Capítulo 4

Programación.

Índice

Capítulo 4
Programación.

4.1. Introducción.

El DSP TMS320C240 soporta una gran colección de herramientas para la generación de código y la ejecución de este, incluyendo un compilador optimizador de C, un ensamblador, un enlazador (linker), un archivador, un simulador de software, un emulador de gran velocidad, y una placa de desarrollo de software:

1) Compilador de C: recibe código fuente en lenguaje C (ANSI C) y lo traduce en código de lenguaje ensamblador entendible por el TMS320C240. En el paquete del compilador van incluidos un sistema operativo que posee un optimizador, y una utilidad de interlistado. El sistema operativo te permite compilar, ensamblar, y unir módulos fuente automáticamente. La utilidad de interlistado se encarga de mostrar el lenguaje ensamblador generado para cada declaración del archivo fuente C.

2) Ensamblador: traduce archivos de lenguaje fuente a archivos objeto (.obj) basados en lenguaje máquina COFF (common object file format).

3) Library-build utility: es una herramienta para construir tu propia librería runtime-support adapatada.

4) Archivador: permite reunir un grupo de archivos en un solo fichero, llamado librería. Adicionalmente, el archivador permite modificar una librería mediante borrado, reemplazado, extracción, o adición de miembros. Una de las aplicaciones más útiles del archivador es la construcción de una librería de módulos objeto. Para el TMS320C240 en especial tenemos la librería rtsxx.lob que contiene funciones de runtime-support escritas en ANSI estándar y utilidades de compilación.

5) Linker: combina archivos objeto en un solo módulo objeto ejecutable. A la misma vez que crea el modulo ejecutable, lleva acabo la recolocación y resuelve las referencias externas. El linker acepta como entrada ficheros objeto COFF reubicados y librerías objeto.

6) Conversión hexadecimal: esta utilidad convierte un fichero objeto COFF en ASCII-hex. El fichero convertido puede ser descargado en un programador EPROM.

7) Listador absoluto: genera un fichero que puede ser reensamblado para producir un listado de las direcciones absolutas de un fichero objeto.

8) Listador de referencia-cruzada: usa ficheros objeto para producir símbolos de muestra del listado de la referencia-cruzada, así como su definición y sus referencias dentro de los ficheros fuente enlazados.

El principal propósito de este proceso de desarrollo es producir un módulo que pueda ser ejecutado en un sistema TMS320C240. Para ello, todas estas herramientas se interrelacionan jerárquicamente como muestra el siguiente diagrama de flujo:

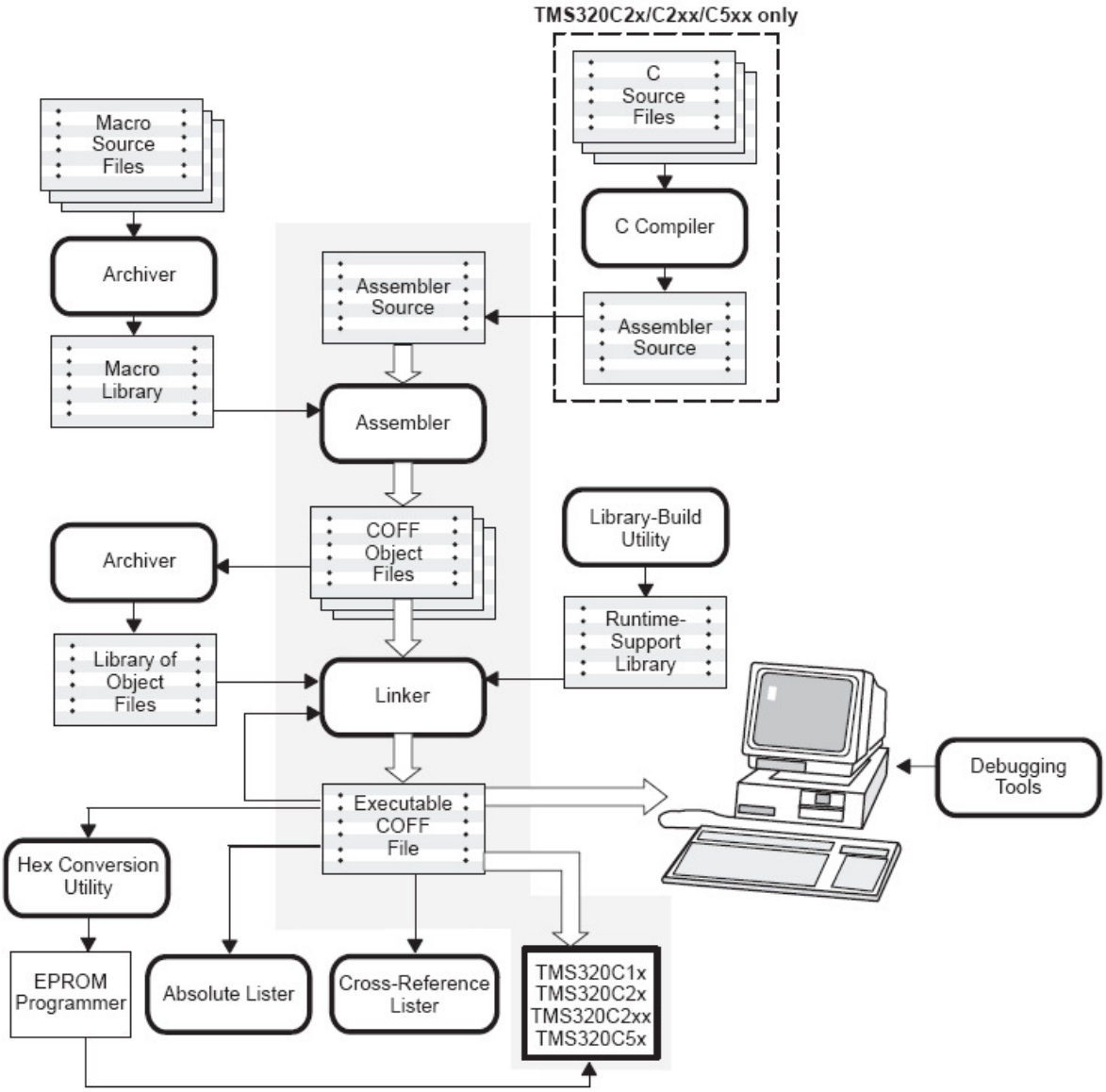


Figura 4.1 – Diagrama de flujo de desarrollo de software para su ejecución en el C240..

En el diagrama observamos una porción sombreada, esta representa el camino más común para el desarrollo de software ejecutable por el TMS320C240.

Todas las anteriores utilidades de desarrollo se encuentran en el programa de desarrollo de software *"Code Composer Studio V2"* de la marca Texas Instruments.

4. 2. Code Composer Studio V2.

Code Composer Studio V2 de Texas Instruments es una potente herramienta de desarrollo y simulación de software para las familias de DSPs de la misma marca. Este entorno nos da la posibilidad de realizar nuestros programas tanto en ensamblador como en lenguaje C, esto es gracias al compilador ANSI C Standard que nos permite convertir programas en lenguaje C a programas en lenguaje ensamblador. Además contiene la capacidad de escribir en un mismo fichero funciones en lenguaje C y otras en lenguaje ensamblador, pudiendo insertar en el código C llamadas a funciones escritas directamente en lenguaje máquina.

4.2.1. Instalación de Code Composer.

Al ejecutar el archivo de instalación nos aparece esta pantalla, en ella elegimos la aplicación de instalación y el programa empezará a descargar en nuestro sistema todos los archivos necesarios.

Antes de ejecutar *Code Composer Studio* por primera vez, tendremos que ejecutar el programa de configuración para nuestro modelo de DSP, que encontramos en el escritorio de Windows (*setup CC C2000).*

Una vez dentro, nos aparecen dos pantallas, una para configuración rápida y otra para configuración detallada o para agregar varias placas de emulación diferentes para varios DSPs. En la pantalla de configuración rápida (figura 4.2) seleccionaremos el modelo de emulador correspondiente al C240 que será el "C2xx XDS510 PP emulator", y finalmente haciendo clic sobre el botón guardar y salir se almacena la configuración y se ejecuta el Code Composer.

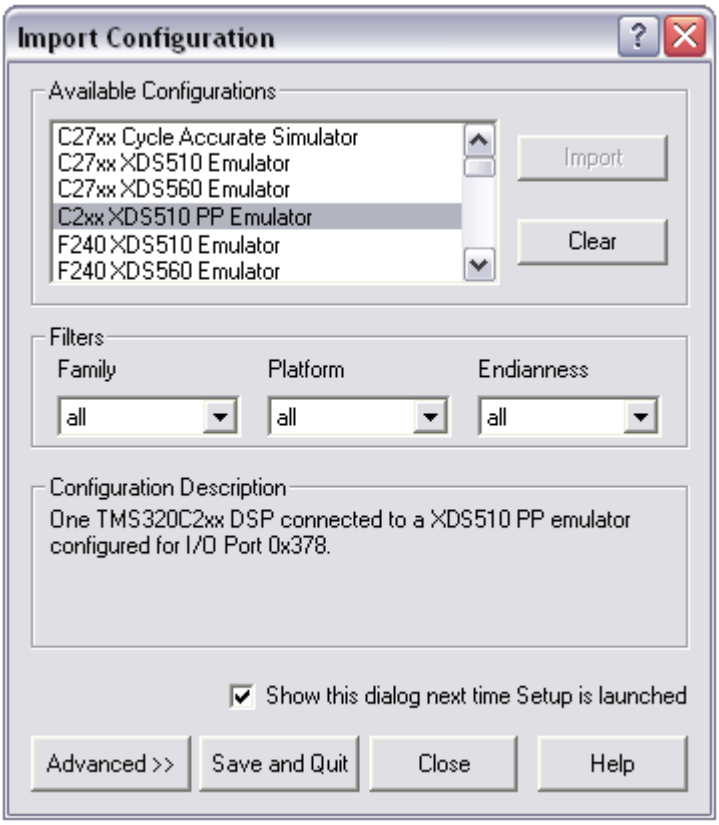


Figura 4.2 – Configuración rápida del Emulador XDS510 PP.

En la pantalla de configuración avanzada del Code Composer (figura 4.3) tendremos que elegir la placa de emulación de nuestro DSP C240 y una vez hecho esto

pinchamos en el menú *File* > *Save*, la configuración quedará guardada y ya podemos pinchar el menú *File>Exit* para salir y ejecutar el Code Composer.

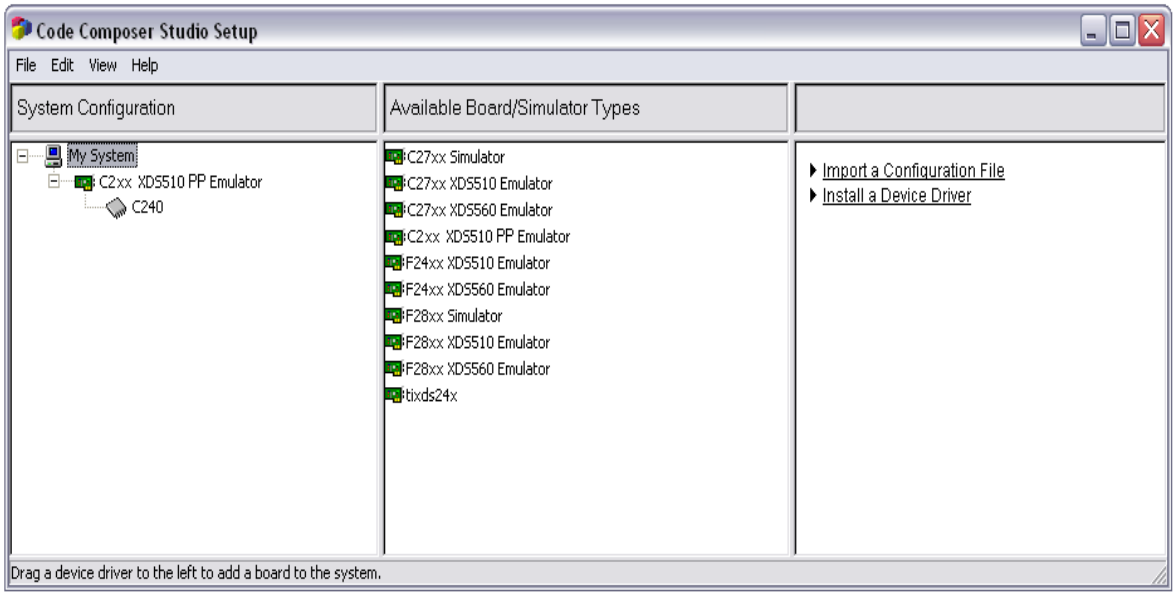


Figura 4.3 – Configuración avanzada del Emulador XDS510 PP.

Si cuando intentamos abrir el Code Composer nos aparece la ventana de la figura 4.4, esto significa que se ha producido un fallo o error debido a una de las siguientes causas:

1) No está conectada la alimentación de la placa.
2) El cable de impresora conectado al emulador, o el cable de emulación JTAG que va a la placa no están bien conectados.
3) Existe un error en la configuración de los parámetros de nuestro emulador.

Figura 4.4 – Mensaje de error al ejecutar Code Composer.

Debemos tener en cuenta que si al aparecer la ventana de error seleccionamos la opción de ignorar el error, entraremos en el entorno Code Composer y podremos compilar nuestros programas, aunque estos siempre nos avisarán de un error al no encontrar la placa, de esta manera podemos realizar una simulación fuera de línea de nuestros programas sin necesidad de tener una placa de emulación conectada al ordenador.

4.2.2. Creación de un programa.

La pantalla principal del entorno de programación y simulación de Code Composer Studio V2 es la siguiente:

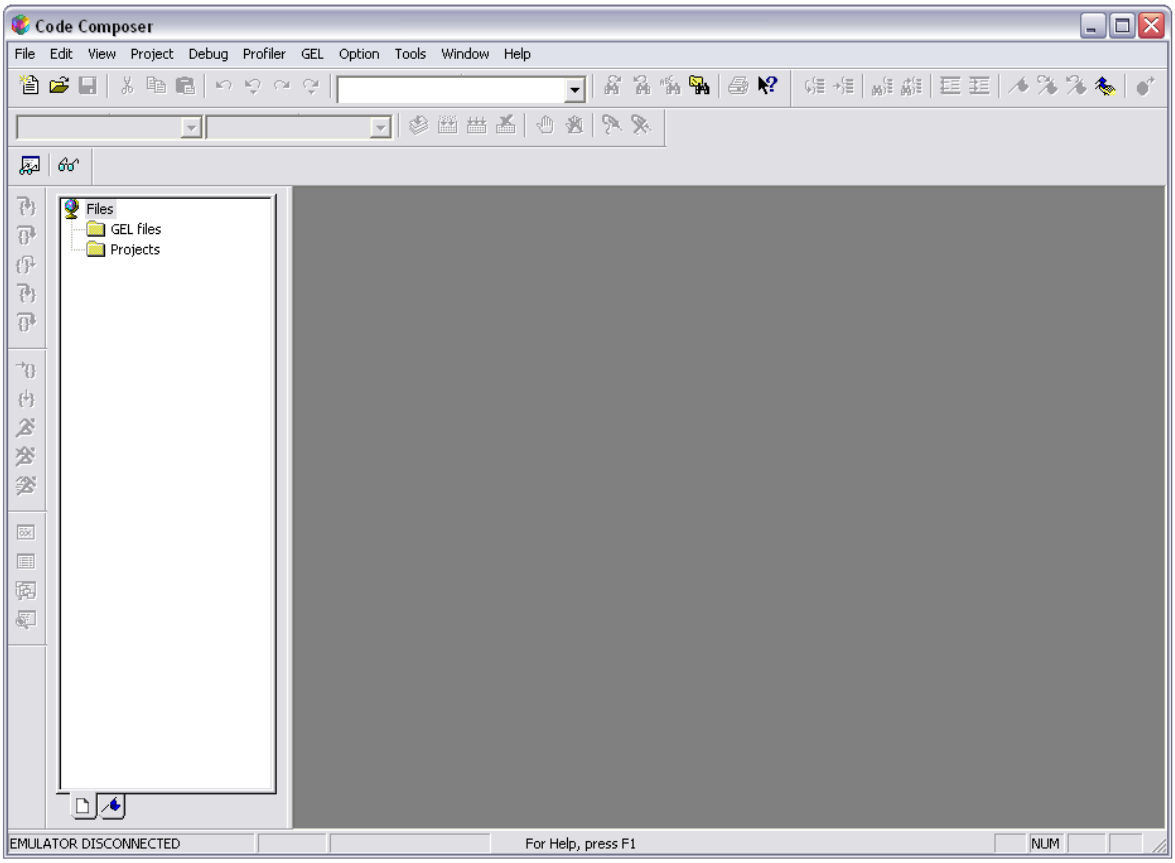


Figura 4.5 - Pantalla principal de Code Composer Studio V2

Para ejecutar nuestro programa tenemos que crear un nuevo proyecto. Para esto nos iremos a la barra de menú a la opción *Project > New,* escribiremos el nombre deseado para nuestro programa y seleccionaremos la opción guardar. Este nos aparecerá en la ventana en blanco de la izquierda, dentro de la carpeta *Projects.* Para su correcta ejecución tendremos que cargar los ficheros fuente necesarios, para esto, sobre el nombre de nuestro proyecto haremos click con el botón derecho y seleccionaremos la función *Add Files*. Primeramente cargaremos nuestro fichero de programa, escrito en C o ensamblador, seguidamente el de configuración de memoria (.cmd), las librerías

necesarias (.lib) así como otros ficheros utilizados. Los últimos ficheros en cargar serán los includes y para ello haremos el mismo procedimiento que antes pero en vez de seleccionar la función *Add Files* seleccionaremos la función *Scan All Dependencies* cargándose los ficheros automáticamente.

Si en vez de crear un nuevo proyecto, lo que deseamos es ejecutar uno ya existente, iremos a la barra menú en la opción *Project* > *Open* y abriremos el proyecto deseado, cargándose automáticamente todos los ficheros necesarios.

Una vez cargado nuestro proyecto, ya sea a partir de la creación de uno nuevo o cargando uno ya existente, lo siguiente a realizar será la compilación de este. Para esto en la barra de menú en la opción *Project* seleccionaremos la opción ***Rebuild All***.

Una vez que hayamos conseguido que no nos de ningún error, hay que recordar que esto no será posible si antes hemos seleccionado la opción omitir en la ventana de error, procederemos a cargar el fichero ejecutable. Para esto en la barra menú en la opción *File* seleccionaremos la opción *Load Program*, recordando que el fichero ejecutable estará en el mismo lugar donde hayamos guardado nuestro proyecto y, además, tendrá el mismo nombre que este pero con la extensión .out.

Hay que decir que los ficheros no podrán ser guardados en carpetas que contengan espacios en su nombre, ya que esto nos provocará en error en la compilación.

4.2.3. Ejecución de un ejemplo.

En este apartado vamos a ejecutar un programa compuesto en lenguaje C y que produce el encendido secuencial de los 8 leds de un array de leds con el que va equipada la placa de emulación del C240.

Para ello, y siguiendo los pasos anteriormente citados, crearemos un nuevo proyecto al que llamaremos “Encendido de leds” y le añadiremos los ficheros necesarios tales como el de la tabla de memoria (leds.cmd) y el del código fuente (leds.c).

Después de esto simplemente pincharemos en el comando *Rebuild All* y el programa queda compilado.

Una vez cargado el programa lo primero que realizaremos será un reset del DSP, para esto en la barra menú en la opción *Debug* seleccionamos la función *Reset DSP*.

Por último ya solo queda ejecutar el programa pinchando en el botón RUN 

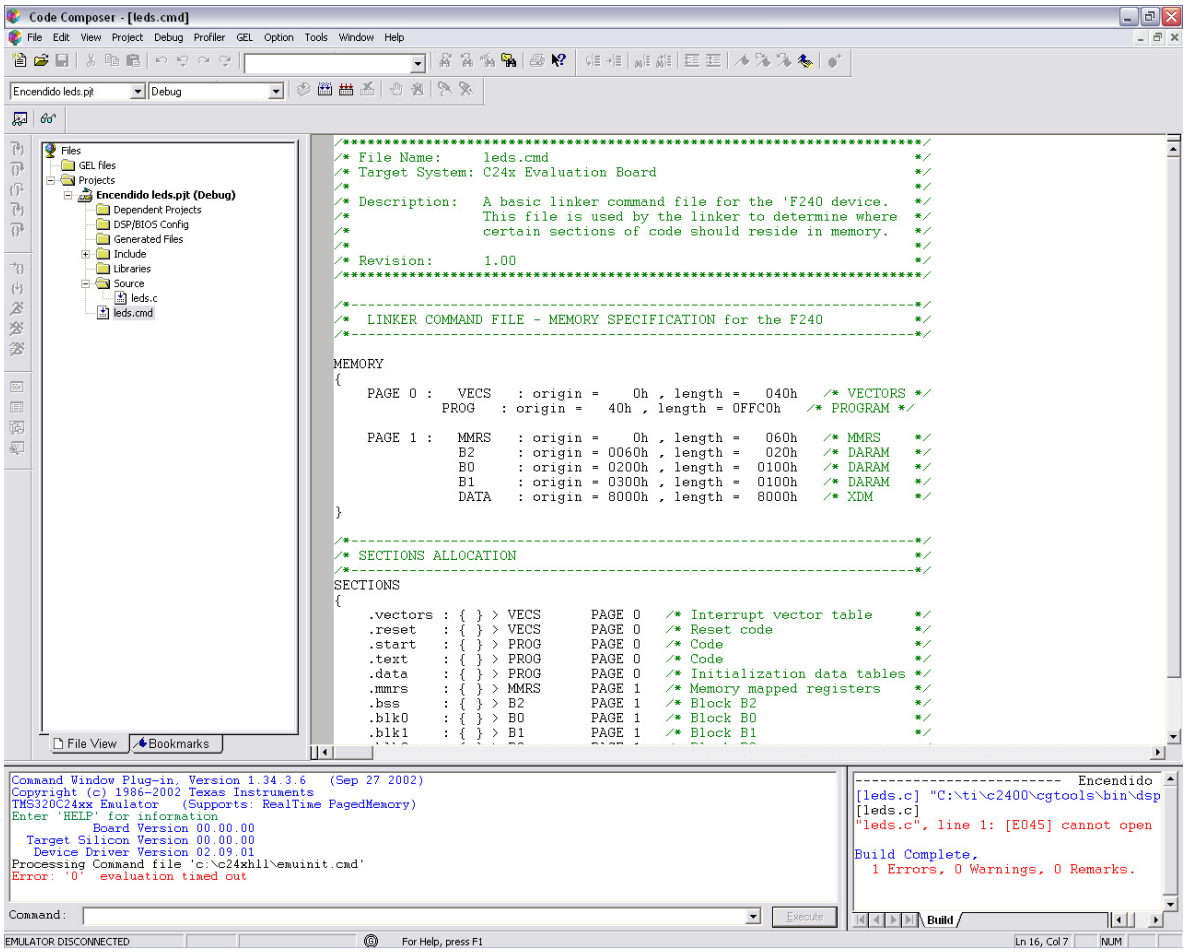


Figura 4.6 – Ejecución de un programa.

4. 3. Programa "LEDS.C"

Este programa está compuesto en lenguaje C y produce el encendido secuencial de los 8 leds de un array de leds con el que va equipada la placa de emulación del C240. Su código fuente es el siguiente:

```
# include "f2407_c.h"
```

Se incluye en la ejecución del programa una librería con información necesaria.

```
ioport unsigned int port0C;
```

El comando ***ioport*** permite el acceso al puerto de entrada del DSP cuya dirección es la 0x0C. Los valores que se encuentren en este registro serán de tipo entero sin signo. En este caso este puerto será solo de salida y se encuentra mapeado con el banco de leds integrado en la placa de evaluación.

```
Unsigned long int i;
```

Definimos la variable global **i** que será necesaria en el desarrollo del programa y que tomará valores de tipo entero sin signo y con tamaño de 4 bits.

```
void main()
{
```

Declaración de la función principal del programa. Establece el punto en el que comienza la ejecución de este.

A continuación se van a declarar los punteros que van a apuntar a los registros de memoria del DSP en los cuales se encuentran los datos de configuración de este. Estos datos son de tipo entero sin signo.

```
    volatile unsigned int *imr=(volatile unsigned int*)0x0004;
```

Direccionamos el puntero **imr** a la dirección **0004h**. Esta dirección corresponde al registro para el enmascarado de interrupciones (IMR).

```
    volatile unsigned int *ifr=(volatile unsigned int*)0x0006;
```

Direccionamos el puntero **ifr** a la dirección **0006h**. Esta dirección corresponde al registro IFR donde se encuentran los indicadores o "flags" que avisan de la ocurrencia de una interrupción.

```
    volatile unsigned int *ckcr0=(volatile unsigned
int*)0x702B;
```

Direccionamos el puntero **ckcr0** a la dirección **702Bh**. Esta dirección corresponde al registro CKCR0 mediante el cual se configura el control de la señal de reloj para el funcionamiento del DSP.

```
volatile unsigned int *ckcr1=(volatile unsigned int*)0x702D;
```

Direccionamos el puntero **ckcr1** a la dirección **702Dh**. Esta dirección corresponde al registro CKCR1 mediante el cual se configura el control de la señal de reloj para el funcionamiento del DSP.

```
volatile unsigned int *syscr=(volatile unsigned int*)0x7018;
```

Direccionamos el puntero **syscr** a la dirección **7018h**. Esta dirección corresponde al registro SYSCR mediante el cual se configura el control del sistema.

```
volatile unsigned int *wdcr=(volatile unsigned int *)0x7029;
```

Direccionamos el puntero **wdcr** a la dirección **7029h**. Esta dirección corresponde al registro WDCR mediante el cual se controla el temporizador del Perro Guardián.

A continuación, en la siguiente parte del código se procede a configurar todos los registros anteriormente direccionados:

```
*imr=0;
```

Le damos el valor **0** al contenido del registro al que apunta el puntero **imr.** Este registro es el que correspondiente a la dirección **0004h** que es el Registro de Enmascarado de Interrupciones. Al darle el valor 0 lo que provocamos es que todos los bits de este registro tomen el valor 0 y de esta forma se enmascaran o deshabilitan todas las interrupciones que afectan al DSP ya que en el caso de este programa ejemplo no serán necesarias.

Bits 15-6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reservado	INT6 máscara	INT5 máscara	INT4 máscara	INT3 máscara	INT2 máscara	INT1 máscara
0	L/E-0	L/E-0	L/E-0	L/E-0	L/E-0	L/E-0

Nota: L = acceso de lectura, E = acceso de escritura, 0- valor después de reinicio.

Figura 4.7. Registro de Enmascarado de Interrupciones (IMR). Dirección 0004h.

```
*ifr=0x3F;
```

Le damos el valor **3Fh** al contenido del registro al que apunta el puntero **ifr.** Este registro es el que correspondiente a la dirección **0006h** que es el Registro de Petición o Aviso de Interrupciones. Al darle el valor 3Fh (111111b) lo que provocamos es que todos los bits de este registro tomen el valor 0 ya que funcionan con lógica negada. De esta manera se anularan todos los avisos de interrupción anteriores. Los bits en las posiciones de la 15 a la 6 permanecen constantemente con valor 0.

Bits 15-6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reservado	INT6 Flag	INT5 Flag	INT4 Flag	INT3 Flag	INT2 Flag	INT1 Flag
0	L/E1B-0	L/ E1B-0	L/ E1B-0	L/ E1B-0	L/ E1B-0	L/ E1B-0

Nota: L = acceso de lectura, E = escribir en este bit 1 para borrarlo, 0- valor después de reinicio.

Figura 4.8 - Registro de Indicadores de Interrupciones (IFR). Dirección 0006h.

```
*ckcr0=0x00C3;
```

Le damos el valor **00C3h** al contenido del registro al que apunta el puntero **ckcr0.** El registro CKCR0 (Registro 0 de Control del Reloj) se utiliza para realizar el control general del módulo generador de pulsos . Algunos de los elementos que controla son el modo reloj, selección del modo de bajo consumo, selección de la predivisión del SYSCLK e indicadores (flags) de estado. Este registro se encuentra en la dirección **702Bh.**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CLKMD(1)	CLKMD(0)	PLLOCK(1)	PLLOCK(0)	PLLPM(1)	PLLPM(0)	Reservado	PLLPS
L/E-x	L/E-x	L-x	L-x	L/E-0	L/E-0	0	L/E-0

Nota: L = acceso de lectura, E = acceso de escritura, -0 valor cero después de reinicio, -x no cambia valor con el reinicio.

Figura 4.9 - Registro 0 de Control del Reloj (CKCR0). Dirección 702Bh.

Bits 7 y 6: CLKMD(1), CLKMD(0). Mediante estos bits se selecciona el modo de operación del módulo de reloj. Las configuraciones según los valores de los bits 7 y 6 son las siguientes:

CLKMD(1:0)	Modo
00	CLKIN / 2
01	CLKIN
10	PLL Inactivo
11	PLL Inactivo

Bits 5 y 4: PLLOCK(1), PLLOCK(2). Estos bits indican cuando el PLL ha introducido el modo seleccionado mediante los bits CLKMD(1:0).

Bits 3 y 2: PLLPM(1), PLLPM(0). Estos bits especifican qué modo de bajo consumo será utilizado. Estos bits toman el valor 0 durante el encendido o reseteo del sistema.

Bit 1. Reservado. Por defecto vale 0.

Bit 0. PLLPS. Este bit especifica cual de dos valores de preescala será elegido para los relojes del sistema. Este bit toma el valor 0 durante el encendido y reseteo del sistema, determinando CPUCLK/4 como frecuencia de reloj (SYSCLK) por defecto para el sistema.

Si este bit vale 0 → frecuencia de SYSCLK = frecuencia CPUCLK / 4
Si este bit vale 1 → frecuencia de SYSCLK = frecuencia CPUCLK / 2

```
*ckcr1=0x00BB;
```

Le damos el valor **00BBh** al contenido del registro al que apunta el puntero **ckcr1,** el cual es el registro CKCR1 (Registro 1 de Control del Reloj). Este registro especifica el factor de multiplicación del PLL (si se encuentra activado) y la frecuencia del reloj de entrada. Este registro se encuentra en la dirección **702Dh.**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CKINF(3)	CKINF(2)	CKINF(1)	CKINF(0)	PLLDIV(2)	PLLFB(2)	PPLLFB(1)	PLLPB(0)
L/E-x	L/E-x	L/E-x	L/E-x	L/E-x	L/E-x	L/E-x	L/E-x

Nota: L = acceso de lectura, E = acceso de escritura, -x no cambia valor con el reinicio.

Figura 4.10 - Registro 1 de Control del Reloj (CKCR1). Dirección 702Dh.

Bits 7 al 4. CKINF(3)-CKINF(0). Estos bits indican la frecuencia de entrada de reloj utilizada. Esto es utilizado por el divisor del WDCLK (Reloj del Watch Dog) para asegurar que se produce una señal de reloj de 16.386/15.625 MHz. A continuación se muestra una tabla con la correspondencia entre el valor de los bits de CKINF y el valor en frecuencia de la señal de reloj.

CKINF(3:0)	Frecuencia (MHz)	CKINF(3:0)	Frecuencia (MHz)
0000	N/A	1000	16
0001	N/A	1001	14
0010	N/A	1010	12
0011	N/A	1011	10
0100	N/A	1100	8
0101	N/A	1101	6
0110	20	1110	4
0111	18	1111	2

Bit 3. PLLDIV(2). Este bit especifica si la entrada al PLL está dividida por 2. Si el bit PLLDIV vale 0, no se divide la entrada del PLL, mientras que si vale 1 entonces la entrada se divide por 2.

Bits 2 al 0. PLLB(2) al PLLFB(0). Estos bits especifican uno de los 6 posibles relaciones de multiplicación del PLL. A continuación se muestra una tabla con estos valores.

PLLFB(2:0)	Relación de multiplicaión del PLL
000	1
001	2
010	3
011	4
100	5
101	9
110	1
111	1

```
*syscr=0x40C0;
```

Le damos el valor **40C0h** al contenido del registro al que apunta el puntero **syscr,** el cual es el registro SYSCR (Registro de Control del Sistema) que se encuentra situado en la dirección de memoria 7018h.

Bit 15	Bit 14	Bits 13-8	Bit 7	Bit 6	5-0
RESET1	RESET0	Reservado	CLKSRC1	CLKSRC0	Reservado
L/E-0	L/E-1		L/E-1*	L/E-1*	

Nota: L= Acceso a lectura, E= Acceso a escritura, -n= valor después de reinicio.
*= No afectado por el reinicio, puesto a 1 al encender.

Bits 15-14. **RESET1, RESET0.** Estos son los bits para el reinicio o "reset" del programa en el procesador. Debe ser establecidos los dos simultaneamente. A continuación se expone una tabla con la combinación de valores que deben tener estos bits para provocar el reinicio.

RESET1	RESET0	Acción Resultante
0	0	Reincio general
0	1	----------
1	0	Reinicio general
1	1	Reinicio general

Bits 13-8. **Reservados.** Pueden tener cualquier valor aleatorio. No se puede escribir sobre ellos.

Bits 7-6. **CLKSRC1, CLKSRC0.** Tienen efecto sólo si está seleccionado como fuente el pin CLKOUT. Estos bits controlan la selección de la función del pin de CLKOUT.

CLKSRC1	CLKSRC0	Función del pin de CLKOUT
0	0	Modo digital de I/O (controlado por los bits del registro de I/O).
0	1	WDCLK: Modo de salida para el temporizador Watchdog.
1	0	SYSCLK: reloj del sistema.
1	1	CPUCLK: Modo de salida para el reloj de la CPU.

Bits 5-0. **Reservados.** Pueden tener cualquier valor aleatorio. No se puede escribir sobre ellos.

wdcr=0x06F

Le damos el valor **06Fh** al contenido del registro al que apunta el puntero **wdcr,** el cual es el registro WDCR (Registro de Control del Temporizador del Watchdog) que se encuentra situado en la dirección de memoria **7029h.** De esta manera conseguimos la deshabilitación del perro guardián.

7	6	5	4	3	2	1	0
WD FLAG	WDDIS	WDCHK2	WDCHK1	WDCHK0	WDPS2	WDPS1	WDPS0
LE-x		LE-0	LE-0	LE-0	LE-0	LE-0	LE-0

Nota: L=acceso de lectura, E=Acceso a escritura, -0=valor después de reset (x-indeterminado).

Bit 7. WD FLAG. Bit de "flag" de WD. Este bit indica si se ha producido un reset del sistema debido a la acción del Watchdog. Este bit se pone a 1 únicamente mediante la acción de un reset por parte del WD.

0 = Indica que el WD no ha realizado ningún reset.
1 = Indica que el WD ha realizado algún reset.

Bit 6. WDDIS. Deshabilitación del Watchdog.

0 = El Watchdog está habilitado.
1 = El Watchdog está deshabilitado.

Bit 5. WDCHK2. Bit 2 de comprobación del Watchdog. Este bit debe ser puesto a 1 cuando se produce la escritura sobre WDCR o se ha llevado a cabo un reset del sistema. El valor de su lectura siempre será 0.

0 = Se ha producido un reset del sistema.
1 = La operación normal continúa si todos los bits de comprobación están correctamente.

Bit 4. WDCHK1. Bit 1 de comprobación del Watchdog. Este bit debe ser puesto a 0 cuando se produce la escritura sobre WDCR o se ha llevado a cabo un reset del sistema. El valor de su lectura siempre será 0.

0 = La operación normal continúa si todos los bits de comprobación están correctamente.
1 = Se ha producido un reset del sistema.

Bit 3. WDCHK0. Bit 0 de comprobación del Watchdog. Este bit debe ser puesto a 1 cuando se produce la escritura sobre WDCR o se ha llevado a cabo un reset del sistema. El valor de su lectura siempre será 0.

0 = Se ha producido un reset del sistema.

1 = La operación normal continúa si todos los bits de comprobación están correctamente.

Bits 2-0. WDPS2-0. Bits de selección de preescala del WD. Estos bits determinan el tiempo de conteo del WD para que se produzca el desbordamiento de este con el consiguiente proceso de reset del sistema.

En la siguiente tabla podemos ver los mínimos valores de tiempo para el desbordamiento del WD según la selección de los bits WDPS2-0:

Bits de selección de preescala de WD			Divisor de WDCLK	16.384 kHz WDCLK		15.625 kHz WDCLK	
WDPS2	WDPS1	WDPS0		Frecuencia (Hz)	Desbordamiento Mínimo	Frecuencia (Hz)	Desbordamiento Mínimo
0	0	X†	1	64	15.63 ms	61.04	16.38 ms
0	1	0	2	32	31.25 ms	30.52	32.77 ms
0	1	1	4	16	62.50 ms	15.26	65.54 ms
1	0	0	8	8	125.00 ms	7.63	131.07 ms
1	0	1	16	4	250.00 ms	3.81	262.14 ms
1	1	0	32	2	500.00 ms	1.91	524.29 ms
1	1	1	64	1	1.0 s	0.95	1.05 s

```
asm("    CLRC OVM");
```

El comando "asm" introduce directamente una línea de lenguaje ensamblador dentro del código C. En este caso pone a cero el bit del OVM, o sea, deshabilita el modo desbordamiento, esto se podría haber hecho directamente con el comando ROVM.

Con la ejecución del código anterior ya quedan inicializados todos los registros de configuración del DSP. A continuación se desarrollará el código que ejerce la operación de encendido secuencial del banco de LEDs que va incorporado en la placa de evaluación del C240:

```
/************************ APLICACION ***************************/

/**** LED 1 ****/
```

En este bloque de programa se realiza el encendido secuencial de los LEDs de forma que sólo se encenderá y desplazará un led a lo largo de la parrilla de LEDs.

```
port0C=0x0;
```

Con este comando le pasamos el valor **0h** a la variable **port0C,** la cual se encuentra instalada en la dirección de memoria de entrada **0x0Ch** que va mapeada con el banco de memoria de los leds. Con esto lo que conseguimos es que todos los LEDs permanezcan apagados.

La secuencia de encendido resultante es: 

Nota: El compilador de C del DSP C240 sólo admite como valores hexadecimales para introducir en los registros de memoria.

```
port0C=0x1;
```

Con este comando le pasamos el valor **1h** a la variable **port0C.** De esta manera se producirá el encendido del primer LED de la parrilla de LEDs, el cual se corresponde con el bit menos significativo del registro de memoria mapeado de los LEDs (0x0C).

La secuencia de encedido resultante es: 

```
for ( i=0; i<40000; i++ );
```

Con este bucle **for** lo que hacemos es incrementar la variable **i** hasta que alcance el valor **40000,** de esta manera aprovechando el tiempo que tarda la función en realizar esto, conseguimos un retardo de 0.8 segundos. Así conseguimos que regular la velocidad con la cual se producirá el encendido secuencial de los LEDs.

El código de programa restante que se relata a continuación es una repetición de los comandos **for** (para temporizar el encendido de los LEDs) y el paso de un valor a la variable **port0C** (que determina qué LEDs se encienden en cada momento).

```
port0C=0x2;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x4;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x8;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x10;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x20;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
```

```
port0C=0x40;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x80;
```

La secuencia de encendido resultante es: 

repite:

Esta etiqueta sirve para que la ejecución del programa pueda volver directamente a este punto en posteriores llamadas.

/** LED 2 ****/**

Este es el bloque en el cual se produce el encendido secuencial de una serie de 2 LEDs.

```
for ( i=0; i<40000; i++ );
port0C=0x1;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x3;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x6;
```

La secuencia de encendido resultante es: 

```
 for ( i=0; i<40000; i++ );
port0C=0xC;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x18;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x30;
```

La secuencia de encendido resultante es: 

```
        for ( i=0; i<40000; i++ );
        port0C=0x60;
```

La secuencia de encendido resultante es: 

```
        for ( i=0; i<40000; i++ );
        port0C=0xC0;
```

La secuencia de encendido resultante es: 

```
        for ( i=0; i<40000; i++ );
        port0C=0x80;
```

La secuencia de encendido resultante es: 

```
/**** LED 3 ****/
```

Este es el bloque en el cual se produce el encendido secuencial de una serie de 3 LEDs.

```
        for ( i=0; i<40000; i++ );
        port0C=0x1;
```

La secuencia de encendido resultante es: 

```
        for ( i=0; i<40000; i++ );
        port0C=0x3;
```

La secuencia de encendido resultante es: 

```
      for ( i=0; i<40000; i++ );
        port0C=0x7;
```

La secuencia de encendido resultante es: 

```
      for ( i=0; i<40000; i++ );
        port0C=0xE;
```

La secuencia de encendido resultante es: 

```
      for ( i=0; i<40000; i++ );
        port0C=0x1C;
```

La secuencia de encendido resultante es: 

```
      for ( i=0; i<40000; i++ );
        port0C=0x38;
```

La secuencia de encendido resultante es: 

```
      for ( i=0; i<40000; i++ );
        port0C=0x70;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xE0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xC0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x80;
```

La secuencia de encendido resultante es: 

```
/**** LED 4 ****/
```

Este es el bloque en el cual se produce el encendido secuencial de una serie de 4 LEDs.

```
for ( i=0; i<40000; i++ );
  port0C=0x1;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x3;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x7;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x1E;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x3C;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x78;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xE0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xC0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x80;
```

La secuencia de encendido resultante es: 

```
/**** LED 5 ****/
```

Este es el bloque en el cual se produce el encendido secuencial de una serie de 5 LEDs.

```
for ( i=0; i<40000; i++ );
  port0C=0x1;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x3;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x7;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x1F;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x3E;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x7C;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF8;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xE0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0xC0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x80;
```

La secuencia de encendido resultante es: 

```
/**** LED 6 ****/
```

Este es el bloque en el cual se produce el encendido secuencial de una serie de 6 LEDs.

```
for ( i=0; i<40000; i++ );
  port0C=0x1;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x3;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x7;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x1F;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x3F;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x7E;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xFC;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF8;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF0;
```

La secuencia de encendido resultante es: 

```
  for ( i=0; i<40000; i++ );
  port0C=0xE0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
```

```
  port0C=0xC0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x80;
```

La secuencia de encendido resultante es: 

/** LED 7 ****/**

Este es el bloque en el cual se produce el encendido secuencial de una serie de 7 LEDs.

```
for ( i=0; i<40000; i++ );
  port0C=0x1;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x3;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x7;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x1F;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x3F;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x7F;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xFE;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xFC;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF8;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xE0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xC0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x80;
```

La secuencia de encendido resultante es: 

/** LED 8 ****/**

Este es el bloque en el cual se produce el encendido secuencial de una serie de 8 LEDs.

```
for ( i=0; i<40000; i++ );
  port0C=0x1;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x3;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x7;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x1F;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x3F;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x7F;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xFF;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xFE;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xFC;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF8;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF0;
```

La secuencia de encendido resultante es: 

```
   for ( i=0; i<40000; i++ );
port0C=0xE0;
```

La secuencia de encendido resultante es:

```
for ( i=0; i<40000; i++ );
  port0C=0xC0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x80;
```

La secuencia de encendido resultante es: 

```
/************final incremento*************************/

/**** LED 7 VUELTA ****/

for ( i=0; i<40000; i++ );
  port0C=0x1;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x3;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x7;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x1F;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x3F;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x7F;
```

La secuencia de encendido resultante es:

```
for ( i=0; i<40000; i++ );
  port0C=0xFE;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xFC;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF8;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF0;
```

La secuencia de encendido resultante es: 

```
  for ( i=0; i<40000; i++ );
  port0C=0xE0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xC0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x80;
```

La secuencia de encendido resultante es: 

```
/**** LED 6 VUELTA ****/
```

Este es el bloque en el cual se produce el encendido secuencial de una serie de 6 LEDs.

```
for ( i=0; i<40000; i++ );
  port0C=0x1;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x3;
```

La secuencia de encendido resultante es:

```
for ( i=0; i<40000; i++ );
  port0C=0x7;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x1F;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x3F;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x7E;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xFC;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF8;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xE0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xC0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x80;
```

La secuencia de encendido resultante es: 

```
/**** LED 5 VUELTA ****/
```

Este es el bloque en el cual se produce el encendido secuencial de una serie de 5 LEDs.

```
for ( i=0; i<40000; i++ );
  port0C=0x1;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x3;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x7;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x1F;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x3E;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x7C;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF8;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xE0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0xC0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x80;
```

La secuencia de encendido resultante es: 

```
/**** LED 4 VUELTA ****/
```

Este es el bloque en el cual se produce el encendido secuencial de una serie de 3 LEDs.

```
for ( i=0; i<40000; i++ );
  port0C=0x1;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x3;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x7;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x1E;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x3C;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x78;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xF0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xE0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xC0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x80;
```

La secuencia de encendido resultante es: 

```
/**** LED 3 VUELTA ****/
```

Este es el bloque en el cual se produce el encendido secuencial de una serie de 3 LEDs.

```
for ( i=0; i<40000; i++ );
port0C=0x1;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x3;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x7;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
```

```
port0C=0xE;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x1C;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x38;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x70;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xE0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0xC0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
  port0C=0x80;
```

La secuencia de encendido resultante es: 

```
/**** LED 2 VUELTA ****/
```

Este es el bloque en el cual se produce el encendido secuencial de una serie de 2 LEDs.

```
for ( i=0; i<40000; i++ );
port0C=0x1;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x3;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x6;
```

La secuencia de encendido resultante es:

```
 for ( i=0; i<40000; i++ );
port0C=0xC;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x18;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x30;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x60;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0xC0;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x80;
```

La secuencia de encendido resultante es: 

```
/**** LED 1 VUELTA ****/
```

En este bloque de programa se realiza el encendido secuencial de los LEDs de forma que sólo se encenderá y desplazará un led a lo largo de la parrilla de LEDs.

```
for ( i=0; i<40000; i++ );
port0C=0x1;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x2;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x4;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x8;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x10;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x20;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x40;
```

La secuencia de encendido resultante es: 

```
for ( i=0; i<40000; i++ );
port0C=0x80;
```

La secuencia de encendido resultante es: 

```
goto repite;

}
```

4. 4. Programa "LEDS.ASM"

Este programa está compuesto en lenguaje ensamblador y produce el encendido secuencial de los 8 leds de un array de leds con el que va equipada la placa de emulación del C240. Su código fuente es el siguiente:

```
1.    .include    f240regs.h        ;Agrega una librería necesaria.

;---------------------------------------------------------------------
; Declaración de los Registros mapeados de I/O del EVM.
;---------------------------------------------------------------------

2. DAC0           .set    0000h          ;Registro del canal 0 del
DAC.
3. DAC1           .set    0001h          ;Registro del canal 1 del
DAC.
4. DAC2           .set    0002h          ;Registro del canal 2 del
DAC.
5. DAC3           .set    0003h          ;Registro del canal 3 del
DAC.
6. DAC_UPDATE     .set    0004h          ;Registro de actualización
del                                      DAC.

7. SWITCHES       .set    0008h          ;Registro de los
interruptores                                        DIP.

8. LEDS           .set    000Ch          ;Registro de los LEDs.
9. temp           .set    8000h          ;Registro temporal.

                                         ;El comando .set define el
valor                                          de una variable, osea,
su                                             dirección.

;---------------------------------------------------------------------
; Declaración de variables para la memoria RAM interna
;---------------------------------------------------------------------

10.         .bss  GPR0,1            ;Registro de propósito general.
11.         .bss  LED_STATUS,1      ;Registro de estado de los LEDs
12.         .bss  RPT_NUM,1         ;Valor de repetición utilizado en
la                                  subrutina mS_DELAY.

13.         .bss  mSEC,1            ;Valor de retardo utilizado por la
                                          subrutina mS_DELAY

                                    ;la sección .bss reserva memoria
para                                variables no inicializadas.

;---------------------------------------------------------------------
; Declaración de las direcciones de los vectores
;---------------------------------------------------------------------

14.         .sect ".vectors"        ;Creación de la sección "vectores".

15. RSVECT        B    START        ; Vector de reinicio.
```

```
16. INT1          B    PHANTOM      ; Nivel de interrupción 1.
17. INT2          B    PHANTOM      ; Nivel de interrupción 2.
18. INT3          B    PHANTOM      ; Nivel de interrupción 3.
19.   INT4        B    PHANTOM      ; Nivel de interrupción 4.
20.   INT5        B    PHANTOM      ; Nivel de interrupción 5.
21.   INT6        B    PHANTOM      ; Nivel de interrupción 6.
22.   RESERVED    B    PHANTOM      ; Reservado.
23.   SW_INT8     B    PHANTOM      ; Interrupción de usuario S/W.
24.   SW_INT9     B    PHANTOM      ; Interrupción de usuario S/W.
25.   SW_INT10    B    PHANTOM      ; Interrupción de usuario S/W.
26.   SW_INT11    B    PHANTOM      ; Interrupción de usuario S/W.
27.   SW_INT12    B    PHANTOM      ; Interrupción de usuario S/W.
28.   SW_INT13    B    PHANTOM      ; Interrupción de usuario S/W.
29.   SW_INT14    B    PHANTOM      ; Interrupción de usuario S/W.
30.   SW_INT15    B    PHANTOM      ; Interrupción de usuario S/W.
31.   SW_INT16    B    PHANTOM      ; Interrupción de usuario S/W.
32.   TRAP        B    PHANTOM      ; Vector de trampa
33.   NMINT       B    PHANTOM      ; Interrupción no enmascarable.
34.   EMU_TRAP    B    PHANTOM      ; Trampa del emulador.
35.   SW_INT20    B    PHANTOM      ; Interrupción de usuario S/W.
36.   SW_INT21    B    PHANTOM      ; Interrupción de usuario S/W.
37.   SW_INT22    B    PHANTOM      ; Interrupción de usuario S/W.
38.   SW_INT23    B    PHANTOM      ; Interrupción de usuario S/W.

                                    ;Todas las anteriores
interrupciones,                                  en caso de producirse,
producen la                                           llamada a la
subrutina PHANTOM, la                                       cual
resetea el Watch dog y las anula.


;=====================================================================
; Código principal:
;
; Desplazamiento de 1 bit hacia la izquierda del contenido del
registro ;LED_STATUS y escritura del nuevo valor sobre el registro
mapeado de I/O ;de los leds encendiendo un led más a la derecha.
;
;=====================================================================


39.         .text                   ;Aquí comienza el código del
                                    programa.

40.         NOP                     ;Introduce un retardo de un
                                    ciclo.

41.   START:                        ;Etiqueta o nombre de la
                                    rutina principal.

42.         SETC  INTM              ;Pone a 1 el bit INTM lo que
                                    produce el deshabilitado de
                                    todas las interrupciones.

43.         SPLK  #0000h,IMR        ;Graba el valor 0h en el
                                    registro de interrupciones
IMR,                                enmascarando todas las
                                    interrupciones del micro.
```

```
44.         LACC  IFR                 ;Carga el valor del registro
IFR                                   en el acumulador. Lectura de
los                                   flags de interrupciones.

45.         SACL  IFR                 ;Graba    el    valor    del
acumulador                            en el registro IFR, borrando
                                      todos los flags de las
                                      interrupciones.

46.         CLRC  SXM                 ;Borra el bit de signo de
                                      extensión (SXM). Las
                                      interrupciones   se   activan
                                      inmediatamente   después   de
                                      ejecutar CLRC.

47.         CLRC  OVM                 ;Borra el bit OVM, por tanto
se                                    reinicia        el        modo
desbordamiento                        (Overflow).


48.         CLRC  CNF                 ;Borra el bit CNF. Establece
el                                    bloque  B0  como  memoria  de
datos.

49.         LDP   #00E0h              ;Carga los 9 bits LSB de la
                                      dirección   00E0h   en   el
registro                              DP (puntero memoria de datos)
y                                     los une con otro registro de
7                                     bits para formar una palabra
de                                    16 bits. Reserva para datos
las                                   direcciones 7000h-707Fh.


50.         SPLK  #00BBh,CKCR1        ;Graba el valor literal 00BBh
en                                    el registro CKCR1. Con esto
se                                    establece que

      :CLKIN(OSC)=10MHz,CPUCLK=20MHz.

51.         SPLK  #00C3h,CKCR0        ;Graba el literal 00C3h en el
                                      registro   CKCR0.   Establece
                                      CKLMD=PLL y habita el divisor
de                                    frecuencia : SYSCLK=CPUCLK/2.


52.         SPLK  #40C0h,SYSCR        ;Graba el literal 40C0h en el
                                      registro   SYSCR.   Establece
                                      CLKOUT=CPUCLK.

53.         SPLK  #006Fh, WDCR        ;Graba el literal 006Fh en el
                                      registro WDCR. Deshabilita el
                                      Watch Dog si VCCP=5V (JP5 en
                                      posición 2-3).

54.         KICK_DOG                  ;Reinicia el Watch Dog.

55.         SPLK  #0h,GPR0            ;Graba el literal 0h en el
                                      registro GPR0. Establece 0
                                      espacios de espera en el
                                      generador de espacios de
espera                                para el espacio de programa,
```

```
el                                      espacio de datos y el espacio
de                                      I/O.

56.         OUT GPR0,WSGR               ;Copia el contenido de GPR0
en                                      el registro WSGR. Configura
cero                                    estados de espera para el
                                        espacio de programa, el
espacio                                 de datos y el espacio de I/O.


57.         SPLK  #0b,LED_STATUS        ;Graba el literal 0b en el
                                        registro LED_STATUS (registro
de                                      los leds). Se ponen a 0 todos
                                        los bits registro LEDS.

58.         OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS (valor 0b)en el
                                        puerto de E/S. Todos los leds
                                        permanencen apagados.

59.         SPLK    #1000,mSEC          ;Graba el literal 1000 en el
                                        registro mSEC. Establece un
                                        retardo de 100mseg.


60.         SPLK  #1h,LED_STATUS        ;Graba el literal 1h en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 el LSB
del                                     registro LEDS.

61.         OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S.                                    Se produce el encendido de
los                                     leds.

62.         LAR     AR0,#7h             ;Carga el literal 7h en el
                                        registro auxiliar AR0. Con
esto                                    creamos un contador que
repite 7                                veces el bloque siguiente.
63.

64.   PASO1:                            ;Etiqueta o nombre de la
                                        subrutina.

65.         CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

66.         KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

67.         MAR   *,AR0                 ;Modifica el registro
auxiliar.                               Establece ARP = AR0.

68.         LACC  LED_STATUS            ;Carga el acumulador con el
                                        contenido de LED_STATUS.

69.         SFL                         ;Desplaza el contenido del
                                        acumulador 1 bit hacia la
                                        izquierda.
```

```
70.         SACL  LED_STATUS          ;Copia los 16 LSB del
acumulador                            en el registro LED_STATUS.

71.         OUT   LED_STATUS,LEDS     ;Copia el contenido de
                                      LED_STATUS en el puerto de
E/S                                   de los leds. Se produce el
                                      desplazamiento hacia la
derecha                               del led/s encendido/s.

72.         BANZ    PASO1             ;Si  AR0 es distinto de 0, el
                                      programa llama a la subrutina
                                      PASO1 y decrementa AR0 en una
                                      unidad, si no pasa a la
                                      instrucción siguiente.

73.         CALL  mS_DELAY            ;Llama a la subrutina
mS_DELAY                              la cual produce un retardo de
                                      100mseg.

74.         KICK_DOG                  ;Reinicia el WD si este está
                                      activo.

75.         SPLK  #1h,LED_STATUS      ;Graba el literal 1h en el
                                      registro LED_STATUS (registro
de                                    los leds). Pone a 1 el LSB
del                                   registro LEDS.

76.         OUT   LED_STATUS,LEDS     ;Copia el contenido de
                                      LED_STATUS en el puerto de
E/S.                                  Se produce el encendido de
los                                   leds.

77.         CALL  mS_DELAY            ;Llama a la subrutina
mS_DELAY                              la cual produce un retardo de
                                      100mseg.

78.         KICK_DOG                  ;Reinicia el WD si este está
                                      activo.

79.         SPLK    #11b,LED_STATUS    ;Graba el literal 11b en el
                                      registro LED_STATUS (registro
de                                    los leds). Pone a 1 los dos
                                      primeros bits del registro
LEDS.

80.         OUT    LED_STATUS,LEDS    ;Copia el contenido de
                                      LED_STATUS en el puerto de
E/S.                                  Se produce el encendido de
los                                   leds.

81.         LAR     AR0,#7h           ;Carga el literal 7h en el
                                      registro auxiliar AR0. Con
esto                                  creamos un contador que
repite 7                              veces el bloque siguiente.
82.

83.   PASO2:                          ;Etiqueta o nombre de la
                                      subrutina.
```

```
84.         CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

85.         KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

86.         MAR   *,AR0                 ;Modifica el registro
auxiliar.                               Establece ARP = AR0.

87.         LACC  LED_STATUS            ;Carga el acumulador con el
                                        contenido de LED_STATUS.

88.         SFL                         ;Desplaza el contenido del
                                        acumulador 1 bit hacia la
                                        izquierda.

89.         SACL  LED_STATUS            ;Copia los 16 LSB del
acumulador                              en el registro LED_STATUS.

90.         OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

91.         BANZ    PASO2               ;Si  AR0 es distinto de 0, el
                                        programa llama a la subrutina
                                        PASO2 y decrementa AR0 en una
                                        unidad, si no pasa a la
                                        instrucción siguiente.

92.         CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

93.         KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

94.         SPLK  #1h,LED_STATUS        ;Graba el literal 1h en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 el LSB
del                                     registro LEDS.

95.         OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S.                                    Se produce el encendido de
los                                     leds.

96.         CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

97.         KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

98.         SPLK    #11b,LED_STATUS      ;Graba el literal 11b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los dos
                                        primeros bits del registro
LEDS.
```

```
99.         OUT    LED_STATUS,LEDS      ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S.                                    Se produce el encendido de
los                                     leds.

100.        CALL  mS_DELAY               ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

101.        KICK_DOG                     ;Reinicia el WD si este está
                                        activo.

102.        SPLK    #111b,LED_STATUS    ;Graba el literal 111b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los tres
                                        primeros bits del registro
LEDS.

103.        OUT   LED_STATUS,LEDS        ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

104.        LAR     AR0,#7h              ;Carga el literal 7h en el
                                        registro auxiliar AR0. Con
esto                                    creamos un contador que
repite 7                                veces el bloque siguiente.


105.  PASO3:                             ;Etiqueta o nombre de la
                                        subrutina.

106.        CALL  mS_DELAY               ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

107.        KICK_DOG                     ;Reinicia el WD si este está
                                        activo.

108.        MAR   *,AR0                  ;Modifica el registro
auxiliar.                               Establece ARP = AR0.

109.        LACC  LED_STATUS             ;Carga el acumulador con el
                                        contenido de LED_STATUS.

110.        SFL                          ;Desplaza el contenido del
                                        acumulador 1 bit hacia la
                                        izquierda.

111.        SACL  LED_STATUS             ;Copia los 16 LSB del
acumulador                              en el registro LED_STATUS.

112.        OUT   LED_STATUS,LEDS        ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.
```

```
113.        BANZ    PASO3               ;Si  AR0 es distinto de 0, el
                                        programa llama a la subrutina
                                        PASO3 y decrementa AR0 en una
                                        unidad, si no pasa a la
                                        instrucción siguiente.

114.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

115.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

116.        SPLK  #1h,LED_STATUS        ;Graba el literal 1h en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 el LSB
del                                     registro LEDS.

117.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S.                                    Se produce el encendido de
los                                     leds.

118.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

119.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

120.        SPLK    #11b,LED_STATUS      ;Graba el literal 11b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los dos
                                        primeros bits del registro
LEDS.

121.        OUT    LED_STATUS,LEDS      ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S.                                    Se produce el encendido de
los                                     leds.

122.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

123.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

124.        SPLK   #111b,LED_STATUS     ;Graba el literal 111b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los tres
                                        primeros bits del registro
LEDS.

125.        OUT    LED_STATUS,LEDS      ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S.                                    Se produce el encendido de
los                                     leds.
```

```
126.        CALL mS_DELAY              ;Llama a la subrutina
mS_DELAY                               la cual produce un retardo de
                                       100mseg.

127.        KICK_DOG                   ;Reinicia el WD si este está
                                       activo.

128.        SPLK   #1111b,LED_STATUS   ;Graba el literal 1111b en el
                                       registro LED_STATUS (registro
de                                     los leds). Pone a 1 los
cuatro                                 primeros bits del registro
LEDS.

129.        OUT   LED_STATUS,LEDS      ;Copia el contenido de
                                       LED_STATUS en el puerto de
E/S                                    de los leds. Se produce el
                                       desplazamiento hacia la
derecha                                del led/s encendido/s.

130.        LAR     AR0,#7h            ;Carga el literal 7h en el
                                       registro auxiliar AR0. Con
esto                                   creamos un contador que
repite 7                               veces el bloque siguiente.

131.  PASO4:                           ;Etiqueta o nombre de la
                                       subrutina.

132.        CALL   mS_DELAY            ;Llama a la subrutina
mS_DELAY                               la cual produce un retardo de
                                       100mseg.

133.        KICK_DOG                   ;Reinicia el WD si este está
                                       activo.

134.        MAR    *,AR0               ;Modifica el registro
auxiliar.                              Establece ARP = AR0.

135.        LACC   LED_STATUS          ;Carga el acumulador con el
                                       contenido de LED_STATUS.

136.        SFL                        ;Desplaza el contenido del
                                       acumulador 1 bit hacia la
                                       izquierda.

137.        SACL   LED_STATUS          ;Copia los 16 LSB del
acumulador                             en el registro LED_STATUS.

138.        OUT    LED_STATUS,LEDS     ;Copia el contenido de
                                       LED_STATUS en el puerto de
E/S                                    de los leds. Se produce el
                                       desplazamiento hacia la
derecha                                del led encendido.

139.        BANZ    PASO4              ;Si  AR0 es distinto de 0, el
                                       programa llama a la subrutina
                                       PASO4 y decrementa AR0 en una
                                       unidad, si no pasa a la
                                       instrucción siguiente.
```

```
140.        CALL    mS_DELAY            ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

141.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

142.        SPLK    #1h,LED_STATUS      ;Graba el literal 1h en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 el primer
                                        bit del registro LEDS.

143.        OUT     LED_STATUS,LEDS     ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led encendido.

144.        CALL    mS_DELAY            ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

145.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

146.        SPLK    #11b,LED_STATUS     ;Graba el literal 11b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los dos
                                        primeros bits del registro
LEDS.

147.        OUT     LED_STATUS,LEDS     ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led encendido.

148.        CALL    mS_DELAY            ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

149.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

150.        SPLK    #111b,LED_STATUS    ;Graba el literal 111b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los tres
                                        primeros bits del registro
LEDS.

151.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

152.        CALL    mS_DELAY            ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.
```

```
153.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

154.        SPLK    #1111b,LED_STATUS   ;Graba el literal 1111b en el
                                        registro LED_STATUS (registro
de                                      los  leds).  Pone  a  1  los
cuatro                                  primeros bits del registro
LEDS.

155.        OUT     LED_STATUS,LEDS     ;Copia el contenido de
                                        LED_STATUS (valor 11b)en el
                                        puerto de E/S. Se produce el
                                        encendido de los leds.

156.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

157.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

158.        SPLK    #11111b,LED_STATUS  ;Graba el literal 11111b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los cinco
                                        primeros bits del registro
LEDS.

159.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

160.        LAR     AR0,#7h             ;Carga el literal 7h en el
                                        registro auxiliar AR0. Con
esto                                    creamos un contador que
repite 7                                veces el bloque siguiente.

161.  PASO5:                            ;Etiqueta o nombre de la
                                        subrutina.

162.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

163.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

164.        MAR   *,AR0                 ;Modifica el registro
auxiliar.                               Establece ARP = AR0.

165.        LACC  LED_STATUS            ;Carga el acumulador con el
                                        contenido de LED_STATUS.

166.        SFL                         ;Desplaza el contenido del
                                        acumulador 1 bit hacia la
                                        izquierda.

167.        SACL  LED_STATUS            ;Copia los 16 LSB del
acumulador                              en el registro LED_STATUS.
```

```
168.        OUT   LED_STATUS,LEDS      ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

169.        BANZ    PASO5               ;Si  AR0 es distinto de 0, el
                                        programa llama a la subrutina
                                        PASO5 y decrementa AR0 en una
                                        unidad, si no pasa a la
                                        instrucción siguiente.

170.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

171.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

172.        SPLK  #1h,LED_STATUS        ;Graba el literal 1b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 el primer
                                        bit del registro LEDS.

173.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

174.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

175.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

176.        SPLK    #11b,LED_STATUS      ;Graba el literal 11b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los dos
                                        primeros bits del registro
LEDS.

177.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

178.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

179.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

180.        SPLK    #111b,LED_STATUS    ;Graba el literal 111b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los tres
                                        primeros bits del registro
LEDS.
```

```
181.        OUT   LED_STATUS,LEDS        ;Copia el contenido de
                                          LED_STATUS en el puerto de
E/S                                       de los leds. Se produce el
                                          desplazamiento hacia la
derecha                                   del led/s encendido/s.

182.        CALL  mS_DELAY                ;Llama a la subrutina
mS_DELAY                                  la cual produce un retardo de
                                          100mseg.

183.        KICK_DOG                      ;Reinicia el WD si este está
                                          activo.

184.        SPLK    #1111b,LED_STATUS     ;Graba el literal 1111b en el
                                          registro LED_STATUS (registro
de                                        los leds). Pone a 1 los
cuatro                                    primeros bits del registro
LEDS.

185.        OUT   LED_STATUS,LEDS         ;Copia el contenido de
                                          LED_STATUS en el puerto de
E/S                                       de los leds. Se produce el
                                          desplazamiento hacia la
derecha                                   del led/s encendido/s.

186.        CALL  mS_DELAY                ;Llama a la subrutina
mS_DELAY                                  la cual produce un retardo de
                                          100mseg.

187.        KICK_DOG                      ;Reinicia el WD si este está
                                          activo.

188.        SPLK    #11111b,LED_STATUS    ;Graba el literal 11111b en el
                                          registro LED_STATUS (registro
de                                        los leds). Pone a 1 los cinco
                                          primeros bits del registro
LEDS.

189.        OUT   LED_STATUS,LEDS         ;Copia el contenido de
                                          LED_STATUS en el puerto de
E/S                                       de los leds. Se produce el
                                          desplazamiento hacia la
derecha                                   del led/s encendido/s.

190.        CALL  mS_DELAY                ;Llama a la subrutina
mS_DELAY                                  la cual produce un retardo de
                                          100mseg.

191.        KICK_DOG                      ;Reinicia el WD si este está
                                          activo.

192.        SPLK    #111111b,LED_STATUS   ;Graba el literal 111111b en
el                                        registro LED_STATUS (registro
de                                        los leds). Pone a 1 los seis
                                          primeros bits del registro
LEDS.

193.        OUT   LED_STATUS,LEDS         ;Copia el contenido de
                                          LED_STATUS en el puerto de
E/S                                       de los leds. Se produce el
```

```
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

194.        LAR     AR0,#7h             ;Carga el literal 7h en el
                                        registro auxiliar AR0. Con
esto                                    creamos un contador que
repite 7                                veces el bloque siguiente.


195.  PASO6:                            ;Etiqueta o nombre de la
                                        subrutina.

196.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

197.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

198.        MAR   *,AR0                 ;Modifica el registro
auxiliar.                               Establece ARP = AR0.

199.        LACC  LED_STATUS            ;Carga el acumulador con el
                                        contenido de LED_STATUS.

200.        SFL                         ;Desplaza el contenido del
                                        acumulador 1 bit hacia la
                                        izquierda.

201.        SACL  LED_STATUS            ;Copia los 16 LSB del
acumulador                              en el registro LED_STATUS.

202.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

203.        BANZ    PASO6               ;Si  AR0 es distinto de 0, el
                                        programa llama a la subrutina
                                        PASO6 y decrementa AR0 en una
                                        unidad, si no pasa a la
                                        instrucción siguiente.

204.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

205.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

206.        SPLK  #1h,LED_STATUS        ;Graba el literal 1b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 el primer
                                        bit del registro LEDS.

207.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
```

```
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

208.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

209.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

210.        SPLK    #11b,LED_STATUS     ;Graba el literal 11b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los dos
                                        primeros bits del registro
LEDS.

211.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

212.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

213.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

214.        SPLK    #111b,LED_STATUS    ;Graba el literal 111b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los tres
                                        primeros bits del registro
LEDS.

215.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

216.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

217.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

218.        SPLK   #1111b,LED_STATUS    ;Graba el literal 1111b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los
cuatro                                  primeros bits del registro
LEDS.

219.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.
```

```
220.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

221.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

222.        SPLK    #11111b,LED_STATUS  ;Graba el literal 11111b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los cinco
                                        primeros bits del registro
LEDS.

223.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

224.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

225.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

226.        SPLK    #111111b,LED_STATUS ;Graba el literal 111111b en
el                                      registro LED_STATUS (registro
de                                      los leds). Pone a 1 los seis
                                        primeros bits del registro
LEDS.

227.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

228.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

229.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

230.        SPLK    #1111111b,LED_STATUS ;Graba el literal 1111111b en
el                                      registro LED_STATUS (registro
de                                      los leds). Pone a 1 los siete
                                        primeros bits del registro
LEDS.

231.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

232.        LAR     AR0,#7h             ;Carga el literal 7h en el
                                        registro auxiliar AR0. Con
esto                                    creamos un contador que
repite 7                                veces el bloque siguiente.
```

```
233.  PASO7:                          ;Etiqueta o nombre de la
                                      subrutina.

234.        CALL  mS_DELAY            ;Llama a la subrutina
mS_DELAY                              la cual produce un retardo de
                                      100mseg.

235.        KICK_DOG                  ;Reinicia el WD si este está
                                      activo.

236.        MAR   *,AR0               ;Modifica el registro
auxiliar.                             Establece ARP = AR0.

237.        LACC  LED_STATUS          ;Carga el acumulador con el
                                      contenido de LED_STATUS.

238.        SFL                       ;Desplaza el contenido del
                                      acumulador 1 bit hacia la
                                      izquierda.

239.        SACL  LED_STATUS          ;Copia los 16 LSB del
acumulador                            en el registro LED_STATUS.

240.        OUT   LED_STATUS,LEDS     ;Copia el contenido de
                                      LED_STATUS en el puerto de
E/S                                   de los leds. Se produce el
                                      desplazamiento hacia la
derecha                               del led/s encendido/s.

241.        BANZ    PASO7             ;Si  AR0 es distinto de 0, el
                                      programa llama a la subrutina
                                      PASO7 y decrementa AR0 en una
                                      unidad, si no pasa a la
                                      instrucción siguiente.


242.        CALL  mS_DELAY            ;Llama a la subrutina
mS_DELAY                              la cual produce un retardo de
                                      100mseg.

243.        KICK_DOG                  ;Reinicia el WD si este está
                                      activo.

244.        SPLK  #1h,LED_STATUS      ;Graba el literal 1b en el
                                      registro LED_STATUS (registro
de                                    los leds). Pone a 1 el primer
                                      bit del registro LEDS.

245.        OUT   LED_STATUS,LEDS     ;Copia el contenido de
                                      LED_STATUS en el puerto de
E/S                                   de los leds. Se produce el
                                      desplazamiento hacia la
derecha                               del led/s encendido/s.

246.        CALL  mS_DELAY            ;Llama a la subrutina
mS_DELAY                              la cual produce un retardo de
                                      100mseg.
```

```
247.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

248.        SPLK    #11b,LED_STATUS     ;Graba el literal 11b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los dos
                                        primeros bits del registro
LEDS.

249.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

250.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

251.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

252.        SPLK    #111b,LED_STATUS    ;Graba el literal 111b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los tres
                                        primeros bits del registro
LEDS.

253.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

254.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

255.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

256.        SPLK   #1111b,LED_STATUS    ;Graba el literal 1111b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los
cuatro                                  primeros bits del registro
LEDS.

257.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

258.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

259.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.
```

```
260.        SPLK    #11111b,LED_STATUS    ;Graba el literal 11111b en
el                                        registro LED_STATUS (registro
de                                        los leds). Pone a 1 los cinco
                                          primeros bits del registro
LEDS.

261.        OUT   LED_STATUS,LEDS         ;Copia el contenido de
                                          LED_STATUS en el puerto de
E/S                                       de los leds. Se produce el
                                          desplazamiento hacia la
derecha                                   del led/s encendido/s.

262.        CALL  mS_DELAY                ;Llama a la subrutina
mS_DELAY                                  la cual produce un retardo de
                                          100mseg.

263.        KICK_DOG                      ;Reinicia el WD si este está
                                          activo.

264.        SPLK    #111111b,LED_STATUS   ;Graba el literal 111111b en
el                                        registro LED_STATUS (registro
de                                        los leds). Pone a 1 los seis
                                          primeros bits del registro
LEDS.

265.        OUT   LED_STATUS,LEDS         ;Copia el contenido de
                                          LED_STATUS en el puerto de
E/S                                       de los leds. Se produce el
                                          desplazamiento hacia la
derecha                                   del led/s encendido/s.

266.        CALL  mS_DELAY                ;Llama a la subrutina
mS_DELAY                                  la cual produce un retardo de
                                          100mseg.

267.        KICK_DOG                      ;Reinicia el WD si este está
                                          activo.

268.        SPLK    #1111111b,LED_STATUS ;Graba el literal 1111111b en
el                                        registro LED_STATUS (registro
de                                        los leds). Pone a 1 los siete
                                          primeros bits del registro
LEDS.

269.        OUT   LED_STATUS,LEDS         ;Copia el contenido de
                                          LED_STATUS en el puerto de
E/S                                       de los leds. Se produce el
                                          desplazamiento hacia la
derecha                                   del led/s encendido/s.

270.        CALL  mS_DELAY                ;Llama a la subrutina
mS_DELAY                                  la cual produce un retardo de
                                          100mseg.

271.        KICK_DOG                      ;Reinicia el WD si este está
                                          activo.

272.        SPLK    #11111111b,LED_STATUS ;Graba el literal 11111111b
en                                         el registro LED_STATUS
                                          (registro delos leds). Pone a
```

```
1                                       los ocho primeros bits del
                                        registro LEDS.

273.        OUT   LED_STATUS,LEDS      ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

274.        LAR     AR0,#7h             ;Carga el literal 7h en el
                                        registro auxiliar AR0. Con
esto                                    creamos un contador que
repite 7                                veces el bloque siguiente.


275.  PASO8:                            ;Etiqueta o nombre de la
                                        subrutina.

276.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

277.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

278.        MAR   *,AR0                 ;Modifica el registro
auxiliar.                               Establece ARP = AR0.

279.        LACC  LED_STATUS            ;Carga el acumulador con el
                                        contenido de LED_STATUS.

280.        SFL                         ;Desplaza el contenido del
                                        acumulador 1 bit hacia la
                                        izquierda.

281.        SACL  LED_STATUS            ;Copia los 16 LSB del
acumulador                              en el registro LED_STATUS.

282.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

283.        BANZ    PASO8               ;Si  AR0 es distinto de 0, el
                                        programa llama a la subrutina
                                        PASO8 y decrementa AR0 en una
                                        unidad, si no pasa a la
                                        instrucción siguiente.

284.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

285.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

286.        SPLK  #1h,LED_STATUS        ;Graba el literal 1b en el
                                        registro LED_STATUS (registro
```

```
de                                  los leds). Pone a 1 el primer
                                    bit del registro LEDS.

287.        OUT   LED_STATUS,LEDS  ;Copia el contenido de
                                    LED_STATUS en el puerto de
E/S                                 de los leds. Se produce el
                                    desplazamiento hacia la
derecha                             del led/s encendido/s.

288.        CALL  mS_DELAY          ;Llama a la subrutina
mS_DELAY                            la cual produce un retardo de
                                    100mseg.

289.        KICK_DOG                ;Reinicia el WD si este está
                                    activo.

290.        SPLK    #11b,LED_STATUS  ;Graba el literal 11b en el
                                    registro LED_STATUS (registro
de                                  los leds). Pone a 1 los dos
                                    primeros bits del registro
LEDS.

291.        OUT   LED_STATUS,LEDS   ;Copia el contenido de
                                    LED_STATUS en el puerto de
E/S                                 de los leds. Se produce el
                                    desplazamiento hacia la
derecha                             del led/s encendido/s.

292.        CALL  mS_DELAY          ;Llama a la subrutina
mS_DELAY                            la cual produce un retardo de
                                    100mseg.

293.        KICK_DOG                ;Reinicia el WD si este está
                                    activo.

294.        SPLK    #111b,LED_STATUS ;Graba el literal 111b en el
                                    registro LED_STATUS (registro
de                                  los leds). Pone a 1 los tres
                                    primeros bits del registro
LEDS.

295.        OUT   LED_STATUS,LEDS   ;Copia el contenido de
                                    LED_STATUS en el puerto de
E/S                                 de los leds. Se produce el
                                    desplazamiento hacia la
derecha                             del led/s encendido/s.

296.        CALL  mS_DELAY          ;Llama a la subrutina
mS_DELAY                            la cual produce un retardo de
                                    100mseg.

297.        KICK_DOG                ;Reinicia el WD si este está
                                    activo.

298.        SPLK   #1111b,LED_STATUS ;Graba el literal 1111b en el
                                    registro LED_STATUS (registro
de                                  los leds). Pone a 1 los
cuatro                              primeros bits del registro
LEDS.
```

```
299.        OUT   LED_STATUS,LEDS        ;Copia el contenido de
                                          LED_STATUS en el puerto de
E/S                                       de los leds. Se produce el
                                          desplazamiento hacia la
derecha                                   del led/s encendido/s.

300.        CALL  mS_DELAY                ;Llama a la subrutina
mS_DELAY                                  la cual produce un retardo de
                                          100mseg.

301.        KICK_DOG                      ;Reinicia el WD si este está
                                          activo.

302.        SPLK    #11111b,LED_STATUS    ;Graba el literal 11111b en
el                                        registro LED_STATUS (registro
de                                        los leds). Pone a 1 los cinco
                                          primeros bits del registro
LEDS.

303.        OUT   LED_STATUS,LEDS         ;Copia el contenido de
                                          LED_STATUS en el puerto de
E/S                                       de los leds. Se produce el
                                          desplazamiento hacia la
derecha                                   del led/s encendido/s.

304.        CALL  mS_DELAY                ;Llama a la subrutina
mS_DELAY                                  la cual produce un retardo de
                                          100mseg.

305.        KICK_DOG                      ;Reinicia el WD si este está
                                          activo.

306.        SPLK    #111111b,LED_STATUS   ;Graba el literal 111111b en
el                                        registro LED_STATUS (registro
de                                        los leds). Pone a 1 los seis
                                          primeros bits del registro
LEDS.

307.        OUT   LED_STATUS,LEDS         ;Copia el contenido de
                                          LED_STATUS en el puerto de
E/S                                       de los leds. Se produce el
                                          desplazamiento hacia la
derecha                                   del led/s encendido/s.

308.        CALL  mS_DELAY                ;Llama a la subrutina
mS_DELAY                                  la cual produce un retardo de
                                          100mseg.

309.        KICK_DOG                      ;Reinicia el WD si este está
                                          activo.

310.        SPLK    #1111111b,LED_STATUS ;Graba el literal 1111111b en
el                                        registro LED_STATUS (registro
de                                        los leds). Pone a 1 los siete
                                          primeros bits del registro
LEDS.

311.        OUT   LED_STATUS,LEDS         ;Copia el contenido de
                                          LED_STATUS en el puerto de
E/S                                       de los leds. Se produce el
```

```
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

312.        LAR     AR0,#7h             ;Carga el literal 7h en el
                                        registro auxiliar AR0. Con
esto                                    creamos un contador que
repite 7                                veces el bloque siguiente.


313.  PASO7V:                           ;Etiqueta o nombre de la
                                        subrutina.

314.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

315.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

316.        MAR   *,AR0                 ;Modifica el registro
auxiliar.                               Establece ARP = AR0.

317.        LACC  LED_STATUS            ;Carga el acumulador con el
                                        contenido de LED_STATUS.

318.        SFL                         ;Desplaza el contenido del
                                        acumulador 1 bit hacia la
                                        izquierda.

319.        SACL  LED_STATUS            ;Copia los 16 LSB del
acumulador                              en el registro LED_STATUS.

320.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

321.        BANZ    PASO7V              ;Si  AR0 es distinto de 0, el
                                        programa llama a la subrutina
                                        PASO7V y decrementa AR0 en
una                                     unidad, si no pasa a la
                                        instrucción siguiente.

322.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

323.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

324.        SPLK  #1h,LED_STATUS        ;Graba el literal 1b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 el primer
                                        bit del registro LEDS.

325.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.
```

```
326.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

327.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

328.        SPLK    #11b,LED_STATUS     ;Graba el literal 11b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los dos
                                        primeros bits del registro
LEDS.

329.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

330.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

331.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

332.        SPLK    #111b,LED_STATUS    ;Graba el literal 111b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los tres
                                        primeros bits del registro
LEDS.

333.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

334.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

335.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

336.        SPLK   #1111b,LED_STATUS    ;Graba el literal 1111b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los
cuatro                                  primeros bits del registro
LEDS.

337.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

338.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.
```

```
339.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

340.        SPLK    #11111b,LED_STATUS  ;Graba el literal 11111b en
el                                      registro LED_STATUS (registro
de                                      los leds). Pone a 1 los cinco
                                        primeros bits del registro
LEDS.

341.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

342.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

343.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

344.        SPLK    #111111b,LED_STATUS ;Graba el literal 111111b en
el                                      registro LED_STATUS (registro
de                                      los leds). Pone a 1 los seis
                                        primeros bits del registro
LEDS.

345.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.


346.        LAR     AR0,#7h             ;Carga el literal 7h en el
                                        registro auxiliar AR0. Con
esto                                    creamos un contador que
repite 7                                veces el bloque siguiente.


347.  PASO6V:                           ;Etiqueta o nombre de la
                                        subrutina.

348.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.


349.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

350.        MAR   *,AR0                 ;Modifica el registro
auxiliar.                               Establece ARP = AR0.

351.        LACC  LED_STATUS            ;Carga el acumulador con el
                                        contenido de LED_STATUS.

352.        SFL                         ;Desplaza el contenido del
                                        acumulador 1 bit hacia la
                                        izquierda.
```

```
353.        SACL  LED_STATUS          ;Copia los 16 LSB del
acumulador                            en el registro LED_STATUS.

354.        OUT   LED_STATUS,LEDS     ;Copia el contenido de
                                      LED_STATUS en el puerto de
E/S                                   de los leds. Se produce el
                                      desplazamiento hacia la
derecha                               del led/s encendido/s.

355.        BANZ    PASO6V            ;Si  AR0 es distinto de 0, el
                                      programa llama a la subrutina
                                      PASO6V y decrementa AR0 en
una                                   unidad, si no pasa a la
                                      instrucción siguiente.

356.        CALL  mS_DELAY            ;Llama a la subrutina
mS_DELAY                              la cual produce un retardo de
                                      100mseg.

357.        KICK_DOG                  ;Reinicia el WD si este está
                                      activo.

358.        SPLK  #1h,LED_STATUS      ;Graba el literal 1b en el
                                      registro LED_STATUS (registro
de                                    los leds). Pone a 1 el primer
                                      bit del registro LEDS.

359.        OUT   LED_STATUS,LEDS     ;Copia el contenido de
                                      LED_STATUS en el puerto de
E/S                                   de los leds. Se produce el
                                      desplazamiento hacia la
derecha                               del led/s encendido/s.

360.        CALL  mS_DELAY            ;Llama a la subrutina
mS_DELAY                              la cual produce un retardo de
                                      100mseg.

361.        KICK_DOG                  ;Reinicia el WD si este está
                                      activo.

362.        SPLK    #11b,LED_STATUS    ;Graba el literal 11b en el
                                      registro LED_STATUS (registro
de                                    los leds). Pone a 1 los dos
                                      primeros bits del registro
LEDS.

363.        OUT   LED_STATUS,LEDS     ;Copia el contenido de
                                      LED_STATUS en el puerto de
E/S                                   de los leds. Se produce el
                                      desplazamiento hacia la
derecha                               del led/s encendido/s.

364.        CALL  mS_DELAY            ;Llama a la subrutina
mS_DELAY                              la cual produce un retardo de
                                      100mseg.

365.        KICK_DOG                  ;Reinicia el WD si este está
                                      activo.
```

```
366.        SPLK    #111b,LED_STATUS    ;Graba el literal 111b en el
                                            registro LED_STATUS (registro
de                                          los leds). Pone a 1 los tres
                                            primeros bits del registro
LEDS.

367.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                            LED_STATUS en el puerto de
E/S                                         de los leds. Se produce el
                                            desplazamiento hacia la
derecha                                     del led/s encendido/s.

368.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                    la cual produce un retardo de
                                            100mseg.

369.        KICK_DOG                    ;Reinicia el WD si este está
                                            activo.

370.        SPLK   #1111b,LED_STATUS    ;Graba el literal 1111b en el
                                            registro LED_STATUS (registro
de                                          los leds). Pone a 1 los
cuatro                                      primeros bits del registro
LEDS.

371.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                            LED_STATUS en el puerto de
E/S                                         de los leds. Se produce el
                                            desplazamiento hacia la
derecha                                     del led/s encendido/s.

372.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                    la cual produce un retardo de
                                            100mseg.

373.        KICK_DOG                    ;Reinicia el WD si este está
                                            activo.

374.        SPLK    #11111b,LED_STATUS  ;Graba el literal 11111b en el
                                            registro LED_STATUS (registro
de                                          los leds). Pone a 1 los cinco
                                            primeros bits del registro
LEDS.

375.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                            LED_STATUS en el puerto de
E/S                                         de los leds. Se produce el
                                            desplazamiento hacia la
derecha                                     del led/s encendido/s.

376.        LAR     AR0,#7h             ;Carga el literal 7h en el
                                            registro auxiliar AR0. Con
esto                                        creamos un contador que
repite 7                                    veces el bloque siguiente.


377.  PASO5V:                           ;Etiqueta o nombre de la
                                            subrutina.
```

```
378.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

379.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

380.        MAR   *,AR0                 ;Modifica el registro
auxiliar.                               Establece ARP = AR0.

381.        LACC  LED_STATUS            ;Carga el acumulador con el
                                        contenido de LED_STATUS.

382.        SFL                         ;Desplaza el contenido del
                                        acumulador 1 bit hacia la
                                        izquierda.

383.        SACL  LED_STATUS            ;Copia los 16 LSB del
acumulador                              en el registro LED_STATUS.

384.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

385.        BANZ    PASO5V              ;Si  AR0 es distinto de 0, el
                                        programa llama a la subrutina
                                        PASO5V y decrementa AR0 en
una                                     unidad, si no pasa a la
                                        instrucción siguiente.

386.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

387.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

388.        SPLK  #1h,LED_STATUS        ;Graba el literal 1b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 el primer
                                        bit del registro LEDS.

389.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

390.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

391.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

392.        SPLK    #11b,LED_STATUS     ;Graba el literal 11b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los dos
```

```
                                        primeros bits del registro
LEDS.

393.        OUT   LED_STATUS,LEDS      ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

394.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

395.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

396.        SPLK    #111b,LED_STATUS    ;Graba el literal 111b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los tres
                                        primeros bits del registro
LEDS.

397.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

398.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

399.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

400.        SPLK   #1111b,LED_STATUS    ;Graba el literal 1111b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los
cuatro                                  primeros bits del registro
LEDS.

401.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

402.        LAR     AR0,#7h             ;Carga el literal 7h en el
                                        registro auxiliar AR0. Con
esto                                    creamos un contador que
repite 7                                veces el bloque siguiente.


403.  PASO4V:                           ;Etiqueta o nombre de la
                                        subrutina.

404.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.
```

```
405.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

406.        MAR   *,AR0                 ;Modifica el registro
auxiliar.                               Establece ARP = AR0.

407.        LACC  LED_STATUS            ;Carga el acumulador con el
                                        contenido de LED_STATUS.

408.        SFL                         ;Desplaza el contenido del
                                        acumulador 1 bit hacia la
                                        izquierda.

409.        SACL  LED_STATUS            ;Copia los 16 LSB del
acumulador                              en el registro LED_STATUS.

410.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

411.        BANZ    PASO4V              ;Si  AR0 es distinto de 0, el
                                        programa llama a la subrutina
                                        PASO4V y decrementa AR0 en
una                                     unidad, si no pasa a la
                                        instrucción siguiente.



412.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

413.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

414.        SPLK  #1h,LED_STATUS        ;Graba el literal 1b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 el primer
                                        bit del registro LEDS.

415.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

416.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

417.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

418.        SPLK    #11b,LED_STATUS     ;Graba el literal 11b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los dos
                                        primeros bits del registro
LEDS.
```

```
419.        OUT   LED_STATUS,LEDS        ;Copia el contenido de
                                          LED_STATUS en el puerto de
E/S                                       de los leds. Se produce el
                                          desplazamiento hacia la
derecha                                   del led/s encendido/s.

420.        CALL  mS_DELAY                ;Llama a la subrutina
mS_DELAY                                  la cual produce un retardo de
                                          100mseg.

421.        KICK_DOG                      ;Reinicia el WD si este está
                                          activo.

422.        SPLK    #111b,LED_STATUS     ;Graba el literal 111b en el
                                          registro LED_STATUS (registro
de                                        los leds). Pone a 1 los tres
                                          primeros bits del registro
LEDS.

423.        OUT   LED_STATUS,LEDS        ;Copia el contenido de
                                          LED_STATUS en el puerto de
E/S                                       de los leds. Se produce el
                                          desplazamiento hacia la
derecha                                   del led/s encendido/s.

424.        LAR     AR0,#7h               ;Carga el literal 7h en el
                                          registro auxiliar AR0. Con
esto                                      creamos un contador que
repite 7                                  veces el bloque siguiente.


425.  PASO3V:                             ;Etiqueta o nombre de la
                                          subrutina.

426.        CALL  mS_DELAY                ;Llama a la subrutina
mS_DELAY                                  la cual produce un retardo de
                                          100mseg.

427.        KICK_DOG                      ;Reinicia el WD si este está
                                          activo.

428.        MAR   *,AR0                   ;Modifica el registro
auxiliar.                                 Establece ARP = AR0.

429.        LACC  LED_STATUS              ;Carga el acumulador con el
                                          contenido de LED_STATUS.

430.        SFL                           ;Desplaza el contenido del
                                          acumulador 1 bit hacia la
                                          izquierda.

431.        SACL  LED_STATUS              ;Copia los 16 LSB del
acumulador                                en el registro LED_STATUS.

432.        OUT   LED_STATUS,LEDS        ;Copia el contenido de
                                          LED_STATUS en el puerto de
E/S                                       de los leds. Se produce el
                                          desplazamiento hacia la
derecha                                   del led/s encendido/s.
```

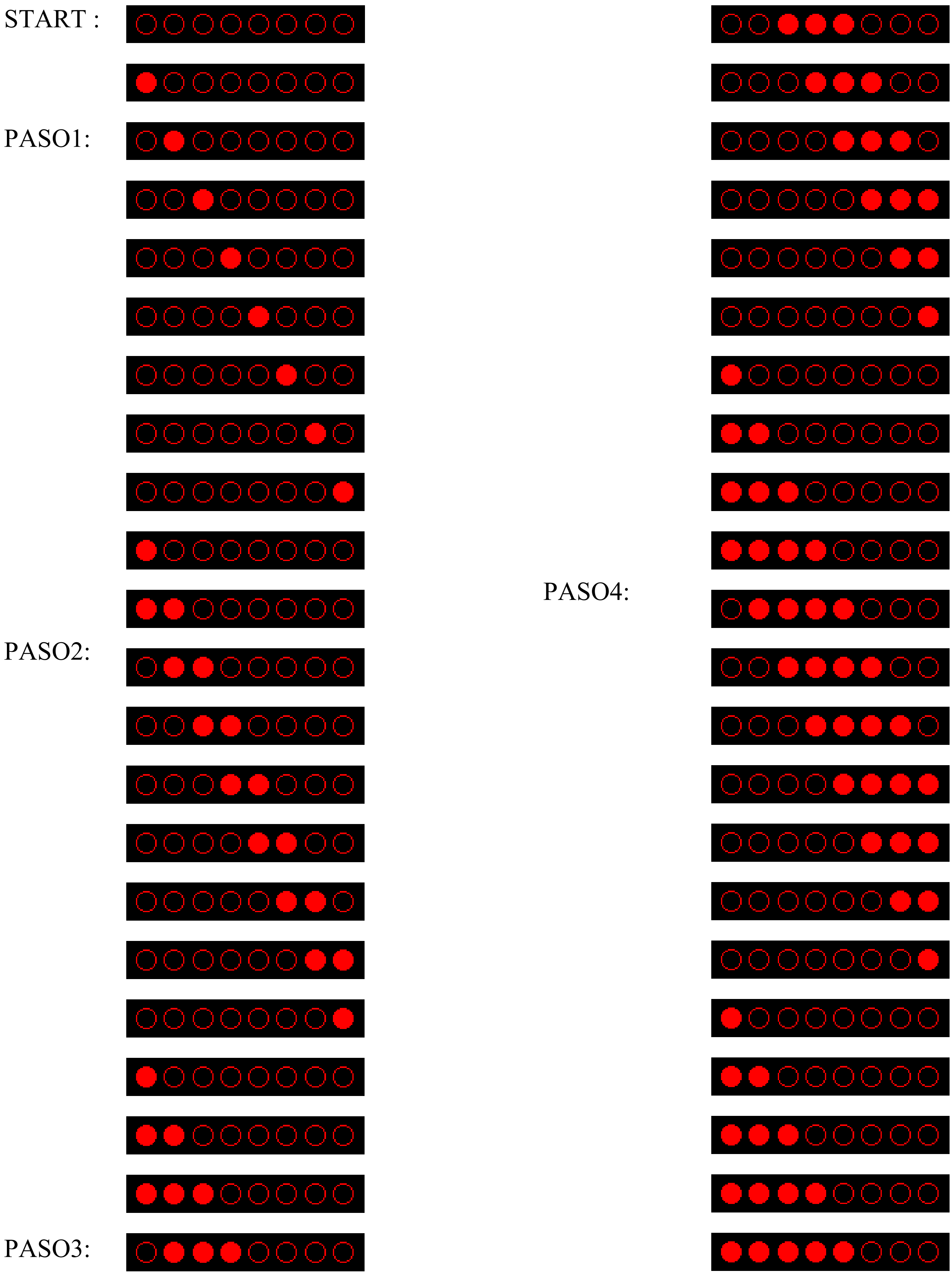
```
433.        BANZ    PASO3V              ;Si  AR0 es distinto de 0, el
                                        programa llama a la subrutina
                                        PASO3V y decrementa AR0 en
una                                     unidad, si no pasa a la
                                        instrucción siguiente.

434.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

435.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

436.        SPLK  #1h,LED_STATUS        ;Graba el literal 1b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 el primer
                                        bit del registro LEDS.

437.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

438.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

439.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

440.        SPLK    #11b,LED_STATUS     ;Graba el literal 11b en el
                                        registro LED_STATUS (registro
de                                      los leds). Pone a 1 los dos
                                        primeros bits del registro
LEDS.

441.        OUT   LED_STATUS,LEDS       ;Copia el contenido de
                                        LED_STATUS en el puerto de
E/S                                     de los leds. Se produce el
                                        desplazamiento hacia la
derecha                                 del led/s encendido/s.

442.        LAR     AR0,#7h             ;Carga el literal 7h en el
                                        registro auxiliar AR0. Con
esto                                    creamos un contador que
repite 7                                veces el bloque siguiente.


443.  PASO2V:                           ;Etiqueta o nombre de la
                                        subrutina.

444.        CALL  mS_DELAY              ;Llama a la subrutina
mS_DELAY                                la cual produce un retardo de
                                        100mseg.

445.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

446.        MAR   *,AR0                 ;Modifica el registro
auxiliar.                               Establece ARP = AR0.
```

```
447.        LACC  LED_STATUS          ;Carga el acumulador con el
                                      contenido de LED_STATUS.

448.        SFL                       ;Desplaza el contenido del
                                      acumulador 1 bit hacia la
                                      izquierda.

449.        SACL  LED_STATUS          ;Copia los 16 LSB del
acumulador                            en el registro LED_STATUS.

450.        OUT   LED_STATUS,LEDS     ;Copia el contenido de
                                      LED_STATUS en el puerto de
E/S                                   de los leds. Se produce el
                                      desplazamiento hacia la
derecha                               del led/s encendido/s.

451.        BANZ    PASO2V            ;Si  AR0 es distinto de 0, el
                                      programa llama a la subrutina
                                      PASO2V y decrementa AR0 en
una                                   unidad, si no pasa a la
                                      instrucción siguiente.

452.        LAR     AR0,#7h           ;Carga el literal 7h en el
                                      registro auxiliar AR0. Con
esto                                  reamos un contador que repite
7                                     veces el bloque siguiente.

453.        CALL  mS_DELAY            ;Llama a la subrutina
mS_DELAY                              la cual produce un retardo de
                                      100mseg.

454.        KICK_DOG                  ;Reinicia el WD si este está
                                      activo.

455.        SPLK  #1h,LED_STATUS      ;Graba el literal 1b en el
                                      registro LED_STATUS (registro
de                                    los leds). Pone a 1 el primer
                                            bit del registro LEDS.
456.        OUT   LED_STATUS,LEDS     ;Copia el contenido de
                                      LED_STATUS en el puerto de
E/S                                   de los leds. Se produce el
                                      desplazamiento hacia la
derecha                               del led/s encendido/s.

457.  PASO1V:                         ;Etiqueta o nombre de la
                                      subrutina.

458.        B       START             ;Salta al bloque START del
                                      principio, volviendo a
repetir                               todo el programa completo.
```

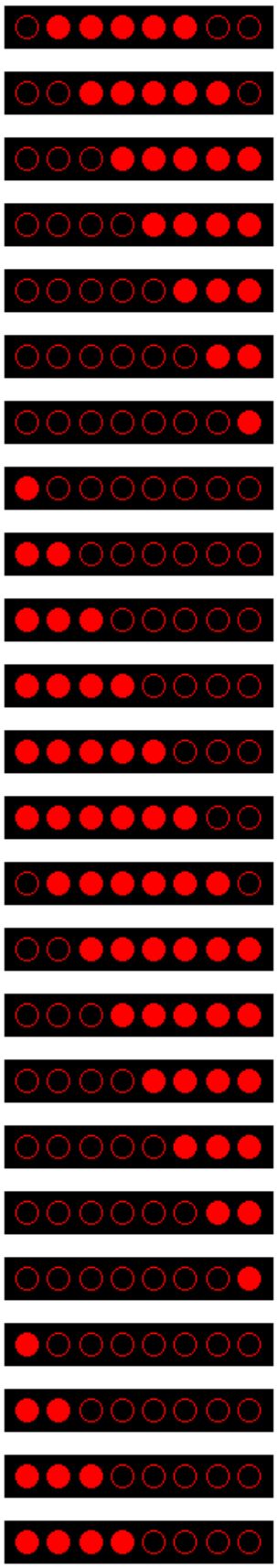
```
;=====================================================================
; Rutina :  mS_DELAY
;
;     Produce retardos multiples de 0.1ms mediante el uso de la

;     instrucción RPT. El retardo producido está basado en el valor

;     grabado en la variable mSEC (Retardo=mSEC·0.1ms). Para contar

;     el número de veces que se repite el bucle de retardo, se

;     utiliza el direccionamiento indirecto.
;
;=====================================================================

mS_DELAY:                       ;Etiqueta o nombre de la
                                subrutina.

        LDP   #0h               ;Carga los 9 bits LSB de la
                                dirección 0h en el registro
                                DP (puntero memoria de datos)
y                               los une con otro registro de
7                               bits para formar una palabra
de                              16 bits. Reserva para datos
las                             direcciones 0000h-007Fh.

        LACC  #2000             ;Carga el acumulador con el
                                literal 2000 que es el valor
de                              la repetición.

        SACL  RPT_NUM           ;Copia los 16 LSB del
acumulador                      en el registro RPT_NUM.

        LAR   AR1,mSEC          ;Carga el contenido de la
                                variable mSEC en el registro
                                auxiliar AR1. Genera un bucle
de                              retardo de (AR0*0.1)mSEC.

        MAR   *,AR1             ;Modifica el registro
auxiliar.                       Establece ARP = AR1


mS_LOOP:                        ;Etiqueta o nombre de la
                                subrutina.

        LDP   #0h               ;Carga los 9 bits LSB de la
                                dirección 0h en el registro
                                DP (puntero memoria de datos)
y                               los une con otro registro de
7                               bits para formar una palabra
de                              16 bits. Reserva para datos
las                             direcciones 0000h-007Fh.

        RPT   RPT_NUM           ;Carga los 8 LSB del registro
                                RPT_NUM, y repite la
instrucción                     siguiente el número de veces
que                             este indica. Esto es 2000
ciclos                          = 0.1 ms.

        NOP                     ;Introduce un retardo de un
                                ciclo.
```

```
469.        BANZ  mS_LOOP               ;Si  AR1 es distinto de 0, el
                                        programa llama a la subrutina
                                        mS_LOOP y decrementa AR0 en
una                                     unidad, si no pasa a la
                                        instrucción siguiente.

470.        RET                         ;Sale de la subrutina mS_LOOP
                                        y extrae el contenido
superior                                de la pila para llevarlo al
                                        contador de programa.


;=====================================================================
; Rutina: PHANTOM
;
; Bucle para poner trampas a las falsas interrupciones.;
;
;=====================================================================

471.  PHANTOM:

472.        KICK_DOG                    ;Reinicia el WD si este está
                                        activo.

473.        B     PHANTOM               ;Salta al bloque PHANTOM
                                        volviendo a repetirlo.
```

La secuencia de encendido de los leds provocada por cada bucle del código ensamblador es la que se muestra a continuación:

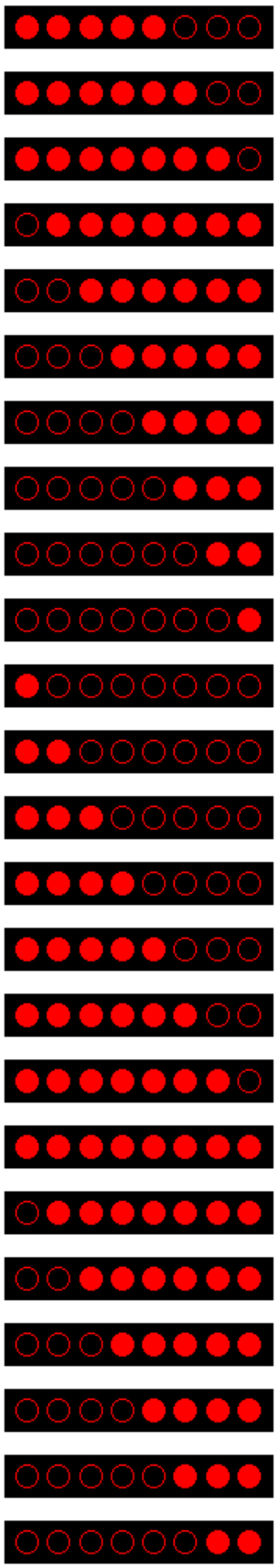


PASO5:



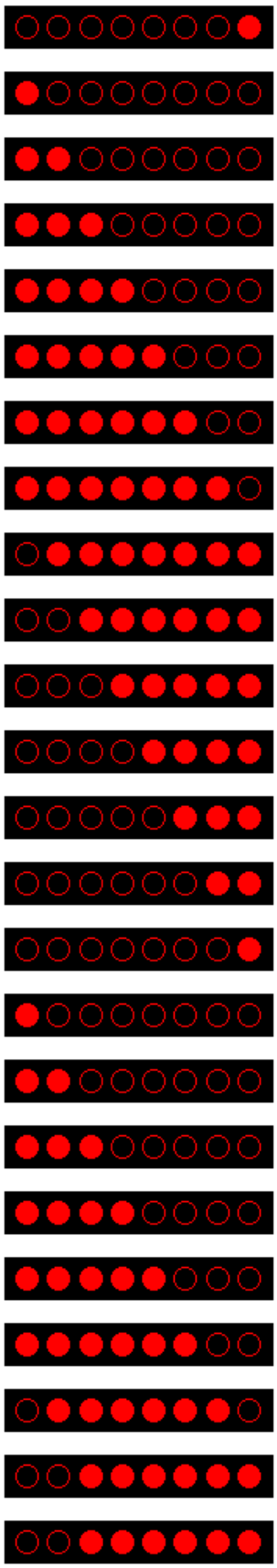
PASO6:

PASO7:



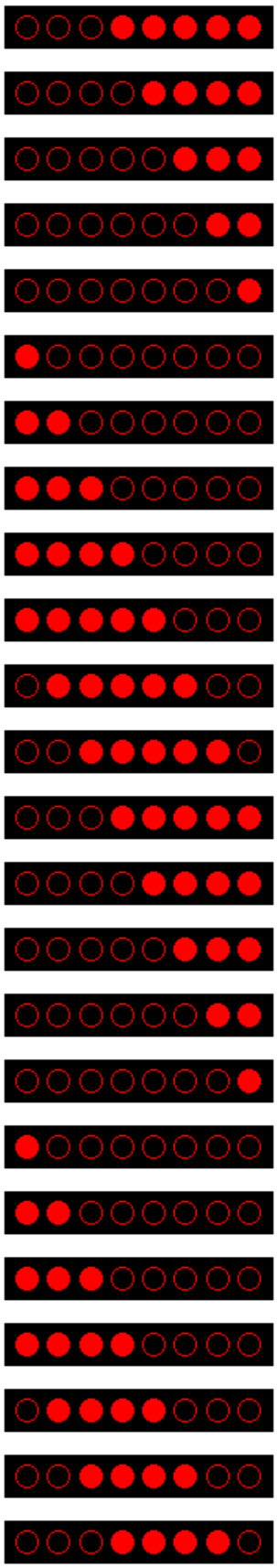
PASO8:

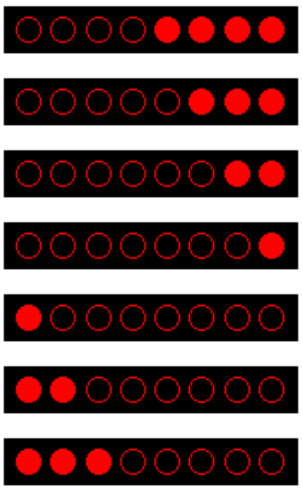
PASO7V:



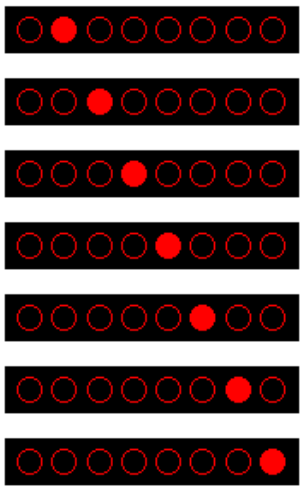
PASO6V:

PASO5V:

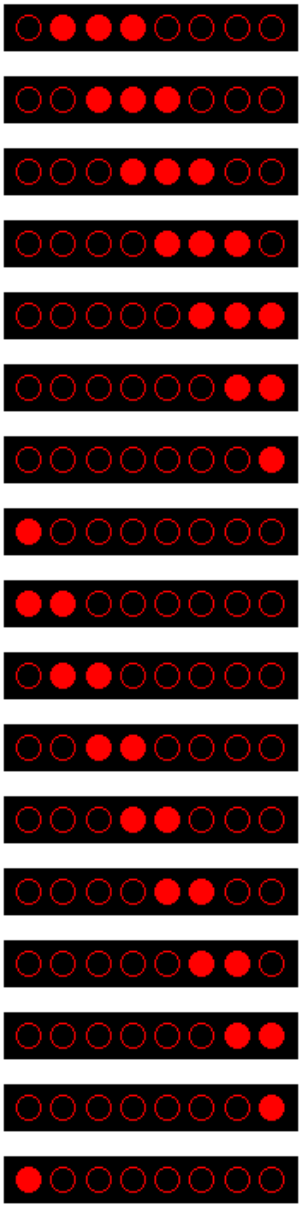




PASO1V:



PASO3V:



PASO2V: