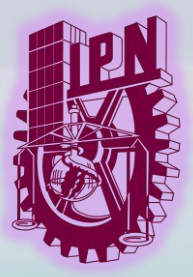

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Introducción a los microcontroladores

3CM16

Practica 08

Cronometro de 59.9 segundos

Equipo 7

Integrantes:

- **Bocanegra Heziquio Yestlanezi**
- **Dominguez Durán Alan Axel**
- **Hernandez Mendez Oliver Manuel**
- **Martinez Cruz José Antonio**

Profesor: Aguilar Sanchez Fernando

Índice

Índice de imagenes 2
Objetivo 3
Introducción 3
Cronometro 3
Materiales y Equipo empleado 5
Desarrollo 6
Circuito simulado 6
Circuito armado 7
Estructura del programa 8
Código CodeVisionAVR 8
Conclusiones 13
Bocanegra Heziquio Yestlanezi 13
Alan Axel Domínguez Durán 13
Martínez Cruz José Antonio 13
Hernández Méndez Oliver Manuel 13
Referencias 14

Índice de imágenes

Imagen 1 Circuito simulado en proteus 6
Imagen 2 Circuito armado en protoboard 7

Objetivo

Al término de la sesión, los integrantes del equipo contaran con la habilidad de realizar un cronómetro de 59.9 segundos.

Introducción

Cronometro

los cronómetros o timers son módulos específicos que se utilizan para medir y controlar el tiempo de forma precisa. Estos timers son componentes integrados en los microcontroladores y se pueden configurar para cumplir diversas funciones dependiendo de las necesidades del sistema [1].

descripción general de cómo se utilizan los cronómetros en los microcontroladores:

- Configuración del cronómetro: En primer lugar, se debe configurar el cronómetro dentro del microcontrolador. Esto implica establecer la resolución del cronómetro, es decir, el tiempo base que se utilizará para las mediciones, así como otros parámetros como el modo de funcionamiento y la fuente de reloj [1].
- Iniciar el cronómetro: Una vez configurado, se puede iniciar el cronómetro. Esto activará el conteo del tiempo según la configuración establecida previamente. Por ejemplo, si se estableció una resolución de 1 microsegundo, el cronómetro comenzará a contar en incrementos de 1 microsegundo [1].
- Medir el tiempo transcurrido: Se puede utilizar el cronómetro para medir el tiempo transcurrido entre eventos específicos dentro del microcontrolador. Esto implica tomar la marca de tiempo en el inicio del evento y luego, en el momento en que se produce el evento deseado, tomar otra marca de tiempo y calcular la diferencia para obtener el tiempo transcurrido [1].
- Generar temporizadores: Los cronómetros también se pueden utilizar para generar temporizadores y controlar acciones en base al tiempo. Por ejemplo,

se puede configurar el cronómetro para que genere una interrupción después de un cierto período de tiempo, lo que puede ser útil para activar tareas programadas o sincronizar acciones [1].

- ♥ Contar eventos externos: Algunos cronómetros en los microcontroladores pueden contar eventos externos, como pulsos o señales, y proporcionar información sobre la frecuencia o duración de esos eventos [1].

Un cronómetro de 59.9 segundos es un dispositivo o función que permite medir con precisión el tiempo transcurrido en intervalos de hasta 59.9 segundos.

El cronómetro de 59.9 segundos generalmente muestra los segundos en una pantalla digital y tiene botones que permiten iniciar, detener y reiniciar la medición del tiempo. Algunos cronómetros también pueden tener funciones adicionales, como un temporizador de cuenta regresiva o una opción de memoria para almacenar múltiples tiempos medidos [].

Pasos para utilizar un cronómetro de 59.9 segundos:

- ♥ Asegúrate de que el cronómetro esté en cero. Si no es así, busca el botón de reinicio o "reset" para ponerlo a cero.
- ♥ Presiona el botón de inicio para iniciar la medición del tiempo. El cronómetro comenzará a contar los segundos.
- ♥ Cuando desees detener el cronómetro, presiona el botón de parada. El tiempo medido se mantendrá en la pantalla.
- ♥ Si necesitas reiniciar el cronómetro para una nueva medición, presiona el botón de reinicio para volver a cero y estar listo para comenzar nuevamente.

Materiales y Equipo empleado

- ♥ CodeVision AVR
- ♥ AVR Studio 4
- ♥ Microcontrolador ATmega 8535
- ♥ 3 Display cátodo común
- ♥ 21 Resistores de 330 Ω a 1⁄4 W
- ♥ 3 Push Button

Desarrollo

1. Diseñe un programa en el que coloque dos Displays, uno en el Puerto A y el otro en el Puerto B y con una terminal del Puerto D detecte la cuenta a través de un par infrarrojo.

Circuito simulado

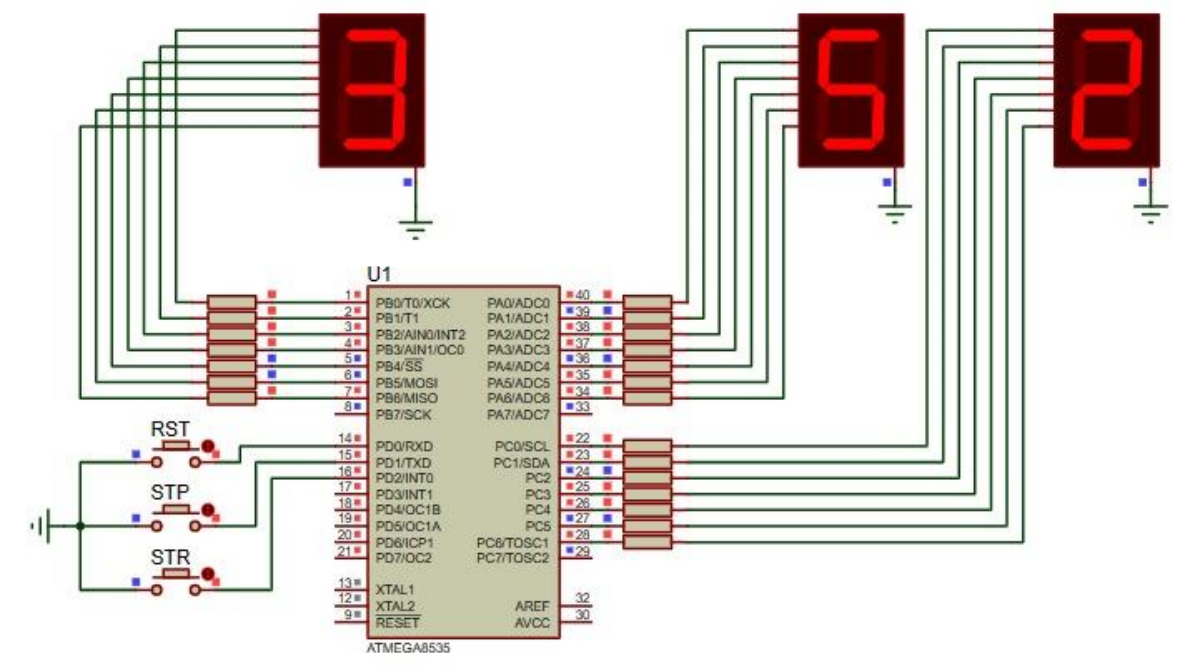

Imagen 1 Circuito simulado en proteus

En la imagen 1 podemos apreciar el circuito funcionado en una simulación mediante el uso de Proteus, con base en el circuito fue construido en la protoboard como se muestra en la imagen 2.

Circuito armado

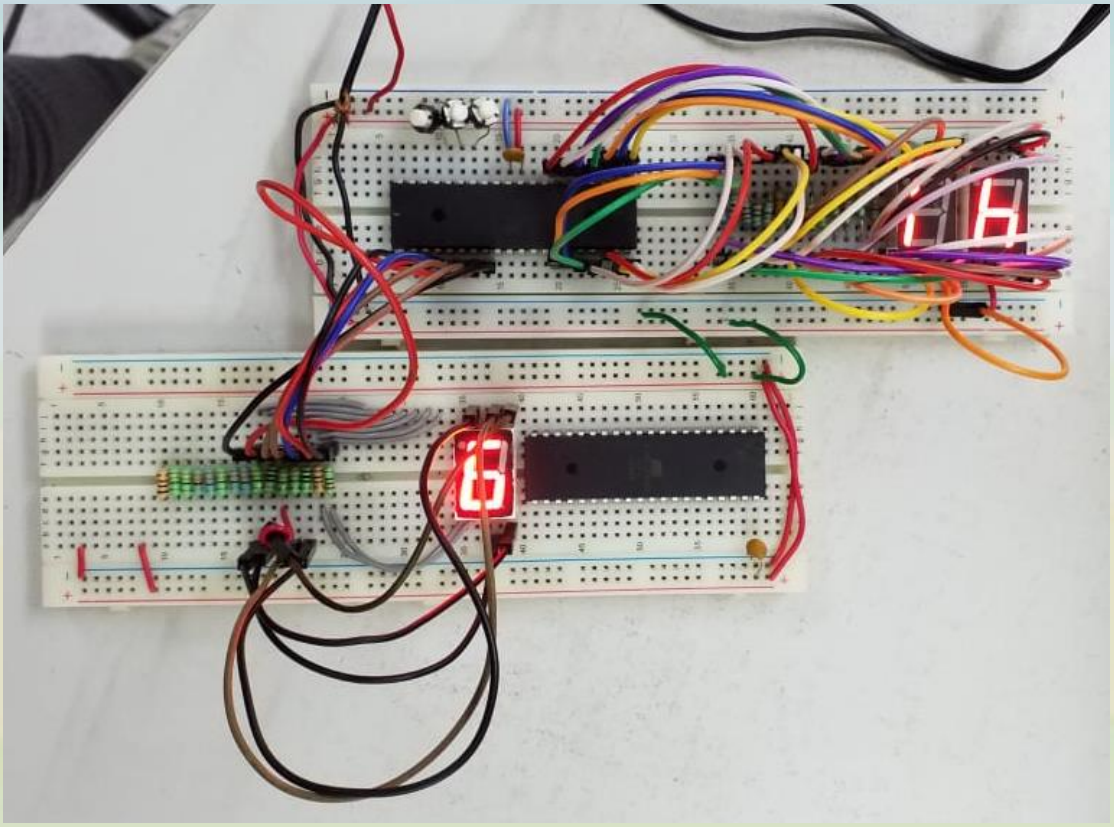

Imagen 2 Circuito armado en protoboard

Como podemos observar en la imagen 2 se encuentran colocados los dos microcontroladores ATMega5835, aunque en realidad solo se utilizó y programo uno, pero se olvidó quitarlo uno, ya que no había espacio suficiente en una sola protoboard para armar el circuito y se reutilizo de la practica 7.

Estructura del programa

Código CodeVisionAVR

```
/*****************
This program was created by the CodeWizardAVR V3.51
Automatic Program Generator
© Copyright 1998-2023 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro

Project :
Version :
Date    : 20/04/2023
Author  :
Company :
Comments:


Chip type               : ATmega8535
Program type            : Application
AVR Core Clock frequency: 1.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 128
*****************/

// I/O Registers definitions

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


#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);
      }
}
```

Conclusiones

Bocanegra Heziquio Yestlanezi

En conclusión, al usar un microcontrolador ATMega8535 para programar un cronómetro de 59.9 segundos nos da una solución precisa para medir intervalos de tiempo. El microcontrolador, con su capacidad de procesamiento y control de periféricos, permite una implementación eficiente y versátil de la funcionalidad de cronometraje. Al utilizar el ATMega8535, se pueden aprovechar las características del microcontrolador, como su alta velocidad de reloj y la capacidad de generar interrupciones, para desarrollar un sistema de cronometraje preciso. Además, el microcontrolador cuenta con suficientes puertos de entrada y salida para conectar y controlar los elementos externos necesarios para la visualización del tiempo, como pantallas LED o LCD.

Alan Axel Domínguez Durán

En esta práctica se implementó un contador, en este caso se agregaron los campos de segundos y milisegundos, además de 3 botones para activar, detener o reiniciar el contador. De esta manera podemos observar la flexibilidad del micro para controlar acciones de hardware en ciclos de tiempo específicos, de forma que podría ser implementado en procesos más largos, puntuales y complejos

Martínez Cruz José Antonio

Existieron inconvenientes con los segmentos del display debido a la cantidad que ya se necesitaba para esta práctica, pero al contrario de eso se logró la correcta muestra del cronometro en cuestión de segundos y milisegundos, así como su correcta finalización cuando alcanzaba el número 59.9 en la que se reiniciaba de nuevo

Hernández Méndez Oliver Manuel

Esta práctica tuvo que ver más con los ciclos de tiempo, con el manejo de diversos contadores, segundos y milisegundos, así como el manejo de botones para diversas tareas, se trata de una implementación que nos permitió jugar con los ciclos de reloj y familiarizarnos aún más con el manejo del tiempo con el microcontrolador, cosa que es de suma importancia ya que en muchas ocasiones los ciclos y procesos temporales son necesarios para implementaciones, tanto simples, como más avanzadas.

Referencias

[1] M. Mazidi, J. G. Mazidi y R. McKinlay, "8051 Microcontroller and Embedded Systems: Using Assembly and C", 2nd ed., Upper Saddle River, NJ: Prentice Hall, 2005.