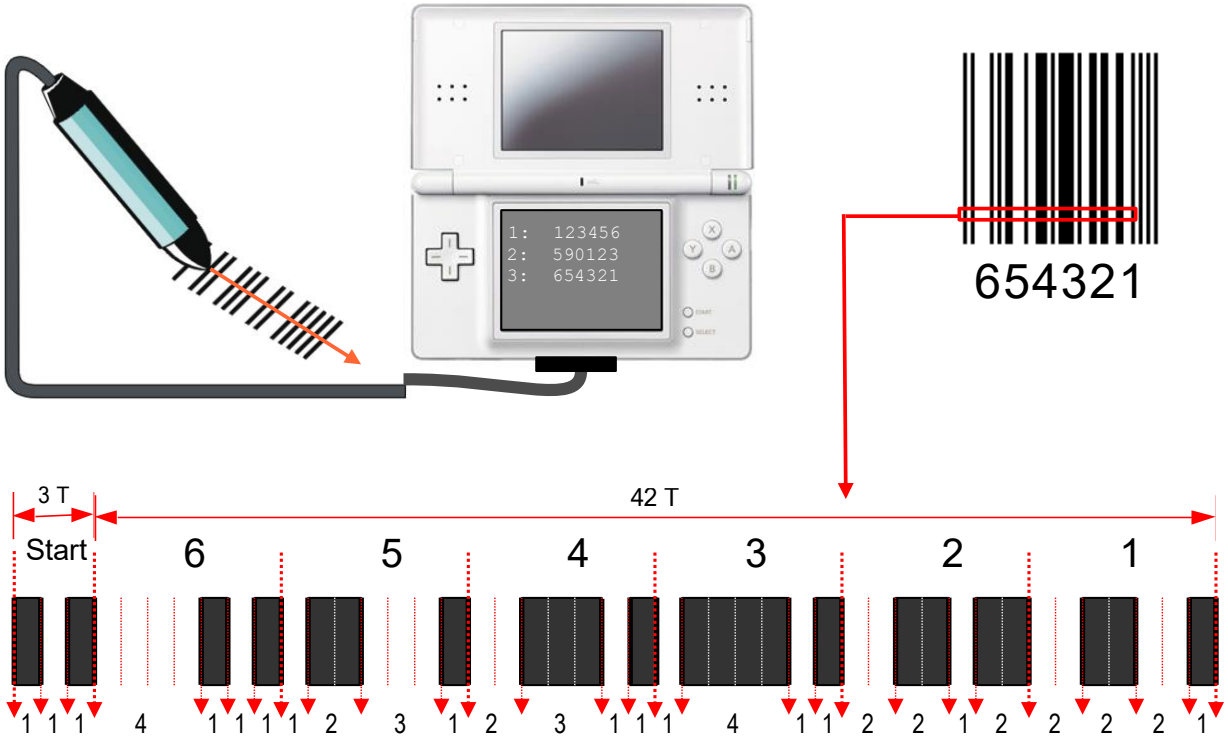
Problema 11: Lector de códigos de barras (Ex. 1ª Conv. 2012-13)

Se propone trabajar con un lector de códigos de barras de tipo “lápiz”:



Este dispositivo periférico no presenta ningún registro, simplemente, generará una interrupción específica (IRQ_CART) cada vez que su haz de luz detecte un cambio de intensidad, es decir, cada paso de claro a oscuro o de oscuro a claro. Estos cambios se producirán cuando el usuario pase el lápiz por encima del código de barras, a una velocidad más o menos constante. En el gráfico del ejemplo, cada interrupción está indicada con una flecha hacia abajo.

Todos los códigos empiezan por una marca de inicio o *Start*, que son dos barras negras separadas por un espacio en blanco. Cada uno de estos tres elementos (2 barras + 1 espacio) presenta una anchura de referencia que llamaremos **anchura unitaria**.

A continuación aparecen las barras y espacios que codifican los dígitos del número del código de barras. Cada dígito se codifica con dos espacios y dos barras, donde cada elemento puede presentar una anchura de 1, 2, 3 o 4 veces la anchura unitaria. La suma de las anchuras de los cuatro elementos de cada dígito siempre será igual a 7 veces la anchura unitaria. Para simplificar, vamos a suponer que siempre se leerán números de 6 dígitos.

La anchura de cada barra o espacio se convertirá en un tiempo (absoluto) según la velocidad de movimiento del lápiz. El programa a realizar tendrá que obtener los tiempos absolutos correspondientes a las anchuras de las barras y espacios. Denominaremos `T` al tiempo absoluto que corresponderá a la anchura unitaria. Dividiendo los tiempos absolutos por `T`, obtendremos unos tiempos relativos para cada barra y cada espacio, que siempre serán valores entre 1 y 4,

independientemente de la velocidad del lápiz. Por ejemplo, la secuencia del gráfico anterior tendrá que proporcionar los siguientes tiempos relativos:

{1, 1, 1, 4, 1, 1, 1, 1, 2, 3, 1, 2, 3, 1, 1, 1, 4, 1, 1, 2, 2, 1, 2, 2, 2, 2, 1}

Se dispone de las siguientes rutinas, ya implementadas:

Rutina	*Descripción*
`inicializaciones()`	Inicializa pantalla e interrupciones
`tareas_independientes()`	Tareas que no dependen de la lectura del código de barras actual (ej. enviar por Wifi el último código)
`swiWaitForVBlank()`	Espera el retroceso vertical
`cpuStartTiming(int timer)`	Inicia un cronómetro de precisión usando el *timer* que se pasa por parámetro (0-2) y el siguiente *timer*
`cpuGetTiming()`	Devuelve el contaje de tics del cronómetro desde que se inició (entero de 32 bits)
`decodificar_codigo(char tiempos[])`	Decodifica un código de barras de 6 dígitos a partir de los tiempos relativos y lo escribe en pantalla

Para obtener el tiempo absoluto de las barras y los espacios hay que utilizar la rutina `cpuStartTiming()` en una interrupción y la rutina `cpuGetTiming()` en la siguiente interrupción, la cual retornará el número de tics transcurridos entre las dos interrupciones. Estas rutinas tardan menos de 5 microsegundos.

Los tics se incrementan a la frecuencia base de la NDS (≈ 33,5 MHz). Si admitimos velocidades del lápiz entre 5 cm/s y 100 cm/s, los tiempos absolutos de la anchura unitaria oscilarán entre los 220.120 y los 11.060 tics. Los tiempos absolutos de toda la secuencia (incluida la marca de inicio) se almacenarán en un vector y, cuando se hayan obtenido todos, se tendrá que realizar su división por el tiempo unitario `T`, utilizando 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 μs. Todas las divisiones se tienen que realizar en una rutina que almacenará los tiempos relativos en otro vector (acceso por referencia):

```
void normalizar_tiempos(int t_abs[], char t_rel[]);
```

Después se tendrán que decodificar los tiempos relativos obtenidos y escribir los dígitos correspondientes, invocando a la rutina `decodificar_codigo()`.

Se pide:

Programa principal en C, RSI del lector y rutina `normalizar_tiempos()` en ensamblador.