INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

PRÁCTICA 4

CONTADOR DE 0 A 9

SISTEMAS EN CHIP

6CM3

PROFESOR:

AGUILAR SÁNCHEZ FERNANDO

INTEGRANTES:

MIGUEL VÁSQUEZ MARIANO JOSUÉ

FRANCO OLVERA ODER

GONZALEZ RAMOS ANGEL DAVID

Fecha de Elaboración

06/04/2024

INTRODUCCIÓN

Los sistemas en chip (SoC) representan una convergencia de múltiples elementos de hardware y software en un solo dispositivo, ofreciendo soluciones compactas y eficientes para una variedad de aplicaciones. En el ámbito de la electrónica digital, la implementación de contadores es fundamental para diversas aplicaciones, desde sistemas de temporización hasta control de procesos y sistemas de medición.

En este contexto, el microcontrolador ATmega8535, perteneciente a la familia AVR de Microchip, destaca como una opción versátil y potente para la implementación de contadores de 0 a 9 en sistemas embebidos. Este microcontrolador ofrece una combinación única de potencia de procesamiento, recursos de entrada/salida (E/S) y flexibilidad de programación, adecuado para aplicaciones en el campo de los sistemas en chip.

Un contador de 0 a 9 es un tipo específico de contador que cuenta en secuencia del 0 al 9 y luego regresa al 0 para iniciar nuevamente. Este contador se usa en aplicaciones como sistemas de visualización numérica, temporizadores, control de acceso y otros dispositivos que requieren una secuencia de conteo limitada. La implementación de este contador utilizando el ATmega8535 aprovecha las capacidades del microcontrolador para realizar esta tarea de manera eficiente y confiable.

En esta práctica exploraremos los fundamentos teóricos y prácticos de la implementación de un contador de 0 a 9 utilizando el ATmega8535. Comenzaremos describiendo los principios de funcionamiento de los contadores digitales y cómo se aplican en este contexto específico. Luego, nos adentraremos en la arquitectura y características clave del ATmega8535, destacando cómo se pueden utilizar para implementar este tipo de contador. Finalmente, presentaremos ejemplos de código y circuitos para ilustrar la programación y el uso práctico de un contador de 0 a 9 con este microcontrolador en el contexto de sistemas en chip.

CÓDIGO

```
/*****************************************************
This program was created by the CodeWizardAVR V4.01
Automatic Program Generator
© Copyright 1998-2024 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro

Project :
Version :
Date    : 19/02/2024
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>

#define boton PIND.0


// Declare your global variables here

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

unsigned char var1;


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
```

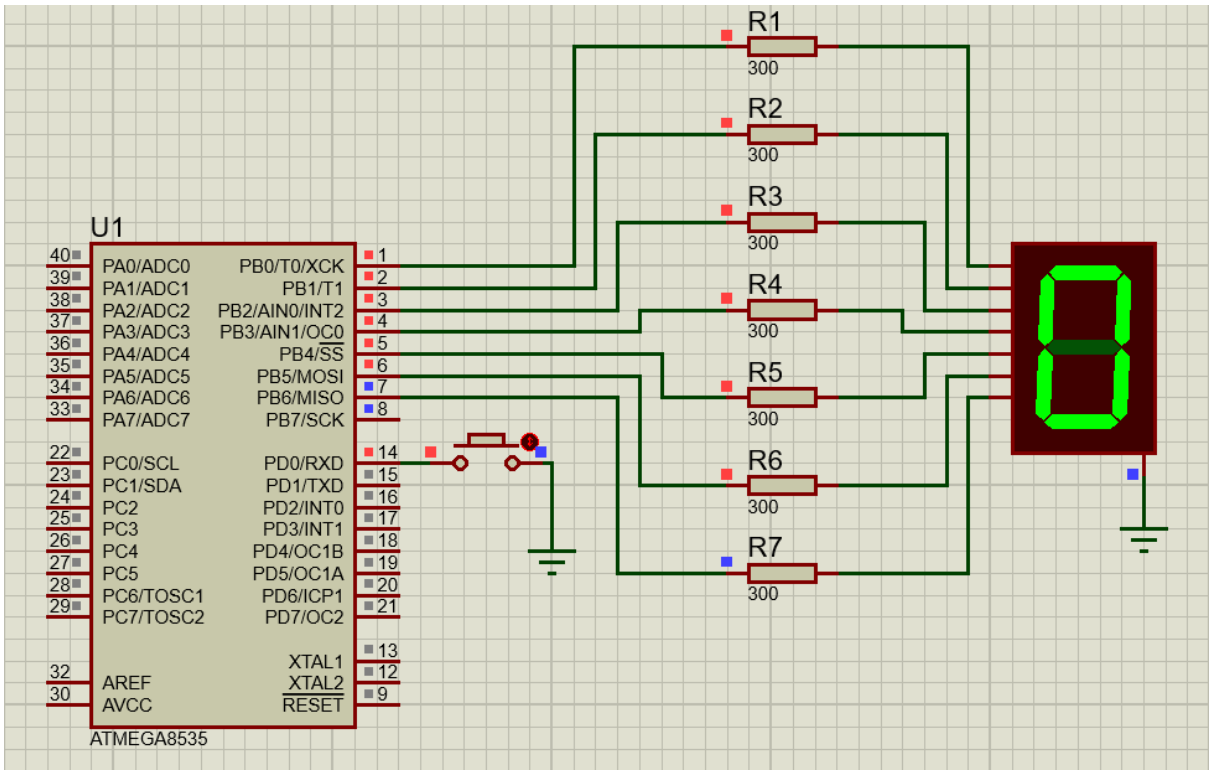
```
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out
Bit0=Out

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
```

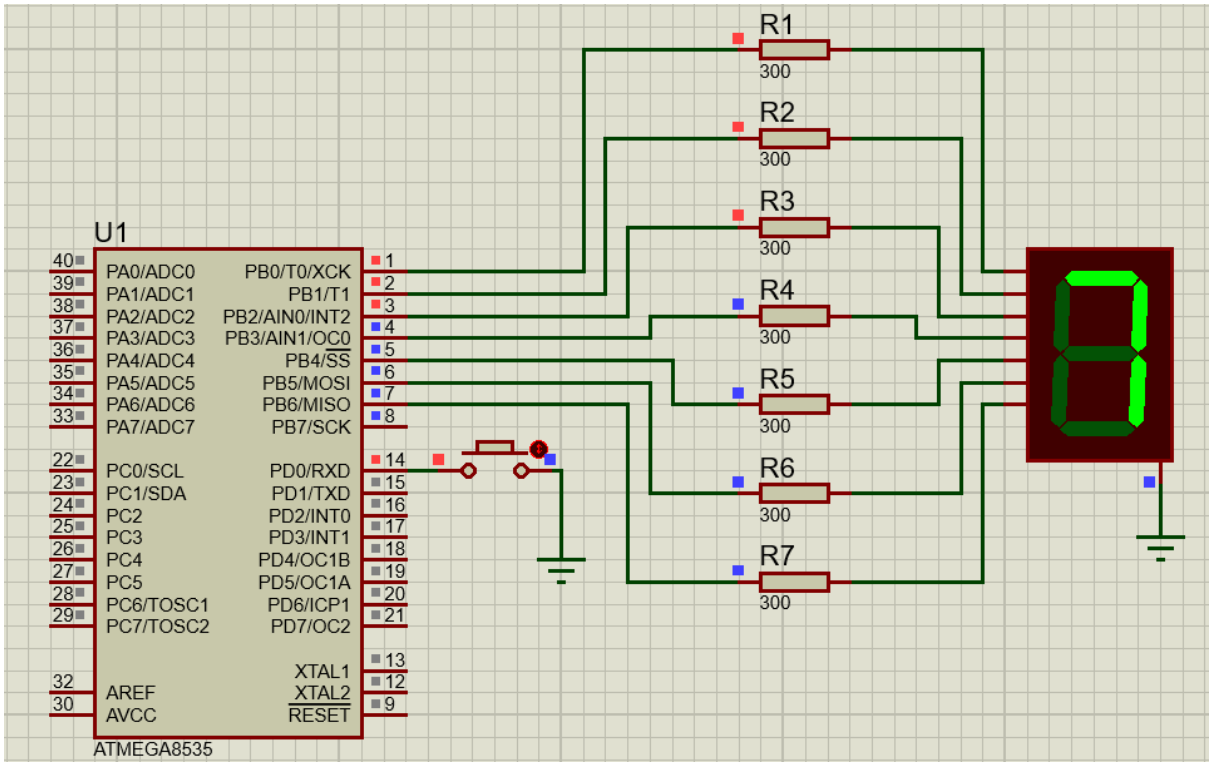
```
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Disconnected
// OC1B output: Disconnected
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: 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: 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);
```

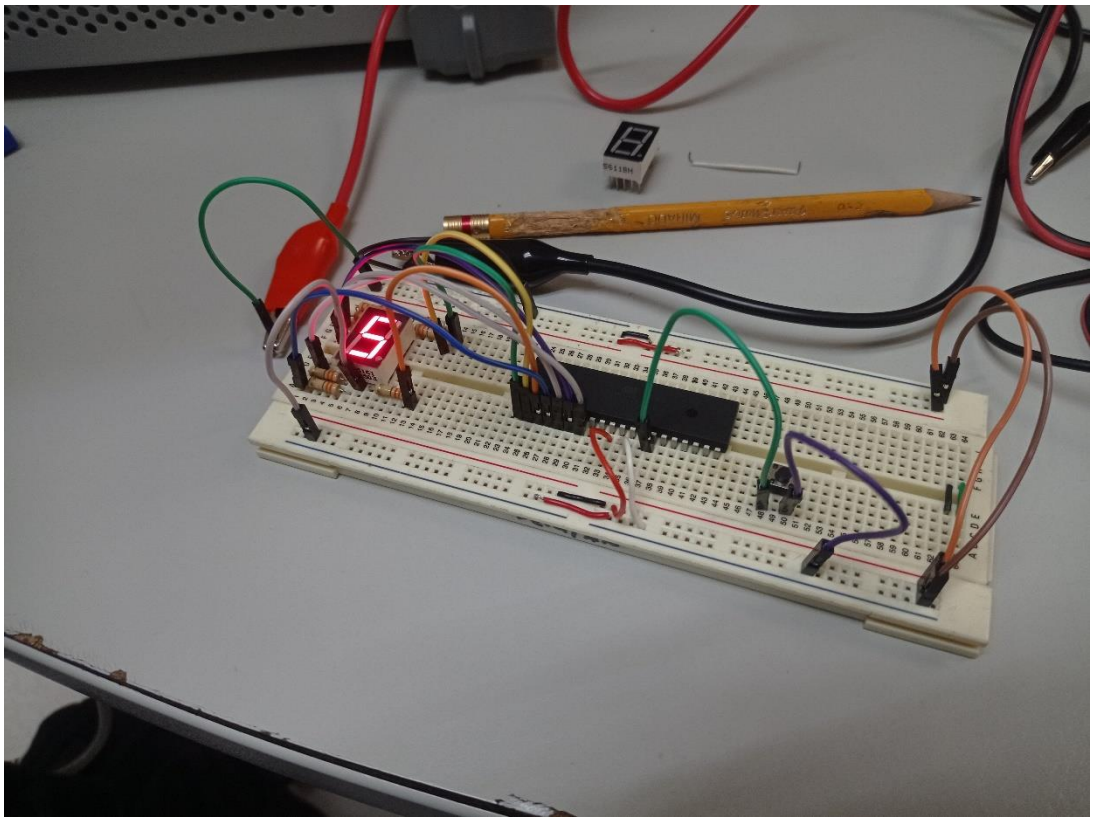
```
// TWI initialization
// TWI disabled
TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);


while (1)
      {
        if (boton == 0)
            var1++;


        if (var1 == 10)
            var1 = 0;


        PORTB = mem[var1];
      }
}
```

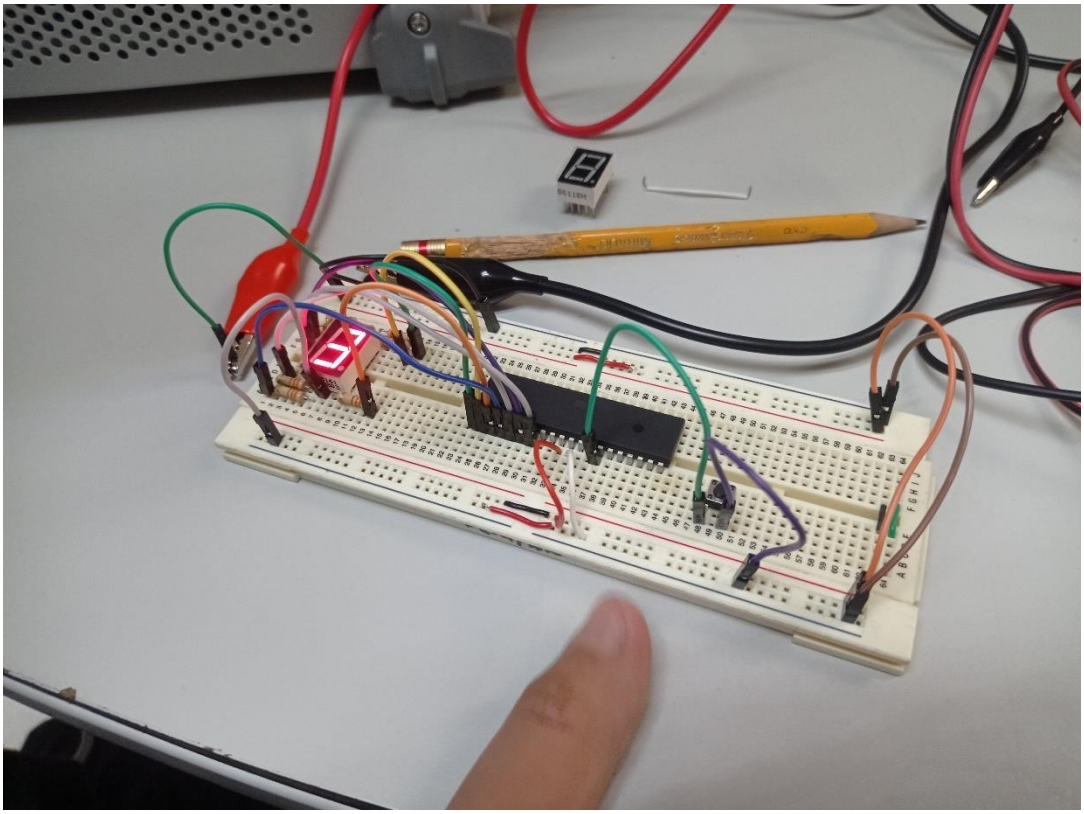
SIMULACIÓN EN PROTEUS

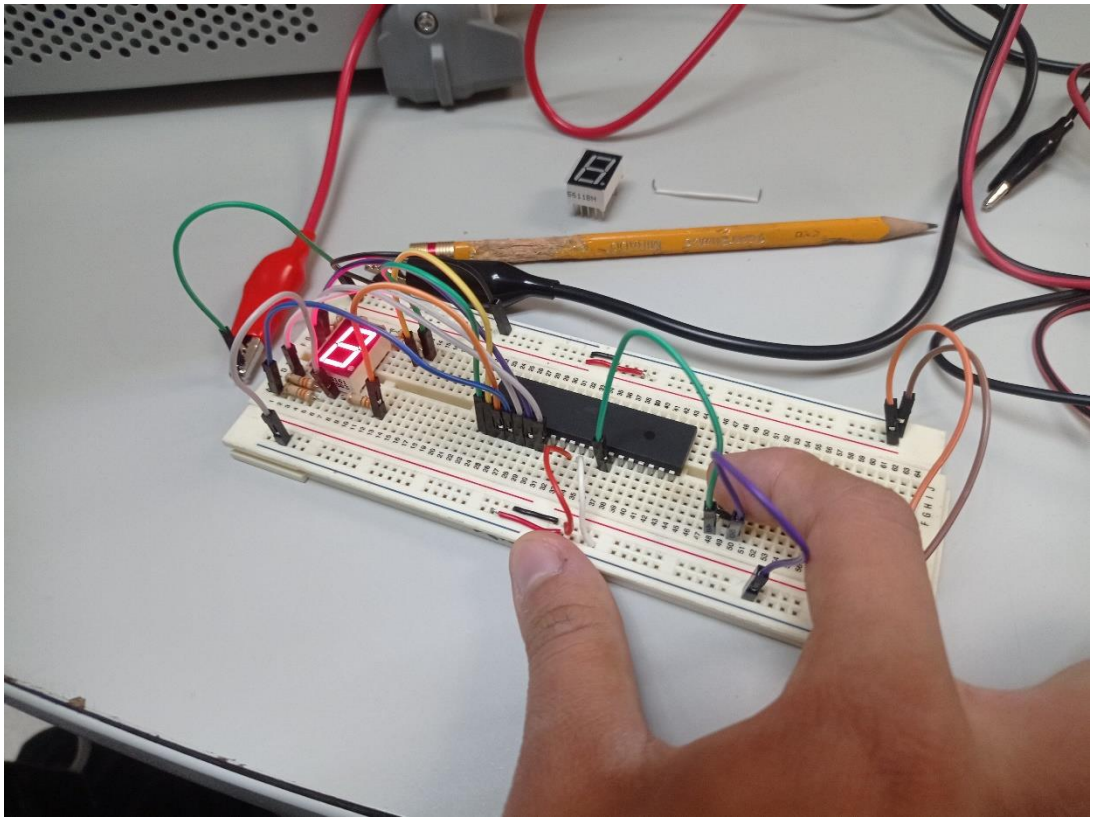



R1
300
R2
300
R3
300
R4
300
R5
300
R6
300
R7
300
U1
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PC0/SCL
PC1/SDA
PC2
PC3
PC4
PC5
PC6/TOSC1
PC7/TOSC2
AREF
AVCC
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
XTAL1
XTAL2
RESET
ATMEGA8535

FOTOS







CONCLUSIONES INDIVIDUALES

Franco Olvera Demian Oder

Esta práctica permitió implementar un programa sencillo que implica usar una técnica que seguramente se utilizará en prácticas futuras. Se trata de la manipulación directa de una variable secuencialmente mediante la interacción del usuario con el circuito, que es una forma elegante de decir que se mueve una variable con un Push Button. Como el flujo de la variable está ligado al ciclo de reloj interno del ATMEGA, el resultado no es el esperado; parece que el movimiento de la variable es errático y no se tiene mucho control sobre él. Esto ocurre porque las instrucciones y secuencias transcurren al cabo de microsegundos y es indistinguible para el ojo humano. En prácticas subsecuentes, se aprovecharán diversas técnicas de programación para corregir esta situación.

Gonzalez Ramos Angel David

La práctica de implementar un contador de 0 a 9 en un microcontrolador como el ATmega8535 proporciona una comprensión sólida de la manipulación de datos y la lógica de control. Esto implica la configuración adecuada de los registros del microcontrolador para contar secuencialmente de 0 a 9, así como la gestión de desbordamientos para reiniciar el contador. Además, esta práctica permite explorar técnicas de visualización, como la conexión a un display de siete segmentos, lo que facilita la representación numérica de forma clara. Al final, desarrollar un contador de 0 a 9 en un microcontrolador fortalece los fundamentos de la programación y la electrónica digital y da una base sólida para proyectos más complejos que requieren control secuencial y visualización de datos.

Miguel Vásquez Mariano Josué

Con el desarrollo de esta practica pudimos implementar un contador del 0 al 9 reutilizando código de la practica 2 para el proceso de conteo asi como de la practica 3 para la salida en el display. Sin embargo, ahora como entrada en vez de utilizar un dip switch, utilizamos un push button. El problema que se presentó a la hora de desarrollar la practica es sobre la acción del conteo a la hora de presionar el botón, ya que como el atmega ejecuta en todo momento cada instrucción a una velocidad de 1MHz, si simplemente se llama a la instrucción de contar mientras el botón este presionado, el microcontrolador hará a el conteo a una velocidad aproximadamente

de 1 conteo por cada milisegundo hasta que el botón deje de presionarse. Esto hace que el contador sea imposible de controlar para una persona y lo ideal sería que no se volviera a incrementar si no hasta que suelte y presione nuevamente el botón. Este es un problema que será resuelta en la posterior práctica.

REFERENCIAS

- Arpita, M. D. (2012, 14 enero). CONTADOR DIGITAL DE 0 a 9 CON 7490. Mikitronic Tutoriales Electrónica. Recuperado 28 de marzo de 2024, de https://mikitronic.blogspot.com/2012/01/contador-digital-de-0-9.html