Version List

Version Index	Date	Author	Description
0.1	13-Jul-2015	Francisco Martinez-Chavez	First revision of Scheduler Specification
0.2	23-Aug-2016	Francisco Martinez-Chavez	Minor fixes
0.3	10-Nov-2017	Francisco Martinez-Chavez	Minor updates

Scheduler Module

Functional Specification

Scheduler Mechanism

Scheduling refers to making a sequence of time execution decisions at specific intervals, this 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.

Tasks Partitioning

Task 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 and keeping the CPU/Core occupied which 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 a timer interrupt, typically called OS Tick.

A mask is a number defined by: mask = (2^n)-1

> Where n represents the counter data size (8bits, 16bits …) which depends on the number of tasks to be provided by the scheduler module, n should be choosen at desing stage by the scheduler designer.

The mask is used to mark a task for execution, when the binary counter and the mask:

    (mask & counter) == mask

From the previous definition, the task is assigned to a range of time-slices.

Given the mask and the OS tick period we can obtain the task rate.
Therefore the task rate is:

    task rate = OS tick * (mask + 1)

Note: There shall be one mask per task.

Offset Concept

A collision may occur between the tasks when the tasks share the same time-slice. 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-slices.
The offset can only be defined in the range from the count of zero up to the value defined by the mask.
When the counter matches the mask, and the matched value is the same as the given offset the task is ready to be executed.

    (mask & counter) == offset

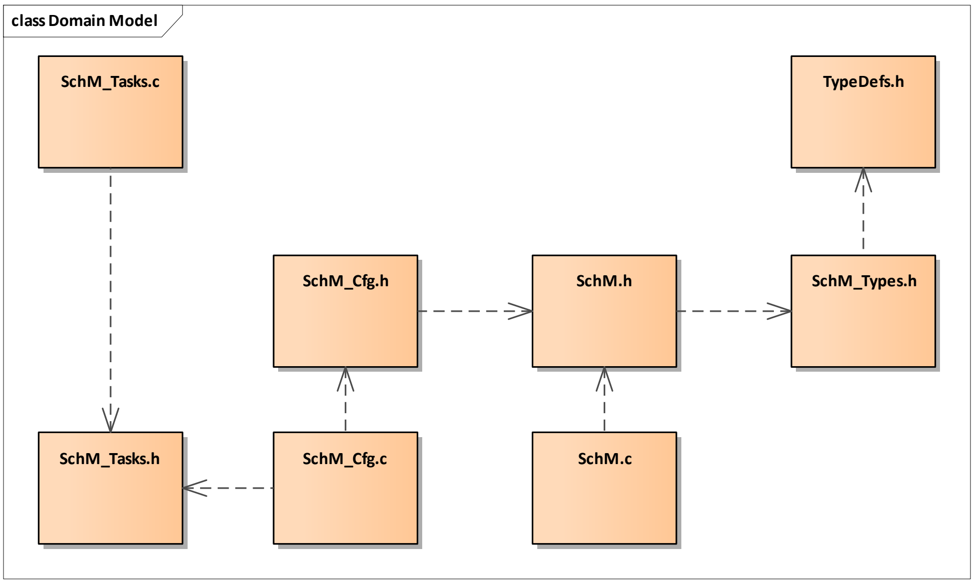
With this approach the task collision is avoided.

Dependencies to other modules

The scheduler module has dependencies on project specific timer module.

File Structure

The include structure of the SchM module shall be as follows:



Note: Scheduler user module header files shall be included only in SchM_Tasks.c file.

File Name	Description
SchM_Cfg.c	Provides the general scheduler configuration and tasks descriptor table
SchM_Cfg.h	Exports the general scheduler configuration
SchM.c	Provides the scheduler main functionality
SchM.h	Exports the public scheduler interfaces
SchM_Types.h	Scheduler module types
SchM_Tasks.c	Provides the timed task definitions
SchM_Tasks.h	Export the timed task interfaces to the scheduler configuration file

* **Only** scheduler user module interfaces should be called from this file.

API Specification

Type Definitions

Scheduler type definitions shall be defined in SchM_Types.h file.

SchM_TaskMaskType

Name:	SchM_TaskMaskType		
Type:	u8		
Range:	SCHM_MASK_3P125MS	0x03	Mask required for 3.125 ms task
	SCHM_MASK_6P25MS	0x07	Mask required for 6.25 ms task
	SCHM_MASK_12P5MS	0x0F	Mask required for 12.5 ms task
	SCHM_MASK_25MS	0x1F	Mask required for 25 ms task
	SCHM_MASK_50MS	0x3F	Mask required for 50 ms task
	SCHM_MASK_100MS	0x7F	Mask required for 100 ms task
Description:	The mask values to generate the task periods		

SchM_TaskIDType

Name:	SchM_TaskIDType	
Type:	u8	
Range:	SCHM_TASKID_BKG	Background Task ID
	SCHM_TASKID_3P125MS	3.125 ms Task ID
	SCHM_TASKID_6P25MS	6.25 ms Task ID
	SCHM_TASKID_12P5MS	12.5 ms Task ID
	SCHM_TASKID_25MS	25 ms Task ID
	SCHM_TASKID_50MS	50 ms Task ID
	SCHM_TASKID_100MS	100 ms Task ID
Description:	Task ID values	

SchM_TaskStateType

Name:	SchM_TaskStateType	
Type:	u8	
Range:	SCHM_TASK_STATE_SUSPENDED	Tasks state initial value
	SCHM_TASK_STATE_READY	Task state indicates the task is ready to be executed
	SCHM_TASK_STATE_RUNNING	Task state indicates the task is currently running
Description:	Task States	

SchM_SchedulerStateType

Name:	SchM_SchedulerStateType	
Type:	u8	
Range:	SCHM_UNINIT	Scheduler state initial value
	SCHM_INIT	Scheduler state after initialization
	SCHM_IDLE	Scheduler state when background task is executed
	SCHM_RUNNING	Scheduler state when a task is executed
	SCHM_OVERLOAD	Scheduler state when task collision was present
	SCHM_HALTED	Scheduler state when scheduler has been stopped
Description:	Task ID values	

SchM_ConfigType

Name:	SchM_ConfigType	
Type:	Structure	
Range:	Implementation Specific Structure	Structure to hold the module's configuration set. The contents of this data structure are implementation specific.
Description:	Structure for the purpose of configuration.	

Public Function Definitions

Public functions shall be exported in SchM.h file and defined in SchM.c file.

SchM_Init

Service Name:	SchM_Init	
Syntax:	void SchM_Init (const SchM_ConfigType *SchMConfig)	
Parameters (in-out):	SchM_ConfigType	Structure for the purpose of configuration.
Parameters (out):	none	
Return Value:	none	
Description:	Function initialization of Scheduler module	

The SchM_Init function shall allocate and initialize the resources to be used by the Scheduler Module, including the timer module initialization used for the tick reference and resources requested by the SchMConfig parameter, this means:

- Initialize the callback funtion passed as reference to the timer module used for the tick reference.
- Initialize all the tasks according to the task descriptor to suspended state.
- Initialize the scheduler state to initialized.

SchM_Start

Service Name:	SchM_Start
Syntax:	void SchM_Start (void)
Parameters (in-out):	none
Parameters (out):	none
Return Value:	none
Description:	Function starts the execution of Scheduler module

The SchM_Start function shall start the timer channel used for the tick reference, set the scheduler state to Idle state and call the SchM_Background function.

SchM_Stop

Service Name:	SchM_Stop
Syntax:	void SchM_Stop (void)
Parameters (in-out):	none
Parameters (out):	none
Return Value:	none
Description:	Function stops the execution of Scheduler module

The SchM_Stop function shall stop the timer channel used for the tick reference and set the scheduler state to halted.

Private Function Definitions
Private functions shall be defined in SchM.c file.

SchM_OsTick

Service Name:	SchM_OsTick
Syntax:	void SchM_OsTick(void)
Parameters (in-out):	none
Parameters (out):	none
Return Value:	none
Description:	Callback function periodically called from the timer module providing the tick reference

The SchM_OsTick function shall be indirectly called by the timer module used for the tick reference, when the timer expires.
This function shall increment by one the internal counter and set the correspondig task state to ready as per the defined rate monotonic scheduler algorithm based on the following task descriptor fileds:

- Mask
- Offset

SchM_Background

Service Name:	SchM_Background
Syntax:	void SchM_Background(void)
Parameters (in-out):	none
Parameters (out):	none
Return Value:	none
Description:	Background function executed when scheduler state is idle

The SchM_Background function shall execute in an infinite loop.
This function searchs for all the tasks to be in the ready state to be executed and:

- Before the task execution:
  - Set the scheduler state to running.
  - Set the task state to be executed to running.
- After the task execution:
  - Set the scheduler state to idle.
  - Set the task state to suspended.

Task Function Definitions

Task functions shall be exported in SchM_Tasks.h file and defined in SchM_Tasks.c.

SchM_<TaskPrefix>_Task

Service Name:	SchM_<TaskPrefix>_Task
Syntax:	void SchM_<TaskPrefix>_Task(void)
Parameters (in-out):	none
Parameters (out):	none
Return Value:	none
Description:	Timed Task executed with a predefined period

Task functions shall be referred as per the task period, e.g. for 3.125ms task:

SchM_<TaskPrefix>_Task -> SchM_3p125ms_Task

The number of task functions shall exist according to the number of tasks as per Scheduler configuration from the task descriptor.