Instituto Politécnico Nacional

Escuela Superior de Cómputo

Departamento de Ciencias e Ingeniería de la Computación

Sistemas en Chip

Practica 03 “Convertidor BCD a 7 Segmentos”

Profesor: Fernando Aguilar Sánchez

Grupo: 6CM1

Equipo 3

Alumnos:

Ocampo Téllez Rodolfo

Ruvalcaba Flores Martha Catalina

Sandoval Hernández Eduardo

Fecha de entrega: 17/11/2022

Objetivo

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

El microcontrolador ATMEGA8535 fabricante ATMEL. En la figura 1 se muestra el microcontrolador ATMEGA8535 la cual maneja datos de 8 bits es decir su bus de datos de 8 bits. Aunque este contiene tres registros los cuales son el *x*, *y* y *z*, los cuales manejan datos de 16 bits.

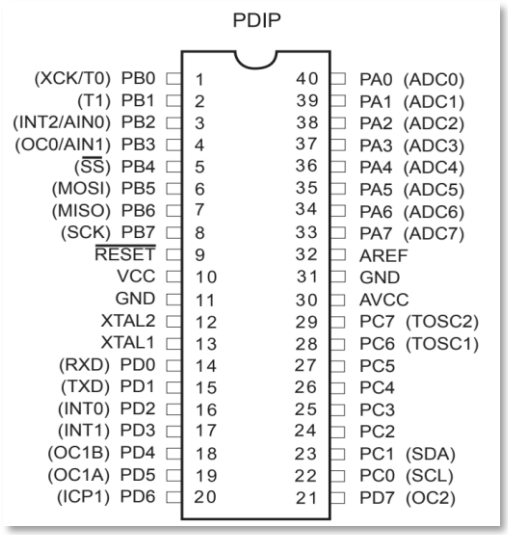

Figura 1. Configuración de pines ATmega8535

El microcontrolador utiliza una arquitectura cerrada, es decir, aquel que es inmodificable por los programadores ajenos a la compañía propietaria del código fuente. Por lo tanto, a este sistema no se le pueden colocar dispositivos periféricos, solo se usa el hardware de la compañía propietaria ya que los dispositivos ajenos a dicha compañía no son compatibles.

El microcontrolador ATMEGA8535 utiliza un encapsulado DIP-40, común en la construcción de circuitos integrados que consiste en un bloque con dos hileras paralelas de pines, observar la figura 2.

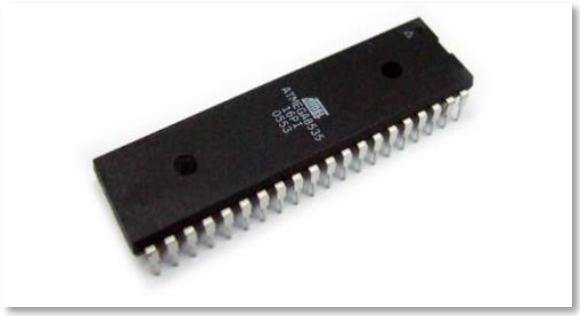

Figura 2. Microcontrolador atmega853.

La comunicación interna del microcontrolador se categoriza en 4 puertos, en la figura 1 se puede analizar la etiquetación de los puertos PA0 al PA7, PB0 al PB7, PC0 al PC7 y PD0 al PD7. Cabe recordar que maneja datos de 8 bits.

El microcontrolador se alimenta de las terminales 10 y 11 como lo muestra la figura 1, los cuales son el VCC (5 V y una tolerancia de ± 0.5V) y GND. Sin embargo, el convertidor se alimenta de forma externa en la terminal 31 y 32, los cuales son GND y AREF.

Posee un oscilador interno de 1MHz, sin embargo, como es un oscilador RC, es susceptible a variar la frecuencia con respecto a las variaciones de temperatura. Por otro lado, puede conectarse un oscilador de 0 a 8 MHz o de 0 a 16 MHz.

Display Cátodo y Ánodo Común

El display de 7 segmentos es un dispositivo opto-electrónico que permite visualizar números del 0 al 9 y o algunas letras. El dispositivo cuenta con 7 leds, uno por cada segmento, a cada uno de estos se le asigna una letra que va de la "a" a la "g" y se encuentran organizados como indica la figura 3.

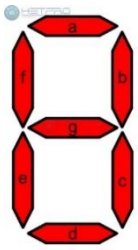

Figura 3. Disposición de los leds en un display de 7 segmentos.

Existen dos tipos de displays de 7 segmentos, de ánodo y cátodo comunes cuya diferencia se encuentra en la forma en que van conectados como lo indica la figura 4.

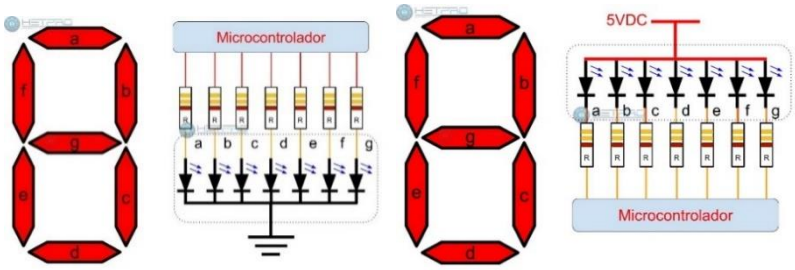

Figura 4. Conexiones en display de 7 segmentos.

Es importante mencionar que los display de 7 segmentos, dado que están construidos con diodos LED, requieren colocar una resistencia para limitar la corriente. Dicha resistencia depende de la corriente que se quiera suministrar al LED así como de la caída de voltaje. Para calcular la resistencia usamos la Ley de Ohm.

Arreglos

Un arreglo es un conjunto de datos que pueden ser seleccionados a través de un índice. A continuación, se muestra su sintaxis: "flash o const tipo_de_dato _nombre_del_arreglo [número de elementos] = {elemento1, elemento2, … , elemento n};"

Flash o const son lo mismo, ya que guardan los datos en flash, pero por la compatibilidad con el lenguaje C, se recomienda usar la palabra const.

Operadores para el manejo de Bits

En la tabla 1, se muestran los operadores y su descripción para el manejo de Bits.

Símbolo	Descripción
&	AND Bit a Bit
|	OR Bit a Bit
^	OR exclusivo Bit a Bit
<<	Corrimiento a la Izquierda
>>	Corrimiento a la Derecha
~	Complemento a unos (inversión de bits)

Tabla 1. Operadores para el manejo de Bits.

Operadores de relación

En la tabla 2, se muestran los operadores de relación y su descripción. Dichos operadores se utilizan en las instrucciones de if, while y do while. Cabe resaltar que los operadores "=", " | " y "&", se deben de colocar dos veces para que realice la evaluación y no el manejo de Bits como se muestra en la tabla 1.

Operador	Descripción
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor Igual que
==	Igual que
!=	Distinto de
&&	Y también si
||	O si

Tabla 2. Operadores de relación.

Código BCD

El código BCD (decimal codificado, en inglés Binary-Coded Decimal), es una representación de números decimales en el sistema binario, lo cual cada dígito decimal es una codificación con secuencia de 4 bits, con esta representación se puede ver la relación que existe entre un número decimal y ese número en codificación binaria. Este sistema de numeración es ponderado, lo cual quiere decir que cada posición de una secuencia de dígitos obtiene cierto valor.

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Tabla 3. Código BCD.

Esta representación llega a utilizar 4 dígitos binarios para representar 0 al 9. Para poder el representar el equivalente se pone el valor 1 donde se ocupa y 0 donde no se ocupa, con 4 dígitos, se pueden representar 16 números (0000-1111), pero en el código BCD, sólo se usan diez de ellos, las 6 combinaciones que no se emplean (1010, 1011, 1100, 1101, 1110 y 1111) no son válidas en el código BCD, tal como se observa en la tabla 3**.**

Materiales 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

Desarrollo Experimental

1.- Diseñe un convertidor BCD a 7 Segmentos para un Display Cátodo común. Observe la siguiente tabla 3, recordar que se debe elaborar la codificación para el Display Ánodo común.

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 3. Display Cátodo común.

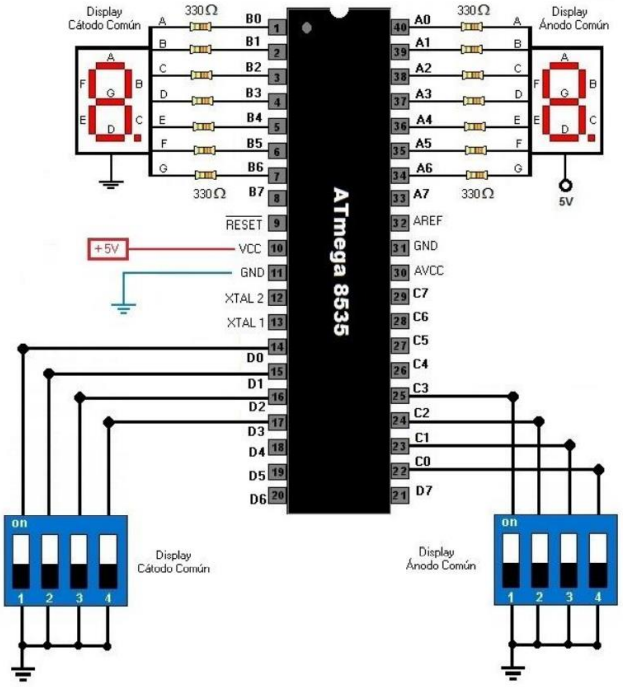

Figura 5. Circuito para el convertidor BCD a 7 segmentos con los displays ánodo y cátodo común.

Estructura del programa

Código de la configuración de los periféricos utilizados y código del programa principal en C, proporcionados por el IDE de CodeVision.

```
#include <mega8535.h>
unsigned char variableA;
unsigned char variableB;
const char tabla7segmentos[16]= {0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,
0x6f,0x77,0x7c,0x58,0x5e,0x79,0x71};

DDRA=(1<<DDA7) | (1<<DDA6) | (1<<DDA5) | (1<<DDA4) | (1<<DDA3) | (1<<DDA2) |
(1<<DDA1) | (1<<DDA0);

PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) |
(0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);

DDRB=(1<<DDB7) | (1<<DDB6) | (1<<DDB5) | (1<<DDB4) | (1<<DDB3) | (1<<DDB2) |
(1<<DDB1) | (1<<DDB0);

PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) |
(0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);

DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) |
(0<<DDC1) | (0<<DDC0);

PORTC=(1<<PORTC7) | (1<<PORTC6) | (1<<PORTC5) | (1<<PORTC4) | (1<<PORTC3) |
(1<<PORTC2) | (1<<PORTC1) | (1<<PORTC0);

DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) |
(0<<DDD1) | (0<<DDD0);

PORTD=(1<<PORTD7) | (1<<PORTD6) | (1<<PORTD5) | (1<<PORTD4) | (1<<PORTD3) |
(1<<PORTD2) | (1<<PORTD1) | (1<<PORTD0);

TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) |
(0<<CS00);

TCNT0=0x00;

OCR0=0x00;

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;

ASSR=0<<AS2;

TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<CS21) |
(0<<CS20);

TCNT2=0x00;

OCR2=0x00;

TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) |
(0<<TOIE1) | (0<<OCIE0) | (0<<TOIE0);

MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);

MCUCSR=(0<<ISC2);

UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<UCSZ2) |
(0<<RXB8) | (0<<TXB8);

ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) |
(0<<ACIS1) | (0<<ACIS0);

SFIOR=(0<<ACME);

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

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

TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);

.
.
    while (1)
    {
        variableB = PIND&0x0f; //Enmascaramos los 4 bits menos significativos
        if(variableB < 10) PORTB=tabla7segmentos[variableB];

        else PORTB = 0x79;

        variableA = PINC&0x0f;
        PORTA = ~tabla7segmentos[variableA];
    };
```

Simulación

La simulación se realizó en el software Proteus Design Suite, previamente cargado el programa en formato ".hex" obtenido del software CodeVision AVR, el cual es un software para el microcontrolador Microchip AVR y sus contrapartes XMEGA.

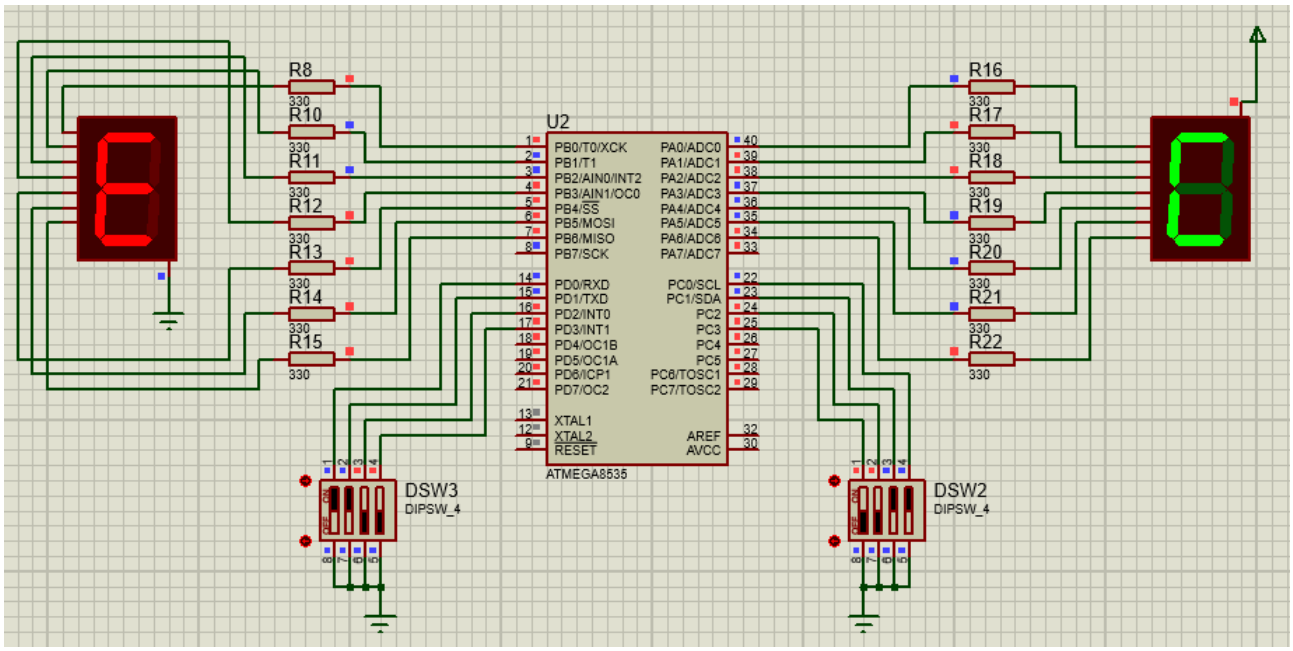

Figura 6. Simulación de la práctica 3.

Fotografía del circuito armado

A continuación, se muestra la figura 7 la evidencia sobre la realización y prueba de la practica 3.

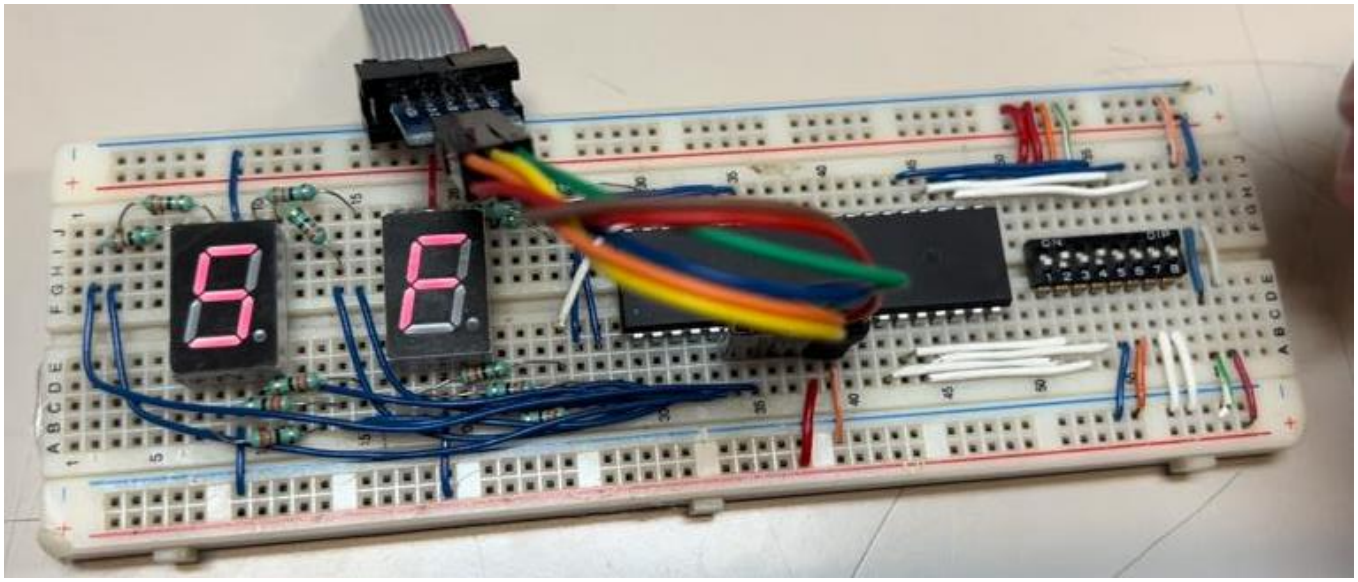

Figura 7. Circuito armado y funcionamiento de la práctica 3.

Observaciones y conclusiones Individuales

Ocampo Téllez Rodolfo

En esta práctica al ocuparse dos displays de 7 segmentos se tuvieron que realizar un número mayor de conexiones en nuestra protoboard, misma que fue utilizada en la práctica 2, cada display mostraría un comportamiento distinto, lo cuál se logró mediante la programación del software. Fue necesario el manejo de arreglos y registros, ya que era la forma más eficiente de realizar el programa, se requirió este conocimiento previo para comprender la lógica del código que fue proporcionado por el profesor. Mientras que en el display cátodo común, que fue limitado a 9 en su conteo mediante un condicional, se utilizó de forma normal nuestro arreglo previamente definido, en el display ánodo común y cuyo conteo fueron los dígitos hexadecimales completos, se tuvo que negar el arreglo para que funcionará de forma correcta en nuestro circuito, una vez realizado ese mínimo cambio se obtuvo un funcionamiento correcto.

Ruvalcaba Flores Martha Catalina

El conocimiento para realizar la practica 3, fue relativamente sencillo puesto que solo era programar un contador BCD a 7 segmentos utilizando arreglos, se cumplió con el objetivo de mostrar en el display de cátodo común los números del 0 al 9, cuando excedía este límite, mostraba una “E” de error. Por otro lado, para el display de ánodo común, se mostraría en hexadecimal del 0 al 15. En esta práctica se aprendió a como negar todo un arreglo con el símbolo “~”. De esta forma, no se tuvo que crear otro arreglo para el display de ánodo común, puesto que con negar el arreglo de la tabla de 7 segmentos fue suficiente para que funcionara dicho display.

Sandoval Hernández Eduardo

Esta práctica resultó ser un poco más complicada al momento de armar el circuito en la protoboard debido a que hubo que realizar más conexiones con los displays de 7 segmentos y se llegaron a conectar de manera intercambiada, una vez armado el circuito se comprobó que funcionara tal y como se pidió en la especificación de la práctica: un display (cátodo común) mostraría los números del 0 al 9 y para números posteriores mostraría una “E” para “Error” mientras que el segundo display (ánodo común) mostraría los números del 0 al 15 en notación hexadecimal, en un inicio el display de ánodo común no respondió adecuadamente ya que se había omitido el símbolo “~” en el código pero una vez corregido ese detalle la practica funcionó de manera satisfactoria.

Bibliografía

- F. Aguilar. (2020, Octubre 12). 02 Práctica de Contador Ascendente/Descendente. Introducción a los Microcontroladores. [Online]. Disponible en: https://youtu.be/wpXVv6_KNkQ
- Atmel Corporation. (2006). 8-bit Microcontroller with 8K Bytes In-System Programmable Flash. [Online]. Disponible en: https://drive.google.com/file/d/1acpQaDlsyLHr3w3ReSgrlXRbshZ5Z-lb/view
- F. Cereijo. (2004, Diciembre 28). Microcontroladores PIC. [Online]. Disponible en: https://drive.google.com/file/d/1mAydVlkZYqtYXhFQ0Tnr_UjZyjiwIp90/view
- HetPro. (2019). Display 7 Segmentos ánodo y cátodo común. [En línea]. Disponible en https://hetpro-store.com/TUTORIALES/display-7-segmentos-anodo-catodo-comun/
- G. Panton. (2017). Código BCD. [En línea]. Disponible en: https://arquiconsamuel.blogspot.com/2017/08/codigo-bcd.html