INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO.

INTRODUCCIÓN A LOS MICROCONTROLADORES.

REPORTE PRÁCTICA 2.

PROFESOR: AGUILAR SANCHEZ FERNANDO

Grupo: 3CM6

Alumno: Rojas Alvarado Luis Enrique

Práctica 2: Contador ascendente/descendente

Introducción teórica

Un contador es un circuito en el que sus salidas siguen una secuencia fija que cuando acaba vuelve a empezar, o circuitos que reciben sus datos en forma serial ordenados en distintos intervalos de tiempo. Los pulsos de entrada pueden ser pulsos de reloj u originarse en una fuente externa y pueden ocurrir a intervalos de tiempo fijos o aleatorios. El número de salidas limita el máximo número que se puede contar.

Al aplicar un pulso de reloj se observarán cambios en las salidas. Se da inicio a la cuenta que comienza en el número cero, 0000b, (todos los leds apagados) y termina en 15, 1111b (todos los leds encendidos). Del mismo modo hemos colocado al lado de las salidas luminosas (diodos led), un display 7 segmentos para comparar la salida numérica con la salida luminosa.

Vamos a pasar a representar una tabla de visualización del código binario. En esta tabla solo se ven representados unos y ceros, seguidamente pondremos para su entendimiento, el equivalente decimal. [1]

A Mayor peso	B	C	D menor peso	Nº DECIMAL
0	0	0	0	**0**
0	0	0	1	**1**
0	0	1	0	**2**
0	0	1	1	**3**
0	1	0	0	**4**
0	1	0	1	**5**
0	1	1	0	**6**
0	1	1	1	**7**
1	0	0	0	**8**
1	0	0	1	**9**
1	0	1	0	**10 (A)**
1	0	1	1	**11 (B)**
1	1	0	0	**12 (C)**
1	1	0	1	**13 (D)**
1	1	1	0	**14 (E)**
1	1	1	1	**15 (F)**

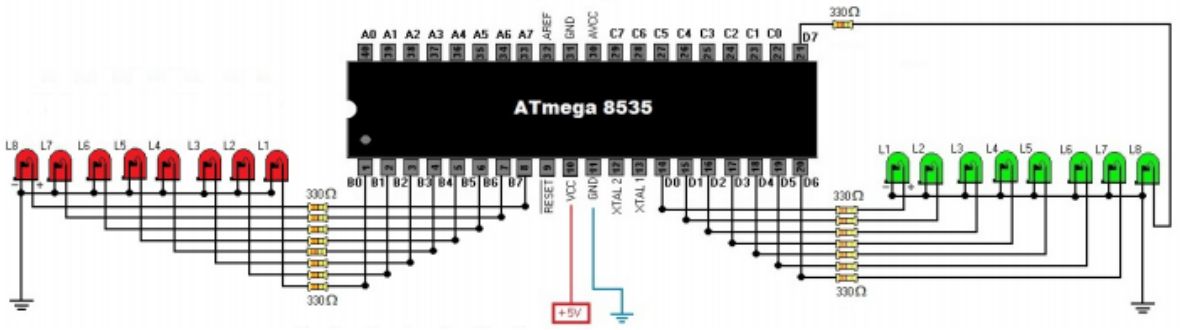
Los pesos de la tabla indican el bit más importante ó significativo de una secuencia.

-Clasificación de los contadores:

- Según la forma en que conmutan los biestables, podemos hablar de contadores síncronos (todos los biestables conmutan a la vez, con una señal de reloj común) o asíncronos (el reloj no es común y los biestables conmutan uno tras otro).
- Según el sentido de la cuenta, se distinguen en ascendentes, descendentes y UP-DOWN (ascendentes o descendentes según la señal de control).
- Según la cantidad de números que pueden contar, se puede hablar de contadores binarios de n bits (cuentan todos los números posibles de n bits, desde 0 hasta 2^n-1), contadores BCD (cuentan del 0 al 9) y contadores Módulo N (cuentan desde el 0 hasta el N-cuarto. El número máximo de estados por los que pasa un contador se denomina módulo del contador. Este número viene determinado por la expresión 2^n donde n indica el número de bits del contador. Ejemplo, un contador de módulo 4pasa por 4 estados, y contaría del 0 al 3.

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.



Estructura del programa

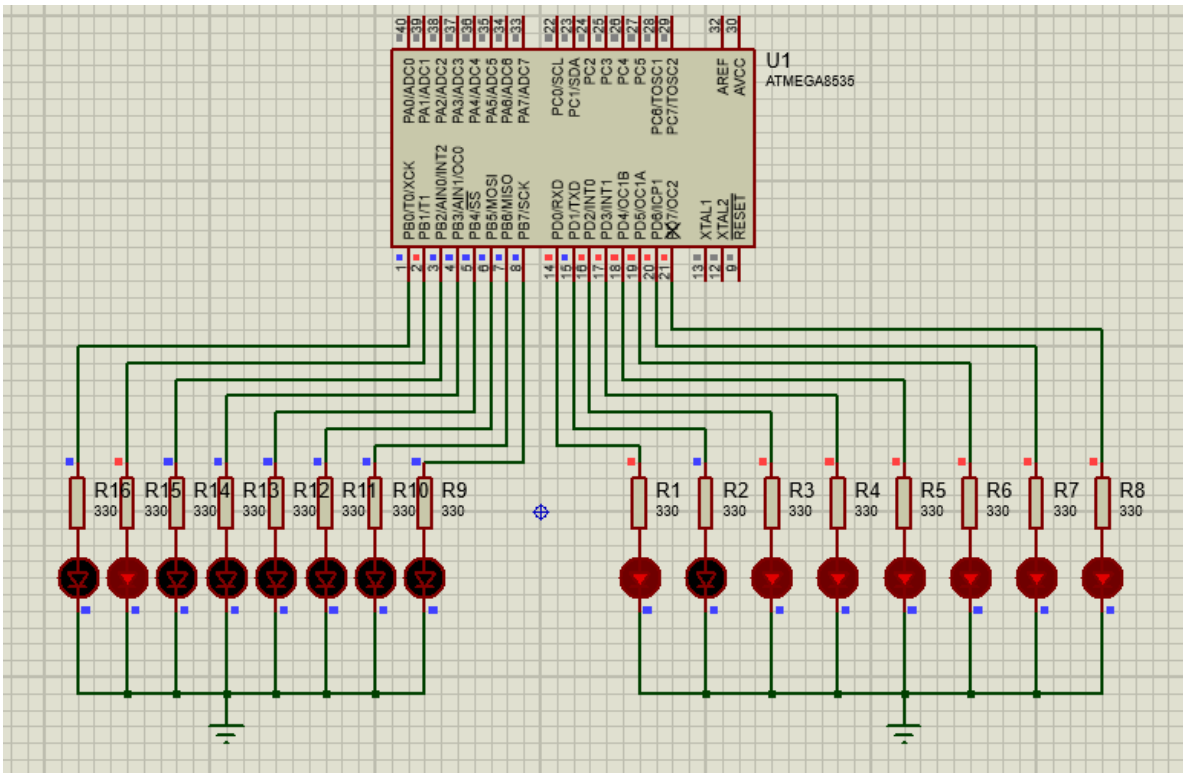
```
/*****************************************************
This program was created by the
CodeWizardAVR V2.60 Evaluation
Automatic Program Generator
© Copyright 1998-2012 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 12/10/2020
Author  :
Company :
Comments:
```

```
14.
15.
16. Chip type               : ATmega8535
17. Program type           : Application
18. AVR Core Clock frequency: 1.000000 MHz
19. Memory model           : Small
20. External RAM size      : 0
21. Data Stack size        : 128
22. *****************************************************/
23.
24. #include <mega8535.h>
25. #include <delay.h>
26. // Declare your global variables here
27.
28. void main(void)
29. {
30. // Declare your local variables here
31.
32. // Input/Output Ports initialization
33. // Port A initialization
34. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
35. DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) | (0<<D
    DA1) | (0<<DDA0);
36. // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
37. PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (0<<PO
    RTA2) | (0<<PORTA1) | (0<<PORTA0);
38.
39. // Port B initialization
40. // Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=O
    ut
41. DDRB=(1<<DDB7) | (1<<DDB6) | (1<<DDB5) | (1<<DDB4) | (1<<DDB3) | (1<<DDB2) | (1<<D
    DB1) | (1<<DDB0);
42. // State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
43. PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PO
    RTB2) | (0<<PORTB1) | (0<<PORTB0);
44.
45. // Port C initialization
46. // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
47. DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<D
    DC1) | (0<<DDC0);
48. // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
49. PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PO
    RTC2) | (0<<PORTC1) | (0<<PORTC0);
50.
51. // Port D initialization
52. // Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=O
    ut
53. DDRD=(1<<DDD7) | (1<<DDD6) | (1<<DDD5) | (1<<DDD4) | (1<<DDD3) | (1<<DDD2) | (1<<D
    DD1) | (1<<DDD0);
54. // State: Bit7=1 Bit6=1 Bit5=1 Bit4=1 Bit3=1 Bit2=1 Bit1=1 Bit0=1
55. PORTD=(1<<PORTD7) | (1<<PORTD6) | (1<<PORTD5) | (1<<PORTD4) | (1<<PORTD3) | (1<<PO
    RTD2) | (1<<PORTD1) | (1<<PORTD0);
56.
57. // Timer/Counter 0 initialization
58. // Clock source: System Clock
59. // Clock value: Timer 0 Stopped
60. // Mode: Normal top=0xFF
61. // OC0 output: Disconnected
62. TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) |
    (0<<CS00);
63. TCNT0=0x00;
```

```
64. OCR0=0x00;
65. 
66. // Timer/Counter 1 initialization
67. // Clock source: System Clock
68. // Clock value: Timer1 Stopped
69. // Mode: Normal top=0xFFFF
70. // OC1A output: Disconnected
71. // OC1B output: Disconnected
72. // Noise Canceler: Off
73. // Input Capture on Falling Edge
74. // Timer1 Overflow Interrupt: Off
75. // Input Capture Interrupt: Off
76. // Compare A Match Interrupt: Off
77. // Compare B Match Interrupt: Off
78. TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WG
    M10);
79. TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) |
     (0<<CS10);
80. TCNT1H=0x00;
81. TCNT1L=0x00;
82. ICR1H=0x00;
83. ICR1L=0x00;
84. OCR1AH=0x00;
85. OCR1AL=0x00;
86. OCR1BH=0x00;
87. OCR1BL=0x00;
88. 
89. // Timer/Counter 2 initialization
90. // Clock source: System Clock
91. // Clock value: Timer2 Stopped
92. // Mode: Normal top=0xFF
93. // OC2 output: Disconnected
94. ASSR=0<<AS2;
95. TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<CS21) |
    (0<<CS20);
96. TCNT2=0x00;
97. OCR2=0x00;
98. 
99. // Timer(s)/Counter(s) Interrupt(s) initialization
100.       TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) |
    (0<<TOIE1) | (0<<OCIE0) | (0<<TOIE0);
101. 
102.       // External Interrupt(s) initialization
103.       // INT0: Off
104.       // INT1: Off
105.       // INT2: Off
106.       MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
107.       MCUCSR=(0<<ISC2);
108. 
109.       // USART initialization
110.       // USART disabled
111.       UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<U
    CSZ2) | (0<<RXB8) | (0<<TXB8);
112. 
113.       // Analog Comparator initialization
114.       // Analog Comparator: Off
115.       ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) |
    (0<<ACIS1) | (0<<ACIS0);
116.       SFIOR=(0<<ACME);
117. 
118.       // ADC initialization
```

```
119.       // ADC disabled
120.       ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<AD
    PS2) | (0<<ADPS1) | (0<<ADPS0);
121.
122.       // SPI initialization
123.       // SPI disabled
124.       SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA)
    | (0<<SPR1) | (0<<SPR0);
125.
126.       // TWI initialization
127.       // TWI disabled
128.       TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);
129.
130.       while (1)
131.             {
132.             // Place your code here
133.             PORTB++;      //incrementa el puerto en una unidad
134.             PORTD--;      //decrementa el puerto en una unidad
135.             delay_ms(500);//retardo que se genera para visualizar
136.             }
137.       }
```

Simulación.



Conclusiones

En esta práctica pudimos hacer conocimiento de un contador binario, el cual sería ascendente y descendente por los diferentes puertos establecidos, también supimos configurar cada uno de estos puertos e incrementando en una unidad cada valor dentro de los puertos de salida (B o D), para que se muestre en los LEDS y

sepamos que cada uno está haciendo lo que le configuramos, deberemos jugar con la librería delay.h y la función delay, ya sea en mili segundos o en micro segundos, según el intervalo de tiempo que queramos que se muestre el contador en los diferentes puertos de salida.

Bibliografía

[1] Electronicasi, “contador binario”, 2013. Accedido: 12/10/2020,

http://www.electronicasi.com/wpcontent/uploads/2013/06/contador-binario.pdf

[2] Microcontroladores, “Contador binario”, 18/12/2013, Accedido: 12/10/2020,

http://microcontroladores-ipn-2013.blogspot.com/2013/12/contador-binario.html