**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

Лабораторная работа №1

Дисциплина: ЭВМ и периферийные устройства

по теме Изучение принципов программного управления внешними устройствами на примере вывода информации на цифровой индикатор

Выполнил: ст. группы ВТ-31  
Новожен Н.В

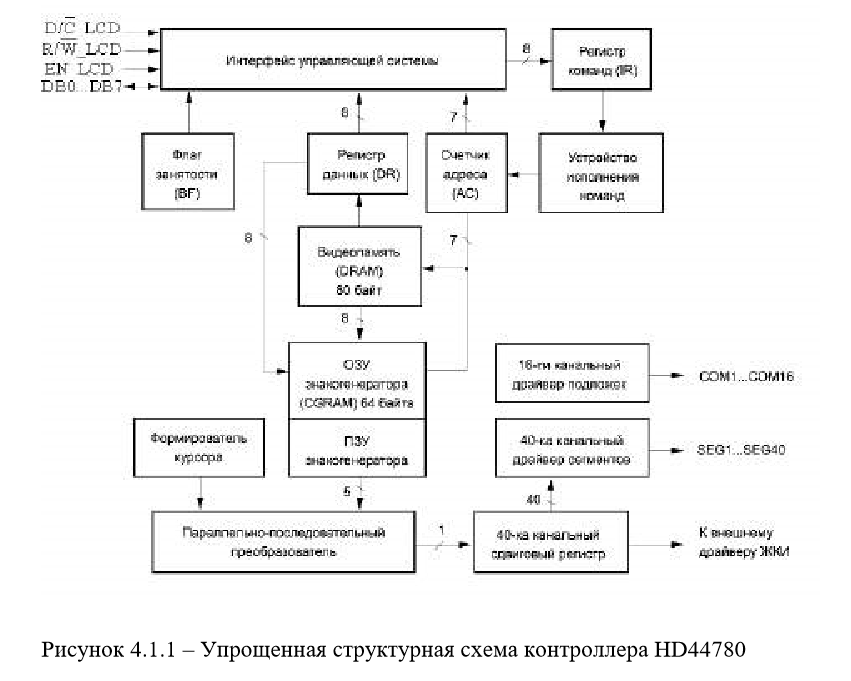
Проверил: Шамраев А,А

**Белгород 2020**

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

**Ход работы**

1. – запустить компилятор IAR Embedded Workbench.
2. – создать пустой проект.
3. – создать файл ресурса для кода программы и подключить его к проекту.
4. – выполнить компиляцию исходного модуля программы и устранить ошибки, полученные на данном этапе.
5. – проверить работоспособность программы и показать результаты работы преподавателю.



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.

*/\**

*\** ***main.c***

*\*/*

*void main(void) {*

*WDTCTL = WDTPW + WDTHOLD;*

*Init\_System();*

*Init\_System\_Clock();*

*LCD\_init();*

*LCD\_message("ВТ-31 Новожен");*

*LCD\_WriteCommand(64);*

*LCD\_WriteByte(0b00000000,1);*

*LCD\_WriteByte(0b00000000,1);*

*LCD\_WriteByte(0b00010010,1);*

*LCD\_WriteByte(0b00000000,1);*

*LCD\_WriteByte(0b00001100,1);*

*LCD\_WriteByte(0b00000000,1);*

*LCD\_WriteByte(0b00010010,1);*

*LCD\_WriteByte(0b00001100,1);*

*LCD\_set\_pos(0,0);*

*LCD\_WriteByte(0,1);*

*while(1){}*

*}*

*// 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*

*}*

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) );

}

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

// \_\_

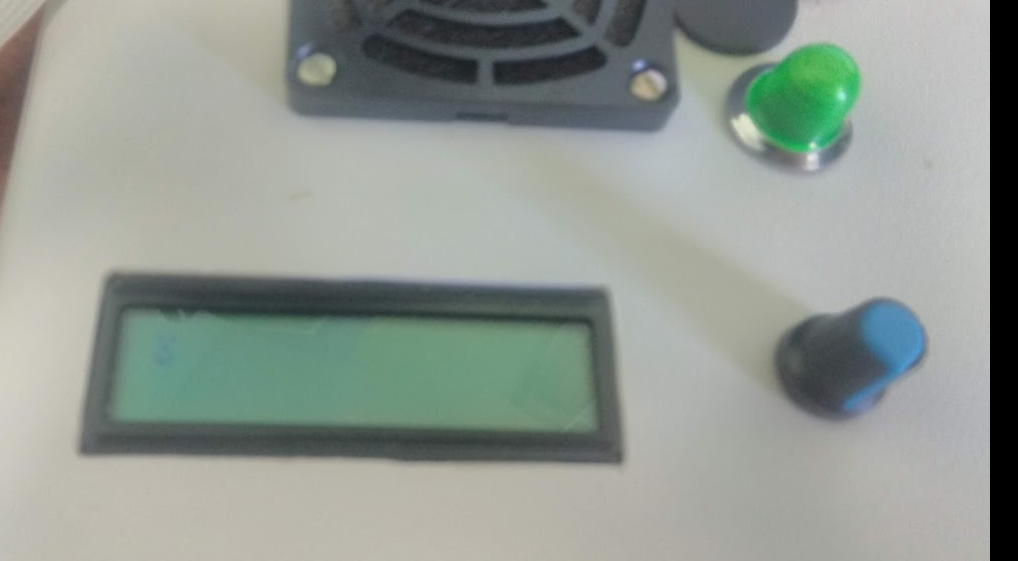
Set\_nSS(); // Перевести сигнал OE\_BF\_LCD =1 в неактивное состояние SS = 1

DB\_DIR = 0x00; // Шина данных на вход

// \_\_ \_\_\_ \_

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

}



Вывод: мы научились выводить на дисплей текст и создавать свои символы и познакомились с архитектурой MSP430;

**Контрольные вопросы и задания**

1. Поясните принцип функционирования цифрового индикатора, подключаемого к лабораторному макету.

2. Поясните алгоритм программного управления контроллером цифрового индикатора.

3. Каким образом можно осуществлять вывод информации на цифровой индикатор в фиксированные позиции?

4. Поясните принципы использования команд установки и сброса отдельных битов; приведите примеры.

5. Приведите алгоритм универсальной программы управления цифровым индикатором.