

**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 №5
дисциплина «ЭВМ и периферийные устройства»
по теме «Изучение принципов работы со встроенным в микроконтроллер
аналого-цифровым преобразователем на примере измерения относительной
влажности воздуха и потребляемого стендом тока»

Выполнил: студент группы ВТ-31

Проверил:

Макаров Д.С.

Шамраев А.А.

Белгород 2020

Лабораторная работа №5

«Изучение принципов работы со встроенным в микроконтроллер аналого-цифровым преобразователем на примере измерения относительной влажности воздуха и потребляемого стендом тока»

Цель работы:изучить принципы функционирования встроенного в микроконтроллер MSP430F1611 АЦП и методику измерения относительной влажности и потребляемого тока с помощью датчиков влажности и тока.

Вариант 6

Задание: Разработать программу, выполняющую измерение потребляемого тока в режиме одиночного преобразования (делитель частоты равен 7) и отображающую результат измерений на ЖКИ.

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

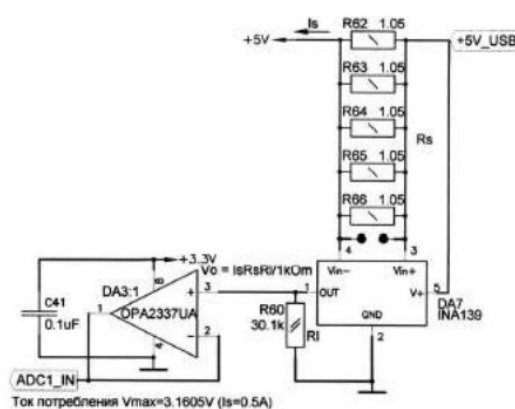


Рис. 1: Схема подключения датчика потребления тока

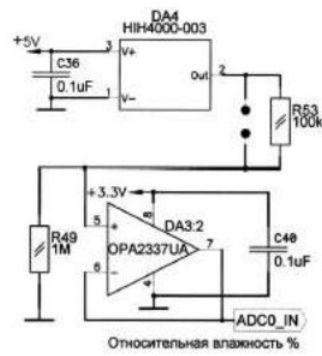


Рис. 2: Схема подключения датчика влажности воздуха

Вывод: Я изучил принцип работы модуля АЦП12 в микроконтроллерах семейства **MSP430F16xx**, а так же научился снимать показания с датчиков влажности **HIN4000-003** и датчиков потребления тока **INA139**.

Приложение

Содержимое файла lcd.c

```
// LCD-display functions
#include "function_prototype.h"
#include "sysfunc.h"
#include "lcd.h"

//Таблица кириллицы
char LCD_table[64]={
    0x41,0xA0,0x42,0xA1,    //0xC0...0xC3 <=> А Б В Г
    0xE0,0x45,0xA3,0x33,    //0xC4...0xC7 <=> Д Е Ж З
    0xA5,0xA6,0x4B,0xA7,    //0xC8...0xCB <=> И Й К Л
    0x4D,0x48,0x4F,0xA8,    //0xCC...0xCF <=> М Н О П

    0x50,0x43,0x54,0xA9,    //0xD0...0xD4 <=> Р С Т У
    0xAA,0x58,0xE1,0xAB,    //0xD5...0xD7 <=> Ф Х Ц Ч
    0xAC,0xE2,0xAC,0xAE,    //0xD8...0xDB <=> Ш Щ Ъ Ы
    0x62,0xAF,0xB0,0xB1,    //0xDC...0xDF <=> Ь Э Ю Я

    0x61,0xB2,0xB3,0xB4,    //0xE0...0xE4 <=> а б в г
    0xE3,0x65,0xB6,0xB7,    //0xE5...0xE7 <=> д е ж з
    0xB8,0xA6,0xBA,0xBB,    //0xE8...0xEB <=> и й к л
    0xBC,0xBD,0x6F,0xBE,    //0xEC...0xEF <=> м н о п

    0x70,0x63,0xBF,0x79,    //0xF0...0xF4 <=> р с т у
    0xE4,0xD5,0xE5,0xC0,    //0xF5...0xF7 <=> ф х ц ч
    0xC1,0xE6,0xC2,0xC3,    //0xF8...0xEB <=> ш щ ъ ы
    0XC4,0xC5,0xC6,0xC7    //0xFC...0xEF <=> ь э ю я
};

byte LCD_row, LCD_col, n;

void LCD_init()
{
    wait_1ms(20);    // пауза 20 мс после включения модуля
    P3DIR |= (D_nC_LCD + EN_LCD); // Настроить порты, которые управляют LCD на вывод
    Reset_EN_LCD(); // Перевести сигнал "Разрешение обращений к модулю LCD" в неактивное
    ↪ состояние

    // Команда Function Set      0 0 1 DL N F * *
    // установка разрядности интерфейса DL=1 =>8, бит DL=0 =>4 бит
    // N=1 => две строки символов, N=0 => одна строка символов
    // F=0 => размер шрифта 5x11 точек, F=1 => размер шрифта 5x8 точек
    // Выбор режима передачи команд для LCD и вывод байта без ожидания броса влага BF
    LCD_WriteCommand(0x3C);
    wait_1ms(1);

    LCD_WriteCommand(0x3C);
    wait_1ms(1);

    // Команда Display ON/OFF control 0 0 0 0 1 D C B
    // включает модуль D=1 и выбирает тип курсора (C,D)
    // C=0, B=0 - курсора нет, ничего не мигает
    // C=0, B=1 - курсора нет, мигает весь символ в позиции курсора
    // C=1, B=0 - курсора есть (подчеркивание), ничего не мигает
    // C=1, B=1 - курсора есть (подчеркивание), и только он и мигает
    LCD_WriteCommand(0x0C);
```

```

LCD_clear();

// Команда Entry Mode Set      0 0 0 0 0 1 ID SH
// установка направления сдвига курсора ID=0/1 - сдвиг влево/вправо
// и разрешение сдвига дисплея SH=1 при записи в DDRAM
LCD_WriteCommand(0x06);
}

//Вывод сообщение на LCD дисплей
void LCD_message(const char * buf)
{
    n = 0;
    while (buf[n])
    {
        // если выходим за границу строки - переход на следующую
        if ( (LCD_row < LCD_MAXROWS-1) && (LCD_col >= LCD_MAXCOLS) )
            LCD_set_pos(++LCD_row, 0);
        if (LCD_col >= LCD_MAXCOLS )
            LCD_set_pos(0,0); // если вышли за границы экрана - начинаем с начала
        // break; // или если вышли за границы экрана - перестаем выводить символы
        LCD_WriteData( LCD_recode(buf[n]) );
        LCD_col++;
        n++;
    }
}

// Функция очистки экрана
void LCD_clear()
{
    // Команда Clear Display      0 0 0 0 0 0 0 1
    // очищает модуль и помещает курсор в самую левую позицию
    LCD_WriteCommand(0x01);
    LCD_row=0;
    LCD_col=0;
}

// Установка позиции курсора:
// row - номер строки (0...1)
// col - номер столбца (0...15)
void LCD_set_pos(byte row, byte col)
{
    if (row > LCD_MAXROWS-1) // проверка на неправильные значения
        row = LCD_MAXROWS-1;
    if (col > LCD_MAXCOLS-1) // проверка на неправильные значения
        col = LCD_MAXCOLS-1;
    LCD_row = row;
    LCD_col = col;
    LCD_WriteCommand( BIT7 | ((0x40 * LCD_row) + LCD_col) );
}

byte LCD_get_row()
{
    return LCD_row;
}

```

```

byte LCD_get_col()
{
    return LCD_col;
}

// Установка режима отображения курсора:
// 0 - курсора нет, ничего не мигает
// 1 - курсора нет, мигает весь символ в позиции курсора
// 2 - курсор есть(подчеркивание), ничего не мигает
// 3 - курсор есть(подчеркивание) и только он мигает
void LCD_set_cursor(byte cursor)
{
    if (cursor > 3)           // проверка на неправильные значения
        cursor = 2;
    LCD_WriteCommand(cursor | BIT2 | BIT3);    // Выполняем команду Display ON/OFF Control
                                                // с нужным режимом отображения курсора
}

void LCD_WriteCommand(char byte)
{
    // Выбор режима передачи команд для LCD и вывод байта
    LCD_WriteByte(byte, 0);    //
}

void LCD_WriteData(char byte)
{
    // Выбор режима передачи данных LCD и вывод байта
    LCD_WriteByte(byte, 1);
}

// Вывод байта на индикатор, параметры:
// byte - выводимый байт
// dnc=0 - режим передачи команд, dnc=1 - данных
void LCD_WriteByte(char byte, char D_nC)
{
    DB_DIR = 0x00;           // Шина данных на прием
    Set_MCU_SEL_0();         // Выбор модуля LCD                               MCU_SEL_0 = 1
    Set_MCU_SEL_1();         // при помощи дешифратора DD7                     MCU_SEL_0 = 1

    ↪ //
    Reset_D_nC_LCD();        // Выбор режима передачи команд для LCD D/C_LCD = 0
    ↪ // --
    ↪ ---
    Set_nWR_nRST();          // Сигнал WR/RST = 1 => сигнал R/W_LCD = 1, т.е. в неактивном
    ↪ состоянии

    ↪ //
    Reset_nSS();             // Сформировать сигнал "OE_BF_LCD" SS = 0

    ↪ //
    Set_EN_LCD();            // Сформировать строб данных для LCD EN_LCD =
    ↪ 1                       |
    Set_EN_LCD();            // Сформировать строб данных для LCD EN_LCD =
    ↪ 1                       |

```

```

Set_EN_LCD();          // Сформировать строб данных для LCD  EN_LCD =
↪ 1      ----/      /-----

while (DB_IN & BIT7); // ожидание сброса флага занятости BUSY
Reset_EN_LCD();        // Перевести сигнал "EN_LCD_OUT" в неактивное состояние EN_LCD = 0

                                ↪      //
                                ↪  --

Set_nSS();              // Перевести сигнал "OE_BF_LCD" в неактивное состояние SS = 1

if (D_nC) Set_D_nC_LCD(); // Выбрать режим записи данных (D_nC = 1)
else Reset_D_nC_LCD();    // или записи команды (D_nC = 0)

                                ↪      //
                                ↪  -- ---- -

Reset_nWR_nRST();       // Сформировать сигнал WR/RST = 0 => R/W_LCD = 0

                                ↪      //
                                ↪  --

Reset_nSS();            // Сформировать сигнал "OE_BF_LCD" SS = 0
DB_DIR = 0xFF;          // Шина данных на выход
DB_OUT = byte;          // Выставить данные на шину данных

                                ↪      //

Set_EN_LCD();           // Сформировать строб данных для LCD  EN_LCD =
↪ 1      |      |
Set_EN_LCD();           // Сформировать строб данных для LCD  EN_LCD =
↪ 1      |      |
Set_EN_LCD();           // Сформировать строб данных для LCD  EN_LCD =
↪ 1      ----/      /-----
Reset_EN_LCD();         // Перевести сигнал "EN_LCD_OUT" в неактивное состояние EN_LCD = 0

                                ↪      //
                                ↪  --

Set_nSS();              // Перевести сигнал OE_BF_LCD =1 в неактивное состояние SS = 1
DB_DIR = 0x00;          // Шина данных на вход

                                //      --
                                ↪  ---
                                ↪  -

Set_nWR_nRST();         // Сигнал WR/RST = 1 => сигнал R/W_LCD = 1, т.е. в неактивном
↪ состоянии

}

//Функция перекодировки символа в кириллицу
char LCD_recode(char b)
{
    if (b<192) return b;
    else return LCD_table[b-192];
}

```

Содержимое файла main.c

```

#include <msp430.h>
#include "stdio.h"
#include "system_define.h"
#include "system_variable.h"

```

```

#include "function_prototype.h"
#include "main.h"

const float HIH_zero_offset = 0.958;
const float HIH_slope = 0.03068;
const float HIH_ion = 3.3;
const float HIH_divisor = 1.1;
const float INA_RS = 0.21;
const float INA_RL = 30.1;

float current_meansurment, humidity_meansurment;
char msg_str[16];

void init_uninterrupted_meansurement(){
    P6SEL |= BIT0 + BIT1;
    ADC12CTL0 = ADC12ON;
    ADC12CTL1 = SHS_1 + SHP + CSTARTADD_0 + ADC12DIV1 + ADC12DIV2 + CONSEQ0 + CONSEQ1;
    ADC12MCTL0 = SREF_3 + INCH_0;
    ADC12MCTL1 = SREF_3 + INCH_1 + EOS;
    ADC12IE = 2;
    ADC12CTL0 |= ENC;

    TBCTL = TBSSEL0 + MCO;
    TBCCTL0 = OUTMOD_2;
    TBCCR0 = 164;
};

#pragma vector = ADC12_VECTOR
__interrupt void adc12_interupt(void)
{
    if (ADC12IFG & BIT1 == 1){
        humidity_meansurment = (((ADC12MEM0/4095.0)* HIH_ion * HIH_divisor)-HIH_zero_offset) /
        ↪ HIH_slope;
        current_meansurment = (ADC12MEM1*3.3) / (4095.0 * INA_RS * INA_RL);
        LCD_set_pos(0,0);
        sprintf(msg_str,"Влажность: %5.3f",humidity_meansurment);
        LCD_message(msg_str);
        LCD_set_pos(1,0);
        sprintf(msg_str,"Ток: %3.2f",current_meansurment);
        LCD_message(msg_str);
        ADC12IFG = 0;
    }
}

void main(void) {
    WDTCTL = WDTPW + WDTHOLD;
    __enable_interrupt();
    Init_System_Clock();
    Init_System();
    LCD_init();
    LCD_message("Ожидание");
    LCD_set_pos(1,0);
    LCD_message("преоб...");
    init_uninterrupted_meansurement();
    while(1){

    };
};

```


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;
    BCCTL1 &= ~XT2OFF; // включение осциллятора XT2
    // 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?
    BCCTL2 |= 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++);
}
```