INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

MICROCONTROLADORES

PRÁCTICA N°1

PROFESOR:

PÉREZ PÉREZ JOSÉ JUAN

ALUMNOS:

TORRES SEGURA RICARDO

VARELA CRUZ CÉSAR

Gutiérrez González Norel

GRUPO: 3CM6

MATERIALES

- ✓ 1 Resistencias de 330 ohms
- ✓ 1 dip switch de 8 entradas
- ✓ 6 2n2222
- ✓ 1 push button
- ✓ 1 capacitor de 1MF
- ✓ 6 Displays de 7 segmentos ánodo común
- ✓ Microcontrolador ATmega8535

EQUIPO

- ✓ AVR studio 4
- ✓ 1 Equipo de cómputo con XP
- ✓ 1 programador
- ✓ 1 fuente de voltaje

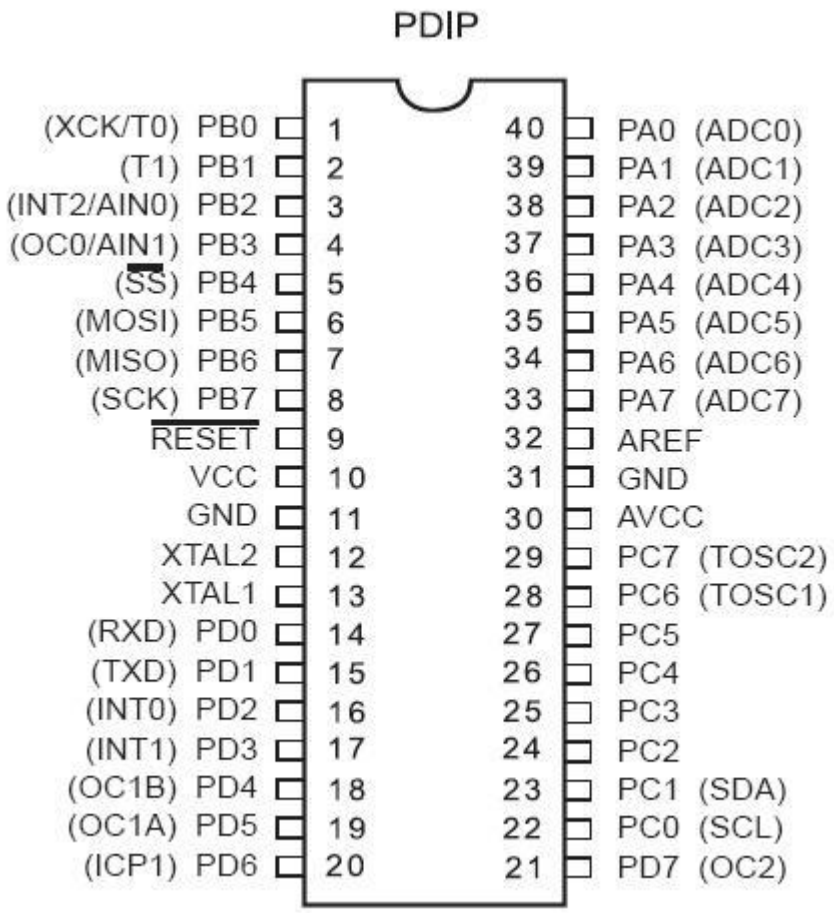
INTRODUCCION TEORICA

Atmel.

Es una compañía de semiconductores, fundada en 1984. Su línea de productos incluye microcontroladores y entre los cuales se encuentra el **ATMEGA8535.**

Características generales del ATMEGA8535

- Pertenece a la compañía Atmel
- Pertenece a la familia ATMEGA
- Es de 8 bits
- Maneja una arquitectura RISC, 130 instrucciones
- Interrupciones externas e internas
- Comparador analógico
- Comparador analógico digital
- 32 líneas programables I/O
- Un contador a tiempo real con oscilador separado



CÓDIGO

```
.include "m8535def.inc"
    Ldi R16, low(RAMEND)
    Ldi R17, high(RAMEND)
    out SPL, R16
    out SPH, R17
    ser R16
    out DDRA, R16
    out DDRC, R16
    out PORTB, R16
ciclo:  in R17, PINB
```

```
out PORTA, R17
com R17
out PORTC, R17
rjmp ciclo
```

CONCLUSIONES

RICARDO TORRES SEGURA

En esta práctica me sentí muy emocionada de poder tener un primer acercamiento con el microcontrolador aunque no entendí muy bien el código. Creo que me hace falta investigar más sobre la sintaxis para poder desarrollar otras prácticas. Me gustó el hecho de ver funcionar el micro.

VARELA CRUZ CÉSAR

En esta práctica vimos algunas instrucciones del mnemónico donde haciendo uso de ellas podemos programar en lenguaje ensamblador, también nos pudimos ver como funciona la interfaz gráfica para programar y poder compilar el código y revisarlo paso por paso.

Gutiérrez González Norel

En fin, la práctica fue muy útil para comprender la manera en que se debe programar el microcontrolador ATMEGA8535 haciendo uso de sus puertos y los recursos de los que dispone. A su vez, sirvió para familiarizarme con el dispositivo y cómo deben de realizarse las conexiones para que funcione correctamente.

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

MICROCONTROLADORES

PRÁCTICA N°2

PROFESOR:

PÉREZ PÉREZ JOSÉ JUAN

ALUMNOS:

TORRES SEGURA RICARDO

VARELA CRUZ CÉSAR

Gutiérrez González Norel

GRUPO: 3CM6

MATERIALES

- ✓ 16 Resistencias de 330 ohms
- ✓ 1 dip switch de 8 entradas
- ✓ 1 push button
- ✓ Microcontrolador ATmega8535

EQUIPO

- ✓ AVR studio 4
- ✓ 1 Equipo de cómputo con XP
- ✓ 1 programador
- ✓ 1 fuente de voltaje

INTRODUCCION TEORICA

Para poder desarrollar esta práctica fue importante el conocer ciertas instrucciones que nos ayudaron a desarrollar el código. Estas instrucciones vienen dentro del datasheet del microcontrolador, las cuales se ubican en una tabla en donde te muestran la sintaxis de cada una de ellas.

Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← 0xFF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • (0xFF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← 0xFF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2

Y aquí se muestran algunas de ellas, y como se mencionó anteriormente viene la sintaxis para usarla y además lo que hace.

CÓDIGO

```
.include "m8535def.inc"
    ser R16
    out DDRA, R16
    out DDRC, R16
    out PORTB, R16
    out PORTD, R16
lee:
    in R16, PINB
    in R17, PIND
    add R16, R17
    in R17, SREG
    out PORTA, R16
    out PORTC, R17
    rjmp lee
```

CONCLUSIONES

RICARDO TORRES SEGURA

En esta práctica aprendí un poco más sobre los puertos y cómo se declaran. Aprendí también que es necesario poner pull-ups a las entradas del dip si no queremos poner resistencias a las entradas del micro y que si se ponen los pull-ups, las entradas del dip van a tierra.

VARELA CRUZ CÉSAR

En esta práctica pudimos ver mas a detalle cómo se asignan los puertos como entrada o como salida para su posible utilización, vimos otras instrucciones las cuales nos mostraban su estado en las banderas y de esa manera pudimos ver un poco más a fondo cómo funcionaba cada una.

Gutiérrez González Norel

En conclusión, seguimos trabajando con los puertos y las operaciones que se pueden realizar en el microcontrolador. También, hicimos uso del registro de estado y mostramos su valor en el puerto C, lo cual demuestra que se realizó una suma (add) sin acarreo.

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

MICROCONTROLADORES

PRÁCTICA N°3

PROFESOR:

PÉREZ PÉREZ JOSÉ JUAN

ALUMNOS:

TORRES SEGURA RICARDO

VARELA CRUZ CÉSAR

Gutiérrez González Norel

GRUPO: 3CM6

MATERIALES

- ✓ 16 Resistencias de 330 ohms
- ✓ 2 dip switch de 8 entradas
- ✓ 2 barra de leds
- ✓ Un alambre de conexión
- ✓ Microcontrolador ATmega8535

EQUIPO

- ✓ AVR studio 4
- ✓ 1 Equipo de cómputo con XP
- ✓ 1 programador
- ✓ 1 fuente de voltaje

INTRODUCCION TEORICA

Las instrucciones nuevas que fueron usadas para el desarrollo de esta práctica fueron:

swap: Lo que hace esta instrucción es intercambiar los nibbles del acumulador.

SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7

ori: Esta instrucción realiza un or bit a bit entre dos operandos.

ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K

Otra cosa importante que se ha manejado en los códigos de las prácticas es lo siguiente:

DDRx: Es un registro de 8 bits que almacena información de configuración para los pines de PORTx. Escribir un 1 en la ubicación del pin en la DDRx hace que el pin físico de ese puerto sea un pin de salida y escribir un 0 hace que el pin sea un pin de entrada.

PORTx: Es un registro de 8 bits que almacena los valores lógicos que se emiten actualmente en los pines físicos de PORTx si los pines están configurados como pines de salida.

PINx: Es un registro de 8 bits que almacena el valor lógico, el estado actual, de los pines físicos en PORTx.

CÓDIGO

```
.include "m8535def.inc"

    ser r16
    out ddra, r16
    out ddrc, r16
    out portb, r16
    out portd, r16

otro:
    in r16, pinb
    in r17, pind
    cp r16, r17

    brlo chc
    out porta, r16
    out portc, r17
    rjmp otro
chc:
    out porta, r17
    out portc, r16
    rjmp otro
```

CONCLUSIONES

RICARDO TORRES SEGURA

En esta práctica aprendí una instrucción muy importante que me va a ayudar a manipular de mejor manera las entradas de un solo puerto. Ví cómo es que se programa un comparador y como es que se mandan los datos a la salida. Lo que más me gustó es que el código no es tan largo y se puede entender mejor que en las primeras prácticas.

VARELA CRUZ CÉSAR

En esta práctica utilizamos instrucciones que comparan un resultado y dependiendo de la condición podemos dirigirnos a otra parte del programa, ya sea repetir o saltar a la siguiente línea de código, es como un if en otro lenguaje de programación.

Gutiérrez González Norel

Finalmente, la práctica nos sirvió para apreciar el funcionamiento de los saltos condicionados como brlo y, también, las instrucciones que nos sirven para hacer comparaciones. Lo cual es útil para apreciar todo lo que nos permite realizar el microcontrolador con tan pocas líneas de código.

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

MICROCONTROLADORES

PRÁCTICA N°4

PROFESOR:

PÉREZ PÉREZ JOSÉ JUAN

ALUMNOS:

TORRES SEGURA RICARDO

VARELA CRUZ CÉSAR

Gutiérrez González Norel

GRUPO: 3CM6

MATERIALES

- ✓ 16 Resistencias de 330 ohms
- ✓ 2 Displays de 7 segmentos ánodo común
- ✓ Microcontrolador ATmega8535L
- ✓ 1 DIp switch de 8 entradas

EQUIPO

- ✓ AVR studio 4
- ✓ 1 Equipo de cómputo con XP
- ✓ 1 programador
- ✓ 1 fuente de voltaje

INTRODUCCION TEORICA

El tema visto para poder desarrollar esta práctica fueron las **rutinas de retardo.**

Lo importante de una rutina como estas es el entender que un retardo se logra por medio de un lazo que se repite varias veces, desde otro lazo que se repite otro número de veces y así sucesivamente hasta alcanzar el tiempo que se necesita para obtener lo requerido.

Y algunas instrucciones:

andi: Realiza un AND lógico entre el contenido del registro Rd y una constante, y deja el resultado en el registro destino Rd.

ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K

brne: Realiza un desvío si no son iguales.

BRNE	k	Branch if Not Equal

nop: Realiza un simple ciclo sin operar.

NOP		No Operation

Código

```
.include "m8535def.inc"

        I:
            ser r16
            out ddra, r16
            out portb, r16
            ldi r20, $5F
            ldi r21, $03
            ldi r22, $6D
            ldi r23, $67
            ldi r24, $33
            ldi r25, $76
            ldi r26, $7E
            ldi r27, $53
            ldi r28, $7F
            ldi r29, $77
            clr ZH

        otro:
            ldi ZL, 20
            in r16, pinb
            andi r16, $0f
    add ZL, r16
    ld r16, Z
            out porta, r16
            rjmp otro
```

```
.include "m8535def.inc"
      I:  ser R16
          out DDRA, R16
          out PORTB, R16
          out DDRD, R16
          ldi R20, $7E
          ldi R21, $30
          ldi R22, $6D
          ldi R23, $79
          ldi R24, $33
          ldi R25, $5B
          ldi R26, $5F
          ldi R27, $70
          ldi R28, $7F
          ldi R29, $7B
      otro:   clr ZH
          ldi ZL, 20
          in R16, PINB
          mov R17, R16
          andi r16,$0f
          add ZL, R16
          ld R16, Z
out PORTA, R16
swap r17
          andi r17,$0f
          ldi zl,20
          add zl,r17
          ld r17,z
```

```
    out PORTD, R17
    rjmp otro
.include"m8535def.inc"
    ldi R16, low(RAMEND)
    out spl, R16
    ldi R16, high(RAMEND)
    out sph, R16
    ser R16
    out DDRA, R16
    out PORTB, R16
    ldi R20, $7E
    ldi R21, $30
    ldi R22, $6D
    ldi R23, $79
    ldi R24, $33
    ldi R25, $5B
    ldi R26, $5F
    ldi R27, $70
    ldi R28, $7F
    ldi R29, $7B
nvo:    clr R16
otro:   in R17, PINB
    andi R17, 3
    inc R17
    rcall deco
    out PORTA, R18
ndly:   rcall delay
    dec R17
```

```
        brne ndly
        inc R16
        cpi R16, 10
        breq nvo
        rjmp otro
    delay:  push R17
        push R18
        ldi R17, $A7
    VGLOOP0:ldi R18, $02
    VGLOOP1:ldi R19, $F0
    VGLOOP2:dec R19
        brne VGLOOP2
        dec R18
        brne VGLOOP1
        dec R17
        brne VGLOOP0
        nop
        pop R18
        pop R17
        ret
    deco:   clr ZH
        ldi ZL, 20
        add ZL, R16
        ld R18, Z
ret
```

CONCLUSIONES

RICARDO TORRES SEGURA

En conclusión, los retardos son de mucha utilidad cuando se busca un proceso más preciso, dependiendo de lo que se busca hacer haciendo que el tiempo de ejecución se alargue. También, se observó la utilidad que tiene la decodificación en un microcontrolador y las pocas líneas de código que se generan con el lenguaje ensamblador.

VARELA CRUZ CÉSAR

En esta práctica utilizamos instrucciones ´para poder comparar nuestros registros, dependiendo del resultado el número se decrementa o incrementa, también vimos cómo se aplican los retardos creando llamadas a subrutinas que nosotros tuvimos que calcular.

Gutiérrez González Norel

En esta práctica aprendimos a decodificar de bcd a 7 segmentos, yo recuerdo que antes lo hacía con binario. Otra cosa importante que aprendimos es a poder usar una especie de ciclo de reloj, el cual hizo avanzar de número dependiendo de qué tan rápido queríamos que aumentara. Me gustó mucho esta práctica porque retomé lo que vi en Fundamentos de Diseño Digital y Diseño de Sistemas Digitales pero con código ensamblador.

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

MICROCONTROLADORES

PRÁCTICA N°5

PROFESOR:

PÉREZ PÉREZ JOSÉ JUAN

ALUMNOS:

TORRES SEGURA RICARDO

VARELA CRUZ CÉSAR

Gutiérrez González Norel

GRUPO: 3CM6

MATERIALES

- ✓ 7 Resistencias de 330 ohms
- ✓ 6 2n2222
- ✓ 6 Displays de 7 segmentos ánodo común
- ✓ Microcontrolador ATmega8535L

EQUIPO

● 1 Equipo de computo

● 1 fuente de voltaje

● 1 programador

● AVR studio 4

INTRODUCCION TEORICA

Multiplexado Proceso consistente en recibir mensajes de diferentes fuentes y enviarlas a un destino común. A la inversa, la técnica de multiplexado permite enviar a puntos de destino diversos datos que proceden de una fuente común.

CÓDIGO

```
. 1. .include "m8535def.inc"

2. .def col = r17

3. .def dato = r16

4. .def borra = r18

5. ldi dato, low(RAMEND)

6. out spl, dato

7. ldi dato, high(RAMEND)

8. out sph, dato

9. ser dato

10. out DDRA, dato

11. out DDRC, dato

12. ldi dato, $fe

13. mov r0, dato

14. ldi dato, $90

15. mov r1, dato

16. ldi dato, $8f
```

```
17. mov r2, dato
18. ldi dato, $81
19. mov r3, dato
20. ldi dato, $92
21. mov r4, dato
22. ldi dato, $fe
23. mov r5, dato
24. clr zh
25.
26. uno:
27. clr zl
28. ldi col, 4
29.
30. dos:
31. out PORTC, col
32. ld dato, z+
33. out PORTA, dato
34. rcall delay
35. ser borra
36. out PORTA, borra
37. LSL col
38. brcc dos
39. rjmp uno
40.
41. delay:
42. ldi r18, 6
43. ldi r19, 49
44. L1: dec r19
45. brne L1
```

46. dec r18

47. brne L1

CONCLUSIONES

RICARDO TORRES SEGURA

En esta práctica pudimos enviar un mensaje en los displays de 7 segmentos el cual solo se mostrará mientras el circuito este alimentado, esta práctica fue interesante ya que pudimos implementar un mensaje a través del microcontrolador.

VARELA CRUZ CÉSAR

En la realización de esta práctica pudimos ver como copiar datos específicos en los registros para así luego poderlos llamar en distintos displays de 7 segmentos los cuales se encontraban multiplexados

Gutiérrez González Norel

Al realizar esta práctica utilizamos seis displays los cuales multiplexamos en el circuito con transistores 2n222 lo que nos permitió mostrar una palabra “-hola-” también se hiso el uso del delay

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

MICROCONTROLADORES

PRÁCTICA N°6

PROFESOR:

PÉREZ PÉREZ JOSÉ JUAN

ALUMNOS:

TORRES SEGURA RICARDO

VARELA CRUZ CÉSAR

Gutiérrez González Norel

GRUPO: 3CM6

MATERIALES

- ✓ 7 Resistencias de 330 ohms
- ✓ 2 PUSH Botton
- ✓ 1 Displays de 7 segmentos ánodo común
- ✓ Microcontrolador ATmega8535

EQUIPO

● 1 Equipo de computo

● 1 fuente de voltaje

● 1 programador

● AVR studio 4

INTRODUCCION TEORICA

El bloque funcional Contador descendente (CTD) cuenta hacia atrás desde el valor prefijado al producirse un flanco positivo en la entrada de contaje atrás (CD). Si el valor actual (VA) es igual a cero, se activa el bit del contador. El contador se inicializa y carga el valor actual (CV) en el valor prefijado (PV) cuando se habilita la entrada de carga (LD). El contador atrás se detiene al alcanzar el valor cero.

El bloque funcional Contador ascendente (CTU) cuenta adelante desde el valor actual hasta el valor prefijado al producirse un flanco positivo en la entrada de contaje adelante (CU). Si el valor actual (VA) es mayor o igual al valor prefijado (PV), se activa el bit del contador. El contador se inicializa al activarse la entrada de desactivación (R). El contador ascendente no se detiene hasta llegar al máximo valor que puede albergar en la variable (VA), es decir, 32.767.

Una interrupción es un evento que hace que el microcontrolador deje de ejecutar la tarea que está realizando para atender dicho acontecimiento y luego regrese y continúe la tarea que estaba realizando antes de que se presentara la interrupción.

El pic 16F628 (y el 16F628A) tiene 10 fuentes de interrupción, si las interrupciones están habilitadas cada vez que una de estos acontecimientos se presente el pic dejará de ejecutar el programa para ir a atender la interrupción y al término de la misma continuará ejecutando el programa donde lo había dejado. Las fuentes de interrupción son:

Interrupción externa RB0/INT

Interrupción por cambio lógico en el puerto B (pines RB7 a RB4)

Interrupción por desborde del timer 0 (TMR0)

Interrupción por desborde del timer 1 (TMR1)

Interrupción por comparación exitosa exitosa en TMR2

Interrupción del comparador

Interrupción del transmisor del USART

Interrupción del receptor del USART

Interrupción del módulo CCP

Interrupción del EEPROM

Aunque el pic cuenta con 10 fuentes distintas de interrupción solamente tiene un Vector de interrupción por lo que si se habilitan varias interrupciones al momento de Presentarse cualquiera de ellas el programa saltara a la misma rutina de interrupción y es responsabilidad del programador crear una rutina que identifique la fuente de la Interrupción.

Los registros asociados con las interrupciones son el registro de control de Interrupción INTCON, el registro habilitación de interrupciones de periféricos PIE1 y El registro de interrupciones de periféricos PIR1. En el registro INTCON se Encuentra el bit de habilitación global de interrupciones GIE, el bit de habilitación de Interrupción por periféricos PEIE y los bits de habilitación de algunas interrupciones Como la interrupción externa del pin RB0 (INTE), la interrupción por cambio de Estado en los pines RB4 a RB7 (RBIE) y la interrupción por desborde del timer 0 (T0IE), así como las banderas correspondientes a cada interrupción (INTF, RBIF y T0IF). En el registro PIE1 se encuentran los bits de habilitación de las demás Interrupciones y en el registro PIR1 se encuentran las banderas asociadas con cada Interrupción.

CÓDIGO

```
1. include "m8535def.inc"
2. .def aux = r16
3. .def cont = r18
4. .def cond = r17
5. rjmp main
6. rjmp apaga
7. rjmp prende
8. main:
9. ldi aux, low(RAMEND)
10. out spl, aux
11. ldi aux, high(RAMEND)
12. out sph, aux
13. ser aux
14. out DDRA, aux
15. out PORTD, aux
16. ldi aux, $0f
17. out mcucr, aux
18. ldi aux, $c0
19. out gicr, aux
20. sei
21. ldi r20, $81
22. ldi r21, $f3
23. ldi r22, $c8
24. ldi r23, $e0
25. ldi r24, $b2
26. ldi r25, $a4
27. ldi r26, $84
```

```
28. ldi r27, $f1
29. ldi r28, $80
30. ldi r29, $a0
31. clr zh
32. clr cont
33. ser aux
34.
35. fin:
36. rcall deco
37. rjmp fin
38.
39. apaga:
40. rcall delay
41. cpi cont, 9
42. breq reset
43. inc cont
44. rcall deco
45. reti
46.
47. prende:
48. rcall delay
49. cpi cont, 0
50. breq reset2
51. dec cont
52. rcall deco
53. reti
54.
55. delay:
```

```
56. ldi r18, 208
57. ldi r19, 202
58. L1: dec r19
59. brne L1
60. dec r18
61. brne L1
62. nop
63.
64. deco:
65. clr zh
66. ldi zl, 20
67. add zl, cont
68. ld cond, z
69. out PORTA, cond
70. ret
71.
72. reset:
73. clr cont
74. reti
75.
76. reset2:
77. ldi cont, $09
78. rcall deco
79. reti
```

CONCLUSIONES

RICARDO TORRES SEGURA

En esta práctica pudimos observar el funcionamiento del contador ascenderme y descendente el cual funcionara presionando un pushbotton para hacer que la señal cambie y comience a contar en la forma ascendente o descendente.

VARELA CRUZ CÉSAR

Para la realización de esta práctica utilizamos las interrupciones, esto hace que cuando el micro estuviera haciendo alguna rutina esta se interrumpiera al nosotros mandar la señal con un pushbotton.

Gutiérrez González Norel

En el desarrollo de esta práctica aprendimos como hacer uso de las interrupciones del microcontrolador, programándole acciones específicas que queramos que realicen, en esta práctica las acciones fueron incrementar o decremento a un numero mostrado en un display

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

MICROCONTROLADORES

PRÁCTICA N° 7

PROFESOR:

PÉREZ PÉREZ JOSÉ JUAN

ALUMNOS:

TORRES SEGURA RICARDO

VARELA CRUZ CÉSAR

GUTIÉRREZ GONZÁLEZ NOREL

GRUPO: 3CM6

MATERIALES

- ✓ 7 Resistencias de 330 ohms
- ✓ 6 2n2222
- ✓ 6 Displays de 7 segmentos ánodo común
- ✓ Microcontrolador ATmega8535L

EQUIPO

- ✓ AVR studio 4
- ✓ 1 Equipo de cómputo con XP
- ✓ 1 programador
- ✓ 1 fuente de voltaje

INTRODUCCION TEORICA

El Registro de control del temporizador / contador - TCCR0 es el siguiente:

Bit	7	6	5	4	3	2	1	0	
	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	TCCR2
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR2 Registrarse

Registro TCCR0 En este momento, nos concentraremos en los bits resaltados. Los otros bits serán discutidos cuando sea necesario. Al seleccionar estos tres bits de selección de reloj, CS02: 00, configuramos el temporizador seleccionando el preescalador adecuado. Las posibles combinaciones se muestran a continuación.

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Clock Select Bit Descripción

El Registro de control del temporizador / contador - TCCR2 es el siguiente:

Bit	7	6	5	4	3	2	1	0	
	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	TCCR2
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR2 Registrarse

Ya que nos ocuparemos del modo CTC más adelante, solo nos preocupan los Bits2: 0 - CS22: 20 - Bits de selección de reloj. A diferencia de otros temporizadores, TIMER2 nos ofrece una amplia gama de prescalers para elegir. En TIMER0 / 1, los preescaladores disponibles son 8, 64, 256 y 1024, mientras que en TIMER2, tenemos 8, 32, 64, 128, 256 y 1024.

CS22	CS21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk_{T2S}/(No prescaling)
0	1	0	clk_{T2S}/8 (From prescaler)
0	1	1	clk_{T2S}/32 (From prescaler)
1	0	0	clk_{T2S}/64 (From prescaler)
1	0	1	clk_{T2S}/128 (From prescaler)
1	1	0	clk_{T2S}/256 (From prescaler)
1	1	1	clk_{T2S}/1024 (From prescaler)

Clock Select Bit Descripción

CÓDIGO

```
.include "m8535def.inc"
.def aux = r16
.def col = r17
.def aux2 = r18
    rjmp main
    .org $004
    rjmp cuenta
    .org $009
    rjmp cuenta2

main:
    ldi aux, low(RAMEND)
    out spl, aux
    ldi aux, high(RAMEND)
    out sph, aux
    ser aux
    out DDRA, aux
    out DDRC, aux
    ldi aux, $fe
    mov r0, aux
    ldi aux, $90
```

```
22.     mov r1, aux
23.     ldi aux, $8f
24.     mov r2, aux
25.     ldi aux, $81
26.     mov r3, aux
27.     ldi aux, $92
28.     mov r4, aux
29.     ldi aux, $fe
30.     mov r5, aux
31.     ldi aux, $02
32.     out tccr0, aux
33.     ldi aux, $07
34.     out tccr2, aux
35.     ldi aux, $41
36.     out timsk, aux
37.     sei
38.
39. fin:
40.     out PORTA, aux2
41.     out PORTC, col
42.     rjmp fin
43.
44. cuenta:
45.     clr zl
46.     ldi col, $04
47.     ld aux2, z+
48.     reti
49.
50. cuenta2:
51.     lsl col
52.     ld aux2, z+
53.     reti
54.
```

CONCLUSIONES

RICARDO TORRES SEGURA

En esta práctica se muestra un mensaje corriendo en los displays. Haciendo uso de timers se puede llegar a recorrer un mensaje largo usando displays limitados.

VARELA CRUZ CÉSAR

Al igual que en la práctica 5 se mostró la palabra hola, con la excepción de que esta vez la palabra fuera recorriendo los displays de izquierda a derecha, para esto hicimos uso de dos timers, uno para mostrar la palabra y otro que fuera recorriendo las letras en los displays, lo que resultando complicado fue poder mostrar el corrimiento a una velocidad que se pudiera distinguir.

Gutiérrez González Norel

Para desarrollar esta práctica utilizamos la Practica 5 como base, solo que en vez de usar un delay utilizamos timers, esto con el fin de lograr mostrar mejor nuestro mensaje utilizando preescalas del reloj del microcontrolador, y además,

este programa muestra como nuestro mensaje se va recorriendo por todos los displays conectados a nuestro microcontrolador.

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

MICROCONTROLADORES

PRÁCTICA N° 8

PROFESOR:

PÉREZ PÉREZ JOSÉ JUAN

ALUMNOS:

TORRES SEGURA RICARDO

VARELA CRUZ CÉSAR

GUTIÉRREZ GONZÁLEZ NOREL

GRUPO: 3CM6

Materiales

- ✓ 1 LED
- ✓ 1 resistencia de 330 ohms
- ✓ 2 potenciómetros de 10 kilo ohms
- ✓ Microcontrolador ATMEGA8535

Equipo

- ✓ AVR studio 4
- ✓ 1 Equipo de cómputo con XP
- ✓ 1 programador
- ✓ 1 fuente de voltaje

Introducción Teórica

Esta práctica es un elemento de seguridad pasiva. Esto significa que no evitan una situación anormal, pero sí son capaces de advertir de ella, cumpliendo así, una función disuasoria frente a posibles problemas.1

Por ejemplo:

1. La intrusión de personas.
2. Inicio de fuego.
3. El desbordamiento de un tanque.
4. La presencia de agentes tóxicos.
5. Cualquier situación que sea anormal para el usuario.

Son capaces además de reducir el tiempo de ejecución de las acciones a tomar en función del problema presentado, reduciendo así las pérdidas.

Código

```
.include "m8535def.inc"
.def aux = r16
.def aux2 = r17
.def aux3 = r18
rjmp main
.org $004
rjmp tono
.org $008
rjmp Cseg
rjmp cuenta
main: ldi aux,low(ramend)
out spl, aux
ldi aux, high(ramend)
```

```
out sph,aux
ser aux
out ddrc,aux
ldi aux,1
out portb,aux
ldi aux,6
out tccr0,aux
ldi aux,2
out tccr2,aux
ldi aux,4
out tccr1b,aux
ldi aux,1
out timsk,aux
sei
ldi aux,250
out tcnt0,aux
loop:
nop
nop
rjmp loop
tono:
ldi aux2,113
out tcnt2,aux2
ser aux
in r17,pinc
eor r17,aux
out portc,r17
reti
Cseg:
ldi aux2,1
out timsk,aux2
reti
cuenta:
ldi aux3,250
out tcnt0,aux3
ldi aux3,$b5
out tcnt1l,aux3
ldi aux3,$b3
out tcnt1h,aux3
ldi aux3,$45
```

out timsk,aux3
reti

Conclusiones

TORRES SEGURA RICARDO

En esta práctica se desarrolló un pequeño sistema de alarma, en el cual, cada que pasaban 5 personas, se activaba una alarma durante un periodo de tiempo establecido.

VARELA CRUZ CÉSAR

Durante la práctica pudimos aprender a programar un timer que contaba las veces que se presionaba un botón, y otro timer que contaba el tiempo que iba a estar sonando la alarma.

GUTIÉRREZ GONZÁLEZ NOREL

En el desarrollo de esta práctica pudimos verificar el correcto funcionamiento de un mecanismo de alarma, en el cual, tenía un botón y al presionarse 5 veces seguidas, sonaba una alarma durante 5 segundos.

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

MICROCONTROLADORES

PRÁCTICA N° 9

PROFESOR:

PÉREZ PÉREZ JOSÉ JUAN

ALUMNOS:

TORRES SEGURA RICARDO

VARELA CRUZ CÉSAR

GUTIÉRREZ GONZÁLEZ NOREL

GRUPO: 3CM6

Materiales:

- ✓ 1 LED
- ✓ 1 resistencia de 330 ohms
- ✓ 2 potenciómetros de 10 kilo ohms
- ✓ Microcontrolador ATMEGA8535

Equipo:

- ✓ AVR studio 4
- ✓ 1 Equipo de cómputo con XP
- ✓ 1 programador
- ✓ 1 fuente de voltaje

Introduccion Teorica:

La delincuencia en general sigue en aumento, ahora tener instalada una alarma se hace necesario. ahora les ofrecemos un sistema de alarma de seguridad simple y compacta para proteger tu casa, taller y objetos de valor. Esta alarma utiliza como control el microcontrolador PIC12F675P. Además, de un sensor de infrarrojos pasivo (PIR), el módulo está integrado con el sistema de alarma para la detección de movimiento.

Esperamos que la ensamblen y pueda serles útil para la protección de casas y todo lo que sea de valor.

Como la salida del sistema de alarma puede ser conectado a lámparas externas o sirenas de alarma, estos dispositivos se activan al instante cuando se detecta el movimiento. Como resultado de ello, el intruso que entró en el área protegida, incluso en la oscuridad total, al instante estará expuesto.

Código

```
.include "m8535def.inc"
.macro motor
ldi r16,1
out porta,r16
ldi r16,@0
loop:
rcall delay
dec r16
brne loop
out porta,r16
ldi r16,@1
loop2:
```

```
rcall delay
dec r16
brne loop2
.endm
ldi r16,low(RAMEND)
out spl,r16
ldi r16,high(RAMEND)
out sph,r16
ser r16
out ddra,r16
out portd,r16
prgnt:
sbis pind,7
rjmp cero
sbis pind,6
rjmp cuatro5
sbis pind,5
rjmp nueve0
rjmp prgnt
cero:
motor 3,37
rjmp prgnt
cuatro5:
motor 2,38
rjmp prgnt
nueve0:
motor 4,36
rjmp prgnt
delay:
ldi r17,$A6
WGLOOP0:
dec r17
brne WGLOOP0
nop
ret
```

Conclusiones

TORRES SEGURA RICARDO

Aprendimos a usar un servomotor en esta practica haciendo uso del lenguaje ensamblador. Hay que ser un poco cuidadosos con las instrucciones ya que puede que el servomotor no funcione correctamente.

VARELA CRUZ CÉSAR
En esta práctica usamos un servomotor y pudimos cotrolar su angulo de giro haciendo uso de 3 botone, donde cada botón le daba un angulo diferente de giro.

GUTIÉRREZ GONZÁLEZ NOREL
En la práctica pudimos hacer girar un servomotor de 180 grados. Utilizando 3 botonoes, le pudimos dar diferentes rotaciones al presionar cada botón. Fue un poco difícil porque al principio nos equivocamos en el código y pensamos que no servia el servomotor.