Práctica 6
Contadores Digitales

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 covers a wide range of concepts with the main purpose of designing a seven segment display capable of showing a 4 units value. Ideas are conceived by writing them in a flowchart, in this case is very useful for the optimization of peripherals ports. The main control is done by the ATMega16, but important components are working in collaboration for the decoding and latch of the data.

I. Nomenclatura

Decimal Codificado en Binario,	*BCD.*
Unidad de microcontrolador,	*MCU.*
Resistencia-Capacitor,	*RC.*
Unidad de frecuencia, hercio,	*Hz.*
Diodo Emisor de Luz,	*LED.*

II. Introducción

Dentro del presente documento se cubren diversos conceptos de Hardware, dichos conceptos se han ido consolidando mediante la realización de previas prácticas, para así llegar a un control óptimo por parte del atmega16, que en este caso se encargará de controlar a otros componentes. El control de otros componentes tiene el propósito de optimizar los periféricos para así poder a futuro conectar más componentes. En este caso los componentes correspondientes son demultiplexores de 1 a 4 con 2 bits de selección, así como los correspondientes decodificadores de bcd a 7 segmentos. Dentro de las labores a realizar por este circuito están: por una parte un conteo ascendente de uno a uno desde un valor de 0 a un valor de 9999, esto con la activación del primer switch, y por otra parte se cuenta con un conteo de dos en dos el cual estará definido por el switch dos y un botón que cada vez que se ha presionado sucederá dicho incremento. Dichas labores demandan una correcta implementación de los registros ya que el conteo de 2 en 2 tendrá un punto de partida desde el último valor que se escribió por la rutina del switch1, esto sin alterar dicho valor ya que si se regresa a la primera rutina se tendrá que tomar de partida dicho punto.

III. Marco Teórico

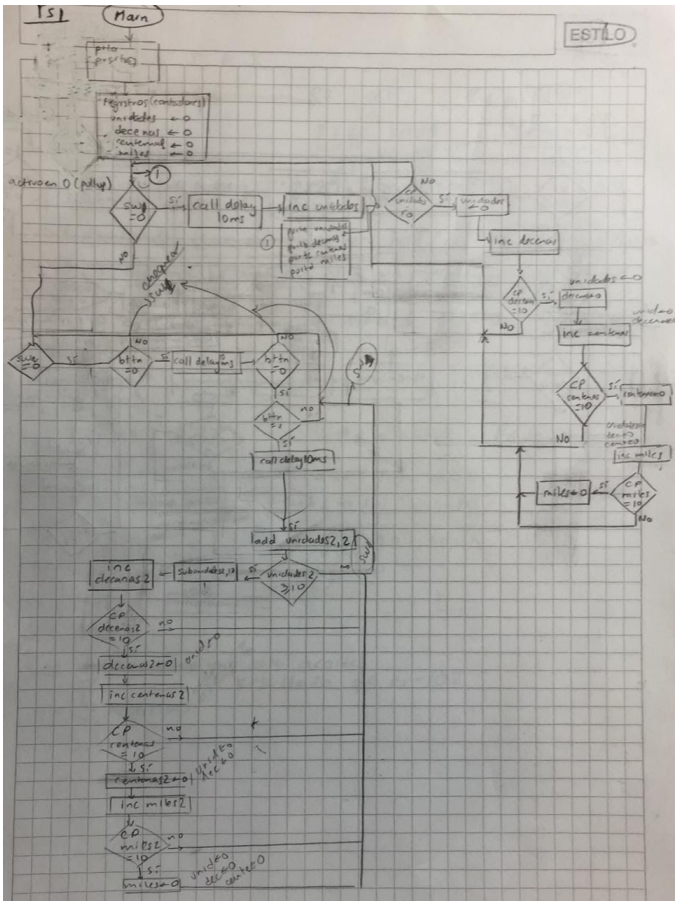
El tiempo es una variable que afecta a todo tipo de procesos y actividades; además, puede ser aprovechada indirectamente para medir otros tipos de magnitudes. Los contadores son una buena herramienta para el manejo de la variable tiempo, por cuanto permiten medirla con precisión y también definir intervalos temporales precisos.

Así mismo, se tiene que considerar la diferencia entre el los contadores en base 10 y en base 2. El microcontrolador realiza el conteo en base 2, sin embargo, se tiene que hacer un sistema que reinicie la unidad de conteo dado caso que llegue a 10 y eso justamente se logra como si fuera en un código de BCD. Es así que, el sistema decimal es universal para las personas y los contadores con esta base son muy importantes en distintas tareas que se pueden realizar.

IV. Desarrollo

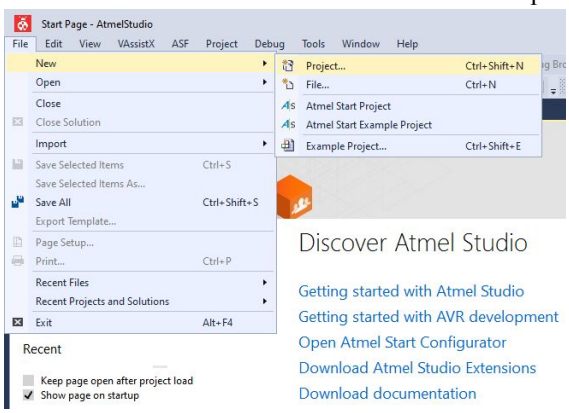
La presente práctica propone la elaboración de un contador digital de 0 a 9999, que será visualizado en 4 displays 7 segmentos, la lógica del contador estará sometido a atender 2 switches que desencadenarán variaciones de ese conteo. Siendo el primer switch el encargado de que empiece a contar de forma ascendente de 0 a 9999 con un periodo de 10 ms entre cada incremento. Dentro de este switch se deben considerar que el conteo se reinicia de forma automática además de que el número en el que va el conteo siempre se guarda, si el switch llegase a desactivarse y posteriormente se vuelve a activar, el conteo inicia donde se quedó. Al activar el segundo switch se debe mostrar en los displays el número en el que va el conteo (del primer switch). Cada vez que se presione 1 botón, el conteo mostrado en los displays deberá incrementar en 2. El incremento ocurrirá hasta que se libere el botón

Una buena práctica es la realización de un diagrama de flujo que explique a grandes rasgos el funcionamiento y la lógica del programa.



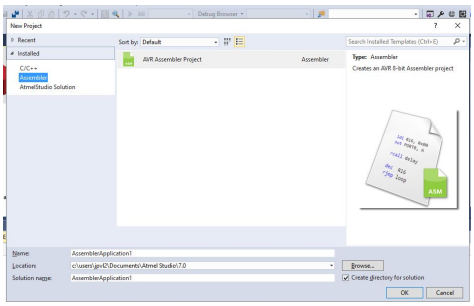
Captura 1: Diagrama de Flujo.

Para iniciar un nuevo proyecto, es necesario dirigirse a la pestaña "File", luego "New" y al final "Project". O al presionar las tres teclas "Ctrl+Shift+N" al mismo tiempo.



Captura 2: Creación de un nuevo proyecto.

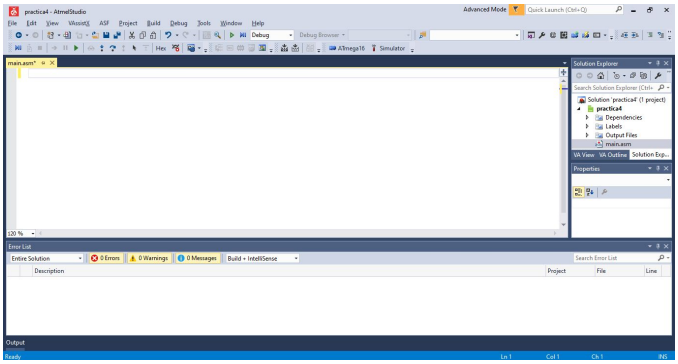
Luego de haber seleccionado la opción de crear un nuevo proyecto, una ventana de asistencia emergerá, en la que debemos elegir la opción de "Assembler" por sobre la C/C++ o Atmel Studio Solution, pues se pretende programar el microcontrolador en lenguaje ensamblador. Dentro de la misma ventana, también se escoge el directorio donde se guardará el proyecto así como el nombre bajo el cual se podrá identificar el mismo.



Captura 3: Ventana de Asistencia, Creación de Proyecto. Lenguaje Ensamblador.

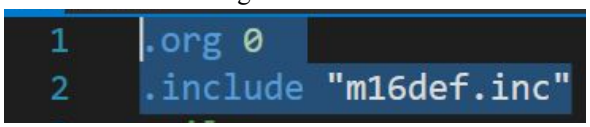
También se debe buscar el modelo del microcontrolador a utilizar, en este caso será el "ATMEGA16", es importante recalcar puesto que una versión muy similar es el "ATMEGA16A" pero este no se ocupará dentro del curso. En realidad este aspecto es a consideración del programador pues depende de cuál tenga disponible físicamente.

Una vez elegidos los detalles anteriores, una ventana emergerá con nombre "main.asm" la cual llevará escrita una plantilla modelo para poder programar con cierta guía. Sin embargo, para esta y futuras ocasiones, se eliminarán las líneas anteriores mencionadas. Por lo que la ventana quedará:



Captura 4: Ventana "main.asm".

Las primeras "instrucciones" serán para el compilador, son directrices para funcionar de cierta manera. En este caso las 2 instrucciones resaltadas significan:



Captura 5: Instrucciones Directrices para el compilador.

Instrucción - Directriz	Interpretación
.include "*mcu*.inc"	Incluye las instrucciones y comandos, memorias, relacionadas con el (m)ega(16)(def)initions
.org #	Empieza a escribir desde la dirección de memoria flash # en adelante.

Tabla 1: Instrucciones directrices para el compilador.

Es necesario preparar la pila que hará uso de las últimas direcciones de la memoria RAM. Si bien es cierto que la pila no es ocupada en este programa en específico, sirve para guardar la dirección de retorno de una interrupción o subrutina además de guardar datos importantes ocupados mientras se estaba ejecutando otras instrucciones de tu programa. (PUSH/POP).

```
;pila
ldi r16, low(ramend)
out spl, r16
ldi r16, high(ramend)
out sph, r16
```

Captura 6: Configuración Pila.

Como ya se mencionó anteriormente, el programa pretende iniciar un conteo con la activación de un Switch, detenerlo y mostrarlo con la activación de otro, además de ser capaz de incrementar en 2 cada vez que un botón es pulsado. Para mayor claridad, cierto número de registros fueron definidos, es decir que serán interpretados por el compilador con el nombre que se les asignó al número de registro correspondiente.

```
;Definicion de registros para mayor claridad
.def unidades = r17
.def decenas  = r18
.def centenas = r19
.def miles    = r20
```

Captura 7: Definición de Registros r17, r18, r19, r20.

La preparación de los puertos es el siguiente paso cuando se programa un microcontrolador.

```
;puertos como salida
ldi r16, 0b00001111
out ddra, r16
out ddrb, r16
out ddrc, r16
out ddrd, r16
;sw1 -> porta 4
;sw2 -> porta 5
;bttn -> porta 6
```

Captura 8: Código - Preparación de puertos.

```
sw1: ;etiqueta a la cual se regresará constantemente para pollear el switch.
;saco #s por los puertos, es una acción principal así que cada vez que reviso el switch
;tengo que sacar los contados por los displays (todo el tiempo)
out porta, unidades;puerto a (display 1) unidades
out portb, decenas;puerto b (display 2) decenas
out portc, centenas;puerto c (display 3) centenas
out portd, miles;puerto d (display 4) miles
    sbic pina, 4;Si el bit 4 del puerto a está en 0 (desactivado) se salta esa la linea siguiente. (Llama el delay10ms)
    rjmp sw2
    call delay10ms;ejecuta delay10ms
    inc unidades;incrementa unidades (intervalo 10ms)
    cpi unidades, 10;comparo unidades con 10, para incrementar contador del 2do display
    brne sw1;reviso el SW1. Salta si unidades no son igules a 10 (Sigo aumentando el contador de Unidades)
    clr unidades;limpia unidades, comprobé que unidades llegaron a 10.
    inc decenas; incremento decenas
    cpi decenas, 10;repito lógica utilizada con unidades.
    brne sw1
    clr unidades
    clr decenas
    inc centenas
    cpi centenas, 10
    brne sw1
```

Captura 8.1: Código - Sensado de Switch 1.

```
    clr unidades
    clr decenas
    clr centenas
    inc miles
    cpi miles, 10
    brne sw1
    clr unidades
    clr decenas
    clr centenas
    clr miles
    rjmp sw1
sw2:
    sbic pina, 5;Si el bit 5 del puerto a está en 0 (desactivado) se salta esa la linea siguiente. (SW2 está apagado)
    rjmp sw1;revisa SW1
bttn:;eliminación de rebote bttn
    sbic pina,6;Si el bit 5 del puerto a está en 0 (desactivado) se salta esa la linea siguiente. (BTTN no ha sido presionado)
    rjmp sw1;revisa SW1
    call delay10ms;llama delay10ms
    sbic pina,6
    rjmp sw1; fue ruido
bttn2:
    sbis pina,6
```

Captura 8.2: Código - Sensado de Switch 2.

```
    rjmp bttn2;espero a que se suelte
    call delay10ms
    subi unidades, -2;"sumo +2" al contador
    cpi unidades, 10
    brlo sw1
    subi unidades, 10
    inc decenas
    cpi decenas, 10
    brne sw1
    clr unidades
    clr decenas
    inc centenas
    cpi centenas, 10
    brne sw1
    clr unidades
    clr decenas
    clr centenas
    inc miles
    cpi miles, 10
    brne sw1
    clr unidades
    clr decenas
```

Captura 8.3: Código Botón..

```
        clr centenas
        clr miles
        rjmp sw1
    delay10ms:
        LDI R16,104
        ciclo2:
        LDI R21, 255
            ciclo1:
                DEC R21
                BRNE ciclo1
            DEC R16
            BRNE ciclo2
    ret
```

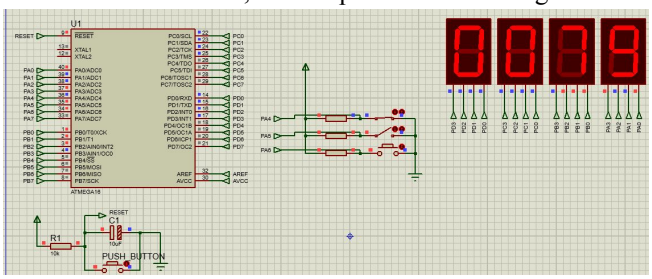
Captura 8.4: Código - Sensado de Switch 3.

En principio, como es descrito en las líneas de código que integran este programa, se sensa el estado del primer switch, dentro de la etiqueta “sw1”, donde la primera acción es la de sacar por los puertos a, b, c, d, los contadores; luego se pregunta por el estado del bit 4 del puerto a, donde se conecta este switch. que fue configurado activo en nivel lógico bajo (0), por medio de una configuración de resistencia pull-up. Una vez preguntado si este no está activo, incrementa el contador de unidades y realiza toda la lógica para incrementar las decenas, centenas y miles. Siguiendo eso, se salta a preguntar por el switch 2.

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.

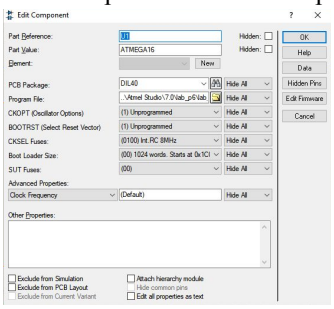
Comprobación por Proteus.

Para esta comprobación se hará uso del programa anterior diseñado, el sistema mínimo del ATMega16, que básicamente consiste en el etiquetado y programación de un botón de RESET para el MCU.
Las entradas para el sistema serán 2 switches configurados para ser activos en nivel lógico bajo (resistencia pull-up) y un botón.
Las salidas en este caso, serán 4 puertos del ATMega 16.



Captura 9: Sistema Mínimo del ATMega16 en conexión a sus entradas y salidas.

Para cargar el archivo de instrucción de código máquina al MCU, se da click derecho sobre él y se selecciona en "Edit Properties". Donde emergerá una ventana, allí en "Program File" se carga la dirección del programa con extensión .hex de donde se servirá el MCU para simular el comportamiento.

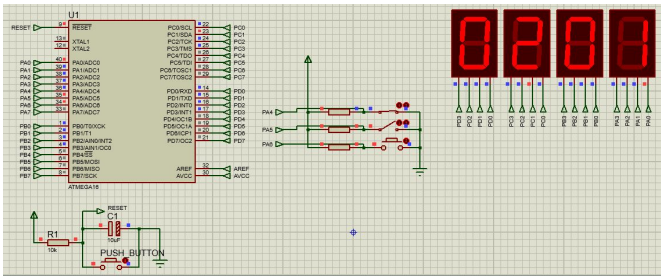


Captura 10: Carga de archivo ".hex" para la simulación de corrimiento de LEDs.

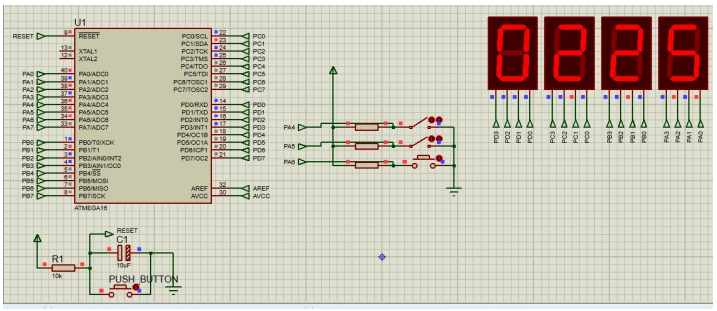
Para la selección del la frecuencia a la que trabajará el MCU, se lo hace en el apartado de "CKSEL Fuses" que prepara al MCU a trabajar de cierta forma, en este caso específico "0100", lo que significa que funcionará con su reloj interior, hecho con un arreglo de RC a 8MHz.

V. Resultados

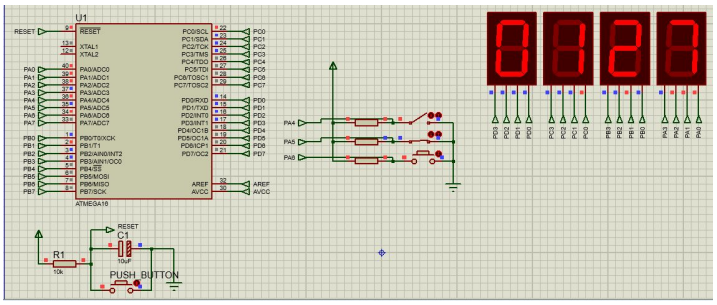
A. Switch 1 - Corrimiento Secuencia A



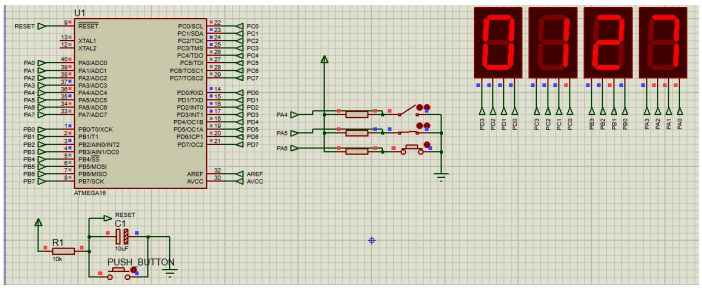
Captura 11: Ejecución de Conteo apretando el Switch1.



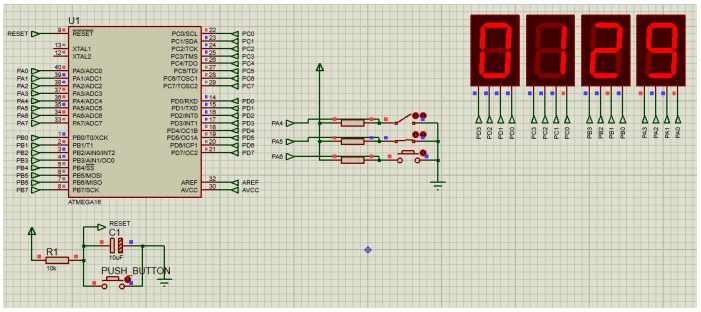
Captura 12: Levantamiento Switch1, pausado de conteo.



Captura 13: Encendido de Switch 2. Conteo de Switch 1 reflejado en Switch 2.



Captura 14: Opresión de botón.



Captura 15: Incrementó 2 en el contador.

VI. Conclusiones

Andrés Islas Bravo:

Conforme va incrementando la dificultad de las labores a realizar en los controladores, la realización de diagramas de flujo empieza a demostrar su importancia de una manera más notoria. En esta práctica la previa planeación permitió la optimización de recursos en los periféricos, ya que tan sólo por un puerto se logró transmitir la información necesaria para 4 displays 7 segmentos. En este caso se pudieron optar por dos caminos diferentes, uno donde se hacía el uso de componentes del Latch para así mantener un valor sobre el display, otro donde se aprovechaba las deficiencias del ojo humano para así hacer un barrido lo suficientemente rápido que no fuera

perceptible, dando como resultado que el valor se visualiza como estático. Por otro parte aprendimos un correcto manejo de los registros para así cumplir de manera exitosa las labores solicitadas.

Juan Pablo Valverde López:

Al igual que en problemas pasados, como en problemas futuros, la solución al problema puede presentarse de muchas maneras, en este caso en particular es importante mantener un estricto control en las unidades quienes son las responsables de que los demás contadores aumenten. En ambos casos, y como la lógica lo exigía, las unidades se revisaban primero y su lógica era repetida en los demás contadores. Es importante considerar que el diseño exterior también afecta el cómo presentas la solución, si se escoge la activación de los switches en alto (1) implica su activación de diferente manera dentro del código.

Alfredo Zhu Chen:

Esta práctica me permitió realizar un contador de cuatro dígitos en BCD. Comparando la programación de mi código con los de mis compañeros, me he dado cuenta que el problema se puede resolver de varias maneras y es interesante observar estos distintos métodos que se presentan. La práctica fue interesante y pude reforzar y aprender conocimientos con respecto a las instrucciones del lenguaje ensamblador. Finalmente, logré probar el código escrito en el programa de simulación de Proteus para observar el funcionamiento de manera visual.

VII. 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