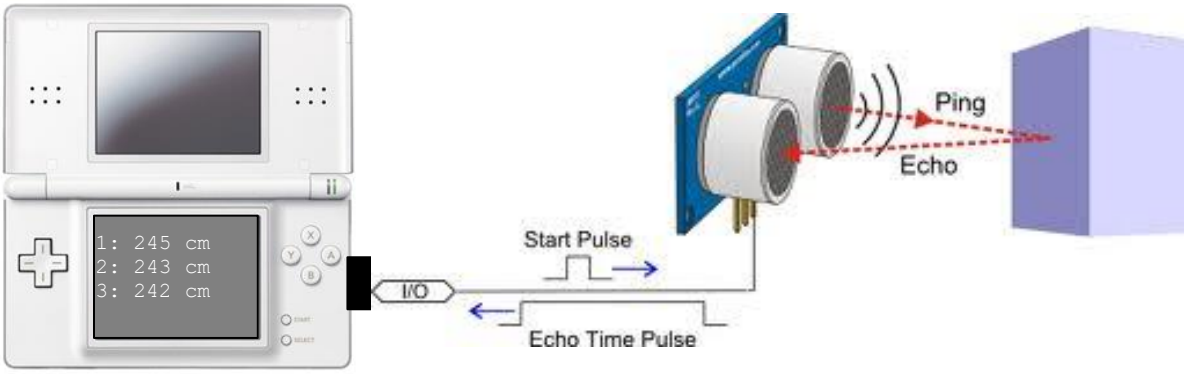
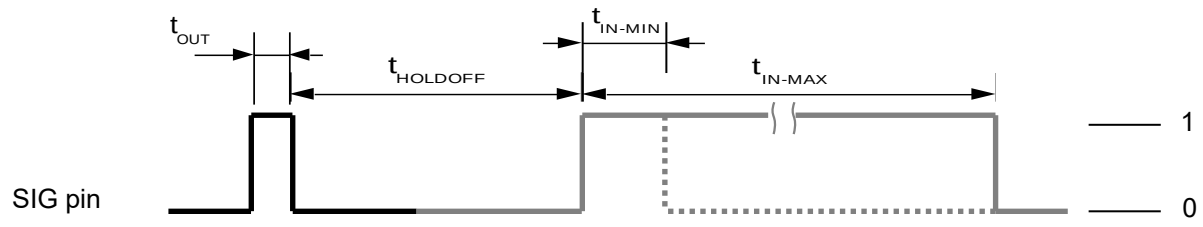
Problema 12: Sensor de distancia (Ex. 1ª Conv. 2013-14)

Se propone controlar con la NDS un sensor de distancia por ultrasonidos PING)))™, de la empresa *Parallax*(R):



Cuando el computador envía un pulso eléctrico de inicio (*Start Pulse*, t_{OUT}) por el único cable de datos disponible (*SIG pin*), el dispositivo PING))) emite una pequeña ráfaga de impulsos ultrasónicos, los cuales rebotan en los objetos cercanos en forma de ecos. Después de emitir la ráfaga ($t_{HOLDOFF}$), el dispositivo activa *SIG pin* (1) hasta que detecta el primer eco, momento en el cual desactiva *SIG pin* (0). A continuación se muestra un cronograma esquemático del proceso de emisión-recepción de la ráfaga de ultrasonidos:



▬▬	Host	*Start Pulse*	t_{OUT}	2 µs (min), 5 µs (típico)
——	PING)))	*Echo Holdoff*	$t_{HOLDOFF}$	750 µs
		Echo Time Pulse Minimum	t_{IN-MIN}	115 µs
		Echo Time Pulse Maximum	t_{IN-MAX}	18.5 ms
		Delay before next measurement		200 µs (min)

Midiendo el tiempo del pulso que genera el dispositivo (*Echo Time Pulse*, t_{IN}) es posible saber la distancia del objeto más cercano. Suponiendo una velocidad del sonido de 340 m/s, el

tiempo mínimo (t_{IN-MIN}) corresponderá a 2 cm, el tiempo máximo (t_{IN-MAX}) corresponderá a 315 cm, y un tiempo intermedio corresponderá a una distancia proporcional entre estos dos valores. Si el primer objeto estuviera a menos de 2 cm o a más de 315 cm, el dispositivo generaría el pulso de tiempo mínimo o el de tiempo máximo, respectivamente.

El dispositivo se conectará a la NDS mediante el puerto de cartuchos de juegos GBA ROM. La señal *SIG pin* se podrá leer y escribir a través del bit 0 del registro `0x040001A2`. Además, cada vez que este bit pase de 1 a 0, se activará la interrupción 13 (IRQ_CART), ya sea cuando el bit lo modifica la propia NDS (pulso de inicio) o cuando el bit lo modifica el dispositivo (pulso de tiempo de eco).

El programa de control, además de realizar ciertas tareas independientes, debe generar el pulso de inicio, detectar el tiempo de eco, calcular la distancia al objeto más cercano en función de dicho tiempo y escribir por pantalla dicha distancia, todo ello continua e indefinidamente. Delante de cada distancia se indicará su número de medida (ver pantalla de la NDS en el primer gráfico).

Se dispone de las siguientes rutinas, ya implementadas:

Rutina	*Descripción*
`inicializaciones()`	Inicializa el *Hardware* (pantalla, interrupciones, etc.)
`tareas_independientes()`	Tareas que no dependen del cálculo de distancias (ej. control del movimiento de un robot); tiempo de ejecución < 1 s
`swiWaitForVBlank()`	Espera hasta el próximo retroceso vertical
`cpuStartTiming(int timer)`	Inicia un cronómetro de precisión usando el *timer* que se pasa por parámetro (0, 1 o 2) y el siguiente *timer*
`cpuGetTiming()`	Devuelve el contaje de tics del cronómetro desde que se inició (entero de 32 bits)
`printf(char *format, ...)`	Escribe por pantalla la información especificada
`startPulse()`	Genera un pulso de inicio de 5 microsegundos

Para poder contar tiempo con precisión hay que utilizar las rutinas `cpuStartTiming()` y `cpuGetTiming()`, las cuales permiten controlar dos *timers* encadenados que contarán tics a la frecuencia base de la NDS (≈ 33,5 MHz). Estas dos rutinas tardan menos de 5 microsegundos en ejecutarse.

Para poder realizar el programa de control sin tener que usar la calculadora, a continuación se muestra la equivalencia entre los tiempos de referencia del cronograma y el número de tics correspondiente:

t_{OUT}:	5 µs	≈	168 tics;
$t_{HOLDOFF}$:	750 µs	≈	25.135 tics;
t_{IN-MIN}:	115 µs	≈	3.854 tics;
t_{IN-MAX}:	18,5 ms	≈	620.009 tics;
retardo mínimo entre mediciones:	200 µs	≈	6.703 tics;

El cálculo de la distancia se tendrá que implementar dentro de una rutina escrita en lenguaje ensamblador, que recibirá por parámetro el número de tics correspondientes al tiempo del pulso de eco (t_{IN}) y devolverá la distancia correspondiente, en centímetros:

```
int calcular_distancia(int t_in);
```

Para realizar las divisiones que sean necesarias se tendrá que utilizar la rutina BIOS `swi 9` (entrada: R0 = numerador, R1 = divisor; salida: R0 = cociente, R1 = resto, R3 = cociente absoluto). Cada división puede tardar entre 5 y 20 microsegundos.

Se pide:

Programa principal en C, RSI del dispositivo y rutina `calcular_distancia()` en ensamblador.