Tema 4: GPIO y Funciones Alternativas

Sistemas Digitales basados en Microprocesador

Grado en Ingeniería Telemática

© Raúl Sánchez Reíllo

ÍNDICE

- Conceptos Previos
- Funcionalidad del pin
- GPIO: Registros de Control
- GPIO: Registros de Datos
- GPIO: Registros de Estado
- Ejemplo de Uso de GPIO
- Ejercicios

Conceptos Previos

- El STM32L152RB tiene 19 pines disponibles:
  - GPIOA: 16 pines de propósito general (PA0 – PA15)
    - En el sistema de desarrollo del curso sólo tiene accesibles el **PA0 (Botón USER), PA4, PA5, PA11 y PA12**
  - GPIOB: 16 pines de propósito general (PB0 – PB15)
    - Disponibles solo el **PB6 (LED Azul) y PB7 (LED Verde)**
  - GPIOC: 16 pines de propósito general (PC0 – PC15)
    - Disponibles solo el **PC12 y PC13**
  - GPIOD: 1 pin de propósito general PD2
    - **PD2** disponible en el sistema de desarrollo del curso
- La mayoría de los pines del microcontrolador tienen varias funcionalidades, es decir, además de ser pines de entrada/salida, tienen otras ***funciones alternativas***
- Por lo tanto primero habrá que seleccionar la funcionalidad del pin, luego configurarlo y luego usarlo.

FUNCIONALIDAD DEL PIN

- Cada pin puede ser configurado como:
  - Entrada digital (digital input)
    - Flotante, con resistencia de pull-up, o con resistencia de pull-down
  - Salida digital (digital output)
    - Con salida push-pull o con salida en drenador abierto con posibilidad de configurar resistencia de pull-up o de pull-down
  - Analógica (analog)
  - Función alternativa (alternate function – AF)

GPIO: Registros de Control

- GPIOx→MODER – Selección del modo de uso del PIN:
  - Se trata de un registro de 32 bits para cada uno de los puertos
    - `GPIOA->MODER`
    - `GPIOB->MODER`
    - `GPIOC->MODER`
    - `GPIOD->MODER`
  - Cada pin tiene 4 posibilidades de configuración:
    - 00 – Digital Input
    - 01 – Digital Output
    - 10 – AF
    - 11 – Analógico

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

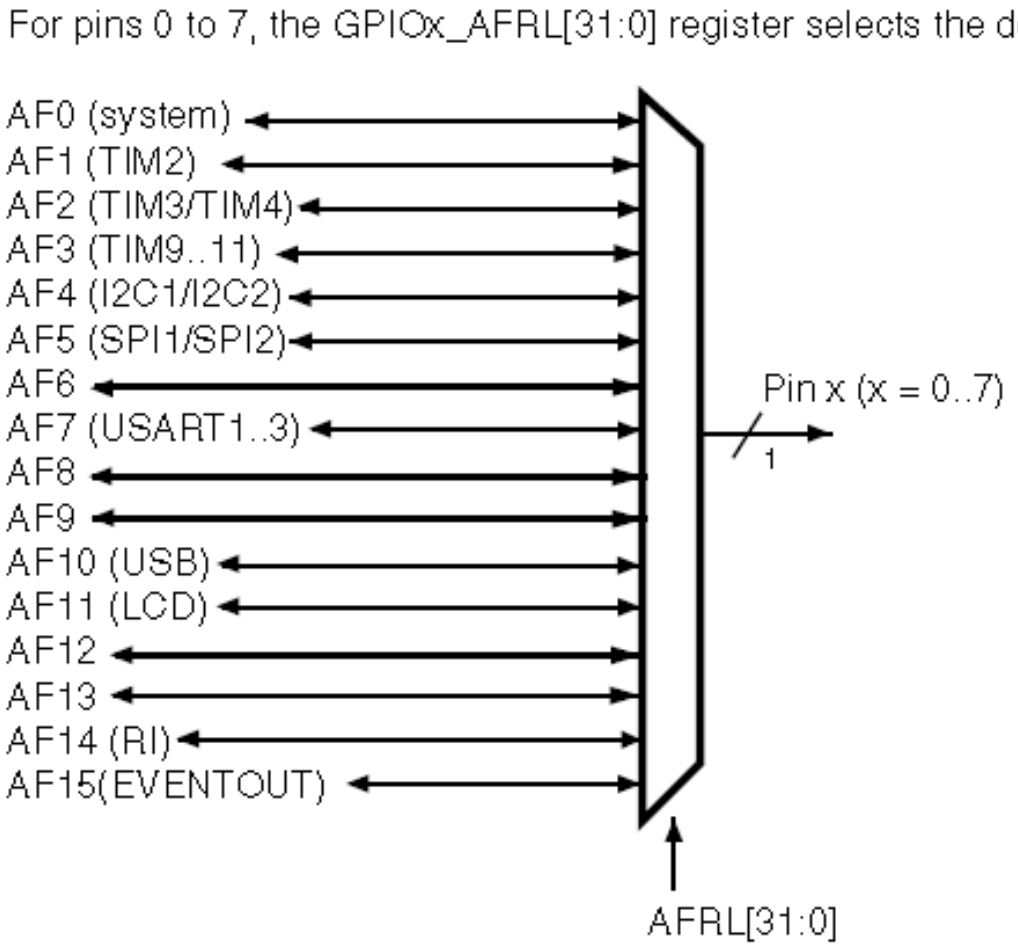
GPIO: Registros de Control

- GPIOx→AFR – Configuración de la AF (en el caso de que se haya seleccionado funcionalidad AF) :
  - Se trata de dos registros de 32 bits para cada uno de los puertos. Por ejemplo:
    - `GPIOA->AFR[0]` para los pines 0 – 7
    - `GPIOA->AFR[1]` para los pines 8 – 15
  - Cada pin tiene las 16 posibilidades siguientes:
    - 0000 – AF0, 0001 – AF1, …, 1111 – AF15

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

GPIO: Registros de Control

- GPIOx→AFR – Cada uno de estos valores (0000 … 1111) significa una función alternativa determinada



GPIO: Registros de Control

- GPIOx→OTYPER – Output Type Register:
  - Registro de 32 bits (p.ej. `GPIOA->OTYPER`), con 16 bits útiles, uno para cada pin:
    - 0 – Salida por push-pull (configuración por defecto)
    - 1 – Salida en drenador abierto

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- GPIOx→OSPEEDR – Output Speed Register:
  - Registro de 32 bits (p.ej. `GPIOA->OSPEEDR`), con 2 bits por pin:
    - 00 – 400KHz (configuración por defecto)
    - 01 – 2MHz
    - 10 – 10MHz
    - 11 – 40MHz

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

GPIO: Registros de Control

- GPIOx→PUPDR – Pull-up/Pull-down register:
  - Registro de 32 bits (p.ej. `GPIOA->PUPDR`) con 2 bits por pin:
    - 00 – Sin pull-up ni pull-down, flotante (configuración por defecto)
    - 01 – Pull-up
    - 10 – Pull-down
    - 11 – Reservada

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

GPIO: Registros de Datos

- GPIOx→IDR – Input Data Register:
  - Registro de 32 bits (p. ej. `GPIOA->IDR`), con solo 16 útiles, 1 por bit
  - Se obtiene el valor que tiene el bit en cada momento

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

GPIO: Registros de Datos

- GPIOx→BSRR – Bit Set/Reset Register:
  - Se descompone en 2 registros de 16 bits:
  - `GPIOA->BSRRL`: Registro Set
    - En aquellos bits donde se escribe un 1, ese pin (si es una salida) se pone a '1'.
    - Aquellos bits en los que se escribe un 0, no sufren cambios
  - `GPIOA->BSRRH`: Registro Reset (o Clear)
    - En aquellos bits donde se escribe un 1, ese pin (si es una salida) se pone a '0'.
    - Aquellos bits en los que se escribe un 0, no sufren cambios

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

GPIO: Registros de Estado

- El periférico GPIO no tiene Registros de Estado

EJEMPLO DE USO (1)

- El siguiente ejemplo configura el uso del LED verde y del LED azul, y establece un ciclo de encendidos y apagados:

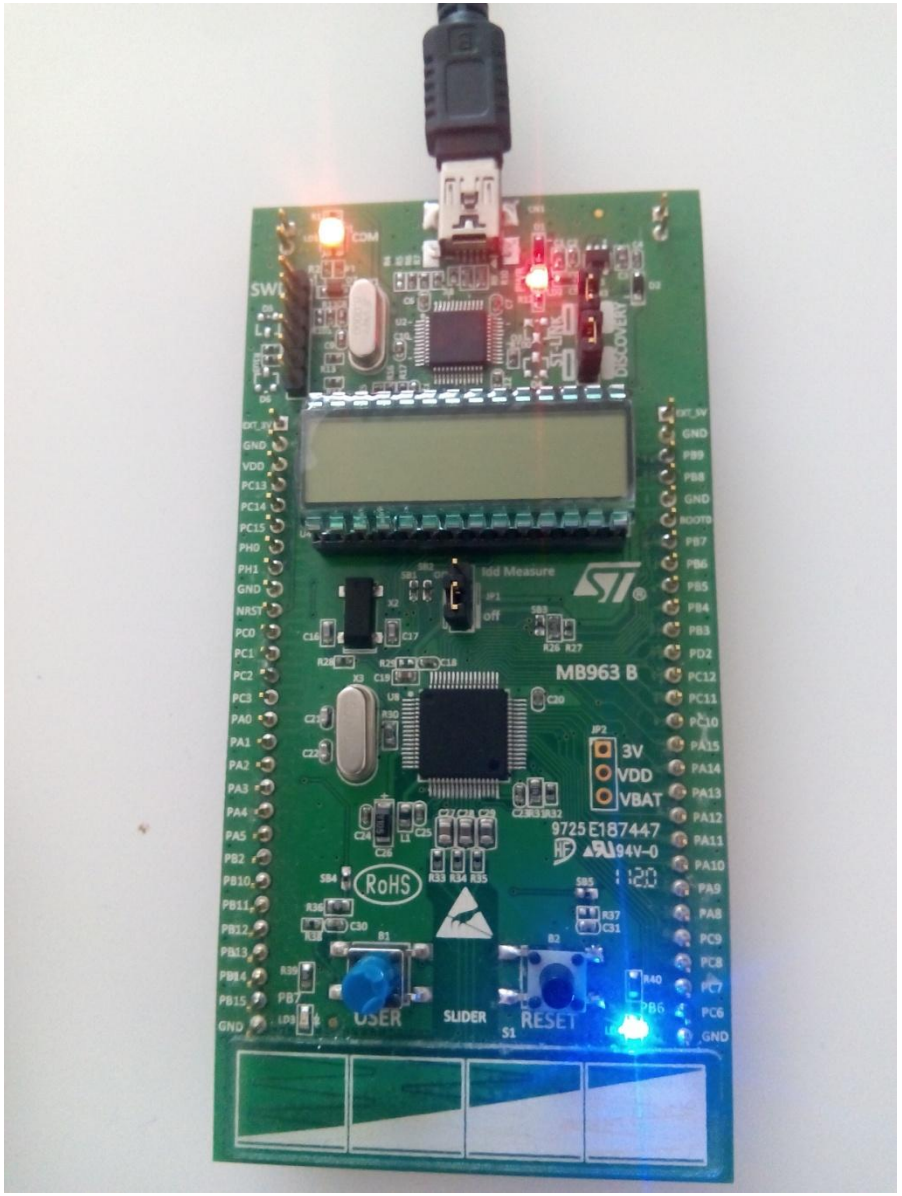
```
#include "stm32l1xx.h"
#include "Biblioteca_SDM.h"
#include "Utiles_SDM.h"

int main(void){
   Init_SDM();

   // 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));

   // Salidas como push-pull (0) y velocidad lenta (00 = 400kHz)
   GPIOB->OTYPER &= ~(1 << 6);
   GPIOB->OTYPER &= ~(1 << 7);
   GPIOB->OSPEEDR &= ~(1 << (6*2 +1));
   GPIOB->OSPEEDR &= ~(1 << (6*2));
   GPIOB->OSPEEDR &= ~(1 << (7*2 +1));
   GPIOB->OSPEEDR &= ~(1 << (7*2));
```

Ejemplo de Uso (2)

- El siguiente ejemplo configura el uso del LED Verde y del LED Azul, y establece un ciclo de encendidos y apagados:

```
while (1) {

  // Enciende el Led Verde y no el Led Azul
  GPIOB->BSRRL = (1<<7);
  GPIOB->BSRRH = (1<<6);
  espera(5000000);

  // Enciende el Led Verde y el Led Azul
  GPIOB->BSRRL = (1<<7);
  GPIOB->BSRRL = (1<<6);
  espera(5000000);

  // Enciende el Led Azul y no el Led Verde
  GPIOB->BSRRH = (1<<7);
  GPIOB->BSRRL = (1<<6);
  espera(5000000);

  // Apaga el Led Verde y el Led Azul
  GPIOB->BSRRH = (1<<7);
  GPIOB->BSRRH = (1<<6);
  espera(5000000);

  }
}
```

PRUEBA DEL EJEMPLO EXPLICADO



EJERCICIOS

- 1) Realice el diagrama de flujo del ejemplo.
- 2) Cree el proyecto y al escribir el código comente con sus propias palabras lo que hace cada línea (a nivel funcional). Ejecútelo y depúrelo. Como ve, este ejemplo ya incluye la librería de funciones útiles que se mandó crear en el tema anterior. Aprenda a crear sus propias librerías de funciones (ficheros .c y .h), para ir incluyendo aquellas funciones que pueda tener que utilizar en futuros ejercicios y prácticas. Incluso a lo largo del curso puede plantear crear varias bibliotecas.
- 3) Amplíe el programa quitándole la rutina de espera que tiene para que los LEDs sólo cambien cuando esté pulsado el botón USER. Primero haga el diagrama de flujo.
  - ¿Cuantos cambios se producen por pulsación y por qué?
- 4) Modifique el programa para que con cada pulsación sólo se haga una modificación de los LED. Primero haga el diagrama de flujo.
  - ¿Lo consigue?
  - Si no lo consigue, piense que en el momento de la pulsación pueden ocurrir oscilaciones hasta estabilizarse (denominadas **rebotes**). ¿Se le ocurre alguna forma de eludir esos rebotes?