

Утверждаю:  
Декан АВТФ  
Мельников Ю.С.

\_\_\_\_\_ 2002г.

**"Изучение лабораторного комплекса SDK – 1.1 "**

Методические указания для проведения лабораторной работы по курсу "Аппаратные и программные средства встраиваемых компьютеров" и микроконтроллеров для студентов специальности 220100 "Вычислительные машины, комплексы, системы и сети"

УДК 681.142

Изучение основ проектирования программного обеспечения микроконтроллеров на базе учебного лабораторного комплекса SDK -1.1  
Метод. указ. для проведения лабораторной работы по курсу «Микропроцессорные системы» для студентов специальности 220100 "Вычислительные машины, комплексы, системы и сети".

Составители: Столяров А.Г., Салит В.В., Меркулов С.В – Томск: Изд. ТПУ, 2002- 10 с.

Рецензент Ким В.Л.

Методические указания рассмотрены и рекомендованы методическим семинаром кафедры Вычислительной техники протокол №\_ от \_\_\_\_\_г.

Зав. Кафедрой ВТ, проф. д.т.н. \_\_\_\_\_ Н.Г. Марков

### Цель работы:

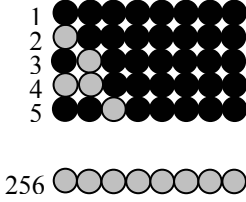
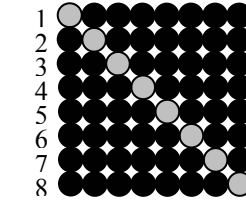
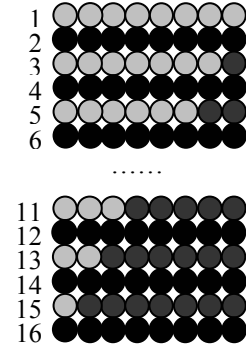
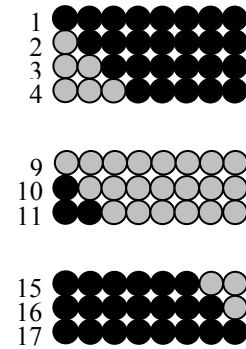
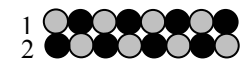


Изучение методов архитектуры и методов проектирования систем на базе микропроцессоров, однокристальных микроЭВМ, встраиваемых контроллеров, систем сбора данных, периферийных блоков вычислительных систем, подсистем ввода-вывода встраиваемых систем.

### Задание на лабораторную работу:

Изучить структуру учебного лабораторного комплекса SDK-1.1 В соответствии с вариантом разработать функциональную спецификацию программного обеспечения, составить алгоритмы работы программы, закодировать и отладить программы на учебном макете SDK-1.1. При реализации вариантов задания необходимо использовать таймеры микроконтроллера AduC812 в режиме прерывания.

### Варианты задания на лабораторную работу:

Варианты заданий	Графическое пояснение к варианту
1. Написать программу, обеспечивающую поочередное «зажигание» и «гашение» светодиодов, расположенных на учебном макете, таким образом, что бы получить визуальный эффект «перемещающегося огня». Интервал между гашением и повторным зажиганием 0.5 сек.	
2. Написать программу, обеспечивающую поочередное «зажигание» и «гашение» светодиодов, расположенных на учебном макете, таким образом, что бы получить визуальный эффект «перемещающегося огня» от концов светодиодной линейки к середине. Частота перемещения 3Гц.	
3. Написать программу, обеспечивающую поочередное «зажигание» и «гашение» светодиодов, расположенных на учебном макете, таким образом, что бы получить визуальный эффект «перемещающегося огня» от одного края светодиодной линейки до другого и обратно. Частота «перемещения» 10 Гц.	
4. Написать программу, обеспечивающую поочередное «зажигание» и «гашение» светодиодов, расположенных на учебном макете, таким образом, что бы получить визуальный эффект «перемещающегося огня» от одного края светодиодной линейки до середины и обратно. Вторая половина линейки должны быть инверсией первой. Частота «перемещения» 1 Гц.	
5. Написать программу, обеспечивающую поочередное «гашение» светодиодов, расположенных на учебном макете, от середины к краям. Вторая половина линейки должны быть инверсией первой. Интервал «гашения» 1сек.	

<p>6 Написать программу, выводящую на светодиоды, расположенных на учебном макете, значение 8-ми разрядного двоичного счетчика, инкрементирующегося с частотой 10Гц.</p>	
<p>7. Написать программу, обеспечивающую поочередное «зажигание» и «гашение» светодиодов, расположенных на учебном макете, таким образом, что бы получить визуальный эффект «перемещающегося огня». Количество проходов – 10. Время каждого последующего прохода должно быть короче предыдущего на 0.1 сек.</p>	
<p>8. Написать программу, обеспечивающую поочередное «зажигание» и «гашение» светодиодов, следующим образом.  1. Зажечь n=8 светодиодов  2. Потушить светодиоды  3. Зажечь n=n-1 светодиодов  4. Потушить светодиоды  5. Повторять пункты 3 и 4 до тех пор пока n&gt;1  Частота переключения 10Гц</p>	
<p>9. Написать программу, обеспечивающую поочередное «зажигание» и «гашение» светодиодов справа на лево и их «гашение» в том же порядке, с частотой 1Гц.</p>	
<p>10. Написать программу, обеспечивающую поочередное «зажигание» светодиодов через один и их «гашение» в том же порядке, с частотой 1Гц.</p>	
<p>Примечание:   - светодиод горит   - светодиод потушен</p>	

## Введение

Учебный лабораторный комплекс SDK-1.1 предназначен для освоения студентами архитектуры и методов проектирования:

- систем на базе микропроцессоров и однокристальных микроЭВМ;
- встраиваемых контроллеров и систем сбора данных;
- периферийных блоков вычислительных систем;
- подсистем ввода-вывода встраиваемых систем.

С использованием стенда SDK-1.1 для студентов высших учебных заведений могут проводиться лабораторные работы по курсам:

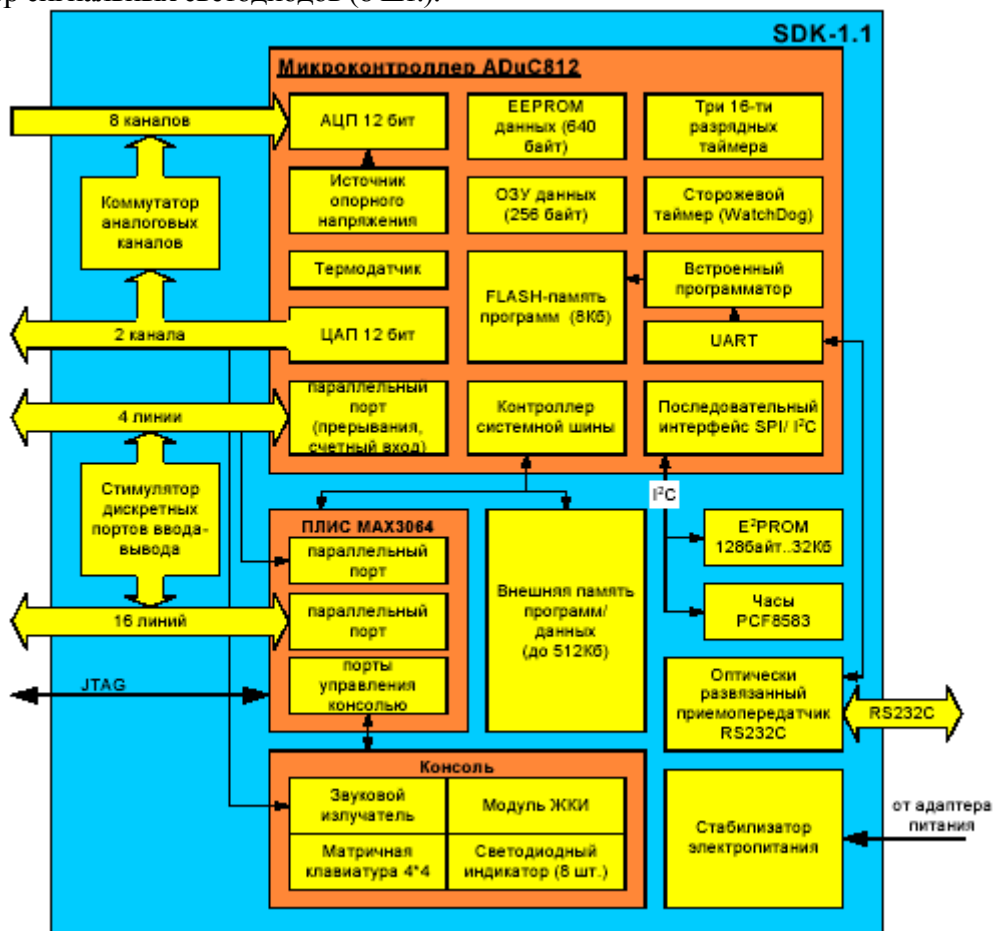
- Организация ЭВМ и вычислительных систем;
- Прикладная теория цифровых автоматов;
- Системы ввода-вывода;
- Информационно-управляющие системы;
- Распределенные управляющие системы;
- Операционные системы реального времени.

## Архитектура стенда SDK-1.1

### Структура аппаратной части

В состав учебного стенда SDK 1.1 входят:

- микроконтроллер ADuC812BS;
- внешняя EEPROM объемом 256 байт;
- клавиатура AK1604A-WWB фирмы ACCORD;
- жидкокристаллический индикатор ЖКИ WH1602B-YGK-CP фирмы Winstar Display;
- часы реального времени PCF8583;
- 128К внешней SRAM с возможностью расширения до 512К;
- набор сигнальных светодиодов (8 шт.).



## Микроконтроллер ADuC812BS

Процессор ADuC812 является клоном Intel 8051 со встроенной периферией.

Основные характеристики:

- рабочая частота 11.0592МГц;
- 8- канальный 12-битный АЦП со скоростью выборок 200К/С (в режиме ПДП);
- два 12-битных ЦАП (код-напряжение);
- внутренний температурный сенсор;
- 640 байт программируемого FLASH/ЕЕ со страничной организацией (256 страниц по 4 байта);
- 256 байт внутренней памяти данных;
- 16Мб адресное пространство;
- режим управления питанием;
- асинхронный последовательный ввод-вывод;
- I2C интерфейс;
- три 16-битных таймера/счетчика и таймер WatchDog.

## Внешняя EEPROM

EEPROM является перепрограммируемым электрически стираемым постоянным запоминающим устройством. Объем памяти EEPROM, установленной в стенде SDK 1.1, составляет 128 байт (возможна установка EEPROM большего объема, до 32Кб). Микросхема EEPROM взаимодействует с процессором посредством интерфейса I2C.

Адрес I2C							
1	0	1	0	0	0	1	R/W

Основные характеристики:

- возможность перезаписи до 1 млн. раз;
- возможность побайтной и постраничной записи (в текущей конфигурации размер страницы 8 байт).

## Матричная клавиатура АК1604А-WWW

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).

## ЖКИ WH1602В-YGK-CP

ЖКИ работает в текстовом режиме (2 строки по 16 символов), имеет подсветку (цвет: желто-зеленый). Основные характеристики:

- 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 (0.1ф). 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 I<sup>2</sup>C.

Адрес I2C							
1	0	1	0	0	0	0	R/W

## Распределение памяти в SDK-1.1

Память в SDK1.1 распределяется следующим образом:

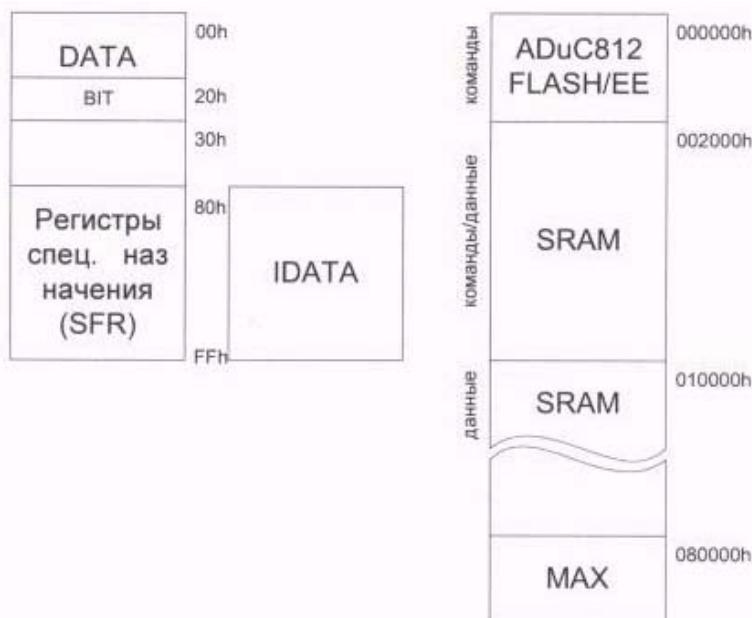


Рис. 2. Карта памяти SDK-1.1

Стандартная для архитектуры 8051 структура внутренней памяти представлена четырьмя банками по 8 регистров общего назначения (диапазоны адресов 00h-07h, 08h-0Fh, 10h-17h, 1Fh-20h), битовым сегментом (20h-2Fh), свободным участком 30h-7Fh, областью размещения SFR (регистров специального назначения, 80h-FFh, доступна при прямой адресации) и свободной областью 80h-FFh, доступной при косвенной адресации.

Внешняя память SDK-1.1 разбита на следующие области.

**ADuC812 Flash/EE.** Это область, в которой располагается таблица векторов прерываний (см. раздел «Система прерываний») и резидентный загрузчик файлов в формате HEX в память SRAM (см. раздел «Резидентный загрузчик HEX202»).

**SRAM.** Статическая память SRAM в SDK-1.1 имеет страничную организацию (максимум 8 страниц по 64К) и условно разделяется на две области. Первая занимает младшие 64Кбайт (страница 0) и доступна для выборки команд микроконтроллером ADuC812. Таким образом, программы могут располагаться только в этих младших 64К адресного пространства. Остальные страницы доступны только для размещения данных. Для адресации ячейки памяти определенной страницы необходимо записать номер страницы в регистр специального назначения DPP ADuC812 (адрес 84h, см. руководство по ADuC812).

**MAX** В младших адресах 8-й страницы адресного пространства (080000h-080007h) располагается 8 ячеек-регистров ПЛИС MAX8064 (MAX8128). Эта область предназначена для взаимодействия с периферийными устройствами стенда (см. раздел «Карта портов ввода-вывода»).

За вычетом 8К Flash-памяти ADuC812, которая отображается в самые младшие адреса (0000h-1FFFh). Фактически для размещения программ доступно 56К статической памяти.

## Карта портов ввода-вывода

В стенде SDK1.1 ввод-вывод данных осуществляется с помощью портов микроконтроллера и микросхемы ПЛИС, которая имеет 8 регистров, отображаемых во внешнее адресное пространство процессора.

**Таблица 1. 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, активный 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

**Регистры ПЛИС****Таблица 2 Перечень регистров ПЛИС**

Адрес	Регистр	Доступ	Назначение
080000H	KB	R/W	Регистр клавиатуры
080001	DATA IND	R/W	Регистр шины данных ЖКИ
080002	EXT LO	R/W	Регистр данных параллельного порта (разряды 0..7)
080003	EXT HI	R/W	Регистр данных параллельного порта (разряды 8..15)
080004	ENA	W	Регистр управления портами ввода-вывода, звуком и сигналом INT0
080006	C IND	W	Регистр управления ЖКИ
080007	SV	W	Регистр управления светодиодами

**Регистр клавиатуры KB**

Адрес 080000H. Значение после сброса 00000000B.

7	6	5	4	3	2	1	0
R	R	R	R	W	W	W	W
ROW				COL			

Биты	Поле	Описание
0..3	COL	Поле предназначено для сканирования клавиатуры (колонки матрицы). Сканирование производится посредством записи логического «0» в один из разрядов поля.
4..7	ROW	Поле предназначено для считывания данных с клавиатурной матрицы (строки). Если ни одна из кнопок в строке не нажата все биты поля ROW содержат логические «1». Если кнопка нажата и на ее колонку подан логический «0», то в поле ROW также появится логический «0».



## Таблица соответствия пересечений строк и столбцов клавиатуры и значения клавиш.

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

### Регистр шины данных ЖКИ DATA\_IND

Адрес 080001H. Значение после сброса 00000000B.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
D7	D6	D5	D4	D3	D2	D1	D0

Биты	Поле	Описание
0..7	D0..D7	Регистр DATA_IND позволяет устанавливать данные на шине данных ЖКИ и считывать их оттуда. Для организации взаимодействия с ЖКИ (для формирования временных диаграмм чтения и записи) необходимо использование регистра C_IND.

### Регистр данных параллельного порта EXT\_LO

Адрес 080002H. Значение после сброса 00000000B.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
D7	D6	D5	D4	D3	D2	D1	D0

Биты	Поле	Описание
0..7	D0..D7	Регистр EXT_LO позволяет считывать и записывать биты 0..7 параллельного порта. Для того, чтобы данные из регистра попали на выход необходимо установить бит EN_LO в логическую «1» (см. регистр ENA). Для чтения данных необходимо установить бит EN_LO в логический «0».

### Регистр данных параллельного порта EXT\_HI

Адрес 080003H. Значение после сброса 00000000B.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
D7	D6	D5	D4	D3	D2	D1	D0

Биты	Поле	Описание
0..7	D0..D7	Регистр EXT_HI позволяет считывать и записывать биты 8..15 параллельного порта. Для того, чтобы данные из регистра попали на выход необходимо установить бит EN_HI в логическую «1» (см. регистр ENA). Для чтения данных необходимо установить бит EN_HI в логический «0».

## Регистр управления ENA

Адрес 080004H. Значение после сброса XX000000B.

7	6	5	4	3	2	1	0
-	-	W	W	W	W	W	W
-	-	INT0	SND2	SND1	SND0	EN_HI	EN_LO

Биты	Поле	Описание
0	EN_LO	Бит EN_LO нужен для управления младшими 8-ю разрядами (биты 0..7) 16-ти разрядного порта ввода-вывода. Если записать в EN_LO логический «0», то порт ввода-вывода переводится в Z состояние и появляется возможность чтения данных из EXT_LO. При записи в данный бит логической «1» порт EXT_LO переключается на вывод и данные записанные в регистр EXT_LO попадают на выход порта ввода-вывода.
1	EN_HI	Полностью аналогичен EN_LO. Управляет старшей частью 16-ти разрядного порта ввода-вывода {биты 8..15}).
2..4	SND0 – SND2	Выход звукового ЦАП. Задаёт уровень напряжения на динамике. Позволяет формировать звуковые сигналы различной тональности и громкости.
5	INT0	При записи логического «0» в этот бит, на вход INT0 ADuC812 также попадает логический «0». Бит можно использовать для формирования внешнего прерывания для микроконтроллера.

## Регистр управления ЖКИ C\_ND

Адрес 080006H. Значение после сброса XXXXX000B.

7	6	5	4	3	2	1	0
-	-	-	-	-	W	W	W
-	-	-	-	-	RS	RW	E

Биты	Поле	Описание
0	E	Бит управления входом «Е» ЖКИ. Наличие положительного импульса на входе «Е» позволяет зафиксировать данные на шине ЖКИ (данные, сигналы RW и RS к этому моменту должны быть уже установлены).
1	RW	Бит переключения шины данных ЖКИ на чтение или запись. 0 - запись, 1 - чтение.
2	RS	Бит переключения режимов команды/данные ЖКИ. 1-данные, 0 – команды.

## Регистр управления светодиодами SV

Адрес 080007H. Значение после сброса 00000000B.

7	6	5	4	3	2	1	0
W	W	W	W	W	W	W	W
D7	D6	D5	D4	D3	D2	D1	D0

Биты	Поле	Описание
0..7	D0..D7	Биты управления светодиодами. Подача логической «1» зажигает светодиоды.

### Доступ к регистрам ПЛИС

Для доступа к регистрам ПЛИС нужно переключить страничный регистр DPP на 8-ую страницу памяти. Адреса регистров внутри страницы находятся в диапазоне от 0 до 7. Доступ к регистрам возможен через указатель: `unsigned char xdata 'regnum'`

Ниже приведен пример функций для доступа к регистрам ПЛИС (вывод на светодиодную сборку непосредственных данных 55h).

```
CSEG                                     ;описание сегмента программы

      ORG      2000H
      LJMP    START                       ;переход на начало программы

DPP   EQU     84h                         ;DPP ставится в соответствие число 84h
      ORG     2100H                       ;Адрес начала основной программы

START:

      MOV     DPP, #08                    ;выбор страницы адресов ПЛИС
      MOV     DPTR, #7h                   ;установка адреса светодиодной сборки
      MOV     A, #55h                     ;подготавливаем данные для
                                           ;высвечивания
      MOVX    @DPTR, A                    ;загрузка данных в регистр ПЛИС
                                           ;содерж. А по адресу в DPTR
      LJMP    START
```

### Проблемы, часто возникающие при доступе к регистрам ПЛИС

Необходимо помнить, что при переключении страниц становятся недоступными все данные размещенные в странице 0.

Во-первых, для того, чтобы избежать проблем со страничным регистром DPP используйте специальные функции для доступа к ПЛИС. Эти функции будут запоминать старое значение страничного регистра, работать с регистрами ПЛИС и возвращать обратно старое значение DPP.

Во-вторых, следите, чтобы передаваемые в регистры ПЛИС значения хранились во внутренней памяти микроконтроллера (DATA, IDATA). Убедиться, что передаваемая информация не содержится во внешней памяти контроллера (XBATA) достаточно просто. Для доступа к внешней памяти в микроконтроллерах семейства C51 используется регистр DPTR. Загляните в листинг программы и убедитесь в том, что для доступа к переменным компилятор не использует DPTR.

## Система прерываний

Микроконтроллер ADuC812 обеспечивает восемь источников и два уровня приоритета прерываний (см. табл. 1). Соответствующий определенному прерыванию приоритет можно установить в регистре специального назначения IP (адрес B8h, см. руководство по ADuC812).

Таблица 3. Прерывания ADuC812

Прерывание	Наименование	Адрес вектора	Приоритет
PSMI	Источник питания ADuCS12	43H	1
IEO	Внешнее прерывание INTO	03H	2
ADCI	Конец преобразования АЦП	33H	3
TFO	Переполнение таймера 0	0BH	4
IE1	Внешнее прерывание INT1	13H	5
TF1	Переполнение таймера 1	1BH	6
I2C/I2SPI	Прерывание последовательного интерфейса (I2C, SPI)	3BH	7
RI/TI	Прерывание UART	23H	8
TF2/EXF2	Переполнение таймера 2	2BH	9

Прерывания ADuC812 имеют вектора в диапазоне 0003h-0043h, которые попадают в область младших адресов памяти программ. Это пространство соответствует 8Кб (0000h-2000h) флэш-памяти. Следовательно, пользователь, не имеющий возможности записи во флэш-память, не может подставить свои процедуры обработки прерываний (точнее команды перехода к процедурам) по адресам, соответствующим векторам прерываний.

Проблема использования прерываний в пользовательских программах решается следующим образом:

1. По адресам (0003h-0043h) векторов прерываний во флэш-памяти SDK-1.1 располагаются команды переходов на вектора пользовательской таблицы, которая располагается в адресах 2003h-2043h.
2. По адресам векторов пользовательской таблицы пользователем указываются команды переходов на процедуры обработки прерываний.

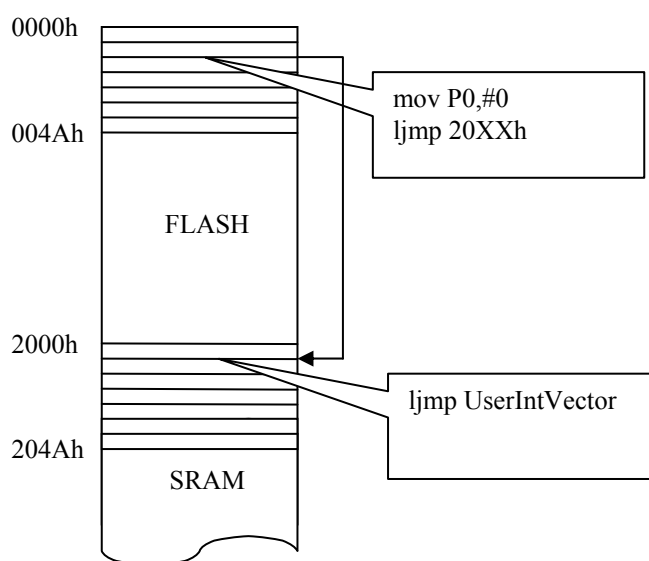


Рис. 3 Использование прерываний в SDK-1.1

Приведем пример помещения собственного вектора в пользовательскую таблицу. Пусть требуется осуществить обработку прерываний от таймера 0 (прерывание OVh). В программу можно вставить следующий код:

В таблице векторов, находящейся во FLASH, перед переходом в пользовательскую таблицу в порт P0 записывается значение 0. Это связано с аппаратной ошибкой в процессоре ADuC812, которая заключается в некорректной выпорке команд при передаче управления из младших 8К памяти команд в старшие адреса. Ошибка устраняется путем обнуления регистра защелки порта 0 (P0) непосредственно перед передачей управления. Рекомендуем обратиться к документу **ADuC812 Errata Sheet** на редакцию чипа со штампом даты больше 9933.

---

```
ORG 200bh          ;описание вектора прерывания
    LJMP t0_0      ;от таймера 0
CSEG              ; сегмент кода программ
    DPP EQU 84h
    ORG 2100H
start:
                ;Далее идет текст основной программы

    LJMP $        ;команда перехода по тому же
                ;адресу ("программная ловушка")

t0_0:
                ;далее идет текст подпрограммы обработки
                ;прерывания от таймера/счетчика 0

    RETI         ;команда выхода из подпрограммы
                ;обработки прерывания

END
```

## Программирование и отладка

Для программирования стенда может использоваться любой транслятор ассемблера или C для ядра 8051, например, пакет uVision (Keil Software). До начала программирования на языке C рекомендуется внимательно ознакомиться с документацией по используемому компилятору (т.к. компиляторы для микроконтроллеров имеют нестандартные расширения).

Основными этапами в программировании стенда являются:

- подготовка программы в текстовом редакторе (или среде программирования);
- транслирование исходного текста и получение загрузочного hex-модуля программы;
- подготовка и загрузка hex-модуля в стенд через интерфейс RS232C с помощью поставляемых инструментальных систем. Под подготовкой понимается добавление в конец модуля строки со стартовым адресом программы, т.е. адреса, по которому передается управление после загрузки в стенд;
- прием и обработка hex-модуля резидентным загрузчиком HEX202, передача управления загруженной программе.



**Рис. 4 Этапы программирования стенда SDK-1.1**

В соответствующих разделах главы «Программное обеспечение стенда SDK-1.1» и в главе «Инструментальные средства фирмы Keil Software» дано описание основных инструментальных средств, участвующих в каждом этапе программирования стенда.

## Программное обеспечение стенда SDK-1.1

### Резидентный загрузчик HEX202

Резидентный загрузчик HEX202 располагается во Flash-памяти ADuC812, начиная с адреса 0100h. Он обеспечивает начальную инициализацию системы, загрузку программ в hex-формате в память SDK-1.1 и передачу им управления.

**Начальная инициализация.** При включении питания (или при передаче управления на ячейку с адресом 0) происходит повторная инициализация **всех** регистров специального назначения их значениями по умолчанию. Это сделано для того, чтобы при случайной передаче управления на ячейку с адресом 0 вследствие возможной ошибки в пользовательской программе не происходило сбоя системы, а сама система вела себя как при включении питания. Эта же процедура повторяется непосредственно перед передачей управления загруженной программе. В случае успешной инициализации на ЖКИ на мгновение выводится надпись «SDK-1.1, 2001 ©LMT Ltd» и на резонатор выдается короткий сигнал.

**Загрузка программ в память SDK-1.1.** После процедуры инициализации системы последовательный канал настраивается в режим 9600 бит/сек, 8 бит данных, 1 стоп-бит, без контроля по четности и в него выдается строчка «HEX202-XX», где XX — номер версии загрузчика. Далее с интервалом примерно в 200мс выдается символ V и ожидается появление символа со стороны инструментальной системы на PC. При появлении символа, если это первый символ строки в hex-формате, то есть двоеточие (:'), выдача символа V прекращается и производится прием остальной части hex-строки. По завершении приема очередной hex-строки вычисляется ее контрольная сумма. Если она не совпадает с принятой, то в последовательный канал выдается символ '-', сигнализирующий об ошибке приема. В противном случае выдается '+' и принятая строка обрабатывается в соответствии с указанной в ней командой (запись данных в память, конец блока или передача управления). Далее, если не было команды передачи управления, вывод в последовательный канал символа '.' возобновляется и ожидается следующая hex-строка.

**Передача управления загруженной программе.** Передача управления происходит по приему hex-строки вида :02AAAA060000SS<cr>, где AAAA есть hex-адрес, по которому необходимо передать управление, SS есть контрольная сумма hex-строки, <cr> - символ возврата каретки. Такая строчка должна быть добавлена в конец каждого hex-файла, загружаемого в SDK-1.1. Для этого в поставляемых с SDK-1.1 инструментальных системах есть команда `addhexstart` (см. соответствующие разделы).

## Инструментальная среда для Win9x/NT – t2.exe

Инструментальная среда T2.exe призвана решать следующие задачи:

1. Преобразование HEX и BIN файлов
2. Передача загрузочных модулей различных форматов в целевую систему с протоколами различного уровня сложности
3. Получение информации из целевой системы
4. Обеспечение элементарных операций с последовательным каналом (прием и передача байта, эмуляция терминала, настройка скорости)
5. Обеспечение быстрой адаптации к целевой системе

Команда	Действие
OPENCHANNEL	Открытие COM порта с установкой сигнала RTS
OPENCOM1,OPENCOM2	Открытие портов COM1 или COM2 с установкой сигнала RTS
CLOSECHANNEL	Закрытие COM порта
TERM	Включение эмулятора терминала
+ECHO	Включение режима копирования консольного вывода в файл echo.txt
-ECHO	Выключение режима копирования консольного вывода в файл echo.txt
LOADHEX	HEX загрузка
ADDHEXSTART	Добавление стартового адреса в конец HEX-файла
BYE	Выход из T2
LFILE	Интерпретация командного файла

**OPENCHANNEL (baud -> com)** - Открытие последовательного порта на заданной скорости. Числовой параметр baud определяет скорость в бодах, например 19200. Параметр com может иметь два значения «com1» или «com2».

Пример: 9600 openchannel com1

**OPENCOM1,OPENCOM2** – открытие COM1 или COM2 на скорости 9600 бод

Пример: opencom1

**TERM (w ->)** Включение эмулятора терминала. w=0 – бинарный, w=1 – HEX.

Пример: 0 term

**LOADHEX (-> filename.hex)** Загрузка HEX- файла в целевую систему по протоколу HEX202. Этот протокол предполагает последовательную пересылку строк из HEX- файла filename.hex. После посылки очередной строки ожидается подтверждение со стороны HEX202 в виде символа «+» или запрос на повторную посылку в виде «-». Необходимо заметить, что перед посылкой HEX- файла сгенерированного в какой либо среде разработки, необходимо добавить в его конец стартовый адрес командой addhexstart.

Пример: loadhex myfile.hex

**ADDHEXSTART (addr,seg->) filename.hex** - Добавление в конец файла filename.hex строки, которая нужна для передачи управления загрузчиком HEX202 по адресу addr после загрузки в целевую систему. Поле seg необходимо указывать на в настоящее время оно не используется

Пример: 0x200 0x0 addhexstart myfile.hex

**BYE** - выход из программы T2.exe

**LFILE – filename.ext** Интерпретация командного файла filename.ext. Файл представляет набор строк текста, содержащих команды T2.exe в том же виде, в котором они представлены в командной строке T2.