INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Introducción a los microcontroladores

3CM16

Practica 03

Convertidor BCD a 7 segmentos

Contador Hexadecimal

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 tablas

Índice de imágenes

Objetivo

Al término de la sesión, los integrantes del equipo contaran con la habilidad de realizar un contador BCD empleando arreglos.

Introducción

Convertidor BCD a 7 segmentos

El código BCD es un sistema numérico binario que utiliza 4 bits para representar cada dígito decimal del 0 al 9. Por lo tanto, el convertidor BCD a 7 segmentos convierte cada dígito BCD en una combinación de segmentos de 7 LEDs que forman el número correspondiente en el display [1].

El circuito básico del convertidor BCD a 7 segmentos consta de 4 entradas para los dígitos BCD y 7 salidas que corresponden a los segmentos del display. Cada entrada BCD se conecta a un decodificador BCD a 10 líneas que activa las salidas correspondientes a los segmentos necesarios para visualizar el dígito en el display [2].

Existen diferentes tipos de displays de 7 segmentos, pero en general, los 7 segmentos se denominan "a", "b", "c", "d", "e", "f" y "g". Dependiendo del tipo de display, puede haber dos o más segmentos adicionales para representar puntos decimales, signos u otros caracteres [3].

Contador hexadecimal

Un contador hexadecimal es un circuito digital que cuenta en base 16, es decir, tiene 16 estados diferentes. Cada estado se puede representar con cuatro bits binarios, lo que permite contar desde 0 hasta F en hexadecimal, equivalente a 0 hasta 15 en decimal [4].

Los contadores hexadecimales se utilizan en una amplia variedad de aplicaciones, como en la generación de direcciones de memoria, en la medición de frecuencia, en la generación de señales de reloj, en sistemas de temporización y en muchos otros dispositivos electrónicos [5].

Existen diferentes tipos de contadores hexadecimales, 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 [6].

Material y equipo empleado

- CodeVision AVR
- AVR Studio 4
- Microcontrolador ATmega 8535
- 1 display ánodo común
- 1 display cátodo común
- 14 resistores de 330 ohm a un cuarto de watt

Desarrollo experimental

1. Diseñe un convertidor BCD a 7 Segmentos para un Display Cátodo común. Observe la siguiente tabla:

Número Display	Combinaciones .	g	f	e	d	c	b	a	Valor Hexadecimal
0	0	0	1	1	1	1	1	1	0x3F
1	0	0	0	0	0	1	1	0	0x06
2	0	1	0	1	1	0	1	1	0x5B
3	0	1	0	0	1	1	1	1	0x4F
4	0	1	1	0	0	1	1	0	0x66
5	0	1	1	0	1	1	0	1	0x6D
6	0	1	1	1	1	1	0	1	0x7C
7	0	0	0	0	0	1	1	1	0x07
8	0	1	1	1	1	1	1	1	0x7F
9	0	1	1	0	1	1	1	1	0x6F

Tabla 1 BCD a 7 segmentos

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.

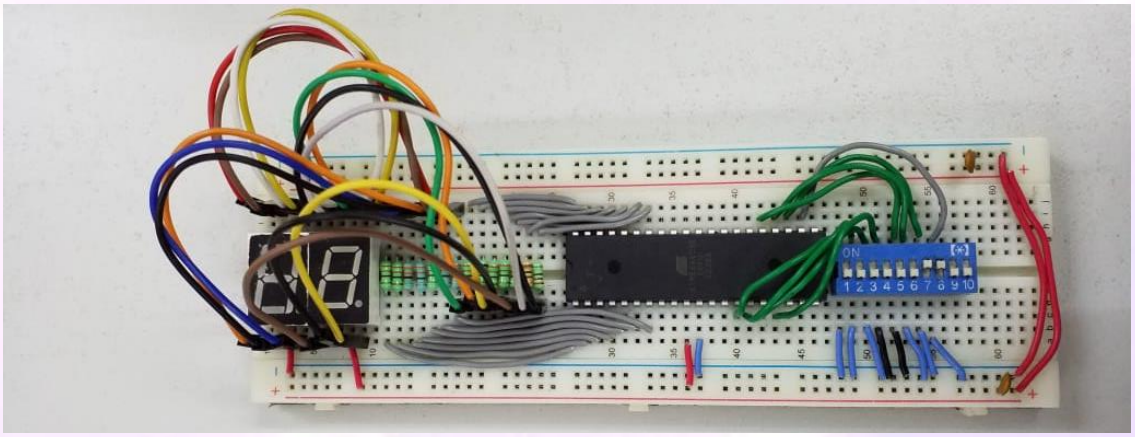

Imagen 1 Circuito armado en protoboard

A continuación, presentamos el circuito en funcionamiento en la imagen 2.

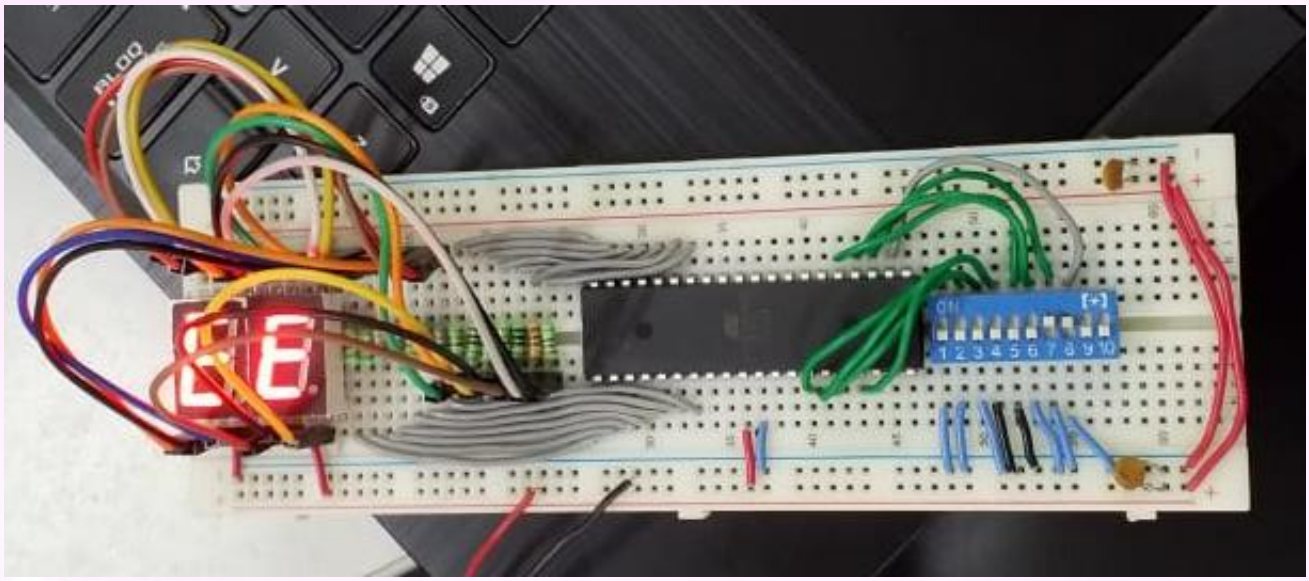

Imagen 2 Circuito en funcionamiento

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>
```

```
unsigned char num, num2;
const char mem[16] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F,
                      0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71};

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)
      {
        num = PIND & 0x0F;
        num2 = PINC & 0x0F;

        if (PIND & 0x10) {
```

```
            PORTB = mem[num];
            PORTA = ~mem[num2];
        } else {
           if (num < 10) {
            PORTB = mem[num];
           } else {
            PORTB = 0x79;
           }

           if (num2 < 10) {
            PORTA = ~mem[num2];
           } else {
            PORTA = ~0x79;
           }
        }
      }
}
```

Simulación – Proteus

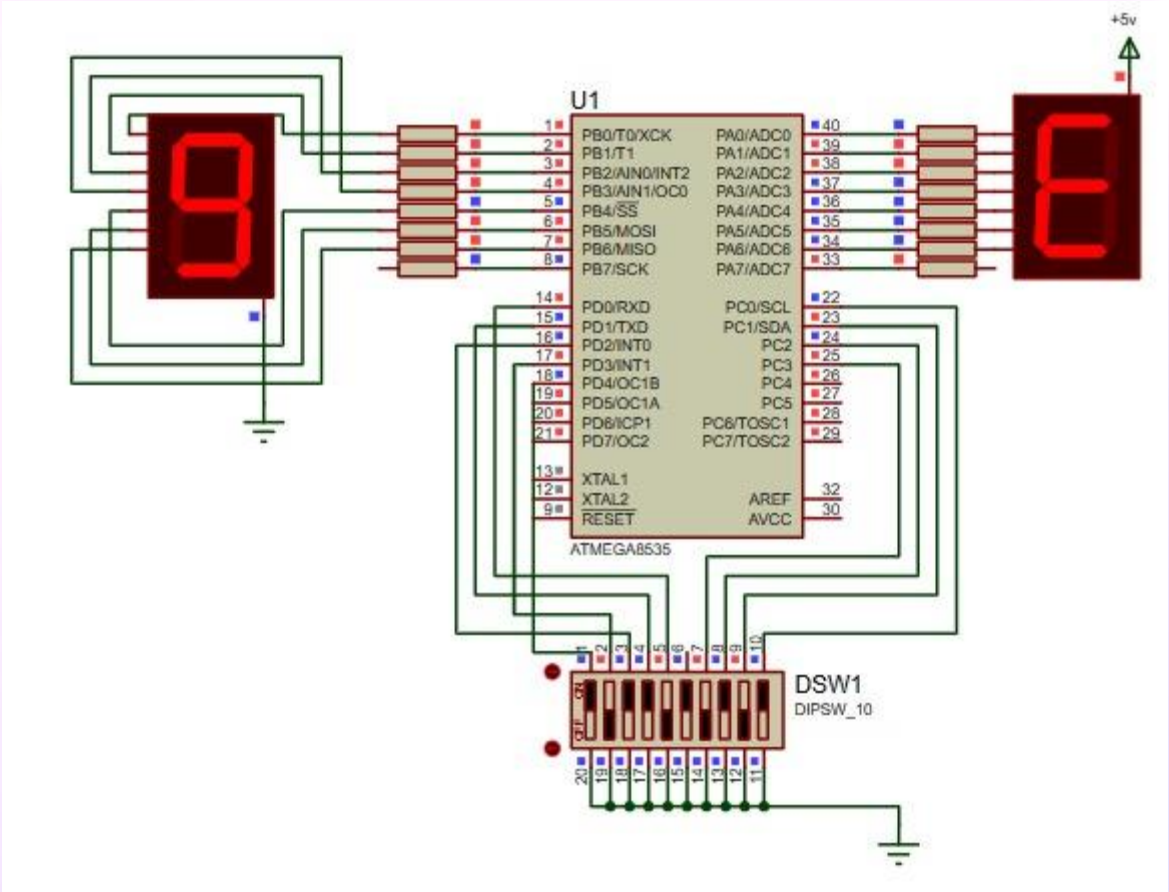

Imagen 3 Simulación en proteus del circuito

Conclusiones

Bocanegra Heziquio Yestlanezi

De la siguiente practica puedo concluir que un convertidor BCD a 7 segmentos, se trata de un circuito digital utilizado para convertir números binarios codificados en decimal (BCD) a su representación visual en un display de 7 segmentos. Por otro lado, el contador hexadecimal es un circuito digital que cuenta en base 16, es decir, utiliza 16 símbolos (0-9 y A-F) para representar los valores numéricos.

Dominguez Durán Alan Axel

Al termino de esta práctica, pudimos ver más a fondo las capacidades del microcontrolador, las cuales son bastante amplias ya que con unas cuantas líneas de código, cambiamos el contador de base numérica, además de que podemos personalizar con libertad los caracteres que se muestran en el display; todo depende del objetivo y caso de uso.

Hernandez Mendez Oliver Manuel

En particular esta práctica represento un reto principalmente por la conversión de las unidades, el tener ambas funciones en un solo circuito fue un tanto complicado, pero al final pudimos resolverlos y nuevamente tuvimos la oportunidad de sacar provecho del microcontrolador para hacer una tarea relativamente compleja de forma simplificada gracias a la implementación de código en C

Martinez Cruz José Antonio

Existieron muchos inconvenientes en esta práctica, resultando en un retraso significativo en comparación con las practicas anteriores. A pesar de eso, logramos sobrellevarlos y lograr los objetivos establecidos en esta práctica, tales como la implementación de los sistemas numéricos decimal y hexadecimal, así como el uso de la E como indicación de desborde de datos.

Referencias

[1] BCD to Seven-Segment Decoder, en IEEE Xplore Digital Library, https://ieeexplore.ieee.org/document/8659859.

[2] Al-Dabagh, M., & Qasim, A. (2018). Design and Implementation of BCD to Seven Segment Decoder. International Journal of Advanced Computer Science and Applications, 9(6), 259-263. doi: 10.14569/ijacsa.2018.090635

[3] Sedra, A. S., & Smith, K. C. (2016). Microelectronic Circuits. Oxford University Press.

[4] "Design and Implementation of Hexadecimal Counter using VHDL", en IEEE Xplore Digital Library, https://ieeexplore.ieee.org/document/8479504.

[5] Tocci, R. J., Widmer, N. S., & Moss, G. L. (2011). Digital Systems: Principles and Applications (11th ed.). Pearson.

[6] Katz, R. H., & Borriello, G. (2010). Contemporary Logic Design (2nd ed.). Pearson.