

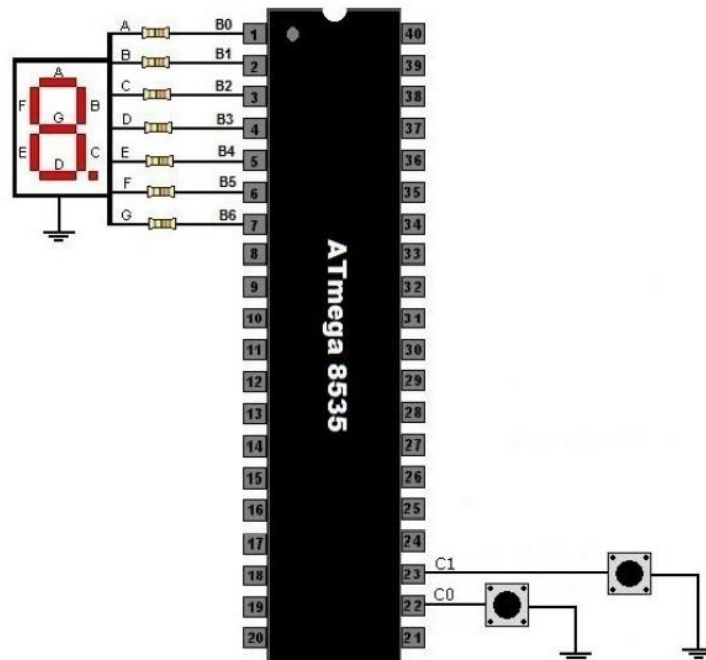


INSTITUTO POLITÉCNICO NACIONAL ESCUELA SUPERIOR DE CÓMPUTO

SISTEMAS EN CHIP

PRACTICA No. 11

"Memoria EEPROM"



ALUMNO: RENTERIA ARRIAGA JOSUE.

GRUPO: 6SCM1

PROFESOR: FERNANDO AGILAR SÁNCHEZ.

05/NOVIEMBRE/2022

I. OBJETIVO GENERAL:

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

II. MATERIAL Y EQUIPO EMPLEADO

- ✓ CodeVision AVR.
- ✓ AVR Studio 4.
- ✓ Microcontrolador ATmega 8535.
- ✓ 1 Display cátodo común.
- ✓ 7 Resistores de $330\ \Omega$ a $\frac{1}{4}\text{ W}$.
- ✓ 2 Push Button.

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 integrar una memoria EEPROM para que se pueda observar en un Display de 7 Segmentos como esta se comporta.

En esta práctica se estudiará la implementación de un Programa y Circuito para programar el puerto C como entrada de nuestros 2 botones uno que va de uno en uno (del 0 al 9), y el otro para guardar el numero en que nos encontramos (en mi caso agregue otro botón para reiniciar el contador y poder observar que si se guardara el número que queríamos) y el puerto B como la salida que se conectará al Display de 7 segmentos (Cátodo Común). La práctica nos mostrara el numero seleccionado de nuestro botone y otro guardara el numero en que nos encontramos en el Display de 7 segmentos. 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 (Display de 7 Segmentos), concepto de memoria EEPROM que se vio en Arquitectura y de Electrónica Analógica para poder realizar correctamente el armado de nuestro circuito y que este funcione correctamente.

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 implementará un Display para los puertos de salida y 3 botones para el puerto de entrada de nuestro circuito.

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

La EEPROM (Electrically Erasable Programmable Red-Only Memory) es un tipo de memoria ROM, es decir, de memoria no volátil en la que los datos se almacenarán de forma permanente, incluso si se retira el suministro de energía. Eso las sitúa en el otro lado de las RAM (Random Access Memory), que pierden todos sus datos cuando no se alimentan de energía.

En el caso de la EEPROM, no es una memoria como la ROM, en la que se graban los datos y ya no pueden ser alterados. La EEPROM, al igual que la flash, admite ser alterada cuando se necesite. Es decir, se pueden almacenar unos datos y borrarlos para almacenar otros diferentes.



De hecho, como bien indican sus siglas, es una memoria borrrable eléctricamente (electrically erasable) para su reprogramación. Esto difiere con otros tipos de ROM, que también son borrrables como las EPROM, pero que en este caso no se usa electricidad para borrar las células de memoria, sino que tenían una «ventanita» de cuarzo en el chip para poder proyectar una luz UV con la que se borrraba.

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. 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.

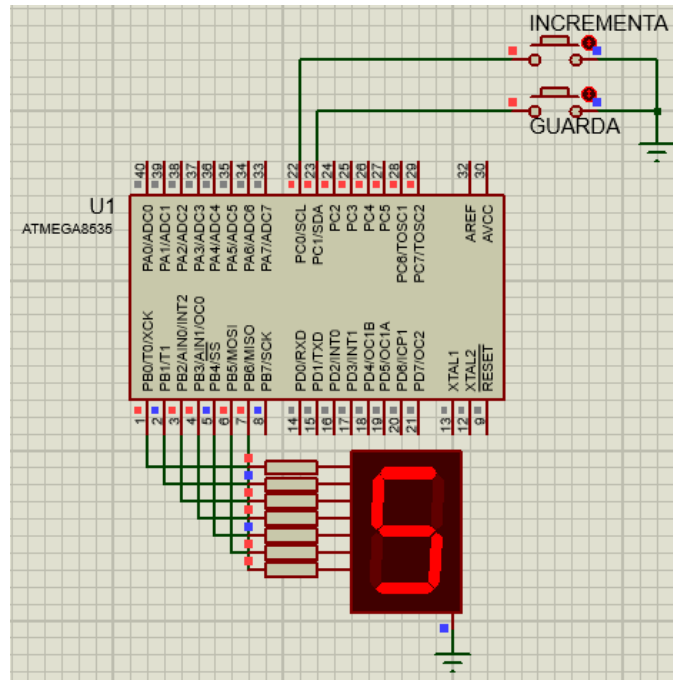


Fig.1_Circuito Armado y Simulado en Proteus.

CIRCUITO EN FISICO ARMADO.

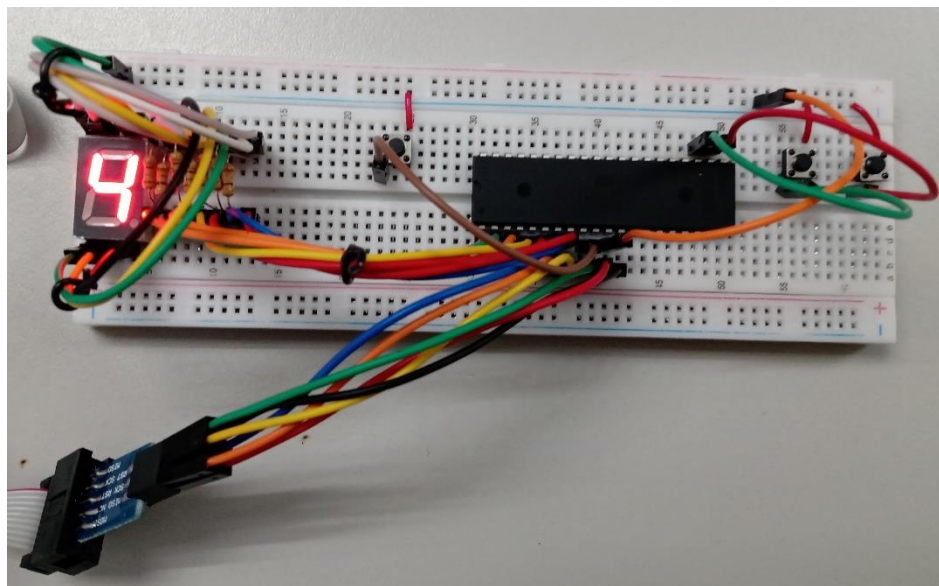


Fig.2_Circuito Armado en Físico (Memoria EEPROM).

V. CÓDIGO

```
#include <mega8535.h>
#include <delay.h>
#define btn_incrementa PINC.0
#define btn_guarda PINC.1

bit pasado_i, actual_i, pasado_g, actual_g;
eeprom unsigned char guardable;
unsigned char index;
const char tabla7segmentos [10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};

void main(void)
{
    // Declare your local variables here

    // Input/Output Ports initialization
    // Port A initialization
    // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
    DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) | (0<<DDA1) |
    (0<<DDA0);
    // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
    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=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
    PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (0<<PORTD2) |
    (0<<PORTD1) | (0<<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);

index = guardable;

while (1)
{
    actual_i = btn_incrementa;
    actual_g = btn_guarda;

    //Boton de incremento
    if (pasado_i == 1 && actual_i == 0) {
        index++;
        if (index > 9) index = 0;
        delay_ms(40);
    }

    if (pasado_i == 0 && actual_i == 1) {
        delay_ms(40);
    }

    //Boton de guardado
    if (pasado_g == 1 && actual_g == 0) {
        guardable = index;
        delay_ms(40);
    }

    if (pasado_g == 0 && actual_g == 1) {
        delay_ms(40);
    }

    pasado_i = actual_i;

```

```
    pasado_g = actual_g;
    PORTB = tabla7segmentos[index];
}
}
```


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. En esta práctica al momento de armar el circuito y simularlo pudimos observar cómo se implementaría una memoria EEPROM, que realmente no fue tan difícil ya que el profesor nos dio el Código.

Tras la investigación y explicación de la práctica previa me quedo un poco más claro como poder implementar una memoria EEPROM y programar el ATMega8535. También al observar el data set del Display se comprendido mejor como conectarlo en físico este componente.

Para concluir esta práctica podemos decir que fue una experiencia interesante ya que decidí yo programar un botón para reiniciar el programa y ver si realmente se estaba ejecutando correctamente, aunque después vi que poniendo un botón en el puerto del RESET se podría hacer sin programarlo. Gracias a la anterior practica ya fue más fácil de hacerlo, ya que es casi lo mismo que esta.

VII. BIBLIOGRAFÍA

- ✓ CodeVisionAVR. (2018). CodeVisionAVR. 20/Septiembre/2022, de CodeVisionAVR Sitio web: <http://www.hpinfotech.ro/cvavr-download.html>
- ✓ N/A. (2022). EEPROM. 05/Noviembre/2022, de Hwelibre, Sitio web: https://www.hwlibre.com/eeeprom/#Que_es_EEPROM