REPORT - PROJECT 3

Advanced Real Time Systems - ECE 5550G

PART 1 -

I) A Brief Description of Rate Monotonic (RM) Scheduling Algorithm :

- It is a priority-based preemptive scheduling algorithm that assigns fixed priorities to each task based on their periods. The tasks with shorter periods receive a higher priority.
- We assume that the tasks are periodic and that their execution times are deterministic and known beforehand, and the context switching overhead is negligible compared to the execution time of the tasks.
- The task with the highest priority i.e. shortest period is always executed first, this ensures that all the deadlines are met. If a task misses its deadline, it is dropped and rescheduled to the next period of the task.

II) Code Implementation of the Rate Monotonic (RM) Scheduling Algorithm :

By implementing the following edits to the scheduler.cpp, we can the RM algorithm to work :

- **prvFindEmptyElementIndexTCB** : This function iterates over xTCBArray, and checks which TCB block is not in use and returns the index of that TCB block.

```
static BaseType_t prvFindEmptyElementIndexTCB( void )
{
    /* your implementation goes here */
    BaseType_t xIndex = -1;
    BaseType_t xIterator;

    for (xIterator = 0; xIterator < schedMAX_NUMBER_OF_PERIODIC_TASKS; xIterator++ )
    {
        if (pdFALSE == xTCBArray[xIterator].xInUse)
        {
            xIndex = xIterator;
            break;
        }
    }

    return xIndex;
}
```

- **prvDeleteTCBFromArray** : This function deletes the TCB block at the index specified by the user by setting the xInUse parameter to pdFALSE.

```
static void prvDeleteTCBFromArray( BaseType_t xIndex )
{
    configASSERT(0 <= xIndex && xIndex < schedMAX_NUMBER_OF_PERIODIC_TASKS);
    configASSERT(pdTRUE == xTCBArray[xIndex].xInUse);

    if (pdTRUE == xTCBArray[xIndex].xInUse)
    {
        xTCBArray[xIndex].xInUse = pdFALSE;
        xTaskCounter--;
    }
}
```

- **prvCreateAllTasks** : This function creates a task and adds it to the list of tasks that are ready to run for each TCB block that is in use in the xTCBArray.

- **vSchedulerPeriodicTaskCreate** : This function adds the task's properties specified by the user to an empty TCB block in the xTCBArray.

- **vSchedulerPeriodicTaskDelete** : This function is used for deleting a task from the FreeRTOS kernel as well as its data in the xTCBArray.

- **prvPeriodicTaskRecreate** : This function recreates the task that has been deleted using the data from the task's corresponding TCB block in the xTCBArray.

- **prvPeriodicTaskCode** : This is a wrapper function that wraps each task created by the user. This function is called by FreeRTOS when a task is in the ready state and has highest priority within the list of ready tasks. This function in turn calls the corresponding task that needs to be run and delays until the release time after executing the task.

```c
static void prvPeriodicTaskCode( void *pvParameters )
{
    TickType_t xStartTick, xEndTick;
    BaseType_t xIndex;
    SchedTCB_t *pxThisTask;
    TaskHandle_t xCurrentTaskHandle = xTaskGetCurrentTaskHandle();

    /* Check the handle is not NULL. */
    configASSERT(NULL != xCurrentTaskHandle);
    xIndex = prvGetTCBIndexFromHandle(xCurrentTaskHandle);
    configASSERT(-1 != xIndex);
    pxThisTask = &xTCBArray[xIndex];

    #if( schedUSE_TIMING_ERROR_DETECTION_DEADLINE == 1 )
        pxThisTask->xExecutedOnce = pdTRUE;
    #endif /* schedUSE_TIMING_ERROR_DETECTION_DEADLINE */

    PRINTF("FUNC: %s", __func__);
    PRINTF(" -> TASK: %s, INIT RUN\n", pxThisTask->pcName);

    if( 0 == pxThisTask->xReleaseTime )
    {
        pxThisTask->xLastWakeTime = xSystemStartTime;
    }
    else
    {
        xTaskDelayUntil(&pxThisTask->xLastWakeTime, pxThisTask->xReleaseTime);
    }

    for( ; ; )
    {
        PRINTF("TASK: %-2s\n",pxThisTask->pcName);

        pxThisTask->xWorkIsDone = pdFALSE;

        xStartTick = xTaskGetTickCount();
        pxThisTask->pvTaskCode( pvParameters );
        xEndTick = xTaskGetTickCount();

        pxThisTask->xWorkIsDone = pdTRUE;
        pxThisTask->xExecTime = 0;

        PRINTF("STAT: %-2s, ST:%04u, ET:%04u, RT:%02u, DT: %04u\n",
        pxThisTask->pcName, xStartTick, xEndTick, (xEndTick - xStartTick), pxThisTask->xAbsoluteDeadline);

        xTaskDelayUntil(&pxThisTask->xLastWakeTime, pxThisTask->xPeriod );
    }
}
```

- **prvSetFixedPriorities** : This function assigns priorities to the tasks according to the RM Algorithm. We keep iterating through the xTCBArray until all tasks have been assigned priorities; We find the task with the shortest period and assign the highest priority and so on and so forth.

```
for( xIter = 0; xIter < xTaskCounter; xIter++ )
{
    xShortest = portMAX_DELAY;
    /* search for shortest period */
    for( xIndex = 0; xIndex < xTaskCounter; xIndex++ )
    {
        pxTCB = &xTCBArray[xIndex];
        configASSERT(pdTRUE == pxTCB->xInUse);

        if (pdFALSE == pxTCB->xPriorityIsSet)
        {
            #if( schedSCHEDULING_POLICY == schedSCHEDULING_POLICY_RMS )
                if (pxTCB->xPeriod <= xShortest)
                {
                    xShortest = pxTCB->xPeriod;
                    pxShortestTaskPointer = pxTCB;
                }
            #endif /* schedSCHEDULING_POLICY */
        }
    }
    if (xShortest != xPreviousShortest)
    {
        xHighestPriority--;
    }
    configASSERT(tskIDLE_PRIORITY < xHighestPriority);
    pxShortestTaskPointer->uxPriority = xHighestPriority;
    pxShortestTaskPointer->xPriorityIsSet = pdTRUE;
    xPreviousShortest = xShortest;
    PRINTF(" Task : %s, Priority : %d, Tick : %d\n",
    pxShortestTaskPointer->pcName, pxShortestTaskPointer->uxPriority, xShortest);

}
```

- **prvSchedulerCheckTimingError** : This function checks for 2 types of timing errors: Deadline Miss and Maximum Execution Time.
    - For a Deadline miss, If a task has executed once and hasn't finished running, **prvCheckDeadline** function is called to check for a deadline miss. If a deadline miss has occurred, the task is deleted and recreated with new timing values.
    - For a Maximum Execution Exceeded, If a task has exceeded its WCET, the task is suspended. For a task that has already been suspended, check if the next release time of the task has arrived and resume the task. (**prvExecTimeExceedHook** function checks if any task has exceeded its execution time and sets the required flags).

```c
static void prvSchedulerCheckTimingError( TickType_t xTickCount, SchedTCB_t *pxTCB )
{

    #if( schedUSE_TIMING_ERROR_DETECTION_DEADLINE == 1 )
        if ((pdTRUE == pxTCB->xWorkIsDone) && (( signed ) ( xTickCount - pxTCB->xLastWakeTime ) > 0))
        {
            pxTCB->xWorkIsDone = pdFALSE;
        }

        /* check if task missed deadline */
        if ((pdTRUE == pxTCB->xExecutedOnce) && (pdFALSE == pxTCB->xWorkIsDone ))
        {
            prvCheckDeadline(pxTCB, xTickCount);
        }
    #endif /* schedUSE_TIMING_ERROR_DETECTION_DEADLINE */


    #if( schedUSE_TIMING_ERROR_DETECTION_EXECUTION_TIME == 1 )
    if( pdTRUE == pxTCB->xMaxExecTimeExceeded )
    {
        pxTCB->xMaxExecTimeExceeded = pdFALSE;
        vTaskSuspend( *pxTCB->pxTaskHandle );
    }
    if( pdTRUE == pxTCB->xSuspended )
    {
        if( ( signed ) ( pxTCB->xAbsoluteUnblockTime - xTickCount ) <= 0 )
        {
            pxTCB->xSuspended = pdFALSE;
            pxTCB->xLastWakeTime = xTickCount;
            vTaskResume( *pxTCB->pxTaskHandle );
        }
    }
    #endif /* schedUSE_TIMING_ERROR_DETECTION_EXECUTION_TIME */

    return;
}
```

- **prvSchedulerFunction** : This function is used to check for timing errors that might have occurred for any of the tasks. We iterate through the xTCBArray and for every task that is in use, we pass it to the **prvSchedulerCheckTimingError** to check for timing errors.

```
static void prvSchedulerFunction( void *pvParameters )
{

    for( ; ; )
    {
        #if( schedUSE_TIMING_ERROR_DETECTION_DEADLINE == 1 || schedUSE_TIMING_ERROR_DETECTION_EXECUTION_TIME == 1 )
            TickType_t xTickCount = xTaskGetTickCount();
            UBaseType_t xIndex;
            SchedTCB_t *pxTCB;

            for (xIndex = 0; xIndex < schedMAX_NUMBER_OF_PERIODIC_TASKS; xIndex++)
            {
                pxTCB = &xTCBArray[xIndex];
                /* Only check tasks which are in use */
                if (pdTRUE == pxTCB->xInUse)
                {
                    prvSchedulerCheckTimingError(xTickCount, pxTCB);
                }
            }
        #endif /* schedUSE_TIMING_ERROR_DETECTION_DEADLINE || schedUSE_TIMING_ERROR_DETECTION_EXECUTION_TIME */

        #if (schedOVERHEAD == 1)
            TickType_t xTicks = schedOVERHEAD_TICKS;
            while(xTicks--)
            {
                // Runs the CPU for 15 ms at FCLK = 16Mhz; 15 ms is WatchDog Timer Interrupt
                _delay_loop_2(60000);
            }
        #endif

        ulTaskNotifyTake( pdTRUE, portMAX_DELAY );
    }
}
```

- **prvDeadlineMissedHook** : This function is called if a task has had a deadline miss. The task is deleted from the FreeRTOS kernel and recreated using **prvPeriodicTaskRecreate** function.

```
static void prvDeadlineMissedHook( SchedTCB_t *pxTCB, TickType_t xTickCount )
{
    PRINTF("FUNC: %s", __func__);
    PRINTF(" -> TASK: %s, T : %d\n", pxTCB->pcName, xTickCount);

    /* Delete the pxTask and recreate it. */
    vTaskDelete(*(pxTCB->pxTaskHandle));
    pxTCB->xExecTime = 0;
    prvPeriodicTaskRecreate(pxTCB);

    pxTCB->xReleaseTime = pxTCB->xLastWakeTime + pxTCB->xPeriod;
    pxTCB->xAbsoluteDeadline = pxTCB->xLastWakeTime + pxTCB->xPeriod + pxTCB->xRelativeDeadline;
}
```

- **prvCheckDeadline** : This function checks if a given task has exceeded its absolute deadline and calls **prvDeadlineMissedHook** function.

```
static void prvCheckDeadline( SchedTCB_t *pxTCB, TickType_t xTickCount )
{
    pxTCB->xAbsoluteDeadline = pxTCB->xLastWakeTime + pxTCB->xRelativeDeadline;

    if (( signed ) ( pxTCB->xAbsoluteDeadline - xTickCount ) < 0)
    {
        prvDeadlineMissedHook( pxTCB, xTickCount );
    }
}
```

PART 2 -

I) A Brief Description of Deadline Monotonic (DM) Scheduling Algorithm :

- It is similar to the Rate Monotonic (RM) Scheduling Algorithm. But, instead of prioritizing tasks based on their periods, the Deadline Monotonic (DM) Scheduling Algorithm prioritizes tasks based on their relative deadlines. The tasks with shorter relative deadlines will have a higher priority.
- The algorithm works by continuously monitoring the state of the system and selecting the highest priority task that is ready to execute. If a higher-priority task becomes ready while a lower-priority task is executing, the lower-priority task is preempted, and the higher-priority task is scheduled to run.

II) Code Implementation of the Rate Monotonic (RM) Scheduling Algorithm :

By implementing the following edits to the scheduler.cpp, we can the RM algorithm to work :

- **prvSetFixedPriorities** : This function assigns priorities to the tasks according to the DM Algorithm. We keep iterating through the xTCBArray until all tasks have been assigned priorities; We find the task with the earliest deadline and assign the highest priority and so on and so forth.

```
for( xIter = 0; xIter < xTaskCounter; xIter++ )
{
    xShortest = portMAX_DELAY;
    /* search for shortest period */
    for( xIndex = 0; xIndex < xTaskCounter; xIndex++ )
    {
        pxTCB = &xTCBArray[xIndex];
        configASSERT(pdTRUE == pxTCB->xInUse);

        if (pdFALSE == pxTCB->xPriorityIsSet)
        {
            #elif( schedSCHEDULING_POLICY == schedSCHEDULING_POLICY_DMS )
                if (pxTCB->xRelativeDeadline <= xShortest)
                {
                    xShortest = pxTCB->xRelativeDeadline;
                    pxShortestTaskPointer = pxTCB;
                }
            #endif /* schedSCHEDULING_POLICY */
        }
    }
    if (xShortest != xPreviousShortest)
    {
        xHighestPriority--;
    }
    configASSERT(tskIDLE_PRIORITY < xHighestPriority);
    pxShortestTaskPointer->uxPriority = xHighestPriority;
    pxShortestTaskPointer->xPriorityIsSet = pdTRUE;
    xPreviousShortest = xShortest;
    PRINTF(" Task : %s, Priority : %d, Tick : %d\n",
    pxShortestTaskPointer->pcName, pxShortestTaskPointer->uxPriority, xShortest);

}
```

PART 3 -

For Task Set 1, the periods and relative deadlines of the tasks are such that they are assigned the same priorities if you use RM or DM Algorithm. Hence there is no difference that can be observed between the outputs of both algorithms.

- Running Task Set 1 using RM Algorithm

```
---- Opened the serial port COM5 ----
----- Program Started -----
FUNC: vSchedulerInit
FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T1
Phase Tick          : 0
Max. Execution Tick : 6
Rel. Deadline Tick  : 24
Period Tick         : 24
Priority            : 0
----------------------

FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T2
Phase Tick          : 0
Max. Execution Tick : 12
Rel. Deadline Tick  : 42
Period Tick         : 48
Priority            : 0
----------------------

FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T3
Phase Tick          : 0
Max. Execution Tick : 9
Rel. Deadline Tick  : 63
Period Tick         : 63
Priority            : 0
----------------------

FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T4
Phase Tick          : 0
Max. Execution Tick : 18
Rel. Deadline Tick  : 309
Period Tick         : 309
Priority            : 0
----------------------

FUNC: vSchedulerStart
FUNC: prvSetFixedPriorities
----Using RM Scheduling Algorithm----
 Task : T1, Priority : 5, Tick : 24
 Task : T2, Priority : 4, Tick : 48
 Task : T3, Priority : 3, Tick : 63
 Task : T4, Priority : 2, Tick : 309
-------------------------------------

FUNC: prvCreateSchedulerTask
---- Scheduler Details ----
Period Tick  : 3
Priority     : 6
```

```
FUNC: prvCreateSchedulerTask
---- Scheduler Details ----
Period Tick  : 3
Priority     : 6
Overhead     : 0
---------------------------

FUNC: prvCreateAllTasks
FUNC: prvPeriodicTaskCode -> TASK: T1, INIT RUN
TASK: T1
STAT: T1, ST:0000, ET:0004, RT:04, DT: 0024
FUNC: prvPeriodicTaskCode -> TASK: T2, INIT RUN
TASK: T2
STAT: T2, ST:0004, ET:0013, RT:09, DT: 0042
FUNC: prvPeriodicTaskCode -> TASK: T3, INIT RUN
TASK: T3
STAT: T3, ST:0014, ET:0020, RT:06, DT: 0063
FUNC: prvPeriodicTaskCode -> TASK: T4, INIT RUN
TASK: T4
TASK: T1
STAT: T1, ST:0024, ET:0028, RT:04, DT: 0048
STAT: T4, ST:0020, ET:0039, RT:19, DT: 0309
TASK: T1
STAT: T1, ST:0048, ET:0052, RT:04, DT: 0072
TASK: T2
STAT: T2, ST:0052, ET:0061, RT:09, DT: 0090
TASK: T3
STAT: T3, ST:0063, ET:0069, RT:06, DT: 0126
TASK: T1
STAT: T1, ST:0072, ET:0076, RT:04, DT: 0096
TASK: T1
STAT: T1, ST:0096, ET:0100, RT:04, DT: 0120
TASK: T2
STAT: T2, ST:0100, ET:0109, RT:09, DT: 0138
TASK: T1
STAT: T1, ST:0120, ET:0124, RT:04, DT: 0144
TASK: T3
STAT: T3, ST:0126, ET:0132, RT:06, DT: 0189
TASK: T1
STAT: T1, ST:0144, ET:0148, RT:04, DT: 0168
TASK: T2
STAT: T2, ST:0148, ET:0157, RT:09, DT: 0186
TASK: T1
STAT: T1, ST:0168, ET:0172, RT:04, DT: 0192
TASK: T3
TASK: T1
STAT: T1, ST:0192, ET:0196, RT:04, DT: 0216
TASK: T2
STAT: T2, ST:0196, ET:0205, RT:09, DT: 0234
STAT: T3, ST:0189, ET:0209, RT:20, DT: 0252
TASK: T1
STAT: T1, ST:0216, ET:0220, RT:04, DT: 0240
TASK: T1
STAT: T1, ST:0240, ET:0244, RT:04, DT: 0264
TASK: T2
STAT: T2, ST:0244, ET:0253, RT:09, DT: 0282
```

- Running Task Set 1 using DM Algorithm

```
---- Opened the serial port COM5 ----
----- Program Started -----
FUNC: vSchedulerInit
FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T1
Phase Tick          : 0
Max. Execution Tick : 6
Rel. Deadline Tick  : 24
Period Tick         : 24
Priority            : 0
----------------------

FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T2
Phase Tick          : 0
Max. Execution Tick : 12
Rel. Deadline Tick  : 42
Period Tick         : 48
Priority            : 0
----------------------

FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T3
Phase Tick          : 0
Max. Execution Tick : 9
Rel. Deadline Tick  : 63
Period Tick         : 63
Priority            : 0
----------------------

FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T4
Phase Tick          : 0
Max. Execution Tick : 18
Rel. Deadline Tick  : 309
Period Tick         : 309
Priority            : 0
----------------------

FUNC: vSchedulerStart
FUNC: prvSetFixedPriorities
----Using DM Scheduling Algorithm----
 Task : T1, Priority : 5, Tick : 24
 Task : T2, Priority : 4, Tick : 42
 Task : T3, Priority : 3, Tick : 63
 Task : T4, Priority : 2, Tick : 309
-------------------------------------

FUNC: prvCreateSchedulerTask
---- Scheduler Details ----
Period Tick  : 3
Priority     : 6
```

```
FUNC: prvCreateSchedulerTask
---- Scheduler Details ----
Period Tick  : 3
Priority     : 6
Overhead     : 0
---------------------------

FUNC: prvCreateAllTasks
FUNC: prvPeriodicTaskCode -> TASK: T1, INIT RUN
TASK: T1
STAT: T1, ST:0000, ET:0004, RT:04, DT: 0024
FUNC: prvPeriodicTaskCode -> TASK: T2, INIT RUN
TASK: T2
STAT: T2, ST:0004, ET:0013, RT:09, DT: 0042
FUNC: prvPeriodicTaskCode -> TASK: T3, INIT RUN
TASK: T3
STAT: T3, ST:0014, ET:0020, RT:06, DT: 0063
FUNC: prvPeriodicTaskCode -> TASK: T4, INIT RUN
TASK: T4
TASK: T1
STAT: T1, ST:0024, ET:0028, RT:04, DT: 0048
STAT: T4, ST:0020, ET:0039, RT:19, DT: 0309
TASK: T1
STAT: T1, ST:0048, ET:0052, RT:04, DT: 0072
TASK: T2
STAT: T2, ST:0052, ET:0061, RT:09, DT: 0090
TASK: T3
STAT: T3, ST:0063, ET:0069, RT:06, DT: 0126
TASK: T1
STAT: T1, ST:0072, ET:0076, RT:04, DT: 0096
TASK: T1
STAT: T1, ST:0096, ET:0100, RT:04, DT: 0120
TASK: T2
STAT: T2, ST:0100, ET:0109, RT:09, DT: 0138
TASK: T1
STAT: T1, ST:0120, ET:0124, RT:04, DT: 0144
TASK: T3
STAT: T3, ST:0126, ET:0132, RT:06, DT: 0189
TASK: T1
STAT: T1, ST:0144, ET:0148, RT:04, DT: 0168
TASK: T2
STAT: T2, ST:0148, ET:0157, RT:09, DT: 0186
TASK: T1
STAT: T1, ST:0168, ET:0172, RT:04, DT: 0192
TASK: T3
TASK: T1
STAT: T1, ST:0192, ET:0196, RT:04, DT: 0216
TASK: T2
STAT: T2, ST:0196, ET:0205, RT:09, DT: 0234
STAT: T3, ST:0189, ET:0209, RT:20, DT: 0252
TASK: T1
STAT: T1, ST:0216, ET:0220, RT:04, DT: 0240
TASK: T1
STAT: T1, ST:0240, ET:0244, RT:04, DT: 0264
TASK: T2
```

For Task Set 2, the RM algorithm produces a priority order of T1>T2>T3 >T4 and the DM Algorithm produces a priority order of T2>T1>T3>T4. Hence we can see in the figures that T2 is executed first in the case of DM Algorithm and T1 is executed first in the case of RM Algorithm

- Running Task Set 2 using RM Algorithm

```
---- Opened the serial port COM5 ----
----- Program Started -----
FUNC: vSchedulerInit
FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T1
Phase Tick          : 0
Max. Execution Tick : 6
Rel. Deadline Tick  : 24
Period Tick         : 24
Priority            : 0
----------------------

FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T2
Phase Tick          : 0
Max. Execution Tick : 9
Rel. Deadline Tick  : 12
Period Tick         : 30
Priority            : 0
----------------------

FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T3
Phase Tick          : 0
Max. Execution Tick : 12
Rel. Deadline Tick  : 42
Period Tick         : 48
Priority            : 0
----------------------

FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T4
Phase Tick          : 0
Max. Execution Tick : 9
Rel. Deadline Tick  : 63
Period Tick         : 63
Priority            : 0
----------------------

FUNC: vSchedulerStart
FUNC: prvSetFixedPriorities
----Using RM Scheduling Algorithm----
 Task : T1, Priority : 5, Tick : 24
 Task : T2, Priority : 4, Tick : 30
 Task : T3, Priority : 3, Tick : 48
 Task : T4, Priority : 2, Tick : 63
-------------------------------------

FUNC: prvCreateSchedulerTask
---- Scheduler Details ----
Period Tick  : 3
Priority     : 6
```

```
FUNC: prvCreateSchedulerTask
---- Scheduler Details ----
Period Tick  : 3
Priority     : 6
Overhead     : 0
---------------------------

FUNC: prvCreateAllTasks
FUNC: prvPeriodicTaskCode -> TASK: T1, INIT RUN
TASK: T1
STAT: T1, ST:0000, ET:0004, RT:04, DT: 0024
FUNC: prvPeriodicTaskCode -> TASK: T2, INIT RUN
TASK: T2
STAT: T2, ST:0004, ET:0011, RT:07, DT: 0012
FUNC: prvPeriodicTaskCode -> TASK: T3, INIT RUN
TASK: T3
STAT: T3, ST:0011, ET:0020, RT:09, DT: 0042
FUNC: prvPeriodicTaskCode -> TASK: T4, INIT RUN
TASK: T4
TASK: T1
STAT: T1, ST:0024, ET:0028, RT:04, DT: 0048
TASK: T2
STAT: T2, ST:0030, ET:0036, RT:06, DT: 0042
STAT: T4, ST:0020, ET:0038, RT:18, DT: 0063
TASK: T1
STAT: T1, ST:0048, ET:0052, RT:04, DT: 0072
TASK: T3
TASK: T2
STAT: T2, ST:0060, ET:0066, RT:06, DT: 0072
STAT: T3, ST:0052, ET:0068, RT:16, DT: 0090
TASK: T4
TASK: T1
STAT: T1, ST:0072, ET:0076, RT:04, DT: 0096
STAT: T4, ST:0068, ET:0079, RT:11, DT: 0126
TASK: T2
TASK: T1
STAT: T1, ST:0096, ET:0100, RT:04, DT: 0120
STAT: T2, ST:0090, ET:0101, RT:11, DT: 0102
TASK: T3
STAT: T3, ST:0101, ET:0110, RT:09, DT: 0138
TASK: T1
STAT: T1, ST:0120, ET:0124, RT:04, DT: 0144
TASK: T2
STAT: T2, ST:0124, ET:0131, RT:07, DT: 0132
TASK: T4
STAT: T4, ST:0131, ET:0138, RT:07, DT: 0189
TASK: T1
STAT: T1, ST:0144, ET:0148, RT:04, DT: 0168
TASK: T3
TASK: T2
STAT: T2, ST:0150, ET:0156, RT:06, DT: 0162
STAT: T3, ST:0148, ET:0164, RT:16, DT: 0186
TASK: T1
STAT: T1, ST:0168, ET:0172, RT:04, DT: 0192
TASK: T2
STAT: T2, ST:0180, ET:0186, RT:06, DT: 0192
```

- Running Task Set 2 using DM Algorithm

```
----- Program Started -----
FUNC: vSchedulerInit
FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T1
Phase Tick          : 0
Max. Execution Tick : 6
Rel. Deadline Tick  : 24
Period Tick         : 24
Priority            : 0
----------------------

FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T2
Phase Tick          : 0
Max. Execution Tick : 9
Rel. Deadline Tick  : 12
Period Tick         : 30
Priority            : 0
----------------------

FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T3
Phase Tick          : 0
Max. Execution Tick : 12
Rel. Deadline Tick  : 42
Period Tick         : 48
Priority            : 0
----------------------

FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T4
Phase Tick          : 0
Max. Execution Tick : 9
Rel. Deadline Tick  : 63
Period Tick         : 63
Priority            : 0
----------------------

FUNC: vSchedulerStart
FUNC: prvSetFixedPriorities
----Using DM Scheduling Algorithm----
 Task : T2, Priority : 5, Tick : 12
 Task : T1, Priority : 4, Tick : 24
 Task : T3, Priority : 3, Tick : 42
 Task : T4, Priority : 2, Tick : 63
-------------------------------------

FUNC: prvCreateSchedulerTask
---- Scheduler Details ----
Period Tick  : 3
Priority     : 6
Overhead     : 0
```

```
FUNC: prvCreateSchedulerTask
---- Scheduler Details ----
Period Tick  : 3
Priority     : 6
Overhead     : 0
--------------------------

FUNC: prvCreateAllTasks
FUNC: prvPeriodicTaskCode -> TASK: T2, INIT RUN
TASK: T2
STAT: T2, ST:0000, ET:0006, RT:06, DT: 0012
FUNC: prvPeriodicTaskCode -> TASK: T1, INIT RUN
TASK: T1
STAT: T1, ST:0007, ET:0011, RT:04, DT: 0024
FUNC: prvPeriodicTaskCode -> TASK: T3, INIT RUN
TASK: T3
STAT: T3, ST:0011, ET:0020, RT:09, DT: 0042
FUNC: prvPeriodicTaskCode -> TASK: T4, INIT RUN
TASK: T4
TASK: T1
STAT: T1, ST:0024, ET:0028, RT:04, DT: 0048
TASK: T2
STAT: T2, ST:0030, ET:0036, RT:06, DT: 0042
STAT: T4, ST:0020, ET:0038, RT:18, DT: 0063
TASK: T1
STAT: T1, ST:0048, ET:0052, RT:04, DT: 0072
TASK: T3
TASK: T2
STAT: T2, ST:0060, ET:0066, RT:06, DT: 0072
STAT: T3, ST:0052, ET:0068, RT:16, DT: 0090
TASK: T4
TASK: T1
STAT: T1, ST:0072, ET:0076, RT:04, DT: 0096
STAT: T4, ST:0068, ET:0079, RT:11, DT: 0126
TASK: T2
STAT: T2, ST:0090, ET:0096, RT:06, DT: 0102
TASK: T1
STAT: T1, ST:0096, ET:0101, RT:05, DT: 0120
TASK: T3
STAT: T3, ST:0101, ET:0110, RT:09, DT: 0138
TASK: T2
STAT: T2, ST:0120, ET:0126, RT:06, DT: 0132
TASK: T1
STAT: T1, ST:0126, ET:0131, RT:05, DT: 0144
TASK: T4
STAT: T4, ST:0131, ET:0138, RT:07, DT: 0189
TASK: T1
STAT: T1, ST:0144, ET:0148, RT:04, DT: 0168
TASK: T3
TASK: T2
STAT: T2, ST:0150, ET:0156, RT:06, DT: 0162
STAT: T3, ST:0148, ET:0164, RT:16, DT: 0186
TASK: T1
STAT: T1, ST:0168, ET:0172, RT:04, DT: 0192
TASK: T2
```

PART 4 -

- The scheduler task is augmented to have a scheduler overhead by adding a code snippet within the **prvSchedulerFunction** which utilizes the CPU for much longer time. The code snippet uses the **_delay_loop_2** function provided by the AVR library which just runs a loop on the CPU for the specified number of iterations (also called busy-waiting). By using this and calling the function for a specified number of ticks we can achieve scheduler overhead.

```
#if (schedOVERHEAD == 1)
    TickType_t xTicks = schedOVERHEAD_TICKS;
    while(xTicks--)
    {
        // Runs the CPU for 15 ms at FCLK = 16Mhz; 15 ms is WatchDog Timer Interrupt
        _delay_loop_2(60000);
    }
#endif
```

- By implementing the above code snippet and running the RM and DM Algorithm for Task Set 2, we can notice that some of the tasks are exceeding their maximum execution time and some tasks are missing their deadlines because the scheduler task is taking a significant amount of time to complete, thereby delaying the execution start time of the user defined tasks which are ready to run.

- Running Task Set 2 using RM Algorithm with Augmented Scheduler Overhead

```
---- Opened the serial port COM5 ----
----- Program Started -----
FUNC: vSchedulerInit
FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T1
Phase Tick          : 0
Max. Execution Tick : 6
Rel. Deadline Tick  : 24
Period Tick         : 24
Priority            : 0
----------------------

FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T2
Phase Tick          : 0
Max. Execution Tick : 9
Rel. Deadline Tick  : 12
Period Tick         : 30
Priority            : 0
----------------------

FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T3
Phase Tick          : 0
Max. Execution Tick : 12
Rel. Deadline Tick  : 42
Period Tick         : 48
Priority            : 0
----------------------

FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T4
Phase Tick          : 0
Max. Execution Tick : 9
Rel. Deadline Tick  : 63
Period Tick         : 63
Priority            : 0
----------------------

FUNC: vSchedulerStart
FUNC: prvSetFixedPriorities
----Using RM Scheduling Algorithm----
 Task : T1, Priority : 5, Tick : 24
 Task : T2, Priority : 4, Tick : 30
 Task : T3, Priority : 3, Tick : 48
 Task : T4, Priority : 2, Tick : 63
-------------------------------------

FUNC: prvCreateSchedulerTask
---- Scheduler Details ----
Period Tick  : 3
Priority     : 6
```

```
FUNC: prvCreateSchedulerTask
---- Scheduler Details ----
Period Tick  : 3
Priority     : 6
Overhead     : 1
--------------------------

FUNC: prvCreateAllTasks
FUNC: prvPeriodicTaskCode -> TASK: T1, INIT RUN
TASK: T1
FUNC: prvExecTimeExceedHook -> TASK: T1, T : 9
FUNC: prvPeriodicTaskCode -> TASK: T2, INIT RUN
TASK: T2
FUNC: prvDeadlineMissedHook -> TASK: T2, T : 15
FUNC: prvPeriodicTaskCode -> TASK: T2, INIT RUN
FUNC: prvPeriodicTaskCode -> TASK: T3, INIT RUN
TASK: T3
STAT: T1, ST:0001, ET:0026, RT:25, DT: 0024
FUNC: prvExecTimeExceedHook -> TASK: T3, T : 30
TASK: T2
FUNC: prvExecTimeExceedHook -> TASK: T2, T : 38
FUNC: prvPeriodicTaskCode -> TASK: T4, INIT RUN
TASK: T4
FUNC: prvDeadlineMissedHook -> TASK: T2, T : 45
FUNC: prvDeadlineMissedHook -> TASK: T3, T : 45
FUNC: prvPeriodicTaskCode -> TASK: T2, INIT RUN
FUNC: prvPeriodicTaskCode -> TASK: T3, INIT RUN
TASK: T1
FUNC: prvExecTimeExceedHook -> TASK: T1, T : 54
TASK: T3
FUNC: prvExecTimeExceedHook -> TASK: T3, T : 65
FUNC: prvDeadlineMissedHook -> TASK: T4, T : 65
FUNC: prvPeriodicTaskCode -> TASK: T4, INIT RUN
TASK: T4
STAT: T1, ST:0049, ET:0077, RT:28, DT: 0096
FUNC: prvExecTimeExceedHook -> TASK: T4, T : 79
TASK: T2
FUNC: prvDeadlineMissedHook -> TASK: T3, T : 93
TASK: T1
FUNC: prvExecTimeExceedHook -> TASK: T1, T : 102
FUNC: prvExecTimeExceedHook -> TASK: T2, T : 104
FUNC: prvDeadlineMissedHook -> TASK: T2, T : 104
FUNC: prvPeriodicTaskCode -> TASK: T2, INIT RUN
FUNC: prvPeriodicTaskCode -> TASK: T3, INIT RUN
STAT: T1, ST:0097, ET:0125, RT:28, DT: 0144
STAT: T4, ST:0067, ET:0134, RT:67, DT: 0189
TASK: T1
FUNC: prvExecTimeExceedHook -> TASK: T1, T : 150
TASK: T3
FUNC: prvExecTimeExceedHook -> TASK: T3, T : 162
STAT: T1, ST:0145, ET:0173, RT:28, DT: 0192
FUNC: prvDeadlineMissedHook -> TASK: T3, T : 189
FUNC: prvPeriodicTaskCode -> TASK: T3, INIT RUN
TASK: T4
TASK: T1
FUNC: prvExecTimeExceedHook -> TASK: T1, T : 198
```

- Running Task Set 2 using DM Algorithm with Augmented Scheduler Overhead

```
---- Opened the serial port COM5 ----
----- Program Started -----
FUNC: vSchedulerInit
FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T1
Phase Tick          : 0
Max. Execution Tick : 6
Rel. Deadline Tick  : 24
Period Tick         : 24
Priority            : 0
---------------------

FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T2
Phase Tick          : 0
Max. Execution Tick : 9
Rel. Deadline Tick  : 12
Period Tick         : 30
Priority            : 0
---------------------

FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T3
Phase Tick          : 0
Max. Execution Tick : 12
Rel. Deadline Tick  : 42
Period Tick         : 48
Priority            : 0
---------------------

FUNC: vSchedulerPeriodicTaskCreate
---- Task Details ----
Name                : T4
Phase Tick          : 0
Max. Execution Tick : 9
Rel. Deadline Tick  : 63
Period Tick         : 63
Priority            : 0
---------------------

FUNC: vSchedulerStart
FUNC: prvSetFixedPriorities
----Using DM Scheduling Algorithm----
 Task : T2, Priority : 5, Tick : 12
 Task : T1, Priority : 4, Tick : 24
 Task : T3, Priority : 3, Tick : 42
 Task : T4, Priority : 2, Tick : 63
-----------------------------------

FUNC: prvCreateSchedulerTask
---- Scheduler Details ----
Period Tick  : 3
Priority     : 6
```

```
FUNC: prvCreateSchedulerTask
---- Scheduler Details ----
Period Tick  : 3
Priority     : 6
Overhead     : 1
--------------------------

FUNC: prvCreateAllTasks
FUNC: prvPeriodicTaskCode -> TASK: T2, INIT RUN
TASK: T2
FUNC: prvExecTimeExceedHook -> TASK: T2, T : 14
FUNC: prvDeadlineMissedHook -> TASK: T2, T : 14
FUNC: prvPeriodicTaskCode -> TASK: T2, INIT RUN
FUNC: prvPeriodicTaskCode -> TASK: T1, INIT RUN
TASK: T1
FUNC: prvExecTimeExceedHook -> TASK: T1, T : 22
FUNC: prvPeriodicTaskCode -> TASK: T3, INIT RUN
TASK: T3
STAT: T1, ST:0016, ET:0029, RT:13, DT: 0048
TASK: T2
FUNC: prvExecTimeExceedHook -> TASK: T2, T : 37
FUNC: prvDeadlineMissedHook -> TASK: T2, T : 45
FUNC: prvDeadlineMissedHook -> TASK: T3, T : 45
FUNC: prvPeriodicTaskCode -> TASK: T2, INIT RUN
FUNC: prvPeriodicTaskCode -> TASK: T3, INIT RUN
FUNC: prvPeriodicTaskCode -> TASK: T4, INIT RUN
TASK: T4
TASK: T1
FUNC: prvExecTimeExceedHook -> TASK: T1, T : 54
TASK: T3
FUNC: prvExecTimeExceedHook -> TASK: T3, T : 65
FUNC: prvDeadlineMissedHook -> TASK: T4, T : 65
FUNC: prvPeriodicTaskCode -> TASK: T4, INIT RUN
TASK: T4
STAT: T1, ST:0049, ET:0077, RT:28, DT: 0096
FUNC: prvExecTimeExceedHook -> TASK: T4, T : 79
TASK: T2
FUNC: prvDeadlineMissedHook -> TASK: T3, T : 93
FUNC: prvExecTimeExceedHook -> TASK: T2, T : 98
TASK: T1
FUNC: prvExecTimeExceedHook -> TASK: T1, T : 104
FUNC: prvDeadlineMissedHook -> TASK: T2, T : 104
FUNC: prvPeriodicTaskCode -> TASK: T2, INIT RUN
FUNC: prvPeriodicTaskCode -> TASK: T3, INIT RUN
STAT: T1, ST:0100, ET:0125, RT:25, DT: 0144
STAT: T4, ST:0067, ET:0134, RT:67, DT: 0189
TASK: T1
FUNC: prvExecTimeExceedHook -> TASK: T1, T : 150
TASK: T3
FUNC: prvExecTimeExceedHook -> TASK: T3, T : 162
STAT: T1, ST:0145, ET:0173, RT:28, DT: 0192
FUNC: prvDeadlineMissedHook -> TASK: T3, T : 189
FUNC: prvPeriodicTaskCode -> TASK: T3, INIT RUN
TASK: T4
TASK: T1
FUNC: prvExecTimeExceedHook -> TASK: T1, T : 198
```

REFERENCES

1. Please read the README.md before getting started with the code
2. Help on Reading the Output ➖
    a. FUNC : <Function Name> refers to the function called
    b. TASK : <Task Name> refers to the task that started executing.
    c. STAT : <Task Name>, ST : <num>, ET : <num>, RT : <num>, DT : <num> shows the stats of the task that finished executing: -
        i. ST is the tick at which the task starts executing.
        ii. ET is the tick at which the task finished executing.
        iii. RT is the number of ticks that task took to complete (including how long the task was preempted, if it was preempted)
        iv. DT is the absolute deadline of the task in ticks.