

A thick dark green vertical bar is positioned on the left side of the page. A green arrow-shaped banner points to the right from this bar, containing the date '3-10-2021'. In the bottom-left corner, there are several thin, curved lines in dark green and light gray, resembling stylized grass or abstract brushstrokes.

3-10-2021

Cronómetro de 60 segundos

Martínez Coronel Brayan Yosafat

3CM17 Introducción a los microcontroladores
FERNANDO AGUILAR SÁNCHEZ

Objetivo

Al término de la sesión, los integrantes del equipo contarán con la habilidad de realizar un cronómetro de 60 segundos.

Introducción teórica

Los contadores son uno de los circuitos digitales de mayor aplicación en la construcción de sistemas lógicos electrónicos. Éstos pueden ser sincrónicos o asincrónicos. Los contadores sincrónicos constituyen un caso particular de circuito secuencial sincrónico en donde los pulsos de entrada que se van a contar se inyectan a la entrada de reloj o de pulsos de todos los flip-flops que componen el contador, obligando a los flip-flops a cambiar de estado de forma simultánea con la sola diferencia en los tiempos de operación de los flip-flops individuales. El cambio de estado de cada uno de los flipflops será función de los valores lógicos presentes en las entradas de excitación (SR, JK o D) de cada uno de los flip-flops.

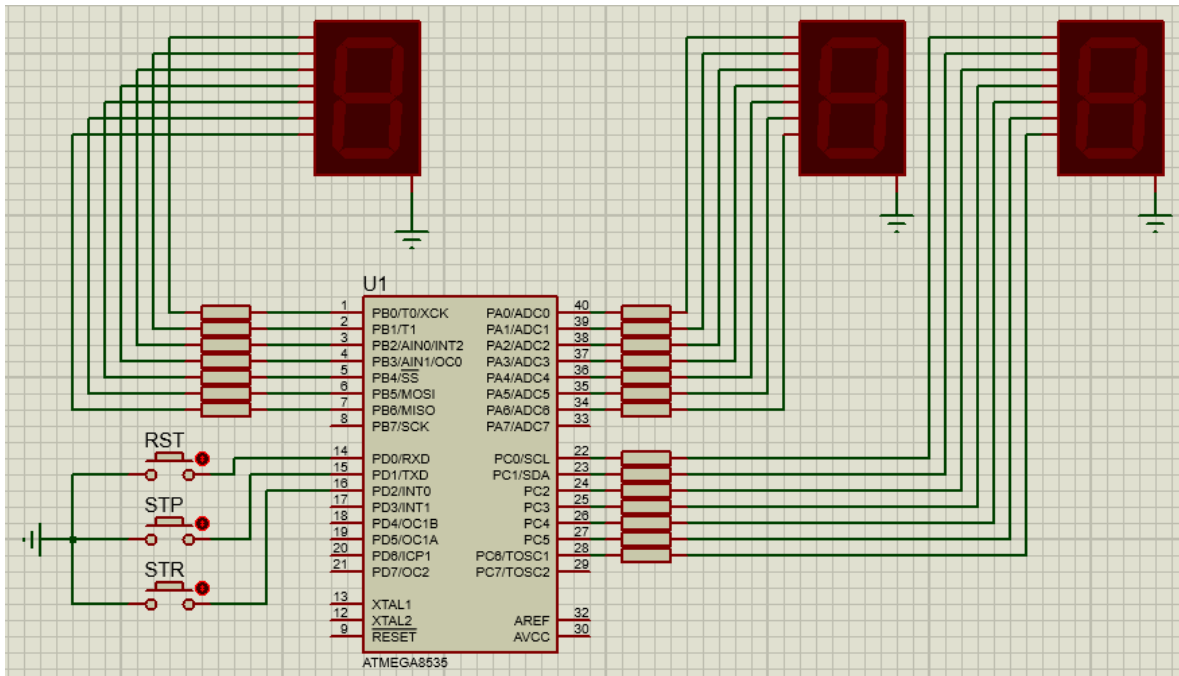
A diferencia de los contadores sincrónicos, en los contadores asincrónicos los pulsos de cuenta no se inyectan a todas las entradas de reloj de los flip-flops de forma simultánea, causando una mayor simplicidad en el circuito del contador, pero una mayor lentitud en su operación. Por supuesto, todo esto se puede ahorrar con el uso de un microcontrolador.

Materiales y Equipo empleado

- CodeVision AVR
- AVR Studio 4
- Microcontrolador ATmega 8535
- 3 Display cátodo común
- 3 botones
- 21 Resistores de 330 Ω a 1/4 W

Desarrollo experimental

Diseñe un programa que coloque tres displays para el conteo de 0 a 60.



Estructura del programa

This program was created by the CodeWizardAVR V3.45

Automatic Program Generator

© Copyright 1998-2021 Pavel Haiduc, HP InfoTech S.R.L.

<http://www.hpinfotech.ro>

Project :

Version :

Date : 28/09/2021

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

*****/

```
#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 según 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);
}
}

```

Observaciones y Conclusiones

Me agrada que podamos usar cosas de las practicas anteriores, no es tan trivial solucionar estos problemas. Es interesante ver cómo cada vez vamos haciendo cosas más complicadas, es como aplicar divide y vencerás para el proyecto final.

Bibliografía

Contador:

<http://www.usc.edu.co/files/LABORATORIOS/GUIAS/INGENIERIA/INGENIERIA%20EN%20SISTEMAS/CIRCUITOS%20DIGITALES%20%20I/CRONOMETRO%20DIGITAL.pdf>