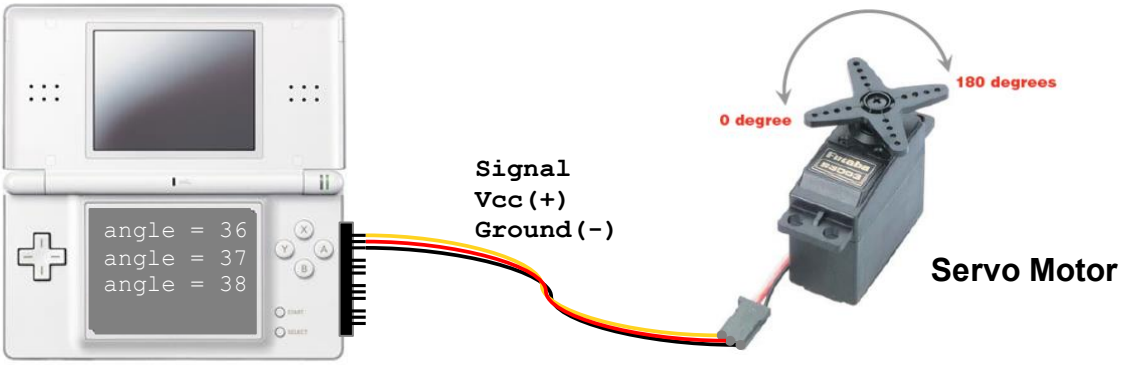
Problema 7: Servomotor

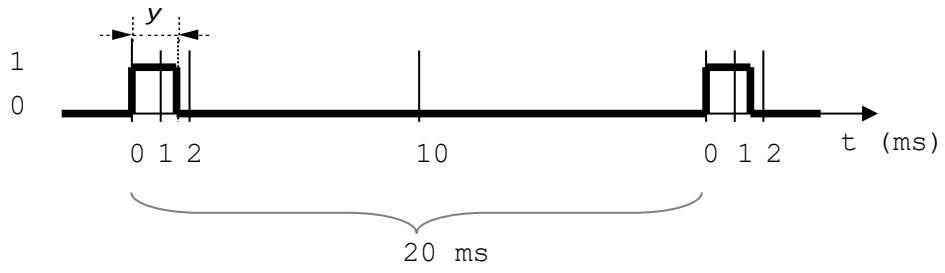
Se propone controlar el ángulo de giro de un servomotor tipo SG90 con la NDS. El programa a realizar permitirá al usuario seleccionar un valor del ángulo entre 0º y 180º:



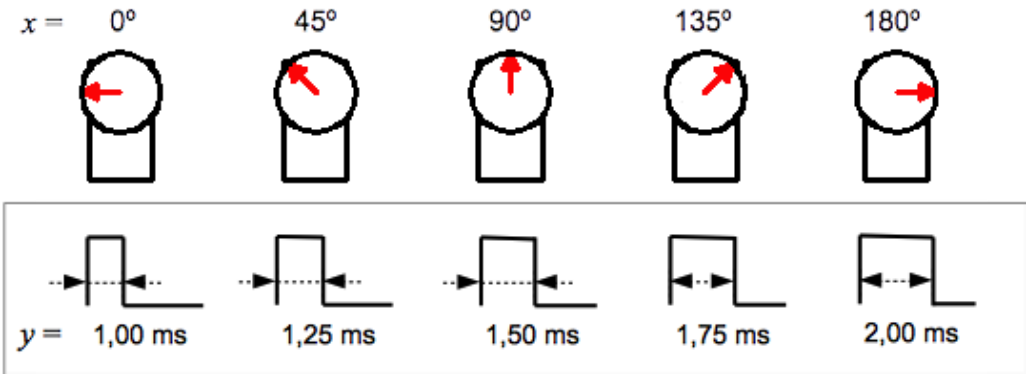
En el esquema se han representado los pines de un adaptador de Entrada/Salida que permite a la NDS controlar hasta 4 servomotores simultáneamente. El controlador de E/S del adaptador se gestiona a través de un registro de E/S denominado REG_SERVO, de 8 bits, de los cuales solo se utilizan los 4 bits de menor peso.

Cada uno de estos 4 bits permite controlar el cable de señal (amarillo) de uno de los servomotores. Los otros dos cables de cada servomotor se conectan a pines de corriente (rojo) y masa (negro) correspondientes. El cable de señal del servomotor a controlar está conectado al bit 3 del registro.

Para regular el ángulo del servomotor, el cable de señal debe emitir un pulso de control periódico de 20 milisegundos, donde el tiempo en que el pulso está a 1 (y) determinará el ángulo de giro (x):



Concretamente, el tiempo debe variar entre 1 ms para 0º y 2 ms para 180º, obteniendo todas las orientaciones posibles según la variación proporcional de dicho tiempo.



En los esquemas anteriores se ha mostrado el estado del servomotor para 5 orientaciones concretas (0º, 45º, 90º, 135º, 180º), aunque se puede conseguir prácticamente cualquier valor (entero) de grados. La siguiente fórmula permite calcular el número de milisegundos del tiempo a 1 (y) a partir del valor del ángulo en grados sexagesimales (x):

$$y = 1 + \frac{x}{180} \ (ms) \quad , \quad x \in [0..180] \ (^o)$$

Como en lenguaje máquina no podemos trabajar con valores reales (coma flotante), podemos adaptar la fórmula anterior para expresar el tiempo en microsegundos, lo cual nos permitirá realizar los cálculos necesarios utilizando solo variables enteras.

El programa a implementar deberá permitir al usuario modificar el ángulo actual de giro en todo momento, utilizando los botones de las flechas derecha e izquierda para aumentar y disminuir el valor del ángulo en unidades, y los botones alternativos de derecha e izquierda (situados en la parte trasera de la consola) para aumentar y disminuir el valor del ángulo en decenas, todo ello sin superar los límites del rango permitido, obviamente. Además, cada vez que se cambie el valor actual del ángulo, éste se deberá mostrar por la pantalla inferior de la NDS.

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 servomotor conectado al bit 3 (ej. controlar otros servomotores según otras fuentes de control, como un sensor de distancia); tiempo ejecución < 100 ms
`scanKeys()`	Captura el estado actual de los botones de la NDS
`int keysDown()`	Devuelve un patrón de bits con los botones activos

`swiWaitForVBlank()`	Espera hasta el próximo retroceso vertical
`printf(char *format,...)`	Escribe un mensaje en la pantalla inferior de la NDS

Para generar la forma del pulso correspondiente al ángulo requerido, se pide utilizar la RSI del *timer* 0, que se configurará (por la rutina de inicialización) para que se invoque cada vez que se active la correspondiente interrupción. El tiempo en que tardará a generarse la interrupción deberá alternar entre el tiempo para el estado del pulso a 1 y el tiempo restante del ciclo para el estado del pulso a 0.

Se propone usar las siguientes variables globales:

```
unsigned char x;              // valor actual del ángulo
unsigned char pulse_state;    // estado actual del pulso
unsigned short y_mic;         // número de microsegundos a 1
```

La variable `pulse_state` permitirá saber el estado actual del pulso, de modo que la RSI del *timer* pueda cambiar alternativamente el estado del bit 3 del registro REG_SERVO. La variable `y_mic` almacenará el tiempo en que el pulso debe estar a 1, en microsegundos.

Para activar el *timer* 0 con un determinado periodo, se pide realizar una rutina específica:

```
void fijar_divfrectim0(unsigned short micros);
```

la cual recibe por parámetro el número de microsegundos del período del *timer*. Por lo tanto, esta rutina debe calcular el divisor de frecuencia correspondiente para que, después del tiempo especificado, se active la interrupción correspondiente. Como frecuencia de entrada, se sugiere utilizar la F/64.

Para realizar las divisiones desde lenguaje ensamblador se puede utilizar la rutina de la BIOS `swi 9` (entrada: R0 = numerador, R1 = divisor; salida: R0 = cociente, R1 = resto, R3 = cociente sin signo), que puede tardar entre 5 y 20 microsegundos.

Se pide:

Programa principal en C, RSI del *timer* 0 y rutina `fijar_divfrectim0()` en ensamblador.