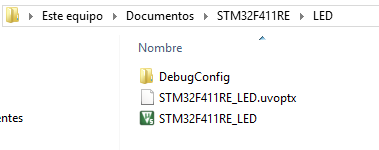
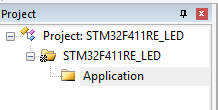
Laboratorio 2 – CMSIS RTOS / KEIL RTX

1 – Crear un nuevo Proyecto y guardarlo, en mi caso lo llamé STM32F411RE\_LED

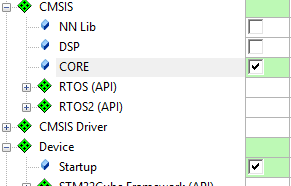


2 – Nombre el proyecto de una manera descriptiva, en mi caso le di un nombre igual a la carpeta

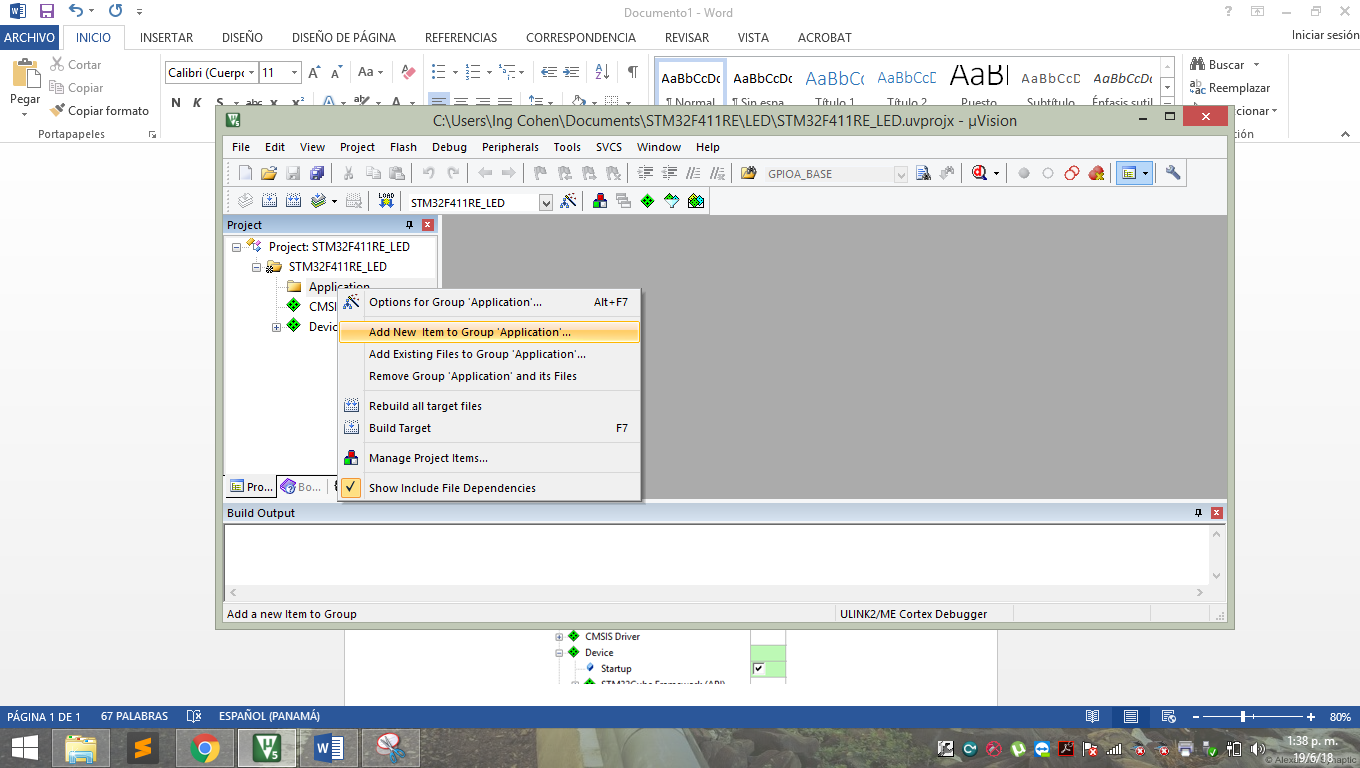


3 – Ahora administraremos los recursos a utilizar haciendo click en el siguiente icono 

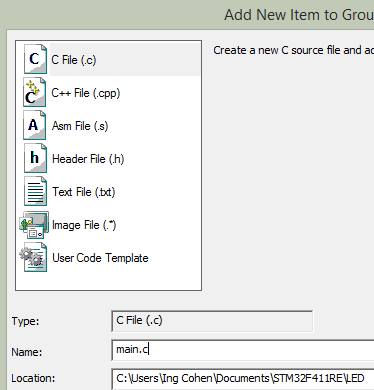
4 – Para nuestro ejemplo solamente utilizaremos los recursos de CMSIS y el archivo de arranque



5 – Añadiremos un unevo ítem al grupo, el cual será nuestro archivo principal

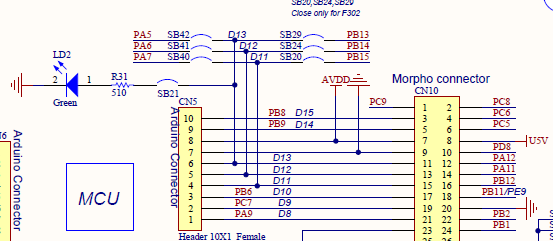


6 – Nuestro objetivo es añadir entonces un archivo principal en este caso que tenga la función main



7 – Antes de empezar a codificar tenemos que tener claros y saber cuales son los periféricos que queremos acceder, en este caso el LED2 que está en el PA2 si observamos el esquemático

<https://goo.gl/vYKFLw>



8 – Lo siguiente es realizar el código para habilitar los puertos de E/S y relojes del CPU, para esto necesitamos incluir la librería que tiene ya los registros del microcontrolador creado.

#include "stm32f4xx.h"

Lo siguiente es declarar por ejemplo, macros para definir el puerto, el estado del LED para cuando esté encendido, apagado o lo que considere necesario para su programa.

#define PA5\_OUTPUT (1 << 10) // ver punto 10 para entender línea

#define LED2 (1 << 5) // ver punto 11 para entener esta línea

Finalmente creamos la rutina del programa principal

int main(void) {

// aqui se escribirá el código de inicialización

while(1) {

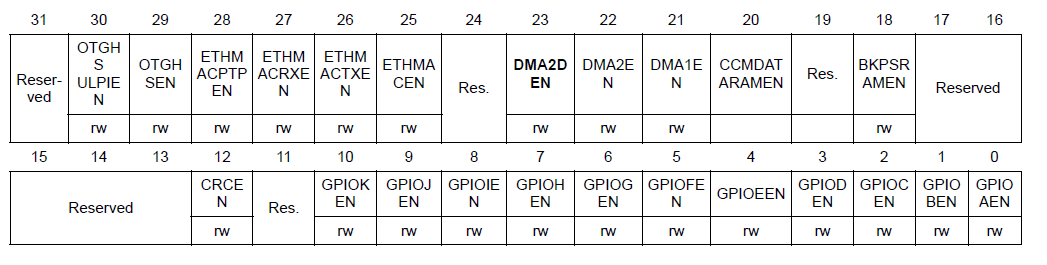
// aquí se escribirá el código para realizar el encendido y apagado de LED

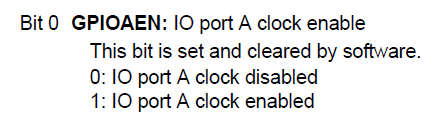
}

}

9 – Para escribir el código necesitamos ver el manual de referencia del microcontrolador.

<https://goo.gl/48b9go>

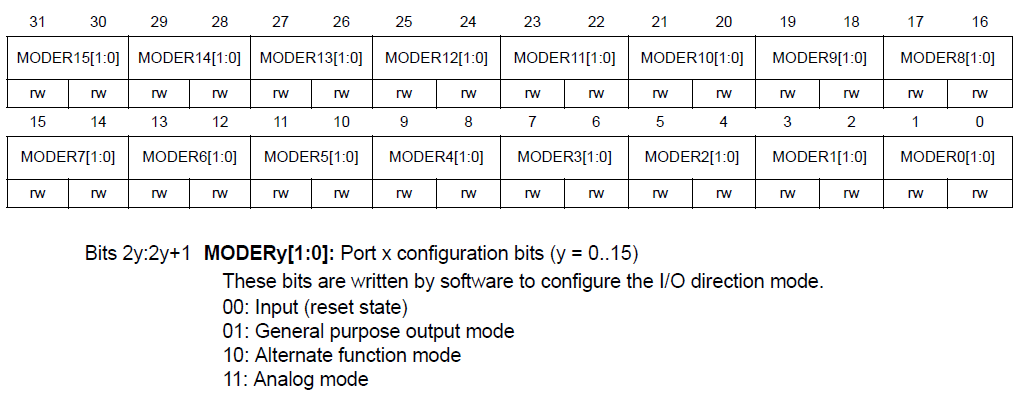




Sabemos que para habilitar el puerto A necesitamos poner este bit a 1 lógico. Para Keil, algunos registros ya están creados así que activamos en este caso el reloj del puerto A, sin esto, el puerto no haría ninguna acción

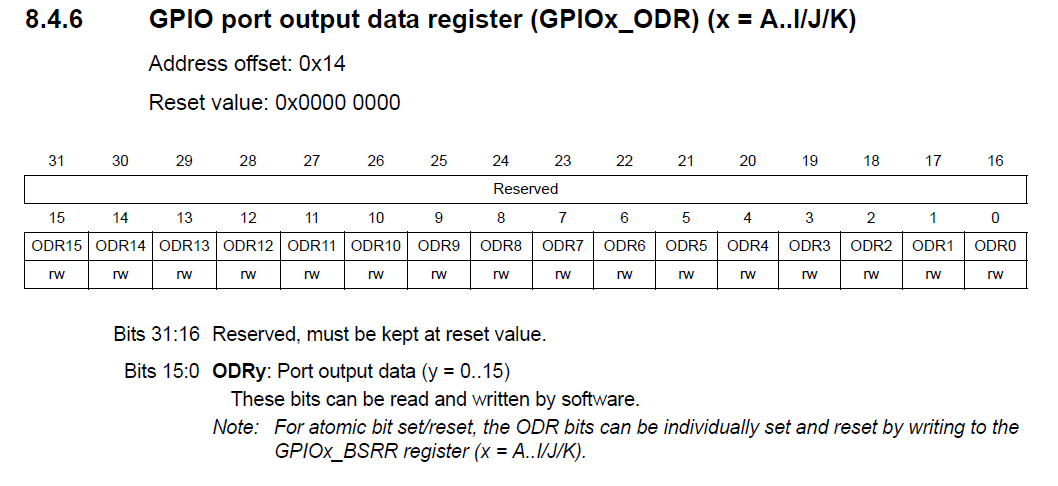
RCC->AHB1ENR |= RCC\_AHB1ENR\_GPIOAEN;

10 – Ya habilitado el reloj lo siguiente es decirle al microcontrolador que este es un puerto de Entrada o Salida, para nuestro caso salida.



GPIOA->MODER |= PA5\_OUTPUT;

11 – Lo siguiente es utilizar el registro de datos de salida para dictar el valor de encendido o apagado



GPIOA->ODR |= LED2; // para encender el LED

GPIOA->ODR &= ~LED2; // para apagar el LED

12 – Si quema el programa y depura paso a paso verá que el LED verde de su tarjeta enciende y apaga.

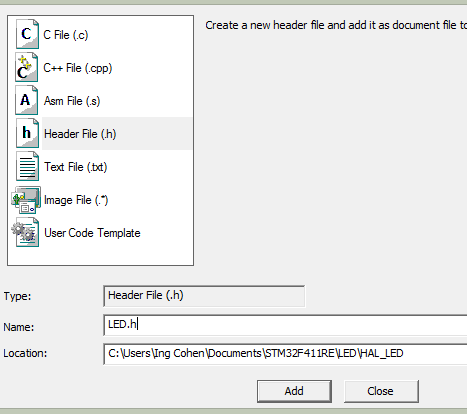
13 – Cuando desconecte el depurador verá que el LED se mantiene encendido. Lo cual no es cierto. Esto se debe a que el microcontrolador conmuta rápidamente la salida y es imperceptible al ojo humano. Es por esto que tendremos que poner un retardo de software para ver el parpadeo. También haremos una pequeña modificación al código para solo poner una línea de código que cambie el bit a 1 o 0.

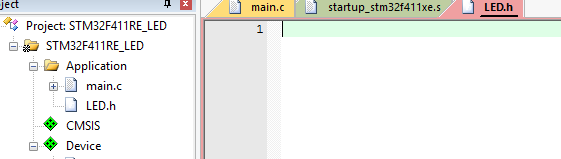
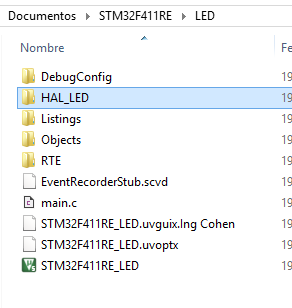
GPIOA->ODR ^= LED2;

for (int i = 0; i < 100000; i++);

Al quemar el programa veremos que ahora el parpadeo es evidente en tiempo real. Actualmente no nos preocuparemos del “timing”.

14 – Ahora que nuestro programa funciona lo modificaremos para entender como hacer nuestros propios archivos cabecera (librerías) creando un archivo llamado LED.h dentro de una carpeta (para tener orden) llamada HAL\_LED (Hardware Abstraction Layer LED)





Arrojaremos dentro lo siguiente:

#include "stm32f4xx.h"

#define PA5\_OUTPUT (1 << 10)

#define LED2 (1 << 5)

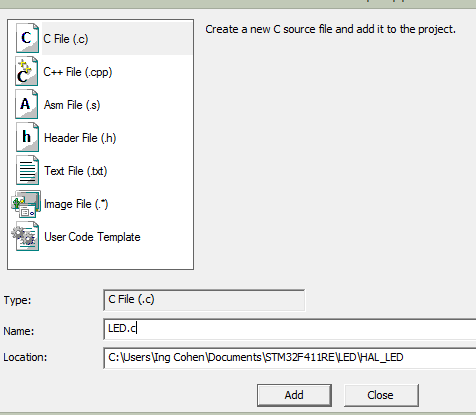
Y crearemos para nuestro uso la declaración de funciones de LED

void LED2\_On(void);

void LED2\_Off(void);

void LED2\_Toggle(void);

15 – Lo siguiente es crear nuestro archivo LED.c



E inicializarlo con el archivo cabecera de las funciones de LED más las definiciones de las funciones de LED

#include "LED.h"

void LED2\_On(void) {

GPIOA->ODR |= LED2;

}

void LED2\_Off(void) {

GPIOA->ODR &= ~LED2;

}

void LED2\_Toggle(void) {

GPIOA->ODR ^= LED2;

}

16 – El archivo main debe quedar de la siguiente manera:

#include "stm32f4xx.h"

#include "./HAL\_LED/LED.h"

int main(void) {

RCC->AHB1ENR |= RCC\_AHB1ENR\_GPIOAEN;

GPIOA->MODER |= PA5\_OUTPUT;

while (1) {

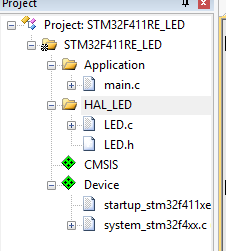
LED2\_Toggle();

for (int i = 0; i < 100000; i++);

}

}

17 – Finalmente, como opcional, no es necesario, solo para orden, creemos una carpeta dentro de nuestro proyecto para insertar la librería creadas.



18 – Como últimos pasos del laboratorio ud. debe crear las librerías Delay y System.

Delay tendrá una función llamada Delay de la siguiente manera:

Librería: delay.h delay.c

Funcion: void delay(unsinged int cycles)

Librería: system.h system.c

Funcion: void SystemInit(void)

void SystemInit(void) {

RCC->AHB1ENR |= RCC\_AHB1ENR\_GPIOAEN;

GPIOA->MODER |= PA5\_OUTPUT;

}

**Laboratorio Individual. 1 x Persona:**

Nota: Debe conseguir 3 resistencias de 3.3k y conectarlas a las salidas o puede visualizarlas con un osciloscopio.

A – Realice un programa que encienda y apague 3 LEDs de la siguiente manera

Encender…

LED1, luego de 500ms encender LED2, luego de 1000ms Encender LED3, luego de 1500ms.

Apagar…

LED1, luego de 500ms apagarLED2, luego de 1000ms apagar LED3, luego de 5000ms repetir rutina de encendido.

B – Realice un programa que encienda y apague 3 LEDs de la siguiente manera

Encender…

LED3, luego de 600ms encender LED1, luego de 4000ms Encender LED2, luego de 2000ms.

Apagar…

LED3, luego de 600ms apagarLED1, luego de 4000ms Encender LED2, luego de 2000ms repetir rutina de encendido.

C – Realice un programa que encienda y apague 3 LEDs de la siguiente manera

Encender…

LED2, luego de 3000ms encender LED3, luego de 600ms Encender LED1, luego de 1800ms.

Apagar…

LED2, luego de 3000ms encender LED3, luego de 600ms Encender LED1, luego de 1800ms repetir rutina de encendido.