Práctica 8

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
(Diagrama, Funcionamiento, Registros de configuración y operación)

Desarrollo:

1) Crear y compilar proyecto:

a) Utilice el programa AVR/Atmel/Microchip Studio para crear un proyecto llamado **Prac8** donde los archivos del proyecto deberán ser los correspondientes a los dados en el repositorio. **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 (Prac8.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).

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 los 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 64 bits) de milisegundos. Implementar la función:
uint64_t *millis*(**void**)**:**
La cual retorna el conteo actual de milisegundos.

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.

d) Implementar funciones relacionadas con el manejo del reloj:

void *Clock_Ini*(uint64_t millis): función para inicializar el reloj en milisegundos iniciando desde 01/01/1970 00:00:00 (UNIX Epoch).

void *Clock_Date_Display*( ): función para el desplegado del estado del reloj y fecha en formato "***hh:mm:ss dd/mm/aaaa***".

Después de dejar correr el programa durante **una hora**, por favor responder la siguiente pregunta: ***¿Por qué existe la diferencia de tiempo transcurrido?*** (Asumiendo que el temporizador fue configurado correctamente)

Comentarios y Conclusiones.

Bibliografía.