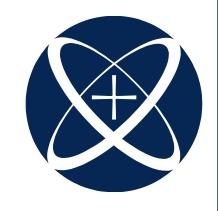
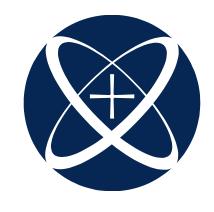
Practice II

Scheduler



- > Scheduling Mechanism
 - Scheduling refers to making a sequence of time execution decisions at specific intervals.
 - The decision that is made is based on a predictable algorithm.
 - An application that does not need its current allocation leaves the resource available for another application's use.
 - The underlying algorithm defines how the term "controlled" is interpreted. In some instances, the scheduling algorithm might guarantee that all applications have some access to the resource.
 - The Binary Progression Scheduler (BPS) manages the access to the CPU resources in a controlled way.



- > Partitioning Mechanism
 - Partitioning is used to bind a task to a subset of the system's available resources.
 - This binding guarantees that a known amount of resources is always available to the task.
 - Those resources are taken by time-slicing the available processing time.
 - Systems that use time-slicing take advantage of the CPU/Core utilization.
 - Keeping the CPU/Core occupied enhance the use of the MCU resources.
 - A processor always have a task to execute even though all the other tasks are idle.
 - When no tasks are executed the processor is running a Background Task

Mask Concept

- The Scheduler is based on a binary counter incremented at a given time, this time is controlled by an interrupt, typically called OS Tick.
- > A **mask** is a number defined by
 - $mask = (2^n)-1$
- > The mask is used to **mark** a task for execution.
- When the binary counter and the mask matches the task is executed.
- > From the mask and the OS tick period we can obtain the task rate.
- > Therefore the task rate is:
 - task rate = OS tick * (mask + 1)

Offset Concept



- > A collision may occur between the tasks.
- If a collision is present some tasks will start being executed in a not desirable time.
- An offset is defined to allow the task execution being moved in different time slots.
- The offset can only be in the range already defined by the mask starting from the count of zero.
- > With this approach the task collision is avoided.

Scheduler Implementation



- > Task Table Definition
 - Three elements are required to define a task:
 - > mask
 - > offset
 - > callback function
 - A task table can be defined with different task rates (calculated from the mask and the OS tick) and offsets to avoid task collisions.

Tasks Rates

Odd Team s- Task rate (ms)	Even Teams - Task rate (ms)
1.56	1
6.25	4
12.5	8
25	16
50	32
100	64

> The Scheduler shall provide a Background task (which runs when no tasks are scheduled).

- > The design is based on the OS tick, mask and offset concepts.
- > The scheduler module shall provide interface to support the following purposes:
 - Scheduler Initialization.
 - > SchM_Init(SchM_TaskConfigType *SchM_Config)
 - Scheduler De-initialization.
 - > SchM_DeInit(void)
 - Scheduler Start.
 - SchM_Start(void)
 - OS tick callback function.
 - > SchM_OsTick(void)
 - Backgroud Task
 - > SchM_Background(void)
 - Scheduler task callback functions.
 - > Callback functions shall be referred as per the task period
 - SchM_Task_##period(void)
 - E.g. SchM_Task_1p56ms(void)
- > The scheduler is designed in a manner that it can be portable between different platforms.

- Scheduler Module Shall be located at BSW and Services layer from AUTOSAR.
- > The following files are to be provided:
 - Provide the task configuration table
 - > SchM_Cfg.c
 - > SchM_Cfg.h
 - Provides main functionality of the scheduler OS Tick callback shall be allocated on those files
 - > SchM.c
 - > SchM.h
 - Scheduler Module type definitions
 - > SchM_Types.h
 - Periodic tasks are allocated in the following files:
 - > SchM_Tasks.c
 - > SchM_Tasks.h

Scheduler – Running Project



- > Scheduler Exercise
 - From the tasks defined in the tasks table:
 - > Turn a pin level ON at the entrance of a Task and turn the pin level OFF at the end of a task execution.
 - Measure the current CPU load
 - Hint: Turn a pin level ON before entering the Background Task and turn the pin level OFF at the end of the Background Task.
 - Modify the CPU Load by adding workload to the tasks.

Scheduler – Running Project

- > Test log shall contain the following screenshots from the scope:
 - All tasks execution
 - Task Periods
 - CPU Load