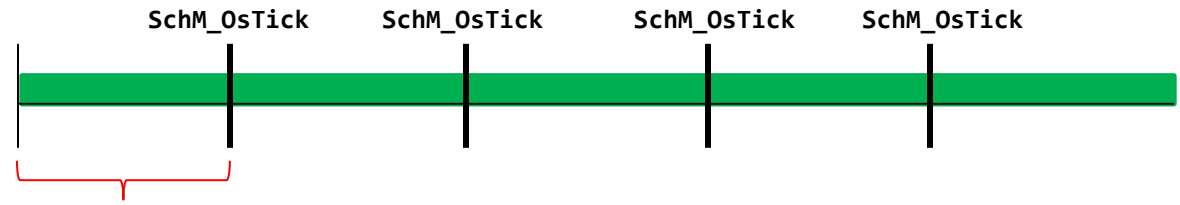
Students:
Olvera Contreras Jorge Alberto
Luis Eduardo Archundia Picazzo

RTOS SCHEDULER MODULE

Implementation

We implemented a Binary Progression Scheduler (BPS) that manages the access to the CPU recourses in a controlled way.



SysTick 781.25 micro-seconds

When there is not a Systick interrupt the **SchM_Background** function is executed

SchM_OsTick

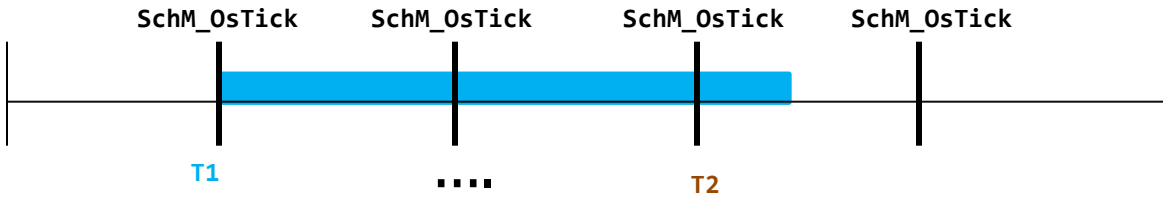
```
void SchM_OsTick( void )
{
//Counter (int) to tasks´ id
//Ostick counter increments
//Set Ready the Task if mask match counter
for(… ){ // Loop controlled by the number of tasks
if( ){    // if((Counter & Mask) == Offset)
// set the Task´s state in READY state
}
}
}
```

Students:
Olvera Contreras Jorge Alberto
Luis Eduardo Archundia Picazzo

SchM_Background

```
void SchM_Background( void )
{
//Counter (int) to tasks´ id
for(;;) {// Infinite loop
for(… ){ // Loop controlled by the number of tasks
if( ){    // if a Task is in READY state
//set the Task´s state in RUNNING state
//set the scheduler´s state in RUNNING state
//call the task (callback function)
//set the Task´s state in SUSPENDED state
//set the scheduler´s state in IDLE state
}
}
}
}
```

We added 2 bit flags to control and indicate when an overload occurs



((Counter & Maskt1) == Offsett1)

((Counter & Maskt2) == Offsett2)

OVERLOAD

```
typedef struct{
   uint8_t FlagOverLoad: 1;    //Flag for indicating that an Overload have occured
   uint8_t FlagTaskReady:1;    //Flag for indicating that there is a task in Ready/Running state
} Flags;
```

The system turns on a LED when overload occurs

Students:
Olvera Contreras Jorge Alberto
Luis Eduardo Archundia Picazzo

SchM_OsTick whit overload

```
void SchM_OsTick( void )
{
//Counter (int) to tasks´ id
//Ostick counter increments
//Set Ready the Task if mask match counter
for(… ){ // Loop controlled by the number of tasks
if( ){    // if((Counter & Mask) == Offset)
if(){ //Review if there is any Task in Running or Start State before activating other task (FlagTaskReady)
//Set the OverLoad Flag
//TurnOn the OVERLOADPIN  -> TurnOnOverloadPin();
}
// set the Task´s state in READY state
//Set the Flag that indicates the activation of a Task (FlagTaskReady=1)
}
}
}
```

Students:
Olvera Contreras Jorge Alberto
Luis Eduardo Archundia Picazzo

SchM_Background whit overload

```
void SchM_Background( void )
{
//Counter (int) to tasks´ id
for(;;) {// Infinite loop
for(… ){ // Loop controlled by the number of tasks
if( ){    // if a Task is in READY state
//set the Task´s state in RUNNING state
//set the scheduler´s state in RUNNING state
//call the task (callback function)
//set the Task´s state in SUSPENDED state
//Clear the Flag that indicates that there is one Task Activated (FlagTaskReady=0)
//set the scheduler´s state in IDLE state
}
}
}
}
```

Students:
Olvera Contreras Jorge Alberto
Luis Eduardo Archundia Picazzo

Testing the system (Tasks)

```
#define PINBKG      13  // Before: 9
#define PIN3P125MS  7
#define PIN6P25MS   17
#define PIN12P5MS   14
#define PIN25MS     15
#define PIN50MS     16
#define PIN100MS    3
#define PINOVERLOAD 14 // Before: 16
```

```
typedef enum
{
    PORTCH_A,
    PORTCH_B,
    PORTCH_C,
    PORTCH_D,
    PORTCH_E
}DioPortType;
```

```
void SchM_3p125ms_Task ( void )
{

    Dio_PortTooglePin(PORTCH_C, PIN3P125MS);
    //static int counter;
    //for(counter=0;counter <= NumberOfCycles; counter++){}

}
```

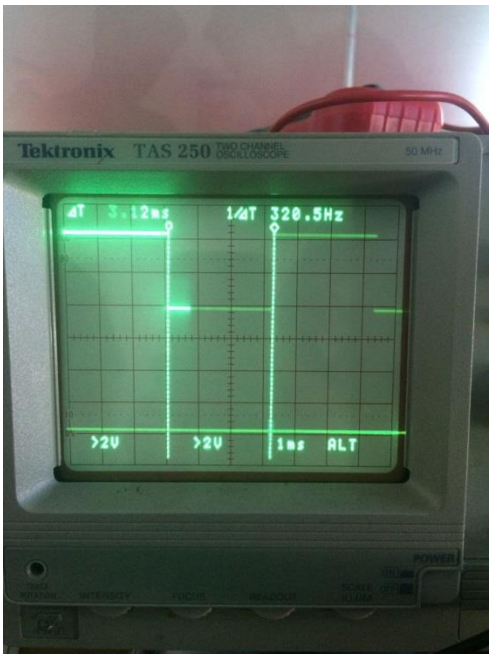
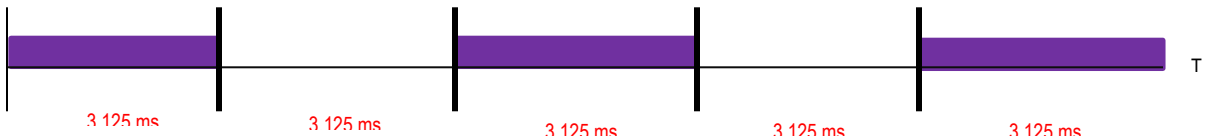


Ilustración 1. Task 3.125 ms

.

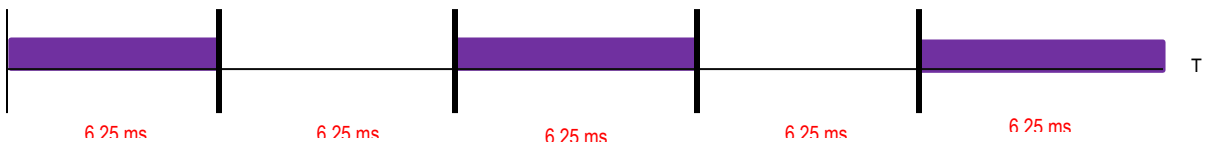


Students:
Olvera Contreras Jorge Alberto
Luis Eduardo Archundia Picazzo

```
void SchM_6p25ms_Task ( void )
{
    Dio_PortTooglePin(PORTCH_B, PIN6P25MS);
    //static int counter;
    //for(counter=0;counter <= NumberOfCycles; counter++){}

}
```

Ilustración 2. Task 6.25 ms

.



```
extern void SchM_12p5ms_Task    ( void ){
    Dio_PortTooglePin(PORTCH_B, PIN12P5MS);
    //static int counter;
    //for(counter=0;counter <= NumberOfCycles; counter++){}
}
```

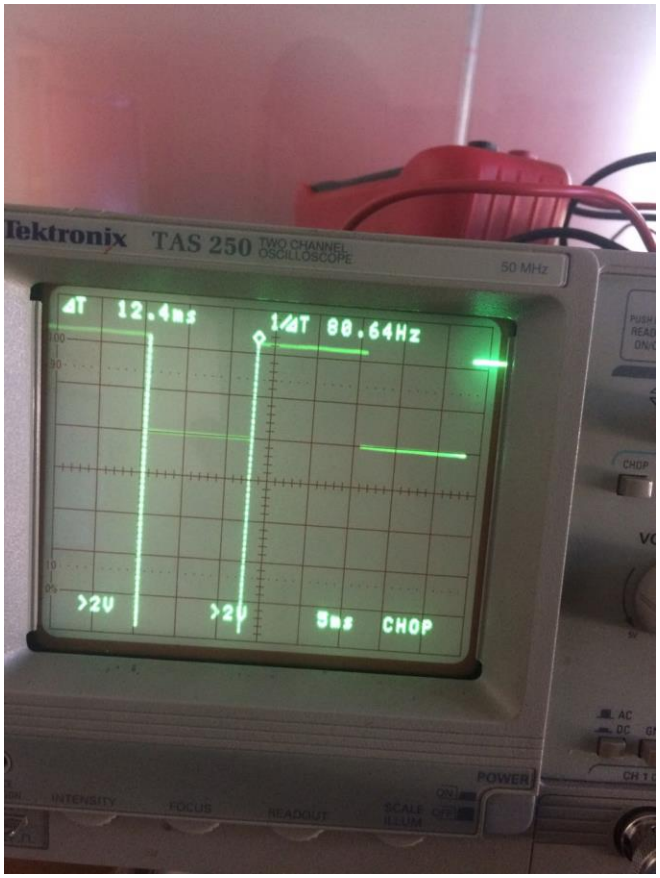
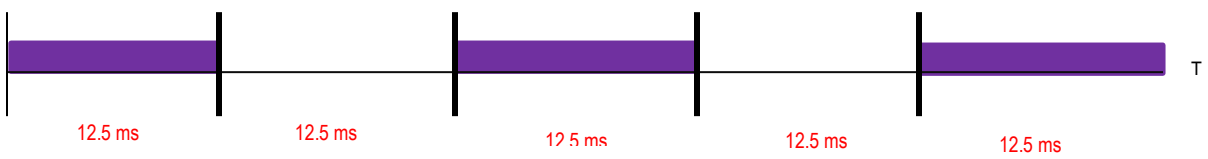


Ilustración 3. Task 12.5 ms

.



```
extern void SchM_25ms_Task     ( void ){
    Dio_PortTooglePin(PORTCH_B, PIN25MS);
    //static int counter;
    //for(counter=0;counter <= NumberOfCycles; counter++){}
}
```

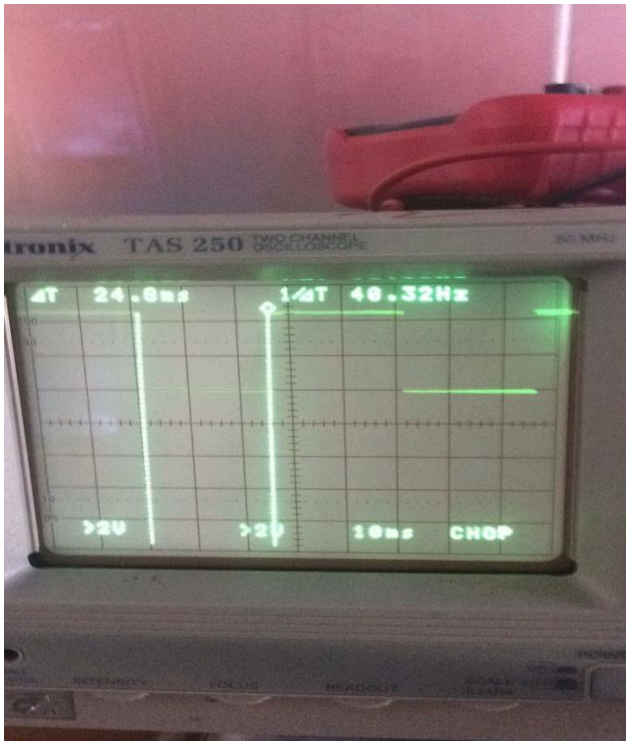
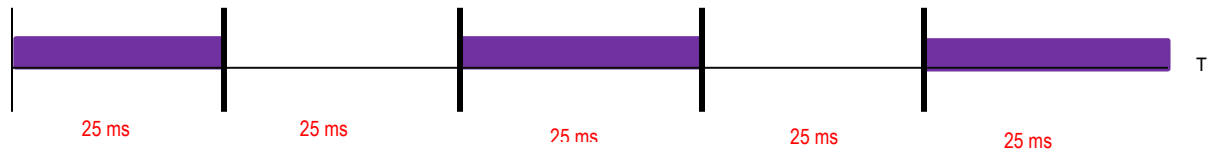


Ilustración 4. Task 25 ms

.



```
extern void SchM_50ms_Task      ( void ){
    Dio_PortTooglePin(PORTCH_B, PIN50MS);
    //static int counter;
    //for(counter=0;counter <= NumberOfCycles; counter++){}
}
```

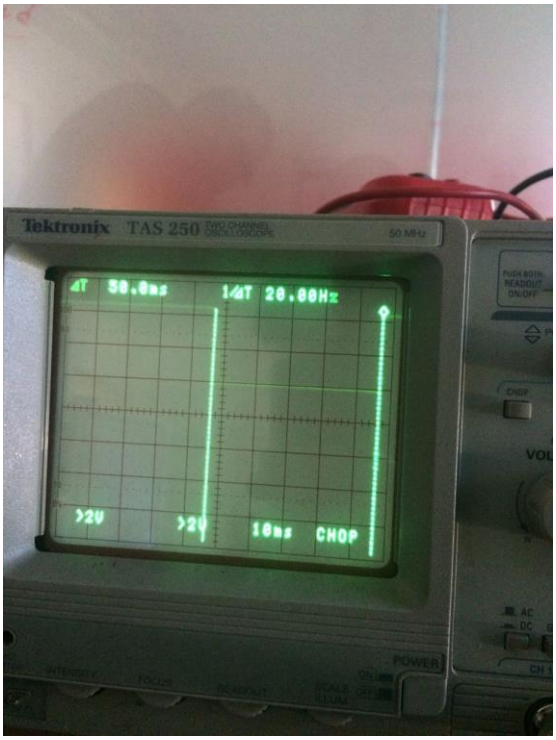
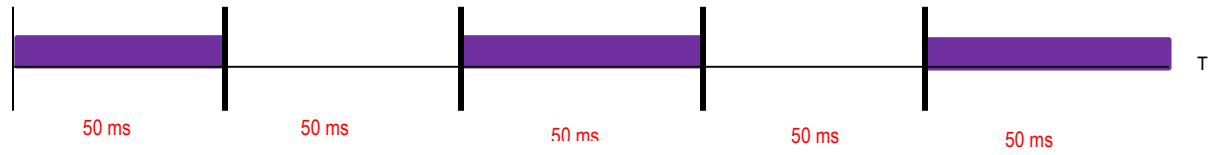


Ilustración 5. Task 50 ms



```
extern void SchM_100ms_Task     ( void ){
    Dio_PortTooglePin(PORTCH_C, PIN100MS);
    //static int counter;
    //for(counter=0;counter <= NumberOfCycles; counter++){}

}
```

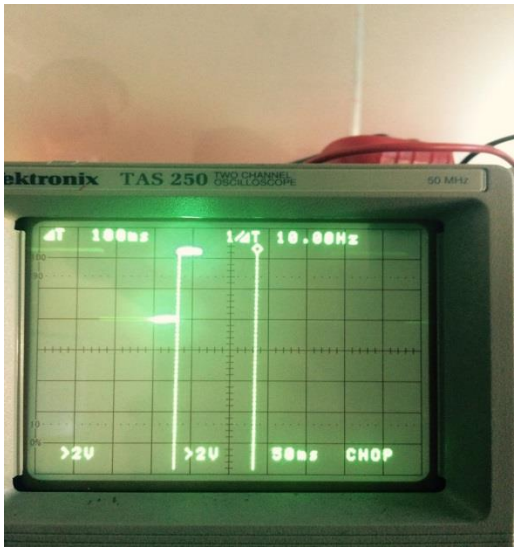
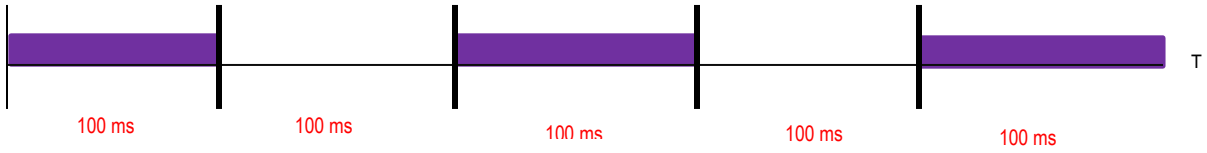


Ilustración 6. Task 100 ms

.



Students:
Olvera Contreras Jorge Alberto
Luis Eduardo Archundia Picazzo

Testing the system (CPU)

```
void SchM_<TaskPrefix>_Task ( void )
 {

       Dio_PortTooglePin(PORT, PIN);
       static int counter;
       for(counter=0;counter <= NumberOfCycles; counter++){}

}
```

```
void SchM_Background( void )
{
    uint8_t LocTaskIdx;
    for(;;)
    {
        for(LocTaskIdx = 0; LocTaskIdx < GlbSchMConfig->NumOfTasks; LocTaskIdx++)
        {
            if ( SCHM_TASK_STATE_READY == SchM_TaskControlBlock[LocTaskIdx].SchM_TaskState )
            {
                SchM_TaskControlBlock[LocTaskIdx].SchM_TaskState = SCHM_TASK_STATE_RUNNING;
                SchM_SchedulerStatus.SchM_SchedulerState = SCHM_RUNNING;
                TurnOffBackgroundPin(); //Turn off the pin that indicates when the Background fu
                GlbSchMConfig->TaskConfig[LocTaskIdx].TaskCallback();
                SchM_TaskControlBlock[LocTaskIdx].SchM_TaskState = SCHM_TASK_STATE_SUSPENDED;
                //Clear the Flag that indicates that there is one Task Activated.
                FlagsScheduler.FlagTaskState = 0 ;
                SchM_SchedulerStatus.SchM_SchedulerState = SCHM_IDLE;
                TurnOnBackgroundPin(); //Turn on the pin that indicates when the Background func
            }
        }
    }
}
```

Students:
Olvera Contreras Jorge Alberto
Luis Eduardo Archundia Picazzo

For this analysis we didn´t consider the instruction of **Toogle Dio_PortTooglePin(PORT,PIN)**

For this analysis when NumberOfCycles gets 10000 the time of the Task´s execution is 2800 microseconds.

When NumberOfCycles is configured whit 2700 the approximated time of the Task´s execution is 756 micros.

When NumberOfCycles is configured whit 0 the approximated time of the Task´s execution is 0 micros.

```
#define NumberOfCycles (uint32_t)2700
```


```
#define NumberOfCycles (uint32_t)0
```

Students:
Olvera Contreras Jorge Alberto
Luis Eduardo Archundia Picazzo

Testing the system (OVERLOAD)

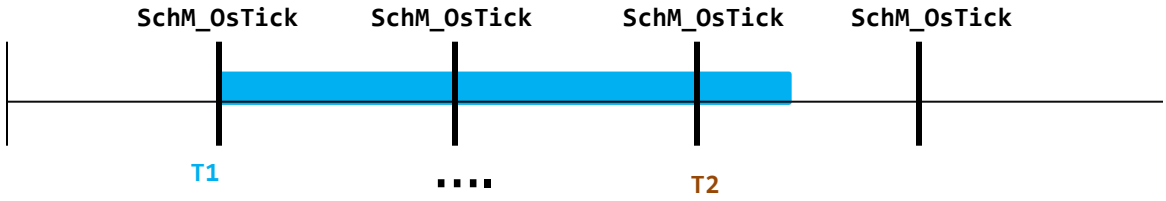
```
void SchM_<TaskPrefix>_Task ( void )
 {

      Dio_PortTooglePin(PORT, PIN);
      static int counter;
      for(counter=0;counter <= NumberOfCycles; counter++){}

}
```

```
#define NumberOfCycles (uint32_t) 3000
```

For this analysis we didn´t consider the instruction of **Toogle Dio_PortTooglePin(PORT,PIN)**

For this analysis when NumberOfCycles gets 10000 the time of the Task´s execution is 2800 micro seconds.

When NumberOfCycles is configured whit 3000 the approximated time of the Task´s execution is 840 micro seconds .



((Counter & Maskt1) == Offsett1)

((Counter & Maskt2) == Offsett2)

OVERLOAD