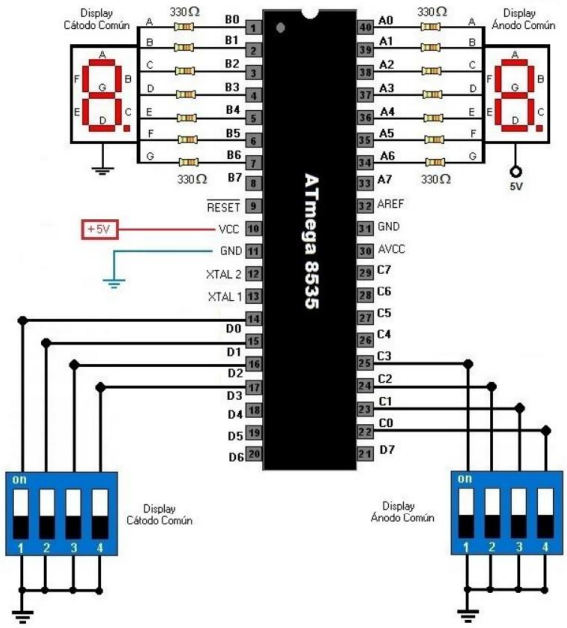
INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

SISTEMAS EN CHIP

PRACTICA No. 03

"BCD a 7 Segmentos"



ALUMNO: RENTERIA ARRIAGA JOSUE.

GRUPO: 6SCM1

PROFESOR: FERNANDO AGILAR SÁNCHEZ.

31/OCTUBRE/2022

I. OBJETIVO GENERAL:

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

II. MATERIAL Y EQUIPO EMPLEADO

- ✓ CodeVision AVR.
- ✓ AVR Studio 4.
- ✓ Microcontrolador ATmega 8535.
- ✓ 1 Display ánodo común.
- ✓ 1 Display cátodo común.
- ✓ 14 Resistores de 330 Ω a ¼ W.

III. INTRODUCCIÓN TEÓRICA

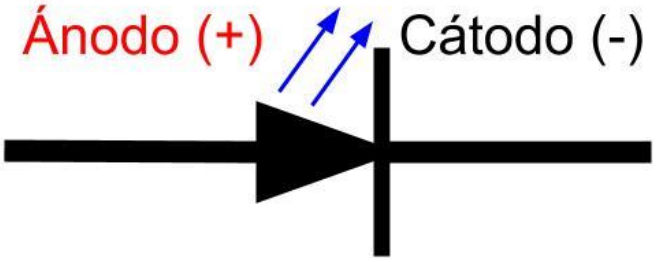
En esta práctica podremos saber utilizar correctamente y aplicar la implementación del código en el lenguaje de programación C de los conceptos de un contador implementando un display (cátodo y ánodo común), implementación de entradas con switchs y la creación de una tabla con los valores de salida en el display.

En esta práctica se estudiará la implementación de un Programa y Circuito para programar el puerto B para la salida de nuestro Display Cátodo Común, nuestro puerto A para la salida de nuestro Display Ánodo Común y los puertos D y C para la entrada de nuestros datos con su respectivo Switch. La práctica se dividirá en dos una que cuente del 0 al 9 y los valores restantes se muestre una E de error y otro que nos cuente del 0 al 9, luego muestre los valores A, B, C, D, E y F. En este caso de la práctica se ocuparán algunos conceptos obtenidos anteriormente en el curso como lo es la implementación de código en C para poder programar nuestro microcontrolador, algunos conceptos de Fundamentos de Diseño Digital y de Electrónica Analógica para poder realizar correctamente el armado de nuestro circuito y que este funcione correctamente, así como la implementación de una tabla con los valores en vinario y hexadecimal de nuestras salidas.

Ocuparemos algunos elementos como lo dice el formato de la practica ya que ocuparemos el simulador Code Vision para poder implementar el código para después programar nuestro ATMega8535, en mi caso se ocupará el programa Khazama AVR con su programador para poder programar nuestro ATMega con el archivo .hex que se generó anteriormente. Para la parte del armado del circuito será algo sencillo ya que solo se implementarán Leds para los puertos de salida.

Algunos conceptos que se necesitan para poder entender algunos conceptos de la practica son los siguientes:

Existen dos tipos principales para los display 7 segmentos. Esta diferencia depende principalmente del arreglo como están conectados los leds que forman a cada segmento. Sabemos que un led tiene dos terminales que se denominan: cátodo y ánodo. El ánodo es la parte positiva del LED, mientras que el cátodo es el pin negativo. Entonces los tipos de display de 7 segmentos se dividen en aquellos de cátodo común y los de ánodo común. Entonces el display tendrá además de los 7 segmentos, 1 pin común. Este pin común se conecta al cátodo o al ánodo dependiendo del tipo de display.



Esta práctica nos ayudara a entender cómo se pueden implementar los circuitos con un microcontrolador y todo el proceso que conlleva realizar la programación del microcontrolador.

Con esta introducción se podrá comprender y realizar el desarrollo de la práctica que se desarrolla más adelante.

IV. 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_*Tabla para la primera parte del problema (0 al 9 y E de Error).*

Numero Display	.	g	f	e	d	c	b	a	Hexadecimal
A	0	1	1	1	0	1	1	1	0x77
B	0	1	1	1	1	1	1	1	0x7F
C	0	0	1	1	1	0	0	1	0x39
D	0	0	1	1	1	1	1	1	0x3F
E	0	1	1	1	1	0	0	1	0x79
F	0	1	1	1	0	0	0	1	0x71

Tabla.2_*Tabla para la segunda parte del problema (0 al 9 y del A al F).*

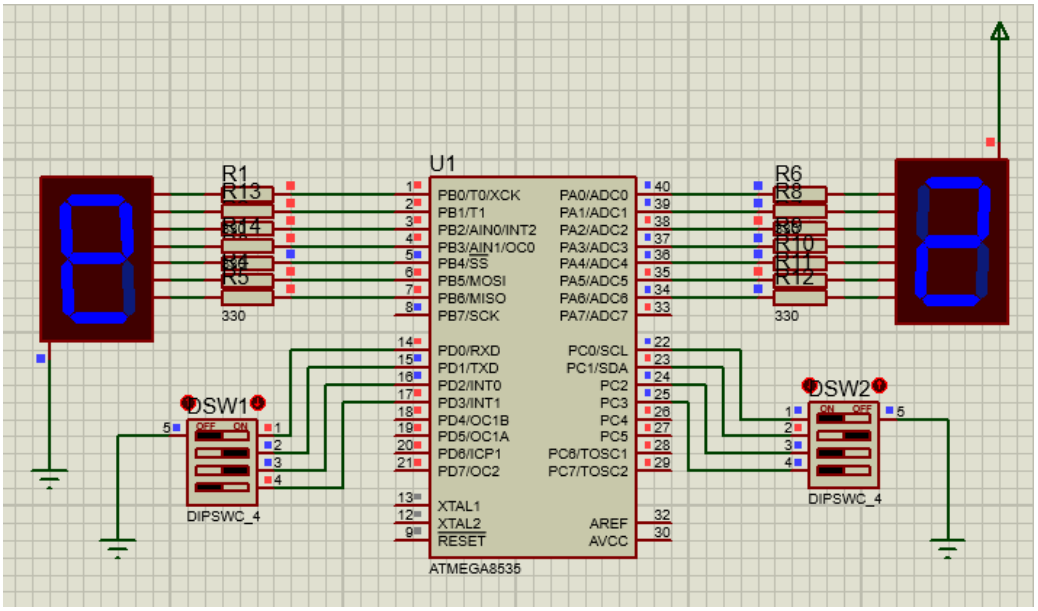


Fig.1_*Circuito Armado y Simulado en Proteus.*

CIRCUITO EN FISICO ARMADO.

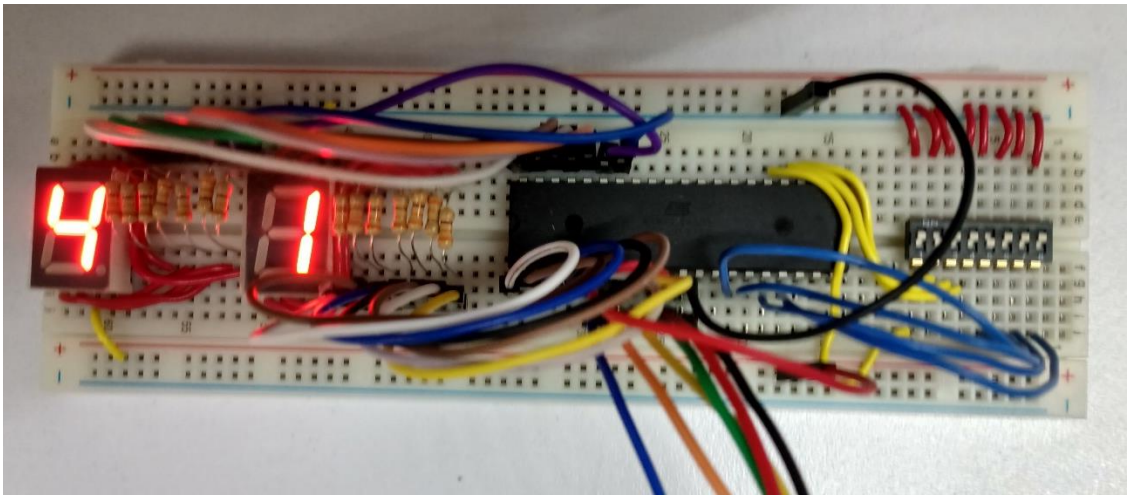


Fig.2_*Circuito Armado en Físico (0 al 9 y E de Error).*

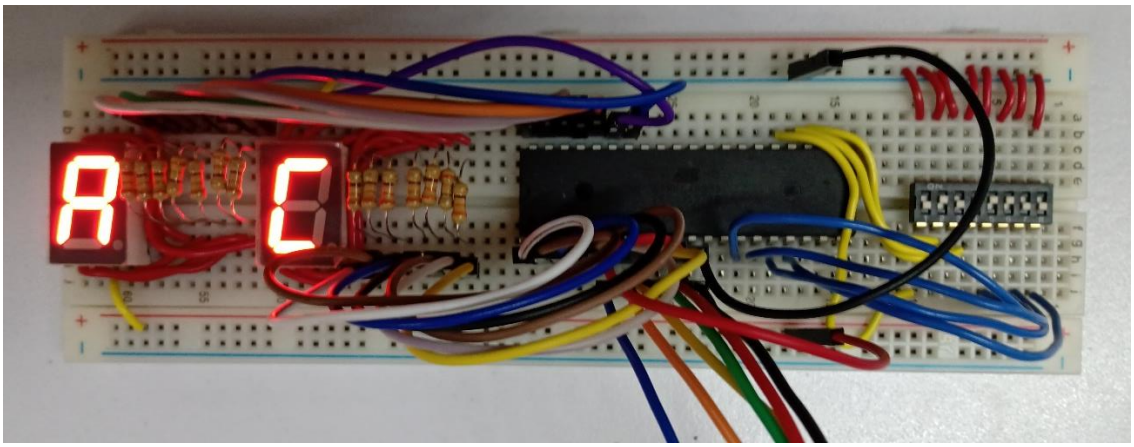


Fig.3_*Circuito Armado en Físico (0 al 9 y del A al F).*

V. CÓDIGO

```
// I/O Registers definitions
#include <mega8535.h>

unsigned char variable1;
unsigned char variable2;
const char tabla7segmentos
[16]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7c,0x07,0x7f,0x6f,0x77,0x7f,0x39,0x3f,0x79,0x71};
// 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=1 Bit6=1 Bit5=1 Bit4=1 Bit3=1 Bit2=1 Bit1=1 Bit0=1
PORTA=(1<<PORTA7) | (1<<PORTA6) | (1<<PORTA5) | (1<<PORTA4) | (1<<PORTA3) | (1<<PORTA2) |
(1<<PORTA1) | (1<<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)
      {
      variable1 = PIND&0x0f; //Enmascaramos los 4 bits menos significativos
      variable2 = PINC&0x0f; //Enmascaramos los 4 bits menos significativos
      //del puerto A ya que los dem?s no interesan.
        PORTB = tabla7segmentos[variable1];
        PORTA = ~tabla7segmentos[variable2];

      //if (variable1>=16) //Si lo que leemos es mayor o igual de 10 que dibuje en el
display una E de ERROR
        //PORTB = 0x79;

      //if (variable2>=16) //Si lo que leemos es mayor o igual de 10 que dibuje en el
display una E de ERROR
        //PORTA = ~0x79;
      }
}
```

VI. OBSERVACIONES Y CONCLUSIONES

Gracias a la realización de la práctica podemos observar cómo se utiliza e implementa un circuito con un microcontrolador (ATMega8535) y como gracias a su programador pudimos logra programar el integrado con nuestro código en lenguaje C. Esta practica no fue tan difícil, pero, es muy importante para poder familiarizarnos con el entorno de programación y como este funciona. Durante la practica si se tuvieron muchos problemas, en este caso se aprendió a realizar un contador con un display y a cómo utilizar los valores hexadecimales.

Tras la investigación y explicación de la práctica previa me quedo un poco más claro como poder implementar y programar el ATMega8535. También conocí el concepto de un display de 7 segmentos (cátodo y ánodo común) y como utilizarlas en nuestro entorno de programación. En el caso de esta práctica se divido en dos los circuitos para que fuera mas fácil.

Para concluir esta práctica podemos decir que fue una experiencia interesante ya que nunca había armado circuitos en físico, solo en simulaciones y es muy diferente. Gracias a la anterior practica ya fue más fácil de hacerlo.

VII. BIBLIOGRAFÍA

- ✓ CodeVisionAVR. (2018). CodeVisionAVR. 20/Septiembre/2021, de CodeVisionAVR Sitio web: http://www.hpinfotech.ro/cvavr-download.html

- ✓ Portal Académico. (2017). CONVERSIÓN DE BINARIO A HEXADECIMAL. 31/Octubre/2022,Sitio web: https://portalacademico.cch.unam.mx/cibernetica1/sistemas-de-numeracion/conversion-de-binario-a-hexadecimal