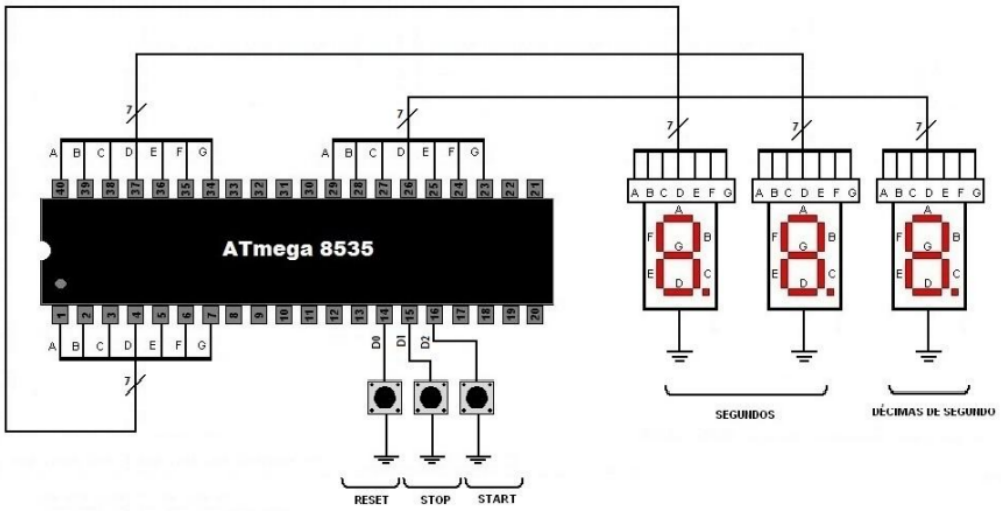
INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

SISTEMAS EN CHIP

PRACTICA No. 08

"Cronometro de 60 segundos"



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 contaran con la habilidad de realizar un cronómetro de 59.9 segundos.

II. MATERIAL Y EQUIPO EMPLEADO

- ✓ CodeVision AVR.
- ✓ AVR Studio 4.
- ✓ Microcontrolador ATmega 8535.
- ✓ 3 Display cátodo común.
- ✓ 21 Resistores de 330 Ω a ¼ W.
- ✓ 3 Push Button.

III. INTRODUCCIÓN TEÓRICA

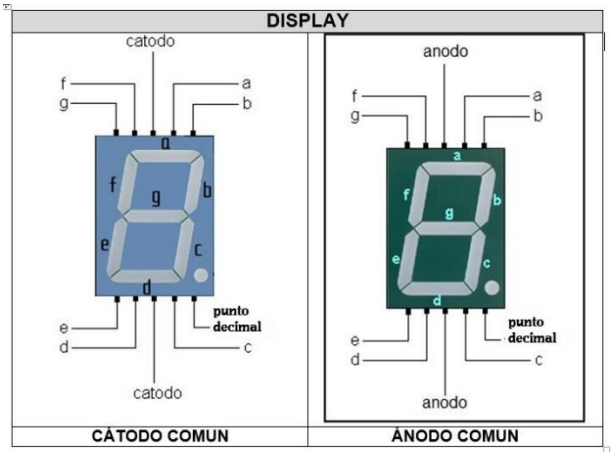
En esta práctica podremos saber utilizar correctamente y aplicar la implementación del código en el lenguaje de programación C de los conceptos de cronometro que llegara hasta 59.9 mostrando nuestros resultados en 3 Display activados con 3 diferentes botones (Reset, Stop y Start) en la entrada y creación del circuito para poder dar solución a la práctica.

En esta práctica se estudiará la implementación de un Programa y Circuito para programar el puerto D como entrada de nuestros 3 botones (Reset, Stop y Start) y el puerto A, B y C como la salida que se conectará al Display de 7 segmentos (Cátodo Común). La práctica solo hará un contador del 00.0 hasta en 60.0, cuando se aprieta el botón de Start iniciara nuestro cronometro, al apretar el botón de stop nuestro cronometro se detendrá en el numero que se quedo y si se vuelve a apretar el botón de Start este continuara en el numero que se quedó y el botón de Reset nos iniciara el conteo en 00.0, todo esto se vera reflejado en nuestros Display. 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) 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 implementarán tres 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:

Cuando hablamos de display de 7 segmentos, estamos hablando nada más ni nada menos de 7 leds que internamente están unidos necesariamente a un ánodo o cátodo común, en este caso es ánodo común, es decir, que tiene todos los polos positivos unidos, e ahí recibe el nombre de ANADO común, lo mismo sucederá con el CATODO común estarán unidos los pines negativos.



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 programa en el que coloque tres Displays, uno en el Puerto A, el otro en el Puerto B y en el puerto C y con una terminal del Puerto D detecte la cuenta a través de 3 botones.

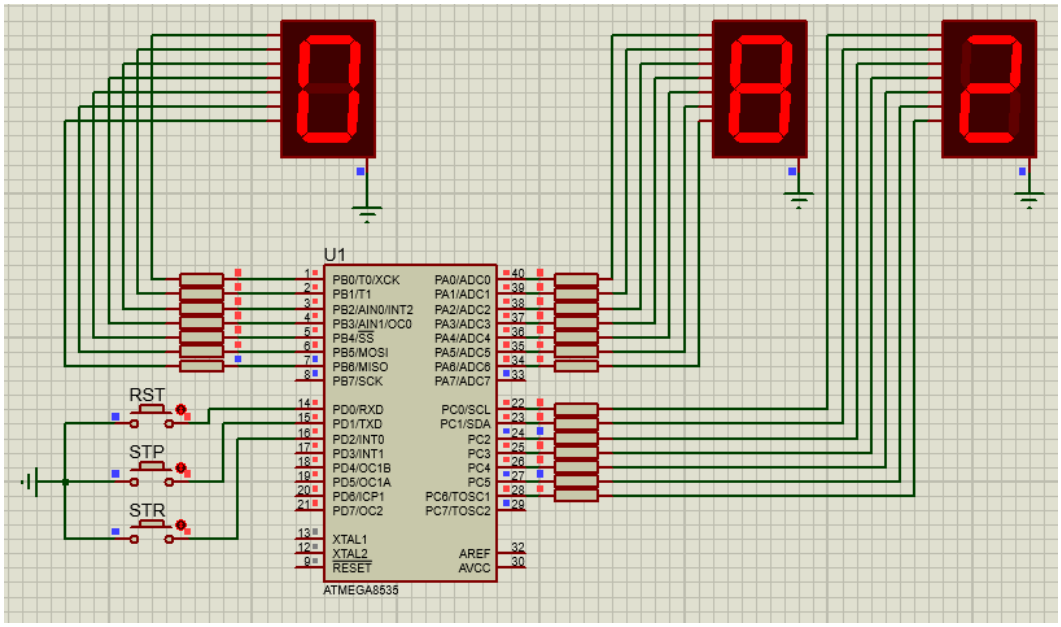


Fig.1_*Circuito Armado y Simulado en Proteus.*

CIRCUITO EN FISICO ARMADO.

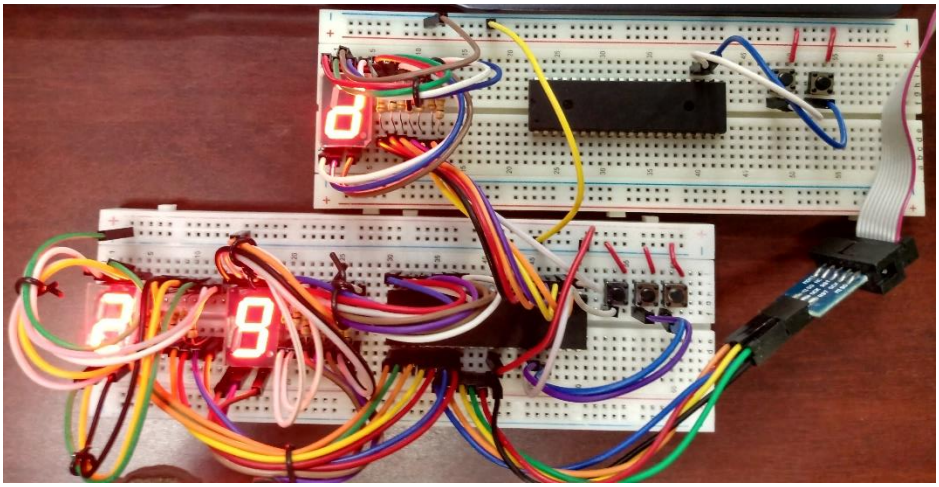


Fig.2_*Circuito Armado en Físico (Cronometro del 00.0 al 60.0).*

V. CÓDIGO

```
#include <mega8535.h>
#include <delay.h>

// Declare your global variables here
#define RESET_BTN PIND.0
#define STOP_BTN PIND.1
#define START_BTN PIND.2

const char mem[10] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F};
unsigned char unidad, decena, decima;
bit debe_avanzar;

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=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
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=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
DDRC=(1<<DDC7) | (1<<DDC6) | (1<<DDC5) | (1<<DDC4) | (1<<DDC3) | (1<<DDC2) | (1<<DDC1) |
(1<<DDC0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) |
(0<<PORTC1) | (0<<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)
      {
        //Revisar por los botones
        if (!START_BTN) debe_avanzar = 1;
        if (!STOP_BTN) debe_avanzar = 0;

        if (!RESET_BTN) {
            decima = 0;
            unidad = 0;
            decena = 0;
            debe_avanzar = 0;
        }

        //Actuar seg?n el estado
        if (decena == 6) debe_avanzar = 0;

        if (debe_avanzar) {
            decima++;

            if (decima == 10) {
                decima = 0;
                unidad++;
            }

            if (unidad == 10) {
                unidad = 0;
```

```
                decena++;
            }
        }

        PORTA = mem[unidad];
        PORTB = mem[decena];
        PORTC = mem[decima];
        delay_ms(30);
      }
}
```

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 un cronometro, que realmente no fue tan difícil.

Tras la investigación y explicación de la práctica previa me quedo un poco más claro como poder implementar 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 nunca había hecho yo mi propio código, ya que, en las practicas anteriores el profesor nos dio el código de como hacer los programas en esta practica y la anterior a mi me toco realizar el código. 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

- ✓ E, Gonzalez. (2019). Display 7 Segmentos ánodo y cátodo común. 26/Septiembre/2022, de Het Prostore, Sitio web: https://hetpro-store.com/TUTORIALES/display-7-segmentos-anodo-catodo-comun/