Федеральное агентство по образованию

Государственное образовательное учреждение высшего профессионального образования

Московский государственный технический университет им. Н.Э. Баумана

Кафедра «Робототехнические системы и мехатроника»

**Отчет по лабораторной работе №2**

по курсу:

***«Микроконтроллерные устройства управления»***

Тема:

***«Программирование и отладка устройства управления на базе микроконтроллеров семейства х51»***

(реализация домашнего задания: 1 и 2 задачи)

**Вариант №4**

*Преподаватель:*

***к.т.н. доц. Солнцев В.И.***

*Студенты группы СМ7-84:*

***Аблятифов С.А.***

***Балкунов К.С.***

2024

**Цель лабораторной работы:** отладить программы домашнего задания (задачи 1 и 2) для микроконтроллеров семейства х51 на микроконтроллерном стенде в системе реального времени.

**Ход выполнения лабораторной работы:**

1. Ознакомление с программно-аппаратными средствами составления и отладки программ на базе CISC микроконтроллеров семейства х51 в системе реального времени.
2. Подготовка кода программы задачи 1 для микроконтроллера семейства x51 с CISC архитектурой AT89S8252 с кварцевым резонатором 22.11 МГц для отладки на стенде.
3. Подготовка кода программы задачи 2 для микроконтроллера семейства x51 с CISC архитектурой AT89S8252 с кварцевым резонатором 22.11 МГц для отладки на стенде.
4. Отладка двух программ на экспериментальном стенде.
5. Составление отчета о лабораторной работе.

**Задача №1**

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

2) При нажатии на любую кнопку клавиатуры стенда соответствующий символ выводится на ЖКИ стенда и семисегментный индикатор. Зуммер дает 3 сигнала длительностью 300 мс с паузой 100 мс, после паузы 300 мс – 3 сигнала длительностью 100 мс с паузой 100 мс, после паузы 300 мс - 3 сигнала дли-тельностью 300 мс с паузой 100 мс.

3) Реализовать удаление последнего введенного сим-вола на экране ЖКИ кнопкой “\*”, а на семисегментном индикаторе при этом должна выводиться цифра, стоящая последней перед удаленной.

**Текст программы задачи 1 и описание ее работы:**

**1) Реализация прерываний в программе**

**1. Прерывание по таймеру**

Таймер 0 используется для управления временем задержки и периодами прерываний для зуммера. Регистры TL0 и TH0 задают начальное значение таймера, которое определяет, когда произойдёт прерывание.

**Расчёт периода прерывания:**

1. #define CLOCK 22110000
2. #define TIMER\_FREQ\_HZ 1000
3. #define RELOAD\_VALUE (65536 - (CLOCK / 12 / TIMER\_FREQ\_HZ))
4. #define RELOAD\_VALUE\_H ((unsigned char)(RELOAD\_VALUE >> 8))
5. #define RELOAD\_VALUE\_L ((unsigned char)(RELOAD\_VALUE))

где fРЕЗ=22118400 Гц, f = 1000 Гц.

TH0 = RELOAD\_VALUE\_H = 112 - старшие 8 бит таймера

TL0 = RELOAD\_VALUE\_L = 248 - младшие 8 бит таймера

**Инициализация таймера 0:**

1. void init\_timer0(void)
2. {// инициализация таймера
3. TMOD = 0x01;
4. TL0 = RELOAD\_VALUE\_L;
5. TH0 = RELOAD\_VALUE\_H;
6. ET0 = 1;
7. TR0 = 1;
8. }

**Обработчик прерываний таймера:**

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. void ISR(void) interrupt 1 using 0
2. {// обработка прерывания
3. if (ms\_ctr % 10 == 0)
4. button\_handler();
5. if (ms\_ctr % 100 == 0)
6. beep\_handler();
7. ms\_ctr++;
8. }

**2. Прерывание по UART**

1. Для реализации прерываний по UART используем автоперезагружаемый таймер 1 в 8 - битном режиме для этого меняем регистр TMOD.
2. TMOD = 0x21;
3. Расчет частоты

Последовательный порт 8051 будет работать в 3 режиме, поэтому частота приёма/передачи задается частотой таймера 1.

Выберем частоту UART 9600 бод и SMOD = 0.

Расчет частоты для таймера 1.

где fРЕЗ=22118400 Гц

Для кварца 22118400 Гц нужно поместить константу 0xFA (250) в TH1 и в TL1 для получения частоты UART 9600 бод.

**2) Описание выполнения каждой операции по заданию**

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. key\_name = KEY\_getkeynumber(key); // получаем цифру по адресу клавиши
18. change\_key\_handler();
19. prev\_key = key;
20. prev\_key\_name = KEY\_getkeyname(prev\_key);
21. }
22. }

**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. При нажатии \* с ЖК дисплея символ удаляется, а на семисегментном индикаторе воспроизводится предыдущий символ.

1. void change\_key\_handler()
2. {// обработчик изменения нажатой клавиши клавиатуры
3. beep\_ctr = ms\_ctr;
4. curr\_signal\_dur\_idx = 0;
5. is\_beep\_seq\_started = TRUE;
6. is\_need\_delay = FALSE;
8. //отчистка дисплеев
9. LCD\_clrscr();
10. SHIFT\_write(0);
11. if (key == KEY\_ASTERISK) {
12. LED7\_setdigit( prev\_key\_name );
13. }
14. else {
15. LED7\_setdigit( curr\_key\_name );
16. LCD\_print( curr\_key\_name );
17. }
18. beep\_on();
19. }

**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\_beep\_seq\_started = FALSE;
15. return;
16. }
17. beep\_on();
18. }
19. }

**Задача №2**

1) При запуске программы на экран терминала выводится сообщение «Задача №2».

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

кото-рые затем устанавливаются как текущие в RTC DS1307.

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

во второй дата.

4) При нажатии клавиши 3 происходит очистка ЖКИ.

Нажатие на другие клавиши игнорируется.

**Текст программы задачи 2** **и описание ее работы:**

**1) Реализация прерываний в программе**

**1. Прерывание по таймеру**

Таймер 0 используется для управления временем задержки и периодами прерываний для зуммера. Регистры TL0 и TH0 задают начальное значение таймера, которое определяет, когда произойдёт прерывание.

**Расчёт периода прерывания:**

1. #define CLOCK 22110000
2. #define TIMER\_FREQ\_HZ 1000
3. #define RELOAD\_VALUE (65536 - (CLOCK / 12 / TIMER\_FREQ\_HZ))
4. #define RELOAD\_VALUE\_H ((unsigned char)(RELOAD\_VALUE >> 8))
5. #define RELOAD\_VALUE\_L ((unsigned char)(RELOAD\_VALUE))

где fРЕЗ=22118400 Гц, f = 1000 Гц.

TH0 = RELOAD\_VALUE\_H = 112 - старшие 8 бит таймера

TL0 = RELOAD\_VALUE\_L = 248 - младшие 8 бит таймера

**Инициализация таймера 0:**

1. void init\_timer0(void)
2. { // инициализация таймера
3. TMOD = 0x01;
4. TL0 = RELOAD\_VALUE\_L;
5. TH0 = RELOAD\_VALUE\_H;
6. ET0 = 1;
7. TR0 = 1;
8. }

**Обработчик прерываний от таймера:**

1. void ISR(void) interrupt 1 using 0
2. { // обработка прерывания милисекундного счётчика
3. if (mode == 1)
4. change\_time\_n\_date();
5. if (mode == 2 && ms\_ctr % 250 == 0)
6. display\_time\_n\_date();
7. ms\_ctr++;
8. }

**2. Прерывание по UART**

1. Для реализации прерываний по UART используем автоперезагружаемый таймер 1 в 8 - битном режиме для этого меняем регистр TMOD.
2. TMOD = 0x21;
3. Расчет частоты

Последовательный порт 8051 будет работать в 3 режиме, поэтому частота приёма/передачи задается частотой таймера 1.

Выберем частоту UART 9600 бод и SMOD = 0.

Расчет частоты для таймера 1.

где fРЕЗ=22118400 Гц, f = 9600 бод.

Для кварца 22118400 Гц нужно поместить константу 0xFA (250) в TH1 и в TL1 для получения частоты UART 9600 бод.

1. Обработчик прерываний

Обработчик прерываний производит обработку отправленных через консоль клавиш ПК. В зависимости от нажатой клавиши меняется глобальная переменная mode, отвечающая за режимы работы программы.

1. void Serial\_ISR(void) interrupt 4 using 0
2. {// обработчик прерываний от UART
3. if (RI)
4. {
5. scanf("%bu", &pc\_keyboard\_data);
6. if (pc\_keyboard\_data == '1')
7. mode = 1; // режим изменения даты и времени
8. if (pc\_keyboard\_data == '2')
9. mode = 2; // режим отображения даты и времени
10. if (pc\_keyboard\_data == '3')
11. { // отчистка дисплея и установка режима бездейстивя
12. mode = 0;
13. LCD\_clrscr();
14. }
15. }
16. }

mode = 0 - режим бездействия

mode = 1 - режим изменения даты и времени

mode = 2 - режим отображения даты и времени

**Описание подпрограмм**

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
3. unsigned char hour, minute, second;
4. unsigned char day, month, year;
5. printf("Enter hour: ");
6. scanf("%bu", &hour);
7. printf("Enter minute: ");
8. scanf("%bu", &minute);
9. printf("Enter second: ");
10. scanf("%bu", &second);
11. DS1307\_settime(hour, minute, second);
12. printf("Enter day: ");
13. scanf("%bu", &day);
14. printf("Enter month: ");
15. scanf("%bu", &month);
16. printf("Enter year: ");
17. scanf("%bu", &year);
18. DS1307\_setdate(day, month, year);
19. mode = 0; // тк нужно выполнить однократно устанавливаем режим бездейстивя
20. }

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

во второй дата.

1. void display\_time\_n\_date()
2. {  // вывести время и дату на ЖК дисплей
3. LCD\_clrscr();
4. char time[9];
5. char date[9];
6. DS1307\_gettime(time);
7. DS1307\_getdate(date);
8. LCD\_gotoxy(0, 0);
9. LCD\_print(time);
10. LCD\_gotoxy( 0, 1 );
11. LCD\_print(date);
12. }

**Приложение**

**Задача 1**

#include <stdio.h>

#include <intrins.h>

#include <string.h>

#include "KEYBOARD.H"

#include "SPI.H"

#include "LCD\_HD44780.H"

#include "DELAY.H"

#include "LED7.H"

#include "DS1307.H"

#include "DS1820.H"

#include "AT24CXX.H"

#include "BEEP.H"

#include "I2C.H"

#include "SHIFT.H"

/\*

Задача №1

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

2) При нажатии на любую кнопку клавиатуры стенда соответствующий символ выводится на ЖКИ стенда

и семисегментный индикатор. Зуммер дает 3 сигнала длительностью 300 мс с паузой 100 мс,

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

дли-тельностью 300 мс с паузой 100 мс.

3) Реализовать удаление последнего введенного сим-вола на экране ЖКИ кнопкой “\*”, а на

семисегментном индикаторе при этом должна выводиться цифра, стоящая последней перед удаленной.

\*/

#define CLOCK 22110000

#define TIMER\_FREQ\_HZ 1000

#define RELOAD\_VALUE (65536 - (CLOCK / 12 / TIMER\_FREQ\_HZ))

#define RELOAD\_VALUE\_H ((unsigned char)(RELOAD\_VALUE >> 8))

#define RELOAD\_VALUE\_L ((unsigned char)(RELOAD\_VALUE))

#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

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

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

char curr\_key\_name = 0;      // значение кнопки по ключу key

char prev\_key\_name = 0;      // значение кнопки по ключу prev\_key

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(void)

{// инициализация таймера

    TMOD = 0x01;

    TL0 = RELOAD\_VALUE\_L;

    TH0 = RELOAD\_VALUE\_H;

    ET0 = 1;

    TR0 = 1;

}

void init\_UART()

{// инициализация UART

    SCON = 0x50; // 8-битовый UART

    TMOD = 0x20; // Таймер 1: 8-битовый режим, авто-перезагружаемый

    TH1 = 250;   // Задаем начальное значение таймера для скорости 9600 бод

    TL1 = 250;

    TR1 = 1; // Запускаем таймер

    TI = 1;

    ES = 1; // Разрешаем прерывание от UART

}

void change\_key\_handler()

{// обработчик изменения нажатой клавиши клавиатуры

    beep\_ctr = ms\_ctr;

    curr\_signal\_dur\_idx = 0;

    is\_beep\_seq\_started = TRUE;

    is\_need\_delay = FALSE;

    //отчистка дисплеев

    LCD\_clrscr();

    SHIFT\_write(0);

    if (key == KEY\_ASTERISK) {

        LED7\_setdigit( prev\_key\_name );

    }

    else {

        LED7\_setdigit( curr\_key\_name );

        LCD\_print( curr\_key\_name );

    }

    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\_beep\_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\_counter;

            return;

        }

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

            button\_read\_flag = FALSE;

        else

            return;

        key\_name = KEY\_getkeynumber(key); // получаем цифру по адресу клавиши

        change\_key\_handler();

        prev\_key = key;

        prev\_key\_name = KEY\_getkeyname(prev\_key);

    }

}

void ISR(void) interrupt 1 using 0

{// обработка прерывания

    if (ms\_ctr % 10 == 0)

        button\_handler();

    if (ms\_ctr % 100 == 0)

        beep\_handler();

    ms\_ctr++;

}

void main() // Основная функция

{

    // Иниц LED, LCD, SPI, I2C, прерывания разрешены

    P0 = 0xFF;

    P1 = 0xFF;

    P2 = 0xFF;

    P3 = 0xFF;

    EA = 1;

    SCON  = 0x50;

    TMOD  = 0x20;

    TH1   = 250;

    TL1   = 250;

    TR1   = 1;

    TI    = 1;

    ES=1;

    init\_timer0();

    init\_UART();

    SPI\_init();

    I2C\_init();

    LCD\_init();

    printf("Task\_1");

    while (1)

    {

    }

}

**Задача 2**

#include <stdio.h>

#include <intrins.h>

#include <string.h>

#include "KEYBOARD.H"

#include "SPI.H"

#include "LCD\_HD44780.H"

#include "DELAY.H"

#include "LED7.H"

#include "DS1307.H"

#include "DS1820.H"

#include "AT24CXX.H"

#include "BEEP.H"

#include "I2C.H"

#include "SHIFT.H"

/\*

1) При запуске программы на экран терминала выводится сообщение «Задача №2».

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

кото-рые затем устанавливаются как текущие в RTC DS1307.

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

во второй дата.

4) При нажатии клавиши 3 происходит очистка ЖКИ.

Нажатие на другие клавиши игнорируется.

\*/

#define CLOCK 22110000

#define TIMER\_FREQ\_HZ 1000

#define RELOAD\_VALUE (65536 - (CLOCK / 12 / TIMER\_FREQ\_HZ))

#define RELOAD\_VALUE\_H ((unsigned char)(RELOAD\_VALUE >> 8))

#define RELOAD\_VALUE\_L ((unsigned char)(RELOAD\_VALUE))

int mode = 0;

unsigned char pc\_keyboard\_data = 0;

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

void init\_timer0(void)

{ // инициализация таймера

    TMOD = 0x01;

    TL0 = RELOAD\_VALUE\_L;

    TH0 = RELOAD\_VALUE\_H;

    ET0 = 1;

    TR0 = 1;

}

void init\_UART()

{// инициализация UART

    SCON = 0x50; // 8-битовый UART

    TMOD = 0x20; // Таймер 1: 8-битовый режим, авто-перезагружаемый

    TH1 = 250;   // Задаем начальное значение таймера для скорости 9600 бод

    TL1 = 250;

    TR1 = 1; // Запускаем таймер

    TI = 1;

    ES = 1; // Разрешаем прерывание от UART

}

void change\_time\_n\_date()

{ // установить новое время в микросхему DS1307 через UART

    unsigned char hour, minute, second;

    unsigned char day, month, year;

    printf("Enter hour: ");

    scanf("%bu", &hour);

    printf("Enter minute: ");

    scanf("%bu", &minute);

    printf("Enter second: ");

    scanf("%bu", &second);

    DS1307\_settime(hour, minute, second);

    printf("Enter day: ");

    scanf("%bu", &day);

    printf("Enter month: ");

    scanf("%bu", &month);

    printf("Enter year: ");

    scanf("%bu", &year);

    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

    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);

}

void Serial\_ISR(void) interrupt 4 using 0

{// обработчик прерываний от UART

    if (RI)

    {

        scanf("%bu", &pc\_keyboard\_data);

        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();

        }

    }

}

void ISR(void) interrupt 1 using 0

{ // обработка прерывания милисекундного счётчика

    if (mode == 1)

        change\_time\_n\_date();

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

        display\_time\_n\_date();

    ms\_ctr++;

}

void main()

{

    // Иниц LED, LCD, SPI, I2C, прерывания разрешены

    P0 = 0xFF;

    P1 = 0xFF;

    P2 = 0xFF;

    P3 = 0xFF;

    EA = 1;

    SCON = 0x50;

    TMOD = 0x20;

    TH1 = 250;

    TL1 = 250;

    TR1 = 1;

    TI = 1;

    ES = 1;

    init\_timer0();

    init\_UART();

    SPI\_init();

    I2C\_init();

    LCD\_init();

    printf("Task\_2");

    while (1)

    {

    }

}