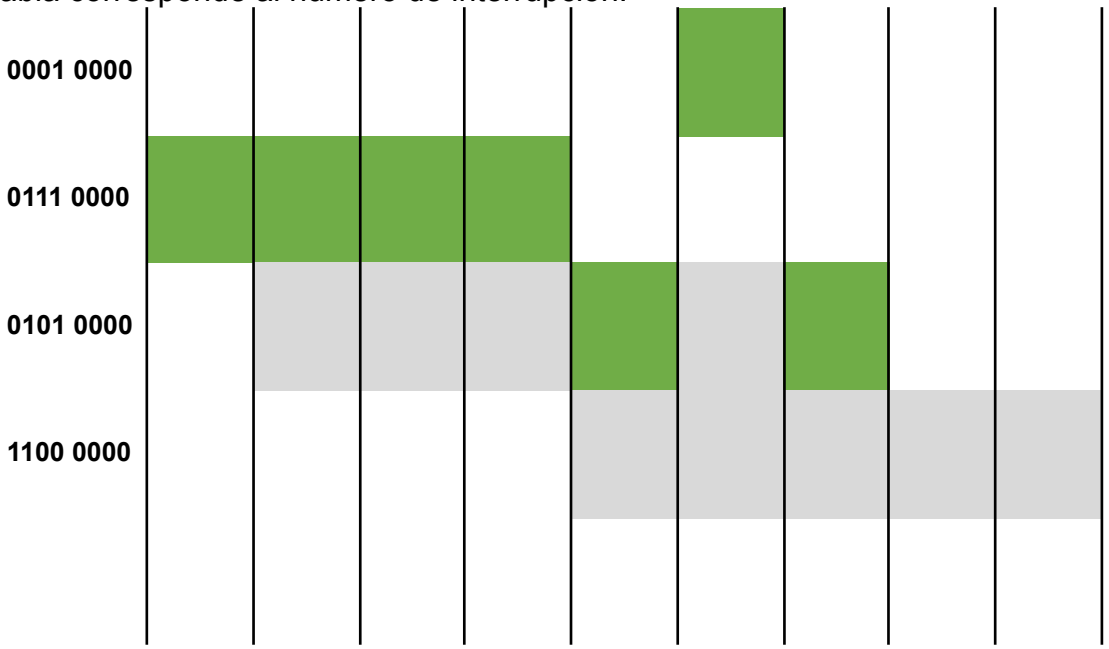
Práctica 1 - Parte 2

1. ¿Cual es la tasa de error del clock HSI? Si se intentan contar 300 segundos con un programa basado en el clock HSI a 8 MHz, ¿entre qué valores es esperable que cuente efectivamente?. Si en cambio se utiliza un cristal de la misma frecuencia con un error de 10 ppm ¿Qué tiempos resultan?

2. A partir del siguiente esquema de ejecución de interrupciones deduzca la configuración de los bits PRIGROUP del registro AIRCR y del registro BASEPRI. Los segmentos verdes significan que la interrupción se apropió de la ejecución y los grises que se pidió la interrupción pero no se apropió de la ejecución. La primera columna de la tabla corresponde al número de interrupción.



3. Cree un proyecto nuevo y siga los pasos enumerados a continuación para manipular un pin GPIO con herramientas de bajo nivel. Antes de continuar con el paso siguiente, verifique usando el debugger que se lograron realizar las acciones pedidas.

   i. Habilitar el clock del periférico que se desea utilizar. Esto se realiza mediante el hardware RCC (Reset and clock control) .

   a. Del *Reference Manual* obtenga la dirección de memoria base de los registros que configuran el RCC y defina una constante

      ```
      #define RCC_BASE …
      ```

   b. Averigüe el bus de periféricos (APB1, APB2) al que está conectado el periférico de interés y el offset del registro encargado de habilitar el clock para los perifericos de dicho bus. Defina un puntero constante a dicho registro:

      ```
      #define OFFSET_APBXENR …

      uint32_t *const RCC_APBXENR = RCC_BASE + OFFSET_APBXENR;
      ```

ii. Averigüe el bit del registro que habilita el clock para el periférico deseado y habilite dicho periférico poniendo el bit en 1:

```
#define BIT_PERFIFERICO …

*reg = *reg | (1 << BIT_PERIFERICO);
```

iii. Una vez habilitado el clock del periférico, puede configurar el periférico. Para un GPIO por ejemplo, debe configurar el bit como entrada o salida. Obtenga del *reference manual*:

a. La dirección base de los registros del GPIOx
b. El offset del registro de configuración correspondiente al pin que quiere configurar (GPIOx_CRL para pines 0 a 7 y GPIOx_CRH del 8 al 15)
c. los bits que deben configurarse para el pin de interés

iv. Una vez configurado puede manipular el periférico a través de sus registros. Para el GPIO puede usar los registros GPIOx_IDR, GPIOx_ODR y máscaras para leer/escribir un bit particular, que se corresponde directamente con el estado del pin.

v. Pruebe si efectivamente logró realizar estos pasos correctamente poniendo el pin en estado alto y bajo.

4. A partir de lo realizado en el ejercicio anterior, configure el pin conectado al LED de la placa blue pill como salida y escriba un código que lo haga destellar a 10 Hz. La espera de tiempo puede hacerse con un bucle vacío que itera una cantidad determinada de veces. Modularice el código para que el programa principal se vea de la siguiente manera:

```
#define ITER_ESPERA 400000
void main()
{
    configurar_led();

    while(1){
        prender_led();
        esperar(ITER_ESPERA);
        apagar_led();
        esperar(ITER_ESPERA);
    }

}
```

Utilice la ventana de depuración correspondiente para visualizar los registros de periféricos, ejecute paso a paso y observe la modificación de los registros utilizados.

Utilice el analizador lógico para ajustar la frecuencia del destello a 10 Hz (ciclo de trabajo del 50%).

5. A la consigna del ejercicio anterior agréguele un pulsador que se conectará a un GPIO configurado como entrada con pull-down interno.

   Modifique el código de tal modo que al estar el switch presionado el LED destelle a 1 Hz y al estar sin presionar lo haga a 10 Hz.

   Organice el código de manera modular, en dos funciones:

```
void configurar_switch()//configura la entrada

uint8_t leer_switch() //retorna el nivel de pin (0 o 1)
```

6. En un proyecto de prueba (puede ser en el que hizo el ejercicio 1) corra el siguiente código:

```
uint32_t r,x,y;
x = 7;
y = 0;
r = x/y;
```

   Ejecute utilizando el debugger y observe en qué rutina queda “atrapado”
   a) ¿Puede describir lo que ocurrió?
   b) Utilize el panel Fault Analyzer para observar las características de la falla. Considerando el tipo verdadero de falla, y viendo que es posible reconocerlo, ¿por qué terminó en una Hard Fault?
   c) Usando el panel de SFRs ubique, dentro del System Control Block, el Configurable Fault Satuts Register y el bit que se activó ante la falla.
   d) Repita el mismo procedimiento (y explique lo que pasa) para el código:

```
uint32_t *pz = 0x20005001;
uint32_t z = *pz;
```

7. Cuando se crea un proyecto en STM32CubeIDE el entorno utiliza la librería HAL e inicializa algunos periféricos por defecto. Por ejemplo, el SysTick del Cortex M3 se configura para lanzar una interrupción cada 1 ms.
   a) Encuentre las líneas donde se configura el systick, la prioridad de la interrupción, y se habilita la interrupción (pista: puede comenzar por la función HAL_Init() y rastrear desde ahí)
   b) La librería ya prepara los vectores de interrupción (como el del HardFault que examinó en el ejercicio anterior) asociados a funciones las cuales incluye en los archivos stm32f1xx_it .h y .c y les llama “Handlers”. Encuentre en el archivo .c correspondiente la función que maneja la interrupción del SysTick, coloque un breakpoint y observe el efecto.
   c) Pruebe las siguientes líneas de assembles y observe qué sucede ahora con el breakpoint:

```
asm("MOV R0,#1;");
asm("MSR PRIMASK,R0;");
```