Tema 5: Pines de Entrada/Salida de Propósito General

SOLUCIÓN DE EJERCICIOS PROPUESTOS

Ejercicio 1

Realizamos la siguiente implementación y posteriormente la comentamos:

```
  /* USER CODE BEGIN 1 */
  unsigned char estado = 0;
  /* USER CODE END 1 */
```

```
  /* USER CODE BEGIN 2 */

  // USER button configuration (PA0)
  // PA0 as digital input(00)
  GPIOA->MODER &= ~(1 << (0*2 +1));
  GPIOA->MODER &= ~(1 << (0*2));

  // PB6 (LED Azul) como salida digital (01)
  GPIOB->MODER &= ~(1 << (6*2 +1)); // 0 en el bit deseado = AND de MODER con el inverso de un
                                    // "1" en la posición
                                    // 13 y el resto "0" (1 << (6*2 +1) -> "1" desplazado 13
                                    // veces desde la derecha)
  GPIOB->MODER |= (1 << (6*2));     // 1 en el bit deseado = OR de MODER con un "1" en la
                                    // posición 12 y el resto "0".
                                    // (1 << (6*2)) -> "1" desplazado 12 veces desde la derecha)

  // PB7 (LED Verde) como salida digital (01)
  GPIOB->MODER &= ~(1 << (7*2 +1));
  GPIOB->MODER |= (1 << (7*2));

  /* USER CODE END 2 */
```

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1) {
    if ((GPIOA->IDR&0x00000001)!=0) { // Si PA0 = 1 (USER pressed)
                                      // incrementa estado
      estado++;
      if (estado>3) estado = 0;
      switch (estado) {
        case 0: // Enciende el Led Verde y no el Led Azul
          GPIOB->BSRR = (1<<7);
          GPIOB->BSRR = (1<<6)<<16;
          break;
        case 1: // Enciende el Led Verde y el Led Azul
          GPIOB->BSRR = (1<<7);
          GPIOB->BSRR = (1<<6);
          break;
        case 2: // Enciende el Led Azul y no el Led Verde
          GPIOB->BSRR = (1<<7)<<16;
          GPIOB->BSRR = (1<<6);
          break;
        default: // Apaga el Led Verde y el Led Azul
          GPIOB->BSRR = (1<<7)<<16;
          GPIOB->BSRR = (1<<6)<<16;
          break;
      }
```

```
  }
  /* USER CODE END WHILE */
```

Se modifica el valor de los pines de forma, aparentemente, aleatoria. Realmente lo que ocurre es que los valores de los LEDs van cambiando continuamente mientras está pulsado el botón, por lo que no se tiene ningún control respecto al número de veces que cambia.

Ejercicio 2

Para que sólo se ejecute una única vez con cada pulsación, ponemos una espera hasta que se suelte el botón. La modificación del código está marcada en amarillo

```
/* USER CODE BEGIN 1 */
unsigned char estado = 0;
/* USER CODE END 1 */
```

```
/* USER CODE BEGIN 2 */
// USER button configuration (PA0)
// PA0 as digital input(00)
GPIOA->MODER &= ~(1 << (0*2 +1));
GPIOA->MODER &= ~(1 << (0*2));

// PB6 (LED Azul) como salida digital (01)
GPIOB->MODER &= ~(1 << (6*2 +1)); // 0 en el bit deseado = AND de MODER con el inverso de un
                                  // "1" en la posición
                                  // 13 y el resto "0" (1 << (6*2 +1) -> "1" desplazado 13
                                  // veces desde la derecha)
GPIOB->MODER |= (1 << (6*2));     // 1 en el bit deseado = OR de MODER con un "1" en la
                                  // posición 12 y el resto "0".
                                  // (1 << (6*2)) -> "1" desplazado 12 veces desde la derecha)

// PB7 (LED Verde) como salida digital (01)
GPIOB->MODER &= ~(1 << (7*2 +1));
GPIOB->MODER |= (1 << (7*2));
/* USER CODE END 2 */
```

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1) {
  if ((GPIOA->IDR&0x00000001)!=0) { // Si PA0 = 1 (USER pressed)
                                    // incrementa estado
    estado++;
    if (estado>3) estado = 0;
    switch (estado) {
      case 0: // Enciende el Led Verde y no el Led Azul
        GPIOB->BSRR = (1<<7);
        GPIOB->BSRR = (1<<6)<<16;
        break;
      case 1: // Enciende el Led Verde y el Led Azul
        GPIOB->BSRR = (1<<7);
        GPIOB->BSRR = (1<<6);
        break;
      case 2: // Enciende el Led Azul y no el Led Verde
        GPIOB->BSRR = (1<<7)<<16;
        GPIOB->BSRR = (1<<6);
        break;
      default: // Apaga el Led Verde y el Led Azul
        GPIOB->BSRR = (1<<7)<<16;
        GPIOB->BSRR = (1<<6)<<16;
        break;
    }
    while ((GPIOA->IDR&0x00000001)!=0) { // Si PA0 = 1 (USER pressed),
                                         // espera hasta soltar el boton
        }
```

```
    }
    /* USER CODE END WHILE */
```

Dependiendo del tipo de botón utilizado, pueden provocarse rebotes al soltar el botón. Esto se puede evitar poniendo una espera adicional al final del while para soltar.

Ejercicio 3

El código resultante para el main.c es el siguiente:

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

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

  // USER button configuration (PA0)
  // PA0 as digital input(00)
  GPIOA->MODER &= ~(1 << (0*2 +1));
  GPIOA->MODER &= ~(1 << (0*2));

  BSP_LCD_GLASS_Clear();
  BSP_LCD_GLASS_DisplayString((uint8_t *)" FREE");
  /* USER CODE END 2 */
```

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    if ((GPIOA->IDR&0x00000001)!=0) { // Si PA0 = 1 (USER pressed)
                                      // incrementa estado
      BSP_LCD_GLASS_Clear();
      BSP_LCD_GLASS_DisplayString((uint8_t *)" PUSH");
      while ((GPIOA->IDR&0x00000001)!=0) { // Si PA0 = 1 (USER pressed),
                                           // espera hasta soltar el boton
        }
      BSP_LCD_GLASS_Clear();
      BSP_LCD_GLASS_DisplayString((uint8_t *)" FREE");
    }
    /* USER CODE END WHILE */
```

Ejercicio 4

El código resultante para el main.c es el siguiente:

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

```
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define UN_SEG  2500000
/* USER CODE END PD */
```

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

```
unsigned char estado = 0;
/* USER CODE END 1 */
```

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

// USER button configuration (PA0)
// PA0 as digital input(00)
GPIOA->MODER &= ~(1 << (0*2 +1));
GPIOA->MODER &= ~(1 << (0*2));

// PB6 (LED Azul) como salida digital (01)
GPIOB->MODER &= ~(1 << (6*2 +1)); // 0 en el bit deseado = AND de MODER con el inverso de un
                                  // "1" en la posición
                                  // 13 y el resto "0" (1 << (6*2 +1) -> "1" desplazado 13
                                  // veces desde la derecha)
GPIOB->MODER |= (1 << (6*2));     // 1 en el bit deseado = OR de MODER con un "1" en la
                                  // posición 12 y el resto "0".
                                  // (1 << (6*2)) -> "1" desplazado 12 veces desde la derecha)

// PB7 (LED Verde) como salida digital (01)
GPIOB->MODER &= ~(1 << (7*2 +1));
GPIOB->MODER |= (1 << (7*2));

BSP_LCD_GLASS_Clear();
BSP_LCD_GLASS_DisplayString((uint8_t *)" FREE");
/* USER CODE END 2 */
```

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  estado++;
  if (estado>1) estado = 0;
  switch (estado) {
    case 0: // Enciende el Led Verde y no el Led Azul
      GPIOB->BSRR = (1<<7);
      GPIOB->BSRR = (1<<6)<<16;
      if ((GPIOA->IDR&0x00000001)!=0) { // Si PA0 = 1 (USER pressed)
          BSP_LCD_GLASS_DisplayString((uint8_t *)" PUSH");
      }
      else {
          BSP_LCD_GLASS_DisplayString((uint8_t *)" FREE");
      }
      espera(UN_SEG);
      break;
    default: // Apaga el Led Verde y enciende el Led Azul
      GPIOB->BSRR = (1<<7)<<16;
      GPIOB->BSRR = (1<<6);
      if ((GPIOA->IDR&0x00000001)!=0) { // Si PA0 = 1 (USER pressed)
          BSP_LCD_GLASS_DisplayString((uint8_t *)" PUSH");
      }
      else {
          BSP_LCD_GLASS_DisplayString((uint8_t *)" FREE");
      }
      espera(UN_SEG);
      break;
  }
  /* USER CODE END WHILE */
```

Esta solución no es ideal, ya que, al pulsar el botón, se tarda en que cambie el mensaje, ya que sólo se actúa en el momento que cambia el LED. Cuando se trate el tema de temporización y de interrupciones, se verá la posibilidad de implementar esta solución de forma exacta.

Ejercicio 5

El código resultante para el main.c es 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 char estado = 1;
  uint8_t mensaje[7] = "      ";
  /* USER CODE END 1 */
```

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

  // USER button configuration (PA0)
  // PA0 as digital input(00)
  GPIOA->MODER &= ~(1 << (0*2 +1));
  GPIOA->MODER &= ~(1 << (0*2));

  // PB6 (LED Azul) como salida digital (01)
  GPIOB->MODER &= ~(1 << (6*2 +1)); // 0 en el bit deseado = AND de MODER con el inverso de un
                                    // "1" en la posición
                                    // 13 y el resto "0" (1 << (6*2 +1) -> "1" desplazado 13
                                    // veces desde la derecha)
  GPIOB->MODER |= (1 << (6*2));     // 1 en el bit deseado = OR de MODER con un "1" en la
                                    // posición 12 y el resto "0".
                                    // (1 << (6*2)) -> "1" desplazado 12 veces desde la derecha)

  // PB7 (LED Verde) como salida digital (01)
  GPIOB->MODER &= ~(1 << (7*2 +1));
  GPIOB->MODER |= (1 << (7*2));

  BSP_LCD_GLASS_Clear();
  BSP_LCD_GLASS_DisplayString((uint8_t *)" 1");
  GPIOB->BSRR = (1<<7)<<16;
  GPIOB->BSRR = (1<<6)<<16;
  /* USER CODE END 2 */
```

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    if ((GPIOA->IDR&0x00000001)!=0) { // Si PA0 = 1 (USER pressed)
                                      // incrementa estado
      estado++;
      if (estado>4) estado = 1;
      switch (estado) {
        case 2: // Enciende el Led Verde y no el Led Azul
          GPIOB->BSRR = (1<<7);
          GPIOB->BSRR = (1<<6)<<16;
          mensaje[1] = '2';
          break;
        case 3: // Enciende el Led Verde y el Led Azul
          GPIOB->BSRR = (1<<7);
          GPIOB->BSRR = (1<<6);
          mensaje[1] = '3';
          break;
        case 4: // Enciende el Led Azul y no el Led Verde
```

```
        GPIOB->BSRR = (1<<7)<<16;
        GPIOB->BSRR = (1<<6);
        mensaje[1] = '4';
        break;
      default: // Apaga el Led Verde y el Led Azul
        GPIOB->BSRR = (1<<7)<<16;
        GPIOB->BSRR = (1<<6)<<16;
        mensaje[1] = '1';
        break;
    }
    BSP_LCD_GLASS_DisplayString(mensaje);
    while ((GPIOA->IDR&0x00000001)!=0) { // Si PA0 = 1 (USER pressed),
                                         // espera hasta soltar el boton
        }
  }
  /* USER CODE END WHILE */
```