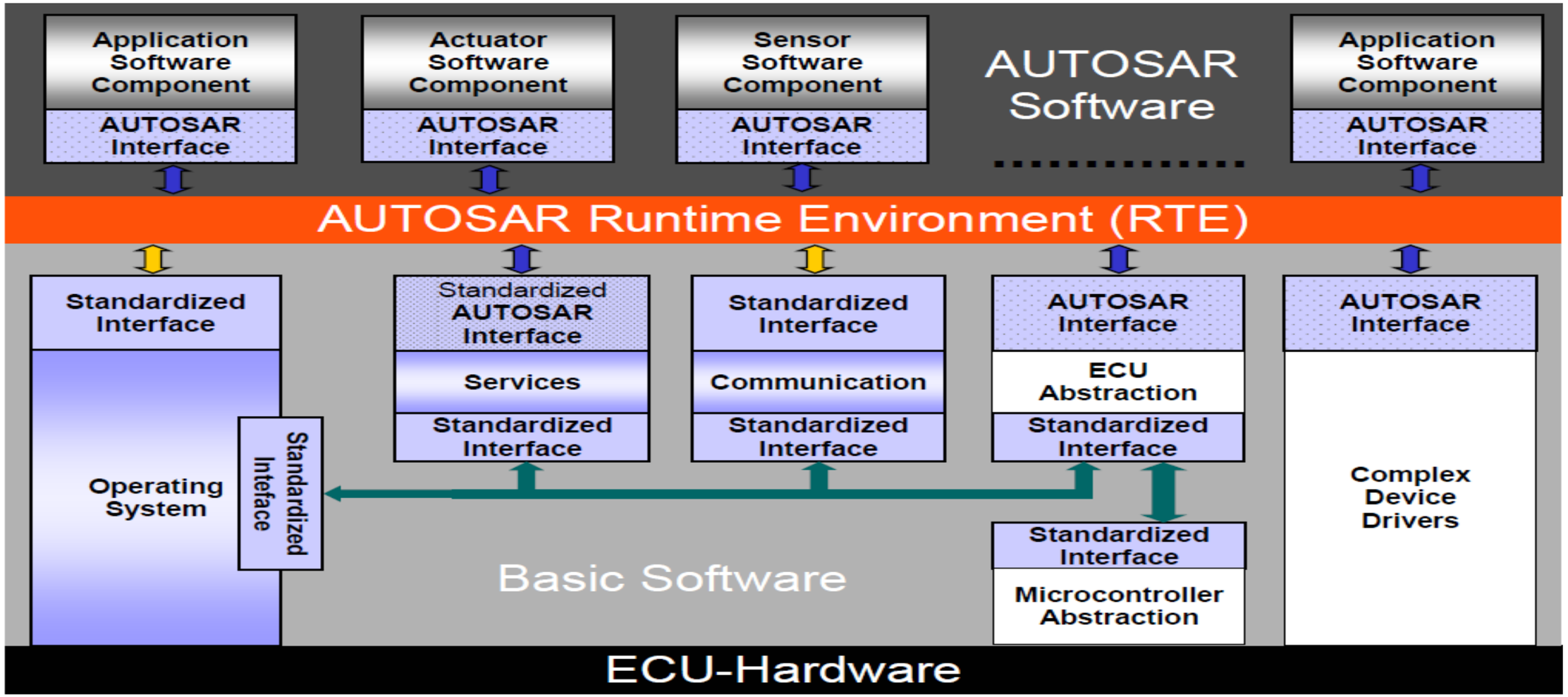
AUTOSAR OS

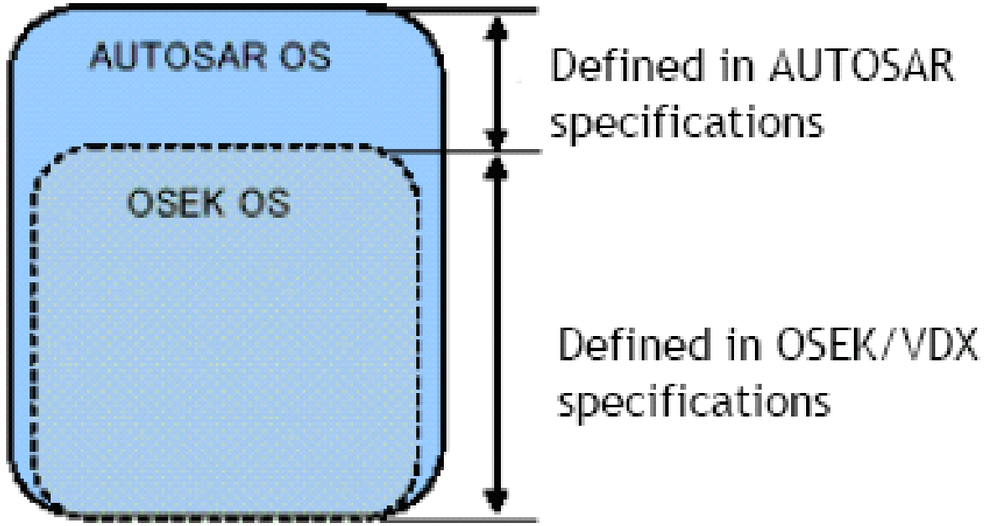
KPIT

Introduction to AUTOSAR OS



System Overview of AUTOSAR Architecture

AUTOSAR OS



- Based on OSEK operating system
- AUTOSAR Os (like OSEK) is configured statically
- Provides a mixed kind of scheduling mechanism
- Provides protective functions at run time

The AUTOSAR OS includes all the functions of OSEK OS and basically has upward compatibility.

Introduction of AUTOSAR OS

Available in Scalability Classes 1, 2, 3, 4 to fit the needs of different applications

- SC1 –Deterministic RTOS baseline (tasks, events, counters, alarms, resources, schedule table)
- SC2 –Timing based task determinism (low-latency, precise timing for periodic tasks, global time synchronization) & Synchronization of Schedule table (used for FlexRay)
- SC3 –Protected memory for tasks avoids memory collisions for safety systems
- SC4 –Timing and memory protected tasks, utilizes the full capabilities of the silicon for secure and protected RTOS designed specifically for the automotive

AUTOSAR OS: Scalability class details

Feature	SC1	SC2	SC3	SC4	Hardware Requirements
OSEK OS (All Conformance Classes)	●	●	●	●	
Counter Interface	●	●	●	●	
Schedule Tables	●	●	●	●	
Stack Monitoring	●	●	●	●	
Protection Hook		●	●	●	
Timing Protection		●		●	Timers/Traps for Timing protection
Global Time / Synchronization support		●		●	Global Time Source
Memory Protection			●	●	MPU/MMU
OS-Applications	●	●	●	●	
Service Protection			●	●	
Call TrustedFunction			●	●	Support for User modes

OSEK OS Features

- **OSEK/VDX™ OS :**

- Core Features
  - TASK : Task provides the framework for execution of the function
  - Events : Mechanism for concurrent processing synchronization
  - Resource : Mechanism for mutual concurrent access exclusion
  - Counter & Alarms : Mechanisms for processing recurring events based on timer interrupt
  - Interrupt Handler : Mechanism for processing asynchronous events
  - Fixed Priority Scheduling
  - Statically configured application

AUTOSAR OS Features

AUTOSAR OS is OSEK/VDX™OS plus:

New core features

- Software Counter : Mechanisms for processing recurring events based on external events
- Stack Monitoring : Mechanism for monitoring the stack of Task or ISR2
- Schedule tables with global time synchronization : Mechanism for processing recurring event based on hardware event or external event with global time synchronization

Protection features

- Timing Protection, Memory Protection and Service protection
- OS applications, trusted and non-trusted OS Application

AUTOSAR OS Core Feature

❖ **TASK:**

```
TASK EntryPoint( void )
{
...
}
```

❖ It provides the framework for the execution of the function
- ❑ Kinds of Task : BASIC TASK and EXTENDED TASK
- ❑ Provides the following services

Service Name	**Description**
Activate Task	Activate the Task i.e. put it from the suspended in to ready state
Terminate Task	Terminate the Task i.e. put it from the ready in to suspended state
Chain Task	Terminate the Task and activates a new one immediately
Schedule	Yields control to a higher-priority ready task (If any exits)
GetTaskId	Gets the identifier of the running task
GetTaskState	Gets the status of the specified task

❖ Task has following configuration parameters :
Stack, Priority, Pre-emptive, MultipleActivatation, etc..

AUTOSAR OS Core Feature

- **Event:**
- Tasks can be synchronized via an Event in OS. Examples are: the signaling of a timer's expiry, the availability of a resource, the receipt of a message, etc
  - OS provides the following services which can be called for extended task

Service Name	Description
SetEvent	Sets events of the given task according to the event mask
ClearEvent	Clears events of the calling task according to the event mask
GetEvent	Gets the current event setting of the given mask
WaitEvent	Transfers the calling task into the waiting state until specified events are set

```
TASK (TASK1)
  {
      while(1) {
      WaitEvent(ONMESSAGE_100);  P10 = !(P10);}
}
/* SetEvent services will be called on the reception of a message 0x100 */
```


AUTOSAR OS Core Feature

❖ **Counters and Alarms:**

❖ The OS comprises a two level concept to make use of recurring events like periodic interrupt of timers, interrupt of the sensors on rotating angles, or any recurring software events.

Service Name	Description
SetRelAlarm	Sets alarm at a relative offset
SetAbsAlarm	SetAlarm at a absolute offset
GetAlarm	Returns the remaining alarm ticks before the alarm expiry
GetAlarmBase	Returns the alarm info (MaxallowedValue, MinCycle, TickesPerBase)
IncrementCounter	Increments the software counter tick

❖ Examples of possible using of alarms are:

- "Activate a certain task, after the counter has been advanced 60 times"
- "Set a certain event, after the counter has reached a value of 90".
- "Invoke a function, after the counter has reached a value of 100".
- "Increment the software counter". Generally used to create RTC

AUTOSAR OS Core Feature

- **Interrupt Processing:**

- Two types of interrupt : CAT1 and CAT2 Interrupt
  - Os services are only allowed to call in CAT2 Interrupt. Os provides the frame work to execute the ISR handler. This frame work provides the mechanism to decouple the ISR to task which can be scheduled based on the priority. However, this method will have slight overhead than CAT1 interrupt.

AUTOSAR OS Core Feature

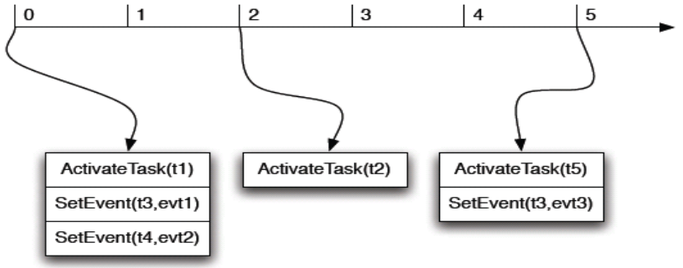
- **Software Counter:**

- Mechanism to increment counter by one tick upon call of IncrementCounter service based on external event

- Advantages
  - This service can be used to activate task / set the event depending on the external interrupt

AUTOSAR OS Core Feature

❖**Schedule Table and Global Synchronization:**



❖Mechanism to process recurring events

Service Name	Description
StartScheduleTableRel	Start schedule table with relative offset
StartScheduleTableAbs	Start schedule table with absolute offset
StopScheduleTable	Stops the running schedule table
NextScheduleTable	Switched the processing from one schedule table to other
GetScheduleTableStatus	Returns the schedule table status

❖Advantages

- ❖Can have multiple expiry points and action on each expiry point is statically configured
- ❖Significant reduction in usage of multiple Alarms i.e only one alarm
- ❖Optimize use of Design patterns
- ❖Easy to implement and less error prone
- ❖Provides the facility to synchronize the operation to adjust local time drift (FlexRay bus)

AUTOSAR OS Core Feature

❖**Stack Monitoring:**

❖AUTOSAR OS provides the facility to monitor the stack

❖Advantages

- Stack overflow in Task/ISR2 can be detected during development stage
- Depending on the stack usage, memory can be allocated in configuration. Hence, it saves the consumption of RAM

AUTOSAR OS Core Feature

❖ **Timing Protection:**

❖ AUTOSAR OS provides the mechanism to monitor task/ISR execution time, arrival and termination rate , resource lock time and interrupt suppression time

❖ Advantages

- ➢ Predictable behavior of OS in case of design errors e.g Unexpected infinite loop, interrupt suppression time
- ➢ Predictable behavior of OS in case of Hardware failure e.g. interrupt rate higher than the expected one
- ➢ Prevention of failure of ECU and lastly failure of the vehicle itself

AUTOSAR OS Core Feature

- **Memory Protection:**

- AUTOSAR OS provides the facility to forbid an access of memory by unauthorized task/ISR (data, instruction, stack)

- Advantages
  - Pointer to memory zone is protected from unauthorized access
  - This prevents the memory corruption

AUTOSAR OS Core Feature

❖ **Service Protection:**

❖ The AUTOSAR OS prevents the propagation of faulty behavior in kernel whenever input parameter or call level error occurs

❖ Advantages

- ➢ Detect fault at early stage
- ➢ Less time for debugging

MULTICORE OS Core Feature

- An extension to AUTOSAR OS
- Shares same configuration & most of the code
- Operates on isolated data structures of entity
- Independent scheduling on each core
- Locatable entity (OsApplication)
- Allocation of entity to a core is static and not dynamic

MULTICORE OS – Functionalities

- **Extended functionalities**
  - Task
  - Event
  - Alarm
  - Counter
  - ScheduleTable
- **Functionalities not supported**
  - Resource
- **Functionalities Added**
  - Spinlock
  - IOC (Inter Os application Communicator)

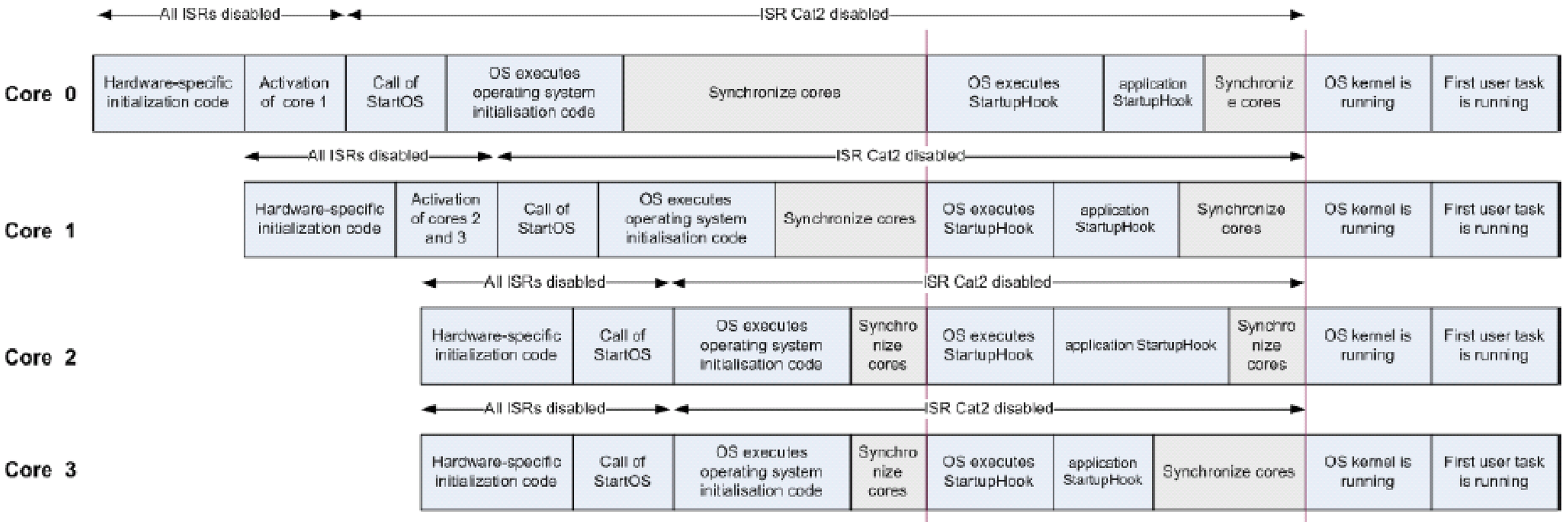
MULTICORE OS – Configuration

- Generation of Os derived from single configuration
- Object Ids should be unique across cores
- Os Application should be configured independent of scalability class
- Core assignment to all objects is done by application

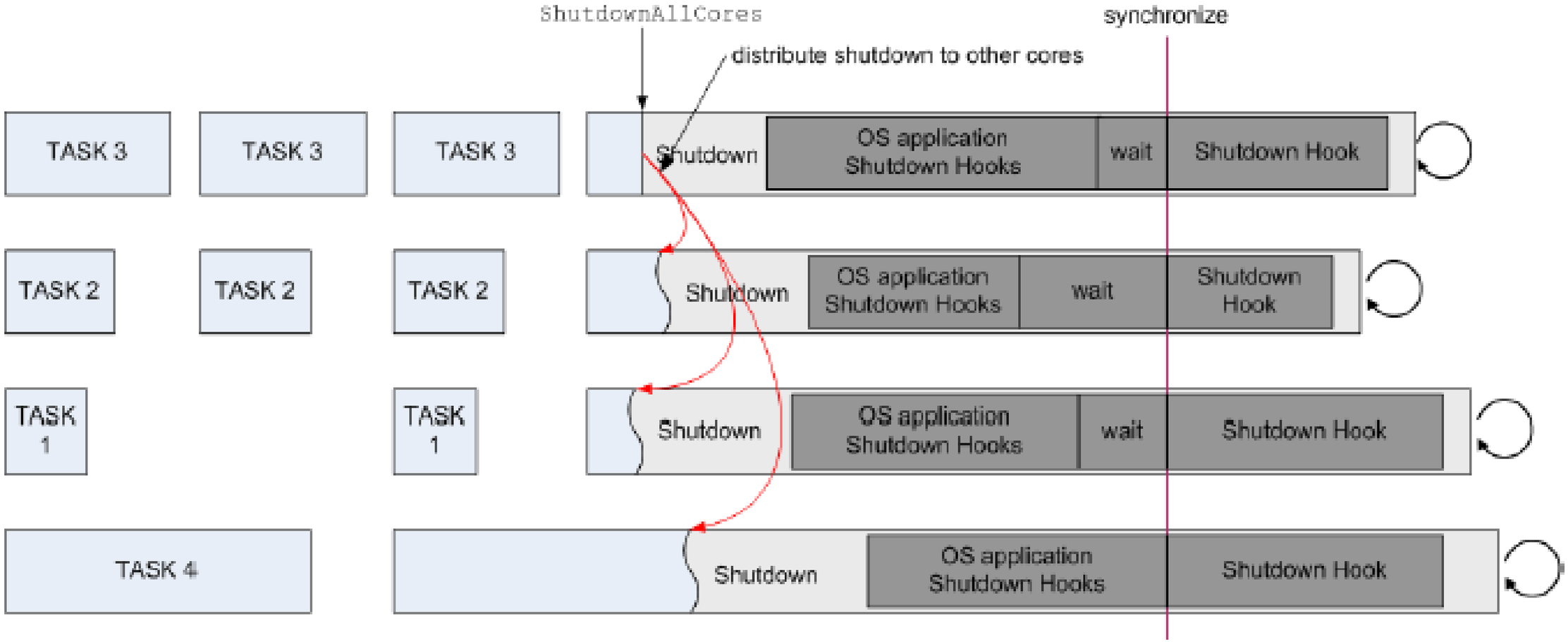
MULTICORE OS – Startup & Shutdown

- **Startup**
  - Cores started in Master Slave concept
  - Startcore called on each core for core specific initialization
  - Single Mode should be used for startup
  - All cores should be synchronised before starting scheduling

MULTICORE OS – Start Up Synchronization



MULTICORE OS - Shutdown Synchronization



MULTICORE OS – Spinlock

- **Locking mechanism - Spinlock**
  - Replacement of Resources
  - Support mutual exclusion for Tasks & CAT2 ISRs across cores
  - Support nested call of spinlock

Service Name	Description
GetSpinlock	Acquires the spinlock
ReleaseSpinlock	Releases the acquired spinlock
TryToGetSpinlock	Checks the availability of the spinlock and acquires if available

MULTICORE OS - IOC

- **Communication in AUTOSAR**
  - Intra OS-Application communication
  - Inter ECU communication
  - Inter OS-Application Communicator

- Supports Cross memory & Cross Core

- Responsible for
  - communication between Os Applications
  - communication crossing core or memory protection boundaries

- Data transferred across cores using memory block

Thank You

www.kpit.com