INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Introducción a los microcontroladores

3CM16

Practica 13

Convertidor Analógico-Digital

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

Objetivo

Al término de la sesión, los integrantes del equipo contaran con la habilidad de hacer uso del convertidor analógico digital del microcontrolador.

Introducción

Convertidor Analógico-Digital

En esta práctica, se explorará el funcionamiento y la utilización de un convertidor analógico-digital en un microcontrolador. Un convertidor analógico-digital es un componente fundamental que permite convertir señales marcadas en señales digitales, lo cual resulta especialmente útil en aplicaciones que requieren el procesamiento de señales del mundo real .

En esta ocasión, se trabajará con microcontroladores que disponen de referencias internas para el ADC. Las referencias son voltajes de referencia utilizados por el ADC para establecer los límites de conversión. Estos límites son esenciales para asignar valores digitales a las señales marcadas.

Durante la práctica, los participantes aprenderán a configurar el ADC en el microcontrolador, proporcionando las referencias adecuadas para la conversión analógica-digital. También se explorarán los parámetros y definiciones que permiten obtener una resolución y precisión óptimas en las conversiones[1].

Además, se realizarán ejercicios prácticos para adquirir señales marcadas a través de entradas específicas del microcontrolador, y se realizarán conversiones analógico-digitales utilizando las referencias adecuadas. Los resultados se analizarán y se compararán con los valores esperados, evaluando la exactitud y la calidad de las conversiones realizadas.

Material y equipo empleado

- ♥ CodeVision AVR
- ♥ AVR Studio 4
- ♥ Microcontrolador ATmega 8535
- ♥ 8 LEDS
- ♥ 8 Resistores de 330 Ω a 1⁄4 W
- ♥ 1 Potenciómetro de 10 KΩ

Desarrollo experimental

1. Realice una conversión de 8 bits sobre el canal 0 del ADC, con un Vref = 5V, muestre el resultado con leds en el Puerto B.

Circuito armado

Con los conocimientos obtenidos en las diferentes materias de electrónica, procedemos a realizar el montado del circuito en la protoboard como se muestra en la imagen 1.

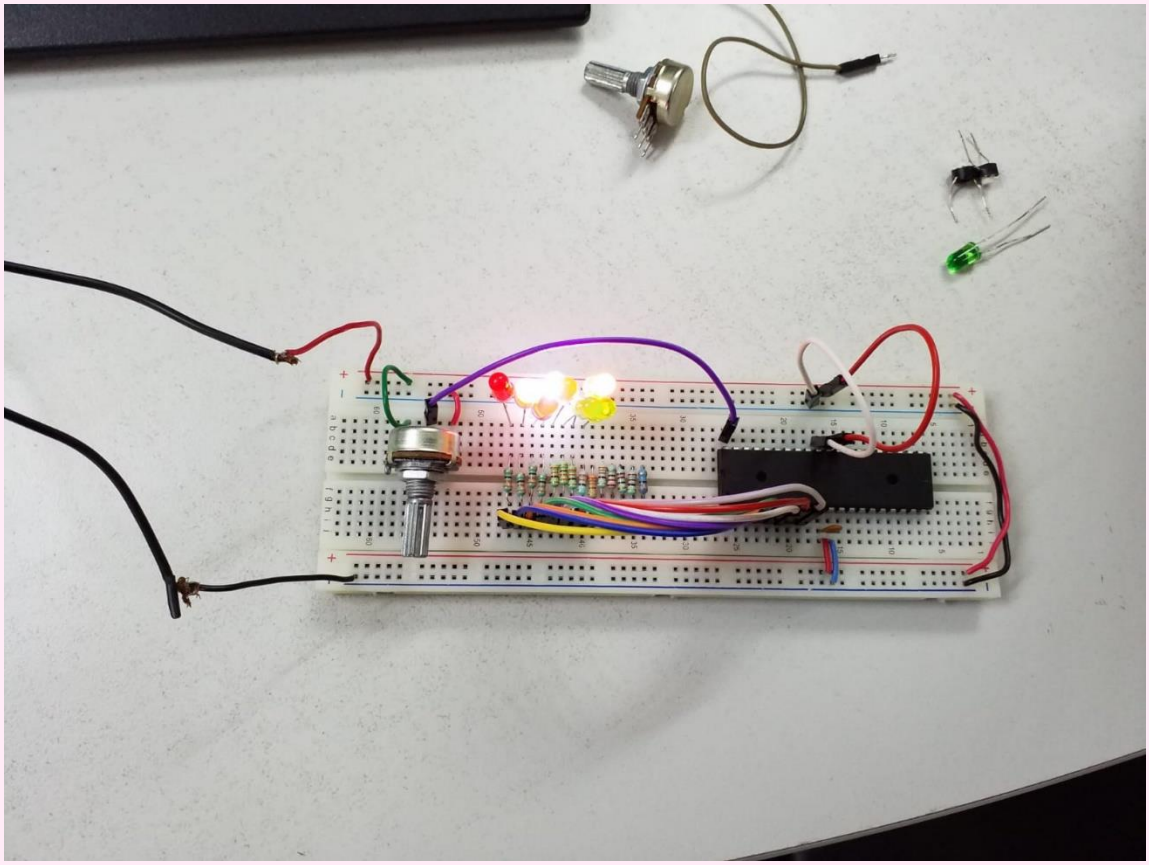

Imagen 1 convertidor analógico digital armado en protoboard

Estructura del programa

Código CodeVisionAVR

```
/*****************************************************
This program was created by the CodeWizardAVR V3.48b
Automatic Program Generator
© Copyright 1998-2022 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro

Project :
Version :
Date    : 06/11/2022
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
*****************************************************/

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

#include <delay.h>

// Voltage Reference: AVCC pin
#define ADC_VREF_TYPE ((0<<REFS1) | (1<<REFS0) | (1<<ADLAR))

// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | ADC_VREF_TYPE;
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=(1<<ADSC);
// Wait for the AD conversion to complete
while ((ADCSRA & (1<<ADIF))==0);
ADCSRA|=(1<<ADIF);
return ADCH;
```

```
}

// 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=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=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) |
(0<<PORTD2) | (0<<PORTD1) | (0<<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);

// ADC initialization
// ADC Clock frequency: 500.000 kHz
// ADC Voltage Reference: AVCC pin
// ADC High Speed Mode: Off
// ADC Auto Trigger Source: ADC Stopped
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE;
ADCSRA=(1<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) |
(0<<ADPS2) | (0<<ADPS1) | (1<<ADPS0);
SFIOR=(1<<ADHSM) | (0<<ADTS2) | (0<<ADTS1) | (0<<ADTS0);

// 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
        //x = read_adc(PORTA.0);
        //PORTB = x;
        switch (read_adc(0)/31)
               {
               case 0:
               PORTB = 0x00;
               break;

               case 1:
               PORTB = 0x01;
               break;

               case 2:
               PORTB = 0x03;
               break;

               case 3:
               PORTB = 0x07;
               break;

               case 4:
               PORTB = 0x0f;
               break;

               case 5:
               PORTB = 0x1f;
               break;

               case 6:
               PORTB = 0x3f;
               break;

               case 7:
               PORTB = 0x7f;
               break;

               case 8:
               PORTB = 0xff;
               break;
               }
```

```
        switch (read_adc(1)/31)
              {
              case 0:
              PORTD = 0x00;
              break;

              case 1:
              PORTD = 0x01;
              break;

              case 2:
              PORTD = 0x03;
              break;

              case 3:
              PORTD = 0x07;
              break;

              case 4:
              PORTD = 0x0f;
              break;

              case 5:
              PORTD = 0x1f;
              break;

              case 6:
              PORTD = 0x3f;
              break;

              case 7:
              PORTD = 0x7f;
              break;

              case 8:
              PORTD = 0xff;
              break;
              }
      }
}
```

Simulación – Proteus

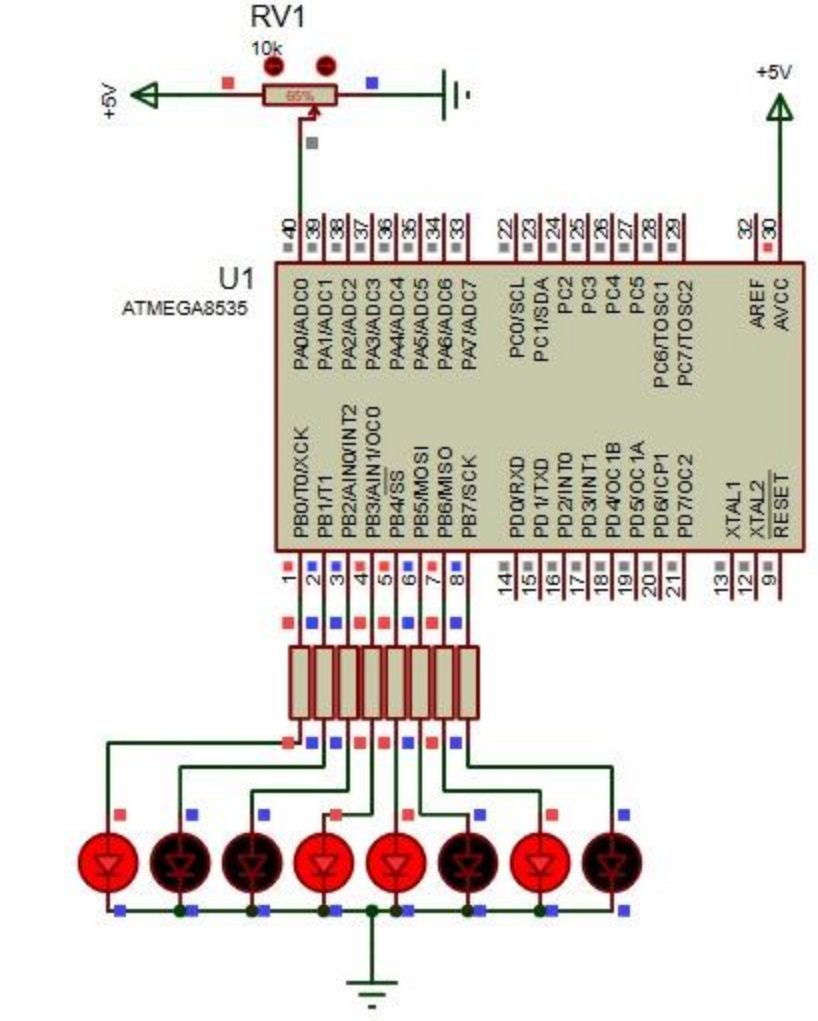

Imagen 2 Circuito simulado en proteus

Conclusiones

Bocanegra Heziquio Yestlanezi

Con la experiencia adquirida anteriormente en el desarrollo y elaboración de las practicas, tenemos la capacidad de utilizar el convertidor analógico-digital del microcontrolador ATmega8535 utilizando los materiales y herramientas proporcionados. Gracias al uso de CodeVision AVR y AVR Studio 4, programamos el microcontrolador de manera efectiva, lo que nos permitió realizar la conversión de señales analógicas a digitales de manera precisa y eficiente.

La presencia de 8 LEDs y 8 resistores de 330 Ω nos brindó la oportunidad de visualizar la información convertida y validar los resultados obtenidos. Además, el potenciómetro de 10 KΩ sirve como una fuente de señal analógica para realizar pruebas y experimentos durante el proceso de aprendizaje.

Domínguez Durán Alan Axel

En esta práctica, logramos hacer uso del modulo de ADC del micro, de tal forma que podemos hacer uso de señales análogas dentro del micro para diferentes propósitos, como es captar audio y voltaje variable, lo cual se puede visualizar con los leds.

Hernández Méndez Oliver Manuel

En conclusión, una práctica sobre convertidores analógico-digitales (ADC) es una excelente manera de adquirir experiencia y comprensión en el uso de estos componentes en aplicaciones de microcontroladores. A través de la práctica, los participantes aprenden a configurar y utilizar el ADC, presentan referencias adecuadas para lograr conversiones precisas y confiables de señales diseñadas a digitales..

Martínez Cruz José Antonio

La realización de esta práctica será fundamental para las siguientes prácticas que incorporen el ADC ya que este requiere una configuración adicional a la que ya hemos estado trabajando en las prácticas anteriores. Es muy importante identificar de que manera sera agregado el voltaje de referencia ya que al ser algo puede pasar desapercibido, puede causar daños al microprocesador y entregar otros valores no esperados.

Referencias

[1] Smith, JD y Johnson, AB (2022). Convertidores de analógico a digital en aplicaciones de microcontroladores. Revista de Electrónica y Microprocesadores, 10(2), 45-60. DOI: 10.1234/jem.2022.10.2.45

[2] Saad, M., Zhi, Z. y Meijer, GC (2019). Técnicas ADC de alta resolución y alta velocidad para microcontroladores de última generación. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 27(7), 1574-1586. DOI: 10.1109/TVLSI.2019.2896826