

dígitos. En ambos casos se debe almacenar el valor numérico del dígito anterior en un vector global denominado `NTel[]`. Como esta tarea se debe realizar en ambas RSIs, se aconseja implementarla en la siguiente rutina auxiliar:

```
unsigned int capture_digit(unsigned int n_puls);
```

Esta rutina recibe como único parámetro el número de pulsos del dígito marcado y siempre devuelve cero como resultado.

Se propone el siguiente programa principal:

```
#define MAXDIGITS 14
unsigned char NTel[MAXDIGITS]; // vector para almacenar número tel.
unsigned char ind_digit = 0; // índice del dígito actual
unsigned char new_digit = 0; // si vale 1 es que hay nuevos dígitos
unsigned char num_pulses = 0; // número de pulsos actual

void main(void)
{
    unsigned char ind_digit_ant = 0; // índice del dígito anterior
    inicializaciones();
    TIMERO_DATA = ?_a_?; // fijar divisor de frecuencia máximo
    do
    {
        tareas_independientes();
        if (new_digit) // si se han marcado nuevos dígitos
        {
            swiWaitForVBlank();
            if (?_b_?) // si se trata del primer dígito
            {
                printf("Número tel.: ");
            }
            while (ind_digit_ant < ind_digit) // visualizar dígitos pendientes
            {
                printf(?_c_?);
                ind_digit_ant++;
            }
            if (num_pulses == 0) // si se ha terminado la marcación
            {
                printf("\n");
                realizar_llamada(NTel, ind_digit);
                ind_digit_ant = 0;
                ind_digit = 0; // reiniciar proceso de marcación
            }
            new_digit = 0;
        }
    } while (1);
}
```

El programa principal entrelaza la realización de tareas independientes con la visualización

por pantalla de los nuevos dígitos marcados por el usuario. Hay que prever que se pueden haber introducido más de un nuevo dígito mientras se estaban realizando las tareas independientes. Cuando se detecten nuevos dígitos, pero la variable `num_pulses` sea igual a cero, significará que la marcación del número ha terminado y que se puede realizar la llamada telefónica, invocando a la rutina `realizar_llamada()`.

Para simplificar la complejidad del problema, vamos a suponer que la rutina `realizar_llamada()` no finaliza hasta que el usuario cuelga el teléfono, y que el usuario nunca marcará números telefónicos de más de 14 dígitos.

Además, se pide que a cada detección de pulso se active una de las diez luces LED que se han instalado en el teléfono, al lado de cada dígito. Concretamente, se debe acceder al registro `REG_TEL` (16 bits):

	10	9	8	7	6	5	4	3	2	1	0
REG_TEL:	x	x	x	x	0	0	0	0	1	1	x
	:0	:9	:8	:7	:6	:5	:4	:3	:2	:1	

En el esquema anterior se supone que se llevan detectados cuatro pulsos: hay que ir activando los LEDs a medida que se detecta cada nuevo pulso, pero hay que dejar activos los LEDs anteriores, hasta que se detecte un nuevo dígito o se acabe la marcación; entonces hay que apagarlos todos.

Por último, a modo de recordatorio se proporcionan los siguientes datos respecto a la programación del `timer 0`:

- **TIMERO\_DATA (0x04000100):** registro de datos (16 bits) del `timer 0`:
  - se escribe para fijar el valor del divisor de frecuencia,
  - se lee para obtener el valor actual del contador de tics, queirá incrementando desde el valor del divisor de frecuencia hasta cero, momento en el que se disparará la IRQ del `timer` (si las interrupciones están habilitadas) y se reiniciará el contador de tics.
- **TIMERO\_CR (0x04000102):** registro de control (16 bits) del `timer 0`; dispone de los siguientes campos (el resto de bits no se utilizan):
  - bits 1..0: indican la selección de la frecuencia de entrada  
 00 → F/1 (=33.513.982 Hz)      01 → F/64 (=523.656 Hz)  
 10 → F/256 (=130.914 Hz)      11 → F/1024 (=32.728,5 Hz)
  - bit 2: encadena el `timer` con el anterior (la salida del `timer` anterior es la entrada del `timer` actual),
  - bit 6: permite habilitar la generación de interrupciones,
  - bit 7: 0 → `timer` parado, 1 → `timer` en marcha; la secuencia "poner a cero, poner a uno" reinicia el contador de tics.

#### Se pide:

RSI del sensor y rutina `capture_digit()` en lenguaje ensamblador, partes del programa principal marcadas con `?_X_?` en lenguaje C. No es necesario implementar la RSI del `timer`.

### Teléfono vintage

Se propone controlar un teléfono antiguo (analógico) con la plataforma NDS. Concretamente, en este problema se pide detectar la introducción de los dígitos del número de teléfono que se desea marcar. Esta introducción (o marcación) se realiza mediante un dial rotatorio, como el mostrado en la parte central del teléfono de la siguiente figura:

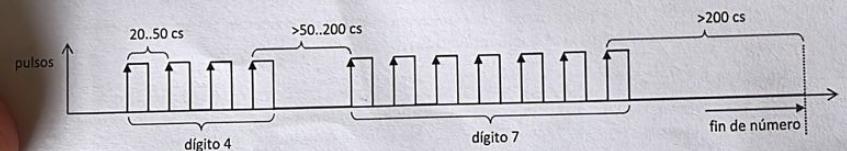


El dial funciona de la siguiente manera:

- el usuario introduce la punta de un dedo en el agujero correspondiente al dígito que desea marcar,
- luego gira el dial en sentido horario, hasta llegar a la cuña metálica que sirve de tope,
- después saca el dedo,
- entonces el dial gira automáticamente en sentido antihorario (gracias a un muelle), hasta que llega a su posición de inicio, preparado para marcar el siguiente dígito.

Para conseguir detectar la selección de los dígitos con la NDS, se ha acoplado una circuitería específica que envía pulsos hacia un pin del puerto de conexión GBA, cuando el dial está girando automáticamente. Este sistema permite generar un número de pulsos igual al dígito marcado, excepto para el dígito 0, que genera diez pulsos.

El siguiente cronograma muestra un ejemplo de introducción de dos dígitos, el 4 y el 7, seguidos de un tiempo de espera de 2 segundos:



En el cronograma anterior se han indicado los rangos de tiempo para distinguir entre pulsos de un mismo dígito y de dígitos diferentes, así como el final del número. Concretamente, el tiempo entre pulso y pulso de un mismo dígito oscilará entre 20 y 50 cs (centésimas de segundo), dependiendo de las características físicas del sistema que hace retroceder el dial. Cuando el tiempo entre el pulso actual y el anterior sea superior a 50 cs, se interpretará que el pulso actual es el primero del nuevo dígito. Si pasan 2 segundos sin introducir nuevos pulsos, se interpretará que el usuario ha terminado de introducir dígitos.

Se dispone de las siguientes rutinas ya implementadas:

Rutina	Descripción
inicializaciones()	Inicializa el hardware (pantalla, interrupciones, etc.)
tareas_independientes()	Tareas que no dependen de la detección de los pulsos de un nuevo número de teléfono, por ejemplo, gestión de una agenda telefónica (tiempo de ejecución < 1s)
swiWaitForVBlank()	Espera hasta el próximo retroceso vertical
printf(char *format,...)	Escribe un mensaje en la pantalla inferior de la NDS
realizar_llamada(unsigned char numtel[], unsigned char ndig)	Se comunica con la central telefónica para efectuar la marcación de los dígitos almacenados en el vector que se pasa por referencia, con el número de dígitos pasado por valor.

Para detectar los pulsos se usará la IRQ\_CART (del cartucho GBA), donde está conectado el cable que envía los pulsos a la NDS, de modo que cada pulso generará una activación de la correspondiente RSI. Esta RSI, que denominaremos rsi\_sensor(), se debe encargar de contar los pulsos de cada dígito, pero también debe controlar el tiempo entre pulsos.

Para realizar dicho control del tiempo se utilizará el timer 0, configurado con la frecuencia de entrada mínima ( $\approx 32.728.498$  Hz) y con el divisor de frecuencia máximo, de modo que generará una interrupción cada 2 segundos aproximadamente.

Por lo tanto, la RSI del sensor NO podrá utilizar las interrupciones del timer 0, sino que deberá leer el registro de datos de dicho timer. De este modo, podrá obtener el valor actual del contador de tics (de entrada), el cual se carga con el divisor de frecuencia cuando se inicia o reinicia el timer, y se va incrementando hasta llegar a cero. De este modo, si la diferencia del número de tics actual respecto al valor del divisor de frecuencia es superior a 16.364 tics, significará que han pasado un poco más de 50 cs desde el último (re)inicio del timer.

Las interrupciones del timer 0 se usarán para detectar si han pasado 2 segundos desde la última vez que se (re)inició el timer. De este modo, la RSI del sensor debe detectar cuándo se cambia de dígito, mientras que la RSI del timer 0 debe detectar cuándo el usuario ha dejado de marcar