Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Оренбургский государственный университет»

Кафедра вычислительной техники

Е.В. Бурькова Е.И. Ряполова

МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ НА СТЕНДЕ SDK-1.1

Рекомендовано к изданию Редакционно-издательским советом Федерального государственного бюджетного образовательного учреждения высшего профессионального образования «Оренбургский государственный университет» в качестве методических указаний для студентов, обучающихся по программам высшего профессионального образования по направлениям подготовки 230100.62 Информатика и вычислительная техника, 090900.62 Информационная безопасность

Оренбург
2012

УДК 004.3(076.5)
ББК 32.973.26-04 я 73
Б 91

Рецензент – кандидат технических наук, доцент А.В. Хлуденев

Бурькова Е.В.

Б 91 Микропроцессорные системы. Лабораторный практикум на стенде SDK-1.1: методические указания к лабораторным работам / Е.В. Бурькова, Е.И. Ряполова; – Оренбургский гос. ун-т. – Оренбург: ОГУ, 2012. – 74 с.

В методических указаниях рассмотрены архитектурные особенности учебного лабораторного микропроцессорного комплекса SDK-1.1, этапы программирования и возможности его применения для изучения курса «Микропроцессорные системы».

Методические указания предназначены для студентов направлений подготовки 230100.62 Информатика и вычислительная техника, 090900.62 Информационная безопасность.

УДК 004.3(076.5)
ББК 32.973.26-04 я 73

Содержание

Введение

Учебные микропроцессорные комплексы (стенды) на базе микроконтроллеров предназначены для изучения принципов организации и работы микропроцессорной элементной базы, вспомогательных элементов (память, контроллеры ввода-вывода и др.), получения навыков проектирования и программирования микропроцессорных систем различного назначения. Внимания заслуживает опыт ООО «ЛМТ» (Санкт-Петербург), которое разработало и последовательно развивает семейство микропроцессорных стендов инструментального и учебного назначения - SDK.

Основу лабораторного комплекса составляет контроллер-конструктор (микропроцессорный стенд) SDK-1.1 на базе ОКЭВМ фирмы Analog Devices ADuC812. Сам лабораторный комплекс представляет собой совокупность контроллера-конструктора, подключенного к персональному компьютеру, и программного обеспечения для ПК и SDK-1.1. Подключение осуществляется к COM-порту ПК через кабель RS232, комплекс инструментальных программ обеспечивает весь процесс программирования SDK-1.1: компиляцию, доставку и запуск программ в SDK-1.1. Контроллер-конструктор имеет в своем составе устройства для ввода и отображения информации, снабжен блоком питания и может работать автономно от ПК.

Основными областями использования комплекса являются:

- обучение основам вычислительной и микропроцессорной техники, систем управления;
- автоматизация простых технологических процессов и лабораторных исследований;
- макетирование микропроцессорных систем, отладка программного обеспечения для систем на базе широко распространенного ядра Intel MCS-51;
- радиолюбительство, управление бытовой техникой.

Контроллер SDK-1.1 оснащен устройствами для обработки и формирования аналоговых и дискретных сигналов, а также приспособлениями для замыкания выходных цепей на входные и симуляции внешних событий. Это дает возможность использовать SDK-1.1 для обучения основам цифровой обработки сигналов, в качестве контрольно-измерительной панели при проведении экспериментов, при настройке оборудования, а также для формирования сигналов с заданными параметрами в процессе управления различными объектами.

Главной областью применения микропроцессорного стенда SDK-1.1, безусловно, является обучение различным аспектам встраиваемой вычислительной техники. Студенты имеют возможность ознакомиться на практике с проектированием, программированием, отладкой и использованием создаваемой ими из "конструктора" SDK-1.1. Разнообразные устройства, входящие в состав стенда, позволяют изучить круг вопросов, связанных с организацией взаимодействия с ними через типичные интерфейсы, применяемые во встраиваемых вычислительных системах.

Методические указания предназначены для студентов направлений подготовки 230100.62 «Информатика и вычислительная техника», 090900.62 «Информационная безопасность» при изучении курса «Микропроцессорные системы».

В первом разделе учебного пособия рассматриваются особенности архитектуры учебного стенда, приводится полное описание функциональных блоков, входящих в его состав, основные режимы работы стенда, способы программирования стенда, инструментальные средства отладки и загрузки программ.

В разделах со второго по седьмой рассмотрены вопросы организации работы со светодиодами, матричной клавиатурой дисплеем и портами стенда, даны примеры программ, варианты заданий для выполнения лабораторных работ по курсу «Микропроцессорные системы». В восьмом разделе приведены контрольные вопросы для проверки полученных знаний и навыков работы со стендом SDK-1.1.

1 Архитектура лабораторного комплекса SDK-1.1

Учебный лабораторный комплекс SDK–1.1 предназначен для освоения студентами архитектуры и методов проектирования вычислительных систем на базе микроконтроллера ADuC812. Процессор ADuC812 является клоном Intel 8051 со встроенной периферией, имеет ядро MCS–51. Разнообразные устройства, входящие в состав стенда, позволяют изучить круг вопросов, связанных с организацией взаимодействия с ними через типичные интерфейсы, применяемые во встраиваемых вычислительных системах.

Стенд SDK–1.1 отличается следующими особенностями:

1) Вычислительное ядро:

а) центральный процессор имеет распространенную архитектуру MCS–51. Объем памяти контроллера позволяет реализовывать программные комплексы средней сложности;

б) развитая структура подсистемы временной синхронизации (встроенные часы реального времени, таймеры–счетчики) обеспечивает возможность глубокого исследования принципов и проблем организации систем реального времени, планирования, синхронизации процессов и т. п.;

в) центральный микроконтроллер ADuC812 позиционируется как микроконтроллер для построения простейших систем обработки сигналов. Он обладает широким набором каналов дискретного и аналогового ввода–вывода.

2) Система ввода–вывода:

а) дискретные входы и выходы позволяют подключать к SDK–1.1 внешние устройства для изучения их архитектуры и цифровых интерфейсов;

б) параллельная шина предназначена для подключения платы расширения SDX, увеличивающей количество портов дискретного и аналогового ввода–вывода комплекса SDK–1.1.

3) Разработка и отладка программного обеспечения:

а) SDK–1.1 поддерживает загрузку и запуск тестовых программ в ОЗУ контроллера без перепрограммирования энергонезависимой памяти программ;

б) обновление встроенного системного программного обеспечения в энергонезависимой памяти программ производится с персонального компьютера по стандартному последовательному каналу (RS–232C) без применения специальных программаторов.

1.1 Структура аппаратной части

Структура аппаратной части стенда SDK-1.1 представлена на рисунке 1.1.

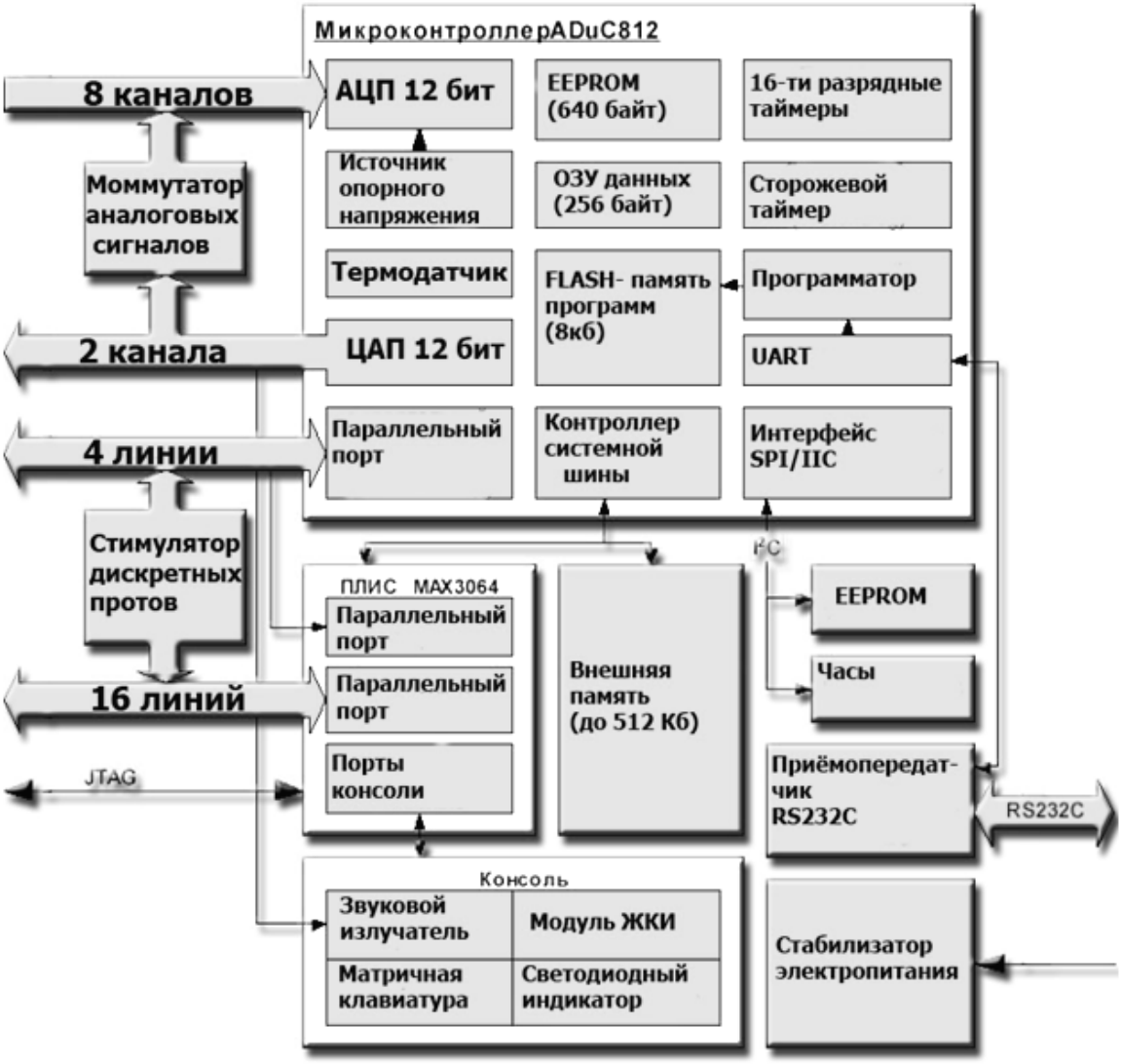


Рисунок 1.1 – Структура аппаратной части стенда SDK-1.1

В состав учебного стенда SDK-1.1 входят:

– вычислительное ядро на основе ОКЭВМ ADuC812;
– внешняя память E2PROM объемом 256 байт;

– 128К внешней памяти SRAM с возможностью расширения до 512К;

– FLASH-память (8 Кб);

– гальванически изолированный порт RS232C для связи с ПК;

– 8, 16, 20-ти разрядный порт дискретного ввода-вывода;

– аналоговый порт ввода на базе 8-канального 12-разрядного высокоскоростного АЦП со встроенным термодатчиком и возможностью работы в режиме прямого доступа к памяти (ПДП);

– аналоговый порт вывода на основе двух 12-рарядных ЦАП;

– второй блок EEPROM-памяти емкостью до 32 Кб, подключенный к вычислителю через интерфейс I2C;

– три 16-разрядных таймера-счетчика с внешними счетными входами (возможностью подачи сигналов через переключатели стенда) и блоком захвата/сравнения для измерения параметров и/или формирования дискретных сигналов;

– сторожевой таймер (Watchdog);

– жидкокристаллический индикатор для вывода текста;

– линейка из 8 сигнальных светодиодов;

– акустический пьезокерамический излучатель;

– матричная клавиатура на 16 клавиш;

– переключатели-стимуляторы 10 линий параллельного порта, сигналов от внешних источников прерываний, коммутаторы сигналов с выходов ЦАП на входы АЦП;

– часы/календарь с возможностью подключения внешней батареи питания.

Несмотря на большое количество устройств, входящих в стенд SDK-1.1, он имеет небольшие габариты: 13 x 12,5 x 2 (см), выполнен в прочном пластмассовом корпусе и снабжен защитой от возможных повреждений, связанных с постоянным использованием студентами. На рисунке 1.2 показан внешний вид стенда SDK-1.1.

Рисунок 1.2 – Внешний вид стенда SDK-1.1

На рисунке 1.3 дано схематическое изображение стенда SDK-1.1. Расшифровка обозначений на схеме дана в таблице 1.1.

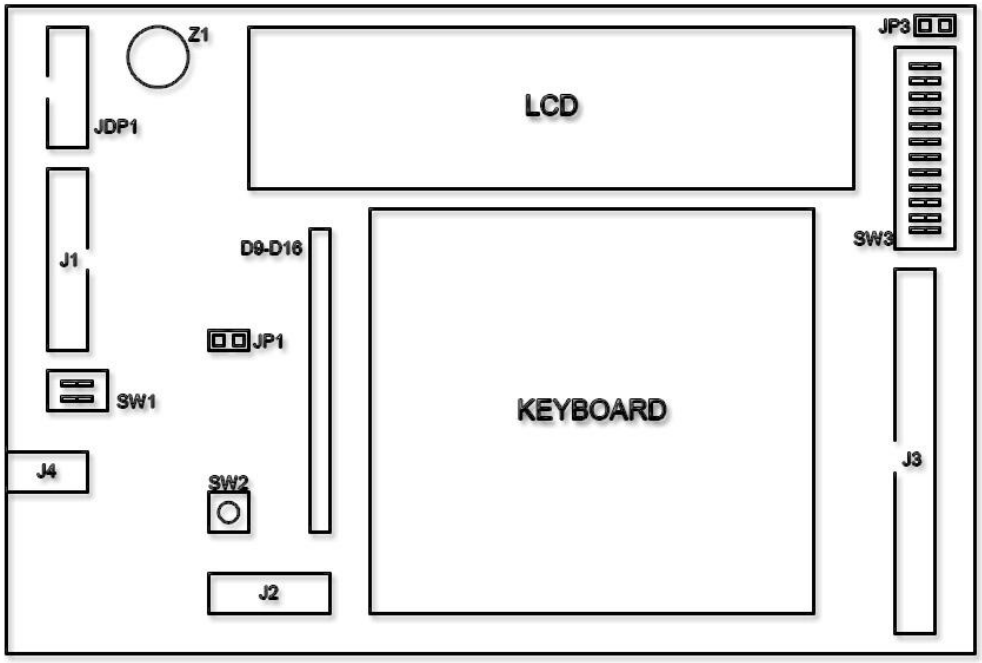


Рисунок 1.3 - Схематическое изображение стенда SDK-1.1.

Рассмотрим более подробно основные функциональные модули стенда.

Микроконтроллер ADuC812BS. Процессор ADuC812 является клоном Intel 8051 со встроенной периферией.

Таблица 1.1 – Расшифровка обозначений на схеме стенда SDK-1.1

Элемент	Описание
LCD	Жидкокристаллический индикатор WH1602B-YGK-CP.
KEYBOARD	Матричная клавиатура AK1604A-WWB.
Z1	Звуковой пьезокерамический излучатель.
SW2	Кнопка сброса RESET.
J4	Разъем питания стенда типа "JACK", полярность безразлична.
JDP1	Разъем последовательного порта стенда.
J1	Выводы каналов АЦП и ЦАП.
SW1	Переключатель, занимающий каналы 0 и/или 1 ЦАП на входы соответствующих (0, 1) каналов ЦАП.
J3	16 линий параллельного порта ПЛИС MAX и 4 линии параллельного порта P3 микроконтроллера ADuC812 (INT0/1,T0/1).
SW3	Набор переключателей, замыкающих соответствующие выводы J3 на корпус (переключение в лог. "0").
J2	Выводы JTAG - интерфейса ПЛИС MAX.
JP1	Перемычка, замыкающая вывод PSEN микроконтроллера ADuC812 на корпус.
Элемент	Описание.
JP3	Разъемы подключения внешней батареи питания часов реального времени PCF8583.
D9-D16	Набор сигнальных светодиодов.

Основные характеристики микропроцессора:

– рабочая частота 11,0592 МГц;

– 8-канальный 12-битный АЦП со скоростью выборок 200 К/с (в режиме ПДП);

– два 12-битных ЦАП (код-напряжение);

– внутренний температурный сенсор;

– 640 байт программируемого E2PROM со страничной организацией (256 страниц по 4 байта);

– 256 байт внутренней памяти данных (участок регистров общего назначения, битовый сегмент, свободный участок, участок регистров специального назначения);

– адресное пространство 16 Мб;

– режим управления питанием;

– асинхронный последовательный ввод-вывод;

– интерфейс I2C;

– три 16-битных таймера/счетчика и таймер WatchDog.

Внешняя E2PROM. E2PROM - перепрограммируемое электрически стираемое постоянное запоминающее устройство. Объем памяти E2PROM, установленной в стенде SDK-1.1, составляет 128 байт (возможна установка E2PROM большего объема, до 32 Кб). Микросхема E2PROM взаимодействует с процессором посредством интерфейса I2C. В таблице 1.2 дана адресация микросхем E2PROM.

Основные характеристики E2PROM: возможность перезаписи до 1 млн. раз; возможность побайтной и постраничной записи (в текущей конфигурации размер страницы составляет 8 байт).

Таблица 1.2 – Адресация микросхем E2PROM

Бит	Значение	Описание
0x7	RW	Переключатель.
0x6	A0	Адреса устройств (для расширения).
0x5	A1	
0x4	A2	
0x3	0	Обязательная по2следовательность для всех устройств E2PROM.
0x2	1	
0x1	0	
0x0	1	

Матричная клавиатура АК1604А-WWB. Клавиатура организована в виде матрицы 4x4. Доступ к колонкам и рядам организован как чтение/запись

определенного байта внешней памяти (4 бита соответствуют 4 колонкам, другие 4 бита - рядам).

Жидкокристаллический индикатор WH1602B-YGK-CP. ЖКИ работает в текстовом режиме (2 строки по 16 символов), имеет подсветку (цвет желто-зеленый).

Основные характеристики жидкокристаллического индикатора:

– габариты: 80x36x13.2 мм;

– активная область 56.21x11.5 мм;

– размеры точки 0.56x0.66 мм; размеры символа 2.96x5.56 мм;

– встроенный набор 256 символов (ASCII + кириллица);

– генератор символов с энергозависимой памятью на 8 пользовательских символов.

Часы реального времени PCF8583. PCF8583 - часы/календарь с памятью объемом 256 байт, работающие от кварцевого резонатора с частотой 32.768 кГц. Питание осуществляется ионистором (0.1 ф). Из 256 байт памяти собственно часами используются только первые 16 (8 постоянно обновляемых регистров-защелок на установку/чтение даты/времени и 8 на будильник), остальные 240 байт доступны для хранения данных пользователя. Точность измерения времени - до сотых долей секунды. Взаимодействие с процессором осуществляется через интерфейс I2C.

1.2 Распределение памяти в SDK-1.1

Адресное пространство процессора разделяется на два, не отображаемых друг на друга участка - внешнюю память и внутреннюю.

На рисунке 1.4 показана схема распределения памяти в SDK-1.1. На представленной схеме внутренняя память расположена в левой части, а внешняя – в правой части.

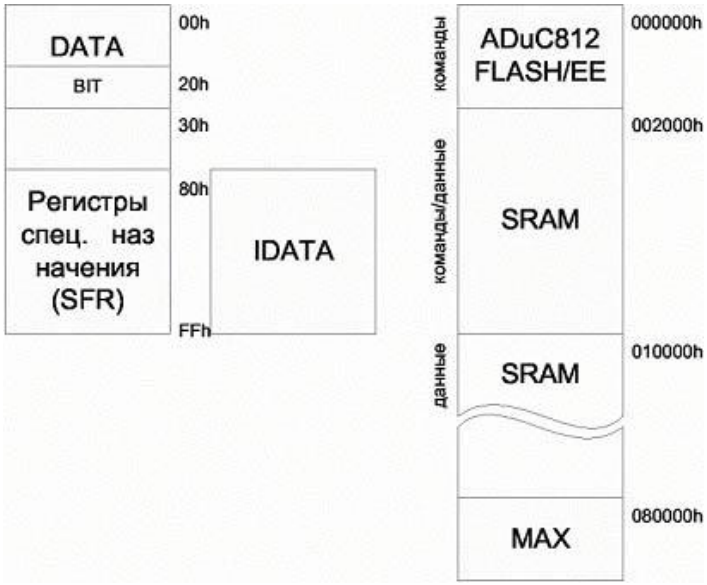


Рисунок 1.4 - Схема распределения памяти SDK-1.1

Внутренняя память. Микропроцессор ADuC812BS, являясь аналогом процессора Intel 8051, унаследовал типичную для процессоров этого семейства структуру организации внутренней памяти. Внутренняя память (256 байт) разделена на 4 участка:

- участок регистров общего назначения;
- битовый сегмент;
- свободный участок;
- участок регистров специального назначения.

В таблице 1.3 содержится информация о распределении внутренней памяти в SDK-1.1.

Стандартная для архитектуры 8051 структура внутренней памяти представлена четырьмя банками по восемь регистров общего назначения (диапазоны адресов 00h-07h, 08h-0Fh, 10h-17h, 18h-lFh), битовым сегментом (20h-2Fh), свободным участком 30h-7Fh, областью размещения SFR (регистров специального назначения) 80h-FFh, доступной при прямой адресации, и свободной областью 80h-FFh, доступной при косвенной адресации.

Таблица 1.3 – Распределение внутренней памяти

	Регистры общего назначения: 4 банка				Битовый сегмент	Свободный участок	Регистры спец. назначения
	1	2	3	4			
Адрес	00-07	08-0F	10-17	18-1h	20-2F	30-7F	80-FF

Внешняя память. Внешняя память SDK-1.1 разбита на следующие области: AduC812 Flash/EE, SRAM, MAX. Распределение адресов внешней памяти представлено в таблице 1.4.

Таблица 1.4 - Распределение адресов внешней памяти

Flash/EE	SRAM			
	Page 1	Pages 2..7	Page 8	
0x0000 0x2000	0x02000 0x0FFFF	0x10000 0x7FFFF	0x80000 0x8FFFF	Диапазон адресов

ADuC812 Flash/EE. Память Flash/EE представляет собой постоянную память, в которой хранится сервисная программа обслуживающая стенд, в ней находится набор тестов оборудования и драйвер RS232, позволяющий загружать пользовательский программы. При подаче питания или сбросе управление передаётся по нулевому адресу и происходит инициализация всех регистров. Если пользовательская программа обратится к адресу 0, то стенд пройдёт процедуру реинициализации, что равносильно нажатию кнопки «сброс». Запись в эту область памяти возможна только в режиме программирования Flash-памяти, в простом режиме доступ к ним закрыт. Это необходимо учесть при разработке программы - код должен располагаться по адресам не ниже 0x2000. Это область, в которой располагается таблица векторов прерываний и резидентный загрузчик файлов в формате HEX в память SRAM.

Память SRAM. Статическая память имеет страничную организацию и представляет собой восемь страниц размером 64Кб каждая. Условно всю память

SRAM разделяют на три участка: первая страница, страницы со второй по седьмую и восьмая страница.

Выборка кода может осуществляться только из первой страницы, и поэтому весь код должен располагаться именно здесь. Страницы со второй по седьмую могут быть использованы только для хранения данных. Восьмая страница особенна тем, что в первых 8 байтах расположены регистры MAX 8064.

В младших адресах восьмой страницы адресного пространства (080000h-080007h) располагается 8 ячеек-регистров ПЛИС MAX8064 (MAX8128). Эта область предназначена для взаимодействия с периферийными устройствами стенда.

1.3 Карта портов ввода-вывода

В стенде SDK-1.1 ввод-вывод данных осуществляется с помощью портов микроконтроллера и микросхемы ПЛИС, которая имеет восемь регистров, отображаемых на внешнее адресное пространство процессора. Информация о портах ввода-вывода микроконтроллера и их назначении в таблице 1.5.

Таблица 1.5 – Порты ввода-вывода микроконтроллера

Порт	Назначение
P0.7-P0.0	Шина адреса/данных AD(7-0) системного интерфейса.
P1.7-P1.0	Аналоговый вход, линии которого мультиплексируются с линиями 7-0 АЦП.
P2.7-P2.0	Адресная шина системного интерфейса A(15-8).
P3.0	RxD - входные данные приемопередатчика UART.
P3.1	TxD - выходные данные приемопередатчика UART.
P3.2	#INT0 - сигнал внешнего прерывания 0, активный уровень - лог. "0".
P3.3	#INT1 - сигнал внешнего прерывания 1, фиктивный уровень - лог. "0".
P3.4	Счетный вход таймера-счетчика T0, активный уровень - лог. "0".
P3.5	Счетный вход таймера-счетчика T1, активный уровень - лог. "0".
P3.6	#WR - сигнал записи во внешнюю память XRAM.
P3.7	#RD - сигнал чтения из внешней памяти XRAM.

2 Организация работы со светодиодами стенда SDK1.1

Ввод-вывод данных в стенде SDK1.1 может осуществляться как непосредственно через порты процессора так и через консоль опосредованно ПЛИС микросхемой MAX8064.

Программирование MAX8064.

Стенд SDK1.1 имеет четыре устройства ввода-вывода, объединённых в «консоль управления». В состав консоли входят:

– 8 светодиодов;

– матричная клавиатура;

– динамик;

– LCD дисплей (16x2 символов).

Связь между консолью и процессором опосредована микросхемой MAX8064. Микросхема MAX имеет восемь однобайтных регистров, с помощью которых происходит управление вводом-выводом. Данные регистры отображаются во внешнем адресном пространстве микроконтроллера как интервал адресов от 0x080000-0x80007, то есть находятся в начале восьмой страницы расширенной памяти. Ниже, в таблице 2.1, дана информации об адресах размещения регистров MAX8064.

Таблица 2.1 - Размещение регистров MAX8064

Адрес в странице	Регистр	Доступ	Описание
0x0	KB	Чтение	Регистр клавиатуры.
0x1	DATA_IND		Регистр шины данных LCD.
0x2	EXT_LO	Запись	Регистр параллельного порта (младший байт).
0x3	EXT_HI		Регистр параллельного порта (старший байт).
0x4	ENA		Регистр разрешений.
0x6	C_IND		Регистр управления LCD.
0x7	SV		Регистр состояния светодиодов.

Для записи в регистр необходимо создать указатель на байт во внешней памяти с помощью использования селектора xdata и указать в нём адрес регистра.

Пример:

```
unsigned char xdata ucxMaxReg = 0x7;
```

В данном примере создаётся указатель на регистр управления светодиодами. Перед записью по этому указателю, т.е. перед разыменованием необходимо переключить страницу внешней памяти на 8-ую, а после записи вернуться к исходной. Номер используемой страницы памяти находится в регистре DPP. Если модуль “ADuC812.H” не подключен, то переменную регистра можно определить вручную:

```
sfr   DPP = 0x84;
unsigned char oldPage = DPP;
DPP = 0x08;
*ucxMaxReg = 0xff;
DPP = oldDPP;
```

2.1 Лабораторная работа №1. Подключение светодиодов

Цель работы:

- изучить структуру аппаратной части стенда;
- изучить распределение памяти SDK 1.1;
- изучить ПЛИС регистры, их адреса, назначение битов и доступ к регистрам;
- научиться использовать команды чтения и записи регистров ПЛИС MAX8064;
- научиться использовать команды управления светодиодами и динамиком.

Пример. Зажечь светодиоды поочередно. В результате горит один светодиод, затем два, три и т.д. После зажигания всех светодиодов необходимо воспроизвести звук любой тональности. Интервал времени между зажиганиями следующих светодиодов должен быть подобран исходя из соображений возможности восприятия.

Листинг программы

```
// подключение необходимых модулей

#include <ADUC812.H>
#include "led.h"
#include "sound.h"

void pause(unsigned int interval); // объявление функции pause
int main(void)
{
char svet = 1;  // объявление и инициализация переменной svet
        int i;
        for (i=0; i<8; i++) // цикл поочередно зажигающий светодиоды
        {
        SetLed(svet); // зажигание светодиодов
                svet <<= 1; // сдвиг влево
                svet |= 1; // установка последнего бита в единицу
                pause(60000); // пауза между зажиганием в 60000 тактов          }

        while (1) // запуск бесконечного цикла {
        Snd_Enable(1); // включение динамика
                pause(1000); // задержка
                Snd_Enable(0); // выключение динамика
                pause(1000); // задержка
        } }
void pause(unsigned int interval) // процедура задержки
{
        unsigned int i;
        for (i = 0; i < interval; i++); }
```

Варианты заданий:

Вариант 1

Осуществить наглядный счёт (на светодиодах с задержкой) в двоичном коде от 0 до 64. По окончании счёта необходимо «зажечь» все светодиоды и воспроизвести звук любой тональности.

Вариант 2

Воспроизвести звуковой сигнал 8 раз. Каждый следующий звуковой сигнал должен иметь большую длительность, чем предыдущий. После воспроизведения звукового сигнала зажигать светодиод соответствующий его порядковому номеру. После зажигания светодиода организовать задержку примерно равную 0,5 секунды.

Вариант 3

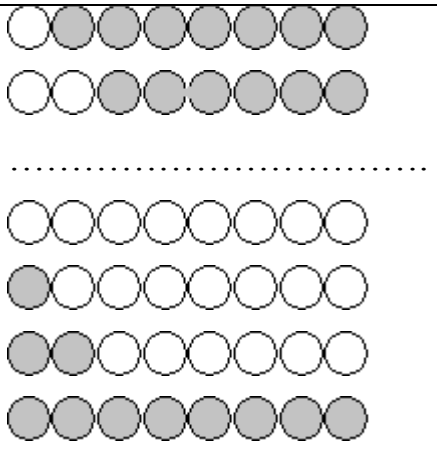
Зажечь одновременно все светодиоды стенда, затем погасить их и воспроизвести звуковой сигнал любой тональности. Повторить эти действия 5 раз. Между повторениями организовать задержку примерно равную 0,5 секунды.

Вариант 4

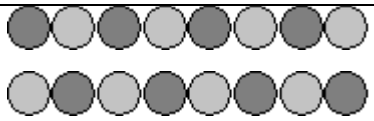
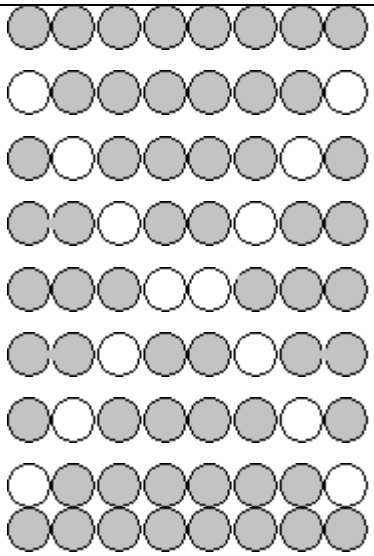
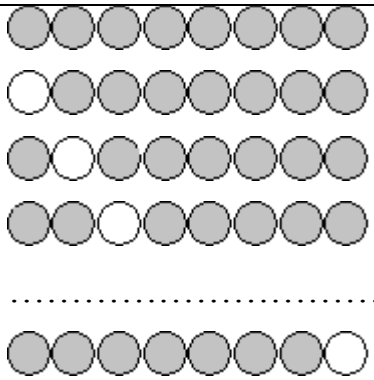
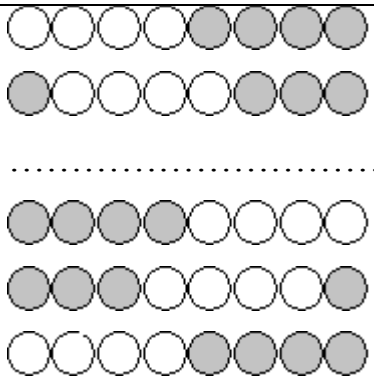
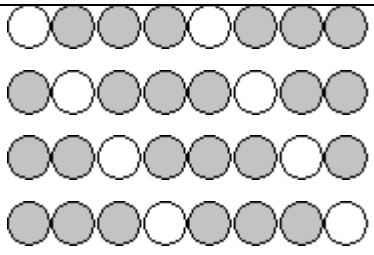
Зажигать светодиоды, начиная с крайних и постепенно дойти до центральных светодиодов. После зажигания всех светодиодов воспроизвести звуковой сигнал любой тональности и погасить светодиоды. Повторить эти действия 5 раз. Между повторениями организовать задержку примерно равную 0,5 секунды.

Задание для вариантов 5-14: написать программу, позволяющую зажигать светодиоды в указанных последовательностях. В таблице 2.2 приведены варианты заданий.

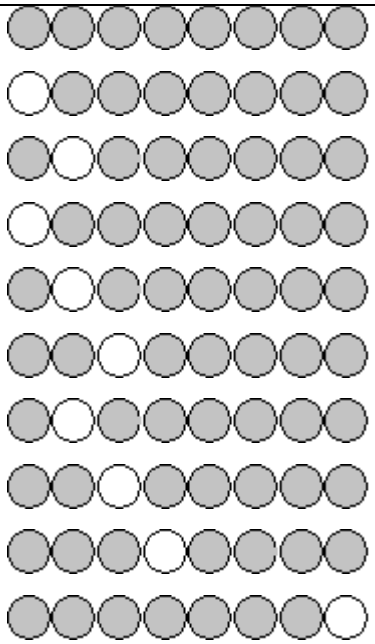
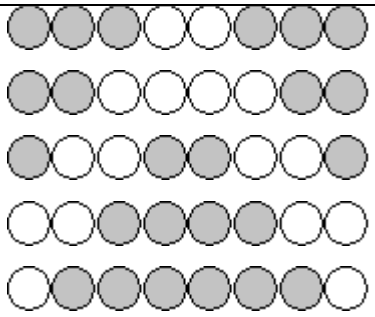
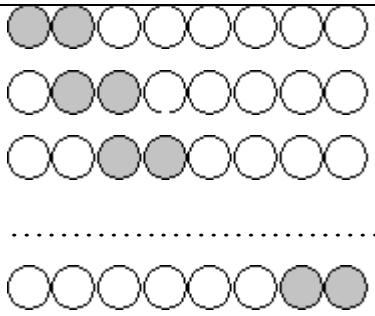
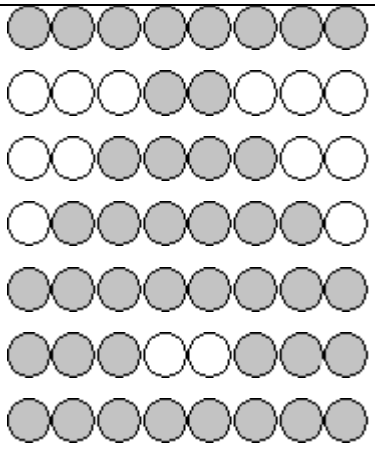
Таблица 2.2 – Варианты заданий 5-14

Номер варианта	Задание
5	

Продолжение таблицы 2.2

Номер варианта	Задание
6	
7	
8	
9	
10	

Продолжение таблицы 2.2

Номер варианта	Задание
11	
12	
13	
14	

Пример. Написать программу, позволяющую зажигать светодиоды в указанных последовательностях (рисунок 2.1).

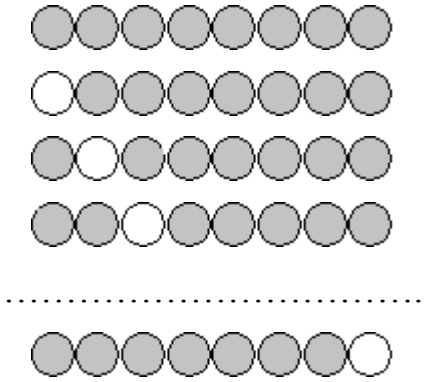


Рисунок 2.1- Последовательность светодиодов

Листинг программы

Файл main.c

```
#include "ADuC812.h" // подключение модуля внутренней памяти
#include "LED.c" // подключение модуля светодиодов

void main(void)  // заголовок главной функции
{
// определение объектов
        unsigned int i;
        unsigned char Leds = 0;
        while(1)        //запуск бесконечного цикла
        {
                SetLed(Leds); // вызов функции зажигания светодиодов
                Leds <<=1; // сдвиг влево
                if(Leds & 0x80) // если зажжен последний светодиод
                {
                        for(i = 0; i < 6000; i++); // пустой цикл для задержки
                }
                else
                {
                        Leds |=1;       //  or 1
                        for(i = 0; i < 6000; i++); // пустой цикл для задержки
                }
        }
}
```

Файл LED.c

```
extern void SetLed(unsigned char val); // функция зажигания светодиодов
void SetLed(unsigned char LedStat)
{
  unsigned char oldDPP=DPP; // определение объектов
  unsigned char xdata *reg = 0x7;
  DPP = 0x8;
    *reg = LedStat;
  DPP=oldDPP;
}
```

3 Работа с матричной клавиатурой стенда SDK-1.1

3.1 Организация матричной клавиатуры

Ввод данных непосредственно от пользователя осуществляется с помощью матричной клавиатуры. Клавиатура представлена в виде матрицы 4x4, а доступ к колонкам и рядам клавиатуры осуществляется путём чтения и записи управляющего байта, располагающегося в регистре KB (0x0 offset) микросхемы MAX. Информация о регистре клавиатуры содержится в таблице 3.1.

Таблица 3.1 – Регистр клавиатуры

Бит	Поле	Доступ	Описание
0..3	Сканируемая колонка	Чтение/Запись	Указывает клавиатуре результат сканирования какой колонки необходимо поместить в старшие биты.
4..7	Состояние колонки	Запись	Возвращает состояние указанной колонки: клавиша нажата клавиша не нажата

При указании более одного нуля приоритет имеет тот, который находится в младшем разряде. После указания номера колонки в старшие биты подается состояние ряда в данной колонке. При нажатии какой либо клавиши бит состояния устанавливается в «0». Ниже, в таблице 3.2, показана архитектура клавиатуры.

Сканирование происходит путем записи в младшие 4 бита номера сканируемой колонки, номер определяется логическим «0» установленным в соответствующем разряде: первая колонка - нулевой бит.

Во избежание ложного срабатывания при обнаружении нажатой клавиши рекомендуемся сделать задержку порядка 50 мс и повторить сканирование. Это поможет избежать ложных срабатываний из-за «дребезга» контактов и случайных «шумовых» помех.

Таблица 3.2 - Архитектура клавиатуры

1	2	3	A	Ряд 1
4	5	6	B	Ряд 2
7	8	9	C	Ряд 3
*	0	#	D	Ряд 4
Колонка 1	Колонка 2	Колонка 3	Колонка 4	

3.2 Лабораторная работа № 2. Приобретение навыков работы с матричной клавиатурой стенда SDK-1.1

Цель работы:

- изучить организацию матричной клавиатуры AK1604-WWB;

- научиться использовать команды считывания данных с клавиатуры и использовать полученные результаты для решения поставленной задачи.

Пример. Ввести с клавиатуры последовательно два числа в диапазоне от 0 до 9. Осуществить перемножение этих чисел и по нажатию нецифровой клавиши стенда (одной по выбору) отобразить данное число в двоичном коде на светодиодах.

Листинг программы

```
// подключение необходимых модулей
#include <ADUC812.H>
#include "led.h"
#include "kbd.h"
#include "sound.h"
unsigned char KeyCode(unsigned char ikey);
int main(void)  {
        unsigned char key1,key2,key3; // определение типов данных

        while (1) // запуск бесконечного цикла      {
                key1=key2=key3=0;  // обнуление переменных
                while (!KB_Hit(&key1)); // сканирование клавиатуры до тех пор пока не будет нажата клавиша
                SetLed(KeyCode(~key1)); // вывести значение клавиши на светодиоды
```

```
        pause(600*10); // задержка
        while (!KB_Hit(&key2));
        SetLed(KeyCode(~key2));
        pause(600*10000);
        key1 = KeyCode(~key1); // получение числового значение первой клавиши
        key2 = KeyCode(~key2); // получение числового значения второй клавиши

        while (1) // бесконечный цикл                {
            if (!KB_Hit(&key3))  // ожидание нажатия клавиши «A»
                if ((~key3)==0x18)                    {
            SetLed(~(key1*key2)); // вывод значение на светодиоды
            break;                }                }
        pause(600*10000);
        SetLed(0);      // отключение светодиодов }
    while(1); }
unsigned char KeyCode(unsigned char ikey) // функция перевода кода нажатой клавиши в цифровое значение {
    switch(ikey)    {
        case 0x82:      return 0;
        case 0x44:      return 9;
        case 0x42:      return 8;
        case 0x41:      return 7;
        case 0x24:      return 6;
        case 0x22:      return 5;
        case 0x21:      return 4;
        case 0x14:      return 3;
        case 0x12:      return 2;
        case 0x11:      return 1;       }       return 0; }
```

Варианты заданий представлены в таблице 3.3.

Таблица 3.3 – Варианты заданий

Номер варианта	Задание
1	Зажечь светодиоды при вводе с клавиатуры указанной последовательности из 3 нажатий и погасить после повторного ввода последовательности.
2	Мигать светодиодами, пока полностью зажат одна строка и не нажато ни одной другой клавиши.
3	Мигать светодиодами, пока полностью зажат один столбец и не нажато ни одной другой клавиши.
4	Зажечь один светодиод и управлять его перемещением с помощью клавиш 4-6.

Продолжение таблицы 3.3

Номер варианта	Задание
5	Реализовать функции сдвига влево, вправо светодиодного ряда и установки-сброса состояния нулевого светодиода.
6	Зажигать линию светодиодов, начинающуюся с младшего разряда и имеющую длину, равную сумме столбца сканирования и строки нажатой клавиши.
7	На светодиодном индикаторе отображать сумму любых трех нажатых цифровых клавиш.
8	Зажечь светодиоды так, чтобы первые четыре соответствовали номеру сканируемого столбца, а последние - его состоянию.
9	С помощью светодиодов вывести ASCII код нажатой клавиши.
10	Зажигать первый и последний светодиод клавишами расположенными в шахматном порядке. (Клавиши на месте белых клеток зажигают один диод, клавиши на месте чёрных - другой).
11	Написать программу для стенда SDK1.1 осуществляющую инверсию соответствующего разряда байта при нажатии клавиши микроконтроллера от 1 до 8. Значение инвертируемого байта записано в программе и отображается на светодиодах.
12	Написать программу для стенда SDK1.1, которая отображает на светодиодах двоичный код цифры, введённой с клавиатуры микроконтроллера. Отображать код при каждом нажатии клавиши. При нажатии клавиши “D” инвертировать значение регистра светодиодов.
13	Прибавлять к переменной, отражающей сумму, число соответствующее номеру нажатой клавиши микроконтроллера и отображать сумму на светодиодах. Начальное значение суммы равно 0. При значении суммы превышающее 255 погасить все светодиоды.
14	Задать значение некой переменной, отражающей сумму, в диапазоне от 0 до 255. При нажатии клавиши “1” вычитать из суммы единицу, при нажатии “2” осуществить прибавление единицы. При выходе за диапазон 0-255 справа продолжать счёт с 0, при выходе за границу диапазона слева продолжать счёт с 255. Отображать двоичный код значения переменной суммы при каждом нажатии клавиши. При нажатии клавиши “D” инвертировать значение переменной суммы.

4 Работа с динамиком стенда SDK-1.1

4.1 Принцип программного управления динамиком стенда

Вывод звука производится путем подачи на динамик периодических сигналов. Сигнал формируется на основе содержимого регистра ENA. Ниже в таблице 4.1 содержится информация о регистре разрешений динамика.

Для генерации звука используются биты SND2..0, отвечающие за уровень напряжения на обмотке динамика, задавая различные уровни напряжения можно генерировать звуки разной громкости. Однако, некоторые стенды не поддерживают изменение уровня напряжения. На таких стендах сигнал формируется на основе бита SND2.

Алгоритм формирования звуковой волны:

– подача сигнала на динамик;
– пауза равная полупериоду волны;
– отключение сигнала (изменения напряжения);
– пауза равная полупериоду волны.

Таблица 4.1 – Регистр разрешений динамика

Номер бита	Название	Доступ	Описание
0x7	-	Запись	
0x6	-		
0x5	INT0		Бит прерывания.
0x4	SND2		
0x3	SND1		Уровень напряжения на динамике.
0x2	SND0		
0x1	EN_HI		Чтение/Запись из параллельного порта.
0x0	EN_LO		

4.2 Лабораторная работа № 3. Получение навыков работы с динамиком и портами ввода вывода

Цель: программирование регистров ПЛИС MAX3064 с целью получения звукового сигнала. Написание библиотеки с функциями управления динамиком для последующего использования.

Задание: написать программу, позволяющую выполнить действия, соответствующие номеру варианта.

В таблице 4.2 приведены варианты заданий.

Таблица 4.2 – Варианты заданий

Номер варианта	Задание
1	Имитировать сигнал «внимание всем», частота которого медленно меняется по синусоиде с задержками в максимуме и минимуме на какое-то время.
2	Имитировать стук колес вагона с определенной частотой.
3	Воспроизвести сигнал если нажаты все кнопки одной строки.
4	Воспроизвести звуковую гамму, состоящую из звуков разной тональности, но одинаковой длительности.
5	Воспроизводить звук, частота которого контролируется с помощью клавиатуры: повышение частоты нажатием одной клавиши, понижение – другой.
6	Воспроизводить звук, частота которого зависит от нажатой клавиши.
7	Имитировать сигнал о переполнении буфера клавиатуры. Выдавать короткий звуковой сигнал при нажатии более двух клавиш.
8	Воспроизвести сигнал, если нажаты все кнопки одного столбца.
9	Имитация импульсного набора номера.
10	При правильном наборе пароля (последовательности из 3 клавиш) имитировать звук установки машины на сигнализацию.

Пример. Воспроизвести звуковую гамму, состоящую из звуков разной тональности, но одинаковой длительности.

Листинг программы

```
#include <ADUC812.H> // подключение модуля внутренней памяти
#include "sound.c" // подключение модуля звука
#include "LED.c" // подключение модуля светодиодов
void main(void) // заголовок главной функции{
// определение объектов
        unsigned int i,j;
        unsigned char Leds = 0, Val= 0;       while(1)
        // Создание задержек для мигания светодиодов и формирования звука
{               Leds = 0; // обнуление переменной leds
                for(Val = 0; Val<8; Val++) // запуск цикла          {
                        Leds <<= 0x1; // сдвиг влево
                        Leds  |= 0x1; // or 1
                        SetLed(Leds); // зажигание светодиодов
                        for(i = 0; i < 60 + (Val > 4? -20: 0); i++) // запуск цикла
                        {       Snd_Enable(1); // включение динамика
                                for(j = 0; j < 300; j++ ); // задержка (время звучания)
                                Snd_Enable(0); // отключение динамика
                                for(j = 0; j <  300 - Val *40; j++); // задержка }}} }
```

4.3 Лабораторная работа № 4. Получение навыков работы с портами ввода вывода стенда SDK-1.1

Цель работы:

- изучить регистры управления портами ввода-вывода EXT_HI, EXT_LO и регистр управления звуком ENA;

- научиться использовать команды управления звуком: изменение тональности и длительности звукового сигнала.

Описание работы

Требуется написать драйвер звукового излучателя, позволяющий задавать частоту (в герцах) и длительность звука (в миллисекундах). Для управления звуковым излучателем используется регистр ПЛИС ENA (адрес 080004h). 2-4 биты регистра ENA управляют величиной напряжения на динамике, т.е. позволяют задавать громкость звука: чем больше “единиц” выставлено в этих битах, тем громче звук. Частота звука задается частотой смены нулей и единиц в

управляющих битах регистра ENA. Ниже, в таблице 4.3, приведены частоты нот первой октавы.

Таблица 4.3 - Частоты музыкальных нот

Нота	Частота, Гц	Нота	Частота, Гц
До	261,63	Фа	349,22
Ре	293,67	Соль	391,99
Ми	329,63	Ля	440,00
Фа	349,22	Си	493,88

Варианты заданий:

Во всех вариантах задания единичный разряд в шестнадцатеричном коде – это соответствующий DIP-переключатель в положении «ON», нулевой разряд – это соответствующий DIP-переключатель в положении «OFF».

Вариант 1

В случае установки на DIP-переключателях кода 0x11 (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться анимация как на рисунке 4.1. Во всех остальных случаях светодиодные индикаторы отражают значение, выставленное на DIP- переключателях.

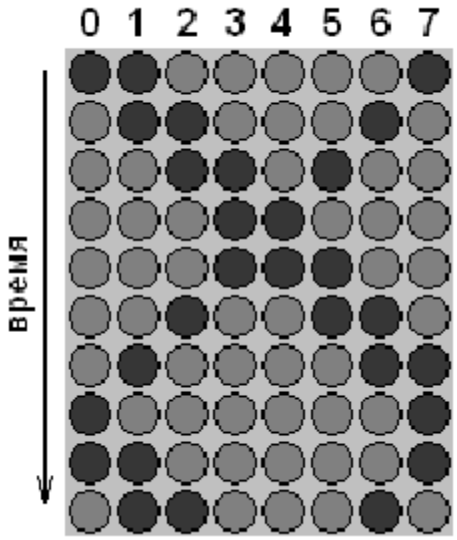


Рисунок 4.1- Анимация для варианта 1

Вариант 2

В случае установки на DIP-переключателях кода 0x22 (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться первая анимация, в случае установки кода 0xDD (шестнадцатеричное значение) – вторая анимация представленная на рисунке 4.2. Во всех остальных случаях светодиодные индикаторы отражают инвертированное значение, выставленное на DIP-переключателях.

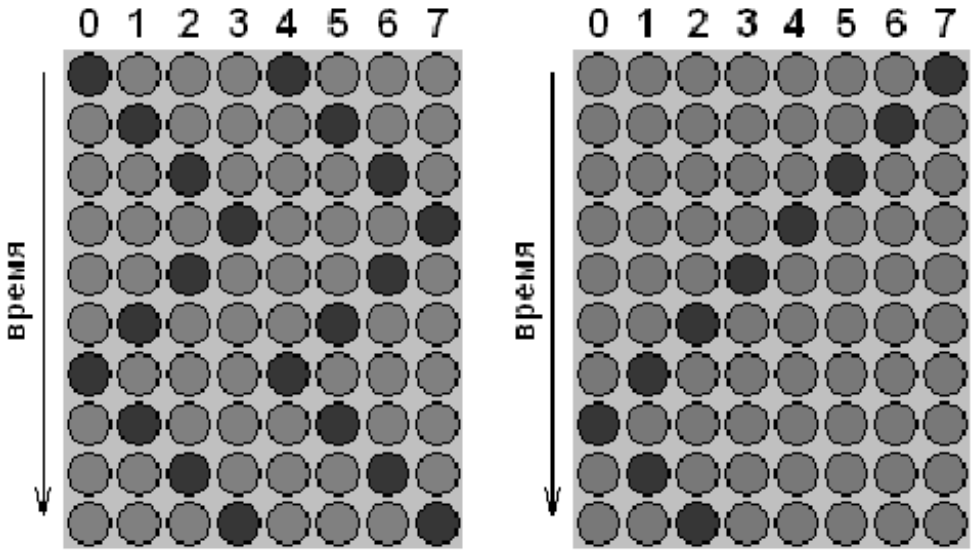


Рисунок 4.2- Анимация для варианта 2

Вариант 3

В случае установки на DIP-переключателях кода 0x33 (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться первая анимация, в случае установки кода 0xCC (шестнадцатеричное значение) – вторая анимация представленная на рисунке 4.3. Во всех остальных случаях светодиодные индикаторы отражают значение, выставленное на DIP-переключателях.

Вариант 4

В случае установки на DIP-переключателях кода 0x44 (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться анимация как на

рисунке 4.4. Во всех остальных случаях светодиодные индикаторы отражают инвертированное значение, выставленное на DIP-переключателях.

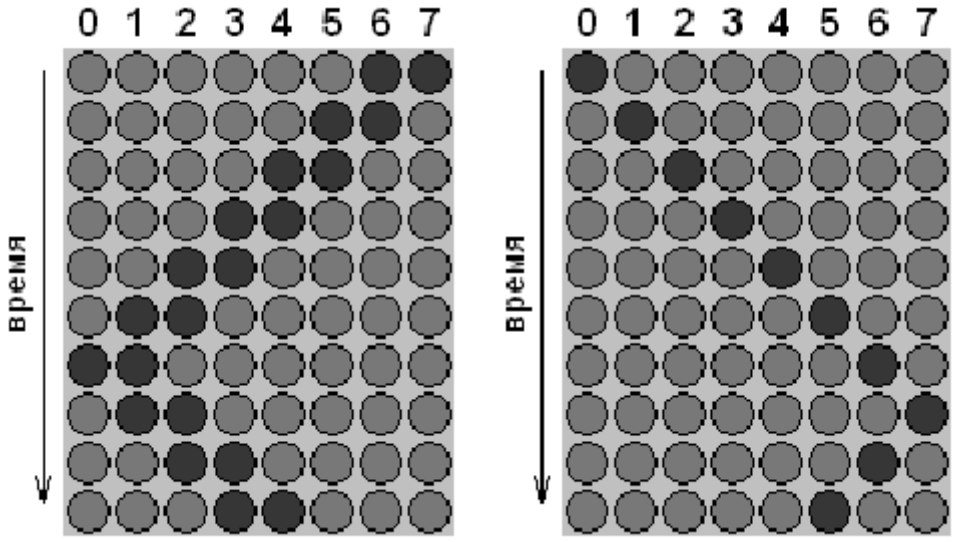


Рисунок 4.3- Анимация для варианта 3

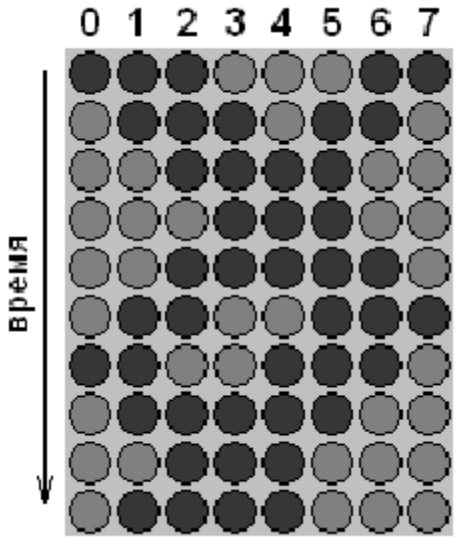


Рисунок 4.4- Анимация для варианта 4

Вариант 5

В случае установки на DIP-переключателях кода 0x55 (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться анимация как на рисунке 4.5. Во всех остальных случаях светодиодные индикаторы отражают значение, выставленное на DIP- переключателях.

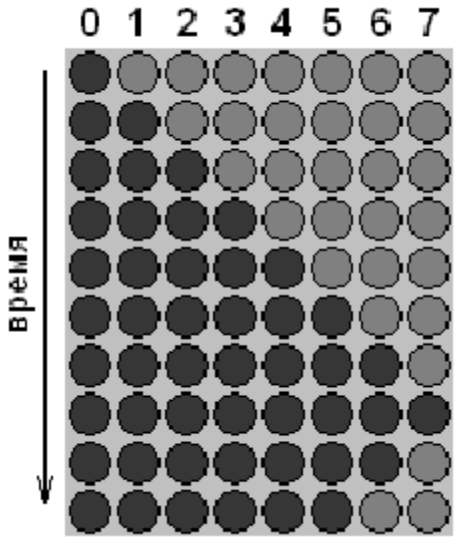


Рисунок 4.5- Анимация для варианта 5

Вариант 6

В случае установки на DIP-переключателях кода 0x66 (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться анимация, показанная на рисунке 4.6. Во всех остальных случаях светодиодные индикаторы отражают инвертированное значение, выставленное на DIP-переключателях.

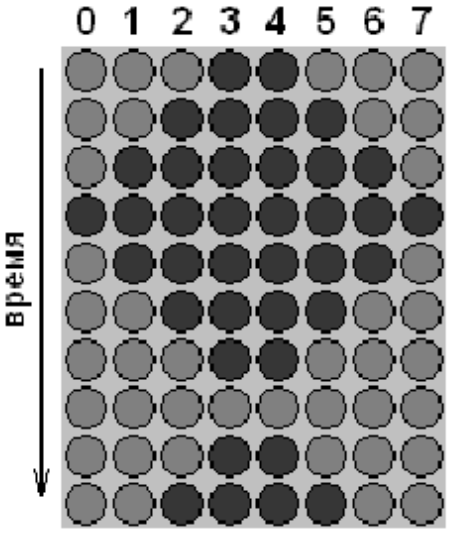


Рисунок 4.6- Анимация для варианта 6

Вариант 7

В случае установки на DIP-переключателях кода 0x77 (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться анимация, показанная

на рисунке 4.7. Во всех остальных случаях светодиодные индикаторы отражают значение, выставленное на DIP- переключателях.

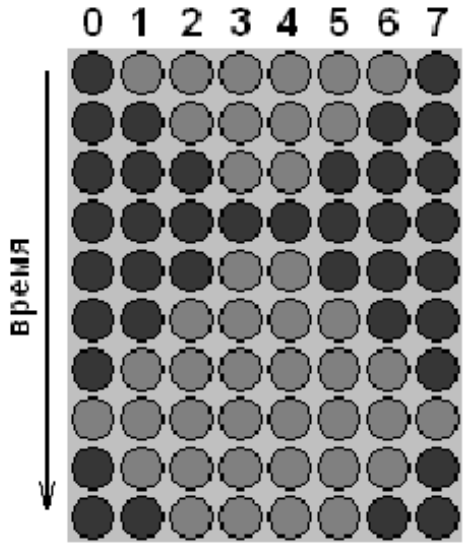


Рисунок 4.7- Анимация для варианта 7

Вариант 8

В случае установки на DIP-переключателях кода 0xAA (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться анимация, показанная на рисунке 4.8. Во всех остальных случаях светодиодные индикаторы отражают инвертированное значение, выставленное на DIP-переключателях.

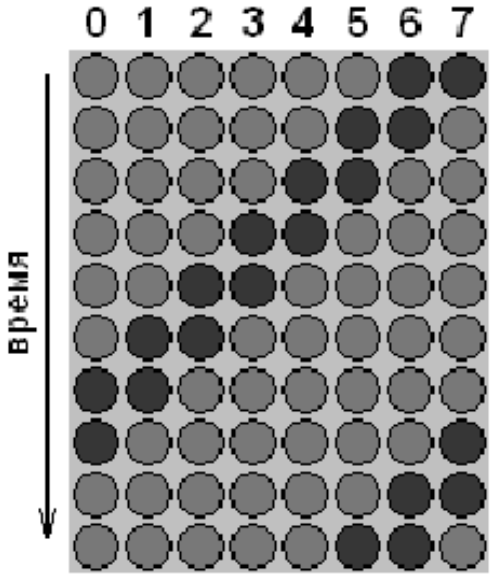


Рисунок 4.8- Анимация для варианта 8

Вариант 9

В случае установки на DIP-переключателях кода 0xBB (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться первая анимация, в случае установки кода 0x44 (шестнадцатеричное значение) – вторая анимация как показано на рисунке 4.9. Во всех остальных случаях светодиодные индикаторы отражают значение, выставленное на DIP-переключателях.

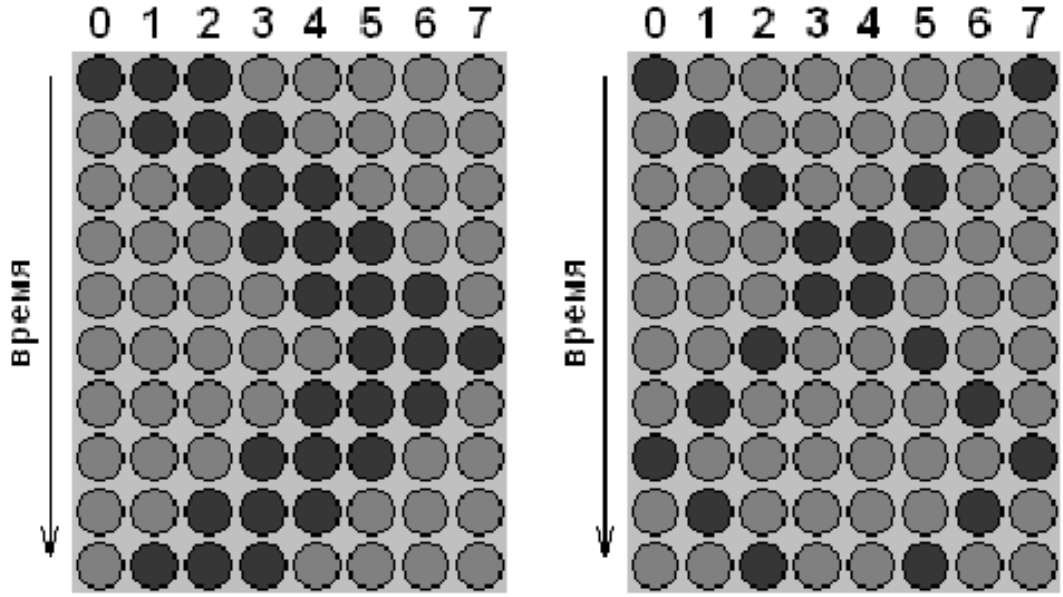


Рисунок 4.9- Анимация для варианта 9

Вариант 10

В случае установки на DIP-переключателях кода 0xCC (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться анимация как на рисунке 4.10. Во всех остальных случаях светодиодные индикаторы отражают инвертированное значение, выставленное на DIP-переключателях.

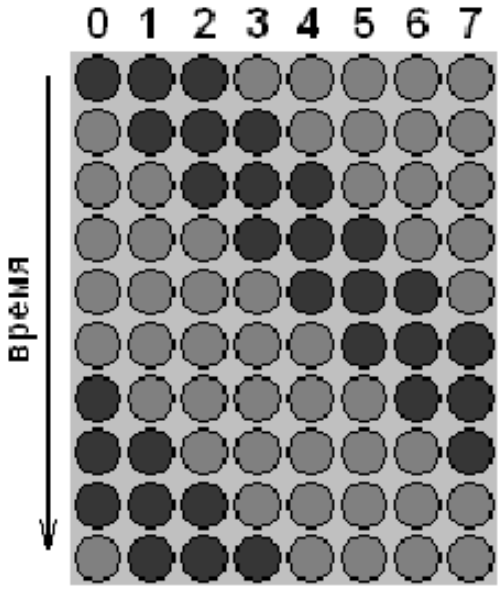


Рисунок 4.10- Анимация для варианта 10

5 Работа с LCD-дисплеем стенда SDK-1.1

5.1 Принцип программного управления LCD-дисплем стенда

Для вывода текстовой информации в стенде SDK1.1 используется жидкокристаллический дисплей WH1602B-YGK-CP вместимостью 32 символа (2 строки по 16 символов). Модуль LCD (далее БИС) подключен к контроллеру, который имеет три регистра: регистр команд (IR), регистр данных (DR) и счетчик адреса (AC). Регистр IR хранит коды операций, и доступен только для записи, при чтении возвращается только один бит BF – флаг занятости, регистр DR предназначен для записи и чтения из DDRAM (память отображаемых символов) и CGRAM (память генератора символов). Запись и чтение из памяти происходит по адресу указанному в AC. В таблицах 5.1 и 5.2 дано описание регистра DATA_IND и C_IND.

Запись в регистры осуществляется через регистры MAX.

Таблица 5.1 – Регистр DATA_IND

Бит	Доступ	Описание
0..7	чтение/запись	Данные отправляемые БИС.

Таблица 5.2 – Регистр C_IND

Бит	Название флага	Доступ	Описание
0	E	Запись	Бит управляет синхроимпульсом. При подаче «1» в этот бит происходит захват данных на шине БИС (соединена с DATA_IND) и выполнение команды.
1	RW		Переключатель чтение запись: 1- Чтение 0- Запись
2	RS		Переключатель команда/данные 1 – Данные 0 – Команда

Продолжение таблицы 5.2

3	Reserved		В некоторых экземплярах используется для питания и других тех. целей. Для корректной работы необходимо этот бит всегда устанавливать в «1».
4..7	Not used	-	-

Выполнение команд осуществляется подачей флага E в регистре C_IND. Перед подачей которого необходимо установить данные в регистр DATA_IND и флаги RW и RS. После подачи разрешающего сигнала БИС начинает выполнение команды, и до тех пор, пока команда не будет выполнена БИС будет игнорировать все запросы, поэтому необходимо либо обеспечивать необходимые задержки, либо проверять флаг занятости перед тем как отправить следующую команду. В таблице 5.3 представлены комбинации флагов RS и RW.

Контроллер обладает собственной памятью на 80 байт, представленной в виде строк длиной 16 символов. Для адресации внутренней памяти контроллер использует 7-ми разрядные адреса. В таблице 5.4 показана адресация памяти.

Таблица 5.3 – Комбинации флагов RS и RW

Комбинация		Описание
RS	RW	
0	0	Запись из DATA_IND в регистр команды IR и её выполнение.
0	1	Чтение флага занятости. Если БИС занята то новые команды не принимаются.
1	0	Запись в память DDRAM или CGRAM.
1	1	Чтение в DATA_IND из DDRAM или CGRAM.

Таблица 5.4 – Адресация памяти

Старшие биты (строка)			Младшие биты (символ в строке)			
A6	A5	A4	A3	A2	A1	A0

Для отображения на экране используются две строки (так называемая DDRAM):

– нулевая: верхняя строка на дисплее;

– четвёртая: нижняя строка на дисплее.

Адреса символов на экране представлены в таблице 5.5.

Таблица 5.5 - Адреса символов на экране

0x00	0x01	0x02	0x03	…	0x0E	0x0F
0x40	0x41	0x42	0x43	…	0x4E	0x4F

Инициализация LCD

Перед использованием дисплея его сначала нужно инициализировать в соответствии с протоколом указанным производителем. Протокол инициализации LCD показан на рисунке 5.1.

Рисунок 5.1 - Протокол инициализации LCD

Текст программы для инициализации дисплея

```
void SetLcdData(unsigned char ucNewData)  // помещает байт в регистр данных LCD
{
        unsigned char xdata * ucxLcdData = 0x01;
        unsigned char ucLastDPP = DPP;

        DPP = 0x08;
        *ucxLcdData   = ucNewData;
        DPP = ucLastDPP;
}
void LcdBisEnable(char cReadWrite /*1-read 0-write*/, char cDataCode /*1-data 0-command*/)
{
        unsigned char ucStrobe = 0;
        unsigned char xdata * ucxLcdData = 0x06;
        unsigned char ucLastDPP = DPP;
        int i = 0;
        DPP = 0x08;
        ucStrobe |= 0x9; // сигнал E = 1  и Reserved = 1;
        if( cReadWrite)ucStrobe |= 0x2; // бит RW
        if( cDataCode) ucStrobe |= 0x4; // бит RS
        *ucxLcdData   = ucStrobe;  // запись в регистр
        for( i = 0; i < 10; i++)continue;
        *ucxLcdData   = ucStrobe & 0xFE; // обнуление разрешающего сигнала E
        DPP = ucLastDPP;
        for( i = 0; i < 300; i++) continue; // задержка на исполнение команды
}
void LcdInit()
{
        int i = 0;
        for(i = 0; i < 4000; i++)continue; // задержка для ожидания конца процесса перехвата
        SetLcdData(0x30);    // 00110000 – установка битности дисплея 8bit
        LcdBisEnable(0x0,0x0);  //
        for(i = 0; i < 1500; i++)continue; // задержка по протоколу
        SetLcdData(0x30);    // 00110000 – установка битности дисплея 8bit
        LcdBisEnable(0x0,0x0);  //
        for(i = 0; i < 50; i++)continue; // задержка по протоколу
        SetLcdData(0x30);    // 00110000 – установка битности дисплея 8bit
        LcdBisEnable(0x0,0x0);  //
        for(i = 0; i < 150; i++)continue; // задержка по протоколу
        SetLcdData(0x38);    // 00111000 – установка битности дисплея
        LcdBisEnable(0x0,0x0);
        for(i = 0; i < 150; i++)continue; // задержка по протоколу
        SetLcdData(0x08);    // 00001000 – отключение дисплея
        LcdBisEnable(0x0,0x0);
        for(i = 0; i < 150; i++)continue; // задержка по протоколу
        SetLcdData(0x01);    // 00000001 – отключение дисплея
        LcdBisEnable(0x0,0x0);
        for(i = 0; i < 1500; i++)continue; // задержка по протоколу

        SetLcdData(0x06);    // 00000110 – начальные установки – направление текста  лев->прав
```

```
    LcdBisEnable(0x0,0x0);
    for(i = 0; i < 150; i++)continue; // задержка по протоколу

    SetLcdData(0x0c);    // 00001100 – включение дисплея
    LcdBisEnable(0x0,0x0);
    for(i = 0; i < 150; i++)continue; // задержка по протоколу
}
```

5.2 Лабораторная работа № 5. Получение навыков работы с LCD дисплеем

Цель: написание библиотеки с функциями управления дисплеем для последующего использования.

Задание: написать программу, позволяющую выполнить действия, соответствующие номеру варианта.

Основными функциями системного таймера является измерение интервалов времени и выполнение периодических задач. В данной работе с помощью таймеров требуется управлять светодиодным индикатором или звуковым излучателем, входящими в состав контроллера SDK-1.1. Драйвер системного таймера должен включать функции, указанные в таблице 5.6.

Таблица 5.6 – Функции системного таймера

Функция	Описание
void InitTimer (void)	Инициализация таймера.
unsigned long GetMsCounter (void)	Получение текущей метки времени в миллисекундах.
unsigned long DTimer (unsigned long t0)	Измерение количества миллисекунда, прошедших с временной метки t0 до текущего времени.
void DelayMs (unsigned long t)	Задержка на t миллисекунд.

Драйвер ЖКИ должен включать функции, указанные в таблице 5.7.

Таблица 5.7 – Функции ЖКИ

Функция	Описание
void InitLCD (void)	Инициализация ЖКИ.
void WriteControlLCD (unsigned char ch)	Запись значения в регистр управления ЖКИ C_IND (ПЛИС): ch – значение, записываемое в C_IND.
bit ReadFLCD (void)	Чтение флага BF (флаг занятости контроллера ЖКИ).
unsigned char ClearLCD (void)	Очистка дисплея с возвратом результата выполения операции.
unsigned char GotoXYLCD (unsigned char x, bit y)	Переход в заданную позицию дисплея с возвратом результата выполнения операции: x, y – координаты позиции.
unsigned char PrintCharLCD (unsigned char symbol)	Вывод символа на дисплей с возвратом результата выполнения операции: symbol – выводимый символ.

Кроме того, может быть реализована функция вывода строки на ЖКИ, функция дополнительной настройки ЖКИ (отображение, мерцание курсора). Драйвер клавиатуры использует прерывание таймера, в котором производится опрос состояния кнопок. В данном случае можно применить автоматное программирование. В драйвер клавиатуры рекомендуется включить функцию чтения нажатых кнопок из буфера, связывающего ее с обработчиком прерывания. Реакции на нажатия кнопок клавиатуры должны формироваться в главной программе. Вся обработка нажатий кнопок не должна быть локализована в обработчике прерываний таймера.

В таблице 5.8 приведены варианты заданий.

Таблица 5.8 – Варианты заданий

Номер варианта	Задание
1	Заполнить экран случайными символами из латинского алфавита.
2	Вывести символ «*» и управлять его передвижением с помощью клавиатуры.

Продолжение таблицы 5.8

Номер варианта	Задание
3	Вывести несколько ASCIIZ строк, используя вертикальную прокрутка с помощью клавиатуры.
4	С помощью символа 0xFF и пробела заполнить дисплей в шахматном порядке.
5	Заполнить экран символами, полученными с клавиатуры и затем вывести количество нажатых кнопок на экран.
6	Получить в программе число типа float (например, с помощью функции sin()) и вывести его в формате с фиксированной точкой (4-5 символов после точки).
7	Вывести случайное число в десятичной системе и продублировать его же в двоичной и шестнадцатеричной системах счисления.
8	Разработать устройство, измеряющее интервал времени, в пределах от 0 до 10 секунд с точностью до 0,01 секунды. Начало и конец интервала задаётся нажатием клавиши на клавиатуре учебного стенда SDK 1.1. Результат измерения вывести на ЖКИ стенда в виде четырёхпозиционного десятичного кода. Измеренный интервал сравнить с эталоном, указанным преподавателем. Если измеренное значение интервала времени больше эталона, то зажечь светодиод 1, если меньше, то светодиод 8, а если равно, то все светодиоды.
9	Разработать устройство, имитирующее работу таймера обратного отсчёта. На клавиатуре должны быть отведены три клавиши для таймера: Старт, Стоп и Сброс. Точность отображения времени равна 0,1 секунды. Ход времени необходимо отображать на ЖКИ микроконтроллера. Перед началом работы вводится значение таймера в интервале от 00.1 до 99.0 секунд. При нажатии Сброса таймер должен возвращаться к первоначальному значению, а при достижении нуля микроконтроллером воспроизводится звук любой тональности.
10	Разработать устройство, осуществляющее счёт времени в прямом или обратном порядке. На клавиатуре должны быть отведены три клавиши: Счёт, Обратный счёт, Стоп. Точность отображения времени 0,1 секунды, а интервал счёта равен от 00.0 до 99.0 секунд. При достижении границ интервала счёта таймер останавливается.

В результате работы устройства на дисплее микроконтроллера должна быть отражена величина наибольшего из измеренных отрезков времени в виде четырёхпозиционного десятичного кода. А также номер измерения. Сигналом начала и окончания отсчёта служит нажатие клавиши на клавиатуре микроконтроллера, а индикатором счёта является мерцание светодиодов (1 раз в секунду или в полсекунды).

Пример. Написать программу, обеспечивающую получение строки из символов и выведение её на экран дисплея.

Листинг программы

Файл main.c

```
#include <ADUC812.H> // подключение модуля внутренней памяти
#include "lcd.c" // подключение модуля дисплея
#include "LED.c"// подключение модуля светодиодов

void main(void)  // заголовок главной функции
{
// описание типов данных
        unsigned char ucCol = 1;
        unsigned char ucRow = 0;
        char cDir = 1;
        unsigned char ucComand;
        unsigned short i = 0;


   SetLed(0xfe); // зажигание светодиода
        LcdInit(); // инициализация LCD
        SetLed(0x00); // выключение светодиодов

        while(1) // запуск бесконечного цикла
        {
                ucCol   += cDir; // увеличение ucCol на значение cDir
                if(ucCol == 0) // сравнение ucCol с 0
                {
                        cDir = +1; // увеличение cDir
                        ucRow = 0; // обнуление ucRow
                }
                if(ucCol == 15) // если ucCol = 15
                {
                        cDir = -1; // уменьшение значения переменной cDir
                   ucRow = 1; // присвоение ucRow значения равного единице
                }

                ucComand = 0x80 | (ucCol & 0xf) | (ucRow ? 0x40:0);
                SetLed(ucComand); // зажечь светодиоды
```

```
                SetLcdData(ucComand);  // вывод на LCD
                LcdBisEnable(0,0);

                SetLcdData(0x43);
                LcdBisEnable(0,1);
                for(i = 0; i < 20000; i++)continue;
                SetLcdData(1);
                LcdBisEnable(0,0);
                for(i = 0; i < 20000; i++)continue;

        }
    }
```

Файл lcd.c

```
    void SetLcdData(unsigned char ucNewData)  // помещает байт в регистр данных LCD
    {
        // определение типов данных
        unsigned char xdata * ucxLcdData = 0x01;
        unsigned char ucLastDPP = DPP;

        DPP = 0x08; // присвоение DPP адреса памяти 0x08
        *ucxLcdData   = ucNewData;
        DPP = ucLastDPP;
    }

    void LcdBisEnable(char cReadWrite /*1-read 0-write*/, char cDataCode /*1-data 0-
command*/)
    {
        // определение типов данных
        unsigned char ucStrobe = 0;
        unsigned char xdata * ucxLcdData = 0x06;
        unsigned char ucLastDPP = DPP;
        int i = 0;
        DPP = 0x08;

        ucStrobe |= 0x9; // сигнал E = 1  и Reserved = 1;
        if( cReadWrite)ucStrobe |= 0x2; // бит RW
        if( cDataCode) ucStrobe |= 0x4; // бит RS

        *ucxLcdData   = ucStrobe;  // запись в регистр
        for( i = 0; i < 10; i++)continue;
        *ucxLcdData   = ucStrobe & 0xFE;  //обнуление разрешающего сигнала E
        DPP = ucLastDPP;
        for( i = 0; i < 300; i++) continue; // задержка на исполнение команды
    }

    void LcdInit() // функция инициализации LCD
    {
        int i = 0;
        for(i = 0; i < 4000; i++)continue; // задержка для ожидания конца переходных
процессов
```

```
        SetLcdData(0x30);    // 00110000 – установка битности дисплея 8bit
        LcdBisEnable(0x0,0x0);  //
        for(i = 0; i < 1500; i++)continue; // задержка по протоколу

        SetLcdData(0x30);    // 00110000 – установка битности дисплея 8bit
        LcdBisEnable(0x0,0x0);  //
        for(i = 0; i < 50; i++)continue; // задержка по протоколу
                SetLcdData(0x30);    // 00110000 - установка битности дисплея 8bit
LcdBisEnable(0x0,0x0);  //
        for(i = 0; i < 150; i++)continue; // задержка по протоколу
                                        SetLcdData(0x38);    // 00111000 - установка битности
                                дисплея 8bit и 2 линии
        LcdBisEnable(0x0,0x0);
        for(i = 0; i < 150; i++)continue; // задержка по протоколу

        SetLcdData(0x08);    // 00001000 – отключение дисплея
        LcdBisEnable(0x0,0x0);
        for(i = 0; i < 150; i++)continue; // задержка по протоколу

        SetLcdData(0x01);    // 00000001 - отключение дисплея
        LcdBisEnable(0x0,0x0);
        for(i = 0; i < 1500; i++)continue; // задержка по протоколу

                                        SetLcdData(0x06);    // 00000110 – начальные установки –
                                направление                  текста влево->вправо
        LcdBisEnable(0x0,0x0);
        for(i = 0; i < 150; i++)continue; // задержка по протоколу

        SetLcdData(0x0c);    // 00001100 – включение дисплея
        LcdBisEnable(0x0,0x0);
        for(i = 0; i < 150; i++)continue; // задержка по протоколу
}
```

Пример. Разработать устройство, имитирующее работу секундомера. На клавиатуре должны быть отведены три клавиши: Старт, Стоп, Сброс. Точность отображения времени секундомером равна 0,1 секунды. Ход времени отображать на ЖКИ микроконтроллера. При достижении 99 секунд секундомер автоматически останавливается и микроконтроллером воспроизводится звук любой тональности.

Листинг программы

```
// подключение необходимых модулей
#include <ADUC812.H>
#include "led.h"
#include "kbd.h"
#include "sound.h"
#include "interput.h"
```

```
#include "lcd.h"

void TimeToStr(void);          // функция перевода времени в строку
// объявление переменных
unsigned int uiCounter = 0;
unsigned int uiTime = 1;       // 1/10 seconds
char szTime[6]="-----";
unsigned char ucKey;
void TimerInt(void) interrupt 1 using 2 // функция, срабатывающая по прерыванию от таймера
{
        unsigned char ucOldPage;     // переменная хранит старое значение регистра DPP
        unsigned int i = 0;
        ET0 = 0;         // отключение таймера
        ucOldPage = DPP;
        uiCounter++;  // инкремент счетчика срабатывания прерываний
        if (uiCounter >= 0x15) // если количество прерываний больше 15h
        {
                SetLed(uiTime);        // вывод значения времени
                uiTime += 1;   // инкремент переменной времени
                if (uiTime == 10000) // если переменная времени = 10000
                {
                        TimeToStr(); // преобразование времени в строку
                   LCD_GotoXY(5,0);          // установка курсора LCD
                        LCD_Type(szTime); // вывод времени на LCD
                        SetLed(0); // выключение светодиодов
                        for (i=0; i<100000; i++) // воспроизвести звуковой сигнал
                        {
                                Snd_Enable(1); // включить динамик
                                pause(100); // задержка
                                Snd_Enable(0); // выключить динамик
                                pause(100); // задержка
                        }
                        return;
                }
                uiCounter = 0;// обнуление счетчика
                TimeToStr();  // преобразование времени в строку

                if (uiTime%10 == 5) // вывод значения на экран через каждые 0,5 с
                {
                        LCD_GotoXY(5,0);
                        LCD_Type(szTime);
                }
        }
        else
        {
                TH0 = TL0 = 0; // обнуление таймера
                TR0 = 1; // включение прерывания
        }
        DPP = ucOldPage; // вернуть старое значение регистра
      ET0 = 1; // включить таймер
}
```

```
int main(void)
{
	SetVect(0x0B,(void*)TimerInt); // установить вектор прерывания таймера по адресу 0x0B
	TH0 = TL0 = 0; // обнуление таймера
	LCD_InitDefault(); // инициализация LCD
		while(1)
	{
		if (KB_Hit(&ucKey)) // ожидание нажатия клавиши
			if (~ucKey==0x11) // если нажата единица
			{
				IE = 0x82; // разрешение прерывания
				TMOD = 0x2;
				TCON = 0x10;
				uiTime = 0;
				break;
			}
	}
		EA = 1;		// включение прерывания
	while(1)
	{
		ET0 = 0; // выключить таймер
		if (KB_Hit(&ucKey)) // ожидание нажатия клавиши
		{
			 ET0 = 1; // включить таймер
			 switch(~ucKey) // выбор действия
			 {
				case 0x11:  // включение секундомера
					TMOD = 0x2;
					TCON = 0x10;
					break;
				case 0x12: // выключение секундомера
					TR0 = 0;
					break;
				case 0x21: // сброс секундомера
					SetLed(2);
					uiTime = 0;
					TH0 = TL0 = 0;
					TR0 = 0;
					TimeToStr();
					LCD_GotoXY(5,0); // установка курсора LCD
					LCD_Type(szTime); // вывод на LCD
					SetLed(0); // выключение светодиодов
					break;
			 }
		}
		else
			ET0 = 1;
		pause(500);
	}
}
```

```
void TimeToStr(void) // функция преобразования значения переменной времени в строку
{
        szTime[0] = (uiTime/1000) % 10 + '0';
        szTime[1] = (uiTime/100) % 10 + '0';
        szTime[2] = ',';
        szTime[3] = (uiTime/10) % 10 + '0';
        szTime[4] = uiTime % 10 + '0';
        szTime[5] = 0; }
```

5.3 Лабораторная работа № 6. Реализация функции калькулятора

Цель: написание программы, выполняющей одно из арифметических действий. Операнды вводятся с клавиатуры. Результат отображается на ЖКИ.

Задание: написать программу, позволяющую выполнить действия, соответствующие номеру варианта. В таблице 5.9 приведены варианты заданий.

Таблица 5.9 – Варианты заданий

№	Задание
1	Производить целочисленное деление 10-тизначного десятичного числа на 9-тизначное.
2	Производить вычитание 10-значных десятичных чисел.
3	Производить умножение 6-значных десятичных чисел.
4	Производить сложение 10-значных десятичных чисел.
5	Производить вычитание 10-значных десятичных чисел.
6	Производить сложение 10-значных десятичных чисел.
7	Производить умножение 6-значных десятичных чисел.
8	Производить целочисленное деление 10-значного десятичного числа на 9-значное.
9	Производить сложение 10-значных десятичных чисел.
10	Производить умножение 6-значных десятичных чисел.

Пример. Написать программу, позволяющую производить сложение 10-значных десятичных чисел.

Листинг программы

```
#include <ADUC812.H>  // подключение модуля внутренней памяти
```

```
#include <string.h>  // подключение модуля строк
#include<stdlib.h> подключение модуля библиотечной функции для формирования суммы
#include <stdio.h>// подключение модуля библиотечной функции для вывода информации
// заголовок главной функции
void main(void)
// описание типов данных
{
unsigned short i = 0;
unsigned short j = 0;
unsigned char key;
char buffer [33];
int num=0;
int rez;

// инициализация дисплея
LcdInit();
  while(1)
// выбираем ключ
{
if(KB_Hit(&key))
{
switch(key)
// формируем задержки, вводим значение, считаем и выводим на экран дисплея
{
case 0x7d: SetLcdData('0'); LcdBisEnable(0,1);for(i = 0; i < 20000; i++)continue; num*=10; num+=0; break;
case 0xee:  SetLcdData('1'); LcdBisEnable(0,1);for(i = 0; i < 20000; i++)continue; num*=10; num+=1; break;
case 0xed:  SetLcdData('2'); LcdBisEnable(0,1);for(i = 0; i < 20000; i++)continue; num*=10; num+=2; break;
case 0xeb:  SetLcdData('3'); LcdBisEnable(0,1);for(i = 0; i < 20000; i++)continue; num*=10; num+=3; break;
case 0xde:  SetLcdData('4'); LcdBisEnable(0,1);for(i = 0; i < 20000; i++)continue; num*=10; num+=4; break;
case 0xdd:  SetLcdData('5'); LcdBisEnable(0,1);for(i = 0; i < 20000; i++)continue; num*=10; num+=5; break;
case 0xdb:  SetLcdData('6'); LcdBisEnable(0,1);for(i = 0; i < 20000; i++)continue; num*=10; num+=6; break;
case 0xbe:  SetLcdData('7'); LcdBisEnable(0,1);for(i = 0; i < 20000; i++)continue; num*=10; num+=7; break;
case 0xbd:  SetLcdData('8'); LcdBisEnable(0,1);for(i = 0; i < 20000; i++)continue; num*=10; num+=8; break;
case 0xbb:  SetLcdData('9'); LcdBisEnable(0,1);for(i = 0; i < 20000; i++)continue; num*=10; num+=9; break;
case 0x7e: SetLed(0x10); SetLcdData(1); LcdBisEnable(0,0); rez=num; num=0; break; //плюс
case 0x7b: // переменные результата;
 rez+=num; // вывод строки
sprintf(buffer,"%d",rez); // вывести на экран
toLCD(&buffer,5,1,10000);
break;  //равно}}}}
```

6 Работа с прерываниями

6.1 Система прерываний стенда SDK-1.1

Для обработки некоторых медленно работающих устройств можно использовать, аппаратные прерывания. Процессор ADuC812BS установленный на стенде обеспечивают восемь источников прерывания и два уровня приоритета. Как и во всех процессорах семейства Intel, таблица векторов прерываний расположена в начале памяти программ, однако, на стенде первые восемь килобайт адресного пространства отведены под сервисные программы и находятся под защитой от записи, и доступ к ним возможен только в режиме перепрограммирования FLASH-памяти. Для решения этой проблемы в начале памяти были размещены команды перехода на адрес равный 0x2000 + адрес данного вектора, то есть вектора программным путём отображаются на интервал памяти 0x2003- 0x2043 названный пользовательской таблицей векторов. Именно поэтому рекомендуется записывать программы начиная с адреса 0x2100. В таблице 6.1 приведен список прерываний ADuC812.

Таблица 6.1 - Прерывания ADuC812

Прерывание	Наименование	Адрес вектора	Приоритет
PMSI	Источник питания ADuC812.	43H	1
IE0	Внешнее прерывание INT0.	03H	2
ADCI	Конец преобразования АЦП.	33H	3
TF0	Переполнение таймера 0.	0BH	4
IE1	Внешнее прерывание INT1.	13H	5
TF1	Переполнение таймера 1.	1BH	6
I2CI/ISPI	Прерывание последовательного интерфейса (I^2C, SPI).	3BH	7
R1/T1	Прерывание UART.	23H	8
TF2/EXF2	Переполнение таймера 2.	2BH	9

Приведем пример помещения собственного вектора в пользовательскую таблицу. Пусть требуется осуществить обработку прерываний от таймера 0 (прерывание 0Bh). В программу на языке Си (компилятор фирмы Keil Software) можно вставить следующий код:

```
void TO_ISR(void) interrupt 1 // Обработчик прерывания от таймера 0
{
// Действия, выполняемые обработчиком
}
void SetVector(unsigned char xdata * Address, void * Vector)
// Функция, устанавливающая вектор прерывания Vector по адресу Address
{
unsigned short xdata * TmpVector;  // Временная переменная
*Address = 0x02;   // Первым байтом по указанному адресу записывается код команды
// передачи управления ljmp, равный 02h
TmpVector = (unsigned short xdata *) (Address+1) ;
*TmpVector = (unsigned short) Vector; // Далее записывается адрес перехода Vector
// Таким образом, по адресу Address теперь располагается инструкция ljmp Vector
}
void main(void)
{
/*...*/
SetVector(0x200B, (void *) TO_ISR); // Установка вектора в пользовательской таблице
ET0=1; EA=1;           // Разрешение прерываний от таймера 0
/*...*/
}
```

Ниже, на рисунке 6.1 проиллюстрировано использование прерываний в SDK-1.1.

В блоке регистров специальных функций есть два регистра, предназначенных для управления режимом прерываний IE и уровнями приоритета IP. Возможность программной установки/сброса любого управляющего бита в этих двух регистрах делает систему прерываний 8051 исключительно гибкой. В таблицах 6.2 и 6.3 описаны назначения битов регистров IE и IP.

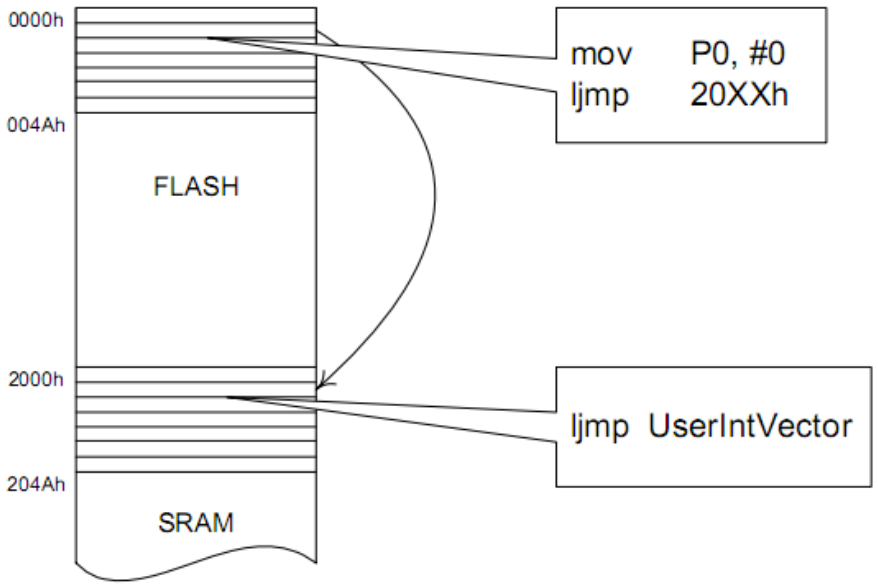


Рисунок 6.1 – Использование прерываний в SDK-1.1

Таблица 6.2 – Регистр масок прерывания (IE)

Символ	Позиция	Имя и назначение
1	2	3
EA	IE.7	Снятие блокировки прерывания. Сбрасывается, программно для запрета всех прерываний независимо от состояний IE.4 - IE.0 .
	IE.6	Не используется.
	lE.5	Не используется.
ES	IE.4	Бит разрешения прерывания, от приемопередатчика Установка/сброс программой для разрешения/запрета прерываний от флагов TI или RI .
ET1	IE.3	Бит разрешения прерывания от таймера. Установка/сброс программой для разрешения/запрета прерываний от таймера 1 .
EX1	IE.2	Бит разрешения внешнего прерывания 1. Установка/сброс программой для разрешения/запрета прерывания 1 .
ET0	IE.1	Бит разрешения прерывания от таймера 0. Установка/сброс программой для разрешения/запрета прерываний от таймера 0.
EX0	IE.0	Бит разрешения внешнего прерывания 0. Установка/сброс программой для разрешения/запрета прерывания 0 .

Таблица 6.3 – Регистр приоритетов прерываний (IP)

Символ	Позиция	Имя и назначение
-	IP.7 - IP.5	Не используется.
PS	IP.4	Бит приоритета приемопередатчика. Установка/сброс программой для присваивания прерыванию от приемопередатчика высшего/низшего приоритета.
PT1	IP.3	Бит приоритета таймера 1. Установка/сброс программой для присваивания прерыванию от таймера 1 высшего/низшего приоритета.
PX1	IP.2	Бит приоритета внешнего прерывания 1. Установка/сброс программой для присваивания высшего/низшего приоритета внешнему прерыванию INT1.
PT0	IP.1	Бит приоритета таймера 0. Установка/сброс программой для присваивания прерыванию от таймера 0 высшего/низшего приоритета.
PX0	IP.0	Бит приоритета внешнего прерывания 0. Установка/сброс программой для присваивания высшего/низшего приоритета внешнему прерыванию INT0.

Система прерываний формирует аппаратный вызов (LCALL) соответствующей подпрограммы обслуживания, если она не заблокирована одним из следующих условий:

- в данный момент обслуживается запрос прерывания равного или высокого уровня приоритета;
- текущий машинный цикл - не последний в цикле выполняемой команды;
- выполняется команда RETI или любая команда, связанная с обращением к регистрам IE или IP.

Отметим, что если флаг прерывания был установлен, но по одному из указанных выше условий не получил обслуживания и к моменту окончания блокировки уже сброшен, то запрос прерывания теряется и нигде не запоминается.

По аппаратно сформированному коду LCALL система прерывания помещает в стек только содержимое счетчика команд (PC) и загружает в него

адрес вектора соответствующей подпрограммы обслуживания. По адресу вектора должна быть расположена команда безусловной передачи управления (JMP) к начальному адресу подпрограммы обслуживания прерывания. В случае необходимости она должна начинаться командами записи в стек (PUSH) слова состояния программы (PSW), аккумулятора, расширителя, указателя данных и т.д. и должна заканчиваться командами восстановления из стека (POP). Подпрограммы обслуживания прерывания должны завершаться командой RETI, по которой в счетчик команд перезагружается из стека сохраненный адрес возврата в основную программу. Команда RET также возвращает управление прерванной основной программе, но при этом не снимут блокировку прерываний, что приводит к необходимости иметь программный механизм анализа окончания процедуры обслуживания данного прерывания.

Существует два способа перехвата прерываний: писать программу с адреса перехватываемого прерывания; запись команды безусловного перехода на функцию-обработчик в пользовательскую таблицу векторов.

После того как прерывание было перехвачено необходимо разрешить процессору вызывать эти прерывания. Список разрешенных прерываний, находящихся в регистре IE, приведен в таблице 6.4.

Таблица 6.4 – Список разрешенных прерываний

IE		
Бит	Битовый регистр	Описание.
0x7	EA	Все прерывания.
0x6	EADC	Прерывания от ЦАП.
0x5	ET2	Прерывание второго таймера.
0x4	ES	Прерывание от последовательного порта.
0x3	ET1	Прерывание первого таймера.
0x2	EX1	Внешние прерывания 1.
0x1	ET0	Прерывания нулевого таймера.
0x0	EX0	Внешние прерывания.

После того как прерывания будут разрешены, они будут обрабатываться назначенным обработчиком.

6.2 Лабораторная работа № 7. Работа с прерываниями и часами реального времени

Цель работы:

- изучить систему прерываний микроконтроллера и источники возможных прерываний;

- научиться создавать собственные процедуры обработки прерываний;

- изучить работу микросхемы PCF8583 – часы/календарь;

- изучить команды для программирования регистров управления и состояния микросхемы часов времени.

Описание работы

Драйвер часов реального времени должен включать функции, приведенные в таблице 6.5.

Таблица 6.5 – Функции часов реального времени

Функция	Описание
void InitRTC (void)	Инициализация часов реального времени.
unsigned char ReadRTC (TimeDate *td)	Чтение даты и времени из RTC с возвратом результата выполнения операции: td – буфер даты и/или времени в виде структуры специального формата.
Unsigned char WriteRTC (TimeDate *td)	Запись даты и времени из RTC с возвратом результата выполнения операции: td – буфер для даты и/или времени в виде структуры специального формата.

Варианты заданий:

Вариант 1

Перехватить внешнее прерывание INT1. Процедура обработки данного прерывания должна запускать таймер (секунды) встроенный в часы реального

времени. На ЖКИ выводится значения таймера, а по окончании счёта микроконтроллер должен автоматически воспроизвести сигнал, т.е. должен быть разрешён сигнал по таймеру.

Вариант 2

Перехватить внешнее прерывание INT0. Процедура обработки данного прерывания должна устанавливать ежедневный сигнал для часов реального времени на 15 секунд больше, чем текущее время и отобразить текущее время в формате ЧАС: МИН:СЕК на ЖКИ.

Вариант 3

Создать свои процедуры обработки следующих прерываний: переполнение таймера; внешнее прерывание.

Перехватить внешнее прерывание INT1 или INT0 (по выбору), процедура обработки данного прерывания должна запускать таймер (один из таймеров по выбору).

Перехватить прерывание переполнения таймера, процедура обработки его должна вывести на ЖКИ текущее время в формате 12-ти часового отображения (AM/PM) с точностью до сотых секунды и дату (месяц и число). Время и дату брать с часов реального времени микроконтроллера.

Вариант 4

Перехватить прерывание переполнения таймера 0. Процедура обработки данного прерывания должна вывести на ЖКИ текущую дату в следующем формате: День (прописью) : Число : Месяц (прописью) : Год.

Вариант 5

Перехватить прерывание переполнения таймера 1. Процедура обработки данного прерывания должна запускать таймер (минуты) встроенный в часы реального времени а также отобразить текущее время в формате ЧАС:МИН:СЕК на ЖКИ.

Первоначальное значение таймера выбирается так, чтобы задержка была не более 2-х минут. При переполнении таймера микроконтроллер должен

автоматически воспроизвести сигнал, т.е. должен быть разрешён сигнал по таймеру.

Пример. Перехватить внешнее прерывание INT0. Процедура обработки данного прерывания должна устанавливать ежедневный сигнал для часов реального времени на 15 секунд больше, чем текущее время и отобразить текущее время в формате ЧАС:МИН:СЕК на ЖКИ.

Листинг программы

```
// подключение необходимых модулей
#include <ADUC812.H>
#include "led.h"
#include "interput.h"
#include "rtc.h"
#include "sound.h"
#include "lcd.h"
#include "string.h"
// описание переменных
TIME xdata time,time2;
bit req = 0;
void OutTime(); // функция вывода времени
void Int0Handler(void) interrupt 0 using 2 // функция, срабатывающая по внешнему прерыванию int0
{ bit res = 0; // объявление и обнуление переменной res
  EX0 = 0; // отключение внешнего прерывания
  if (req == 1) return; // проверяем на рекурсию
  req = 1; // установка переменной рекурсии в 1
  EX0 = 1;   // включение внешнего прерывания }
void main()
{       unsigned char n,r; // объявление переменных
        int res = 0;
        LCD_InitDefault(); // инициализация LCD
        EA  = 1; // отключение прерывания
        EX0 = 1; // включение прерывания
        SetVect(0x3,Int0Handler); // установка вектора внешнего прерывания
        time2.sec = 0; // обнуление секунд
        res = SetTime(&time2); // установка времени
        while(1)
        {       EA = 0;  // отключение прерывания
                OutTime(); // вывод времени
                EA = 1; // включение прерывания
                pause(1000); // задержка
                EA = 0;  // отключение прерывания
                if (req == 1)  // если прерывание уже сработало
                {       GetTime(&time); // получение значения времени
                time.sec += 15; // прибавление 15 с к значению времени
```

```
                        n = time.sec / 60; // нормализация времени
                        r = time.sec % 60;
                        time.min += n;
                        time.sec = r;
                        SetTime(&time);       // установка времени
                        req = 0; // обнуление переменной req
                        pause(1000); // задержка
                        EX0 = 1; // включить внешнее прерывание
                }
                EA = 1;  // разрешить прерывание
        }
}
void OutTime() // функция вывода времени
{       char str[11];
        memset(str,'-',11);
        GetTime(&time2); // получение значения времени
        str[10] = 0;
        str[9] = ((time2.sec ) % 10)  + '0';
        str[8] = ((time2.sec / 10) % 10) + '0';
        str[7] = ':';
        str[6] = ((time2.min ) % 10)  + '0';
        str[5] = ((time2.min / 10) % 10) + '0';
        str[4] = ':';
        str[3] = ((time2.hour ) % 10)  + '0';
        str[2] = ((time2.hour / 10) % 10) + '0';
        str[1] = ':';
        str[0] = 'T';
        SetLed(time2.sec); // вывод секунд на светодиоды
         LCD_GotoXY(0,0); // установка курсора на LCD
        LCD_Type(str); // вывод строки со значение времени
}
```

7 Многозадачность

7.1 Многозадачность при работе с SDK 1.1

Концепция ЭВМ с программным управлением подразумевает выполнение программы последовательно, шаг за шагом. Таким образом, одно вычислительное ядро может выполнять только одну программу в один момент времени, однако часто требуется выполнение нескольких программ одновременно. До 80-х годов двадцатого века существовало два способа решения данной проблемы: либо наращивание количества вычислительных ядер, либо включение в основную программу нескольких функций и поочередное их выполнение. Каждый способ не лишён недостатков, наращивание числа вычислительных ядер приводило к большим финансовым затратам, а также к увеличению размеров ЭВМ, а они в то время и так были немаленькими. В случае последовательного выполнения все подпрограммы были зависимыми друг от друга, и в случае отказов одной из подпрограммы, например выхода в бесконечный цикл или фатальной ошибки, все остальные подпрограммы так же прекращали выполнение. В обоих случаях изменение количества одновременно выполняемых процессов было достаточно затруднено.

Исключающая многозадачность.

В восьмидесятые годы была разработана так называемая концепция исключающей многозадачности. При таком построении основная программы (ядро ОС) оперировала с дескрипторами процессов, в которых описывалась среда выполнения каждого процесса. Под ресурсами в то время понимались процессорное время, область памяти и устройства ввода-вывода. Такие отдельно выполняемые программы получили название процесса, и стали независимыми друг от друга. При таком устройстве вычислительного процесса, добавление новых и завершение работающих процессов перестало представлять какие либо трудности. Необходимо было только добавить дескриптор процесса в очередь на использование ресурсов, либо удалить его из очереди.

Реализация многозадачности в среде uVision

В данном разделе будет описан способ реализации многозадачности на стенде SDK 1.1 с помощью перехвата прерывания таймера. Менеджер ресурсов будет рассчитан всего на два процесса, дабы не осложнять понимание самой концепции многозадачности.

Переменные с фиксированным адресом создаются с помощью директивы _AT_ Для простого менеджера ресурсов потребуется только три переменных: номер активного процесса, и программные счетчики каждого из процессов. А также одна внутренняя переменная – адрес возврата из прерывания. Их определение будет выглядеть так:

```
unsigned char  xdata  cThread    _at_ 0xff00;
unsigned short xdata ucCurAdr    _at_ 0xff10;
unsigned short xdata usRetAdr[2] _at_ 0xff12;
```

Данные переменные должны быть заполнены до того как будет разрешено выполнение прерывания. cThread необходимо приравнять к нулю как показано на примере *((unsigned char xdata*)(0x00ff00)) = 0x00;

Внутри менеджера ресурсов необходимо описать чередование активных процессов и сохранение программного счетчика для активного процесса. Также здесь необходимо установить таймер на необходимое время.

```
TH0 = 0xf0;
ET0 = 1;
if(cThread == 0)
{       usRetAdr[0] = ucCurAdr;
        cThread     = 0x1;
        ucCurAdr    = usRetAdr[1]; }
else
{       usRetAdr[1] = ucCurAdr;
        cThread = 0x0;
        ucCurAdr    = usRetAdr[0]; }
```

Данный код сначала сохраняет содержимое программного (ucCurAdr) счетчика в память и устанавливает новое значение для нового процесса.

Для сохранения значения программного счетчика в переменой ucCurAdr необходимо считать его из адреса возврата прерывания. Например, используя данный код Ассемблера:

```
#pragma asm
        MOV   DPTR,#0xff10
        mov A, SP;
        subb A, #0x5;  // глубина на которой находится младший байт адреса возврата
        MOV R1, A
        MOV A, @r1;
        MOVX @DPTR, A
        DEC R1;
        INC DPTR;
        MOV A, @R1;
        MOVX @DPTR, A
#pragma endasm
```

Для определения смещения адреса возврата относительно вершины стека необходимо откомпилировать проект и просмотреть сгенерированный SRC код и посчитать количество операций PUSH. Для сохранения нового адреса возврата можно использовать такой код:

```
#pragma asm
MOV   DPTR,#ucCurAdr
mov A, SP;
subb A, #0x5;
MOV R1, A
MOVX A, @DPTR
MOV  @R1, A;
DEC R1;
INC DPTR;
MOVX A, @DPTR
MOV  @R1, A;
#pragma endasm
```

Регистр-указатель DPTR можно использовать только совместно с командой MOVX, обрабатывающей обращения к расширенной памяти.

После написания менеджера процессов необходимо указать стартовые адреса для процессов. Узнать их можно во время процесса отладки в окне

дизассемблированного кода. На рисунке 7.1 показан процесс выбора окна дизассемблированного кода.

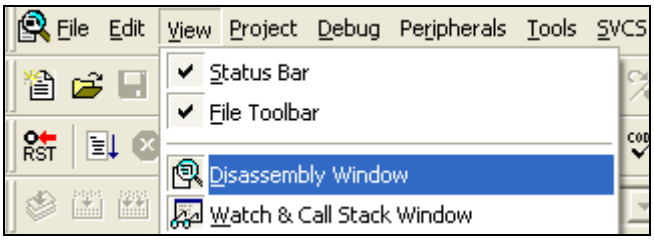


Рисунок 7.1 – Выбор команды дизассемблированного кода

А записать в переменные, используемые менеджером ресурсов, с помощью обращения к ячейке памяти, записав адрес команды с которой начинается процесс: *((unsigned short xdata*)(0x00ff14)) = 0x21db;

Если все сделано правильно, то два процесса будут выполняться одновременно. Отдельные процессы лучше всего размещать в отдельных бесконечных циклах.

7.2 Лабораторная работа № 8. Многозадачность при работе с SDK-1.1

Цель: написание программы, в которой бы выполнялись одновременно задания для лабораторных работ № 1 и № 2, используя исключающую многозадачность.

Задание: расположить в двух бесконечных циклах соответствующие подпрограммы и добиться их одновременного выполнения.

7.3 Лабораторная работа № 9. Разработка системы сбора и обработки информации при работе с SDK-1.1

Цель работы:

- закрепление навыков работы со всеми узлами учебного стенда SDK-1.1;

- изучение команд передачи данных между узлами учебного стенда SDK-1.1.

Описание работы

В рамках лабораторной работы необходимо разработать программу для контроллера SDK-1.1 (ведомый), который обменивается данными с персональным компьютером (ведущий). В качестве канала связи используется последовательный канал RS232. Устройством вывода терминал персонального компьютера. Драйвер клавиатуры должен содержать: функцию инициализации, обработчик прерывания от таймера, циклический буфер клавиатуры (нажатые кнопки), При инициализации необходимо указать задержку перед повтором символа (первый параметр) и скорость повтора символа (второй параметр). Рекомендуемая величина задержки перед повтором – 1 секунда (не меньше).

Варианты заданий.

Вариант 1

Выполнить задачу сбора голосов и их подсчёт. На экран монитора компьютера необходимо вывести список кандидатов с указанием их порядкового номера. При голосовании запрашивается номер кандидата (запрос на ЖКИ). При вводе номера необходимо вывести на ЖКИ учебного стенда SDK-1.1 фамилию кандидата и получить подтверждение о правильности выбора. Количество голосующих задаётся программно. После сбора информации необходимо вывести на экран монитора компьютера результаты голосования и количество голосов отданных за каждого кандидата.

Вариант 2

Подсчёт рейтинга группы. Программно вводится список имён экзаменуемых студентов. После команды начала выставления оценок, на ЖКИ учебного стенда SDK-1.1 последовательно выводятся фамилии студентов. Напротив каждой фамилии надо поставить оценку 1..5 и подтвердить правильность ввода. После сбора информации обо всех студентах, на экран монитора компьютера необходимо вывести общее количество студентов, фамилии студентов с отображением оценки и средний балл (рейтинг) группы.

Вариант 3

Кодировка текста. На ЖКИ учебного стенда SDK-1.1 вводится числовая последовательность, которая представляет собой закодированную текстовую информацию. Разделителем кода символа служит нечисловой символ (*,#,A,B,C,D). Количество символов в последовательности не должно быть более 80, а при заполнении всей рабочей области ЖКИ (16X2), она очищается. При подаче сигнала на раскодирование (нажатие клавиши) на экране монитора компьютера должна быть отображена текстовая строка и количество символов в ней.

Вариант 4

Фильтрация информации. На экране монитора компьютера отобразить фамилии людей (не менее десяти), с указанием года, месяца (числом) и дня рождения. Предусмотреть три варианта фильтрации данных: по году, по месяцу и по дню рождения. На ЖКИ учебного стенда SDK-1.1 отобразить варианты фильтра. Необходимо осуществить выбор фильтра и указать значение для осуществления отбора, при выборе дать запрос на правильность ввода. На экране монитора должна быть отображена отсортированная информация. Необходимо также предусмотреть клавишу возвращения в первоначальный список (неотфильтрованный).

Вариант 5

Тест. На экран монитора компьютера по порядку выводятся вопросы для теста (не менее десяти). На ЖКИ учебного стенда SDK-1.1 должны быть отражены варианты ответов. При вводе ответа должен инициироваться запрос на подтверждение правильности. При окончании тестирования на ЖКИ учебного стенда SDK-1.1 выводится средний бал, а на экране монитора статистика ответов, т.е. номера вопросов и правильность (неправильность) ответа на них, а также общее количество вопросов и правильных ответов.

Пример. Подсчёт рейтинга группы. Программно вводится список имён экзаменуемых студентов. После команды начала выставления оценок, на ЖКИ учебного стенда SDK-1.1 последовательно выводятся фамилии студентов.

Напротив каждой фамилии надо поставить оценку 1..5 и подтвердить правильность ввода. После сбора информации обо всех студентах, на экран монитора компьютера необходимо вывести общее количество студентов, фамилии студентов с отображением оценки и средний балл (рейтинг) группы.

Листинг программы

```
// подключение необходимых модулей

#include <ADUC812.H>
#include "sound.h"
#include "lcd.h"
#include "kbd.h"
#include "string.h"
#include "sio.h"

// объявление переменных

int N = 5;
char mas[5][10];
char* pMas = mas;
char xdata ball[5];
char xdata nCurrentLine;
char key;

void Output(); // функция вывода на LCD
void OutputCom(); // функция вывода на com-порт

void main()

{
        unsigned char iNum = 0; // определение переменной iNum
        nCurrentLine = 0;
        N = 5;  // размерность массива
        strcpy(mas[0],"mr.Ivanov"); // инициализация массива
        strcpy(mas[1],"mr.Petrov");
        strcpy(mas[2],"mr.Eprst");
        strcpy(mas[3],"mr.Andrew");
        strcpy(mas[4],"mr.Pink");

        LCD_InitDefault(); // инициализация LCD
        memset(ball,0,N); // обнуление массива оценок
        Output(); // вывод на экран
        while(1) // запуск бесконечного цикла

        {
          if(KB_Hit(&key)) // сканирование клавиатуры

                {
```

```
                        switch(~key) // определение нажатой клавиши

                        {
                        case 0x18:
                         if (nCurrentLine)
                         nCurrentLine--; // уменьшение значения текущей строки
                        break;

                        case 0x28:
                         if (nCurrentLine!=N-1) // проверка на допустимость значения текущей записи
                        nCurrentLine++; // увеличение значения текущей строки
                        break;

                        case 0x48: // вывод на com-порт
                        OutputCom();
                        break;

                        default: // если нажата клавиша от 0 до 5

                        iNum = KeyCode(~key); // получение текущего значения

                                if (iNum>5) iNum = 5;
                                if (iNum<1) iNum = 1;
                                ball[nCurrentLine] = iNum; // установить оценку текущему пользователю

                        }

                        Output(); // функция вывода
                }
        }

        while(1);
}

void Output() // функция вывода на LCD
{
        char str[16] = "               ";
        double avg = 0;
        int i = 0;
        strncpy(str,"Shtuk:",6); // функция копирования строки в строку
        str[6] = N + '0'; // преобразовать в символ
        for(i = 0; i < N; i++) // подсчет среднего значения оценки

        {
                avg += ball[i];
        }

        avg /= (double)(N);

        strncpy(str+8,"Avg:",4); // формирование строки для вывода
        str[12] = (int)avg % 10  + '0';
        str[13] = ',';
```

```
	str[14] = (int)(avg*10) %10  + '0';
	str[15] = (int)(avg*100) % 10  + '0';
	str[16] = 0;

	LCD_Clear(); // очистка LCD
		LCD_GotoXY(0,0); // установка курсора LCD
	LCD_Type(str);		// вывод на LCD

	LCD_GotoXY(0,1); // установка курсора LCD
	LCD_Type(&mas[nCurrentLine][0]); // вывод на LCD

	str[0] = ball[nCurrentLine] % 10  + '0'; // вывод оценок
	str[1] = 0;
	LCD_GotoXY(11,1); // установка курсора LCD
	LCD_Type(str);		// вывод на LCD

}
void OutputCom() // вывод на com-порт
{
	char str[16] = "               ";
	int j = 0;
	double avg = 0;
	int i = 0;
	strncpy(str,"Shtuk:",6); // формирование строки для вывода
	str[6] = N + '0';
	for(i = 0; i < N; i++)
	{
		avg += ball[i];
	}
	avg /= (double)(N);
	strncpy(str+8,"Avg:",4);
	str[12] = (int)avg % 10  + '0';
	str[13] = ',';
	str[14] = (int)(avg*10) %10  + '0';
	str[15] = (int)(avg*100) % 10  + '0';
	str[16] = 0;

	SIO_Init(S9600,0); // инициализация порта на скорости 9600
	Wsio('\n'); // команда перевода строки
	Wsio('\r'); // команда возврата каретки
	Type(str); // послать сформированную строку
	for (i=0; i<N; i++)
	{
	   Wsio('\n');
	   Wsio('\r');
		Type(&mas[i][0]);
		memset(str,' ',15);
		str[15] = 0;
		str[13] = ball[i] % 10  + '0';
		str[14] = 0;
		Type(str);
	}	}
```

8 Контрольные вопросы

1. Перечислить модули учебного стенда SDK-1.1, описать их функциональное назначение.
2. К какому семейству относится микроконтроллер ADuK812? Назовите тип архитектуры, приведите основные характеристики этого микроконтроллера.
3. Показать визуально и рассказать о назначении всех узлов, переключателей и разъёмов стенда.
4. Какие виды памяти реализованы в стенде? Начертить карту памяти стенда и рассказать о каждом виде памяти стенда.
5. Какой объем занимает внутренняя память стенда, из каких частей она состоит?
6. Какой вид оперативной памяти используется в стенде?
7. Опишите регистры ПЛИС: виды, назначение, адреса, примеры использования.
8. Как осуществляется доступ к регистрам ПЛИС?
9. В чем назначение регистров управления ENA?
10. Назовите все периферийные микросхемы стенда SDK-1.1.
11. Какие порты ввода-вывода есть в стенде? Каким образом осуществляется доступ к портам?
12. Сколько источников прерываний обеспечивает микроконтроллер ADuC812? Сколько уровней приоритетов прерываний? Как установить приоритет прерывания? Где хранятся адреса векторов прерываний?
13. Какие регистры предназначены для управления режимами прерываний?
14. Как используются прерывания в пользовательских программах?
15. ЖКИ – описание функций. Какие регистры имеет ЖКИ? Что хранится в этих регистрах?
16. Что представляет собой память данных ЖКИ? Что такое генератор символов ОЗУ?

17. Что представляет собой микросхема часы/календарь? Какая память в ней присутствует?

18. Какие основные узлы входят в структуру микросхемы часов/календаря?

19. Что представляет собой устройство памяти часов/календаря?

20. Опишите основные функции, режим счета, режим сигнализации часов/календаря.

21. Что представляет собой регистр управления/состояния часов/календаря?

22. Опишите регистры-счетчики, регистр управления сигналом часов/календаря.

23. Какие вы знаете режимы работы таймера часов/календаря.

24. Сколько таймеров-счетчиков входит в состав стенда, опишите схемотехнику таймеров.

25. Для чего предназначены регистры TH0, TL0, TH1, TL1?

26. Какую информацию содержит регистр TCON? Является ли он программно доступным регистром? Какие команды используются для доступа к нему?

27. Какую информацию содержит регистр TMOD? Привести примеры использования данного регистра.

28. Назовите основные этапы программирования стенда.

29. Что такое резидентный загрузчик HEX202? Как осуществляется загрузка программы в память стенда SDK-1.1?

30. Что представляет собой формат записи Intel Hex?

31. Что такое инструментальная система T2? Назовите основные команды этой системы.

32. Что включает в себя набор средств тестирования стенда SDK-1.1?

33. Какие средства разработки для микроконтроллера 8051 включает в себя пакет Keil Software?

34. Назовите отличительные черты компилятора языка Си фирмы Keil Software.

35. Какие типы данных поддерживает компилятор C51?

36. Какие модификаторы памяти и модели памяти поддерживает компилятор C51?

37. Назовите основные функции, используемые при разработке программ.

38. Назовите этапы программирования и отладки стенда SDK-1.1.

39. Для чего используется память EEPROM в стенде SDK-1.1?

40. Что представляет собой интерфейс I2C?

41. Какие вы знаете средства ввода-вывода информации в стенде SDK-1.1?

42. Каким образом можно запрограммировать работу светодиодов?

43. Какие средства необходимо использовать для считывания клавиатуры стенда?

44. В чем заключается инициализация LCD стенда?

45. Как реализовать многозадачность в стенде SDK-1.1?

46. Что такое исключающая многозадачность?

Список использованных источников

1 Андреев, Д.В. Программирование микроконтроллеров MCS-51 : учебное пособие/ Д.Б. Андреев. - Ульяновск: УлГТУ, 2000. - 88с.

2 Белов А.В. Конструирование устройств на микроконтроллерах. СПб.: Наука и техника, - 2005. – 256 с.

3 Бурькова, Е.В. Микропроцессорные системы: Электронное гиперссылочное учебное пособие / Е.В. Бурькова, А.Ю. Батов. Рег. № 90. - Оренбург: УФАП ГОУ ОГУ.-2005. – 112 с.

4 Бурькова, Е.В. Микропроцессорный комплекс SDK-1.1. Архитектура и программирование учебное пособие / Е.В. Бурькова, А.Ю., А.С. Боровский, Оренбургский гос. ун-т. - Оренбург: ОГУ, 2010. – 107 с.

5 Применение инструментального учебного микроконтроллера SDK-1.1 и программного симулятора при изучении дисциплины «Микропроцессорные системы» / Е.В. Бурькова // Современные информационные технологии в науке, образовании и практике: материалы Всероссийской научно-практической конференции. - Оренбург, ГОУ ОГУ, 2004. - С. 96-99.

6 Лукичев, А.Н. Расширение возможностей лабораторного комплекса SDK-1.1. Научно-технический вестник СПбГИТМО (ТУ). Выпуск 10. Информация и управление в технических системах. - СПб.: СПбГИТМО (ТУ), 2003.-С. 86-90.

7 Каспер Э. Программирование на языке Ассемблера для микроконтроллеров семейства i8051. – М.: Горячая линия – Телеком, 2004 – 191 с.

8 Магда, Ю.С. Микроконтроллеры серии 8051: практический подход. — М.: ДМК Пресс, 2008. —228 с.

9 Новиков, Ю.В. Основы микропроцессорной техники / Ю.В. Новиков, П.К. Скоробогатов - М.: ИНТУИТ.РУ. «Интернет-Университет Информационных технологий», 2003.-440с.

10 Новиков, Ю.В. Основы цифровой схемотехники. Базовые элементы и схемы. Методы проектирования / Ю.В. Новиков –М.: Мир 2001.–379 с.

11 Предко, М. Руководство по микроконтроллерам: в 2 т / М. Предко - М.: Постмаркет, 2001. – Т. 1 – 416 с.

12 Предко, М. Руководство по микроконтроллерам: в 2 т / М. Предко - М.: Постмаркет, 2001. – Т. 2 –488 с.

12 Пухальский, Г.И. Проектирование микропроцессорных устройств: учебное пособие для вузов / Г.И. Пухальский - СПб.: Политехника, 2001-544с.

14 Таненбаум, Э. Архитектура компьютера / Э. Таненбаум – СПб.: ПИТЕР, 2003. – 704 с.

15 Учебный стенд SDK-1.1 Руководство пользователя. - СПб.: ООО "ЛМТ", 2001.-99с.

16 Цилькер, Б.Я. Организация ЭВМ и систем: учебник для вузов / Б.Я. Цилькер, С.А. Орлов – СПб.: ПИТЕР, 2006. – 668 с.