INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

PRÁCTICA 1

PUERTOS DE ENTRADA Y SALIDA

SISTEMAS EN CHIP

6SCM3

PROFESOR:

AGUILAR SÁNCHEZ FERNANDO

INTEGRANTES:

MIGUEL VÁSQUEZ MARIANO JOSUÉ

FRANCO OLVERA ODER

GONZALEZ RAMOS ANGEL DAVID

Fecha de Elaboración

28/02/2024

INTRODUCCIÓN

El atmega8535 cuenta con 32 pines de entrada y de salida las cuales se encuentran divididos en 4 grupos (el A,B,C,D). Cada pin se puede configurar para que sea de entrada o salida. Esto se puede hacer mediante el código del programa especificándolo como “In” para el caso de entrada u “Out” para el caso de salida.

Cada pin utiliza una resistencia Pull Up interna, una resistencia Pull Up es una configuración entre una resistencia y un interruptor el cual permite que a su salida sea un “1” lógico cuando el interruptor este abierto y un “0” lógico cuando este cerrado; para el caso de una configuración que funcione al revés, es decir, que cuando el interruptor este abierto su salida sea “0” lógico y cuando este cerrado su salida sea “1” lógico, se le llama resistencia de Pull Down. El hecho de que el Atmega8535 utilice una resistencia Pull Up, significa que considerara un “1” lógico cuando el interruptor conectado al mismo este abierto y “0” lógico cuando no. Si se desea que el resistor funcione al revés, se puede realizar desde la configuración de los pines en el código utilizando el valor de “T” para que con el interruptor cerradp se considere un “1” lógico y un “0” lógico cuando este abierto. Si se desea utilizar la configuración por defecto, se utiliza el valor de “P”.

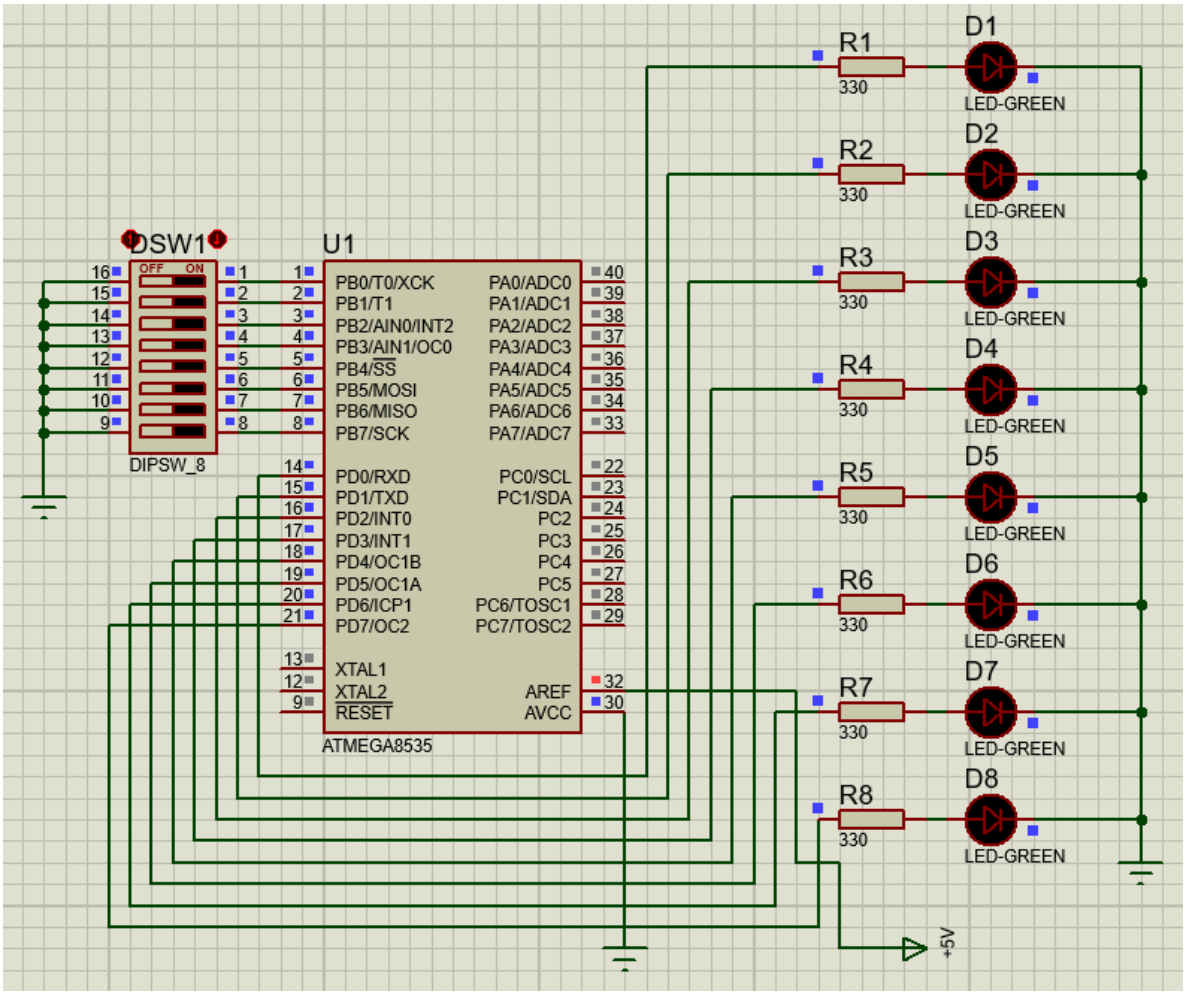
CÓDIGO

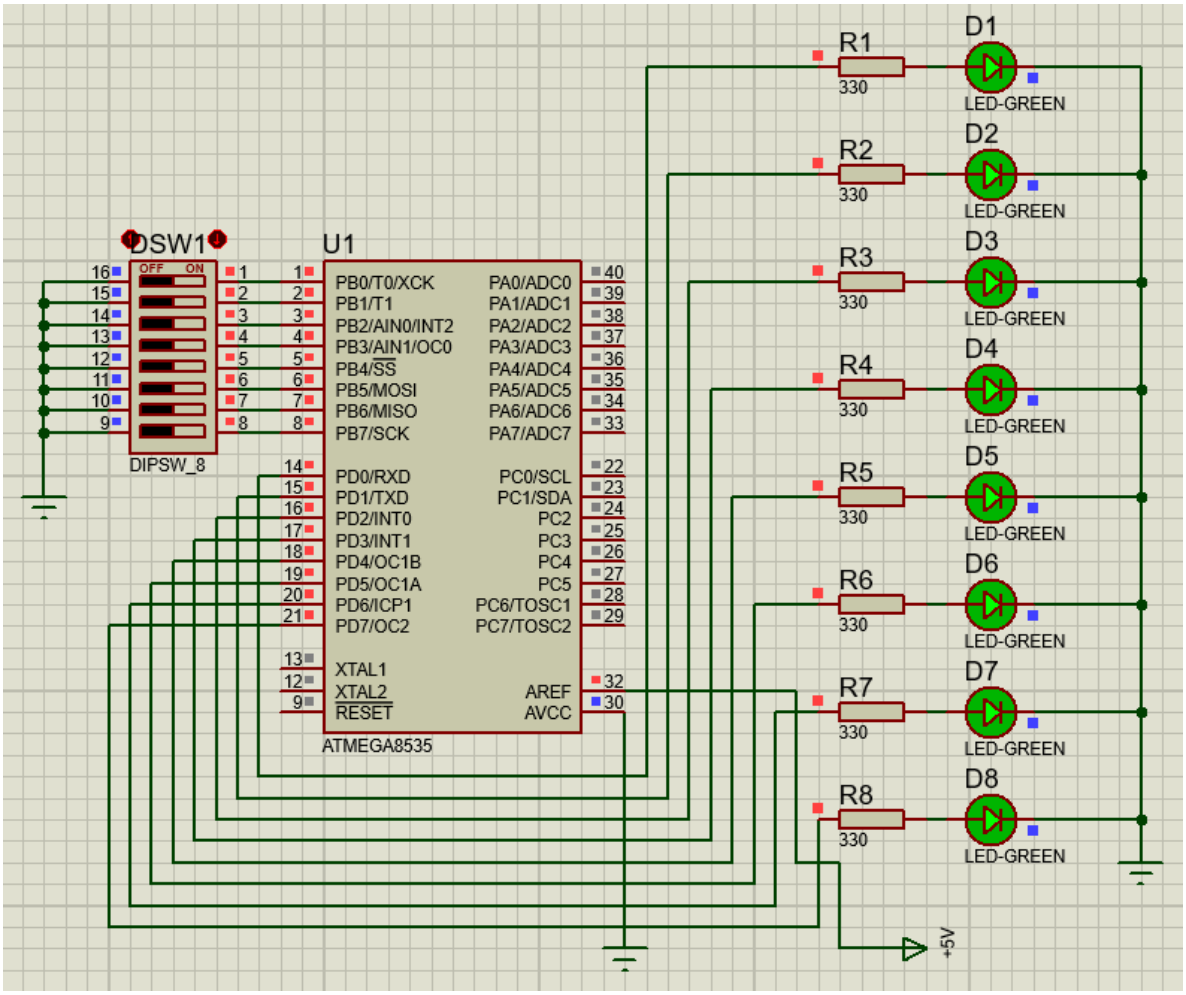
```
/*****************************************************
This program was created by the CodeWizardAVR V4.01
Automatic Program Generator
© Copyright 1998-2024 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro

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
*****************************************************/

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

// 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=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) | (0<<DDB2) |
(0<<DDB1) | (0<<DDB0);
// State: Bit7=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P
PORTB=(1<<PORTB7) | (1<<PORTB6) | (1<<PORTB5) | (1<<PORTB4) | (1<<PORTB3) |
(1<<PORTB2) | (1<<PORTB1) | (1<<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=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out
Bit0=Out
DDRD=(1<<DDD7) | (1<<DDD6) | (1<<DDD5) | (1<<DDD4) | (1<<DDD3) | (1<<DDD2) |
(1<<DDD1) | (1<<DDD0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
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)
```

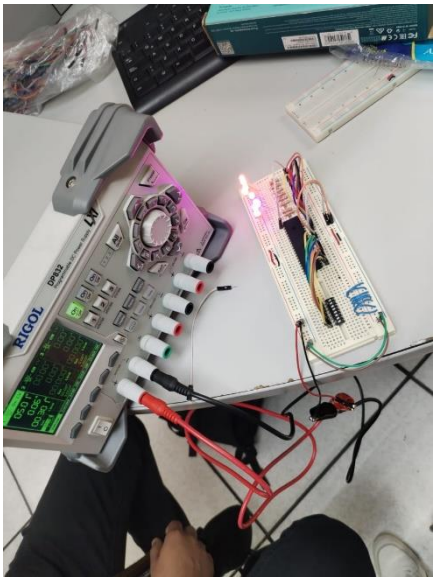
```
| (0<<CS20);
TCNT2=0x00;
OCR2=0x00;

// 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);
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)
      {
      // Place your code here
      PORTD=PINB;
      }
}
```

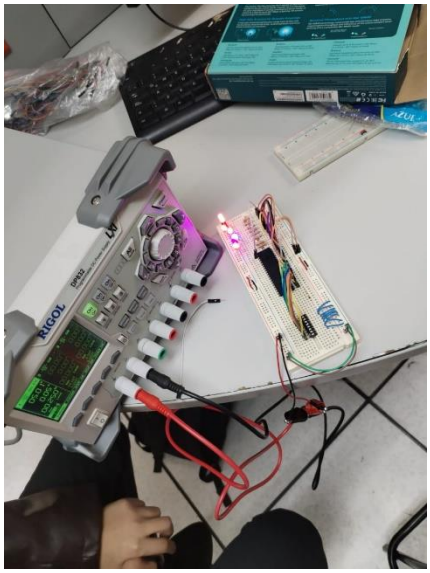
SIMULACIÓN EN PROTEUS




DSW1
OFF ON
DIPSW_8
U1
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
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
ATMEGA8535
R1
R2
R3
R4
R5
R6
R7
R8
330
D1
D2
D3
D4
D5
D6
D7
D8
LED-GREEN
+5V

FOTOS





CONCLUSIONES INDIVIDUALES

Franco Olvera Demian Oder

Este fue el primer acercamiento práctico que se tuvo con el Microcontrolador ATMEGA y, antes de realizar cualquier práctica más elaborada, se comenzó a utilizar únicamente un programa simple que una conexión directa entre entradas y salidas del CI. Aquí se aprendieron elementos básicos, como su arquitectura, programación y forma de alimentación y conexión.

La programación no podría ser más sencilla ya que, gracias al IDE CodeVision es posible adaptar código C al Microcontrolador, facilitando muchos pasos y ahorrando el tiempo y necesidad de aprender en otros lenguajes como el Ensamblador específico de su arquitectura.

Miguel Vásquez Mariano Josué

Con el desarrollo de esta práctica, pudimos conocer el funcionamiento de los pines de entrada y salida del Atmega8535; ya que estos utilizan de manera interna una resistencia Pull Up. Asi mismo, aprendimos a utilizar AVRStudio para la configuración de este. También, implementamos nuestro primer código utilizado para programar el funcionamiento del programa y con ella generar un archivo. hex que nos permitiese programarlo en el microcontrolador.

Gonzalez Ramos Angel David

La práctica de puertos de entrada y salida en un microcontrolador como el ATmega8535 es esencial para el desarrollo efectivo de sistemas embebidos, proporcionando flexibilidad, control y eficiencia en el uso de recursos, junto con una comprensión más profunda de la programación a bajo nivel y las consideraciones de diseño.

REFERENCIAS

- How and why to add pull-up and pull-down resistors to Microcontroller I/O pins | CNMAT. (s. f.). https://cnmat.berkeley.edu/content/how-and-why-add-pull-and-pull-down-resistors-microcontroller-io-pins
- ATMEL. 8-bit Microcontroller with 8K Bytes In-System Programmable Flash. ATmega8535 ATmega8535L. 2502HS–AVR–04/06. 2006. [Revisión Agosto 2006 – Octubre 2006].