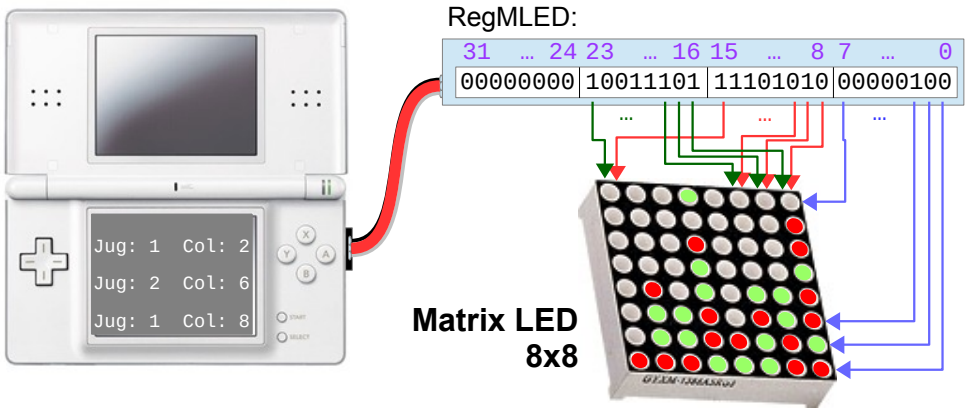
Problema de Examen (1ª Conv.): matriz 8x8 LEDs

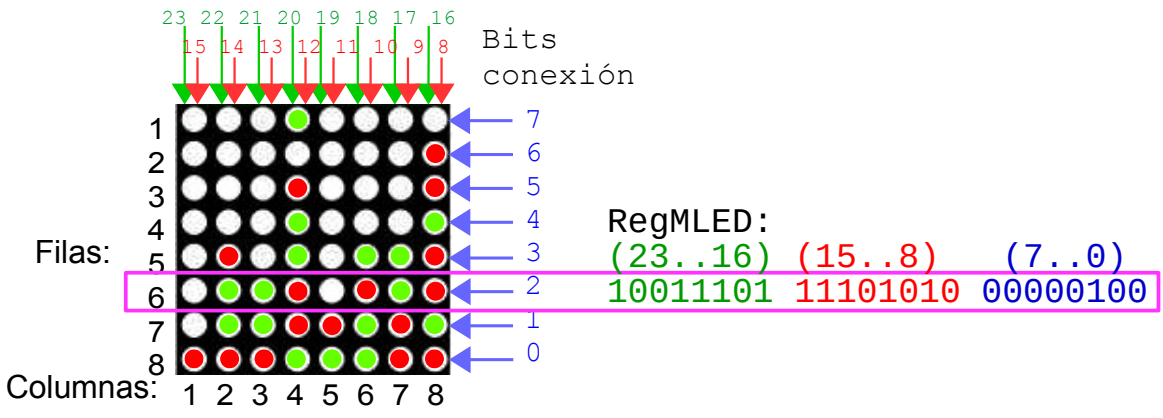
Se propone conectar una interfaz para controlar una matriz de 8x8 LEDs bicolor (1588ABEG-5) con la NDS, con el fin de implementar el juego del 4 en ralla:



La interfaz dispone de un único registro de Entrada/Salida de 32 bits, de nombre simbólico `RegMLED`, pero solo se utilizarán los 24 bits de menor peso, los cuales están divididos en los siguientes campos:

Campo	Bits	Función
`Columnas en Verde`	23..16	Activan (=0) o desactivan (=1) el color verde de los 8 LEDs de la fila seleccionada por el campo de Filas
`Columnas en Rojo`	15..8	Activan (=0) o desactivan (=1) el color rojo de los 8 LEDs de la fila seleccionada por el campo de Filas
`Filas`	7..0	Activan (=1) la fila que se tiene que iluminar en cada momento

El problema de gestionar 64 LEDs, cada uno con 2 colores, es que resulta muy costoso conectar los 128 cables necesarios para el control individual del color de cada LED. Por este motivo el dispositivo dispone de un cable por cada fila más dos cables por cada columna, de manera que se pueda aplicar la técnica del "barrido" por filas, que consiste en enviar corriente eléctrica sobre un único cable de fila y utilizar el voltaje de los 16 cables de las columnas para indicar el estado del color de los 8 LEDs de esa fila. El siguiente esquema muestra un ejemplo de cómo se controla el color de los LEDs de la fila 6 (bit 2 = 1):



Si se recorren todas las filas secuencialmente, fijando el color de las 8 columnas de cada fila, repitiendo el proceso a una frecuencia suficientemente alta (≥ 30 Hz), el ojo humano no detecta el truco de que solo una fila está emitiendo luz en todo momento, sino que tiene la sensación de que toda la matriz iluminada completamente (persistencia visual).

La interfaz no genera interrupciones, de modo que se pide utilizar la RSI del *timer* 0, programada a una frecuencia de 240 Hz, para realizar el barrido periódico de la matriz.

Por otro lado, para implementar la dinámica del juego del 4 en ralla se pide utilizar la RSI del *timer* 1, programada a una frecuencia de 3 Hz, que se encargará de generar el efecto de caída de las fichas (ver más adelante).

El programa a implementar controlará las fichas de dos jugadores. El jugador 1 llevará las rojas y el jugador 2 las verdes. El tablero consistirá en 7 filas (de la 2 a la 8) por 8 columnas. Por turnos, cada jugador tendrá que seleccionar la columna sobre la que quiere soltar su ficha. Para esto, sobre la fila superior (la 1) se mostrará una ficha del color del jugador que tiene el turno, posicionada inicialmente en la columna 4. El jugador podrá cambiar la posición (columna) de la ficha con las teclas `KEY_RIGHT` y `KEY_LEFT`. El jugador podrá soltar la ficha pulsando `KEY_SELECT`, si la columna actual no está llena con otras fichas. Cuando se realice la selección de la columna, aparte de escribir dicha selección por la pantalla inferior de la NDS, la ficha empezará a caer hacia las filas inferiores hasta que llegue a la fila 8 o hasta que encuentre otra ficha en la posición inferior. Entonces el programa debe comprobar si hay cuatro en ralla; en caso negativo la partida continúa, en caso positivo se declara el vencedor por la pantalla de la NDS y la partida finaliza; el programa ya no hace nada más, hasta que se reinicie la NDS (no hay tareas independientes).

Para realizar este programa se dispone de las siguientes rutinas ya implementadas:

Rutina	*Descripción*
`inicializaciones()`	Inicializa el *hardware* (pantalla, interrupciones, etc.)
`scanKeys()`	Captura el estado actual de los botones de la NDS
`int keysDown()`	Devuelve un patrón de bits con los botones activos
`int comprobar_4(unsigned char *tablero, int nfil, int ncol)`	Comprueba si existe 4 en ralla en un tablero de `nfil` x `ncol` posiciones, devolviendo 1 en caso afirmativo y 0 en caso negativo (tiempo de ejecución > 0,0001 s)
`mostrar_ganador(int jugador)`	Muestra por pantalla el mensaje de resultado de la partida
`swiWaitForVBlank()`	Espera hasta el próximo retroceso vertical
`printf(char *format,...)`	Escribe un mensaje en la pantalla inferior de la NDS

En la pantalla inferior de la NDS del esquema inicial se muestran diversos ejemplos de la salida de texto que indican la selección de cada columna, que siguen el siguiente formato:

```
Jug: i (tabulador) Col: j       (1 ≤ i ≤ 2)  (1 ≤ j ≤ 8)
```

Se sugiere el uso de las siguientes variables globales:

```
unsigned char matriz[8][8];     //valores de los LEDs, en cada posición:
                                //   = 0 → vacía (apagado)
                                //   = 1 → ficha jugador 1 (rojo)
                                //   = 2 → ficha jugador 2 (verde)
unsigned char turno = 1;        // número de jugador actual
unsigned char fil = 1, col = 4; // fila y columna de la ficha que
                                // se está colocando
```

Para la implementación del programa principal y la RSI del *timer* 1, puede ser conveniente usar una variable que indique en qué fase se encuentra el juego en cada instante (seleccionar columna, caída ficha, comprobar ganador, final partida).

A cada activación de la RSI del *timer* 1, si el programa se encuentra en la fase de caída, la ficha solo bajará una posición (fila inferior), en el caso de que sea posible, obviamente. Si la ficha se encuentra en la fila 8 o si se detecta otra ficha en la fila inferior a la actual, se habrá terminado la fase de caída.

A cada activación de la RSI del *timer* 0 habrá que acceder a una de las filas de la matriz y generar los 24 bits de información para el refresco de esa fila. Obviamente, este proceso se debe realizar periódicamente para todas las filas de la matriz.

Se pide:

Programa principal y variables globales en C, RSIs de los *timers* en ensamblador.

Solución Problema Examen 1ª Convocatoria

```
unsigned char matriz[8][8];     //valores de los LEDs, en cada posición:
                                //   = 0 → vacía (apagado)
                                //   = 1 → ficha jugador 1 (rojo)
                                //   = 2 → ficha jugador 2 (verde)
unsigned char turno = 1;        // número de jugador actual
unsigned char fil = 1, col = 4; // fila y columna de la ficha que
                                // se está colocando
unsigned char fase = 0;         // fase actual del programa:
                                //   = 0 → seleccionar columna
                                //   = 1 → caída de ficha
                                //   = 2 → comprobar ganador
                                //   = 3 → final de partida
unsigned char f_refresco = 0;   // índice de fila que se tiene
                                // que refrescar en el display

void main()
{    int keys;
     inicializaciones();        // la matriz estará toda a ceros,
     matriz[0][col-1] = turno;  // excepto la posición de salida
     do
     {  swiWaitForVBlank();     // relajar bucle principal
        switch (fase)
        {
          case 0:               // selección de columna
                  scanKeys();
                  keys = keysDown();
                  if ((keys & KEY_LEFT) && (col > 1))
                  {  matriz[0][col-1] = 0;
                     col--;
                     matriz[0][col-1] = turno;
                  }
                  if ((keys & KEY_RIGHT) && (col < 8))
                  {  matriz[0][col-1] = 0;
                     col++;
                     matriz[0][col-1] = turno;
                  }
                  if ((keys & KEY_SELECT) && (matriz[1][col-1] == 0))
                  {
                     printf("Jug: %d\tCol: %d\n", turno, col);
                     fase = 1;
                  }
                  break;
          case 1:                       // esperar caída de ficha
                  break;
          case 2: if (comprobar_4(&matriz[1][0], 7, 8))
                  {                     // detección de ganador
                     mostrar_ganador(turno);
                     fase = 3;                  // programa finalizado
                  }
                  else                  // cambio de turno
                  {  turno = 3 - turno;
                     fil = 1; col = 4;          // inicializaciones para
                     matriz[0][col-1] = turno;  // la nueva selección
                     fase = 0;
                  }
                  break;
          }
        }
     } while (1);
}
```

```
@;RSI del timer 1: se activa periódicamente (3 Hz) para realizar la
@;    caída de la ficha, una fila por cada activación de la RSI, hasta
@;    que detecte otra ficha en la fila inferior o hasta que llegue a
@;    la última fila.
RSI_timer1:
      push {r0-r8, lr}

      ldr r0, =fase
      ldrb r1, [r0]         @;R1 = valor de la fase del programa
      cmp r1, #1
      bne .LRSIt1_final     @;salir si no estamos en fase de caída

      ldr r2, =fil
      ldrb r3, [r2]         @;R3 = valor de fila
      cmp r3, #8
      beq .LRSIt1_fincaida  @;finalizar caída por llegar a última fila

      ldr r4, =col
      ldrb r4, [r4]         @;R4 = valor de columna
      sub r4, #1            @;ajustar índice de columna a rango (0..7)
      ldr r5, =matriz       @;R5 = dirección inicial de la matriz de juego
      add r6, r4, r3, lsl #3     @;R6 = desplazamiento (col-1 + fil*8)
      ldrb r7, [r5, r6]     @;comprobar posición inferior (fil indica el
                            @;    desplazamiento de la fila inferior,
                            @;    porque el índice de filas empiezan en 0
                            @;    y la variable fil empieza en 1)
      cmp r7, #0
      bne .LRSIt1_fincaida  @;finalizar caída por detectar ficha debajo

      ldr r8, =turno
      ldrb r8, [r8]         @;R8 = valor del turno
      strb r8, [r5, r6]     @;si se puede bajar la ficha, guardar turno en
      sub r6, #8            @;posición inferior y borrar posición anterior
      strb r7, [r5, r6]     @;    (R7 seguro que contiene un 0)
      add r3, #1            @;bajar una fila
      strb r3, [r2]         @;actualizar variable global de fila
      b .LRSIt1_final

  .LRSIt1_fincaida:
      mov r1, #2            @;cambiar a fase 2 cuando finaliza la caída
      strb r1, [r0]         @;actualizar variable global de fase

  .LRSIt1_final:
      pop {r0-r8, pc}
```

```
@;RSI del timer 0: se activa periódicamente (240 Hz) para realizar el
@;   refresco de una fila de LEDs, en función del contenido de la
@;   variable global 'matriz'.
RSI_timer0:
     push {r0-r7, lr}     @;salvar registros a modificar

     ldr r0, =f_refresco
     ldrb r1, [r0]        @;R1 = fila de refresco actual, rango (0..7)
     ldr r2, =matriz
     add r2, r1, lsl #3   @;R2 = dirección inicial de fila de refresco

     mov r3, #0           @;R3 = índice de columna
     mov r4, #0x80        @;R4 = máscara de activación de una columna
     mov r5, #0xFF        @;R5 = patrón de bits de columnas en rojo
     mov r6, #0xFF        @;R6 = patrón de bits de columnas en verde
  .LRSIt0_for:
     ldrb r7, [r2, r3]    @;R7 = valor en la matriz (columna actual)
     cmp r7, #1           @;comprobar si es jugador 1
     biceq r5, r4         @;activar bit (=0) en patrón de rojos
     cmp r7, #2           @;comprobar si es jugador 2
     biceq r6, r4         @;activar bit (=0) en patrón de verdes
     mov r4, r4, lsr #1   @;desplazar máscara
     add r3, #1           @;   y avanzar índice de siguiente columna
     cmp r3, #8
     blo .LRSIt0_for      @;cerrar bucle

     mov r7, r6, lsl #16  @;R7 = patrón de bits verdes (desplazados)
     orr r7, r5, lsl #8   @;añadir patrón de bits rojos (desplazados)

     rsb r3, r1, #7       @;invertir f_refresco a núm. bit (R3 = 7 - R1)
     mov r4, #1
     mov r4, r4, lsl r3   @;crear máscara de activación del bit de fila
     orr r7, r4           @;añadir al total de bits
     ldr r2, =RegMLED
     str r7, [r2]         @;escribir registro de Entrada/Salida

     add r1, #1           @;pasar a la siguiente fila de refresco
     cmp r1, #7           @;si supera el último índice (7),
     movhi r1, #0         @;volver a empezar por la fila 0
     strb r1, [r0]        @;actualizar variable global 'f_refresco'

     pop {r0-r7, pc}      @;restaurar registros modificados
```