Práctica 10
Interrupciones

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 is mainly focused in the appliance of interruptions in ATMega16. Interruptions appear as a consequence of the multiple tasks that are demand by the code, because in one hand the switches are being pulled, and on the other hand the timers or the buttons needs to give the counting indication. So buttons and timers are checked by interruptions.

I. Introducción

En el documento de continuación se abordan los aspectos tanto físicos como de código ensamblador de la práctica realizada. Principalmente, se dará énfasis al manejo de interrupciones ya que en esta actividad será el valor agregado respecto a las previas. se tendrá un polling constante para saber el estado en el que se encuentran los switches, estos determinarán la acción a realizar. Por una parte el primer switch habilitará un conteo respecto a tiempo y el segundo switch habilitaran un conteo que dependerá de las pulsaciones de botones. Dichos parámetros de tiempo y pulsaciones de botones serán atendidas por medio de interrupciones. Las interrupciones generadas por el vector de timer generarán un conteo ascendente de uno a uno, por otra parte los botones; uno generará un conteo ascendente y otro generará un conteo descendente.

II. Marco Teórico

Las interrupciones son recursos del microcontrolador que permiten atender una subrutina de servicio de interrupción al detectar un evento en específico[1]. Es una gran ventaja utilizar interrupciones, ya que no requiere un sondeo constante o una ciclación extensa del código para indicar que se deba realizar una tarea.

ATMega 16 cuenta con 21 fuentes de interrupciones, cada una se activa de una manera específica y se tiene que escribir una línea de instrucción en el vector específico de la dirección del programa cuando se quiera utilizar, en práctica se hace llamar la subrutina de interrupción en el vector indicado. Así mismo, este cuenta con la posibilidad de 3 interrupciones externas, interrupciones por desbordamiento(*Overflow*) y encuentro (*Match*) para los temporizadores(*Timers*), interrupciones por transmisión y recepción completa para comunicaciones seriales y entre otras más.

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	TIMER2 COMP	Timer/Counter2 Compare Match
5	$008	TIMER2 OVF	Timer/Counter2 Overflow
6	$00A	TIMER1 CAPT	Timer/Counter1 Capture Event
7	$00C	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	$00E	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	$010	TIMER1 OVF	Timer/Counter1 Overflow
10	$012	TIMER0 OVF	Timer/Counter0 Overflow
11	$014	SPI, STC	Serial Transfer Complete
12	$016	USART, RXC	USART, Rx Complete
13	$018	USART, UDRE	USART Data Register Empty
14	$01A	USART, TXC	USART, Tx Complete
15	$01C	ADC	ADC Conversion Complete
16	$01E	EE_RDY	EEPROM Ready
17	$020	ANA_COMP	Analog Comparator
18	$022	TWI	Two-wire Serial Interface
19	$024	INT2	External Interrupt Request 2
20	$026	TIMER0 COMP	Timer/Counter0 Compare Match
21	$028	SPM_RDY	Store Program Memory Ready

Captura 1:Tabla de vectores para colocar instrucciones como subrutinas de servicio de interrupción. Tomado desde:
http://ww1.microchip.com/downloads/en/devicedoc/doc2466.pdf

El proceso de manipulación de interrupciones es la siguiente: se completa la ejecución de instrucción en curso; se salva en la pila la dirección de retorno; se inhabilitan todas las interrupciones con la bandera de interrupciones I=0 en el registro SREG(*Status Register*); se salta a la dirección correspondiente en el vector de interrupción; se habilita nuevamente las interrupciones al finalizar la subrutina poniendo la bandera I de vuelta a 1 para permitir nuevas solicitudes de interrupción.

Para habilitar las interrupciones externas, se debe de poner en 1 los últimos bits del registro GICR(*General Interrupt Control Register*), el bit 5 es para habilitar interrupción externa INT2 por medio de PB4, bit 6 para INT0 por medio de PD2 y bit 7 para INT1 por medio de PD3. Por un lado, el registro MCUCR(*MCU Control Register*) permite indicar la forma en que se detectan las interrupciones INT0 e INT1 por medio de los primeros cuatro bits(ISC, *Interrupt Sense Control*). Por otro lado, MCUCSR(MCU Control and Status Register) permite indicar lo mismo para INT2 en el bit 6(ISC2). El registro GIFR(General Interrupt Flag Register) contiene las banderas de estas interrupciones externas indica cuando se solicita la interrupción y cuando se está atendiendo, es importante mencionar que no hay necesidad de leerlas y apagarlas porque se el microcontrolador lo hace propiamente en el momento[2].

TABLA I
CONFIGURACIÓN DE SOLICITUD DE INTERRUPCIÓN PARA INT0 E INT1 EN LOS PRIMERO CUATRO BITS DEL REGISTRO MCUCR

Forma de solicitud de las interrupciones INT0/INT1	ISCn1	ISCn0
nivel bajo	0	0
Detección por cualquier cambio de nivel	0	1
Detección por flanco de bajada	1	0
Detección por flanco de subida	1	1

TABLA II
CONFIGURACIÓN DE SOLICITUD DE INTERRUPCIÓN PARA INT2 EN EL REGISTRO MCUCSR

Forma de solicitud de las interrupciones INT2	ISC2
Detección por flanco de bajada	0
Detección por flanco de subida	1

Para habilitar las interrupciones por *Timers,* se debe de realizar en el registro TIMSK(*Time Interrupt MasK register*). Las banderas que indican la atención y solicitud de interrupciones se encuentran en el registro TIFR(Timer Interrupt Flag Register), al igual que las interrupciones externas, no hay necesidad de apagarlas manualmente porque el microcontrolador se encarga de eso en el momento. En la siguiente tabla se muestra cada bit de TIMSK para habilitar la interrupción correspondiente de los *Timers*[2].

TABLA III
CONFIGURACIÓN DE SOLICITUD DE INTERRUPCIÓN PARA LOS TIMERS EN EL REGISTRO TIMSK

OCIE2|TOIE2|TICIE1|OCIE1A|OCIE1B|TOIE1|OCIE0|TOIE0	
TOIE0	*Timer* 0 por encuentro(*Match*)
OCIE0	*Timer* 0 por desbordamiento(*Overflow*)
TOIE1	*Timer* 1 por encuentro(*Match*)
OCIE1B	*Timer* 1 B por encuentro(*Match*)
OCIEA	*Timer* 1 Apor encuentro(*Match*)
TICIE1	*Timer* 1 por entrada de captura(*Match*)
TOIE2	*Timer* 2 por encuentro(*Match*)
OCIE2	*Timer* 2 por desbordamiento(*Overflow*)

III. DESARROLLO

La presente práctica emplea cuatro displays de 7 segmentos, como método de visualización de un contador de 4 dígitos. Este contador tendrá dos modos de funcionamiento, que serán activados por los respectivos switches. Además para esta práctica se emplearán interrupciones dentro del ATMega16. La primera interrupción será la de RESET donde se configurará el main del problema, se inicializa y se estructura los diferentes recursos que se emplearán. La segunda interrupción estará accionada por un botón, definido por el manual del microcontrolador en el PIND2, el botón incrementará en 1 el contador que se visualizará en los 4 displays. La siguiente interrupción se activará al presionar un botón conectado al PIND2 del ATMega16. Por úlitmo, la interrupción que de ahora en adelante será mencionada como de refrescamiento, será ejecutada cada 10ms que es el tiempo para el cual el ojo humano no percibe cambios por lo rápido que se producen y es una ventaja dentro de la visualización del contador de la presente práctica. Los requerimientos de hardware del sistema son:

- 4 Displays 7 SEG.
- 2 Botones.
- 2 Switches.

Para codificación del problema dentro del "main" estará la lógica de sensado de switches e indirectamente se activarán la

bandera “t” que será usada como método de identificación del botón que fue accionado.

```
.org 0
rjmp main

.org 0x2; INT0 - Externa 0 (Button1)
rjmp button1_inc

.org 0x4; INT1 - Externa 1 (Button2)
rjmp button2_dec

.org 0x26; Timer0 - Compare (10ms)
rjmp conteo_refresh
```

Captura 2: Configuración Vectores de Interrupción.

```
pollsw1:
    clt
    sbis pind, sw1
    rjmp pollsw2
    set
pollsw2:
    sbis pind, sw2
    rjmp nosws
    ;Habilito interrupciones externas solo cuando SW2 ON
    ldi r16, 0b11000000
    out gicr, r16
nosws:
    clr r16
    out gicr,r16
    clr cont1
    clr cont2
    clr cont3
    clr cont4
    sts 0x60, cont4
    sts 0x61, cont3
    sts 0x62, cont2
    sts 0x63, cont1
    rjmp pollsw1
```

Captura 2.1: Lógica de Activación de Switches

```
refresh:
    ld r0, x+
    mov r21, barrido
    andi r21, 0b11110000
    add r0, r21
    out porta, r0;saco el #
    rol barrido
    dec displays
    brne salir
    salir:
    sts 0x60, cont4
    sts 0x61, cont3
    sts 0x62, cont2
    sts 0x63, cont1
pop r16
out sreg, r16
reti
```

Captura 2.2: Interrupción de refrescamiento

Si ningún switch está activo, se visualiza un “0000” en los displays. El primer switch iniciará el conteo ascendente de 0 a 9999 y los mostrará en los 4 displays. Cada dígito mostrado es almacenado en una localidad de memoria RAM para que la interrupción a la cual se accede cada 10 ms lea esas localidades y las muestre en cada displays.

El segundo switch mostrará inicialmente un “0000” en lso displays y este conteo podrá ser manipulado por medio de los botones, donde en cada interrupción se configuró para que uno de ellos incrementa en 1 el contador y otro lo decrementa en 1.

Las palabras de control para los requerimientos de este programa fueron configuradas de la siguiente manera.

- **Cálculo de Constante para Timer 0**
  - $\frac{T}{Tmin\ x\ Prescales} - 1 = 77.12$
  - **OCR0 - Output Compare Register**
- **Palabra de Control**
  - **Modo CTC, Pres 1024 - TCCR0**
  - **00001101**
- **Habilitar interrupción Timer 0 - TIMSK.**
  - **00000010**
- **Configuro cómo trabajan las interrupciones externas - MCUCR**
  - **00001010**

- **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 incremento y decremento. Para poder asociar esta subrutina con las interrupción de refrescamiento se la mandó a llamar dentro de la interrupción de la siguiente manera.

```
button1_inc:
in r16, sreg
push r16
    call conteo
pop r16
out sreg, r16
reti

button2_dec:
in r16, sreg
push r16
    call decremento
pop r16
out sreg, r16
reti
```

Captura 3: Unión de subrutina de incremento/decremento con interrupción.

La subrutina de conteo a su vez emplea una lógica similar a la anteriormente desarrollada con la única diferencia de guardar

los registros involucrados en las localidades de RAM a las cuales se accedía y se visualizaban en los cuatros displays.

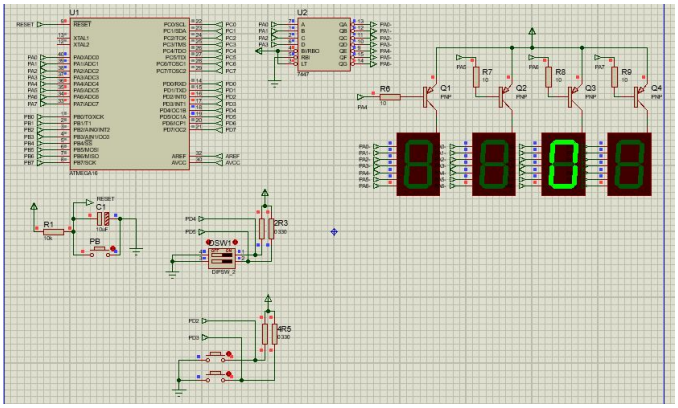
```
conteo:
    inc cont1;
    cpi r16, 10
    brne regreso;
    clr cont1;
    inc cont2;
    cpi cont2, 10;
    brne regreso;
    clr cont2
    inc cont3
    cpi cont3, 10
    brne regreso
    clr cont3
    inc cont4
    cpi cont4, 10
    brne regreso
    clr cont1
    clr cont2
    clr cont3
    clr cont4
regreso:
    sts 0x60, cont4
    sts 0x61, cont3
    sts 0x62, cont2
    sts 0x63, cont1
ret
```

Captura 4: Subrutina de conteo.

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.

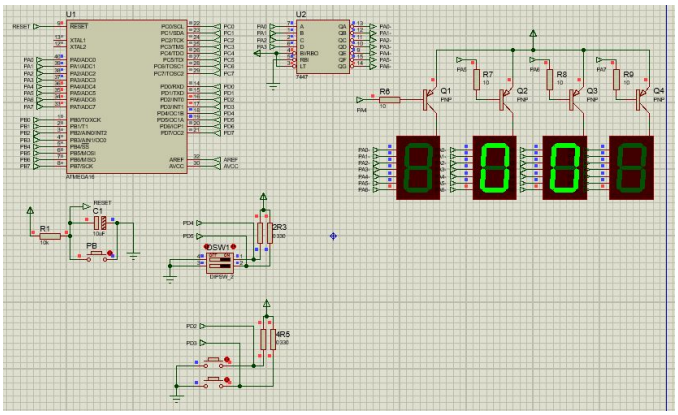
Las salidas para ese sistema serán los cuatro displays 7 SEG conectados por medio de un decodificador LS7447 utilizado para cátodo común y transistores que permitirán el encendido y apagado de acuerdo al código de barrido de la interrupción de refrescamiento.



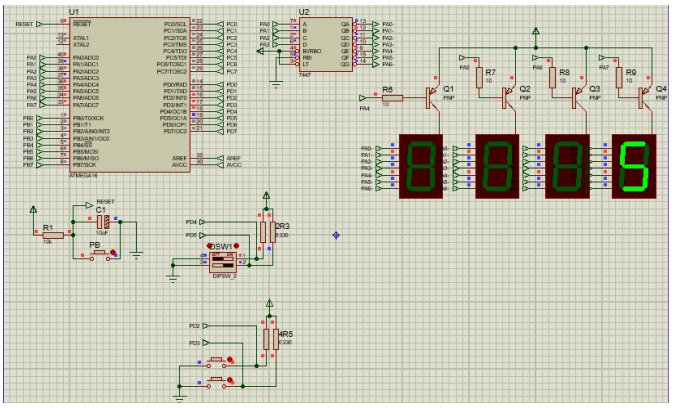
Captura 5: Sistema que integra al ATMega16, LCD, Switches y Osciloscopio.

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.

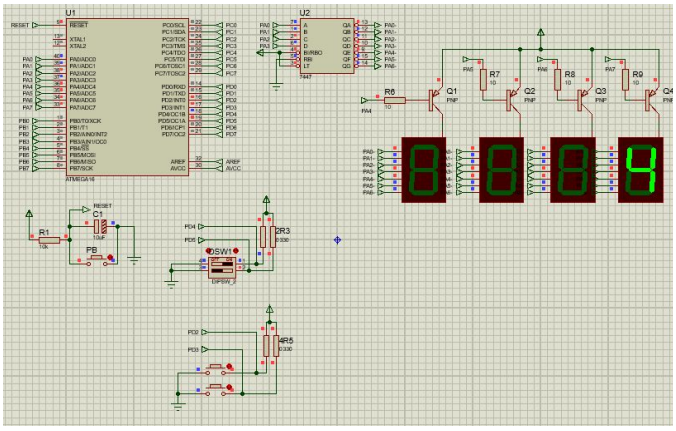
IV. Resultados



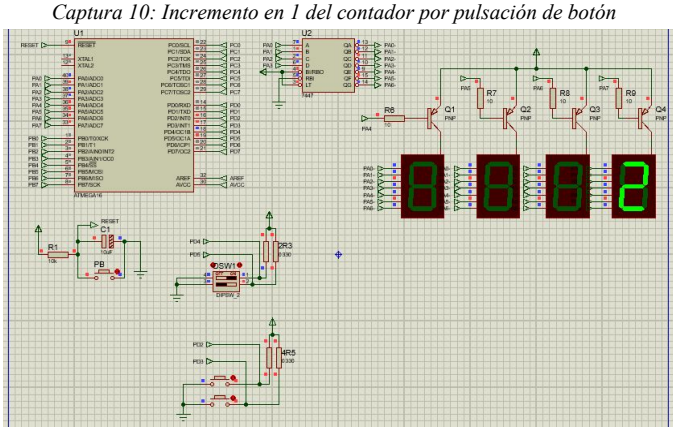
Captura 8: Visualización de "0000" mientras ningún switch esté encendido..



Captura 9: Activación de conteo por medio del switch 1.



Captura 10: Incremento en 1 del contador por pulsación de botón



Captura 10: Decremento en 1 del contador por pulsación de botón

V. Conclusiones

Andrés Islas Bravo:

Esta práctica me permitió aplicar las interrupciones en una situación dónde la presencia de múltiples demandas por parte del código orillaba a la necesidad de que ciertos componentes y acciones fueran atendidos por las mismas. El el valor agregado de mayor significancia lo sentí en la estructuración del pensamiento, ya que programar por interrupciones es muy diferente al pensamiento lineal que existe al solo programar en el Main.

Juan Pablo Valverde López:

La implementación de interrupciones hace del sistema algo más sólido y mejor acondicionado para el trabajo real, muchas veces en simulación o problemáticas iniciales las situaciones eran idóneas para que nuestra solución sea la correcta. Sin embargo, conforme se descubren nuevas herramientas y recursos disponibles su integración hace del sistema algo más robusto.

Alfredo Zhu Chen:

Esta práctica me permitió manejar las interrupciones al aplicarlas en una práctica que se había hecho anteriormente. El objetivo de la práctica se cumplió porque se pudo implementar el conteo por timer y por botones a partir de interrupciones. Fue muy interesante la actividad ya que se tuvo que es la primera práctica que se ocupa interrupciones y pude observar que es un proceso mucho más eficiente. 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. (2010). Manejo de interrupciones. [Archivo PDF]. Obtenido desde: http://www3.fi.mdp.edu.ar/electrica/opt_archivos/arduino/Manejo_de_Interrupciones.pdf

[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