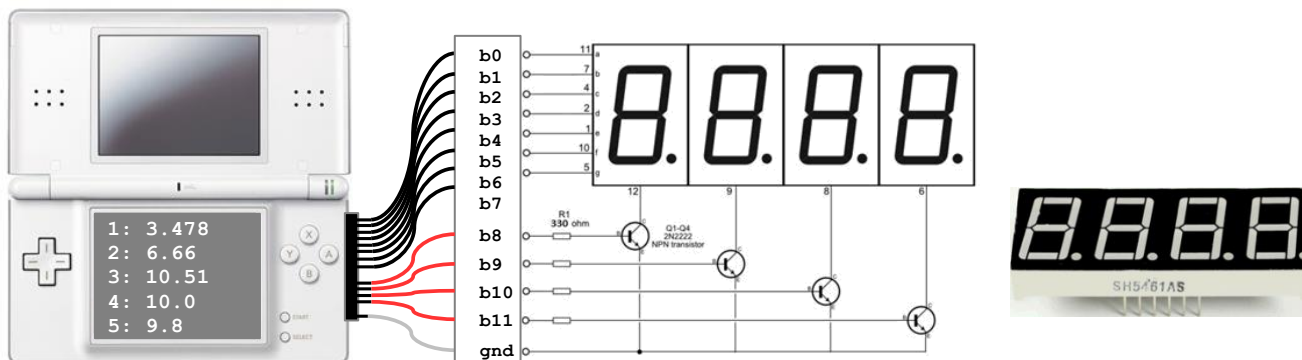
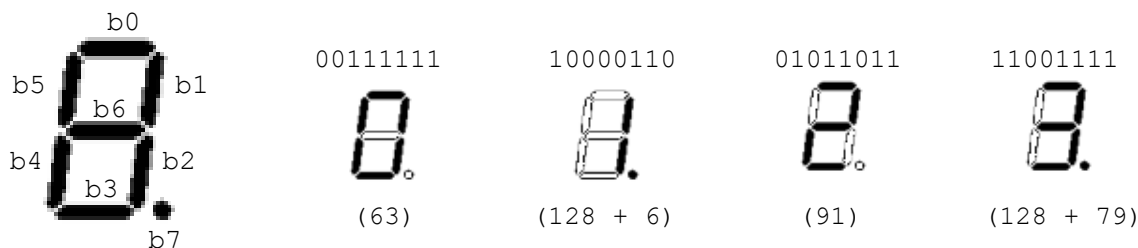


Problema 27: Display numérico 4 dígitos (Ex. 1ª Conv. 2018-19)

Se propone conectar a la NDS un dispositivo para visualizar números de hasta 4 dígitos, compuestos cada uno de ellos por 7 segmentos más un punto decimal. En el siguiente esquema se muestra la conexión lógica de los 12 bits necesarios para gestionar el display, junto con una foto de un dispositivo real (SH5461AS):



La interfaz que vamos a utilizar consiste en un único registro de Entrada/Salida de 16 bits, de nombre simbólico `REG_DATA`. Los 8 bits de menos peso sirven para determinar la activación (1) o no (0) de cada uno de los 7 segmentos más el punto decimal de cada dígito individual. El siguiente gráfico muestra la correspondencia entre los bits (`b0`, `b1`, `b2`, etc.) y los elementos luminosos de un dígito, así como la codificación numérica, en binario y en decimal, de los bits para generar los dígitos del 0 al 3, con el punto decimal activado en los casos 1 y 3:



En la codificación en base diez de los dígitos de ejemplo hemos separado expresamente el valor del punto decimal (128) del valor de los otros 7 bits, ya que la activación de dicho punto decimal es opcional. Para generar los 7 segmentos de cada uno de los diez dígitos decimales podemos utilizar el siguiente vector:






```
//vector de generación de los dígitos decimales en 7 segmentos
unsigned char Vsegments[] = {63, 6, 91, 79, 102, 109, 125, 7, 127, 111};
```

Para poder activar los 7 segmentos más el punto decimal de las 4 posiciones del display, será necesario realizar una multiplexación temporal usando los bits del `b8` al `b11` del registro `REG_DATA`. Esto es, para controlar la activación/desactivación de los elementos luminosos del dígito de más peso (de más a la izquierda), los bits `b11-b8` deberán fijarse a 0001 durante un instante, al mismo tiempo que los bits `b7-b0` deberán enviar el código correspondiente a

dicho dígito. En el instante siguiente, los bits $b_{11}-b_8$ se fijarán a 0010, mientras que los bits b_7-b_0 enviarán la codificación para el dígito de la segunda posición. El proceso se repetirá para las posiciones tercera ($b_{11}-b_8=0100$) y cuarta ($b_{11}-b_8=1000$), para después volver a empezar por la primera posición. En cada instante, solo los elementos luminosos de una posición estarán emitiendo luz. Sin embargo, si el instante es suficientemente largo (≥ 25 ms) al mismo tiempo que la frecuencia de cambio de posiciones es lo suficientemente alta (≥ 25 Hz), el ojo humano no percibirá dicho proceso de multiplexación temporal sino que experimentará la sensación de que todas las posiciones están emitiendo luz al mismo tiempo (persistencia visual).

El programa a implementar deberá capturar un número real (tipo `float`) con un dispositivo independiente al display, por ejemplo, un sensor de distancia, de temperatura, de presión, etc (tarea independiente). La captura se debe realizar durante todo del tiempo que el programa esté en funcionamiento, es decir, se deben ir obteniendo diversas muestras del valor real, aunque no se requiere que el proceso de captura siga una periodicidad estricta. Sin embargo, se pide que se capturen unos 10 valores por segundo y que se realice una media de todos los valores obtenidos durante cada segundo, con el fin de evitar una excesiva fluctuación del valor que se está visualizando.

A cada segundo, el resultado del promedio se convertirá en un vector de 4 valores decimales individuales, correspondientes a los dígitos de cada posición. Esta conversión la realizará una rutina ya implementada (ver tabla de rutinas), la cual también proporcionará el número de dígitos enteros y el número de dígitos decimales que contiene el vector. El siguiente esquema muestra cinco ejemplos de números promedio junto con su correspondiente transformación a vector de dígitos, números de dígitos entero y decimal, y visualización en el display:

Número	Vdigitos	Dent	Ddec	Visualización				
3,478	<table><tr><td>3</td><td>4</td><td>7</td><td>8</td></tr></table>	3	4	7	8	1	3	
3	4	7	8					
6,66	<table><tr><td>6</td><td>6</td><td>6</td><td>x</td></tr></table>	6	6	6	x	1	2	
6	6	6	x					
10,51	<table><tr><td>1</td><td>0</td><td>5</td><td>1</td></tr></table>	1	0	5	1	2	2	
1	0	5	1					
10,0	<table><tr><td>1</td><td>0</td><td>0</td><td>x</td></tr></table>	1	0	0	x	2	1	
1	0	0	x					
9,8	<table><tr><td>9</td><td>8</td><td>x</td><td>x</td></tr></table>	9	8	x	x	1	1	
9	8	x	x					
	0 1 2 3							

Como se puede observar, es posible que para representar un número se requieran menos de cuatro dígitos ($Dent + Ddec < 4$). En este caso, la visualización temporal de dígitos solo deberá activar las posiciones del display necesarias, empezando siempre por la primera posición de la izquierda.

Para gestionar la visualización temporal, así como para controlar el paso de cada segundo para realizar el promedio de valores capturados, se deberá utilizar la RSI del *timer* 0, programada con un periodo de 25 milisegundos.

Para realizar este programa se dispone de las siguientes rutinas ya implementadas:

<i>Rutina</i>	<i>Descripción</i>
<code>inicializaciones()</code>	Inicializa el <i>hardware</i> (pantalla, interrupciones, etc.); no inicializa el <i>timer</i> 0
<code>inicializar_timer0(unsigned short freq)</code>	Inicializa el <i>timer</i> 0 para que genere interrupciones a la frecuencia especificada por parámetro, en hercios.
<code>float capturar_dato()</code>	Captura y devuelve el valor del sensor; tiempo ejecución mínimo = 15 ms, tiempo ejecución máximo = 25 ms
<code>convertir_numero(float valor, unsigned char *Vdigitos, int *Dent, int *Ddec)</code>	Convierte un valor real en una serie de 4 dígitos decimales (redondeando el valor de entrada), que se guardan en el vector <i>Vdigitos</i> a partir de la primera posición del vector; devuelve por referencia el número de dígitos enteros (<i>Dent</i>) y el número de dígitos decimales (<i>Ddec</i>); tiempo ejecución máximo = 800 μ s
<code>swiWaitForVBlank()</code>	Espera hasta el próximo retroceso vertical
<code>printf(char *format,...)</code>	Escribe un mensaje en la pantalla inferior de la NDS

Se sugiere el uso de las siguientes variables globales, además del vector `Vsegment[]` descrito anteriormente:

```
#define MAX_CAPTURAS    10           // máximo número de capturas
                                   // (en un segundo)
unsigned char n_vals = 0;           // número de capturas actual
float valores[MAX_CAPTURAS];       // vector de valores capturados

unsigned char Vdigits[4];           // vector de dígitos
unsigned char num_dent;              // número de dígitos enteros
unsigned char num_ddec;              // número de dígitos decimales
```

La variable `n_vals` permitirá contar el número de elementos almacenados en todo momento dentro del vector `valores[]`, cuyo propósito es registrar todos los valores capturados por el sensor en el último segundo. Las variables `num_dent` y `num_ddec` permitirán registrar el formato decimal del número que se descompondrá en dígitos individuales sobre el vector `Vdigits[]`.

Cada segundo, el contenido de `Vdigits[]` se tienen que visualizar por la pantalla inferior de la NDS, junto con el número de segundo actual (valor secuencial). No se deberá escribir

directamente el valor del número promediado porque éste puede contener más decimales de los que se obtienen con la rutina `convertir_numero()`. Tampoco se permite el uso de llamadas a funciones matemáticas como `round()`. Se puede suponer que siempre habrá al menos un dígito entero y un dígito decimal, aunque sean 0.

Se pide:

Programa principal y variables globales adicionales en C, RSI del timer0 en ensamblador.