UNIVERSIDAD NACIONAL DE COLOMBIA

Primer proyecto
Led ON

Alejandro Ramírez Jaramillo

Universidad Nacional de Colombia sede Manizales Email: alramirezja@unal.edu.co

Introducción

El primer código que se estudiará en el curso encenderá un led usando los pines de salida del microcontrolador. Para esto tendremos que familiarizarnos con los registros que controlan los GPIO (General purpose input/output) e identificar la distribución de los pines en la tarjeta, la cual cuenta con varios puertos, pines y algunos elementos propios. Los documentos que se usarán en esta guía se encuentran en las referencias, junto con un link para abrirlos.

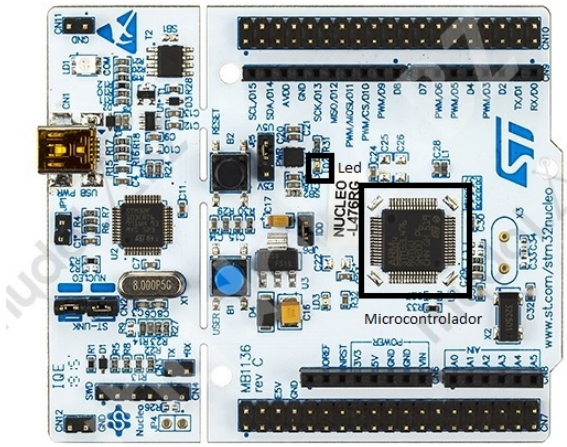
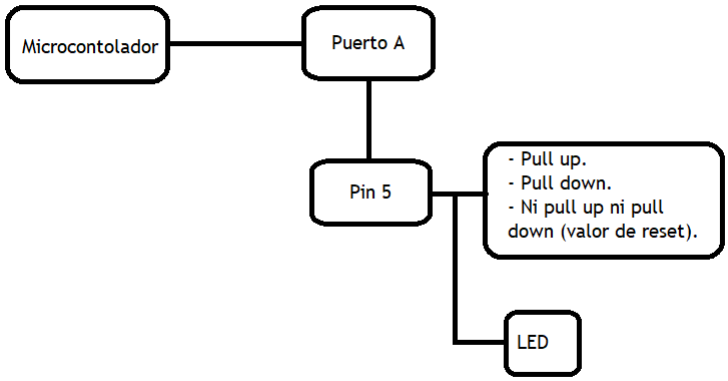


Figure 1. STM32L476



Contenido

1.Registros 3
1.1 RCC_AHB2ENR 3
1.2 GPIOx_MODER 3
1.3 GPIOx_ODR 4

2.Enmascaramiento 5

2.1 AND 5

2.2 OR 5

3.Código Led-on.....7

3.1 Inicialización 7

3.2 Configuración pines y puertos 7

3.3 Salida 8

4.Ejercicio Práctico 9

5.Referencias.....10

1 Registros

Los registros son elementos de memoria con direcciones que almacenan la configuración de diferentes funciones del microcontrolador. Determinan los parámetros de: timers, interrupciones internas y externas, así como periféricos. Los registros se componen de 32 bits (algunos de los cuales son fijos y no se pueden modificar) que corresponden a una cierta característica o función según su posición, estos bits se cambian a través de operaciones de enmascaramiento para que así tomen los valores requeridos. Los registros tienen un valor de reset y un offset que servirá para inicializarlos.

+ RCC (Reset and clock control) address: 0x4002 1000
+ GPIOA (General purpose input/output)address: 0x4800 0000

La inicialización se hace sumando a la dirección del primer registro (RCC, GPIOA) el valor de offset del segundo (ODR,MODER, AHB2ENR).

1.1 RCC_AHB2ENR

AHB2 peripherial clock enable register, registro que nos permite activar el reloj para ciertas secciones de la tarjeta, en este caso se usará para activar el reloj del puerto del pin que estará conectado al led.

+ Adress offset: 0x4C
+ Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEN	HASHEN	AESEN (1)
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DCMIEN	ADCEN	OTGFSEN	Res.	Res.	Res.	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 2. Distribución bits AHB2ENR, RM0351 Reference Manual, Pag. 251

Para el ejercicio el usuario usará el led 1 que se encuentra en la tarjeta, que está conectado al pin 5 del puerto A, lo cual podemos ver si accedemos al documento MB1136 schematic board que se encuentra en el link: https://www.st.com/content/ccc/resource/technical/document/user_manual/98/2e/fa/4b/e0/82/43/b7/DM00105823.pdf/files/DM00105823.pdf/jcr:content/translations/en.DM00105823.pdf
Donde encontrarán la distribución y conexiones de la tarjeta en la página 63 (Electrical schematics). Teniendo en cuenta que se usará el puerto A, cargaremos al registro el valor correspondiente para activar el reloj del puerto.

1.2 GPIOx_MODER

GPIO port mode register, con este registro se configura el modo de funcionamiento de los pines de un puerto específico. Con este el usuario establecerá el pin 5 del puerto A como una salida. La x en el nombre del registro se cambia por la letra del puerto que se va a usar (A) y se compone de 32 bits que se agrupan en pares, cada par corresponde a un pin del puerto, por ejemplo los bits con el nombre MODE3 corresponden al pin 3.

+ Adress offset: 0x00

+ Reset value puerto A: 0xABFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[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
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 3. Distribución bits MODER, RM0351 Reference Manual, Pag. 303

Modos:

00: Input mode

01: General purpose output mode

10: Alternate function mode

11: Analog mode

Para el ejercicio se necesita configurar el pin 5 como una salida y así poder encender el led, por lo cual es necesario cargar en MODE5 los bits 01 sin alterar lo valores que se encuentran en los otros bits, para lo cual utilizaremos enmascaramiento.

Figure 26. Output configuration

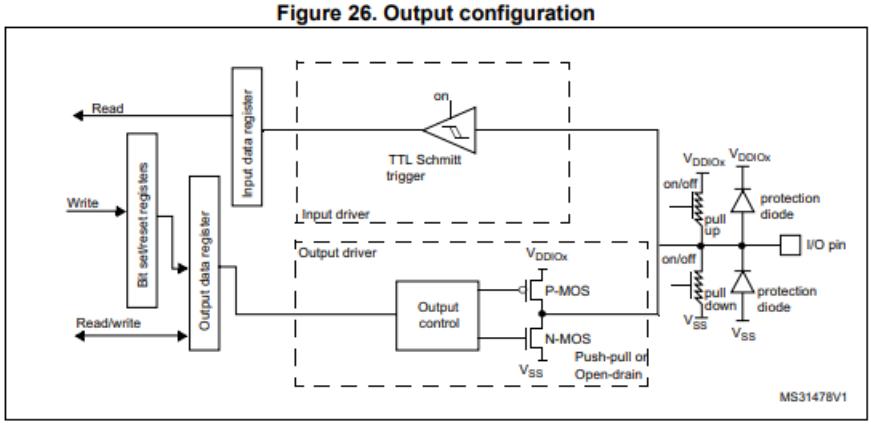


Figure 4. Output configuration, RM0351 Reference Manual, Pag. 301

1.3 GPIOx_ODR

GPIO port output data register, con este registro se manejan las salidas de un puerto (determinado por la letra que se ponga en el lugar de la x), escogiendo cuales están en HIGH y cuales en LOW (en otras palabras en que salidas hay voltaje y en cuales no). Solo se trabaja con los 16 primeros bits que están enumerados de 0 a 15, que corresponde al pin con el mismo número.

+ Adress offset: 0x14
+ Reset value puerto A: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 5. Distribución bits ODR, RM0351 Reference Manual, Pag. 305

Para poner en HIGH un pin específico basta con escribir un uno en el bit cuya posición corresponda con el número del pin.

2 Enmascaramiento

El uso de enmascaramiento es escencial en la programación en ensamblador y en C, es una operación usada para modificar los valores que se encuentran en los registros de una manera más precisa, alterando los bits que necesitamos sin cambiar los demás. Esto es útil en la medida en que es posible que no se conozca el valor que tenía el registro antes de modificarlo. El enmascaramiento se hace principalmente con las operaciones lógicas and y or.

2.1 AND

El enmascaramiento con AND permite convertir cualquier valor que se encuentra en un bit (0,1) en un 0, por lo tanto se usa para convertir los bits en cero cuando sea necesario y para los demas bits se aplica el AND con unos, para que mantengan su valor original.

AND	0	1
0	0	0
1	0	1

Table 1. Combinaciones AND.

En la tabla 1 se ilustra los resultados a las posibles operaciones AND entre bits, con lo cuál queda más claro, al hacer un AND con un cero cualquier valor se vuelve cero y al hacer AND con un uno, el valor del bit permanece igual.

EJM: Es necesario cambiar los bits 3 y 4 de un registro de 16 bits por 0 pero no conocemos que valores tiene ya (0xXXXX), entonces realizamos un enmascaramiento con and:
Como se van a cambiar los bits 3 y 4 y no se van a alterar los demas, escogemos el valor binario con ceros en los bits 3 y 4 y 1 en el resto: 0b1111111111100111 el cual corresponde en hexadecimal a 0xFFE7. al realizar el and obtenemos que:
0bXXXXXXXXXXXXXXXX AND 0b1111111111100111 = 0bXXXXXXXXXXX00XXX

2.2 OR

El enmascaramiento con OR se usa para convertir cualquier valor que se encuentre en un bit en un 1, por lo tanto se usa para forzar el valor de los bits a uno mientras que los bits que no se necesitan modificar se les aplica el OR con cero para que permanezcan igual.

En la tabla 2 se ilustra cuales son los resultados a los posibles operaciones OR entre bits, con lo cual queda más claro, al hacer un OR con un uno cualquier valor se vuelve uno y al hacer OR con un cero, el valor del bit permanece igual.

OR	0	1
0	0	1
1	1	1

Table 2. Combinaciones OR.

EJM: Es necesario cambiar los bits 6 y 7 de un registro de 16 bits por 1 pero no conocemos que valores tiene ya (0xXXXX), entonces realizamos un enmascaramiento con or:
Como se van a cambiar los bits 6 y 7 y no se van a alterar los demas, escogemos el valor binario con unos en los bits 6 y 7 y 0 en el resto: 0b0000000011000000 el cual corresponde en hexadecimal a 0x00C0. al realizar el OR obtenemos que:
0bXXXXXXXXXXXXXXXX OR 0b0000000011000000 = 0bXXXXXXXX11XXXXXX

3 Código Led-on

El código para encender el led se puede dividir en tres partes: inicialización, configuración de pines y puertos y la salida.

3.1 Inicialización

Como se explicó anteriormente para usar los registros es necesario inicializarlos al principio del código usando una dirección de base y el valor del offset. En esta primera sección del codigo se inicializan los registros AHB2ENR, MODER y ODR.

```
// Constants
.equ     LEDDELAY,      100000

// Register Addresses
// You can find the base addresses for all peripherals from Memory Map section
// RM0351 on page 78. Then the offsets can be found on their relevant sections.

//       RCC   base address is 0x40021000
//   AHB2ENR register offset is 0x4C
.equ     RCC_AHB2ENR,   0x4002104C // RCC AHB2 peripheral clock reg (page 251)

//       GPIOA base address is 0x48000000
//   MODER register offset is 0x00
//   ODR   register offset is 0x14
.equ     GPIOA_MODER,   0x48000000 // GPIOA port mode register (page 303)
.equ     GPIOA_ODR,     0x48000014 // GPIOA output data register (page 305)
// Start of text section
.section .text
////////////////////////////////////////////////////////////////////////////////
// Vectors
////////////////////////////////////////////////////////////////////////////////
// Vector table start
// Add all other processor specific exceptions/interrupts in order here
        .long    __StackTop                // Top of the stack. from linker script
        .long    _start +1                 // reset location, +1 for thumb mode
```

3.2 Configuración

Los registros AHB2ENR y MODER se usan para la configuración de los puertos y pines. Para el registro AHB2ENR es necesario activar (poner en uno el bit) el reloj del puerto A, el cual corresponde al bit 0, para hacer este cambio cargue la dirección del registro en otro (r6), a otro registro (r5) aplíquele enmascaramiento para dejarlo en el valor de reset y cambiar los bits que necesita, después se almacenA el valor del segundo registro (r5) en el primer registro (r6) que contiene la dirección del registro AHB2ENR.

```
_start:
        // Enable GPIOA Peripheral Clock (bit 0 in AHB2ENR register)
        ldr r6, = RCC_AHB2ENR       // Load peripheral clock reg address to r6
        ldr r5, [r6]                // Read its content to r5
        orr r5, 0x00000001          // Set bit 0 to enable GPIOA clock
        str r5, [r6]                // Store result in peripheral clock register
```

El procedimiento anterior se repite para el registro MODER cambiando el valor que se va a poner en los bits, esta vez reemplace los bits 11 y 10 (correspondientes al pin 5) por 01 (General purpose output mode).
Hexadecimal: F, Binario: 1111

AND
Hexadecimal: 7, Binario: 0111
Resultado:0111

```
        // Make GPIOA Pin5 as output pin (bits 1:0 in MODER register)
        ldr r6, = GPIOA_MODER       // Load GPIOA MODER register address to r6
        ldr r5, [r6]                // Read its content to r5
        and r5, 0xABFFFFFF          // Clear bits 11, 10 for P5
        and r5, 0xFFFFF7FF          // Write 01 to bits 11, 10 for P5
        str r5, [r6]                // Store result in GPIOA MODER register
```

3.3 Salida

Después de haber configurado los pines de salida y el reloj para el puerto, es momento de encender el led poniendo el pin 5 del puerto A en HIGH, para lo cual se utiliza el registro ODR. Solo es necesario encender uno de los pines del puerto (el 5) así que para los 16 bits (editables) del registro ODR que tienen un valor de reset de 0x0000 cambie por 1 el bit 5, lo cual se convierte a hexadecimal y es 0x0020.
Después de ejecutar el código pasa a un ciclo infinito (loop) que se seguirá ejecutando por el resto del programa.

```
        ldr r6, = GPIOA_ODR         // Load GPIOA output data register
        ldr r5, = 0x0020                // Read its content to r5
        str r5, [r6]                // Store result in GPIOA output data register
        // Read GPIOC Pin13


loop:
        nop                         // No operation. Do nothing.
        b loop                      // Jump to loop
```

4 Ejercicio práctico

1. Use el delay en ensamblador aprendido en clase para hacer parpadear el led que se encendió en el código y pruebe hacer diferentes los tiempos de encendido y apagado. Para esto use el registro ODR y la constante LEDDELAY que se creo al principio del código.
2. Use el botón que se encuentra conectado al pin 13 del puerto C para encender y apagar el led. Para esto use el registro GPIOx_IDR (Página 305, RM0351 Reference Manual), configure el puerto C de la misma manera que el puerto A pero cambiando el pin y el modo de funcionamiento. Pista: El comando "cmp" (Página 88, PM0214 Programming Manual) compara el valor de dos registros y de ser iguales activa la bandera equal, el comando "beq" (Página 142, PM0214 Programming Manual) hace un salto a una etiqueta específica si la bandera equal está activada.

5 Referencias

+ PM0214 Programming manual, STM32 Cortex-M4 MCUs and MPUs programming manual.
+ RM0351 Reference manual, STM32L4x5 and STM32L4x6 advanced Arm-based 32-bit MCUs.