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 5

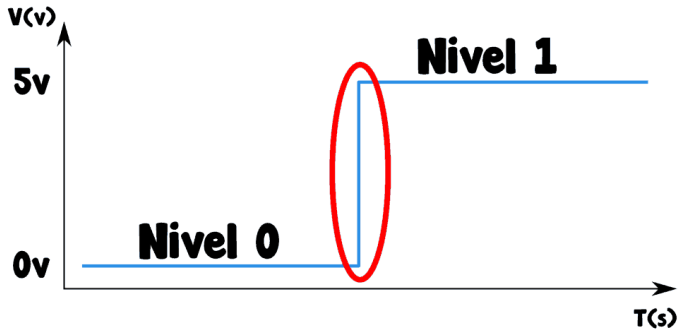
Contador activado por flancos

Introducción

En electrónica digital, se representa, traslada, encripta, calcula y almacena la información de forma binaria. El concepto es sencillo ya que solo tenemos dos estados 0 y 1, pero en realidad estos estados hay que pasarlos a niveles eléctricos. En la realidad trabajamos con diferencias de potencial.

Una forma clásica de definir el 0 y el 1 es con dos niveles de tensión estándar como se aprecia en la imagen anterior. En este sistema el 1 se representa con 5v y el 0 con 0 voltios. Aun siendo una forma clásica de entender la transmisión de datos es una de las más usadas. En realidad, no tienen por qué ser 5v, puede ser otro valor de tensión, por ejemplo 3.3v o 1.8v.

Lo más importante de este método no es el valor de tensión que representa el 1 lógico, sino que lo más importante es que el 1 se representa con un valor de tensión positivo y el 0 con la masa del circuito.

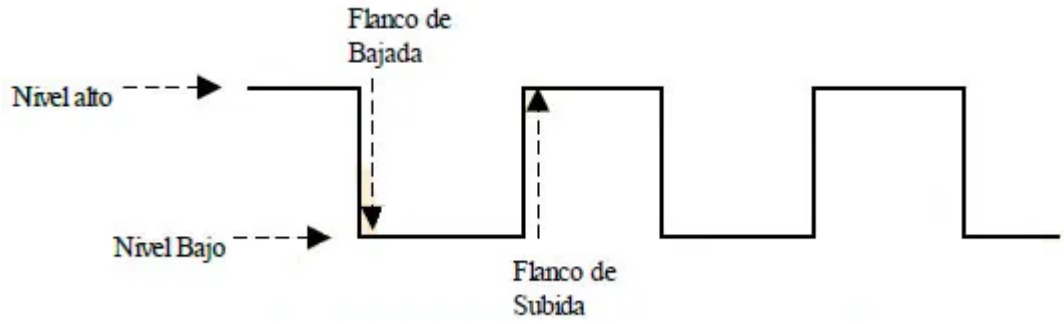


Gráfica 1 Flanco de subida

Se puede decir que el diagrama anterior representa dos estados, el primero es el estado 0 que se representa con la línea horizontal baja y el segundo el estado 1 que está representado con la línea horizontal alta.

La línea vertical es el FLANCO. Un flanco es la transición entre dos estados lógicos, en este caso concreto pasamos de estar en un estado bajo a un estado alto.

En inglés al flanco se le denomina EDGE, cuya traducción literal podría ser borde o canto y es que es justamente esto, el borde entre dos estados, el borde entre pasar de un estado bajo a un estado alto. ¿Y si se le da la vuelta? Es decir, y si pasamos de un estado alto a un estado bajo, pues también es un flanco.



Gráfica 2 Flancos de bajada y subida

Los flancos son indiferentes a si se pasa de un estado bajo a un estado alto o si se pasa de un estado alto a un estado bajo, ambos son flancos, pero no son iguales.

Un flanco de subida es el que pasa de estar en nivel bajo a estar en nivel alto ¿por qué? Piensa que si se desplaza desde la izquierda a la derecha y se requiere seguir por encima de la línea se tiene que subir, es como subir un escalón. Los flancos de subida se denominan en inglés rising.

Por otro lado, el flanco de bajada es aquel en el que, si se va de izquierda a derecha y se requiere de bajar el escalón, es decir estar debajo de la línea. Los flancos de bajada se denominan en inglés falling.

Como norma general los flancos de subida se suelen representar con una flecha hacia arriba y los flancos de bajada con una flecha hacia abajo, como se muestra en las dos gráficas anteriores.

Códigos

1. Código de la configuración de los periféricos

```
1.  // I/O Registers definitions
2.  #include <mega8535.h>
3.  #define boton PIND.0
4.  const char tabCatodo [10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7c,0x07,0x7f,0x6f};
5.  unsigned char cont;
6.  bit Bx;
7.  bit By;
8.
9.  // Declare your global variables here
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. PORTD=(1<<PORTD7) | (1<<PORTD6) | (1<<PORTD5) | (1<<PORTD4) | (1<<PORTD3) | (1<<PORTD2) |
    (1<<PORTD1) | (1<<PORTD0);
```

```
33. // Timer/Counter 0 initialization
34. // Clock source: System Clock
35. // Clock value: Timer 0 Stopped
36. // Mode: Normal top=0xFF
37. // OC0 output: Disconnected
38. TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) | (0<<CS00);
39. TCNT0=0x00;
40. OCR0=0x00;
41. // Timer/Counter 1 initialization
42. // Clock source: System Clock
43. // Clock value: Timer1 Stopped
44. // Mode: Normal top=0xFFFF
45. // OC1A output: Disconnected
46. // OC1B output: Disconnected
47. // Noise Canceler: Off
48. // Input Capture on Falling Edge
49. // Timer1 Overflow Interrupt: Off
50. // Input Capture Interrupt: Off
51. // Compare A Match Interrupt: Off
52. // Compare B Match Interrupt: Off
53. TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WGM10);
54. TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) | (0<<CS10);
55. TCNT1H=0x00;
56. TCNT1L=0x00;
57. ICR1H=0x00;
58. ICR1L=0x00;
59. OCR1AH=0x00;
60. OCR1AL=0x00;
61. OCR1BH=0x00;
62. OCR1BL=0x00;
63. // Timer/Counter 2 initialization
64. // Clock source: System Clock
65. // Clock value: Timer2 Stopped
66. // Mode: Normal top=0xFF
67. // OC2 output: Disconnected
68. ASSR=0<<AS2;
69. TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<CS21) | (0<<CS20);
70. TCNT2=0x00;
71. OCR2=0x00;
72. // Timer(s)/Counter(s) Interrupt(s) initialization
73. TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) | (0<<OCIE0)
    | (0<<TOIE0);
74.
75. // External Interrupt(s) initialization
76. // INT0: Off
```

```
77. // INT1: Off
78. // INT2: Off
79. MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
80. MCUCSR=(0<<ISC2);
81. // USART initialization
82. // USART disabled
83. UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<UCSZ2) | (0<<RXB8) |
    (0<<TXB8);
84. // Analog Comparator initialization
85. // Analog Comparator: Off
86. // The Analog Comparator's positive input is
87. // connected to the AIN0 pin
88. // The Analog Comparator's negative input is
89. // connected to the AIN1 pin
90. ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) | (0<<ACIS1) | (0<<ACIS0);
91. SFIOR=(0<<ACME);
92. // ADC initialization
93. // ADC disabled
94. ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<ADPS2) | (0<<ADPS1) |
    (0<<ADPS0);
95. // SPI initialization
96. // SPI disabled
97. SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) | (0<<SPR1) | (0<<SPR0);
98. // TWI initialization
99. // TWI disabled
100.       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))
        cont++;
      if (cont==10)
        cont=0;

      PORTB = tabCatodo [cont];
      if (Bx==1)
        By=1;
      if ((Bx==0)&&(By==1))
        By=0;
};
```

Circuitos

1. Circuito simulado en Proteus

En la figura 1 se muestra el circuito armado que es el mismo que el de la práctica anterior, sin embargo, en este caso se cuenta con un código que permite detectar el flanco de subida, por lo que únicamente tras soltar el botón se realiza el incremento, al menos en el caso de la simulación puesto que el simulador no detecta rebotes.

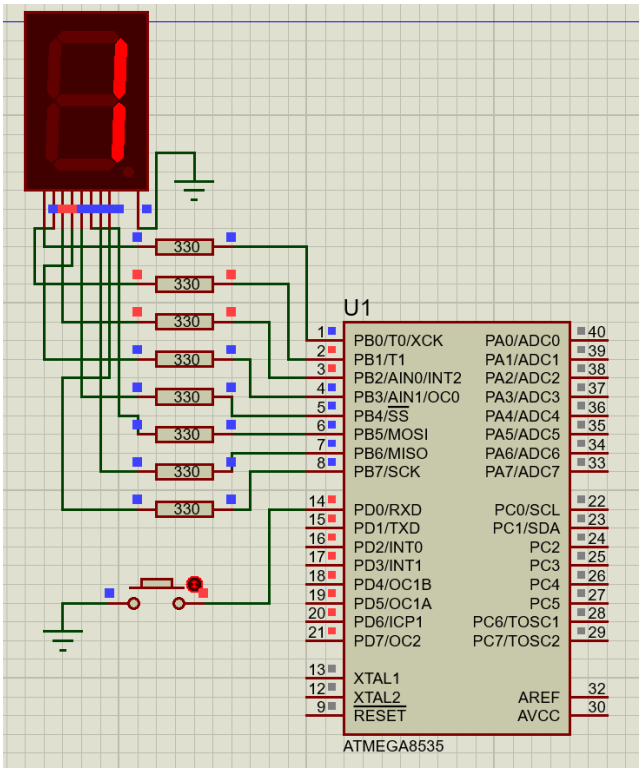
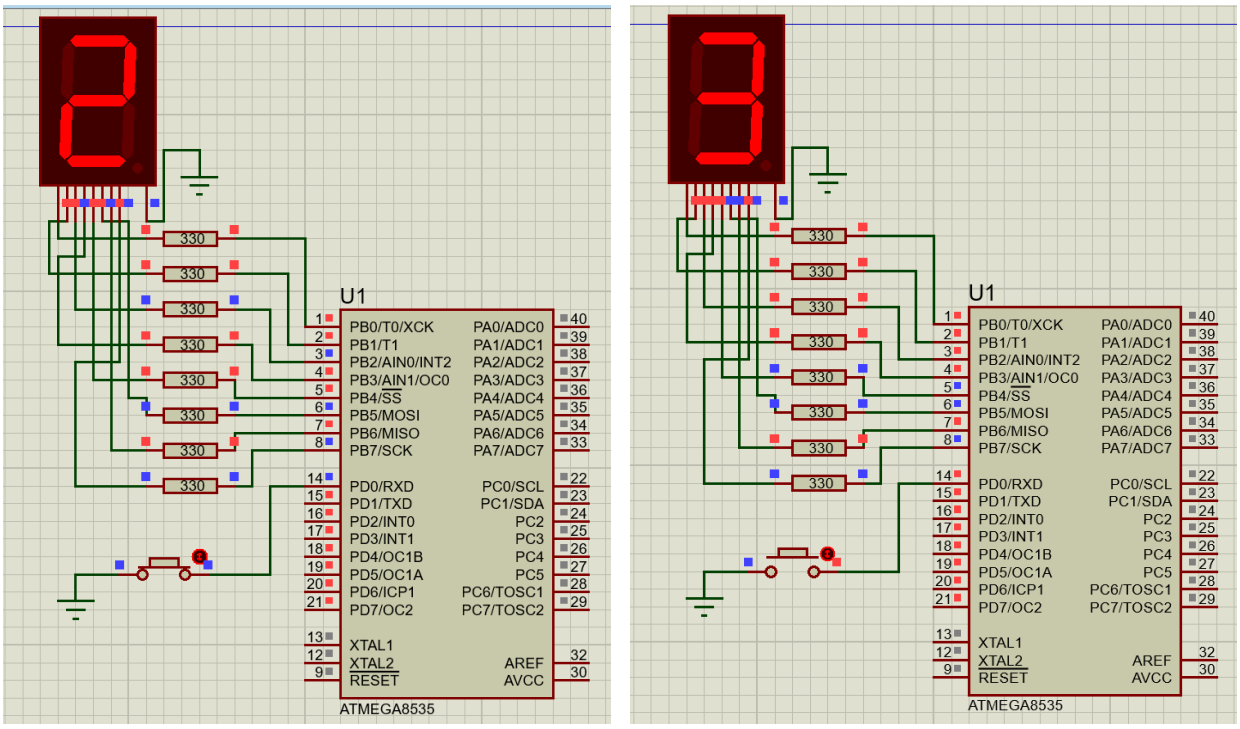


Figura. 1 Primer incremento del contador

Al realizar la implementación se utilizó parte de la lógica implementada en la práctica 2, puesto que, al llegar al nueve, el contador se reinicia, además de utilizar el arreglo antes visto para pasar los valores al display cátodo común.

En la figura 2 se muestra como se realiza el incremento siento la sección a el antes y la sección b el después.

Figura. 2 Incremento con flanco de subida



a) Botón presionado

b) Incremento tras soltar

1. Circuito armado en la Proto

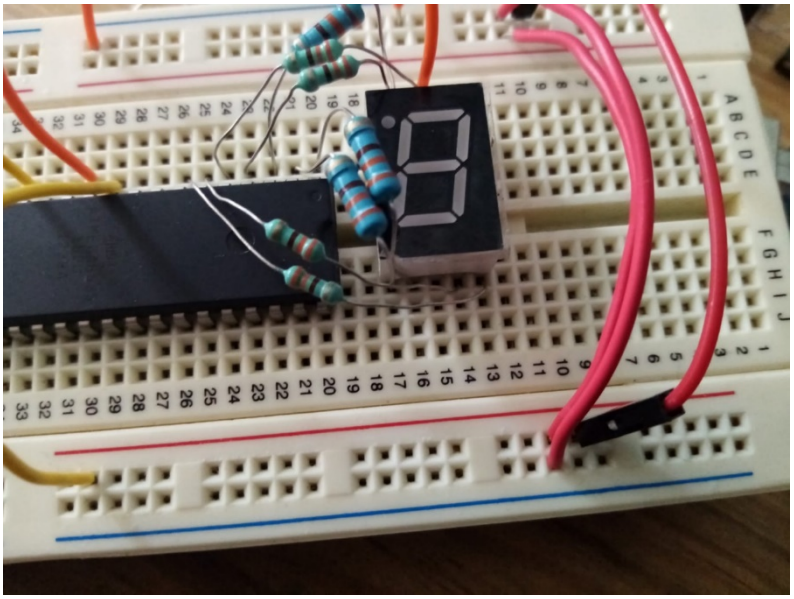


Figura.2 Display con Flancos

Conclusiones

Al realizar la presente práctica se optó por presentar un detector de flanco de subida puesto que el ejemplo aportado por el profesor correspondía a un detector de bajada y se pensó en desarrollar algo distinto, cabe mencionar que en las primeras versiones de la implementación de consideraba el valor del pin d (push button) como una variable para determinar los flancos, sin embargo esto provocaba errores como no mostrar ningún cambio o incrementar más de una vez el contador, puesto que el pin d puede ir cambiando conforme se ejecuta el código, generando errores en las condiciones.

Finalmente, se obtuvo el detector tomando la idea de usar bits independientes para marcar los cambios como se muestra en el formato de la práctica otorgado por el profesor obteniendo un contador que incrementa sin problemas en la simulación y que por ende presenta algunos errores debes en cuando al realizarlo de forma física debido al rebote que se pretender solucionar en la siguiente práctica.

Arruti Sánchez Alondra

En lo personal tenía mucho que no trabajaba con Flancos ya que en arquitectura de computadoras, la práctica no la pudimos realizar (en mi caso, mi compañera parece que si la logro realizar) debido a que nos cambiaron de profesora en el curso pasado, sin embargo tenía vago conocimiento de estos hasta ahora, donde pude observar el funcionamiento gracias al push button del circuito realizado ya que se vio un cambio con la practica anterior ya que el otro se veía como daba saltos al momento de presionar el botón y aquí se controlo ese aspecto en donde podemos observar como ya no da ese error y además de eso ya no entra ruido como antes, al igual que como un simple circuito puede hacer un funcionamiento tan complicado como lo son los flancos que en realidad ya vi que no es difícil de manejar.

Carrillo Soto Cristian Eduardo

Bibliografía

[1] Gómez, E. (2021, 6 mayo). Flanco de subida y bajada ¿Qué son? Rincón Ingenieril. Recuperado 5 de octubre de 2022, de https://www.rinconingenieril.es/flanco-subida-bajada/