INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO.

INTRODUCCIÓN A LOS MICROCONTROLADORES.

REPORTE PRÁCTICA 4.

PROFESOR: AGUILAR SANCHEZ FERNANDO

Grupo: 3CM6

Alumno: Rojas Alvarado Luis Enrique

Boleta: 2014010995

Práctica 4. Contador de 0 a 9

Introducción teórica.

Un contador es un circuito secuencial construido a partir de biestable y puertas lógicas capaz de almacenar y contar los impulsos (a menudo relacionados con una señal de reloj), que recibe en la entrada destinada a tal efecto, así mismo también actúa como divisor de frecuencia.

Los pulsos de entrada pueden ser pulsos de reloj u originarse en una fuente externa y pueden ocurrir a intervalos de tiempo fijos o aleatorios.

El número de salidas limita el máximo número que se puede contar. [1]

Contadores sincrónicos: Todos los flip-flops cambian simultáneamente con cada pulso del reloj (de acuerdo con el estado de sus entradas de control).

Contadores asincrónicos: Todos los flip-flops no cambian simultáneamente con cada pulso del reloj.

Contadores asincrónicos cuya base no es potencia de dos: Este tipo de contador puede ser construido realimentando convenientemente las salidas a algunas de las entradas, incluyendo las entradas directas, para eliminar estados de un contador 2n superior. [2]

Según el formato de salida del conteo: Binario. BCD (Decimal Codificado en Binario). Arbitrario.

Según sentido de conteo: Conteo ascendente ó progresivo. Conteo descendente ó regresivo. [3]

El número máximo de estados por los que pasa un contador se denomina módulo del contador (Número MOD). Este número viene determinado por la expresión 2^n donde n indica el número de bits del contador. Ejemplo, un contador de módulo 4 pasa por 4 estados, y contaría del 0 al 3. Si necesitamos un contador con un módulo distinto de 2^n, lo que haremos es añadir un circuito combinacional.

Este contador decodificador (BCD) está diseñado para un display de 7 segmentos con cátodo común. Se alimenta con 5 voltios. Puedes agregar otras entradas, como todo el puerto de entrada del microcontrolador que se vaya a ocupar, así como que si se quiere visualizar números de 2 dígitos, se puede usar la misma configuración y combinación de entradas o agregar más entradas, así como una salida para visualizar el segundo dígito, puede ser un display de ánodo o cátodo. Como generador de pulsos se incluye el que utiliza el ATMEGA8535, para variar los pulsos deberás cambiar la frecuencia del oscilador interno del microcontrolador, ya que el programa en lenguaje de programación C, se ejecuta a una frecuencia de 1MHz que se traduce a 1 milisegundo en toda la ejecución del programa principal.

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.

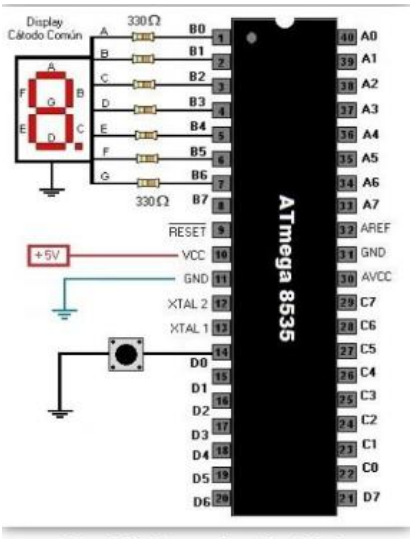


Figura 1. Circuito para el contador de 0 a 9.

Código

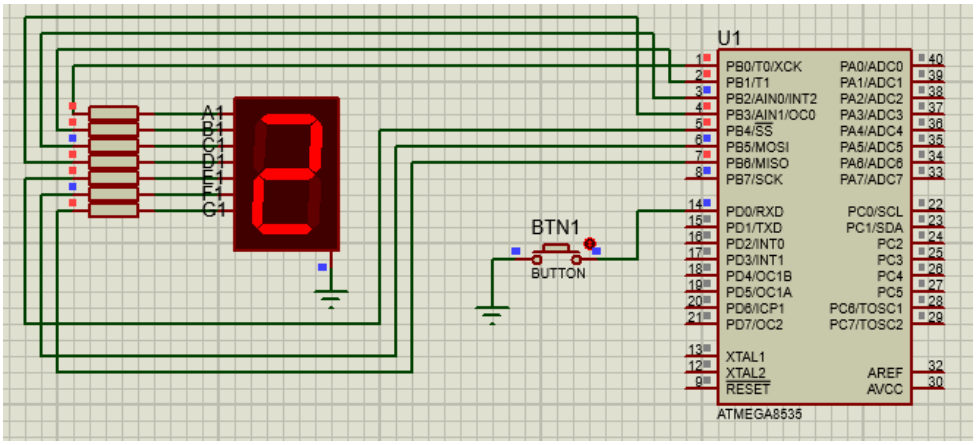
```
/*****************************************************
This program was created by the
CodeWizardAVR V2.60 Evaluation
Automatic Program Generator
© Copyright 1998-2012 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 27/10/2020
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
*****************************************************/

#include <mega8535.h>
#define boton PIND.0
const char tabla7segmentos [10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7c,0x07,0x7f,0x6f
};
unsigned char var1;


void main(void)
{
```

```
32. // Declare your local variables here
33.
34. // Input/Output Ports initialization
35. // Port A initialization
36. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
37. DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) | (0<<D
    DA1) | (0<<DDA0);
38. // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
39. PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (0<<PO
    RTA2) | (0<<PORTA1) | (0<<PORTA0);
40.
41. // Port B initialization
42. // Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=O
    ut
43. DDRB=(1<<DDB7) | (1<<DDB6) | (1<<DDB5) | (1<<DDB4) | (1<<DDB3) | (1<<DDB2) | (1<<D
    DB1) | (1<<DDB0);
44. // State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
45. PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PO
    RTB2) | (0<<PORTB1) | (0<<PORTB0);
46.
47. // Port C initialization
48. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
49. DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<D
    DC1) | (0<<DDC0);
50. // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
51. PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PO
    RTC2) | (0<<PORTC1) | (0<<PORTC0);
52.
53. // Port D initialization
54. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
55. DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) | (0<<D
    DD1) | (0<<DDD0);
56. // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=P
57. PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (0<<PO
    RTD2) | (0<<PORTD1) | (1<<PORTD0);
58.
59. // Timer/Counter 0 initialization
60. // Clock source: System Clock
61. // Clock value: Timer 0 Stopped
62. // Mode: Normal top=0xFF
63. // OC0 output: Disconnected
64. TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) |
    (0<<CS00);
65. TCNT0=0x00;
66. OCR0=0x00;
67.
68. // Timer/Counter 1 initialization
69. // Clock source: System Clock
70. // Clock value: Timer1 Stopped
71. // Mode: Normal top=0xFFFF
72. // OC1A output: Disconnected
73. // OC1B output: Disconnected
74. // Noise Canceler: Off
75. // Input Capture on Falling Edge
76. // Timer1 Overflow Interrupt: Off
77. // Input Capture Interrupt: Off
78. // Compare A Match Interrupt: Off
79. // Compare B Match Interrupt: Off
80. TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WG
    M10);
```

```
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<<U
CSZ2) | (0<<RXB8) | (0<<TXB8);

      // Analog Comparator initialization
      // Analog Comparator: Off
      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<<AD
PS2) | (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++;
```

```
135.                if (var1==10)
136.                var1=0;
137.                PORTB=tabla7segmentos [var1];
138.
139.            }
140.        }
```

Simulación de circuito



Conclusiones

En esta práctica pudimos observar el comportamiento de un contador de 0-9 BCD ya que la representación al final la pudimos ver representada en un display 7 segmentos de cátodo común. Las instrucciones dentro del programa principal se ejecutan prácticamente en 1 milisegundo (esto puede variar de acuerdo con la configuración del oscilador interno del ATMEGA8535). Pero de igual manera, la acción más rápida que podamos hacer es inmensamente lenta comparada con esa velocidad, así es que por esa razón los números que se representan en el display pasan muy rápido y no es posible percibirlos todos con precisión. Solo se tuvo que conectar el puerto B como salida (Tal como la práctica anterior) y el pin 0 del puerto D como entrada, activando el pull-up porque recibirá un 0 lógico como entrada.

Bibliografía

[1] Rubén González, Márco Brandón, José Antonio Salcines, “Contadores”, Recuperado de:

https://personales.unican.es/manzanom/Planantiguo/EDigitalI/CONTG5.pdf,. Recuperado el 27/10/2020

[2] Microelectronics, Jacob Millman, 1979.

[3] Sergio Noriega, “Contadores”, Introducción a los sistemas lógicos y digitales, 2009, Recuperado de:

https://catedra.ing.unlp.edu.ar/electrotecnia/islyd/apuntes/Tema%205%20Contadores%202010.pdf,. Recuperado el 27/10/2020