Омский государственный технический университет

Курсовой проект

по дисциплине

«Проектирование современных микроконтроллерных устройств управления электрическими аппаратами»

Выполнил: магистрант
гр. ЭЭм-144 Иванов А.Б.

Омск - 2015

Техническое задание на проектирование.

1. Разработать программу микропроцессорного устройства для вычисления активной мощности, потребляемой от источника и действующих значений напряжения и тока нагрузки.
2. Для получения сигналов, подаваемых на входы АЦП, используется измерительный трансформатор напряжения и измерительный трансформатор тока, амплитудное значение выходных сигналов не должно превышать 2,5 В.
3. Нагрузка подключается к однофазной сети с напряжением 220±20 В, предельное значение тока 20 А, частота сети 50 Гц.
4. Произвести симуляцию работы программы.

Содержание

Введение

В соответствии с техническим заданием была разработана программа для микроконтроллерного устройства вычисления активной и реактивной мощности, потребляемой от источника синусоидальной ЭДС, косинуса угла сдвига между напряжением и током нагрузки и действующих значений напряжения и тока нагрузки.

Программа написана на языке С для микроконтроллера Atmega16 фирмы ATMEL в программе CVAVR, симуляция была произведена при помощи программного пакета Proteus (рис. 1).

Сигналы тока и напряжения, подаваемые на входы МК в реальной схеме устройства формируются измерительным трансформатором напряжения и измерительным трансформатором тока, при этом амплитудные значения этих сигналов не должны превышать 2,5 В. В связи с тем что МК не воспринимает сигналы отрицательной полярности, вводится напряжение смещения, равное 2,5 В.

При симуляции процесса измерения в программе Proteus источники сигналов, действующие на входы МК, задаются в виде постоянной составляющей (2,5 В) и синусоидальных функций напряжения и тока с амплитудным значением не более 2,5 В.

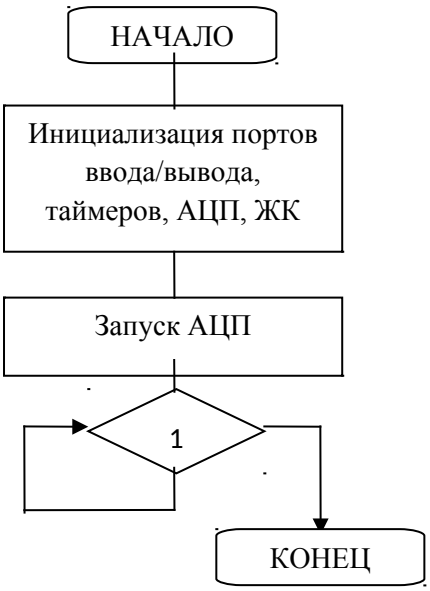
Описание алгоритма работы программы

В процессе работы производится постоянное считывание мгновенных значений тока и напряжения со входов АЦП микроконтроллера. После этого вычисляется мгновенная мощность и происходит накопление сумм мгновенных мощностей и квадратов тока и напряжения. Накопление производится в течение 10 периодов напряжения сети с целью усреднения полученных значений, после чего вычисляются действующие значения тока и напряжения, активная и реактивная мощность и косинус угла сдвига между напряжением и током нагрузки. Полученные значения выводятся на ЖК-дисплей.

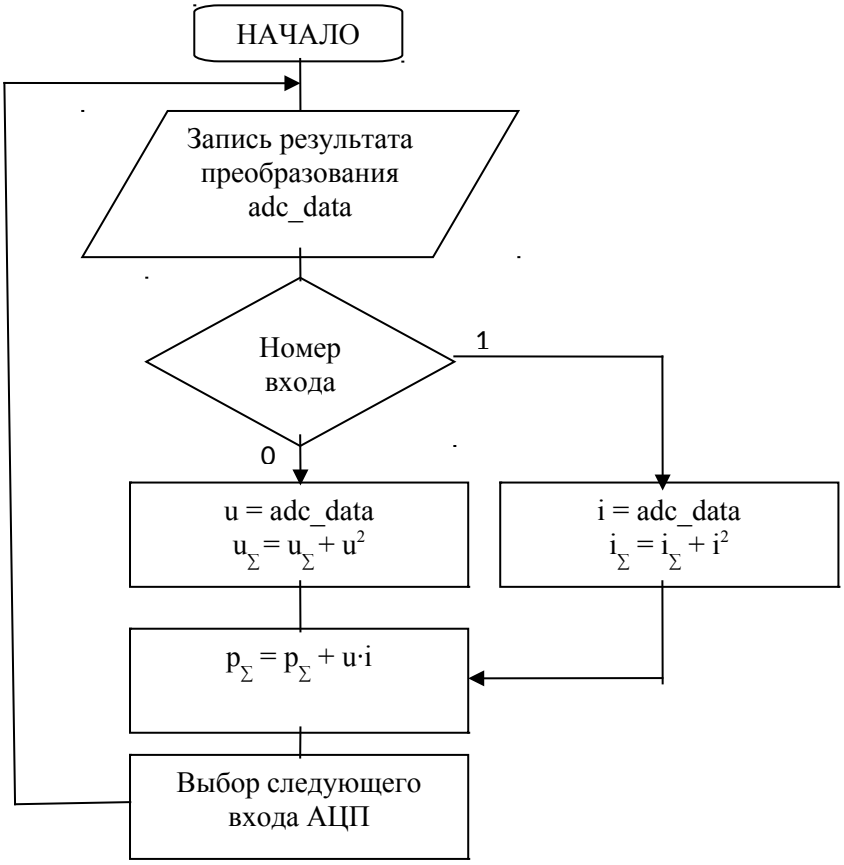
При тактовой частоте работы АЦП, равной 500 КГц, за 10 периодов сети (200 мс) АЦП делает 1176 отсчетов, т.е. за один период сети (20 мс) производится 117 отсчетов тока и напряжения. Сдвиг по времени отсчетов тока и напряжения составляет примерно 85 мкс без учета потерь времени на возведение в квадрат и суммирование. В данном случае отсчеты тока отстают от отсчетов напряжения примерно на 85 мкс.

Блок-схема алгоритма вычисления активной и реактивной мощности, потребляемой нагрузкой, действующих значений напряжения и тока нагрузки и косинуса угла сдвига фаз между напряжением и током.

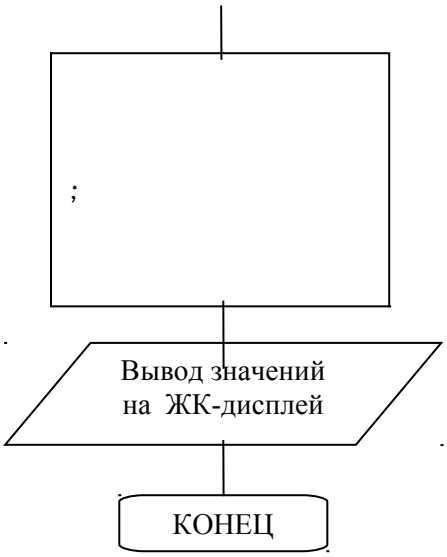
Основная программа



Подпрограмма обработки прерывания АЦП (обеспечивает непрерывное накопление сумм квадратов тока и напряжения и мгновенных мощностей)



Подпрограмма обработки прерывания таймера (обеспечивает процедуру вычисления активной и реактивной мощности, действующих значений тока и напряжения и косинуса угла сдвига между напряжением и током нагрузки за 10 периодов сети, при этом АЦП не работает, по окончании 10 периодов АЦП продолжает работу)



Листинг программы с комментариями

```
#include <mega16.h>     // подключение модуля atmega16
// выбор порта для подключения ЖК-дисплея
#asm
   .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>        // модуль для работы с ЖК-дисплеем
#include <delay.h>      // модуль с функциями задержки
#include <stdio.h>      // модуль стандартных функций ввода/выода
#include <math.h>
#define FIRST_ADC_INPUT 0       // первый вход АЦП
#define LAST_ADC_INPUT 1        // второй вход АЦП
int adc_data[LAST_ADC_INPUT-FIRST_ADC_INPUT+1]; // массив выходных данных АЦП
#define ADC_VREF_TYPE 0x20  // выбор источника опорного напряжения
#define c_u 124      // калибровочный коэффициент напряжения на нагрузке (коэффициент трансформации изм. тр-ра напряжения)
#define c_i 12      // калибровочный коэффициент  тока  нагрузки (коэффициент трансформации изм. тр-ра тока)

unsigned char  str1[16], str2[16], str3[16], str4[16]; // строки для вывода на ЖК-дисплей

float Us,Is, Ps, PP, cosf, Q;    //  Us - действ. знач. напряжения ,Is - тока , Ps - активная мощность, PP - сумма мгновенных мощностей, cosf - косинус ф, Q - реактивная
long P,U,I,Pm,Um,Im;  //   P, U, I - для накопления значений с АЦП;   Pm, Um, Im - сохранение накопленных значений для дальнейших вычислений

unsigned int cnt,          // количество выборок
          N,           // количество отсчетов АЦП за 10 периодов напряжения сети (частота сети 50 Гц)
          pr;          // переменная для задания количества периодов сети в течении которых накапливаются значения


// подпрограмма вывода данных на ЖК-дисплей
void lcd_output(void)
{
    sprintf(str1,"U:%.1f V", Us);       // составление 1ой строки
    sprintf(str2,"I:%.1f A", Is);
```

```
    sprintf(str3,"P:%.0f Q:%.0f", Ps, Q);   // составление 2ой строки
    sprintf(str4,"cos:%.1f N:%u", cosf, N);   // составление 3ей строки
    lcd_clear();                        // очистка ЖК-дисплея
    lcd_gotoxy(0,0);                    // определение позиции для вывода (1й символ 1й строки ЖК)
    lcd_puts(str1);                     // вывод 1ой строки
    lcd_gotoxy(0,1);                    // определение позиции для вывода (1й символ 2й строки ЖК)
    lcd_puts(str2);                     // вывод 2ой строки
    lcd_gotoxy(0,2);                    // определение позиции для вывода (1й символ 3й строки ЖК)
    lcd_puts(str3);                     // вывод 2ой строки
    lcd_gotoxy(0,3);                    // определение позиции для вывода (1й символ 3й строки ЖК)
    lcd_puts(str4);                     // вывод 2ой строки

}

// подпрограмма обработки перывания таймера_0
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{

TCNT0=0xB2;     // реинициализация стартового значения счетчика таймера_0
pr++;                       // наращиваем счетчик количества периодов сети
if (pr==10) {             // если количество периодов = 10
N=cnt; Pm=P; Um=U;Im=I;   // сохранение количества выборок за период
cnt=0; P=0;U=0;I=0;pr=0;     // сброс переменных

      Us=sqrt(Um/N)*c_u*5/256;  // вычисляем среднеквадратичное значение напряжения
      Is=sqrt(Im/N)*c_i*5/256;  // среднеквадратичное значение тока
      PP=Pm*25/65536;    // сумма мгновенных мощностей
      Ps=(PP/N)*c_u*c_i;                // активная мощность = среднее значение мгновенной мощности
за период
      cosf=fabs(Ps)/(Us*Is);   // вычисляем косинус ф
      Q=fabs(Us*Is)*sqrt(1-cosf*cosf);  // реактивная мощность
      lcd_output();   // вывод на ЖК-дисплей
}

}


// подпрограмма обработки прерывания АЦП с автоматическим сканированием входов
interrupt [ADC_INT] void adc_isr(void)
{
register static unsigned char input_index=0;   // номер текущего входа

adc_data[input_index]=ADCH;  // запись результата преобразования в массив

if (input_index==0) {U+=(long)(adc_data[0]-128)*(adc_data[0]-128);} // если считывали с 1го вывода
АЦП, то накапливаем квадрат напряжения
// из результата АЦП вычитаем 128 бит, соответствуюшие напряжению смещения 2.5 В

if (input_index==1) {                           // если считывали со 2го вывода АЦП, то
    I+=(long)(adc_data[1]-128)*(adc_data[1]-128);   // накапливаем квадрат тока
    P+=(long)(adc_data[0]-128)*(adc_data[1]-128);  // накапливаем мгновенную мощность
    cnt++;                  // наращиваем счетчик количества подсчитанных значений
}

if (++input_index > (LAST_ADC_INPUT-FIRST_ADC_INPUT)) {input_index=0;} // выбираем номер
следующего входа АЦП
ADMUX=(FIRST_ADC_INPUT | (ADC_VREF_TYPE & 0xff))+input_index;         // задаем номер входа
АЦП
ADCSRA|=0x40;       // начиннаем новое преобразование
}

// основная подпрограмма
void main(void)
```

```
{

// инициализация порта A
PORTA=0x00;
DDRA=0x00;

// инициализация порта B
PORTB=0x00;
DDRB=0x01;

// инициализация порта C
PORTC=0x00;
DDRC=0x00;

// инициализация порта D
PORTD=0x78;
DDRD=0x07;

// инициализация таймера_0
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// инициализация таймера_1 (не используется)
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// инициализация таймера_2 (не используется)
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// Инициализация обработки внешних прерываний (не используется)
MCUCR=0x00;
MCUCSR=0x00;

// Инициализация обработки прерываний таймеров
TIMSK=0x01;

// инициализация аналогового компаратора (не используется)
ACSR=0x80;
SFIOR=0x00;

// инициализация АЦП
ADMUX=FIRST_ADC_INPUT | (ADC_VREF_TYPE & 0xff);
ADCSRA=0x8B;    //выбор частоты 500 кГц (0x8B - 500 kHz   0x8C - 250 kHz)

// Инициализация ЖК-дисплея
lcd_init(16);
// Глобально разрешить прерыывания
#asm("sei")
lcd_output();  // вывод начальных значений на ЖК

ADCSRA.7=1;          // включаем АЦП
```

```
ADCSRA|=0x40;       // запускаем преобразование АЦП
TCNT0=0xB2;         // задаем начальное значение для таймера
TCCR0=0x05;         // задаем частоту таймера

while (1)     // основной цикл
        {
        }
}
```

Результаты симуляции в программе Proteus

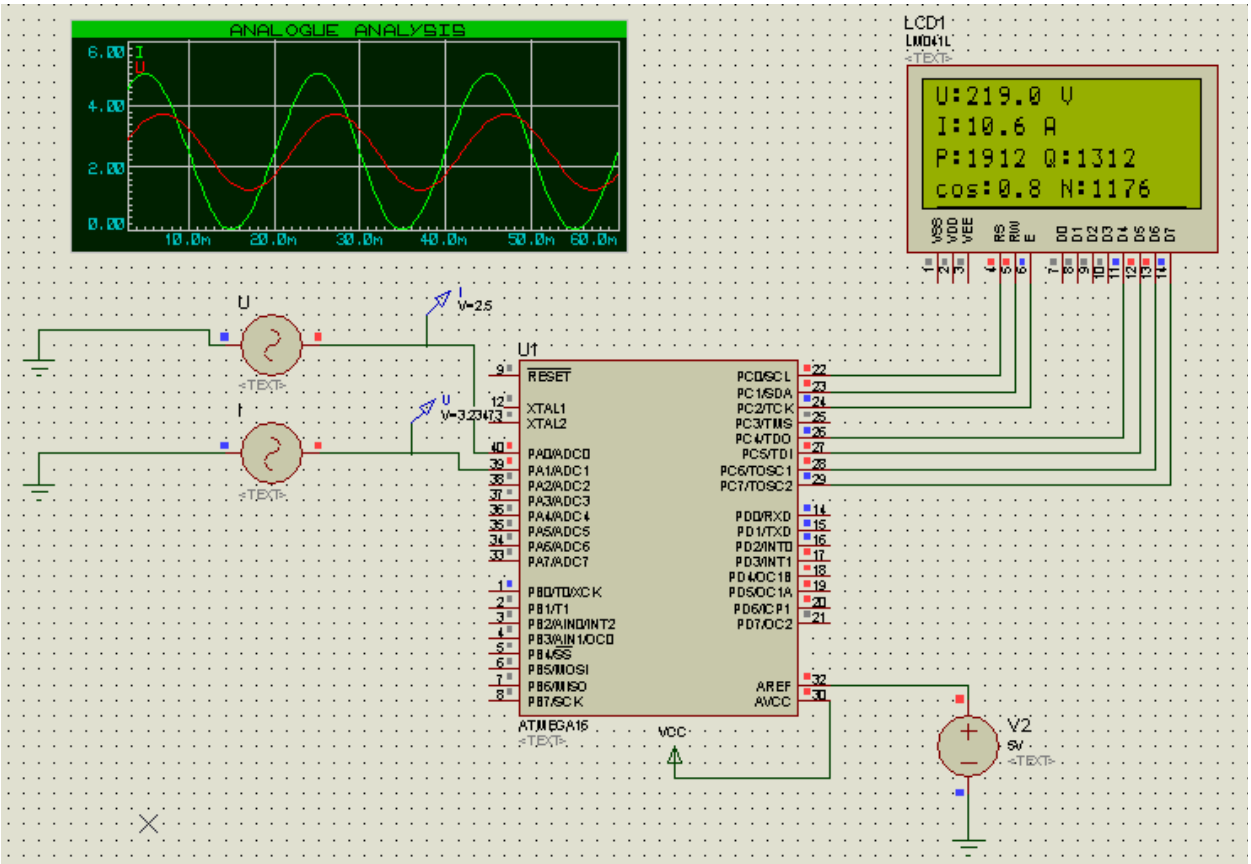


Рис. 1 Симуляция в программе Proteus

Заключение

Список литературы

1. Лебедев М.Б. CodeVisionAVR: пособие для начинающих, Москва, 2008
2. Шпак Ю.А. Программирование на языке С для AVR и PIC микроконтроллеров, Киев, 2006.