Tema 10: Comunicación Serie Asíncrona: USART

SOLUCIÓN DE EJERCICIOS PROPUESTOS

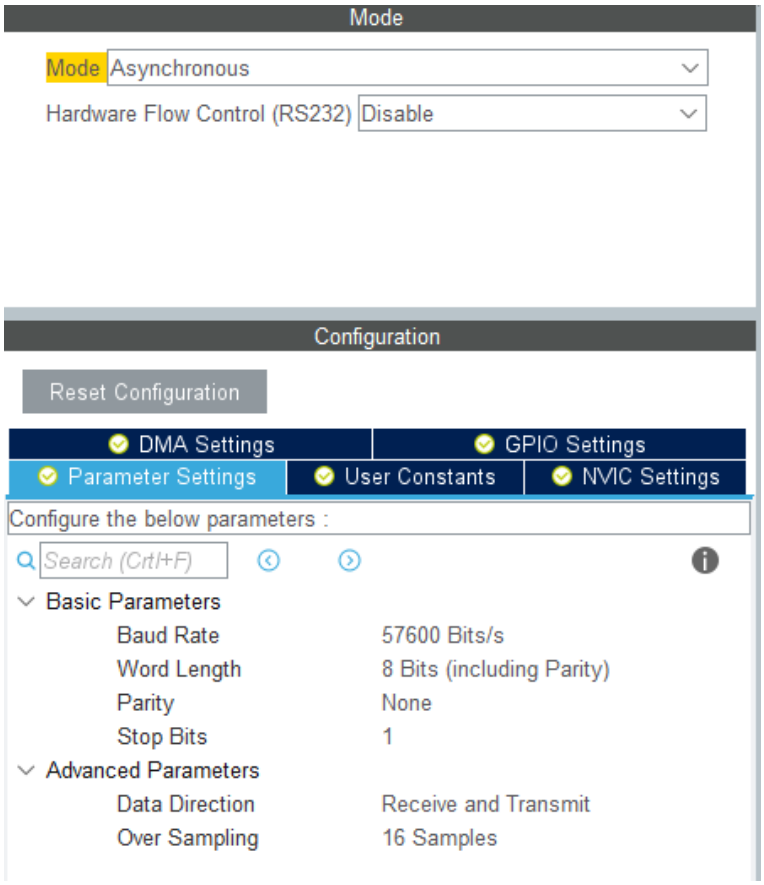
Ejercicio 1

Atendiendo a las especificaciones del enunciado, necesitamos:

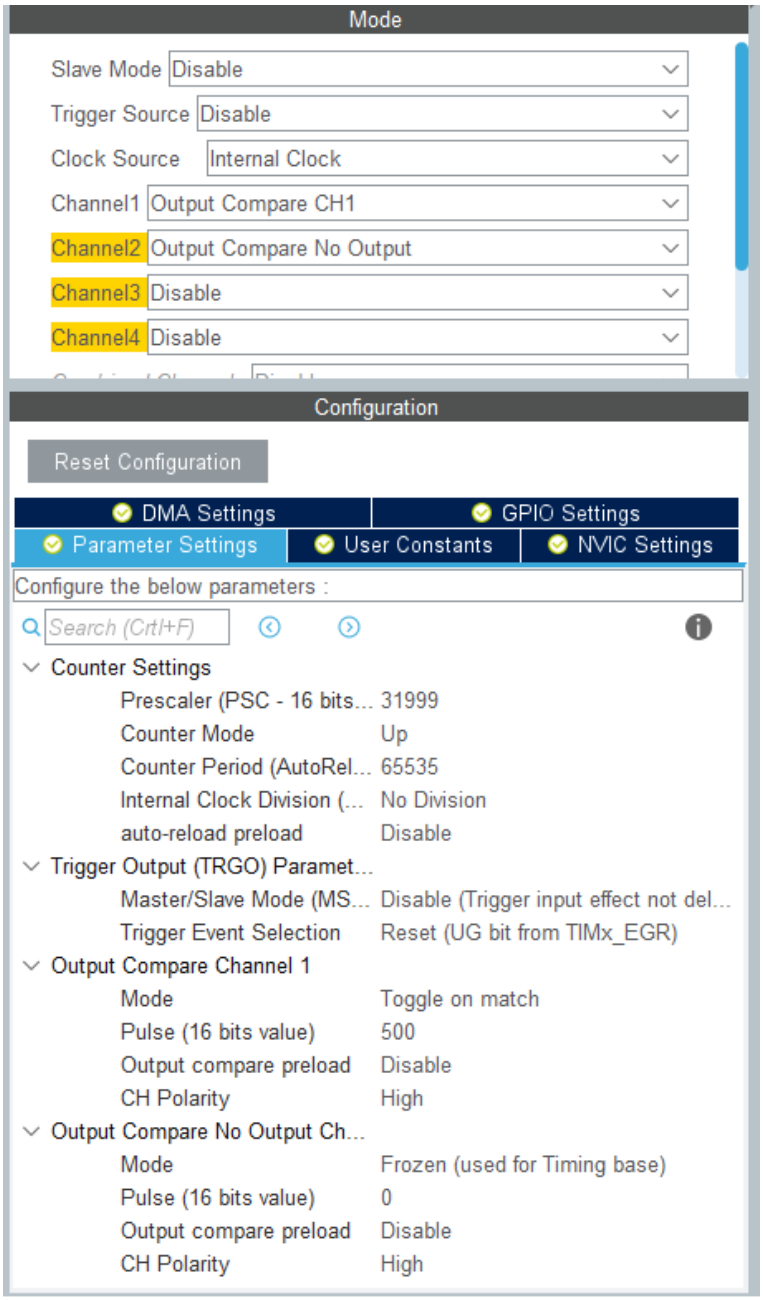
- Una comunicación con la USART, que podrá ser utilizada con los pines PB6 y PB7. USART1
- Un temporizador en modo TOC, con salida hardware, pudiendo ser esa salida el PA5. TIM2_CH1
- Un temporizador en modo TOC, sin salida hardware, para contar los 2 segundos. Usamos TIM2_CH2
- Un temporizador en modo TIC, con entrada por PA0 que tiene conectado otro temporizador (que es el TIM5) que no hemos visto en el curso. Pero de esta forma, vamos a ver lo fácil que supone utilizar otros periféricos utilizando la Abstracción Hardware. TIM5_CH1

Hay que tener en cuenta que, como medida de protección, las salidas hardware son inhibidas por las IRQs cuando se utilizan las HAL, por lo que habrá que re-activar la salida hardware del TIM2_CH1.

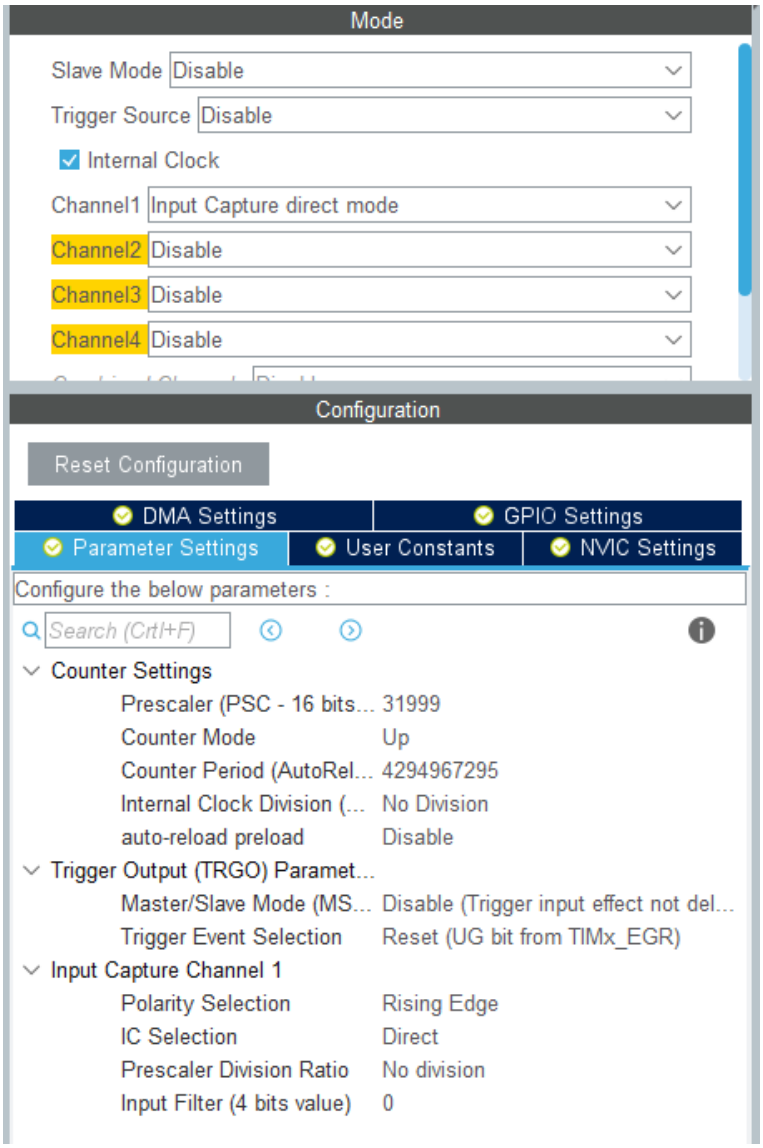
La configuración de la USART queda así (junto con la activación del USART1 global interrupt):



La configuración del TIM2 queda así (junto con la activación del TIM2 global interrupt):



Y la configuración del TIM5 queda así (junto con la activación del TIM5 global interrupt). Hay que tener en cuenta que es un temporizador de 32 bits (en lugar de 16) con un pre-escalado de 16 bits (esto se detecta fácilmente en la perspectiva CubeMX).



En cuanto al código, hay que hacer las siguientes aclaraciones:

- Todo se va a gestionar por interrupciones, por lo que el programa principal (en su bucle infinito), no va a tener contenido.
- No se necesita una callback para la transmisión, porque ya la RAI proporcionada por la HAL se encarga de todo (al tener claro el número de caracteres que se van a enviar).
- La callback de recepción se ejecuta cuando se hayan recibido todos los caracteres indicados (es decir, 4 o 5 dependiendo de su elección sobre el retorno de carro)
- Para permitir la activación de las salidas, se ha tenido que escribir directamente en el CCER del TIM2.
- No se ha realizado una gestión completa de errores en la trama recibida (se plantea que siempre sea correcta, es decir, siempre 3 números seguidos por un retorno de carro compuesto por CR+LF)

```
/* USER CODE BEGIN Includes */
#include "Utiles_SDM.h"
/* USER CODE END Includes */
```

```
/* USER CODE BEGIN PV */
unsigned short semiperiodo_TOC = 500;
```

```
unsigned periodo_TIC = 0;
unsigned short tiempo_inicio = 0;
uint8_t texto_out[8];
unsigned char texto_in[5];
TIM_OC_InitTypeDef my_sConfigOC1 = {0};
TIM_OC_InitTypeDef my_sConfigOC2 = {0};
/* USER CODE END PV */
```

```
/* USER CODE BEGIN 0 */

// Callback para la Rx de la USART
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
  // Traducción sin gestionar errores
  semiperiodo_TOC = 1000/((texto_in[0]-'0')*100 + (texto_in[1]-'0')*10 + (texto_in[2]-'0'));
  semiperiodo_TOC = semiperiodo_TOC /2;
  my_sConfigOC1.Pulse += semiperiodo_TOC;
  if (HAL_TIM_OC_ConfigChannel(&htim2, &my_sConfigOC1, TIM_CHANNEL_1) != HAL_OK) {
    Error_Handler();
  }
  //HAL_TIM_OC_Start_IT(&htim2, TIM_CHANNEL_1);
  HAL_UART_Receive_IT(&huart1, texto_in, 5); // Vuelve a activar Rx por haber acabado el buffer
}

// Callback para TOC del Timer2
void HAL_TIM_OC_DelayElapsedCallback(TIM_HandleTypeDef *htim) {
  htim2.Instance->CCER = 0x0001;
  if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1){
    my_sConfigOC1.Pulse += semiperiodo_TOC;
    if (HAL_TIM_OC_ConfigChannel(&htim2, &my_sConfigOC1, TIM_CHANNEL_1) != HAL_OK) {
      Error_Handler();
    }
  }
  else {
    my_sConfigOC2.Pulse += 2000;
    if (HAL_TIM_OC_ConfigChannel(&htim2, &my_sConfigOC2, TIM_CHANNEL_2) != HAL_OK) {
      Error_Handler();
    }
    Bin2Ascii((unsigned short)periodo_TIC,texto_out);
    texto_out[6]='\r';
    texto_out[7]='\n';
    HAL_UART_Transmit_IT(&huart1, texto_out, 8);
  }
}

// Callback para TIC del Timer5
void HAL_TIM_IC_CaptureCallback (TIM_HandleTypeDef * htim) {
  periodo_TIC = HAL_TIM_ReadCapturedValue(&htim5, TIM_CHANNEL_1);
  periodo_TIC = periodo_TIC - tiempo_inicio;  // El tiempo transcurrido es la resta de TIM2->CCR1
                                              // menos el tiempo de inicio, que inicialmente es 0
  if (periodo_TIC<0) periodo_TIC += 0x0FFFFFFF;  // Para evitar que de la vuelta al contador, le
doy
                                              // yo la vuelta y me aseguro que es correcta
  tiempo_inicio = HAL_TIM_ReadCapturedValue(&htim5, TIM_CHANNEL_1); // El nuevo tiempo inicio
}

/* USER CODE END 0 */
```

```
  /* USER CODE BEGIN 2 */
  // Init para TOC de generación de la señal
  my_sConfigOC1.OCMode = TIM_OCMODE_TOGGLE;
  my_sConfigOC1.Pulse = semiperiodo_TOC;
  my_sConfigOC1.OCPolarity = TIM_OCPOLARITY_HIGH;
  my_sConfigOC1.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_OC_ConfigChannel(&htim2, &my_sConfigOC1, TIM_CHANNEL_1) != HAL_OK) {
    Error_Handler();
  }
  // Init para TOC de 2 segundos
```

```
  my_sConfigOC2.OCMode = TIM_OCMODE_TIMING;
  my_sConfigOC2.Pulse = 2000;
  my_sConfigOC2.OCPolarity = TIM_OCPOLARITY_HIGH;
  my_sConfigOC2.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_OC_ConfigChannel(&htim2, &my_sConfigOC2, TIM_CHANNEL_2) != HAL_OK) {
    Error_Handler();
  }
  HAL_TIM_OC_Init(&htim2);
  HAL_TIM_OC_Start_IT(&htim2, TIM_CHANNEL_2);
  HAL_TIM_OC_Start_IT(&htim2, TIM_CHANNEL_1);
  htim2.Instance->CCER = 0x0001;
  // Init para TIC de medida del periodo
  HAL_TIM_IC_Init(&htim5);
  HAL_TIM_IC_Start_IT(&htim5, TIM_CHANNEL_1);
  // Init para USART en Rx
  HAL_UART_Receive_IT(&huart1, texto_in, 5);
  /* USER CODE END 2 */
```

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
```

En las pruebas, el Tera Term se ha configurado de la siguiente manera: