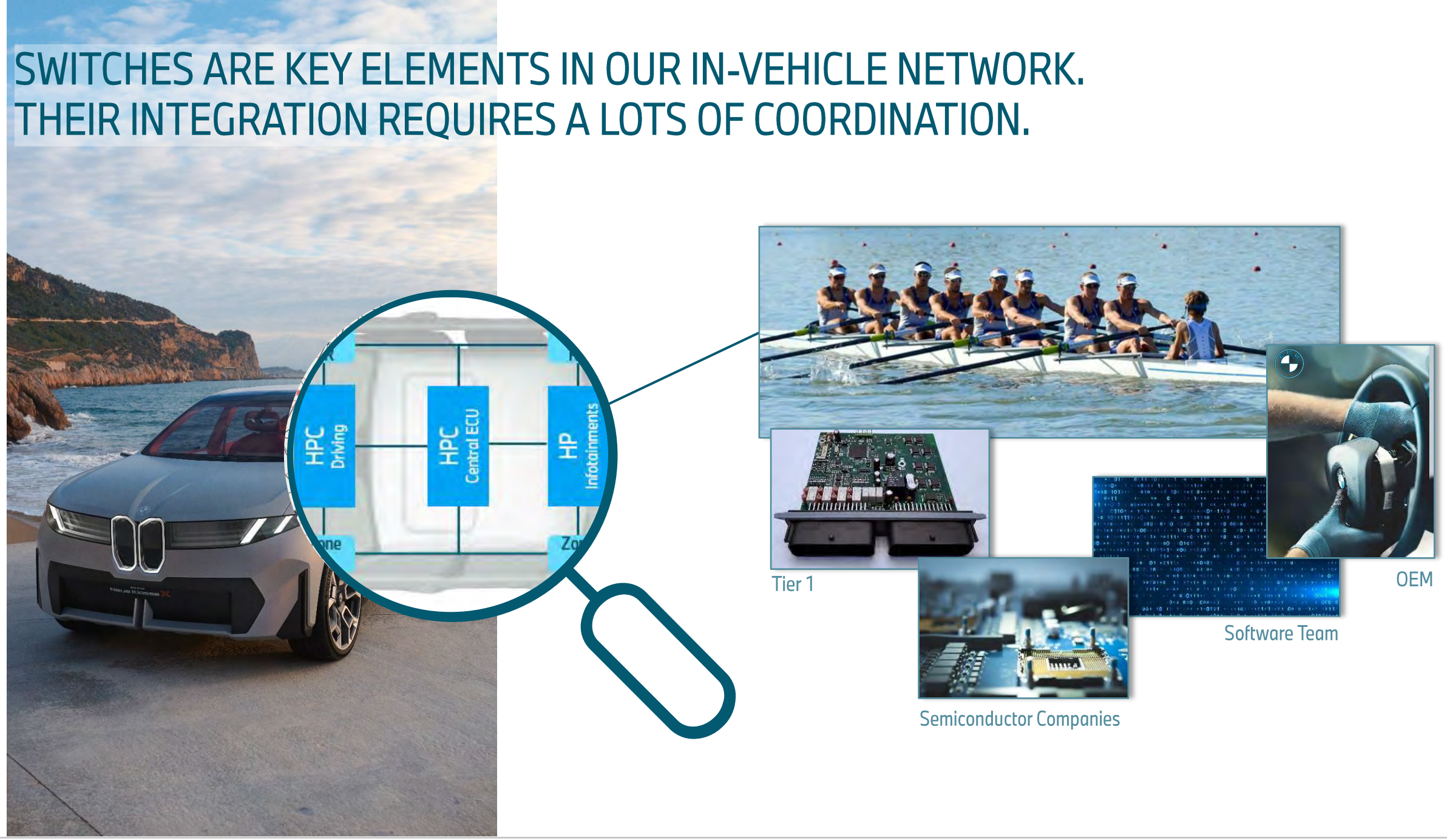
Brain

Heart

Digestive system

Blood vessel, hormones, nerves

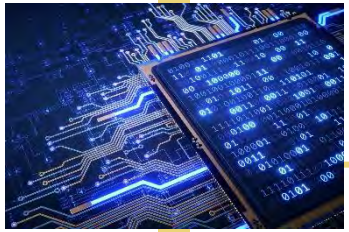
SWITCHES ARE KEY ELEMENTS IN OUR IN-VEHICLE NETWORK. THEIR INTEGRATION REQUIRES A LOTS OF COORDINATION.



AGENDA.

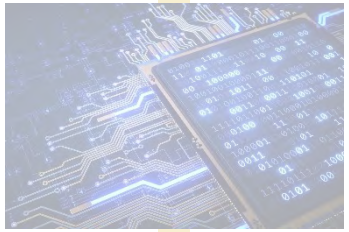
Switch Integration TODAY

Why standardization ?



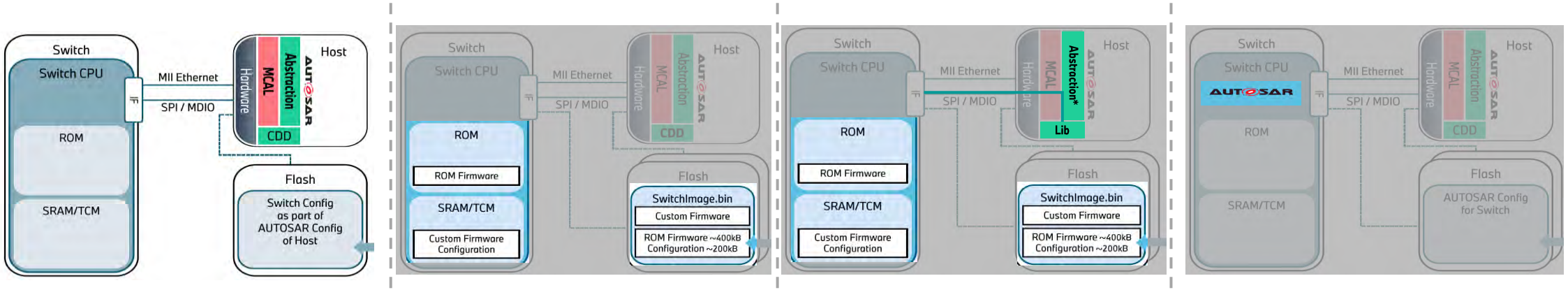
Switch Integration TOMORROW

Mission North star, our objectives



SWITCH INTEGRATION TODAY

DIFFERENT SCENARIOS.
FLASHBACK AEC 2023.

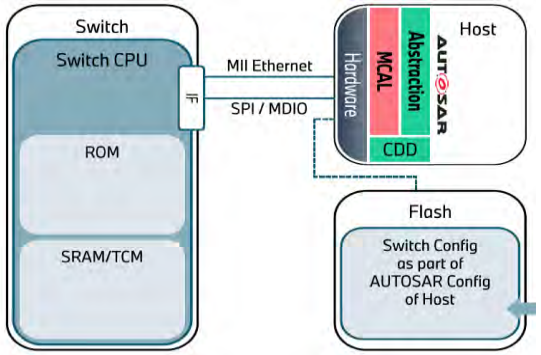


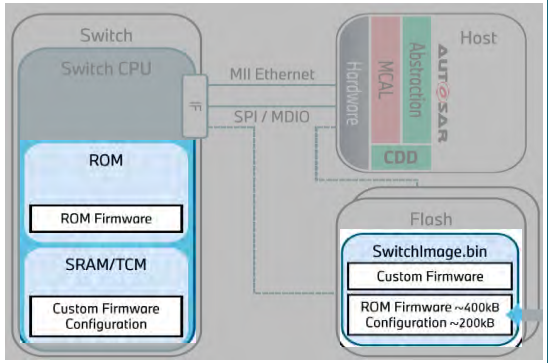
	Unmanaged Switch	Classic Switch	Hybrid Switch	AUTOSAR Switch
Host CPU utilization	− −	+ +	+ +	+ +
Memory foot-print	+ +	+ +	+	+
Extensibility of feature set	− −	+ +	+ +	− −
Compliance, Inter-OP & Co	− / +	+	+	− / +
Start-up limiations / speed	− −	+ +	+ +	+
Debugging standards	+	−	− / +	+
Know-how transfer	+ +	− −	+	+ +

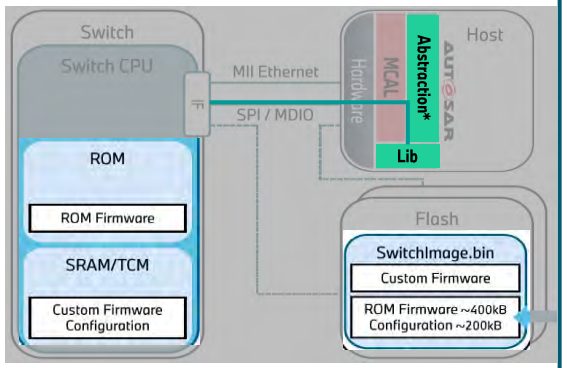
2023 - K. Budweiser, L. Jürgensen, Automotive Ethernet Congress

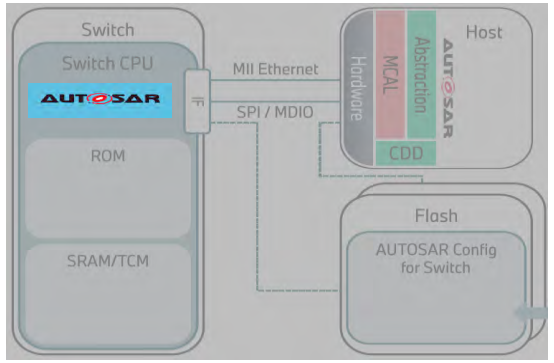
DIFFERENT SCENARIOS.
FLASHBACK AEC 2023.

→ How does this look like in series projects?







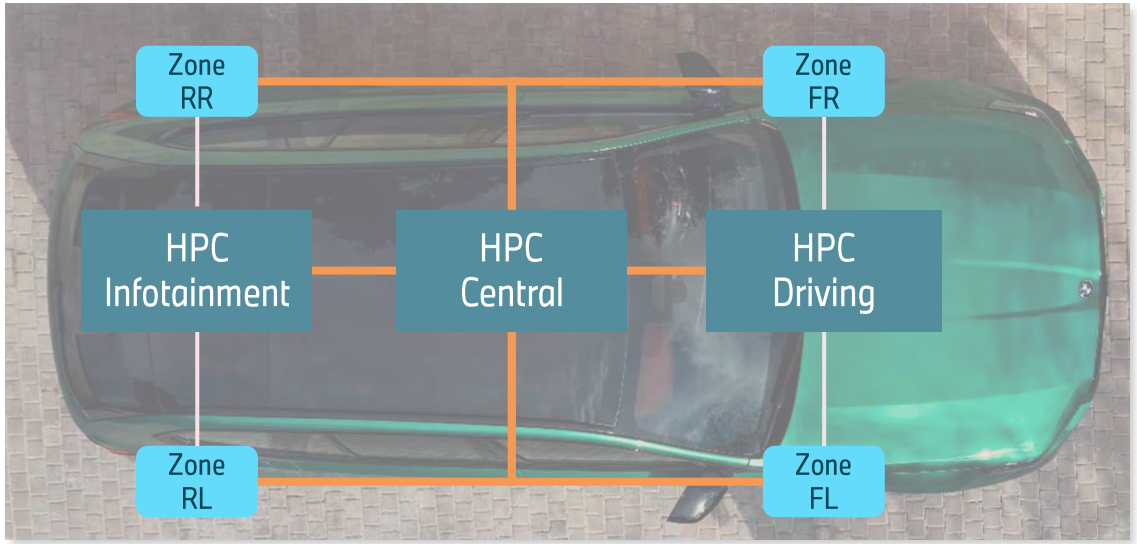


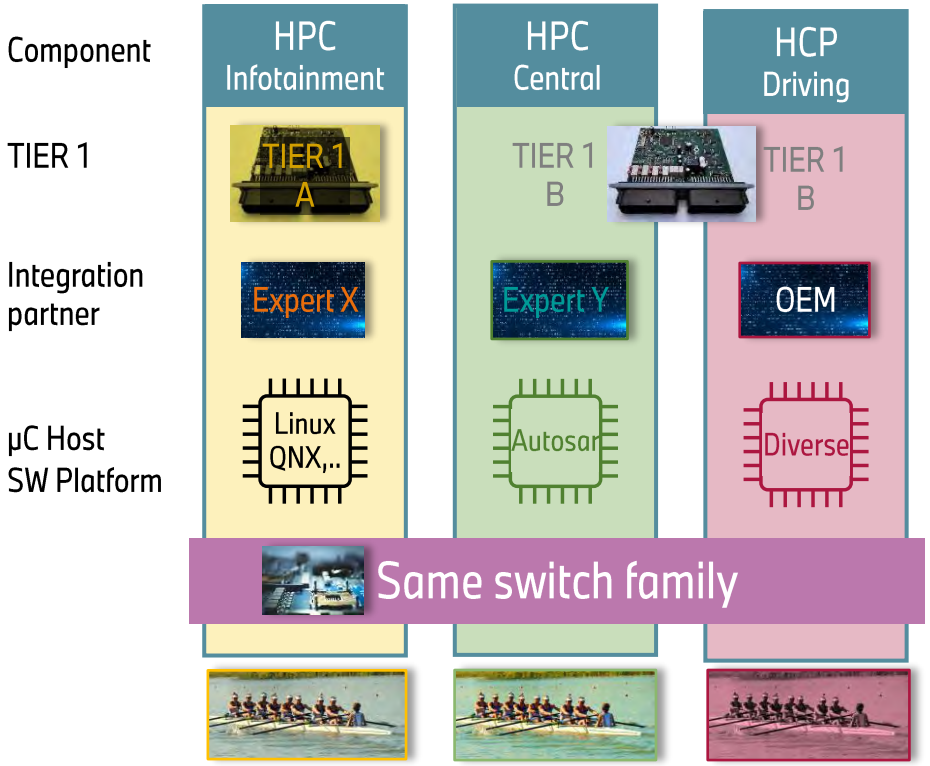
	Unmanaged Switch	Classic Switch	Hybrid Switch	AUTOSAR Switch
Host CPU utilization	− −	+ +	+ +	+ +
Memory foot-print	+ +	+ +	+	+
Extensibility of feature set	− −	+ +	+ +	− −
Compliance, Inter-OP & Co	− / +	+	+	− / +
Start-up limiations / speed	− −	+ +	+ +	+
Debugging standards	+	−	− / +	+
Know-how transfer	+ +	− −	+	+ +

2023 - K. Budweiser, L. Jürgensen, Automotive Ethernet Congress

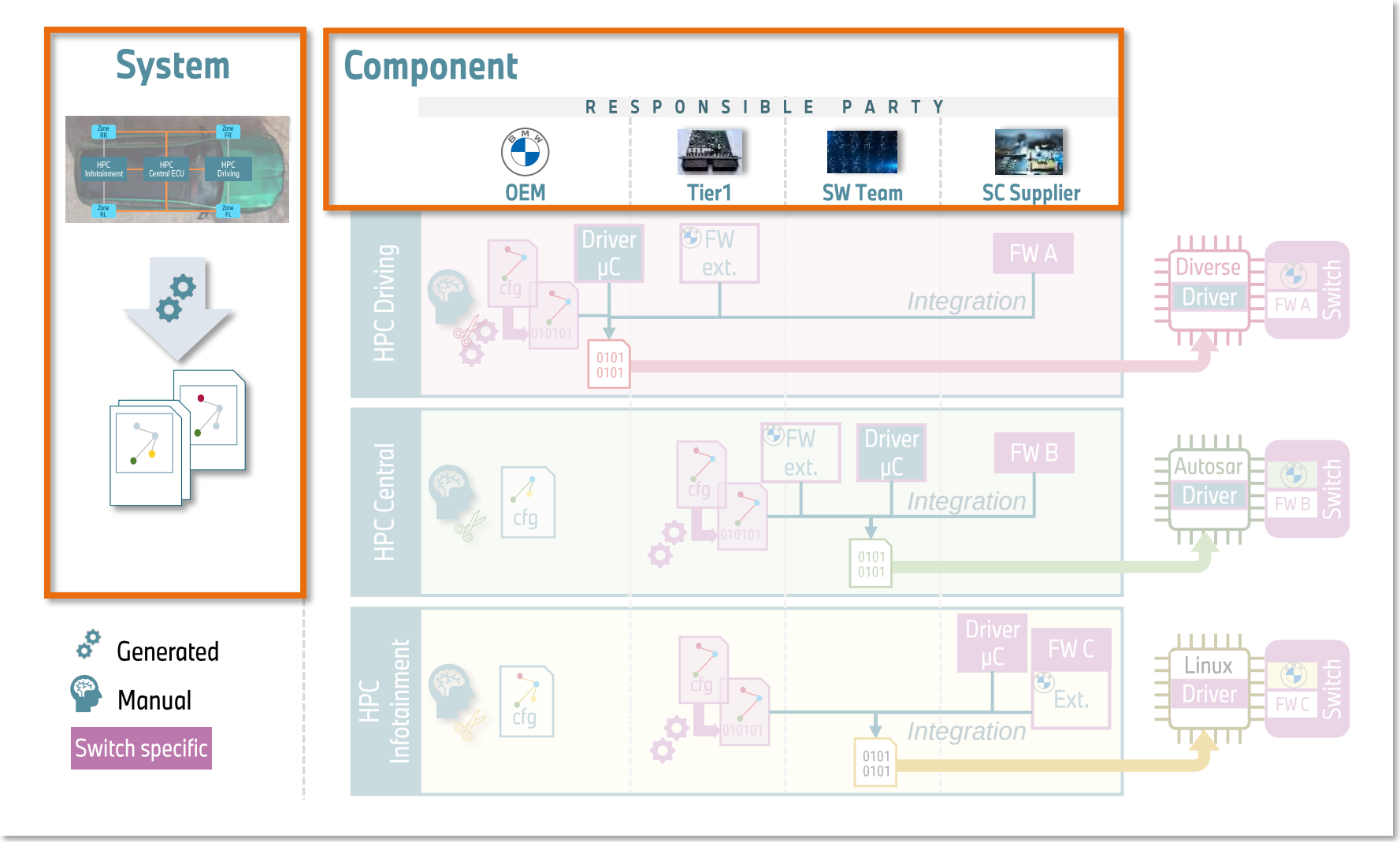
PLATFORM DEVELOPMENT FOR A ZONAL ARCHITECTURE.
CASE STUDY | HYBRID SWITCH.

System overview as an example:

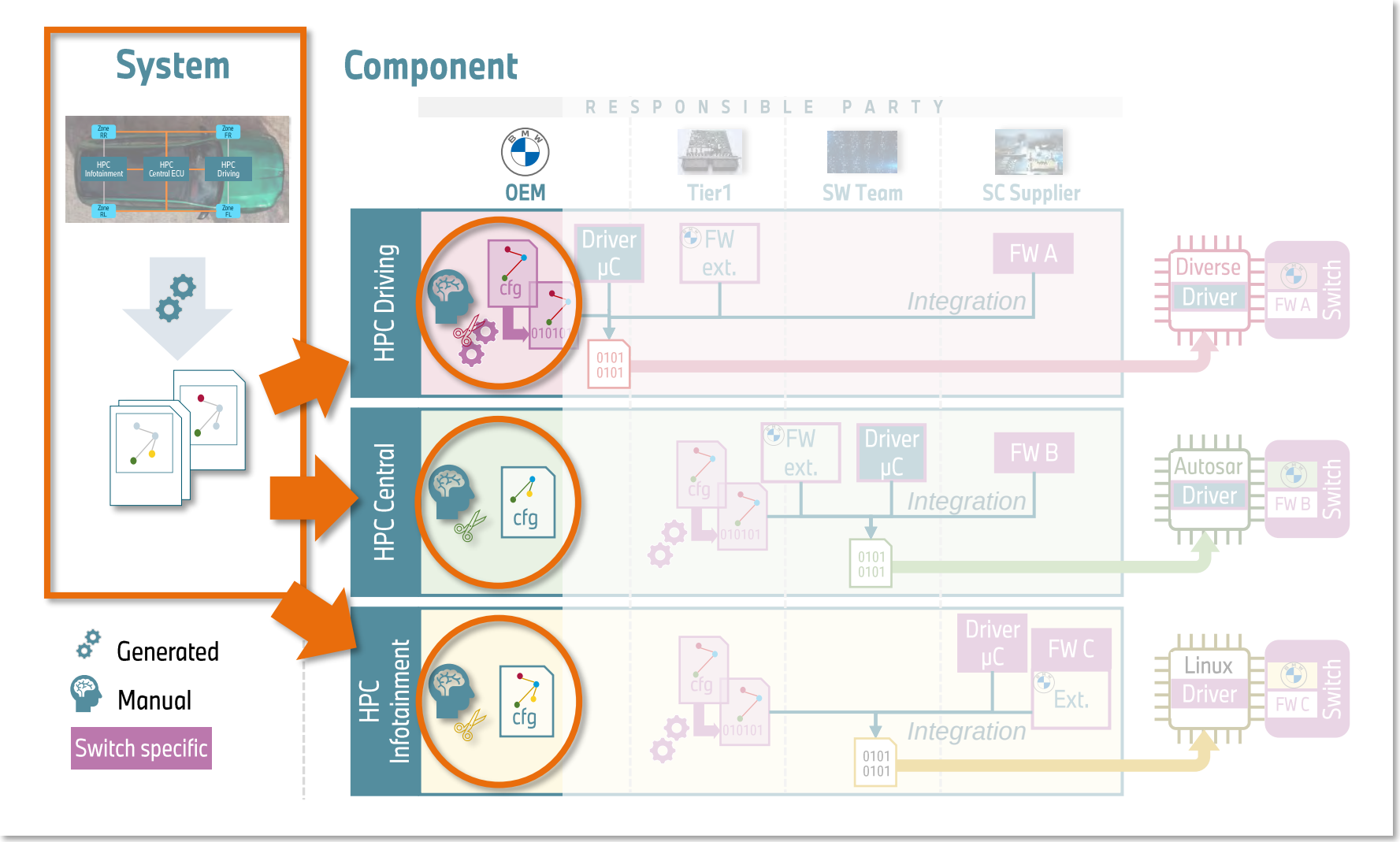




WORKFLOW AND ORGANIZATION ARE DIFFERENT.
CASE STUDY.

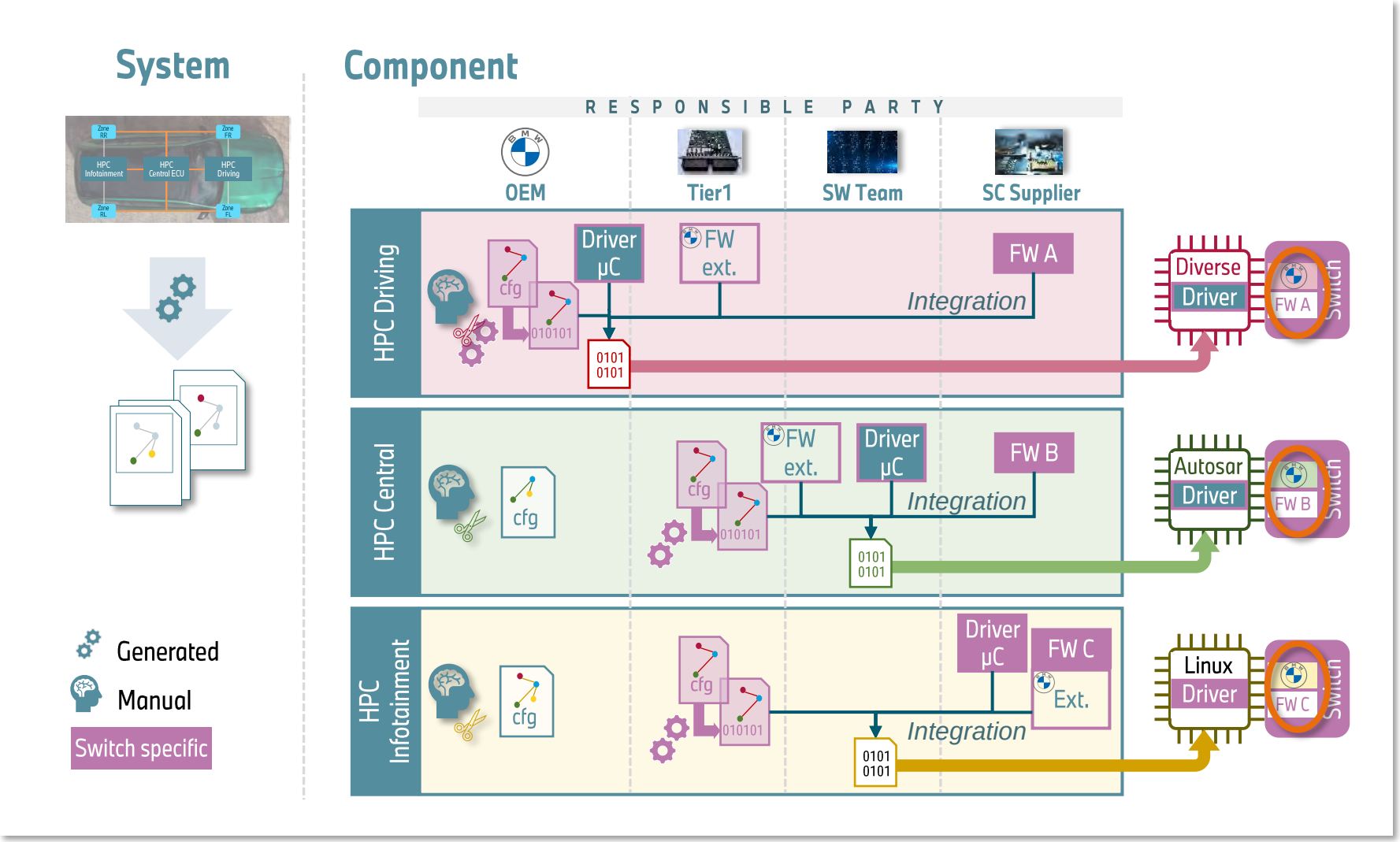


WORKFLOW AND ORGANIZATION ARE DIFFERENT.
CASE STUDY.



- **System configuration** is **manually tailored** on component level by OEM
- Proprietary toolchain

WORKFLOW AND ORGANIZATION ARE DIFFERENT.
CASE STUDY.

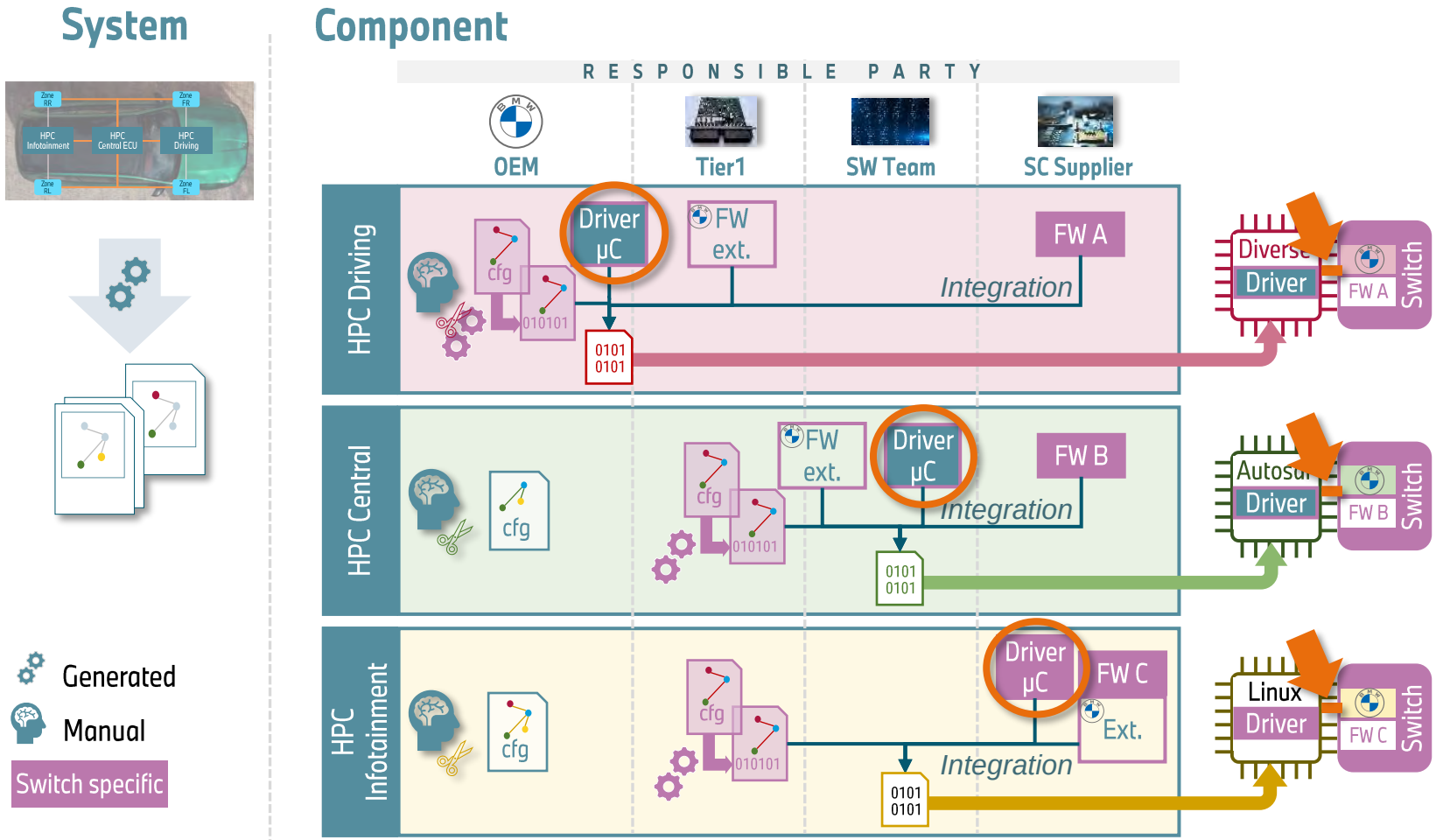


System configuration is **manually tailored** on component level by OEM

Proprietary toolchain

FW Dependency FW & OEM extensions

WORKFLOW AND ORGANIZATION ARE DIFFERENT.
CASE STUDY.

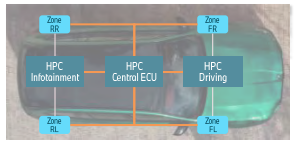


- **System configuration** is **manually tailored** on component level by OEM
- Proprietary toolchain
- FW Dependency FW & OEM extensions
- Driver OS-specific driver for proprietary protocol
  - ➢ **No synergy nor reuse**

Current situation doesn't provide efficient workflow: a standardized development path (i.e. ECU process) is not available yet.

WORKFLOW AND ORGANIZATION ARE DIFFERENT.
CASE STUDY.

System

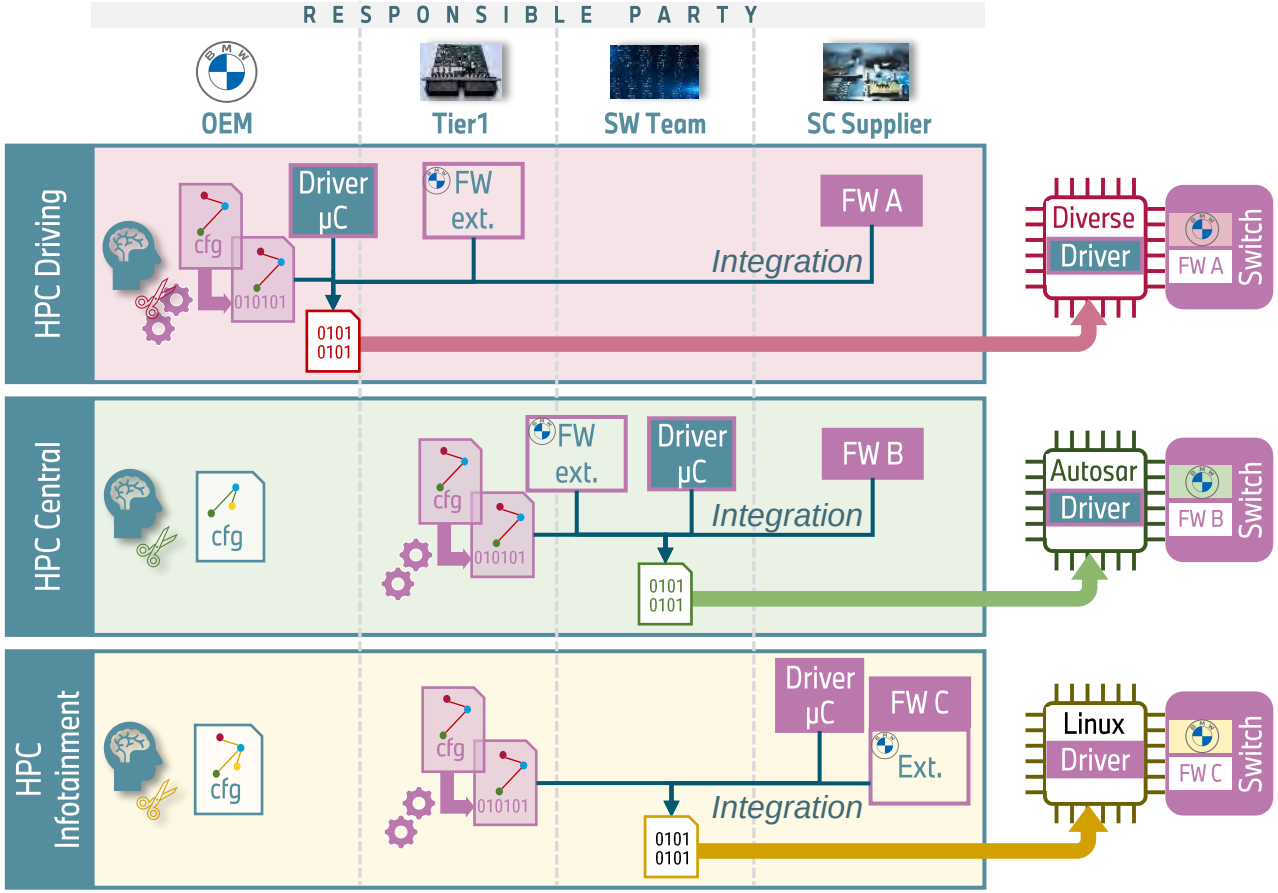


Generated

Manual

Switch specific

Component



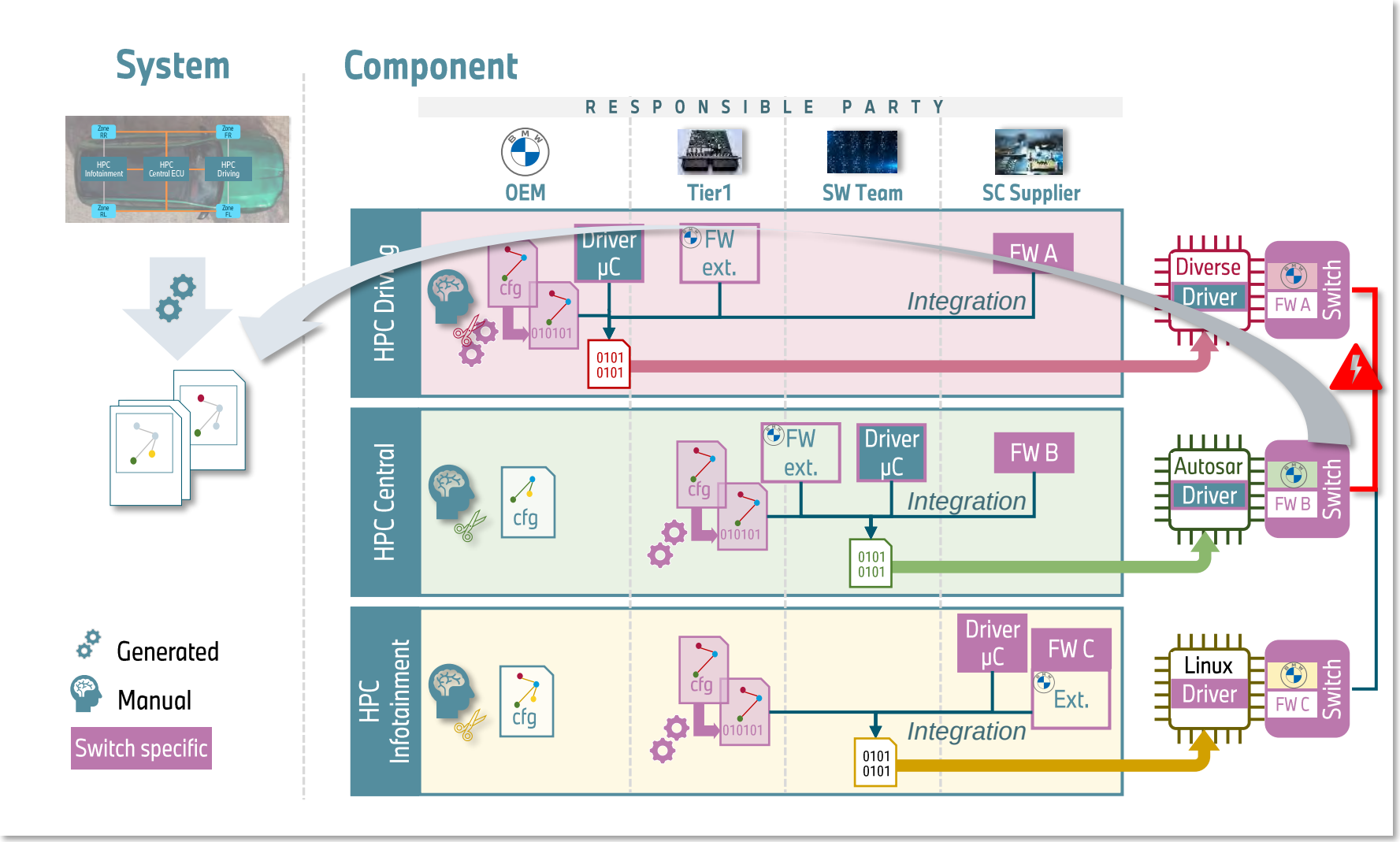
- **System configuration** is **manually tailored** on component level by OEM
- Proprietary toolchain
- FW Dependency FW & OEM extensions
- Driver OS-specific driver for proprietary protocol
  - ➢ **No synergy nor reuse**

- Most switch vendors provide toolchain for configuration
  - ➢ **Proprietary solutions arose**
- Switch and PHY initialization in switch FW (not in main HPC)

Bug handling ?

Current situation doesn't provide efficient workflow: a standardized development path (i.e. ECU process) is not available yet.

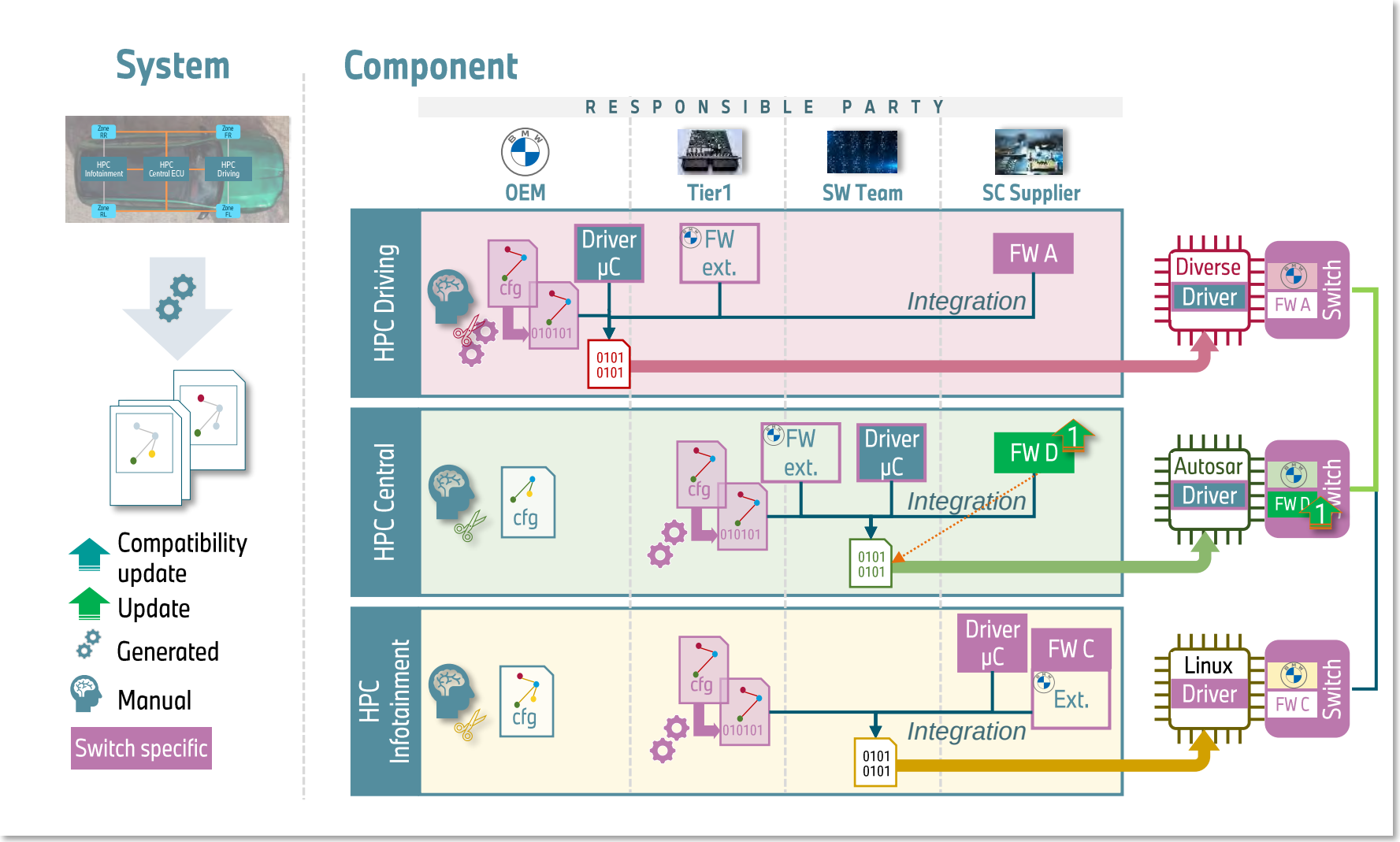
DEBUGGING REMAINS COMPLEX.
CASE STUDY.



- At system integration (OEM)
- Error reproduction for analysis (all parties)
  - Time and effort consuming

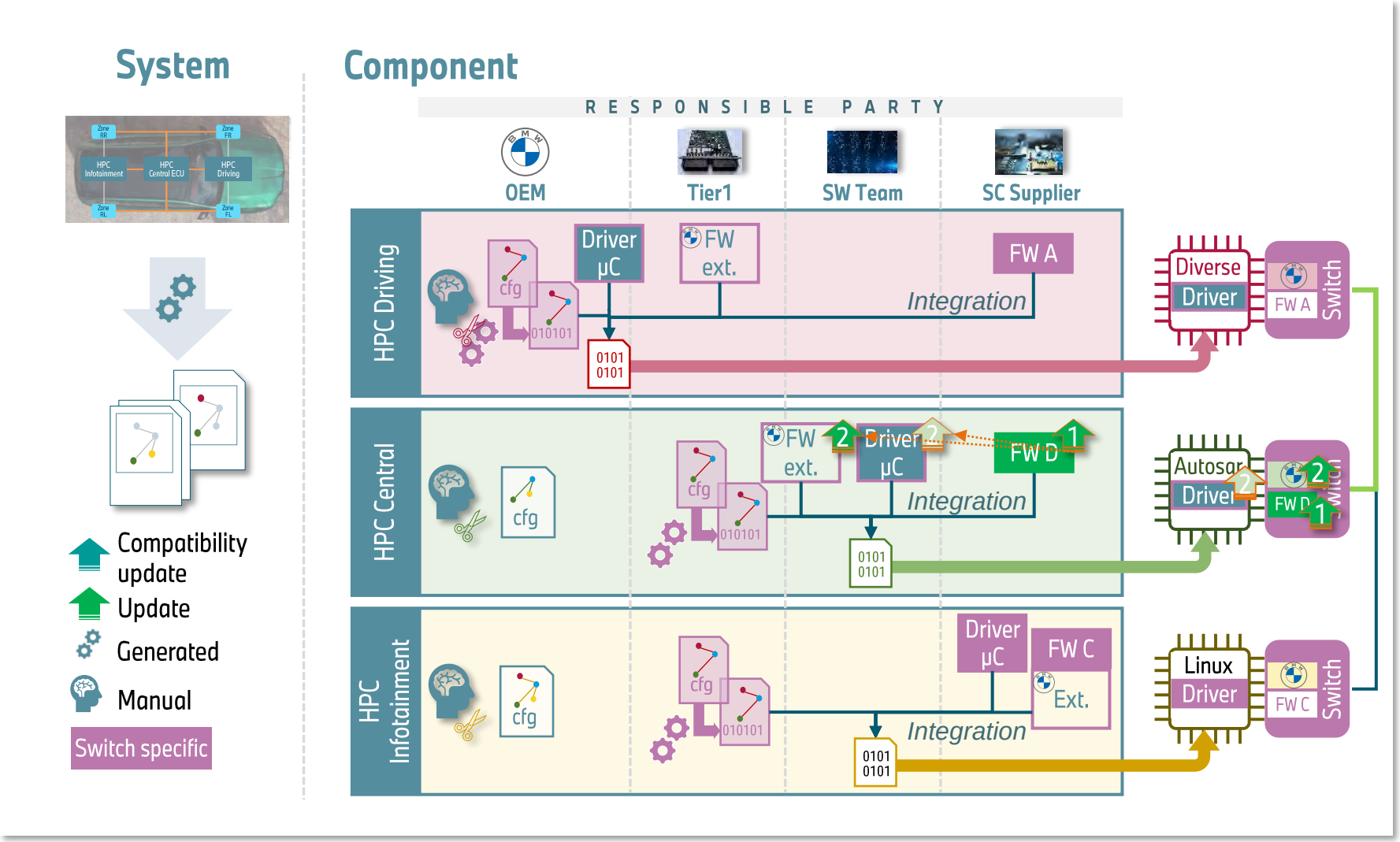
Critical Firmware and HW fix
- SW dependencies
- Distribution
- Version compatibility

DEBUGGING REMAINS COMPLEX.
CASE STUDY.



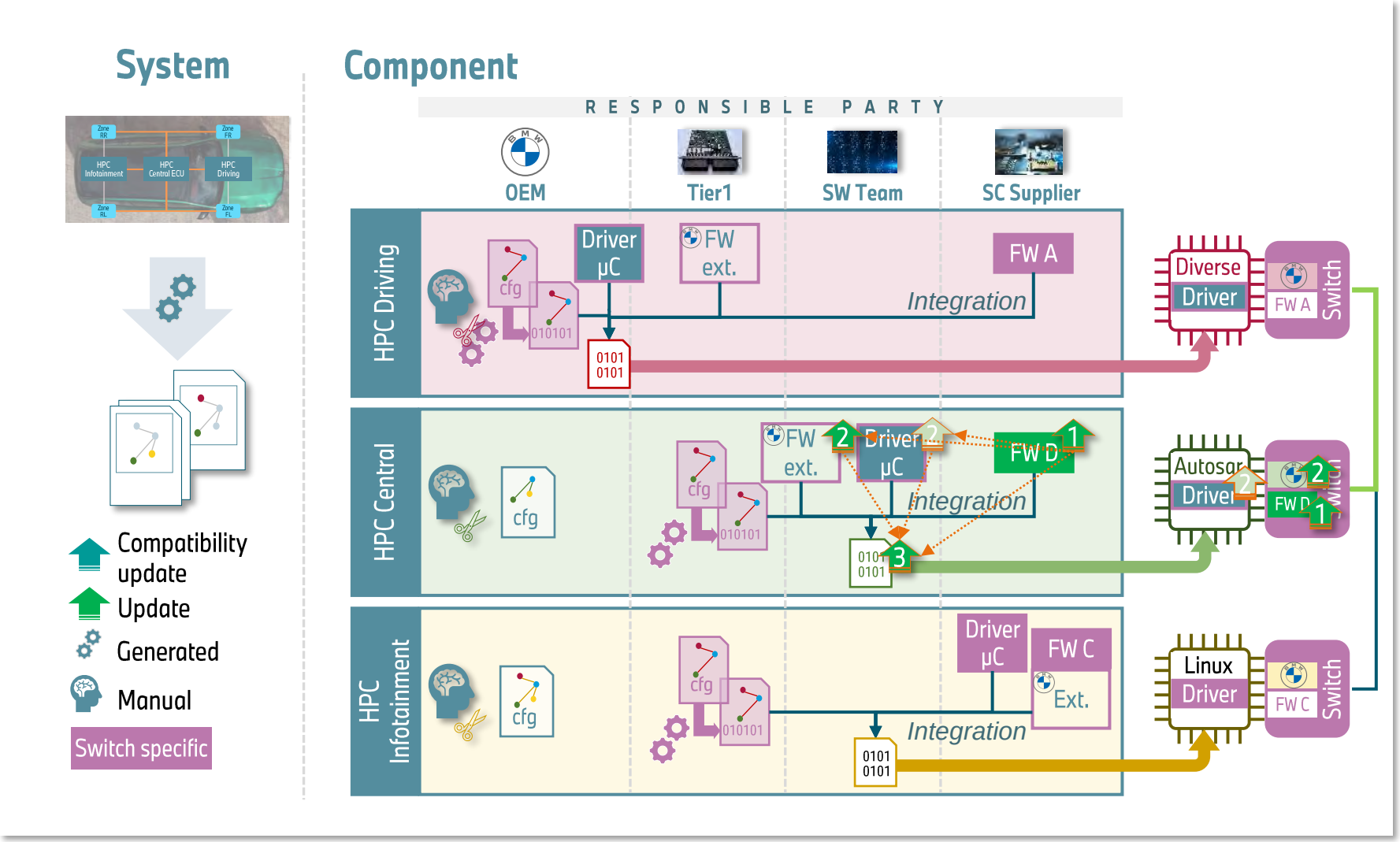
- Firmware update: new integration iteration is triggered
  - 1 Upgrade firmware (switch vendor)

WORKFLOW AND ORGANIZATION ARE DIFFERENT.
CASE STUDY.



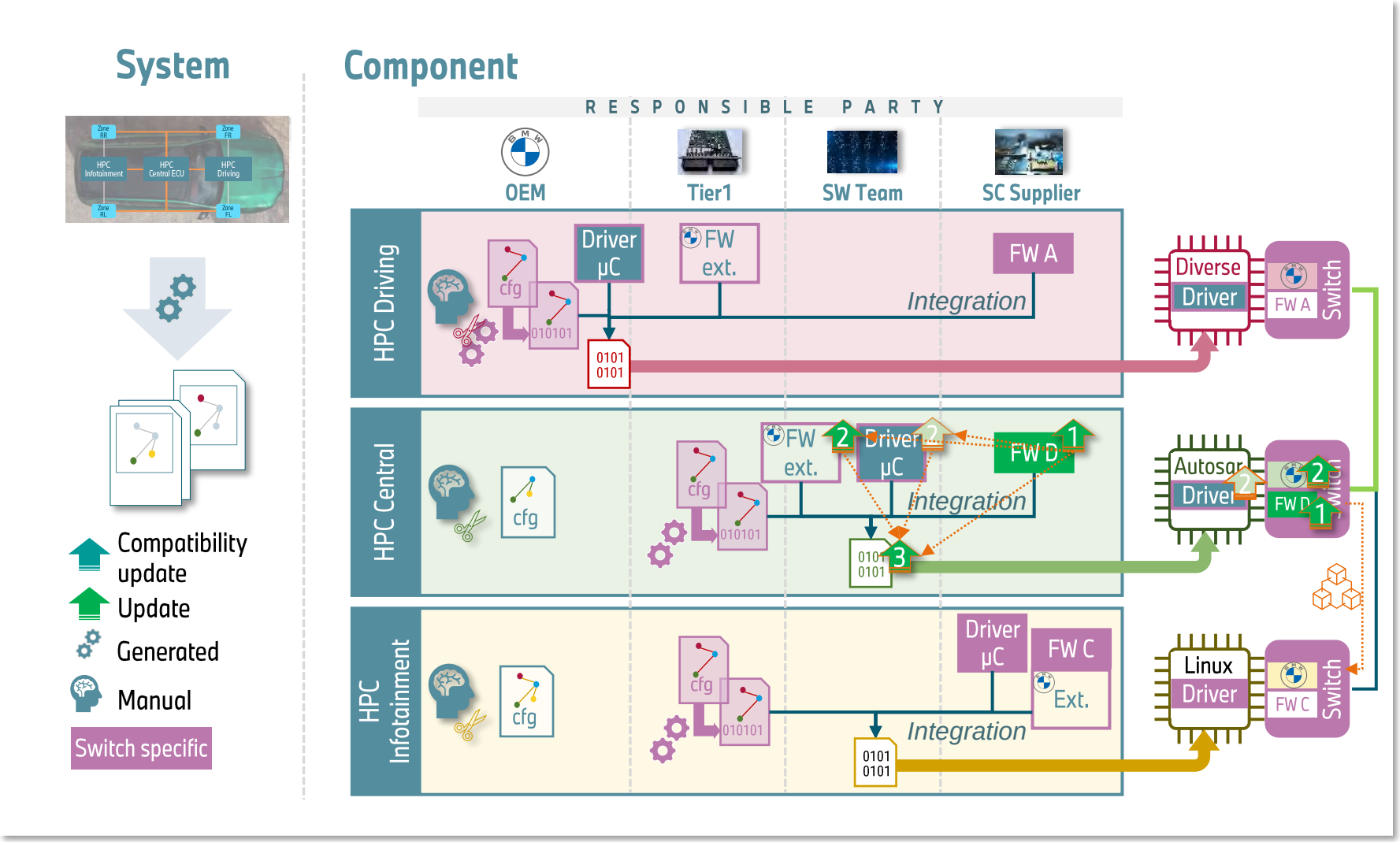
- Firmware update: new integration iteration is triggered
  - 1 Upgrade firmware (switch vendor)
  - 2 Reintegrate OEM extensions + update µC driver

WORKFLOW AND ORGANIZATION ARE DIFFERENT.
CASE STUDY.



- Firmware update: new integration iteration is triggered
  - 1 Upgrade firmware (switch vendor)
  - 2 Reintegrate OEM extensions + update µC driver
  - 3 Regenerate build and validate

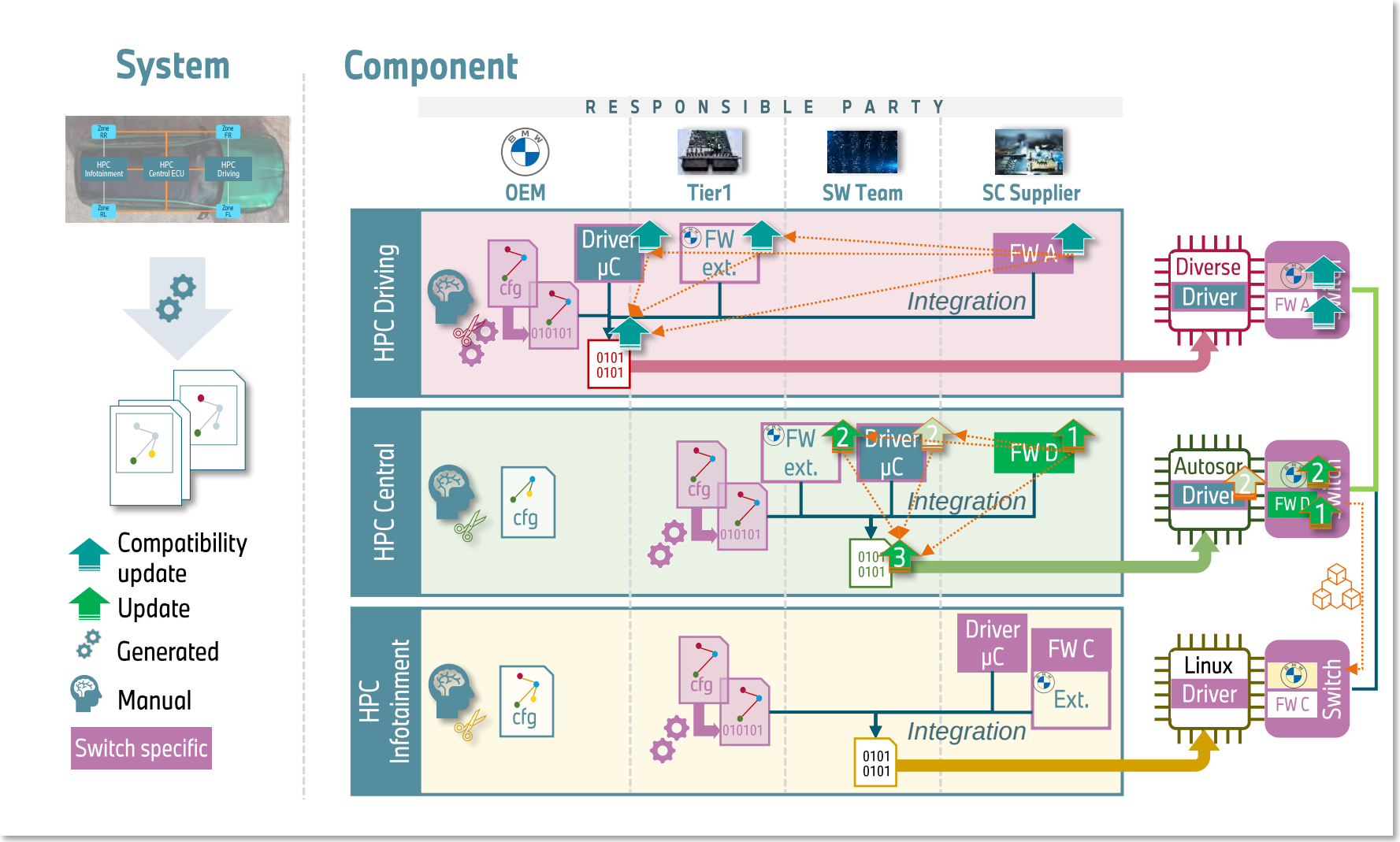
WORKFLOW AND ORGANIZATION ARE DIFFERENT.
CASE STUDY.



- Firmware update: new integration iteration is triggered
  - 1 Upgrade firmware (switch vendor)
  - 2 Reintegrate OEM extensions + update µC driver
  - 3 Regenerate build and validate

Revision maintenance – API

WORKFLOW AND ORGANIZATION ARE DIFFERENT.
CASE STUDY.

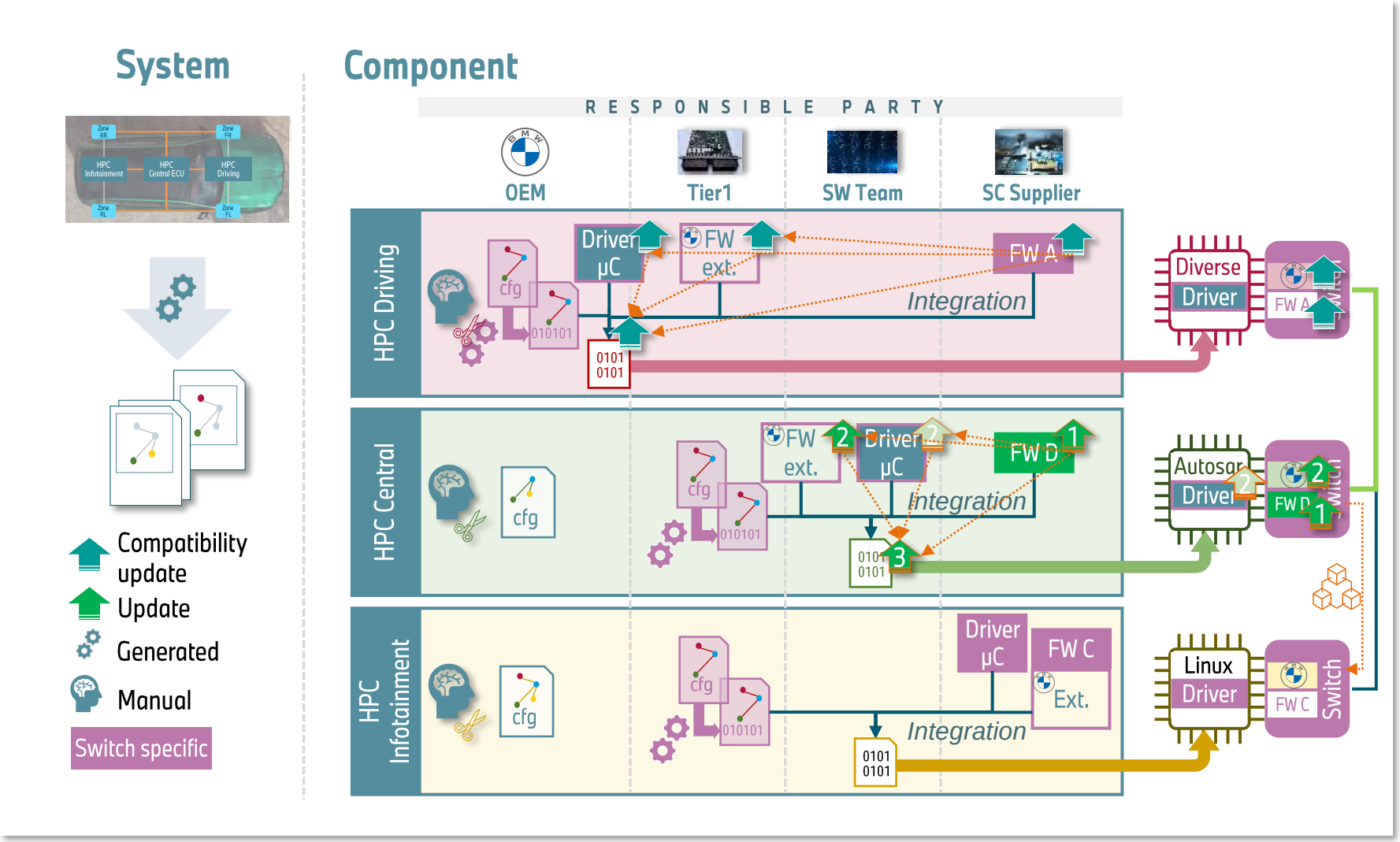


- Firmware update: new integration iteration is triggered
  - 1 Upgrade firmware (switch vendor)
  - 2 Reintegrate OEM extensions + update µC driver
  - 3 Regenerate build and validate

Revision maintenance – API

Rolling out fix/patch

WORKFLOW AND ORGANIZATION ARE DIFFERENT.
CASE STUDY.



- Firmware update: new integration iteration is triggered
  - 1 Upgrade firmware (switch vendor)
  - 2 Reintegrate OEM extensions + update µC driver
  - 3 Regenerate build and validate

Revision maintenance – API

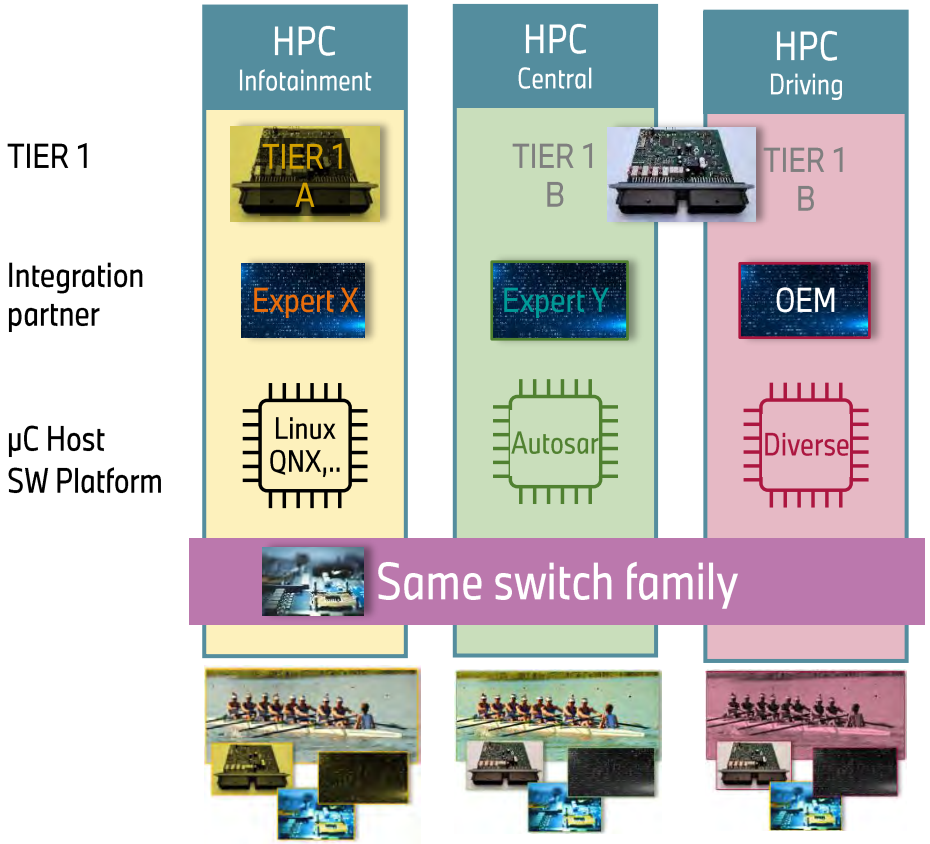
Rolling out fix/patch

Fixes post homologation or post SOP?

How many years switch vendor maintain their SW release ?

SWITCH MANAGEMENT AND CONFIGURATION IS CHALLENGING AND AFFECTS EVERYONE !

- Switches are one of the most complex subsystem within a component and yet are not involved in customer functionality
- HW abstraction is not 100% possible
- **Standardized interfaces** enable greater reuse of software to reduce time, effort and costs and ensure compatibility between systems
- **Generic configuration** reduce manually caused errors: automated workflow and greater compatibility between systems
- Testing and bug fixing is an aspect that should not be neglected: SW Maintenance 10 years after SOP?



Keep in mind:

*Our study case was **simplified** and did not included other switch vendors.*

WHY STANDARDIZATION ?

WHY STANDARDIZATION ?

Generic SW for Efficiency and reliability

- **SW re-use** via generic SW interfaces
  - Reduce risk towards delay
  - Focus on function development
- Less **interdependencies and early phase** validation (SW/HW)
- **Efficient** bug finding and bug fixing

HW abstraction for modularity and flexibility

- Stronger design for new feature and architecture scalability
- Reduce **time** and **complexity** in ECU development
- **Flexibility** increased against **country regulation** fluctuation

… lead to:

- Common base / language to become more efficient in a multi-party projects
- We need to establish an ecosystem
- Non OEM specific solution can be profitable for other stakeholder

Universal configuration format

- Toolchain **automation**
- **Uniform** configuration
- Smooth **porting** and **migration**

WHY STANDARDIZATION ?

Generic SW for Efficiency and reliability

- **SW re-use** via generic SW interfaces
  - Reduce risk towards delay
  - Focus on function development
- Less **interdependencies and early phase** validation (SW/HW)
- **Efficient** bug finding and bug fixing

Universal configuration format

- Toolchain **automation**
- **Uniform** configuration
- Smooth **porting** and **migration**

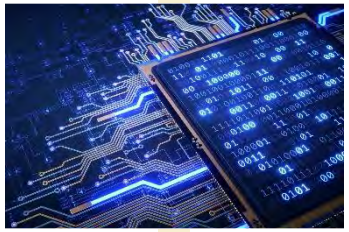
HW abstraction for modularity and flexibility

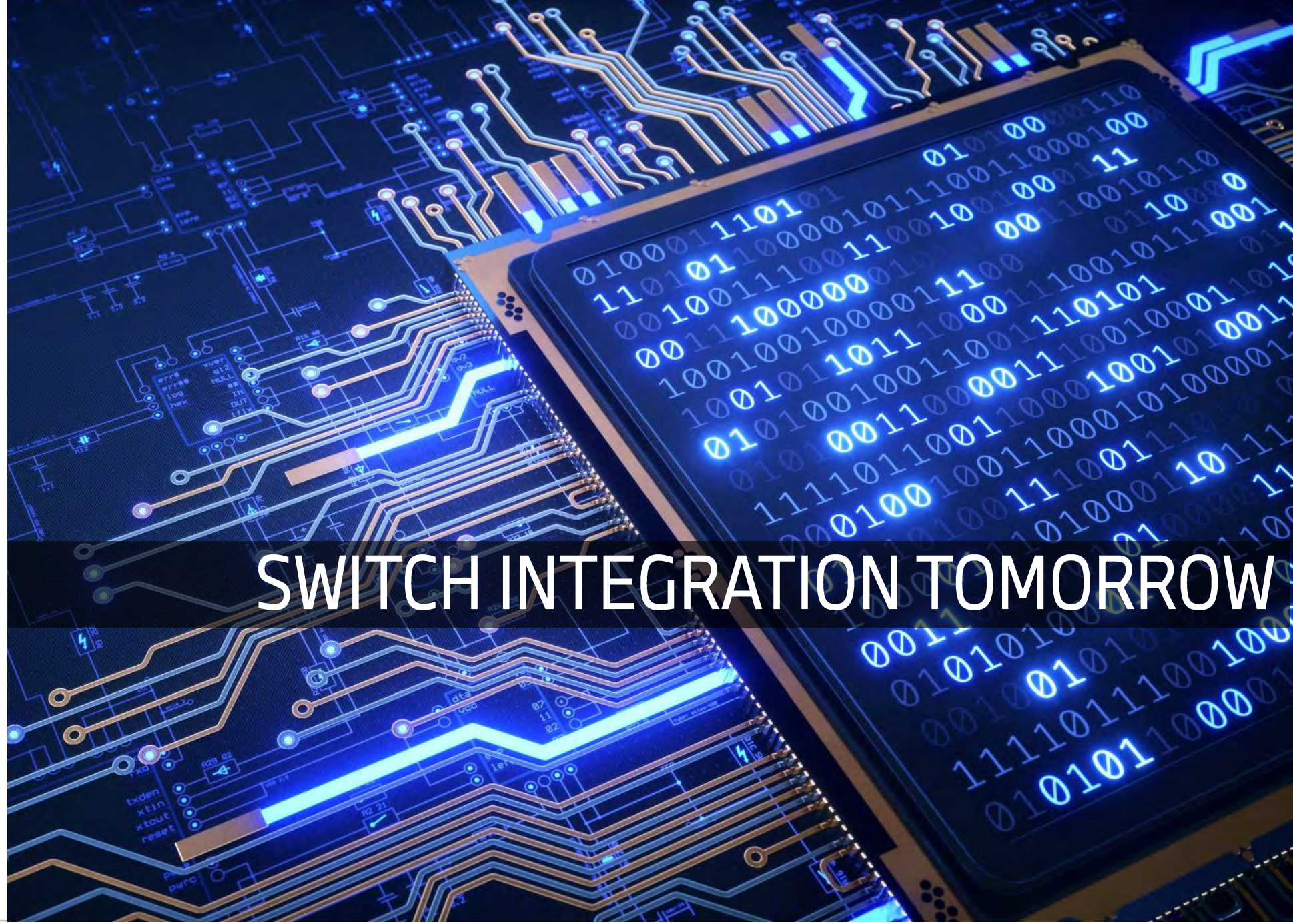
- Stronger design for new feature and architecture scalability
- Reduce **time** and **complexity** in ECU development
- **Flexibility** increased against **country regulation** fluctuation

… lead to:

- Common base / language to become more efficient in a multi-party projects
- We need to establish an ecosystem
- Non OEM specific solution can be profitable for other stakeholder

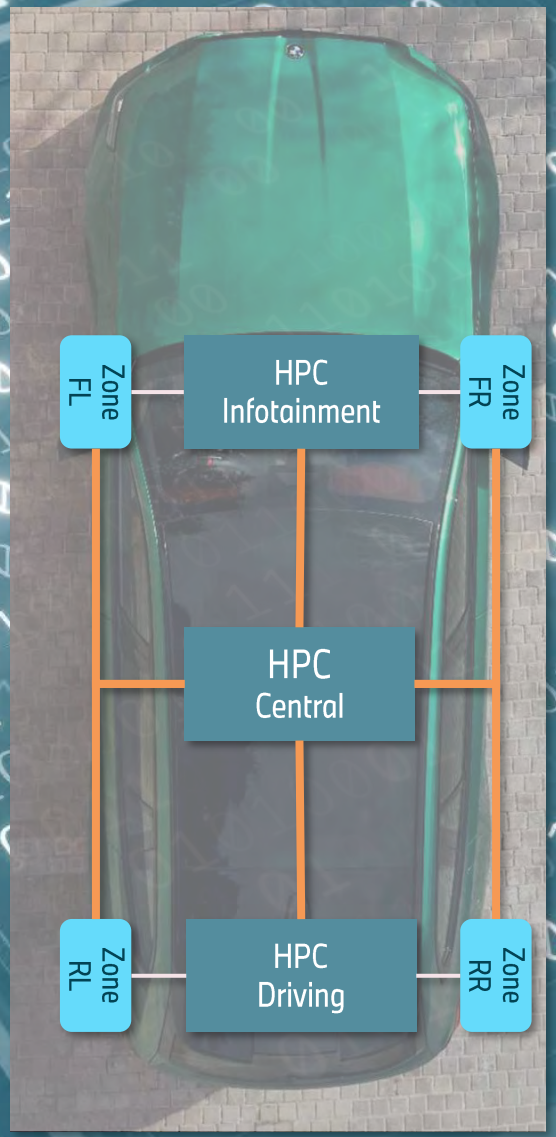
OEM, TIER 1, switch vendors and SW experts see improvement potential



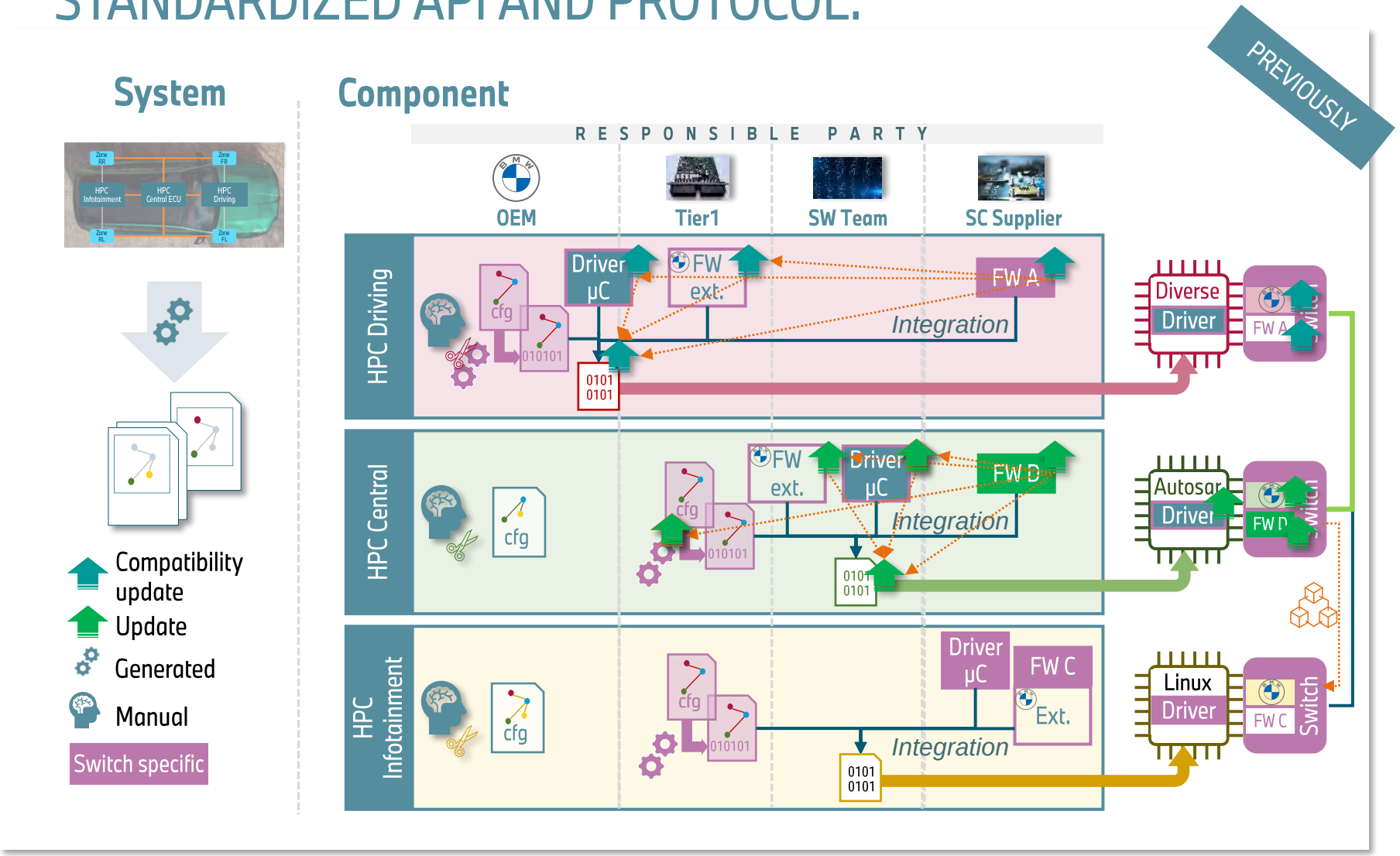


SWITCH INTEGRATION TOMORROW

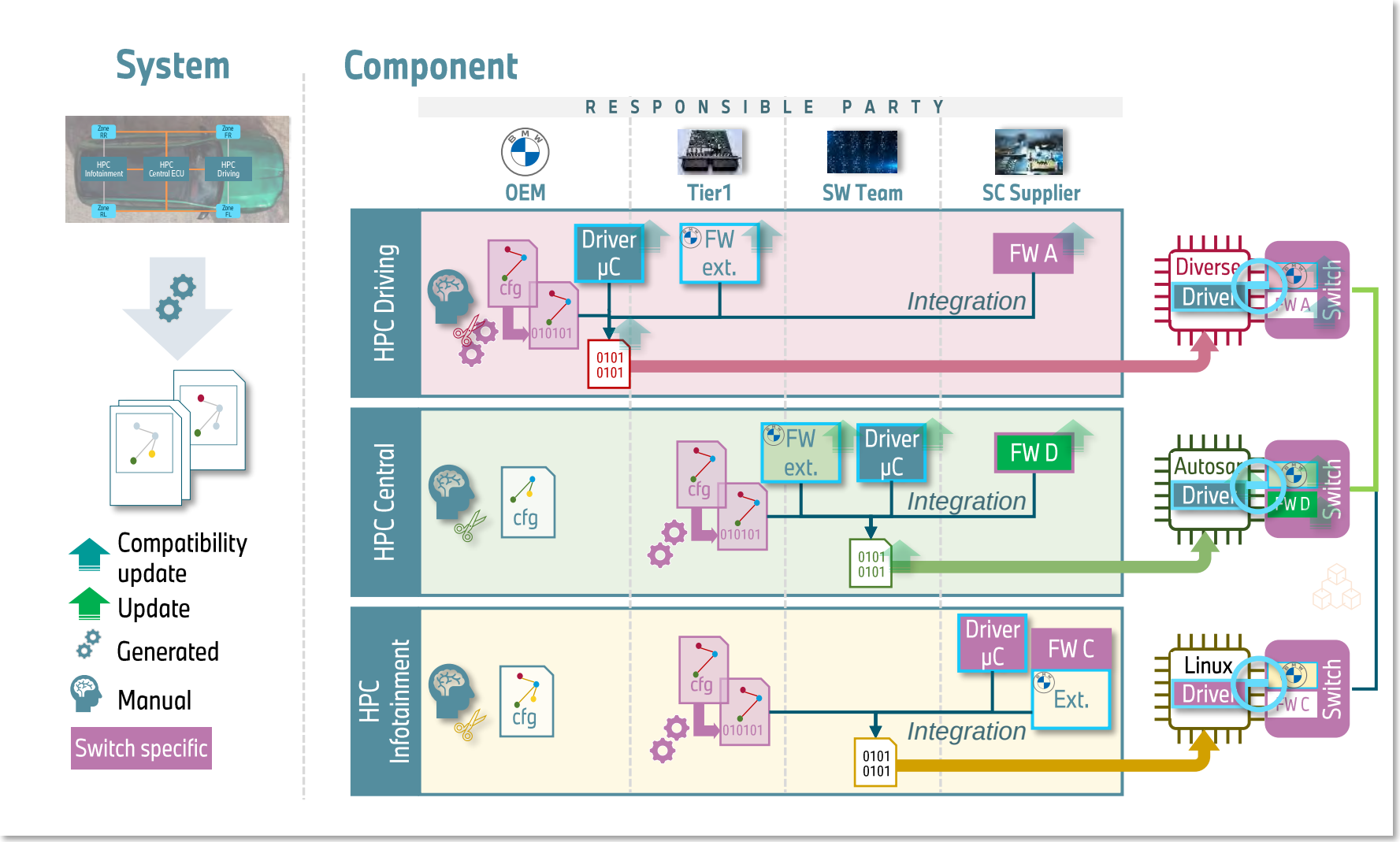
CHANCES OF STANDARDIZATION.
PORTING ON THE CASE STUDY.



EFFICIENCY AND RELIABILITY IN SW APPLIED IN CASE STUDY.
STANDARDIZED API AND PROTOCOL.



EFFICIENCY AND RELIABILITY IN SW APPLIED IN CASE STUDY.
STANDARDIZED API AND PROTOCOL.

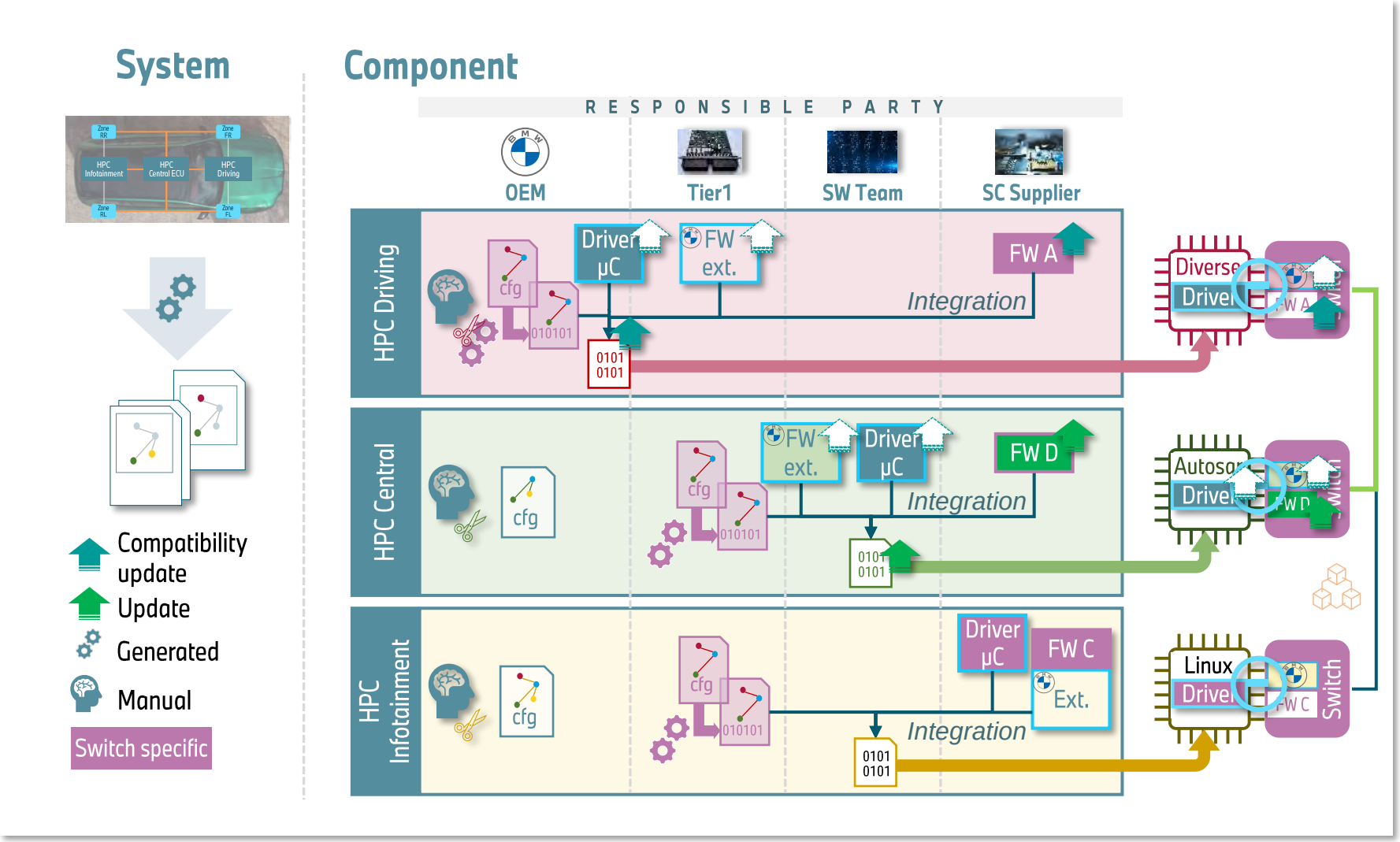


- SW reuse with compatible API on switch
- Host development less impacted

Driver Compatibility is easier to maintain

Universal protocol enables a multiple OS compatibility

EFFICIENCY AND RELIABILITY IN SW APPLIED IN CASE STUDY.
STANDARDIZED API AND PROTOCOL.



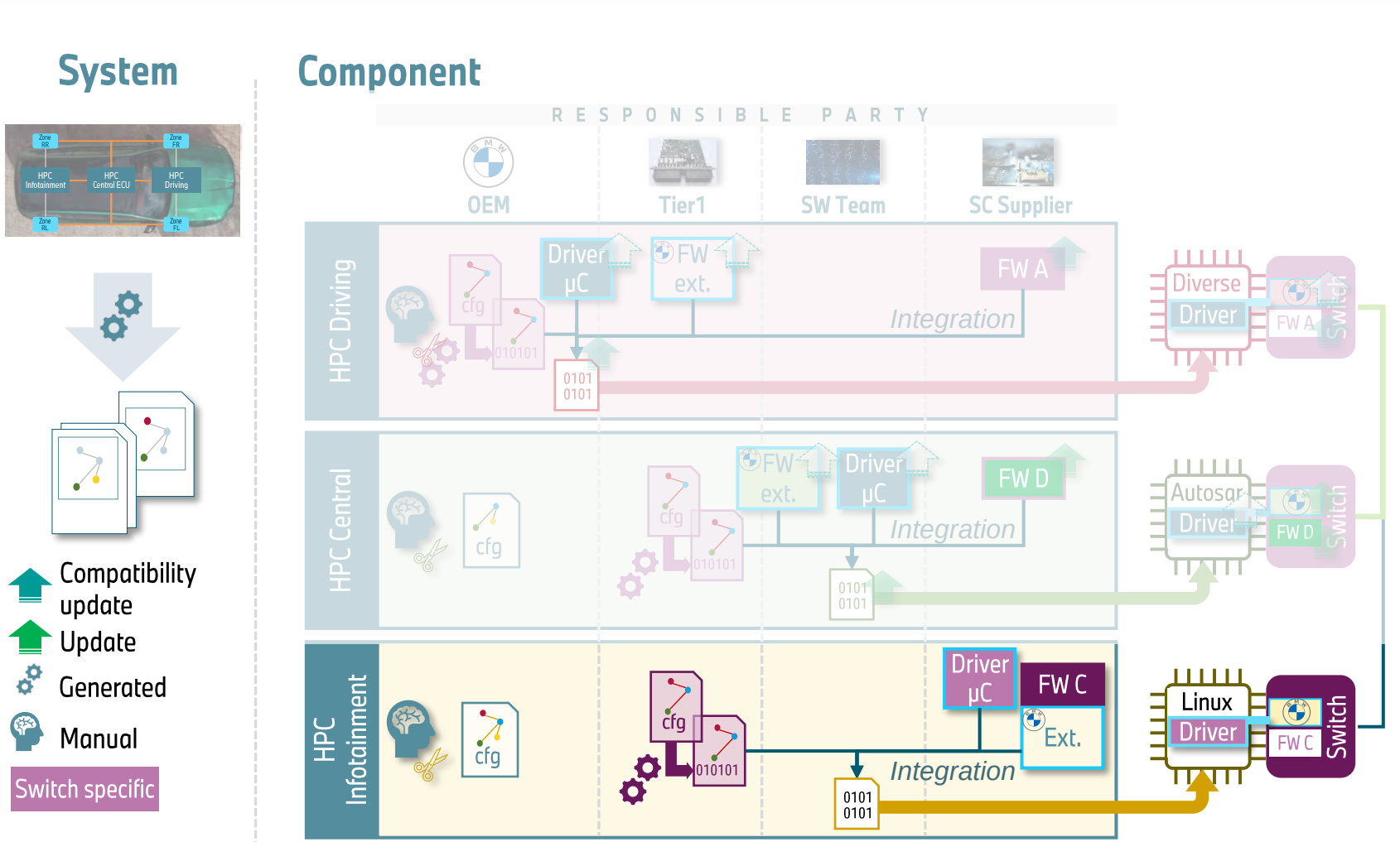
- SW reuse with compatible API on switch
- Host development less impacted

Driver Compatibility is easier to maintain

Universal protocol enables a multiple OS compatibility

Less updates / reintegration loops

EFFICIENCY AND RELIABILITY IN SW APPLIED IN CASE STUDY.
STANDARDIZED API AND PROTOCOL.



- SW reuse with compatible API on switch
- Host development less impacted

Driver Compatibility is easier to maintain

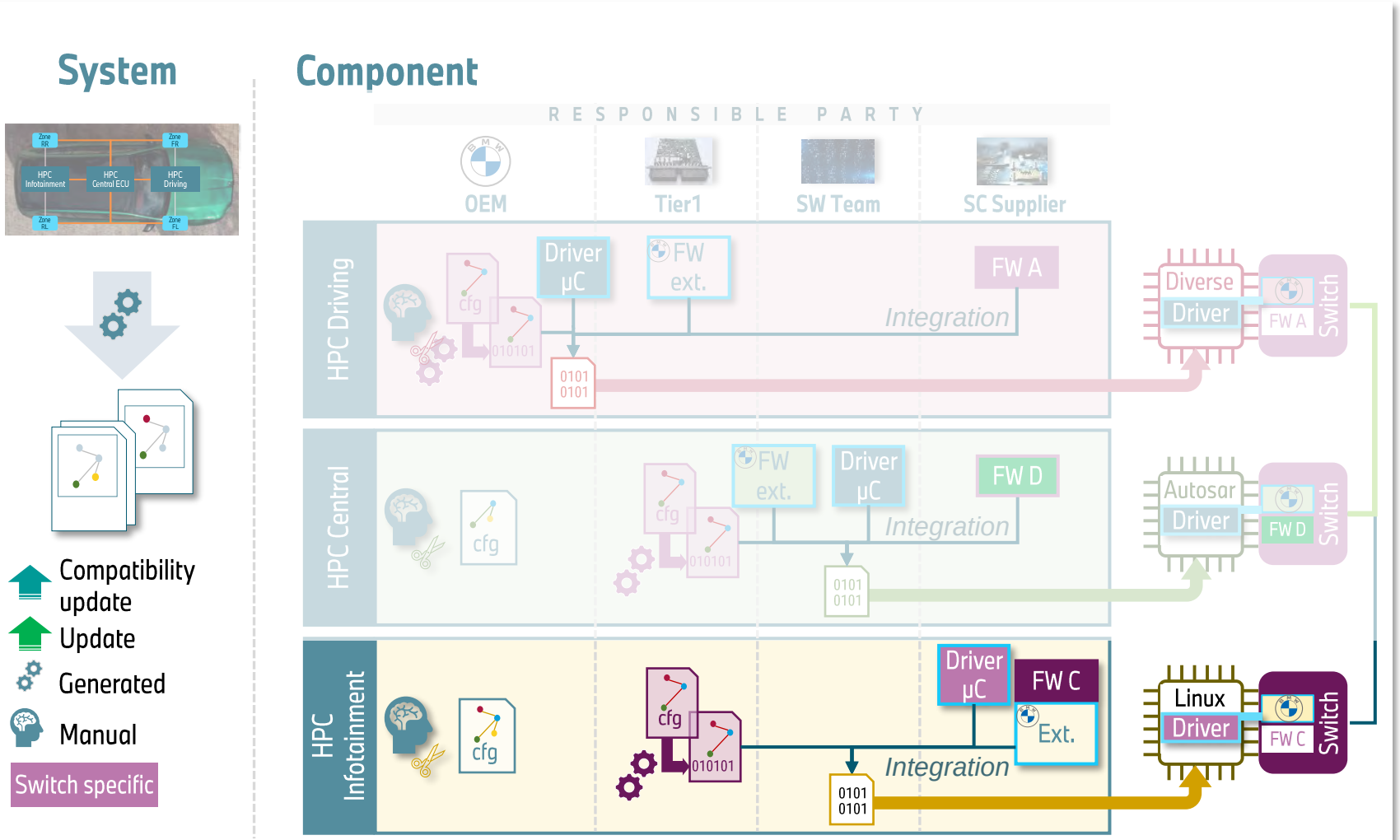
Universal protocol enables a multiple OS compatibility

Less updates / reintegration loops

Switch Compatibility with other switch vendors depending on application or local regulation

- Open Source for all and better maintenance

EFFICIENCY AND RELIABILITY IN SW APPLIED IN CASE STUDY.
STANDARDIZED API AND PROTOCOL.



- SW reuse with compatible API on switch
- Host development less impacted

Driver Compatibility is easier to maintain

Universal protocol enables a multiple OS compatibility

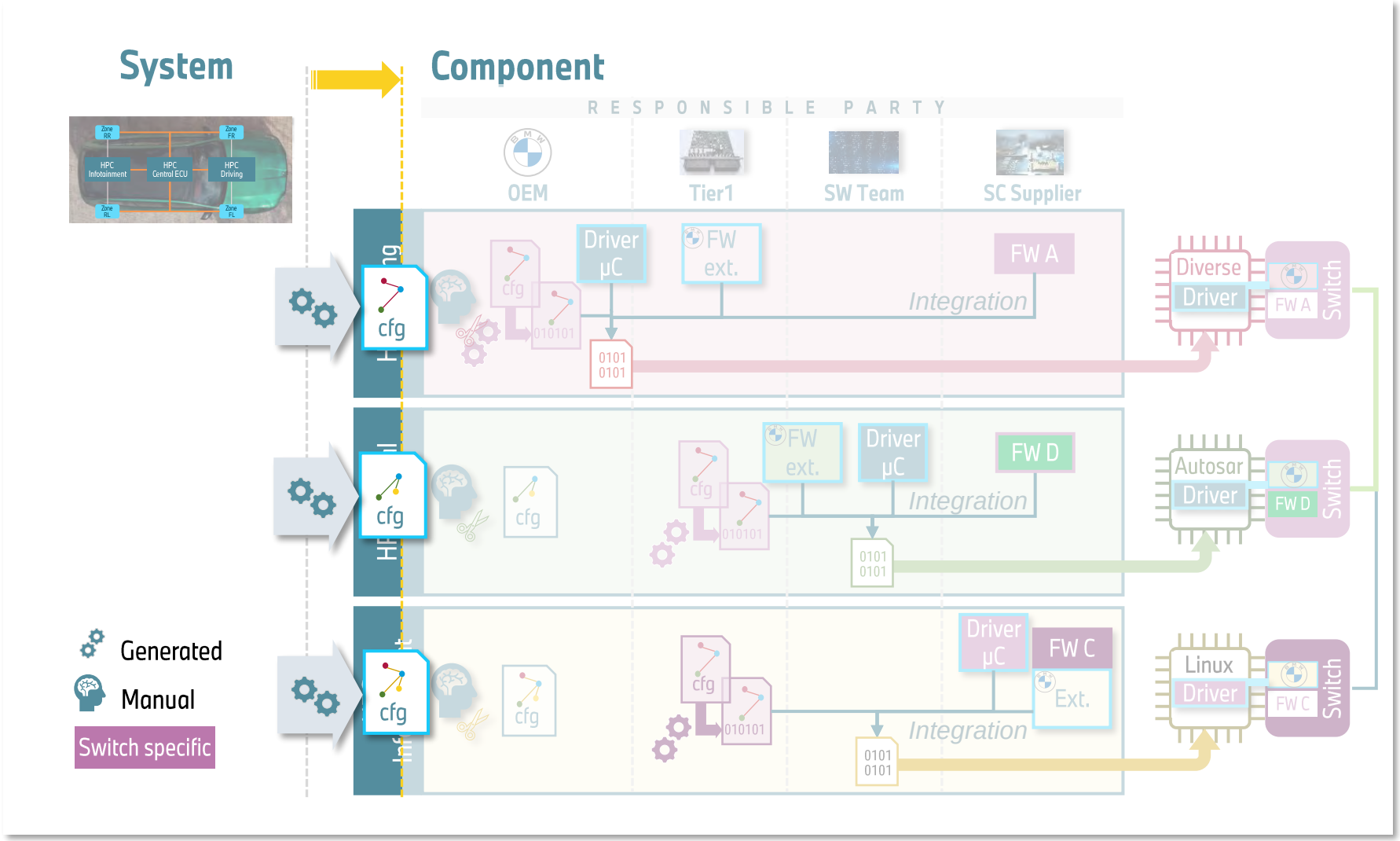
Less updates / reintegration loops

Switch Compatibility with other switch vendors depending on application or local regulation

- Open Source for all and better maintenance

Next step: configuration !

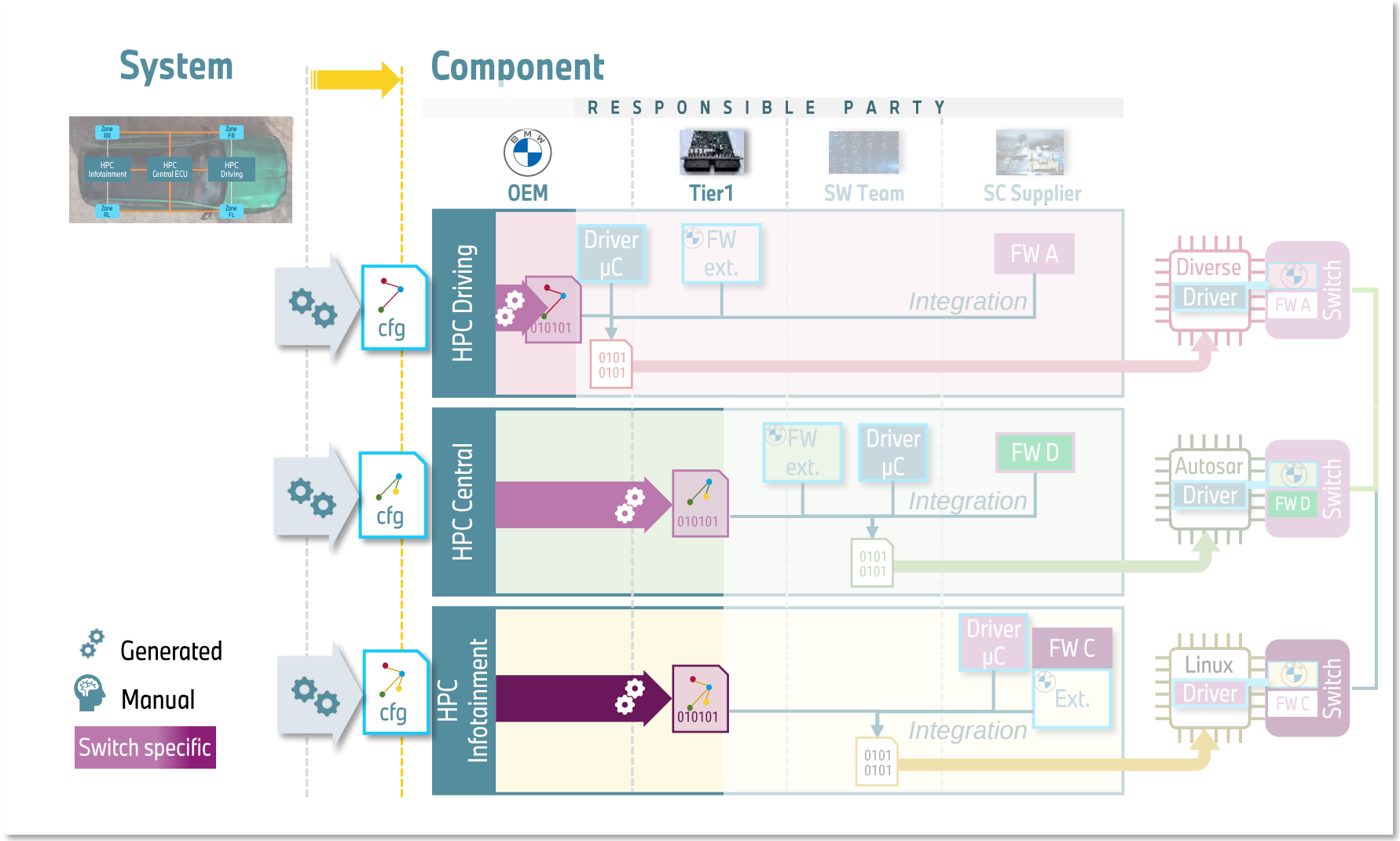
EFFICIENCY AND RELIABILITY IN SW APPLIED IN CASE STUDY.
CONFIGURATION.



More efficient with automated toolchain:

- Configuration generation in standardized format at system level
- Continuous integration at system level

EFFICIENCY AND RELIABILITY IN SW APPLIED IN CASE STUDY.
CONFIGURATION.



- More efficient with automated toolchain:
  - Configuration generation in standardized format at system level (no manual errors)
  - Continuous integration at system level
  - Less effort for HPC team

➢ Enables system abstraction and makes it easy to transfer to other providers (i.e. component reuse for China)

GENERIC INTERFACE AND CONFIGURATION.
DEPENDENCIES HW – SW.

	Security	Flash & Diagnostic		Timing	Networking		Life cycle
SW support	Secured boot	Update firmware	Monitoring	SCT	Networking Filtering	Vlan filt.	Wake-up
	MKA	New config	Configset change	Time validation	L3 Routing	Prioritization	UDP-NM3
	Firewall, Filtering	Interaction protocol	Mirroring	gPTP	TC11, TC8	AVB	Partial Network
HW support	HSM	Flash	SPI /SMI	1588	Packet buffer	QoS Queues	...
	MACSec	Dual banking	MIB	HW clock	Schaper Scheduler	ARL	
	Policing				Ports		
	TCAM				SPI		

→ Non exhaustive list, need to be completed!

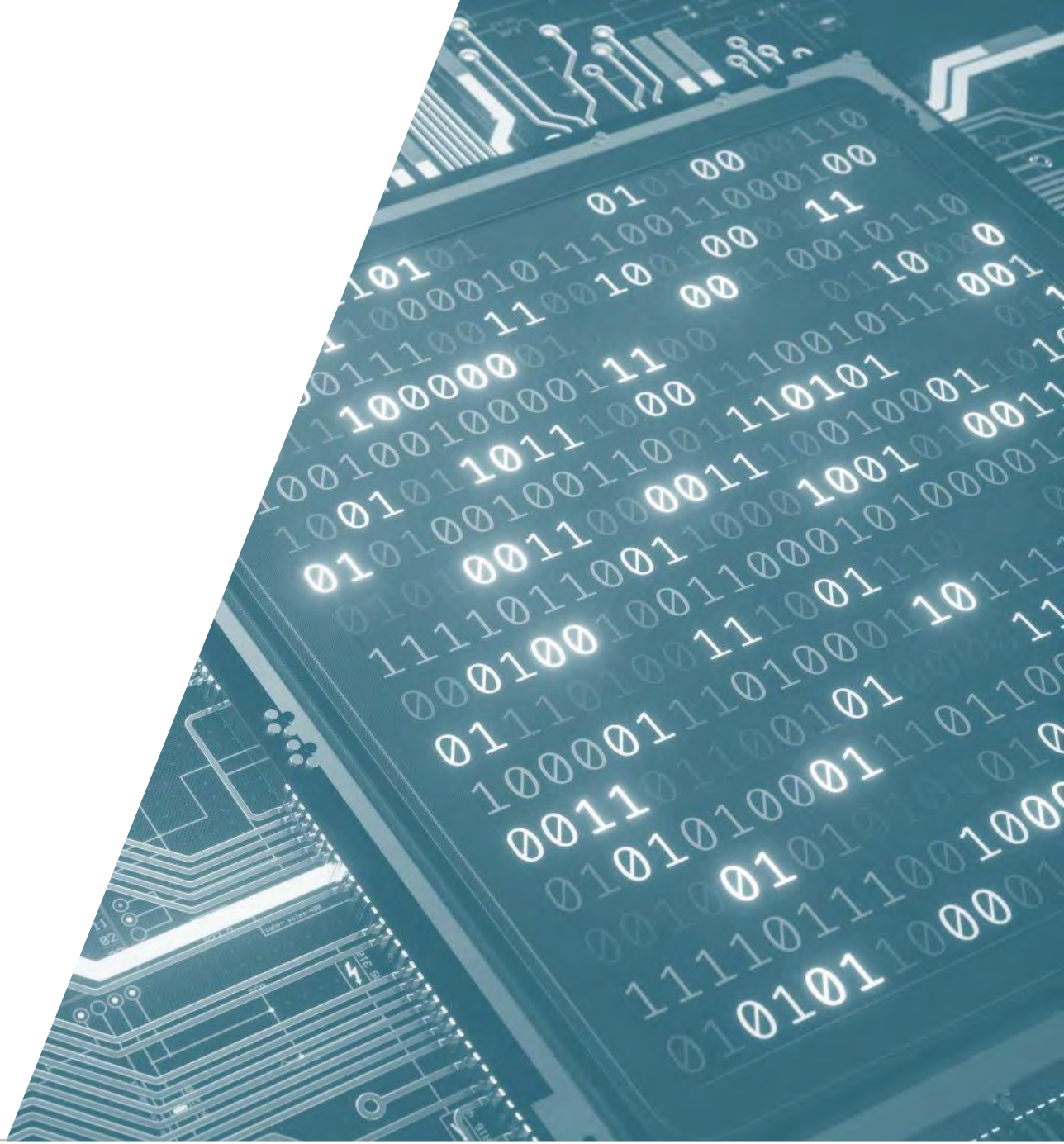
High importance

To-Dos for standardization :

- Define scope and minimal feature set
- Identify relevant part for HW (configuration) and SW (API)
- Identify dependencies between HW and SW

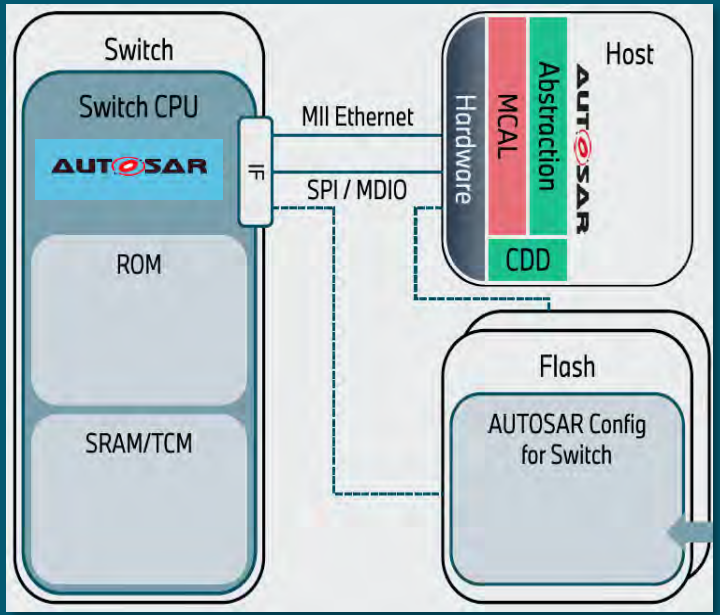
It must be generic to reserve **HW design freedom** but specific enough for **OEM needs !**

OUTLOOKS.



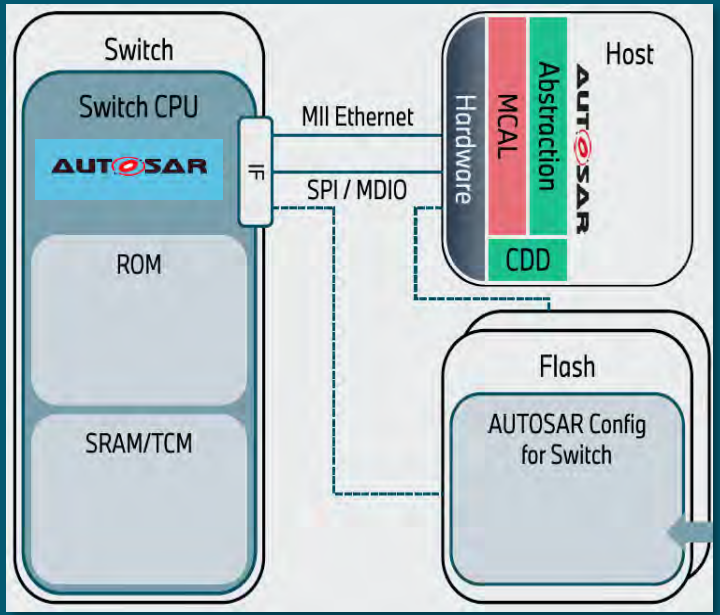
IS AUTOSAR SWITCH THE SOLUTION ?
OTHER APPROACHES AVAILABLE.

- Solves partly the problems for AUTOSAR :
  - Export format equal as host (*.arxml)
  - Integrated toolchain
- Comes with the same limitations from AUTOSAR (cost, complexity, limitations)
- Implements a switch-host RPC protocol: compatibility with different stack vendors and non-AUTOSAR system



IS AUTOSAR SWITCH THE SOLUTION ?
OTHER APPROACHES AVAILABLE.

- Solves partly the problems for AUTOSAR :
  - Export format equal as host (*.arxml)
  - Integrated toolchain
- Comes with the same limitations from AUTOSAR (cost, complexity, limitations)
- Implements a switch-host RPC protocol: compatibility with different stack vendors and non-AUTOSAR system



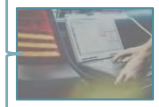
AUTOSAR is **not the solution for non AUTOSAR projects** !

- License cost
- Additional effort to understand this new ecosystem and cost for toolchain

EXISTING MECHANISM IN IT WORLD.
OUTLOOK.

Existing solutions in the IT world [vs. Automotive]

- Mechanism: SNMP, NETCONF, RESTCONF [DoIP, Prop.]
- Protocols: SNMP, RCP over SSH, HTTP [SOME/IP, Prop.]
  - ➢ Monitoring vs. management
- Configuration model: MIB, YANG [AUTOSAR, MIB support]
- Format: XML, JSON [AUTOSAR, MIB support]



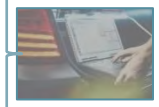
Available none automotive standards

- SNMP: IETF RFC1157, RFC3411, RFC3413, RFC3414
- RESTCONF: RFC 8040
- NETCONF: RFC6241, RFC5277, RFC5539, RFC6242
- YANG: RFC6020, RFC6021, RC6022

EXISTING MECHANISM IN IT WORLD.
OUTLOOK.

Existing solutions in the IT world [vs. Automotive]

- Mechanism: SNMP, NETCONF, RESTCONF [DoIP, Prop.]
- Protocols: SNMP, RCP over SSH, HTTP [SOME/IP, Prop.]
  - ➢ Monitoring vs. management
- Configuration model: MIB, YANG [AUTOSAR, MIB support]
- Format: XML, JSON [AUTOSAR, MIB support]



Available none automotive standards

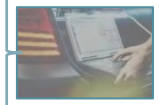
- SNMP: IETF RFC1157, RFC3411, RFC3413, RFC3414
- RESTCONF: RFC 8040
- NETCONF: RFC6241, RFC5277, RFC5539, RFC6242
- YANG: RFC6020, RFC6021, RC6022

- Issue solved ?
- Why do solutions for other industries not work in Automotive?

EXISTING MECHANISM IN IT WORLD.
OUTLOOK.

Existing solutions in the IT world [vs. Automotive]

- Mechanism: SNMP, NETCONF, RESTCONF [DoIP, Prop.]
- Protocols: SNMP, RCP over SSH, HTTP [SOME/IP, Prop.]
  - ➢ Monitoring vs. management
- Configuration model: MIB, YANG [AUTOSAR, MIB support]
- Format: XML, JSON [AUTOSAR, MIB support]



Available none automotive standards

- SNMP: IETF RFC1157, RFC3411, RFC3413, RFC3414
- RESTCONF: RFC 8040
- NETCONF: RFC6241, RFC5277, RFC5539, RFC6242
- YANG: RFC6020, RFC6021, RC6022

- Issue solved ?
- Why do solutions for other industries not work in Automotive?

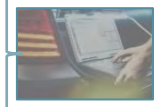
Pro / cons

- Maturity
- Resources, embedded solution, complexity
- Capability for extended management
- Security

EXISTING MECHANISM IN IT WORLD.
OUTLOOK.

Existing solutions in the IT world [vs. Automotive]

- Mechanism: SNMP, NETCONF, RESTCONF [DoIP, Prop.]
- Protocols: SNMP, RCP over SSH, HTTP [SOME/IP, Prop.]
  - ➢ Monitoring vs. management
- Configuration model: MIB, YANG [AUTOSAR, MIB support]
- Format: XML, JSON [AUTOSAR, MIB support]



Available none automotive standards

- SNMP: IETF RFC1157, RFC3411, RFC3413, RFC3414
- RESTCONF: RFC 8040
- NETCONF: RFC6241, RFC5277, RFC5539, RFC6242
- YANG: RFC6020, RFC6021, RC6022

- Issue solved ?
- Why do solutions for other industries not work in Automotive?

Pro / cons

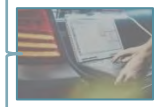
- Maturity
- Resources, embedded solution, complexity
- Capability for extended management
- Security

What can be adapted for Automotive ? **SOME/IP**

EXISTING MECHANISM IN IT WORLD.
OUTLOOK.

Existing solutions in the IT world [vs. Automotive]

- Mechanism: SNMP, NETCONF, RESTCONF [DoIP, Prop.]
- Protocols: SNMP, RCP over SSH, HTTP [SOME/IP, Prop.]
  - ➢ Monitoring vs. management
- Configuration model: MIB, YANG [AUTOSAR, MIB support]
- Format: XML, JSON [AUTOSAR, MIB support]



Available none automotive standards

- SNMP: IETF RFC1157, RFC3411, RFC3413, RFC3414
- RESTCONF: RFC 8040
- NETCONF: RFC6241, RFC5277, RFC5539, RFC6242
- YANG: RFC6020, RFC6021, RC6022

- Issue solved ?
- Why not already switch vendors selling product in other sectors have not introduced theses mechanisms in automotive ?

Pro / cons

- Maturity
- Resources, embedded solution, complexity
- Capability for extended management
- Security

What can be adapted for Automotive ? **SOME/IP**

→ „Automotive“ profile ?

SWITCHES ARE ALREADY COVERED BY MANY STANDARDS.
AVAILABLE AUTOMOTIVE SPECIFICATIONS.

Complex system with different usage

Implementation is not specified due to high OS diversity

Common exchange format and management needed

AUTOSAR

System_Template

SWS_EthernetSwitchDriver

AVNU

TSN Avnu Automotive Profile

OPEN ALLIANCE

TC11:

- Switch Requirements
- Test Specification Switch

TC17:

- MACSec

IEEE

IEEE 802.1D – MAC Bridge

IEEE 802.1X – Port-based Net. Access Ctrl

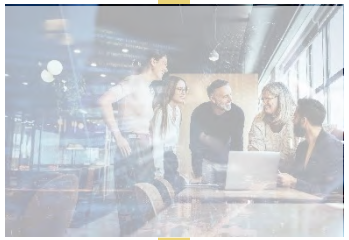
IEEE 802.1AE - MACsec

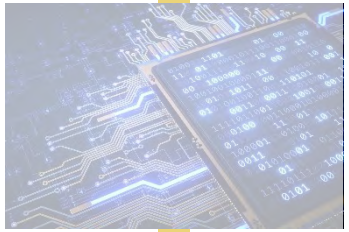
IEEE 802.1Q – VLAN

IEEE 1588 – PTP 1 step transparent clock

IEEE 802.1AS

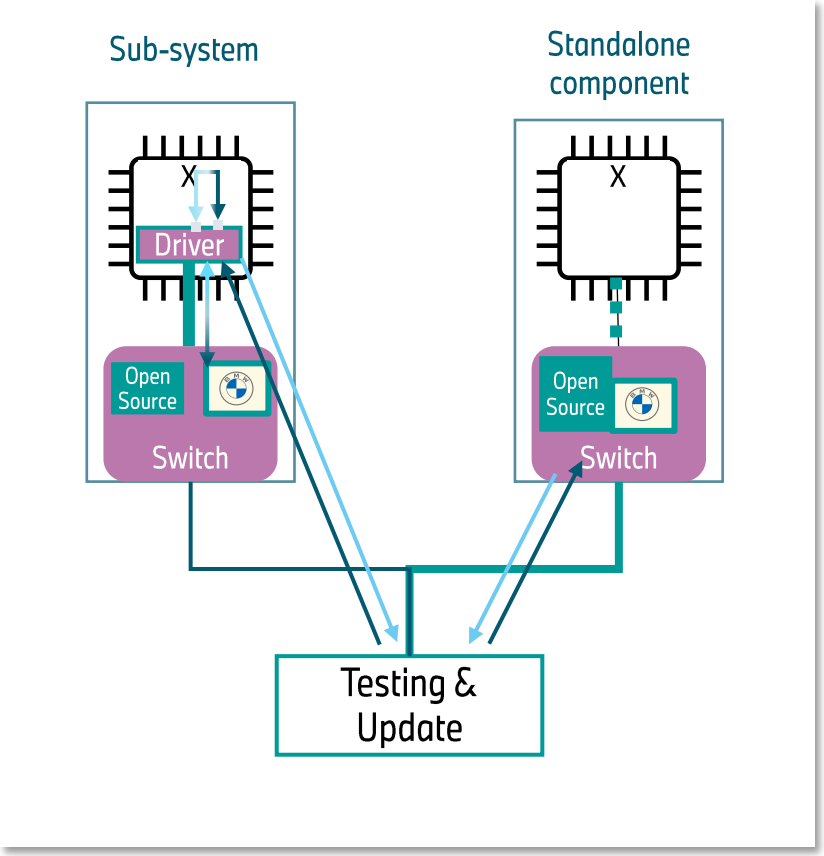
IEEE1722



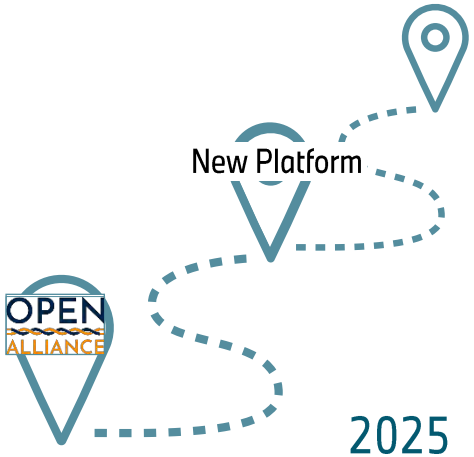


MISSION NORTH STAR, OUR OBJECTIVES

MISSION NORTH STAR, OUR OBJECTIVES
FINAL SOLUTION STILL TO BE DEFINED BUT **CLEAR GOALS**.



1. Switch remains a subsystem or becomes an independent system component located within a HPC
2. Generic interface and configuration are the first steps we have identified
3. Open-source solution is attractive and there is still work to be done



2030+ Clear picture about the position of the switch within the system

2029 Generic interface and configuration
Open-source project

2025 **Prototype** and begin/expand a TC within **Open Alliance**

TAKEAWAYS.

1. SW API and configuration standardization is a necessity to solve current and future struggles

2. The solution cannot be OEM specific

3. Participation of all concerned parties is valuable to draw the solution

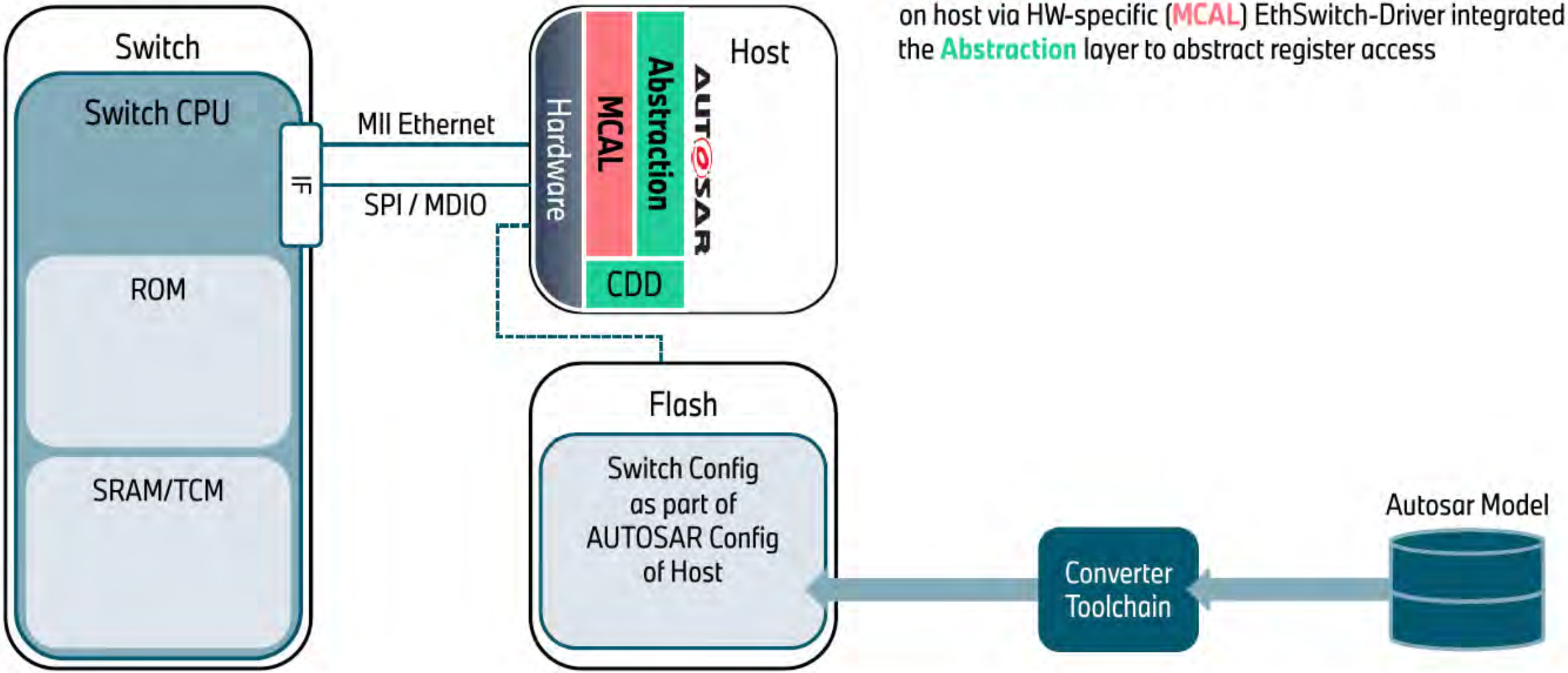
4. Is there within IEEE a similar working group or a possible liaison with Open Alliance ?

THANK YOU FOR YOUR ATTENTION

Backup

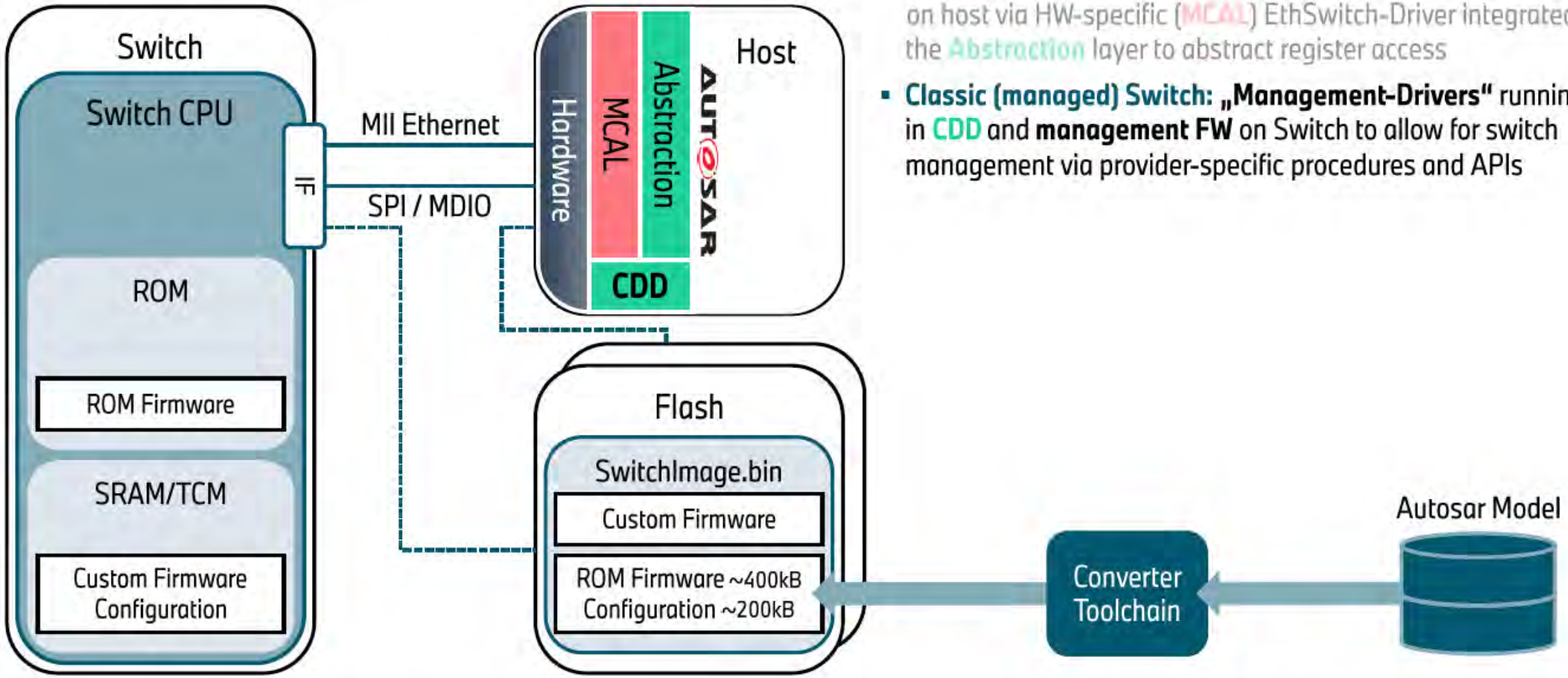
AN OVERVIEW OF DIFFERENT SWITCH CONFIGURATION APPROACHES. SWITCH MANAGEMENT, RE-CONFIGURATION & FLASH UPDATES.

- **Unmanaged Switch:** AUTOSAR-based switch management on host via HW-specific (MCAL) EthSwitch-Driver integrated in the Abstraction layer to abstract register access

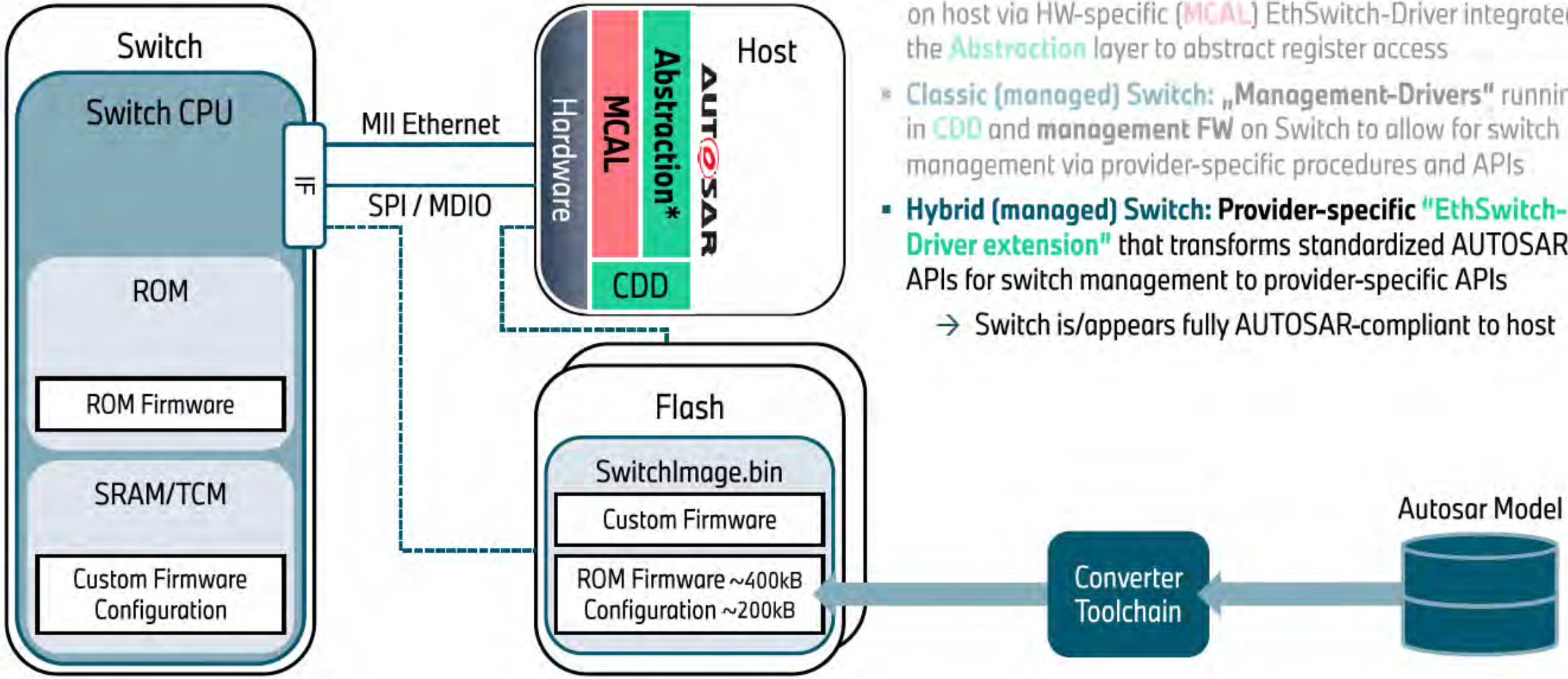


AN OVERVIEW OF DIFFERENT SWITCH CONFIGURATION APPROACHES. SWITCH MANAGEMENT, RE-CONFIGURATION & FLASH UPDATES.

- **Unmanaged Switch:** AUTOSAR-based switch management on host via HW-specific (MCAL) EthSwitch-Driver integrated in the Abstraction layer to abstract register access
- **Classic (managed) Switch: „Management-Drivers"** running in CDD and **management FW** on Switch to allow for switch management via provider-specific procedures and APIs

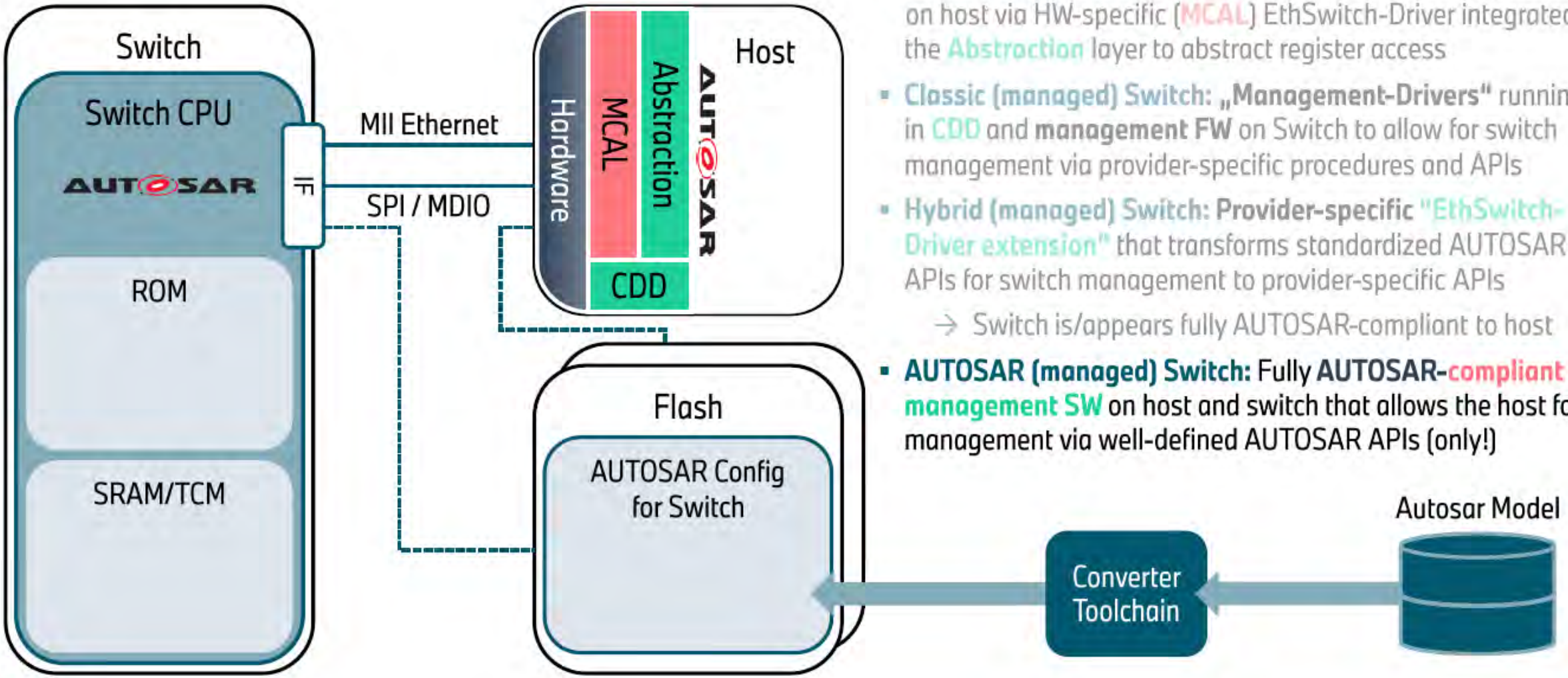


AN OVERVIEW OF DIFFERENT SWITCH CONFIGURATION APPROACHES. SWITCH MANAGEMENT, RE-CONFIGURATION & FLASH UPDATES.



- **Unmanaged Switch:** AUTOSAR-based switch management on host via HW-specific (MCAL) EthSwitch-Driver integrated in the Abstraction layer to abstract register access
- **Classic (managed) Switch:** **„Management-Drivers"** running in CDD and **management FW** on Switch to allow for switch management via provider-specific procedures and APIs
- **Hybrid (managed) Switch: Provider-specific "EthSwitch-Driver extension"** that transforms standardized AUTOSAR APIs for switch management to provider-specific APIs
  → Switch is/appears fully AUTOSAR-compliant to host

AN OVERVIEW OF DIFFERENT SWITCH CONFIGURATION CONCEPTS. SWITCH MANAGEMENT, RE-CONFIGURATION & FLASH UPDATES.



- **Unmanaged Switch:** AUTOSAR-based switch management on host via HW-specific (MCAL) EthSwitch-Driver integrated in the Abstraction layer to abstract register access
- **Classic (managed) Switch:** **„Management-Drivers"** running in CDD and **management FW** on Switch to allow for switch management via provider-specific procedures and APIs
- **Hybrid (managed) Switch:** **Provider-specific** "EthSwitch-Driver extension" that transforms standardized AUTOSAR APIs for switch management to provider-specific APIs
  → Switch is/appears fully AUTOSAR-compliant to host
- **AUTOSAR (managed) Switch:** Fully **AUTOSAR-compliant management SW** on host and switch that allows the host for management via well-defined AUTOSAR APIs (only!)