Práctica 8
Timers - Parte 1

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 a counter mode and a PWM mode. The last one can be controlled with two buttons increasing and decreasing its duty cycle. A substantial difference from other practices are the complexity and the supervision of more elements which can modify the system.

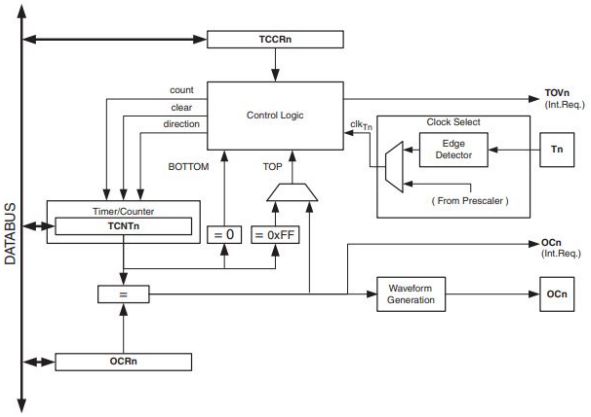
I. Introducción

El presente documento aborda todo lo acontecido durante el control de un LCD. Se muestra todo lo que se necesita para controlar en un modo de 4 bits el display. Esto va desde su configuración inicial donde se le indica que estará trabajando en dicho modo y posteriormente se mencionan todas las instrucciones necesarias para completar su configuración, esto sin olvidar que dicha configuración será mandada en bloques de 4 bits, correspondientes a la parte alta y baja de lo que normalmente sería una instrucción de 8 bits. Asimismo las líneas de códigos presentes corresponden a un mensaje de "Hola Mundo!", esto con una distribución donde el "Hola" se encuentra en la primera fila y el "mundo" en la segunda, ambas centradas. El motivo de este arreglo es meramente para un mayor dominio de las memorias que están presentes en el LCD, y el manejo de una de una mayor cantidad de instrucciones para obtener una mayor familiarización con el instrumento.

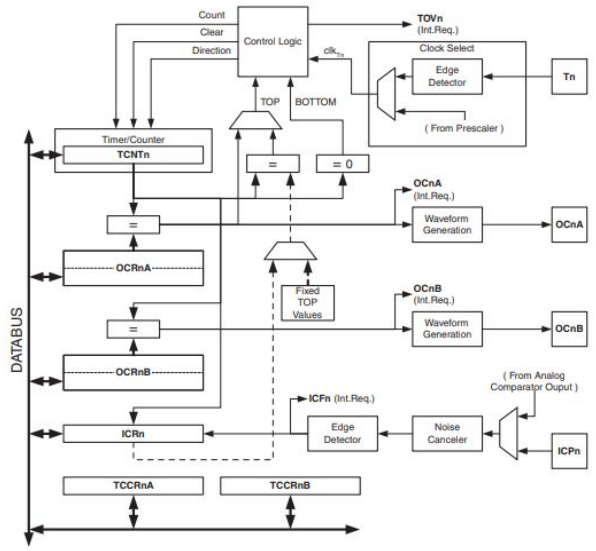
II. Marco Teórico

Los *Timers* o los Temporizadores son elementos importantes en los microcontroladores, ya que en diversas aplicaciones se requieren para temporizar y controlar ciertas partes del propio sistema. Un gran uso de los *Timers* es la realización de retardos, estos se pueden implementar en el programa directo, pero tienen imprecisión debido a la necesidad de considerar los ciclos de máquinas de las instrucciones para alcanzar el tiempo, este problema de imprecisión no se encontraría con los *Timers*. Además, otras de sus aplicaciones se encuentran en la generación de PWM (Modulación por ancho de pulsos) para modular el ciclo útil de una señal o también para contar pulsos externos de alguna entrada proveniente.

El microcontrolador ATMega16 cuenta con 3 *Timers*. El primero es el *Timer 0* de 8 bits y puede trabajar como temporizador o como contador de pulsos externos. Esto se puede configurar con su respectivo registro de control TCCR0. El *Timer 2* es igual de 8 bits que el anterior. A diferencia del primero, este solo puede trabajar como temporizador y no como contador de pulsos externos, de esta forma tendrá más opciones de pre-escalamiento en las elecciones de frecuencias y se puede configurar con su registro de control TCCR2. El *Timer 1* es mucho más potente que los dos anteriores, ya que es de 16 bits y tiene muchas más ventajas. Todos los 3 *Timers* pueden trabajar con PWM, sin embargo, el *Timer 1* permite variar el periodo de la señal con mucha más flexibilidad y con una mayor duración al configurar sus dos registros de control TCCR1A y TCCR1B . Adicionalmente, este temporizador tiene una unidad de captura para realizar mediciones específicas de una señal de entrada.



Captura 1: Diagrama de bloques del Timer 0 y del Timer 2. Tomado desde: http://ww1.microchip.com/downloads/en/devicedoc/doc2466.pdf



Captura 2: Diagrama de bloques del Timer 1. Tomado desde: http://ww1.microchip.com/downloads/en/devicedoc/doc2466.pdf

Los 4 modos principales de los *Timers* en el ATMega16 que se pueden configurar en la palabra de control son: normal, CTC(*Clear TIme on Compare*), *Fast PWM* y *Phase Correct PWM*. En el modo normal, se realiza el conteo con el registro de TCNTn(*Time Counter of Timer n*) y se activa la bandera de TOVn(*Time Counter Overflow of Timer n*) cuando pase del máximo, es decir cuando haya desbordamiento. En el modo CTC, se realiza el conteo con el mismo TCNTn y se activa la bandera OCFn(*Output Compare Flag of TImer n*) cuando TCNTn coincida con un valor guardado en OCRn. Tanto la bandera de OCFn como la de TOVn se pueden encontrar en el reigstro TIFR(*Time/Counter Interrupt Flag Register*). Finalmente, tanto *Fast PWM* como *Phase Correct PWM* crean señales de modulación por ancho de pulso, la diferencia es que el segundo puede crear periodos dos veces más largos que el primero debido a que trabaja con doble pendiente de la señal.

III. Desarrollo

La presente práctica propone el uso de una pantalla LCD de 2 líneas con capacidad de escritura de 16 caracteres, como método de visualización del estado del programa, mostrando 3 mensajes diferentes dependiendo el estado que haya sido activado. Este tipo de pantallas son de las más básicas y cuentan con un microcontrolador integrado para su funcionamiento. Para llevar a cabo esta práctica se debe entender los requerimientos del problema.

El sistema contará con:

- LCD [1].
- Switches [2].
- Botones [2].
- LED [1].

El primer switch activará un contador ascendente con periodo de 10 ms en formato 00:00 que se mostrará en el LCD (centrado en la primera línea). El número máximo será 59:99. Se requiere que el periodo de 10ms esté dado por el Timer1.

El segundo switch activará el Timer0 en modo Fast PWM y la salida OC0 (Pin 3 del puerto B, utilizado en su función especial [PB3]) deberá conectarse a un LED. Mientras esté el switch activado, con dos botones se podrá regular el PWM del LED (modificando OCR0), con un botón se incrementará y con otro se decrementará. Mientras este switch esté activado, la línea 2 del LCD deberá mostrar el mensaje "PWM" centrado. La línea 1 deberá estar vacía. Para probar el funcionamiento del PWM, se observará la señal en un osciloscopio.

Si ningún switch está activo, la LCD deberá mostrar un mensaje de aviso para indicarle al usuario que debe activar un switch.

Para el funcionamiento del LCD se tiene dos subrutinas principales, una es la de inicialización y otra la de envío de los caracterectreres a la pantalla. Es importante señalar que el método de trabajo de esta pantalla será el de 4 bits, es importante hacer esta aclaración pues también cuenta con un modo de trabajo de 8 bits sin embargo, al tratarse del ATMega16, un problema recurrente será la administración del número de puertos para poder llevar a cabo la resolución de problemas. Las dos subrutinas principales son:

- **"initlcd_4bits"**

```
initlcd_4bits:
;initialization
    call delay10msctc
    ;functionset
    clt
    ldi dato,0x28
    call sendlcd_4bits
    call delay10msctc
    ;Display on control
    clt
    ldi dato,0x0f
    call sendlcd_4bits
    call delay10msctc
    ;clear display
    clt
    ldi dato,0x01
    call sendlcd_4bits
    call delay10msctc
ret
```

Captura 3: Rutina de inicialización display.

- **"send_LCD_4bits"**

```
sendlcd_4bits:
push r22
    mov r22, dato
    andi r22,0b11110000 ; mascara nibble superior
    ori r22, 0b00000100;enable a 1
    bld r22, 0
    out porta, r22
    cbi porta,2;genero flanco de bajada
    call delay40usctc
    sbi porta,2;genero flanco de subida
    call delay40usctc
    mov r22, dato
    swap r22
    andi r22,0b11110000 ; mascara nibble superior
    ori r22, 0b00000100;enable a 1
    bld r22, 0
    out porta, r22
    cbi porta,2;flanco de bajada
    call delay40usctc
    sbi porta,2;flanco de subida
    call delay40usctc
pop r22
ret
```

Captura 4: Enviado de instrucciones/datos a display.

En la siguiente figura se observa el esquema o diagrama de flujo del programa. Al tratarse de varias subrutinas que se deben ejecutar, algunas de ellas que fueron utilizadas en prácticas pasadas, serán resumidas para un mejor entendimiento del desarrollo del programa.

```
clt
ldi dato,0b1000011
call sendlcd_4bits
set
ldi dato,0x30
add dato, cont4
call sendlcd_4bits
ldi dato,0x30
add dato, cont3
call sendlcd_4bits
ldi dato,':'
call sendlcd_4bits
ldi dato,0x30
add dato,cont2
call sendlcd_4bits
ldi dato,0x30
add dato, cont1
call sendlcd_4bits
```

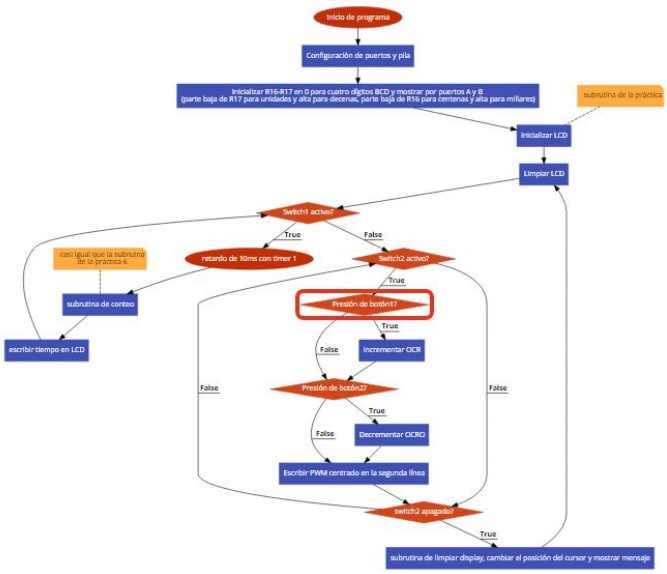


Diagrama 1: Diagrama de Flujo Subrutinas

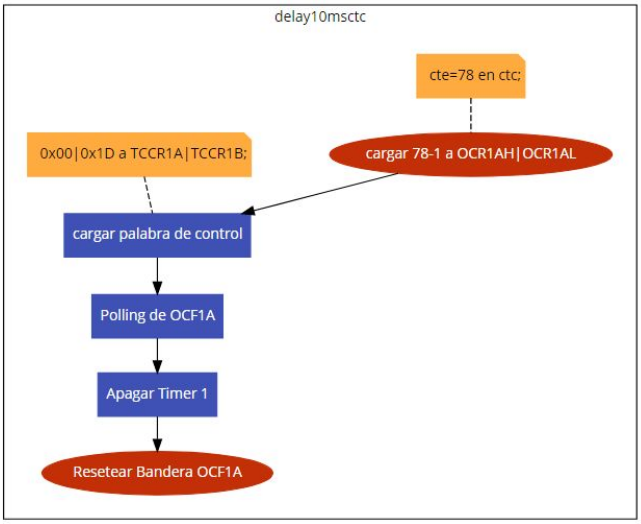


Diagrama 1: Diagrama de Flujo subrutina de delay10msctc

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. modificando el límite superior de conteo. 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. Antes de ellos, como se ve en la captura a continuación, se debe colocar el cursor en la mitad del LCD, resultando cómo:

Captura 5: Unión de subrutina de conteo con subrutina de envío de datos.

Para el segundo switch que activaba un PWM en OC0, se configuró el timer 0 en Fast PWM tal y como se solicitaba. Sin embargo antes de ser incrementado o decrementado por los botones, el PWM fue configurado para que se muestre en 50% de DutyCycle.

```
;pwm 50%
ldi r23, 0b01101101
out tccr0, r23
ldi r16, 127;50% DutyCycle
out ocr0, r16
```

Captura 5: Configuración inicial para el switch 2, PWM en Duty Cycle del 50%.

Una vez se muestre la amplitud de pulso modulada por OC0 [PB3], se solicitaba que este pulso pueda ser modificado por dos botones, uno encargado de aumentar el ciclo útil y oto para decrementarlo. El registro que contiene la constante para el ciclo útil es el OCR0, por lo que su modificación resultaría en el aumento o disminución del ciclo útil. Lo que se puede conseguir con una operación básica:

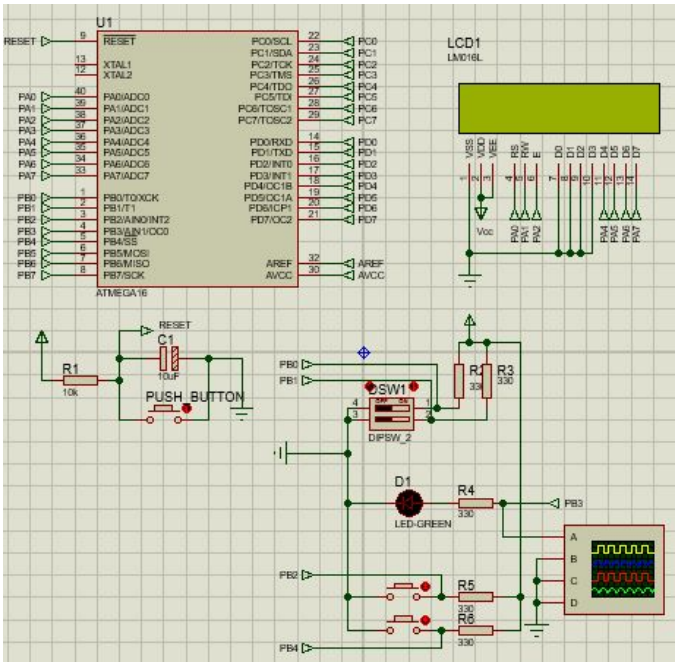
```
pollbttn1:
    sbic pinb, bttn1
    rjmp pollbttn2
    subi r16,-5;aumentar pwm
    out ocr0, r16
    call delay10msctc
    rjmp pollbttn1
pollbttn2:
    sbic pinb, bttn2
    rjmp switch2
    subi r16,5;disminuir pwm
    out ocr0, r16
    call delay10msctc
```

Captura 6: Aumento Disminución del ciclo útil .

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 serán el módulo LM016L, un LED para el PWM.

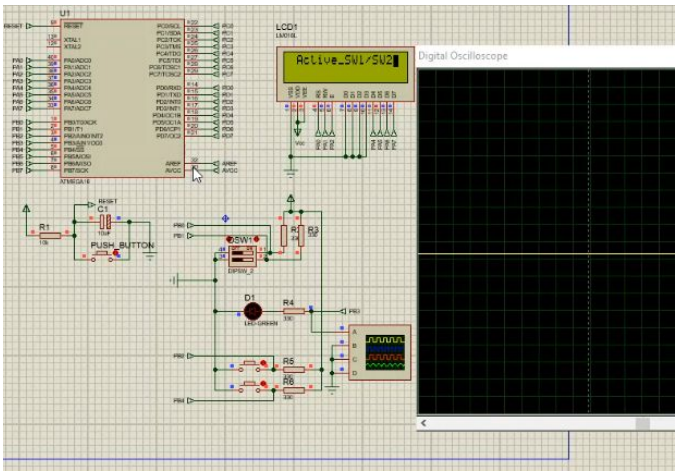


Captura 7: Sistema que integra al ATMega16, LCD, Switches, Botones y LED.

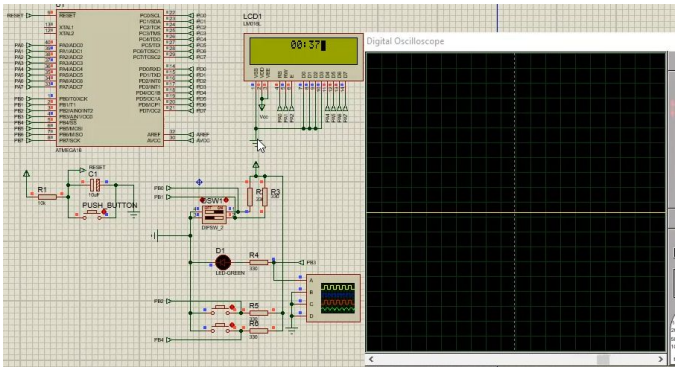
La frecuencia a la cual trabajará el ATMega16 se mantiene, por lo que se configura “CKSEL Fuses” que prepara al MCU a trabajar de cierta forma, en este caso específico “0100”, lo que significa que funcionará con su reloj interior, hecho con un arreglo de RC a 8MHz.

IV. Resultados

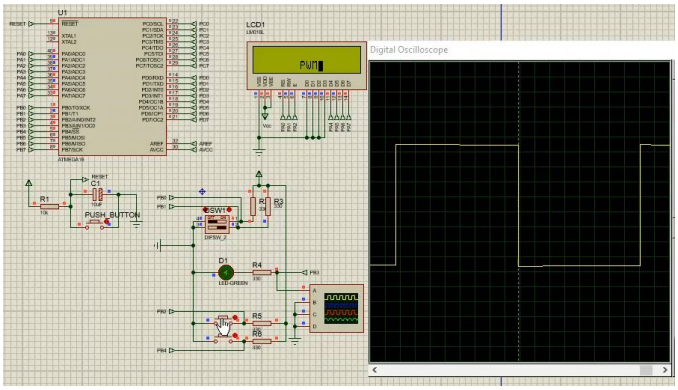
A. *Switch 1 - Corrimiento Secuencia A*



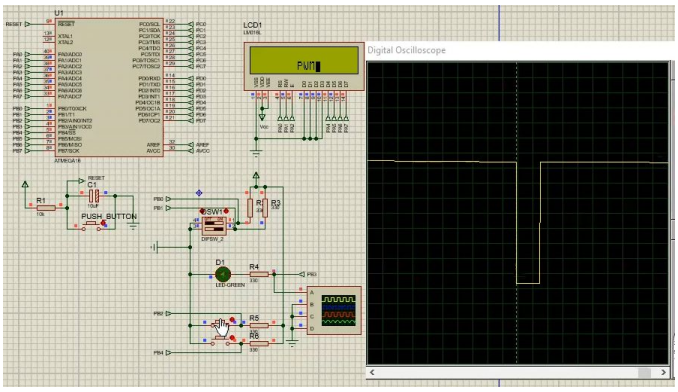
Captura 8: Mensaje de aviso para la activación de switch 1 o switch 2.



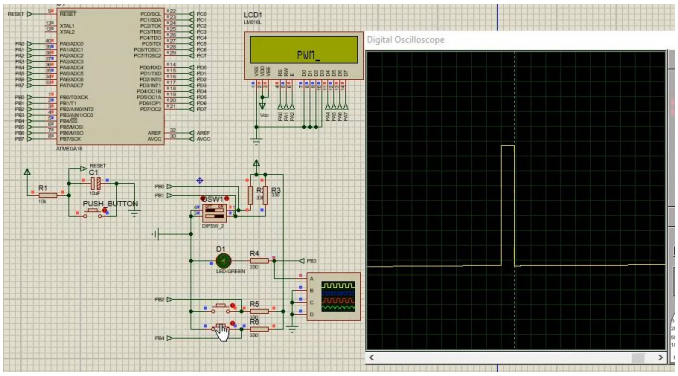
Captura 9: Activación del SW1, conteo.



Captura 10: Activación del SW2, PWM



Captura 10.1: Activación del SW2, Aumento de PWM



Captura 10.2: Activación del SW2, Disminución de PWM

V. Conclusiones

Andrés Islas Bravo:

Esta práctica permitió retomar conceptos que se habían manejado en previas materias de semestres anteriores, pero en esta ocasión abordarlos desde la perspectiva de un lenguaje ensamblador sobre un ATMega 16. Dentro de lo más importante y nuevo respecto conocimientos fue la parte de enviar en modo serial las instrucciones, con el propósito de sólo usar 4 bits. Esto fue importante ya que se tuvieron que considerar nuevos parámetros de tiempos entre cada instrucción y entender a mayor profundidad la comunicación entre el controlador y el display.

Juan Pablo Valverde López:

Se consiguió integrar las diferentes rutinas que fueron desarrolladas en el pasado. Además del desarrollo de nuevos timers que verdaderamente hacen uso de las disposiciones del ATMega16 con lo que se mejoró el rendimiento del programa. Creo que es importante resaltar la utilidad del desarrollo de programas por bloques, pues se puede hacer uso de ellos de manera individual, ajustando características externas para que modelen la rutina donde son requeridos.

Alfredo Zhu Chen:

Esta práctica me permitió integrar conocimientos vistos en las prácticas anteriores para desarrollar el sistema. El objetivo de la práctica se cumplió porque se pudo implementar el conteo correctamente para cuando se active el primer switch, modular el ciclo útil del LED para cuando se active el segundo switch y mostrar los respectivos mensajes en distintos casos mediante el LCD. Fue muy interesante la actividad ya que se tuvo que utilizar el *Timer 0* y 1. Finalmente, logré probar el código escrito en el programa sobre la simulación de Proteus para observar el funcionamiento de manera visual.

VI. Referencias

[1] S. a. (2014). LCD Interfacing Tutorial. Obtenido desde: https://www.8051projects.net/lcd-interfacing/introduction.php

[2] 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