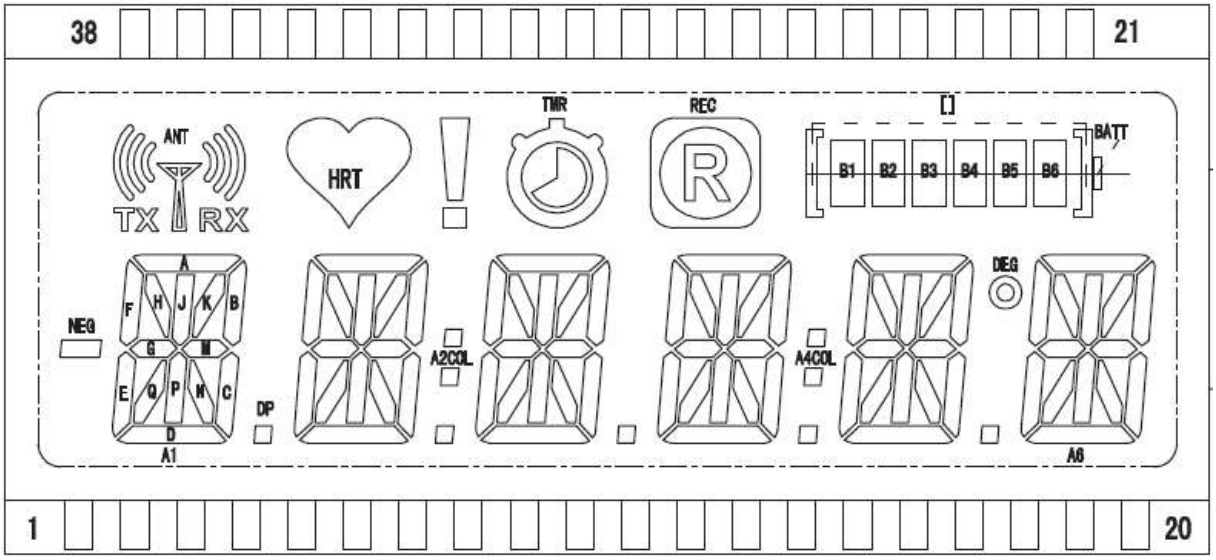
SISTEMAS BASADOS EN MICROPROCESADOR
PRÁCTICA 2. C+ASM. PANTALLA LCD

Grado en Ingeniería Electrónica Industrial (3er curso)
Curso 2026/2027

1 INTRODUCCIÓN

El MSP-EXP430FR6989 *Launchpad* dispone de un *display* LCD controlable directamente por el procesador. Puede visualizar hasta 6 caracteres alfanuméricos usando dígitos de 14 segmentos, y algunos símbolos especiales.



1.1 Objetivos

- Desarrollar programas mixtos en C y ensamblador.
- Controlar una pantalla LCD.
- Hacer una librería de uso de la pantalla.

2 ESTUDIO TEÓRICO

2.1 Trabajando con el controlador LCD

Busque y estudie en la documentación de examen de la página web de la asignatura la documentación del controlador LCD (`lcd.pdf`). En ella se recogen los registros de control del módulo de *display* del FR6989. Este módulo es muy complejo y versátil para que se puedan acomodar diversas pantallas (hasta 320 segmentos) y escenarios. No obstante, dada la pantalla conectada en nuestro caso en el *Launchpad*, la configuración es muy concreta:

- f_{LCD}=1kHz (aprox.), modo 4 mux, segmentos activados y bajo consumo.
- V_{LCD}=2’6V (generado internamente), tensiones V2-V4 internas, V5=Vss, bomba de carga activa y sincronizada con el reloj.
- Borrar la pantalla.
- Activar la pantalla.

Los segmentos del controlador de LCD están multiplexados con los pines del microcontrolador. Puede encontrar en el manual de usuario del *Launchpad* qué segmentos del módulo LCD se usan para controlar el *display* físico. Lo ideal es activar sólo los segmentos que se usan realmente con nuestra pantalla, para dejar los demás libres para otras funciones.

Observe la distribución de segmentos en cada registro de la memoria de vídeo. Preste atención a los segmentos adicionales ubicados en cada uno de los dígitos: NEG y A1DP en el dígito A1, A2COL y A2DP en A2, etc. Note también la distribución aparentemente caprichosa de los dígitos en la memoria. Se ha hecho así para simplificar el conexionado de las señales desde el micro a la pantalla, pero dificulta la programación.

3 ESTUDIO PRÁCTICO

3.1 Proyecto mixto C-ensamblador

Es posible realizar programas escritos en parte en C y en parte en ensamblador. Para ello se deben incluir distintos ficheros dentro de lo que se conoce como un *proyecto* en CCS. Cada fichero tiene una extensión en función del lenguaje usado: `.c` para C y `.asm` para ensamblador. No es posible mezclar C y ASM en el mismo fichero.

- Cree un nuevo proyecto pulsando *Project/New CCS Project...*
- Si no aparece por defecto el modelo del procesador, introdúzcalo en la casilla "Target".
- Nombre al proyecto "P2.1".
- Seleccione la opción por defecto *Empty Project (with main.c)*.
- Pulse *Finish*. Se activa el nuevo proyecto y abre el esqueleto de un fichero `main.c`. En el mismo ya se ha puesto la instrucción que desactiva el perro guardián, pero no la que desbloquea los puertos. Tendrá que hacerlo manualmente con la instrucción:

```
PM5CTL0 &= ~LOCKLPM5;   //Desbloquea los puertos de E/S
```

- Es necesario cambiar algunas opciones por defecto del proyecto. Pulse con el botón derecho del ratón sobre el nombre del mismo y seleccione *Properties* (o pulse Alt-Enter):
  - Pasar a modelo de memoria pequeño. Esto hace que sólo se usen los primeros 64KB de memoria tanto para código como para datos, lo que nos permitirá usar instrucciones de la CPU de 16 bits (CPU) en lugar de la de 20 bits (CPUX): vaya a *Build/MSP430 Compiler/Processor Options*. Especifique modelo *small* para código y datos.
  - Desactivar optimizaciones. Esto evita que el compilador elimine código que puede considerar inútil, como los bucles de espera activa: vaya a *Build/MSP430 Compiler/Optimization*. En *Optimization level*, seleccione *off*.
  - Pulse OK.

3.2 Módulo de gestión de la pantalla `lcd.asm`

Un módulo es un fichero de código que recoge subrutinas y variables relacionadas con una tarea o sección específica del microcontrolador. En esta práctica se va a partir de un módulo precreado, al que se le añadirán nuevas funciones.

Descargue el fichero `lcd.asm` de la carpeta `Práctica 2` de la web de la asignatura y anote dónde lo guarda (por ejemplo, Escritorio). Pulse con el botón derecho del ratón sobre el proyecto y seleccione *Add Files...*. Busque `lcd.asm` donde lo guardó y haga doble click sobre él. En el cuadro de diálogo que se abre, asegúrese de que está activa la opción *Copy files* y pulse OK.

Abra el fichero para editarlo en CCS. Estudie el contenido:

- La directiva `.cdecls` permite incluir un fichero de cabecera de C y lo traduce a ensamblador. En nuestro caso carga las definiciones del MSP430.
- La directiva `.global` declara una serie de etiquetas que se van a referenciar desde fuera. En nuestro caso nombra las subrutinas que se van a definir en este módulo. Si en el futuro añade subrutinas a este módulo, tendrá que añadir aquí sus etiquetas si van a ser accedidas desde otro módulo.
- Hay un vector en el que se define la tabla de 14 segmentos. Cada carácter necesita dos bytes. En el bajo se guardan los segmentos A-M y en el alto los siguientes. Observe que los bits 0 y 2 del byte alto se usan para otros símbolos y no se definen aquí. Es muy interesante incluir una cabecera como la que tiene esta tabla para documentar cada subrutina.

Haga ahora el fichero de cabecera. Pulse de nuevo con el botón derecho del ratón sobre el proyecto y seleccione *New/Header file*. Nombre el fichero `lcd.h`. Se abrirá el esqueleto de un fichero de cabecera de C. Debe incluir las definiciones y prototipos del módulo en el hueco de las líneas 10 a 12 (puede usar más si lo necesita). Los prototipos de este módulo son:

3.2.1 Inicializar la pantalla `lcdIni`

```
void lcdIni (void);
```

Inicializa la pantalla para que se pueda usar. Esta subrutina ya está incluida en `lcd.asm`. Estudie el código y compárelo con las instrucciones de la sección 2.1 del estudio teórico.

3.2.2 Conversor de ASCII a 14 segmentos `lcda2seg`

```
unsigned int lcda2seg (char c);
```

Recibe en `c` un carácter ASCII imprimible (códigos 32 a 127) y devuelve su representación a 14 segmentos según el lcd del *Launchpad*. Devuelve `0xFFFF` si el carácter está fuera de rango. Ayúdese de la tabla de conversión presente en el módulo `Tab14Seg`.

3.2.3 Escribir un carácter con desplazamiento a la izquierda `lcdPintaIzq`

```
void lcdPintaIzq (char c);
```

Si `c` es un carácter ASCII imprimible, desplaza el contenido de la pantalla un carácter a la izquierda (el carácter de la izquierda se pierde) y escribe `c` en el dígito de la derecha. En caso contrario no hace nada. Llama a `lcda2seg()` para obtener su representación en segmentos.

Preste atención al proceso de desplazamiento, ya que no deben desplazarse los segmentos externos de cada dígito (bits 0 y 2 del MSB). Es decir, cuando se mueva la información de segmentos del dígito 6 para ponerlo en el dígito 5 (por ejemplo), debe mover todos los bits excepto los bits 0 y 2 del MSB, que deberán conservarse.

3.2.4 Escribir un carácter con desplazamiento a la derecha `lcdPintaDer`

```
void lcdPintaDer (char c);
```

Igual que la anterior, pero desplaza el contenido de la pantalla un carácter a la derecha y escribe `c` en el dígito de la izquierda.

3.2.5 Borrar la pantalla `lcdBorraTodo`

```
void lcdBorraTodo (void);
```

Borra el contenido de la pantalla, incluyendo símbolos.

3.2.6 Borrar los dígitos `lcdBorra`

```
void lcdBorra (void);
```

Borra los dígitos de la fila inferior de la pantalla.

3.2.7 Control del icono de batería `lcdBateria`

```
uint8_t lcdBateria (uint8_t b);
```

Establece el estado de los segmentos del indicador de nivel de batería. Los bits 0 a 5 de `b` indican, respectivamente, los estados de los segmentos 1 a 6 de la batería. El bit 6 indica el estado de la forma de la batería `[ ]` y el 7 el polo positivo. 0 para apagado y 1 para encendido. Devuelve el estado anterior de dichos iconos.

3.2.8 Función iconos `lcdIconos`

```
uint8_t lcdIconos (uint8_t map);
```

Enciende/apaga diversos iconos de la pantalla LCD. `map` es un mapa de bits que guarda el estado de cada icono. La distribución es la siguiente:

7	6	5	4	3	2	1	0
°	TX	ANT	RX	TMR	HRT	REC	!

Los bits usan lógica positiva: un 1 enciende el icono y un 0 lo apaga. Devuelve el estado anterior de dichos iconos.

3.2.9 Función InterDigito `lcdInterDigito`

```
void lcdInterDigito (uint8_t map);
```

Enciende/apaga los segmentos de los puntos decimales, los dos puntos y el signo menos. `map` en un mapa de bits que guarda el estado de cada icono. La distribución es la siguiente:

7	6	5	4	3	2	1	0
A2COL	A4COL	A5DP	A4DP	A3DP	A2DP	A1DP	NEG

Los bits usan lógica positiva: un 1 enciende el icono y un 0 lo apaga. Devuelve el estado anterior de dichos iconos.

3.3 Programa principal en C

Para poder usar el módulo anterior desde C, debe incluir el fichero de cabecera correspondiente:

```
#include "lcd.h"
```

La instrución del preprocesador de C `#include` permite dos formatos para especificar el nombre del fichero a incluir. Si el mismo se rodea de los caracteres '<' y '>', se busca el fichero en el fichero del sistema del compilador. Si se rodea de dobles comillas, se busca en el directorio del proyecto.

```
#include <msp430.h>  //msp430.h está en el directorio del sistema
#include "lcd.h"     //lcd.h está en el directorio del proyecto
```

Dentro de `main()` llame a las funciones de inicialización de los módulos. En nuestro caso el módulo lcd:

```
int main (void)
{
  WDTCTL = WDTPW | WDTHOLD;          //Detener perro guardián
  PM5CTL0 &= ~LOCKLPM5;              //Desbloquear puertos E/S

  lcdIni ();                         //Inicializar display LCD

  //Resto del código
}
```

Inicialmente el programa hace una prueba de la pantalla enciendo TODOS los segmentos durante 2 segundos[1]. Después la apaga y pasa a hacer ella animación siguiente:

Aparece el mensaje "`***Sistemas Basados en Microprocesador***`" en la pantalla desplazándose alternativamente a izquierda y derecha repetidamente[2]. Después de imprimir cada carácter llame a una función en C que haga una pausa de unos 0'2 segundos[1]. La visualización del mensaje debe simultanearse con el movimiento de un segmento del icono de la batería. Inicialmente se encenderá el de más a la derecha. Cada vez que se muestre un carácter del mensaje, se cambiará el segmento encendido al que está a su izquierda. Cada vez que se toque un extremo, se invertirá el movimiento.

Después de hacer el tercer ciclo (primero hacia la izquierda, luego a la derecha y finalmente a la izquierda otra vez), borre la pantalla completamente y empiece el proceso de animación desde cero.

1. Haga un bucle de espera activa para hacer la pausa. El tiempo es aproximado.
2. Cuando termine de hacer la impresión hacia la izquierda el contenido de la pantalla será "`dor***`". Cuando inicie la impresión hacia la derecha debe tenerlo en cuenta para que la siguiente imagen sea "`ador**`". El comportamiento debe ser análogo cuando termine la impresión hacia la derecha.