FINAL INFORMATICA II	Total Hojas	Finaliza	30 Julio 2014
		20:45hs (duración 1:30hs)	

Nombre y Apellido del alumno	Nº Legajo	Calificación	Docente a cargo / Firma	

Se la ha pedido participar en el desarrollo de un reproductor mp3 basado en un micro cortex M3 (u 8051) que posee un dispositivo de audio capaz de decodificar, amplificar y emitir por los parlantes el archivo de audio que se le solicite por teclado, y que se encuentra almacenado en una tarjeta SD que posee su propia electrónica de control.

Ud. deberá ocuparse de:

PUNTO 1) Ingresar por medio de un conjunto de switchs configurados como una matriz de 2x6, el número de tema a reproducir (se supone un máximo de 99 temas).

1	3	5	7	9	
2	4	6	8	0	ENTER

Teclado de 2x6

1. Para conformar el número de tema se debe oprimir necesariamente la tecla ENTER.
2. Cada vez que se oprime un número, debe dispararse un temporizador de 10s que, en caso de vencer, descarta la tecla ingresada.
3. Cuando se oprime ENTER se obtiene el valor de las últimas dos teclas validas ingresadas

Sabiendo que el systick (Timer0), ha sido configurado a 500us, se le pide codificar en C:

- la función que realice la lectura del teclado (driver y primitiva).
- la función antirebote (se valida tecla luego de 4 verificaciones). [dejarlo para el final si no llega]
- La función main que contemple el algoritmo necesario para determinar que numero de canción se escuchara.
- La/s funciones correspondientes de temporización, invocando apropiadamente la lectura de teclado sabiendo que, por cuestiones de diseño se requiere su lectura cada 5ms.

PUNTO 2)

Al oprimir la tecla ENTER, además de calcular el número de la canción, se debe proceder a la reproducción del tema seleccionado. Un compañero suyo se ocupo de la lógica necesaria para que la controladora de la SD deposite la información a reproducir en un buffer global (uint8_t buf_audio[100]) y de despachar el primer byte. Todo ello se realiza dentro de la función comSerie () de la cual Ud. no debe ocuparse.
Pero si debe realizar la rutina de atención de interrupciones del puerto serie (UART0) para comunicar el buffer y la controladora de audio.

La trama asincrónica será transmitida por interrupciones a 9600 bits/seg (velocidad compatible con la del controlador de la SD que al mismo tiempo está cargando el buffer circular de 100 bytes).

La condición de fin de transmisión será la lectura de un byte de contenido 0xFF.

Nota: NO debe considerar la situación que mientras esté sonando una canción se ingresen nuevas teclas. Asimismo, la función ComSerie() se encargará de poner en 0 a start una vez que haya finalizado la transmisión.

Nota2: La inicialización de la UART0 establece que solo se transmitirá.

El main tendrá el siguiente aspecto:

```
uint8_t buf_audio[100]; uint8_t buff_cancion[2];
#define     NO_KEY    (uint8_t) 0xFF
……..

void main (void) {
    uint8_t tecla, start;
    inicializaciones();                 //no debe ocuparse
    while (1) {
            tecla = Teclado ();
            if (tecla != NO_KEY) {
                ……..
                start = AnalizarTeclas(tecla);
                if (start)
                        comSerie ();      //no debe ocuparse
            }
} }
```

Se le pide codificar en C:

- la funcion de interrupcion de la comunicación serie que asegure la correcta lectura del buffer circular uint8_t buf_audio[100]

Resolución

```
#define     fila0     PORTn,m
#define     fila1     PORTn,m
#define     col0      PORTn,m
#define     col1      PORTn,m
#define     col2      PORTn,m
#define     col3      PORTn,m
#define     col4      PORTn,m
#define     col5      PORTn,m
#define     col6      PORTn,m
#define     NO_KEY            (uint8_t) 0xFF
#define     ACTIVO_BAJO       0
#define     Col0      Get_PIN (col0, ACTIVO_BAJO)
#define     Col1      Get_PIN (col1, ACTIVO_BAJO
#define     Col2      Get_PIN (col2, ACTIVO_BAJO
#define     Col3      Get_PIN (col3, ACTIVO_BAJO

#define     INIC      11
#define     ENTER     10

uint8_t barridoTeclado (void) {
            SetPIN (fila0, 0); SetPIN (fila1, 1);
            if ( (Col0) )             return 1 ;
            if ( (Col1) )             return 3 ;
            if ( (Col2) )             return 5 ;
            if ( (Col3) )             return 7 ;
            if ( (Col4) )             return 9 ;
            if ( (Col5) )             return INIC ;

            SetPIN (fila0, 1); SetPIN (fila1, 0);
            if ( (Col0) )             return 2 ;
            if ( (Col1) )             return 4 ;
            if ( (Col2) )             return 6 ;
            if ( (Col3) )             return 8 ;
            if ( (Col4) )             return 0 ;
            if ( (Col5) )             return ENTER ;
}
//------------------------------------------------------------------------
//primitiva de teclado
extern  uint8_t buff_key;
extern  uint8_t buff_cancion[2];

uint8_t Teclado (void){
            uint8_t key = NO_KEY;

            if (buff_key != NO_KEY) {
                        key = buff_key;
                        buff_key = NO_KEY;
                        tiempoTecla = 0; //reseteo bandera
            }
            return key;
}


//------------------------------------------------------------------------
extern volatile uint8_t buff_key;
#define REBOTES   4

/*función que atiende el debounce
Se utiliza aquí la técnica de verificar CANT_REBOTES veces la tecla antes de darla por buena. CANT_REBOTES es empírico. Depende del HW
Solo escribiremos en bufKey el valor de una tecla, si la variable cont vale cero, luego de haber sido cargada por la macro CANT_REBOTES, lo que significará que se ha verificado la misma tecla CANT_REBOTES veces.*/
void debounceTeclado ( uint8_t CodigoActual )
{
            static uint8_t CodigoAnterior = NO_KEY;
            static uint8_t EstadosEstables;

            if( CodigoActual == NO_KEY ) {
                        CodigoAnterior = NO_KEY;
                        EstadosEstables = 0;
                        return;
            }

            if( EstadosEstables == 0  ) {
                        CodigoAnterior = CodigoActual;
                        EstadosEstables = 1;
                        return;
            }

            if( CodigoActual != CodigoAnterior ) {
                        CodigoAnterior = NO_KEY;
                        EstadosEstables = 0;
                        return;
            }

            if( EstadosEstables == REBOTES ) {
                        key = CodigoActual;
                        EstadosEstables++;
                        return;
            }

            if( EstadosEstables == REBOTES + 1)
                        return;

            EstadosEstables ++;

            return;
}

//------------------------------------------------------------------------
extern  uint8_t buff_cancion[2];
extern uint8_t index;

void systick_Handler (void)
{
            static uint8_t  tiempo = 10;

            tiempo--;
            if (!tiempo){
                  tiempo = 10;
                  debounceTeclado (barridoTeclado);
            }
            if (tiempoTecla) {
               tiempoTecla--;
               if (!tiempoTecla){
                        buff_cancion[0]= buff_cancion[1]=0;
                        index = 0;
               }
            }
}
```

```
//-------------------------------------------------------------------------
uint8_t buf_audio[100]
uint8_t buff_cancion[2];
static uint8_t index = 0;
static uint8_t inx_out;

void main (void) {
    uint8_t tecla;
    inicializaciones();

    while (1) {
        tecla = Teclado ();
        if (tecla != NO_KEY) {
            tiempoTecla = 2000; //para 10s
            start = AnalizarTeclas(tecla);
            if (start)
                comSerie ();
        }
    }
}
```

```
//-------------------------------------------------------------------------
uint8_t  AnalizarTeclas(uint8_t teclaRecibida)
{
   uint8_t numCancion = 0;

    if (teclaRecibida < ENTER) {
        if ( index < 2) {
            buff_cancion[index] = teclaRecibida;
            index++;
        } else {
            buff_cancion[0] = buff_cancion[1] ;
            index = 1;
        }
        return  0;
    }
    if (teclaRecibida ==ENTER) {
        numCancion = buff_cancion[0] + buff_cancion[1] *10;
        index = 0;
        return numCancion;
    }

//-------------------------------------------------------------------------

extern uint8_t buf_audio[];
extern uint8_t inx_out;

void UART0_IRQHandler (void)
{
    uint8_t iir;
    iir = U0IIR;

    if ( iir & 0x02 ) { //THRE
        if (inx_in != inx_out)
            if (buf_audio [inx_out] != 0xFF)
            {
                U0THR = (buf_audio [inx_out++]);
                inx_out %= 100; //garantizo el buffer circular
            }
    }
```