

Ficheros que contiene la práctica 2

E/S programada

Ficheros que contiene la práctica 2



- **ld_script.ld**: script de enlazado utilizado en todas las prácticas
 - La dirección de comienzo del programa en esta práctica es **0x0C100000**
- **44b.h**: fichero de cabecera con definiciones de macros para facilitar el acceso a los controladores de los dispositivos de nuestro sistema.

```
...  
...  
#define rPCONB    (*(volatile unsigned *)0x1d20008)  
#define rPDATB    (*(volatile unsigned *)0x1d2000c)  
...  
...
```

Variables



Variables definidas en gpio.h

```
enum digital {  
    LOW = 0,  
    HIGH = 1  
};
```

```
enum port_mode {  
    INPUT = 0,  
    OUTPUT = 1,  
    SIGOUT = 2,  
    EINT = 3  
};
```

```
enum trigger {  
    LLOW = 0,  
    LHIGH = 1,  
    FALLING = 2,  
    RISING = 3,  
    EDGE = 4  
};
```

- 000 = Interrupción por nivel bajo = LLOW
- 001 = Interrupción por nivel alto = LHIGH
- 01x = Disparado por flanco de bajada = FALLING
- 10x = Disparado por flanco de subida = RISING
- 11x = Disparado por ambos flancos = EDGE

Variable definida en 44.h

```
enum eneable {  
    ENEABLE = 0,  
    DISABLE = 1  
};
```

Ficheros que contiene la práctica 2



- **gpio.h y gpio.c:** interfaz e implementación de funciones para el **manejo de los puertos B y G del controlador GPIO**
 - Para el **puerto B** el alumno deberá **completar las siguientes funciones** :
 - **int portB_conf (int pin, enum port_mode mode)**
 - Con esta función se configuran los leds
 - **mode** indica la configuración a poner
 - Sólo trabaja con dos opciones: SIGOUT o OUTPUT
 - **pin** indica el bit correspondiente al led
 - **int portB_write (int pin, enum digital val)**
 - Con esta función se escribe el valor **val** en el bit indicado por **pin**
 - Esta función sólo se usa si los pines están configurados como salida

Importante

tenéis que saber **con cuál de los registros del puerto B trabaja cada función**

Ficheros que contiene la práctica 2



■ **gpio.h y gpio.c**

– Para el puerto G el alumno deberá **completar las siguientes funciones** :

- **int portG_conf** (int **pin**, enum port_mode **mode**)
 - Con esta función se configuran los pulsadores
 - **mode** indica la configuración a poner
- **int portG_eint_trig** (int **pin**, enum trigger **trig**)
 - Con esta función se configura cómo se quiere detectar la interrupción de los pulsadores
 - **trig** indica el modo en que se quiere detectar la interrupción
- **int portG_conf_pup** (int **pin**, enum enable **st**)
 - Permite activar o no la resistencia de pull-up del pulsador indicado por pin
 - **st** indica si se quiere activar o no la resistencia

NOTA: en todas estas funciones **pin** indica el **bit correspondiente al pulsador**

Importante

tenéis que saber **con cuál de los registros del puerto G trabaja cada función**

Ficheros que contiene la práctica 2



■ **gpio.h y gpio.c**

- Para el puerto G el alumno deberá **completar las siguientes funciones** :
 - **int portG_read** (int **pin**, enum digital* **val**)
 - Con esta función se lee el estado del pulsador indicado por **pin**
 - Si está **pulsado devuelve val =LOW**
 - Si **NO** está **pulsado devuelve val =HIGH**
 - Esta función sólo se usa si los pines están configurados como INPUT o EINT
 - **int portG_write** (int **pin**, enum digital **val**)
 - Con esta función se escribe el valor de **val** en el bit indicado por **pin**
 - Esta función sólo se usa si los pines están configurados como OUTPUT o SIGUOT

Importante

tenéis que saber **con cuál de los registros del puerto G trabaja cada función**

Ficheros que contiene la práctica 2



- **leds.h y leds.c**: ficheros para implementar las funciones para el **manejo de los leds**
 - Está definida la variable status: indica si un led se quiere encender o apagar
 - Status= 0 apagar los dos Leds
 - Status= 1 encender el Led 1
 - Status= 2 encender el Led 2
 - Status= 3 encender los dos Leds
 - El valor a esta variable se lo asignan las funciones ➡
 - El alumno deberá **completar las siguientes funciones**
 - **void leds_init** (void)
 - Esta función tiene que:
 - » Configurar el puerto de los leds como salida
 - » Apagar los dos leds
 - **void leds_display** (unsigned int leds_status)
 - Esta función enciende o apaga el/los leds en función del valor que tenga la variable status

```
void led1_on( void )  
void led1_off( void )  
void led2_on( void )  
void led2_off( void )  
void led1_switch( void )  
void led2_switch( void )  
void leds_switch( void )
```

Nota: Recordad que los leds se enciende con "0"

Ficheros que contiene la práctica 2



- **D8Led.h y D8Led.c:** ficheros para implementar las funciones para el **manejo del display 8 segmentos**
 - Están **definidas las máscaras** que configuran las **segmentos y los dígitos**
 - Se han definido dos arrays
 - **Segments[]** contiene las máscaras de los segmentos (a-g)
 - **Digits []** contiene las máscaras de los dígitos (0-F)
 - El alumno deberá **completar las siguientes funciones:**
 - **void D8Led_segment (int value)**
 - Esta función enciende en el display el segmento Segments[value]
 - **void D8Led_digit (int value)**
 - Esta función muestra en el display el dígito Digits [value]

Nota: Recordad que los segmentos del display se encienden con “0”

Ficheros que contiene la práctica 2



- **button.h y button.c:** interfaz para el manejo de los pulsadores
 - El alumno deberá **completar** la función **unsigned int read_button(void)**
 - Esta función lee el estado de los botones y devuelve que botón se ha pulsado. Si no se ha pulsado ninguno devuelve un “0”

```
{  
    Leer botón 1  
    Si se ha pulsado devolver 1  
    Leer botón 2  
    Si se ha pulsado devolver 2  
    Si no se ha pulsado ninguno devolver 0  
}
```

Nota: Para **Leer botón** hay que usar la función **portG_read** (int **pin**, enum digital* **val**) que habéis definido en el **fichero GPIO.c**

Ficheros que contiene la práctica 2



- **utils.h y utils.c**: interfaz e implementación de funciones auxiliares.
 - Este fichero contiene la función **void Delay** (int **time**)
 - Realiza una espera activa de valor **time**
 - **time** se pone en **unidades de 0.1 ms**
 - Ejemplo, **Delay (1000)** espera 100ms
 - Esta función **se llama desde la función setup ()** del programa principal (main.c) **con el argumento 0** para su calibración

Este fichero os lo damos completo, **no hay que modificarlo**

Ficheros que contiene la práctica 2



- **init.asm:** fichero de inicialización, **está en ensamblador**
 - Contiene el símbolo start, donde empieza la ejecución de nuestro programa
 - **Configura las pilas** de los distintos **modos de ejecución**
 - Inicializa la región de datos *bss* a 0
 - Deshabilita las interrupciones en el controlador de interrupciones
 - Habilita las interrupciones del procesador (registro de estado)
 - **No es necesario para esta parte de la práctica**
 - Llama al main

Este fichero os lo damos completo, no hay que modificarlo

Programa principal de la práctica 2



- **main.c:** fichero con el código del programa principal de la práctica 2.a
 - Este fichero contiene las siguientes funciones que **deberán ser codificadas** por el **alumno**
 - **int setup (void)**
 - Configurar los controladores HW de los dispositivos que vamos a manejar
 - **int loop (void)**
 - Comprobar si se ha pulsado alguno de los pulsadores
 - Si se ha pulsado el pulsador 1:
 - » Permutar el estado del led 1 (lo apagará si estaba encendido y lo encenderá si estaba apagado)
 - » Y permutar la dirección de giro del segmento en el display de 8 segmentos
 - Si se ha pulsado el pulsador 2,
 - » Permutar el estado del led 2
 - » Y permutar el estado de movimiento (en marcha o parado) del segmento en el display
 - Si el segmento del display de de 8 segmentos no está detenido hay que mover una posición

Programa principal de la práctica 2



- **int setup (void):**
 - **Inicializar los leds** y configurarlos para que sean pines de salida
 - Utilizar las funciones definidas en leds.c
 - **Inicializar el Display 8 segmentos**
 - No necesita configuración inicial. Si queremos, podemos encender el led en la posición inicial deseada.
 - **Configurar los pulsadores:** Configurar los pulsadores como entrada y Activar la resistencia pull-up
 - Utilizar las funciones definidas en gpio.c
 - **Rutina Delay:**
 - Debe ser invocada con el valor 0 para su calibración.