**2 ЛАБОРАТОРНАЯ РАБОТА №2**

**ЖИДКОКРИСТАЛЛИЧЕСКИЙ ИНДИКАТОР**

**Цель работы**: изучить внутреннюю структуру, особенности организации памяти модуля жидкокристаллического индикатора, организацию обмена данными с ним, а также научиться отображать символьные данные на жидкокристаллическом индикаторе.

## 2.1 Краткие теоретические сведения

Блок коммутации выводов позволяет производить индивидуальное конфи­гурирование выводов микроконтроллера для назначения им требуемых функ­ций. Регистры управления блока коммутации выводов управляют мультиплек­сорами, которые производят соединение выводов микроконтроллера со встро­енными периферийными устройствами. Периферийные устройства должны быть подключены к соответствующим выводам ещё до то­го, как они будут активизированы, а также до разрешения любых связанных с ними прерываний. Активизация любой периферийной функции, которая не подключена к связанному с ней выводу, приведет к неопределенному результату.

Выбор одной из функций для подключения к выводу порта полностью исклю­чает все другие функции, доступные этому выво­ду. Частичным исключением из этого правила являются вхо­ды АЦП. Независимо от функции, выбранной для вывода порта, к которому подключен вход АЦП, этот вход может читаться микроконтрол­лером в любое время и изменение входного напряжения на нем будет отражено в регистре результата аналого-цифрового преобразования.

Однако корректное чтение результата преобразования может быть произведено, если для данного вывода выбрана функция аналогового входа АЦП, поскольку толь­ко тогда будет активизирована соответствующая схема интерфейса между выводом и модулем АЦП. Во всех других случаях, подключение цифровых логичес­ких схем, необходимых для активизации этой функции, приведет к некоррект­ному проведению аналого-цифрового преобразования.

После выполнения сброса микроконтроллера, значения всех битов в регистрах PINSEL0 и PINSEL1 равно “0”. Значения битов регистра PINSEL2 в этом случае будет определяться тем, используются ли для данного микроконтроллера внутрисистемная отладка или трассировка, или нет.

Регистры PINSEL0 и PINSEL1 управляют функциями выводов микроконтроллера, согласно перечисленным в таблице 3.15 параметрам настройки. Направление передачи данных (ввод/вывод) для каждого вывода определяется состоянием служебного бита выбора направления в регистре IODIR0 лишь в том случае, если для вывода назначена функция линии ввода/вывода общего назначения (GPIO). Если вывод выбран для выполнения других функций, то направление передачи данных устанавливается автоматически соответствующим периферийным устройством.

## 2.2 Порядок выполнения работы

* + 1. Изучите до начала выполнения работы внутреннее устройство жидкокристаллического индикатора, основные принципы его работы, а также схему подключения ЖКИ-модуля к микроконтроллеру, реализованную в учебно-отладочном стенде LPC2148 Education Board.
    2. Изучите примеры программирования работы ЖКИ-модуля, приведенные в разделе 2 данных методических указаний.
    3. Разработайте схему алгоритма управления ЖКИ-модулем с учетом установленного варианта задания по таблице 5.1.

Таблица 2.1 – Варианты заданий

|  |  |
| --- | --- |
| №  варианта | Формулировка задания |
| 1 | Реализовать алгоритм вывода данных на ЖКИ. Данные в виде целых чисел от 0 до 9 циклически поступают и выводятся в произвольном месте индикатора при срабатывании прерывания от таймера TIMER0, частота работы которого – 2 Гц. |

* + 1. Запустите интегральную среду разработки Keil uVision.
    2. Создайте проект, в котором опишите разработанный алгоритм управления при помощи языка С.
    3. Откомпилируйте созданный проект. При наличии сообщений об ошибках или предупреждениях вернитесь к предыдущему пункту и внесите необходимые изменения.
    4. Включите учебно-отладочный стенд. При помощи утилиты Flash Magic загрузите файл с расширением \*.hex, находящийся в папке проекта, в стенд.
    5. Визуально оцените правильность функционирования разработанного алгоритма.

**2.3 Результаты выполнения работы**

Диаграмма без названия

Рисунок 2.2 – Схема алгоритма реализованной программы

#include <lpc21xx.h>

#define LCD\_DATA\_BUS 16

#define LCD\_RS 24

#define LCD\_E 25

#define LCD\_BLACK\_CTL 30

#define LCD\_RW 22

int delay(int \_slp) {

int i, j = 0;

for (i = 0; i < \_slp \* 1000; i++)

j = i + 1;

return j;

}

int flag = 0;

void write\_lcd\_command(unsigned char command) {

IODIR1 |= (0xFF << LCD\_DATA\_BUS); //Set data bus as outputs

IODIR1 |= (1 << LCD\_RS); //Set RS as output

IODIR1 |= (1 << LCD\_E); //Set E as output

IODIR0 |= (1 << LCD\_BLACK\_CTL); //Set Backlight control as output

IODIR0 |= (1 << LCD\_RW); //Set R/W as output

IOCLR1 = (1 << LCD\_E); //E down

IOCLR1 = (1 << LCD\_RS); //RS = 0

IOCLR0 = (1 << LCD\_RW); //RW = 0

IOCLR1 = (0xFF) << LCD\_DATA\_BUS; //Clear data bus

IOSET1 = (command) << LCD\_DATA\_BUS; //Write data to bus

IOSET1 = (1 << LCD\_E); //E up

delay(10); //Wait

IOCLR1 = (1 << LCD\_E); //E down

}

void write\_lcd\_data(unsigned char data) {

IODIR1 |= (0xFF << LCD\_DATA\_BUS); //Set data bus as outputs

IODIR1 |= (1 << LCD\_RS); //Set RS as output

IODIR1 |= (1 << LCD\_E); //Set E as output

IODIR0 |= (1 << LCD\_BLACK\_CTL); //Set Backlight control as output

IODIR0 |= (1 << LCD\_RW); //Set R/W as output

IOCLR1 = (1 << LCD\_E); //E down

IOSET1 = (1 << LCD\_RS); //RS = 1

IOCLR0 = (1 << LCD\_RW); //RW = 0

IOCLR1 = (0xFF) << LCD\_DATA\_BUS; //Clear data bus

IOSET1 = (data) << LCD\_DATA\_BUS; //Write data to bus

IOSET1 = (1 << LCD\_E); //E up

delay(10); //Wait

IOCLR1 = (1 << LCD\_E); //E down

}

\_\_irq void Timer0ISR(void) {

flag = 1;

T0IR = 0x01;

VICVectAddr = 0;

return;

}

void InitTimer0(void) {

VICDefVectAddr = (unsigned int) &Timer0ISR;

VICIntEnable = 0x10; //Channel#4 is the Timer0

VICIntSelect = 0x00; //all interrupts are IRQs

T0MR0 = 22500000; //Timer match (~ 0.4 second)

T0MCR = 0x03; //Interrupt on Match0, reset timer on match

T0PC = 0x01; // Prescaler to 2

T0TC = 0x00; // reset Timer counter

T0TCR = 0x01; // enable Timer

return;

}

int incr\_number(int nmb) {

if (nmb == 9){

return 0;

}

return ++nmb;

}

int main(){

int nmb = 0;

write\_lcd\_command(0x38);

write\_lcd\_command(0x0E);

write\_lcd\_command(0x01);

write\_lcd\_command(0x06);

write\_lcd\_command(0x81);

InitTimer0();

while(1) {

if (flag) {

nmb = incr\_number(nmb);

write\_lcd\_data(0x30 | nmb);

write\_lcd\_command(0x81);

flag = 0;

}

}

}

**2.4 Особенности IDE Keil uVision3 выявленные в ходе выполнения лабораторной работы**

В ходе лабораторной работы было выявлено, что компилятор в этой IDE позволяет определять переменные только в начале блоков выполнения перед кодом, что существенно затрудняет программирование.

**Выводы**

В данной лабораторной работе научились работать с жидкокристаллическим индикатором, а также изучили основные возможности контроллера прерываний (VIC). Ознакомились с возможностями и особенности функционирования среды разработки Keil uVision 3. Была разработана программа которая отображает цифры на жидкокристаллическом индикаторе.