INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Introducción a los microcontroladores

3CM16

Practica 04

Contador de 0 a 9

Equipo 7

Integrantes:

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

Profesor: Aguilar Sanchez Fernando

Índice

Índice de circuitos

Índice de imagen

Objetivo

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

Introducción

Contador de 0 a 9

Es un circuito digital que cuenta en base 10, es decir, tiene 10 estados diferentes, del 0 al 9. Cada estado se puede representar con cuatro bits binarios, lo que permite contar de 0000 a 1001 en binario, equivalente a 0 hasta 9 en decimal [1].

Los contadores de 0 a 9 se utilizan en una amplia variedad de aplicaciones, como en la generación de direcciones de memoria, en la medición de tiempo, en la generación de señales de reloj, en sistemas de temporización y en muchos otros dispositivos electrónicos [2].

Existen diferentes tipos de contadores de 0 a 9, pero el más común es el contador sincrónico ascendente. Este tipo de contador tiene un reloj de entrada que activa el cambio de estado en cada ciclo de reloj. Los estados se generan mediante la conexión de flip-flops (biestables) en cascada, de manera que la salida del flip-flop menos significativo (LSB) se conecta a la entrada del flip-flop siguiente (segundo LSB) y así sucesivamente [3].

Además, se utilizan puertas lógicas para generar las señales de control que permiten la carga de los datos iniciales, el reseteo del contador a un valor inicial y la habilitación o deshabilitación del contador [4].

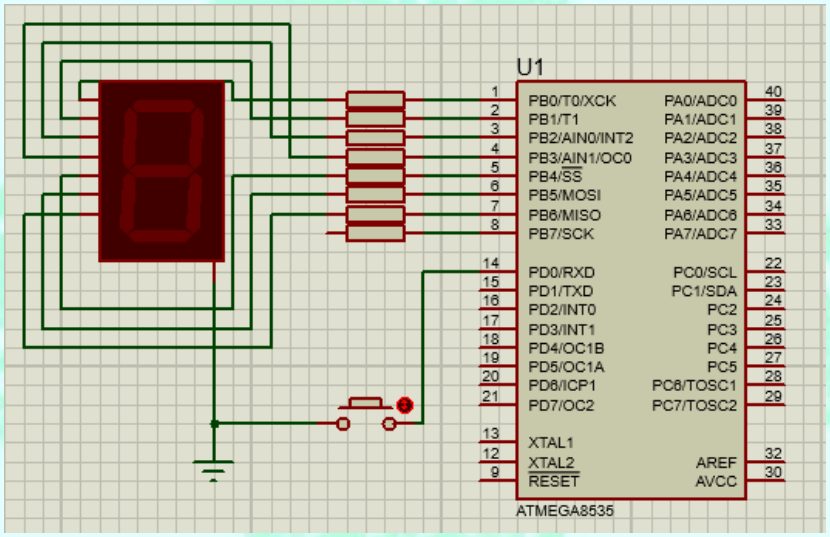
También es posible implementar un contador de 0 a 9 utilizando un decodificador BCD (Binary Coded Decimal) y un display de 7 segmentos. En este caso, el contador cuenta en binario y el decodificador convierte el número binario en la representación decimal para ser visualizado en el display de 7 segmentos [4].

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

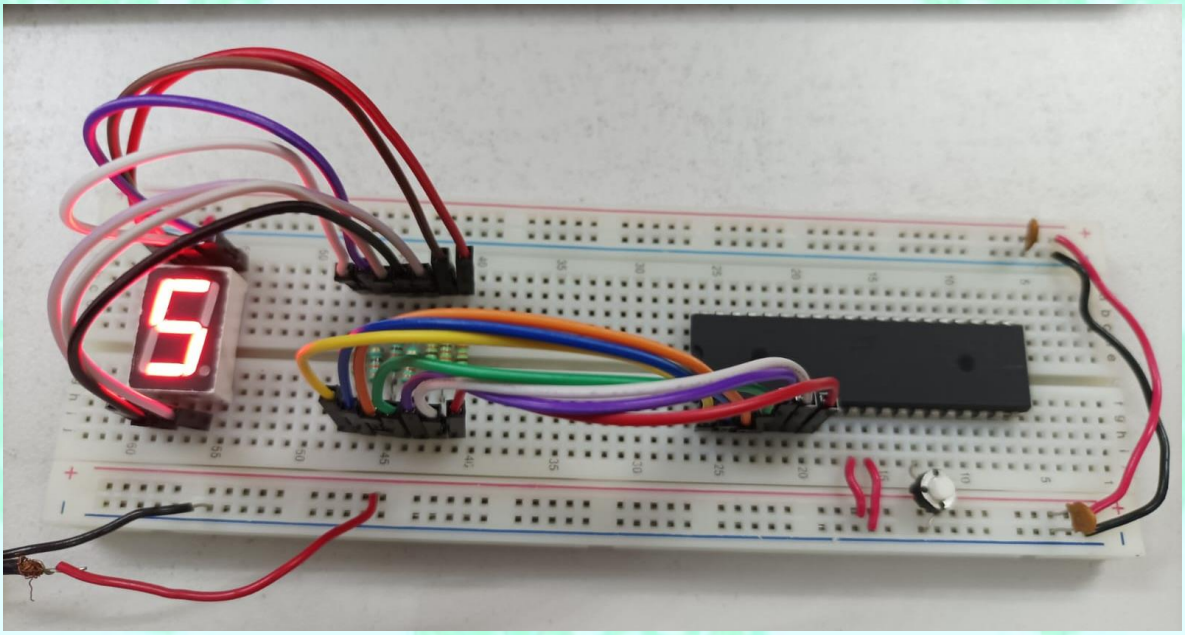
1. Diseñe un programa colocando en el Puerto B un Display. Coloque un Push Button en la terminal 0 del Puerto D para incrementar su cuenta del 0 al 9.



Circuito 1 0 a 9

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.



Circuito 2 contador de 0 a 9 armado en protoboard

Estructura del programa

Código CodeVisionAVR

```
/*****************************************************
This program was created by the CodeWizardAVR V3.45
Automatic Program Generator
© Copyright 1998-2021 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro

Project :
Version :
Date    : 15/03/2023
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
*****************************************************/

#include <mega8535.h>
#define boton PIND.0

// Declare your global variables here
const char mem[10] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F,
0x6F};
unsigned char var1;

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

// Input/Output Ports initialization
// Port A initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) |
(0<<DDA1) | (0<<DDA0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
```

```
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=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) |
(0<<PORTC2) | (0<<PORTC1) | (0<<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)
            var1++;

        if (var1 == 10)
            var1 = 0;

        PORTB = mem[var1];
      }
}
```

Simulación – Proteus

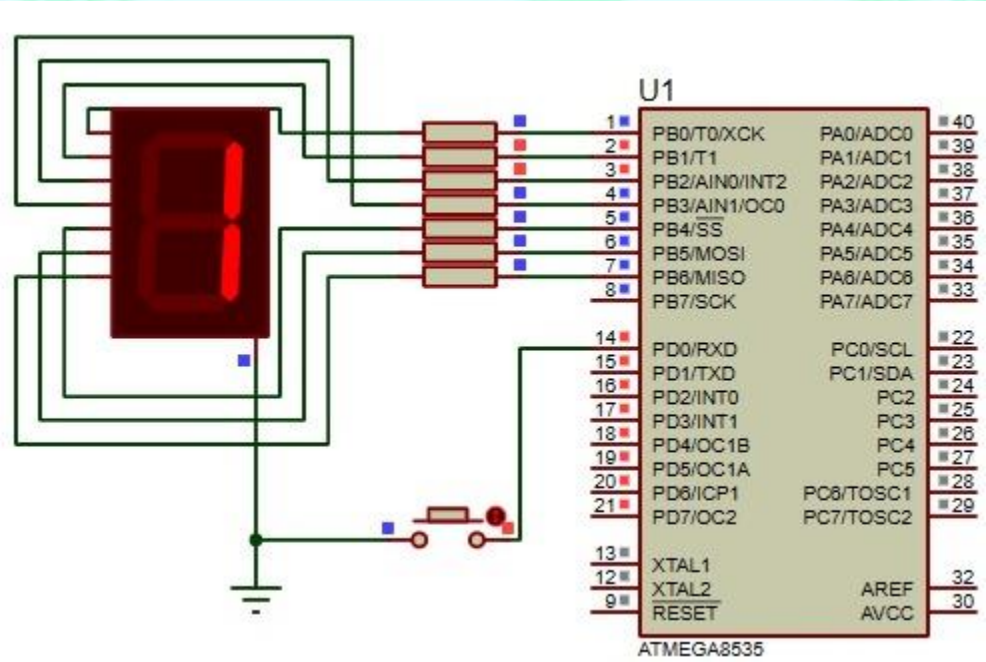


Imagen 1 Circuito simulado en proteus

Observaciones y conclusiones individuales

Bocanegra Heziquio Yestlanezi

De la siguiente practica puedo concluir que un contador de 0 a 9 es un circuito digital que cuenta de cero a nueve y luego reinicia su conteo. En su forma más simple, el circuito cuenta del 0 al 9 utilizando una serie de flip-flops y compuertas lógicas para implementar la lógica de conteo. Una vez que el contador alcanza el valor 9, se reinicia automáticamente a cero y el proceso de conteo comienza de nuevo.

Dominguez Durán Alan Axel

En esta práctica implementamos nuevamente un contador, en este caso en lugar de utilizar el dip switch, usamos los pulsos de un push button para aumentar el contador, simplificando de esta forma la construcción y armado del circuito a través de software. De esta forma podemos probar que integrar el software con el hardware puede ayudar a que las implementaciones de los circuitos sean más eficaces y sencillas; pero que conserven su funcionalidad.

Hernandez Mendez Oliver Manuel

Esto nos hizo darnos cuenta de la flexibilidad que te brinda el uso de un microcontrolador si se tiene la capacidad de asignar de forma correcta los puertos que se usarán, en este caso en vez de hacer uso de un dip switch implementamos la funcionalidad mediante push buttons, al principio consideramos que la práctica no funcionaba de forma correcta sin embargo los saltos que se visualizaban eran debido a factores externos que se mitigaron en la práctica siguiente.

Martinez Cruz José Antonio

A comparar de la practica anterior esta logro realizarse de manera exacta y ágil, mostrando los resultados esperados en la simulación elaborada en Proteus. El único detalle, así como en todas las practicas que conlleva el uso de un display, es reconocer si es cátodo o ánodo.

Referencias

[1] "Design and Simulation of a Synchronous BCD Up Counter", en IEEE Xplore Digital Library, https://ieeexplore.ieee.org/document/8506064.

[2] "Design and Implementation of BCD Counter with Digital Display using VHDL", en IEEE Xplore Digital Library, https://ieeexplore.ieee.org/document/7601309.

[3] Floyd, T. L. (2016). Digital Fundamentals (11th ed.). Pearson.

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