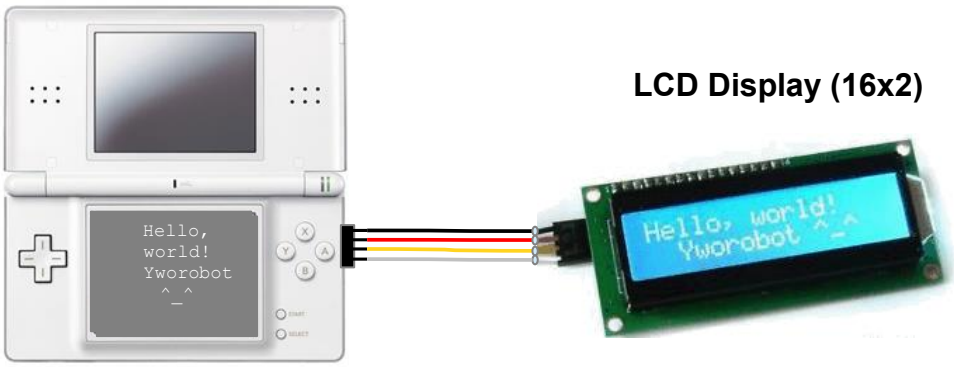
Problema 8: Display LCD

Se propone controlar con la NDS un display LCD de 2 filas x 16 columnas tipo HD44780U. El programa a realizar recibirá mensajes por wifi y los representará en el display:



En el esquema anterior se han representado los pines de un adaptador de Entrada/Salida que permite a la NDS enviar comandos al LCD mediante dos cables de datos (más dos cables de alimentación). En realidad, el display recibirá los datos serializados, pero de este proceso ya se encargará el controlador de E/S del adaptador.

Desde el punto de vista del programa a realizar, el controlador dispone de un registro de E/S denominado REG_DISPLAY, de 16 bits, de los cuales solo se utilizan los 10 bits de menor peso. La siguiente tabla muestra los posibles comandos que se pueden enviar con estos 10 bits, junto con sus respectivos parámetros:

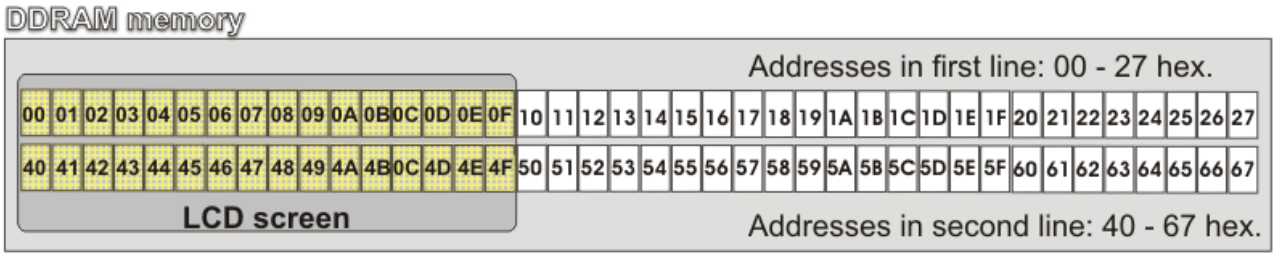
Instruction	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Clear display	0	0	0	0	0	0	0	0	0	1
Cursor home	0	0	0	0	0	0	0	0	1	x
Entry mode set	0	0	0	0	0	0	0	1	I/D	S
Display on/off control	0	0	0	0	0	0	1	D	C	B
Cursor/display shift	0	0	0	0	0	1	S/C	R/L	x	x
Function set	0	0	0	0	1	DL	N	x	BR1	BR0
CGRAM address set	0	0	0	1	CGRAM address					
DDRAM address set	0	0	1	DDRAM address						
Address counter read	0	1	BF=0	AC contents						
DDRAM or CGRAM write	1	0	Write data							
DDRAM or CGRAM read	1	1	Read data							

x = don't care

De todos estos comandos, para resolver este problema de examen nos interesan tres:

- `Cursor/display shift:` REG_DISPLAY = `000001(S/C)(R/L)xx`
  - `S/C: 1 → screen, 0 → cursor`
  - `R/L: 1 → right, 0 → left`
- `DDRAM Address set:` REG_DISPLAY = `001(DDRAM address)`
  - `DDRAM address (7 bits)`
- `DDRAM or CGRAM write:` REG_DISPLAY = `10(Write data)`
  - `Write data (8 bits)`

Internamente, el display dispone de una memoria RAM de 80 posiciones de 1 byte cada una, denominada DDRAM (*Display Data RAM*), distribuidas en 2 filas de 40 columnas, aunque en la pantalla LCD (*screen*) solo se visualizan 2 filas por 16 columnas:



En el gráfico anterior, para cada posición de la DDRAM se muestra su dirección de memoria, en hexadecimal. Sin embargo, hay que tener en cuenta que en cada posición se guardará el código del carácter a visualizar, típicamente con su codificación ASCII.

Como la pantalla LCD solo dispone de 16 columnas, para poder visualizar todo el contenido de la memoria será necesario desplazar la posición inicial de la pantalla hacia la derecha o hacia la izquierda, con el comando "`Cursor/display shift`".

El comando "`DDRAM address set`" permite fijar la dirección de la posición de memoria que se requiere leer o escribir. El comando "`DDRAM or CGRAM write`" permite escribir un dato (un byte) en la posición de memoria previamente fijada con el comando anterior. Cuando se fija una dirección de memoria, se puede escribir un conjunto de caracteres consecutivamente, sin tener que especificar la dirección para cada carácter, ya que el propio display se encargará de aumentar automáticamente la posición actual de escritura en memoria.

El programa a implementar deberá recibir mensajes de hasta 32 caracteres por la wifi y transferirlos a la memoria DDRAM, mediante los comandos adecuados. Cada nuevo mensaje que se reciba se escribirá en la segunda linea de la DDRAM, mientras que el contenido anterior de la segunda linea se deberá copiar a la primera linea, realizando un efecto de *scroll* vertical. Los mensajes recibidos también se deberán mostrar por la pantalla inferior de la NDS.

Además, la visualización de la pantalla del display deber ir desplazándose para que se pueda leer todo el contenido de los mensajes, cada cierto tiempo, realizando un efecto de *scroll* horizontal. Se propone el siguiente algoritmo:

- Visualización de las 16 primeras columnas; espera de 3 segundos
- Desplazamiento gradual de 16 columnas a la derecha, durante 2 segundos
- Visualización de las 16 columnas siguientes; espera de 3 segundos
- Desplazamiento gradual de 16 columnas a la izquierda, durante 2 segundos

Estos pasos se deben repetir indefinidamente, es decir, cuando se acaba el último paso se vuelve a empezar por el primero. El algoritmo está simplificado, en el sentido de que no tiene en cuenta la longitud concreta de los mensajes cuando realiza el desplazamiento. Tampoco será necesario reiniciar el algoritmo en el momento que se inserte un nuevo mensaje.

Se dispone de las siguientes rutinas, ya implementadas:

Rutina	*Descripción*
`inicializaciones()`	Inicializa el *hardware* (pantalla, interrupciones, etc.)
`tareas_independientes()`	Tareas que no dependen del acceso al display LCD (ej. realizar una animación en la pantalla superior); tiempo ejecución < 200 ms
`int wifiReceiveText(char *t)`	Rutina de recepción de un mensaje por la interfaz wifi de la NDS; si hay un nuevo mensaje (desde la última llamada), copiará los bytes de dicho mensaje sobre el string que se pase por referencia (min. 33 posiciones) y devolverá 1; si no hay nuevo mensaje, devolverá 0
`sincro_display()`	Espera a que el display esté preparado para recibir un nuevo comando; como máximo, tardará 41 μs

`strcpy(char *dest, const char *source)`	Copia el string que se pasa por segundo parámetro sobre el string que se pasa por primer parámetro
`swiWaitForVBlank()`	Espera hasta el próximo retroceso vertical
`printf(char *format,...)`	Escribe un mensaje en la pantalla inferior de la NDS

Para realizar el desplazamiento horizontal del display, se pide utilizar la RSI del *timer* 0, que se configurará (por la rutina de inicialización) para que se invoque 8 veces por segundo.

Para transferir los caracteres sobre las dos líneas de la DDRAM, se pide realizar la siguiente rutina específica:

```
void insertar_strings(char str1[], char str2[]);
```

la cual recibe por parámetro y por referencia dos vectores de caracteres, de 32 posiciones cada uno, como mínimo, desde los cuales se copiarán los códigos ASCII de cada carácter sobre las 32 primeras columnas de la DDRAM.

Para sincronizarse con el display, se debe invocar a la rutina `sincro_display()` antes de enviar cualquier comando al display. Esta rutina no retornará hasta que el display no esté preparado, pero se nos asegura que, como máximo, tardará 41 microsegundos en retornar.

Se pide:

Programa principal y variables globales en C, RSI del *timer* 0 y rutina `insertar_string()` en ensamblador.