Измерение частоты сигнала с помощью микроконтроллеров AVR. Простой частотомер.

Цель: Изучить принцип измерения частоты сигнала на примере микроконтроллера ATmega8.

Существует много схемных решений и способов измерения частоты сигнала, в том числе и при помощи микроконтроллеров. Мы разберем пример самого простого частотомера, построенного на контроллере **Atmega8**. Схема частотомера показана на рисунке 1. Для обеспечения хорошей точности измерения микроконтроллер тактируется от генератора с внешним кварцем частотой 8MHz. Измеренные показания выводятся на **LCD 1602** с контроллером **HD44780**. VD1 стабилитрон ограничивает амплитуду входного сигнала, R2 токоограничительный резистор.

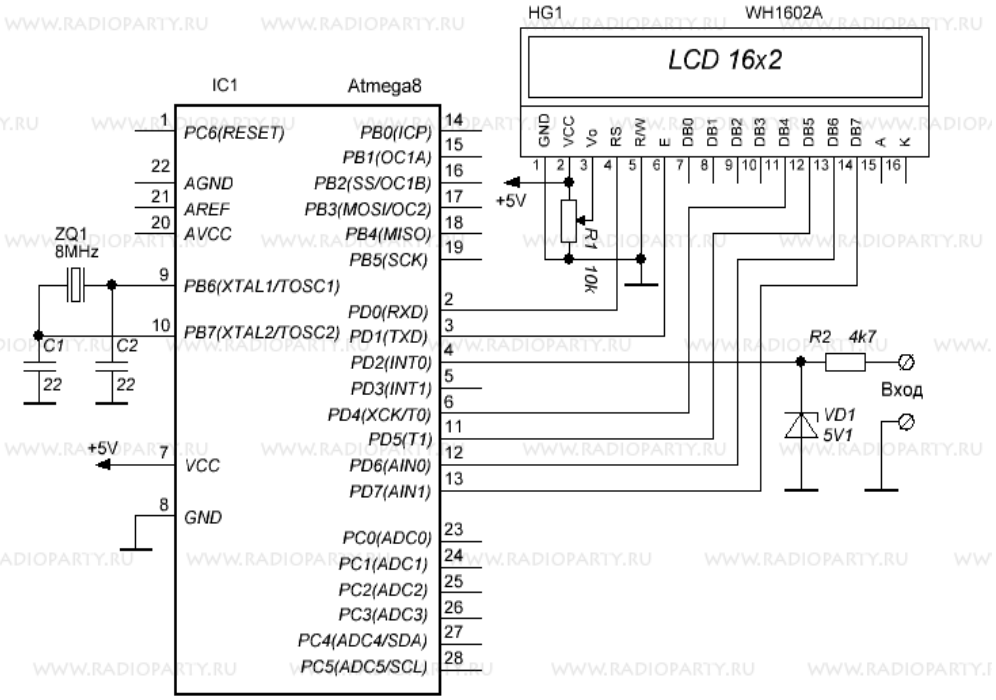

Рисунок 1.

Чтобы измерить частоту сигнала необходимо подсчитать количество импульсов, поступающих на вход микроконтроллера, за единицу времени. Для этого в нашей программе используем два типа прерывания: прерывание по переполнению таймера T0 и внешнее прерывание по изменению сигнала на входе **INT0**. Количество поступающих на вход сигналов будем подсчитывать за время - 1 секунда. Восьмибитный таймер T0 будет работать с частотой 1MHz, для этого включим предделитель на 8. Обработчик прерывания по переполнению таймера вызывается 4000 раз в секунду, при этом переменная **counter** увеличивает свое значение на единицу. Как только эта

переменная станет равна 4000, т.е. пройдет 1 секунда, на дисплей уйдет информация о переменной **edgecounter**, затем обе переменные обнуляются. Все это происходит уже в главном цикле. Переменная **edgecounter** увеличивает свое значение на единицу каждый раз, когда на входе **INT0** происходит смена фронта сигнала, т.е. поступает 1 импульс.

Исходный код программы:

```
// Измерение частоты сигнала с помощью микроконтроллеров AVR. Простой частотомер.
#include<avr/io.h>
#include<avr/interrupt.h>
#include<util/delay.h>

volatile unsigned int edgecounter = 0, counter = 0;

// Обработчик прерывания по переполнению T0, вызывается 4000 раз в секунду
ISR(TIMER0_OVF_vect)
{
TCNT0 = 6; // Счетчик T0 начинает считать с 6, т.к. 1MHz/(256-6) = 4000Hz
counter++;
}

// Обработчик внешнего прерывания
ISR(INT0_vect)
{
edgecounter++;
}
// Функции работы с LCD
#define RS PD0
#define EN PD1

// Функция передачи команды
void lcd_com(unsigned char p)
{
PORTD &= ~(1 << RS); // RS = 0 (запись команд)
PORTD |= (1 << EN); // EN = 1 (начало записи команды в LCD)
PORTD &= 0x0F; PORTD |= (p & 0xF0); // старший нибл
_delay_us(100);
PORTD &= ~(1 << EN); // EN = 0 (конец записи команды в LCD)
```

```
_delay_us(100);
PORTD |= (1 << EN); // EN = 1 (начало записи команды в LCD)
PORTD &= 0x0F; PORTD |= (p << 4); // младший нибл
_delay_us(100);
PORTD &= ~(1 << EN); // EN = 0 (конец записи команды в LCD)
_delay_us(100);
}

// Функция передачи данных
void lcd_data(unsigned char p)
{
PORTD |= (1 << RS)|(1 << EN); // RS = 1 (запись данных), EN - 1 (начало записи команды в LCD)
PORTD &= 0x0F; PORTD |= (p & 0xF0); // старший нибл
_delay_us(100);
PORTD &= ~(1 << EN); // EN = 0 (конец записи команды в LCD)
_delay_us(100);
PORTD |= (1 << EN); // EN = 1 (начало записи команды в LCD)
PORTD &= 0x0F; PORTD |= (p << 4); // младший нибл
_delay_us(100);
PORTD &= ~(1 << EN); // EN = 0 (конец записи команды в LCD)
_delay_us(100);
}

// Функция вывода строки на LCD
void lcd_string(unsigned char command, char *string)
{
lcd_com(0x0C);
lcd_com(command);
while(*string != '\0')
{
lcd_data(*string);
string++;
}
}

// Функция вывода переменной
void lcd_num_to_str(unsigned int value, unsigned char nDigit)
{
 switch(nDigit)
 {
 case 4: lcd_data((value/1000)+'0');
 case 3: lcd_data(((value/100)%10)+'0');
 case 2: lcd_data(((value/10)%10)+'0');
 case 1: lcd_data((value%10)+'0');
```

```
 }
}

// Функция инициализации LCD
void lcd_init(void)
{
_delay_ms(50); // Ожидание готовности ЖК-модуля
// Конфигурирование четырехразрядного режима
PORTD |= (1 << PD5);
PORTD &= ~(1 << PD4);
// Активизация четырехразрядного режима
PORTD |= (1 << EN);
PORTD &= ~(1 << EN);
_delay_ms(5);
lcd_com(0x28); // шина 4 бит, LCD - 2 строки
lcd_com(0x08); // полное выключение дисплея
lcd_com(0x01); // очистка дисплея
_delay_us(100);
lcd_com(0x06); // сдвиг курсора вправо
lcd_com(0x0C); // включение дисплея, курсор не видим
}

int main(void)
{
// Настройка портов ввода/вывода
DDRD = 0b11110011;
PORTD = 0x00;
// Настройка таймера T0
TCCR0 |= (1 << CS01); // Предделитель на 8, частота таймера 1 MHz
TIMSK |= (1 << TOIE0); // Разрешаем прерывание от таймера T0
// Настройка внешнего прерывания
GICR |= (1 << INT0); // Разрешаем внешнее прерывание на входе INT0
MCUCR |= (1 << ISC01)|(1 << ISC00); // Внешнее прерывание формируется по переднему фронту

sei(); // Глобально разрешаем прерывания

lcd_init(); // Инициализация дисплея
lcd_com(0x01);
lcd_string(0x80, "Frequency Meter");
lcd_string(0xC0, "F =      Hz");
while(1)
{
// Выводим показания на дисплей
if(counter == 4000)
```

```
{
lcd_com(0xC4);
lcd_num_to_str(edgecounter, 4);
counter = 0;
edgecounter = 0;
}
}
}
```

Задания для выполнения:

1. Изучить принцип измерения частоты на примере микроконтроллера ATmega8.
2. Реализовать приложенную программу в среде Proteus.

Результаты работы отправить на e-mail: rasov@rambler.ru с темой **Частотомер_ФИО**