



AUTOSAR OS basics and OS terminology

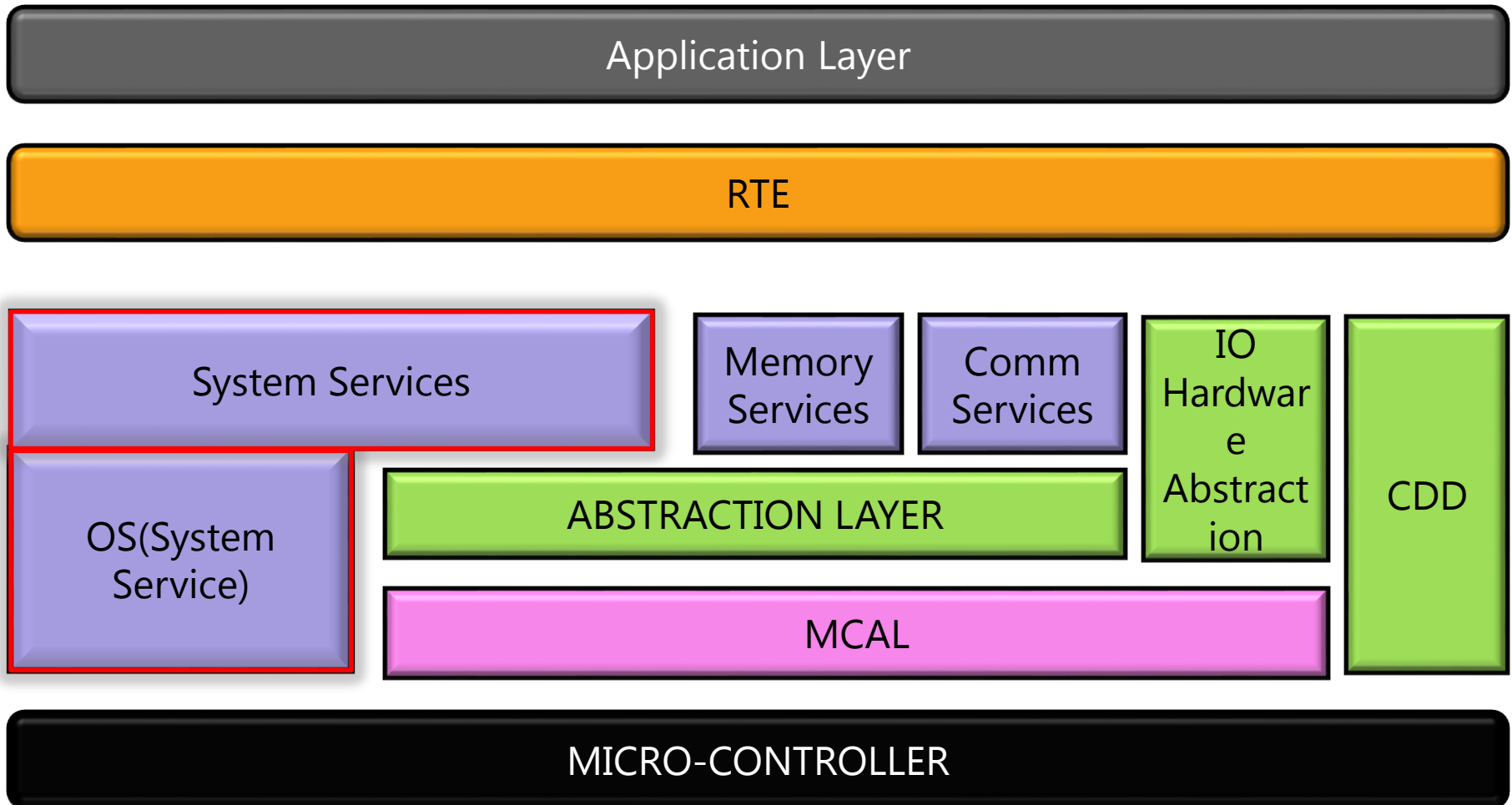
KPIT

Scope of the session

- AUTOSAR OS Responsibilities in a system
- Introduction to OSEK OS
- OS-objects at a glance
- Basic configuration for OS
- Configuration for OS-Task, OS-Alarm and OS-Counter
- Additional features by AUTOSAR OS

- AUTOSAR OS performs following activities in a system :
 - Initialization activities
 - Context management for OS-tasks that invoke Application runnables
 - Scheduling of BSW modules
 - Periodic activation of runnables e.g. Timing event
 - Event based activation of runnable e.g. Data Received Event
 - Handling all the interrupts in the system and configure Interrupt vector table for the same

OS Position in AUTOSAR Architecture



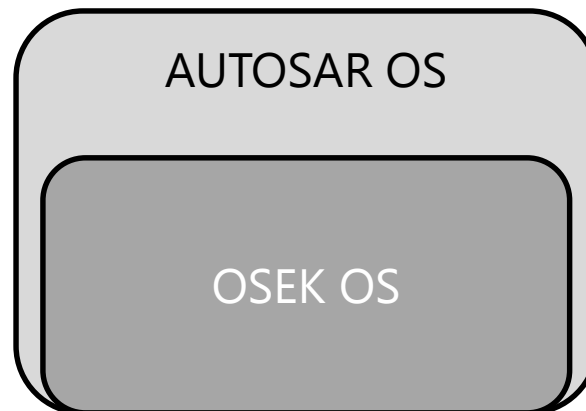
Features of AUTOSAR OS

- OsScalabilityClass should be selected as per requirements
- Scalability class of OS decides features offered by it.

Scalability Class	Services Offered
SC1	Basic Features
SC2	Basic Features + Timing Protection
SC3	Basic Features + Memory Protection
SC4	Basic Features + Timing Protection + Memory Protection

- Number of cores is decided by configuring it in OsNumberOfCores.

- AUTOSAR OS is based on OSEK OS which aims at industry standard for an open-ended architecture for distributed control units in vehicles.
- AUTOSAR OS is an extension to the OSEK OS which is a real time operating system for time critical automotive applications providing advances features like:
 - ❑ Timing protection
 - ❑ Memory protection
 - ❑ Multicore functionalities



OS objects at a glance

OS-Task

Application software can be divided into parts as per their real-time requirements

These parts (Runnable entities) can be scheduled by the means of OS-task

OS-Task is framework for execution of application

OS-Counter

OS object required to drive alarms and schedule table

It can be software driven or hardware driven(STM timer)

OS-Alarm

OS object that provides actions on expiry. The action could be activation of task, setting of event, incrementing an OS software counter or alarm callback routines(SC1 only)

Basic configuration for AUTOSAR OS

*Os_0 *OsOS_0

General OsHooks

[Parameter Name]	[Type]	[Parameter Value]	[Parameter Range]
OsNumberOfCores		<input type="text" value="1"/>	[1.....65535]
OsScalabilityClass		<input type="text" value="SC1"/>	
OsStackMonitoring		<input type="checkbox"/> false	
OsStatus		<input type="text" value="STANDARD"/>	
OsUseGetServiceId		<input type="checkbox"/> false	
OsUseParameterAccess		<input type="checkbox"/> false	
OsUseResScheduler		<input checked="" type="checkbox"/> true	
OsProcessor		<input type="text"/>	
OsProcessorSeries		<input type="text"/>	
OsIncludeHeaderFile		<input type="text"/>	
OsIdleTaskStackSize		<input type="text" value="600"/>	[4.....2147483644]
OsStackType		<input type="text" value="32"/>	[32.....32]
OsWdgSecondsPerTick		<input type="text" value="0.0"/>	[0.0.....4294967295.0]

- OsNumberOfCores
 - Allows you decide whether the system is single core or multicore
- OsScalabilityClass
 - Selection option for scalability class of the operation system
- OsStackMonitoring
 - OS feature of stack monitoring can be turned **ON** or **OFF** using this option
- OsStatus
 - OS provides error logging service which is active only when status selected is EXTENDED

Basic Configuration for OS : OS Hooks

Short Name 







General

[Parameter Name]	[Type]	[Parameter Value]	[Parameter Range]
OsErrorHook		<input type="checkbox"/> false	
OsPostTaskHook		<input type="checkbox"/> false	
OsPreTaskHook		<input type="checkbox"/> false	
OsProtectionHook		<input type="checkbox"/> false	
OsShutdownHook		<input type="checkbox"/> false	
OsStartupHook		<input type="checkbox"/> false	

Basic Configuration for OS : OS Hooks







- Hook routines are part of OS that are implemented by integrator. They are a part of integrator code
- StartUpHook : Called after call to StartOS. Any startup specific routines can be called here
- PreTaskHook : Called before entering any task. GetTaskID() API can be used in PreTaskHook to get identifier of immediate task that will be activated. It can be used for calculating processor time consumed by a particular task
- PostTaskHook : Called after exit from any task. GetTaskID() API can be used in PostTaskHook to get identifier of immediate task that got pre-empted/terminated. It can be used for calculating processor time consumed by a particular task
- ErrorHook : Called whenever there is error in invocation of particular OS service. It can be used for logging errors
- ProtectionHook : It is invoked in case of timing and memory faults in SC2/SC3/SC4 configurations
- ShutdownHook : This hook is invoked before going into shutdown. Reset functionalities may be invoked from here

Configuration for OS-Task

[Parameter Name]	[Type]	[Parameter Value]	[Parameter Range]
OsTaskActivation		<input type="text" value="1"/> 	[1.....4294967295]
OsTaskPriority		<input type="text" value="0"/> 	[0.....4294967295]
OsTaskSchedule		<div>FULL </div>	
OsTaskStackSize		<input type="text" value="800"/> 	[4.....2147483644]
OsTaskPeripheralAccess		<input type="checkbox"/> false	
OsTaskAccessingApplication		▸ OsTaskAccessingApplication	
OsTaskEventRef		▸ OsTaskEventRef	
OsTaskResourceRef		▸ OsTaskResourceRef	
OsTaskAutostart			
OsTaskTimingProtection			

- OsTaskActivation: Number of simultaneous activations of a particular task
- OsTaskPriority: Assign the priority to the task. Higher the number, higher is the priority.
- OsTaskSchedule: NON and FULL are options here. NON indicates non preemptive task that cannot be preempted by other tasks regardless of the priority of the task. FULL indicates a fully preemptive task.
- OsTaskStackSize: Assign the stack size to the task on the basis of usage of stack by the task.
- OsTaskPeripheralAccess: Applies in case of SC3/SC4
- OsTaskAccessingApplication: Applies in case of SC3/SC4

Configuration for OS-Counter

[Parameter Name]	[Type]	[Parameter Value]	[Parameter Range]
OsCounterMaxAllowedValue		<input type="text" value="1"/> 	[1.....18446744073709551615]
OsCounterMinCycle		<input type="text" value="1"/> 	[1.....18446744073709551615]
OsCounterTicksPerBase		<input type="text" value="1"/> 	[1.....4294967295]
OsCounterType		<div>SOFTWARE </div>	
OsDrivingInterrupt		<div></div>	
OsSecondsPerTick		<input type="text" value="0.0000001"/>	[0.....4294967296]
OsSecondsPerHardwareTick		<input type="text"/>	[0.....4294967296]
OsCounterAccessingApplication		▸ OsCounterAccessingApplication	

- OsCounterMaxAllowedValue: Specifies the maximum value of counter at which the counter shall reset and restart
- OsCounterType: OS supports two types of counter : **SOFTWARE** and **HARDWARE**
- OsDrivingInterrupt: Mandatory to configure this parameter for HARDWARE type counter. It selects the underlying hardware timer(STM) for driving the counter
- OsSecondsPerTick: This value provides the tick duration of OS counter. E.g If 1ms counter is needed, this value should be 0.001
- OsSecondsPerHardwareTicks: This value should be $1/(\text{STM frequency})$. E.g If STM operates at 50Mhz, this value should be 0.00000002

Configuration for OS-alarm

GeneralOsAlarmActionOsAlarmAutostart

[Parameter Name]	[Type]	[Parameter Value]	[Parameter Range]
OsAlarmAccessingApplication		OsAlarmAccessingApplication	
OsAlarmCounterRef			

[OsAlarmAction](#)
[OsAlarmAutostart](#)

< *OsAlarmActivateTask_0

Configuration Editor

Module/Container path

AUTOSAR -> Os -> OsAlarm -> OsAlarmAction -> OsAlarmActivateTask

Short Name OsAlarmActivateTask_0

General

[Parameter Name]	[Type]	[Parameter Value]	[Parameter Range]
OsAlarmActivateTaskRef			

Configuration for OS-alarm

- OsAlarmCounterRef: Gives the reference for counter that drives particular alarm
- OsAlarmActivateTaskRef: Reference to the task to be activated on alarm expiry

- Following features of SC1 OS can be used:
 - Stack Protection
 - TASK/ISR can be checked for violating stack boundaries
 - OS shall invoke shutdown for stack fault in case of SC1
 - This features enables to track any stack overflow
 - Sample use case for PreTaskHook and PostTaskHook
 - PreTaskHook is invoked every time before entering a task routine
 - PostTaskHook is invoked every time before exiting a task routine
 - Time stamps can be taken at these routines and running time for task can be easily calculated. This data would be particularly useful to analyze processing time taken up by the tasks

Thank you

www.kpit.com

