MI52 – Final

1h30. Documents allowed. You can answer in French or English.

1. Questions

1. Explain what is a system timer interrupt ?

2. Why does preemptive operating system needs a system timer interrupt ?

3. What does an operation system like FreeRTOS do when an system timer interrupt occurs ?

4. Is it better to have a high or low frequency system timer interrupt ?

5. How can you define a Task in a small operating system like FreeRTOS ?

6. When a FreeRTOS task call vTaskDelay(n), explain what FreeRTOS do.

7. In a virtual memory based system like Linux or Windows, what is the difference between a process and a thread ?

8. What is the difference between a kernel mode process and a user mode process ?

9. How can a user mode process becomes a kernel mode process ? Give an example in which this switch occurs ?

2. Exercise

An embedded application need to copy every X ms N 16 bits data from user memory to a single memory location to place the data on the pin of the SoC and set then clear a pin of the SoC. The system run with FreeRTOS.
The hardware timer counter TIM2 is used to generate interrupt request every X ms.
As the duration of the transfer is not negligible, this operation can't be done in the interrupt service routine. The operation will be performed in a dedicated task. It is chosen to synchronize the transfer task and the ISR with a binary semaphore.

Functions :

Timer :

`void vTIM2_isr ( void )`

Interrupt service routine for the timer. This ISR should clear the bit Nbr 4 of the TIM2_SR to acknowledge the interrupt request of the TIM2 interface and “de-block” the data transfer task.

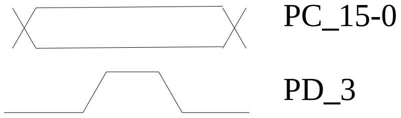
Task :

`void vData_transfer ( void *pvParameters ))`

this task must copy the N user data at USER_DATA_BUFFER address one by one. For each datum :

1. copy a data to the GPIOC (PC) output data register that control the 16 pins for the data
2. set the GPIOD pin 3
3. clear same pin.

All the pins of the GPIOC are used but only the pin Number 3 of the GPIOD is used. The Output data register of the GPIO are R/W registers.
Signal on the pins :



Address :

```
USER_DATA_BUFFER : 0x1000
GPIOC_ODR : 0x40020814
GPIOD_ODR : 0x40020C014
```

```
TIM2_SR : 0x40000010
```

FreeRTOS functions :

```
SemaphoreHandle_t xSemaphoreCreateBinary( void );
xSemaphoreTake( xSemaphoreHandle xSemaphore, TickType_t xTicksToWait );
xSemaphoreGive( xSemaphoreHandle xSemaphore ) ;
xSemaphoreGiveFromISR( SemaphoreHandle_t xSemaphore, signed BaseType_t
*pxHigherPriorityTaskWoken )
portEND_SWITCHING_ISR( xHigherPriorityTaskWoken );
```

1) Questions

1. In your opinion, why is it preferred to do the transfer in a dedicated task instead of the interrupt service routine ?

2. If the transfer of data is not done in the ISR, how can we start a transfer as quickly as possible in a multitask environment like here ?

3. Why should the isr aknowledge the interrupt request ?

4. Explain how to synchronize with the binary semaphore the ISR and the data transfer task

2) Code

Give the code (C) of the ISR and task. Do not forget to comment your code !