Федеральное агентство по образованию

Государственное образовательное учреждение высшего профессионального образования

Московский государственный технический университет им. Н.Э. Баумана

Кафедра «Робототехнические системы и мехатроника»

# Домашнее задание №2

по курсу:

**«Микроконтроллерные устройства управления»**

Тема:

**«Программирование микроконтроллеров с RISC архитектурой на базе микроконтроллеров семейства AVR в прерываниях для создания системы управления реального времени»**

**Вариант №4**

*Преподаватель:*

***к.т.н. доц. Солнцев В.И.***

*Студенты группы СМ7-84:*

***Аблятифов С.А.***

***Балкунов К.С.***

2024

**Цель домашнего задания:** Написать программы в соответствии с вариантом для программирования микроконтроллеров с RISC архитектурой на базе микроконтроллеров семейства AVR в прерываниях для создания системы управления реального времени.

**Задача №3**

При запуске программы на экран терминала выводится сообщение «Задача №3.

При нажатии на любую кнопку клавиатуры PC звучит зуммер с частотой 5 кГц длительностью 1 секунда, и символ выводится на ЖКИ стенда. При нажатии клавиш 0-9 соответствующие цифры также выводятся на семисегментный индикатор.

**Текст программы задачи 3 и описание ее работы:**

**1. Прерывание по таймеру**

1) Расчет частоты таймера

Используем режим CTC (Clear Timer on Compare Match) для таймера 0. Настроим его на прерывания каждую миллисекунду. Для кварца с частотой , выберем предделитель 256. Тогда, найдем величину значения регистра совпадения :

2) Инициализация таймера 0

1. void init\_timer0()
2. {   // Настройка таймера 0 на 1000 Гц (1 мс)
3. // Настройка тактирования таймера 1000 Гц (1 мс)
4. TCCR0 = 0;                    // Очистить конфигурационный регистр таймера 0
5. TCCR0 |= (1 << WGM01);  // устанавливаем режим СТС (сброс по совпадению)
6. TCCR0 |= (1 << CS02);     // Предделитель 256
7. OCR0 = 58;         // Значение для достижения 1000 Гц (1 мс) компаратор А
8. TIMSK\_OCIE0 |= 1;  // Запуск компаратора таймера 0
9. }

3) Обработчики прерываний таймера 0

Запуск обработки нажатия клавиш осуществляете каждый 10 мс, запуск функции для проигрывания последовательности сигналов зуммером – раз в 100 мс, т.к. все длительности сигналов и задержки между ними кратны 100.

1. #pragma vector = TIMER0\_COMP\_vect
2. \_\_interrupt void TIMER0\_COMP(void)
3. {
4. if (ms\_ctr % 10 == 0) button\_handler();
5. if (ms\_ctr % 100 == 0) beep\_handler();
6. ms\_ctr++;
7. }

**2.Прерывание по UART**

1. Инициализация прерываний по UART: UART\_init( CALC\_UBRR( 9600 ) );

Функции UART\_init() и CALC\_UBRR() даны были даны в методических материалах:

1. UART\_init(CALC\_UBRR(9600));
2. \_\_enable\_interrupt();
3. init\_timer1();
4. UART\_sendstring\_flash(UartMessageTask3);
5. OutputStartLCDMessage();
6. Расчет частоты

Установим нормальный асинхронный двунаправленный режим работы со скоростью 9600 бод. Функции UART\_init() и CALC\_UBRR() даны были даны в методических материалах. где f=14756000 Гц, band=9600.

|  |
| --- |
| 1. void UART\_init( unsigned short UBRR\_value ) {  UART\_UBRRL = ( unsigned char ) UBRR\_value;  UART\_UBRRH = ( unsigned char ) ( UBRR\_value >> 8 );    UART\_UCSRB\_RXEN = 1;  UART\_UCSRB\_TXEN = 1;  UART\_UCSRB\_RXCIE = 1; } |

Тогда UBRRH = 0, UBRRL = 51.

**3. Описание подпрограмм**

Обработчик нажатия кнопок клавиатуры стенда **button\_handler().** При регистрации нажатой клавиши запускает функцию **change\_key\_handler().**

1. void button\_handler()
2. {
3. // Обработчик нажатия кнопок клавиатуры
4. key = KEY\_getkey(); // подлучаем адрес нажатой клавиши
5. if (key != prev\_key)
6. {
7. if (button\_read\_flag == FALSE)
8. {
9. button\_read\_flag = TRUE;
10. button\_ms\_ctr = ms\_ctr;
11. return;
12. }
13. if (button\_read\_flag == TRUE && ms\_ctr - button\_ms\_ctr > 20)
14. button\_read\_flag = FALSE;
15. else
16. return;
17. change\_key\_handler();
18. prev\_key = key;
19. }
20. }

**Логика работы учёта дребезга контактов кнопок**

Сперва происходит проверка изменения состояния кнопки. Если текущий адрес клавиши (key) отличается от предыдущего (prev\_key), то если button\_read\_flag = FALSE происходит запись текущего времени работы программы в переменную button\_ms\_ctr измение флага button\_read\_flag на значнеие TRUE и выход из функции. Это нужно, для реализации задержки в 20 мс, которая обеспечит исключение дребезга контактов. При последующих вхождениях в функцию осуществляется проверка ms\_ctr - button\_ms\_ctr > 20. После выполнения этого условия флаг button\_read\_flag выставляется в FALSE, новое значение кнопки можно считать истинным, реализуется обработка нового значения.

Обработчик изменения нажатой клавиши клавиатуры **change\_key\_handler()**. Реализует вывод на данных на ЖК дисплей и семисегментный индикатор.

1. void change\_key\_handler()
2. {// обработчик нового значения нажатой клавиши клавиатуры
3. beep\_ctr = ms\_ctr;
4. curr\_signal\_dur\_idx = 0;
5. is\_seq\_started = TRUE;
6. is\_need\_delay = FALSE;
7. keyname = KEY\_getkeyname(key);
8. if (keyname >= '0' && keyname <= '9') {
9. SHIFT\_write(0);
10. LED7\_setdigit( keyname - '0' );
11. }
13. LCD\_clrscr();
14. LCD\_print(keyname);
15. beep\_on();
16. }

**beep\_handler()** реализует сигналы зуммера. Флаг is\_need\_delay обеспечивает своевременную задержку между сигналами, флаг is\_beep\_seq\_started отвечает за целостность последовательности сигналов.

1. void beep\_handler()
2. {// обработчик сигналов зумера
3. if (ms\_ctr - beep\_ctr > signal\_dur\_seq[curr\_signal\_dur\_idx] && is\_need\_delay == FALSE)
4. {
5. is\_need\_delay = TRUE;
6. beep\_ctr = ms\_ctr;
7. beep\_off();
8. curr\_signal\_dur\_idx++;
9. }
10. if (ms\_ctr - beep\_ctr > PAUSE\_DUR && is\_need\_delay == TRUE) {
11. is\_need\_delay = FALSE;
12. if (curr\_signal\_dur\_idx == 2) {
13. curr\_signal\_dur\_idx = 0;
14. is\_seq\_started = FALSE;
15. return;
16. }
17. beep\_on();
18. }
19. }

**Задача №4**

1) При запуске программы на экран терминала выводится сообщение «Задача №4.

2) При нажатии клавиши 1 клавиатуры PC предлагается ввести время и дату, которые затем устанавливаются как текущие в RTC DS1307. При нажатии клавиши 2 на ЖКИ стенда в первой строке выводится текущее время, во второй дата. При нажатии клавиши 3 происходит очистка ЖКИ. Нажатие на другие клавиши игнорируется.

3) Выводить на экран показания двух АЦП в вольтах, показания второго АЦП

кратные 10 выводить на семисегментный индикатор. При показаниях больше 99 включить зуммер.

**Текст программы задачи 4 и описание ее работы:**

**1. Прерывание по таймеру**

1) Расчет частоты таймера

Используем режим CTC (Clear Timer on Compare Match) для таймера 0. Настроим его на прерывания каждую миллисекунду. Для кварца с частотой , выберем предделитель 256. Тогда, найдем величину значения регистра совпадения :

2) Инициализация таймера 0

1. void init\_timer0()
2. {   // Настройка таймера 0 на 1000 Гц (1 мс)
3. // Настройка тактирования таймера 1000 Гц (1 мс)
4. TCCR0 = 0;                    // Очистить конфигурационный регистр таймера 0
5. TCCR0 |= (1 << WGM01);  // устанавливаем режим СТС (сброс по совпадению)
6. TCCR0 |= (1 << CS02);     // Предделитель 256
7. OCR0 = 58;         // Значение для достижения 1000 Гц (1 мс) компаратор А
8. TIMSK\_OCIE0 |= 1;  // Запуск компаратора таймера 0
9. }
10. Обработчики прерываний таймера 0

В зависимости от текущего значения режима (mode) вызываются соответствующие функции: change\_time\_n\_date() – изменение времени/даты, display\_time\_n\_date() – отображение времени/даты, с помощью функции ADC\_handler() вне зависимости от режима выводятся показания АЦП

1. #pragma vector = TIMER0\_COMP\_vect
2. \_\_interrupt void TIMER0\_COMP(void)
3. {
4. if (mode == 1)
5. change\_time\_n\_date();
6. if (mode == 2 && ms\_ctr % 250 == 0)
7. display\_time\_n\_date();
8. if (ms\_ctr % 50 == 0)
9. ADC\_handler();
10. ms\_ctr++;
11. }

**2.Прерывание по UART**

1) Инициализация прерываний по UART: UART\_init( CALC\_UBRR( 9600 ) );

2) Расчет аналогичен предыдущей задаче.

3) Обработка прерываний по UART

1. #pragma vector = UART\_RXC\_vect
2. \_\_interrupt void UART0\_RX\_interrupt()
3. { // Прерывание по UART
4. pc\_keyboard\_data = UART\_receivevalue();
5. UART\_sendstring("sended: ");
6. UART\_sendvalue(pc\_keyboard\_data);
7. UART\_sendvalue('\n');
8. if (pc\_keyboard\_data == '1')
9. mode = 1;   //режим изменения даты и времени
10. if (pc\_keyboard\_data == '2')
11. mode = 2;   //режим отображения даты и времени
12. if (pc\_keyboard\_data == '3')
13. {               //отчистка дисплея и установка режима бездейстивя
14. mode = 0;
15. LCD\_clrscr();
16. }
17. }

mode = 0 - режим бездействия  
mode = 1 - режим изменения даты и времени   
mode = 2 - режим отображения даты и времени

**Описание основных подпрограмм**

В режиме 2 отображается время на ЖК дисплее стенда

1. void display\_time\_n\_date()
2. { // вывести время и дату на ЖК дисплей
3. if (ms\_ctr % 250 == 0) // обновляем данные каждые 250 мс
4. {
5. LCD\_clrscr();
6. char time[9];
7. char date[9];
8. DS1307\_gettime(time);
9. DS1307\_getdate(date);
10. LCD\_gotoxy(0, 0);
11. LCD\_print(time);
12. LCD\_gotoxy( 0, 1 );
13. LCD\_print(date);
14. }
15. }

При нажатии клавиши 1 клавиатуры PC предлагается ввести время и дату, которые затем устанавливаются как текущие в RTC DS1307.

1. void change\_time\_n\_date()
2. { // установить новое время в микросхему DS1307 через UART-интерфейс
3. unsigned char hour, minute, second;
4. unsigned char day, month, year;
5. UART\_sendstring\_flash(enterHourStr);
6. hour = UART\_receivevalue();
7. UART\_sendstring\_flash(enterMinuteStr);
8. minute = UART\_receivevalue();
9. UART\_sendstring\_flash(enterSecondStr);
10. second = UART\_receivevalue();
11. DS1307\_settime(hour, minute, second);
12. UART\_sendstring\_flash(enterDayStr);
13. day = UART\_receivevalue();
14. UART\_sendstring\_flash(enterMonthStr);
15. month = UART\_receivevalue();
16. UART\_sendstring\_flash(enterYearStr);
17. year = UART\_receivevalue();
18. DS1307\_setdate(day, month, year);
19. mode = 0; // тк нужно выполнить однократно устанавливаем режим бездейстивя
20. }

Для вывода параметров АЦП используется функция-обработчик данных АЦП

1. void ADC\_handler()
2. { // обработчик АЦП
3. adc1 =  ADC\_get( 0 );
4. adc2 =  ADC\_get( 1 );
5. echo\_ADC(adc1, adc2);
6. beep\_if\_ADC\_over99(adc1, adc2) ;
7. LED7\_setdigit(int(adc2/10)); // Вывод данных второго АЦП на семисигментный индикатор
8. }

Функция echo\_ADC(adc1, adc2) отправляет в консоль значения АЦП

1. void echo\_ADC(int adc1, int adc2)
2. { // Вывод консоль по UART показания двух АЦП в вольтах
3. char str\_data\_ADC1[10];
4. char str\_data\_ADC2[10];
5. ShortToString( adc1, str\_data\_ADC1 );
6. ShortToString( adc2, str\_data\_ADC2 );
7. UART\_sendstring( "ADC1: " );
8. UART\_sendstring( str\_data\_ADC1 );
9. UART\_sendstring("\n\r");
10. UART\_sendstring( "ADC2: " );
11. UART\_sendstring( str\_data\_ADC2 );
12. UART\_sendstring("\n\r");
13. }

Функция beep\_if\_ADC\_over99(adc1, adc2) включает зуммер пи приевышении АЦП значения 99

1. void beep\_if\_ADC\_over99(int adc1, int adc2)
2. { // Включение зуммера при преодолении порога 99
3. if ((ADC\_get( 0 ) > 99) || (ADC\_get( 1 ) > 99))
4. BEEP\_PORT = 0;
5. else
6. BEEP\_PORT = 1;
7. }

**Вывод**

Особенностью работы устройства с применением прерываний является экономия процессорного времени и при использовании таймеров организация многозадачного режима работы. При работе с прерываниями в системе реального времени процессы обрабатываются практически одновременно.

**Приложение**

**Задача 3**

#include "SPI.H"

#include "LCD\_HD44780.H"

#include "I2C.H"

#include "KEYBOARD.H"

#include "LED7.H"

#include "UART.H"

#include "DS1307.H"

#include "DS1820.H"

#include "AT24CXX.H"

#include "CONVERT.H"

#include "PINDEF.h"

/\*

Студент:    Балкунов К.С.

Группа:     СМ7-84Б

Вариант: 4

Задача: 3

1) При запуске программы на экран терминала выводится сообщение «Задача №3»;

2) При нажатии на любую кнопку клавиатуры PC зуммер дает 3 сигнала

   длительностью 100 мс с паузой 100 мс,

   после паузы 300 мс – 3 сигнала длительностью 300 мс с паузой 100 мс,

   после паузы 300 мс - 3 сигнала длительностью 100 мс с паузой 100 мс,

   и символ выводится на ЖКИ стенда. При нажатии же клавиш 0-9 соответствующие цифры,

   помимо звукового сигнала, выводятся еще на семисегментный индикатор.

\*/

#define ONESHOT\_DELAY 100    // пауза 100 мс между одиночными сигналами

#define PAUSE\_DUR 300        // пауза 300 мс между сериями сигналов

#define BUTTONS\_ASK\_DUR 10   // время опроса нопок клавиатуры

#define FALSE 0

#define TRUE 1

int signal\_dur\_seq[] = {100, 300, 100}; // последовательность длительности сигналов в серии

flash char UartMessageTask3[] = "Task 3\r\n";

char key = 0;                // ключ символа нажатой кнопки клавиатуры

char prev\_key = 0;           // ключ символа нажатой кнопки клавиатуры на прошлом опросе кнопки

char is\_beep\_seq\_started = 0;     // флажок запуска последовательности сигналов после нажатия кнопки

char is\_need\_delay = 0;      // флажок для организации паузы между сигналами

char button\_read\_flag = FALSE;  //флажок чтения адресов кнопок клавиатуры стенда

int curr\_signal\_dur\_idx = 0; // индекс проигрываемого сигнала в серии сигналов

unsigned int ms\_ctr = 0;     // счетчик миллисекунд от начала работы программы

unsigned int beep\_ctr = 0;   // счетчик для задания длительности сигналов зумера

unsigned int button\_ms\_ctr = 0; // счетчик миллисекнуд для исключения дребезга контактов

void beep\_on() {BEEP\_BIT = 0;}

void beep\_off() {BEEP\_BIT = 1;}

void init\_timer0()

{   // Настройка таймера 0 на 1000 Гц (1 мс)

    // Настройка тактирования таймера 1000 Гц (1 мс)

    TCCR0 = 0;                    // Очистить конфигурационный регистр таймера 0

    TCCR0 |= (1 << WGM01);  // устанавливаем режим СТС (сброс по совпадению)

    TCCR0 |= (1 << CS02);     // Предделитель 256

    OCR0 = 58;         // Значение для достижения 1000 Гц (1 мс) компаратор А

    TIMSK\_OCIE0 |= 1;  // Запуск компаратора таймера 0

}

#pragma vector = TIMER0\_COMP\_vect

\_\_interrupt void TIMER0\_COMP(void)

{

    if (ms\_ctr % 10 == 0) button\_handler();

    if (ms\_ctr % 100 == 0) beep\_handler();

    ms\_ctr++;

}

void change\_key\_handler()

{// обработчик нового значения нажатой клавиши клавиатуры

    beep\_ctr = ms\_ctr;

    curr\_signal\_dur\_idx = 0;

    is\_seq\_started = TRUE;

    is\_need\_delay = FALSE;

    keyname = KEY\_getkeyname(key);

    if (keyname >= '0' && keyname <= '9') {

        SHIFT\_write(0);

        LED7\_setdigit( keyname - '0' );

    }

    LCD\_clrscr();

    LCD\_print(keyname);

    beep\_on();

}

void beep\_handler()

{// обработчик сигналов зумера

    if (ms\_ctr - beep\_ctr > signal\_dur\_seq[curr\_signal\_dur\_idx] && is\_need\_delay == FALSE)

    {

        is\_need\_delay = TRUE;

        beep\_ctr = ms\_ctr;

        beep\_off();

        curr\_signal\_dur\_idx++;

    }

    if (ms\_ctr - beep\_ctr > PAUSE\_DUR && is\_need\_delay == TRUE) {

        is\_need\_delay = FALSE;

        if (curr\_signal\_dur\_idx == 2) {

            curr\_signal\_dur\_idx = 0;

            is\_seq\_started = FALSE;

            return;

        }

        beep\_on();

    }

}

void button\_handler()

{

    // Обработчик нажатия кнопок клавиатуры

    key = KEY\_getkey(); // подлучаем адрес нажатой клавиши

    if (key != prev\_key)

    {

        if (button\_read\_flag == FALSE)

        {

            button\_read\_flag = TRUE;

            button\_ms\_ctr = ms\_ctr;

            return;

        }

        if (button\_read\_flag == TRUE && ms\_ctr - button\_ms\_ctr > 20)

            button\_read\_flag = FALSE;

        else

            return;

        change\_key\_handler();

        prev\_key = key;

    }

}

int main()

{

    I2C\_init();

    SPI\_init();

    LCD\_init();

    LCD\_clrscr();

    UART\_init(CALC\_UBRR(9600));

    \_\_enable\_interrupt();

    init\_timer0();

    UART\_sendstring\_flash(UartMessageTask3);

    OutputStartLCDMessage();

    while (1)

    {

        button\_handler(); // читаем данные и выполняем соответсвующие инструкции

    }

}

**Задача 4**

#include "SPI.H"

#include "LCD\_HD44780.H"

#include "I2C.H"

#include "KEYBOARD.H"

#include "LED7.H"

#include "UART.H"

#include "DS1307.H"

#include "DS1820.H"

#include "AT24CXX.H"

#include "CONVERT.H"

#include "PINDEF.h"

/\*

Студент:    Балкунов К.С.

Группа:     СМ7-84Б

Вариант: 4

Задача: 4

1)  При запуске программы на экран терминала выводится сообщение «Задача №4.

2)  При нажатии клавиши 1 клавиатуры PC предлагается ввести время и дату,

    которые затем устанавливаются как текущие в RTC DS1307.

    При нажатии клавиши 2 на ЖКИ стенда в первой строке выводится текущее время,

    во второй дата. При нажатии клавиши 3 происходит очистка ЖКИ.

    Нажатие на другие клавиши игнорируется.

3)  Выводить на экран показания двух АЦП в вольтах, показания второго АЦП

кратные 10 выводить на семисегментный индикатор. При показаниях больше 99 включить зуммер.

\*/

flash char UartMessageTask4[] = "Task 4\r\n";

flash char enterHourStr[] = "Enter hour: ";

flash char enterMinuteStr[] = "Enter minute: ";

flash char enterSecondStr[] = "Enter second: ";

flash char enterDayStr[] = "Enter day: ";

flash char enterMonthStr[] = "Enter month: ";

flash char enterYearStr[] = "Enter year: ";

flash char sensorNotConnectedStr[] = "Sensor is not connected";

int mode = 0;   //0 - бездействие, 1 - режим изменения даты/времени, 2- режим отображения даты/времени

unsigned char pc\_keyboard\_data = 0;

unsigned int ms\_ctr = 0;     // счетчик миллисекунд от начала работы программы

void init\_ADC() {ADCSRA |= (1 << ADEN);}    // активация АЦП

void init\_timer0()

{   // Настройка таймера 0 на 1000 Гц (1 мс)

    // Настройка тактирования таймера 1000 Гц (1 мс)

    TCCR0 = 0;                    // Очистить конфигурационный регистр таймера 0

    TCCR0 |= (1 << WGM01);  // устанавливаем режим СТС (сброс по совпадению)

    TCCR0 |= (1 << CS02);     // Предделитель 256

    OCR0 = 58;         // Значение для достижения 1000 Гц (1 мс) компаратор А

    TIMSK\_OCIE0 |= 1;  // Запуск компаратора таймера 0

}

void echo\_ADC(int adc1, int adc2)

{ // Вывод консоль по UART показания двух АЦП в вольтах

    char str\_data\_ADC1[10];

    char str\_data\_ADC2[10];

    ShortToString( adc1, str\_data\_ADC1 );

    ShortToString( adc2, str\_data\_ADC2 );

    UART\_sendstring( "ADC1: " );

    UART\_sendstring( str\_data\_ADC1 );

    UART\_sendstring("\n\r");

    UART\_sendstring( "ADC2: " );

    UART\_sendstring( str\_data\_ADC2 );

    UART\_sendstring("\n\r");

}

void beep\_if\_ADC\_over99(int adc1, int adc2)

{ // Включение зуммера при преодолении порога 99

    if ((ADC\_get( 0 ) > 99) || (ADC\_get( 1 ) > 99))

        BEEP\_PORT = 0;

    else

        BEEP\_PORT = 1;

}

void ADC\_handler()

{ // обработчик АЦП

    adc1 =  ADC\_get( 0 );

    adc2 =  ADC\_get( 1 );

    echo\_ADC(adc1, adc2);

    beep\_if\_ADC\_over99(adc1, adc2) ;

    LED7\_setdigit(int(adc2/10)); // Вывод данных второго АЦП на семисигментный индикатор

}

void change\_time\_n\_date()

{ // установить новое время в микросхему DS1307 через UART-интерфейс

    unsigned char hour, minute, second;

    unsigned char day, month, year;

    UART\_sendstring\_flash(enterHourStr);

    hour = UART\_receivevalue();

    UART\_sendstring\_flash(enterMinuteStr);

    minute = UART\_receivevalue();

    UART\_sendstring\_flash(enterSecondStr);

    second = UART\_receivevalue();

    DS1307\_settime(hour, minute, second);

    UART\_sendstring\_flash(enterDayStr);

    day = UART\_receivevalue();

    UART\_sendstring\_flash(enterMonthStr);

    month = UART\_receivevalue();

    UART\_sendstring\_flash(enterYearStr);

    year = UART\_receivevalue();

    DS1307\_setdate(day, month, year);

    mode = 0; // тк нужно выполнить однократно устанавливаем режим бездейстивя

}

void display\_time\_n\_date()

{ // вывести время и дату на ЖК дисплей

    if (ms\_ctr % 250 == 0) // обновляем данные каждые 250 мс

    {

        LCD\_clrscr();

        char time[9];

        char date[9];

        DS1307\_gettime(time);

        DS1307\_getdate(date);

        LCD\_gotoxy(0, 0);

        LCD\_print(time);

        LCD\_gotoxy( 0, 1 );

        LCD\_print(date);

    }

}

#pragma vector = TIMER0\_COMP\_vect

\_\_interrupt void TIMER0\_COMP(void)

{

    if (mode == 1)

        change\_time\_n\_date();

    if (mode == 2 && ms\_ctr % 250 == 0)

        display\_time\_n\_date();

    if (ms\_ctr % 50 == 0)

        ADC\_handler();

    ms\_ctr++;

}

#pragma vector = UART\_RXC\_vect

\_\_interrupt void UART0\_RX\_interrupt()

{ // Прерывание по UART

    pc\_keyboard\_data = UART\_receivevalue();

    UART\_sendstring("sended: ");

    UART\_sendvalue(pc\_keyboard\_data);

    UART\_sendvalue('\n');

    if (pc\_keyboard\_data == '1')

        mode = 1;   //режим изменения даты и времени

    if (pc\_keyboard\_data == '2')

        mode = 2;   //режим отображения даты и времени

    if (pc\_keyboard\_data == '3')

    {               //отчистка дисплея и установка режима бездейстивя

        mode = 0;

        LCD\_clrscr();

    }

}

int main()

{

    I2C\_init();

    SPI\_init();

    LCD\_init();

    LCD\_clrscr();

    init\_ADC();

    UART\_init(CALC\_UBRR(9600));

    \_\_enable\_interrupt();

    init\_timer1();

    UART\_sendstring\_flash(UartMessageTask4);

    OutputStartLCDMessage();

    while (1)

    {

    }

}