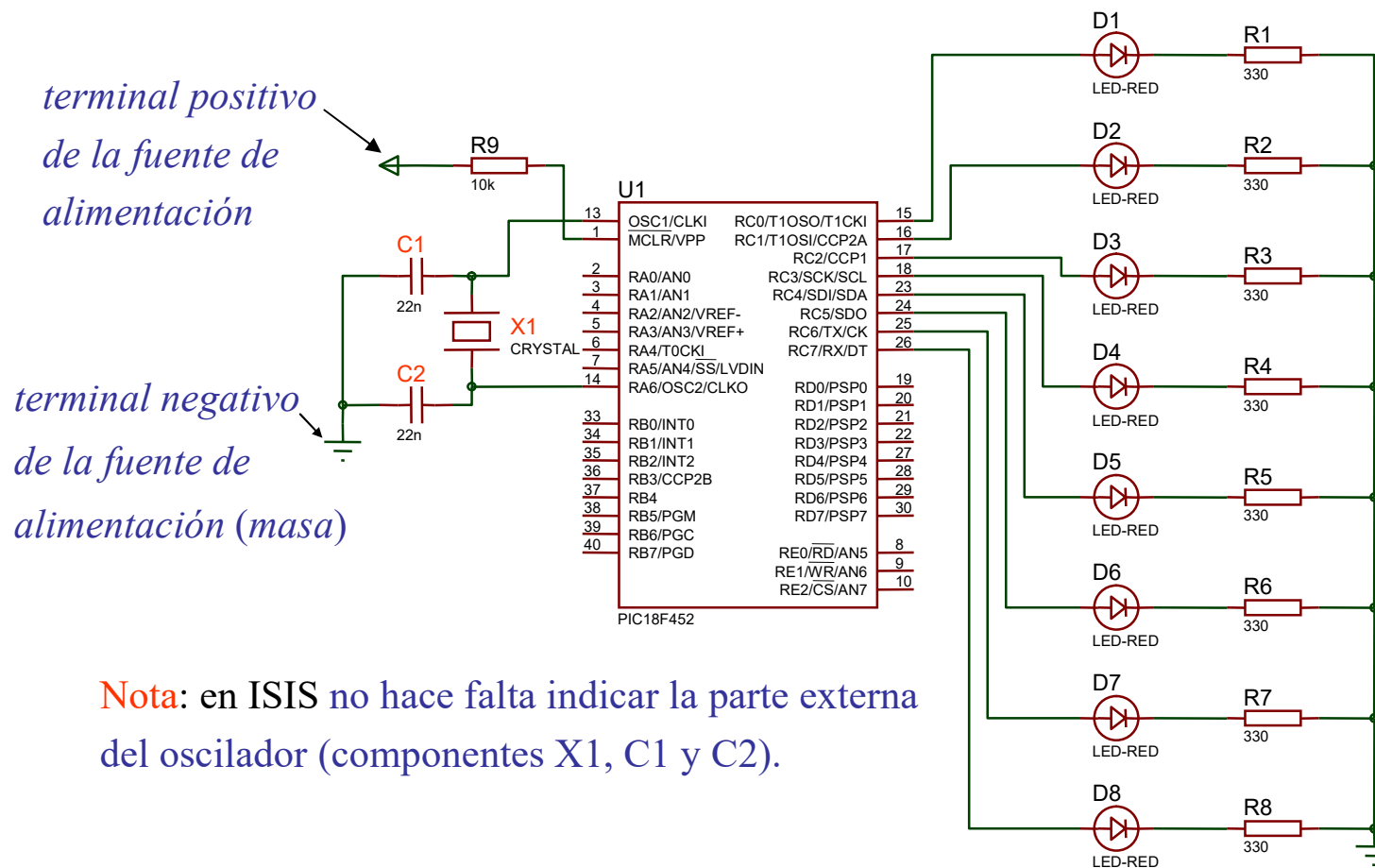


Práctica 1a: Abre el proyecto que está guardado en el escritorio de *Windows*, en la carpeta denominada *Primer programa*. Lee todo lo que se indica (comentarios incluidos) e intenta entender que hará el μC cuando ejecute dicho código. A continuación realiza la práctica indicada en la siguiente página.

Normas:

- Cada proyecto se guarda en una carpeta distinta
- El archivo con la simulación en ISIS debes guardarlo en la misma carpeta en la que guardas los archivos del correspondiente proyecto en *C*.
- Una vez corregida una práctica puedes copiar en un pendrive tanto el proyecto con el código en *C* como el archivo con la simulación. Pero a continuación debes borrar todos los archivos del PC relativos a dicha práctica.

Práctica 1b: Hay que escribir el programa a ejecutar por el microcontrolador indicado en el esquema, de modo que se enciendan los 8 *leds* de forma consecutiva. El tiempo de encendido de cada *led* debe ser igual a 0.2 segundos y el tiempo que transcurre desde que se apaga un *led* hasta que se encienda el siguiente debe ser igual a 0.1 segundos. **Componentes ISIS:** PIC18F452, Res, Led-blue, Crystal, Cap.



Práctica 1c: Hay que escribir el programa a ejecutar por el microcontrolador indicado en el esquema de la página anterior, de modo que se enciendan los 8 *leds* por parejas de forma consecutiva. La secuencia de encendido irá desde la pareja de leds D1-D8 hasta la pareja de leds D4-D5, primero en un sentido y luego en el otro, de forma consecutiva. El tiempo de encendido de cada pareja de *leds* debe ser igual a 0.4 segundos y el tiempo que transcurre desde que se apaga una pareja de *leds* hasta que se encienda la siguiente pareja de leds debe ser igual a 0.2 segundos.

Secuencia de encendido:

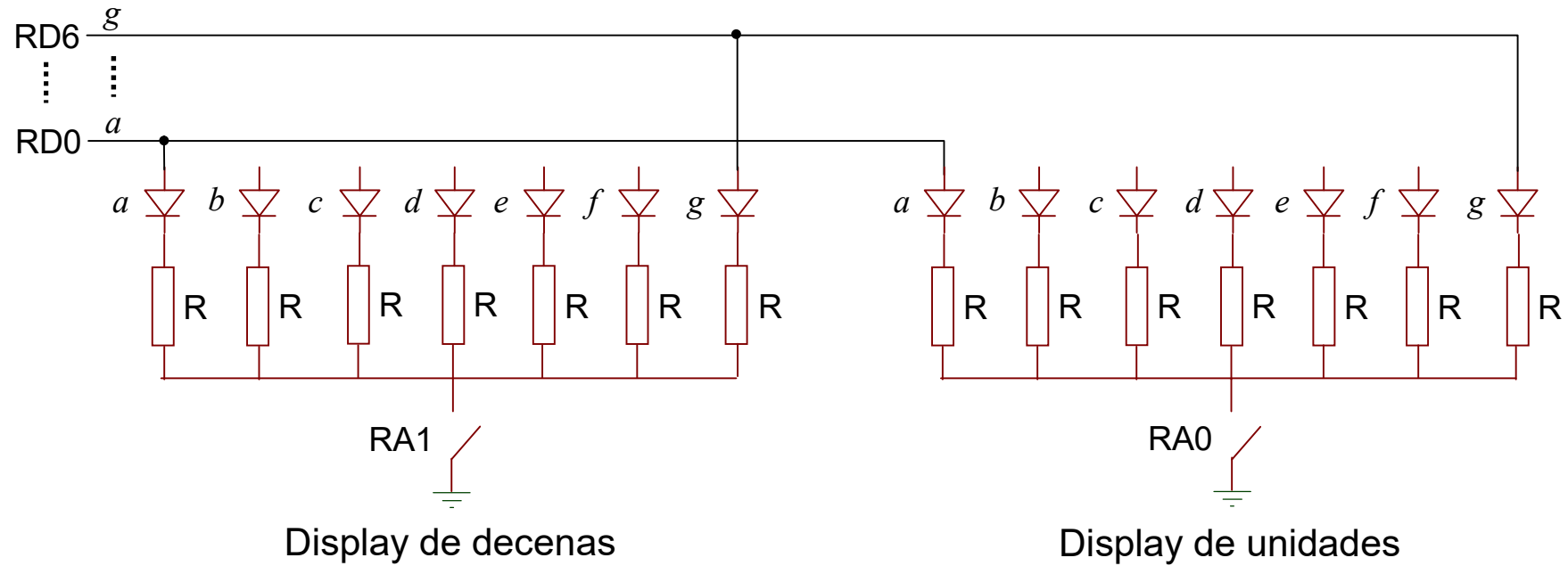
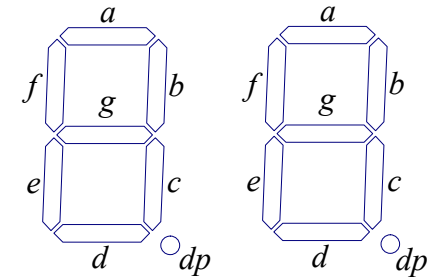
D1&D8 → D2&D7 → D3&D6 → D4&D5 → D3&D6 → D2&D7 → D1&D8 → D2&D7 ····

Práctica 2: Hay que diseñar un contador de módulo 60. El contenido del contador se visualizará en un doble *display* de 7 segmentos de cátodo común. Las señales *a*, *b*, *c*, *d*, *e*, *f* y *g* (ver página siguiente) son comunes a ambos *displays*, por lo que se debe activar alternativamente el *display* de unidades y el de decenas, de modo que parezca que siempre están activos. El contador debe incrementar su contenido cada segundo (periodo = 1seg).

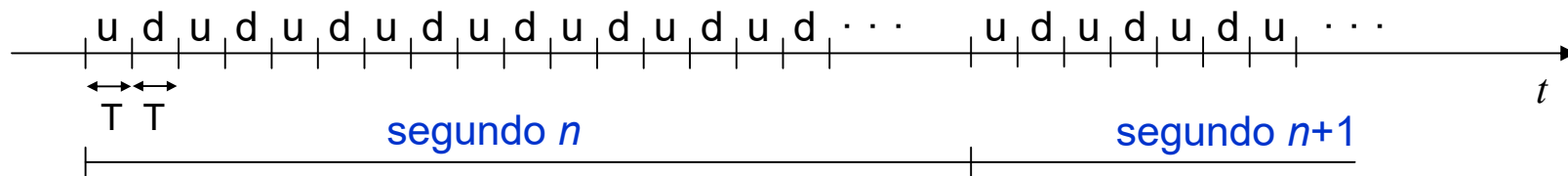
Componentes ISIS: PIC18F452, 7SEG-MPX2-CC-BLUE, RES, RX8, BC846B ZETEX.

Nota: las televisiones antiguas mostraban 25 imágenes (fotografías) por segundo. En este caso, se recomienda mostrar los valores de las *unidades* y de las *decenas* 50 veces por segundo. Es decir, en un segundo se muestran 25 veces las *unidades* y 25 veces las *decenas*.

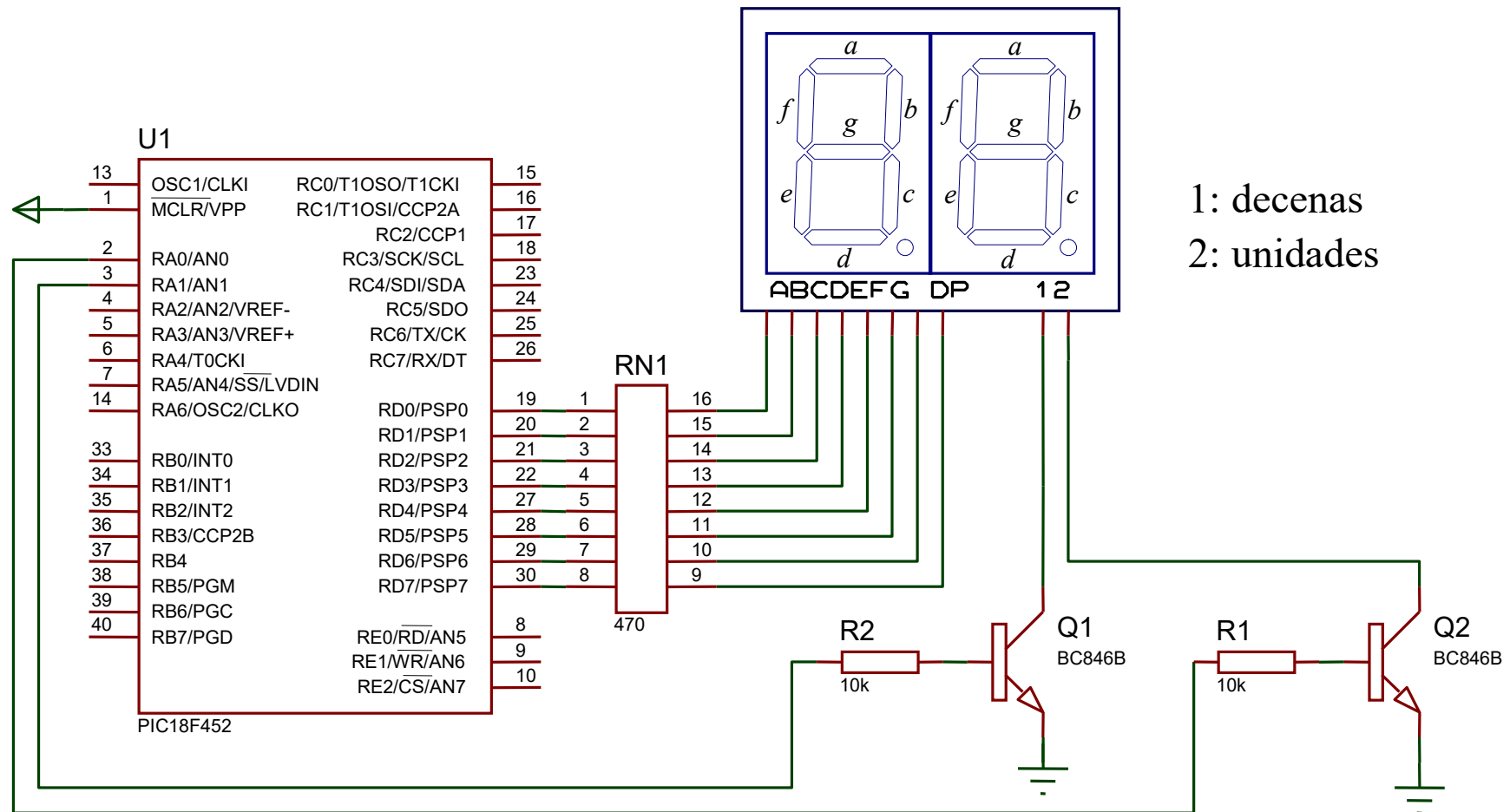
Problema: los terminales dp , g , f , e , d , c , b y a son comunes a ambos displays.



Solución:

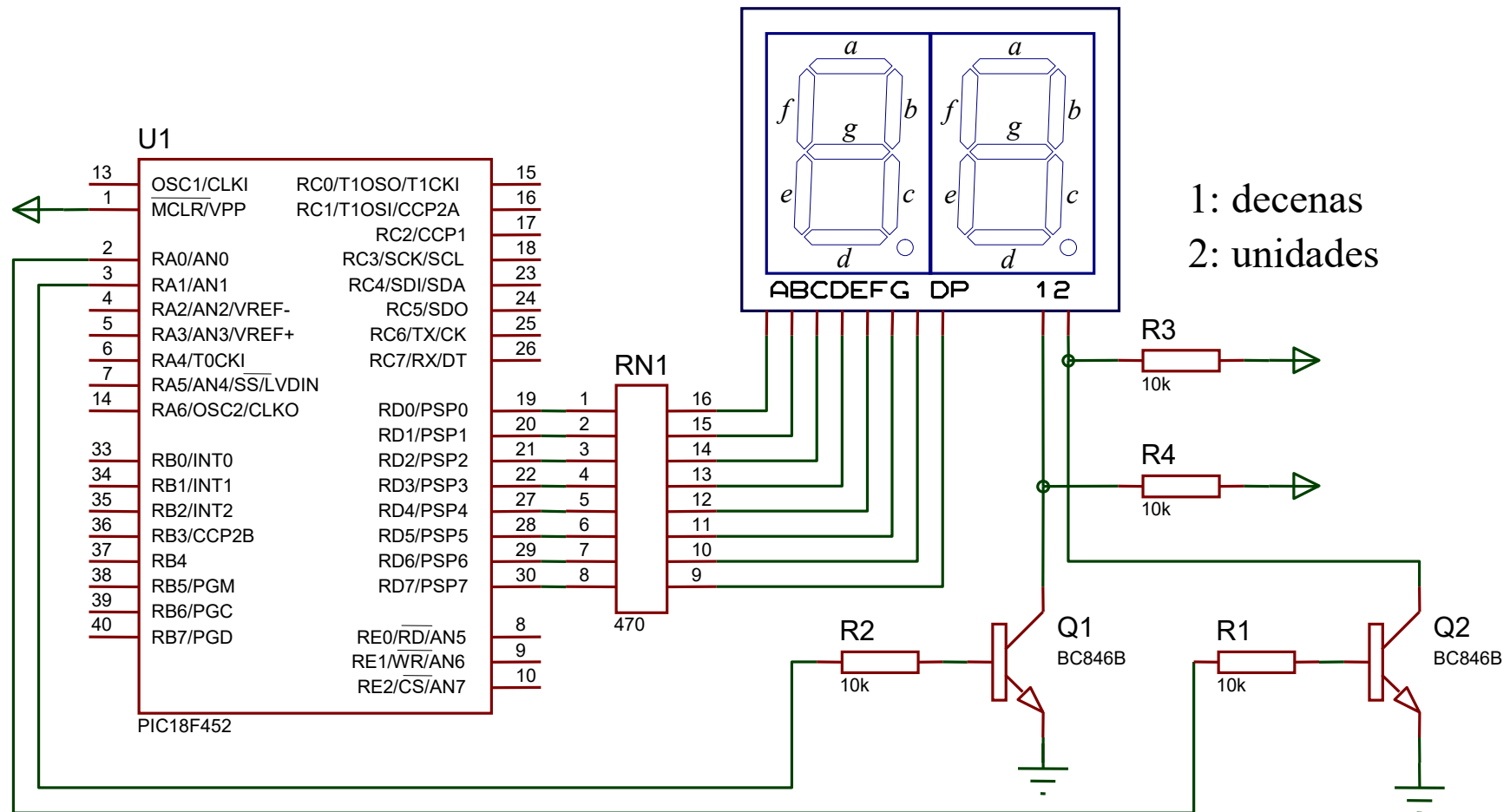


(Circuito real... no simulable en ISIS)



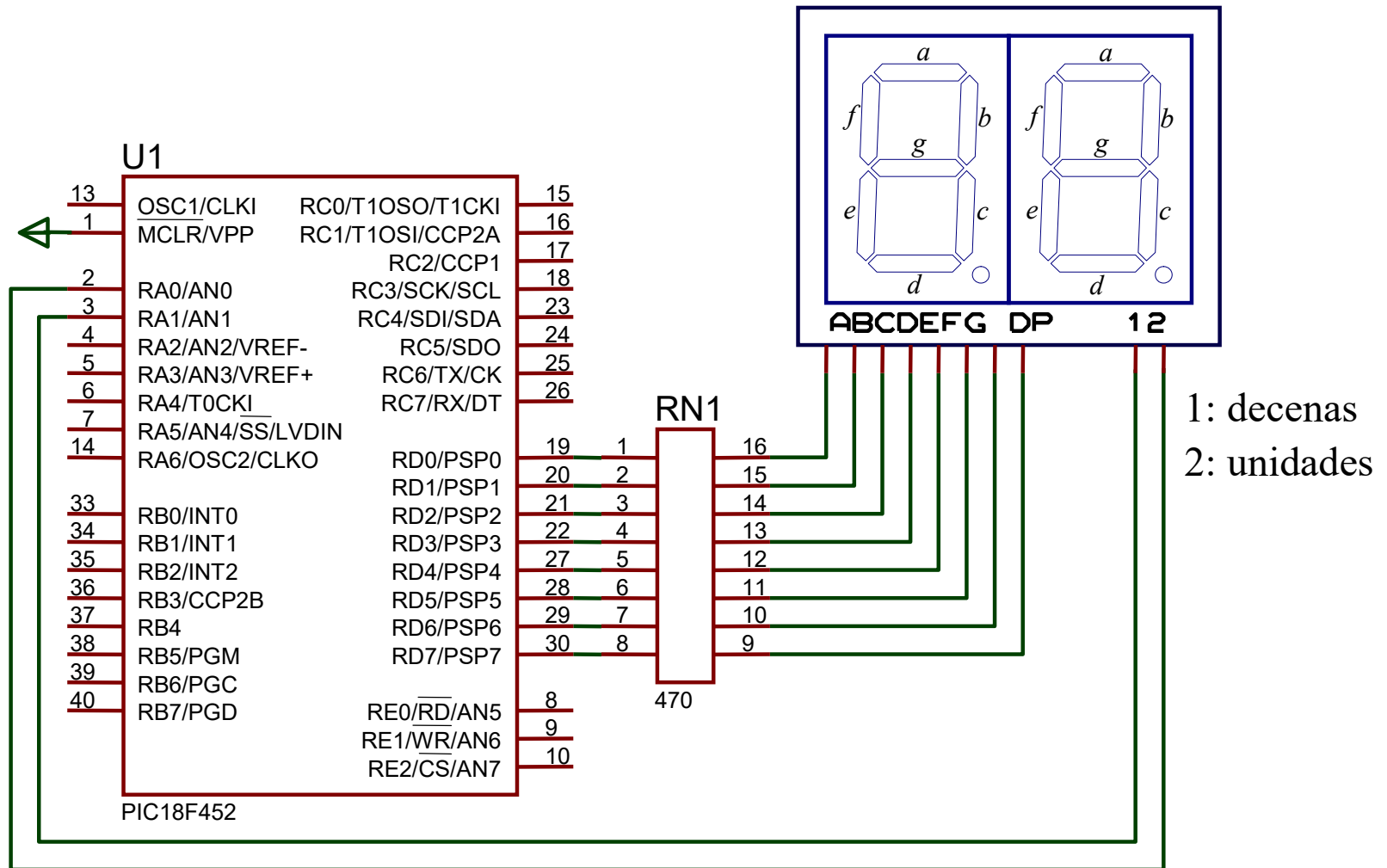
Léeme: con un 1 en RA0 se enciende el display de unidades y con un 0 se apaga
con un 1 en RA1 se enciende el display de decenas y con un 0 se apaga

(Circuito a simular en ISIS **con** los transistores)



Léeme: con un 1 en RA0 se enciende el display de unidades y con un 0 se apaga
con un 1 en RA1 se enciende el display de decenas y con un 0 se apaga

(Circuito a simular en ISIS **sin** los transistores)



Léeme: con un 0 en RA0 se enciende el display de unidades y con un 1 se apaga
con un 0 en RA1 se enciende el display de decenas y con un 1 se apaga

Práctica 3 a): El diodo emisor de luz (led) conectado a la patilla RB1 debe cambiar de estado (encendido/apagado) cada vez que se presiona el pulsador conectado al terminal RB0. En este apartado, se trata de resolver este problema haciendo que el microcontrolador observe periódicamente si el pulsador está presionado o no (técnica de polling \equiv observación periódica). **Componentes ISIS:** PIC18F452, RES, LED-BLUE y SW-SPST-MOM

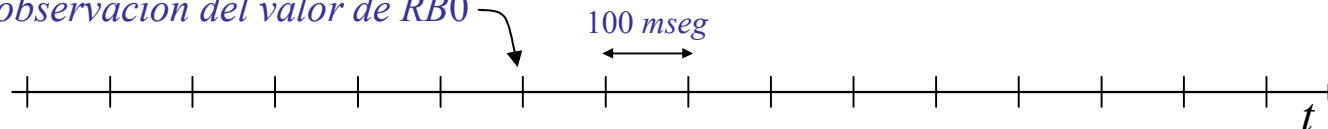
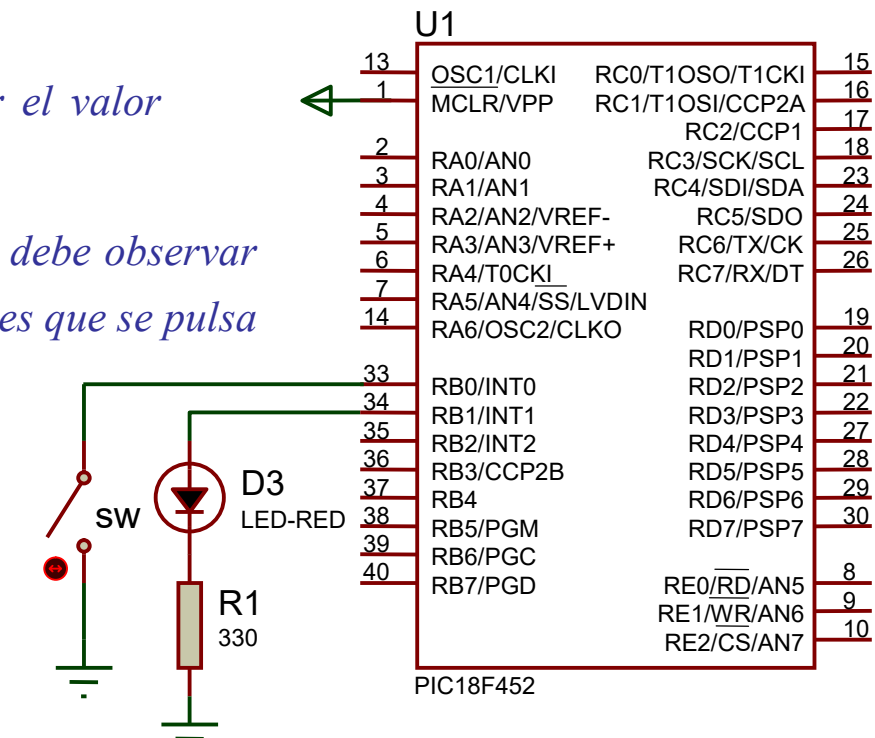
Hay que habilitar la resistencia de pull-up interna del terminal RB0 (RBPU_bit = 0)

El microcontrolador sólo debe comprobar el valor de RB0 una vez cada 100 mseg.

Preguntas: ¿Cuántas veces crees que el μC debe observar el valor de RB0 para que sepa todas las veces que se pulsa el botón?

¿Cuántas veces puede una persona pulsar un botón en 1 segundo?... ¿tantas?... ¿Cuál es el tiempo mínimo que puede mantener pulsado un botón?

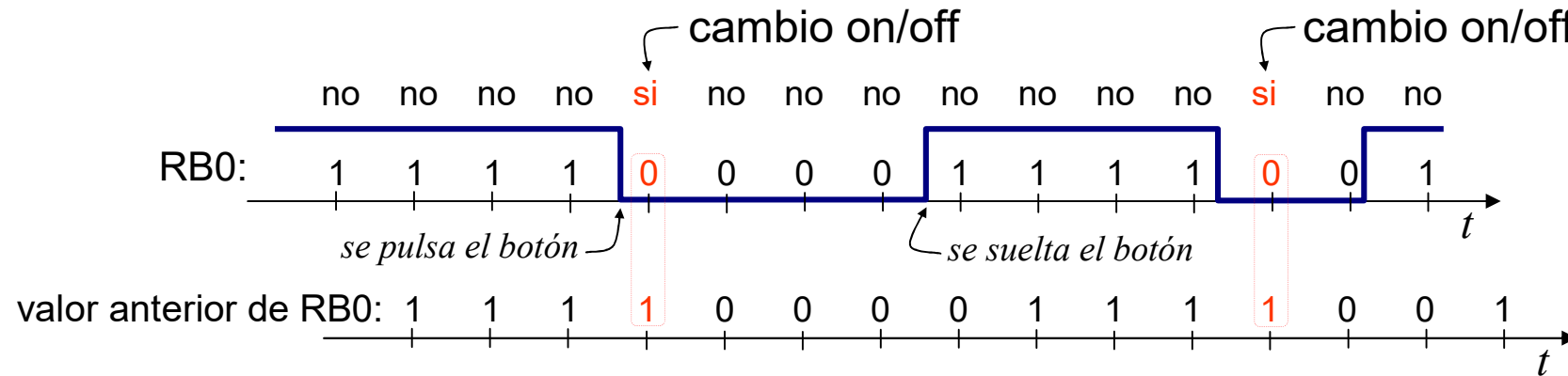
instante de observación del valor de RB0



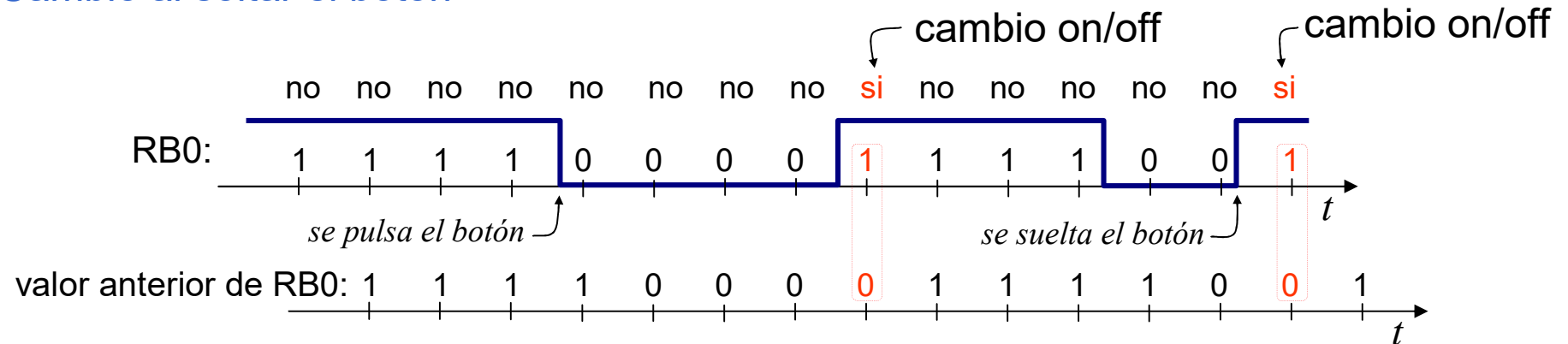
Cambio al pulsar el botón

RB0 = 0: botón pulsado

RB0 = 1: botón no pulsado



Cambio al soltar el botón

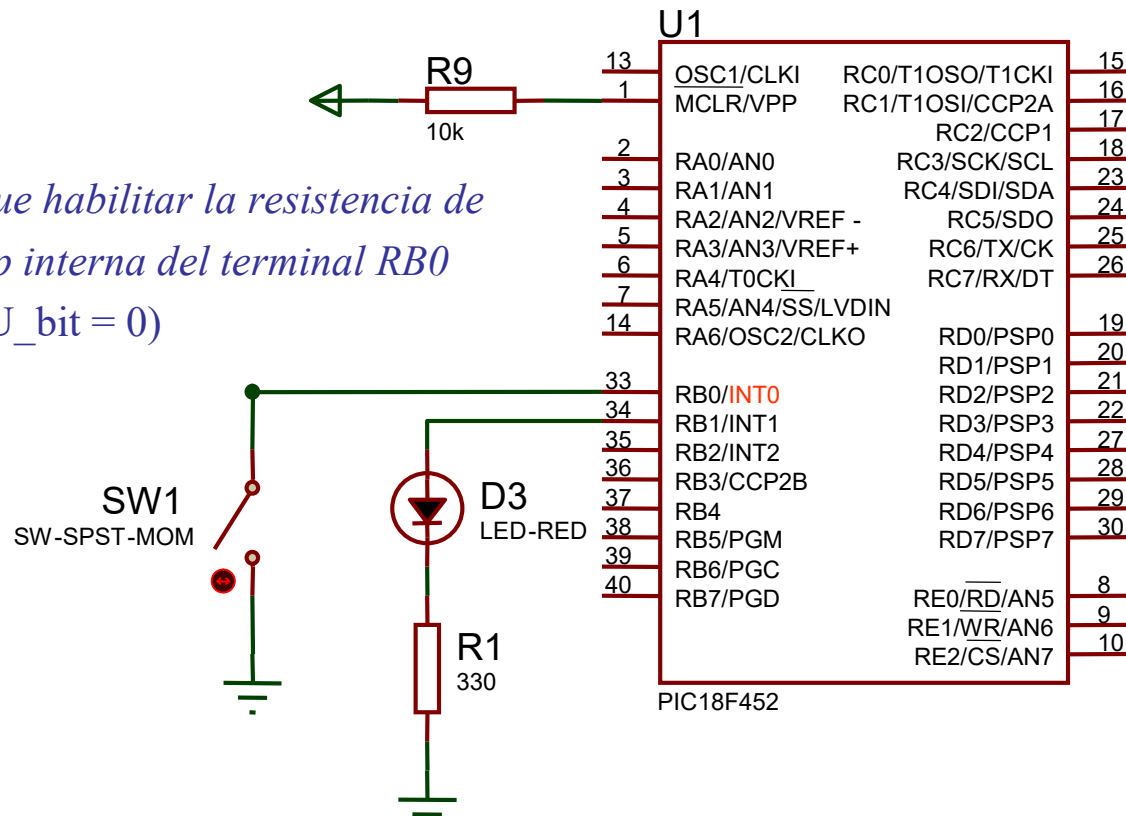


Nota: $\sim x$ [proporciona el $cal(x)$, (ALT+126)] $!x$ [proporciona el negado de x]

Ejemplos: $\sim 0x12 = 0xED$ $\sim 1 = 0$ $!0x12 = 0$ $!0 = 1$ $!1 = 0$ 10

Práctica 3 b): Hay que hacer que el diodo led conectado al terminal RB1 cambie de estado (encendido/apagado) cada vez que se presiona el pulsador conectado al terminal RB0. La diferencia con el apartado anterior reside en que ahora hay que utilizar la *interrupción* (INT0) para detectar los cambios de estado del pulsador. Componentes ISIS: PIC18F452, RES, LED-RED y SW-SPST-MOM.

Hay que habilitar la resistencia de pull-up interna del terminal RB0 (RBPU_bit = 0)



Práctica 3 c): Se trata de diseñar el circuito que indica el ‘*turno*’ o la ‘*vez*’ en los centros de salud, centros comerciales (frutería, carnicería, etc.). La idea es que cada vez que se presione un pulsador, el número decimal representado en un doble display de 7 segmentos incremente su contenido en 1 unidad. Tienes que proponer un circuito a partir de los circuitos de las prácticas anteriores.

Nota 1: Se trata de implementar un contador de módulo 100 del mismo tipo que los estudiados en SD (modo de contaje ascendente), con la diferencia de que en éste caso hay que ver/representar el contenido del contador en base 10, en un doble display de 7 segmentos.

Nota 2: Lo primero que hay que decidir es si la CPU del microcontrolador puede observar el estado del pulsador y controlar, al mismo tiempo, el doble display de 7 segmentos. No se puede utilizar la técnica de *polling* (en lo que queda de curso).

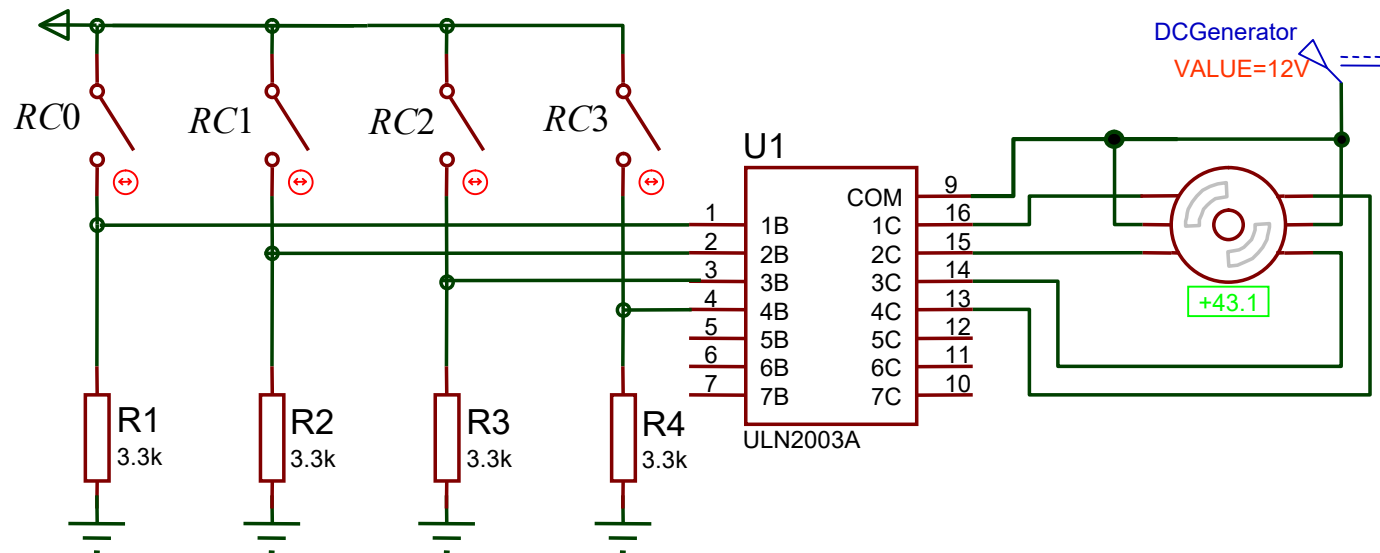
Práctica 4 a): este apartado consiste en estudiar el funcionamiento de un *motor paso a paso* (*Stepper motor*) de tipo *unipolar* (puedes utilizar cualquiera de los circuitos indicados en las siguientes páginas).

i) Con el motor configurado con ángulos de paso de 90° , determina las direcciones en las que se encuentran las 4 bobinas del estator así como la secuencia de excitación para que el rotor realice un giro a derechas de 360° . Indica las direcciones de las bobinas sobre un círculo dibujado en un papel (indica los ángulos correspondientes y el terminal del circuito ULN2003A que hay que poner a 1 para que el rotor se oriente en cada dirección).

ii) Con el motor configurado con ángulos de paso de 45° , determina las $360^\circ/45^\circ = 8$ direcciones en las que se puede orientar el rotor del motor al excitar las bobinas individualmente de forma consecutiva. Indica sobre un círculo dibujado en un papel dichas direcciones y el ángulo (en grados) que corresponde a cada una de las 8 direcciones así como el número del terminal del circuito ULN2003A que hay que poner a 1 para que el rotor se oriente en cada dirección. Ten en cuenta que será el microcontrolador el que tenga que poner a 1 los terminales del ULN2003A para que el rotor gire de acuerdo con las indicaciones del apartado 4 b)

Componentes ISIS: Motor-stepper, ULN2003A, RES, SW-SPST-MOM, DC generator.

Nota: DC generator se obtiene en la columna de la izquierda, en *Generator mode*. Hay que hacer que genere una tensión continua de 12V



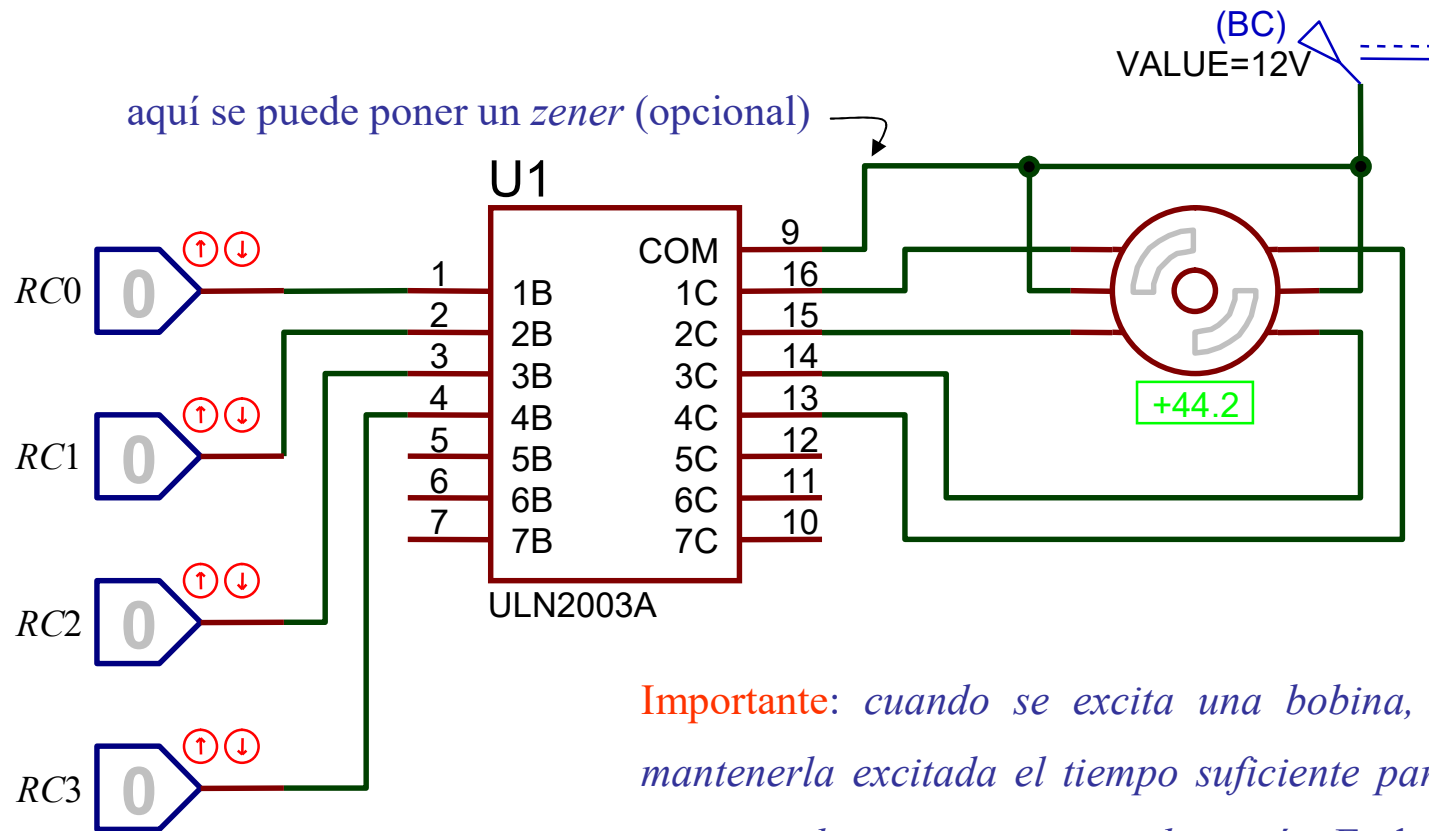
Importante: cuando se excita una bobina, hay que mantenerla excitada el tiempo suficiente para que el rotor del motor pueda orientarse en su dirección. En la práctica, dicho tiempo se determina de forma empírica.

Nota: en la siguiente página se indica un circuito equivalente al anterior para ISIS, que se dibuja en menos tiempo.

A continuación se indica un circuito equivalente al de la página anterior.

Componentes ISIS: Motor-stepper, ULN2003A, Logicstate, DC generator.

Nota: DC generator se obtiene en la columna de la izquierda, en *Generator mode*. Hay que hacer que genere una tensión continua de 12V



Importante: cuando se excita una bobina, hay que mantenerla excitada el tiempo suficiente para que el rotor pueda orientarse en su dirección. En la práctica, dicho tiempo se determina de forma empírica.

Práctica 4 b): Hay que controlar el funcionamiento de un motor paso a paso (*Stepper motor*) de tipo *unipolar*, configurado con pasos de 45° , utilizando un microcontrolador PIC18F452 (ver página siguiente). Componentes ISIS: PIC18F452, Motor-stepper, ULN2003A, RES, SW-SPST-MOM, DC generator.

El control del motor que hay que realizar consiste en lo siguiente:

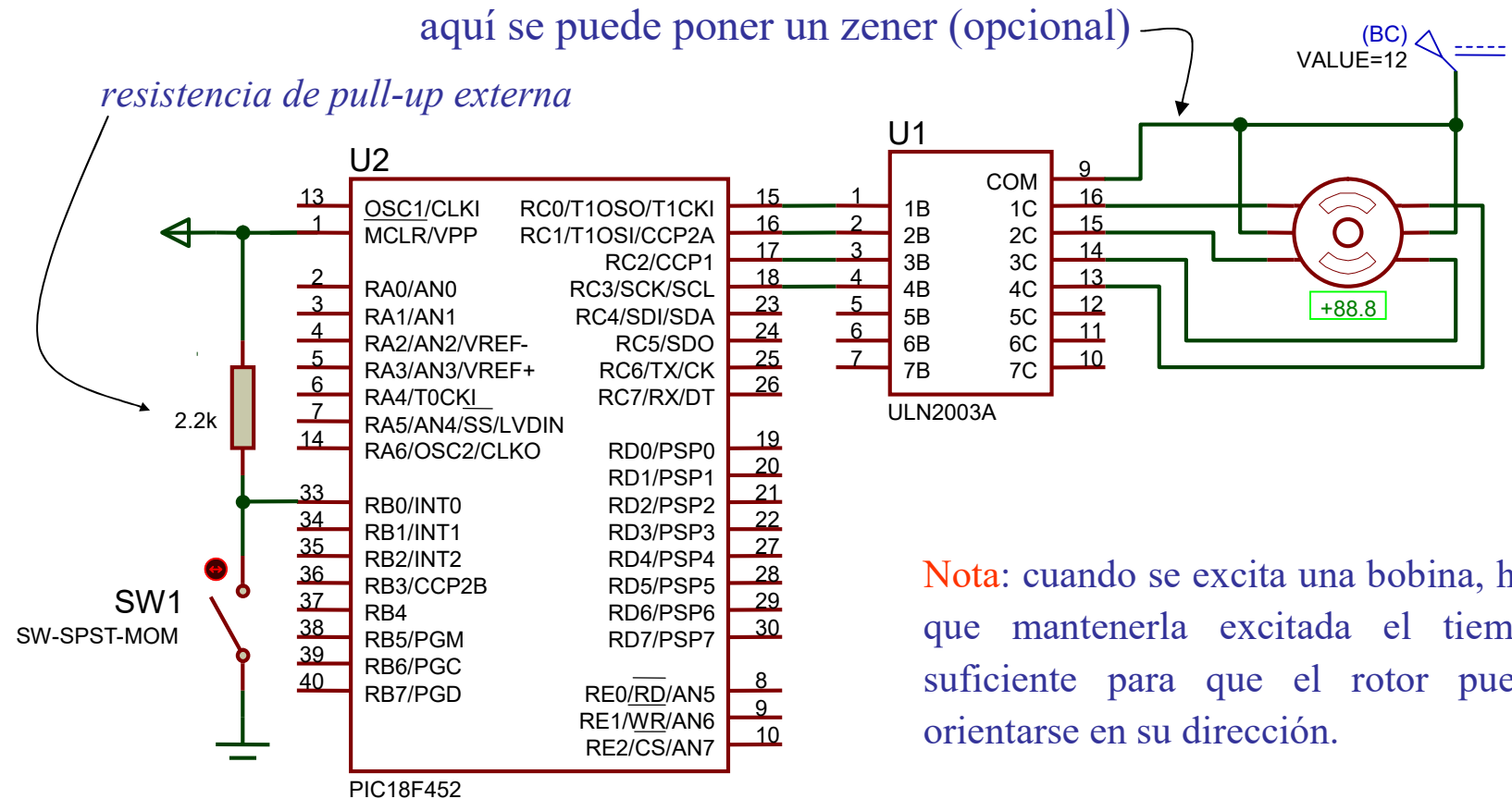
1º: Cuando se inicia ISIS, el eje del motor está siempre en $+0^\circ$ (ver punto amarillo). Hay que hacer que el rotor se sitúe en -90° (posición inicial)

2º: Si se pulsa el botón SW1, el motor debe realizar un giro a izquierdas de 135° y detenerse.

3º: Una vez que el eje del motor ha realizado el giro anterior de 135° , si se pulsa otra vez el botón SW1, el motor debe realizar un giro a derechas de 360° y detenerse. A partir de dicho momento, siempre que se vuelva a pulsar el botón SW1, el motor deberá realizar un giro a derechas de 360° .

Nota: cuando se excita una bobina, hay que mantenerla excitada el tiempo suficiente para que el rotor pueda orientarse en su dirección. Las bobinas se excitan siempre de forma consecutiva

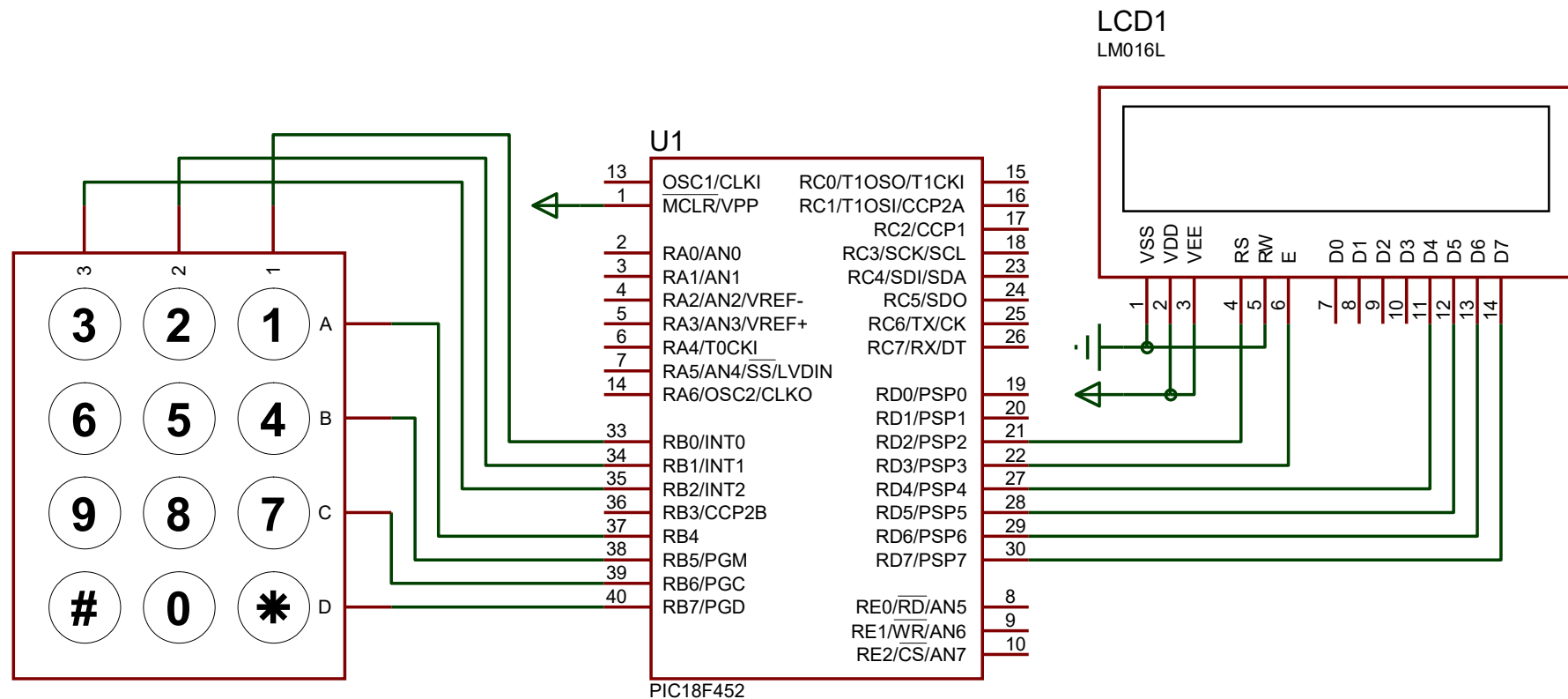
Práctica 4 b) (continuación)



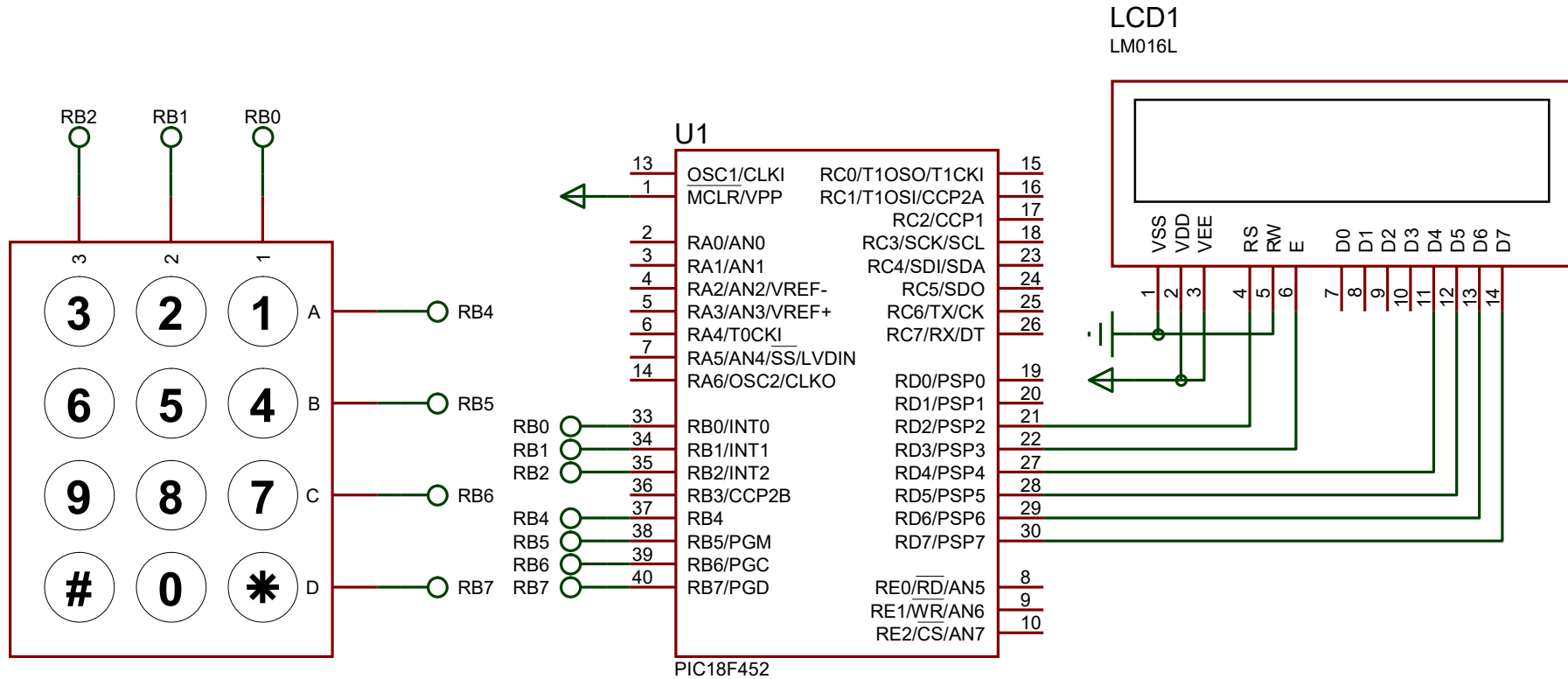
Práctica 5 a): Introducir datos por medio de un teclado y visualizarlos en un LCD.

Componentes ISIS: PIC18F452, LM016L, KEYPAD-PHONE

Nota: la función Tecla12INT.h utiliza interrupciones para detectar cuándo y qué tecla se ha pulsado (`#include "Tecla12INT.h"`).



Un esquema 'equivalente' al anterior para Isis.....



Datos para utilizar la función **Tecla12INT**:

_ Fuera de cualquier función hay que poner:

include "Tecla12INT.h" (hay que poner una copia de este archivo en la carpeta del proyecto)

_ En main() hay que poner lo siguiente (además de otras cosas):

TRISB = 0xF0; // el nibble alto son entradas y el nibble bajo son salidas

PORTB = 0;

_ Configuración interrupción RB4-RB7

INTCON2.RBPU = 0; // se habilitan las resistencias de *pullup* del puerto B

x = PORTB; //para poder borrar el RBIF

INTCON.RBIF=0;

INTCON.RBIE=1;

_ En la rutina de servicio de interrupciones hay que poner:

void interrupt() // se ha pulsado una tecla

{

key = tecla(); // en la variable *key* se guarda el valor ASCII de la tecla pulsada

x = PORTB; // para poder borrar el bit RBIF (define x global)

INTCON.RBIF=0; // se borra el bit RBIF después de llamar a la función **tecla()**

}

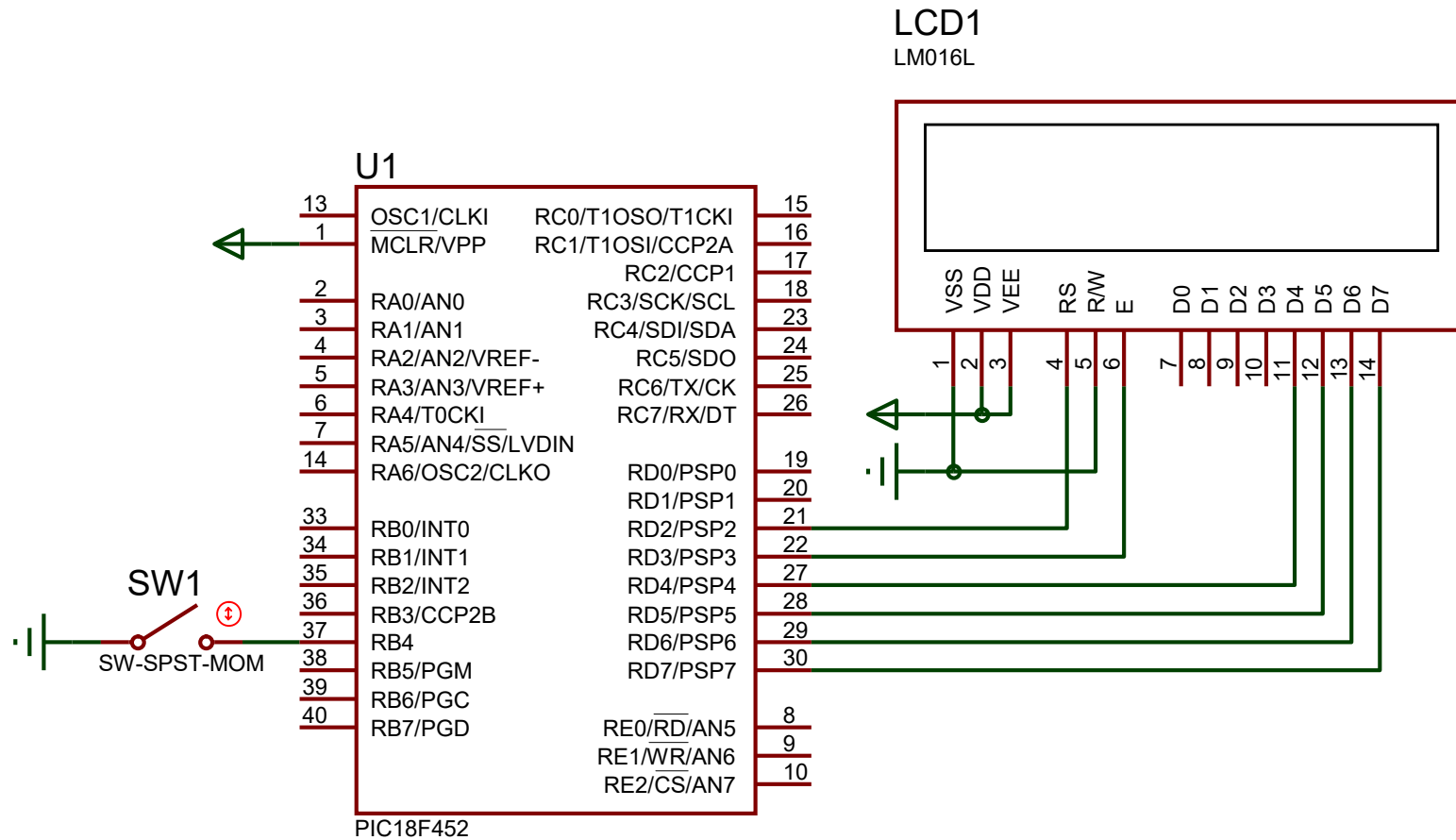
Práctica 5 b): *El objetivo de este ejercicio es comprender y practicar las interrupciones por cambio de nivel de los terminales RB4-RB7.* En el circuito de la página siguiente, el *Lcd* debe mostrar en todo momento el número de veces que se ha pulsado el botón SW1 desde que se ha puesto en funcionamiento el circuito. El sistema debe comenzar a contar en 0 (carácter 48 en ASCII) y una vez que se llegue a 99, cuando se presione otra vez el botón, el sistema debe pasar a 0 (se trata de implementar un contador de módulo 100).

Recordatorio: Para poder borrar el *flag* que indica que se ha producido una interrupción por cambio de nivel hay que LEER el puerto B (al menos hay que leer uno de los terminales del puerto B). Las simulaciones en ISIS sólo funcionan correctamente si la variable que guarda el valor del puerto es global.

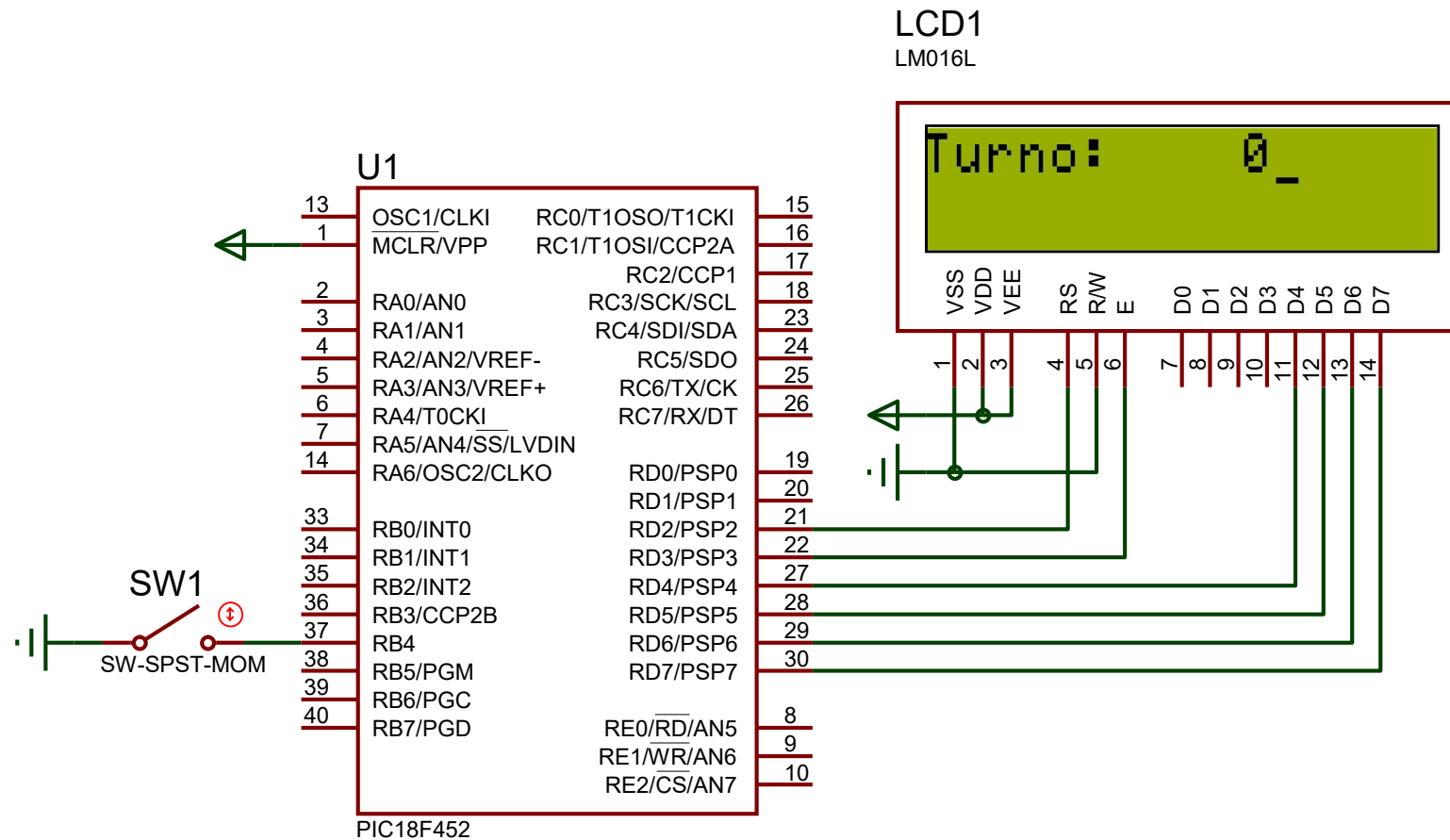
Nota: No olvides que se produce una interrupción al cerrar el interruptor y otra al abrirlo.

Componentes ISIS: PIC18F452, SW-SPST-MOM, LM016L, POWER, GND, RES.

Práctica 5b

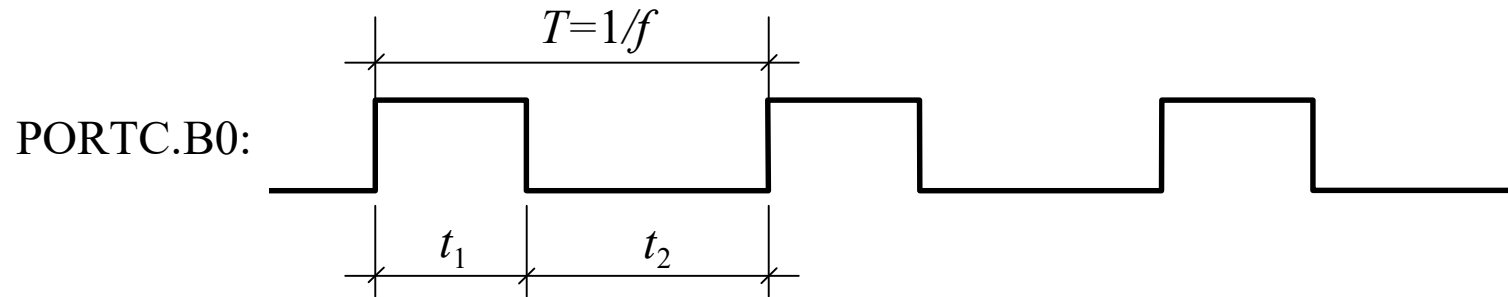


Práctica 5 c): En este apartado se trata de modificar el código escrito para el apartado anterior de modo que cuando se ponga en funcionamiento el circuito, en la pantalla aparezca lo que se indica a continuación (sin que haya que pulsar el botón SW1)



Componente ISIS: PIC18F452, Instrument: OSCILLOSCOPE.





$T = t_1 + t_2 = \text{constante} \equiv$ periodo de la señal a generar

$t_1 = DT$ siendo D el **ciclo de trabajo** (*duty cycle*) $D \in [0,1]$

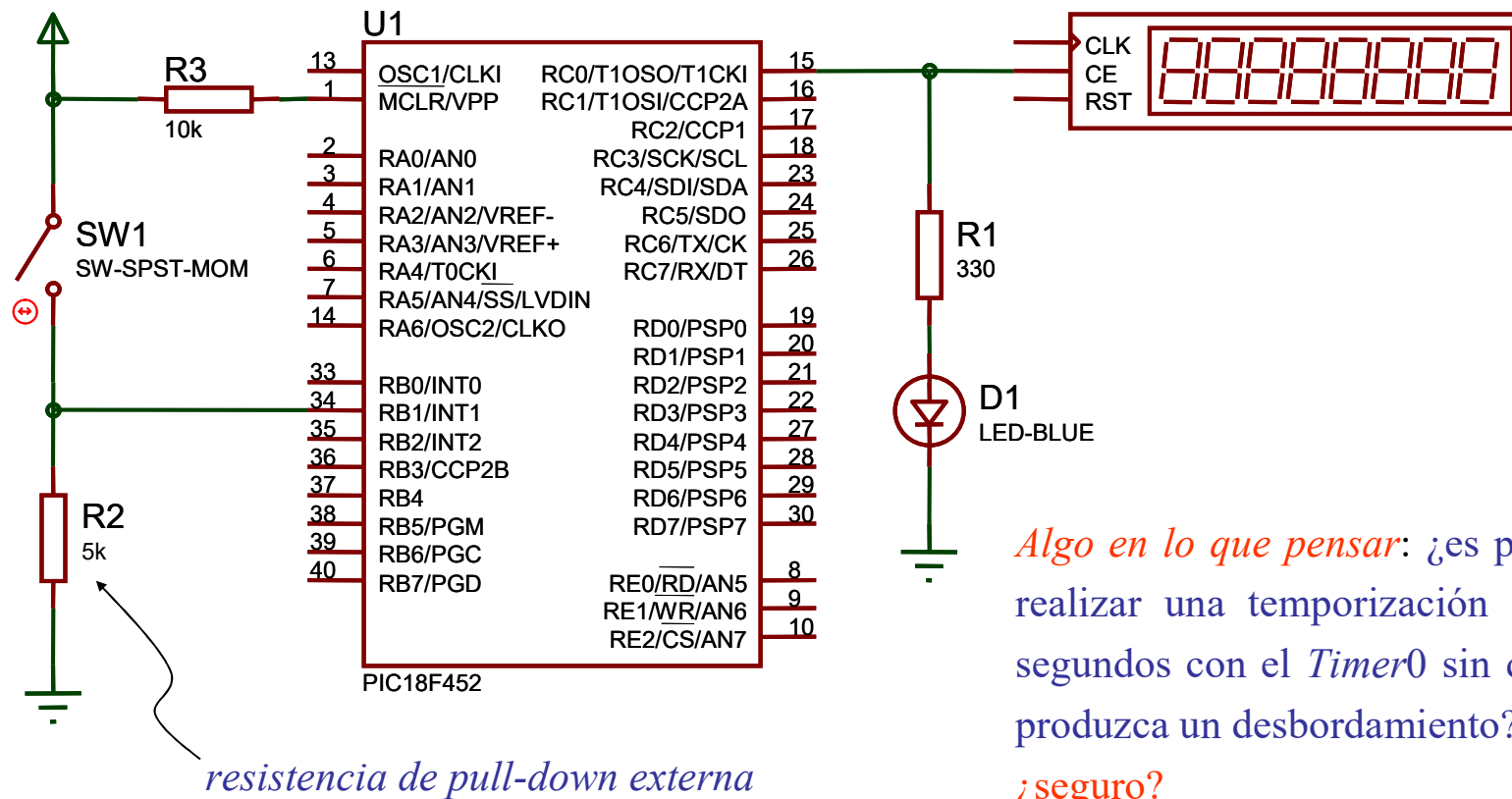
$t_2 = (1-D)T$

IMPORTANTE: antes de ponerte a escribir código, determina los valores de T , de t_1 y de t_2 . A continuación determina los valores a guardar en el registro de configuración del *Timer0* para realizar una temporización de t_1 segundos y de t_2 segundos. Puedes echarle una ojeada al ejemplo de la pág. 62 del archivo NotasPIC18F452.pdf.

Práctica 6 b): Determina el tiempo máximo que se puede temporizar con el Timer 0, teniendo en cuenta que la frecuencia de reloj del microcontrolador PIC18F452 es de 8MHz y que sólo se puede producir 1 desbordamiento del Timer 0

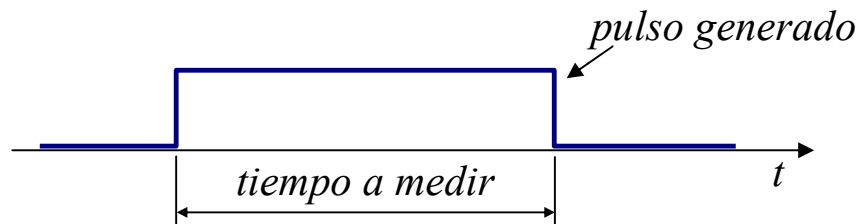
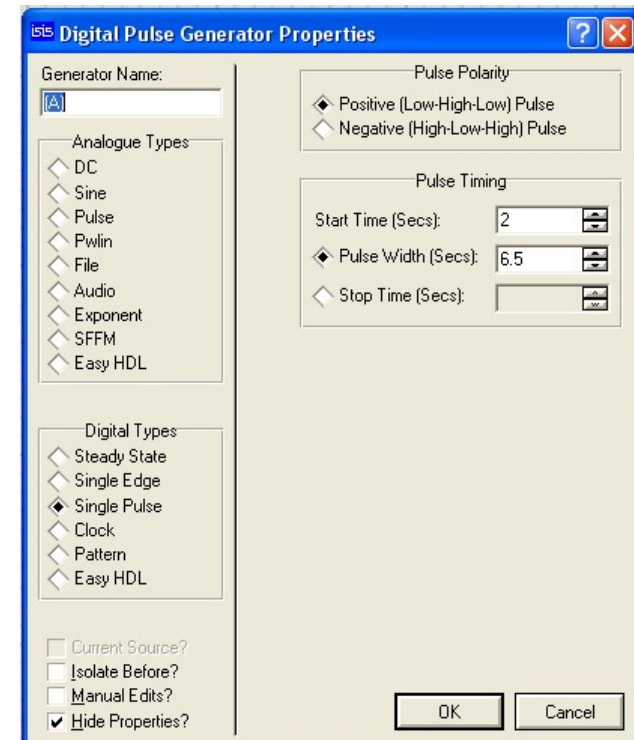
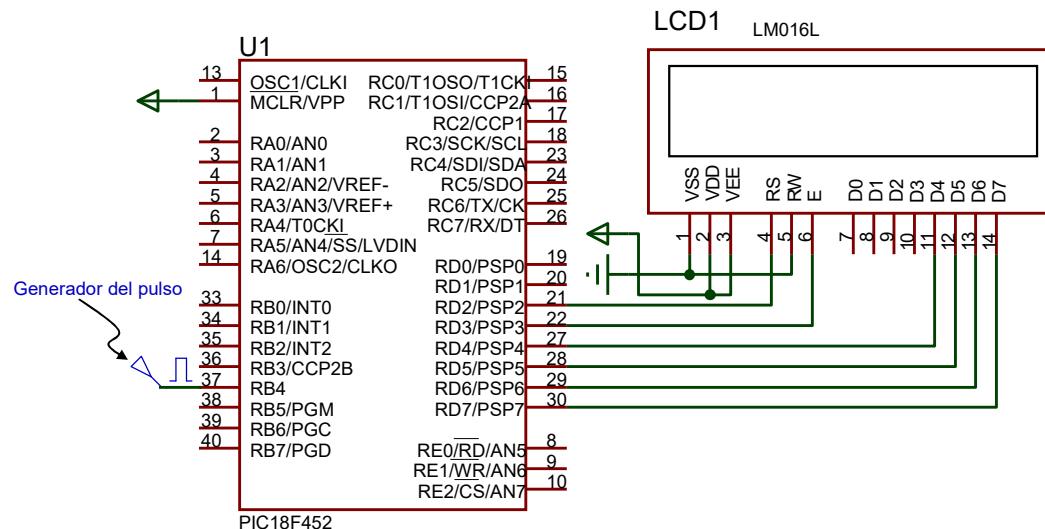
Práctica 6 c): Construir un temporizador de 1 minuto, no redispachable. Para ello se utilizará la interrupción INT1 y el Timer0. Cada vez que se presiona el pulsador, la salida RC0 debe ponerse a nivel alto durante 1 minuto (el *led* deberá estar encendido durante 60 segundos).

Componentes ISIS: PIC18F452, RES, SW-SPST-MOM, LED-BLUE, COUNTER TIMER



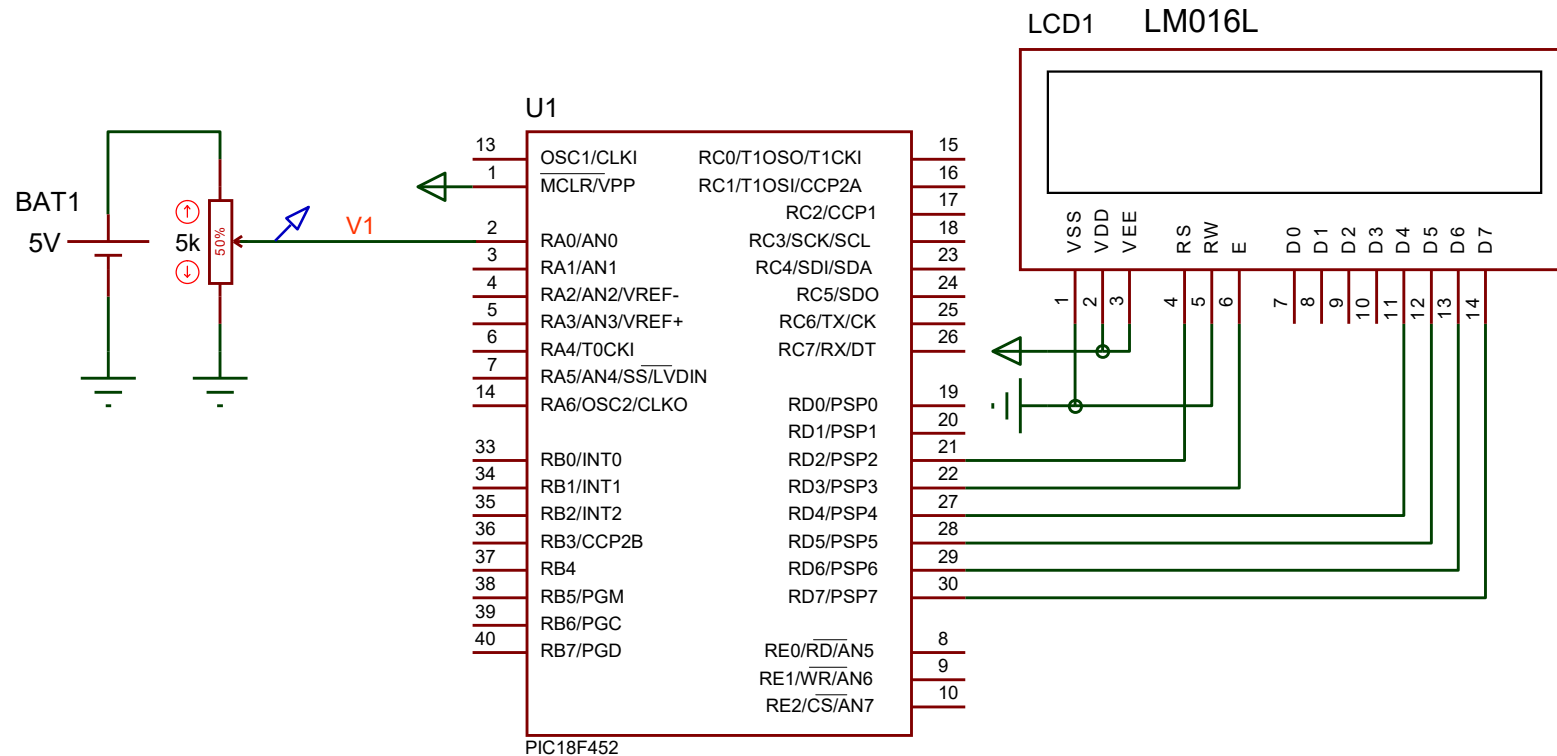
Práctica 6 d): En este apartado hay que medir la duración del (único) pulso que describe la señal aplicada al terminal RB4 del PIC18F452. La duración del pulso se debe representar, en segundos, en el LCD. Nota: para verificar el funcionamiento del código hay que configurar el dispositivo generador de pulsos de ISIS tal como se indica en la figura.

Componentes ISIS: PIC18F452, LM016, DPULSE



Práctica 7 a): Esta práctica consiste en construir un medidor de tensiones continuas (un voltímetro). La tensión (**V1**) a medir deberá muestrearse con una frecuencia de 1Hz. En este apartado se puede utilizar la función `delay_ms()`

Componentes ISIS: PIC18F452, POT-HG, CELL, LM016L



Algo en lo que pensar: ¿crees que existirán muchos cuerpos cuya temperatura cambie significativamente en 1,5 segundos? ¿seguro?

The diagram shows a PIC18F452 microcontroller connected to an LCD1601 display and a temperature sensor (LM35). The PIC18F452 is configured as a PIC18F452. The LCD1601 is connected to the PIC18F452 via a 4-bit data bus (D0-D7) and control lines (RS, RW, E). The PIC18F452 is also connected to a temperature sensor (LM35) via a 1-wire interface. The temperature sensor output is connected to the PIC18F452's VOUT pin. The PIC18F452 is also connected to a 1-wire interface (Vout) via a 1-wire interface.

$$T_{(^{\circ}C)} = 100 \cdot V_{out}$$

$$+2^{\circ}C \text{ to } +150^{\circ}C$$

^o $\equiv 223 \text{ ó } 178_{\text{ASCII}}$: different LCD displays have different char code for degree

Práctica 7 c): Modificar el código y el circuito del apartado 7 b) de modo que al pulsar un botón el sistema cambie la escala de la temperatura representada (grados Celsius, Farenheit y Kelvin). En esta apartado **no** se pueden utilizar las funciones `delay_ms()`, `delay_us()`, `ADC_Get_Sample()`, `ADC_Read()`. Tampoco se puede utilizar la técnica de *polling*. Componentes ISIS: PIC18F452, LM35, LM016L, SW-SPST-MOM

Nota:

$$^{\circ}\text{K} = ^{\circ}\text{C} + 273,15$$

$$^{\circ}\text{F} = 1,8 \text{ } ^{\circ}\text{C} + 32$$

$^{\circ}\text{C}$	$^{\circ}\text{K}$	$^{\circ}\text{F}$
2	275,15	35,6
10	283,15	50
30	303,15	86
60	333,15	140
90	363,15	194
100	373,15	212
120	393,15	248
150	423,15	302

Práctica 8 a): En este apartado hay que construir un medidor de presiones basado en el uso de un sensor MPX4115A. La frecuencia de muestreo de la tensión proporcionada por el sensor debe ser de 1 hercio. La presión medida se representará en un LCD. Las unidades de la presión a representar cambiarán a medida que se presione el pulsador, siguiendo la siguiente secuencia: kPa, PSI, Atm, mBar, mmHg, N/m² y kg/cm². Componentes ISIS: PIC18F452, MPX4115, RES, Button, CAP, LM016L. En esta apartado se puede utilizar la función delay_ms().

Pregunta: ¿a qué crees que se debe el error en el valor de la presión representada?

Nota:

1 Psi = 6'8927 kPa

1 Atm = 101'325 kPa

1 mBar = 0'1 kPa

1 mmHg = 0'13328 kPa

1 N/m² = 1 Pa = 0'001 kPa

1 Kg/cm² = 98'039 kPa

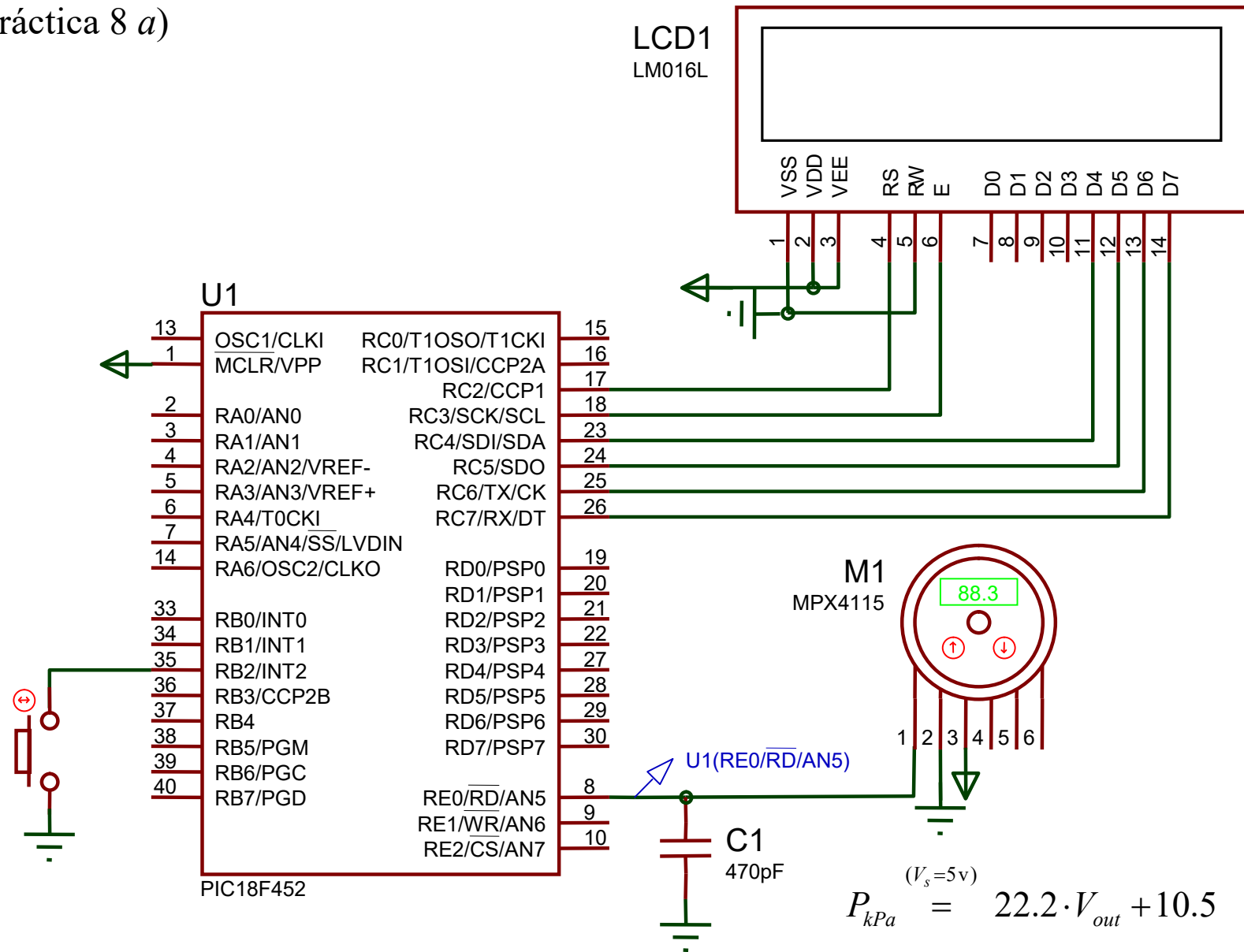
1 kp/cm² = 9'81 kPa

$$P_{kPa}^{(V_s=5v)} = 22.2 \cdot V_{out} + 10.5$$

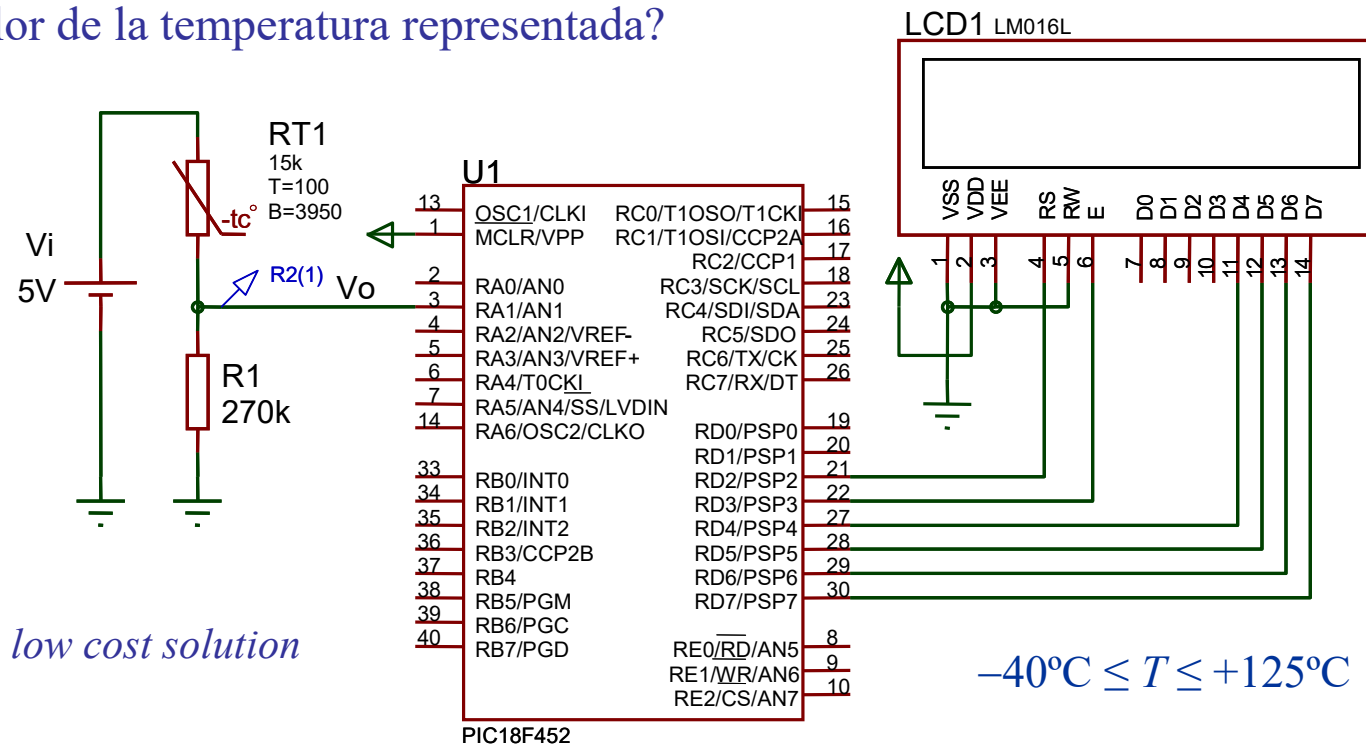
kPa	PSI	Atm	mBar	mmHg	N/m ²	kg/cm ²	kp/cm ²
100	14,5	0,9869	1000	750,3	10 ⁵	1,02	10,19

Nota: El rango de presiones que puede medir el sensor MPX4115A es: $15\text{kPa} \leq P \leq 115\text{kPa}$

(Práctica 8 a)



Práctica 8 b): En este apartado hay que construir un termómetro basado en el uso de un termistor NTC (su resistencia tiene un coeficiente de temperatura negativo). La tensión V_o , a partir de la cual se puede determinar la temperatura a la que se encuentra la NTC (ver notas convertidor AD), se muestreará cada 0.5 segundos. El valor de la temperatura se representará en el LCD en grados Celsius. En esta apartado no se pueden utilizar las funciones `ADC_Get_Sample()`, `ADC_Read()`, `delay_ms()`, `delay_us()`. Componentes ISIS: PIC18F452, RES, LM016L, CELL, NCP15XW153. *Pregunta:* ¿A qué se debe el error en el valor de la temperatura representada?



Práctica 9 a): En esta práctica se trata de diseñar un filtro IIR que elimine el ruido en una señal de audio. El archivo que guarda las muestras de dicha señal se denomina Audio6000.wav y está disponible en el escritorio de los PCs del laboratorio. Los pasos a dar se resumen en lo siguiente:

1º: Reproduce el archivo de audio con el fin de comprobar que la señal tiene ruido.

2º: Determina el contenido en frecuencia de la señal de audio. Para ello hay que calcular la transformada de Fourier (FFT) de dicha señal utilizando el programa dsPICworks. Observando dicho contenido en frecuencia debería resultar evidente la frecuencia (o frecuencias) en la que se concentra el ruido a eliminar.

3º: Diseña un filtro para eliminar el ruido utilizando el programa WFilter. Para ello en File → New elige:

_ El tipo de respuesta en frecuencia del filtro a diseñar (*Implementation*: IIR).

_ El tipo de filtro a utilizar (*Selectivity*), de acuerdo con el valor de las frecuencias en las que se concentra el ruido a eliminar y las frecuencias en las que se concentra la información.

_ La frecuencia de muestreo (*Sampling frequency*: 44100Hz)

_ El tipo de aproximación polinómica que se va a utilizar (*Approximation*)

(haz click en *Next*)

_ La amplitud máxima (*Gain*), en dBs, del rizado que puede presentar el módulo de la respuesta en frecuencia del filtro en su banda (o bandas) de paso. Los valores a indicar en dBs son **negativos**. Las frecuencias de corte en hercios (*Edge Freq*) (*Passband specification*)

_ La atenuación mínima (*Gain*), en dBs, que debe introducir el filtro en su banda prohibida. Los valores a indicar en dBs son negativos. Las frecuencias de corte en hercios (*Edge Freq*) (*Stopband specification*).

(haz click en *Design Filter*)

4º: Comprueba que la respuesta en frecuencia del filtro diseñado presenta las características deseadas. (*Edit – Response parameters*)

5º: Utiliza el programa *WFilter* para generar la señal de salida **flt.wav* del filtro cuando en su entrada se aplica la señal cuyas muestras se guardan en el archivo *Audio6000.wav*

6º: Utilizando el programa *WFilter*, reproduce el archivo de audio que guarda las muestras de la señal de salida del filtro con el fin de comprobar la efectividad del filtro diseñado.

7º: Utiliza el programa *dsPICWorks* para ver la respuesta en frecuencia de la señal de salida del filtro (no olvides que hay que convertir previamente el formato *.wave* a *.tim*).

8º: ¿Cuál es el orden de la función de transferencia (discreta) del filtro diseñado utilizando el programa *WFilter*?

Nota: A la hora de filtrar una señal de audio hay que tener en cuenta lo siguiente:

- Teóricamente, un oído humano sano puede oír señales senoidales (tonos) de frecuencias comprendidas entre 20Hz y 20kHz.
- Los tonos entre ≈ 20 y ≈ 64 Hz no los oímos todas las personas.
- Los tonos entre ≈ 65 y ≈ 250 Hz son tonos graves medios.
- Los tonos entre ≈ 250 y ≈ 2000 Hz contienen el tono fundamental y los primeros armónicos de la mayoría de las fuentes sonoras.

- Los tonos entre ≈ 2000 y $\approx 4096\text{Hz}$ corresponden al rango de tonos al que el oído humano tiene una mayor sensibilidad.
- Los tonos entre ≈ 4097 y $\approx 16000\text{Hz}$ corresponden a sonidos desagradables.
- Los tonos superiores a $\approx 16000\text{Hz}$ no todas las personas los oímos.

En resumen: el rango de frecuencias de la voz humana que contiene la mayor parte de la información relevante está comprendido entre los 200Hz y los 4000Hz .

Práctica 9 b): En este apartado se trata de diseñar un filtro **FIR** que elimine el ruido en la señal de audio del apartado anterior. Los programas a utilizar son los mismos que en el apartado anterior. Utiliza una ventana de *Hamming*. Al comprobar el funcionamiento del filtro diseñado (\equiv filtrar la señal de audio con ruido) deja la opción por defecto (*Convolution*) en *Fir Filtering Method*.

Práctica 9 c): En este apartado se trata de diseñar un filtro **IIR** de tipo *paso alto* cuya banda de transición está comprendida entre 4000Hz y 10000Hz. La idea sigue siendo eliminar el ruido en la señal de audio cuyas muestras se guardan en el archivo Audio6000.wav.. ¿Cuál es el efecto de este filtro sobre la señal de audio?. ¿Por qué?

Práctica 9 d): En este apartado se trata de diseñar un filtro **IIR** de tipo *paso bajo* cuya banda de transición está comprendida entre 100Hz y 5000Hz. La idea sigue siendo eliminar el ruido en la señal de audio cuyas muestras se guardan en el archivo Audio6000.wav. ¿Cuál es el efecto de este filtro sobre la señal de audio?. ¿Por qué?

Práctica 9 e): En este apartado hay que diseñar un filtro que sólo deje pasar el ruido (pitido).

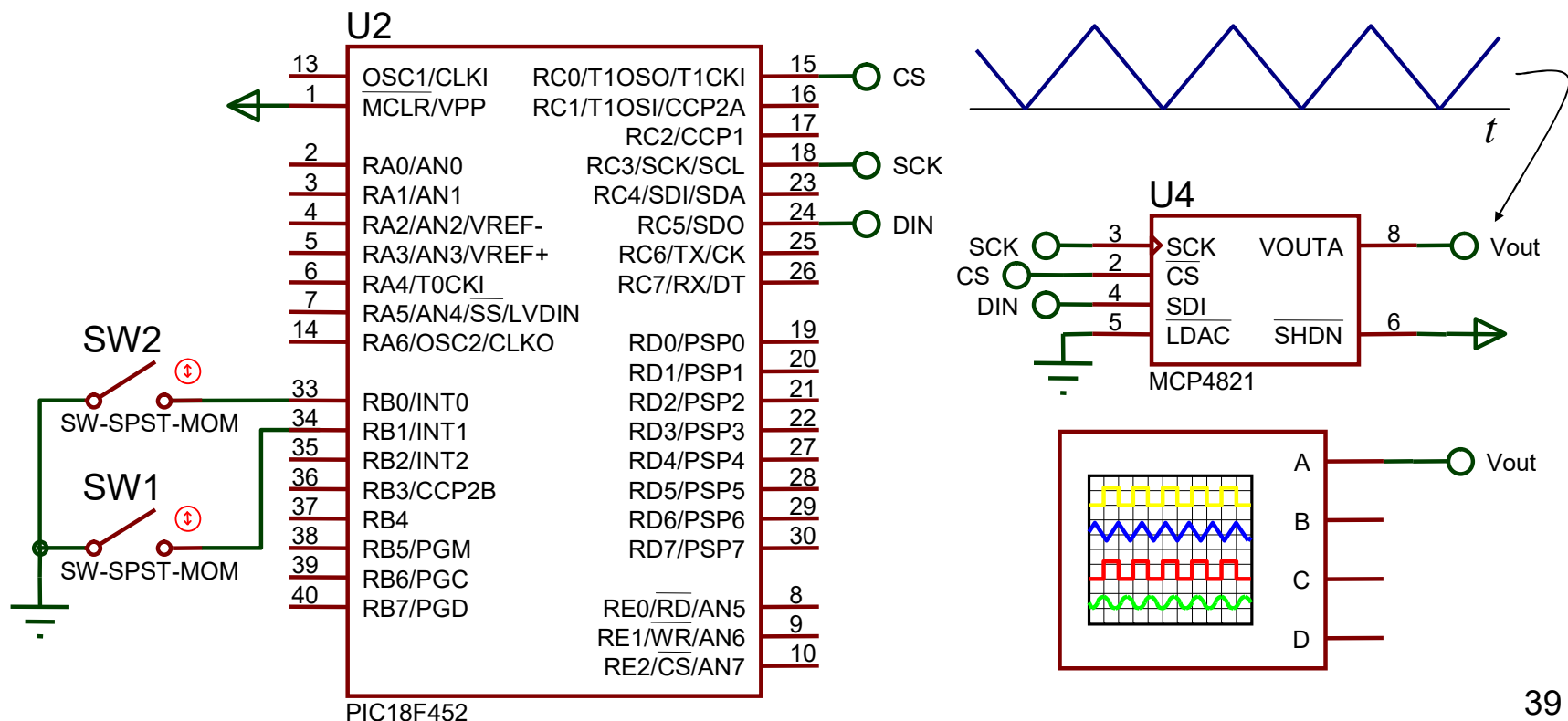
Práctica 10 *a)*: Genera una tensión constante con el convertidor D/A

b) Genera una señal triangular periódica como la indicada más abajo.

c) Genera una señal triangular de modo que cada vez que se pulse SW2 aumente su periodo y cada vez que se pulse SW1 disminuya su periodo.

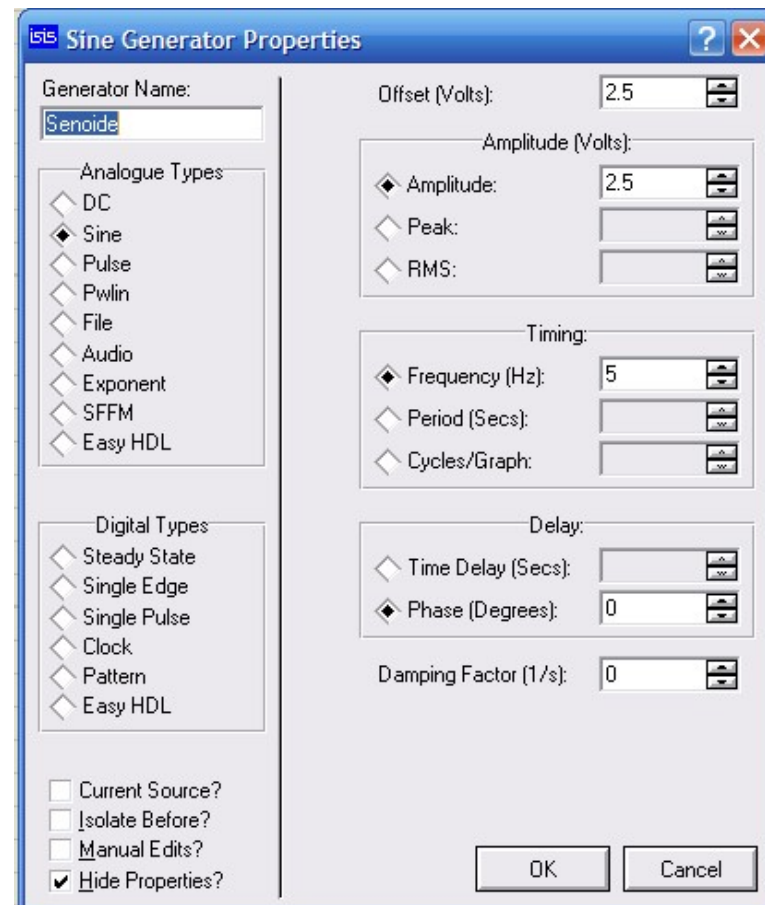
Componentes ISIS: PIC18F452, SW-SPST-MOM, MCP4821, Oscilloscope, DC.

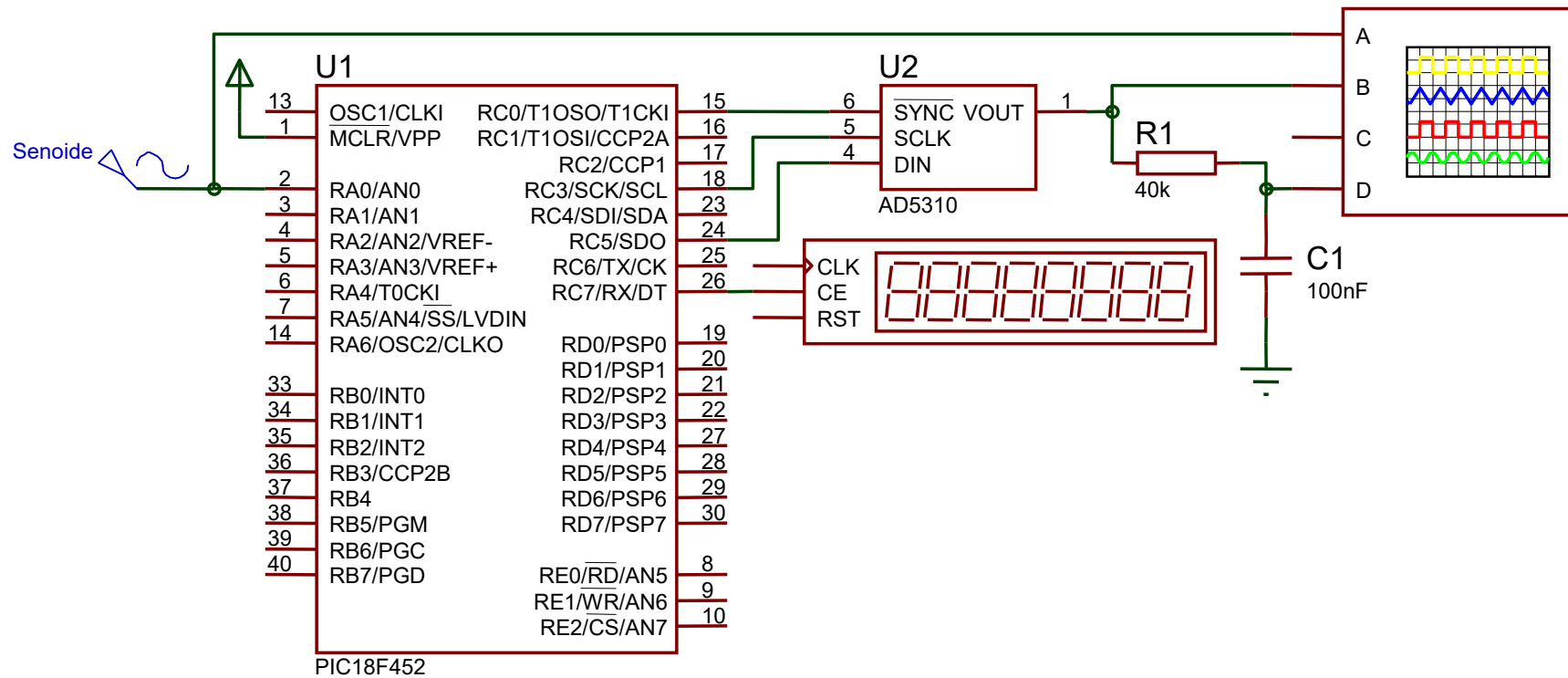
Nota: configura el canal A del osciloscopio para que represente el modo DC.



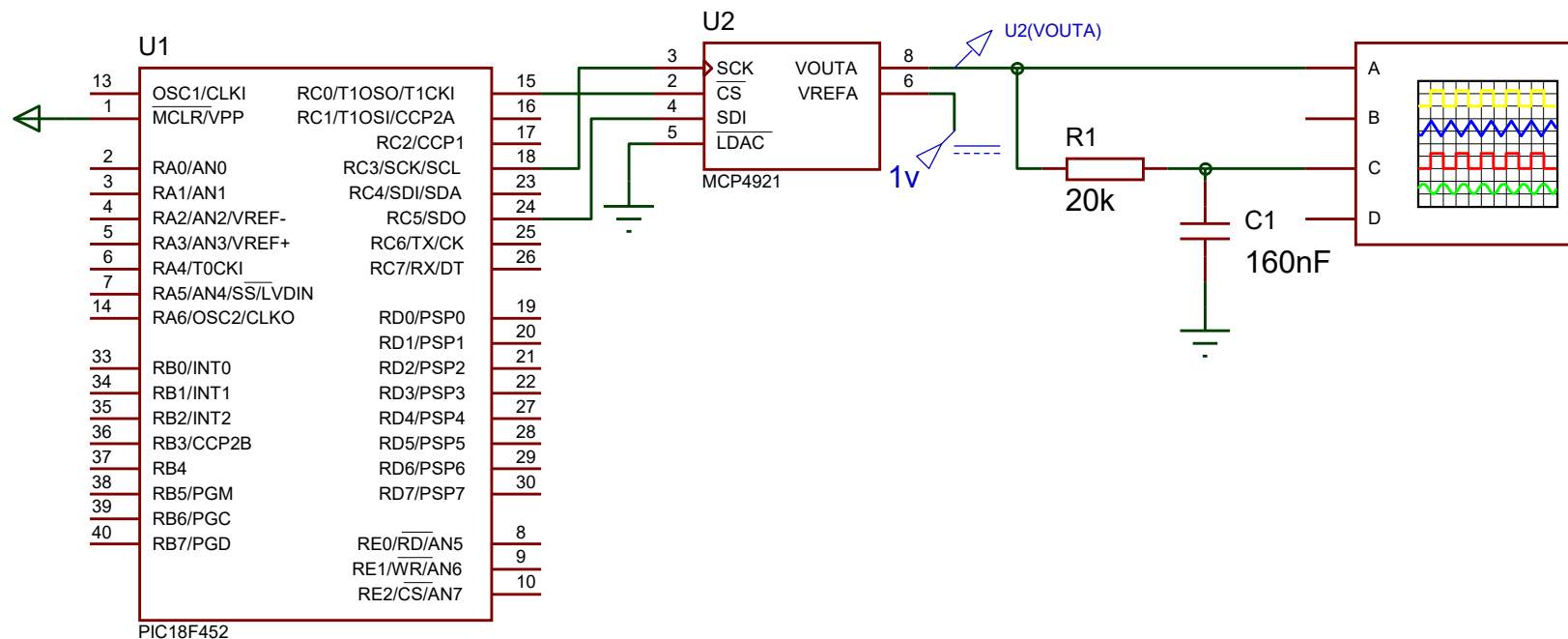
Práctica 10 *d*): Hay que muestrear una señal senoidal caracterizada porque su frecuencia es de 5Hz, su valor de pico es igual a 2,5V y su valor medio es igual a 2,5V. El periodo de muestreo debe ser igual a 10mseg (comprueba su valor con el *Counter Timer*). Las muestras deben enviarse a un convertidor D/A AD5310.

Componentes ISIS: PIC18F452, AD5310, Oscilloscope, Sine generator, Counter Timer.





Práctica 10 e): Genera una señal senoidal de frecuencia 50Hz, de valor de pico igual a 2v y con valor medio igual a 1v. Componentes ISIS: PIC18F452, MCP4921, Oscilloscope, RES, CAP. Nota: los canales A y C del osciloscopio deben representar el modo DC (pon la escala vertical en 0.1v para ver el efecto del filtro paso bajo).

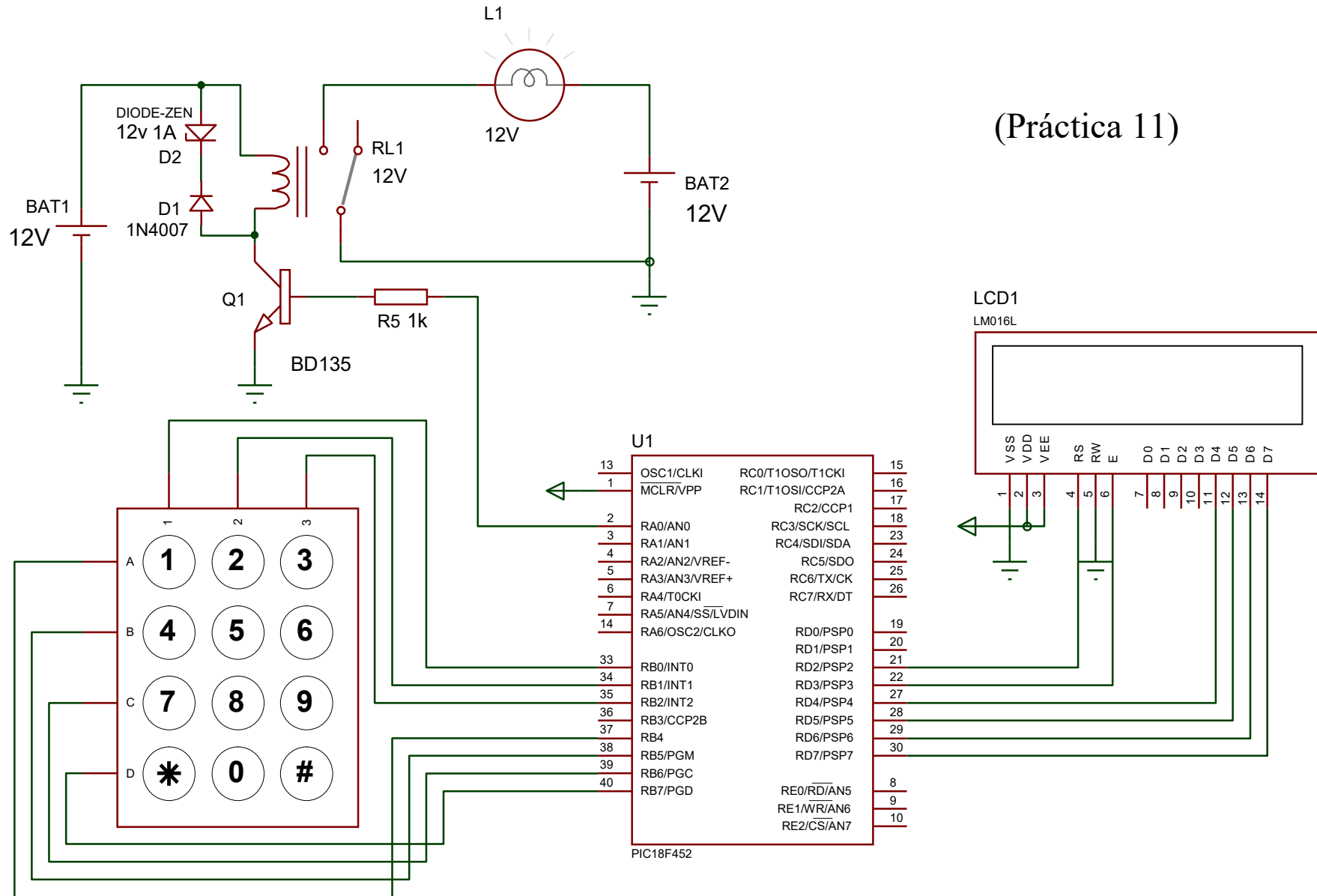


Nota: Los valores de las muestras se pueden generar con el programa *Matlab*

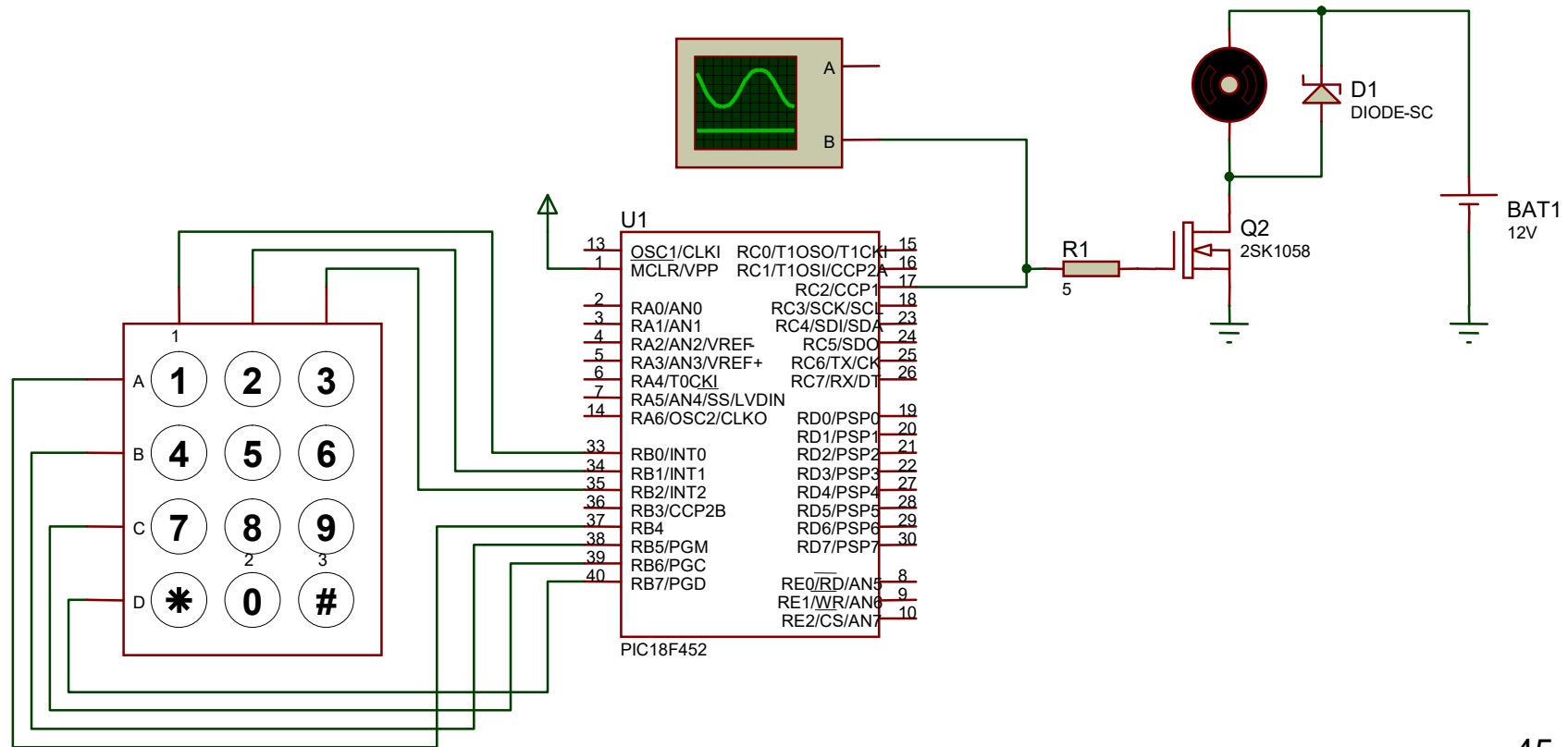
Práctica 11: Diseñar el sistema de control de una cerradura electrónica. Cada vez que se introduzca, por medio del teclado, una clave de 5 dígitos (correcta) se deberá accionar un relé durante 5 segundos (ver página siguiente). En ISIS la lámpara permanece encendida mientras la cerradura está abierta. Cada dígito de la clave introducido por el usuario se indicará en el LCD con el carácter '*'. El listado de claves válidas se guardará en la memoria **EEPROM** del microcontrolador.

Componentes ISIS: PIC18F452, KEYPAD-PHONE, BD135, CELL, RELAY, LAMP, LM016L, RES, POT-HG, DIODE-ZEN, 1N4007.

(Práctica 11)

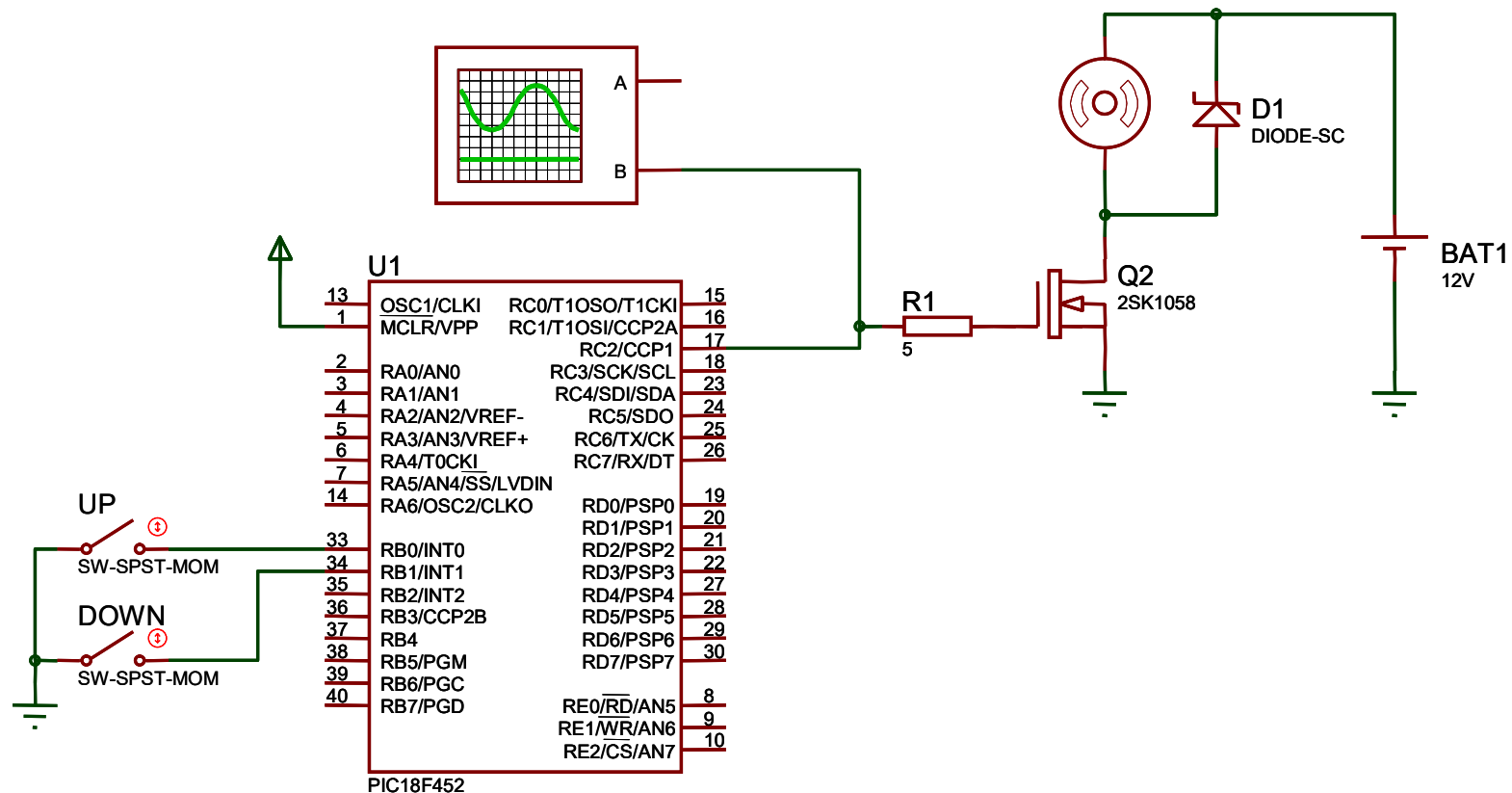


Práctica 12 a): Control de la velocidad de giro, en bucle abierto, de un ventilador accionado por un motor DC, utilizando el módulo **CCP1**. Para controlar la velocidad de giro sólo hay que generar una señal digital de 1KHz, con un ciclo de trabajo proporcional al valor (de 1 dígito) introducido mediante el teclado. Componentes ISIS: PIC18F452, KEYPAD-PHONE, RES, CELL, MOTOR, 2SK1058, DIODE-SC.



Práctica 12 b): Se trata de hacer lo mismo que en el apartado anterior, pero ahora para aumentar/reducir la velocidad de giro sólo se dispone de 2 pulsadores. Cada vez que se pulsa el botón UP/DOWN la velocidad de giro del motor aumenta/disminuye.

Componentes ISIS: PIC18F452, SW-SPST-MOM, RES, CELL, MOTOR, 2SK1058, DIODE-SC



Práctica 12 c): En este caso se trata de diseñar el sistema que controla la orientación de una cámara de video. La orientación de dicha cámara se controla por medio de un *servomotor*, el cual tiene las siguientes características:

Ancho mínimo de los pulsos: 1mseg. (ángulo = -90°)

Ancho máximo de los pulsos: 2 mseg. (ángulo = $+90^\circ$)

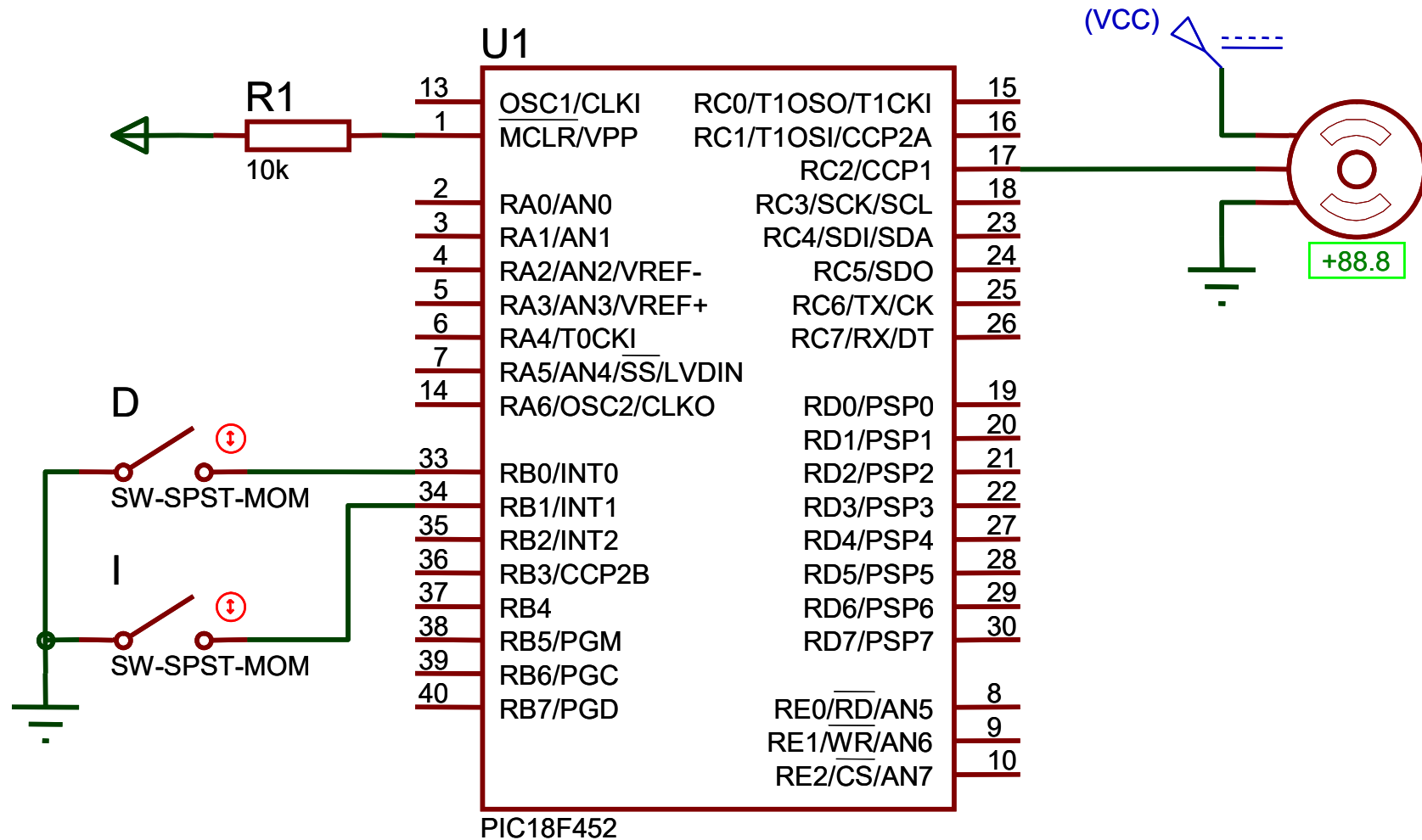
Periodo de la señal PWM: 4mseg.

Cuando se pone en funcionamiento el sistema de vigilancia, la cámara debe situarse en la posición correspondiente a 0° . A partir de dicho instante, cada vez que se pulse el botón D/I, la cámara debe girar aproximadamente 10° hacia la Derecha/Izquierda.

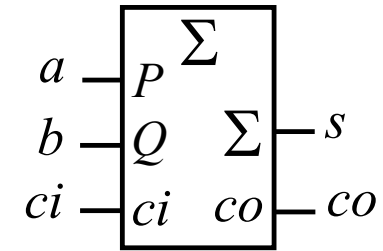
Componentes ISIS: PIC18F452, Motor-PwmServo, SW-SPST-MOM, RES.

Nota: la frecuencia de la señal de reloj del microcontrolador debe ser $f_{osc} = 4\text{MHz}$.

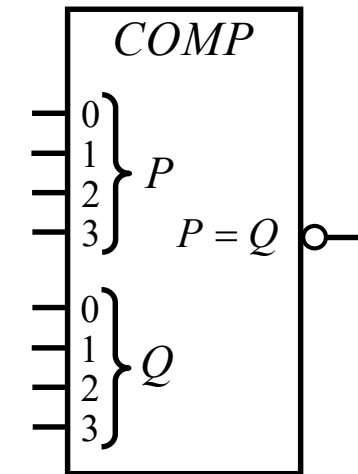
Práctica 12 c)



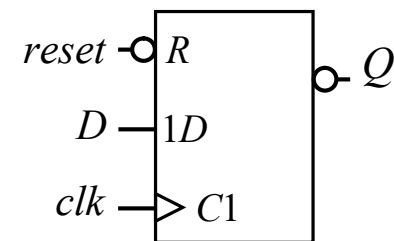
Práctica 13 a): Escribe el código en VHDL que describe el comportamiento del sumador total de 1 bit indicado en la parte derecha y comprueba su correcto funcionamiento en una Basys 2.



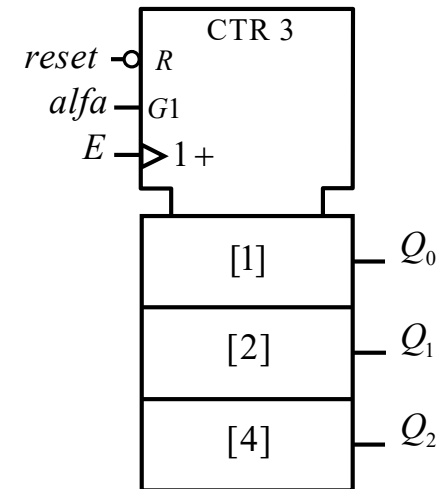
Práctica 13 b): Escribe el código en VHDL que describe el comportamiento del comparador de magnitud de 4 bits indicado en la parte derecha y comprueba su correcto funcionamiento en una Basys 2.



Práctica 13 c): Escribe el código en VHDL que describe el comportamiento del *flip flop D* sincronizado con los flancos de subida indicado en la parte derecha y comprueba su correcto funcionamiento en una Basys 2.



Práctica 13 *d)*: Escribe el código en VHDL que describe el comportamiento del contador de 3 bits indicado en la parte derecha y comprueba su correcto funcionamiento con el simulador ISIM (simulación funcional).



Práctica 13 *e)*: Escribe el código en VHDL que describe el comportamiento de un circuito con 8 entradas, en cuyas salidas se indica (en binario) el doble del número de entradas que están a 1. Para comprobar el funcionamiento del circuito se utilizará una placa de entrenamiento Basys 2.

Práctica 13 *f)*: Escribe el código en VHDL que describe el comportamiento de un circuito al que llegan 4 número binarios, sin signo, $A(a_1 a_0)$, $B(b_1 b_0)$, $C(c_1 c_0)$ y $D(d_1 d_0)$, que proporciona el mayor de los cuatro números. Para comprobar el funcionamiento del circuito se utilizará una placa de entrenamiento Basys 2.

Nota: en previsión de que alguno de los códigos escritos se verifique experimentalmente en una placa Basys 2, al crear un proyecto se debe indicar lo siguiente:

New Project Wizard

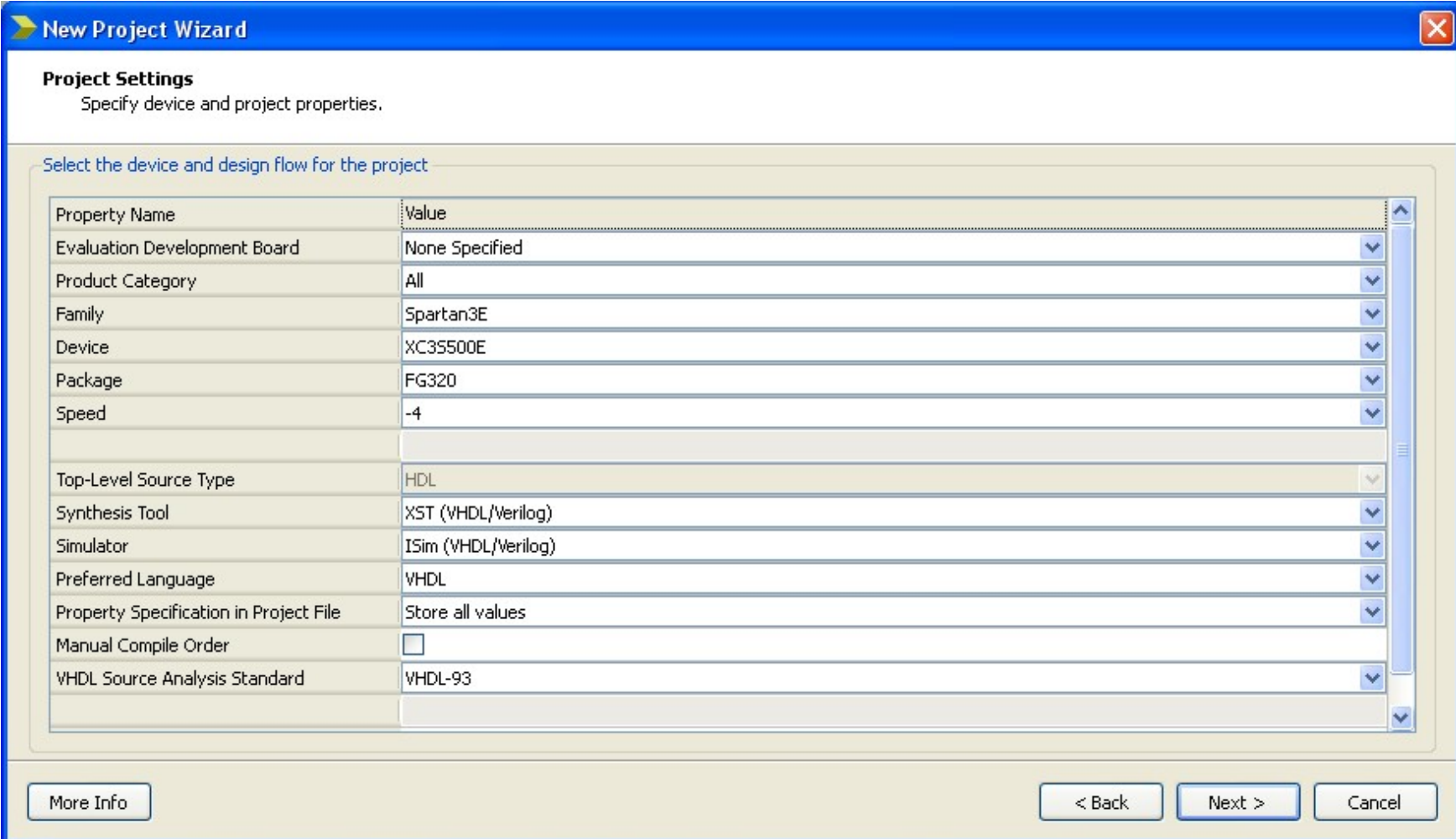
Project Settings
Specify device and project properties.

Select the device and design flow for the project

Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan3E
Device	XC3S100E
Package	CP132
Speed	-4
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project File	Store all values
Manual Compile Order	<input type="checkbox"/>
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	<input type="checkbox"/>

More Info < Back Next > Cancel

En el caso de utilizar una placa Spartan3E de Xilinx, al crear un proyecto se debe indicar lo siguiente:

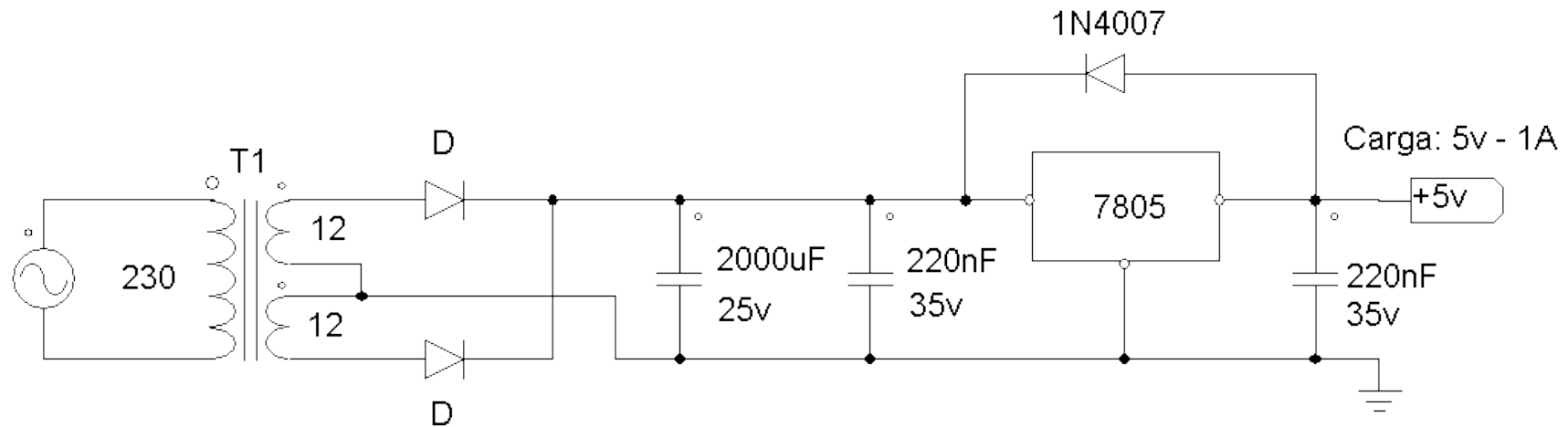


The image shows the 'New Project Wizard' dialog box, specifically the 'Project Settings' step. The title bar reads 'New Project Wizard' with a close button. Below the title bar, the text 'Project Settings' is followed by the instruction 'Specify device and project properties.' The main area is titled 'Select the device and design flow for the project' and contains a table of properties.

Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan3E
Device	XC3S500E
Package	FG320
Speed	-4
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project File	Store all values
Manual Compile Order	<input type="checkbox"/>
VHDL Source Analysis Standard	VHDL-93

At the bottom of the dialog, there are three buttons: 'More Info', '< Back', and 'Next >', along with a 'Cancel' button.

Esquema de una fuente de alimentación lineal (admite fluctuaciones en el valor eficaz de la tensión de red de $\pm 20\%$)



Trafo T1:
REF: 805-136 (RS-AMIDATA)
25+25 = 50VA

Diodos D: $I_{rms} < 1.72A$, $I_{avg} < 0.5A$ $V_{reverse} < 39.2v$

Condensador C: $V_{pico} < 18.8v$, $I_{rms} < 2.21A$, $I_{pico} < 7.5A$