Instituto Politécnico Nacional

Escuela Superior de Cómputo

Departamento de Ciencias e Ingeniería de la Computación

Sistemas en Chip

Practica 04 “Contador de 0 a 9”

Profesor: Fernando Aguilar Sánchez

Grupo: 6CM1

Equipo 3

Alumnos:

Ocampo Téllez Rodolfo

Ruvalcaba Flores Martha Catalina

Sandoval Hernández Eduardo

Fecha de entrega: 11/12/2022

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 Teórica

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.

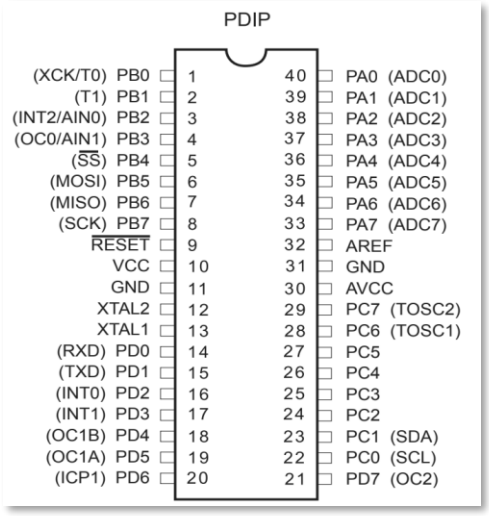

Figura 1. Configuración de pines ATmega8535

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.

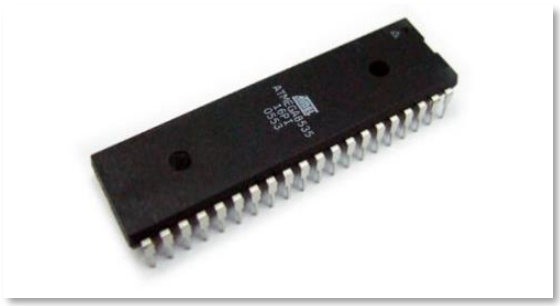

Figura 2. Microcontrolador atmega8535

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.

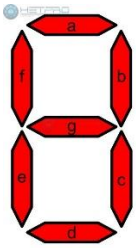

Figura 3. Disposición de los leds en un display de 7 segmentos.

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.

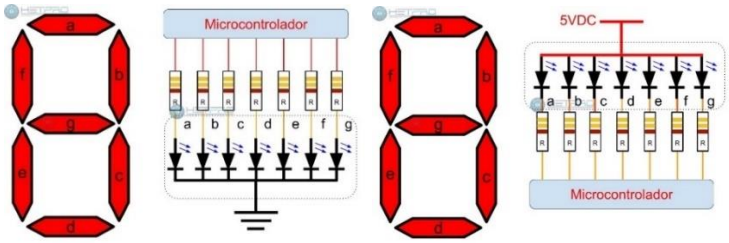

Figura 4 . Conexiones en display de 7 segmentos.

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.

Contador Binario

Un contador electrónico digital es muy útil por ello en la actualidad estamos rodeados de dispositivos que disponen de algún tipo de contador digital, incluso en la mayoría de los electrodomésticos vienen equipados con uno. Nuestro contador digital tiene un campo muy amplio para su aplicación de forma rotacional (cuantas vueltas efectúa un objeto) o de forma secuencial (Ej. en una empresa para el conteo de cajas de producto, etc.)

Este dispositivo permite visualizar la formación numérica de manera ascendiente desde 0 hasta 9 en un display. Consta de pocos componentes electrónicos y es alimentado por una fuente regulable que suministra 5 volts, es ampliamente utilizado a niveles más complejos para fines comerciales en artefactos electrónicos. Contiene circuitos integrados que permiten dicha función.

Materiales y Equipo empleado

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

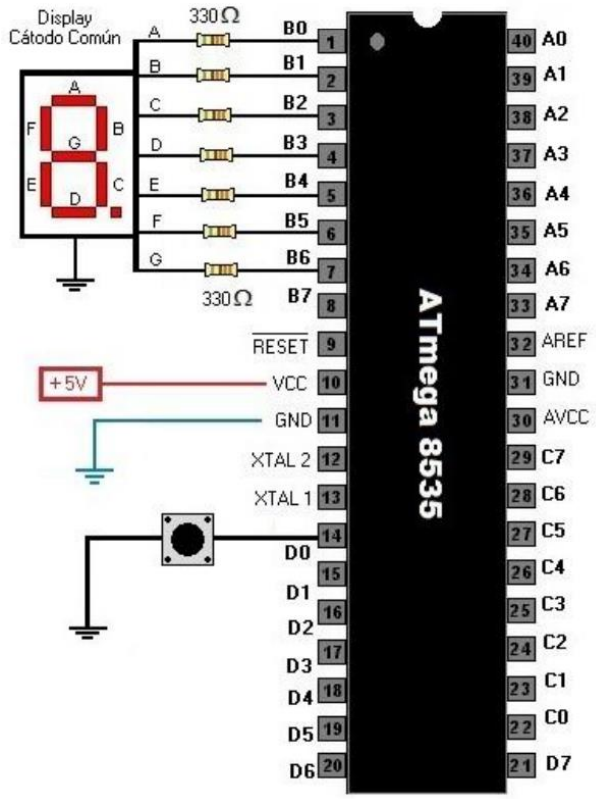

Figura 5. Circuito para el contador de 0 a 9.

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>

#define boton PIND.0

unsigned char variable;

const char
tabla7segmentos[16]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c,0x
58,0x5e,0x79,0x71};

void main(void)

{

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=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) |
(0<<PORTC2) | (0<<PORTC1) | (0<<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) {
      if (boton==0)
        variable++;
      if(variable==10)
        variable=0;
      PORTB = tabla7segmentos[variable]
      PORTA = ~tabla7segmentos[variable]; }
}
```

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.

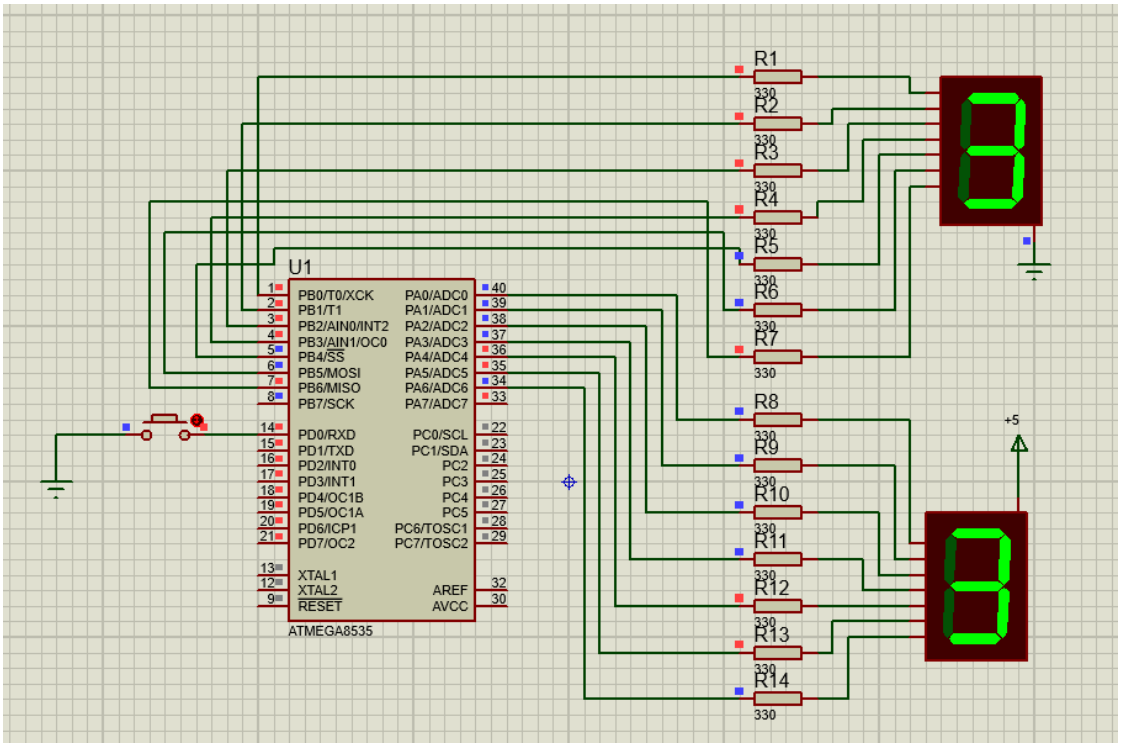

Figura 6. Simulación de la práctica 4.

Fotografía del circuito armado

A continuación, se muestra la figura 7 la evidencia sobre la realización y prueba de la practica 4.

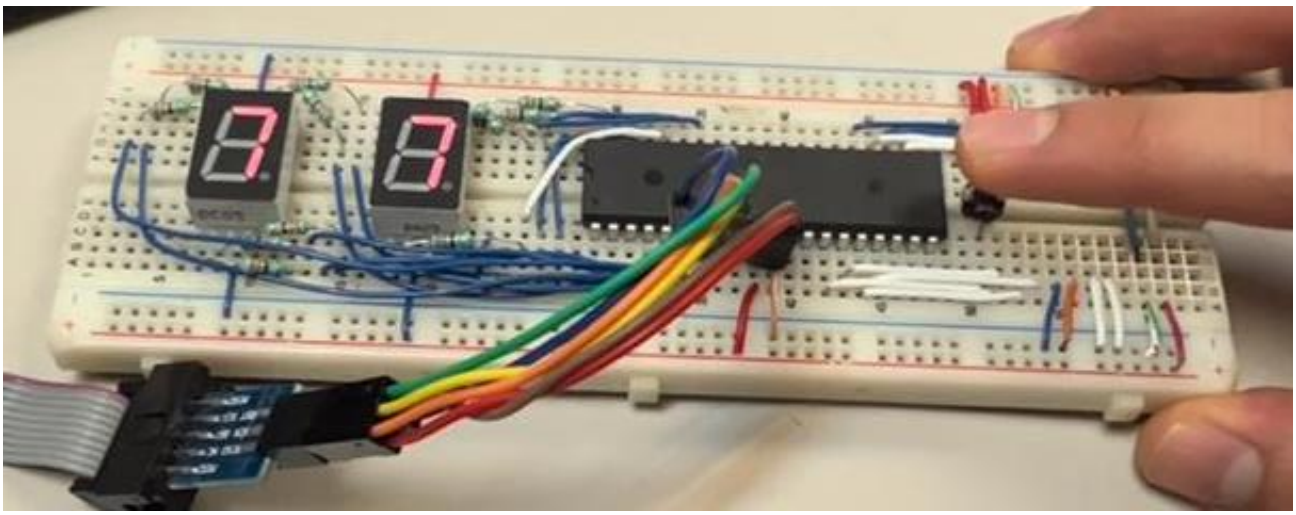

Figura 7. Circuito armado y funcionamiento de la práctica 4.

Observaciones y conclusiones Individuales

Ocampo Téllez Rodolfo

En esta práctica se pudo notar desde el inicio una similitud con la práctica anterior. En cuestión al hardware, apenas se realizó un cambio, la estructura del circuito por tanto fue casi idéntica. El cambio más significativo fue en el software, ya que en esta ocasión se utilizó el etiquetado de puertos facilitando la lectura de código y evitando el uso de los registros de los puertos. En cuanto al funcionamiento del circuito, lo que cabe destacar es que la visualización del conteo era imposible ya que por defecto, el microcontrolador posee una velocidad de funcionamiento muy alta, lo que implicó que a la hora de ejecución, el conteo se mostraba demasiado rápido, esto es velocidades de alrededor de 1 μs, por tanto solo podíamos observar como se 'quedaba' encendido el display en todos sus segmentos, y esto debido a la limitada capacidad del ojo humano y no porque se quedara en el valor '8'. Por lo tanto, aunque funcionaba correctamente, se debe tomar en cuenta que al usar un integrado con esta velocidad de reloj, se tendría que aplicar una ralentización si se desea que sea observable la acción que este realizando.

Ruvalcaba Flores Martha Catalina

En esta práctica, solamente se pide un contador del 0 al 9, y este sea activado por un push-button y reflejado en un display de 7 segmentos, sin embargo, en su programación se pudo observar que, en conteo, este iba ser programado a una alta velocidad, tal que en el display se mostraba el conteo de forma salteada, tal que parecía que no iba en sucesión los números. Al principio, el equipo estaba confundido por como se comportaba, ya que, si se dejaba presionado el botón, el display se prendía por completo. Una vez que el profesor nos explicó por qué sucedía esto, entendimos por qué es importante la frecuencia de funcionamiento del microcontrolador, ya que la frecuencia de reloj permite indicar cuando se abre y cierra el flujo de una corriente eléctrica, cuya unidad es en Herzt (Hz), la cual representa un ciclo por segundo, asimismo la importancia de la activación por flancos la cual será aplicada para la practica 5 y 6. Por lo tanto, debido a la velocidad de ejecución del programa, se muestra en el display unos saltos de números, esto debido a que se ejecuta a la velocidad del microcontrolador (oscilador interno), la cual es de 1 MHz, tal que, cada instrucción del código se ejecuta aproximadamente en un microsegundo. Causando que no exista un control sobre el conteo.

Sandoval Hernández Eduardo

Esta práctica resultó ser muy similar a la anterior pues optamos por dejar el circuito físico muy parecido a la práctica anterior solo cambiando el dip-switch por un push-button e implementar el código necesario para el conteo, se pudo apreciar que el contador no contaba de forma eficiente debido a la velocidad con la que opera el microcontrolador ya que al ser este muy rápido hacía los conteos de igual manera muy veloz y como consecuencia los números parecían más bien ser aleatorios, incluso si de dejaba presionado el botón se podía apreciar como el display se mantenía encendido por completo como si se tratara del número "8" siendo que en realidad estaba mostrando todos los números muy rápido. Considero que esta práctica es esencial para comprender lo que significa que la frecuencia de operación del microcontrolador sea de 1 MHz pues es algo que se deberá tener en consideración en prácticas futuras.

Bibliografía

- F. Aguilar. (2020, Octubre). 04 Práctica de Contador de 0 a 9. Introducción a los Microcontroladores. [Online]. Disponible en: https://youtu.be/xNhjvh_YW7s

- 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/

- A. Solano. (2014). Contador BCD de 0-9 con temporizador 555 (Automatización): [En línea]. Disponible en: https://www.monografias.com/trabajos102/contador-bcd-0-9-temporizador-555-automatizacion/contador-bcd-0-9-temporizador-555-automatizacion