Instituto Politécnico Nacional

Escuela Superior de Cómputo

Departamento de Ciencias e Ingeniería de la Computación

Sistemas en Chip

Práctica 11

"Memoria EEPROM"

Profesor: Fernando Aguilar Sánchez

Grupo: 6CM1

Equipo 3

Alumnos:

Ocampo Téllez Rodolfo

Patlani Mauricio Adriana

Ruvalcaba Flores Martha Catalina

Sandoval Hernández Eduardo

Fecha de entrega: 23/12/2022

Objetivo

Al término de la sesión, los integrantes del equipo contaran con la habilidad de hacer uso de la memoria EEPROM del microcontrolador.

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.

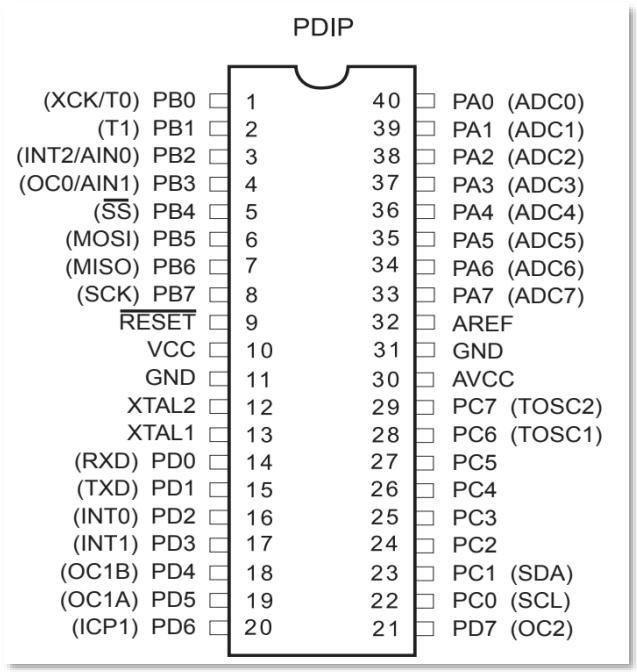

Figura 1. Configuración de pines ATmega8535

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.

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.

Memorias: EPROM y EEPROM

La memoria EPROM, es una subdivisión de la memoria ROM. Así pues, tenemos que la EPROM, acrónimo de Erasable Programmable Read Only Memory, es no volátil, programable y borrable. Puede mantener la información almacenada durante largos períodos de tiempo; adicionalmente, permite que la misma sea leída de forma ilimitada.

EEPROM responde a "Electrically Erasable Programmable Read Only Memory" que se puede traducir como Memoria programable borrable de solo lectura. También se la conoce como E-2-PROM. Como su nombre sugiere, una EEPROM puede ser borrada y programada con impulsos eléctricos. Al ser una pieza que se puede gestionar por estos impulsos eléctricos, podemos realizar todas estas operaciones de reprogramación sin tener que desconectarla de la placa a la cual va conectada.

La EEPROM también se conoce como "non-volatile memory" o memoria no volátil y es debido a que cuando se desconecta la energía, los datos almacenados en la EEPROM no serán eliminados quedando intactos. La información almacenada dentro de este dispositivo puede permanecer durante años sin una fuente de energía eléctrica.

La mayor aplicación de este tipo de memoria se produce en los sistemas micro controlados o micro procesados. De esta manera, la EEPROM se convierte en el medio donde es posible guardar información de forma semipermanente, tal como: sistemas operativos, aplicaciones informáticas, lenguajes de programación y rutinas.

La primera ventaja que debemos mencionar acerca de la memoria EEPROM es la posibilidad de reutilizarla tantas veces como sea necesario. Para ello, debemos siempre de asegurarnos de borrar el contenido y grabar uno nuevo. Por otra parte, la memoria EEPROM representa un importante avance tecnológico, ya que nos permite modificar o corregir la información que se encuentra grabada dentro de ella.

EEPROM en los AVR

Los microcontroladores AVR contienen en su interior una memoria de este tipo, de reducida capacidad pero muy útil para guardar datos que no deseamos que se pierdan si el micro se queda sin energía, por ejemplo para guardar configuraciones del sistema que estén disponibles entre cada reinicio o encendido del micro.

La memoria EEPROM de los AVR se organiza en un espacio separado de la memoria de programa y la memoria RAM. Según el fabricante, la memoria puede resistir al menos 100,000 ciclos de escritura/borrado y el tiempo de almacenamiento de datos es de básicamente es el tiempo de vida del microcontrolador. En específico, el Atmega 8535 tiene 512 bytes de EEPROM con una resistencia: 100.000 ciclos de escritura / borrado.

La programación en lenguaje C/C++ de la memoria EEPROM es muy sencilla, se puede realizar de 2 maneras: una es escribiendo y leyendo sobre los registros del micro asociados y la otra es utilizando la biblioteca eeprom.h incluida en la colección avr-libc.

Para acceder a la memoria EEPROM desde código se necesitan acceder a sus registros. Estos registros se encuentran mapeados en el espacio de dirección de E/S (I/O space) y se escribe y leen como cualquier otro registro. Hay ciertos puntos que se deben de tomar en cuenta a la hora de programar esta memoria en los AVR, por ejemplo, la hoja de datos menciona que la memoria tiene un tiempo de acceso mínimo que se debe cumplir entre cada escritura. También menciona que los voltajes de alimentación deben ser lo más estable posibles para evitar corrupción de datos al momento de escribir.

Materiales y Equipo empleado

- CodeVision AVR
- AVR Studio 4
- Microcontrolador ATmega 8535
- 1 Display cátodo común
- 7 Resistores de 330 Ω a ¼ W
- 2 Push Button

Desarrollo Experimental

1.- Haga un programa en el cual con un botón conectado al pin C0 incrementará el valor en el display conectado en el puerto B y cuando se presione el botón C1 lo guarde en la memoria EEPROM. Después desconecte el microcontrolador de la energía eléctrica y vuélvalo a conectar para que observe que el dato se quedó guardado.

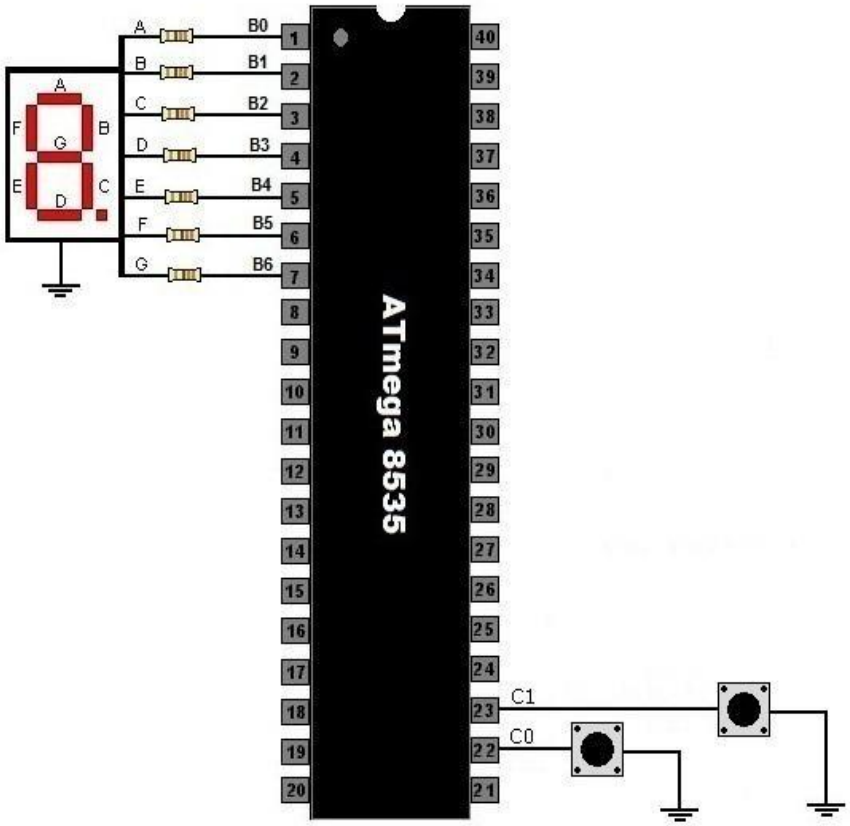

Figura 2. Circuito para el uso de la memoria EEPROM.

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>

#include <delay.h>

#define boton PINC.0

#define boton_guarda PINC.1

bit botonp;

bit botona;

unsigned char var; //Ahora la variable se guarda en eeprom

const char tabla7segmentos [10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7c,0x07,0x7f,0x6f};

eeprom char datoaguardar;

void checa_boton (void); // Aquí se declaran todas las funciones que se van usar

void main(void)

{

// Input/Output Ports initialization

// Port A initialization

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);

// Port B initialization

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);

// Port C initialization

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);
```

```
// Port D initialization
DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) |
(0<<DDD1) | (0<<DDD0);
PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) |
(0<<PORTD2) | (0<<PORTD1) | (0<<PORTD0);
…
…
…
if (datoaguardar>10)
datoaguardar=0;
var=datoaguardar;
while (1)
{
checa_boton();
PORTB=tabla7segmentos [var];
PORTA = ~tabla7segmentos[var];
if (boton_guarda==0) //Si se presiona el botón de guardar
datoaguardar=var; //se grabara la eeprom con el valor de var
};
}
void checa_boton (void)
{
if (boton==0)
botona=0;
else
botona=1;
if ((botonp==1)&&(botona==0)) //hubo cambio de flanco de 1 a 0
{
var++; //Se incrementa la variable
```

```
if (var>=10)
var=0;
delay_ms(40); //Se coloca retardo de 40mS para eliminar rebotes
}
if ((botonp==0)&&(botona==1)) //hubo cambio de flanco de 0 a 1
delay_ms(40); //Se coloca retardo de 40mS para eliminar rebotes
botonp=botona;
}
```

Simulaciones

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.

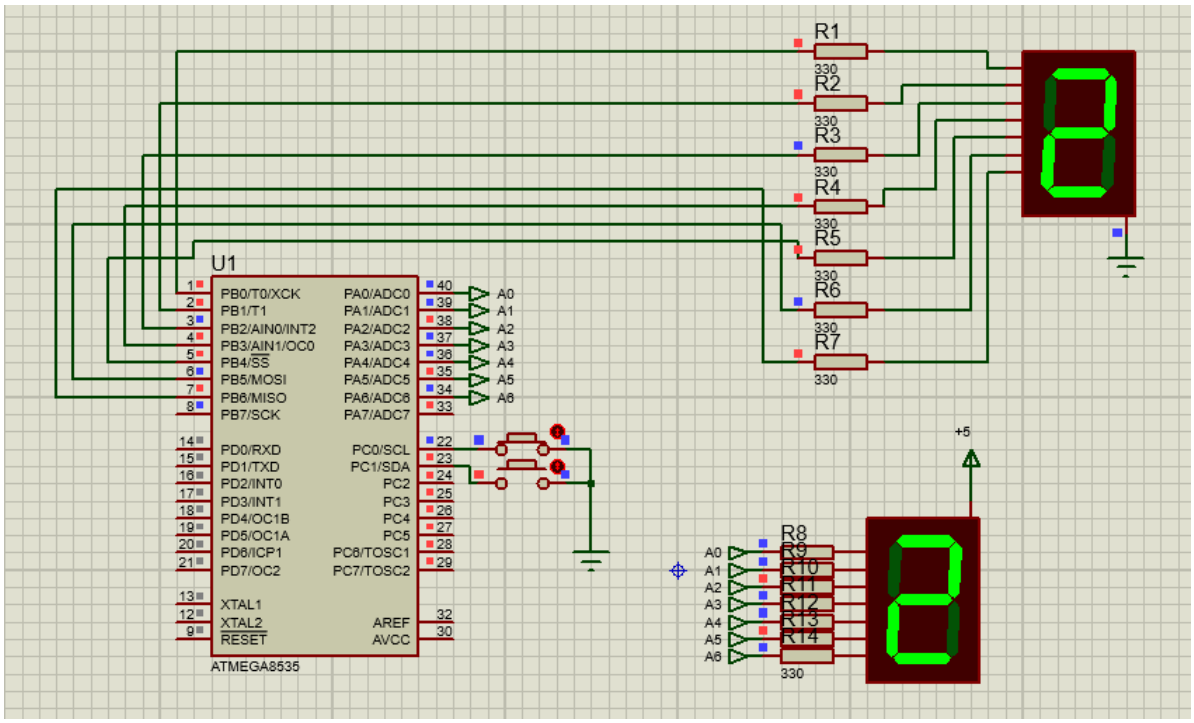

Figura 3. Simulación de la práctica 11.

Fotografía del circuito armado

A continuación, se muestra la figura 4 la evidencia sobre la realización y prueba de la practica 11.

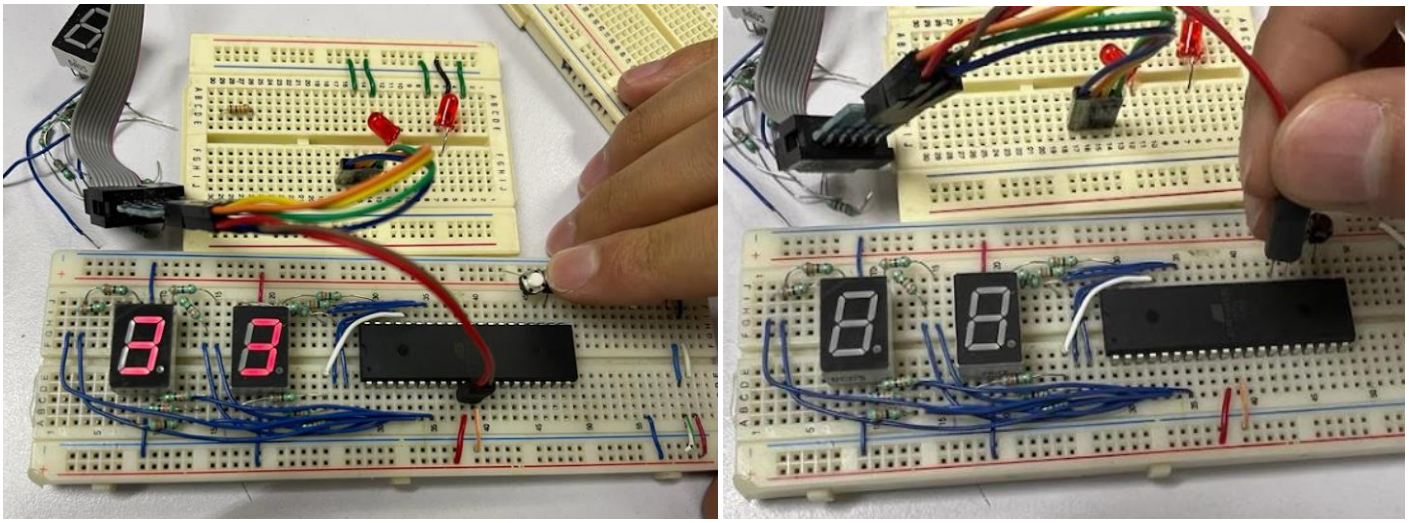

Figura 4. Circuito armado y funcionamiento de la práctica 11.

Observaciones y conclusiones Individuales

Ocampo Téllez Rodolfo

Esta práctica consistió en probar y observar el funcionamiento de la memoria EEPROM que incluyen los microcontroladores, en específico el Atmega 8535. Usarla fue sencillo pues en la biblioteca estándar de este modelo de Atmega incluye el tipo ‘eeprom’ que al usarlo le indica al microcontroladore que los datos que se le pasen serán guardados en esta memoria. En cuánto a su funcionalidad, es útil para no perder datos al momento de que se llegue a cortar la alimentación al circuito, tal como se pudo comprobar satisfactoriamente, y aunque esta aplicación solo registraba un digito del 0 al 9, saber usarla puede ser de ayuda en prácticas posteriores.

Patlani Mauricio Adriana

Con esta práctica se aprendió a explotar aun mas el microcontrolador Atmega usando su memoria EEPROM, el cual se puede visualizar que tal y como se vio en la introducción teórica, es una memoria no volátil y que además el dato no se pierde al dejar de alimentar eléctricamente el circuito. Aunque esto nos deja que nosotros borremos el contenido de la memoria si se requiere guardar un nuevo dato. La práctica fue sencilla de hacer, ya que solo de implementó el circuito y el código fue dado por el profesor.

Ruvalcaba Flores Martha Catalina

Con la práctica de la memoria EEPROM, se entendió que esta sirve para almacenar un dato de entrada, tal que este dato no se pierde cuando el circuito no está alimentado por una corriente, y una vez que se vuelve a alimentar, este reflejó su dato almacenado en el display. Fue interesante observar cómo se comportó el circuito, ya que se utilizaron conocimientos adquiridos de las practicas anteriores. De igual forma, por el código proporcionado por el profesor, fue más sencillo su realización y prueba del circuito.

Sandoval Hernández Eduardo

Siguiendo con la implementación del teclado matricial, es esta ocasión se probó hacer uso de la memoria EPROM para almacenar un dato ingresado desde el teclado, esto fue en particular importante ya que usualmente cuando se corta la alimentación hacia el microcontrolador se suelen perder los datos que no se relacionan con el programa cargado en si sino con los que se ingresan posterior a su programación, con esto se podría implementar programas que almacenen datos en particular como lo fue el número ingresado en esta práctica pero con otras aplicaciones, lo cual será importante tenerlo en cuenta para las prácticas futuras.

Bibliografía

• 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

• Informática para todos. (s.f). ¿Qué es una EEPROM?. [Online]. Disponible en: https://ordenadores-y-portatiles.com/eeprom/

• Vida Bytes. (Marzo, 2022) Memoria EPROM: Significado y características. [Online]. Disponible en: https://vidabytes.com/memoria-eprom/

• A. Orozco. (Arbirl 2018) AVR programación en C – 15 Lectura y escritura en la Memoria EEPROM de los AVR. [Online]. Disponible en: https://vidaembebida.wordpress.com/2018/04/20/avr-programacion-en-c-15-lectura-y-escritura-en-la-memoria-eeprom-de-los-avr/