

## Práctica 2 - Microcontroladores PIC: Proceso de diseño y entorno MPLAB

Fechas: 2, 3, 9 y 10 de Noviembre de 2006

Se parte de un ejemplo simple para mostrar el entorno, realizar la edición de un programa, el ensamblado, la simulación, la depuración sobre la aplicación y finalmente la grabación de un microcontrolador PIC16F877.

Se trata de un ejemplo para la placa de demostración PICDEM2 plus y se va a realizar la depuración mediante el módulo ICD2 de Microchip.

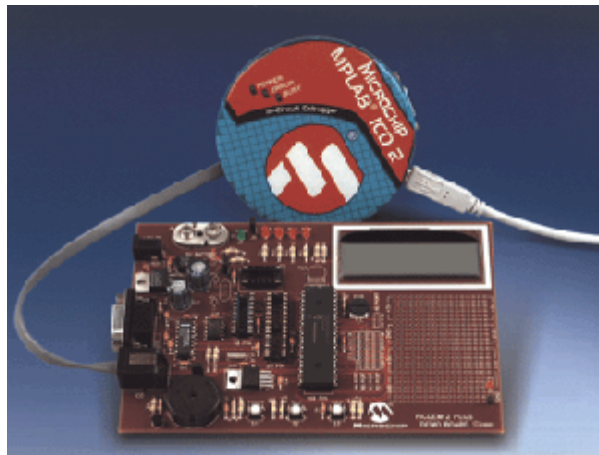


Figura 1.- PICDEM2 plus y módulo ICD2 de Microchip

De todos los elementos disponibles en la placa de prácticas, se utilizará un pulsador conectado a la entrada 4 del PORTA y los 4 diodos LED presentes a la salida del PORTB (véase el esquema de la placa PICDEM2 plus al final de este documento).

El ejemplo que se va a seguir consiste en contar y mostrar en binario por el PORTB, el número de veces que se actúa sobre el pulsador conectado al pin 4 del PORTA (si está pulsado, el pin estará a nivel 0 y si está libre a 1).

Se suministra el código fuente del ejemplo y se prueba la versión en el fichero denominado:

- [Cuenta.asm](#)  
(en esta versión no se tienen en cuenta posibles "rebotes" en el pulsador)

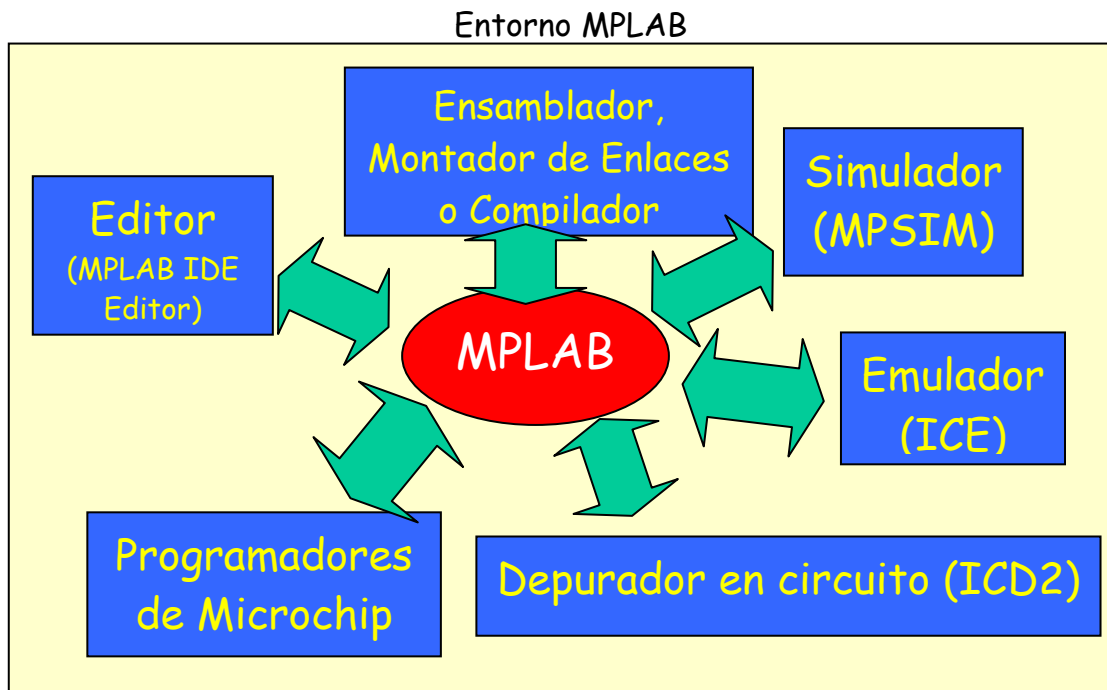


Figura 2.- Entorno de trabajo integrado para microcontroladores PIC:  
MPLAB-IDE

Pasos básicos a dar:

1.- Crear un fichero de código fuente

*File > New*

Se abre el editor de texto y escribimos el código fuente en ensamblador

2.- Una vez escrito el código completo, lo salvamos en un fichero con el nombre que se quiera y con la extensión **.asm** (p.e. cuenta.asm) que se ubicará en un directorio de trabajo determinado

*File > Save*

3.- Ahora hay que definir un proyecto. Hay dos posibilidades para definir un proyecto: manualmente o mediante un asistente (*Project Wizard*) que es la manera más sencilla y "guiada"

*Project > Project Wizard*

Seleccionaremos el dispositivo microcontrolador, la herramienta de generación de código máquina (**mpasmwin**), le daremos un nombre al proyecto e indicaremos el directorio donde lo situamos, a continuación nos interrogará sobre los ficheros a añadir al proyecto y añadiremos el fichero del código fuente que hemos escrito y almacenado en los pasos 1 y 2.

4.- Aparecerá el proyecto con el nombre asignado y la extensión **.mcp**. El paso siguiente será el ensamblado del código:

*Project > Build All*

o bien

*Project > Make*

No hay diferencia entre una opción u otra si hay un único fichero de código fuente. Sería distinto si hubiera varios ficheros que se ensamblan por separado y se enlazan luego con el "montador de enlaces" MPLINK.

En el ensamblado pueden producirse errores que deben corregirse usando como información los ficheros de listado (extensión **.lst**) y de errores (**.err**). Hasta que no desaparezcan los errores, no se podrá generar el código máquina en un formato compatible con el resto de herramientas (extensión **.hex**).

5.- Una vez realizado el ensamblado del programa sin errores, podemos pasar a comprobar el funcionamiento del programa mediante un **Simulador** (sólo software) o bien mediante una herramienta hardware que puede ser un **Depurador** o un **Emulador**.

Para la verificación con el SIMULADOR:

*Debugger > Select Tool > MPLAB SIM*

para activar el simulador

Para poder comprobar si nuestro programa funciona, debemos visualizar registros, bits, puertos, en dónde se encuentra la ejecución (memoria de programa), estado de la pila hardware, tiempo transcurrido, etc.

Para poder apreciar el estado de los registros durante la simulación, abrimos:

*View > Watch*

En esa ventana, añadimos los registros a visualizar y con qué formato de representación (hexadecimal, decimal, binario,...). Se añaden todos a partir de la tabla de símbolos existente o bien de la tabla de registros de funciones especiales.

Otras visualizaciones de interés pueden ser:

*View > Hardware Stack*

Pila hardware para "guardar" el PC

*View > File Register*

Todos los reg. de datos

*View > Special Function Registers*

Reg. de funciones especiales

*Debugger > Stopwatch*

Tiempo transcurrido

6.- Se deben configurar los bits de funcionamiento del dispositivo, aunque durante la simulación sólo tendría efecto el posible desbordamiento del *Watchdog* si se encontrara activo

*Configuration > Configuration Bits* bits de configuración  
*Watchdog Timer > OFF*

También se pueden configurar opciones de la simulación:

*Debugger > Settings...*  
*Clock*  
*Break options*

7.- Prepararemos las entradas externas que tendremos en la aplicación  
*(SE TRATA DE UNA SIMULACIÓN Y NO HABRÁ ENTRADAS QUE NO SEAN AQUELLAS QUE SE INTRODUCAN DESDE UNA VENTANA DE EJECUCIÓN)*

*Debugger >*  
*Stimulus Controller* Estímulos externos al MCU,  
(asíncronos y síncronos)  
*SCL Generator.* Fichero de estímulos síncronos

8.- Probar, ejecutar y depurar el código

Ejecución en diversos modos posibles:

*Debugger > Run* Ejecución continua hasta parada: *Halt*  
*> Animate* Actualizando presentación registros  
*> Step Into* Paso a paso entrando en subprogr.  
*> Step Over* Paso a paso "saltando" sobre subpr.  
*> Step Out* Paso a paso hasta salir de subpr.

Puntos de Ruptura:

*Debugger > Breakpoints* Establecer puntos de ruptura (se detiene la ejecución al llegar a esa instrucción)

9.- Si el programa no funciona, se debería modificar el código con el Editor y volveríamos a dar los pasos de nuevo desde el punto 4: **Ensamblar...**

## FICHEROS:

Ficheros del Entorno de Trabajo: *Workspace (.mcw)*

Ficheros de Proyecto: *Project (.mcp)*

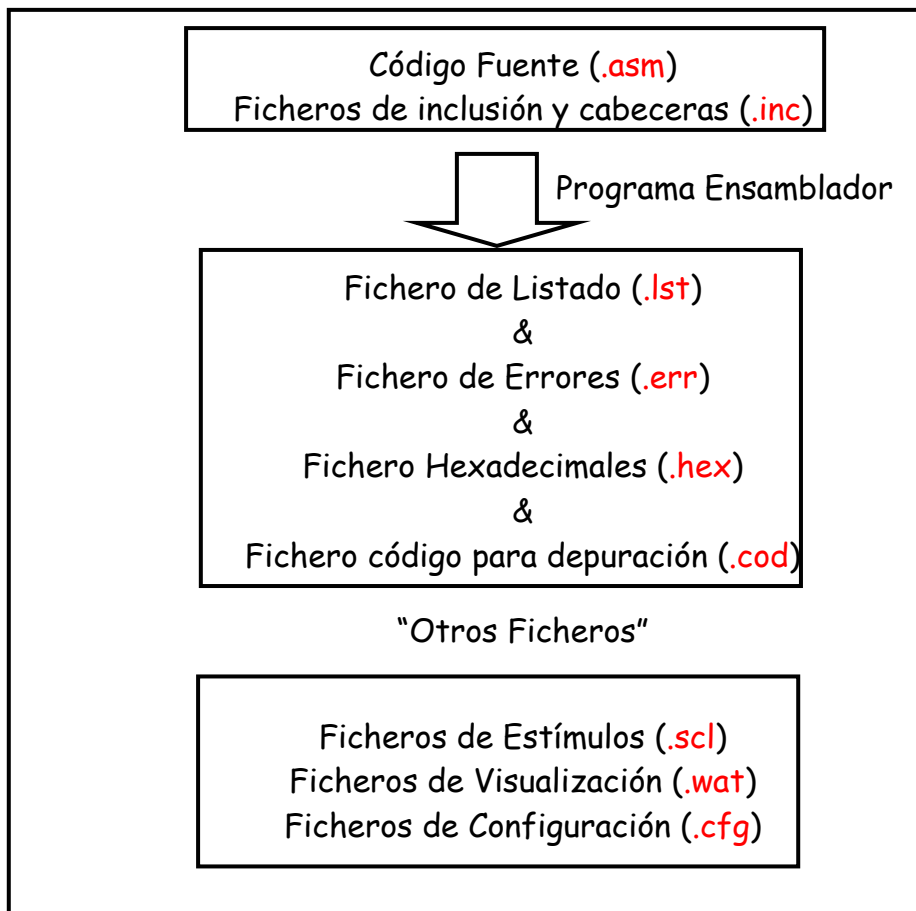


Figura 3.- Ficheros generados con MPLAB

### Proyectos: *Projects*:

Los Proyectos se salvan al cerrarlos

*Project > Close*

y se pueden recuperar posteriormente

*Project > Open...*

con lo que se restauran todas las ventanas abiertas, las herramientas activas y las condiciones en las que se estaba trabajando. De este modo, no resultaría necesario volver a definir los estímulos, ventanas de visión, etc.

Un proyecto engloba pues todos los ficheros necesarios para desarrollar la fase de ensamblado, simulación, depuración y programación

### Entornos de Trabajo: *Workspaces*

Contienen la información sobre el dispositivo, las ventanas abiertas, la configuración, las herramientas activas, etc. Un entorno de Trabajo **podría incluir a varios proyectos** que estuvieran simultáneamente abiertos

También se pueden salvar:

*File > Save Workspace*

y recuperar posteriormente:

*File > Open Workspace...*

## PRUEBAS SOBRE EL HARDWARE DE LA APLICACIÓN:

### ➤ Empleo del ICD2 como depurador real ("In Circuit Debugger 2")

Después de realizada la simulación del programa y considerando que éste "puede" funcionar desde el punto de vista software y trabajando con señales lógicas, procederíamos a utilizar el "debugger" ICD2 que nos permite incluir, en la placa de circuito impreso de nuestra aplicación, un microcontrolador grabado con nuestro código y trabajando en modo "depuración", lo que implica que desde el entorno de trabajo MPLAB IDE podemos ejecutar de manera continua, parando, paso a paso, con animación, estableciendo puntos de ruptura, viendo el estado de ciertos registros, etc.

De esta manera podemos comprobar que nuestro programa funciona o no conjuntamente con el hardware de la aplicación, manejando ya señales eléctricas reales.

Los pasos a dar serían los siguientes:

#### 1.- Activación del ICD2: selección del dispositivo y conexión

*Debugger > Select Tool > MPLAB ICD2*

Configuramos alimentación del ICD2 y comunicaciones

*Debugger > Settings*

*Power* (desactivamos alimentación de la tarjeta desde ICD2)

*Communication* (puerto USB ó COMn si procede)

Ahora alimentamos la tarjeta de la aplicación y nos aseguramos de que está conectado el ICD2 al puerto USB (o al puerto serie):

*Debugger > Connect*

Si todo es correcto habrá pasado el test inicial y estará listo, si no fuera así habrá que revisar si las comunicaciones son correctas, si hay tensión de alimentación, etc.

2.- Cargamos los bits de la palabra de configuración

*Configure > Configuration Bits...*

Y seleccionamos adecuadamente el oscilador, watchdog, etc.  
en este caso, SÍ QUE ES MUY IMPORTANTE HACER LA SELECCIÓN ADECUADA YA QUE TRABAJAMOS DIRECTAMENTE SOBRE EL MICROCONTROLADOR

3.- Antes de programar (grabar) el microcontrolador y hacerlo trabajar en modo "depuración", podemos hacer que el ICD2 ajuste la grabación a lo que ocupa realmente nuestro programa y que no sea necesario grabar la totalidad de la memoria de programa del microcontrolador

*Debugger > Settings*

*Program*

(Casilla: *Allow ICD2 to select memories...*)

4.- Grabamos el microcontrolador con nuestro código más la parte necesaria para llevar a cabo la depuración (son las 256 palabras finales y son transparentes al usuario)

*Debugger > Program*

Ejecutado este paso, el microcontrolador tendrá grabada en su memoria tanto nuestro programa como el "sistema operativo" que permite la depuración (poner en marcha, parar, transferir registros, etc). Pero para la puesta en ejecución real del microcontrolador con el código, será necesario que desde el entorno MPLAB, se dé la orden correspondiente a la EJECUCIÓN

5.- Ejecutamos, pero ahora ya sobre el hardware y en tiempo real

*Debugger > Run*

Ejecución continua hasta parada Halt

*> Animate*

Actualizando presentación registros

*> StepInto/ Over /Out* Ejecuciones paso a paso

6.- Visualización de registros, pila, etc. (como en el caso del simulador)

7.- Para pruebas sucesivas, se repetirían los sucesivos pasos de:

Edición y modificación del programa

Ensamblado

Programación

Ejecución

## PRUEBAS SOBRE EL HARDWARE DE LA APLICACIÓN:

### ➤ Empleo del ICD2 como Programador

Una vez comprobado el funcionamiento del programa con el funcionamiento en modo *debugger*, se trataría ahora de grabarlo en un microcontrolador, para lo cual podríamos utilizar también el ICD2 pero ahora en modo Programador

Los pasos a dar serían los siguientes:

1.- Desactivación del ICD2 como Debugger (MPLAB-ICD2 no puede trabajar a la vez en modo "depuración" y en modo "programador"):

*Debugger > Select Tool > None*

2.- Activación del ICD2 como Programador:

*Programmer > Select Programmer > MPLAB ICD 2*

Configuramos alimentación del ICD2 y comunicaciones

*Programmer > Settings*

*Power* (desactivamos alimentación desde ICD2)

*Communication* (puerto USB ó COMn si procede)

Ahora nos aseguramos de que está conectado al puerto USB y conectamos

*Programmer > Connect*

3.- Cargamos los bits de la palabra de configuración

*Configure > Configuration Bits...*

Y seleccionamos adecuadamente el oscilador, watchdog, etc.

4.- Grabamos el microcontrolador

*Programmer > Program*

Ejecutado este paso correctamente, el microcontrolador tiene en su memoria de programa, únicamente el código que nosotros le hemos transferido y empezará a ejecutarlo, a partir de la posición inicial, **en el momento en que lo liberemos de la conexión al ICD2.**

El microcontrolador YA ES AUTÓNOMO Y FUNCIONA DE ACUERDO AL  
PROGRAMA PRESENTE EN SU MEMORIA  
(No hay dependencia del entorno MPLAB)

## ANEXOS:

### Un posible algoritmo que se podría utilizar para el ejemplo propuesto

#### Inicialización:

- PORTB se define como puerto de salida
- PORTB se pone a cero al principio
- PORTA como puerto de entrada (por defecto son todos así)

#### Bucle de Ejecución continua

Si la tecla está pulsada ( $RA4==0$ ) entonces  
Llamada a Subprograma de Incremento  
si no  
Volvemos al principio del bucle

#### Subprograma de Incremento

Sumamos 1 a la combinación presente en el PORTB  
Bucle de espera a que la tecla esté liberada  
Si la tecla está pulsada ( $RA4==0$ ) entonces  
Seguimos esperando dentro del bucle  
si no  
Retornamos del Subprograma

### Código fuente del ejemplo

```

;Fichero CUENTA.ASM
;
;Programa de Prueba para la placa PICDEM-2 plus
;Por el Puerto B se saca en binario, el numero de veces
;que se pulsó la tecla que está conectada a la entrada RA4
;si pulsada a cero y si libre a 1
;
LIST    P=16F877                ;Directiva para definir listado y microcontrolador
INCLUDE P16F877.INC            ;Inclusión de fichero de etiquetas

ORG    0
BSF    STATUS,RP0              ;Paso al banco 1 de la memoria de datos
CLRF   TRISB                    ;para definir el PORTB como salida
BCF    STATUS,RP0              ;Volvemos al banco 0
CLRF   PORTB                    ;Ponemos a cero el PORTB para que aparezca ese
                                   ;valor cuando se defina como salida

ESPERA    BTFSS PORTA,4          ;Esperamos a que se pulse la tecla
          CALL INCREMENTO        ;en cuyo caso RA4 pasa a 0 y vamos a
          GOTO ESPERA            ;subprograma de INCREMENTO

;Subprograma de INCREMENTO

INCREMENTO INCF PORTB,F         ;Si se pulsó incrementamos PORTB

SOLTAR    BTFSS PORTA,4          ;no salimos hasta que se haya soltado
          GOTO SOLTAR            ;la tecla, en ese caso RA4 pasaría a 1
          RETURN                  ;y volvemos al programa principal
          END

```

### Placa PICDEM2 plus (hardware de la aplicación)

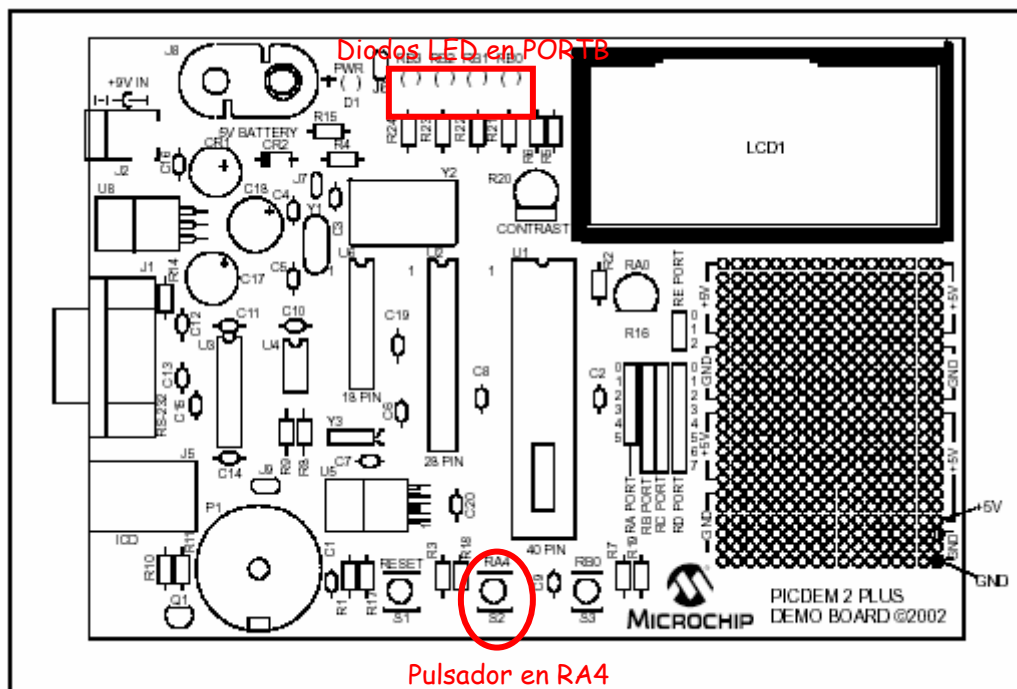


Figura 4.- Disposición Física de componentes en la PICDEM2 plus



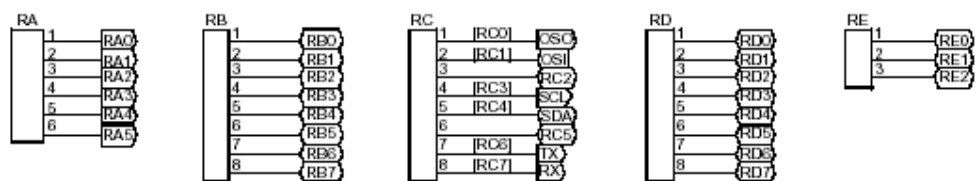
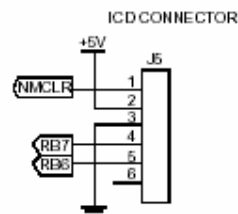
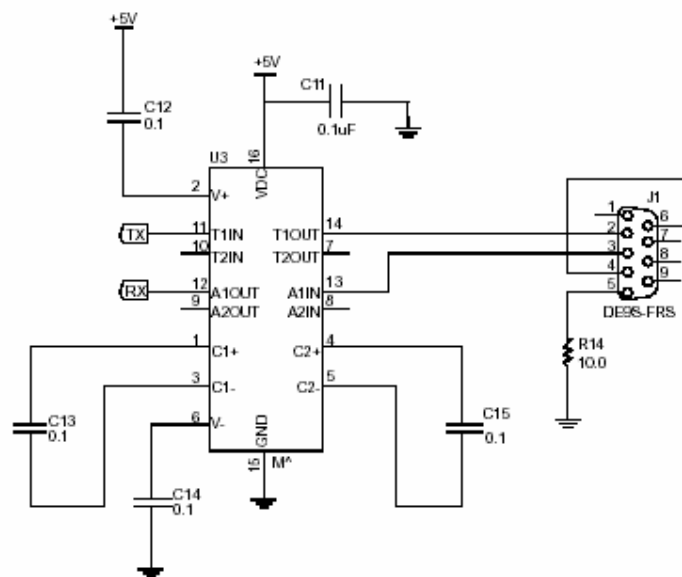
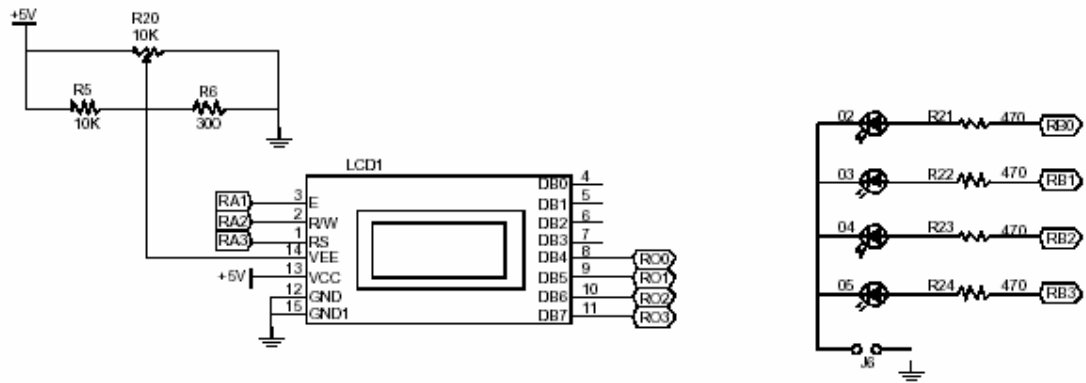


Figura 6.- Esquema de la placa PICDEM2 plus (parte II)