Gestão de tarefas com FreeRTOS

Objectivos

- Criação/suspensão de tarefas;
- Controlo de ciclo de uma tarefa;
- Escalonador preemptivo e não preemptivo;
- *Starvation* de uma tarefa;
- API do FreeRTOS.

Exercícios propostos

Criação de tarefas

1. Estudar e validar a implementação do projeto exemplo fornecido com o FreeRTOS;

2. Baseando-se no projeto exemplo fornecido, criar uma segunda tarefa recorrendo à função *xTaskCreate* para colocar o LED ligado no GPIOB1 a piscar a uma frequência de 0,5 Hz.

3. Recorrendo às funções *vTaskSuspend* e *vTaskResume* criar uma terceira tarefa que seja responsável por suspender e retomar a execução da tarefa 1 de 10 em 10 segundos. Pode obter um *delay* de 10 segundos da seguinte forma:

   ```
   vTaskDelay( ( TickType_t ) 10000 / portTICK_RATE_MS);
   ```

Controlo de ciclo

Nota: A tarefa 3 serviu apenas para estudar as funções *vTaskSuspend* e *vTaskResume*. Para evitar que esta tarefa corra, basta colocar como comentário a linha de código que cria essa tarefa (*xTaskCreate*).

4. Na tarefa 1, na linha imediatamente a seguir à mudança do estado do LED, acrescentar o seguinte ciclo (não esqueçer de declarar a variável `i`: `volatile uint16_t i;`):

```
for ( i=0; i < 65535; i++ )
{
}
```

   Sabendo que o objetivo do ciclo *for* é gastar tempo de CPU, analisar o resultado da sequência liga/desliga dos dois LED.
   Comente o resultado obtido.

5. Usando o exemplo anterior (mantendo o ciclo *for* na tarefa 1) troque o mecanismo de bloqueio *vTaskDelay* pelo *vTaskDelayUntil* para efetuar o controlo de ciclo da tarefa 1 e da tarefa 2.

   Não esquecer que a função *vTaskDelayUntil* tem um novo parâmetro de entrada que deve ser declarado e inicializado da seguinte forma:

```
static void prvFlashTask( void *pvParameters )
{
  /* Declare the variable xLastExecutionTime */
  TickType_t xLastExecutionTime;

  /* Initialize xLastExecutionTime */
  xLastExecutionTime = xTaskGetTickCount();
  for( ;; )
  {
   /* Block 1 second. */
   /* vTaskDelay((TickType_t)1000/portTICK_RATE_MS); comment this line */
   vTaskDelayUntil(&xLastExecutionTime, (TickType_t)1000/portTICK_RATE_MS );

    /* Toggle the LED */
    ...
  }
}
```

Comente o resultado obtido.

Escalonador do FreeRTOS

Nota: Para responder aos problemas seguintes colocar como comentário (ou apagar) o ciclo *for* inserido no código da implementação anterior. Este ciclo serviu apenas para gastar tempo de CPU por forma a evidenciar a diferença entre o *vTaskDelay* e o *vTaskDelayUntil*.

6. Garantir que as tarefas 1 e 2 do problema anterior correm a cada 10ms (em vez dos atuais 1000 ms). A tarefa 1 deve ser alterada por forma a enviar, através da USART disponível na placa de expansão, a seguinte *string*:

```
"Mensagen enviada ao fim de 10 ms.\r\n"
```

A tarefa 2, além de fazer a mudança do estado do LED (GPIOB1), deve enviar através da USART disponível na placa de expansão a seguinte *string*:

```
"O segundo LED mudou de estado agora mesmo.\r\n"
```

É possível usar a seguinte função já implementada no exemplo fornecido (ver também a função de configuração: *prvSetupUSART2*):

```
static void prvSendMessageUSART2( char *message )
```

Exemplo para a tarefa 1:

```
prvSendMessageUSART2( "Mensagen enviada ao fim de 10 ms.\r\n" );
```

Para analisar as mensagens que chegam à porta série do computador usar um programa do tipo: Hyperterminal, Realterm, Cutecom, etc. (configuração: 115200 bps, 8N1).
Comentar o resultado obtido.

7. Recorrendo ao exemplo da pergunta anterior, alterar a prioridade de uma das tarefas por forma a que seja superior à da outra tarefa.

   Para a tarefa 1 usar o valor de prioridade: tskIDLE_PRIORITY + 1

   Para a tarefa 2 usar o valor de prioridade: tskIDLE_PRIORITY + 2

   Alterar também a *string* das duas mensagens enviadas pela USART:

   ```
   "Mensagen enviada ao fim de 10 ms a partir de uma tarefa de
   baixa prioridade.\r\n"
   "O segundo LED mudou de estado agora mesmo a partir de uma
   tarefa de alta prioridade.\r\n"
   ```

   Comentar os resultados.

8. Recorrendo ainda à última aplicação desenvolvida na pergunta anterior, alterar o escalonador para que este seja não preemptivo. No *FreeRTOSConfig.h* alterar:

   ```
   #define configUSE_PREEMPTION 0
   ```

   Discutir as diferenças entre o resultado obtido e o resultado da implementação anterior.

Starvation **de uma tarefa**

Nota: A partir deste ponto colocar o escalonador preemptivo novamente.

9. Criar uma tarefa com prioridade mais elevada do que todas as outras, em que o conteúdo deverá ser apenas um ciclo infinito:

   ```
   for(;;)
   {
       while(1);
   }
   ```

   O que se verifica nesta situação em relação a todas as outras tarefas? O que se pode dizer relativamente ao uso da técnica de *pooling* de uma variável ou porto de I/0?

API do FreeRTOS

10. Criar uma tarefa que seja responsável por imprimir no *display* da placa de desenvolvimento o valor do *tick count* actual do escalonador e o número de tarefas a correr no FreeRTOS. Atualizar estes dados no *display* a cada 500 ms. Para efeitos de *debug* manter uma tarefa com o LED (GPIOB0) a piscar à frequência de 0,5 Hz.

    Por forma a saber que funções permitem aceder a estas variáveis do escalonador, recorrer à API do FreeRTOS disponível em www.freertos.org (*FreeRTOS* → *API Reference* → *Task Utilities*).

11. No seguimento da pergunta anterior, adicionar no *display* informação sobre o valor da prioridade e o nome das duas tarefas que estão a correr.
    De notar que poderá ser necessário incluir alguns parâmetros de configuração no ficheiro *FreeRTOSConfig.h*

12. Nem sempre é fácil fazer *debug* de um sistema multi-tarefa. Uma das informações relevantes em algumas situações, é, por exemplo, saber o estado em que se encontra cada tarefa. Existe uma função no FreeRTOS chamada de *vTaskList* que permite obter a lista de tarefas com essa informação. Pretende-se que seja desenvolvida uma tarefa que permita enviar, através da USART, a lista de tarefas a correr no sistema em questão. Ler a documentação da função *vTaskList* em:

    http://www.freertos.org/a00021.html#vTaskList

    Para validar esta funcionalidade criar três tarefas:

    - Tarefa 1 – LED pisca no GPIOB0;
    - Tarefa 2 – Atualizar o *tick count* no display de 500 em 500 ms
    - Tarefa 3 – Envio do resultado da função *vTaskList* através da USART.

    O resultado deverá ser algo idêntico à seguinte tabela:

```
Name    State   Priority    Stack    Num
task3   R       1           38       3
IDLE    R       0           106      4
task2   B       1           102      2
flash   B       1           102      1
```