**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ **«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. В. Г. ШУХОВА»**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**ЛАБОРАТОРНАЯ РАБОТА №1**

**Дисциплина: ЭВМ и периферийные устройства**

**Тема: Изучение принципов программного управления внешними устройствами на примере вывода информации на цифровой индикатор**

Выполнил: ст. группы ВТ-31

Подкопаев Антон Валерьевич

Проверил: доцент кафедры ПО и ВТАС

Шамраев Анатолий Анатольевич

**Белгород 2020**

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

**Указания по организации самостоятельной работы**

Перед работой необходимо проработать теоретический материал по литературе и конспект лекций, ознакомиться с принципами функционирования и возможностями программного управления выводом символов на экран 32 позиционного цифрового ЖК-индикатора с использованием микроконтроллера MSP430F1611

**Порядок проведения работы и указания по ее выполнению**

Перед началом выполнения практической части лабораторной работы проводится экспресс-контроль знаний по принципам функционирования микроконтроллера MSP430, системе команд и возможностям организации программного управления выводом символов на экран цифрового индикатора. При подготовке к лабораторной работе необходимо составить предварительный вариант листинга программы, в соответствие с индивидуальным заданием.

**Задание.** Разработать в среде программирования IAR Embedded Workbench программу для микроконтроллера MSP430, которая выполняет вывод информации (фамилию, имя и отчество студента, и группу) на экран цифрового индикатора.

**Порядок выполнения задания:**

– запустить компилятор IAR Embedded Workbench.

– 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.

Теоретические сведения

**Принципы программного управления выводом символов на экран цифрового индикатора с помощью микроконтроллера MSP430F1611**

В лабораторном стенде используется индикатор WH1602A-NGG-CT на базе контроллера HD44780. Данный ЖКИ позволяет отображать 2 строки по 16 символов. Символы отображаются в матрице 5х8 (или 5х10) точек. Между символами имеются интервалы шириной в одну отображаемую точку. Каждому отображаемому на ЖКИ символу соответствует его код в ячейке ОЗУ модуля.

Модуль содержит два вида памяти – кодов отображаемых символов и пользовательского знакогенератора, а также логику для управления ЖКпанелью.

Модуль позволяет:

– работать как по 8-ми, так и по 4-х битной шине данных (задается при инициализации);

– принимать команды с шины данных;

– записывать данные в ОЗУ с шины данных; – читать данные из ОЗУ на шину данных;

– читать статус состояния на шину данных;

– запоминать до 8-ми изображений символов, задаваемых пользователем; – выводить мигающий (или не мигающий) курсор двух типов;

– управлять контрастностью и подсветкой;

**Программирование и управление.**

Перед началом рассмотрения принципов управления ЖКИ-модулем, обратимся к внутренней структуре контроллера HD44780. Можно сразу выделить основные элементы, с которыми приходится взаимодействовать при программном управлении: регистр данных (DR), регистр команд (IR), видеопамять (DDRAM), ОЗУ знакогенератора (CGRAM), счетчик адреса памяти (АС), флаг занятости контроллера (BF). Другие элементы не являются объектом прямого взаимодействия с управляющей программой – они участвуют в процессе регенерации изображения на ЖКИ: знакогенератор, формирователь курсора, сдвиговые регистры и драйверы.

**Схема подключения ЖКИ к микроконтроллеру MSP430F1611**

Для согласования электрических уровней в схеме применен преобразователь SN74LVC4245A (3,3V в 5V).

Изображение выглядит как снимок экрана

Автоматически созданное описание

Сигналы OE\_BF\_LCD и \_LCD WR/ поступают от дешифратора DD7, если на нем выбран модуль LCD (A0=1, A1=1).

Изображение выглядит как снимок экрана

Автоматически созданное описание

Изображение выглядит как снимок экрана

Автоматически созданное описание

*Приложение*

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 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;

}

// 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 киррилицу

char LCD\_recode(char b)

{

if (b<192) return b;

else return LCD\_table[b-192];

}

main.c

#include <msp430.h>

#include "stdio.h"

#include "system\_define.h"

#include "system\_variable.h"

#include "function\_prototype.h"

#include "main.h"

/\*

\* main.c

\*/

void main(void) {

Init\_System\_Clock();

Init\_System();

WDTCTL = WDTPW + WDTHOLD;

LCD\_init();

LCD\_message(" ВТ-31 Подкопаев Антон ");

LCD\_WriteCommand(0x40);

LCD\_WriteData(0b000011111);

LCD\_WriteCommand(0x41);

LCD\_WriteData(0b000010001);

LCD\_WriteCommand(0x42);

LCD\_WriteData(0b00010001);

LCD\_WriteCommand(0x43);

LCD\_WriteData(0b00000100);

LCD\_WriteCommand(0x44);

LCD\_WriteData(0b00001010);

LCD\_WriteCommand(0x45);

LCD\_WriteData(0b00010001);

LCD\_WriteCommand(0x46);

LCD\_WriteData(0b00001110);

LCD\_WriteCommand(0x47);

LCD\_WriteData(0b00010001);

LCD\_set\_pos(0,0);

LCD\_WriteByte(0,1);

while(1){

}

}

Результат работы программы

Изображение выглядит как стена, внутренний, сидит

Автоматически созданное описание