INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

Introducción a los microcontroladores
3CM16

Practica 06

Contador de flancos sin rebote

Equipo 7

Integrantes:

- Bocanegra Heziquio Yestlanezi
- Dominguez Durán Alan Axel
- Hernandez Mendez Oliver Manuel
- Martinez Cruz José Antonio

Profesor: Aguilar Sanchez Fernando

Índice

Objetivo

Al término de la sesión, los integrantes del equipo contaran con la habilidad de realizar un contador sin rebotes de 0 a 9 mostrado en un display activado con un Push Button.

Introducción

Contador activado por flancos

Un contador activado por flancos es un tipo de contador que cuenta los flancos (cambios) de nivel de una señal de entrada en lugar de contar los pulsos de la señal. Estos contadores se utilizan a menudo en aplicaciones en las que se requiere contar eventos que pueden no ser constantes, como los pulsos generados por un sensor o un interruptor [1].

Un contador activado por flancos se puede implementar de varias maneras. Una forma común es utilizar una compuerta lógica XOR para detectar los flancos de la señal de entrada. Cuando hay un flanco de subida en la señal de entrada, la salida de la compuerta XOR cambia de estado y activa el contador. El contador luego incrementa su valor en uno y espera el próximo flanco de subida para repetir el proceso [2].

Otra forma común de implementar un contador activado por flancos es utilizando un flip-flop (biestable) D. El flip-flop D tiene una entrada de reloj y una entrada de datos. Cuando la entrada de reloj cambia de estado, el flip-flop captura el valor de la entrada de datos y lo mantiene en su salida hasta el siguiente cambio de estado del reloj. Para contar flancos, se conecta la entrada de reloj del flip-flop a la señal de entrada y la entrada de datos a la salida del flip-flop más significativo del contador. Cuando hay un flanco de subida en la señal de entrada, el flip-flop D captura el valor del contador y lo mantiene hasta el siguiente flanco de subida [3].

Material y equipo empleado

- ♥ CodeVision AVR
- ♥ AVR Studio 4
- ♥ Microcontrolador ATmega 8535
- ♥ 1 Display ánodo o cátodo común
- ♥ 7 Resistores de 330 Ω a 1⁄4 W
- ♥ 1 Push Button

Desarrollo experimental

1. Al término de la sesión, los integrantes del equipo contaran con la habilidad de realizar un contador sin rebotes de 0 a 9 mostrado en un display activado con un Push Button.

Circuito

Con los conocimientos obtenidos en las diferentes materias de electrónica, procedemos a realizar el montado del circuito en la protoboard como se muestra en la imagen 1.

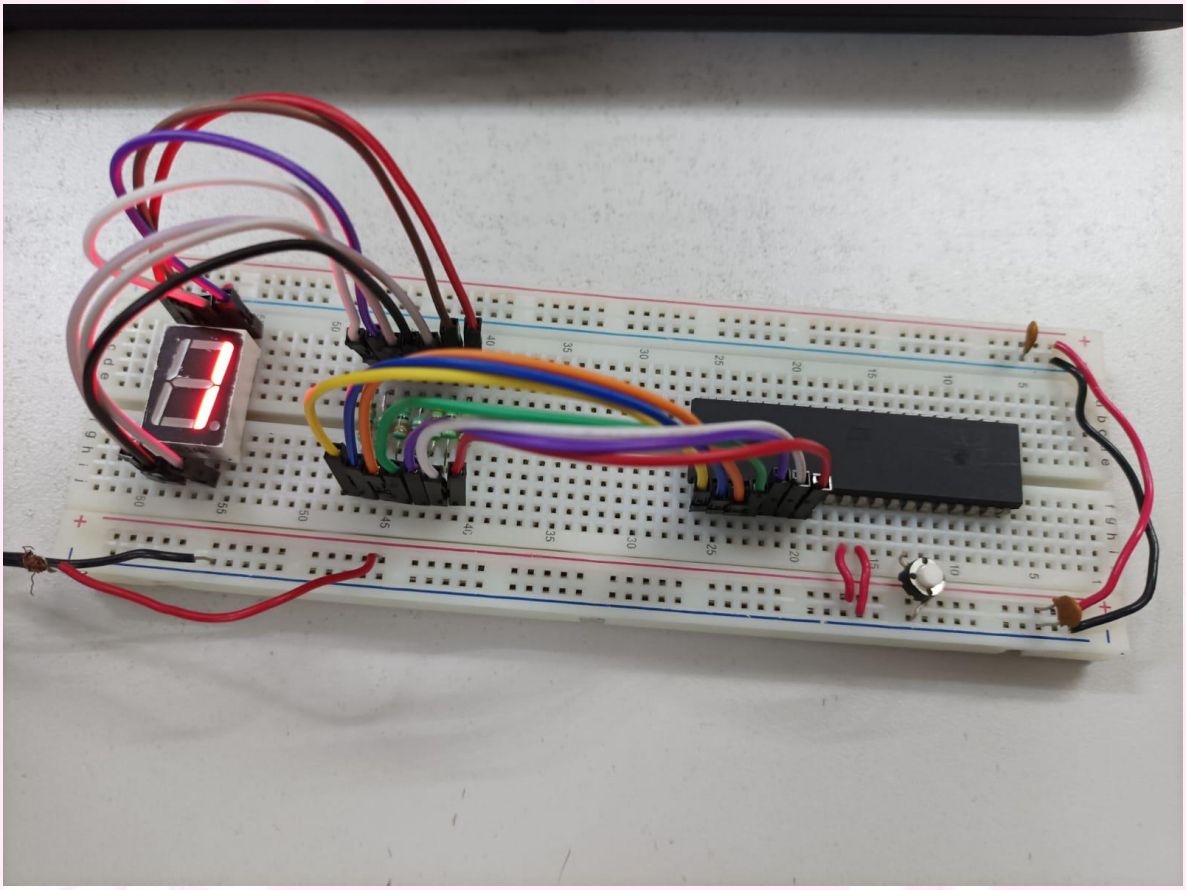

Imagen 1 contador de 0 a 9 armado en protoboard

Estructura del programa

Código CodeVisionAVR

```
/*****************************************************
This program was created by the CodeWizardAVR V3.51
Automatic Program Generator
© Copyright 1998-2023 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro

Project :
Version :
Date    : 29/03/2023
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
*****************************************************/

// I/O Registers definitions
#include <mega8535.h>
#include <delay.h>
#define boton PIND.0
bit botonp;
bit botona;
unsigned char var;
const char tabla7segmentos
[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7c,0x07,0x7f,0x6f};

// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
```

```
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out
Bit0=Out
DDRA=(1<<DDA7) | (1<<DDA6) | (1<<DDA5) | (1<<DDA4) | (1<<DDA3) | (1<<DDA2) |
(1<<DDA1) | (1<<DDA0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) |
(0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);

// Port B initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out
Bit0=Out
DDRB=(1<<DDB7) | (1<<DDB6) | (1<<DDB5) | (1<<DDB4) | (1<<DDB3) | (1<<DDB2) |
(1<<DDB1) | (1<<DDB0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) |
(0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);

// Port C initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) |
(0<<DDC1) | (0<<DDC0);
// State: Bit7=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P
PORTC=(1<<PORTC7) | (1<<PORTC6) | (1<<PORTC5) | (1<<PORTC4) | (1<<PORTC3) |
(1<<PORTC2) | (1<<PORTC1) | (1<<PORTC0);

// Port D initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) |
(0<<DDD1) | (0<<DDD0);
// State: Bit7=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P
PORTD=(1<<PORTD7) | (1<<PORTD6) | (1<<PORTD5) | (1<<PORTD4) | (1<<PORTD3) |
(1<<PORTD2) | (1<<PORTD1) | (1<<PORTD0);

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) |
(0<<CS01) | (0<<CS00);
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
```

```
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Disconnected
// OC1B output: Disconnected
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) |
(0<<WGM10);
TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) |
(0<<CS11) | (0<<CS10);
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0<<AS2;
TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) |
(0<<CS21) | (0<<CS20);
TCNT2=0x00;
OCR2=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) |
(0<<TOIE1) | (0<<OCIE0) | (0<<TOIE0);

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
MCUCSR=(0<<ISC2);
```

```
// USART initialization
// USART disabled
UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) |
(0<<UCSZ2) | (0<<RXB8) | (0<<TXB8);

// Analog Comparator initialization
// Analog Comparator: Off
// The Analog Comparator's positive input is
// connected to the AIN0 pin
// The Analog Comparator's negative input is
// connected to the AIN1 pin
ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) |
(0<<ACIS1) | (0<<ACIS0);
SFIOR=(0<<ACME);

// ADC initialization
// ADC disabled
ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) |
(0<<ADPS2) | (0<<ADPS1) | (0<<ADPS0);

// SPI initialization
// SPI disabled
SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) |
(0<<SPR1) | (0<<SPR0);

// TWI initialization
// TWI disabled
TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);

while (1)
     {
        if (boton==0)
            botona=0;
    else
            botona=1;
                if ((botonp==1)&&(botona==0)) //hubo cambio de flanco de 1 a
0
        {
            var++; //Se incrementa la variable
                if (var==10)
                    var=0;
                    delay_ms(40); //Se coloca retardo de 40mS para eliminar
rebotes
        }
```

```
                if ((botonp==0)&&(botona==1)) //hubo cambio de flanco de 0 a
1
                    delay_ms(40); //Se coloca retardo de 40mS para eliminar
rebotes
                PORTB=tabla7segmentos [var];
                botonp=botona;
};
}
```

Simulación – Proteus

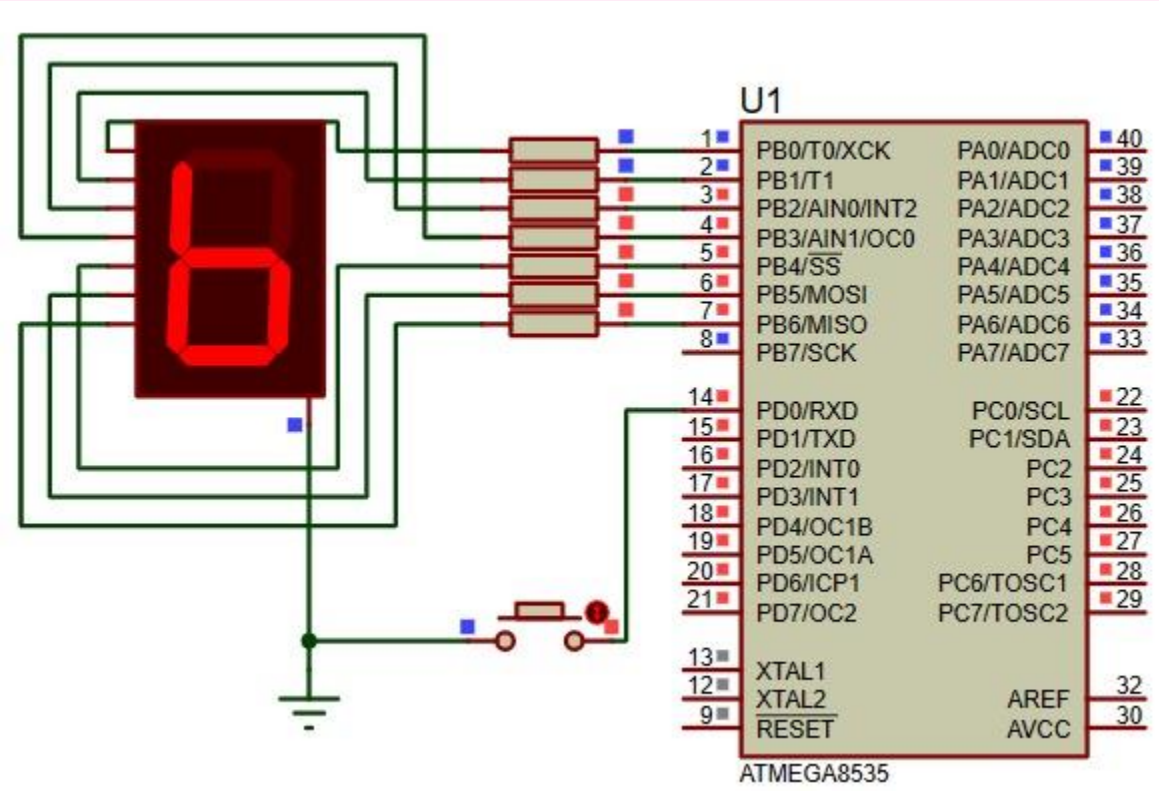

Imagen 1 Circuito simulado en proteus

Conclusiones

Bocanegra Heziquio Yestlanezi

un contador de flancos sin rebote es un circuito digital que utiliza la transición de un pulso de señal sin rebote para actualizar su estado y contar de forma ascendente o descendente. La eliminación del efecto de rebote es fundamental para asegurar una operación fiable del contador. El circuito elimina el efecto de rebote de los contactos mecánicos de un interruptor o pulsador y utiliza señales limpias para actualizar el contador.

Domínguez Durán Alan Axel

En esta práctica también implementamos un contador que aumenta por pulsos de botón, en este caso se corrigen los rebotes del ruido eléctrico ocasionados por la propia construcción del botón, de esta forma, el contador funciona de una manera más estable y no ocasiona confusiones al usuario.

Hernández Méndez Oliver Manuel

Al llegar a este punto el equipo ya tiene un mejor conocimiento acerca del dispositivo microcontrolador, el desarrollo de este contador fue bastante ágil y gracias a la práctica anterior por cada pulso de botón la presencia de ruido era casi nula, sin duda alguna la mejora en cuanto a la experiencia de uso de los proyectos aumenta considerablemente al contar con diversas herramientas.

Martínez Cruz José Antonio

Recordando lo realizado en la práctica anterior, logramos encontrar la diferencia al momento de mostrar el numero en el display entre los flancos de subida y en esta caso sin rebote. A pesar de poseer las casi las mismas características, los pequeños detalles en la codificación logran grandes cambios como es la correcta muestra del número en cuestión.

Referencias

[1] "Design of High-Speed Up/Down Counters Based on a Novel T-Flip-Flop for Next-Generation Digital Systems", en IEEE Transactions on Very Large Scale Integration (VLSI) Systems, https://ieeexplore.ieee.org/document/7963315.

[2] "Design of Low Power, High Speed Digital Counters using Edge Triggered Flip-Flops", en IEEE Xplore Digital Library, https://ieeexplore.ieee.org/document/7152767.

[3] Mano, M. M., & Kime, C. R. (2017). Logic and Computer Design Fundamentals (5th ed.). Pearson.