

## Tema 5: GPIO y Funciones Alternativas

### Sistemas Digitales Basados en Microprocesadores

Universidad Carlos III de Madrid

Dpto. Tecnología Electrónica

*Nota: Las figuras utilizadas para ilustrar las características y funcionalidades del microcontrolador del curso se han obtenido de la documentación técnica disponible en <https://www.st.com/en/microcontrollers-microprocessors/stm32l151-152.html>*

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

# 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
  - GPIOH: **PH1** disponible si no se usa el oscilador externo
- 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

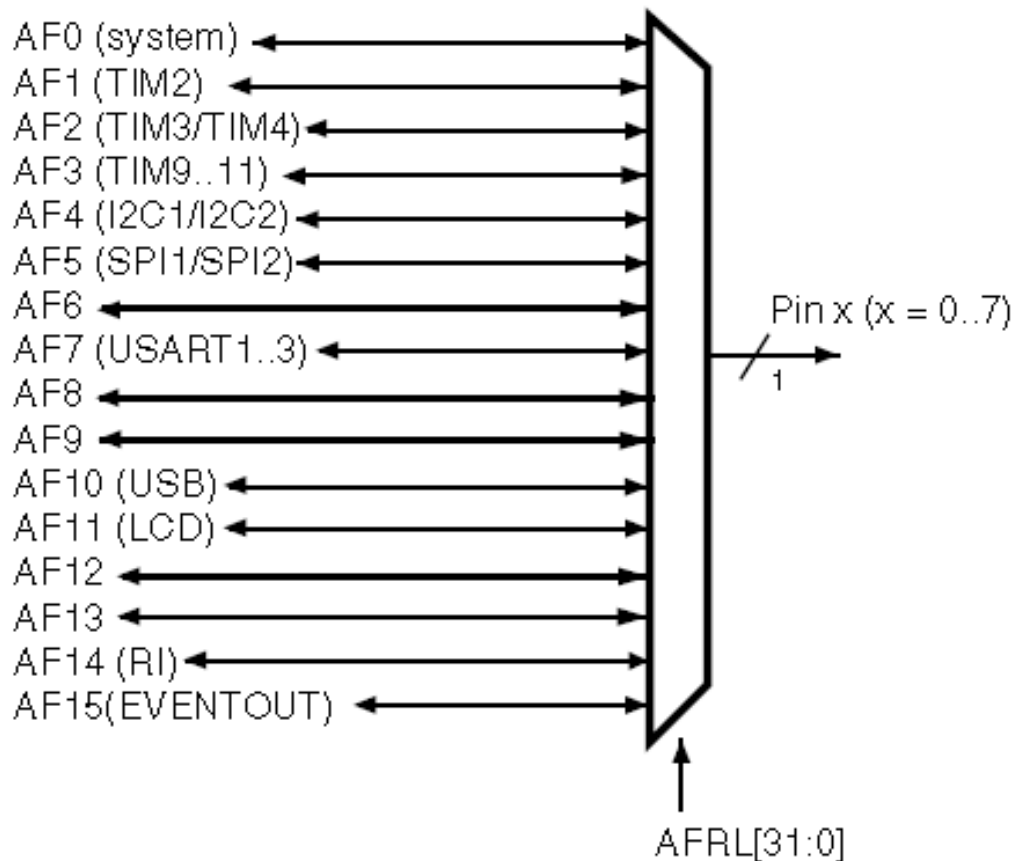
- **GPIOx** → **AFR** – Configuración de la AF (en el caso de que se haya seleccionado funcionalidad AF) :
  - Se trata de dos registros de 32 bits para cada uno de los puertos. Por ejemplo:
    - GPIOA → AFR[0] para los pines 0 – 7
    - GPIOA → AFR[1] para los pines 8 – 15
  - Cada pin tiene las 16 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

For pins 0 to 7, the GPIOx\_AFRL[31:0] register selects the dedicated alternate function



# 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 mitades de 16 bits:

- La mitad menos significativa (los bits del 0 al 15) son para poner '1' en el pin correspondiente (funcionalidad de 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
- La mitad más significativa (los bits del 16 al 31) son para poner '0' en el pin correspondiente (funcionalidad de RESET)
  - 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 Datos

- GPIOx → BSRR – Bit Set/Reset Register:

- Versiones antiguas del uVision dividían este registro en 2:

- Uno llamado GPIOx->BSSRH correspondiente a la parte alta del BSRR, es decir, a la funcionalidad de RESET
- Otro llamado GPIOx->BSSRL correspondiente a la parte baja del BSRR, es decir, a la funcionalidad de SET

- Sólo hay que hacer las siguientes transformaciones en el código:

- GPIOB->BSRRL = (1<<14); pasa a ser GPIOB->BSRR = (1<<14);
- GPIOB->BSRRH = (1<<7); pasa a ser GPIOB->BSRR = (1<<7)<<16;

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

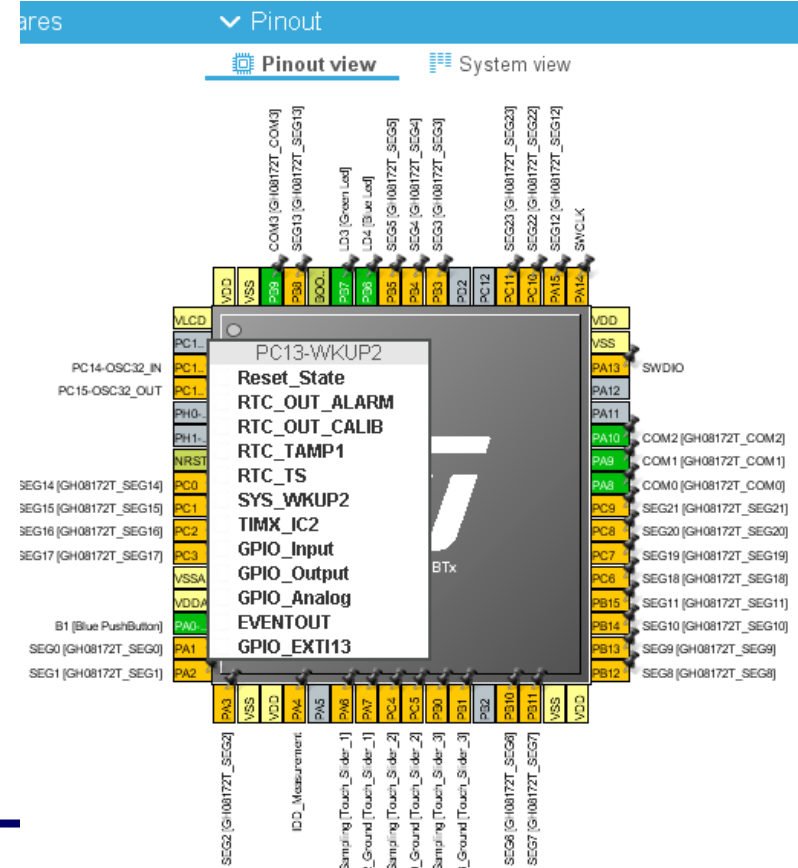
- El periférico GPIO no tiene Registros de Estado

# Peculiaridades al usar la Discovery y el CubeMX

- GPIO A:
  - **PA0: Botón USER** + Wake-up for low IDD
  - PA1 – PA3: LCD
  - PA4: IDD measurement
  - **PA5: Libre**
  - **PA6, PA7: Linear touch**
  - PA8 – PA10: LCD
  - **PA11, PA12: Libre**
  - **PA13, PA14: SWD**
  - PA15: LCD
- GPIO B:
  - **PB0, PB1: Linear touch**
  - **PB2: Boot**
  - PB3 – PB5: LCD
  - **PB6, PB7: LEDs (pull down + LED)**
  - PB8 – PB15: LCD
- GPIO C:
  - PC0 – PC3: LCD
  - **PC4, PC5: Linear touch**
  - PC6 – PC11: LCD
  - **PC12: Libre**
  - PC13: IDD measurement
    - **Modifica el comportamiento del PA0 salvo que se configure como Reset or Input**
  - PC14, PC15: Oscillator 32
- GPIO D:
  - **PD2: Libre**
- GPIO H:
  - PH0, PH1: Oscillator

# Peculiaridades al usar la Discovery y la perspectiva CubeMX

- En la Placa Discovery, PA0 también está conectada a una circuitería activada por el PC13. Por tanto, si se quiere utilizar el PA0, habrá que asegurarse que PC13 está configurado como Input o en Reset
  - Para hacerlo, en Cube MX pulsar en el pin PC13, y seleccionar la opción deseada en el menú emergente
- PH0 no está conectado
- PH1 puede ser usada como GPIO



# Uso de Máscaras en Programación

- Cuando se programa sobre micros es vital mantener la buena práctica de “sólo se modifica lo que queremos modificar, lo demás hay que dejarlo como está”
- Por ejemplo, imaginemos que en un enunciado, se pide que los pines PA0 – PA7 sean de salida. Esto implica que el registro GPIOA->MODER debe tener los bits correspondientes a esos puertos a valor 10
- Si se tuviese la tentación de escribir: **GPIOA->MODER = 0x00005555;**
- Estaría totalmente incorrecto, ya que sí que está poniendo esos pines como salida, pero está poniendo el resto como entradas (y puede que esos otros pines se estén utilizando para otra función).
- Por lo tanto hay que programar dejando los demás sin modificar.
- Para hacer esto habría que utilizar operativa con máscaras. Esto se puede hacer de varias formas pero vamos a mostrar 3 métodos.



# Uso de Máscaras en Programación

- Método 1: Crear una máscara binaria completa con codificación hexadecimal. Por ejemplo:
  - `GPIOA->MODER &= ~(0x0000FFFF); // forzamos todos los bits que queremos manipular, a '0'`
  - `GPIOA->MODER |= 0x00005555; // ponemos los '1' que necesitamos sólo en los bits correspondientes`
- Método 2: Crear la máscara haciendo desplazamiento de '1's un determinado número de bits
  - `GPIOA->MODER &= ~(0x0000FFFF); // forzamos todos los bits que queremos manipular, a '0'`
  - `GPIOA->MODER |= ( (1<<14) | (1<<12) | (1<<10) | (1<<8) | (1<<6) | (1<<4) | (1<<2) | (1<<0) ); // ponemos los '1' que necesitamos sólo en los bits correspondientes (14, 12, 10, 8, 6, 4, 2, 0)`

# Uso de Máscaras en Programación

- Método 3: Crear unos #defines compatibles con la función OR, para ser utilizado en cada uno de los bits correspondientes
  - Para este método debe crear en un fichero .h (por ejemplo Utiles\_SDM.h) los #defines para cada uno de los bits
    - **#define BIT\_0 0x00000001**
    - **#define BIT\_1 0x00000002**
    - **#define BIT\_2 0x00000004**
    - ...
    - **#define BIT\_30 0x40000000**
    - **#define BIT\_31 0x80000000**
  - Y luego, en su código, utilizarlos. Por ejemplo:
    - **GPIOA->MODER &= ~( BIT\_15 | BIT\_13 | BIT\_11 | BIT\_9 | BIT\_7 | BIT\_5 | BIT\_3 | BIT\_1 ); // Pone '0' en los bits 15, 13, 11, 9, 7, 5, 3 y 1**
    - **GPIOA->MODER |= ( BIT\_14 | BIT\_12 | BIT\_10 | BIT\_8 | BIT\_6 | BIT\_4 | BIT\_2 | BIT\_0 ); // Pone '1' en los bits 14, 12, 10, 8, 6, 4, 2 y 0**

# Ejemplo de Uso

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

```
/* USER CODE BEGIN 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));

// Salidas como push-pull (0) y velocidad lenta (00 = 400kHz). NO ES NECESARIO ESTE CURSO
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));

/* USER CODE END 2 */
```

# Ejemplo de Uso

- Funcionamiento continuo:

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

    // Enciende el Led Verde y no el Led Azul
    GPIOB->BSRR = (1<<7);
    GPIOB->BSRR = (1<<6)<<16;
    espera(5000000);

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

    // Enciende el Led Azul y no el Led Verde
    GPIOB->BSRR = (1<<7)<<16;
    GPIOB->BSRR = (1<<6);
    espera(5000000);

    // Apaga el Led Verde y el Led Azul
    GPIOB->BSRR = (1<<7)<<16;
    GPIOB->BSRR = (1<<6)<<16;
    espera(5000000);
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

# Ejemplo de Uso

