Práctica 10

Uso de Temporizadores/Contadores del uC ATmega1280

Objetivo: Mediante esta práctica el alumno aprenderá la programación y uso básico del Temporizador 0 y 2 del microcontrolador ATmega1280.

Material:
- Computadora Personal (con AVR Studio)
- Tarjeta T-Juino.
- Programa Terminal.

Equipo: - Computadora Personal con USB, AVRStudio y WinAVR

Teoría:
- Programación del Timer 0 del microcontrolador
- Programación del Timer 2 del microcontrolador
  (Diagrama, Funcionamiento, Registros de configuración y operación)

Desarrollo:

1) Crear y compilar proyecto:

   a) Utilice el programa AVR Studio para crear un proyecto llamado **Prac10** donde los archivos del proyecto deberán ser los correspondientes al listado 2 (Prac10.c) y listado 3 (Timer0.c) **Nota:** todos los archivos (*.c y *.h) deberán estar en el mismo directorio del proyecto.

   b) Compile el proyecto (realizar correcciones en dado caso que existan)

   c) Una vez compilado el proyecto, el archivo (Prac10.hex) deberá ser cargado al T-Juino. Este archivo se encuentra en la carpeta llamada "default" generada por el compilador en el directorio del proyecto (p.e. C:\uPyuC\Prac10\default).

Listado 1. Timer0.h

```
#ifndef _TIMER0_H
#define _TIMER0_H

#include <inttypes.h>

/* Función para inicializar el Timer0 y  generar      */
/* la temporización de 1 Sec.                         */
void Timer0_Ini ( void );

/* Función para verificar bandera del segundo         */
uint8_t Timer0_SecFlag ( void );

#endif  /* _TIMER0_H */
```

Listado 2. Prac10.c

```
#include <avr/io.h>
#include "Timer0.h"

/* incluir lo necesario para usar UART0 */

int main(){

    /* llamar a función para inicializar puertos E/S */
    /* llamar a función para inicializar UART0       */

    Timer0_Ini();               /* Inicializar Timer0 para 1 sec.*/
    while(1){                   /* == main loop ==               */
        if( Timer0_SecFlag() ){ /* ¿ha pasado 1 Segundo?         */
            /* instrucciones para encender LED */
             UART0_puts("1 segundo\n\r");
            /* instrucciones para apagar LED */
         }
    }                           /* fin del loop principal        */
    return 0;                   /* <-- no se llega aqui          */
```

Listado3. Timer0.c

```
#include <avr/interrupt.h>
#include <inttypes.h>

static volatile uint8_t SecFlag;

void Timer0_Ini ( void ){

    TCNT0=0x06;     /* Inicializar valor para el timer0     */
    TCCR0A=0x00;    /* inicializa timer0 en modo 0 (normal) */
                    /* Inicializar con fuente de osc. Int.  */
    TCCR0B=0x03;    /* con Prescalador 64                   */
    TIMSK0=0x01;    /* habilita interrupcion del Timer0     */
    sei();          /* habilita interrupciones (global)     */
}

uint8_t Timer0_SecFlag ( void ){

   if( SecFlag ){
       SecFlag=0;
       return 1;
   }
   else{
       return 0;
   }
}

ISR (TIMER0_OVF_vect){     /* TIMER0_OVF_vect                          */
   static uint16_t  mSecCnt;
   TCNT0+=0x06;            /* reinicializar Timer0 sin perder conteo   */
   mSecCnt++;              /* Incrementa contador de milisegundos      */
   if( mSecCnt==1000 ){
       mSecCnt=0;
        SecFlag=1;         /* Bandera de Segundos                      */
   }
}
```

d) Una vez cargado el programa, la tarjeta T-Juino deberá estar encendiendo un LED (en algún puerto) cada segundo. Este programa utiliza como base de tiempo el temporizador **Timer0** inicializado en modo 0 (normal) para que se genere una interrupción cada un milisegundo aproximadamente. Esto ocurre cuando el Timer se desborda (pasa de valor FF a 00) y se activa **TOV0**. La rutina de servicio de interrupción (**ISR***: Interrupt Service Routine*) asociada a la interrupción lleva un conteo de los milisegundos en las variable **mSecCnt.** Una vez que el conteo llega a 1000 entonces se inicializa a cero para nuevamente llevar dicho conteo, además otra variable tipo bandera llamada **SecFlag** se activa para indicar que ha transcurrido un segundo.

Modificaciones a realizar al programa:

a) Realice las cambios necesarios para manejar el mismo esquema de tiempo base del **Timer0** pero ahora utilizando el modo **CTC** del temporizador.

b) Cambiar la lógica de la ISR para solo implementar un contador (de 32 bits) de milisegundos. Implementar la función:
uint32_t *millis*(**void**)**:**
La cual retorna el conteo actual de milisegundos desde que inició el sistema.

c) Diseñe e implemente la función **void** *UART0_AutoBaudRate***(void)**, la cual ajusta el baud rate dependiendo de la velocidad del dato recibido, tomando como base la duración del bit de inicio (Start Bit) del dato, **suponiendo que el bit menos significativo será '1'**. Esta función deberá funcionar dentro del rango de **8,000** a **200,000** Bauds.
Nota: Hacer uso del Timer0 para contabilizar el periodo.

Parte2: Uso del Temporizador/Contador 2

Descripción: Añadir al proyecto el **Listado 4** y **5**; y modifique los archivos necesarios para generar en una base de tiempo de 1 segundo usando el temporizador 2 (**Timer2)** con el cristal de 32.768 KHz externo como fuente de oscilación.

Para la implementación considere:

a) El **Listado 4** (Timer2.h) muestra la función prototipo de inicialización del **Timer2** y función de verificación de la bandera indicadora del segundo.

b) Diseñe e implemente la función **Timer2_Ini(uint8_t baseT)** que tiene como objetivo inicializar el Timer2 para generar desbordes a una base de tiempo determinada. Es decir, la función recibe un parámetro con el cual determina la inicialización para el desborde dado por el parámetro. Los valores válidos para este parámetro son de 1 a 8, por tanto se puede inicializar el temporizador de 1 a 8 segundos como base de tiempo.

Ejemplo de uso: Timer2_Ini(4); /* inicializar Timer 2 para generar */
/* hacer Flag igual 1 cada 4 seg. */

d) Modifique el programa para que muestre en la PC vía puerto serie (UART0). El formato del reloj deberá ser tipo 24Hrs, (**hh:mm:ss**). Reutilizar las funciones de la Práctica anterior.

e) Reutilizar las funciones de la Práctica 5:
void Clock_Ini(<parámetros>): función para inicializar el reloj.
void Clock_Update( ): función para actualizar el reloj (segundos, minutos y horas).
void Clock_Display( ): función para el desplegado del estado del reloj.

d) Utilice el programa Terminal (MTTTY) en la PC con la misma configuración del puerto serie para recibir el mensaje enviado por T-Juino.

c) Sustituya el programa principal de la **parte 1** al del **Listado 6** para verificar el funcionamiento correcto mediante el programa Hyperterminal en la PC.

Listado 4. Timer2.h

```
#ifndef _TIMER2_H
#define _TIMER2_H
#include <inttypes.h>

/* Funcion para inicializar el Timer2 para generar N seg.   */
void Timer2_Ini( uint8_t baseT );

/* Funcion para verificar bandera del segundo        */
uint8_t Timer2_Flag ( void );

#endif  /*  TIMER2 H */
```

Listado 5. Timer2.c

```
#include <avr/interrupt.h>
#include <avr/io.h>
#include <inttypes.h>

static uint8_t  Flag;

void Timer2_Ini( uint8_t baseT ){

/* Configurar Timer2 para usar cristal externo según */
/* secuencia dada por la hoja de especificaciones    */

}

uint8_t Timer2_Flag ( void ){
   if( Flag ){
      Flag=0;
      return 1;
   }
   else{
      return 0;
   }
}

/* Rutina de Servicio de Interrupción  para atender la interrupción */
/* por Comparación del Timer2  (TCNT2 y OCR2 son iguales)           */
ISR(SIG_OUTPUT_COMPARE2A){

       Flag = 1;     /* Activa Bandera para indicar los 1 segundos */

}  /* fin de la rutina de servicio de Interrupción (RSI) */
```

Listado 6. Prac10.c

```
#include <avr/io.h>
#include "Timer0.h"
#include "Timer2.h"

/* incluir lo necesario para usar UART0 */

void UART0_AutoBaudRate(void);

int main(){

    /* llamar a función para inicializar puertos E/S */
    /* llamar a función para inicializar UART0       */

    UART0_AutoBaudRate();
    clrscr();
    gotoxy(5,1);
    UART0_puts("Autobauding done. UBRR0=");
    itoa(UBRR0,str,10);
    UART0_puts(str);
    UART0_puts('\n\r');

    Timer0_Ini();                /* Inicializar contador de millis.*/
    Timer2_Ini(7);               /* Inicializar Timer2 para 7 sec.*/
    Clock_Ini(23,59,50);

    while(1){                    /* == main loop ==               */
        if( Timer2_SecFlag() ){  /* ¿ha pasado un Segundo?        */
            Clock_Update();
            gotoxy(5,2);
            Clock_Display();
            gotoxy(5,3);
            UART0_puts("millis=");
            /* itoa solo convertirá los 16bits menos significativos */
            itoa(millis(),str,10);
            UART0_puts(str);
         }
    }                            /* fin del loop principal        */
    return 0;                    /* <-- no se llega aquí          */
}
```

Después de dejar correr el programa durante unos minutos, por favor responder la siguiente pregunta: ***¿Por qué existe la diferencia en el conteo de milisegundos?*** (Asumiendo que ambos temporizadores fueron configurados correctamente y está no es la causa raíz de la discrepancia)
Estos son los dos osciladores utilizados: XTAL 16MHz, XTAL 32KHz

Comentarios y Conclusiones.

Bibliografía.