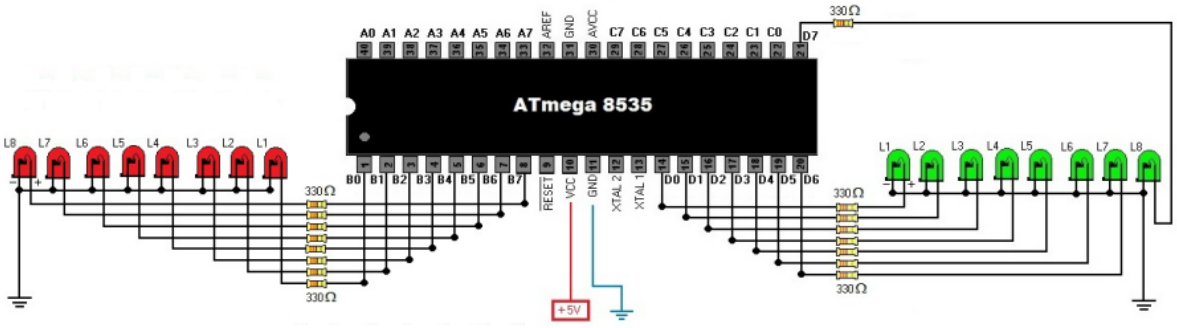
INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

SISTEMAS EN CHIP

PRACTICA No. 02

"Contador Ascendente/ Descendente"



ALUMNO: RENTERIA ARRIAGA JOSUE.

GRUPO: 6SCM1

PROFESOR: FERNANDO AGILAR SÁNCHEZ.

23/OCTUBRE/2022

I. OBJETIVO GENERAL:

Al término de la sesión, los integrantes del equipo contaran con la habilidad de realizar un contador descendente y ascendente utilizando funciones de retardo.

II. MATERIAL Y EQUIPO EMPLEADO

- ✓ CodeVision AVR.
- ✓ AVR Studio 4.
- ✓ Microcontrolador ATmega 8535.
- ✓ 16 LED's.
- ✓ 16 Resistores de 330 Ω a ¼ W.

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 un contador Ascendente y Descendente, codificación 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 salida de un contador Ascendente y Descendente. La práctica solo hará un contador Ascendente y Descendente que se vera reflejado en unos Leds que se prenderán y apagarán. 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 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 Leds para los puertos de salida.

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

Un contador ascendente/descendente (up/down) es aquel capaz de procesar en cualquier dirección a lo largo de una cierta secuencia. Cada una de las condiciones para las entradas J y K de cada flip-flop produce una basculación en el punto apropiado de la secuencia del contador.

La ascendencia de una persona la forman sus padres, abuelos, bisabuelos, etc., es decir, sus antepasados, ancestros o ascendentes. La descendencia, en cambio, hace referencia a los hijos, nietos, bisnietos, etc., es decir, sus descendientes. Se llama enlace ascendente a la señal que llega desde la estación emisora hasta el satélite de comunicaciones. El enlace descendente es el enviado desde el satélite hacia la zona de cobertura sobre la superficie terrestre.

El bloque funcional Contador ascendente (CTU) cuenta adelante desde el valor actual hasta el valor prefijado al producirse un flanco positivo en la entrada de contaje adelante (CU). Si el valor actual (VA) es mayor o igual al valor prefijado (PV), se activa el bit del contador.

Un contador asíncrono binario descendente es un contador en el cual, los biestables que lo conforman no comparten la señal de reloj, realizando sólo el contaje en sistema binario y unidireccional descendente. en la que la salida Q de cada biestable se conecta a la entrada T del siguiente biestable. El orden descendente es aquel que corresponde a los números que siguen una secuencia de mayor a menor.

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 contador binario ascendente en el Puerto B (0-255), y un contador descendente en el Puerto D (255-0). Use un retardo de un segundo para visualizar la información.

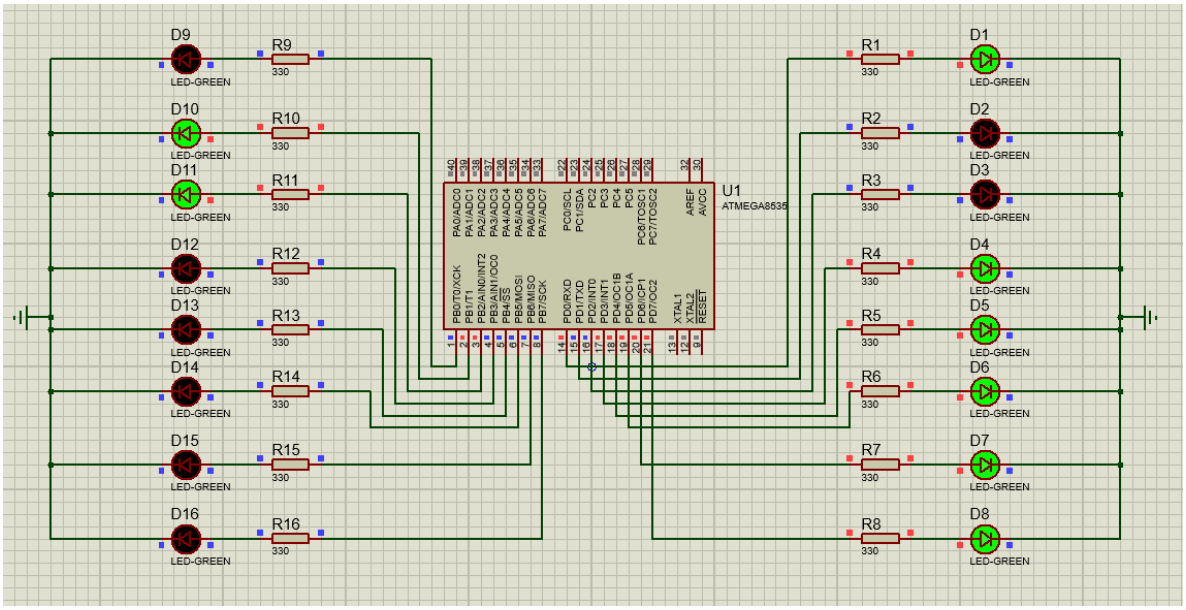


Fig.1_*Circuito Armado y Simulado en Proteus.*

CIRCUITO EN FISICO ARMADO.

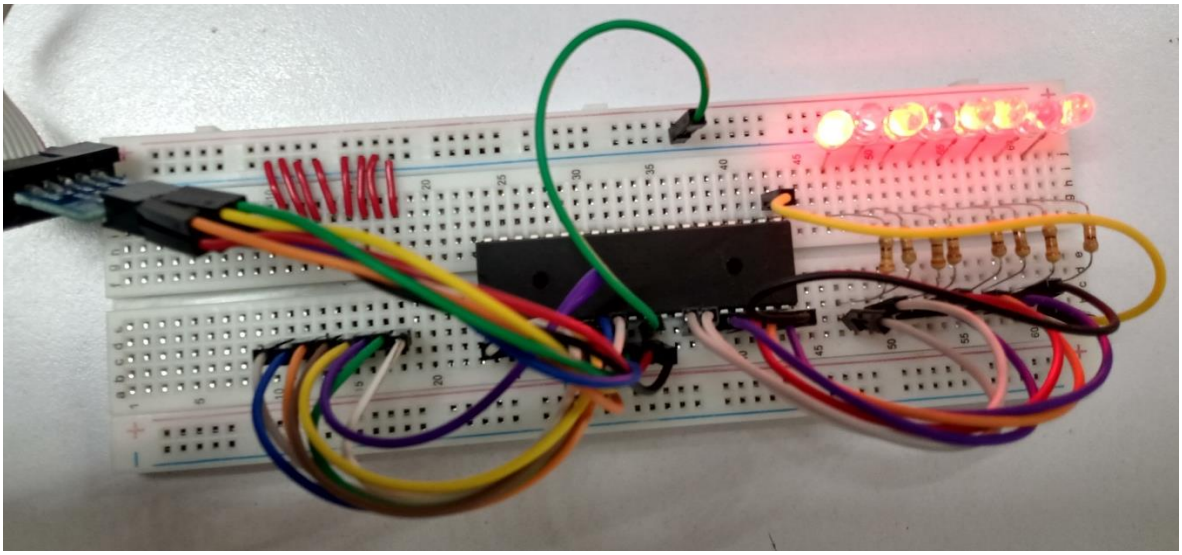


Fig.2_*Circuito Armado en Físico (Contador Descendente).*

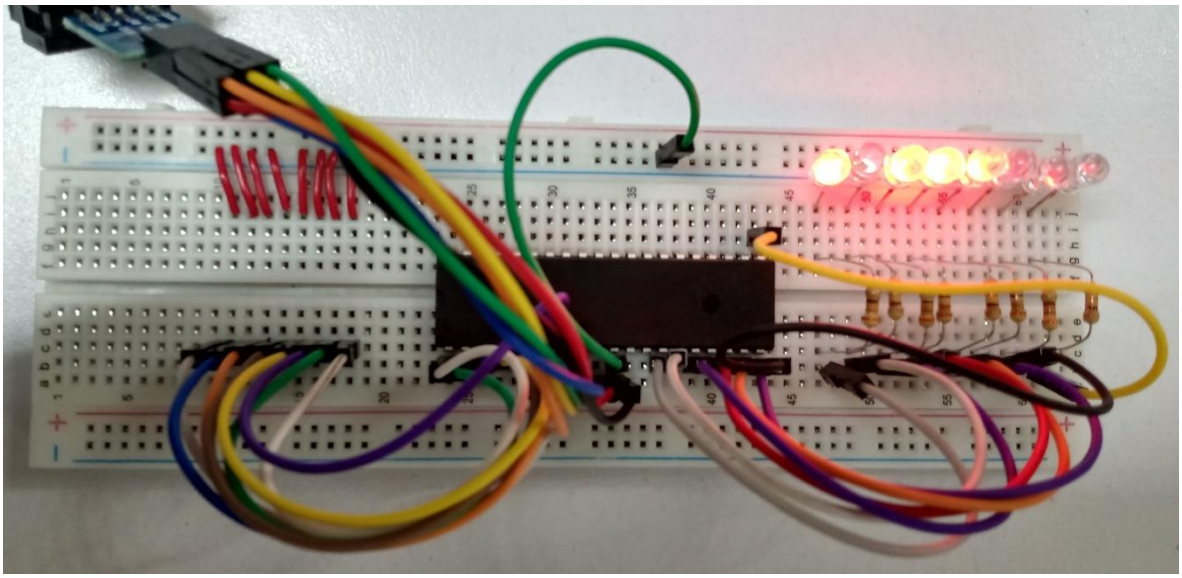


Fig.3_*Circuito Armado en Físico (Contador Ascendente).*

V. CÓDIGO

```
// I/O Registers definitions
#include <mega8535.h>
#include <delay.h>

// Declare your global variables here

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=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) |
(0<<PORTC1) | (0<<PORTC0);

// Port D initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
DDRD=(1<<DDD7) | (1<<DDD6) | (1<<DDD5) | (1<<DDD4) | (1<<DDD3) | (1<<DDD2) | (1<<DDD1) |
(1<<DDD0);
// State: Bit7=1 Bit6=1 Bit5=1 Bit4=1 Bit3=1 Bit2=1 Bit1=1 Bit0=1
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)
      {
        // Place your code here
        PORTB++;
        PORTD--;
        delay_ms(500);
      }
}
```

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. Esta practica no fue tan difícil, pero, es muy importante para poder familiarizarnos con el entorno de programación y como este funciona. Durante la practica si se tuvieron muchos problemas, mas que nada porque no conocía la forma de trabajar con estos elementos físicos.

Tras la investigación y explicación de la práctica previa me quedo un poco más claro como poder implementar y programar el ATMega8535. También conocí el concepto de resistencias pull-up y como utilizarlas en nuestro entorno de programación. En el caso de esta práctica dividí en dos los circuitos para que fuera mas fácil, uno para descendente y otro para descendente.

Para concluir esta práctica podemos decir que fue una experiencia interesante ya que nunca había armado circuitos en físico, solo en simulaciones y es muy diferente. Gracias a la anterior practica ya fue más fácil de hacerlo.

VII. BIBLIOGRAFÍA

- ✓ CodeVisionAVR. (2018). CodeVisionAVR. 20/Septiembre/2021, de CodeVisionAVR Sitio web: http://www.hpinfotech.ro/cvavr-download.html

- ✓ Descubre Arduino. (2012). ¿Como funciona un contador ascendente y descendente? 20/Septiembre/2021,Sitio web: https://respuestasrapidas.com.mx/como-funciona-un-contador-ascendente-y-descendente/