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)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

ЭВМ и ПУ

Лабораторная работа №1

**Изучение принципов программного управления внешними**

**устройствами на примере вывода информации на цифровой индикатор**

Выполнил:

ст. гр. ВТ-31

Клесов М.И.

Проверил:

Шамраев А.А.

Белгород, 2020

**Цель работы**: изучить принципы функционирования и возможности программного управления цифровым индикатором, разработать алгоритм и программу для вывода информации на цифровой индикатор.

Задание: разработать в среде программирования Code Composer программу для микроконтроллера MSP430, которая выполняет вывод информации (ФИО студента и группу) на экран ЖКИ.

Порядок выполнения работы:

* 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**



**Приложение**

**Файл “main.c”:**

#include <msp430.h>

#include "stdio.h"

#include "system\_define.h"

#include "system\_variable.h"

#include "function\_prototype.h"

#include "main.h"

void main(void) {

WDTCTL = WDTPW|WDTHOLD;

Init\_System\_Clock();

Init\_System();

LCD\_init();

LCD\_set\_cursor(0);

const char \*out\_name = "Клесов МИ";

const char \*out\_group = "ВТ-31";

LCD\_set\_pos(0,0);

LCD\_message(out\_name);

LCD\_set\_pos(1,0);

LCD\_message(out\_group);

byte sym\_table[8] = {

0b00000000,

0b00000000,

0b00001110,

0b00010101,

0b00011111,

0b00010101,

0b00001110,

0b00000000

};

char i;

for (i = 0; i < 8; i++){ // формируем сивол в CGRAM (память знакогенератора)

LCD\_WriteCommand(0x40 + i); // set CGRAM Address (0x40 + адресс)

LCD\_WriteData(sym\_table[i]);

}

LCD\_set\_pos(1,8);

LCD\_WriteData(0); // записываем в DDRAM (видеопамять) код запрограммированного символа

while(1);

}

**Файл “lcd.h”:**

#ifndef \_\_LCD\_H\_\_

#define \_\_LCD\_H\_\_

#define LCD\_MAXCOLS 16 //Максимальная длина строки LCD

#define LCD\_MAXROWS 2 //Количество строк LCD

#endif

**Файл “lcd.c”:**

// LCD-display functions

#include "function\_prototype.h"

#include "sysfunc.h"

#include "lcd.h"

//Таблица киррилицы

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...0xE4 <=> р с т у

0xE4,0xD5,0xE5,0xC0, //0xF5...0xE7 <=> 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 5х11 точек, F=1 => размер шрифта 5х8 точек

// Выбор режима передачи команд для LCD и вывод байта без ожидание сброса 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,B)

// 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;

}

// Устновка 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(); // Cигал WR/RST = 1 => сигнал R/W\_LCD = 1, т.е. в неактивном состоянии \_\_

Reset\_nSS(); // Сформировать сигал "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 сигал WR/RST = 0 => R/W\_LCD = 0

\_\_

Reset\_nSS(); // Сформировать сигал "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(); // Cигал 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];

}

**Файл “sysfunc.h”:**

#ifndef \_\_SYSFUNC\_H\_\_

#define \_\_SYSFUNC\_H\_\_

#include "system\_define.h"

extern word wait\_i, wait\_j;

#endif

**Файл “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

// 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++);

}