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

Выполнил: студент группы ВТ-31 Макаров Д.С.
Проверил: Шамраев А.А.

Белгород 2020

Лабораторная работа №3

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

Цель работы:Изучить принципы програмного управления двунаправленным обменом данных по последовательному интерфейсу I2C.

Вариант 6

Задание: Разработать программу, фиксирующую нажатия клавиш 4,8,9 матричной клавиотуры включением светодиодов 1,2,3 соответственно. Выход из цикла осуществить клавишей *. Частота импульсов на линии SCL - 60 кГц.

- модуль UASRT0
- скорость 14400 бит/с
- режим обмена - асинхронный
- 7 битов без бита четности

Порядок выполнения задания:

- включить лабораторный макет.
- запустить Code Composer IDE.
- создать пустой проект.
- создать файл ресурса для кода программы и подключить его к проекту.
- выполнить компиляцию исходного модуля программы и устранить ошибки, полученные на данном этапе.
- проверить работоспособность программы и показать результаты работы преподавателю.

Ход работы

Схема стенда

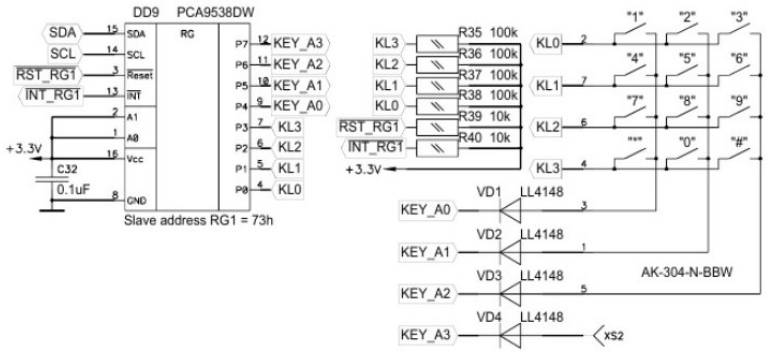

Рис. 1: Схема подключения клавиотуры

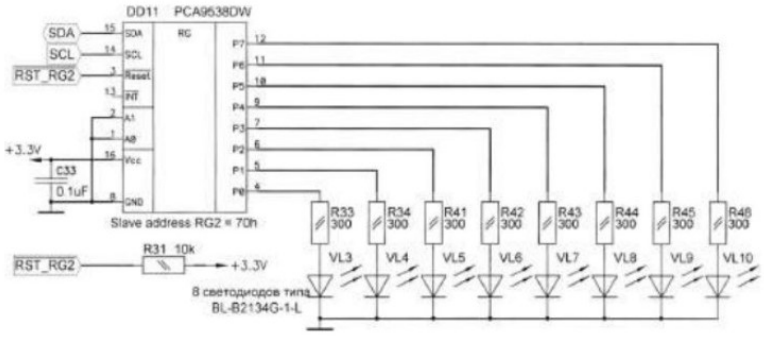

Рис. 2: Схема подключения светодиодов

Вывод: Я изучил принцип работы двухстороннего протокола обмена данных I2C, его реализацию в микроконтроллерах семейства MSP430, а так же принцип работы порта расширения PCA9538.

Приложение

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', '#'};

// Проверка нажатия клавиши в текущий момент, результат:
//  0 - клавиша не нажата
//  ASCII-код клавиши
char KEYS_scannow()
{
  keyline=0;
  // выбираем регистр конфигурации направления (0x03)
  //  и конфигурируем P4-P7 на вывод - для строба столбцов,
  //  а P0-P3 на ввод - для опроса строк (1-ввод, 0-вывод)
  I2C_WriteByte(0x03, 0x0F, KEYS_i2c_addr);

  for (keycol=0; keycol<3; keycol++) {
  // последовательно подаем сигнал низкого уровня на столбцы (P4-P7)
    I2C_WriteByte(0x01, ~(1<<keycol<<4) & 0xf0, KEYS_i2c_addr);
    wait_1ms(1);
    // и опрашиваем строки (P0-P3) на наличие нуля
    keyline = ~(I2C_ReadByte(0x00, KEYS_i2c_addr)) & 0x0f;
    if (keyline) break;
  }
  if (!keyline) return 0;       // если не была нажата никакая клавиша - возвращаем 0
  if (keyline == 4) keyline = 3;  // переводим номера разрядов в номер строки
  if (keyline == 8) keyline = 4;
```

```
  KEYS_last = table_keys[--keyline*3+keycol]; // получаем код нажатой клавиши из таблицы
  return KEYS_last;
}


// Возвращает код последней нажатой клавиши, результат:
//  0 - не нажималась никакая клавиша
//  ASCII-код клавиши
char KEYS_lastkey()
{
  KEYS_scannow();
  return KEYS_last;
}


// Очистка последней нажатой клавиши
void KEYS_clear()
{
  KEYS_last = 0;
  wait_1ms(200);
}

// Ожидание нажатия клавиши, результат - ASCII-код нажатой клавиши
char KEYS_waitkey()
{
  KEYS_clear();           // очистка последней нажатой клавиши
  while (!KEYS_scannow()) // пока не нажата никакая клавиша,
    wait_1ms(1);          //    сделать паузу
  return KEYS_last;       // вернуть код нажатой клавиши
}


// пауза с циклическим опросом клавиатуры, прерывается 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
}


void LED_fx1(int n)
{
  LED_clear();
  LED_set(1);
  wait_1ms(n);
```

```
  LED_set(3);
  wait_1ms(n);
  LED_set(5);
  wait_1ms(n);
  LED_set(7);
  wait_1ms(n);
  LED_reset(1);
  wait_1ms(n);
  LED_reset(3);
  wait_1ms(n);
  LED_reset(5);
  wait_1ms(n);
  LED_reset(7);
  wait_1ms(n);
  LED_clear();
}


void LED_fx2(int n)
{
  LED_clear();
  LED_out(0x81);
  wait_1ms(n);
  LED_out(0x42);
  wait_1ms(n*8);
  LED_out(0x24);
  wait_1ms(n*5);
  LED_out(0x18);
  wait_1ms(n*3);

  LED_out(0x24);
  wait_1ms(n*5);
  LED_out(0x42);
  wait_1ms(n*8);
  LED_clear();
}


void LED_fx3(int n)
{
  LED_clear();
  LED_out(0x01);
  wait_1ms(n*6);
  LED_out(0x02);
  wait_1ms(n*4);
  LED_out(0x04);
  wait_1ms(n*2);
  LED_out(0x08);
  wait_1ms(n*1);
  LED_out(0x10);
  wait_1ms(n*1);
  LED_out(0x20);
  wait_1ms(n*2);
  LED_out(0x40);
  wait_1ms(n*4);
  LED_out(0x80);
  wait_1ms(n*6);

  LED_out(0x40);
  wait_1ms(n*4);
```

```
  LED_out(0x20);
  wait_1ms(n*2);
  LED_out(0x10);
  wait_1ms(n*1);
  LED_out(0x08);
  wait_1ms(n*1);
  LED_out(0x04);
  wait_1ms(n*2);
  LED_out(0x02);
  wait_1ms(n*4);
  LED_clear();
}
```

Содержимое файла main.c

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"
#include "main.h"
#include "keys.h"

#define KEYS_i2c_addr 0x73
/*
 * main.c
 */
char state = 0;
char key_press = 0;
char key = 0;
int delay = 0;

void Custom_Speed_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
  // нужно установить 60000 Гц
  I2CPSC = 2; //пределитель тактовой частоты
  I2CSCLH = 21;          // High period of SCL
  I2CSCLL = 21;          // 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 reset_keyboard(){
    I2C_WriteByte(0x03,0x0F,KEYS_i2c_addr);
    I2C_WriteByte(0x01,0x0F,KEYS_i2c_addr);
};

void set_keyboard(){
    I2C_WriteByte(0x03,0x0F,KEYS_i2c_addr);
    I2C_WriteByte(0x01,0xFF,KEYS_i2c_addr);
};
```

```
void scan_key(){
    key_press = 0;
    key = KEYS_scannow();
            switch(key){
                case '1':
                    delay-=15;
                    break;
                case '2':
                    delay+=15;
                    break;
                case '0':
                    LED_clear();
                    state = 1;
                    break;
                case '*':
                    LED_clear();
                    state = 0;
                    break;
                case '#':
                    LED_clear();
                    state = 2;
                    break;
            }
            reset_keyboard();
};
void main(void) {
    WDTCTL = WDTPW + WDTHOLD;
    Init_System_Clock();
    P1IE |= BIT7;
    P1IFG &= !BIT7;
    P1IES |= BIT7;
    __enable_interrupt();

    Init_I2C();
    LED_clear();
    reset_keyboard();
    while(1){
        if(key_press){
            scan_key();
        }
        switch(state){
            case 0:
                LED_fx1(delay);
                break;
            case 1:
                LED_fx2(delay);
                break;
            case 2:
                LED_fx3(delay);
                break;
    };
}

#pragma vector=PORT1_VECTOR
__interrupt void keyboard_interrupt(void)
{
    key_press = 1;
    P1IFG &= ~BIT7;
}
```

Содержимое файла sysfunc.c

```
// System functions

#include <msp430.h>
#include "sysfunc.h"


// инициализация портов системы
void Init_System()
{
        P1DIR |= (nSS + nWR_nRST + MCU_SEL_0 + MCU_SEL_1); // установка направления портов на
        ↪  вывод
        DB_DIR = 0x00; // шина данных настроена на ввод
}


// инициализация системы тактирования
void Init_System_Clock()
{
        volatile byte i;
        BCSCTL1 &= ~XT2OFF;                         // включение осцилятора XT2
        BCSCTL2 |= DIVS1;
        BCSCTL2 |= DIVS0;
                                                    // MCLK = XT2, SMCLK = XT2
        do                                          // ожидание запуска кварца
        {
                IFG1 &= ~OFIFG;                         // Clear OSCFault flag
                for (i = 0xFF; i > 0; i--);             // Time for flag to set
        }
        while ((IFG1 & OFIFG));                     // OSCFault flag still set?
        BCSCTL2 |= SELM_2 | SELS;                   // установка внешнего модуля тактирования
}

// 2do: сделать точную задержку
void wait_1ms(word cnt)
{
  for (wait_i = 0; wait_i < cnt; wait_i++)
    for (wait_j = 0; wait_j < 1000; wait_j++);
}


void wait_1mks(word cnt)
{
  for (wait_i = 0; wait_i < cnt; wait_i++);
}
```