INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

INTRODUCCIÓN A LOS MICROCONTROLADORES

PROFESOR:

Aguilar Sanchez Fernando

ALUMNOS:

Arruti Sánchez Alondra
Carrillo Soto Cristian Eduardo

Grupo: 3CM14

Fecha de Entrega: 05/10/22

PRÁCTICA 6

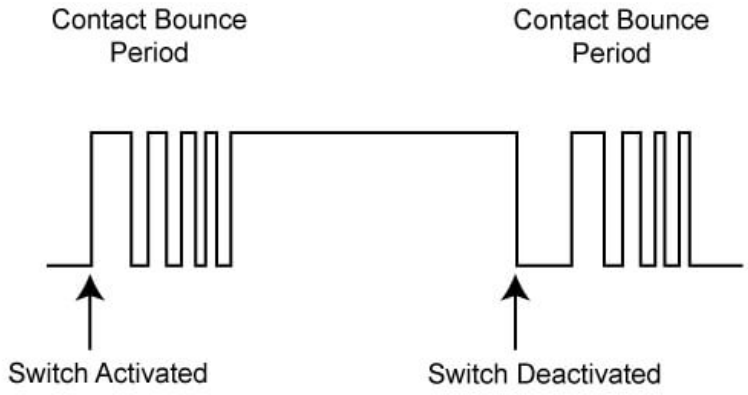
Contador sin rebotes

Introducción

Los botones, push button e interruptores son objetos mecánicos. Esta circunstancia produce un efecto que es muy importante tener en cuenta a la hora de procesar la señal que le envía al PIC: el rebote.

La conexión o interrupción del contacto eléctrico generada por estos cacharros depende mecánicamente de un muelle o fleje que hace que el contacto permanezca estable, sin embargo el proceso de cierre o apertura de dicho contacto no se produce de una sola vez y para siempre, sino que por el contrario hay una secuencia más o menos larga de conexiones y desconexiones, dependiendo de la potencia del muelle o fleje, que hace que en vez de tener un paso claro de ON a OFF, o viceversa, lo que tenemos es todo un tren de pulsos, cada vez más cortos, hasta que se estabiliza en uno u otro estado.

Al ser el PIC mucho más rápido en su ejecución que las múltiples señales que recibe y al ejecutar nuestro programa funciones distintas dependiendo del estado ON/OFF de entrada es fácilmente observable que dichas funciones se dispararán tantas veces como rebotes tengamos en los pulsadores o interruptores.



Los resultados de lanzar la ejecución de estas funciones muchas veces, pudiendo incluso superponerse entre ellas, puede llevar a nuestro programa a cometer errores, inconsistencia o a resultados indefinidos.

En el contexto de la ingeniería eléctrica y electrónica, un interruptor es un componente que puede "hacer" o "romper" un circuito eléctrico, interrumpiendo así una corriente eléctrica o desviándola de un conductor a otro.

Como pueden atestiguar los ingenieros, hay muchos tipos diferentes de interruptores conmutadores, como los de palanca, los basculantes, los pulsadores, los micro interruptores y los interruptores de límite, los magnéticos y los de láminas, y los relés. Todos tienen una cosa en común: rebotan.

Ilustración 1 Ejemplos de interruptores

Normalmente, este rebote tiene poco o ningún efecto en el circuito, pero si el circuito digital es lo suficientemente rápido como para detectar y responder a múltiples rebotes, puede haber graves consecuencias. La tarea de un ingeniero es evitar o mitigar los efectos de este rebote, o "rebotar" el interruptor. Aunque la industria ha practicado durante mucho tiempo el rebote por hardware, más recientemente se ha pasado a enfoques de rebote por software. Sin embargo, hay situaciones en las que el rebote por hardware es una mejor opción.

Códigos

1. Código de la configuración de los periféricos

```
1.  // I/O Registers definitions
2.  #include <mega8535.h>
3.  #include <delay.h>
4.  #define boton PIND.0
5.  const char tabCatodo [10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7c,0x07,0x7f,0x6f};
6.  unsigned char cont;
7.  bit Bx;
8.  bit By;
9.
10. void main(void)
11. {
12. // Declare your local variables here
13. // Input/Output Ports initialization
14. // Port A initialization
15. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
16. DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) | (0<<DDA1) |
    (0<<DDA0);
17. // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
18. PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (0<<PORTA2) |
    (0<<PORTA1) | (0<<PORTA0);
19. // Port B initialization
20. // Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
21. DDRB=(1<<DDB7) | (1<<DDB6) | (1<<DDB5) | (1<<DDB4) | (1<<DDB3) | (1<<DDB2) | (1<<DDB1) |
    (1<<DDB0);
22. // State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
23. PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PORTB2) |
    (0<<PORTB1) | (0<<PORTB0);
24. // Port C initialization
25. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
26. DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<DDC1) |
    (0<<DDC0);
27. // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
28. PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) |
    (0<<PORTC1) | (0<<PORTC0);
29. // Port D initialization
30. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
31. DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) | (0<<DDD1) |
    (0<<DDD0);
32. // State: Bit7=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P
```

```
33. PORTD=(1<<PORTD7) | (1<<PORTD6) | (1<<PORTD5) | (1<<PORTD4) | (1<<PORTD3) | (1<<PORTD2) |
    (1<<PORTD1) | (1<<PORTD0);
34. // Timer/Counter 0 initialization
35. // Clock source: System Clock
36. // Clock value: Timer 0 Stopped
37. // Mode: Normal top=0xFF
38. // OC0 output: Disconnected
39. TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) | (0<<CS00);
40. TCNT0=0x00;
41. OCR0=0x00;
42. // Timer/Counter 1 initialization
43. // Clock source: System Clock
44. // Clock value: Timer1 Stopped
45. // Mode: Normal top=0xFFFF
46. // Noise Canceler: Off
47. // Input Capture on Falling Edge
48. // Input Capture Interrupt: Off
49. // Compare A Match Interrupt: Off
50. // Compare B Match Interrupt: Off
51. TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WGM10);
52. TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) | (0<<CS10);
53. TCNT1H=0x00;
54. TCNT1L=0x00;
55. ICR1H=0x00;
56. ICR1L=0x00;
57. OCR1AH=0x00;
58. OCR1AL=0x00;
59. OCR1BH=0x00;
60. OCR1BL=0x00;
61. // Timer/Counter 2 initialization
62. // Clock source: System Clock
63. // Clock value: Timer2 Stopped
64. // Mode: Normal top=0xFF
65. // OC2 output: Disconnected
66. ASSR=0<<AS2;
67. TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<CS21) | (0<<CS20);
68. TCNT2=0x00;
69. OCR2=0x00;
70. // Timer(s)/Counter(s) Interrupt(s) initialization
71. TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) | (0<<OCIE0)
    | (0<<TOIE0);
72. // External Interrupt(s) initialization
73. MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
74. MCUCSR=(0<<ISC2);
```

```
75. UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<UCSZ2) | (0<<RXB8) |
    (0<<TXB8);
76. // Analog Comparator initialization
77. // Analog Comparator: Off
78. // The Analog Comparator's positive input is
79. // connected to the AIN0 pin
80. // The Analog Comparator's negative input is
81. // connected to the AIN1 pin
82. ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) | (0<<ACIS1) | (0<<ACIS0);
83. SFIOR=(0<<ACME);
84. // ADC initialization
85. // ADC disabled
86. ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<ADPS2) | (0<<ADPS1) |
    (0<<ADPS0);
87. // SPI initialization
88. // SPI disabled
89. SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) | (0<<SPR1) | (0<<SPR0);
90. // TWI initialization
91. // TWI disabled
92. TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);
```

2. Código del programa principal en C

```
while (1){
      if (boton==0)
        Bx=1;
      else
        Bx=0;

      if ((Bx==0) && (By==1))//Cambio de 0 a 1
      {
        cont++;
        if (cont==10)
            cont=0;
        delay_ms(40);
      }

      PORTB=tabCatodo [cont];

      if (Bx==1){      //Cambio de 1 a 0
        By=1;
        delay_ms(40);
      }

      if ((Bx==0)&&(By==1))
        By=0;
};
```

Circuitos

1. Circuito simulado en Proteus

En la figura 1 se puede observar el contador ascendente con el que se ha trabajado en las dos prácticas anteriores, realmente en cuanto a la simulación no presenta cambio alguno con respecto al comportamiento de la práctica anterior, esto debido a que el simulador no considera los rebotes como ya se ha mencionado con anterioridad y por ende no presenta los errores que está práctica pretende resolver, limitando la verificación de la implementación a lo que se presente en el circuito físico.

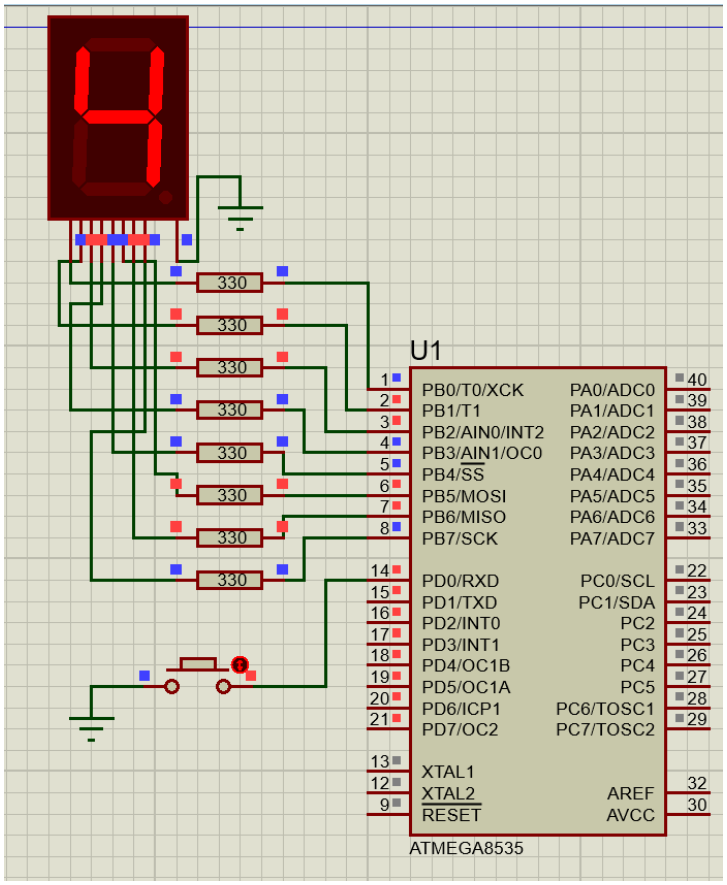


Figura. 1 Contador en su cuarto incremento

1. Circuito armado en la Proto

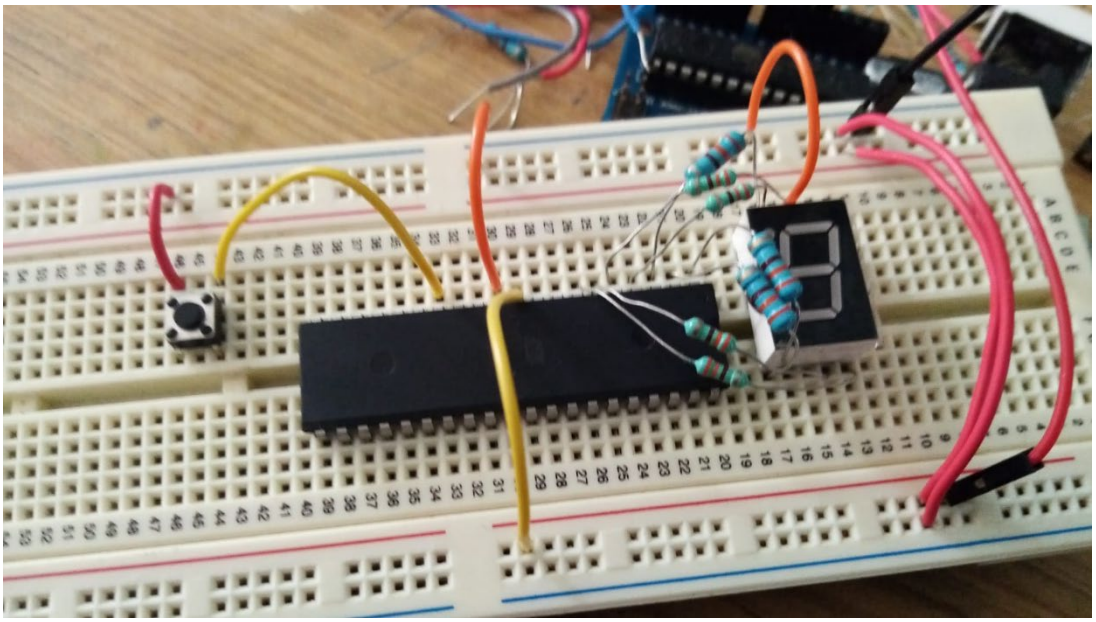


Figura. 2 Contador sin incremento

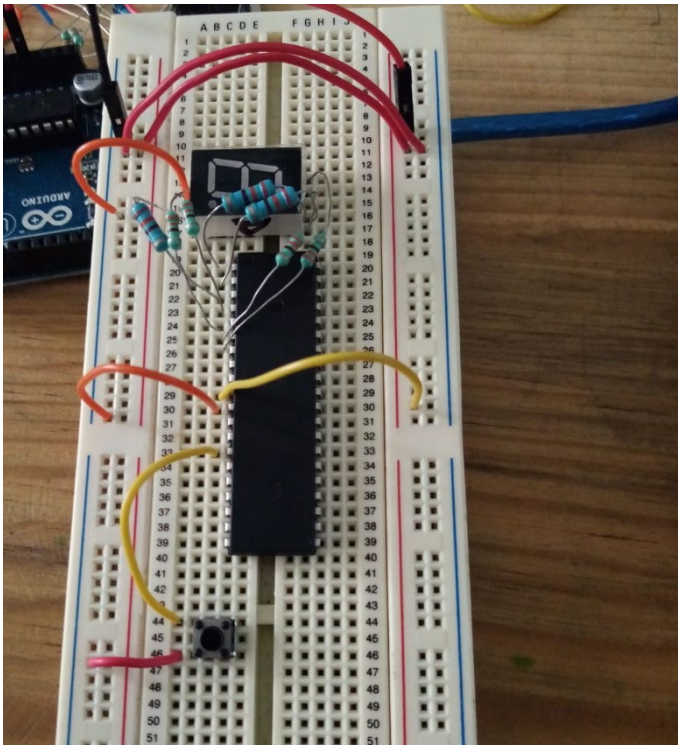


Figura. 3 Contador armado en la protoboard

Conclusiones

Al llegar a la presente práctica ya se ha trabajado con el código de forma previa por lo que no representa un gran desafío el desarrollarla, puesto que se reutiliza el código previamente implementado en la práctica anterior con el pequeño cambio de segmentar el flanco que se ha escogido para incrementar y de esta forma agregarle un delay o retraso que se encarga de discernir el ruido que se genera durante ese flanco, así como agregar otro delay a la bandera que detecta el cambio en el pin de entrada (el flanco que no se analiza) para evitar que el ruido de esa zona intervenga con el valor resultante.

Finalmente, tras realizar tres prácticas relacionadas tal vez la forma en la que se dividen no es la mejor puesto que la primera es una implementación de igualaciones e incrementos que si bien hace conciencia de los errores estos podrían resolverse dentro de la misma práctica como dos partes de un solo trabajo, de igual forma para el caso de la presente práctica no debería ser un trabajo separado del anterior, pues este es el que conlleva la implementación final, para obtener los resultados deseados tanto en las simulaciones como en el desarrollo físico.

Arruti Sánchez Alondra

Al momento de realizar el código, por mi parte se me complico ya que no había programado un microcontrolador por mi cuenta ya que en instrumentación vimos como programar un Arduino mediante software especializado en Arduino y no como tal un compilador, sin embargo gracias a mi compañera comprendí mejor el cómo se debía programar y como funcionaba ya que no entendía muy bien lo de los saltos en los botones, sin embargo, con la investigación que se realizo para esta práctica supe cómo funcionaba y también gracias la circuito vimos dicho funcionamiento.

Aunque el circuito fue el mismo para las dos practicas anteriores observamos como cada una funcionaba de diferente manera y como se controlaban diferente, es como un botón para un videojuego, en un juego con el botón “A” puede ser para atacar, pero con otro puede ser para atacar por lo que me gusta verlo de esa manera para entenderlo mejor.

Carrillo Soto Cristian Eduardo

Bibliografía

[1] Blocked. (s. f.). Recuperado 5 de octubre de 2022, de https://www.digikey.com.mx/es/articles/how-to-implement-hardware-debounce-for-switches-and-relays

[2] García, V. (2010b, noviembre 13). Pulsadores – Electrónica Práctica Aplicada. Recuperado 5 de octubre de 2022, de https://www.diarioelectronicohoy.com/blog/pulsadores-sin-rebotes