



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO



Introducción a los microcontroladores 3CM16

Practica 19

Interrupciones externas

Equipo 7

Integrantes:

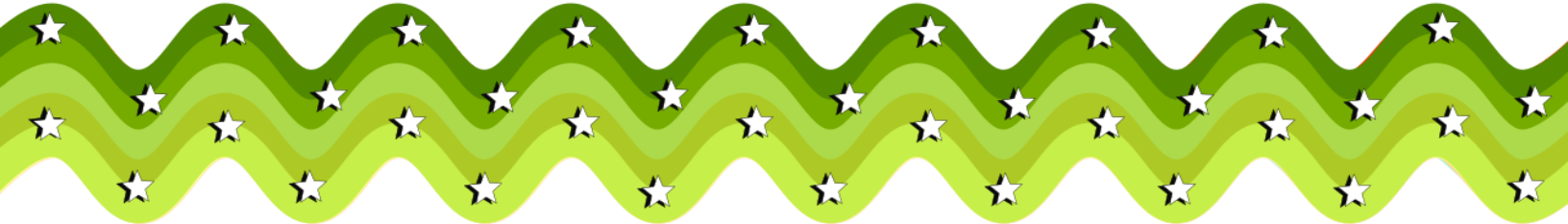
- ♥ Bocanegra Heziquio Yestlanezi
- ♥ Dominguez Durán Alan Axel
- ♥ Hernandez Mendez Oliver Manuel
- ♥ Martinez Cruz José Antonio

Profesor: Aguilar Sanchez Fernando



índice

Índice de imágenes	3
Objetivo	3
Materiales y Equipo empleado.....	3
Introducción.....	4
Interrupciones externas	4
Aspectos clave.....	4
Interrupciones	5
Desarrollo	6
Configuración de pines y registros.....	6
Inicialización del display y contadores	6
Rutinas de interrupción	6
Actualización del display	6
Bucle principal	6
Simulación – proteus	7
Circuito	8
Codificación.....	8
Conclusiones.....	13
Bocanegra Heziquio Yestlanezi.....	13
Dominguez Durán Alan Axel.....	13
Hernandez Mendez Oliver Manuel.....	13
Martinez Cruz José Antonio	13
Referencias	14





índice de imágenes

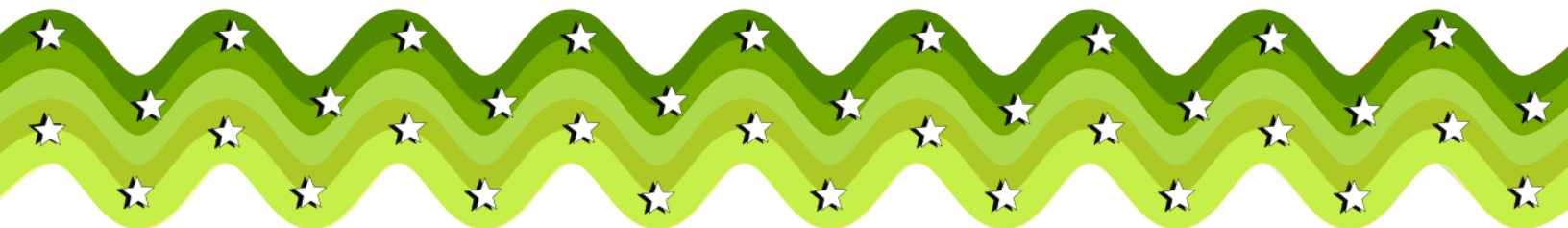
Imagen 1 simulación en proteus - funcionando.....	7
Imagen 2 circuito armado	8

Objetivo

Al término de la sesión, los integrantes del equipo contarán con la habilidad para manejar las interrupciones del Microcontrolador.

Materiales y Equipo empleado

- ▽ CodeVision AVR
- ▽ AVR Studio 4
- ▽ Microcontrolador ATmega 8535
- ▽ 1 Display cátodo común
- ▽ 2 Push Button
- ▽ 1 Resistor de $1K\Omega$






Introducción

Interrupciones externas

Las interrupciones externas en los microcontroladores son mecanismos que permiten que un evento externo, como un cambio en el nivel de voltaje de un pin específico, active una interrupción y detenga temporalmente la ejecución del programa principal. Esto permite que el microcontrolador atienda rápidamente eventos importantes y realice tareas específicas en respuesta a ellos [1].

Aspectos clave

- ▽ Configuración de las interrupciones externas: Para utilizar las interrupciones externas, se deben configurar adecuadamente los registros y las opciones de configuración del microcontrolador. Esto incluye habilitar las interrupciones externas, definir el modo de disparo (flanco de subida, flanco de bajada o nivel lógico) y configurar las resistencias de pull-up o pull-down, según sea necesario [2].
 - ▽ Pines de interrupción externa: Los microcontroladores tienen pines específicos que se pueden configurar como pines de interrupción externa. Estos pines suelen tener una funcionalidad dedicada para gestionar las interrupciones y están conectados a circuitos internos que detectan los eventos externos deseados [2].
 - ▽ Rutinas de interrupción: Cuando se activa una interrupción externa, el microcontrolador detiene temporalmente la ejecución del programa principal y salta a una rutina de interrupción específica. Esta rutina de interrupción contiene el código que se ejecutará en respuesta al evento externo. Es importante tener en cuenta que las rutinas de interrupción deben ser rápidas y eficientes, ya que interrumpen el flujo normal del programa [2].
 - ▽ Prioridades de interrupción: Algunos microcontroladores permiten establecer prioridades para las interrupciones externas. Esto significa que, en caso de que ocurran varias interrupciones al mismo tiempo, se puede establecer un orden de prioridad para determinar qué rutina de interrupción se ejecuta primero [2].
 - ▽ Habilitar y deshabilitar interrupciones: En algunos casos, es posible habilitar o deshabilitar las interrupciones externas durante la ejecución del programa. Esto permite controlar el momento en el que se pueden generar o atender interrupciones, según las necesidades del sistema [2].
- 

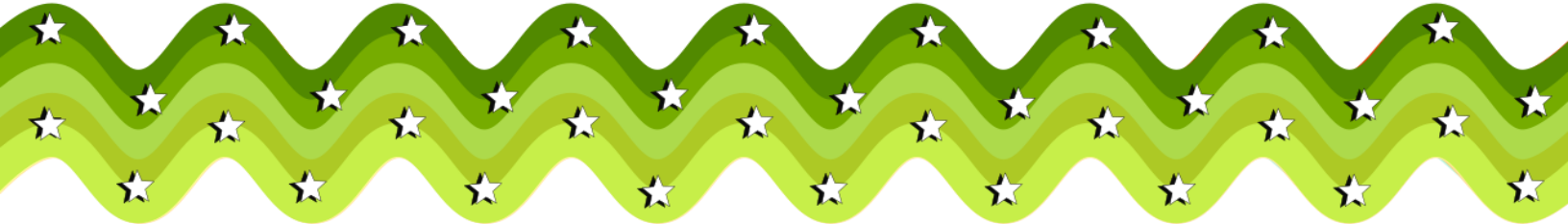


Interrupciones

en un microcontrolador son mecanismos que permiten al microcontrolador detener temporalmente la ejecución de un programa principal para atender una tarea prioritaria. Estas interrupciones pueden ser generadas por eventos externos, como un cambio en el nivel de voltaje en un pin específico, o eventos internos, como desbordamiento de temporizadores o finalización de operaciones de conversión analógico-digital [3].

Para diseñar un programa que incremente el conteo del display cuando haya un flanco de subida en INT0 y decremente el valor del display cuando haya un nivel lógico de 0 en INT1, se pueden utilizar las interrupciones externas del microcontrolador [4].

El microcontrolador debe estar configurado para habilitar y manejar las interrupciones externas en los pines INT0 e INT1. Cuando se detecte un flanco de subida en INT0, se ejecutará una rutina de interrupción que incrementará el conteo del display. Por otro lado, cuando se detecte un nivel lógico de 0 en INT1, se ejecutará otra rutina de interrupción que decrementará el valor del display [4].





Desarrollo

1. Diseñe un programa que cuando haya un flanco de subida en INT0 se incremente el conteo del display que podrá contar de 0 a 9, y que cuando haya un nivel lógico de 0 en INT1 se decremente el valor del display.

Configuración de pines y registros

Primero, es necesario configurar los pines del microcontrolador ATmega8535 y los registros correspondientes para habilitar las interrupciones externas en los pines INT0 e INT1, así como configurar los pines del display y los push buttons.

Inicialización del display y contadores

Se deben inicializar el display y los contadores necesarios para llevar el conteo. El display debe estar configurado en modo cátodo común y se pueden utilizar variables para almacenar el valor actual del contador.

Rutinas de interrupción

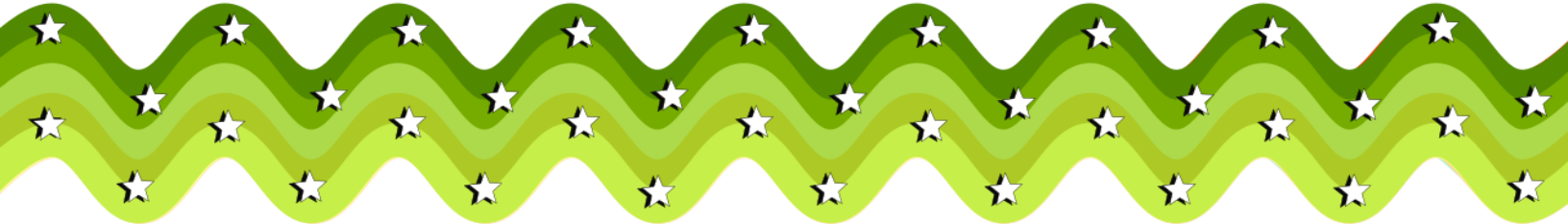
Se deben definir las rutinas de interrupción para manejar los eventos en los pines INT0 e INT1. Cuando se detecte un flanco de subida en INT0, se ejecutará la rutina de incremento del contador, y cuando haya un nivel lógico de 0 en INT1, se ejecutará la rutina de decremento del contador.

Actualización del display

Después de cada incremento o decremento del contador, se debe actualizar el display para mostrar el valor actual del contador. Esto implica enviar los dígitos adecuados al display mediante técnicas de multiplexación o mediante una interfaz de control adecuada.

Bucle principal

Finalmente, se debe crear un bucle principal en el programa para que siga ejecutándose y permita la detección continua de eventos en los pines INT0 e INT1.



Simulación - proteus

A continuación, en la imagen 1 uno podemos observar la simulación del circuito armada en proteus.

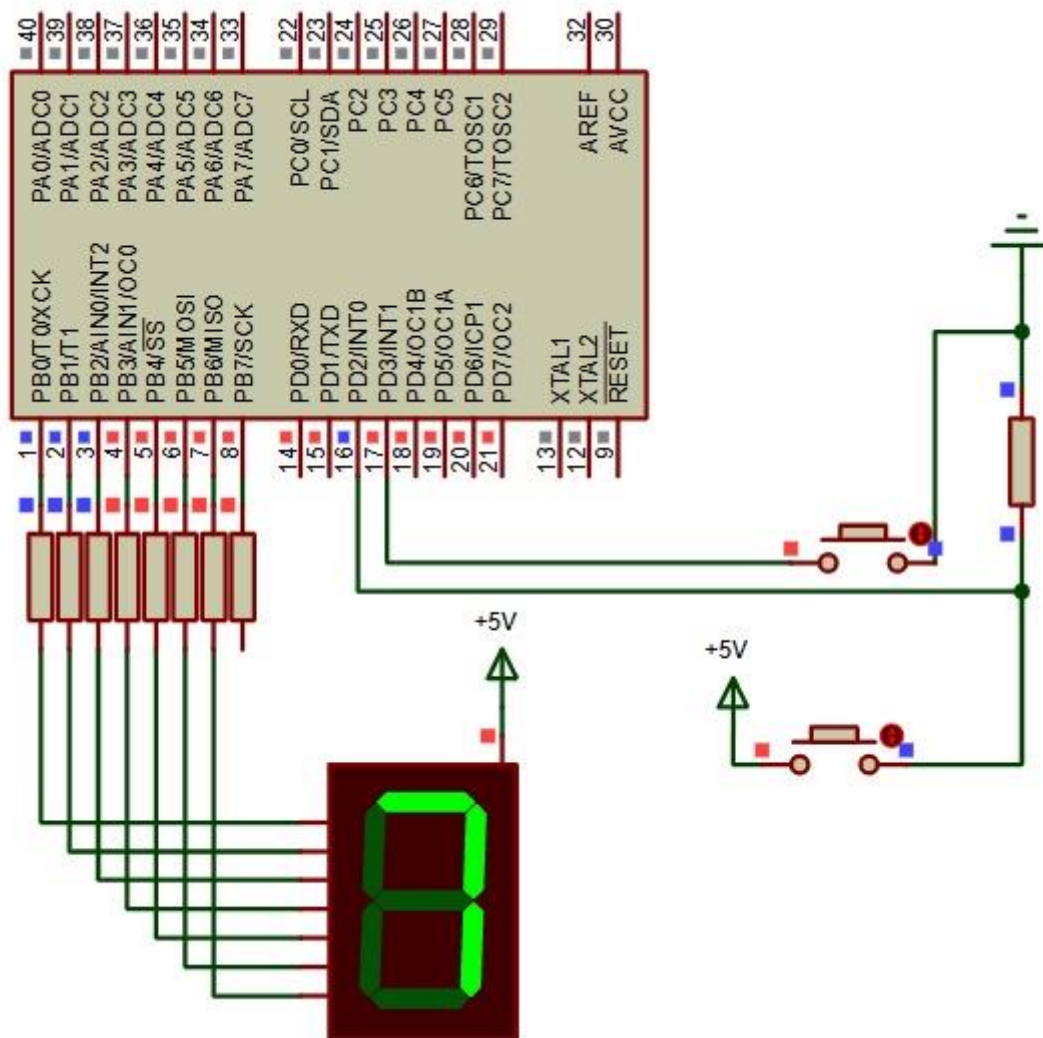


Imagen 1 simulación en proteus - funcionando

Aunque en la imagen no podemos observar la configuración de los pines que utilizamos, nos podemos guiar y darnos cuenta mediante el código que se mostrara mas adelante.

Circuito

A continuación, en la imagen 2 podemos observar el circuito funcionando de manera correcta, el cual fue realizado con base en la simulación de proteus.

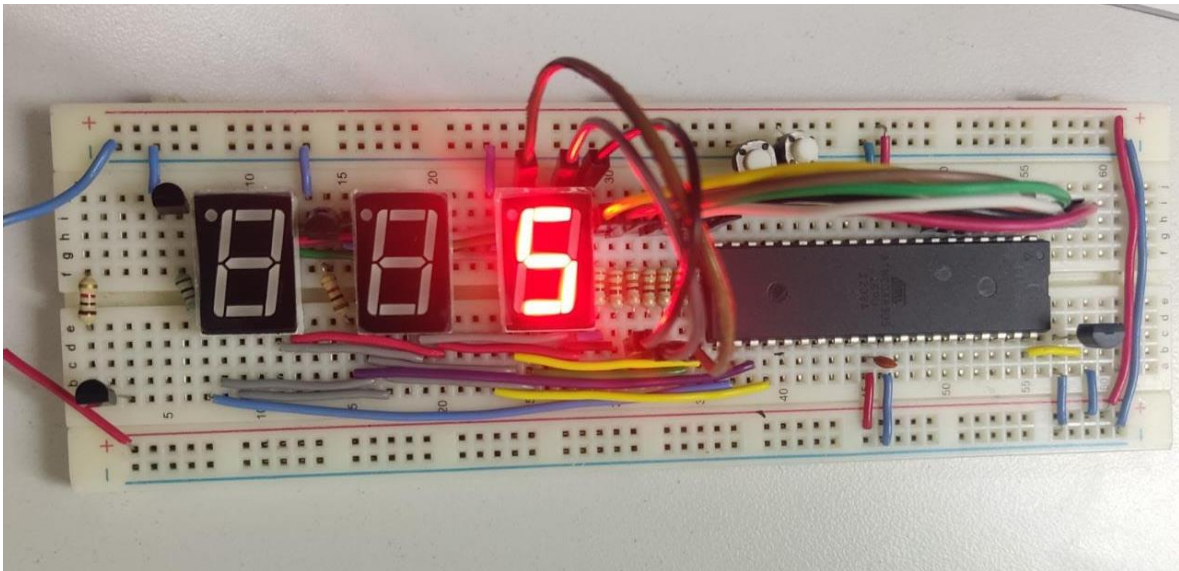


Imagen 2 circuito armado

Codificación

```

/*****
This program was created by the CodeWizardAVR V3.46a
Automatic Program Generator
♦ Copyright 1998-2021 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro

Project :
Version :
Date    : 24/05/2023
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

***** /

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

```
char indice = 0;
```

```
// External Interrupt 0 service routine
```

```
interrupt [EXT_INT0] void ext_int0_isr(void)
```

```
{
```

```
    indice++;
```

```
    if (indice == 10) indice = 0;
```

```
}
```

```
// External Interrupt 1 service routine
```

```
interrupt [EXT_INT1] void ext_int1_isr(void)
```

```
{
```

```
    if (indice == 0) indice = 9;
```

```
    else indice--;
```

```
}
```

```
// Declare your global variables here
```

```
const char mem[10] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F};
```

```
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=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: On
// INT0 Mode: Rising Edge
// INT1: On
// INT1 Mode: Low level
// INT2: Off
GICR|=(1<<INT1) | (1<<INT0) | (0<<INT2);
MCUCR=(0<<ISC11) | (0<<ISC10) | (1<<ISC01) | (1<<ISC00);
MCUCSR=(0<<ISC2);
GIFR=(1<<INTF1) | (1<<INTF0) | (0<<INTF2);

// 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);  
  
// Globally enable interrupts  
#asm("sei")  
  
while (1)  
{  
    PORTB = ~mem[indice];  
}  
}
```



Conclusiones

Bocanegra Heziquio Yestlanezi

el diseño de este programa requerirá habilitar y configurar las interrupciones externas en los pines correspondientes del microcontrolador, y definir las rutinas de interrupción que incrementen o decrementen el valor del display cuando se produzcan los eventos deseados. Esto permitirá contar de 0 a 9 en el display, incrementando o decrementando el valor según las interrupciones generadas en los pines INT0 e INT1.

Dominguez Durán Alan Axel

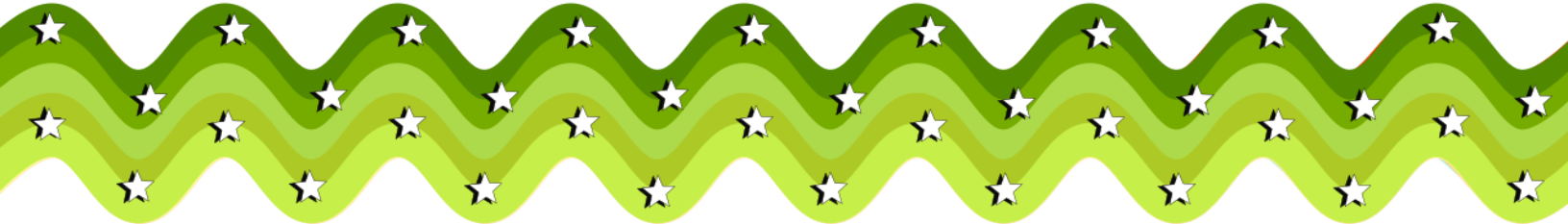
Similar a las primeras prácticas, se implementó un contador con un display de 8 segmentos y dos push buttons, sin embargo, este contador puede regresar a estados de reloj anteriores de tal forma que podemos contar de forma ascendente y descendente; sin embargo por la propia frecuencia del microcontrolador y la forma en que operan los botones, en ocasiones al retroceder, el contador se salta estados de reloj; sin embargo esa es su forma de operar. Con esta implementación de contador nos será posible adecuar los displays de acuerdo con nuestras necesidades.

Hernandez Mendez Oliver Manuel

Al igual que en las prácticas iniciales, se desarrolló un sistema de cuenta utilizando un display de 8 segmentos y dos pulsadores. No obstante, en esta ocasión se agregó la capacidad de retroceder a estados de reloj previos, lo que permite contar tanto en sentido ascendente como descendente. Sin embargo, debido a la frecuencia del microcontrolador y al funcionamiento de los pulsadores, en ocasiones, al retroceder, el contador salta algunos estados de reloj. Aunque esto puede parecer una limitación, es parte del funcionamiento normal del sistema. Esta implementación del contador nos brinda la flexibilidad de adaptar los displays según nuestras necesidades específicas.

Martinez Cruz José Antonio

Esta práctica resulto ser muy sencilla debido a que el circuito en si era muy pequeño a comparar de los anteriores y también el código desarrollado fue más reducido en comparación. Sin embargo, lo principal en esta práctica era comprender de manera clara como es que estas interrupciones nos pueden ayudar para la práctica siguiente y afortunadamente se logró ese objetivo.





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.
 - [2] M. Morris Mano and Charles R. Kime, "Logic and Computer Design Fundamentals," 4th ed. Upper Saddle River, NJ, USA: Pearson, 2008, pp. 542-544.
 - [3] T. L. Floyd, "Digital Fundamentals," 11th ed. Boston, MA, USA: Pearson, 2015, pp. 518-520.
 - [4] 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.
- 