Práctica 2

Programación en lenguaje C

1. Objetivos

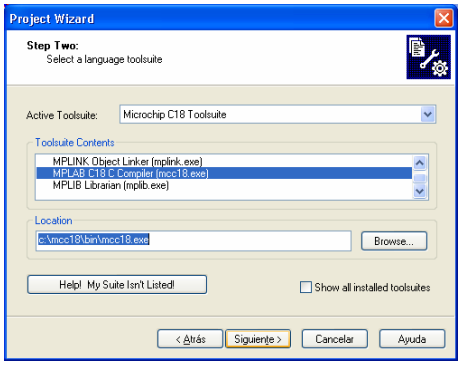
El objetivo de esta sesión es introducir la programación en C de sistemas basados en microcontrolador. En esta práctica el alumno debe comprender las limitaciones que se imponen sobre el lenguaje C con el que se programan las aplicaciones, para tratar de ajustarse a los recursos que tiene un microcontrolador. Así mismo se profundiza en el conocimiento de los puertos de entrada/salida y su utilización para mostrar información a través de displays de 7 segmentos.

2. Introducción

En esta sesión utilizaremos el compilador C18 para poder crear programas escritos en un C bastante básico pero mucho más sencillo de utilizar que el ensamblador normal. También se utilizan elementos periféricos básicos para mostrar información a través de la barra de 4 displays de 7 segmentos.

2.1 Compilador de lenguaje C

El compilador MPLAB C18 es un compilador de C para PICs de la familia 18 de microchip bastante completo y robusto. Su uso es muy simple ya que se integra dentro del entorno de desarrollo MPLAB utilizado en la práctica anterior. Para crear un proyecto utilizando las herramientas de compilación en C del C18 hay que seleccionar en el asistente, durante la creación del proyecto, la opción *Microchip C18 Toolsuite*:



Luego se añadirán los ficheros fuente con el código C que se pueden escribir directamente desde el editor del MPLAB y deben guardarse con un nombre acabado con la extensión *.c.

Toda la información acerca del compilador C18 se pueden consultar en el ***manual que se encuentra disponible en la página WEB de la asignatura*** es recomendable leer este manual para tener información sobre el acceso desde el C a las características propias de los PIC. A continuación se resumen algunas de estas características.

Acceso a los registros internos del PIC

Todos los registros internos del PIC están predefinidos en el propio C18, mediante el fichero de cabecera ***p18f2550.h*** por lo que se pueden utilizar directamente en el programa como si fueran variables de tipo *unsigned char*, o como bits independientes dentro de una estructura tipo *union* (para más información ver el manual del compilador en la página de la asignatura). A continuación se muestra un ejemplo de cómo utilizar la variable PORTA con la que podemos modificar los bits del puerto A:

Como variable *unsigned char:*

```
PORTA = 0x34;       /* Escribe el valor 0x34 en el puerto A */
```

En el caso de una lectura del puerto:

```
unsigner char valor_pa;
…..

valor_pa = PORTA;   /* Lee el valor de entrada del puerto A y lo salva en valor_pa*/
```

También se pueden utilizar la estructura *union* para acceder de forma más cómoda a los bits:

```
PORTAbits.RA0 = 1; /* pone a uno el pin cero RA0 de puerto A */
```

Es importante darse cuenta de que el compilador ya sabe en qué banco se encuentra el registro y ya se encarga de cambiar de banco cada vez que se accede a una variable de este estilo. Para más información sobre la utilización de los registros internos y los bancos de memoria consultar el manual de compilador.

Algunas funciones interesantes

Hay varias funciones implementadas como macros en ensamblador:

- Nop() es una función que equivale al nop del ensamblador y sirve para introducir un retraso de una instrucción.
- Sleep() es la función que equivale a la instrucción sleep del PIC.
- Reset()es una función que ejecuta la instrucción reset del PIC.
- ClrWdt() pone a cero el contador del watchdog.
- Swapf (var, dest, access) cambia los 4 bits más bajos por los más altos de la variable (var) almcenándolos en el destino acumulador o registro: (W, F) y con modo de acceso access bank o normal (0, 1).

También hay funciones para rotar con o sin acumulador a izquierda o derecha un registro:

- Rlcf(var, dest, access), Rlncf(var, dest, access), Rrcf(var, dest, access) y Rrncf(var, dest, access).

Como funciones de librería hay que destacar:

- Delay1TCY(void);
- Delay10TCYx(unsigned char ***unit***);
- Delay100TCYx(unsigned char ***unit***);
- Delay1KTCYx(unsigned char ***unit***);
- Delay10KTCYx(unsigned char ***unit***);

Estas funciones introducen retrasos medidos en múltiplos de periodos de instrucción. La primera función introduce un retraso de un periodo de instrucción (equivalente a una instrucción NOP). Siendo un periodo de instrucción TCY=4/(Freq_oscilador). Recordar que en las tarjetas EduMic la frecuencia del oscilador es 4Mhz.

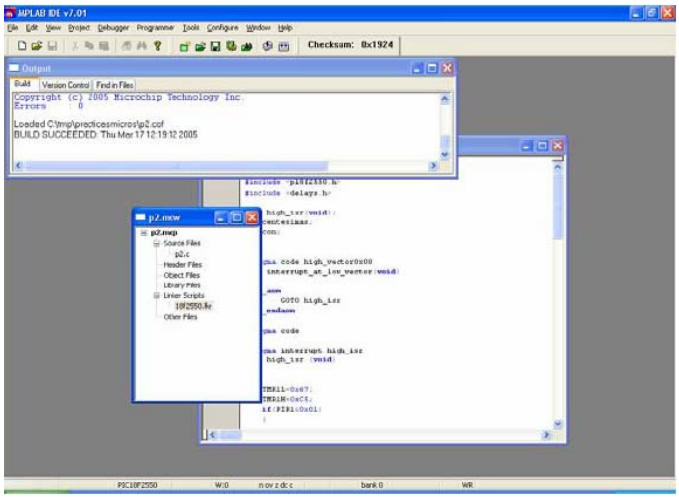
El resto de funciones introducen un número variable de ciclos de retraso dependiendo del parámetro ***unit*** . La función Delay10TCYx produce un retraso de (10TCY)*unit (10 periodos de instrucción por el valor del parámetro unit). El resto de funciones incrementa el número de ciclos por los que se incrementa el valor multiplicado por el parámetro unit en un orden de magnitud: (100TCY)*unit, (1KTCY)*unit y (10KTCY)*unit respectivamente.

Todas ellas están declaradas en el fichero de cabecera ***delays.h***.

Ficheros asociados al proyecto

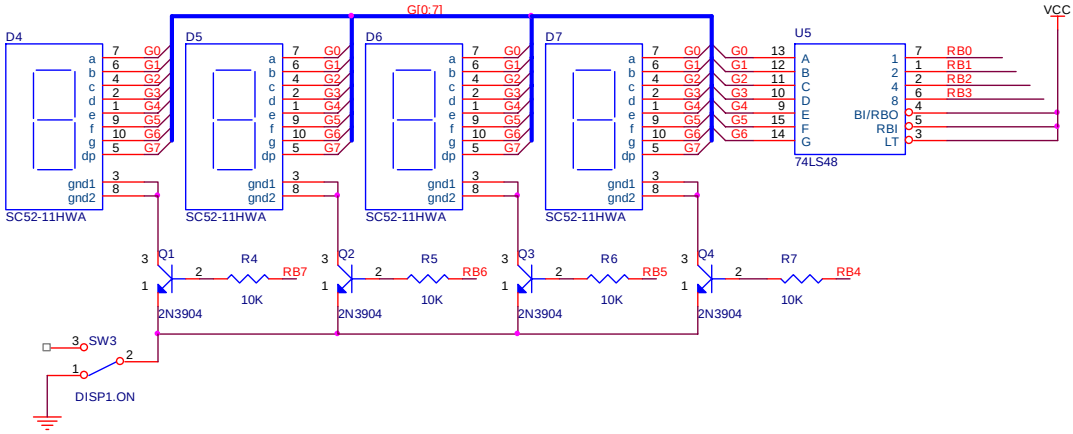
En los ficheros fuente (source files) tendremos un o varios ficheros en c (Ejemplo p2.c). Los ficheros de cabecera (header files) se especifican dentro del fichero fuente y no es necesario añadirlos. Los módulos de arranque y las librerías estándar y del procesador estarían dentro de las clases (object y library files), pero no hace falta especificarlas porque están incluídas en el script del linker (linker script). Si hubiera otros ficheros con módulos o librerías que hay que enlazar juntos habría que añadirlos al proyecto. El fichero de script de linker indica que librerías debe utilizar, la inicialización del micro. Este fichero siempre es el mismo. Para el

microcontrolador 18F2550 el fichero es el 18F2550.lkr, que se encuentra en el directorio “lkr” dentro del directorio principal donde está instalado el compilador C18.



2.2 El display de 4 dígitos

El esquema correspondiente al circuito de este display se encuentra en la siguiente figura:



Para controlar la barra de 4 displays de 7 segmentos se utiliza un único puerto (Puerto B). Este tipo de conexión se utiliza para ahorrar puertos de entrada/salida del micro, ya que en vez de utilizar un puerto por cada display (conectándo por ejemplo: Puerto A con un display, Puerto B con otro display, etc….) se utiliza un único puerto para controlar los 4 displays. El inconveniente de este esquema es que se complica la programación de la aplicación que debe mostrar por cada uno de los 4 displays un dígito diferente, utilizando únicamente el puerto B. Para realizar este objetivo los displays se multiplexan en el tiempo, es decir, durante un pequeño instante de tiempo sólo uno de los displays estará realmente encendido con el valor del dígito que le corresponde. Después de este intervalo el display se apaga y el programa pasa a encender el display siguiente con otro valor correspondiente a su posición dentro de la barra. Si este proceso se realiza de forma sucesiva con todos los displays y de forma muy rápida parecerá que todos los displays están encendidos a la vez.

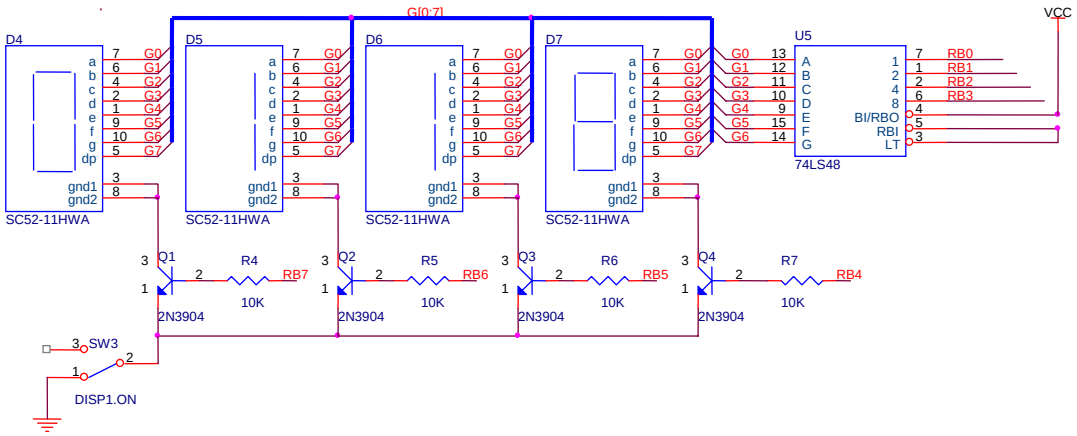
Para llevar a cabo este proceso se utiliza el esquema de conexión mostrado en la figura anterior. El chip a la derecha (74LS48) es un decodificador de binario a 7-segmentos por lo que basta con poner en la parte baja del puerto B el dígito que se quiere sacar por uno de los displays. Los cuatro bits más significativos del puerto B se utilizan para seleccionar el display donde se quiere que aparezca el dígito. Por lo tanto, para sacar un número completo de cuatro dígitos hay que sacar cada vez por un display el dígito correspondiente; si esta operación se realiza de forma rápida (una dígito cada 5 o 10 milisegundos por ejemplo) parecerá que todos los dígitos se encienden a la vez, cuando en realidad se encienden de forma sucesiva. Es posible que si ponemos velocidades más lentas se vea un parpadeo.

3. Trabajos a realizar

En todas las sesiones de laboratorio se pide la realización de unos programas o tareas. **Es imprescindible que se tengan preparados, o al menos esbozados, los programas al principio de cada sesión**. Al inicio de la sesión el profesor comprueba de que efectivamente los programas han sido preparados y se pone la nota correspondiente. Tener los programas preparados al inicio de la sesión es muy importante, de lo contrario, es muy difícil que en una misma sesión se puedan terminar los programas habiendo probado su funcionamiento.

Programa: Escritura de valores en la barra de 4 displays.

En esta práctica se van a utilizar los puertos A y B. El puerto A se utilizará como entradas digitales y el puerto B como salidas para el control de la barra de 4 displays. El programa que se propone debe leer el valor actual de entrada del puerto A y escribir su correspondiente valor decimal en la barra de displays conectada al puerto B (cada dígito del valor decimal en uno de los displays de la barra conectada al puerto B). Por ejemplo si la entrada del puerto A fuera: 0111 0110 su traducción a decimal sería $(0*2^7+1*2^6+1*2^5+1*2^4+0*2^3+1*2^2+1*2^1+0*2^0)=118$ y la barra de 4 displays deberá mostrar:



Para realizar esta operación el programa deberá estar formado por un bucle que constantemente lea el valor actual del puerto A, y escriba los dígitos decimales en las posiciones adecuadas dentro de la barra de 4 displays. Para escribir cada dígito el programa debe primero activar una de las líneas altas del puerto B (bits 7:4) para determinar qué display se enciende y en las líneas más bajas (bits 3:0) se escribe el valor que se desea mostrar. Después de escribir un display hay que esperar de 5-10ms para apagarlo y escribir sobre el siguiente. El retardo de unos 5 milisegundos se debe implementar con la función de librería del compilador *Delay100TCYx (unsigned char **unit**)*. Este proceso se repite para cada display de forma secuencial dentro del bucle que forma el programa.

Recordar que antes de programar el microcontrolador con el software de programación EduMIC, la palabra de configuración config1 hay que modificarla para que su valor sea **config1= '0C00'.**

Apartado opcional: Temporizador de cuenta de minutos y segundos.

Como trabajo opcional se propone que se diseñe un temporizador que cuente minutos y segundos de forma aproximada, utilizando las subrutinas de pérdida de tiempo introducidas en el apartado anterior. El programa utilizará el bucle del apartado anterior para escribir sobre los displays de 7, pero en este caso se escribirán los valores de las dos variables minutos y segundos donde se almacena el tiempo transcurrido desde que arrancó el programa (cada variable se escribirá sobre una pareja de displays 7 segmentos con dos dígitos decimales).