Práctica 9
Timers - Parte 2

Islas Bravo, Andrés.
Estudiante - Laboratorio de Microcontroladores.
Departamento de Arquitectura e Ingeniería
Instituto Tecnológico y de Estudios Superiores de Monterrey
Ciudad de México, México
a01339391@itesm.mx

Valverde López, Juan Pablo.
Estudiante - Laboratorio de Microcontroladores.
Departamento de Arquitectura e Ingeniería
Instituto Tecnológico y de Estudios Superiores de Monterrey
Ciudad de México, México
a01656127@itesm.mx

Zhu Chen, Alfredo.
Estudiante - Laboratorio de Microcontroladores.
Departamento de Arquitectura e Ingeniería
Instituto Tecnológico y de Estudios Superiores de Monterrey
Ciudad de México, México
a01651980@itesm.mx

Abstract
This practice uses the LCD display as the visualization medium of state messages of the system. The state of the system depends on switches from where it can be activated either as a pulse counter or a high duty cycle calculator. The main difference reside in the timer configuration, being a external pulses counter and a capture unit, respectively.

I. Introducción

El presente documento aborda lo acontecido en la práctica para lograr el funcionamiento de las dos rutinas. Símbolo gran conceptos de timer los cuales son configurados acorde a lo demandado por cada labor. Se tiene una primer labor en la cual se requiere contar pulsos externos para lo cual justamente hay una función del timer con el nombre " contador de pulsos externos" la cual, como su nombre lo indica, hace el conteo de pulsos los cuales son configurados para ser contados ya sea en flanco de subida y de bajada. Por otra parte, se tiene la labor de contar el tiempo que una señal está en activo, para lo cual se usa un recurso del timer con el nombre de "unidad de captura". Dicho recurso, al igual que el previo, responde a estímulos externos configurables de flanco de subida y bajada, esto dependiendo de su configuración, sin embargo, su principal diferencia recae en que éste estará con el conteo de tiempo activo, haciendo que a la salida tengamos un valor de segundos y no de número de pulsos.

II. Marco Teórico

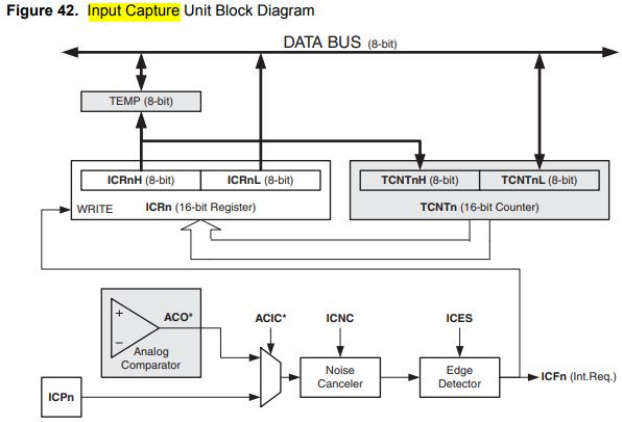
El *Timer* como contador de pulsos externos tiene una gran variedad de aplicaciones en nuestra vida cotidiana. Cada vez que se quiera realizar un conteo de ciertas unidades conforme los pulsos externos lo indiquen en una línea específica del puerto, se puede resolver al manejar el *Timer* como contador de pulsos externos. El ATMega16 puede trabajar de esta manera con el *Timer 0* y el *Timer* 1, esto se logra al configurar la palabra de control, TCCR0(Time Counter Control Register) y TCCR1A|TCCR1B respectivamente, con el selector de reloj en “110” para detectar pulsos por flanco de bajada y “111” por flanco de subida. La señal que entrega los pulsos deberá de conectarse a T0(PB0) para el *Timer 0 y* T1(PB1) para el *Timer 1*. Así mismo, se debe de tomar en cuenta que el número de conteo estará guardado en el registro de 8 bits OCR0(Output Compare Register) para el *Timer 0* y en el registro de 16 bits OCR1A|OCR1B para el *Timer 1*[1].

TABLA I
Configuración del selector de reloj para el timer 0 y timer 1 como contador de pulsos externos

	CSn2	CSn1	CSn0
Contador de pulsos externos por flanco de bajada	1	1	0
Contador de pulsos externos por flanco de subida	1	1	1

La unidad de captura es una herramienta muy útil para obtener información de una señal dada, se puede conocer su tiempo en alto, tiempo en bajo, ciclo útil, periodo y frecuencia de acuerdo con los números de ciclos contados. En el ATMega16, el *Timer* 1 es el único que tiene una unidad de captura por medio del ICP(*Input Capture Pin)* equivalente al PD6. Para la configuración el bit 2 ACIC(*Analog Comparator Input Capture Enable*) del registro ACSR*(Analog Comparator Control and Status Register)* debe estar en “0”, usualmente no se configura porque el microcontrolador arranca para que este mismo bit esté en “0”. En la palabra de control de 16 bits del *Timer 1,*TCCR1A|TCCR1B, el bit 7 ICNC1(Input Capture Noise Canceller) de la parte baja(TCCR1B) de la palabra de control indica si se quiere cancelar el ruido de la señal que se

quiere leer, “0” para inhabilitar y “1” para habilitar; el bit 6 ICES1(Capture Edge Select) de esta misma parte es para seleccionar si se quiere detectar por flanco de subida con “1” o por flanco de bajada con “0”. Al detectar el flanco correspondiente, la bandera ICF1*(Input Capture Flag)* en TIFR(*Time/Counter Interrupt Flag Register*) se levantará y el conteo de 16 bits en TCNT1H|TCNT1L(*Time Counter 1 High*|*Time Counter 1 Low*) pasa capturado en ICR1H|ICR1L(*Input Capture Register 1 HIgh*|*Input Capture Register 1 Low*)[1].



Captura 1: Diagrama de bloques de la unidad de captura en el Timer 1. Tomado desde:
http://ww1.microchip.com/downloads/en/devicedoc/doc2466.pdf

III. Desarrollo

La presente práctica emplea una pantalla LCD de 2 líneas con capacidad de escritura de 16 caracteres, como método de visualización de dos diferentes contadores, los cuales, en su lógica hacen uso de los *Timers* internos del ATMega16. El primer contador empleará el Timer0 en configuración de “contador de pulsos externos” cuyo pin de lectura será el PB0, el cual ya viene especificado por ATMEL. El segundo contador hará uso del Timer1 en su configuración de “unidad de captura”. El programa en su totalidad hace uso de varias subrutinas.Los requerimientos de hardware del sistema son:

- 1 LCD.
- 2 Switches.
- 1 Generador de funciones.

Dentro de la codificación del problema dentro del “main” estará la lógica de sensado de switches e indirectamente se harán llamar a las subrutinas.

Las subrutinas utilizadas en este programa son:

- “conteo”. Propiamente será la subrutina que mantenga la lógica básica de un conteo, en este caso el máximo será 9999, por las condiciones establecidas por el problema.
- “enviopulsos” se enviará el números de pulsos contados a la pantalla lcd.
- “sendlcd_4bits” lógica de envío de datos en 4 bits a la pantalla lcd.
- “initlcd_4bits” lógica de inicialización para la pantalla lcd.
- “delay10msctc” retardo de 10ms empleando el Timer2 en modo CTC.
- “delay40usctc” retardo de 40us empleando el Timer2 en modo CTC.
- “bin_bcd” lógica conversión de binario a bcd.

```
bin_bcd:
    clr r25
    sts 0x60,r25
    sts 0x61,r25
    sts 0x62,r25
    sts 0x63,r25
otro:
        cpi bin_lsb,0
        brne inc_bcd
        cpi bin_msb,0
        brne inc_bcd
        ret
```

Captura 2: Subrutina bin_bcd. Autor: Ms.C. Omar Mata.

```
inc_bcd:
    ldi r26,0
    ldi yl,0x63
    ldi yh,0
        ciclo:
            ld r16,y
            inc r16
            st y,r16
            cpi r16,10
            brne dec_bin
            st y, r26
            dec yl
            cpi yl,0x5f
            brne ciclo
```

Captura 2.1: Subrutina bin_bcd. Autor: Ms.C. Omar Mata

```
dec_bin:
    dec bin_lsb
    cpi bin_lsb,0xff
    brne otro
    dec bin_msb
    rjmp otro
```

Captura 2.2: Subrutina bin_bcd. Autor: Ms.C. Omar Mata

El primer switch iniciará el conteo de la cantidad de pulsos que entren por el pin PB1 cada 2ms y los mostrará en el LCD. El número máximo de pulsos a mostrar en el LCD será de 9999.

El segundo switch iniciará el conteo del “tiempo” en ALTO de una señal cuadrada conectada al pin PD6 y lo mostrará en la LCD. Usar un divisor de 256.

La preparación para la ejecución del programa a realizar responde al esquema que se maneja desde el principio. Esto es:

- **Indicaciones escritura en memoria FlashROM.**
- **Cargado de archivo con definiciones del ATMega 16.**
- **Preparar la pila.** Pues se hará uso de las últimas direcciones de la memoria RAM mediante las instrucciones PUSH/POP.
- **"Definitions" "Equal".** Para claridad del programa y facilidad de modificación.
- **Preparación de puertos como salida.** En cuanto el programa lo requiera.

Es importante recalcar la importancia del uso de los códigos anteriormente desarrollados, específicamente para el conteo esta estructura se mantuvo. Para poder asociar esta subrutina con la subrutina de envío de, en este caso datos, al LCD se decidió que dentro de la lógica del sensado del primer switch, quien era el encargado de activar este conteo, se llamó la subrutina conteo y luego los registros utilizados en esta subrutina fueron cargados individualmente al registro de "dato" que se utilizaba para la subrutina de envío de datos. Este procedimiento es crítico porque se realiza dentro de los 2ms en los que se configuró iba a durar el Timer0.

```
poll2msctc:
    in r16, tifr
    sbrs r16, ocf0
    rjmp pulsonodetectado
    call conteo
    ldi r16,1<<ocf0
    out tifr, r16
    pulsonodetectado:
        sbrs r16, ocf1a
        rjmp poll2msctc
    ;apago timer
    clr r16
    out tccr1b, r16
    out tccr0, r16
    ;apagar ocf0
    ldi r16, 1<<ocf1a
    out tifr, r16
    call enviopulsos
```

Captura 3: Unión de subrutina de conteo con subrutina de envío de datos.

Para el segundo switch que se configuraba en modo unidad de captura, su inicialización y ejecución dependen del estado del switch 2. Los registros a los que se les presta mayor atención en esta circunstancia son los "*Input Capture Register (Parte) Low"* y "*Input Capture Register (Parte) High"* quienes eran leídos a la vez que trrasparasados a los registros conversores de binario a bcd especificados como "bin_lsb" (*Binary Least Significant Byte)* y al "bin_msb" (*Binary Most Significant Byte)* los cuales hacen uso de la subrutina anteriormente mencionada en la captura 3. Seguido de esto, se configura el puntero Y así como se envía los números convertidos a bcd directo a la pantalla lcd.

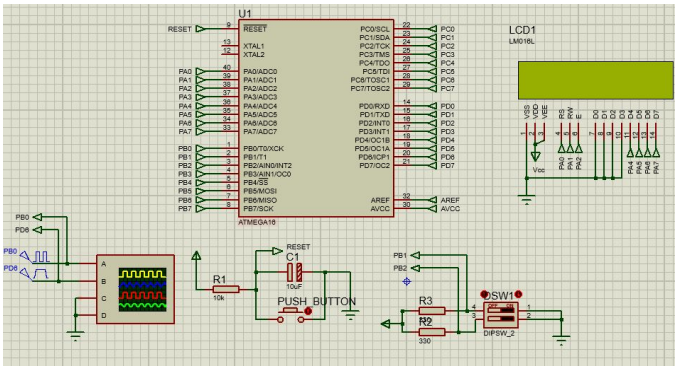
```
set
ldi yh,0;dirección parte alta
ldi yl,0x60
ld r16, y+;registro de contador BCD de 1 a 9, traerse el valor de la memoria RAM
ldi dato, 0x30
add dato, r16
call sendlcd_4bits
ld r16, y+
ldi dato, 0x30
add dato, r16
call sendlcd_4bits
ld r16, y+
ldi dato, 0x30
add dato, r16
call sendlcd_4bits
ld r16, y
ldi dato, 0x30
add dato, r16
call sendlcd_4bits
```

Captura 4: Envío de bcd a la pantalla lcd..

Para comprobar el funcionamiento del sistema, se utilizará Proteus en su versión 8.8, donde se carga el archivo .hex que es generado en el directorio del proyecto.

Comprobación por Proteus.

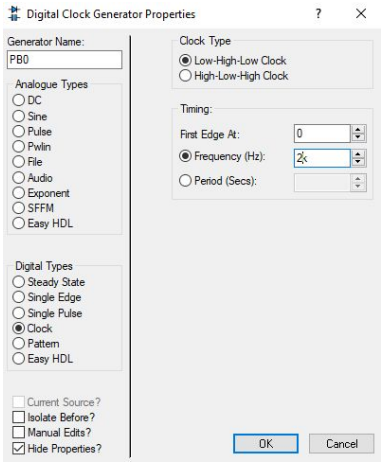
Las salidas para ese sistema será el módulo LM016L, e indirectamente el osciloscopio que fue utilizado para ver los pulsos generados por el generador de funciones.



Captura 5: Sistema que integra al ATMega16, LCD, Switches y Osciloscopio.

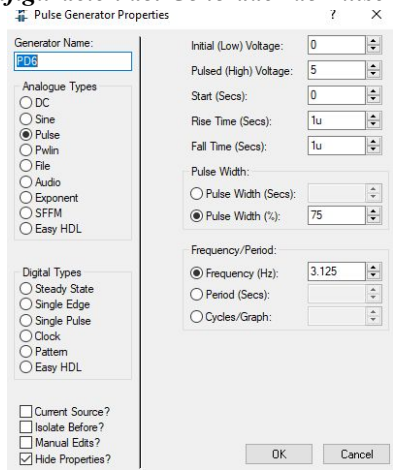
La frecuencia a la cual trabajará el ATMega16 se mantiene, por lo que se configura "CKSEL Fuses" que prepara al MCU a trabajar de cierta forma, en este caso específico "0100", lo que significa que funcionará con su reloj interior, hecho con un arreglo de RC a 8MHz.

A. Configuración del Generador de Funciones a 2k [Hz].



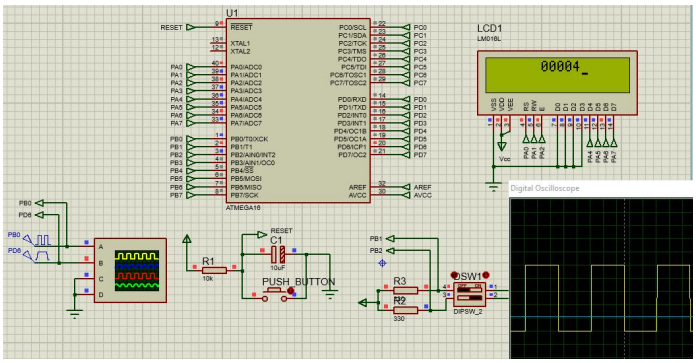
Captura 6: Configuración del Generador de Funciones a 2k Hz.

B. *Configuración del Generador de Pulso a 3.125 [Hz].*

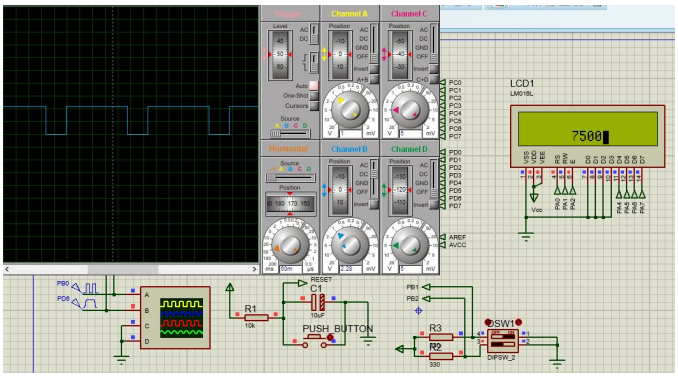


Captura 7: Configuración del Generador de Pulsos. Duty Cycle 75% a 3.125 Hz.

IV. Resultados



Captura 8: Contador de Pulsos Externos cada 2 ms a una frecuencia de 2k Hz.



Captura 9: Contador de Pulso activo en 1, Duty Cycle 75% a 3.125 Hz.

V. Conclusiones

Andrés Islas Bravo:

Esta práctica abordó diversas formas de usar los timers, así mismo, se mantuvo el uso de lsd para así tener una visualización de la información clara e instantánea. Dentro de lo común, se tiende a confundir un timer configurado como contador de pulsos con uno en configuración como unidad de captura, así que el haber realizado esta práctica fue una manera adecuada de diferenciar ambos al estar trabajando en una misma práctica.

Juan Pablo Valverde López:

Se consiguió integrar gran cantidad de rutinas, así como optimizar la inicialización de los diferentes *Timers* utilizados para cumplir los requerimientos de esta práctica. Un momento crítico el conteo de pulsos cada 2ms pues es un tiempo realmente pequeño y el microcontrolador debe realizar varias tareas.

Alfredo Zhu Chen:

Esta práctica me permitió manejar los *Timers* como contador de pulsos externos y como unidad de captura. El objetivo de la práctica se cumplió porque se pudo implementar el conteo de pulsos externos durante 2 ms para cuando se active el primer switch y la mostración del ciclo útil de una señal para cuando se active el segundo switch. Fue muy interesante la actividad ya que se tuvo que utilizar el *Timer* de una manera más amplia. Finalmente, logré probar el código escrito en el programa sobre la simulación de Proteus para observar el funcionamiento de manera visual.

VI. Referencias

[1] ATMEL. (2010). 8-bit AVR Microcontroller with 16K Bytes In-System Programmable Flash – AtMega Atmega16L. [Archivo PDF]. Obtenido desde: http://ww1.microchip.com/downloads/en/devicedoc/doc2466.pdf