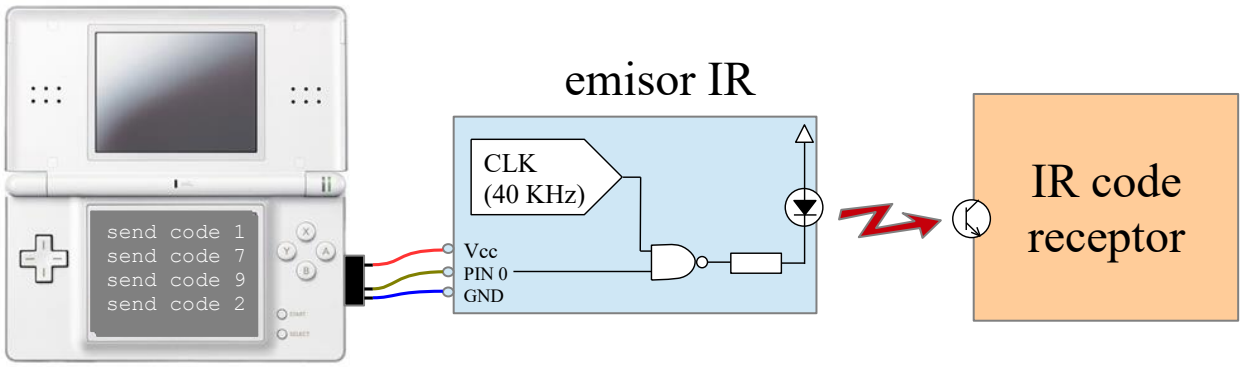
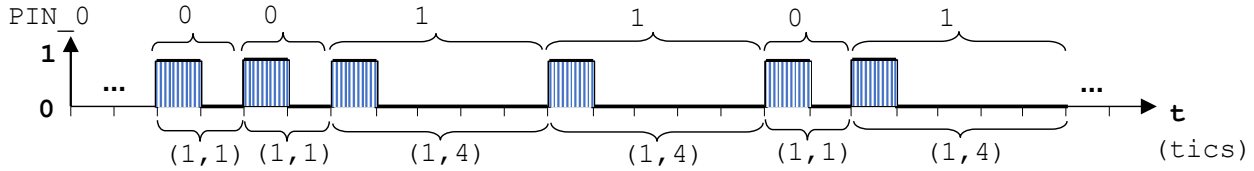
Problema 21: Emisor IR (Ex. 2ª Conv. 2016-17)

Se propone controlar un circuito emisor de luz infrarroja (IR) con la NDS. El programa a realizar deberá permitir al usuario pulsar algunos botones de la NDS para emitir una determinada serie de ráfagas de luz infrarroja, que transmitirá un comando específico (encender/apagar, subir volumen, etc.) a un dispositivo receptor (televisión, reproductor de DVDs, etc.) compatible con el formato de códigos definido por la empresa NEC®:



En el esquema anterior se ha representado la conexión de la NDS con un circuito emisor IR, básicamente con un cable de datos denominado `PIN0` que controlará la transmisión de pulsos de luz a una frecuencia de 40 KHz. El estado del pin se podrá fijar escribiendo un 0 o un 1 en el bit 0 de un registro de E/S de nombre simbólico REG_IR; aunque el registro es de 32 bits, el resto de bits no tiene ninguna función asignada. Debido a la puerta NAND, durante el tiempo en que el PIN0 esté a 1, el LED IR convertirá los pulsos eléctricos generados por el CLK (*CLocK*) en pulsos de luz infrarroja; esta señal periódica se denomina *portadora*. Si PIN0 está a 0, no se enviará la señal portadora. A estos dos estados de enviar y no enviar portadora los denominaremos “ON” y “OFF”, respectivamente.

Para transmitir los códigos de los comandos, la señal portadora se activará y desactivará siguiendo unos determinados patrones de tiempo. Concretamente, se define la unidad 'tic' como el tiempo unitario de referencia. Los tics tendrán una frecuencia de 1.700 Hz, de modo que el tiempo de un tic será aproximadamente de 588 µs.



Para codificar cada bit de información se especificará un par de tiempos expresados en tics, constituidos por un número de tics para el estado "ON" y otro número de tics para el estado "OFF". Un bit de datos 0 se codificará con el par (1, 1), mientras que un bit de datos 1 se codificará con el par (1, 4). Es decir, los dos tipos de bit de datos activan la portadora durante un único tic, pero los bits de datos a 0 la desactivan durante un solo tic, mientras que los bits de datos a 1 la desactivan durante cuatro tics. Esta diferencia en el tiempo de "OFF" es suficiente para que el receptor distinga el valor de cada bit de datos. El cronograma anterior muestra un ejemplo de transmisión de un trozo de secuencia binaria "...001101...".

El programa a implementar deberá testar periódicamente los 10 botones de la NDS que se corresponden con los 10 bits de menos peso del registro REG_KEYINPUT, que son todos los botones menos 'X' e 'Y'. Para cada botón se deberá iniciar la transmisión de una determinada ráfaga de 32 bits de datos (32 pares ON/OFF), precedida de un par ON/OFF de inicio (*Lead In*) i seguida de otro par ON/OFF de final (*Led Out*). Según el formato NEC, el par de tiempos de *Lead In* es (15, 7), mientras que el par de tiempos de *Lead Out* es (1, 59). En resumen, se podrán transmitir hasta 10 comandos diferentes, cada uno de los cuales estará compuesto de 34 pares ON/OFF (1+32+1).

Se dispone de las siguientes rutinas, ya implementadas:

Rutina	*Descripción*
`inicializaciones()`	Inicializa el *hardware* (pantalla, interrupciones, etc.)
`activar_timer0(int freqOut)`	Activa la generación de interrupciones del *timer* 0, a la frecuencia de salida especificada por parámetro
`desactivar_timer0()`	Desactiva la generación de interrupciones del *timer* 0
`scanKeys()`	Captura el estado actual de los botones de la NDS
`int keysDown()`	Devuelve un patrón de bits con los botones activos (estado invertido de bits REG_KEYINPUT, 1 → botón pulsado, 0 → botón no pulsado)
`swiWaitForVBlank()`	Espera hasta el próximo retroceso vertical
`printf(char *format,...)`	Escribe un mensaje en la pantalla inferior de la NDS

Para generar las secuencias de pares ON/OFF de cada comando se proponen las siguientes variables globales:

```
unsigned short VCodes[10][34] =
     {{0x0F07, 0x0101, 0x0101, 0x0101, 0x0104, 0x0101, …, 0x013B},
     {0x0F07, 0x0101, 0x0101, 0x0104, 0x0101, 0x0104, …, 0x013B},
     …
     {0x0F07, 0x0101, 0x0104, 0x0101, 0x0101, 0x0101, …, 0x013B}};

unsigned char current_code;        // índice código actual
unsigned char current_pair;        // índice par ON/OFF actual
unsigned char state;               // estado actual (1=ON, 0=OFF)
unsigned char tics;                // tics pendientes
```

La matriz `VCodes[10][34]` almacenará las 10 secuencias de 34 pares ON/OFF, donde cada par se codifica como un *halfword*, con los 8 bits altos para el tiempo del estado ON y los 8 bits bajos para el tiempo del estado OFF. Por ejemplo, un par (15, 7) se codifica como `0x0F07`, un par (1, 1) se codifica como `0x0101`, etc.

La variable `current_code` permitirá memorizar el índice del código actual (de 0 a 9). La variable `current_pair` permitirá memorizar el índice del par ON/OFF actual (de 0 a 33). La variable `state` permitirá memorizar el estado actual (0 o 1) del par actual. La variable `tics` permitirá memorizar cuantos tics faltan para que termine el estado actual.

En general, el programa principal debe consultar periódicamente los bits de los botones de la NDS. Cuando detecta uno pulsado, debe activar el *timer* 0 a una frecuencia de 1.700 Hz, inicializar las variables globales de control, activar el bit 0 del REG_IR y esperar a que termine la transmisión de toda la secuencia de 34 pares asociada al código del botón; no hay que realizar ninguna tarea independiente. Además, por la pantalla inferior de la NDS se debe escribir el mensaje "`send code x`", donde '*x*' debe indicar el índice del código a transmitir (de 0 a 9). Si se pulsan varios botones a la vez, solo se deberá transmitir el código de un único botón.

Por su parte, la RSI del *timer* 0 debe decrementar el número de tics pendientes del estado actual; cuando este número llegue a cero se debe pasar al siguiente estado y, si es necesario, pasar al siguiente par, actualizando el número de tics pendientes y el bit 0 del registro IR según el nuevo estado actual del par actual del código actual. Cuando se haya transmitido el último par del código actual, se debe detener el *timer* 0.

Según el formato NEC, los bits de información de todo comando siempre contienen 16 unos y 16 ceros, con lo cual se puede calcular el total de tics de cualquier secuencia, incluyendo los pares de *Lead In* y *Lead Out*. Este total es de 194 tics, que corresponden a un tiempo aproximado de 0,1141 segundos.

Para obtener el número de tics del estado actual del par actual del código actual, se pide realizar una rutina específica:

```
unsigned char obtener_tics(unsigned short codes[][34],
                           unsigned char ccode,
                           unsigned char cpair,
                           unsigned char cstate);
```

la cual recibe por parámetro la referencia de la matriz de códigos y los valores del código actual, par actual y estado actual, respectivamente. La rutina devuelve como resultado el número de tics correspondiente.

Se pide:

Programa principal en C, RSI del *timer* 0 y rutina `obtener_tics()` en ensamblador. No es necesario copiar la variables globales indicadas en este enunciado en las hojas de la solución.