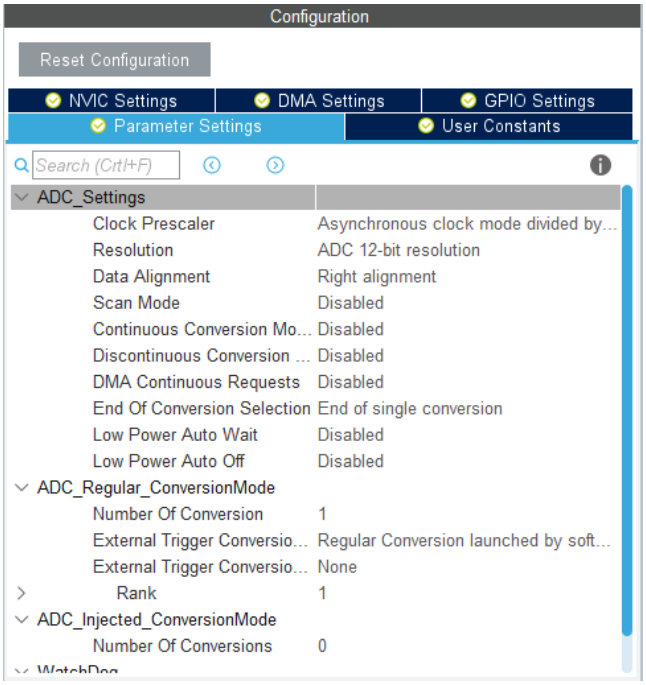
Tema 9: Uso de Abstracción Hardware

SOLUCIÓN DE EJERCICIOS PROPUESTOS

Ejercicio 1

En la perspectiva CubeMX se hace:

- Se pone en modo Reset el PC13
- Se asegura que IN4 está activo en el ADC
- Se configura en ADC



El código escrito quedaría así:

```
/* USER CODE BEGIN Includes */
#include "stm32l152c_discovery.h"
#include "stm32l152c_discovery_glass_lcd.h"
#include "Utiles_SDM.h"
/* USER CODE END Includes */
```

```
  /* USER CODE BEGIN 1 */
  unsigned short valor;
  uint8_t texto[7];
  /* USER CODE END 1 */
```

```
  /* USER CODE BEGIN 2 */
  BSP_PB_Init(BUTTON_USER, BUTTON_MODE_GPIO);
  BSP_LCD_GLASS_Init();
  BSP_LCD_GLASS_BarLevelConfig(0);
  BSP_LCD_GLASS_Clear();
  /* USER CODE END 2 */
```

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    if (BSP_PB_GetState(BUTTON_USER) == 1) {      // Button pressed
      HAL_ADC_Start(&hadc);
      HAL_ADC_PollForConversion(&hadc, 10000);
      valor = HAL_ADC_GetValue(&hadc);
      Bin2Ascii(valor, texto);
      BSP_LCD_GLASS_DisplayString(texto);
      while (BSP_PB_GetState(BUTTON_USER)==1);
    }
    /* USER CODE END WHILE */
```

Ejercicio 2

La configuración en la perspectiva CubeMX es la misma que en el ejercicio anterior, pero cambiando la opción de *Continuous Conversion* a *Enabled*.

Sin embargo, la biblioteca HAL no hace una gran distinción en su uso entre el modo de conversión simple y el de conversión continua, como se puede ver en el código siguiente:

```
/* USER CODE BEGIN Includes */
#include "stm32l152c_discovery.h"
#include "stm32l152c_discovery_glass_lcd.h"
#include "Utiles_SDM.h"
/* USER CODE END Includes */
```

```
  /* USER CODE BEGIN 1 */
  unsigned short valor;
  uint8_t texto[7];
  /* USER CODE END 1 */
```

```
  /* USER CODE BEGIN 2 */
  BSP_LCD_GLASS_Init();
  BSP_LCD_GLASS_BarLevelConfig(0);
  BSP_LCD_GLASS_Clear();
  /* USER CODE END 2 */
```

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    HAL_ADC_Start(&hadc);
    HAL_ADC_PollForConversion(&hadc, 10000);
    valor = HAL_ADC_GetValue(&hadc);
    Bin2Ascii(valor, texto);
    BSP_LCD_GLASS_DisplayString(texto);
    /* USER CODE END WHILE */
```

Ejercicio 3

En la perspectiva CubeMX la configuración es muy sencilla, puesto que sólo se necesita seleccionar el DAC y activar la OUT2 Configuration. El código queda de la siguiente forma:

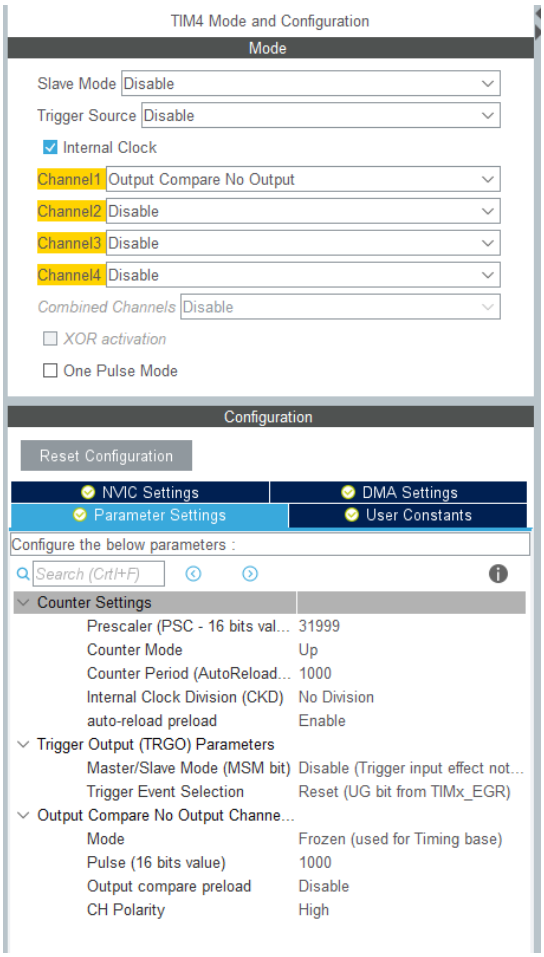
```
  /* USER CODE BEGIN 1 */
  unsigned short i = 0;
  unsigned short onda[16] = {0, 2, 4, 8, 16, 32, 64, 128, 255, 128, 64, 32, 16, 8, 4, 2};
  /* USER CODE END 1 */
```

```
  /* USER CODE BEGIN 2 */
  HAL_DAC_Start(&hdac, DAC_CHANNEL_2);
  /* USER CODE END 2 */
```

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    for (i=0; i<16; i++) {
      HAL_DAC_SetValue(&hdac,DAC_CHANNEL_2, DAC_ALIGN_8B_R,onda[i]);
      HAL_Delay(1000);
    }
    /* USER CODE END WHILE */
```

Ejercicio 4

Este ejemplo trata, simplemente de medir de forma continua un determinado periodo de tiempo. Esta funcionalidad, se puede hacer tal cual se ha comentado en el tema de Temporizadores, pero también se puede hacer utilizando el temporizador en modo pre-carga, al sólo estar utilizando un único canal de ese temporizador. Esa funcionalidad se denomina Time Base. Esto implica configurar el temporizador tal y como está puesto en la figura:



Y además, el código sería el siguiente:

```
/* USER CODE BEGIN Includes */
#include "stm32l152c_discovery.h"
#include "stm32l152c_discovery_glass_lcd.h"
#include "Utiles_SDM.h"
/* USER CODE END Includes */
```

```
  /* USER CODE BEGIN 1 */
```

```
  unsigned short numero=0;
  uint8_t texto[7];
  /* USER CODE END 1 */
```

```
  /* USER CODE BEGIN 2 */
  BSP_LCD_GLASS_Init();
  BSP_LCD_GLASS_BarLevelConfig(0);
  BSP_LCD_GLASS_Clear();
  HAL_TIM_Base_Init(&htim4);
  HAL_TIM_Base_Start(&htim4);
  /* USER CODE END 2 */
```

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
     while (!__HAL_TIM_GET_FLAG(&htim4, TIM_FLAG_CC1));
    __HAL_TIM_CLEAR_FLAG(&htim4, TIM_FLAG_CC1);
    numero++;
    Bin2Ascii(numero, texto);
    BSP_LCD_GLASS_Clear();
    BSP_LCD_GLASS_DisplayString(texto);
   /* USER CODE END WHILE */
```

Ejercicio 5

La configuración de la perspectiva CubeMX es la misma que en el ejercicio 4, pero añadiendo el seleccionar el ***TIM4 global interrupt***, en la pestaña de ***NVIC Settings*** del TIM4.

El código quedaría así:

```
/* USER CODE BEGIN Includes */
#include "stm32l152c_discovery.h"
#include "stm32l152c_discovery_glass_lcd.h"
#include "Utiles_SDM.h"
/* USER CODE END Includes */
```

```
/* USER CODE BEGIN PV */
unsigned short numero=0;
/* USER CODE END PV */
```

```
/* USER CODE BEGIN 0 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim) {
  numero++;
}
/* USER CODE END 0 */
```

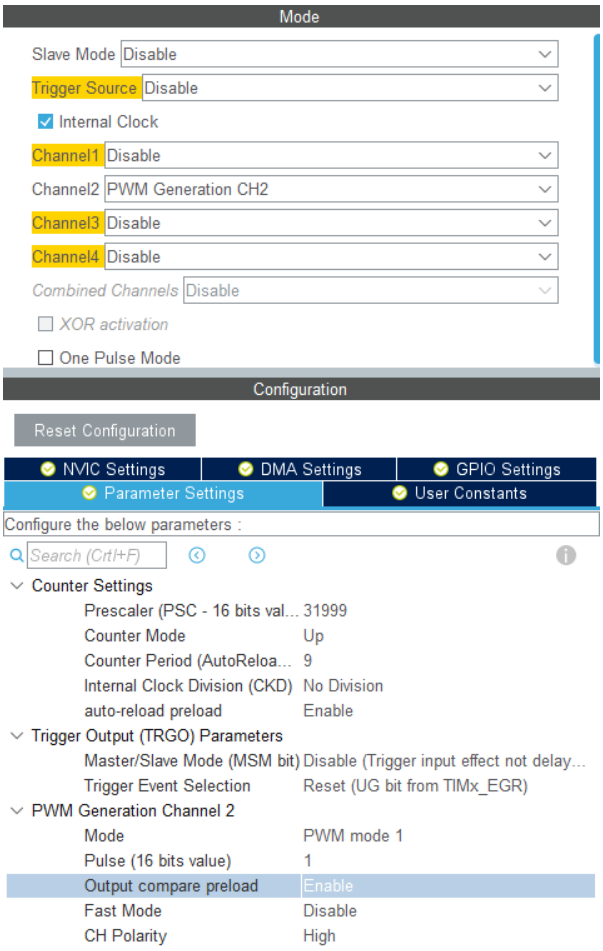
```
  /* USER CODE BEGIN 1 */
  uint8_t texto[7];
  /* USER CODE END 1 */
```

```
  /* USER CODE BEGIN 2 */
  BSP_LCD_GLASS_Init();
  BSP_LCD_GLASS_BarLevelConfig(0);
  BSP_LCD_GLASS_Clear();
  HAL_TIM_Base_Init(&htim4);
  HAL_TIM_Base_Start_IT(&htim4);
  /* USER CODE END 2 */
```

```
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    Bin2Ascii(numero, texto);
    BSP_LCD_GLASS_DisplayString(texto);
    /* USER CODE END WHILE */
```

Ejercicio 6

Para la generación de una señal PWM, la configuración de la perspectiva CubeMX es como la mostrada en la figura (habiendo marcado el PB7 como TIM4_CH2 en el dibujo del pinout):



Y el código sería:

```
/* USER CODE BEGIN Includes */
#include "stm32l152c_discovery.h"
#include "stm32l152c_discovery_glass_lcd.h"
#include "Utiles_SDM.h"
/* USER CODE END Includes */
```

```
  /* USER CODE BEGIN 1 */
  unsigned short DC=1;
  uint8_t texto[7];
  TIM_OC_InitTypeDef my_sConfigOC = {0};
  /* USER CODE END 1 */
```

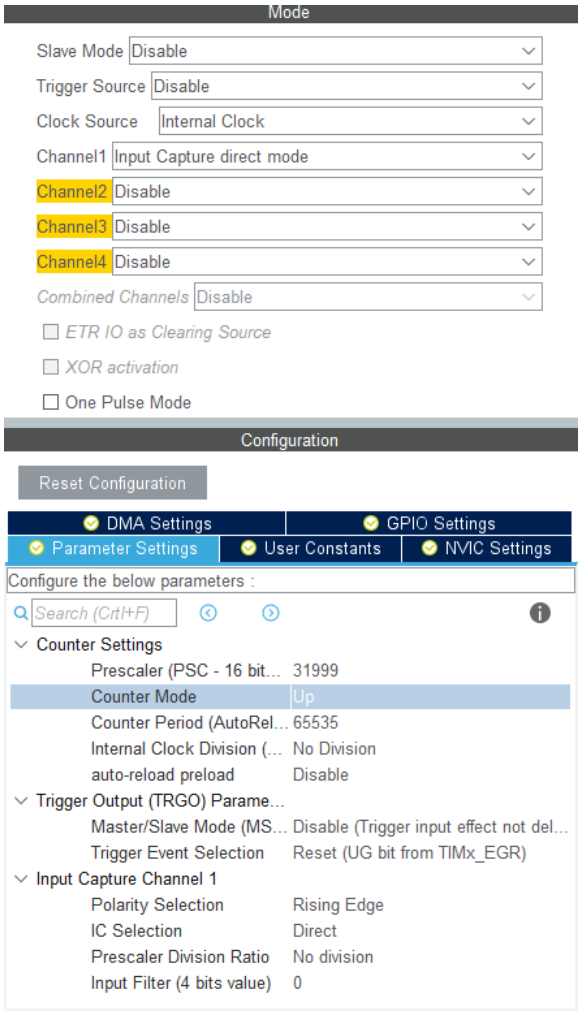
```
  /* USER CODE BEGIN 2 */
  BSP_PB_Init(BUTTON_USER, BUTTON_MODE_GPIO);
  BSP_LCD_GLASS_Init();
  BSP_LCD_GLASS_BarLevelConfig(0);
  BSP_LCD_GLASS_Clear();
  my_sConfigOC.OCMode = TIM_OCMODE_PWM1;
  my_sConfigOC.Pulse = DC;
  my_sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  my_sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_PWM_ConfigChannel(&htim4, &my_sConfigOC, TIM_CHANNEL_2) != HAL_OK) {
    Error_Handler();
  }
  HAL_TIM_PWM_Init(&htim4);
```

```
HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_2);
/* USER CODE END 2 */
```

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  if (BSP_PB_GetState(BUTTON_USER) == 1) {      // Button pressed
    DC++;
    if (DC>=9) DC=1;
    my_sConfigOC.Pulse = DC;
    if (HAL_TIM_PWM_ConfigChannel(&htim4, &my_sConfigOC, TIM_CHANNEL_2) != HAL_OK) {
      Error_Handler();
    }
    HAL_TIM_PWM_Init(&htim4);
    HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_2);
    Bin2Ascii(DC, texto);
    BSP_LCD_GLASS_DisplayString(texto);
    while (BSP_PB_GetState(BUTTON_USER)==1);
  }
  /* USER CODE END WHILE */
```

Ejercicio 7

La configuración del TIC en la perspectiva CubeMX se hace de la siguiente forma (habiendo marcado el PA0 como TIM2_CH1 en el dibujo del pinout):



Y el código, sin interrupciones, quedaría así:

```
/* USER CODE BEGIN Includes */
#include "stm32l152c_discovery.h"
#include "stm32l152c_discovery_glass_lcd.h"
#include "Utiles_SDM.h"
/* USER CODE END Includes */
```

```
  /* USER CODE BEGIN 1 */
  unsigned char texto[7];
  unsigned short tiempo_inicio = 0;
  int tiempo;
  /* USER CODE END 1 */
```

```
  /* USER CODE BEGIN 2 */
  BSP_LCD_GLASS_Init();
  BSP_LCD_GLASS_BarLevelConfig(0);
  BSP_LCD_GLASS_Clear();
  HAL_TIM_IC_Init(&htim2);
  HAL_TIM_IC_Start(&htim2, TIM_CHANNEL_1);
  /* USER CODE END 2 */
```

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    while (!__HAL_TIM_GET_FLAG(&htim2, TIM_FLAG_CC1));
    __HAL_TIM_CLEAR_FLAG(&htim2, TIM_FLAG_CC1);
    tiempo = __HAL_TIM_GET_COMPARE(&htim2, TIM_CHANNEL_1);
    tiempo = tiempo - tiempo_inicio;  // El tiempo transcurrido es la resta de TIM2->CCR1
                                      // menos el tiempo de inicio, que inicialmente es 0
    if (tiempo<0) tiempo += 0x0FFFF;  // Para evitar que de la vuelta al contador, le doy
                                      // yo la vuelta y me aseguro que es correcta
    tiempo_inicio = __HAL_TIM_GET_COMPARE(&htim2, TIM_CHANNEL_1); // El nuevo tiempo inicio
    Bin2Ascii((unsigned short)(tiempo), texto);  // Convierto el número a texto
    BSP_LCD_GLASS_Clear();
    BSP_LCD_GLASS_DisplayString(texto); // Saco el texto por el LCD CD
    /* USER CODE END WHILE */
```

Ejercicio 8

Basándose en el Ejercicio 7, la única diferencia en la perspectiva CubeMX es la activación de la casilla ***TIM2 global interrupt***, dentro de la pestaña ***NVIC Settings*** del TIM2. Con eso quedaría el código de la siguiente manera:

```
/* USER CODE BEGIN Includes */
#include "stm32l152c_discovery.h"
#include "stm32l152c_discovery_glass_lcd.h"
#include "Utiles_SDM.h"
/* USER CODE END Includes */
```

```
/* USER CODE BEGIN PV */
unsigned short tiempo_inicio = 0;
int tiempo;
/* USER CODE END PV */
```

```
/* USER CODE BEGIN 0 */
void HAL_TIM_IC_CaptureCallback (TIM_HandleTypeDef * htim) {
  tiempo = __HAL_TIM_GET_COMPARE(&htim2, TIM_CHANNEL_1);
  tiempo = tiempo - tiempo_inicio;  // El tiempo transcurrido es la resta de TIM2->CCR1
                                    // menos el tiempo de inicio, que inicialmente es 0
  if (tiempo<0) tiempo += 0x0FFFF;  // Para evitar que de la vuelta al contador, le doy
                                    // yo la vuelta y me aseguro que es correcta
  tiempo_inicio = __HAL_TIM_GET_COMPARE(&htim2, TIM_CHANNEL_1); // El nuevo tiempo inicio
```

```
}
/* USER CODE END 0 */
```

```
  /* USER CODE BEGIN 1 */
  unsigned char texto[7];
  /* USER CODE END 1 */
```

```
  /* USER CODE BEGIN 2 */
  BSP_LCD_GLASS_Init();
  BSP_LCD_GLASS_BarLevelConfig(0);
  BSP_LCD_GLASS_Clear();
  HAL_TIM_IC_Init(&htim2);
  HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_1);
  /* USER CODE END 2 */
```

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    Bin2Ascii((unsigned short)(tiempo), texto);  // Convierto el número a texto
    BSP_LCD_GLASS_DisplayString(texto); // Saco el texto por el LCD CD
    /* USER CODE END WHILE */
```