INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO.

INTRODUCCIÓN A LOS MICROCONTROLADORES.

REPORTE PRÁCTICA 5.

PROFESOR: AGUILAR SANCHEZ FERNANDO

Grupo: 3CM6

Alumno: Rojas Alvarado Luis Enrique

Boleta: 2014010995

LINK DE VIDEO EXPLICATIVO: https://youtu.be/9wZP8rAOsIQ

Práctica 5: Contador de 0 a 9 activado por flancos

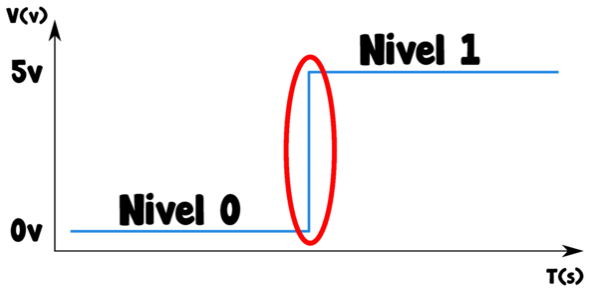
Introducción teórica

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, ya que en la realidad trabajamos con diferencias de potencial.

Un flanco de subida es el que pasa de estar en nivel bajo a estar en nivel alto ¿por qué? Piensa que si te desplazas desde la izquierda a la derecha y quieres seguir por encima de la línea tienes 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 vas de izquierda a derecha tienes que bajar el escalón. 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. [1]



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.

Es indiferente si se pasa de un estado bajo a un estado alto a si se pasa de un estado alto a un estado bajo, ambos son flancos, pero no son iguales.

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. [2]

Desarrollo experimental

1.- Diseñe un programa colocando en el Puerto B un Display. Coloque un Push Button en la terminal 0 del Puerto D para incrementar su cuenta del 0 al 9 activado por flancos.

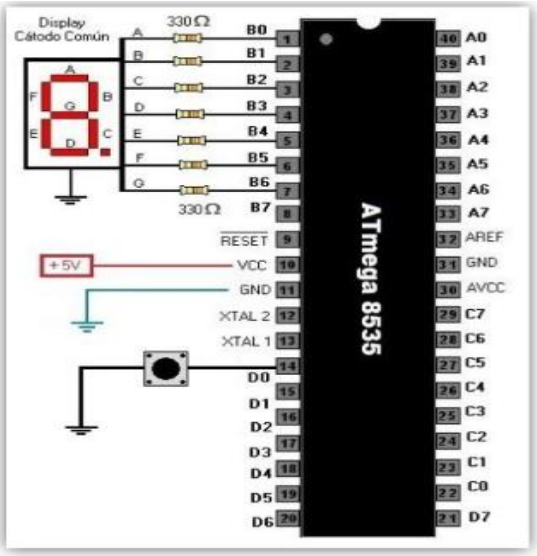


Figura 1. Circuito para el contador de 0 a 9 activado por flancos.

Código

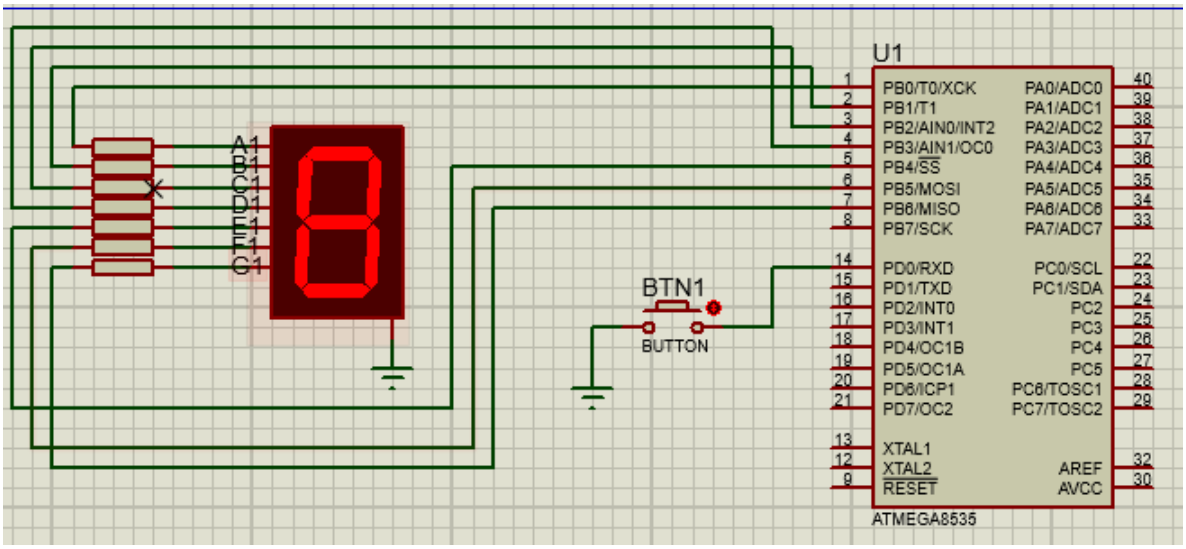
```
/*****************************************************
This program was created by the
CodeWizardAVR V2.60 Evaluation
Automatic Program Generator
© Copyright 1998-2012 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 05/11/2020
Author  :
Company :
Comments:


Chip type               : ATmega8535
Program type            : Application
AVR Core Clock frequency: 1.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 128
```

```
22. ****************************************************/
23.
24. #include <mega8535.h>
25.
26. #define boton PIND.0
27. bit botonp;
28. bit botona;
29. unsigned char var;
30. const char tabla7segmentos [10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7c,0x07,0x7f,0x6f
    };
31.
32.
33. void main(void)
34. {
35. // Declare your local variables here
36.
37. // Input/Output Ports initialization
38. // Port A initialization
39. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
40. DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) | (0<<D
    DA1) | (0<<DDA0);
41. // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
42. PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (0<<PO
    RTA2) | (0<<PORTA1) | (0<<PORTA0);
43.
44. // Port B initialization
45. // Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=O
    ut
46. DDRB=(1<<DDB7) | (1<<DDB6) | (1<<DDB5) | (1<<DDB4) | (1<<DDB3) | (1<<DDB2) | (1<<D
    DB1) | (1<<DDB0);
47. // State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
48. PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PO
    RTB2) | (0<<PORTB1) | (0<<PORTB0);
49.
50. // Port C initialization
51. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
52. DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<D
    DC1) | (0<<DDC0);
53. // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
54. PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PO
    RTC2) | (0<<PORTC1) | (0<<PORTC0);
55.
56. // Port D initialization
57. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
58. DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) | (0<<D
    DD1) | (0<<DDD0);
59. // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=P
60. PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (0<<PO
    RTD2) | (0<<PORTD1) | (1<<PORTD0);
61.
62. // Timer/Counter 0 initialization
63. // Clock source: System Clock
64. // Clock value: Timer 0 Stopped
65. // Mode: Normal top=0xFF
66. // OC0 output: Disconnected
67. TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) |
    (0<<CS00);
68. TCNT0=0x00;
69. OCR0=0x00;
70.
71. // Timer/Counter 1 initialization
```

```
72. // Clock source: System Clock
73. // Clock value: Timer1 Stopped
74. // Mode: Normal top=0xFFFF
75. // OC1A output: Disconnected
76. // OC1B output: Disconnected
77. // Noise Canceler: Off
78. // Input Capture on Falling Edge
79. // Timer1 Overflow Interrupt: Off
80. // Input Capture Interrupt: Off
81. // Compare A Match Interrupt: Off
82. // Compare B Match Interrupt: Off
83. TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WG
    M10);
84. TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) |
     (0<<CS10);
85. TCNT1H=0x00;
86. TCNT1L=0x00;
87. ICR1H=0x00;
88. ICR1L=0x00;
89. OCR1AH=0x00;
90. OCR1AL=0x00;
91. OCR1BH=0x00;
92. OCR1BL=0x00;
93.
94. // Timer/Counter 2 initialization
95. // Clock source: System Clock
96. // Clock value: Timer2 Stopped
97. // Mode: Normal top=0xFF
98. // OC2 output: Disconnected
99. ASSR=0<<AS2;
100.       TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<
    CS21) | (0<<CS20);
101.       TCNT2=0x00;
102.       OCR2=0x00;
103.
104.       // Timer(s)/Counter(s) Interrupt(s) initialization
105.       TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) |
    (0<<TOIE1) | (0<<OCIE0) | (0<<TOIE0);
106.
107.       // External Interrupt(s) initialization
108.       // INT0: Off
109.       // INT1: Off
110.       // INT2: Off
111.       MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
112.       MCUCSR=(0<<ISC2);
113.
114.       // USART initialization
115.       // USART disabled
116.       UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<U
    CSZ2) | (0<<RXB8) | (0<<TXB8);
117.
118.       // Analog Comparator initialization
119.       // Analog Comparator: Off
120.       ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) |
    (0<<ACIS1) | (0<<ACIS0);
121.       SFIOR=(0<<ACME);
122.
123.       // ADC initialization
124.       // ADC disabled
125.       ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<AD
    PS2) | (0<<ADPS1) | (0<<ADPS0);
```

```
126.
127.       // SPI initialization
128.       // SPI disabled
129.       SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA)
    | (0<<SPR1) | (0<<SPR0);
130.
131.       // TWI initialization
132.       // TWI disabled
133.       TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);
134.
135.       while (1){
136.           if (boton==0)
137.               botona=0;
138.           else
139.               botona=1;
140.
141.           if ((botona==0)&&(botonp==1)) //hubo cambio de flanco de 1 a 0
142.               var++; //Se incrementa la variable
143.
144.           if (var==10)
145.               var=0;
146.
147.           PORTB=tabla7segmentos [var];
148.           botonp=botona;
149.
150.           }
151.       }
```

Simulación



Conclusiones

En esta práctica se realizó la detección de flancos con un programa desarrollado en c. Con una variable detectamos cuando se presiona el botón para capturar primero el flanco de bajada con un cero. Ya que necesitamos que cada que se presione un botón cambie el número guardado en la lista de codificación (hecha en la práctica pasada) de cada número en hexadecimal, entonces cada que se pulse el botón y se detecte el flanco de bajada (0 lógico) entonces cambiará el número representado

en el display de 7 segmentos. Posteriormente se requerirá detectar el flanco de subida, para que cada que se suelte el botón (1 lógico), también cambie el número representado en el display de 7 segmentos.

Bibliografía

[1] tecnoplc, «FLANCO POSITIVO Y NEGATIVO EN S7-200.,» 14 Mayo 2015. [En línea]. Available: http://www.tecnoplc.com/flanco-positivo-y-negativo-en-s7-200/. [Último acceso: 5 Noviembre 2020].

[2] E. Gómez, «Flanco de subida y bajada ¿Qué son?,» rinconingenieriI, 31 Enero 2017. [En línea]. Available: https://www.rinconingenieril.es/flanco-subida-bajada/. [Último acceso: 5 Noviembre 2020].