
Prácticas de Laboratorio

Diseño de Sistemas Electrónicos

Fernando Martínez Martí
fmartinezmarti@ugr.es

Diego Pedro Morales Santos
diegopm@ugr.es

Curso 2012/2013

Índice general

Práctica 0: Introducción a las herramientas de prácticas.	1
1. Instalación del PSoC Creator	1
Objetivos	1
Materiales Necesarios	1
Descarga e Instalación	1
2. Familiarización con el PSoC Creator	5
Creación de un nuevo proyecto	5
Práctica 1: Introducción al PSoC de Cypress	10
1. Salidas Digitales	10
Objetivos	10
Materiales Necesarios	10
Configuración Hardware	10
Asignación de Pines	13
Implementación del Firmware	13
Programación y Resultados	14
2. Manejo de Interrupciones	15
Objetivos	15
Materiales Necesarios	15
Configuración Hardware	15
Implementación del Firmware	16
Programación y Resultados	17
3. Iluminación de un LED mediante señales PWM	18
Objetivos	18
Materiales Necesarios	18
Documentación a entregar	20
Práctica 2: Desarrollo de Aplicaciones Digitales con PSoC	21
1. Manejo del LCD	21
Objetivos	21
Materiales Necesarios	21
Configuración Hardware	21
Implementación del Firmware	21
2. Puertas Lógicas en el PSoC	24
Objetivos	24
Materiales Necesarios	24

Índice general

	Configuración Hardware	24
	Implementación del Firmware	24
	Programación y Resultados	25
3.	Utilidades del PSoC para Sistemas Digitales	26
	Objetivos	26
	Materiales Necesarios	26
	Configuración Hardware	26
	Implementación del Firmware	27
	Programación y Resultados	28
4.	Diseño de un multiplicador de 2 bits	29
	Objetivos	29
	Materiales Necesarios	29
	Documentación a entregar	30
5.	Aplicación de un Sistema Digital	31
	Objetivos	31
	Materiales Necesarios	31
	Documentación a entregar	31
	Práctica 3: Comunicaciones con el PSoC	32
1.	UART	32
	0.0.1. Objetivos	32
	Materiales Necesarios	32
	Documentación a entregar	33
	Práctica 4: Desarrollo de Aplicaciones Analógicas con PSoC	34
1.	Voltímetro	34
	0.0.2. Objetivos	34
	Materiales Necesarios	34
2.	Generador de Funciones	34
	0.0.3. Objetivos	34
	Materiales Necesarios	34
	Documentación a entregar de los dos ejercicios	35

Práctica 0: Introducción a las herramientas de prácticas.

1. Instalación del PSoC Creator

Objetivos

- Instalar el software necesario para la programación de los PSoC (Programable System on Chip) de Cypress®.
- Familiarizarse con la herramienta de diseño PSoC Creator.

Materiales Necesarios

1. Ordenador personal.

Descarga e Instalación

En primer lugar, se introduce en el navegador la siguiente dirección <http://www.cypress.com/>, esto nos llevará directamente a la pagina oficial de Cypress Semiconductor®, que es el fabricante de los PSoC. Este proporciona de forma completamente gratuita el software necesario para la programación de los PSoC previo registro en su web.

Actualmente Cypress® ofrece 3 tipos de PSoC, con diferentes prestaciones, los PSoC 1, PSoC 3 y PSoC 5.

Una vez dentro de la web, aparecerán varias secciones, pulsamos sobre la parte de **Support/Software Downloads/PSoC Creator** (ver figura 1). Que será la versión que se utilizará durante todas las practicas.

Práctica 0: Introducción a las herramientas de prácticas.

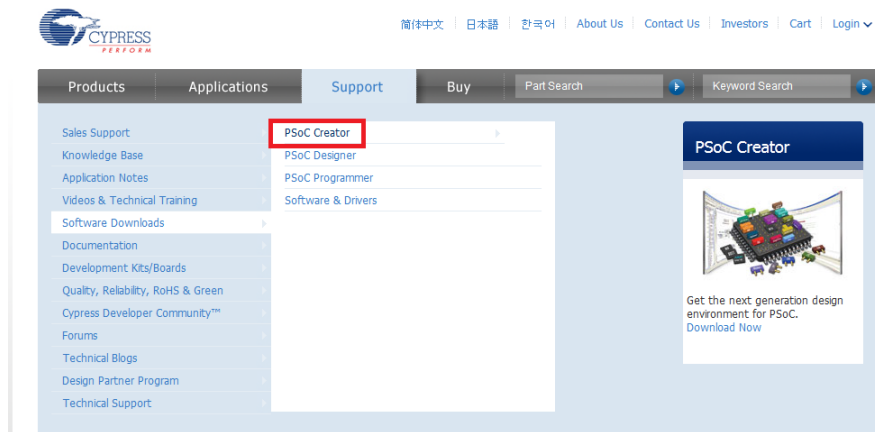


Figura 1: Descarga del Software (Paso 1).

Se nos abrirá una nueva ventana donde pulsaremos **Download Creator**(figura 2). Para poder llevar a cabo la descarga, es necesario registrarse en la web de Cypress®.

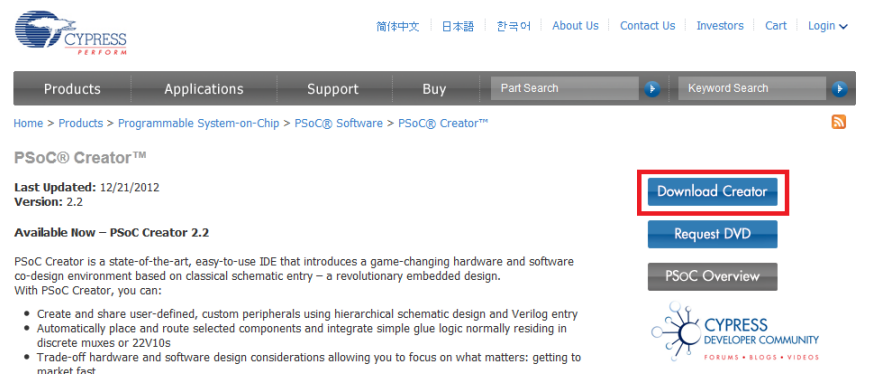


Figura 2: Descarga del Software (Paso 2).

El proceso de instalación es muy simple. Ejecutamos el instalador que nos hemos descargado. En primer lugar, nos preguntará por la ruta donde queremos instalar el programa, y en segundo lugar qué tipo de instalación hacer (figura 3). En principio la instalación por defecto será suficiente. Siempre se puede reinstalar los paquetes que hagan falta más tarde.

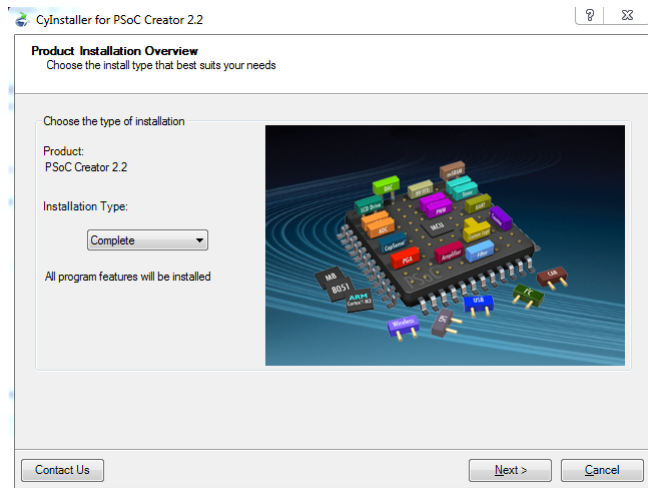


Figura 3: Instalación del Software (Paso 1).

Aceptamos el contrato acuerdo de licencia de Cypress® y el herramientas de desarrollo Keil. Una vez hecho, la instalación comenzará. Cuando todo termine de instalarse, aparecerá una ventana para añadir la información de contacto. PODEIS omitir este paso si pulsais sobre *Continue Without Contact Information*(ver figura 4) y a continuación pulsamos sobre *finish*.

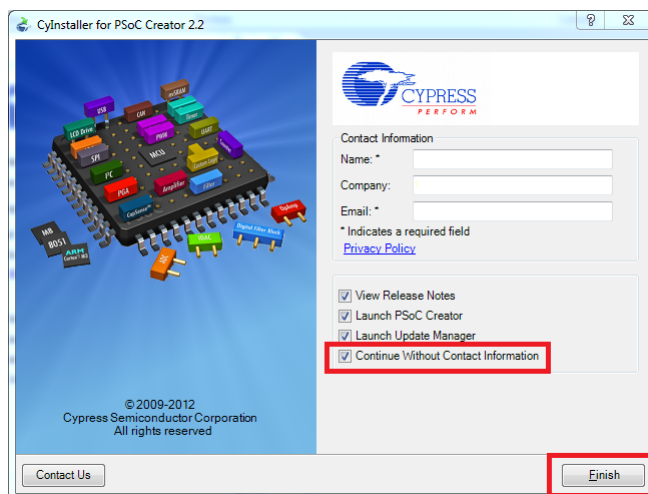


Figura 4: Instalación del Software (Paso 2).

Una vez terminada la instalación, se abrirá el Creator automáticamente, el cual requerirá en primer lugar, el registro del programa(figura 5). Para ello, basta con introducir el correo electrónico y contraseña que se utilizó para registrarse en la web de Cypress® para poder

Práctica 0: Introducción a las herramientas de prácticas.

descargar el programa.

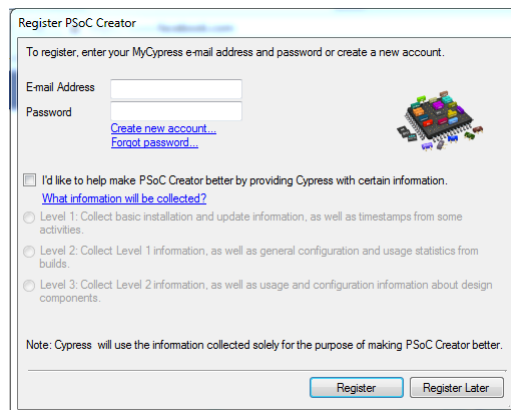


Figura 5: Instalación del Software (Paso 3).

Por lo tanto, una vez finalizada, la ventana principal del Creator será la que se puede ver en la figura 6.

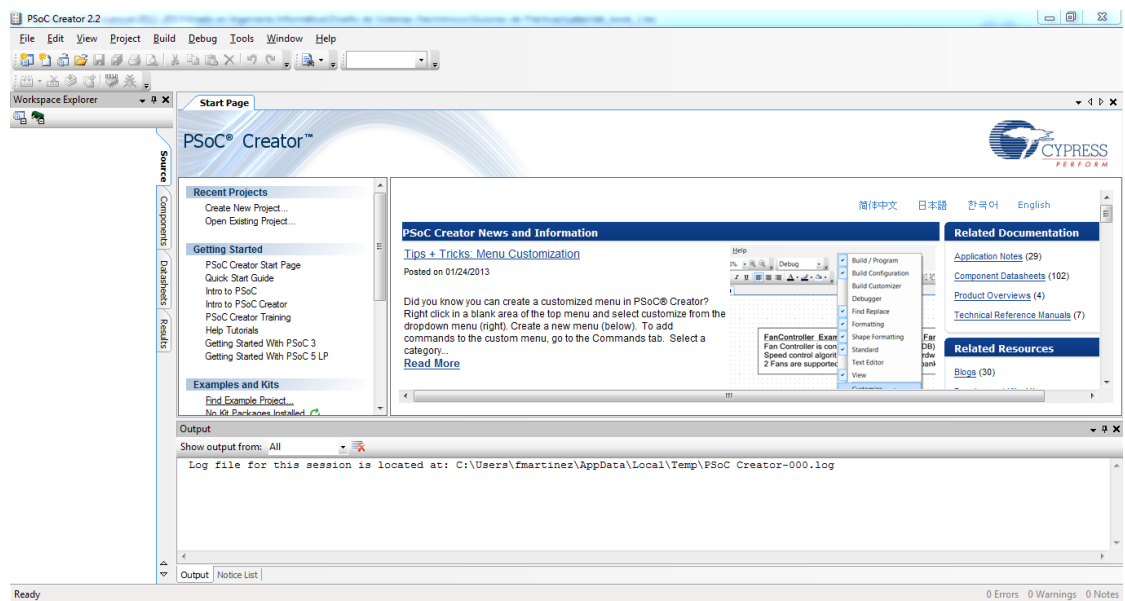


Figura 6: Ventana Principal del Creator.

2. Familiarización con el PSoC Creator

Durante esta práctica, el alumno se familiarizará con los PSoC, su diseño y programación. Se realizarán tareas básicas que permitan al alumno enfrentarse de manera eficiente a las siguientes prácticas.

Este ejercicio permitirá familiarizarse con el Creator, y además servirá para conocer que pasos se deben seguir de cara a la creación de un nuevo proyecto para cada una de las prácticas a desarrollar.

Creación de un nuevo proyecto

Para crear un proyecto nuevo en PSoC Creator, pulsamos sobre File / New / Project (ver figura 7).

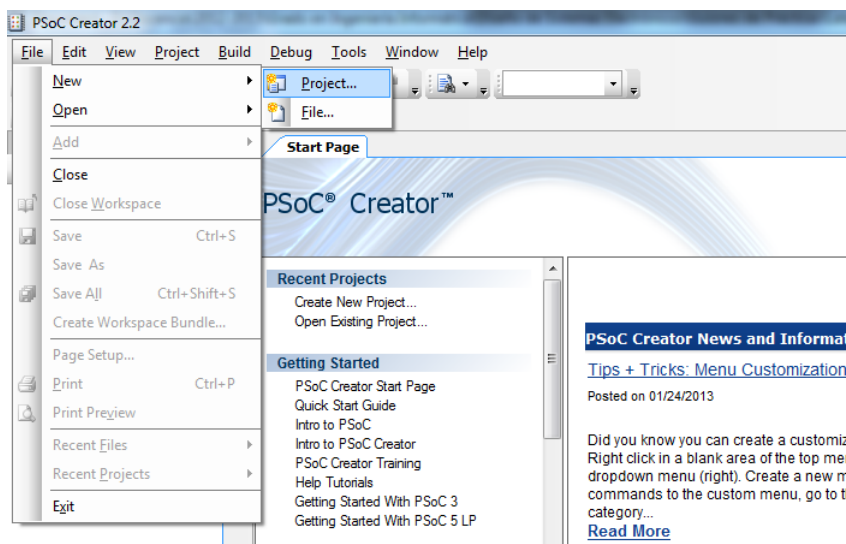


Figura 7: Creación de nuevo proyecto.

A continuación, se puede elegir si se quiere partir de una plantilla de diseño o partir de uno vacío. En nuestro caso partiremos de una plantilla vacía para el PSoC 3, que será el PSoC que utilizaremos durante las prácticas. Escribimos un nombre para el proyecto, por ejemplo Practica 0, y definimos la ruta donde queremos que se guarde como se puede ver en la figura 8. Además, pulsamos sobre “Advanced” y establecemos un nombre de espacio de trabajo para que todas las prácticas se guarden dentro del mismo espacio.

Práctica 0: Introducción a las herramientas de prácticas.

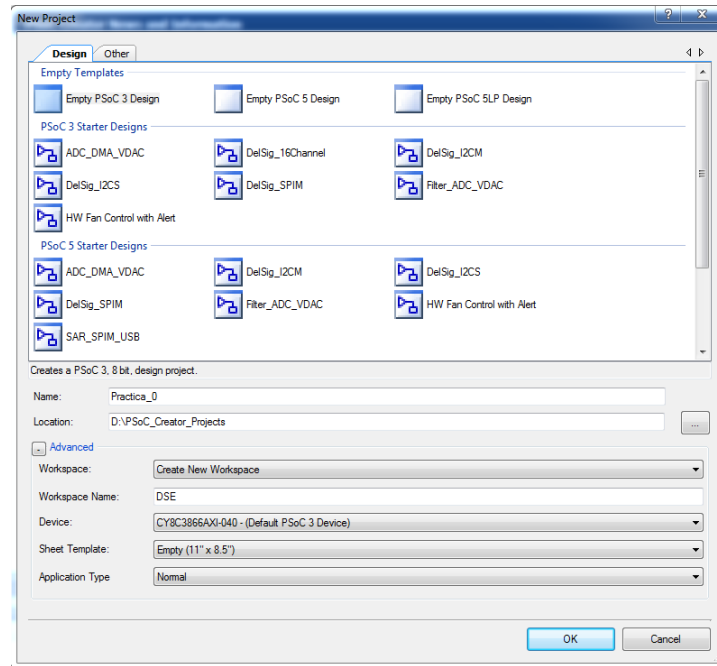


Figura 8: Tipo de proyecto.

Una vez creado el proyecto, hay que activar la licencia del compilador Keil. Para ello pulsamos sobre Help / Register / Keil (ver figura 9).

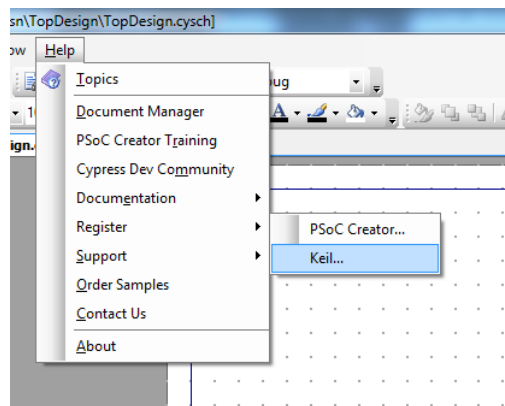


Figura 9: Activación de la licencia Keil.

De esta forma, se nos abrirá la ventana de la figura 10. En esta ventana, en primer lugar pulsamos sobre "Get LIC via Internet".

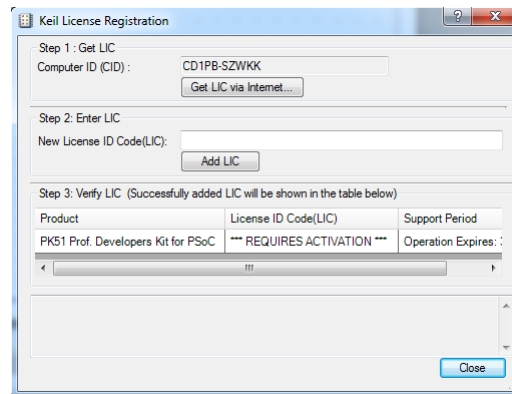


Figura 10: Ventana de activación de la licencia.

Al pulsar, se abrirá una ventana en el navegador para activar la licencia como la de la figura 11. En este formulario, se rellenaran todos los campos señalados en negrita.

Enter Your Contact Information Below

Computer ID (CID): CD1PB-SZWKK

Product Serial # (PSN): IKA1P-M6QOE-8W7ST

PC Description:
Enter a description of the PC on which this license is registered.
For example: LAB PC, Office Computer, Laptop, John's PC, etc.

First Name:

Last Name:

Professional Title:

E-mail:

Company:

Company Web Site:

Address:

City:

State/Province:

Zip/Postal Code:

Country:

Phone:

Fax:

My job is:

Age:

Employees:

Platform:

Internet Connection:

Last year, ☐ CAN ☐ LabVIEW ☐ Bluetooth ☐ C++ ☐ UML

Figura 11: Ventana de activación de la licencia.

Una vez pulsado sobre "Submit", en la siguiente ventana aparecerá un mensaje en el que dirá que la licencia se ha activado satisfactoriamente, y que en el correo electrónico que

Práctica 0: Introducción a las herramientas de prácticas.

hayamos especificado en el formulario, encontraremos un código de activación. Abrimos el correo y copiamos el código LIC en la ventana 10. Y pulsamos sobre “Add LIC”, de esta forma, ya tendremos la licencia activada durante un año.

En el workspace, que aparece a la izquierda de la ventana principal del programa, se puede acceder al esquemático del proyecto a través del archivo “TopDesign.cysch”.

Para entrar en la ventana donde se hace la asignación de los pines del PSoC, lo podemos hacer pulsando sobre “Practica_0.cydwr”, como se puede ver en la ventana 12.

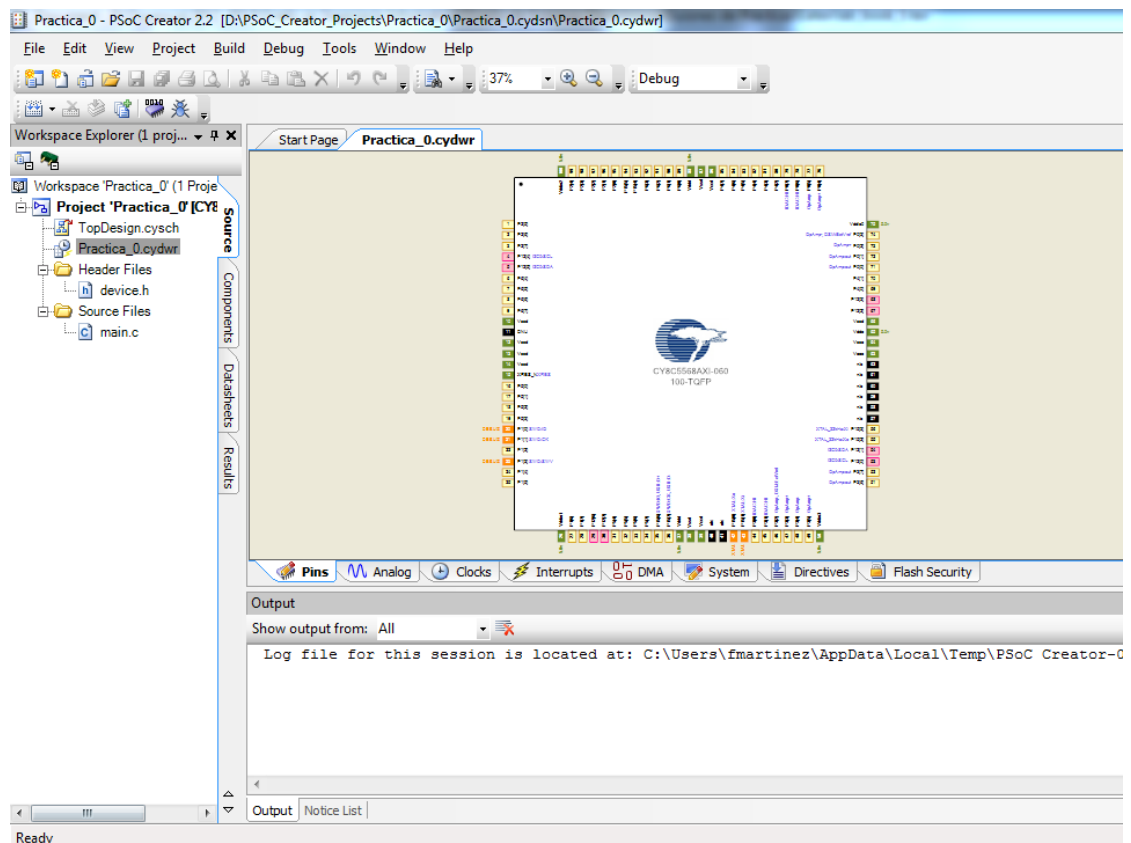
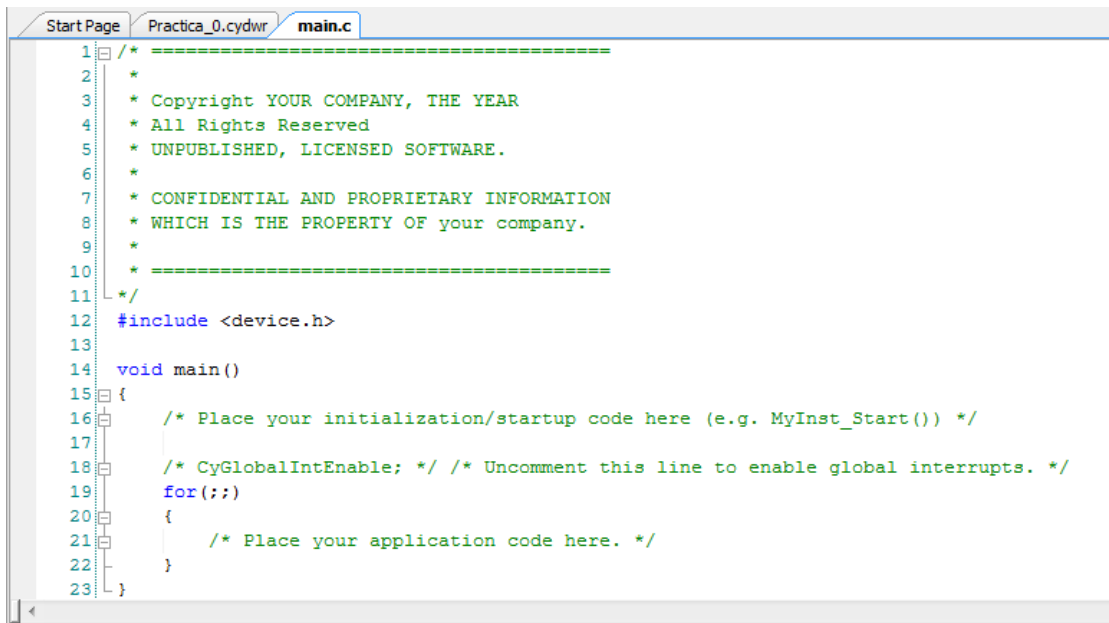


Figura 12: Asignación de pines del PSoC.

Finalmente, para modificar el firmware del PSoC. Basta con abrir el archivo main.c, obteniendo la ventana 13.



```
1  /* =====  
2  *  
3  * Copyright YOUR COMPANY, THE YEAR  
4  * All Rights Reserved  
5  * UNPUBLISHED, LICENSED SOFTWARE.  
6  *  
7  * CONFIDENTIAL AND PROPRIETARY INFORMATION  
8  * WHICH IS THE PROPERTY OF your company.  
9  *  
10 * =====  
11 */  
12 #include <device.h>  
13  
14 void main()  
15 {  
16     /* Place your initialization/startup code here (e.g. MyInst_Start()) */  
17  
18     /* CyGlobalIntEnable; */ /* Uncomment this line to enable global interrupts. */  
19     for(;;)  
20     {  
21         /* Place your application code here. */  
22     }  
23 }
```

Figura 13: Firmware del PSoC.

Práctica 1: Introducción al PSoC de Cypress

1. Salidas Digitales

Objetivos

Implementar un sistema compuesto por una salida digital que se conecte a un LED y hacerlo parpadear mediante una señal de reloj.

Materiales Necesarios

1. Software para la programación PSoC Creator de Cypress.
2. Kit de Desarrollo PSoC CY8KIT-001.
3. Osciloscopio.

Configuración Hardware

En primer lugar crearemos un nuevo proyecto como se ha explicado en la sección [2](#), a este proyecto lo llamaremos "*LAB1A*" dentro del espacio de trabajo "*DSE*". Nos aseguramos que la ventana correspondiente a "*TopDesign.cysch*" esté abierta. Desde el catalogo de componentes disponibles, que es una ventana que aparece a la derecha (figura [1](#)), elegimos System y seleccionamos Clock.

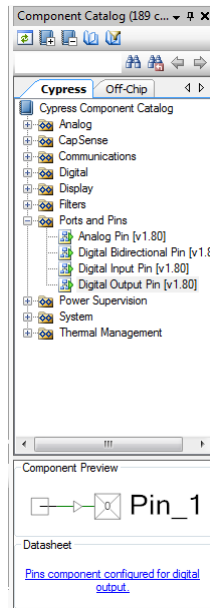


Figura 1: Añadir componente Digital Output Pin del catálogo.

Para acceder a la configuración de este componente, hacemos doble click sobre o pulsamos el botón derecho sobre el y seleccionamos Configure. Abriéndose la ventana de la figura 2. Fijamos el nombre del componente a “Reloj_Test” y la frecuencia del reloj a “1MHz”.

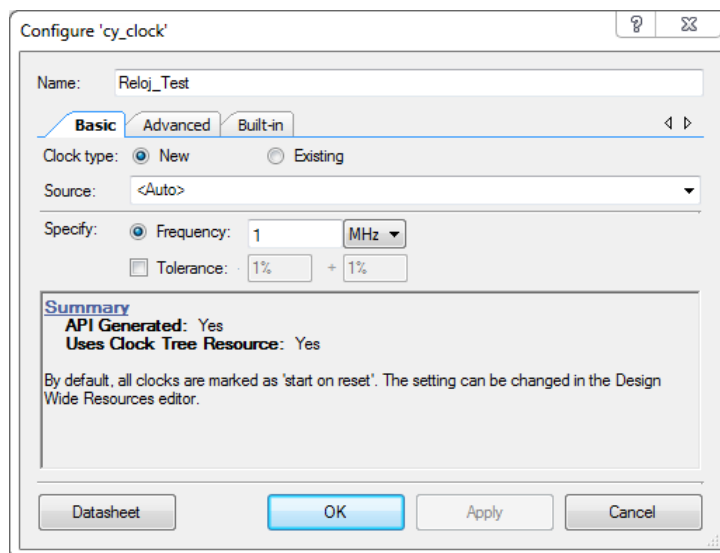


Figura 2: Configuración del componente reloj.

Práctica 1: Introducción al PSoC de Cypress

De nuevo, desde el catalogo de componentes disponibles, elegimos Port and Pins y seleccionamos Digital Output Pin. Accedemos a la configuración de este componente, hacemos doble click sobre o pulsamos el botón derecho sobre el y seleccionamos Configure. Abriéndose la ventana de la figura 3. Fijamos su nombre a por ejemplo “LED_Pin”. En el seleccionamos la opción de HW Connection, ya que queremos conectarlo al reloj añadido anteriormente, fijamos el número de pines a 1, aunque podemos poner más en caso de que queramos (ATENCIÓN: si se pone más de un pin, el componente se comportará como un puerto). En la pestaña de *General* seleccionamos **Strong Drive** para asegurar que proporcionamos suficiente corriente al LED.

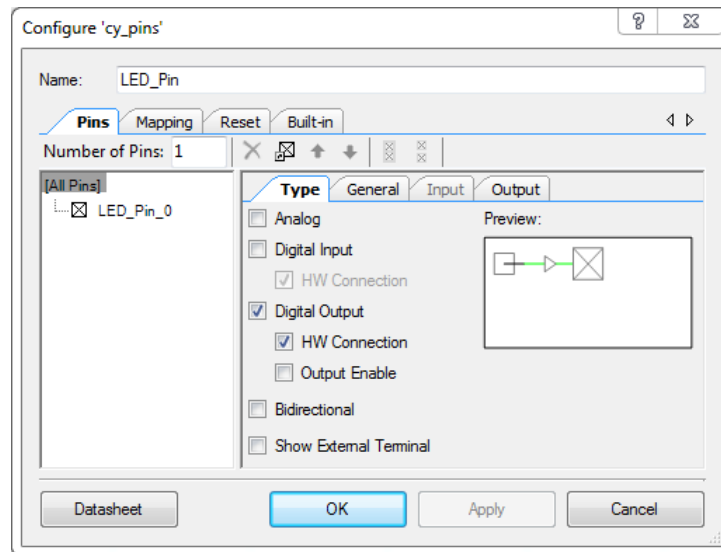


Figura 3: Configuración de los pines digitales.

Una vez que hemos añadido al esquemático ambos componentes, pulsamos **w** para entrar en el modo de conexión y conectamos el reloj y el componente de salida digital como se puede observar en la figura 4.



Figura 4: Esquemático.

Asignación de Pines

Ahora, como ya se vió en la práctica 0, accederemos a los pines del PSoC, abriendo el archivo "LAB1A.cydwr". En el lado derecho de la ventana, asignaremos a LED_Pin al PIN_1[6] de la placa de desarrollo(fig. 5).

Alias	Name /	Port	Pin	Lock
	LED_Pin	P1[6]	27	<input checked="" type="checkbox"/>

Figura 5: Asignaciones de los pines digitales a los LEDs de la placa de desarrollo.

Una vez asignado el pin. Tenemos que hacer la conexión física. Para ello conectamos un cable del pin P1_6 al LED1.

Implementación del Firmware

Abrimos desde el workspace el archivo **main.c**. Y dentro de él copiamos el siguiente código:

```
/* =====
 * Diseño de Sistemas Electronicos
 * LAB1A
 * Fernando Martínez Martí (fmartinezmarti@ugr.es)
 * =====
 */
#include <device.h>

void toggleLED(void){
    // Fijamos el valor del LED a su complementario
    LED_Pin_Write(LED_Pin_Read() ^ 1U);
}

void main(){
    for(;;){ // Loop infinito
        /* Función definida para conmutar el LED */
        toggleLED();
    }
}

/* [] END OF FILE */
```

Si pulsamos dentro Build / Build LAB1A (ver figura 6), todo el proyecto se compilará, generándose el código fuente y las cabeceras necesarias para poder hacer uso de los componentes que hemos añadido en el esquemático.

Práctica 1: Introducción al PSoC de Cypress

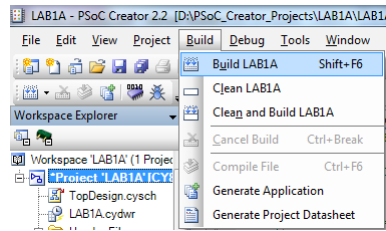


Figura 6: Compilación del proyecto.

Programación y Resultados

Para programar la placa de desarrollo, basta con pulsar sobre Debug / Program. En la figura 7 se puede ver una captura del osciloscopio del pin del LED.

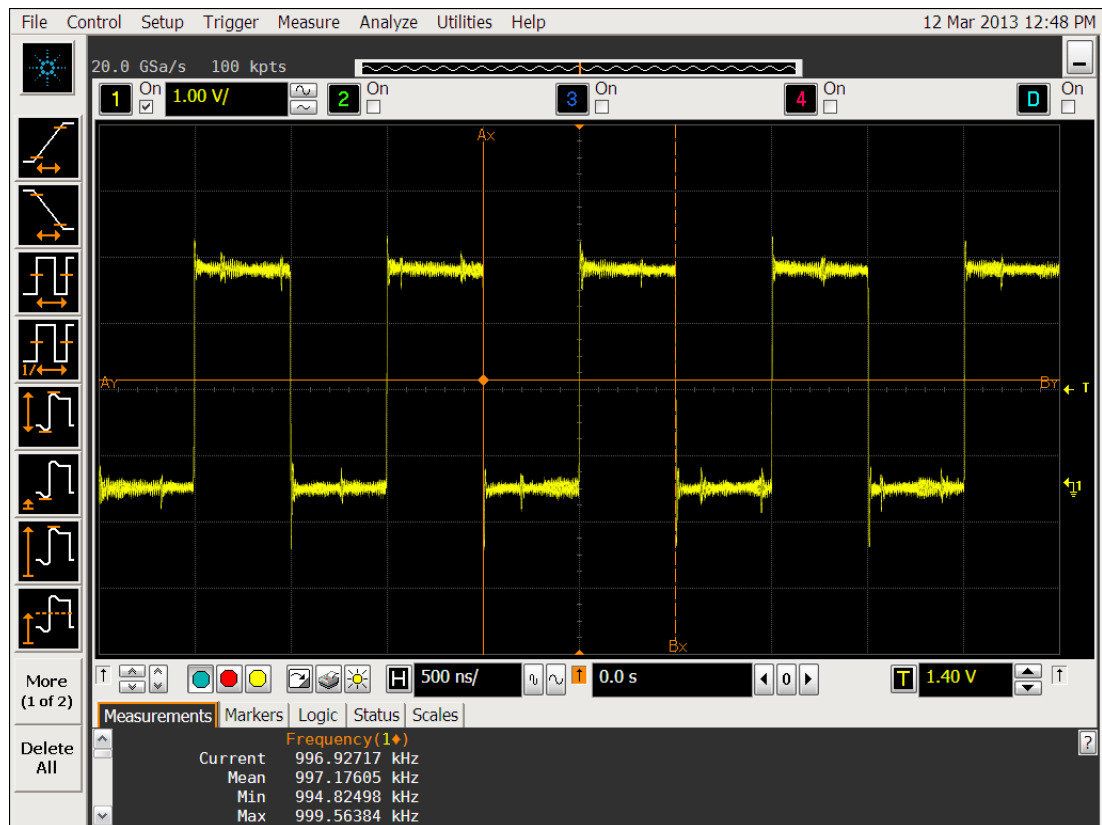


Figura 7: Captura del osciloscopio.

2. Manejo de Interrupciones

Objetivos

Implementar una sistema que encienda un LED por medio una interrupción.

Materiales Necesarios

1. Software para la programación PSoC Creator de Cypress.
2. Kit de Desarrollo PSoC CY8KIT-001.
3. Osciloscopio.

Configuración Hardware

Creamos un nuevo proyecto dentro del espacio de trabajo DSE llamado LAB1B.

En el esquemático, añadimos de el componte Clock de 10kHz y el componente de salida digital (deseleccionando HW connection en este caso) como hicimos en el ejercicio anterior. Además añadimos un componente interrupción desde System / Interrupt al que llamamos por ejemplo “isr_RelojLed”. Los conectamos todos como se puede observar en la figura 8.

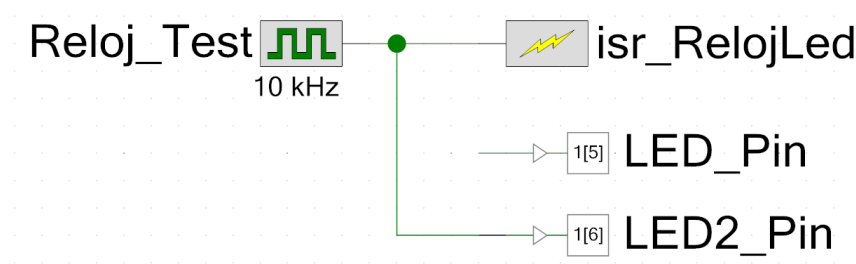


Figura 8: Esquemático.

En la ventana de asignación de pines, volvemos a conectar el Led a mismo pin que en el ejercicio anterior.

Implementación del Firmware

Abrimos desde el workspace el archivo **main.c**. Y dentro de él copiamos el siguiente código y compilamos:

```
/* =====
 * Diseño de Sistemas Electrónicos
 * LAB1B
 * Fernando Martínez Martí ( fmartinezmarti@ugr.es )
 * =====
 */
#include <device.h>

/* Variables globales */
uint8 intep;

void toggleLed(void) {
    // Fijamos el valor del LED a su complementario
    LED_Pin_Write(LED_Pin_Read() ^ 1U);
}

void toggleLed2(void) {
    // Fijamos el valor del LED a su complementario
    LED2_Pin_Write(LED2_Pin_Read() ^ 1U);
}

static CY_ISR(isr_RelojLed_Interrupt) {
    intep = 1;
}

void main() {
    /* Inicializamos los componentes en el orden correcto */
    // 1. Interrupciones
    isr_RelojLed_Start();
    // Fijamos que esta interrupcion sea la unica del sistema
    isr_RelojLed_SetVector(&isr_RelojLed_Interrupt);
    // 2. Fuentes de interrupcion (El reloj se habilita en el hardware)
    // 3. Habilitamos las interrupciones globlaes
    CyGlobalIntEnable; // Macro para habilitarlas
    // Iniciamos el valor de interrupción.
    intep = 0;

    for(;;) {
        toggleLed2();
        /* Place your application code here. */
        if(intep == 1) {
            toggleLed();
            intep = 0;
        }
    }
}

/* [] END OF FILE */
```

Programación y Resultados

Una vez programada la placa, tenemos que conseguir la mitad de frecuencia que la señal de reloj como se observa en la siguiente señal en el osciloscopio:

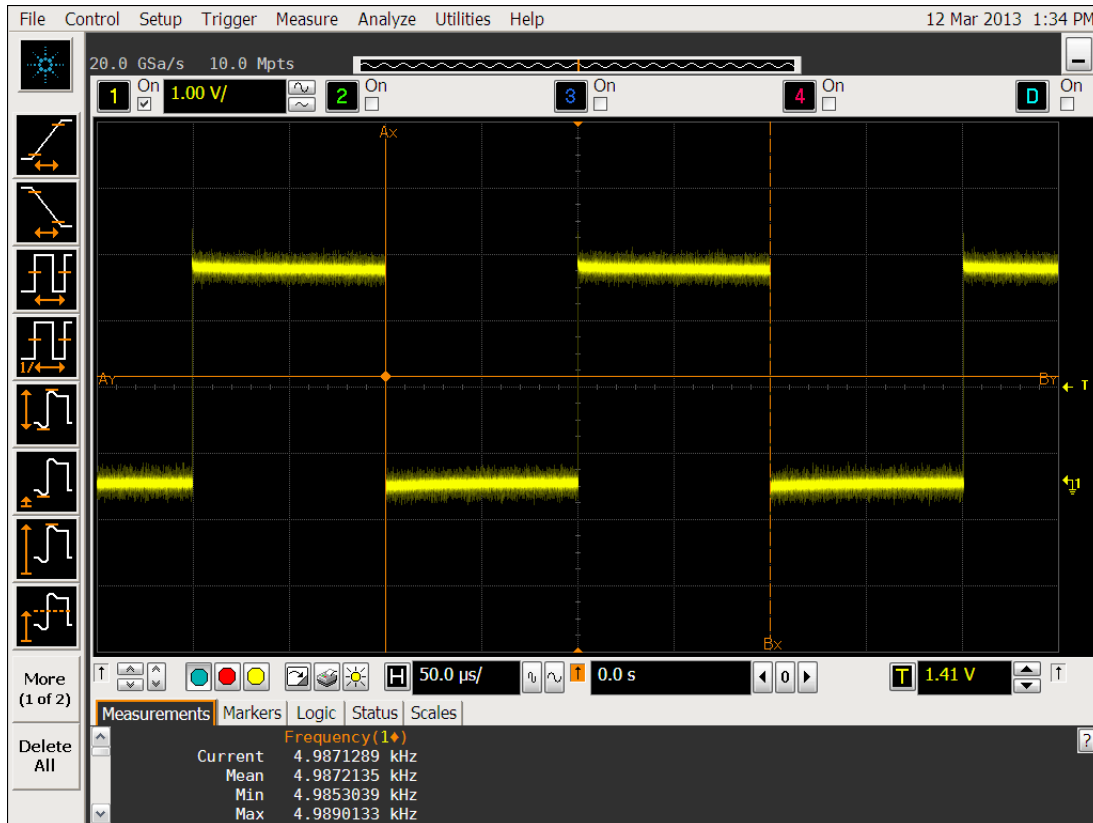


Figura 9: Captura del osciloscopio.

3. Iluminación de un LED mediante señales PWM

Objetivos

Implementar una sistema que haga parpadear dos LEDs. Uno mediante la propia señal PWM y otro mediante una interrupción excitada por la señal PWM, además, este último LED modificará su frecuencia de parpadeo en función de si uno de los botones está o no pulsado. Manteniéndose a 5Hz cuando el botón no esté pulsado, y disminuyendo a 0.5Hz cuando si lo esté.

En este ejercicio se va a crear un sistema de cronometraje que permitir completar unas tareas (encender un Led) cada milésima de segundo, cada centésima de segundo y cada segundo. El sistema temporal consiste en un Timer que alcanza su máximo y activa la interrupción cada milésima de segundo. Se debe de hacer uso de flags para notificar al bucle principal que ha pasado un determinado intervalo temporal.

El utilizar dichos flags nos permitirá usarla en otras partes del firmware para saber que tarea debemos ejecutar. Si tratamos de ejecutar una tarea completa dentro de una interrupción en lugar de usar flags se pueden dar problemas de desbordamiento de la interrupción o sincronismo.

En la figura 10 se puede ver un diagrama de flujo que se debe de seguir en la programación del PSoC.

A modo de resumen, el sistema electrónico a diseñar consistirá:

- Un reloj de sincronización del sistema a 100kHz.
- Un señal PWM de 1ms de periodo con el duty cycle elegido por alumno. Este duty cycle determinará la luminosidad del LED conectado a la salida del PWM.
- Un LED excitado por la señal PWM (PWM_Pin).
- Un LED excitado por la interrupción del PWM (LED_Pin).
- Un botón que modifique la frecuencia de parpadeo del LED (LED_Pin).

Se recuerda al alumno que la documentación de los diferentes módulos y componentes que se pueden añadir al PSoC se puede consultar a través del botón derecho sobre el componente y pulsando sobre “Open Datasheet...”, se recomienda encarecidamente consultar dicha documentación para conocer que métodos dispone cada componente así como es su sintaxis.

Materiales Necesarios

1. Software para la programación PSoC Creator de Cypress.
2. Kit de Desarrollo PSoC CY8KIT-001.
3. Osciloscopio.

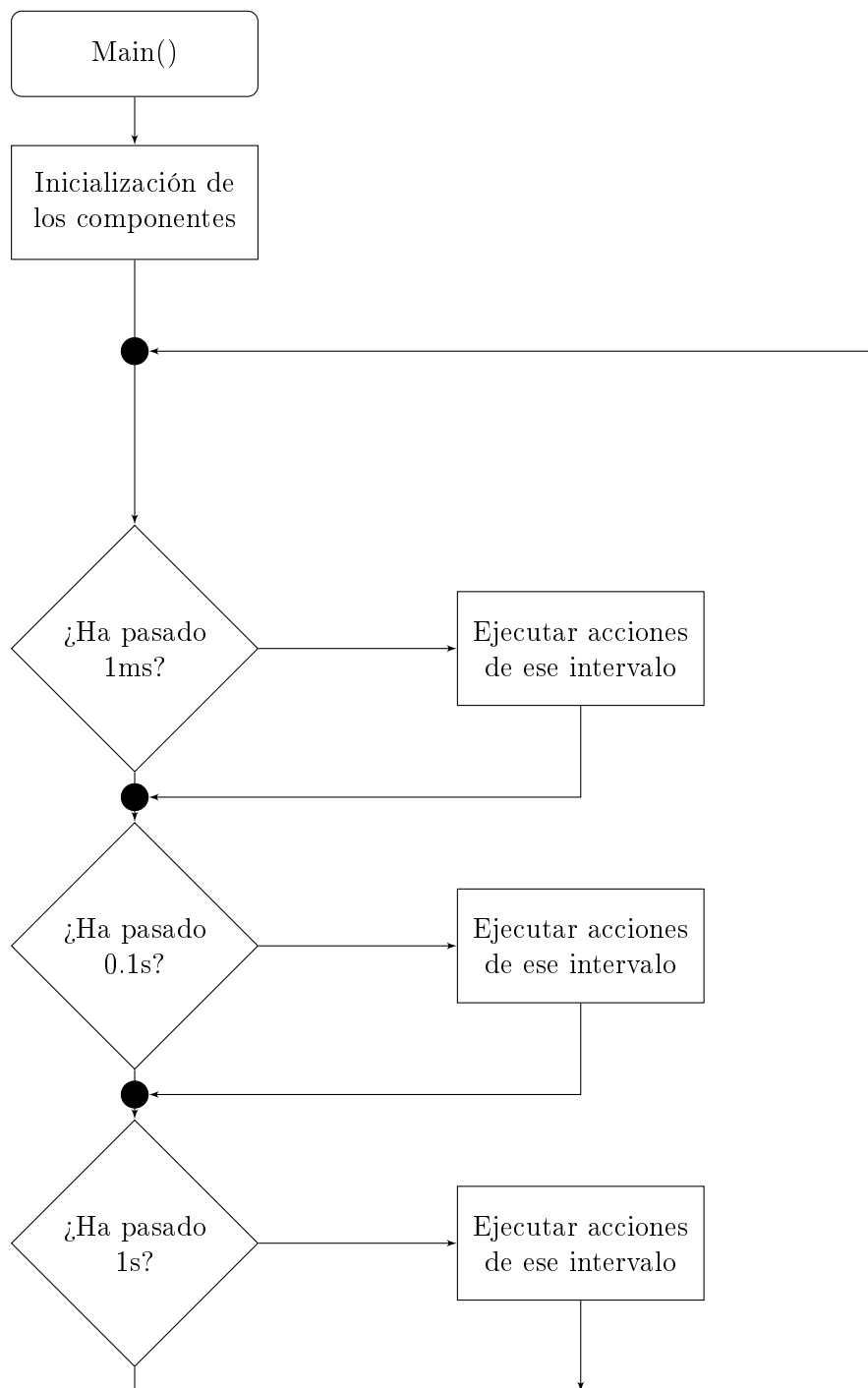


Figura 10: Diagrama de flujo del firmware.

Documentación a entregar

- Un breve documento en PDF (que no incluya código fuente), en el que se presente:
 - Esquemático.
 - Descripción del funcionamiento del sistema planteado.
 - Diagrama de flujo de la máquina de estados (si la hay).
 - Limitaciones del diseño.
- Proyecto del PSoC Creator.

Todo esto se entregará a través de la tarea habilitada en la plataforma Moodle de la asignatura. Además se deberá demostrar en clase el proyecto funcionando.

Práctica 2: Desarrollo de Aplicaciones Digitales con PSoC

1. Manejo del LCD

Objetivos

El alumno deberá familiarizarse con la pantalla LCD incluida en la placa de desarrollo y en su comunicación utilizando el PSoC.

Materiales Necesarios

1. Software para la programación PSoC Creator de Cypress.
2. Kit de Desarrollo PSoC CY8KIT-001.

Configuración Hardware

Creamos un proyecto nuevo, al que llamaremos LAB2A dentro del espacio de trabajo DSE.

En el esquemático, añadimos de la lista de componentes, dentro de Display, “Character LCD”. Editamos el componente, manteniendo a “none” el conjunto de caracteres y asegurándonos que está seleccionado las rutinas de conversión de ASCII a números. Además, fijaremos el nombre a “CharLCD”.

Como se puede observar en el esquemático de la placa de desarrollo proporcionado por el profesor de prácticas, la pantalla LCD está conectada a través de los pines P2[6:0].

Implementación del Firmware

Con el fin de tener nuestra pequeña librería que maneje el display, creamos un fichero cabecera, llamado “display.h” con el siguiente contenido:

```
/* =====  
 * Diseño de Sistemas Electrónicos  
 * LAB2A  
 * Fernando Martínez Martí ( fmartinezmarti@ugr.es )  
 * =====  
 */  
void DisplayWelcome( void );  
void DisplayTitle( void );  
void DisplayCount( uint8 count );
```

Práctica 2: Desarrollo de Aplicaciones Digitales con PSoC

```
/* [] END OF FILE */
```

Mientras que el código fuente asociado denominado “display.c” contiene:

```
/* =====
 * Diseño de Sistemas Electrónicos
 * LAB2A
 * Fernando Martínez Martí ( fmartinezmarti@ugr.es )
 * =====
 */
#include <device.h>
#include "display.h"

void DisplayWelcome( void ){
    CharLCD_ClearDisplay();
    CharLCD_Position(0U, 0U); /* fila , columna */
    CharLCD_PrintString( "PSoC Mola!" );
}

void DisplayTitle( void ){
    CharLCD_Position(1U, 0U);
    CharLCD_PrintString( "Contador:" );
}

void DisplayCount( uint8 count ){
    CharLCD_Position(1U, 12U);
    CharLCD_PrintNumber( ( uint16 ) count );
    CharLCD_PrintString( " " );
}
/* [] END OF FILE */
```

Mientras que en el main, implementamos:

```
/* =====
 * Diseño de Sistemas Electrónicos
 * LAB2A
 * Fernando Martínez Martí ( fmartinezmarti@ugr.es )
 * =====
 */
#include <device.h>
#include "display.h"

void main() {
    uint8 contador = 0u;

    CharLCD_Start(); // Inicializamos la LCD
    DisplayWelcome();
    DisplayTitle();

    for(;;) {
        DisplayCount( contador );
        contador++;
    }
}
```

```
        if (contador>99){
            contador=0;
        }
    }
}
/* [] END OF FILE */
```

2. Puertas Lógicas en el PSoC

Objetivos

Conocer el funcionamiento y las posibilidades de la lógica digital utilizando un PSoC. En este caso, el alumno implementará un simple puerta lógica para comprobar su funcionamiento.

Materiales Necesarios

1. Software para la programación PSoC Creator de Cypress.
2. Kit de Desarrollo PSoC CY8KIT-001.

Configuración Hardware

Para diseñar este esquemático, el alumno simplemente tendrá que añadir dos pines digitales de entrada y un pin digital de salida. Además, se utilizará la puerta lógica que él quiera, en este caso, se ha elegido una puerta AND, que se puede añadir al esquemático desde “Digital/Logic/And”. El esquemático final sería como el de la figura 1.

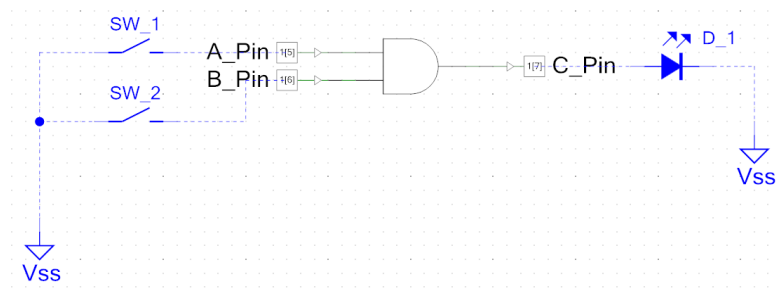


Figura 1: Esquemático LAB2B.

Asignamos los dos pines de entrada a los botones (P1_5 y P1_6) y el pin de salida a un led (P1_7).

Implementación del Firmware

En este caso no es necesario programar el PSoC ya que el comportamiento de nuestro sistema será únicamente a través de los elementos hardware que PSoC Creator nos permite introducir.

Programación y Resultados

Conectamos los pines anteriores de entrada a dos botones (SW1 y SW2 por ejemplo) y otra conexión a uno de los LED. El comportamiento será el de la tabla 1.

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Tabla 1: Tabla Puerta AND.

3. Utilidades del PSoC para Sistemas Digitales

Objetivos

Este ejercicio pretende permitir al alumno que se introduzca en otros bloques digitales que componen el PSoC. Para ello, se realizará un diseño compuesto por los siguientes componentes:

- Un contador básico para contar los flancos de bajada de un pin.
- Un comparador digital para hacer un reset sobre el contador anterior.
- Una constante digital para proveer un valor fijo digital al comparador.
- Un detector de picos que produce un único pulso en los flancos de bajada a la entrada básica del contador.

Materiales Necesarios

1. Software para la programación PSoC Creator de Cypress.
2. Kit de Desarrollo PSoC CY8KIT-001.

Configuración Hardware

Creamos un nuevo proyecto con le nombre LAB2C. En el esquemático añadimos los siguientes componentes.

1. Edge Detector
2. Digital Comparator
3. Basic Counter
4. Status Register
5. Character LCD

A parte de esto, añadiremos un fuente de reloj y la configuraremos como una fuente existente con la misma frecuencia que el reloj del BUS principal (24MHz). El esquemático final seria el de la figura [2](#).

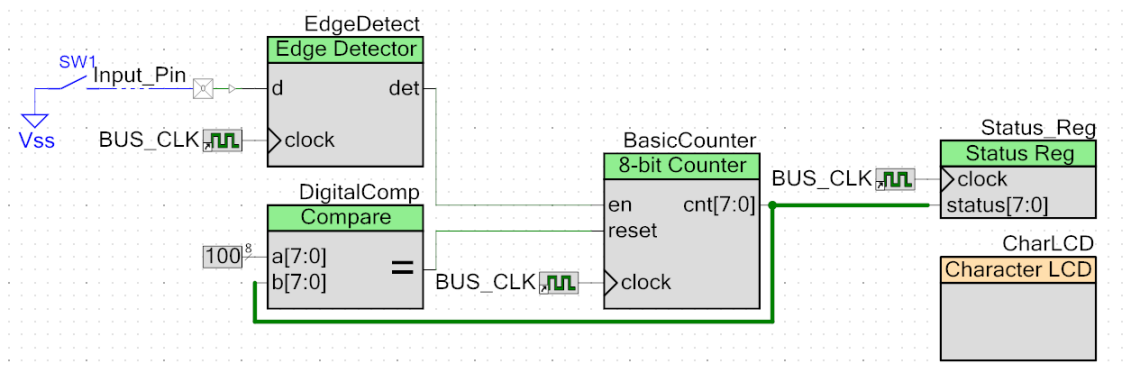


Figura 2: Esquemático LAB2C.

La configuración de todos los componentes será la de por defecto, excepto en el caso del contador que aumentaremos de 4 a 8 bits, el detector de flancos, que se configura en modo bajada (falling) y el comparador digital que se fijará la comparación a igual y se utilizará 8 bits. Además, fijaremos el modo de visualización del registro de estados a modo bus.

Asignamos el pin de entrada al botón (P1_5) y la LCD a los pines especificados por el esquemático(P2[6:0]).

Implementación del Firmware

Implementamos el siguiente código dentro del archivo “main.c”.

```
/* =====
 * Diseño de Sistemas Electrónicos
 * LAB2C
 * Fernando Martínez Martí ( fmartinezmarti@ugr.es )
 * =====
 */
#include <device.h>

void main() {
    uint8 count;
    LCD_Char_1_Start();
    for (;;) {
        LCD_Char_1_ClearDisplay();
        LCD_Char_1_Position(0,0);
        LCD_Char_1_PrintString("Count: ");
        count = Status_Reg_1_Read();
        LCD_Char_1_PrintDecUint16(count);
        CyDelay(100);
    }
}/* [] END OF FILE */
```

Programación y Resultados

El resultado de este ejercicio será ver en la LCD un contador que se actualizará conforme el usuario pulse el botón.

4. Diseño de un multiplicador de 2 bits

Objetivos

El alumno deberá implementar un multiplicador de dos bits, cuyas entradas se controlen con tres botones, un botón servirá para aumentar el valor del multiplicador, otro para disminuirlo, y otro para pasar al siguiente valor o al resultado.

Para el diseño del multiplicador, el alumno utilizará la herramienta de diseño de circuitos digitales “open source” **LogiSim**. Una vez abierta dicha herramienta, pulsando sobre proyecto analizar circuito, podremos añadir las entradas y salidas deseadas, rellenamos la tabla asociada a esas entradas y salidas, y pulsamos sobre “Crear Circuito”.

El LogiSim se puede descargar del siguiente enlace:

<http://ozark.hendrix.edu/~burch/logisim/>

Los valores que debemos fijar se pueden observar en la tabla 2.

A_0	A_1	B_0	B_1	R_0	R_1	R_2	R_3
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	1	0
1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	1	0	0
1	1	0	0	0	0	0	0
1	1	0	1	0	1	1	0
1	1	1	0	1	1	0	0
1	1	1	1	1	0	0	1

Tabla 2: Resultados del Multiplicador de 2 bits.

Materiales Necesarios

1. Software para la programación PSoC Creator de Cypress.
2. Kit de Desarrollo PSoC CY8KIT-001.

Documentación a entregar

- Un breve documento en PDF (que no incluya código fuente), en el que se presente:
 - Esquemático.
 - Descripción del funcionamiento del sistema planteado.
 - Diagrama de flujo de la máquina de estados (si la hay).
 - Limitaciones del diseño.
- Proyecto del PSoC Creator.

Todo esto se entregará a través de la tarea habilitada en la plataforma Moodle de la asignatura. Además se deberá demostrar en clase el proyecto funcionando.

5. Aplicación de un Sistema Digital

Objetivos

El alumno deberá implementar un juego para la resolución del acertijo infantil del “Granjero-Zorro-Gallina-Maiz” que tiene que cruzar un río. Las condiciones de este acertijo son muy simples: el zorro y la gallina no pueden quedarse solos en una orilla del río al igual que la gallina no puede quedarse sola con el maíz.

Para ello se implementará un sistema digital compuesto por tres grandes grupos:

1. Un sistema basado en una look-up-table que recogerá todas las posibles soluciones y asignará el valor 0x0F a las entradas que no correspondan a los pasos de resolución del acertijo.
2. Un sistema basado en biestables D para introducir un retardo hardware a fin de comprobar que el acertijo se ha resuelto en el orden previamente fijado.
3. Un control de errores mediante puertas lógicas (apoyarse de LogiSim).

Para el control del juego se utilizarán tantos botones como sea necesario y se mostrarán los resultados en la pantalla LCD.

Materiales Necesarios

1. Software para la programación PSoC Creator de Cypress.
2. Kit de Desarrollo PSoC CY8KIT-001.

Documentación a entregar

- Un breve documento en PDF (que no incluya código fuente), en el que se presente:
 - Esquemático.
 - Descripción del funcionamiento del sistema planteado.
 - Diagrama de flujo de la máquina de estados (si la hay).
 - Limitaciones del diseño.
- Proyecto del PSoC Creator.

Todo esto se entregará a través de la tarea habilitada en la plataforma Moodle de la asignatura. Además se deberá demostrar en clase el proyecto funcionando.

Práctica 3: Comunicaciones con el PSoC

1. UART

0.0.1. Objetivos

El alumno deberá partir del ejercicio que realizó para la práctica 1. En este caso, se añadirá la comunicación a través de un módulo UART, utilizando un cable UART-USB que se proporcionará en el laboratorio.

En este caso, el objetivo es mandar un mensaje por medio de una tecla que interrumpa o reinicie el parpadeo del LED que se encendía por medio la interrupción, así como modificar el duty cycle del LED excitado por la señal PWM.

Los mensajes se lo que el PSoC vaya a hacer se deben de mostrar en la pantalla LCD: duty 20 %, Parpadeo Off, ... Además, el PSoC debe de enviar algun mensaje de confirmación cuando reciba una orden a través del puerto serie.

En la plataforma Moodle se puede descargar un hyperterminal para aquellos alumnos que utilicen Windows 7 o superior.

Para la configuración del UART del PSoC configuraremos:

- 115200 Baudios
- 8 bits de datos
- Sin paridad
- 1 bit de parada
- Sin control de flujo
- Reloj Interno
- Interrupciones desactivadas
- Tamaño de los buffers de 4 bytes
- Transmisión hardware habilitada para el RS-485

Materiales Necesarios

1. Software para la programación PSoC Creator de Cypress.
2. Kit de Desarrollo PSoC CY8KIT-001.
3. Cable UART-USB.

Documentación a entregar

- Un breve documento en PDF (que no incluya código fuente), en el que se presente:
 - Esquemático.
 - Descripción del funcionamiento del sistema planteado.
 - Diagrama de flujo de la máquina de estados (si la hay).
 - Limitaciones del diseño.
- Proyecto del PSoC Creator.

Todo esto se entregará a través de la tarea habilitada en la plataforma Moodle de la asignatura. Además se deberá demostrar en clase el proyecto funcionando.

Práctica 4: Desarrollo de Aplicaciones Analógicas con PSoC

1. Voltímetro

0.0.2. Objetivos

Realizar un voltímetro digital para muestrear tensiones entre 0 y 3.3 voltios (V_{ssa} y V_{dda}). Los datos muestreados se deben de representar en la pantalla LCD del kit de desarrollo, esta representación se hará con 16 bits y se hará una tasa de conversión de 5000 muestras por segundo.

Para realizar este ejercicio el sistema puede contar únicamente con la entrada de tensión a muestrear.

Se deberá de realizar una calibración para ver como convertir los valores de tensión analógicos a digitales.

Materiales Necesarios

1. Software para la programación PSoC Creator de Cypress.
2. Kit de Desarrollo PSoC CY8KIT-001.

2. Generador de Funciones

0.0.3. Objetivos

Realizar un generador de funciones que cuente con cuatro tipo de funciones (cuadrada, seno, diente de sierra y triangular). Este generador tendrá una tensión de salida mínima de 0 voltios y máximo de 4.08 voltios.

El usuario debe de ser capaz de modificar la amplitud de la señal así como la frecuencia de oscilación.

Materiales Necesarios

1. Software para la programación PSoC Creator de Cypress.
2. Kit de Desarrollo PSoC CY8KIT-001.

Documentación a entregar de los dos ejercicios

- Un breve documento en PDF(que no incluya código fuente), en el que se presente:
 - Esquemático.
 - Descripción del funcionamiento del sistema planteado.
 - Diagrama de flujo de la maquina de estados (si la hay).
 - Limitaciones del diseño.
- Proyecto del PSoC Creator.

Todo esto se entregará a través de la tarea habilitada en la plataforma Moodle de la asignatura. Además se deberá demostrar en clase el proyecto funcionando.