Universidade Vigo

Guía resumida para utilizar el microcontrolador CY7C68013

Camilo Quintáns Graña
Miguel Ángel Domínguez Gómez
Vigo, 2011

Índice

1. DESCRIPCIÓN DEL HARDWARE 3
1.1. Introducción 3
1.2. Recursos de memoria 4
1.3. Control de la CPU 4
1.4. Puertos de entrada/salida 5
1.5. Temporizadores 6
1.6. Interrupciones 7
2. DESCRIPCIÓN DEL SOFTWARE 8
2.1. Notas sobre la documentación 8
2.2. Pasos para crear un proyecto para realizar un programa del microcontrolador 8
3. PRÁCTICA EJEMPLO: UTILIZACIÓN DE LOS PERIFÉRICOS DEL MICROCONTROLADOR 11
3.1. Objetivos 11
3.2. Enunciado 11
3.3. Tareas del alumno previas a la asistencia al laboratorio 11
3.4. Tareas a realizar en el laboratorio 12
3.5. Solución programada en lenguaje C 13

1. DESCRIPCIÓN DEL HARDWARE

1.1. Introducción

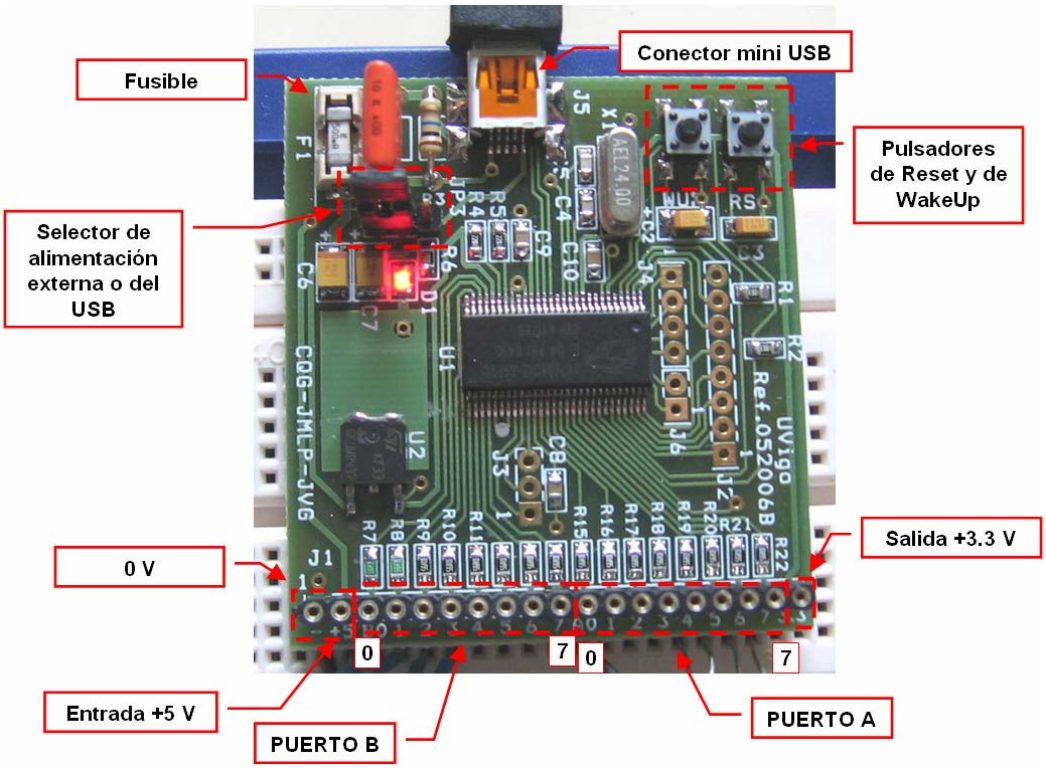

Figura 1.1. Placa para realizar prácticas con el microcontrolador CY7C68013 con conexión USB y memoria EEPROM.

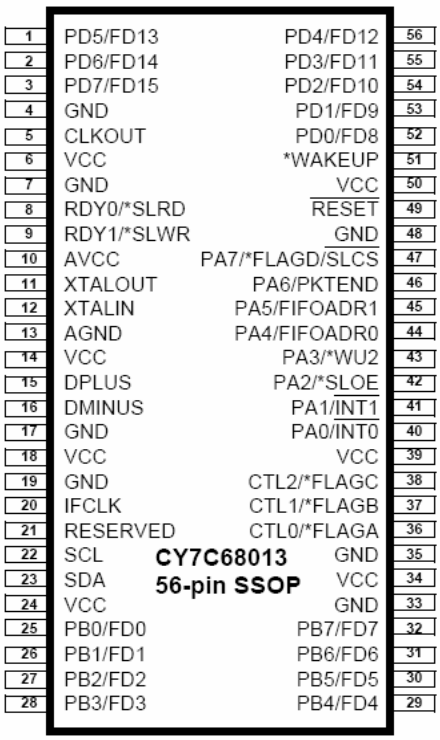

Figura 1.2. Descripción de los terminales de entradas y salidas del microcontrolador CY7C68013.

1.2. Recursos de memoria

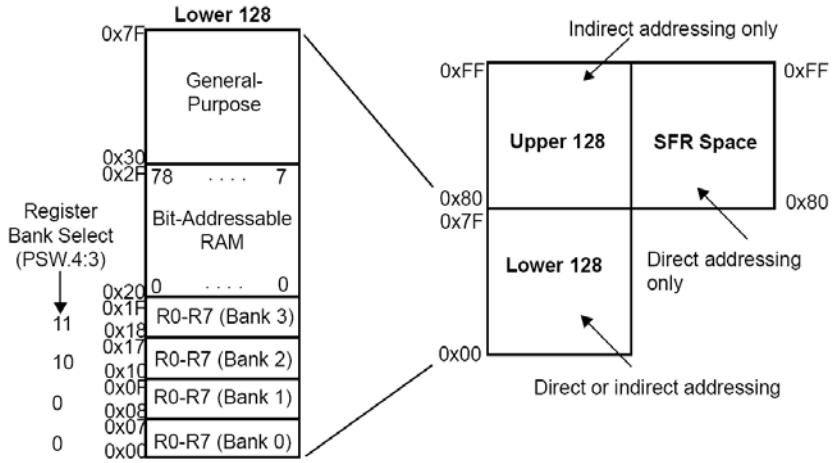

Figura 1.3. Mapa de la memoria interna de datos (256 bytes).

1.3. Control de la CPU

La velocidad de procesado se puede configurar programando los bits 3 y 4 del registro **CPUCS**, de acuerdo a la Tabla 1.1. Sin embargo, esto no es suficiente, para que tenga efecto dicho cambio, se debe incluir en los ficheros fuente del proyecto el archivo de biblioteca **ezusb.lib**, que se encuentra en el siguiente directorio:

C:\Cypress\USB\Target\Lib\FX2

CPUCS — **CPU Control and Status** — **E600**

b7	b6	b5	b4	b3	b2	b1	b0
0	**0**	**PORTCSTB**	**CLKSPD1**	**CLKSPD0**	**CLKINV**	**CLKOE**	**0**
R	R	R/W	R/W	R/W	R/W	R/W	R
0	0	0	0	0	0	1	0

Figura 1.4. Registro de control de la CPU.

Tabla 1.1. Bits para configurar la frecuencia del reloj de la CPU.

CLKSPD1	CLKSPD0	CPU Clock
0	0	12 MHz (Default)
0	1	24 MHz
1	0	48 MHz
1	1	Reserved

Tabla 1.2. Registros especiales del microcontrolador.

x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
0	**IOA**	**IOB**	**IOC**	**IOD**	**SCON1**	PSW	ACC	B
1	SP	**EXIF**	**INT2CLR**	**IOE**	**SBUF1**			
2	DPL0	**MPAGE**	**INT4CLR**	**OEA**				
3	DPH0			**OEB**				
4	**DPL1**			**OEC**				
5	**DPH1**			**OED**				
6	**DPS**			**OEE**				
7	PCON							
8	TCON	SCON0	**IE**	**IP**	**T2CON**	**EICON**	**EIE**	**EIP**
9	TMOD	SBUF0						
A	TL0	**AUTOPTRH1**	**EP2468STAT**	**EP01STAT**	**RCAP2L**			
B	TL1	**AUTOPTRL1**	**EP24FIFOFLGS**	**GPIFTRIG**	**RCAP2H**			
C	TH0		**EP68FIFOFLGS**		**TL2**			
D	TH1	**AUTOPTRH2**		**GPIFSGLDATH**	**TH2**			
E	**CKCON**	**AUTOPTRL2**		**GPIFSGLDATLX**				
F			**AUTOPTRSETUP**	**GPIFSGLDATLNOX**				

1.4. Puertos de entrada/salida

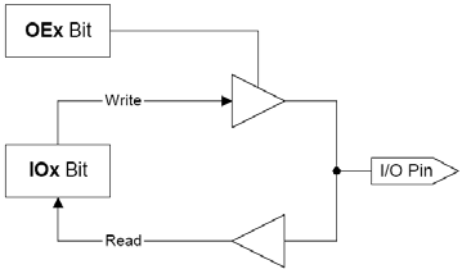

Figura 1.5. Esquema de los puertos de E/S.

Tabla 1.3. Registros de habilitación de los puertos de E/S.

OEA	Port A Output Enable						SFR 0xB2
b7	b6	b5	b4	b3	b2	b1	b0
D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

OEB	Port B Output Enable						SFR 0xB3
b7	b6	b5	b4	b3	b2	b1	b0
D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

OED	Port D Output Enable						SFR 0xB5
b7	b6	b5	b4	b3	b2	b1	b0
D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

Tabla 1.4. Registros de datos de los puertos de E/S.

IOA	Port A (Bit-Addressable)						SFR 0x80
b7	b6	b5	b4	b3	b2	b1	b0
D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
x	x	x	x	x	x	x	x

IOB	Port B (Bit-Addressable)						SFR 0x90
b7	b6	b5	b4	b3	b2	b1	b0
D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
x	x	x	x	x	x	x	x

IOD	Port D (Bit-Addressable)						SFR 0xB0
b7	b6	b5	b4	b3	b2	b1	b0
D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
x	x	x	x	x	x	x	x

1.5. Temporizadores

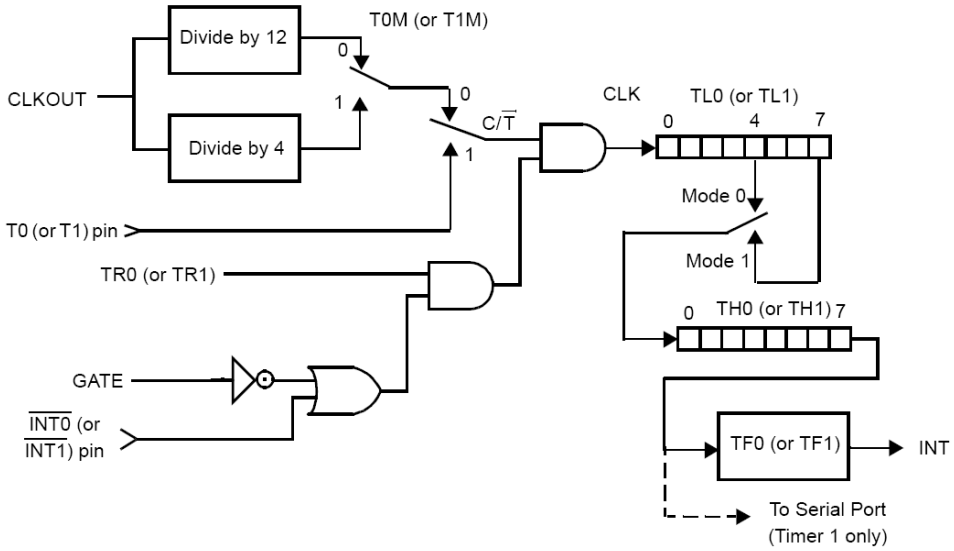

Figura 1.6. Diagrama de bloques de los temporizadores 0/1 modos 0/1.

Tabla 1.5. Registro TMOD (0x89) de configuración del temporizador.

Bit	Function
TMOD.7	**GATE1** - Timer 1 gate control. When GATE1 = 1, Timer 1 will clock only when $\overline{INT1}$ = 1 and TR1 (TCON.6) = 1. When GATE1 = 0, Timer 1 will clock only when TR1 = 1, regardless of the state of $\overline{INT1}$.
TMOD.6	**C/$\overline{T}$1** - Counter/Timer select. When C/$\overline{T}$1 = 0, Timer 1 is clocked by CLKOUT/4 or CLKOUT/12, depending on the state of T1M (CKCON.4). When C/$\overline{T}$1 = 1, Timer 1 is clocked by high-to-low transitions on the T1 pin.
TMOD.5 TMOD.4	**M1** - Timer 1 mode select bit 1. **M0** - Timer 1 mode select bit 0. M1 M0 Mode 0 0 Mode 0 : 13-bit counter 0 1 Mode 1 : 16-bit counter 1 0 Mode 2 : 8-bit counter with auto-reload 1 1 Mode 3 : Timer 1 stopped
TMOD.3	**GATE0** - Timer 0 gate control, When GATE0 = 1, Timer 0 will clock only when $\overline{INT0}$ = 1 and TR0 (TCON.4) = 1. When GATE0 = 0, Timer 0 will clock only when TR0 = 1, regardless of the state of $\overline{INT0}$.
TMOD.2	**C/$\overline{T}$0** - Counter/Timer select. When C/$\overline{T}$0 = 0, Timer 0 is clocked by CLKOUT/4 or CLKOUT/12, depending on the state of T0M (CKCON.3). When C/$\overline{T}$0 = 1, Timer 0 is clocked by high-to-low transitions on the T0 pin.
TMOD.1 TMOD.0	**M1** - Timer 0 mode select bit 1. **M0** - Timer 0 mode select bit 0. M1 M0 Mode 0 0 Mode 0 : 13-bit counter 0 1 Mode 1 : 16-bit counter 1 0 Mode 2 : 8-bit counter with auto-reload 1 1 Mode 3 : Two 8-bit counters

Tabla 1.6. Registro TCON (0x88) de control del temporizador.

Bit	Function
TCON.7	**TF1** - Timer 1 overflow flag. Set to 1 when the Timer 1 count overflows; automatically cleared when the FX2 vectors to the interrupt service routine.
TCON.6	**TR1** - Timer 1 run control. 1 = Enable counting on Timer 1.
TCON.5	**TF0** - Timer 0 overflow flag. Set to 1 when the Timer 0 count overflows; automatically cleared when the FX2 vectors to the interrupt service routine.
TCON.4	**TR0** - Timer 0 run control. 1 = Enable counting on Timer 0.
TCON.3	**IE1** - Interrupt 1 edge detect. If external interrupt 1 is configured to be edge-sensitive (IT1 = 1), IE1 is set when a negative edge is detected on the $\overline{INT1}$ pin and is automatically cleared when the FX2 vectors to the corresponding interrupt service routine. In this case, IE1 can also be cleared by software. If external interrupt 1 is configured to be level-sensitive (IT1 = 0), IE1 is set when the $\overline{INT1}$ pin is 0 and automatically cleared when the $\overline{INT1}$ pin is 1. In level-sensitive mode, software cannot write to IE1.
TCON.2	**IT1** - Interrupt 1 type select. $\overline{INT1}$ is detected on falling edge when IT1 = 1; $\overline{INT1}$ is detected as a low level when IT1 = 0.
TCON.1	**IE0** - Interrupt 0 edge detect. If external interrupt 0 is configured to be edge-sensitive (IT0 = 1), IE0 is set when a negative edge is detected on the $\overline{INT0}$ pin and is automatically cleared when the FX2 vectors to the corresponding interrupt service routine. In this case, IE0 can also be cleared by software. If external interrupt 0 is configured to be level-sensitive (IT0 = 0), IE0 is set when the $\overline{INT0}$ pin is 0 and automatically cleared when the $\overline{INT0}$ pin is 1. In level-sensitive mode, software cannot write to IE0.
TCON.0	**IT0** - Interrupt 0 type select. $\overline{INT0}$ is detected on falling edge when IT0 = 1; $\overline{INT0}$ is detected as a low level when IT0 = 0.

1.6. Interrupciones

Tabla 1.7. Interrupciones del FX2.

FX2 Interrupt	Source	Interrupt Vector	Natural Priority
IE0	$\overline{INT0}$ Pin	0x0003	1
TF0	Timer 0 Overflow	0x000B	2
IE1	$\overline{INT1}$ Pin	0x0013	3
TF1	Timer 1 Overflow	0x001B	4
RI_0 & TI_0	USART0 Rx & Tx	0x0023	5
TF2	**Timer 2 Overflow**	0x002B	6
Resume	**WAKEUP / WU2 Pin or USB Resume**	0x0033	0
RI_1 & TI_1	**USART1 Rx & Tx**	0x003B	7
USBINT	**USB**	0x0043	8
I²CINT	**I²C-Compatible Bus**	0x004B	9
IE4	**GPIF / FIFOs / $\overline{INT4}$ Pin**	0x0053	10
IE5	**$\overline{INT5}$ Pin**	0x005B	11
IE6	**INT6 Pin**	0x0063	12

Nota: Hay que tener en cuenta que el terminal de entrada para de interrupción 0 es el bit 0 del puerto A (**PA.0**), tal y como se muestra en la Figura 1.2.

Tabla 1.8. Registro de habilitación de las interrupciones del FX2.

Bit	Function
IE.7	**EA** - Global interrupt enable. Controls masking of all interrupts except USB wakeup (resume). EA = 0 disables all interrupts except USB wakeup. When EA = 1, interrupts are enabled or masked by their individual enable bits.
IE.6	**ES1** - Enable Serial Port 1 interrupt. ES1 = 0 disables Serial port 1 interrupts (TI_1 and RI_1). ES1 = 1 enables interrupts generated by the TI_1 or RI_1 flag.
IE.5	**ET2** - Enable Timer 2 interrupt. ET2 = 0 disables Timer 2 interrupt (TF2). ET2=1 enables interrupts generated by the TF2 or EXF2 flag.
IE.4	**ES0** - Enable Serial Port 0 interrupt. ES0 = 0 disables Serial Port 0 interrupts (TI_0 and RI_0). ES0=1 enables interrupts generated by the TI_0 or RI_0 flag.
IE.3	**ET1** - Enable Timer 1 interrupt. ET1 = 0 disables Timer 1 interrupt (TF1). ET1=1 enables interrupts generated by the TF1 flag.
IE.2	**EX1** - Enable external interrupt 1. EX1 = 0 disables external interrupt 1 (INT1). EX1=1 enables interrupts generated by the $\overline{\text{INT1}}$ pin.
IE.1	**ET0** - Enable Timer 0 interrupt. ET0 = 0 disables Timer 0 interrupt (TF0). ET0=1 enables interrupts generated by the TF0 flag.
IE.0	**EX0** - Enable external interrupt 0. EX0 = 0 disables external interrupt 0 ($\overline{\text{INT0}}$). EX0=1 enables interrupts generated by the $\overline{\text{INT0}}$ pin.

2. DESCRIPCIÓN DEL SOFTWARE

2.1. Notas sobre la documentación

Para programar el microcontrolador se utiliza el programa **EZ-USB Control Panel** con la opción FX2.

Una vez instalado el programa **µVision2** (de la empresa Keil Software, Inc.), en el directorio **C:\Keil\C51\HLP**, se encuentra el archivo **GS51.pdf** del manual de usuario. En este manual se debe estudiar, principalmente, el Capítulo 4, que explica cómo crear un proyecto, y los capítulos 6 y 7, relativos a cómo usar el depurador.

En el mismo directorio **C:\Keil\C51\HLP** también se encuentran los manuales de programador en lenguaje ensamblador (fichero **A51.pdf**) y en lenguaje C (fichero **C51.pdf**).

Por otro lado, en el directorio **C:\Cypress\USB\Doc\FX2**, se encuentra el archivo FX2 **TechRefManual.pdf** del manual del microcontrolador.

2.2. Pasos para crear un proyecto para realizar un programa del microcontrolador

1.- Ejecutar el programa mediante la orden **Inicio/Todos los programas/Keil uVision2** y crear un nuevo proyecto con la opción **Proyect/Nex Projet**. Luego, en el fichero de codificación del programa, incluir en la cabecera las siguientes directivas de compilación:

En el caso de que se programe se codifique en lenguaje ensamblador:

```
$NOMOD51  ; Inhibe registros del 8051 predefinidos
$INCLUDE (fx2regs.inc)
```

En el caso de que se codifique en lenguaje C:

```
#include "Fx2.h"
#include "Fx2regs.h"
```

- Salvar el archivo del código fuente con extensión **.c** si se codifica en lenguaje C, y con extensión **.A51** si se codifica en ensamblador.
- Fijarse si el módulo fuente (el archivo con el programa) está en la ventana del manejador de proyectos de **uVision2**, si no es así, añadirlo con la opción **Add Files to Group**.

2.- En la opción **Project/Select Device for Target 'Target 1'** seleccionar el dispositivo **EZ-USB FX2** (Figura 2.1).

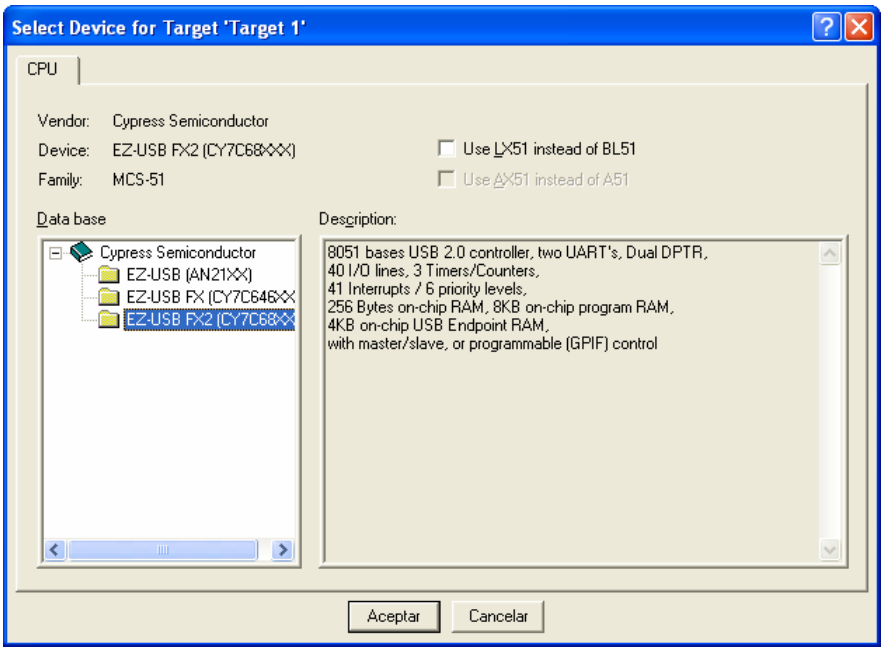

Figura 2.1

3.- En la opción: **Project/Options for Target 'Target 1'/Target** configurar la frecuencia del reloj (Figura 2.2) y, en la opción **Project/Options for Target 'Target 1'/Output** activar la orden de generar el fichero de salida **.hex**.

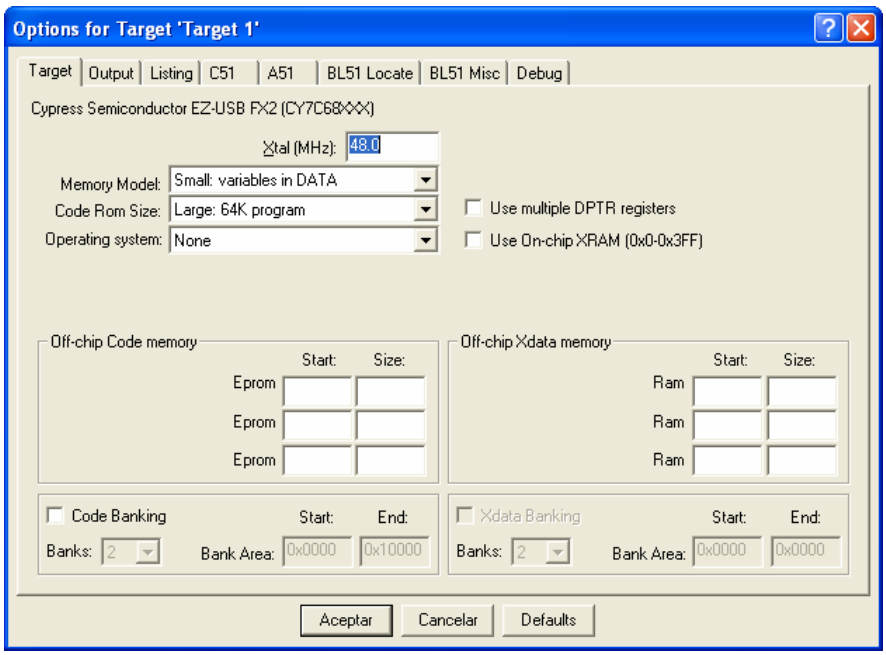

Figura 2.2

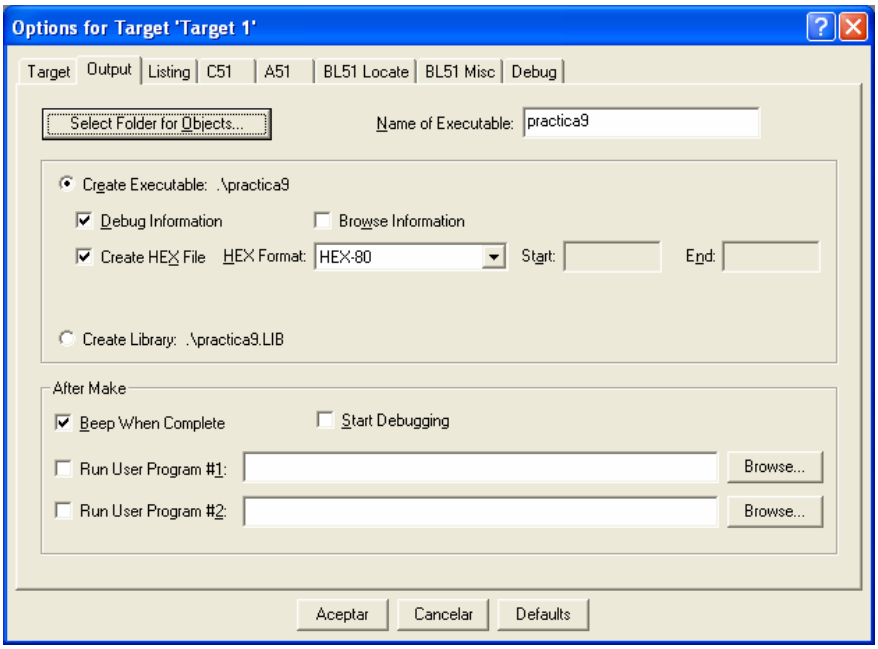

Figura 2.3

4.- En la opción **Project/File Extensions, Books and Enviroment/Enviroment setup** añadir la siguiente ruta de los archivos de las bibliotecas del dispositivo (Figura 2.4):

C:\Cypress\USB\Target\Inc

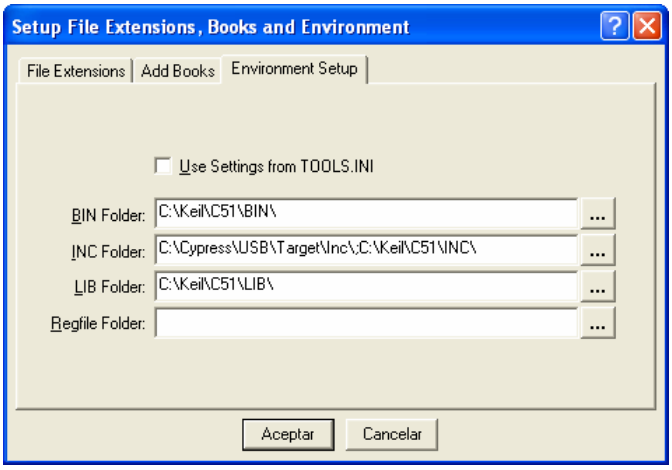

Figura 2.4

5.- Conectar al puerto USB la placa del microcontrolador y ejecutar el programa:

Inicio/Todos los programas/Cypress/EZ-USB Control Panel

Y, con la opción **Download** descargar el archivo **.hex** generado en el paso anterior (Figura 2.5).

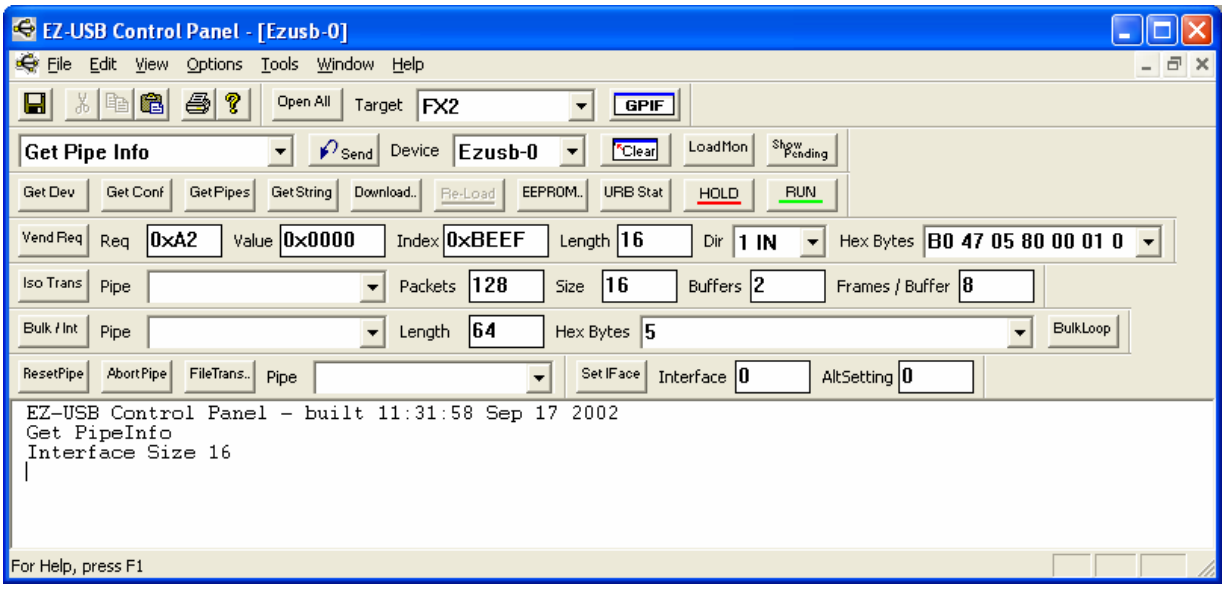

Figura 2.5. Interfaz del panel de control del microcontrolador.

3. PRÁCTICA EJEMPLO: UTILIZACIÓN DE LOS PERIFÉRICOS DEL MICROCONTROLADOR

3.1. Objetivos

El objetivo de esta práctica es que el alumno aprenda a programar un microcontrolador y a controlar dispositivos externos mediante consulta e interrupción. Para alcanzar estos objetivos se siguen los siguientes pasos:

Primero se debe realizar un programa de control mediante el microcontrolador que cumpla las especificaciones del enunciado de la práctica. A continuación, se simula su funcionamiento con el software **µVision2** en modo *debugger*. Y, una vez simulado y comprobado su correcto funcionamiento, se conectan en la placa de prototipos los periféricos necesarios para realizar físicamente el control propuesto y, finalmente, se programa el microcontrolador y se verifican las especificaciones.

3.2. Enunciado

Se desea realizar en un microcontrolador un programa de control sencillo para iluminar secuencialmente una serie de 4 LEDs. El microcontrolador debe tener habilitada una interrupción externa de forma que cuando se reciba un pulso en el terminal correspondiente a dicha interrupción, el programa debe atenderla activando un contador que realizará una cuenta ascendente desde 0 hasta 9. Una vez que el contador llega a 9 el microcontrolador seguirá con el programa principal de encendido de LEDs en el punto donde lo había dejado.

En esta práctica se utilizará el microcontrolador EZ-USB FX2 de 56 terminales de la firma CYPRESS, que es una versión extendida del microcontrolador 8051. Para programar el microcontrolador se utilizará el software µVision2 de la firma KEIL, que es un entorno de desarrollo de programas para la familia de microcontroladores MCS-51. La programación se puede realizar tanto en lenguaje ensamblador como en lenguaje C.

3.3. Tareas del alumno previas a la asistencia al laboratorio

El alumno debe realizar las siguientes tareas antes de asistir al laboratorio:

- Lectura de la documentación sobre el microcontrolador.
- Estudio del diseño y la programación del microcontrolador.
- Estudio del manejo de interrupciones.

3.4. Tareas a realizar en el laboratorio

Se comienza por realizar el montaje del hardware. En la Figura 3.1 se muestra una fotografía de la placa de prototipos con los circuitos y conexiones correspondientes a la práctica.

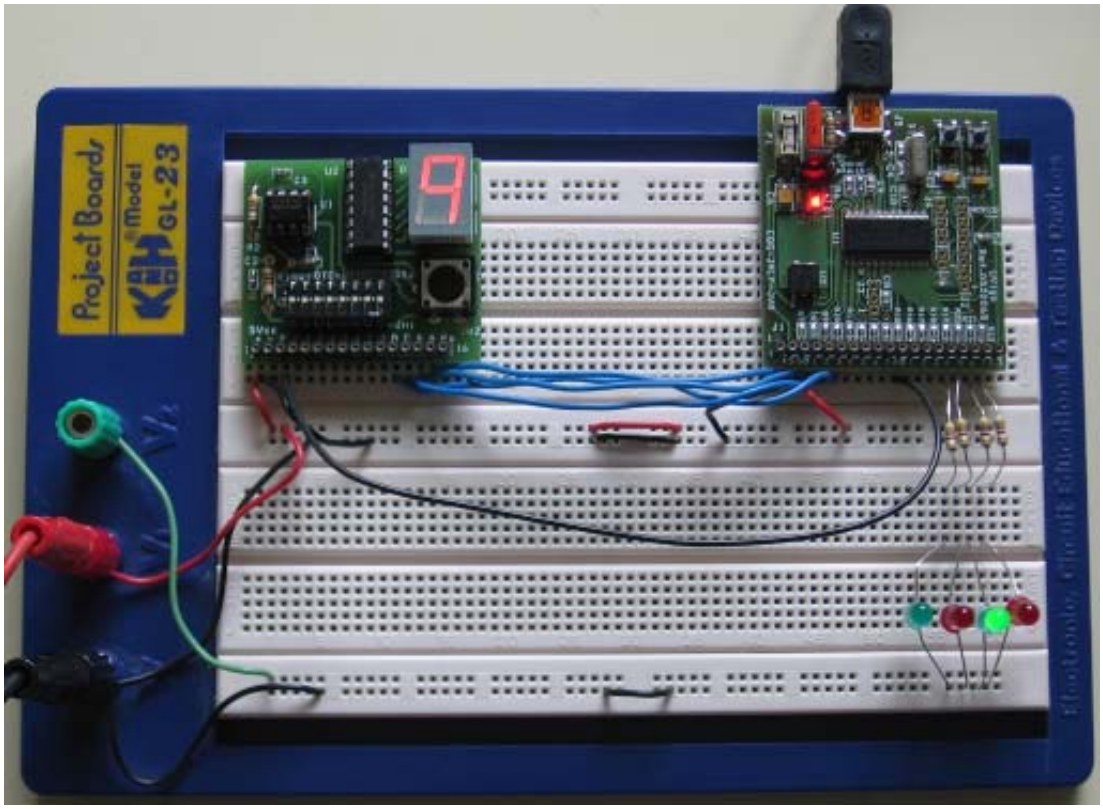

Figura 3.1. Fotografía de la placa de prototipos con los circuitos de la practica.

En la esquina superior izquierda se encuentra la placa de periféricos digitales y, a su derecha, la placa del microcontrolador. Se observa como se conectan los 4 bits altos del puerto A los LEDs con una resistencia en serie. Los cuatro bits de menor peso del puerto B a las entradas A, B, C y D de la placa de periféricos correspondientes al display. Y la interrupción generada con la salida del generador de pulsos manuales (salida P de la placa de periféricos) a la entrada de interrupciones del microcontrolador (bit 0 del puerto A).

3.5. Solución programada en lenguaje C

```
#include "Fx2.h"
#include "Fx2regs.h"
unsigned char contador;

// función que realiza un retardo de 1 segundo
void retardo (void)
{
    int i=0;
    // realizar un bucle para que se realicen 100 temporizaciones (1 segundo)
    for (i=0; i<100; i++)
        {
            TL0 = 0xEF; // Cargar el valor inicial del temporizador en los registros TL0 y TH0 para que
            TH0 = 0xD8; // cuente hasta 10.000 microsegundos
            TCON = 0x11; // iniciar la temporización del temporizador 0
            do
                {
                } while (TF0 == 0);
        }
}

// función que atiende la interrupción 0 (pin PA.0)
void interrup0 (void) interrupt 0
{
    IOB = 0x00; //se inicializa el puerto B
    contador = 0;
    // se realiza un contaje de 0 a 9 con un retardo de 1 segundo entre cada incremento
    // y se asigna el valor del contador al puerto B
    do
    {   retardo ();
        contador++;
        IOB = contador;
    }
    while (contador < 9);
}
//Programa principal
main (void)
{
    unsigned int i=0;
    IT0 = 1; // Interrupción externa 0 se activa en el flanco de bajada
    EX0 = 1; // Desinhibe interrupción externa 0
    EA = 1; // Desinhibe globalmente todas las interrupciones
    OEB = 0x0F; // Selección de los 4 bits de menor peso del puerto B como terminales de salida
    OEA = 0xF0; // Selección de los 4 bits de mayor peso del puerto A como terminales de salida
    IOB = 0x00; // Inicialización del puerto B
    TMOD = 0x01; // Selección del modo 1 (temporizador de 16 bits) para el temporizador 0

    // bucle para ir encendiendo secuencialmente 4 leds con un retardo de 1 segundo
    while (1)
        {   IOA = 0x10;
            retardo ();
            IOA = 0x20;
            retardo ();
            IOA = 0x40;
            retardo ();
            IOA = 0x80;
            retardo ();
        }
}
```