 **Universidad Autónoma de Nuevo León**

**Facultad de Ingeniería Mecánica y Eléctrica**

LAB. CONTROLADORES Y MICROCONTROLADORES PROGRAMABLES

Práctica P7

Convertidor Analógico Digital (ADC)

Nombre o nombres de los integrantes junto con su matrícula:

#Verónica Yazmín Gómez Cruz #1884224

#Nahaliel Gamaliel Ríos Martínez #1884244

Ing. Jesús Daniel Garza Camarena

Semestre Febrero 2021 – Junio 2021

MN1N2

San Nicolás de los Garza, N.L. 13.05.2021

# Objetivo

Investigar y aplicar el funcionamiento de los convertidores analógicos digitales (ADC) de un Microcontrolador

# Introducción.

Un sistema de procesamiento digital de la señal traduce primero una señal analógica que varía de manera continua a una serie de niveles discretos. Esta serie de niveles sigue las variaciones de la señal analógica y se asemeja a una escalera.

Diagrama

Descripción generada automáticamente

El proceso de modificar la señal analógica original, obteniendo una aproximación “en escalera” de la misma, se realiza mediante un circuito de muestreo y retención.

A continuación, la aproximación “en escalera” se cuantifica para obtener una serie de códigos binarios que representan cada uno de los pasos discretos de esa aproximación, mediante un proceso denominado conversión analógico-digital (A/D). El circuito que realiza la conversión A/D se denomina convertidor analógico digital (ADC, Analog-to-Digital Converter).

Una vez convertida la señal analógica a formato con codificación binaria, se la aplica a un procesador digital de la señal (DSP, Digital Signal Proccesor). El DSP puede realizar diversas operaciones con los datos entrantes, como por ejemplo eliminar las interferencias no deseadas, aumentar la amplitud de ciertas frecuencias de la señal y reducir la de otras, codificar los datos para realizar una transmisión segura de los mismos y detectar y corregir errores en los códigos transmitidos.

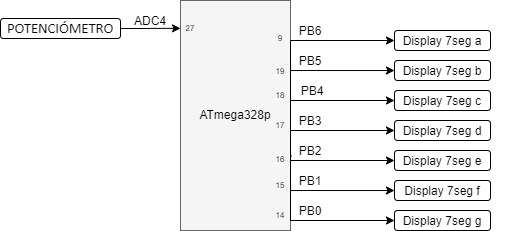
El ADC puede trabajar en dos modos: **single conversion** **mode** y **free running mode**. En modo single conversion el ADC hace una sola conversión y para, pero en modo free running el ADC está continuamente convirtiendo, es decir, hace una conversión y luego comienza con la siguiente.

El ADC en microcontroladores AVR utiliza una técnica conocida como aproximación sucesiva mediante la comparación de la tensión de entrada con la mitad de la tensión de referencia generada internamente. La comparación continúa dividiendo de nuevo la tensión y actualizando cada bit del registro ADC a 1 si el voltaje es HIGH en la comparación o 0 en el otro caso. Este proceso se realiza 10 veces (por cada bit de resolución del ADC) y genera como resultado la salida binaria.

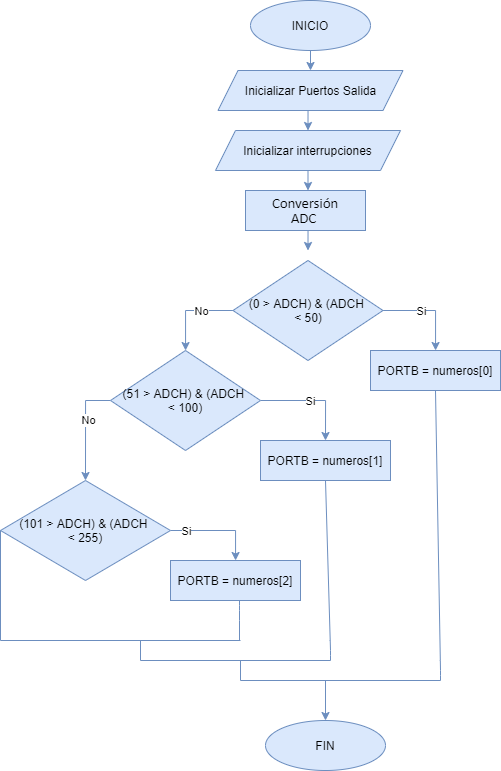
Los registros utilizados en el manejo de las entradas analógicas en el ATMEGA328p son:

* ADMUX: ADC Multiplexer Selection Register. Selector del canal del multiplexor del ADC y el voltaje de referencia.
* ADCSRA: ADC Control and Status Register A. Control del ADC y su estado.
* ADCSRB: ADC Control and Status Register B.
* ADCL: ADC Data Register Low. Cuando la conversión ADC ha finalizado, el resultado se deja en estos dos registros.
* ADCH: Data Register High
* DIDR0: Digital Input Disable Register 0. Para deshabilitar la entrada digital de los pines analógicos. Página 326.

**Diagrama de bloques**

****

# Diagrama de flujo.



# Materiales utilizados

1 ATMEGA328P

1 Push Button

1 Potenciómetro

1 resistencaias (10K)

3 capacitores

1 Display 7 seg

# Código en Atmel.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* LLENAR ESTE ESPACIO CON LOS SIGUIENTES DATOS: \*

\* Nombre: Verónica Yazmín Gómez Cruz \*

\* Nahaliel Gamaliel Rios Martinez \*

\* Hora clase: N1-N2 \*

\* Día: M \*

\* N° de lista: 17, 18 \*

\* N° de Equipo: 7 \*

\* Dispositivo: ATMEGA328P \*

\* Rev: 1.0 \*

\* Propósito de la actividad: \*

\* Implementar un detector de rango el cual muestre \*

\* las siguientes letras en un display dependiendo \*

\* de los niveles existentes en la entrada del \*

\* microcontrolador utilizando un potenciómetro \*

\* (resistencia variable) en la entrada siendo \*

\* leído por el ADC para convertir el nivel de \*

\* voltaje mostrado en un número binario. \*

\* \*

\* - 0 a 50 (numero decimal)? Letra "L" de Low \*

\* (Letra mostrada en el display) \*

\* - 51 a 100 ? Letra "S" de Safe \*

\* - 101 a 255 ? Letra "H" de High \*

\* Fecha: 13.05.2021 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*atmega328P PIN - OUT\*/

/\* PIN - OUT

atmega328P

-------

PC6 |1 28| PC5

PD0 |2 27| PC4

PD1 |3 26| PC3

PD2 |4 25| PC2

PD3 |5 24| PC1

PD4 |6 23| PC0

VCC |7 22| GND

GND |8 21| AREF

PB6 |9 20| AVCC

PB7 |10 19| PB5

PD5 |11 18| PB4

PD6 |12 17| PB3

PD7 |13 16| PB2

PB0 |14 15| PB1

--------

\*/

/\*atmega328P PIN FUNCTIONS\*/

/\*

atmega328P PIN FUNCTIONS

pin function name pin function name

1 !RESET/PCINT14 PC6 15 PCINT1/OC1A PB1

2 RxD/PCINT16 PD0 16 PCINT2/OC1B/SS PB2

3 TxD/PCINT17 PD1 17 PCINT3/OC2A/MOSI PB3

4 INT0/PCINT18 PD2 18 PCINT4/MISO PB4

5 INT1/PCINT19/OC2B PD3 19 PCINT5/SCK PB5

6 PCINT20 PD4 20 ANALOG VCC AVCC

7 +5v VCC 21 ANALOG REFERENCE AREF

8 GND GND 22 GND GND

9 XTAL1/PCINT6 PB6 23 PCINT8/ADC0 PC0

10 XTAL2/PCINT7 PB7 24 PCINT9/ADC1 PC1

11 PCINT21/OC0B PD5 25 PCINT10/ADC2 PC2

12 PCINT22/OC0A/AIN0 PD6 26 PCINT11/ADC3 PC3

13 PCINT23/AIN1 PD7 27 PCINT12/ADC4/SDA PC4

14 PCINT0/AIN1 PB0 28 PCINT13/ADC5/SCL PC5

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Bibliotecas\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <avr/io.h>//se incluyen las Bibliotecas de E/S del AVR atmega328P

#include <avr/interrupt.h>// librería de interrupciones

#include <avr/delay.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Macros y constantes\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define *F\_CPU* 1000000UL //1 Mhz

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Variables globales\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//--Espacio para declarar variables globales

#define a PINB0

#define b PINB1

#define c PINB2

#define d PINB3

#define e PINB4

#define f PINB5

#define g PINB6

*uint8\_t* numeros[3] = {

//gfedcba

0b0111000, //L

0b1101101, //S

0b1110110, //H

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Funciones\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//--Espacio para Establecer funciones

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Declaración de Funciones\*\*\*\*\*\*\*\*\*\*\*\*\*/

//--Espacio para declarar funciones

void initialize\_ports(void); // Inicializar puertos

void ADC\_init(void);

void ADC\_on(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Programa principal\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main(void)

{

//--Inicialización

cli();

initialize\_ports(); // va hacía la inicialización

ADC\_init();

sei();

ADC\_on();

//--Ejecución

while (1) //loop infinito

{

} // END loop infinito

} // END MAIN

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Definición de funciones\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Descripcion de lo que hace la funcion: \*

//initialize\_ports : inicializa los puertos de entrada o \*

//salida \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void initialize\_ports(void)

{

//--Entradas

//--Salidas

DDRB |=\_BV(a);

DDRB |=\_BV(b);

DDRB |=\_BV(c);

DDRB |=\_BV(d);

DDRB |=\_BV(e);

DDRB |=\_BV(f);

DDRB |=\_BV(g);

PORTB = 0x00; //-Por seguridad iniciamos en 0

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Descripcion de lo que hace la funcion: \*

//ADC\_init : Habilitamos la interrupción y configuramos \*

// el ADC \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void ADC\_init(void)

{

//Avcc como pin de referencia

ADMUX &=~ (1<<REFS1);

ADMUX |= (1<<REFS0);

//8 bits

ADMUX |= (1<<ADLAR);

//PIN ADC4

ADMUX &=~ (1<<MUX3);

ADMUX |= (1<<MUX2);

ADMUX &=~ (1<<MUX1);

ADMUX &=~ (1<<MUX0);

//Freeruning

ADCSRA |= (1<<ADATE);

//Habilitar interrupción

ADCSRA |= (1<<ADIE);

//velocidad de muestreo

// 1 MHz clock / 8 = 125 kHz ADC clock debe de estar entre 50 - 200Khz

ADCSRA &=~ (1<<ADPS0);

ADCSRA |= (1<<ADPS1);

ADCSRA |= (1<<ADPS2);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Descripcion de lo que hace la funcion: \*

//ADC\_init : Leer y convertir señal análoga \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void ADC\_on(void)

{

//Encendemos el ADC

ADCSRA |= (1<<ADEN);

*\_delay\_ms*(10);

// Iniciar la conversión

ADCSRA |= (1 << ADSC);

}

ISR(ADC\_vect)

{

//0 a 5V -> 0 a 255bits

//0 a 50 (numero decimal)? Letra "L" de Low (Letra mostrada en el display)

if ( (ADCH >= 0) && (ADCH <= 50) )

{

PORTB = numeros[0];

//51 a 100 ? Letra "S" de Safe

}else if((ADCH >= 51) && (ADCH <= 100)){

PORTB = numeros[1];

//101 a 255 ? Letra "H" de High

}else if((ADCH >= 101) && (ADCH <= 255)){

PORTB = numeros[2];

}

}

# Diagrama del circuito en PROTEUS.

Diagrama

Descripción generada automáticamente con confianza media

**Conclusión**

En esta práctica de laboratorio aprendimos cuales son los registros que debemos configurar para utilizar las entradas analógicas del ATMEGA328P. También vimos otra utilidad de las interrupciones que veníamos viendo en prácticas anteriores, pues son necesarias para utilizar este tipo de entradas. Creo que aprender a utilizar las entradas analógicas y saber configurarlas correctamente dependiendo de los valores que estamos esperando es de vital importancia pues se utilizan mucho para manejar diversos tipos de sensores

**Bibliografía**

Floyd, T. L. (2006). Fundamentos de sistemas digitales. Pearson Educación.

JECRESPOM (2017b, septiembre 5). ADC –. Aprendiendo Arduino. https://aprendiendoarduino.wordpress.com/tag/adc/#:%7E:text=El%20ADC%20puede%20trabajar%20en,luego%20comienza%20con%20la%20siguiente.

ATmega328P. 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash. DATASHEET. https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P\_Datasheet.pdf