UNIVERSIDAD NACIONAL DE COLOMBIA

Primer proyecto
Led ON (GPIO)

Gustavo Adolfo Osorio - Samuel Esteban Reyes - Santiago Matta Amador

Universidad Nacional de Colombia sede Manizales Email: gaosoriol@unal.edu.co sreyesn@unal.edu.co smatta@unal.edu.co

Introducción

En esta guia haremos una breve explicacion de como se configura un boton, en este caso para el encendido de un LED. Como ejemplo usaremos el boton USER de la tarjeta stm32L476.

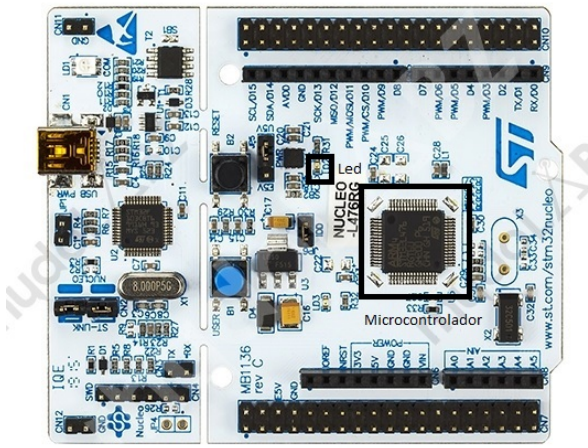

Figure 1. STM32L476

Contenido

1 Botón (Constants)

Otro configuración que se le puede dar a las GPIO es la de entrada, la cual en vez de cambiar el estado de un pin permite leer el estado en ese pin, el cual está determinado por lo que se haya conectado a ese pin. En este ejercicio se va a usar como entrada el pin conectado al botón de la tarjeta y se va a hacer la lectura de dicha entrada, para tomar decisiones de acuerdo a dicha entrada.

1.1 Inicialización

La inicialización se hará igual a como lo haciamos con los LEDS, pero esta vez se debe agregar la dirección de GPIOC junto con la de los otros puertos.

```
.equ GPIOC_BASE, 0x48000800 //(p78)
```

1.2 Configuración

Se programa de la misma forma que en el ejercicio de la MFShield, solo que esta vez se inicializa el clock del puerto C junto con el del puerto A.

```
// Enable GPIOA Peripheral Clock (bit 0 in AHB2ENR register) (RM0351, page 251)
    ldr r6, =RCC_BASE       // Load peripheral clock reg address to r6
    ldr r5, [r6,#RCC_AHB2ENR]   // Read its content to r5
    orr r5, 0x00000005  // AHB2 peripheral clock enable register (RCC_AHB2ENR) (RM0351, PAGE 251
    str r5, [r6,#RCC_AHB2ENR]
```

Figure 2. Configuración Reloj

Además con el registro MODER se configurá el pin 13 del puerto C como una entrada, según los valores definidos para este modo de funcionamiento.

```
54 //CONFIGURACION PIN C13
55
56     ldr r6, =GPIOC_BASE
57     ldr r5, [r6,#GPIO_MODER]
58     bfc r5, #26, #2
59     str r5, [r6,#GPIO_MODER]
60
```

Figure 3. Configuración PC13

1.3 Control del Led usando el botón

Para finalizar se programa el control del led usando el estado del botón, para lo cual se usará el registro IDR.

- Lectura del estado del botón: Usando el registro IDR se almacena en otro registro (r1...) el estados actual del botón, como se encuentra en el pin 13, el bit 13 contendrá el estado (0 o 1).

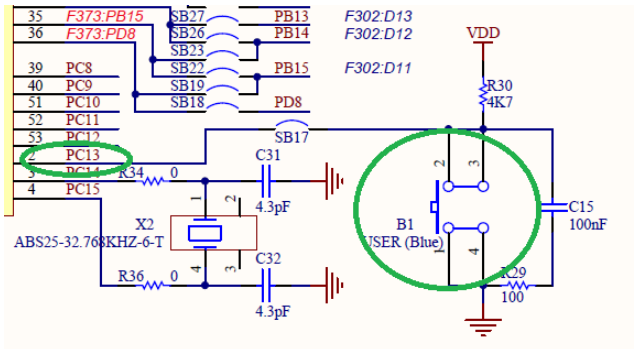

Figure 4. Configuración Reloj

- Como podemos ver en el esquematico anterior, esta señalado el pin PC13 y el boton USER B1, el cual en un estado de NO presionado, el pin PC13 lee un valor de 1, pero cuando el boton SI esta presionado, vamos a leer un 0, entonces debemos tener en cuenta la logica inversa del boton.

```
loop:

    ldr r2, =GPIOC_BASE
    ldr r1, [r2,#GPIO_IDR]

    and r1, r1, #0x2000
    ldr r3, =0x0000
    cmp r1, r3
    bne led_off
```

Figure 5. Configuración Reloj

- Limpiar y comparar: Una vez que se ha guardado el estado del botón, se usa un AND para evitar que el estado de otros pines interfiera, el resultado se compara con una constante que representará uno de los dos estados del botón.
- Salto: Dependiendo del resultado de la comparación, se hace un salto hacia una etiqueta o no, en esta etiqueta se hará el cambio de estado.

```
led_on:

    ldr r6, =GPIOA_BASE
    ldr r5, [r6,#GPIO_ODR]
    orr r5, #(1<<5)
    str r5, [r6,#GPIO_ODR]
    b loop

led_off:

    ldr r6, =GPIOA_BASE
    ldr r5, [r6,#GPIO_ODR]
    bfc r5, #5, #1
    str r5, [r6,#GPIO_ODR]

    b loop

.size   main, .-main
```

Figure 6. Configuración Reloj

2 Referencias

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