МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»

(БГТУ им. В.Г.Шухова)

**Лабораторная работа №3**

дисциплина «ЭВМ и периферийные устройства»

тема: «Изучение принципов организации обмена данными по последовательному интерфейсу i2c на примере управления блоком светодиодов и программного опроса клавиатуры»

Выполнил: студент группы ВТ-31

Ковалёв И. Д.

Проверил: Шамраев А.А.

Белгород 2020

**Цель работы:**

изучить принципы программного управления двунаправленным обменом данных по последовательному интерфейсу I2C.

**Задания к работе:**

Разработать в среде программирования IAR Embedded Workbench программу на языке С, которая выполняет опрос клавиатуры лабораторного стенда и выводит информацию о нажатых клавишах с помощью блока светодиодов.  
**Порядок выполнения задания:**

– включить лабораторный макет.

– запустить компилятор IAR Embedded Workbench.

– создать пустой проект.

– создать файл ресурса для кода программы и подключить его к проекту.

– ввести код исходного модуля программы обмена данными между микроконтроллером MSP430F1611 с регистрами PCA9538 по интерфейсу I2C соответствие с индивидуальным заданием, приведенным в таблице 4.3.8.

– выполнить компиляцию исходного модуля программы и устранить ошибки, полученные на данном этапе.

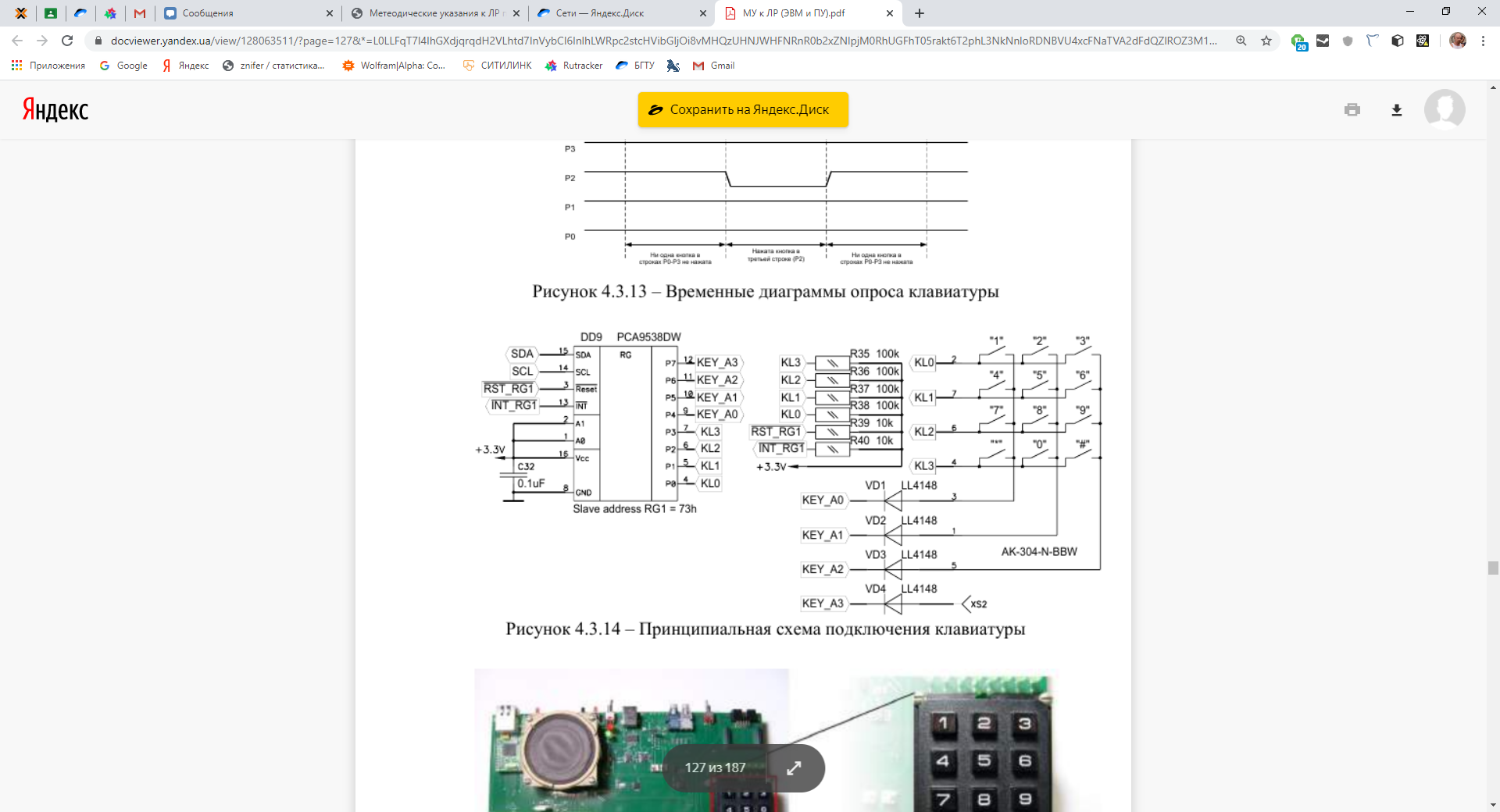
– настроить параметры программатора.

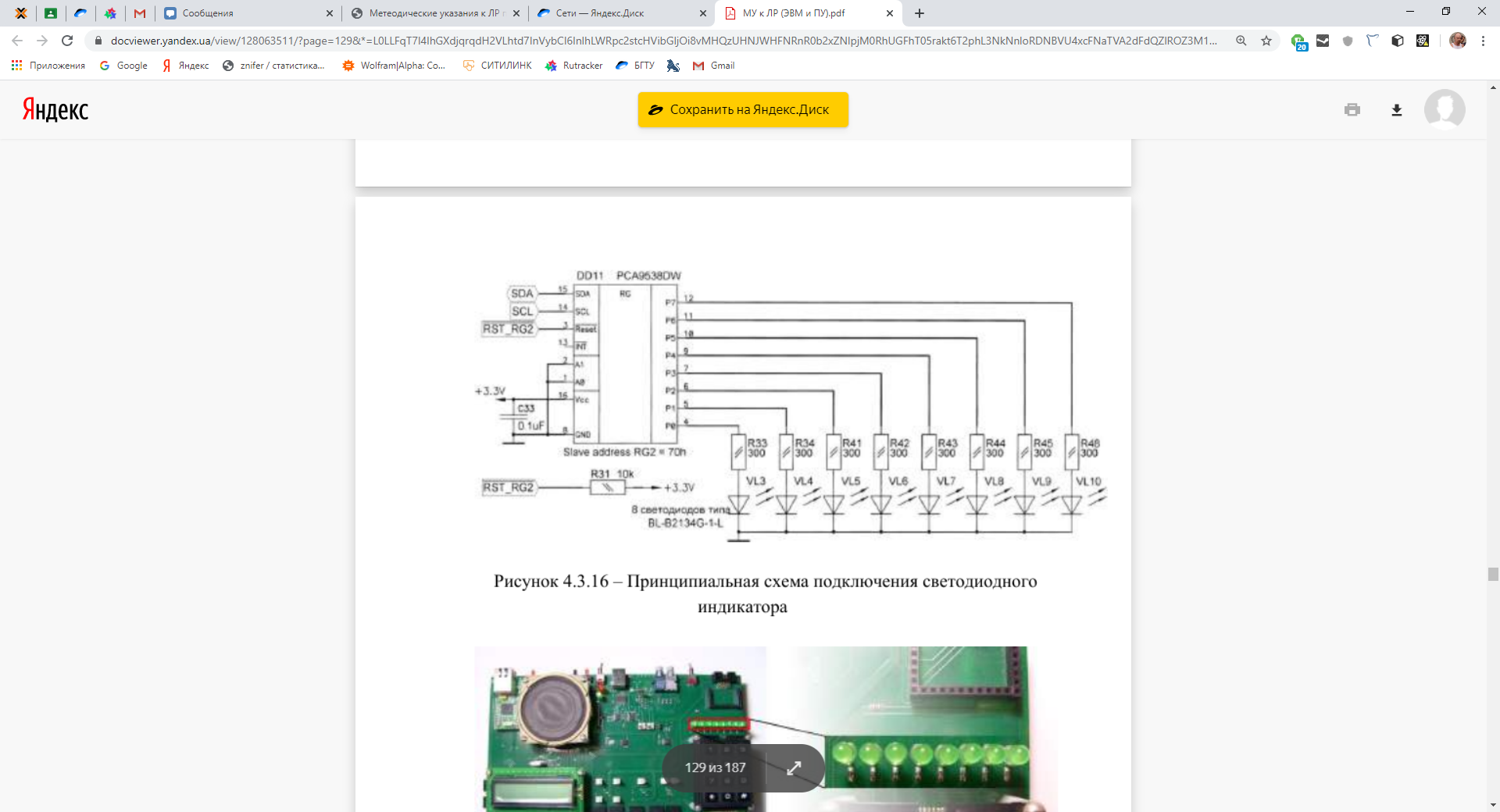
– создать загрузочный модуль программы и выполнить программирование микроконтроллера. Проверить работоспособность загруженной в микроконтроллер программы и показать результаты работы преподавателю. В случае некорректной работы разработанной программы, выполнить аппаратный сброс микроконтроллера, провести отладку исходного модуля программы и заново проверить функционирование программы

**Задание варианта №7:**

Разработать программу, фиксирующую нажатия клавиш 3, 7 и 0 матричной клавиатуры включением светодиодов 1, 2 и 3 соответственно. Выход из цикла опроса осуществляется при нажатии клавиши #. Частота тактовых импульсов на линии SCL – 70 кгц.

**Вывод:** в процессе выполнения лабораторной работы были изучены принципы программного управления двунаправленным обменом данных по последовательному интерфейсу I2C на примере матричной клавиатуры и светодиодных индикаторов.





**Листинг разработанной программы:**

**Main.c:**

#include <msp430.h>

#include "system\_define.h"

#include "system\_variable.h"

#include "function\_prototype.h"

#include "main.h"

/\*

Разработать программу, фиксирующую нажатия клавиш 3, 7 и 0 матричной

клавиатуры включением светодиодов 1, 2 и 3 соответственно. Выход из

цикла опроса осуществляется при нажатии клавиши #. Частота тактовых

импульсов на линии SCL – 70 кГц.

\*/

char interrupted = 0;

#pragma vector=PORT1\_VECTOR

\_\_interrupt void ISR\_p1int(void){

P1IFG &= ~BIT7;

interrupted = 1;

}

void interrupt\_ready(){

I2C\_WriteByte(0x03, 0x0F, 0x73);

I2C\_WriteByte(0x01, 0x0F, 0x73);

}

void poll\_ready(){

I2C\_WriteByte(0x03, 0x0F, 0x73);

I2C\_WriteByte(0x01, 0xFF, 0x73);

}

// Инициализация модуля UART0 для работы в режиме I2C

void Init\_I2C\_70hz(){

P3SEL |= 0x0A; // Выбор альтернативной функции для линий порта P3

// в режиме I2C SDA->P3.1, SCL->P3.3

U0CTL |= I2C + SYNC; // Выбрать режим I2C для USART0

U0CTL &= ~I2CEN; // Выключить модуль I2C

// Конфигурация модуля I2C

//f\_scl = fI2C / (I2CPSC + 1) \* (I2CSCLH + 2 + I2CSCLL + 2)

I2CTCTL=I2CSSEL\_2; // SMCLK

I2CSCLH = 0x21; // High period of SCL

I2CSCLL = 0x21; // Low period of SCL

U0CTL |= I2CEN; // Включить модуль I2C

// формирование строба сброса I2C-регистров PCA9538 - RST\_RG1->P3.1 и RST\_RG2->P3.2

P3DIR |= 0x05; // переключаем эти ножки порта на вывод,

P3SEL &= ~0x05; // выбираем функцию ввода-вывода для них

P3OUT &= ~0x05; // и формируем строб сброса на 1 мс

wait\_1ms(1);

P3OUT |= 0x05;

}

void main(void) {

WDTCTL = WDTPW + WDTHOLD;

//P1DIR &= ~BIT7;

P1IE |= BIT7;

P1IFG &= ~BIT7;

P1IES |= BIT7;

\_enable\_interrupt();

Init\_System\_Clock();

Init\_System();

Init\_I2C();

interrupt\_ready();

char symbol = 0, octPressed = 0, i = 0;

while (1){

if(interrupted){

// Отключим на время?

P1IE &= ~BIT7;

interrupted = 0;

symbol = KEYS\_scannow();

if (!octPressed){

switch (symbol){

case '3':

LED\_set(1);

wait\_1ms(100);

LED\_reset(1);

break;

case '7':

LED\_set(2);

wait\_1ms(100);

LED\_reset(2);

break;

case '0':

LED\_set(3);

wait\_1ms(100);

LED\_reset(3);

break;

case '#':

octPressed = 1;

break;

}

}

interrupt\_ready();

// Включим на время

P1IE |= BIT7;

}

}

}

**Содержимое файла i2c.c:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// I2C function

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "function\_prototype.h"

#include "system\_define.h"

#include "I2C.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//============================================================================

// Инициализация модуля UART0 для работы в режиме I2C

void Init\_I2C()

{

P3SEL |= 0x0A; // Выбор альтернативной функции для линий порта P3

// в режиме I2C SDA->P3.1, SCL->P3.3

U0CTL |= I2C + SYNC; // Выбрать режим I2C для USART0

U0CTL &= ~I2CEN; // Выключить модуль I2C

// Конфигурация модуля I2C

I2CTCTL=I2CSSEL\_2; // SMCLK

I2CSCLH = 0x26; // High period of SCL

I2CSCLL = 0x26; // Low period of SCL

U0CTL |= I2CEN; // Включить модуль I2C

// формирование строба сброса I2C-регистров PCA9538 - RST\_RG1->P3.1 и RST\_RG2->P3.2

P3DIR |= 0x05; // переключаем эти ножки порта на вывод,

P3SEL &= ~0x05; // выбираем функцию ввода-вывода для них

P3OUT &= ~0x05; // и формируем строб сброса на 1 мс

wait\_1ms(1);

P3OUT |= 0x05;

}

//============================================================================

//============================================================================

// отправка данных по протоколу I2C

void Send\_I2C(unsigned char\* buffer,unsigned int num, unsigned char address)

{

while (I2CBUSY & I2CDCTL); // проверка готовности модуля I2C

BufTptr=buffer;

I2CSA = address; // установка адреса приемнмка

I2CNDAT =num; // количество передаваемых байт

I2CIE = TXRDYIE+ALIE; // разрешение прерываний по окончанию передачи байта и по потере арбитража

U0CTL |= MST; // режим Master

I2CTCTL |= I2CSTT + I2CSTP + I2CTRX; // инициализировать передачу

while ((I2CTCTL & I2CSTP) == 0x02); // ожидание условия СТОП

}

//============================================================================

//============================================================================

// прием данных по протоколу I2C

void Receive\_I2C(unsigned char\* buffer,unsigned int num, unsigned char address)

{

while (I2CBUSY & I2CDCTL); // проверка готовности модуля I2C

BufRptr=buffer;

I2CSA=address;

I2CTCTL&=~I2CTRX; // режим приема

I2CNDAT=num;

I2CIE=RXRDYIE; // резрешение прерывания по окончанию приема байта

U0CTL |= MST;

I2CTCTL |= I2CSTT + I2CSTP; // инициализировать прием

while ((I2CTCTL & I2CSTP) == 0x02); // ожидание условия СТОП

}

//============================================================================

//============================================================================

// отправка байта устройству на шине I2C

void I2C\_SendByte(char data, char i2c\_addr)

{

Tx\_Data[0] = data; // отправляемый байт

Send\_I2C(&Tx\_Data[0], 1, i2c\_addr); // вывод по I2C на устройство

}

//============================================================================

//============================================================================

// запись байта в регистр устройства на шине I2C

void I2C\_WriteByte(char reg, char data, char i2c\_addr)

{

Tx\_Data[0] = reg; // выбираем регистр

Tx\_Data[1] = data; // записываемые данные

Send\_I2C(&Tx\_Data[0], 2, i2c\_addr); // вывод по I2C на устройство

}

//============================================================================

//============================================================================

// чтение байта из регистра устройства на шине I2C

byte I2C\_ReadByte(char reg, char i2c\_addr)

{

Tx\_Data[0] = reg; // выбираем регистр

Send\_I2C(&Tx\_Data[0], 1, i2c\_addr);

Receive\_I2C(&Rx\_Data[0], 1, i2c\_addr); // получаем значение из регистра

return Rx\_Data[0];

}

//============================================================================

//============================================================================

// чтение слова (2 байта) из регистра устройства на шине I2C

int I2C\_ReadWord(char reg, char i2c\_addr)

{

Tx\_Data[0] = reg; // выбираем регистр

Send\_I2C(&Tx\_Data[0], 1, i2c\_addr);

Receive\_I2C(&Rx\_Data[0], 2, i2c\_addr); // получаем 2 байта значение из регистра

return Rx\_Data[0] + (Rx\_Data[1] \* 256);

}

//============================================================================

//============================================================================

//Обработка прерывания от модуля USART0, работающего в режиме I2C

// вектор прерываний для модуля I2C

#pragma vector=USART0TX\_VECTOR

\_\_interrupt void I2C\_ISR()

{

switch(I2CIV)

{

case 0: break; // нет прерывания

case 2: break; // потеря арбитража

case 4: break; // нет подтверждения

case 6: break; // прерывание собственного адреса

case 8: break; // регистр доступен для чтения

case 10: // окончание приема байта

\*BufRptr++=I2CDRB;

break;

case 12: // окончание передачи байта

I2CDRB=\*BufTptr++;

break;

case 14: break; // общий вызов

case 16: break; // обнаружено условие СТАРТ

default : break;

}

}

**Содержимое файла keys.c:**

// Keyboard functions

#include "function\_prototype.h"

#include "sysfunc.h"

#include "keys.h"

byte keycol, keyline, KEYS\_last=0;

char table\_keys[12] = {'1', '2', '3', '4', '5', '6', '7', '8', '9', '\*', '0', '#'};

// ѕроверка нажати¤ клавиши в текущий момент, результат:

// 0 - клавиша не нажата

// ASCII-код клавиши

char KEYS\_scannow()

{

keyline=0;

// выбираем регистр конфигурации направлени¤ (0x03)

// и конфигурируем P4-P7 на вывод - дл¤ строба столбцов,

// а P0-P3 на ввод - дл¤ опроса строк (1-ввод, 0-вывод)

I2C\_WriteByte(0x03, 0x0F, KEYS\_i2c\_addr);

for (keycol=0; keycol<3; keycol++) {

// последовательно подаем сигнал низкого уровн¤ на столбцы (P4-P7)

I2C\_WriteByte(0x01, ~(1<<keycol<<4) & 0xf0, KEYS\_i2c\_addr);

wait\_1ms(1);

// и опрашиваем строки (P0-P3) на наличие нул¤

keyline = ~(I2C\_ReadByte(0x00, KEYS\_i2c\_addr)) & 0x0f;

if (keyline) break;

}

if (!keyline) return 0; // если не была нажата никака¤ клавиша - возвращаем 0

if (keyline == 4) keyline = 3; // переводим номера разр¤дов в номер строки

if (keyline == 8) keyline = 4;

KEYS\_last = table\_keys[--keyline\*3+keycol]; // получаем код нажатой клавиши из таблицы

return KEYS\_last;

}

// ¬озвращает код последней нажатой клавиши, результат:

// 0 - не нажималась никака¤ клавиша

// ASCII-код клавиши

char KEYS\_lastkey()

{

KEYS\_scannow();

return KEYS\_last;

}

// ќчистка последней нажатой клавиши

void KEYS\_clear()

{

KEYS\_last = 0;

wait\_1ms(200);

}

// ќжидание нажати¤ клавиши, результат - ASCII-код нажатой клавиши

char KEYS\_waitkey()

{

KEYS\_clear(); // очистка последней нажатой клавиши

while (!KEYS\_scannow()) // пока не нажата никака¤ клавиша,

wait\_1ms(1); // сделать паузу

return KEYS\_last; // вернуть код нажатой клавиши

}

// пауза с циклическим опросом клавиатуры, прерываетс¤ если нажата клавиша

void KEYS\_pause(byte cnt)

{

byte i;

KEYS\_clear(); // очистка последней нажатой клавиши

for (i=0; i<cnt; i++)

if (KEYS\_scannow())

break;

}

**Содержимое файла leds.c:**

// LED-indicator functions

#include "function\_prototype.h"

#include "sysfunc.h"

#include "leds.h"

char LED\_config=0; // хранится конфигурация светодиодов (вкл/выкл)

void LED\_out(char leds)

{

// регистр конфигурации направления 0x03 конфигурируем на вывод информации (1-ввод, 0-вывод)

I2C\_WriteByte(0x03, 0x00, LED\_i2c\_addr);

I2C\_WriteByte(0x01, leds, LED\_i2c\_addr); // выводим данные в регистр OUTPUT (0x01)

LED\_config=leds; // сохраняем новую конфигурацию

}

// Преобразование номера светодиода в бит, с которым нужно проводить операцию

// 1 = 10000000, 2 = 01000000 ... 8 = 00000001

char LED\_convert(char led)

{

if(led<1)

led=1;

if(led>8)

led=8;

led=9-led;

return (1<<(led-1));

}

// Выключить все светодиоды

void LED\_clear()

{

LED\_out(0x00);

}

// Инвертировать все светодиоды

void LED\_invert()

{

LED\_out(LED\_config ^ 0xff);

}

// Включить светодиод с номером от 1 до 8 (слева направо)

void LED\_set(char led)

{

led=LED\_convert(led);

LED\_config |= led; // устанавливаем соответствующий разряд

LED\_out(LED\_config); // выводим в регистр

}

// Выключить светодиод с номером от 1 до 8 (слева направо)

void LED\_reset(char led)

{

led=LED\_convert(led);

LED\_config &= ~(led); // сбрасываем соответствующий разряд

LED\_out(LED\_config); // выводим в регистр

}

// Сменить состояние светодиода с номером от 1 до 8 (слева направо)

void LED\_change(byte led)

{

led=LED\_convert(led);

LED\_config ^= led; // меняем состояние соответствующего разряда (XOR)

LED\_out(LED\_config); // выводим в регистр

}