Universidad Autónoma de Baja California

Materia:

Microprocesadores y Microcontroladores.

Reporte#10

Uso de Temporizadores/contadores del uC ATmega1280.

Alumno:

Montoya Valdivia Omar Antonio: 1252892

Profesor:

Jesús García

Programación del Timer 0 del microcontrolador

El timer0 AVR es un temporizador/contador de 8 bits, el registro donde se guardan los valores del timer0 AVR es el registro temporizador contador representado por TCNT0, cuando es utilizado como temporizador, sus valores aumentaran de uno en uno entre 0 y 255 con cada ciclo de reloj, por ejemplo si el oscilador con el que está funcionando el microcontrolador AVR es de 1MHz, entonces el registro TCNT0 aumentará una unidad en 1 us, si el registro TCNT0 se incrementa en 100 unidades habrán transcurrido 100us; cuando es utilizado como contador el temporizador AVR ya no aumenta su valor de uno en uno en cada ciclo de reloj, sino que lo hará mediante el flanco de subida o el flanco de bajada de alguna señal que llegue a un pin especial (identificado como T0)del AVR conectado al timer0 AVR.

Para programar el timer0 AVR como temporizador será necesario colocar todos los bits del registro TCCR0A a cero, esto es TCCR0A=0, en realidad esto no es necesario ya que el registro se inicializa automáticamente a 0, este registro será útil cuando se utilice el timer0 AVR en modo comparación y para la modulación de ancho de pulso PWM.

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

El registro TCCR0B es el que permitirá utilizar el timer0 como temporizador:

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

El timer0 AVR cuenta con lo que se conoce como el prescaler esto hace que la frecuencia de trabajo FCPU se divida por este prescaler, con lo que se logra que el timer0 AVR tarde un poco mas en aumentar su valor en una unidad; el prescaler puede tomar el valor de 1, 8, 64, 256 o 1024; estos valores se eligen programando los bits 0, 1 y 2 del registro TCCR0B, los bits 7 a 3 en este caso se pondrán a 0

Al utilizar los prescaler se tiene la ventaja de lograr tiempos mas largos para cada incremento en una unidad del registro TCNT0 del timer0 AVR, y por lo tanto realizar temporizaciones más largas mientras el timer0 AVR incrementa sus valores.

De acuerdo a los tiempos que se quieran obtener al utilizar el timer0 AVR, habrá que utilizar el prescaler adecuado así como inicializar el registro TCNT0 también con un valor adecuado, no olvidar que este registro es de 8 bits y por lo tanto solo puede contener valores enteros entre 0 y 255.

El timer0 tiene 4 modos de funcionamiento que se pueden configurar programando sus registros asociados:

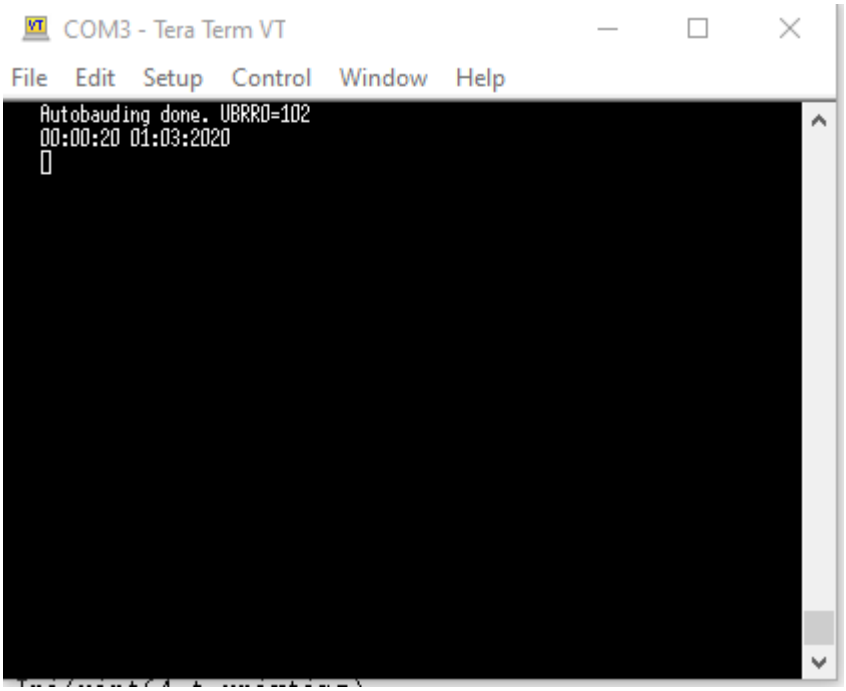
- Modo Normal: El timer cuenta desde 0 a 255 y se desborda reiniciando la cuenta. Puede generar interrupción al desbordarse o cuando la comparación del conteo concuerde con un valor determinado.
- Modo CTC: En este modo el timer0 es reiniciado a 0 cuando una comparación entre el timer y un valor determinado coincide. Opcionalmente puede configurarse para que al haber una coincidencia, genera una interrupción o cambie el estado de un pin.
- Modo Fast PWM: Este modo permite generar una onda PWM de alta frecuencia. El timer cuenta desde 0 a 255 y reinicia la cuenta. Con cada cuenta el valor del timer0 se compara con un valor determinado que cuando coinciden cambia el estado de uno de los pines de salida PWM, y cuando se reinicia el timer este pin vuelve a cambiar su estado.
- Modo Phase Correct PWM: Este modo ofrece una onda PWM de alta resolución a diferencia del modo Fast PWM. El timer cuenta hacia adelante y hacia atrás antes de hacer el cambio de estado del pin PWM, es decir cuenta de 0 a 255 al llegar a 255 cuenta de 255 a 0, obteniendo una salida PWM más limpia pero de menor frecuencia

Programación del Timer 2 del microcontrolador

El timer2 es muy similar a su hermano timer0 sin embargo a pesar de ser tanta similares en muchos sentidos, el timer2 presenta algunas diferencias. este temporizador no funciona como contador, solo como temporizador, y puede ser usado en modo comparador, y también en modo PWM. El timer2 como temporizador tiene dos opciones, el timer2 AVR puede trabajar con el mismo cristal o reloj con el cual trabaja el microcontrolador, o puede trabajar con un cristal o reloj independiente conectado exteriormente a los pines TOSC1 y TOSC2, en este caso se dice que trabaja en forma asíncrona al sistema.

Después de dejar correr el programa durante una hora, por favor responder la siguiente pregunta:

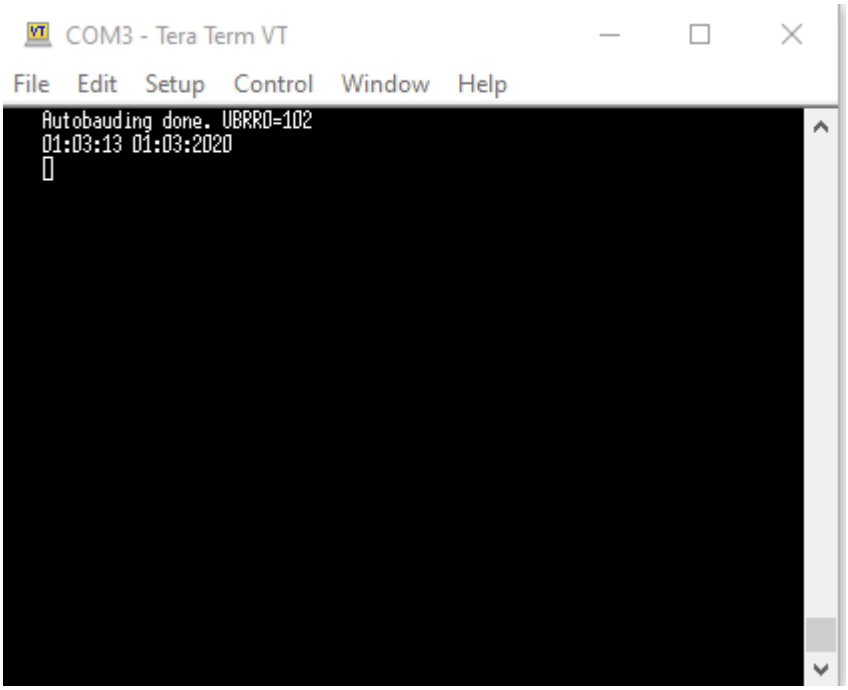
Al iniciar ambos relojes estos son los siguientes datos

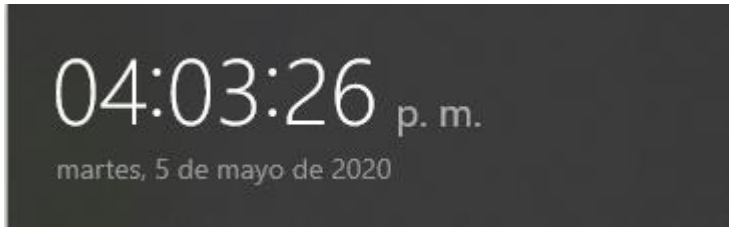


El reloj del ATmega marca **00:00:20 01:03:2020**

El reloj de mi sistema operativo marca **03:00:30 05:05:2020**

Al dejar pasar 01:02:56





Ahora al hacer la resta de ambos tiempos:

Para el caso del ATMEGA 2560

$$\mathbf{01:03:13 - 00:00:20 = 01:02:53}$$

Para el caso del sistema operativo **01:02:56**

¿Por qué existe la diferencia de tiempo transcurrido? (Asumiendo que el temporizador fue configurado correctamente)

Porque no se calculan exactamente los segundos en el microcontrolador debido a que por cada segundo que se calcula pues no son exactamente los segundo entonces esos pequeños pulsos extras que se van acumulando hasta poco a poco ir desfasando el reloj. En este caso al pasar aproximadamente una hora, el reloj se desfaso 3 segundos por lo que, si se hubiera dejado el reloj por más de 20 horas, el reloj estaría desfasado 1 hora debido a los pequeños intervalos de error que presentan, unos ciclos de reloj de más o de menos se van acumulando conforme pasa el tiempo.

Conclusión:

Al realizar la practica comprendí el uso de temporizadores para el Atmega 2560 si como configurarlos y su modo de operación como un contador, cabe recalcar que no se pudo utilizar el Timer2 debido a que la tarjeta no contaba con él, pero se entendió el concepto de cómo funcionan ambos. También se entendió el manejo de interrupciones del temporizador como implementarlas en C para conseguir un reloj. Esto aprendido se puede aplicar para cuando se quiera realizar un sistema embebido que controle ciertas situaciones cada cierto tiempo.

Fuentes:

Timer2 AVR temporizador - MICROCONTROLADORES. (2019). Retrieved from http://microcontroladores-mrelberni.com/timer2-avr-temporizador/

alfreedom, V. (2019). AVR programación en C – 09 Timer/Counter0 del ATmega16/32

parte 1 (Modo Normal). Retrieved from https://vidaembebida.wordpress.com/2014/08/17/avr-programacion-en-c-09-timercounter0-del-atmega16-p