04 PRÁCTICA FINAL

NO. 04

Contador binario con pantalla LCD

SESIÓN	DURACIÓN	UNIDADES	PLATAFORMA
4 de 4	4 horas	7 y 8	PIC16F1829 (en C)

Introducción

En la Práctica 2 el 74LS74 contaba con flip-flops: cada pulso de reloj sumaba uno y los LEDs mostraban el número en binario. Hoy el mismo trabajo lo hace un **microcontrolador** con un programa. Cada vez que se presiona el botón **SUMAR** la cuenta sube en uno; cuatro LEDs la muestran en binario y la pantalla LCD la escribe en decimal y en binario. Un puente llamado **BORRAR** la regresa a cero, como la entrada CLR del flip-flop.

El PIC16F1829 es un microcontrolador de Microchip de 20 terminales. Se programa en lenguaje **C**, el mismo en que se basa el Arduino. A diferencia del Arduino, no viene en una placa ni tiene un cargador por USB: el programa se **compila** en la computadora, se **graba** en el chip con un programador de PIC y después el chip se coloca en el protoboard como cualquier integrado.

El circuito se arma en tres etapas: primero el PIC y los LEDs, luego las entradas y al final la pantalla. Cada etapa se prueba antes de pasar a la siguiente.

NOTA

El circuito hace exactamente lo mismo que el de los grupos con Arduino Uno: el mismo botón, el mismo puente, los mismos cuatro LEDs y la misma pantalla. Solo cambian el microcontrolador y la forma de cargar el programa.

ADVERTENCIA

El PIC16F1829 **no es un PICAXE**. Por dentro el PICAXE-20M2 es el mismo chip, pero viene grabado con un programa de fábrica y su etiqueta dice PICAXE-20M2. Un chip con la etiqueta PIC16F1829-I/P está vacío: el PICAXE Editor no lo reconoce. Se programa en C, como se explica en esta guía.

OBJETIVOS

- Compilar un programa en C y grabarlo en un microcontrolador.
- Armar el circuito mínimo de un microcontrolador: alimentación y reinicio.
- Conectar salidas (LEDs) y entradas (botón y puente) al PIC.
- Leer un número de 4 bits en binario y pasarlo a decimal.
- Mostrar datos en una pantalla LCD por el bus I^2C.
- Cambiar el comportamiento del circuito modificando el programa.

MATERIALES Y EQUIPO

- PIC16F1829-I/P (encapsulado de 20 terminales).
- Programador de PIC con su cable: la tarjeta de zócalo iCP03 con su programador, o un PICkit 3, PICkit 4, PICkit 5 o MPLAB Snap.
- Computadora con los programas de la tabla siguiente, ya instalados.
- MO-MB102 con ADP-5HB y el UT33B+.
- Pantalla LCD 16×2 con módulo I^2C y cuatro cables *jumper* macho-hembra.
- 4 LEDs y 4 resistencias de 330 Ω.
- 1 botón pulsador, 3 resistencias de 10 kΩ y 1 capacitor de 100 nF.
- Protoboard y cables *jumper*.

Programas que debe tener instalados

Instálelos **antes** de la clase: juntos ocupan varios gigabytes, la descarga tarda y la instalación pide permisos de administrador. Todos son gratuitos.

Programa	Para qué sirve	Dónde se obtiene
MPLAB X IDE (versión 6)	Es el editor: ahí se escribe el programa, se compila y, con un PICkit o un Snap, se graba. Trae **MPLAB IPE**, el programa para grabar chips	`microchip.com/mplab/mplab-x-ide`
MPLAB XC8 (modo gratuito, *Free*)	El **compilador**: convierte el programa en C en un archivo `.hex`, que es lo que se graba en el chip	`microchip.com/mplab/compilers`
Paquete PIC16F1xxxx_DFP	Los datos del PIC16F1829 (registros y memoria). MPLAB X lo descarga solo la primera vez que se elige el chip; si no, en `Tools > Packs`	Dentro de MPLAB X
Software del programador	Graba el `.hex` en el chip. Con PICkit 3, 4, 5 o Snap: **MPLAB IPE**. Con el programador de la tarjeta iCP03: el software y el controlador de su fabricante	Con el programador o en la página de su fabricante

1. Instale primero **MPLAB X IDE**. En la instalación deje marcados *MPLAB X IDE* y *MPLAB IPE*.
2. Instale **MPLAB XC8**. Cuando pregunte por la licencia, elija *Free*.
3. Abra MPLAB X y revise en `Tools > Options > Embedded > Build Tools` que aparezca **XC8**. Si no aparece, pulse `Scan for Build Tools`.
4. Conecte el programador. Si es un PICkit o un Snap, abra **MPLAB IPE** y compruebe que lo reconoce. Si es el de la tarjeta iCP03, instale su controlador, abra su software y busque en la lista de chips el **PIC16F1829**.

NOTA

No hacen falta el PICAXE Editor, el adaptador USB-TTL CP2102 ni el 74LS04: esos sirven para el PICAXE. MPLAB X y XC8 funcionan en Windows, macOS y Linux; el software de algunos programadores solo funciona en Windows.

PLANIFICACIÓN DE LA SESIÓN

Parte	Actividad	Tiempo
—	Apertura: contar en binario con 4 bits	20 min
A	Compilar, grabar el PIC y conectar los LEDs	65 min
B	Las entradas: SUMAR y BORRAR	35 min
C	La pantalla LCD	40 min
D	Retos: cambiar el programa	35 min
—	Evaluación en Socrative, desarmado y cierre	25 min
	Total	**220 min + 20 min**

Fundamento teórico

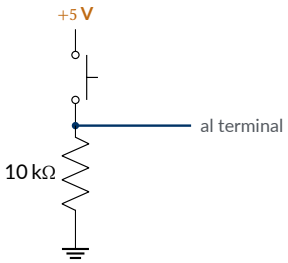
Contar con 4 bits

Cada LED es un bit. De derecha a izquierda valen 1, 2, 4 y 8: el número es la suma de los LEDs encendidos. Con 4 bits se cuenta de 0 a 15. Al sumar uno a 15 (1111) el resultado necesitaría un quinto bit, así que la cuenta **regresa a 0**, igual que un contador de flip-flops.

bit 3	bit 2	bit 1	bit 0	Valor
8	4	2	1	Peso de cada LED
0	1	1	0	$4 + 2 = 6$
1	0	1	1	$8 + 2 + 1 = 11$

Una entrada nunca queda al aire

El botón une el terminal con +5 V. Cuando está suelto, la resistencia de 10 kΩ lo lleva a `GND`. Sin ella la entrada queda **flotando** y la cuenta avanza sola, como pasó con las entradas TTL al aire en la Práctica 1.



- **Suelto:** el terminal lee 0.
- **Presionado:** el terminal lee 1. Por la resistencia pasan apenas $5\,\text{V}/10\,\text{k}\Omega = 0.5\,\text{mA}$.
- **El rebote:** al cerrarse, el contacto vibra unos milisegundos y la entrada cambia varias veces. El programa espera 20 ms y, además, no vuelve a contar hasta que se suelta el botón.

Compilar y grabar

Se trabaja en dos pasos, como en el Arduino, pero con programas separados:



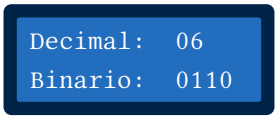
- **Compilar** revisa el programa y lo traduce a las instrucciones del chip. Si hay un error de escritura, aquí aparece.
- **Grabar** copia el `.hex` en la memoria del PIC. El chip lo conserva aunque se apague, y lo ejecuta cada vez que recibe alimentación.

El terminal de reinicio (MCLR)

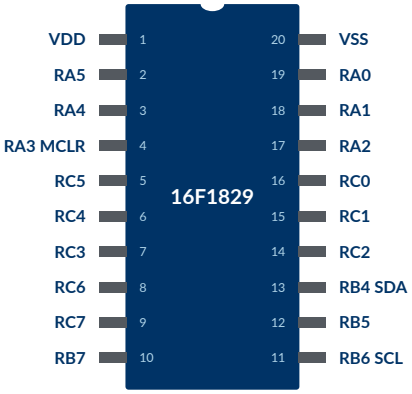
El terminal 4 se llama `MCLR`: mientras está en 0, el PIC se **reinicia** y no ejecuta el programa. Por eso lleva una resistencia de 10 kΩ a +5 V, que lo mantiene en 1. Es la misma idea que el `CLR` del flip-flop: una entrada que borra todo si queda en el nivel equivocado.

La pantalla por I^2C

El módulo soldado detrás de la LCD recibe los datos por solo dos cables: `SDA` (datos) al terminal 13 y `SCL` (reloj) al terminal 11. Su dirección suele ser `0x27`; algunos módulos usan `0x3F`. El programa escribe la dirección multiplicada por dos: `0x4E` o `0x7E`. Lo que se verá al final es esto:



Terminales



Terminal	Nombre	Uso
1 / 20	VDD / VSS	+5 V / GND
4	RA3 (MCLR)	10 kΩ a +5 V
5	RC5	Botón SUMAR
14	RC2	Puente BORRAR
7, 8, 9, 10	RC3, RC6, RC7, RB7	LEDs: bit 3 a bit 0
13 / 11	RB4 / RB6	SDA / SCL

Figura 04.1. PIC16F1829. Los terminales 2, 3, 6, 12 y 15 a 19 no se usan: el programa los deja como salidas en 0.

ADVERTENCIA

La alimentación está en las esquinas de la *izquierda*: VDD es el terminal 1 y VSS el 20. Es al revés que en los 74LS, donde VCC está en la esquina de arriba a la izquierda (el último terminal) y GND abajo a la derecha. Invertir la alimentación destruye el chip.

El circuito

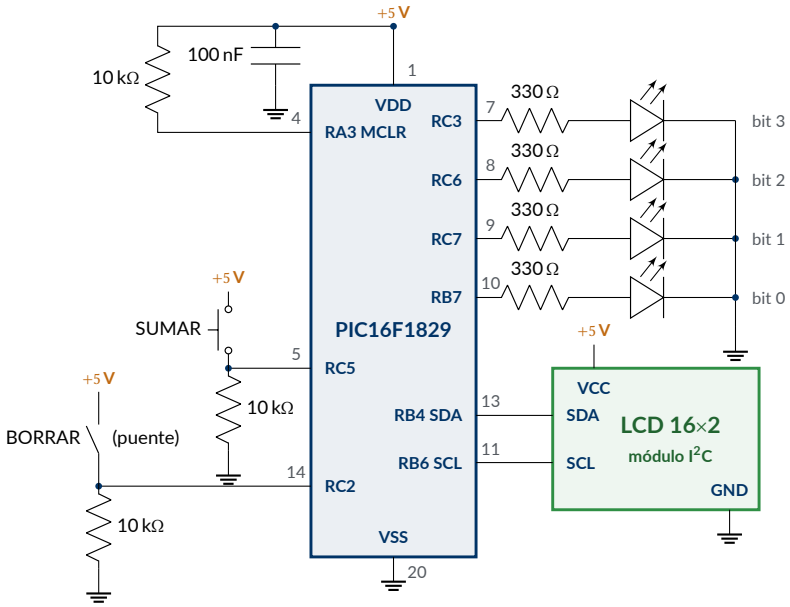


Figura 04.2. Esquema completo. Los números pequeños son los terminales. El puente BORRAR es un cable que se pone y se quita.

Tabla 04.1. Conexiones, terminal por terminal

Terminal	Se conecta a	Parte
1 (VDD)	Riel +	A
20 (VSS)	Riel −	A
4 (MCLR)	10 kΩ a +5 V	A
7, 8, 9, 10	Cada uno a una resistencia de 330 Ω y su LED: el 7 (RC3) es el bit 3 y el 10 (RB7) el bit 0	A
5 (RC5)	Botón SUMAR (el otro lado a +5 V) y 10 kΩ a GND	B
14 (RC2)	Puente BORRAR (el otro lado a +5 V) y 10 kΩ a GND	B
13 (SDA) / 11 (SCL)	SDA / SCL de la LCD	C
2, 3, 6, 12, 15 a 19	Libres	—

El circuito armado

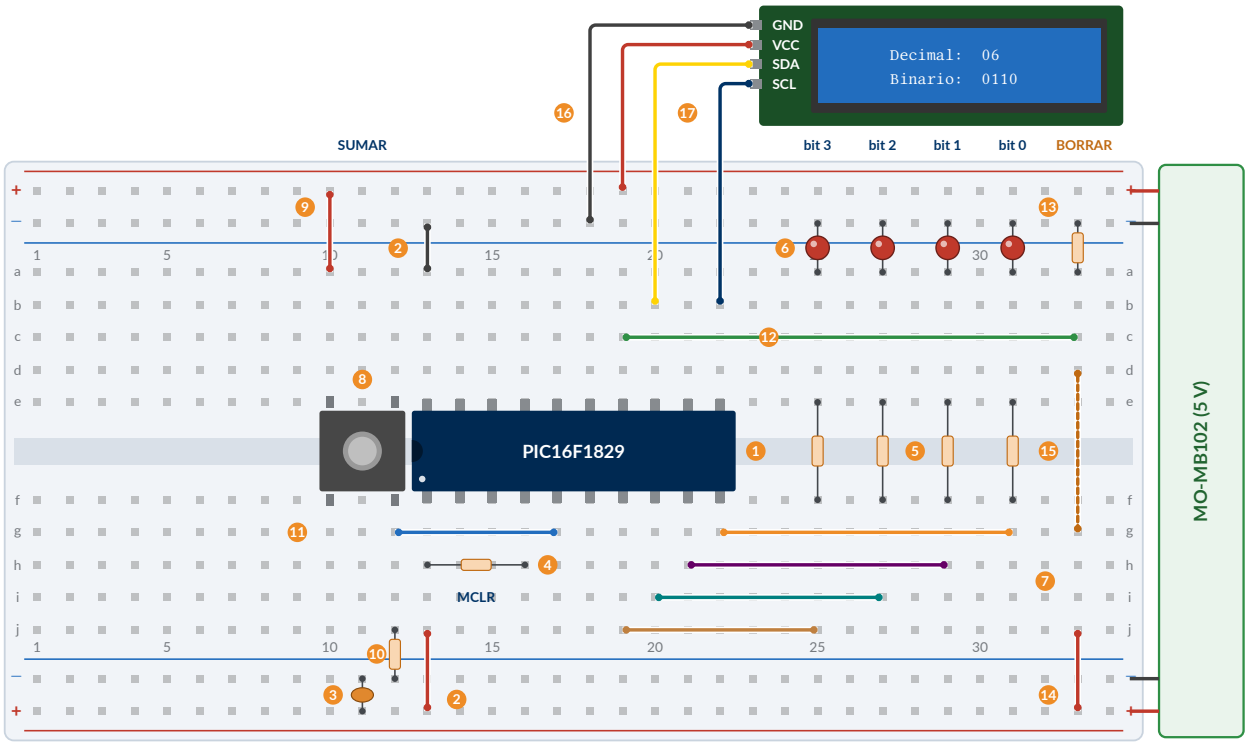


Figura 04.3. El circuito armado. Los números naranjas remiten a la lista de armado. Las letras (a a j) son las filas y los números, las columnas. El PIC tiene la muesca a la izquierda.

Tabla 04.2. Lista de armado

N.º	Qué se coloca y dónde	Parte
1	PIC16F1829, **ya grabado**, en las columnas 13 a 22, muesca a la izquierda	A
2	Alimentación, dos cables: negro de a13 al riel − (VSS, terminal 20); rojo de j13 al riel + (VDD, terminal 1)	A
3	Capacitor de 100 nF entre los dos rieles de abajo, en la columna 11	A
4	Resistencia de 10 kΩ: de h13 a h16 (de VDD al terminal 4, MCLR)	A
5	Cuatro resistencias de 330 Ω cruzando la ranura: f25–e25, f27–e27, f29–e29 y f31–e31	A
6	Cuatro LEDs: pata larga en a25, a27, a29 y a31; pata corta en el riel − de arriba	A
7	Cables: j19 a j25 (bit 3, terminal 7), i20 a i27 (bit 2, terminal 8), h21 a h29 (bit 1, terminal 9) y g22 a g31 (bit 0, terminal 10)	A
8	Botón SUMAR cruzando la ranura, con las patas en e10, e12, f10 y f12	B
9	Cable rojo: de a10 al riel + de arriba	B
10	Resistencia de 10 kΩ: de j12 al riel − de abajo	B
11	Cable: de g12 a g17 (al terminal 5)	B
12	Cable: de c19 a c33 (terminal 14)	B
13	Resistencia de 10 kΩ: de a33 al riel − de arriba	B
14	Cable rojo: de j33 al riel + de abajo	B
15	Puente BORRAR: un cable de d33 a g33. Se pone para borrar y se quita	B
16	LCD: GND al riel − de arriba y VCC al riel + de arriba	C
17	LCD: SDA a b20 (terminal 13) y SCL a b22 (terminal 11)	C

RECOMENDACIÓN

La resistencia de MCLR (elemento 4) aprovecha que la columna 13 de abajo ya está en +5 V por el cable rojo del elemento 2. El botón usa dos patas **en diagonal**: e10 y f12; así funciona aunque se coloque girado.

ADVERTENCIA

Apague el MO-MB102 antes de mover cualquier cable. Coloque el módulo con sus dos puentes en 5 V y verifique con el multímetro cuál riel es el +. Si su protoboard tiene los rieles partidos a la mitad, una las dos mitades con un cable: el circuito llega hasta la columna 33.

Procedimiento

Parte A. Compilar, grabar el PIC y conectar los LEDs — 65 min

Compilar el programa

1. En MPLAB X: `File > New Project`, categoría *Microchip Embedded*, tipo *Application Project(s)* (en versiones anteriores, *Standalone Project*).
2. Dispositivo: **PIC16F1829**. Herramienta: su PICkit o Snap si lo tiene conectado; si usa la tarjeta iCP03, elija *No Tool*. Compilador: **XC8**. Nombre del proyecto: `PF_Contador`.
3. En el árbol del proyecto, clic derecho en *Source Files* y `Add Existing Item`: elija `PF_Contador_LCD_16F1829.c`. También puede crear un archivo nuevo (`New > main.c`) y pegar el programa completo del final de esta guía.
4. Pulse **Clean and Build** (el martillo con la escoba). Abajo debe aparecer **BUILD SUCCESSFUL**. El archivo grabable queda en la carpeta del proyecto, en `dist/default/production/`, con terminación `.hex`.

Grabar el chip

Con la tarjeta iCP03 y su programador:

5. Ponga el selector de la tarjeta en **8-20** y coloque el PIC en el zócalo siguiendo la marca **20-PIN** impresa, con el terminal 1 donde indica la marca. Baje la palanca.
6. En el software del programador elija el **PIC16F1829**, lea el chip (*Read* o *Detect*) para confirmar que lo reconoce, cargue el `.hex` y pulse *Program*. Al terminar, pulse *Verify*.
7. Suba la palanca y saque el PIC.

Con un PICkit o un Snap: en MPLAB X pulse **Make and Program Device**, o abra MPLAB IPE, elija el PIC16F1829 y su herramienta, cargue el `.hex` y pulse *Program*. El PICkit se conecta a los terminales de programación del chip (`MCLR`, `VDD`, `VSS`, `ICSPDAT` en el 19 e `ICSPCLK` en el 18) o a un zócalo adaptador.

Primera prueba

8. Con la alimentación apagada, coloque los elementos 1 a 4 de la lista de armado.
9. Encienda el MO-MB102 y mida con el UT33B+ entre los terminales 1 y 20, y entre el 4 y el 20. Apague.
10. Coloque los LEDs (elementos 5 a 7) y encienda. La luz debe caminar una vez del bit 0 al bit 3. Después los LEDs se apagan: la cuenta está en 0. Apague y encienda el MO-MB102 para repetir la prueba.

Tabla 04.3. Primera prueba

Comprobación	Resultado
El proyecto compiló con *BUILD SUCCESSFUL* (sí / no)	
El programador grabó y verificó sin errores (sí / no)	
Tensión entre los terminales 1 y 20	
Tensión entre los terminales 4 (`MCLR`) y 20	
Orden en que encienden los LEDs al arrancar	
Al terminar la prueba, ¿cuántos LEDs quedan encendidos?	

ADVERTENCIA

Cada vez que cambie el programa hay que compilar y grabar de nuevo: apague el MO-MB102, saque el PIC del protoboard con cuidado (sin doblar los terminales), grábelo en el zócalo y vuelva a colocarlo con la muesca a la izquierda.

Parte B. Las entradas: SUMAR y BORRAR

35 min

1. Apague y coloque los elementos 8 a 15. El puente BORRAR *no* se deja puesto.
2. Encienda. Presione SUMAR una vez por renglón de la tabla y anote los LEDs en el orden bit 3, bit 2, bit 1, bit 0 (encendido = 1).
3. Al final, ponga y quite el puente BORRAR.

Tabla 04.4. El contador en los LEDs

Pulsaciones desde 0	LEDs (bit 3 a bit 0)	En decimal
1		
2		
3		
4		
7		
8		
10		
15		
16		
Después de poner el puente BORRAR		

NOTA

Si la cuenta avanza sola o salta de dos en dos, revise la resistencia de 10 kΩ del elemento 10: sin ella el terminal 5 queda al aire.

Parte C. La pantalla LCD

40 min

1. Apague y conecte la LCD (elementos 16 y 17).
2. Encienda. La pantalla muestra el saludo mientras la luz camina y después las dos líneas `Decimal:` y `Binario:`.
3. Si la luz de fondo enciende pero no se ven letras, gire con un desarmador el potenciómetro azul del módulo (contraste).
4. Lleve la cuenta a cada número de la tabla y anote lo que dice la LCD.

Tabla 04.5. La LCD y los LEDs

Lleve la cuenta a	LCD: Decimal	LCD: Binario	¿Coincide con los LEDs?
5			
9			
12			
14			

Tabla 04.6. El rebote

Prueba	La LCD marca
Ponga el puente BORRAR, quítelo y presione SUMAR 10 veces, lo más rápido que pueda	

Parte D. Retos: cambiar el programa

35 min

Cada reto cambia una o dos líneas. Haga una copia del archivo antes de empezar. Después de cada cambio: compilar, grabar y volver a probar.

1. **Contador de década.** Haga que la cuenta vaya de 0 a 9 y después regrese a 0, como el 74LS90. Pista: busque el número 15 dentro de `while (1)`.
2. **Contar hacia abajo.** Haga que BORRAR *reste* uno en lugar de ir a 0, y que debajo de 0 siga el 15, como el 74LS193.

Si algo no funciona

Síntoma	Revise
Error `xc.h: No such file` o *no compiler* al compilar	XC8 no está instalado o el proyecto no lo tiene elegido: `File > Project Properties > Compiler Toolchain`
Otro error al compilar	Que se haya copiado el programa completo, desde el primer `/*` hasta la última `}`
El programador no reconoce el chip (*Device ID* 0000 o 3FFF)	Orientación del PIC en el zócalo, la marca 20-PIN, el selector en 8-20 y la palanca abajo; que el chip elegido sea el PIC16F1829
Device ID mismatch	En el software se eligió otro chip
Grabó bien, pero en el protoboard no pasa nada	La resistencia de **MCLR** (elemento 4): con el terminal 4 en 0 el PIC se queda reiniciado; la alimentación en los terminales 1 y 20
El PIC se calienta	Alimentación invertida: **VDD** es el terminal 1. Apague de inmediato
Un LED nunca enciende	Su polaridad y que la resistencia cruce la ranura en la misma columna
La cuenta avanza sola	La resistencia de 10 kΩ de j12 al riel −
SUMAR no hace nada	El cable rojo de a10; el cable de g12 a g17
BORRAR no hace nada	El cable de c19 a c33; el cable rojo de j33
Los LEDs muestran otro número	Cables de los terminales 7 a 10 cruzados: el 7 es el bit 3 y el 10 el bit 0
LCD encendida, sin letras	El contraste: potenciómetro azul del módulo
LCD con cuadros en la primera línea	La dirección: cambie `0x4E` por `0x7E` en `DIR_LCD`, compile y grabe otra vez
LCD apagada o el programa se detiene al arrancar	**VCC** y **GND** del módulo; **SDA** (terminal 13) y **SCL** (terminal 11) cruzados. Sin la LCD bien conectada el bus I^2C puede quedarse esperando

Preguntas de cierre

1. ¿Por qué después de 15 la cuenta regresa a 0?

2. ¿Qué diferencia hay entre compilar y grabar?

3. ¿Para qué sirve la resistencia de 10 kΩ del terminal 4 (`MCLR`)?

4. ¿Cuántos cables necesita la LCD y qué lleva cada uno?

El programa completo

</> PF_CONTADOR_LCD_16F1829.C

```
/*
 * Universidad Rural de Guatemala - Electronica Digital FE731
 * Practica final, version PIC16F1829: contador binario con pantalla LCD
 * SUMAR suma 1 y BORRAR regresa a 0. La cuenta, de 0 a 15, se ve en
 * cuatro LEDs y en la LCD (I2C), en decimal y en binario.
 * Compilador: MPLAB XC8, en un proyecto de MPLAB X para el PIC16F1829.
 */
#include <xc.h>
#include <stdint.h>

#pragma config FOSC = INTOSC    // oscilador interno: no lleva cristal
#pragma config WDTE = OFF
#pragma config PWRTE = ON
#pragma config MCLRE = ON       // terminal 4: 10 kohm a +5 V
#pragma config CP = OFF
#pragma config CPD = OFF
#pragma config BOREN = ON
#pragma config CLKOUTEN = OFF
#pragma config IESO = OFF
#pragma config FCMEN = OFF
#pragma config WRT = OFF
#pragma config PLLEN = OFF
#pragma config STVREN = ON
#pragma config BORV = LO
#pragma config LVP = ON         // permite grabarlo tambien con un MPLAB Snap

#define _XTAL_FREQ 4000000      // 4 MHz, lo usa __delay_ms()

// Terminales: los mismos del protoboard de la version PICAXE-20M2
#define SUMAR   PORTCbits.RC5   // terminal 5, boton
#define BORRAR  PORTCbits.RC2   // terminal 14, puente a +5 V
#define LED0    LATBbits.LATB7  // terminal 10, bit 0 (derecha)
#define LED1    LATCbits.LATC7  // terminal 9, bit 1
#define LED2    LATCbits.LATC6  // terminal 8, bit 2
#define LED3    LATCbits.LATC3  // terminal 7, bit 3 (izquierda)

```

```
// Direccion de 8 bits: 0x27 por 2 = 0x4E.
// Si la LCD solo muestra cuadros, use 0x7E (0x3F por 2).
#define DIR_LCD 0x4E

uint8_t cuenta = 0;

// ---- Bus I2C con el modulo MSSP: SDA en el terminal 13, SCL en el 11 ----

void i2cEspera(void) {
    while ((SSP1CON2 & 0x1F) || (SSP1STAT & 0x04)) {
    }
}

void i2cInicio(void) {
    i2cEspera();
    SSP1CON2bits.SEN = 1;
}

void i2cFin(void) {
    i2cEspera();
    SSP1CON2bits.PEN = 1;
}

void i2cEscribir(uint8_t dato) {
    i2cEspera();
    SSP1BUF = dato;
}

// ---- Manejo de la LCD por I2C (modulo PCF8574) ----
// Cada byte del PCF8574 lleva: D7 D6 D5 D4 | luz | EN | RW | RS.
// La LCD recibe cada byte en dos mitades de 4 bits; toma cada mitad
// cuando EN pasa de 1 a 0. modo: 0 = orden para la LCD, 1 = letra.

void enviarMitad(uint8_t mitad, uint8_t modo) {
    i2cInicio();
    i2cEscribir(DIR_LCD);
    i2cEscribir(mitad | 0x0C | modo);   // luz encendida, EN = 1
    i2cEscribir(mitad | 0x08 | modo);   // luz encendida, EN = 0
    i2cFin();
}

void enviarByte(uint8_t dato, uint8_t modo) {
    enviarMitad(dato & 0xF0, modo);
    enviarMitad((uint8_t)(dato << 4), modo);
}

void orden(uint8_t o) {
    enviarByte(o, 0);
    if (o <= 0x02) {
        __delay_ms(2);                  // borrar la pantalla tarda mas
    }
}

void escribir(const char *texto) {
```

```
91      while (*texto) {
92          enviarByte((uint8_t)*texto, 1);
93          texto++;
94      }
95  }
96
97  void iniciarLCD(void) {
98      __delay_ms(50);
99      enviarMitad(0x30, 0); __delay_ms(5);
100     enviarMitad(0x30, 0); __delay_ms(1);
101     enviarMitad(0x30, 0); __delay_ms(1);
102     enviarMitad(0x20, 0); __delay_ms(1);  // modo de 4 bits
103     orden(0x28);                          // 2 lineas
104     orden(0x0C);                          // pantalla encendida, sin cursor
105     orden(0x06);                          // el cursor avanza a la derecha
106     orden(0x01);                          // borrar
107 }
108
109 // ---- El contador ----
110
111 void ponerLEDs(void) {
112     LED0 = cuenta & 1;
113     LED1 = (cuenta >> 1) & 1;
114     LED2 = (cuenta >> 2) & 1;
115     LED3 = (cuenta >> 3) & 1;
116 }
117
118 void mostrar(void) {
119     ponerLEDs();
120     orden(0x89);                          // linea 1, columna 9
121     enviarByte('0' + cuenta / 10, 1);     // decenas
122     enviarByte('0' + cuenta % 10, 1);     // unidades
123     orden(0xC9);                          // linea 2, columna 9
124     for (int8_t b = 3; b >= 0; b--) {
125         enviarByte('0' + ((cuenta >> b) & 1), 1);
126     }
127 }
128
129 void main(void) {
130     OSCCON = 0x68;                        // oscilador interno a 4 MHz
131
132     ANSELA = 0;                           // todos los terminales digitales
133     ANSELB = 0;
134     ANSELC = 0;
135     LATA = 0;
136     LATB = 0;
137     LATC = 0;
138     TRISA = 0b00001000;                   // RA3 (terminal 4) es MCLR
139     TRISB = 0b01010000;                   // RB4 y RB6: SDA y SCL del I2C
140     TRISC = 0b00100100;                   // RC5 SUMAR y RC2 BORRAR: entradas
141
142     SSP1ADD = 9;                          // 4 MHz / (4 x 10) = 100 kHz
143     SSP1STAT = 0x80;
144     SSP1CON1 = 0x28;                      // MSSP como maestro I2C
```

```
145
146     iniciarLCD();
147     orden(0x80);                        // linea 1
148     escribir("URURAL  FE731");
149     orden(0xC0);                        // linea 2
150     escribir("Contador binario");
151
152     // Prueba de los LEDs: la luz camina del bit 0 al bit 3
153     for (uint8_t k = 0; k < 4; k++) {
154         cuenta = (uint8_t)(1 << k);
155         ponerLEDs();
156         __delay_ms(400);
157     }
158
159     cuenta = 0;
160     orden(0x01);                        // borrar la pantalla
161     orden(0x80);
162     escribir("Decimal:");
163     orden(0xC0);
164     escribir("Binario:");
165     mostrar();
166
167     while (1) {
168         if (SUMAR == 1) {
169             cuenta = cuenta + 1;
170             if (cuenta > 15) {
171                 cuenta = 0;
172             }
173             mostrar();
174             __delay_ms(20);             // deja pasar el rebote
175             while (SUMAR == 1) {        // espera a que se suelte
176             }
177             __delay_ms(20);
178         }
179
180         if (BORRAR == 1) {
181             cuenta = 0;
182             mostrar();
183             __delay_ms(20);
184             while (BORRAR == 1) {
185             }
186             __delay_ms(20);
187         }
188     }
189 }
```

Parte del programa	Qué hace
`#pragma config`	Configura el chip al grabarlo: oscilador interno, `MCLR` activo, sin protección de código
`#define`	Da nombre a los terminales: `SUMAR`, `BORRAR`, `LED0` a `LED3` y la dirección de la LCD
`i2cInicio` a `i2cEscribir`	Manejan el bus I^2C con el módulo MSSP del PIC
`enviarMitad` a `iniciarLCD`	Mandan cada letra a la LCD en dos mitades de 4 bits. No hace falta modificarlas
`ponerLEDs` y `mostrar`	Encienden los LEDs con los bits de la cuenta y la escriben en la LCD
`main`, antes del `while`	Prepara los terminales y el I^2C, saluda y hace caminar la luz
`while (1)`	Revisa SUMAR y BORRAR una y otra vez; espera el rebote y que se suelte el botón

EVALUACIÓN EN LÍNEA (SOCRATIVE)

Al terminar la práctica se aplicará una evaluación breve en **Socrative**. Las preguntas combinan la teoría de la clase magistral en línea con lo que se acaba de hacer en el laboratorio.
El instructor indicará el código de la sala. Anótelo aquí, junto con cualquier indicación del día.

Código de sala / enlace

Espacio para la evaluación

Reservado para Socrative: el instructor colocará aquí el acceso o las indicaciones