МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ,
МОЛОДЕЖИ И СПОРТА УКРАИНЫ
ГОСУДАРСТВЕННОЕ ВЫСШЕЕ УЧЕБНОЕ ЗАВЕДЕНИЕ
**ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

# Методические указания

к виполнению лабораторных работ по курсу «Микропроцессоры и ЭВМ» для студентов специальности 6.05100304 НАП (Приборы и системы экологического мониторинга).

Рассмотрено
на заседании кафедры
электронной техники.
Протокол №6 от 26.01.2011г
Утверждено на заседании
учебно-издательского совета
ДОННТУ протокол № 2
От 21. 03 2011 г

**Донецк, 2011**

Методические указания к выполнению лабораторных работ по дисциплине «Микропроцессоры и ЭВМ» для студентов специальности 6.051003 НАП (Приборы и системы экологического мониторинга)/

Д.Н. Кузнецов, В.В.Калашников – ДонНТУ, Донецк, 2011. – 63 с.

Приведены цель и методика выполнения лабораторных работ, необходимые теоретические и справочные данные к их выполнению, порядок выполнения, требования к содержанию отчетов и контрольные вопросы для самоконтроля знаний студентов.

Составители: доцент Д.Н. Кузнецов

ст. преп. В.В. Калашников

Рецензент:

**СОДЕРЖАНИЕ**

# ОСНОВНЫЕ ПОЛОЖЕНИЯ

Методические указания предназначены для студентов специальности «Научные аналитические и экологические приборы и системы» и являются руководством к лабораторным работам по дисциплине «Микропроцессоры и ЭВМ».

Целью этого курса является закрепление лекционного материала, освоение интегрированной среды разработки и отладки программного обеспечения ProView, освоение программы имитационного моделирования микропроцессорных устройств Proteus и специализированного лабораторного стенда EV8031/AVR.

Лабораторные работы проводятся в компьютерной аудитории. При их выполнении используются методы математического и имитационного моделирования с использованием пакетов прикладных программ ProView и Proteus, а также реальный лабораторный стенд EV8031/AVR.

Лабораторные работы начинаются с короткого теоретического материала по основам работы исследуемого устройства. Далее излагаются методика выполнения работы, справочные данные, требования к содержанию отчета и перечень контрольных вопросов для самоконтроля усвоения материала.

Подготовка к лабораторной работе должна выполнятся по лекционному материалу и дополнительной литературе.

По каждой выполненной работе составляется отчет. На титульном листе указывается дисциплина, тема лабораторной работы и автор отчета. Отчет должен содержать цель работы, вариант индивидуального задания, принципиальную схему исследуемого устройства, результаты исследований. В конце следует анализ полученных результатов и вывод по работе.

## Лабораторная работа №1

### Архитектура микроконтроллера 8051(52). Организация памяти. Операции пересылки кодов между различными видами памяти.

**Цель работы:** изучение структуры микроконтроллера 8051(52), организации памяти, а также приобретение начальных навыков программирования на языке СИ операций пересылки кодов.

### Теоретическая часть

**Архитектура МК.**

Микроконтроллер (МК) 8051(52) относится к классу однокристальных микроЭВМ и предназначен для построения несложных цифровых систем управления. Структура МК 8051 приведена на рис.1.

Этот МК можно считать классическим образцом, по образу и подобию которого позднее было создано множество других изделий.

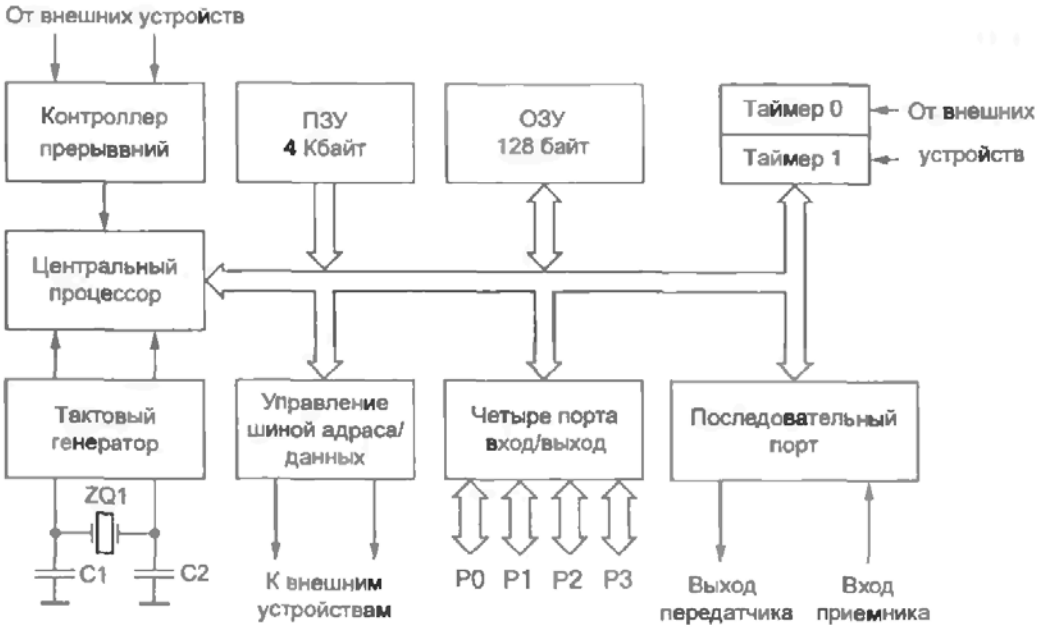



Рис.1 Структурная схема МК 8051

Микроконтроллеры 8051(52) имеют следующие аппаратные особенности:

– встроенную память программ объемом 4(8) кбайт;

– внутреннее ОЗУ объемом 128(256) байт;

– четыре двунаправленных побитно настраиваемых восьмиразрядных порта ввода-вывода;

– два(три) 16-разрядных таймера-счетчика;

– встроенный тактовый генератор;

– адресация 64 КБайт памяти программ и 64 Кбайт памяти данных;

– две линии запросов на прерывание от внешних устройств;

– интерфейс для последовательного обмена информацией с другими микроконтроллерами или персональными компьютерами.

**Организация памяти МК.**

В архитектуре семейства 8051 память программ и память данных разделены (гарвардская архитектура). Организация памяти иллюстрируется рис.2. Память программ может быть целиком внешней (сигнал EA=0), либо при обращении по младшим 4(8) К адресов код извлекается из ячеек внутренней памяти (сигнал EA=1).

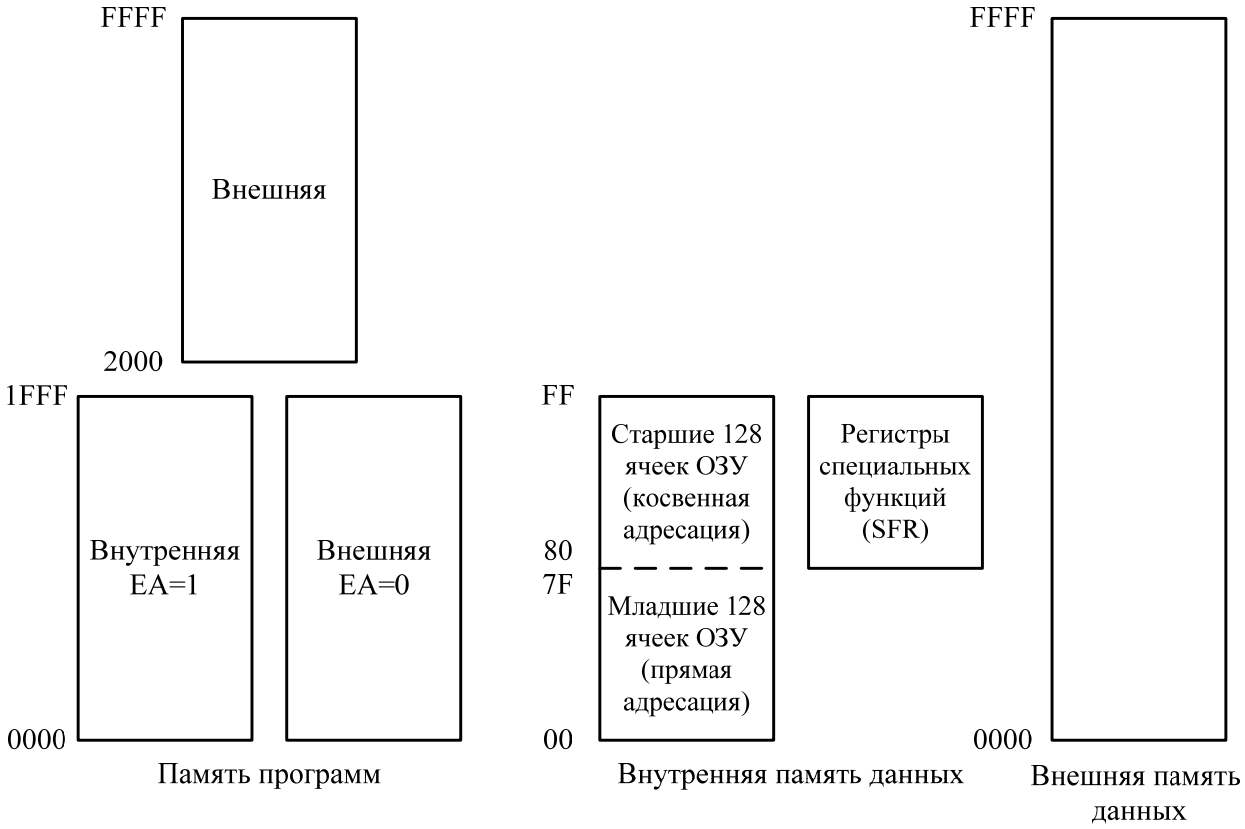



Рис.2 Распределение памяти МК 8052

Память данных делится на внешнюю и внутреннюю, каждая из них имеет свое пространство адресов.

Увеличение размеров внутренней памяти данных в МК 8052 привело к наложению старших 128 байт ОЗУ данных и пространства регистров специальных функций. Выбор той или иной области при обращении осуществляется аппаратурой МК.

Область регистров специальных функций (SFR) МК 8052 содержит 26 регистров, назначение которых приведено в табл.1. Как видно из таблицы, 12 SFR допускают побитовое обращение. Вновь добавленные в МК 8052 (в сравнении с 8051) регистры выделены жирным шрифтом.

При написании программы в пакете ProView для указания области памяти используются следующие ключевые слова:

– CODE – память программ (кодов);
– DATA – внутренняя память данных (прямая адресация);
– IDATA – внутренняя память данных (косвенная адресация);
– XDATA – внешняя память данных;
– BIT – битовый сегмент внутренней памяти данных.

Табл. 1

***Регистры специальных функций МК 8052***

| Наименование | Назначение | Адрес |
|---|---|---|
| P0* | Порт 0 | 80H |
| SP | Указатель стека | 81H |
| DPL | Младший байт указателя данных DPTR | 82H |
| DPH | Старший байт указателя данных DPTR | 83H |
| PCON | Регистр управления потреблением | 87H |
| TCON* | Регистр управления таймеров/счетчиков | 88H |
| TMOD | Регистр режимов таймеров/счетчиков | 89H |
| TL0 | Таймер/счетчик 0. Младший байт | 8AH |
| TL1 | Таймер/счетчик 1. Младший байт | 8BH |
| TH0 | Таймер/счетчик 0. Старший байт | 8CH |
| TH1 | Таймер/счетчик 1. Старший байт | 8DH |
| P1* | Порт 1 | 90H |
| SCON* | Регистр управления последовательным портом | 98H |
| SBUF | Буфер последовательного порта | 99H |
| P2* | Порт 2 | 0A0H |
| IE* | Регистр разрешения прерываний | 0A8H |
| P3* | Порт 3 | 0B0H |
| IP* | Регистр приоритетов прерываний | 0B8H |
| **T2CON*** | **Регистр управления таймером/счетчиком 2** | **0C8H** |
| **RCAP2L** | **Младший регистр данных перезагрузки T2** | **0CAH** |
| **RCAP2H** | **Старший регистр данных перезагрузки T2** | **0CBH** |
| **TL2** | **Таймер/счетчик 2. Младший байт** | **0CCH** |
| **TH2** | **Таймер/счетчик 2. Старший байт** | **0CDH** |
| PSW* | Регистр состояния программы | 0D0H |
| A* | Аккумулятор | 0E0H |
| B* | Регистр B | 0F0H |

**Основные положения из языка СИ51 необходимые для выполнения задания лабораторной работы.**

**Спецификаторы типов**

Язык Си51 поддерживает определения для множества базовых типов данных, называемых "основными" типами. Названия этих типов перечислены в Табл.2.

Табл. 2.

***Спецификаторы типов языка Си***

| Типы целых | Типы плавающих | Другие типы |
|---|---|---|
| signed char<br>signed int<br>signed short int<br>signed long int<br>unsigned char<br>unsigned int<br>unsignet short int<br>unsigned long int | Float<br>double | Void |

Типы signed char, signed int, signed short и signed long int вместе с соответствующими двойниками unsigned называются типами целых.

Спецификаторы типов float и double относятся к типу "плавающих". В объявлениях переменных и функций можно использовать любые спецификаторы "целый" и"плавающий".

Тип void может быть использован только для объявления функций, которые не возвращают значения.

Можно задать дополнительные спецификаторы типа путем объявления typedef.

При записи спецификаторов типов допустимы сокращения как показано в табл.3. В целых типах ключевое слово signed может быть опущено. Так, если ключевое слово unsigned опускается в записи спецификатора типа, то тип целого будет знаковым, даже если опущено ключевое слово signed.

В табл.4 для каждого типа приведены: размер распределяемой памяти и области значений переменных для данного типа. Поскольку тип void не представляет переменных, он не включен в эту таблицу.

Табл.3

***Спецификаторы и сокращения***

| Спецификатор типа | Сокращение |
|---|---|
| signed char | char |
| signed int | signed, int |
| signed short int | short, signed short |
| signed long int | long, signed long |
| unsigned char | - |
| unsigned int | unsigned |
| unsigned short int | unsignet short |
| unsignet long int | unsignet long |
| float | - |
| long float | double |

Табл.4

***Размер распределяемой памяти и области значений переменных различных типов***

| Тип | Представление в памяти | Область значений |
|---|---|---|
| Char | 1 байт | -128 до 127 |
| int | зависит от реализации | |
| short | 2 байта | -32768 до 32767 |
| long | 4 байта | -2.147.483.648 до 2.147.483.647 |
| unsigned char | 1 байт | 0 до 255 |
| unsigned | зависит от реализации | |
| unsigned short | 2 байта | 0 до 65535 |
| unsigned long | 4 байта | 0 до 4.294.967.295 |
| float | 4 байта | IEEE стандартное соглашение |
| double | 8 байт | IEEE стандартное соглашение |

Объявление переменных

Перед использованием переменной в Си её необходимо объявить, т.е. указать компилятору какой тип данных она может хранить и как она называется.

Формат объявления переменной таков:

[at constant] type [memory_space] variable_name [= initialization_value] ;

[at constant] – необязательный элемент, он нужен для указания адреса переменной;

type – тип переменной;

[memory_space] – необязательный элемент, он указывает в какой области памяти будет расположена переменная. Варианты: BIT, CODE, DATA, IDATA, XDATA.

variable_name – имя переменной;

[= initialization_value] – необязательный элемент – начальное значение переменной.

Локальные переменные объявляются в самом начале функции, т.е. сразу после скобки {. Локальные переменные доступны только в той функции где они объявлены.

Вот несколько примеров объявления переменных:

unsigned char my_peremen = 34;

unsigned short big_peremen = 34034;

Это объявлены две переменные и им присвоены значения.

Первая my_peremen – символьного типа – это 1 байт, она может хранить число от 0 до 255. В данном случае в ней хранится число 34.

Вторая big_peremen – целого типа, два байта, в ней может хранится число от 0 до 65535 , а в примере в неё поместили десятичное число 34034.

Пример массива содержащего 3 числа или элемента массива.

char mas[3]={11,22,33};

Нумерация элементов начинается с 0! Т.е. элементы данного массива называются

mas[0], mas[1], mas[2]

и в них хранятся десятичные числа 11, 22 и 33.

Где то в программе вы можете написать:

mas[1] = 210;

Теперь в mas[1] будет хранится число 210

Массивы могут быть многомерными. Можно не присваивать значений элементам массива при объявлении. Но только при объявлении вы можете присвоить значения всем элементам массива сразу ! Потом это можно будет сделать только индивидуально для каждого элемента.

## Задание к лабораторной работе

3. ЗАДАНИЕ

Пусть в памяти программ CODE, начиная с ячейки ADR2, расположена таблица кодов формата (unsigned char или unsigned short) длиной N.

Записать на языке СИ МК К1816ВЕ51 программу, которая выполняет пересылку данного массива в RAM (data или xdata), начиная с адреса ADR3. Программа должна начинаться с ячейки ADR1.Результаты пересылки кодов выдавать на статическую индикацию лабораторного стенда EV8031/AVR в десятичной системе счисления и на UART.

Таблица 2.2. Таблица вариантов заданий

| № варианта | Code Adr2 | Data Adr3 | Xdata Adr3 | Длина массива N | Нач адрес прог ADR1 | Unsigned Char | Unsigned short |
|---|---|---|---|---|---|---|---|
| 01 | 0x100 | 0x35 | - | 0a | 0x200 | + | - |
| 02 | 0x150 | - | 0x100 | 0b | 0x250 | - | + |
| 03 | 0x200 | 0x40 | - | 0c | 0x300 | + | - |
| 04 | 0x250 | - | 0x150 | 0d | 0x350 | - | + |
| 05 | 0x300 | 0x45 | - | 0e | 0x400 | + | - |
| 06 | 0x350 | - | 0x200 | 0f | 0x450 | - | + |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 07 | 0x400 | 0x50 | - | 09 | 0x500 | + | - |
| 08 | 0x450 | - | 0x250 | 0a | 0x550 | - | + |
| 09 | 0x500 | 0x55 | - | 0b | 0x600 | + | - |
| 10 | 0x550 | - | 0x300 | 0c | 0x650 | - | + |
| 11 | 0x600 | 0x35 | - | 0d | Ox700 | + | - |
| 12 | 0x650 | - | 0x350 | 0e | 0x750 | - | + |
| 13 | 0x700 | 0x40 | - | 0f | 0x800 | + | - |
| 14 | 0x750 | - | 0x400 | 09 | 0x850 | - | + |
| 15 | 0x800 | 0x45 | - | 0a | 0x900 | + | - |

Коды данных (unsigned char или unsigned short) задать произвольно.

4. ПРИМЕР ВЫПОЛНЕНИЯ ЗАДАНИЯ

В памяти команд code с адреса ADR2=40h расположено N=5 шестнадцатеричных кодов. Необходимо переписать их в оперативную память данных DATA, начиная с адреса ADR3=0x40. Программа должна начинаться с адреса ADR1=0x0100. Результаты пересылки кодов выдавать на статическую индикацию лабораторного стенда EV8031/AVR в десятичной системе счисления и на UART

Пример программы пересылки кодов приведен ниже.

```
/*программа пересылки кодов*/

/*программа пересылки кодов*/
#include<reg51.h>
#include<stdio.h>
#define adr1 0x100
#define adr2 0x40
#define adr3 0x40
#define n 5 количество пересылаемых элементов n=5
/* массив пересылаемых кодов str1 находится в программной памяти code. Начальный адрес массива adr2 = 0x40. Елементы массива имеют формат unsigned char (1 байт без знака */
at adr2 unsigned char code str1[]={0x10,0x7f,0x30,0x40,0xfe};
/* Резервируем массив oddnums в оперативной памяти data, состоящий из 5 байт. Начальный адрес массива adr3 */
at adr3 unsigned char data oddnums[5];
/*Зарезервируем массив s[4] под строку символов в оперативной памяти. В эту строку оператором sprintf будут записаны коды елементов массива oddnums переведенные в десятичную систему счисления в коде ASC . Коды нужны для преобразования и выдачи их на цифровую индикацию*/
unsigned char s[4];
unsigned char ind1=0,ind2=0,k=0,ind;
#define const1 0x0f
#define const2 0x7fff
int c,j;
/* Начало главной функции программы*/
at adr1 void main() //Главная функция программы начинается с адреса adr1
{
for (c=0;c<=n-1;c++)
{
oddnums[c]=str1[c]; //пересылаем коды из массива str1 в массив oddnums
/* выполняем преобразование oddnums[c] в десятичную систему счисления и записываем в строку s*/
```

```
sprintf(s,"%d",oddnums[c]);
/*выполняем преобразование строки символов s в ASC в строку цифровых символов для индикации. Количество цифровых символов (7-ми сегментных индикаторов) равно 4*/
ind1=0;
ind2=0;
m1:
if(s[k]!=0) //Признаком окончания записи в строке S является код 00 (‘\0’)
{
ind=(ind1>>4)&0x0f;
ind2|=ind;
ind1=ind1<<4;
ind1=ind1|(s[k]&0x0f);
k++;
goto m1;
}
else
{
k=0;
/*Цифровые индикаторы подключены к портам P2 –два старших разряда, P1-два младших разряда*/
P2=ind2;
P1=ind1;
ind=0;
}
/*Вывод на индикацию в буфер UART – номер цикла вывода C и символов в десятичной системе счисления из oddnums */
printf("c=%d  oddnums=%d  ",c,oddnums[c]);
for(j=const2;j>=0;j--);//временная задержка между пересылаемыми символами
}
while(1);
}
 Программа выполняет пересылку кодов с использованием указателей*/
 #include <reg51.h>
#include<stdio.h>
#define adr1 0x100
#define adr2 0x40
#define adr3 0x40
#define n 0x0a
sbit C=0xd7;
char c;
/* массив пересылаемых кодов str1 находится в программной памяти code. Начальный адрес массива adr2 = 0x40. Елементы массива имеют формат unsigned char  (1 байт без знака */
at adr2 unsigned char code str1[]={10,20,30,40,50,60,70,80,90,200};
at adr3 unsigned char data oddnums[n];
at 0xb000 unsigned char xdata ind2;
at 0xa000 unsigned char xdata ind1;
#define const1 0x7fff
short j;
code unsigned char *stbl;
unsigned char *odn;
unsigned char oddnum;
```

```
// Функция перевода пересылаемого кода oddnums[c] в дес. с.с
void prvind()
{
unsigned char rg0;
unsigned char rg1;
unsigned short R;
unsigned char ra1;
unsigned char ra3;
unsigned char rg2;
R=oddnums[c];
rg0=0;
rg1=0;
rg2=0;
C=0;
R-=0x270f;
if(C==1){ind1=0b10101011;ind2=0b11001101;goto m3;}
R=oddnums[c];
m0:
C=0;
R-=0x03e8;
if(C==1){rg0++;goto m0;}
else R+=0x03e8;rg0=rg0<<4;
m1:
C=0;
R-=0x64;
if(C==1){rg1++;goto m1;}
else ra1=R+0x64; ind1=rg0|rg1;
m2:
C=0;
ra1-=0x0a;
if(C==1){rg2++;goto m2;}
else ra3=ra1+0x0a;rg2=rg2<<4;
ind2=rg2|ra3;
m3:
for(j=const1;j>=0;j--);
}
// Главная функция
at adr1 void main()
{
stbl=&str1[0];
odn=&oddnums[0];
for (c=0;c<=(n-1);c++){
oddnum=*stbl++;
*odn++=oddnum;
printf("  %d",oddnum);
prvind();}
while(1);
}
```

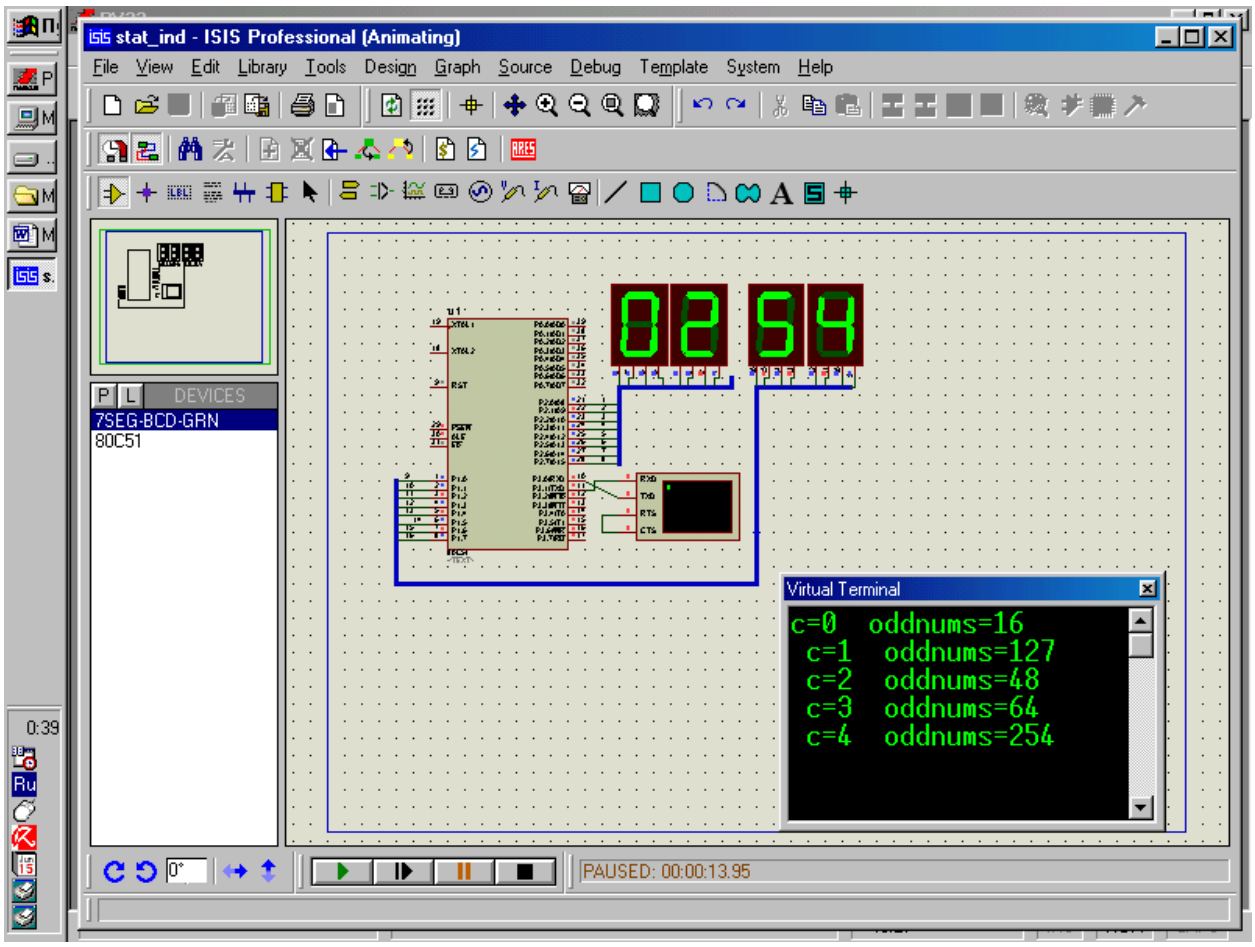



Рис.3. Результат работы программы в PROTEUS.

**Отладка программы в дебагере ProView**

После успешной компиляции программы необходимо выполнить ее отладку во встроенном в ProView дебагере (отладчике). При этом необходимо выполнить следующее:

1) Убедиться, что таблица кодов длиной N расположена в памяти кодов (code) начиная с адреса ADR2. Для этого зайдите в меню View и выберите Code. Откроется окно просмотра памяти кодов (рис.3).
2) Убедиться, что программа расположена в памяти кодов начиная с адреса ADR1.
3) Выполнить пошаговое выполнение программы (по F7).
4) Найти результат пересылки кодов, расположенный по адресу ADR3. Для этого зайдите в меню View, выберите Data dump и далее Data View.

**Содержание отчета**

Отчёт о лабораторной работе должен содержать:

1) Титульный лист.
2) Цель и задачи работы.
3) Исходные данные, в соответствии с вариантом.
4) Контрольный ручной просчет.
5) Текст программы пересылки кодов на языке Си с подробными комментариями.
6) Выводы по работе.

Контрольные вопросы:

1. Приведите ключевые слова в пакете ProView для указания областей памяти микроконтроллера МК51.

2. Перечислите спецификации типа целый и их размеры распределяемой памяти.
3. Приведите формат объявления переменной .
4. Локальные переменные и их объявления.
5. Глобальные переменные и их объявления.
6. Переменные типа массив в различных областях памяти.
7. Объявление указателей и что означают знаки &, *.

## Лабораторная работа №2

### Арифметические и логические операции выполняемые МК51 на языке Си

**Цель работы:** приобретение начальных навыков программирования на языке Си арифметических и логических операций.

### Теоретическая часть

**Операции**

Операции – это специальные комбинации символов, специфицирующие действия по преобразованию различных величин.

В табл.1. представлен список операций языка Си. Операции должны использоваться точно так, как они представлены в таблице: без пробельных символов между символами в тех операциях, которые представлены несколькими символами.

Табл. 1.

***Операции языка Си***

| Операция | Наименование |
|---|---|
| ! | Логическое НЕ |
| ~ | Побитовое дополнение |
| + | Сложение |
| - | Вычитание |
| * | Умножение |
| / | Деление |
| % | Остаток |
| << | Сдвиг влево |
| >> | Сдвиг вправо |
| < | Меньше |
| <= | Меньше или равно |
| > | Больше |
| >= | Больше или равно |
| = = | Равно |
| != | Не равно |
| & | Побитовое И, адрес от |
| \| | Побитовое включающее ИЛИ |
| ^ | Побитовое исключающее ИЛИ |
| && | Логическое И |
| \|\| | Логическое ИЛИ |
| ' | Последовательное выполнение (запятая) |
| ?: | Операция условного выражения |
| ++ | Инкремент |
| -- | Декремент |
| = | Простое присваивание |
| += | Сложение с присваиванием |
| -= | Вычитание с присваиванием |

| | |
|---|---|
| *= | Умножение с присваиванием |
| /= | Деление с присваиванием |
| %= | Остаток с присваиванием |
| >>= | Сдвиг вправо с присваиваиванием |
| <<= | Сдвиг влево с присваиванием |
| &= | Побитовое И с присваиванием |
| \|= | Побитовое включающее ИЛИ с присваиванием |
| ^= | Побитовое исключающее ИЛИ с присваиванием |

**Задание к лабораторной работе**

Пусть в памяти программ, начиная с ячейки ADR2 расположена таблица кодов длиной N (Xi, i=1,2,...,N, формат – unsigned char или unsigned short согласно задания).

Записать на языке Си МК8051 программу, которая выполняет вычисление заданной функции F над этими кодами. Результат вычисления разместить в ячейке ОЗУ с адресом ADR3 = 0x50. Содержимое ячейки памяти 0x50 выдать на индикацию по адресу 0xB000 внешней памяти XDATA лабораторного стенда EV8031/AVR. Программа должна начинаться с ячейки ADR1. Варианты заданий приведены в табл.2.

Табл.2

***Таблица вариантов заданий***

| Номер | ADR1 | ADR2 | Unsigned Char | Unsigned Short | N | Функция F |
|---|---|---|---|---|---|---|
| 1 | 0x714 | 0x431 | + | - | E | (Sum (Xi))*K1) / N |
| 2 | 0x62F | 0x541 | - | + | F | (Max(Xi))*K2 / N |
| 3 | 0x53E | 0x621 | + | - | D | (Min(Xi))*K2 / N |
| 4 | 0x44A | 0x711 | - | + | C | (Sum(Xi) \| X1)*K1 |
| 5 | 0x355 | 0x121 | + | - | B | (Max(Xi) \| X1)*K1 |
| 6 | 0x266 | 0x236 | - | + | A | (Min(Xi) \| X1)*K2 |
| 7 | 0x177 | 0x345 | + | - | 6 | (Sum(Xi) & X1)*K3 |
| 8 | 0x78A | 0x454 | - | + | 7 | (Max(Xi) & X1)*K3 |
| 9 | 0x69D | 0x568 | + | - | F | (Min(Xi) & |
| 10 | 0x5AF | 0x677 | - | + | E | X1)*K1 |
| 11 | 0x4BA | 0x781 | + | - | 5 | (Sum(Xi) \| X2)*K1 |
| 12 | 0x3C9 | 0x231 | - | + | 6 | (Max(Xi) \| X2)*K3 |
| 13 | 0x2D8 | 0x3A1 | + | - | 7 | (Min(Xi) \| X2)*K2 |
| 14 | 0x1E4 | 0x4B2 | - | + | 8 | (Sum(Xi) & X2)*K4 |
| 15 | 0x7F5 | 0x5C2 | + | - | 8 | (Max(Xi) & X2)*K1 |
| | | | | | | (Min(Xi) & X2)*K3 |

Коды данных задать произвольно. Выполнить контрольный просчет.
K1=1,2; K2=1,3; K3=1,4; K4=1,5;

**Пример программы.**

Пусть в памяти программ, начиная с ячейки ADR2 = 0x40, расположена таблица кодов X длиной n=10 Необходимо составить и отладить программу, которая вычисляет функцию F по формуле F= (summa x[I] )*K1/n результат вычисления необходимо вывести на UART оператором printf и на стенд на статическую цифровую индикацию по адресу 0xa000 и 0xb000. Программа должна начинаться с ячейки ADR1= 0x100.

Программа на языке Си.

**/*программа вычисления функции F=(summa(x[i])*k0/n; i=1,2,..10; k=1.2;*/**

**/* При вычислении функции F разрешено использовать целочисленные операции*/**

```
#include<stdio.h>
#include<reg51.h>
#include<math.h>
#define adr1 0x100
#define adr2 0x40
#define adr3 0x40
#define n 0x0a
```

**/* Описание глобальных переменных и констант*/**

```
/* Массив обрабатываемых кодов memr формата unsigned char находится в программной памяти code с начальным адресом adr2 */
at adr2 unsigned char  code memr[]={20,30,20,25,50,35,40,5,7,100};
char i;
at adr3 unsigned short data sum,z;
    /*Зарезервируем массив s[4] под строку символов в оперативной памяти. В эту строку оператором sprintf будут записаны коды результата вычисления Z переведенные в десятичную систему счисления в коде ASC . Коды нужны для преобразования и выдачи их на цифровую индикацию*/
unsigned char s[4];
unsigned char ind1=0,ind2=0,k=0,ind;
#define const1 0x7fff
unsigned short k1,k2;
/*Главная функция начинается с адреса adr1 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////*/
at adr1 void main()
{
sum=0;
for (i=0;i<=(n-1);i++)
sum+=memr[i];//вычисляем сумму элементов массива memr
k1=(sum*12);//сумму умножили на коэф. 12 и резальтат получили в10 раз больше
k2=(n);
z=k1/k2;//
/* полученный результат z переводим оператором sprintf в строку s в десятичной системе счисления в коде ASC*/
sprintf(s,"%d",z);
/*Выполняем преобразование строки десятичных символов s в коды цифровой индикации*/
ind1=0;
ind2=0;
/* Признаком окончания строки символов s является код 00- (‘\0’)*/.
m1:
if(s[k]!=0)
{
ind=(ind1>>4)&0x0f;
ind2|=ind;
ind1=ind1<<4;
```

```
ind1=ind1|(s[k]&0x0f);
k++;
goto m1;
}
else
{
k=0;
```
/* Четыре цифровых семи сегментных индикатора подключены к портам P2 и P1. Два старших разряда подключены к порту P2 – два младших к порту P2.На индикацию выводится результат z в 10 раз больший */
```
P2=ind2;
P1=ind1;
ind=0;
}
```
/* Выводим результат Z на UART  в десятичной системе счисления*/
```
printf(" z=%d\n",z);//вывод на UART
while(1);
}
```

Из сравнения видно, что программа, написанная на языке Си отличается лучшей читабельностью и меньшим объемом записи. Она хорошо структурирована. Для ее написания не требуется знания системы команд МК.

## Отладка программы в дебагере ProView

После успешной компиляции программы необходимо выполнить ее отладку во встроенном в ProView дебагере (отладчике). При этом необходимо выполнить следующее:

5) Убедиться, что таблица кодов длиной N расположена в памяти кодов (code) начиная с адреса ADR2. Для этого зайдите в меню View и выберите Code. Откроется окно просмотра памяти кодов (рис.3).
6) Убедиться, что программа расположена в памяти кодов начиная с адреса ADR1.
7) Выполнить пошаговое выполнение программы (по F7).
8) Найти результат вычисления функции F, расположенный по адресу ADR3. Для этого зайдите в меню View, выберите Data dump и далее Data View.

## Содержание отчета

Отчёт о лабораторной работе должен содержать:

7) Титульный лист.
8) Цель и задачи работы.
9) Исходные данные, в соответствии с вариантом.
10) Контрольный ручной просчет.
11) Текст программы вычисления значения функции F на языке Си с подробными комментариями.
12) Выводы по работе.

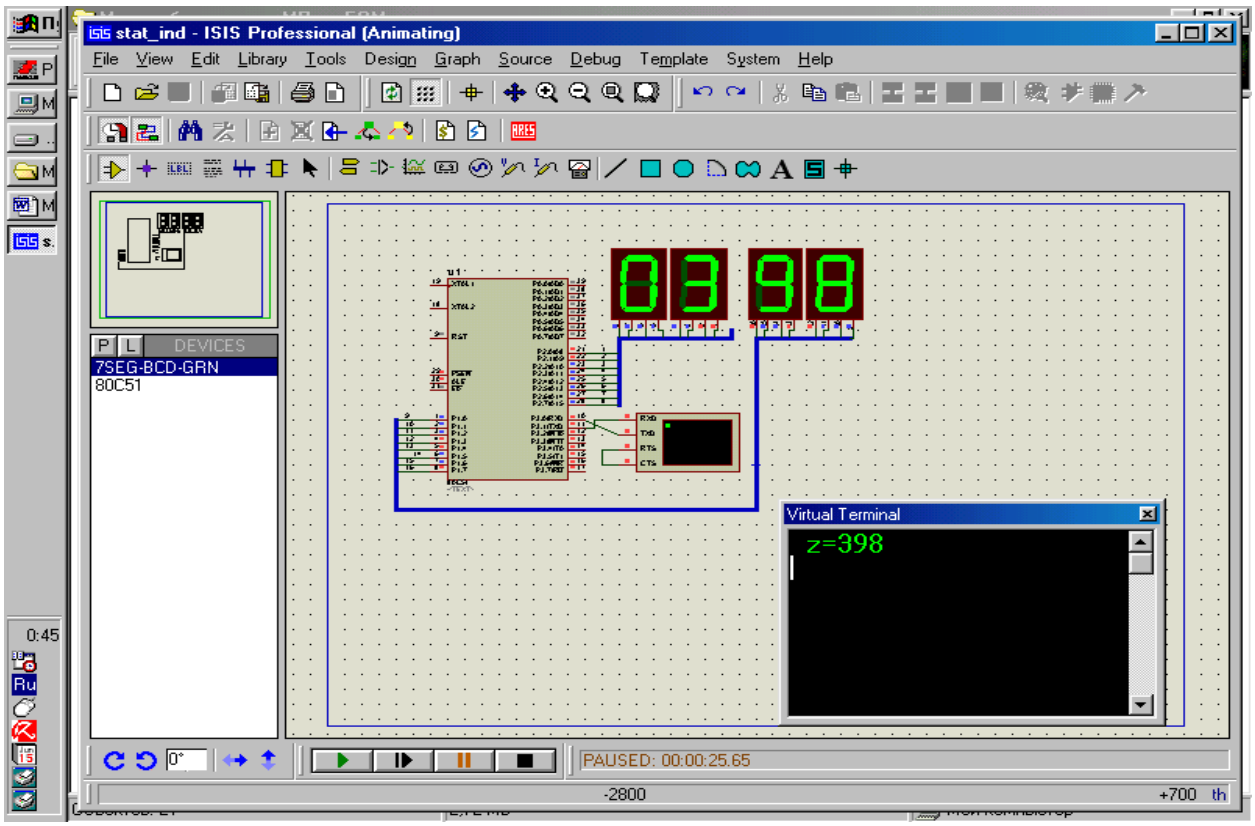



Рисунок 1. Результат работы программы в PROTEUS

Контрольные вопросы:

a. Приведите знаки арифметических операций и арифметических операций с присваиванием. Запишите правильно S=S+B;
b. Приведите знаки логических операций и логических операций с присваиванием. Запишите правильно S=S\/B;
c. Приведите знаки операции отношение.
d. Приведите форму записи оператора ЕСЛИ.
e. В чем состоит разница между логическими операциями и логическими побитовыми операциями, на примере Z=X&Y; Z=X&&Y.
f. Приведите форму записи оператора цикла FOR.

## Лабораторная работа №3

Арифметические операции в форме с плавающей запятой.
**Цель работы:** приобретение начальных навыков использования библиотек стандартных программ и выполнения операций ввода-вывода.

### Краткие теоретические сведения.

### Стандартные функции ввода/вывода

Простейшими функциями ввода/вывода в языке С являются **getchar()** и **putchar()**, которые предназначены для посимвольного обмена данными (обе объявлены в стандартном библиотечном файле **stdio.h**). Функция getchar() возвращает символ, принятый от UART, а функция putchar(), наоборот, выводит символ. Все остальные стандартные функции ввода-вывода базируются на них.

Рассмотрим пример простой программы, выводящей в бесконечном цикле через UART символы “1”, ”2”, “3”, “4”, “5”.

```
#include<reg52.h>
```

```
#include<stdio.h>
char code S[] = "12345";
void main()
{
char i;//Объявили переменную
	SCON=0x50;//Настроили UART на первый режим, прием разрешен
	RCAP2H=0xFE;//Данные для автоперезагрузки T2, скорость 1200 Бод
	RCAP2L=0xC8;
	RCLK=1;//T2 тактирует приемник UART
	TCLK=1;//T2 тактирует передатчик UART
	TR2=1;//Запуст T2
	TI=1;//Установили влаг передатчика
	i=0;
	while(1)
		{
			putchar(S[i]);//Выдали i-й символ через UART
			i++;
			if (i==5) i=0;//Проверка выхода за пределы массива
		}
}
```

Функции **puts ()** и **printf ()** используют функцию putchar() для вывода посимвольно строки в порт RS232, назначенным последним. Предположим, у нас есть определение строки, предназначенной для вывода через последовательный интерфейс:

```
char s[6] = "12345";
```

Вызов функции puts () для вывода этой строки выглядит просто как puts (s) ;.

**Функция printf () несколько сложнее**, поскольку позволяет применить форматированный вывод. Она имеет следующий синтаксис вызова:

```
printf("Строка, в которую подставляются переменные ", переменная]., переменная2, .... ) ;
```

В выводимой строке обычно используются спецификации форматирования значений переменных, переданных в качестве параметров. Каждая такая спецификация начинается со знака "%", после которого следуют обозначения параметров форматирования:

c – вывод параметра в виде символа ASCII;

d – вывод параметра в виде целого числа;

e – вывод параметра в экспоненциальном формате;

f – вывод параметра в виде вещественного числа;

ld – вывод параметра в виде длинного целого со знаком;

lu – вывод параметра в виде беззнакового длинного целого;

Lx –вывод параметра в виде беззнакового длинного целого в шестнадца-теричном представлении, используя нижний регистр символов;

LX –вывод параметра в виде беззнакового длинного целого в шестнадца-теричном представлении, используя верхний регистр символов;

s – вывод параметра в виде строки;

u – вывод параметра в виде беззнакового целого;

x – вывод параметра в виде беззнакового целого в шестнадцатеричном представлении, используя нижний регистр символов;

X – вывод параметра в виде беззнакового целого в шестнадцатеричном представлении, используя верхний регистр символов;

% – вывод знака процента.

Количество спецификаций должно совпадать с количеством переменных. При этом первая встретившаяся спецификация соответствует первой переменной в списке, вторая – второй и т.д.

В случае вывода числовых значений сразу же после знака "%" может быть указано количество символов и формат для отображения чисел, например:

8 – восемь знаков;

08 – восемь знаков с дополнением ведущими нулями; 8.2 – восемь знаков, два знака после десятичной точки. Примеры использования спецификаций форматирования:

```
int   n   =   12 3;
char c = '$';
float f = 23.5;
char str[] = "Hello";
printf("n = %d, str = %s", n, str);      //n = 123, str = Hello
printf("n = %d : 0x%04X", n, n);     //n = 123:0x007B
printf("Salary = %c%8.2f", c, f);     //Salary = $23.50
```

**Задания для проведения лабораторной работы.**

Составить программу вычисления функции F по заданной формуле для заданного диапазона изменения значений аргумента. Вычисления производить в форме с плавающей запятой. Вывод результатов вычисления выполнить на UART.

Программа должна начинаться с ячейки 0x100. Результаты вычислений оформить в массив, начиная с адреса 0x500 внешней памяти. Варианты заданий приведены в табл.1.

| № вар. | Формула | Диапазон значений аргумента |
|---|---|---|
| 01 | Fi = 20*SIN Xi | $\pi/6<=Xi<=\pi$  Xi=Xi-1+h; h=π/6 |
| 02 | Fi = 40*COSXi | $\pi/6<=Xi<=\pi$  Xi=Xi-1+h; h=π/6 |
| 03 | Fi = 40*(SIN Xi + COS Xi) | $\pi/6<=Xi<=\pi$  Xi=Xi-1+h; h=π/6 |
| 04 | Fi = √40*TG Xi | $\pi/6<=Xi<=\pi$  Xi=Xi-1+h; h=π/6 |
| 05 | Fi = LG Xi | 10<=Xi<=100; Xi=Xi-1+h; h=10 |
| 06 | Fi = LN Xi | 10<=Xi<=100; Xi=Xi-1+h; h=10 |
| 07 | Fi = √Xi | 10<=Xi<=100; Xi=Xi-1+h; h=10 |
| 08 | Fi = √80*TG Xi | $\pi/6<=Xi<=\pi$  Xi=Xi-1+h; h=π/6 |
| 09 | Fi = 80*COSXi | $\pi/6<=Xi<=\pi$  Xi=Xi-1+h; h=π/6 |
| 10 | Fi = 60*SIN Xi | $\pi/6<=Xi<=\pi$  Xi=Xi-1+h; h=π/6 |
| 11 | Fi = 80*(SIN Xi + COS Xi) | $\pi/6<=Xi<=\pi$  Xi=Xi-1+h; h=π/6 |
| 12 | Fi = 6,5*LG Xi | 10<=Xi<=100; Xi=Xi-1+h; h=10 |
| 13 | Fi =2,3* LN Xi | 10<=Xi<=100; Xi=Xi-1+h; h=10 |
| 14 | Fi =7,3* √Xi | 10<=Xi<=100; Xi=Xi-1+h; h=10 |
| 15 | Fi = 4,8*(SIN Xi + COS Xi) | $\pi/6<=Xi<=\pi$  Xi=Xi-1+h; h=π/6 |

**Пример выполнения задания.**

```
//Директивы компилятора, присваивающие значения именам
#include <math.h>
#include<stdlib.h>
#include<stdio.h>
#define ADR1 0x714
```

```
#define ADR2 0x431
#define N 0xa
#define ADR3 0x32

//Описание глобальных переменных и констант **********************
at ADR2 float code x[N]={2.2,2.4,6.8,8.4,10.1,
6.8,7.2,6.4,9.1,1.9};
at ADR3 float data result, sum,z;
//Основная функция, располагаемая с адреса ADR1 ******************
at ADR1 void main()
{
//Описание локальных переменных
unsigned char i; //Переменная цикла
        sum=0;
        for(i=0;i<=(N-1);i++)
                {
         sum=sum+x[i];
                 z=600*(sin(x[i])+sqrt(x[i]))/exp(x[i]);
/* Выводим на UART значения переменных x и z в формате с плавающей запятой */
        printf(" x=%2.1f  z=%2.2f  \n",x[i],z);
                }
        result=sum/N;
/* Выводим на UART значения переменных result и sum в формате с плавающей запятой */

        printf("result=%2.2f,sum=%2.2f",result,sum);
        while(1);//Зацикливание
}
```

**Содержание отчета**

Отчёт по лабораторной работе должен содержать:

13) Титульный лист.
14) Цель и задачи работы.
15) Исходные данные, в соответствии с вариантом.
16) Текст программы вычисления значения функции F на языке Си с подробными комментариями.
17) Результаты расчета, выполненного программой.
18) Выводы по работе.

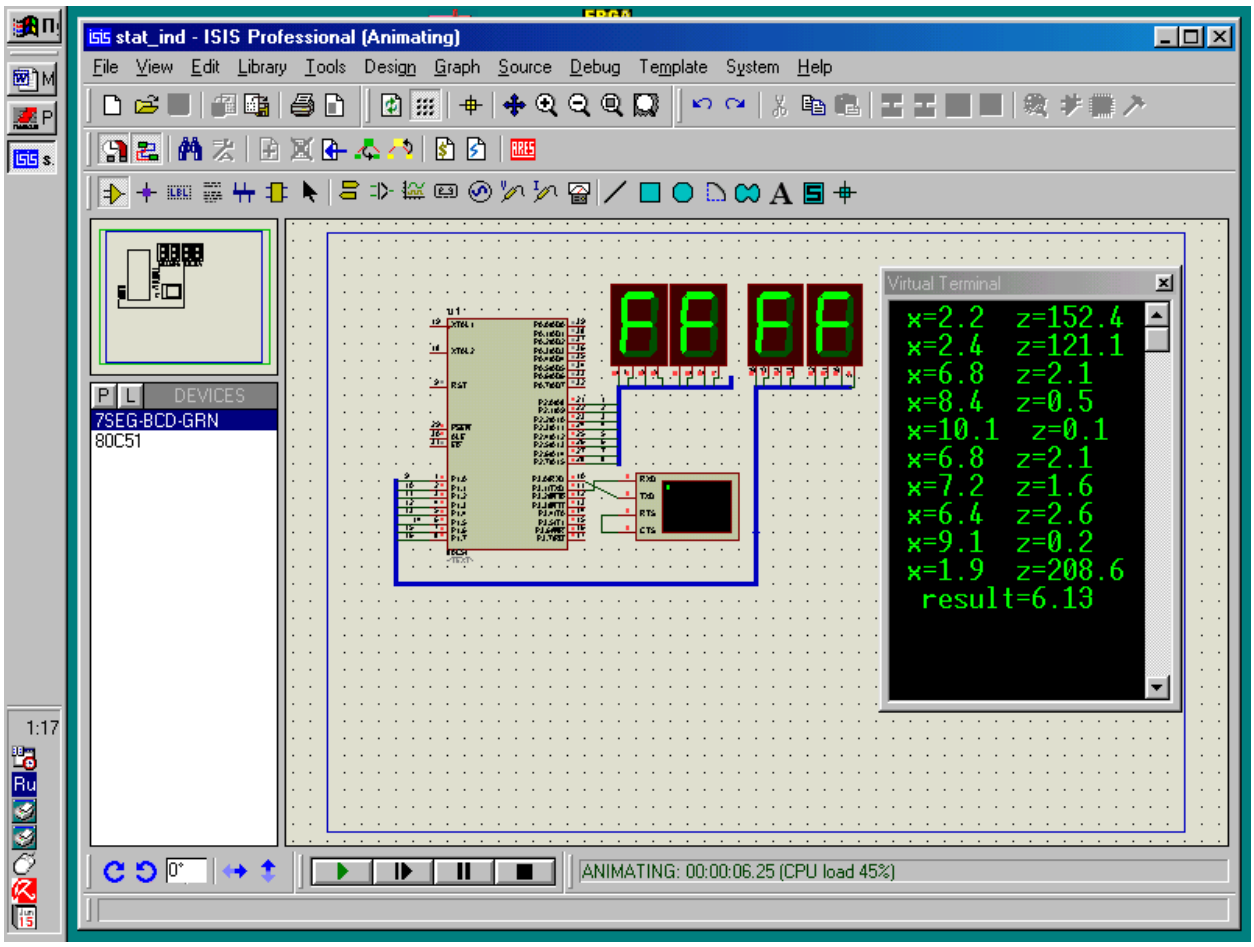



Рисунок Результаты работы программы в PROTEUS

Контрольные вопросы:

a. Приведите знаки арифметических операций и арифметических операций с присваиванием. Запишите правильно S=S+B;
b. Запишите оператор вывода по формату на UART следующих переменных: a=0,94, b=0,9476, c=231;
c. Запишите оператор вывода по формату на UART следующих переменных: a=3,2, b=24,94, c=123,1;
d. Приведите все спецификации вывода.

**Лабораторная работа №4**

**Выполнение операций над битовыми данными**

*Цель работы:* изучение организации битового пространства памяти микроконтроллера КМ1816ВЕ51, программных средств обработки бит.

1. КОМАНДЫ РАБОТЫ С БИТОВЫМИ ДАННЫМИ

Данная группа команд оперирует с однобитными операндами. В качестве операндов могут выступать отдельные биты некоторых регистров специальных функций, биты портов и биты 16 ячеек внутренней памяти данных. Все
адресуемые биты образуют одноразрядное линейно упорядоченное пространство

BSEG емкостью 256 бит. В пространстве BSEG используется только прямая адресация, прямой восьмиразрядный адрес в пространстве BSEG обозначается bit. Адресация в пространстве BSEG иллюстрируется таблицей 1.Группа команд операций с битами (табл. 2) включает 6 операций: три одноместных операции установки (CLR), сброса (SETB) и инверсии (CPL), две двухместных операции конъюнкции и дизъюнкции, и операцию пересылки. В качестве "аккумулятора" в битовых операциях используется триггер (флаг) переноса C. Характеристики битовых команд приведены в табл.2.

Таблица 1. Адресация в пространстве BSEG

| Адрес в пространстве DSEG | Разряд ячейки внутренней памяти данных | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 20 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| 21 | 0F | 0E | 0D | 0C | 0B | 0A | 09 | 08 |
| 22 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 |
| 23 | 1F | 1E | 1D | 1C | 1B | 1A | 19 | 18 |
| 24 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
| 25 | 2F | 2E | 2D | 2C | 2B | 2A | 29 | 28 |
| 26 | 37 | 36 | 35 | 34 | 33 | 32 | 31 | 30 |
| 27 | 3F | 3E | 3D | 3C | 3B | 3A | 39 | 38 |
| 28 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 |
| 29 | 4F | 4E | 4D | 4C | 4B | 4A | 49 | 48 |
| 2A | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 |
| 2B | 5F | 5E | 5D | 5C | 5B | 5A | 59 | 58 |
| 2C | 67 | 66 | 65 | 64 | 63 | 62 | 61 | 60 |
| 2D | 6F | 6E | 6D | 6C | 6B | 6A | 69 | 68 |
| 2E | 77 | 76 | 75 | 74 | 73 | 72 | 71 | 70 |
| 2F | 7F | 7E | 7D | 7C | 7B | 7A | 79 | 78 |

Регистры специального назначения

| Адрес в пространстве DSEG | Разряд регистра специального назначения | | | | | | | | Регистр |
|---|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| 80 | 87 | 86 | 85 | 84 | 83 | 82 | 81 | 00 | PO |
| 81 | 8F | 8E | 8D | 8C | 8B | 8A | 89 | 88 | TCON |
| 90 | 97 | 96 | 95 | 94 | 93 | 92 | 91 | 90 | P1 |
| 98 | 9F | 9E | 9D | 9C | 9B | 9A | 99 | 98 | SCON |
| A0 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | P2 |
| A8 | AF | -- | -- | AC | AB | AA | A9 | A8 | IE |
| B0 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 | P3 |
| B8 | -- | -- | -- | BC | BB | BA | B9 | B8 | IP |
| C8 | CF | CE | CD | CC | CB | CA | C9 | C8 | T2CON |
| | | | | | | | | | |
| D0 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | PSW |
| E0 | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 | A |
| F0 | F7 | F6 | F5 | F4 | F3 | F2 | F1 | F0 | B |

## 2. ЗАДАНИЯ

Пусть в DSEG (DATA), в ячейке ADR2 расположен код CODE. Записать на языке СИ МК К1816ВЕ51 программу, которая выполняет вычисление заданной булевой функции F над этими кодами. Результат вычислений должен быть записан по адресу ADR3 пространства BSEG. Программа должна начинаться с ячейки ADR1.

Таблица 3. Варианты заданий

| Номер | ADR1 | ADR2 | CODE | ADR3 |
|---|---|---|---|---|
| 01 | 333 | 43 | FE | 7A |
| 02 | A45 | 54 | 3F | 4B |
| 03 | 563 | 62 | 5D | 6C |
| 04 | 234 | 71 | 7C | 7D |
| 05 | D41 | 12 | 8B | 1E |
| 06 | 872 | 23 | 7A | 2F |
| 07 | 45F | 34 | 1E | 4F |
| 08 | 32E | 45 | F2 | 5A |
| 09 | 695 | 56 | E3 | 6B |
| 10 | 5A2 | 67 | FA | 3C |
| 11 | 4B3 | 78 | A5 | 4D |
| 12 | 3C4 | 23 | 56 | 5E |
| 13 | 2DF | 3A | F7 | 6F |
| 14 | 1E3 | 4B | A8 | 7A |
| 15 | 7FE | 5C | 48 | 5B |
| 16 | 6E2 | 6D | 29 | 6C |
| 17 | 5A2 | 7E | F3 | 3D |
| 18 | 4D4 | 1F | E2 | 5E |
| 19 | 3C3 | 2E | D3 | 6F |
| 20 | 261 | 3D | A4 | 5A |
| 21 | 3BF | 5D | B5 | 3D |
| 22 | 2CC | 53 | 3D | 2E |
| 23 | 3D3 | 6A | 4F | 3E |
| 24 | 7E3 | 7B | 8F | 48 |
| 25 | 2F4 | 5E | 6D | 5B |
| 26 | 4E4 | 45 | D9 | 6C |

Вид функции F для вычисления определяется следующим образом.

Пусть YZ две последние цифры номера зачетки.

Код CODE имеет формат (поразрядно)

| X7 | X6 | X5 | X4 | X3 | X2 | X1 | X0 |
|---|---|---|---|---|---|---|---|

F=(X7)OP1(X6)OP2(X5)OP3(X4)OP4(X3)OP5(X2)OP6(X1)OP7(X0), где OPi - булева операция, находится из таблицы 4.

Таблица 4. Таблица операций

| Z | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| OP1 | ∧ | ∧ | ∧ | ∧ | ∧ | ∨ | ∨ | ∨ | ∨ | ∨ |
| OP2 | + | + | + | ∧ | ∧ | ∧ | ∨ | ∨ | ∨ | ∨ |
| OP3 | ∨ | ∨ | ∨ | + | + | + | + | ∨ | + | ∧ |
| OP4 | + | + | ∨ | ∧ | ∨ | ∧ | ∧ | ∧ | ∨ | ∨ |
| OP5 | ∧ | ∧ | ∨ | ∨ | ∨ | + | + | + | + | + |
| OP6 | + | ∨ | ∧ | ∨ | ∧ | ∧ | + | + | ∧ | ∧ |
| OP7 | ∨ | + | + | + | + | + | ∨ | ∧ | + | ∨ |

∧ - операция И
∨ - операция ИЛИ
+ - операция Исключающее ИЛИ

## 4. ПРИМЕР ВЫПОЛНЕНИЯ РАБОТЫ

Пусть в DSEG, в ячейке ADR2=31 расположен код COD1=0x2D. Записать на языке СИ МК1816ВЕ51 программу, которая выполняет вычисление заданной булевой функции F=(X5/\X4)+X3\/X0 над этими кодами. Результат вычислений должен быть записан по адресу F1=7F пространства BSEG. Программа должна начинаться с ячейки ADR1=0xF00.

```
/*Реализация логических функций контроллером МК51*/
#include<stdlib.h>
/*Определение символических адресов*/
#define adr1 0x0f00
#define adr2 0x31
#define cod1 0x2d
#define rab 0x20
at adr2 unsigned char data cde1=cod1;
at rab unsigned char data cde;
at 0x00 bit x0;
at 0x01 bit x1;
at 0x02 bit x2;
at 0x03 bit x3;
at 0x04 bit x4;
at 0x05 bit x5;
at 0x06 bit x6;
at 0x07 bit x7;
at 0x08 bit z1;
```

```
at 0x09 bit z2;
at 0x0a bit f;
at 0x7f bit f1;
at adr1 void main()
{
cde=cde1;
z1=x5&x4;
z2=z1^x3;
f=z2|x0;
f1=f;
}
```

**Содержание отчета**

Отчёт о лабораторной работе должен содержать:

19) Титульный лист.
20) Цель и задачи работы.
21) Исходные данные, в соответствии с вариантом.
22) Контрольный ручной просчет.
23) Текст программы вычисления значения функции F на языке Си с подробными комментариями.
24) Выводы по работе.

Контрольные вопросы:

a. Как присваивают имена битовым переменным?
b. Где находится битовое пространство для пользователя? Укажите его адреса в ОЗУ.
c. Приведите знаки для выполнения логических и поразрядных операций.
d. Как адресуются биты регистров специальных функций?
e. Для чего нужен прототип #include<reg51.h>?
f. Приведите программу реализации функции Y=X1\/X2. Значения переменных расположить произвольно в битовом пространстве.

**Лабораторная работа №5**

ИССЛЕДОВАНИЕ РАБОТЫ СИСТЕМЫ ПРЕРЫВАНИЙ МК51

*Цель работы*: изучение системы прерываний микроконтроллера КМ1816ВЕ51.

2. ОРГАНИЗАЦИЯ СИСТЕМЫ ПРЕРЫВАНИЙ

Микроконтроллер поддерживает двухуровневую приоритетную систему прерываний с шестью источниками запросов.

Имеется два внешних запроса на прерывания - линии ~INT0, ~INT1 (разряды 2-й и 3-й третьего порта ввода/вывода, P3.2 и P3.3, соответственно).

Источниками внутренних запросов могут служить;

- счетчик/таймер 0,
- счетчик/таймер 1,
- счетчик/таймер 2,
- порт последовательного ввода/вывода.

## РЕГИСТРЫ СИСТЕМЫ ПРЕРЫВАНИЙ

Блок регистров системы прерываний включает регистр управления TCON, формат которого приведен на рис.5.2, и два регистра для управления режимом прерываний и уровнями приоритетов. Форматы этих регистров приведены на рис. 5.3. и 5.4., соответственно.

Регистр TCON (адрес 88h в пространстве DSEG)

| Адрес бита в пространстве BSEG | 8F | 8E | 8D | 8C | 8B | 8A | 89 | 88 |
|---|---|---|---|---|---|---|---|---|
| Назначение разрядов регистра | TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |

TCON.0 IT0 - управление типом входа ~INT0
IT0 = 1 - по спадающему фронту (динамический вход)
IT0 = 0 - статический вход
TCON.1 IEO - флаг запроса на прерывание ~INT0
при динамическом входе
TCON.2 IT1 - управление типом входа ~INT1
IT1 = 1 - по спадающему фронту (динамический вход)
IT1 = 0 - статический вход
TCON.3 IE1 - флаг запроса на прерывание ~INTI
при динамическом входе
TCON.4 TR0 - флаг программного управления CT0
TCON.5 TF0 - флаг запроса прерывания по переполнению CT0
TCON.6 TR1 - флаг программного управления CT1
TCON.7 TF1 - флаг запроса прерывания по переполнению CT1

Рис.5.2. Регистр управления таймерами /счетчиками
и системой прерываний

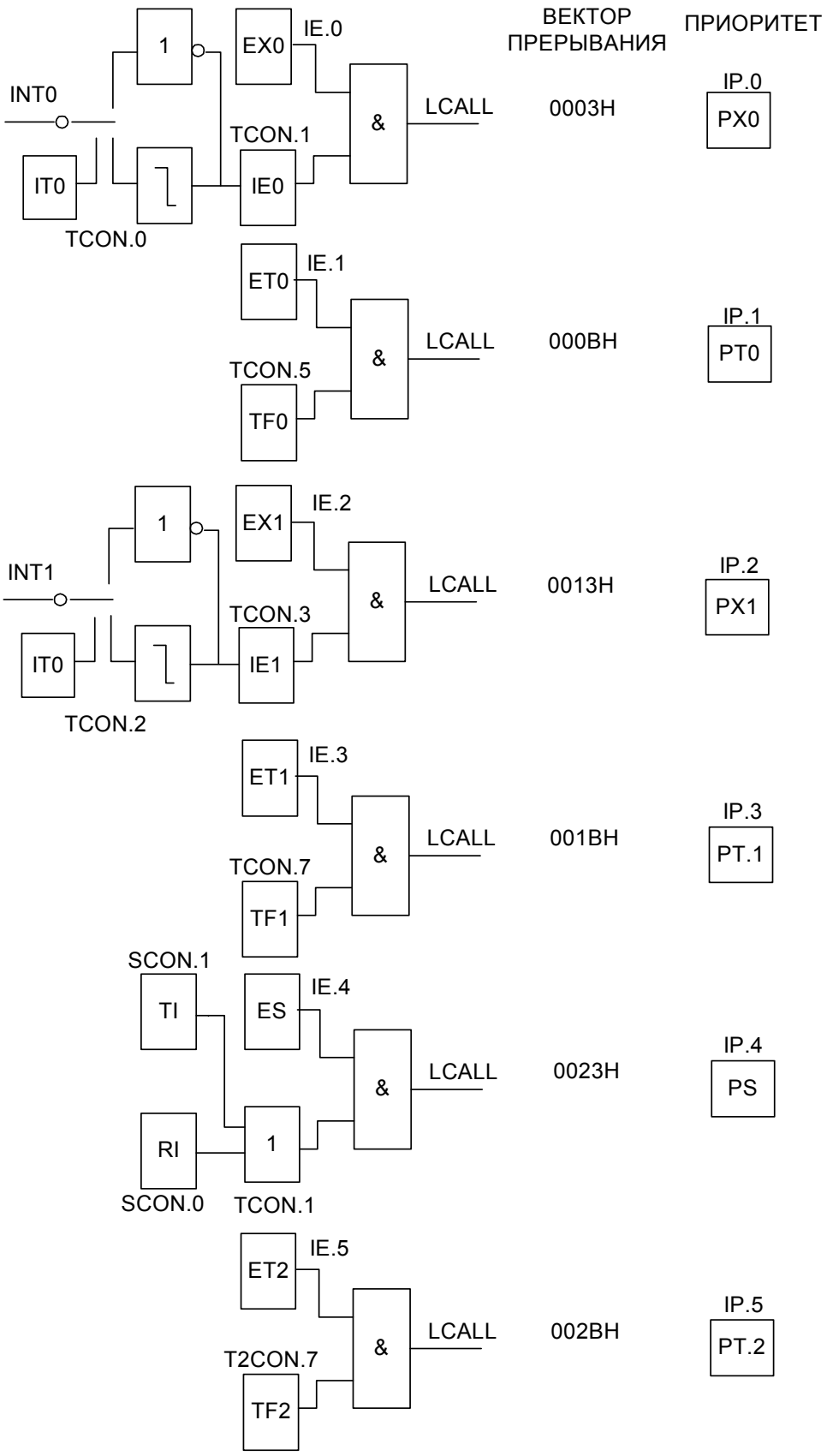



Рис.5.1. Архитектура системы прерываний микроконтроллера AT89C52.

Регистр IE (адрес A8h в пространстве DSEG)

| Адрес бита в пространстве BSEG | AF | AE | AD | AC | AB | AA | A9 | A8 |
|---|---|---|---|---|---|---|---|---|
| Назначение разрядов регистра | EA | — | ET2 | ES | ET1 | EX1 | ET0 | EX0 |

IE.0 EX0 - маска IE0 или ~ INT0
IE.1 ET0 - маска TF0
IE.2 EX1 - маска IE1 или ~ INT1
IE.3 ET1 - маска TF1
IE.4 ES - маска TI/RI
IE.5 ET2 - маска TF2
IE.7 EA - общее разрешение прерывания

Рис. 5.3. Регистр разрешения прерываний

Регистр IP (адрес B8h в пространстве DSEG)

| Адрес бита в пространстве BSEG | BF | BE | BD | BC | BB | BA | B9 | B8 |
|---|---|---|---|---|---|---|---|---|
| Назначение разрядов регистра | — | — | PT2 | PS | PT1 | PX1 | PT0 | PX0 |

IP.0 PX0 - приоритет IE0 или ~INT0
IP.1 PT0 - приоритет TF0
IP.2 PX1 - приоритет IE1 или ~ INT1
IP.3 PT1 - приоритет TF1
IP.4 PS - приоритет T1/ RI
IP.5 PT2 - приоритет TF2

Рис.5.4. Регистр приоритетов прерываний

ОБСЛУЖИВАНИЕ ПРЕРЫВАНИЙ

Линии ~INT0 и ~INT1 (P3.2, P3.3) могут быть запрограммированы на срабатывание, как по спадающему фронту сигнала, так и по низкому уровню сигнала. Управление типом вход выполняется флажками IT0 и IT1, соответственно. При IT0=1, устанавливается режим срабатывания по спадающему фронту сигнала, при IT0=0, устанавливается режим срабатывания по L уровню сигнала на входе ~INT0.

Аналогично и для IT1. Запросы на прерывание от внешних источников устанавливают флажки IE0 и IE1. В случае работы по спадающему фронту эти флажки сбрасываются аппаратно, при приеме соответствующего запроса на обработку. При работе по уровню, флажки IE0, IE1 отслеживают состояние сигнала на входе, их сброс необходимо предусмотреть в программе обработки соответствующим обращением к устройству, выставившему запрос.

Переполнение счетчика/таймера приводит к установке флага переполнения TF0 для счетчика/таймера 0 и TF1 для счетчика/таймера1. При приеме запроса на обработку прерывания от счетчика/таймера соответствующий флажок аппаратно сбрасывается.

Запросы на прерывание могут поступать и от последовательного порта ввода/вывода. Порт может взвести флажок TI - запрос на прерывание от передатчика или

флажок RI - запрос на прерывание от приемника. Оба запроса воспринимаются как запрос от одного источника. Флажки RI,TI при приеме запроса на прерывание от порта аппаратно не сбрасываются. Их обнуление должно быть предусмотрено в обработчике прерывания.

Все флажки, фиксирующие запросы на прерывания могут устанавливаться и сбрасываться программно, с помощью соответствующих команд работы в пространстве BSEG. Т.е. существует возможность вызывать прерывания чисто программным путем. С помощью регистра разрешения прерываний можно замаскировать любой из источников прерываний индивидуально или запретить/разрешить работу системы прерываний в целом.

Каждый источник прерываний может иметь два уровня приоритета, который задается установкой или сбросом соответствующего разряда в регистре уровня приоритета прерывания IP. Установка разряда в 1 соответствует высокому приоритету. Установка разряда в 0 - низкому.
При обслуживании низкоприоритетного прерывания, запрос на высокоприоритетное прерывание может быть принят и обслужен. При обслуживании высокоприоритетного прерывания запрос на низкоприоритетное прерывание игнорируется.

При одновременном появлении нескольких запросов одного уровня используется арбитраж в соответствии со следующими приоритетами:

| Приоритет | Источник |
|---|---|
| 0 (высший) | IE0 |
| 1 | TF0 |
| 2 | IE1 |
| 3 | TF1 |
| 4 (низший) | RI/TI |

При приеме запроса аппаратно генерируется команда LCALL vect, которая обеспечивает переход к начальному адресу vect соответствующей процедуры обслуживания. Каждому источнику соответствует свой адрес:

Вектор прерываний
0003H EXTI0 IE0 - внешний запрос 0
000BH TIMER0 TF0 - по таймеру/счетчику 0
0013H EXTI1 IE1 - внешний запрос 1
001BH TIMER1 TF1 - по таймеру/счетчику 1
0023H SINT Ri/TI - от последовательного порта

Запрос на прерывание может быть принят в конце каждого цикла выполнения команды, кроме команды RETI и команд, работающих с регистрами IE и IP.

При переходе к процедуре обслуживания текущее состояние PC загружается в стек, обеспечивая возврат по команде RETI, которая должна завершать работу обработчика прерываний. Эта команда отличается от обычной RET, тем, что сообщает системе прерываний о конце обслуживания прерывания данного уровня. Это позволяет перейти к обслуживанию прерывания низкого уровня, если на него поступил запрос во время обработки высокоуровневого. Общий алгоритм обработки прерываний микроконтроллера приведен на рис.5.

ЗАДАНИЕ

Составить программу инициирующую прерывание заданного типа TYPE программным путем. Обработчик прерывания типа TYPE выдает очередной элемент массива в порт P0 и на индикацию. Массив длиной N находится в памяти программ по адресу ADR2.

Программа начинается с адреса ADR0. Обработчик прерывания

Располагается в памяти по адресу ADR1.

Таблица 2. Варианты заданий

| Номер | ADR0 | TYPE | ADR1 | ADR2 | N |
|---|---|---|---|---|---|
| 01 | 614 | IE0 | 714 | 431 | E |
| 02 | 52F | TF0 | 62F | 541 | F |
| 03 | 43E | IE1 | 53E | 621 | D |
| 04 | 34A | TF1 | 44A | 711 | C |
| 05 | 255 | RI | 355 | 121 | B |
| 06 | 366 | TI | 266 | 236 | A |
| 07 | 277 | IE0 | 177 | 345 | 6 |
| 08 | 68A | TF0 | 78A | 454 | 7 |
| 09 | 79D | IE1 | 69D | 568 | F |
| 10 | 3AF | TF1 | 5AF | 677 | E |
| 11 | 5BA | RI | 4BA | 781 | 5 |
| 12 | 4C9 | TI | 3C9 | 231 | 6 |
| 13 | 1D8 | IE0 | 2D8 | 3A1 | 7 |
| 14 | 2E4 | TF0 | 1E4 | 4B2 | 8 |
| 15 | 3F5 | IE1 | 7F5 | 5C2 | 8 |

## Описание функций-обработчиков прерываний

Микроконтроллеры 8051 и их модификации имеют систему прерываний, которая используется для измерения временных интервалов, обнаружения и подсчета внешних событий, обмена данными по последовательному порту.
В таблице 5 приведен стандартный базовый набор прерываний МК 8051.

Табл.5

***Прерывания МК 8051***

| Номер прерывания | Источник прерывания | Адрес перехода |
|---|---|---|
| 0 | По INT0 | 0003h |
| 1 | Таймер/счетчик 0 | 000Bh |
| 2 | По INT1 | 0013h |
| 3 | Таймер/счетчик 1 | 001Bh |
| 4 | UART | 0023h |

Функции-обработчики прерываний являются системными функциями. В описании их заголовка указываются ключевое слово ***interrupt*** и **номер прерывания**.

**Примеры.**

1) Объявление функции-обработчика прерываний от таймера T0

```
void timer_0 (void) interrupt 1
    {
        набор операторов…
    }
```

2) Объявление функции-обработчика прерываний по INT1

```
void prerivanie_INT1 (void) interrupt 2
    {
        набор операторов…
    }
```

3) Объявление функции-обработчика прерываний по UART

```
void SerialPort (void) interrupt 4
```

```
{
        набор операторов…
}
```

Пример выполнения задания на языке программирования СИ51

```
/*По каждому прерыванию INT1 на индикацию выдается очередной код из массива
MEMR[]  на UART и статическую цифровую индикацию – 4 десятичные цифры*/
#include <stdio.h>
#include <reg51.h>
#define const1 0x3fff
#define const2 0x7fff
#define adr0 0x0100
#define adr1 0x0200
#define n 10
at 0x0300 unsigned char code
memr[]={0x01,0xf0,0x03,0x04,0x05,0x06,0x07,0x08,0x0e,0xfe};
char i=0;
at 0xb000 unsigned char xdata inm;
at 0xa000 unsigned char xdata ins;
unsigned char s[4],ind1=0,ind2=0,k=0,ind;
short p;
void prerivanie_INT0() interrupt 0
{
}
void timer_0() interrupt 1
{
}
void timer_1() interrupt 3
{
}
void serial_port() interrupt 4
{
}

at adr1 void int_1() interrupt 2
{
printf("i=%d  memr=%d  \n",i,memr[i]);
sprintf(s,"%d",memr[i]);
ind1=0;
ind2=0;
m1:
if(s[k]!=0)
{
ind=(ind1>>4)&0x0f;

ind2|=ind;
ind1=ind1<<4;
ind1=ind1|(s[k]&0x0f);
k++;
goto m1;
}
```

```
else
{
k=0;
P2=ind2;
P1=ind1;
ind=0;
}
i++;
for(p=0;p<=const1;p++);
}
at adr0 void main()
{
IE|=0x84;/*a §aҐиЁвм ўбҐ Їаў Ё Їаў int1*/
TCON|=0x04; /*ўл§®ў Їаў int1*/
m1:
IE1=1;/*§ Їа®б -  Їаў int1*/
if(i<=(n-1))goto m1;else i=0;
while(1);
}
```

## Содержание отчета

Отчёт о лабораторной работе должен содержать:

25) Титульный лист.
26) Цель и задачи работы.
27) Исходные данные, в соответствии с вариантом.
28) Текст программы  на языке Си с подробными комментариями.
29) Выводы по работе.

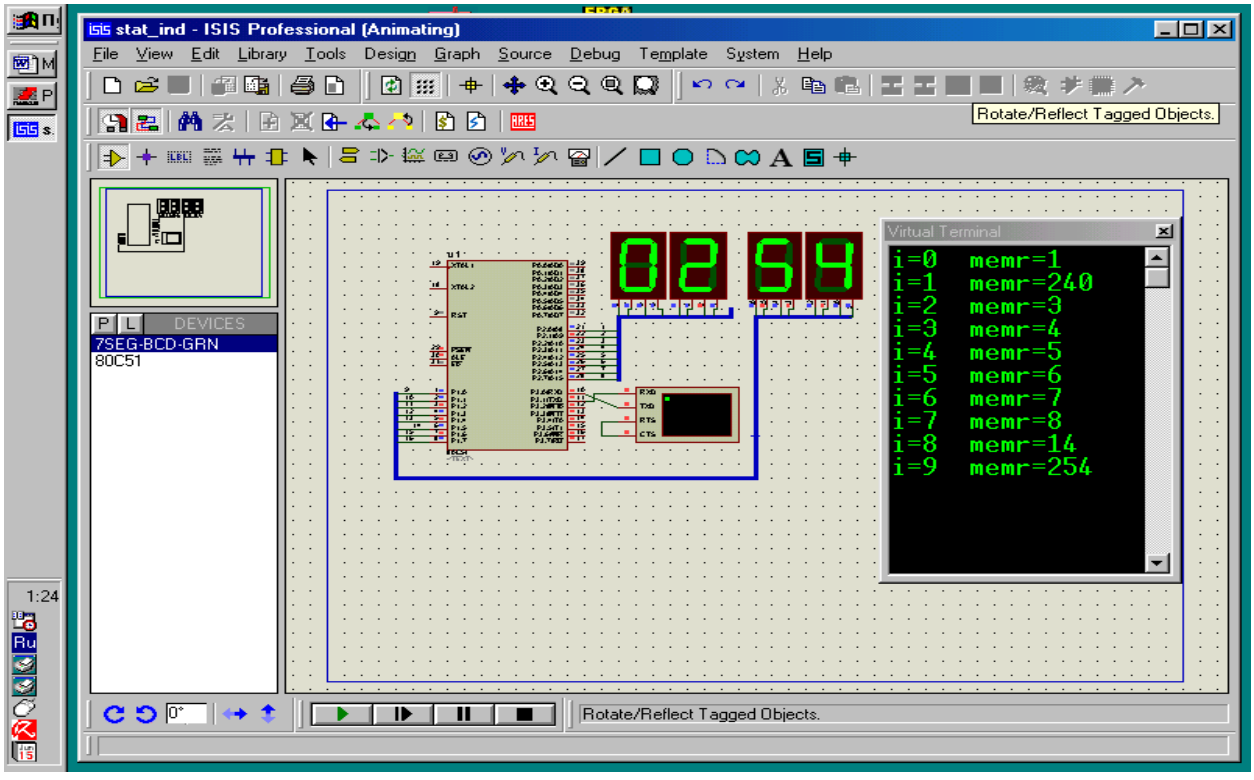



Рисунок  Результаты работы программы в PROTEUS

Контрольные вопросы:

a. Приведите назначение разрядов регистра TCON. Управление состоянием флажков этого регистра на языке программирования СИ.
b. Приведите назначение разрядов регистра IE. Управление состоянием флажков этого регистра на языке программирования СИ.
c. Как оформляется заголовок функции обработчика прерывания?
d. Назначение и действия, которые должна выполнить функция инициализации программы управления устройством, работающим с включенной системой прерываний.
e. Где находятся входы запросов на прерывание INT0, INT1 и как настроить эти входы.
f. Что представляют собой вектор прерывания и уровень прерывания?

## Лабораторная работа N6

ИССЛЕДОВАНИЕ РАБОТЫ ТАЙМЕРОВ МИКРОКОНТРОЛЛЕРА КМ1816ВЕ51

Цель работы: изучение организации службы времени микроконтроллера КМ1816ВЕ51, а также приобретение навыков программирования временных задержек в кодах микроконтроллера с использованием службы времени.

### 1. ОРГАНИЗАЦИЯ ТАЙМЕРОВ

В состав МК51 входит два 16-разрядных таймера/счетчика (рис.6.1),каждый из которых состоит из двух программно-доступных регистров TH0, TL0 (таймер 0), TH1, TL1 (таймер 1), TH2, TL2 (таймер 2).

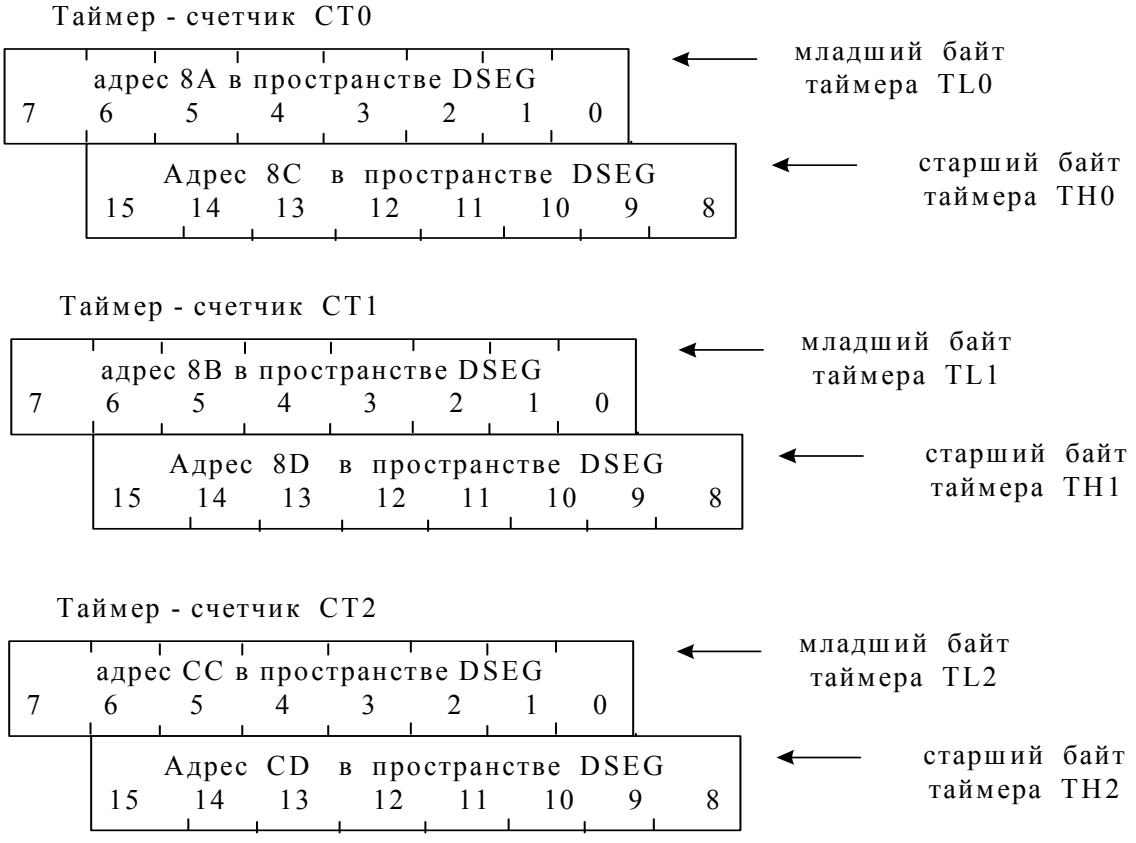



Рис. 6.1. Регистры таймеров/счетчиков

Каждый таймер/счетчик может быть запрограммирован на работу в режиме таймера - подсчета числа машинных циклов, следующих с частотой OSC/12 (обычно 1 Мгц), либо счетчика - подсчета числа переходов из H и L на соответствующих входах T0 и T1. Функцией таймера/счетчика и выбором режима его работы управляют регистры TMOD (таймеры 0 и 1), форматы которых приведены на рис. 6.2.

Регистр TMOD (адрес 89 в пространстве DSEG)

| | T1 MOD | | | | T0 MOD | | | |
|---|---|---|---|---|---|---|---|---|
| Мнемоническое обозначение разрядов регистра | GATE | C/T | M1 | M0 | GATE | C/T | M1 | M0 |

| CT0 | CT1 | | |
|---|---|---|---|
| TMOD.0 | TMOD.4 | M0 | - младший бит поля режима |
| TMOD.1 | TMOD.5 | M1 | - старший бит поля режима |
| TMOD.2 | TMOD.6 | C/T | - Выбор функции: 0-таймер / 1-счетчик |
| TMOD.3 | TMOD.7 | GATE | - Управление работой CT0 / CT1 |

Рис. 6.2. Форматы регистров режимов таймеров/счетчиков

Разряд C/T задает функцию таймера/счетчика. При C/T=0 таймер/счетчик работает в режиме таймера. При C/T = 1 - в режиме счетчика. Разряды M0, M1 определяют режим работы таймера - счетчика. Каждый таймер/счетчик может работать в 4-х режимах.

В режиме 0 счетный регистр имеет длину 13 бит (см. рис.6.3.а), причем 5 младших разрядов расположены в TL, а 8 старших - в TH. Сигнал переполнения фиксируется соответствующим флажком TF.

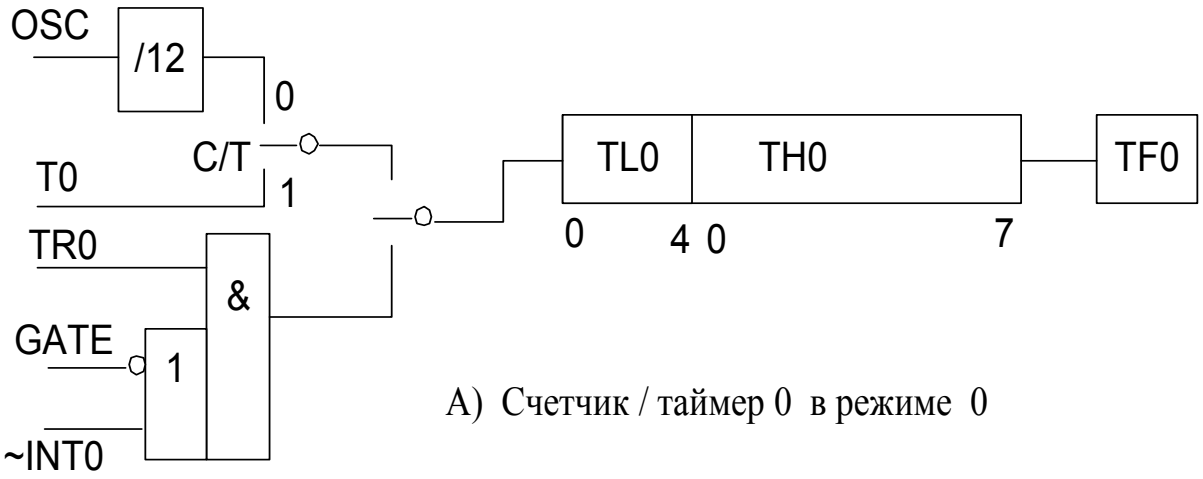
В режиме 1 счетный регистр имеет длину 16 разрядов. В остальном, режим не отличается от режима 0.

В режиме 2 (рис. 6.3.б) задается 8-разрядный регистр таймера/счетчика (TL), с автоматической загрузкой начального кода из TH. Сигнал переполнения таймера/счетчика не только взводит флажок TF, но и перегружает TL содержимым TH.

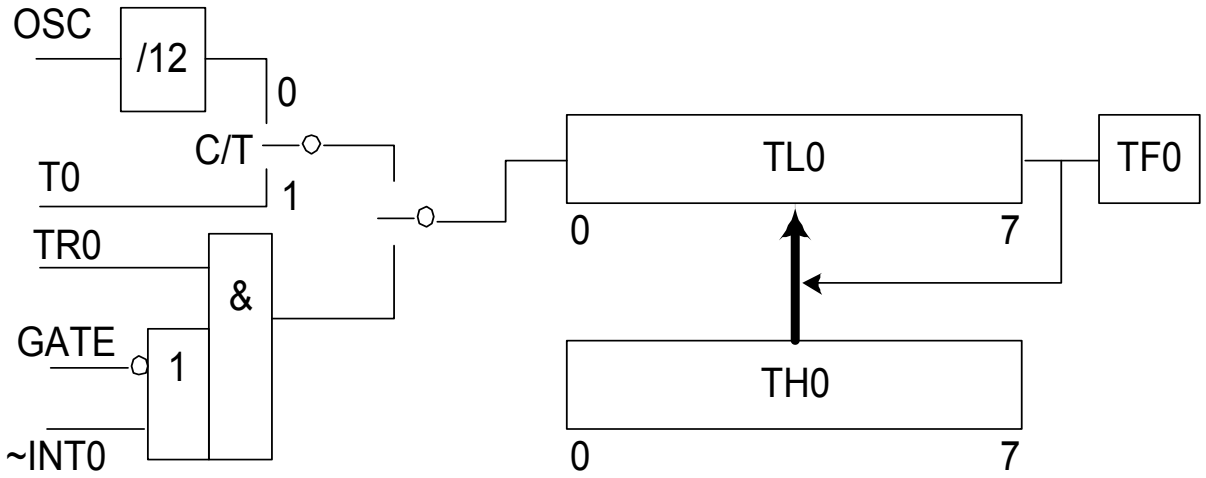
В режиме 3 работает только таймер/счетчик 0. В этом режиме два регистра CT0 рассматриваются как два независимых 8-разрядных таймера/счетчика. Переполнение TL0 вызывает взведение флага TF0, переполнение TH0 устанавливает флаг TF1.

Режимы работы таймера 2 существенно отличаются от режимов работы таймеров 0 и 1 подробно рассматриваться не будут. Автозагрузка счетных регистров таймера 2 выполняется из специальных регистров RCAP2L (загружает TL2) и RCAP2H (загружает TH2).

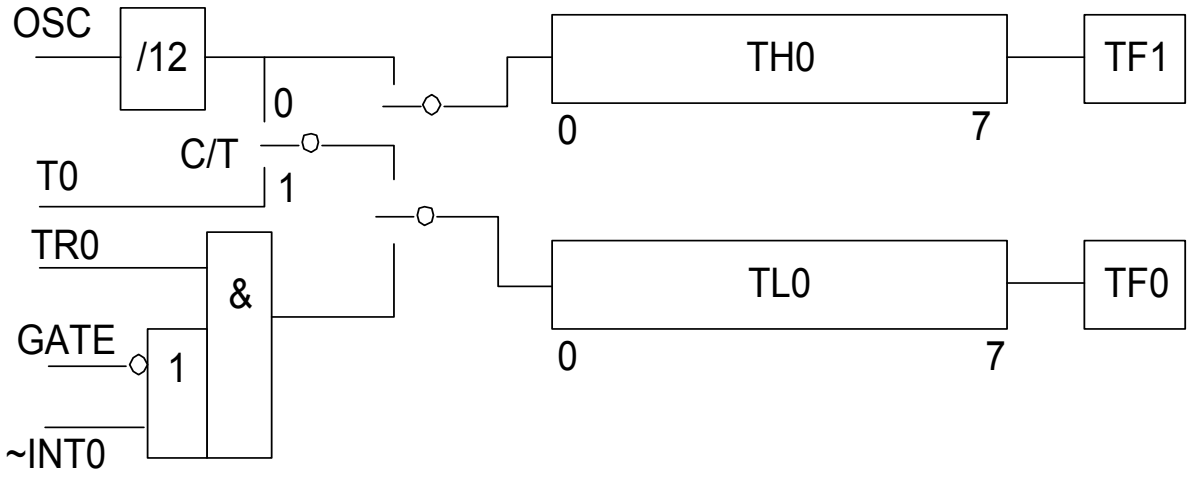
Независимо от режима работы имеется возможность программного и аппаратного управления пуском/остановкой счета импульса. Логика пуска/останова анализирует состояние разряда GATE в регистре TMOD и разряда TR в регистре TCON ( Таймеры 0 и 1) и специальные биты регистра TCON2 ( Таймер 2), форматы которых приведены на рис. 6.4.




А) Счетчик / таймер 0 в режиме 0




Б) Счетчик / таймер 0 в режиме 2




С) Счетчик / таймер 0 в режиме 3

Рис.6.3. Режимы работы счетчиков/таймеров

Регистр TCON (адрес 88h в пространстве DSEG)

| Адрес бита в пространстве BSEG | 8F | 8E | 8D | 8C | 8B | 8A | 89 | 88 |
|---|---|---|---|---|---|---|---|---|
| Назначение разрядов регистра | TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |

TCON.0 IT0 - управление типом входа ~INT0
IT0 = 1 - по спадающему фронту (динамический вход)
IT0 = 0 - статический вход
TCON.1 IEO - флаг запроса на прерывание ~INT0
при динамическом входе
TCON.2 IT1 - управление типом входа ~INT1
IT1 = 1 - по спадающему фронту (динамический вход)
IT1 = 0 - статический вход
TCON.3 IE1 - флаг запроса на прерывание ~INTI
при динамическом входе
TCON.4 TR0 - флаг программного управления CT0: 0 –останов счета / 1 - запуск
TCON.5 TF0 - флаг запроса прерывания по переполнению CT0
TCON.6 TR1 - флаг программного управления CT1: 0 –останов счета / 1 - запуск
TCON.7 TF1 - флаг запроса прерывания по переполнению CT1

Рис. 6.4. Формат регистра управления таймерами/счетчиками

Работа таймера/счетчика разрешается только, если взведен соответствующий разряд TR. При этом разряд GATE должен быть нулевым или на входе ~INT~ - низкий уровень. Для подачи внешних сигналов на таймеры/счетчики и логику их управления используются разряды порта P3 (см. рис.6.5).

Порт P3 (адрес B0 в пространстве DSEG)

| Адрес бита в пространстве BSEG | B7 | B6 | 8D | 8C | 8B | 8A | 89 | 88 |
|---|---|---|---|---|---|---|---|---|
| Спец. Назначение контактов порта | WR | RD | T1 | T0 | INT1 | INT0 | TxD | RxD |

P3.2 INT0 - вход запроса на прерывание 0
P3.3 INTI - вход запроса на прерывание 1
P3.4. T0 - счетный вход таймера/счетчика 0
P3.5. T1 - счетный вход таймера/счетчика 1

Рис. 6.5. Использование порта P3 для подключения внешних сигналов таймеров 0 и 1

ЗАДАНИЯ для выполнения лабораторной работы

Написать систему программ, которая выполняет следующие функции. Обработчик прерывания от таймера 0 инкрементирует содержимое ячейки ADRT и загружает начальное состояние в регистр TL0.Первая программа выполняет циклический анализ содержимого
ADRT и вызывает подпрограмму вывода массива кодов в порт. Программа располагается в CSEG, начиная с адреса ADR0. Программа должна состоять из двух

частей. Первая часть - инициализация системы, вторая - анализ содержимого ADRT и вызов подпрограммы.

Вторая программа - должна располагаться в памяти CSEG, начиная с адреса ADR1. Она должна при каждом обращении передавать в порт P1 массив кодов длиной N, расположенный в CSEG, начиная с адреса ADR2.

Начальный код в таймере и константа сравнения должны быть выбраны так, чтобы вызов подпрограммы выполнялся каждые Тмкс. Считаем, что частота OSC/12 = 1 МГц.

Вход в систему программ по команде LCALL ADR0, расположенной в CSEG по адресу 0000H.

Варианты заданий приведены в табл.1.

Массив кодов должен соответствовать массиву программы 2 из лабораторной работы номер 4.

Таблица 1. Варианты заданий

| Номер | ADRT | ADR0 | ADR1 | ADR2 | N | T |
|---|---|---|---|---|---|---|
| 01 | 21 | 614 | 714 | 431 | E | 500 |
| 02 | 22 | 52F | 62F | 560 | F | 550 |
| 03 | 23 | 43E | 53E | 621 | D | 600 |
| 04 | 24 | 34A | 44A | 711 | C | 650 |
| 05 | 25 | 255 | 355 | 121 | B | 700 |
| 06 | 26 | 366 | 266 | 236 | A | 750 |
| 07 | 27 | 277 | 177 | 345 | 6 | 800 |
| 08 | 28 | 68A | 78A | 454 | 7 | 850 |
| 09 | 29 | 79D | 69D | 568 | F | 900 |
| 10 | 2A | 3AF | 5AF | 677 | E | 950 |
| 11 | 2B | 5BA | 4BA | 781 | 5 | 1000 |
| 12 | 2C | 4C9 | 3C9 | 231 | 6 | 1050 |
| 13 | 2D | 1D8 | 2D8 | 3A1 | 7 | 1100 |
| 14 | 2E | 2E4 | 1E4 | 4B2 | 8 | 1150 |
| 15 | 2F | 3F5 | 7F5 | 5C2 | 8 | 1200 |

ПРИМЕР ВЫПОЛНЕНИЯ ЗАДАНИЯ

Написать программную систему, в которой подпрограмма вывода массива кодов вызывается каждые 1000 мкс.

Обработчик прерывания от таймера 0 инкрементирует содержимое ячейки ADRT=50.

Первая программа выполняет циклический анализ содержимого ADRT и вызывает подпрограмму вывода массива кодов в порт. Программа располагается в CSEG, начиная с адреса 0100H. Программа должна состоять из двух частей. Первая часть - инициализация системы, вторая - анализ содержимого ADRT и вызов подпрограммы.

Вторая программа должна располагаться в памяти CSEG, начиная с адреса 0200H. Она должна при каждом обращении передавать в порт P1 массив кодов длиной 6, расположенный в CSEG, начиная с адреса 0300H.

Выберем начальную константу для таймера таким образом, чтобы прерывание от него поступало каждые 100 мкс. Так как максимальное число в таймере 255,

следовательно начальное его значение должно быть 255-100 = 155 = 9BH. Для задания интервала 1000 мкс в ячейке ADRT должен декрементироваться код 10 = 0AH.

```
/*Пример выполнения задания на языке программирования СИ51*/
/*программа пересылки кодов*/

#include <stdio.h>
#include <reg51.h>
#define tl 0x9b;
#define n 0x06
#define adr1 0x0100
#define adr2 0x0200
at 0x0300 unsigned char code  memr[]={0xfd,0x02,0x03,0x04,0x05,0xfe};/*массив
пересылаемых кодов *.
at  0x50 unsigned char adrt; //переменная ADRT находится по адресу 0x50
char i;
#define const1 0x7f
#define const2 0x7fff
unsigned char s[4],ind1=0,ind2=0,k=0,ind;
/* ///////// Функция передачи кодов …….*/
/*  функция запускается на исполнение если содержимое ADRT равно 10*/
/*  Во время выполнения функции массив MEMR , находящийся в памяти CODE
выводится на UART и на цифровую индикацию в десятичной системе счисления.
Старшие 2 цифры в порт P2 и младшие две цифры в порт P1  */
at adr2 void per_code()
{

char i;
long p;
TR0=0;// запретить счет таймеру CT0
for(i=0; i<=(n-1);i++)
{
printf("memr=%d \n ",memr[i]);//вывод на UART
sprintf(s,"%d",memr[i]);
ind1=0;
ind2=0;
m1:
if(s[k]!=0)
{
ind=(ind1>>4)&0x0f;

ind2|=ind;
ind1=ind1<<4;
ind1=ind1|(s[k]&0x0f);
k++;
goto m1;
}
else
{
k=0;
/*  вывод на цифровую индикацию*/
P2=ind2;
```

```
P1=ind1;
ind=0;
}
for(p=0;p<=const2;p++);
}
}
/* ------------------------Функция обработчик прерывания от таймера CT0---------- */
 void timer0()interrupt 1
{
adrt++;// наращивание содержимого ADRT
}
/*  Главная функция ------------------------------------------------------------    */
at adr1 void main()
{
IE|=0x82;/*Разрешить общие прерывания и прерывание от  c/t0*/
TL0=tl;// установка таймера в исходное состояние
TMOD|=0x02;/* таймер CT0 должен работать в режиме 2 */
TCON|=0x10; /* разрешаем счет таймеру CT0 */
mz1:
adrt=adrt ;
if(adrt<=0x0a)goto mz1;else adrt=0;
per_code();// запуск функции per_code() если содержимое ADRT = 10
while(1);
}
```

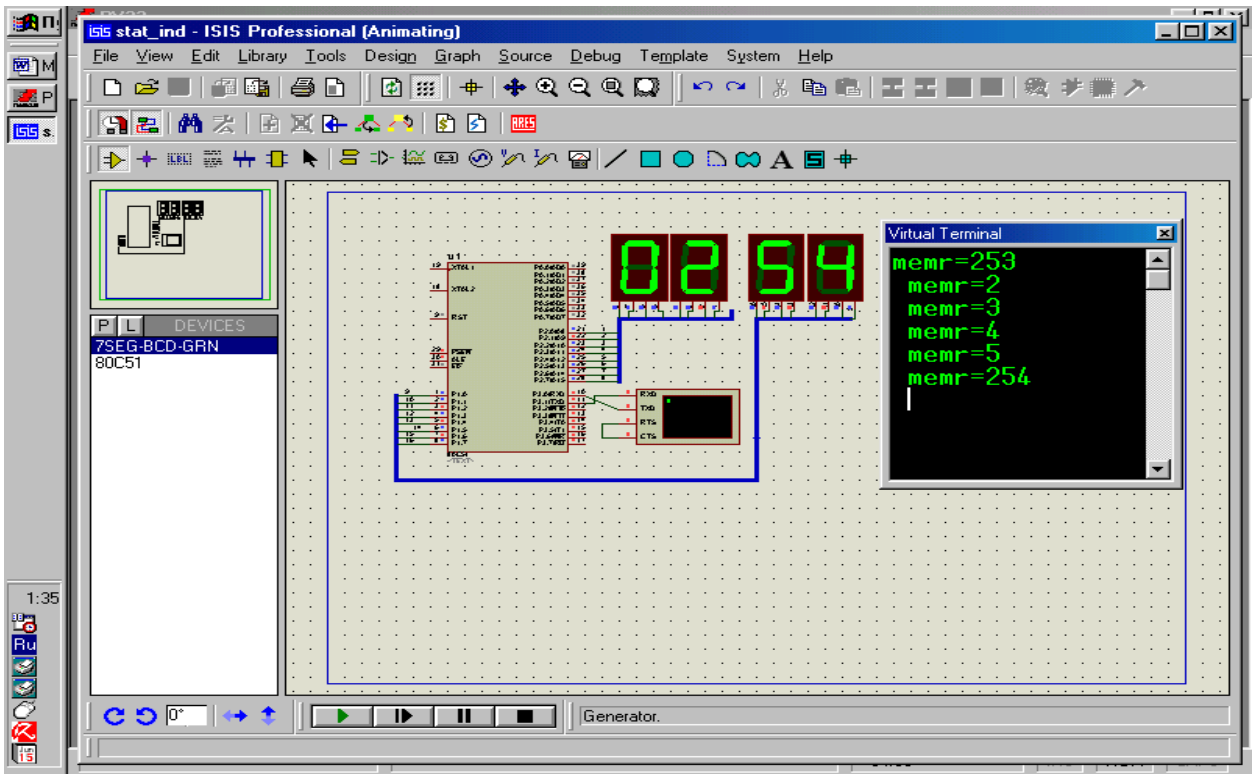



Рисунок 1. Результаты работы программы в PROTEUS

**Содержание отчета**

Отчёт о лабораторной работе должен содержать:

Титульный лист.

Цель и задачи работы.

исходные данные, в соответствии с вариантом.

Текст программы на языке Си с подробными комментариями.

Выводы по работе.

Контрольные вопросы:

Перечислите регистры и разряды управления таймером С/Т0.

Перечислите регистры и разряды управления таймером С/Т1.

Запишите программу на СИ инициализации С/Т0: константа установки 0xf00c; режим1; разрешить ПРВ и запустить счет.

Запишите программу инициализации таймера С/Т0 на выдержку времени 10мС.

.Запишите программу инициализации таймера С/Т1 на выдержку времени 80 мкС.

Запишите программу инициализации таймера С/Т0 и adrt на выдержку времени 8 секунд.

## Лабораторная работа № 7

Тема: Последовательный интерфейс персонального компьютера (РС)

Цель работы: Изучение принципа работы COM-портов РС;

Тестирование COM- порта РС;

Тестирование COM-портов двух РС;

### Теоретические сведения

Последовательная передача данных

При последовательной передаче данных *биты,* образующие коды символов, пересылаются по очереди один за другим по единственному проводнику, возвратным проводом является земля. Персональные компьютеры (РС) часто имеют два коммуникационных порта (обычно называемых COM-портами). Именно эти порты используются для асинхронной последовательной передачи данных путем посылки и получения ASCII кода. Этот протокол передачи данных известен как стандарт RS232C.

На рисунке 1 показано, как выглядит передаваемый сигнал при пересылке символа R. Каждому символу предшествует стартовый бит (логический 0), который передается перед стоповым битом. Длина стопового бита может составлять 1, 1.5 или 2 обычных бита.

Число изменений состояния канала передачи данных в секунду называется скоростью передачи в бодах. В системе с двумя возможными состояниями скорость передачи в бодах равна числу битов, передаваемых за секунду (бит/с).

При пересылке данных между компьютерами (или между компьютером и периферийным устройством) оба компьютера должны работать в одном и том же режиме так, чтобы во время передачи какого- либо символа тактовый генератор на принимающем компьютере работал на той же частоте, что и на передающем компьютере. Такая система является асинхронной, так как минимальное число символов, передаваемых в секунду, определяется скоростью работы передатчика, т. е. передатчик может сделать паузу любой длительности между символами. Во время любых пауз линия находится в состоянии логической единицы, поэтому естественно в качестве стартового бита при передаче стартового бита при передаче использовать логический ноль.

Один из наиболее распространенных способов пересылки битов в системе асинхронной передачи данных показан на рисунке 1. Здесь логический 0 представлен уровнем напряжения, превышающем +3В(обычно 10В), а логическая 1 - уровнем напряжения, меньшим –3В(обычно – 10В). Нулевое напряжение указывает на неработающую систему или на разрыв на линии.

Режим работы ИС приемника передатчика может быть запрограммирован путем использования:

1) пакета программ;
2) режима командной строки MS-DOS (рисунок 2);
3) некоторой команды в СИ программе.

Номер бита старт 0 1 2 3 4 5 6 бит чет .Ст

Битовая посл..0 0 1 0 0 1 0 1 1 1

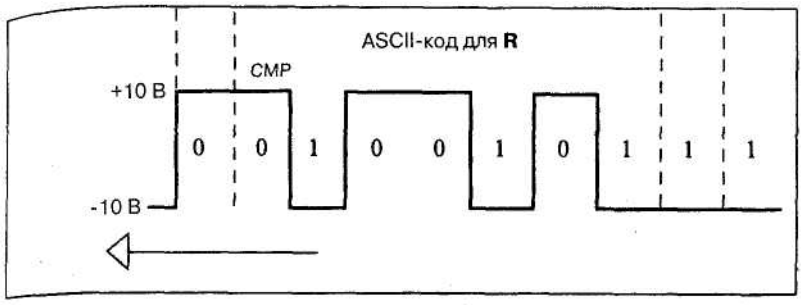



Рисунок1.-Сигнал стандартаRS232при передаче символа «R».


Пример Напечатайте>mode com1:9600,п,8,1,P.

Рисунок 2 - Режим командной строки MS-DOS.

Аппаратные средства последовательного интерфейса

Специальные ИС, называемые *универсальными асинхронными приемопередатчиками* - UART, преобразуют данные из параллельного (TTL) формата, получаемого из шины данных микропроцессора, в последовательный (RS232). Эти устройства можно рассматривать как устройства с двумя внутренними секциями: преобразователем параллельного формата в последовательный, называемый передатчиком, и преобразователем последовательного формата в параллельный, называемый приемником. Передатчик представляет бит четности, стартовый и стоповый биты. Приемник проверяет формат и четность полученных данных. Приемник и передатчик могут работать одновременно, такой способ передачи называют полным дуплексом (передача только в одном направлении называется симплексной связью, передача в обоих направлениях, называется полудуплексной связью).

Скорости передачи и приема данных контролируются генератором тактовых импульсов, работающим на частоте, кратной скорости передачи данных. Внутренний регистр состояния, который хранит флажки состояний приема передачи и ошибок, может быть прочитан программным способом.

Стандартным соединителем является 25-штырьковый разъем д типа (штепсельная часть). Однако в некоторых случаях применяется 9-штырьковый разъем, который пригоден для соединения наиболее часто используемых управляющих линий. Назначение выводов 25-штырькового разьема показано на рисунке 3.


Рисунок 3.- Назначение контактов Сом порта.

**Порядок выполнения лабораторной работы**

Вначале работы следует проверить адрес последовательного порта используемого компьютера. Типичные адреса COM –портов ПК для пересылки и получения данных приведены ниже:

1 последовательный порт: 3F8 hex

2 последовательный порт: 2F8 hex

Задание 1 <Тестирование петлевой конфигурации>

Необходимо отослать некий символ с вывода «Передаваемые данные» (вывод 2) и получить на выводе «Принимаемые данные» (вывод 3) того же порта. Для этого теста нужен лишь один компьютер. Вывод 2 «Передаваемые данные» соединен с выводом 3 «Принимаемые данные» того же самого порта.

Структурная схема программного обеспечения к этому заданию показана на рисунке 4.

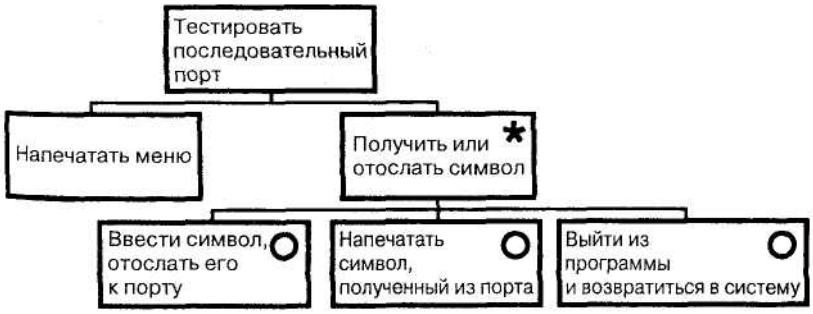



Рисунок 4.- Структурная схема ПО.

```
 /*Serial.c Тестирование последовательного порта*/
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
int Address=0x3f8; /*Адрес порта COM1,заменить на 0x2f8 для COM2*/
void main()
{
char menu='x';
system("cls"); /*Очистка экрана*/
printf("\n Передача данных через последовательный порт");
printf("\n R для получения данных");
printf("\n S для отсылки данных");
printf("\n Q для выхода из программы\n");
while(1)
{
menu=getch();
switch(menu)
{
case 'S': printf("\n Введите символ с клавиатуры для его пересылки");
          outp(Address,getch());
          printf("\nS,R,Q");
          break;
case 'R': printf("\n Полученный символ %c",inp(Address));
          printf("\n S,R,Q");
          break;
case 'Q': exit(1);
}
fflush(stdin);
}
}
```

*Тестирование*

Соедините выводы 2 и 3 COM-порта

Нажмите на клавиатуре S, а затем введите символ.

1. Нажмите R: тот же самый символ должен быть получен на экране.

: 4. Отсоедините выводы 2 и 3. При нажатии R на экране должен быть получен другой символ.

Задание2 «Связь между двумя компьютерами»

Цель данного упражнения - пересылка символа между двумя компьютерами. Таким образом, для выполнения упражнения нужен второй компьютер. Соедините COM- порты двух компьютеров нуль-модемным кабелем.

1. Убедитесь в том, что оба компьютера работают в одном и том же режиме.
2. Запустите программу к предыдущему заданию на обоих компьютерах.
3. Отошлите символ с одного компьютера и получите его на другом. Система должна одинаково работать в обоих направлениях (полу дуплекс).
4. Используйте режим командной строки MS-DOS для изменения режима работы на обоих компьютерах. Выяснить что произойдет в том случае когда компьютеры будут работать с разными скоростями передачи данных или будут передавать символы с разным количеством разрядов данных.

Содержание отчета:

Отчет должен содержать:

1.Текст программы тестирования портов;

2. Ответы на поставленные вопросы.

ВОПРОСЫ

1. Почему каждому символу предшествует стартовый бит?
2. Приведите два важных преимущества использования напряжения противоположной полярности для передачи логических состояний, а не обычных TTL-уровней 5 и 0в.
3. Какое логическое состояние передается между символами?
4. За какое время будет передан 7-разрядный символ с дополнительным битом четности при скорости передачи 1000бод?
5. Каким образом принимающее устройство узнает о том, что передающее устройство не подключено, если используется трехпроводная система, в которой соединены выводы 2, 3 и 7?

## Лабораторная работа №8

## ИССЛЕДОВАНИЕ РАБОТЫ УАПП.

Цель работы: изучение организации системы ввода/вывода микроконтроллера AT89C52 при помощи универсального асинхронного приемо-передатчика.

### 1. ОРГАНИЗАЦИЯ СИСТЕМЫ ПОСЛЕДОВАТЕЛЬНОГО ВВОДА-ВЫВОДА

В состав MK52 входит универсальный асинхронный приемо-передатчик (последовательный порт), обеспечивающий прием и/или передачу информации последовательным кодом (младшими байтами вперед).

В состав порта входят:

- принимающий регистр,
- передающий регистр ,
- буферный регистр ( SBUF, адрес 99H),
- регистр управления - состояния ( SCON, адрес 98H ), формат которого приведен на рис.1.

Регистр SCON (адрес 98 в пространстве DSEG)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Мнемоническое обозначение разрядов регистра | SM0 | SM1 | SM2 | REN | TB8 | RB8 | TI | RI |

Рис. 1. Формат регистра SCON

Биты регистра имеют следующее функциональное назначение :

SM1 SM0 - задают режим работы УАПП
0 0 режим 0 - сдвигающий регистр.
0 1 режим 1 - 8 бит, изменяемая скорость.
1 0 режим 2 - 9 бит , фиксированная скорость.
1 1 режим 3 - 9 бит, изменяемая скорость.

SM2 - бит управления режимом, устанавливается программно для запрета приема кадров с девятым битом равным нулю;

REN - бит разрешения /запрета приема, устанавливается программно для разрешения приема данных;

TB8 - девятый бит передатчика в режимах 2 и 3 ;

RB8 - девятый бит приемника в режимах 2 и 3 ;

TI - флаг прерывания передатчика, устанавливается аппаратно при окончании передачи кадра, сбрасывается программно;

RI - флаг прерывания приемника, устанавливается аппаратно при окончании приема кадров, сбрасывается программно.

Режим работы УАПП должен задаваться программно, заданием двух старших бит регистра SCON.

**1.1. Работа УАПП в режиме 0.**

Информация передается и принимается через внешний вывод входа приемника RxD (P3.0). Принимаются или передаются только 8 бит данных. Через внешний вывод выхода передатчика TxD (P3.1) выдаются импульсы сдвига с частотой OSC/12 , которые сопровождают каждый бит.

Передача начинается после выполнения любой команды, которая загружает код в буферный регистр SBUF. Автоматически содержимое SBUF переписывается в сдвигающий регистр передатчика, и в каждом машинном цикле содержимое сдвигающего регистра сдвигается вправо и подается на выход RxD ( рис.2.). В освобождающиеся биты записываются 0. Когда сдвигающий регистр весь заполнится нулями - передача закончена, устанавливается флаг TI.

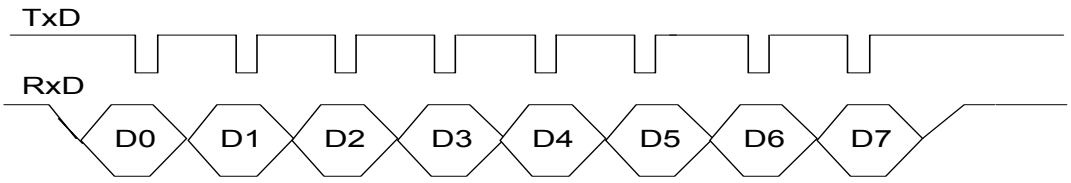



Рис. 2. Формат передачи данных в режиме 0 .

Прием начинается при условии REN=1 и RI=0. Установка REN и сброс RI должны выполняться программно. Как только условие начала приема выполнено, начинается выдача синхросигналов на выход TxD и считывание битов с входа RxD в моменты каждого машинного цикла. После 8 машинных циклов, в начале 9-го, содержимое сдвигающего регистра приемник переписывается в SBUF, аппаратно сбрасывается REN и устанавливается RI.

**1.2. Работа УАПП в режиме 1.**

В этом режиме выполняется асинхронный прием/передача 11-разрядного кадра: стартовый бит (1 разряд), байт данных (8 разрядов) и стоповый бит (1 разряд). Скорость приема/передачи задается таймером.

Данные передаются через TxD и принимаются через RxD. Передача инициируется любой командой, которая записывает в SBUF байт для передачи. При этом в девятый (стоповый) бит передатчика записывается 1. В момент времени SIPI следующей команды на выходе TxD появляется стартовый бит (низкий уровень) , а затем и биты данных. Последним выдается стоповый бит, всегда равный 1. Во время его формирования взводится триггер TI - запрос на прерывание по завершении передачи. Длительность выдачи каждого бита равна 16 периодам работы внутреннего счетчика.

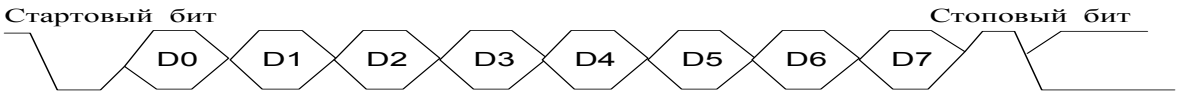



Рис. 3. Кадр передачи данных в режиме 1

Прием начинается при обнаружении на входе RxD перехода сигнала из 1 в 0. При обнаружении перехода в сдвигающий регистр приемника загружается код 1FFh, а внутренний счетчик тактов перезапускается. В состоянии 7,8,9 внутреннего счетчика тактов проверяется уровень сигнала на входе RxD. Если в двух случаях их трех уровень не 0 , считается, что пришла помеха и снова начинается ожидание стартового бита - перехода из 1 в 0 .

Если в эти же моменты в двух случаях из трех зафиксирован уровень 0, считается что пришел стартовый бит. Стартовый бит сдвигается в регистр приемника и продолжается прием остальных бит кадра. При приеме каждого бита происходит фиксация уровня входного сигнала в состояниях 7,8,9 внутреннего счетчика. Значение сигнала принимается равным полученному не менее 2 раз из трех.

Принятый байт загружается в буферный регистр SBUF только если R1=0 и, либо SM2 = 0, либо принятый стоп-бит равен 1. При этом в RB8 загружается 1. В противном случае информация теряется и приемник начинает поиск стартового бита.

В режиме 1 ( как в режиме 3 ) частота передачи приема данных УАПП определяется таймером CT1, вернее частотой его переполнения , и состоянием старшего разряда регистра PCON – SMOD (адрес регистра PCON - 87H ). Частота приема/передачи равна частоте переполнения CT1 деленное на 32, если SMOD = 0. Если SMOD установлен в 1 , частота приема/передачи равна частоте переполнения CT1 деленной на

16 т.е. в два раза выше. СТ1 при синхронизации УАПП может работать в любом режиме, однако запрос на прерывание TF1 по переполнению СТ1 должен быть заблокирован.

**1.3. Работа УАПП в режиме 2.**

В этом режиме происходит асинхронный прием/передача 12-разрядного кадра : стартовый бит (1 разряд), байт данных (8 разрядов), программируемый бит данных (1 разряд) и стоповый бит (1 разряд). Скотрость приема/передачи задается программно и может быть равна 1/32 или 1/64 частоты OSC, в зависимости от состояния разряда SCON регистра PCON.

Режим 2 отличается от режима 1 только наличием дополнительного 9 программируемого бита. При передаче стоповым битом вставляется 9 бит, равный содержимому TB8 регистра SCON.

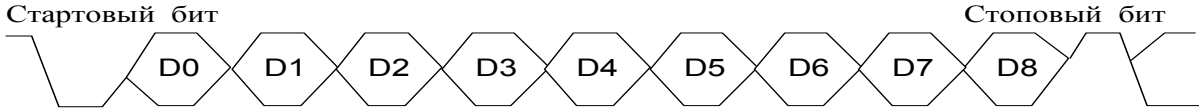



Рис. 4. Кадр передачи данных в режиме 2

При приеме 9 бит записывается в RB8 регистра SCON. Причем и перезапись байта из регистра сдвига в буферный регистр SBUN и перезапись 9 бита в RB8 происходит только в том случае, если при приеме стопового бита выполняется два условия: бит RI = 0, и либо SM2= 0, либо значение принятого 9 бита равно 1.

**1.4. Работа УАПП в режиме 3.**

Режим предусматривает выполнение асинхронного приема/передачи 12-разрядного кадра. Работа аналогично режиму 2, но скорость передачи задается таймером, как в режиме 1.

**1.5. Скорость приема/передачи.**

В режиме 0 частота передачи зависит только от резонансной частота кварцевого резонатора

$$f0=fрез/12.$$

За один машинный цикл последовательный порт передает один бит информации.

В режимах 1, 2 и 3 скорость приема/передачи зависит от значения управляющего бита SMOD.

В режиме 2 частота передачи определяется выражением:

$$f2=(2^{SMOD}/64)fрез.$$

Иными словами, при SMOD=0 частота передачи равна (1/64)/fрез , а при SMOD = 1 равна (1/32)/fрез.

В режимах 1 и 3 в формировании частоты передачи кроме управляющего бита SMOD принимает участие таймер 1. При этом частота передачи зависит от частоты переполнения (OVT1) и определяется следующим образом:

$$f1,3=(2^{SMOD}/32)fovt1.$$

Прерывание от таймера 1 в этом случае должно быть заблокировано. Сам Т/С1 может работать таймер, и как счетчик событий в любом из трех режимов. Однако наиболее удобно использовать режим таймера с автоперезагрузкой (старшая тетрада TMOD=0010B). При этом частота передачи определяется выражением:

$$f1,3=(2^{SMOD}/32)*(fрез/12)*(256 - (TH1)).$$

| Частота приема/передачи (BAUD RATE) | Частота резонатора, МГц | SMOD | Таймер/счетчик | | |
|---|---|---|---|---|---|
| | | | C/T | Режим (MODE) | Перезагружаемое число |
| Режим 0, макс: 1 МГц | 12 | x | x | x | x |
| Режим 2, макс: 375 кГц | 12 | 1 | x | x | x |
| Режимы 1,3: 62,5 кГц | 12 | 1 | 0 | 2 | 0FFH |
| 19,2 кГц | 11,059 | 1 | 0 | 2 | 0FDH |
| 9,6 кГц | 11,059 | 0 | 0 | 2 | 0FDH |
| 4,8 кГц | 11,059 | 0 | 0 | 2 | 0FAH |
| 2,4 кГц | 11,059 | 0 | 0 | 2 | 0F4H |
| 1,2 кГц | 11,059 | 0 | 0 | 2 | 0E8H |
| 110 Гц | 6 | 0 | 0 | 2 | 72H |
| 110 Гц | 12 | 0 | 0 | 1 | 0FEEBH |

Рис. 5. Настройка таймера 1 для управления частотой работы UART.

## 2. ЗАДАНИЕ ПО ЛАБОРАТОРНОЙ РАБОТЕ

```
/*Программа для стенда EV8031/AVR(V3.2)*/
/*Программа обслуживания клавиатуры
и динамической индикации на HL2.
С клавиатуры получаем коды клавиш от 0 до 9
и коды 16сс A,B,C,D,E,F.
Код А получаем двумя нажатиями кнопок вначале нажать и отпустить
 кнопку # а затем например 1(нажать и отпустить) получим символ А
 итого # 1 -A;#4 -B;#7 -C;#3 -D;#8 -E;#9 -F.
При нажатии кнопки, например 3, программа выдает на индикацию
 инверсию кода 3 цифрового индикатора (0xb0)- 0x4f.
Программа выполняет обмен символом в 16-ричной сс с персональным компьютером
код символа набираемый клавиатурой  динамически выдается  на циф. индикатор HL2. */
/*************************************************************************
**/
#include<reg51.h>
#include<stdio.h>
#define uw 0x80
at 0x8000 unsigned char xdata a55;
at 0x8001 unsigned char xdata b55;
at 0x8002 unsigned char xdata c55;
at 0x8003 unsigned char xdata rus55;
at 0x9003 unsigned char xdata s3s12;
at 0x9005 unsigned char xdata s2s11;
at 0x9006 unsigned char xdata s1s10;
unsigned char data buf=0x71;
#define rel96 0xfc;
unsigned char code consti[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,
```

```
0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x8e};
unsigned char simvol;
char k;
sfr pcon=0x87;
unsigned char n;
unsigned char klav;
unsigned char klav1;
unsigned char sim;
unsigned char sim1;//функция инициализации таймера и системы
void inic()
{
ET1=0;
ES=1;
EA=1;
TL1=rel96;
TH1=rel96;
TMOD=0x20;
TCON=0x40;
SCON=0x50;
rus55=uw;
pcon=0x80;
return;
}
void indik(unsigned char sim,unsigned char sim1)
{
b55=sim;
a55=0xfe;
SCON=0x40;
SBUF=sim1;
return;
}
//обработчик прерывания TI/RI
void getch()interrupt 4
{
unsigned char cods;
unsigned char cods1;
if(TI==1){TI=0;REN=1;goto rin3;};
if(RI==1)goto get;
TI=0;
REN=1;
goto rin3;
get:
EA=0;
RI=0;
cods=SBUF;
sim1=cods;
/*преобразование принятого кода ASCII в 16-ричный код
переход к верхнему регистру*/
cods1=cods|0x20;
cods1=cods1&0xf0;
if(cods1==0x30){cods=cods&0x0f;simvol=consti[cods];sim=~(simvol); indik(sim,sim1);
goto rin3;}
```

```
if(cods1>=0x67)goto rin3;
cods=cods&0x0f;
cods=cods+0x09;
simvol=consti[cods];
sim=~(simvol);
indik(sim,sim1);
rin3:
EA=1;
}
void delay()
{
char i,j;
for(i=0x7f;i>=0;i--){
for(j=0x7f;j>=0;j--);}
return;
}
void main ()
{
inic();
m1: klav=s3s12;
klav1=klav|0xf0;
if(klav1==0xff)goto m2; else delay();
klav=s3s12;
klav1=klav|0xf0;
if(klav1==0xf7)goto k0;
if(klav1==0xfe){sim=0x4f;sim1=0x33; goto z1;}//Символ "3"
else if(klav1==0xfd){sim=0x7d;sim1=0x36;goto z1;}//Символ"6"
else if(klav1==0xfb){sim=0x6f;sim1=0x39;goto z1;}//Символ "9"
else goto m1;
z1:indik(sim,sim1);
// Определение факта отпускания клавиши
while(klav1!=0xff){klav=s3s12;klav1=klav|0xf0;}
delay();
goto m1;
m2:
klav=s2s11;
klav1=klav|0xf0;
if(klav1==0xff)goto m3; else delay();
if(klav1==0xf7){sim=0x3f;sim1=0x30;goto z2;}//Символ "0"
if(klav1==0xfe){sim=0x5b;sim1=0x32;goto z2;}//Символ"2"
if(klav1==0xfd){sim=0x6d;sim1=0x35;goto z2;}//символ"5"
if(klav1==0xfb){sim=0x7f;sim1=0x38;goto  z2;}//Символ"8"
else goto m1;
z2:
indik(sim,sim1);
//Определение факта отпускания клавиши
while(klav1!=0xff){klav=s2s11;klav1=klav|0xf0;}
delay();
goto m1;
m3:
klav=s1s10;
klav1=klav|0xf0;
```

```
if(klav1==0xff)goto m1; else delay();
if(klav1==0xfe){sim=0x06;sim1=0x31;goto z3;}//Символ"1"
if(klav1==0xfd){sim=0x66;sim1=0x34;goto z3;}//Символ"4"
if(klav1==0xfb){sim=0x07;sim1=0x37;goto z3;}//Символ"7"
else goto m1;
z3:
indik(sim,sim1);
//Определение факта отпускания клавиши
while(klav1!=0xff){klav=s1s10;klav1=klav|0x0f;}
delay();
goto m1;
//**********************************************************************
*****
//Была нажата клавиша реш.'#'
k0:
while(klav1!=0xff){klav=s3s12;klav1=klav|0xf0;}
delay();
k1:
klav=s3s12;
klav1=klav|0xf0;
if(klav1==0xff)goto am1; else delay();
klav=s3s12;
klav1=klav|0xf0;
if(klav1==0xfe){sim=0x5e;sim1=0x44;goto aut1;}//Символ"Д"
if(klav1==0xfd){sim=0x79;sim1=0x45;goto aut1;}//Символ"E"
if(klav1==0xfb){sim=0x71;sim1=0x46;goto aut1;}//Символ "F"
else goto k1;
aut1:
indik(sim,sim1);
//Определение факта отпускания клавиши
while(klav1!=0xff){klav=s3s12;klav1=klav|0xf0;}
delay();
goto m1;
am1:
klav=s1s10;
klav1=klav|0xf0;
if(klav1==0xff)goto k1; else delay();
klav=s1s10;
klav1=klav|0xf0;
if(klav1==0xfe){sim=0x77;sim1=0x41;goto aut2;}//символ"A"
if(klav1==0xfd){sim=0x7c;sim1=0x42;goto aut2;}//Символ"B"
if(klav1==0xfb){sim=0x39;sim1=0x43;goto aut2;}//Символ"C"
else goto k1;
aut2:
indik(sim,sim1);
//Определение факта отпускания клавиши
while(klav1!=0xff){klav=s1s10;klav1=klav|0xf0;}
//Устранение дребезга контактов
delay();
goto m1;
}
```

```
 /*Serial.c Тестирование последовательного порта*/
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
int Address=0x3f8; /*Адрес порта COM1,заменить на 0x2f8 для COM2*/
void main()
{
char menu='x';
system("cls"); /*Очистка экрана*/
printf("\n Передача данных через последовательный порт");
printf("\n R для получения данных");
printf("\n S для отсылки данных");
printf("\n Q для выхода из программы\n");
while(1)
{
menu=getch();
switch(menu)
{
case 'S': printf("\n Введите символ с клавиатуры для его пересылки");
          outp(Address,getch());
          printf("\nS,R,Q");
          break;
case 'R': printf("\n Полученный символ %c",inp(Address));
          printf("\n S,R,Q");
          break;
case 'Q': exit(1);
}
fflush(stdin);
}
}
```

## Лабораторная работа №8_1

Тема: Последовательный интерфейс персонального компьютера (PC)
Цель работы: Изучение принципа работы COM-портов PC;
Тестирование COM- порта PC;
Тестирование COM-портов двух PC;

### Теоретические сведения

Последовательная передача данных

При последовательной передаче данных *биты,* образующие коды символов, пересылаются по очереди один за другим по единственному проводнику, возвратным проводом является земля. Персональные компьютеры (PC) часто имеют два коммуникационных порта (обычно называемых COM-портами). Именно эти порты используются для асинхронной последовательной передачи данных путем посылки и получения ASCII кода. Этот протокол передачи данных известен как стандарт RS232C.

На рисунке 1 показано, как выглядит передаваемый сигнал при пересылке символа R. Каждому символу предшествует **стартовый** бит (логический 0), который передается перед стоповым битом. Длина стопового бита может составлять 1, 1.5 или 2 обычных бита.

Число изменений состояния канала передачи данных в секунду называется скоростью передачи **в бодах.** В системе с двумя возможными состояниями скорость передачи в бодах равна числу битов, передаваемых за секунду (бит/с).

При пересылке данных между компьютерами (или между компьютером и периферийным устройством) оба компьютера должны работать в одном и том же режиме так, чтобы во время передачи какого- либо символа тактовый генератор на принимающем компьютере работал на той же частоте, что и на передающем компьютере. Такая система является асинхронной, так как минимальное число символов, передаваемых в секунду, определяется скоростью работы передатчика, т. е. передатчик может сделать паузу любой длительности между символами. Во время любых пауз линия находится в состоянии логической единицы, поэтому естественно в качестве стартового бита при передаче стартового бита при передаче использовать логический ноль.

Один из наиболее распространенных способов пересылки битов в системе асинхронной передачи данных показан на рисунке 1. Здесь логический 0 представлен уровнем напряжения, превышающем +3В(обычно 10В), а логическая 1 - уровнем напряжения, меньшим –3В(обычно – 10В). Нулевое напряжение указывает на неработающую систему или на разрыв на линии.

Режим работы ИС приемника передатчика может быть запрограммирован путем использования:

1) пакета программ;
2) **режима** командной строки MS-DOS (рисунок 2);
3) некоторой команды в СИ программе.

^'

Номер бита старт 0 1 2 3 4 5 6 бит чет .Стоповый

Битовая посл..0 __ 0 1 0 0 1 0 1 1 1

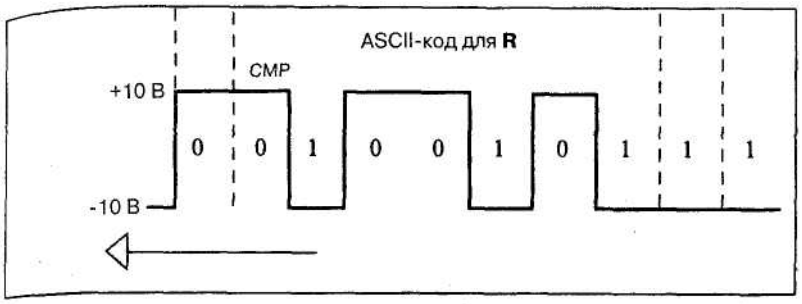



Рисунок1.-Сигнал стандартаRS232при передаче символа «R».


**Пример** Напечатайте>mode com1:9600,п,8,1,P.

**Рисунок 2 -** Режим командной строки MS-DOS.

Специальные ИС, называемые *универсальными асинхронными приемопередатчиками* - UART, преобразуют данные из параллельного (TTL) формата, получаемого из шины данных микропроцессора, в последовательный (RS232). Эти устройства можно рассматривать как устройства с двумя внутренними секциями: преобразователем параллельного формата в последовательный, называемый передатчиком, и преобразователем последовательного формата в параллельный, называемый приемником. Передатчик представляет бит четности, стартовый и стоповый биты. Приемник проверяет формат и четность полученных данных. Приемник и передатчик могут работать одновременно, такой способ передачи называют **полным дуплексом** (передача только в одном направлении называется симплексной связью, передача в обоих направлениях, называется полудуплексной связью).

Скорости передачи и приема данных контролируются генератором тактовых импульсов, работающим на частоте, кратной скорости передачи данных. Внутренний регистр состояния, который хранит флажки состояний приема передачи и ошибок, может быть прочитан программным способом.

Стандартным соединителем является 25-штырьковый разъем д типа (штепсельная часть). Однако в некоторых случаях применяется 9-штырьковый разъем, который пригоден для соединения наиболее часто используемых управляющих линий. Назначение выводов 25-штырькового разъема показано на рисунке 3.


Рисунок 3.- Назначение контактов Сом порта.

**Порядок выполнения лабораторной работы**

Вначале работы следует проверить адрес последовательного порта используемого компьютера. Типичные адреса COM –портов ПК для пересылки и получения данных приведены ниже:

1 последовательный порт: 3F8 hex

2 последовательный порт: 2F8 hex

Задание 1 <Тестирование петлевой конфигурации>

Необходимо отослать некий символ с вывода «Передаваемые данные» (вывод 2) и получить на выводе «Принимаемые данные» (вывод 3) того же порта. Для этого теста

нужен лишь один компьютер. Вывод 2 «Передаваемые данные» соединен с выводом 3 «Принимаемые данные» того же самого порта.

Структурная схема программного обеспечения к этому заданию показана на рисунке 4.

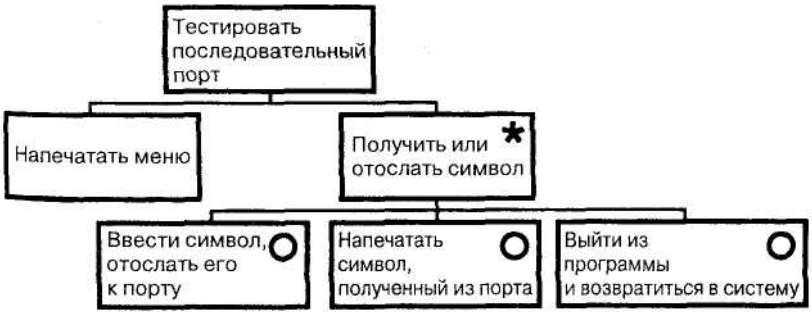



Рисунок 4.- Структурная схема ПО.

```
 /*Serial.c Тестирование последовательного порта*/
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
int Address=0x3f8; /*Адрес порта COM1,заменить на 0x2f8 для COM2*/
void main()
{
char menu='x';
system("cls"); /*Очистка экрана*/
printf("\n Передача данных через последовательный порт");
printf("\n R для получения данных");
printf("\n S для отсылки данных");
printf("\n Q для выхода из программы\n");
while(1)
{
menu=getch();
switch(menu)
{
case 'S': printf("\n Введите символ с клавиатуры для его пересылки");
          outp(Address,getch());
          printf("\nS,R,Q");
          break;
case 'R': printf("\n Полученный символ %c",inp(Address));
          printf("\n S,R,Q");
          break;
case 'Q': exit(1);
}
fflush(stdin);
}
}
```

Тестирование

Соедините выводы 2 и 3 COM-порта

Нажмите на клавиатуре S, а затем введите символ.
Нажмите R: тот же самый символ должен быть получен на экране.
Отсоедините выводы 2 и 3. При нажатии R на экране должен быть получен другой символ.

Задание2 «Связь между двумя компьютерами»

Цель данного упражнения - пересылка символа между двумя компьютерами. Таким образом, для выполнения упражнения нужен второй компьютер. Соедините COM- порты двух компьютеров нуль-модемным кабелем.

1. Убедитесь в том, что оба компьютера работают в одном и том же режиме.
2. Запустите программу к предыдущему заданию на обоих компьютерах.
3. Отошлите символ с одного компьютера и получите его на другом. Система должна одинаково работать в обоих направлениях (полу дуплекс).
4. Используйте режим командной строки MS-DOS для изменения режима работы на обоих компьютерах. Выяснить что произойдет в том случае когда компьютеры будут работать с разными скоростями передачи данных или будут передавать символы с разным количеством разрядов данных.

```
/*  Программа управления UART микроконтроллера МК51  */
/* Программа выполняет функцию посимвольного ввода кодов на скорости
9600бод от персонального компьютера через интерфейс
RS- 232 стенда  EV8051/AVR и посимвольную передачу принятых кодов обратно в
персональный компьютер с целью индикации их на мониторе */
#include<reg51.h>
#include<stdio.h>
#include<stdlib.h>
#define rel96 0xfd //константа таймера для скорости 9600 б.
unsigned char cod;
/*  функция инициализации UART и таймера CT1  */
void inic()
{
IE=0x00; // выключаем систему прерываний микроконтроллера
TMOD=0x20; устанавливаем таймер CT1 в режим 2
TH1=rel96; //устанавливаем канстанты в таймер CT1 для скорости 9600
TL1=rel96; /* контроллер должен иметь кварцевый резонатор на 11.059мгц */
PCON&=0x7f;
SCON=0x50;//UART будет работать в реж.1. Разрешаем прием данных
TR1=1; Приемник готов к приему очередного символа
}
/*  Функция передачи принятого кода приемником UART на персональный
компьютер через RS- 232  */
void put()
{
SBUF=cod|0xf0;/* инициируем передачу загружая в SBUF принятый код  *.
m1:
if(TI!=1)goto m1; else TI=0;/* ожидаем окончание передачи символа и переходим на
режим приема (TI=0) следующего кода  */
```

```
}
/*  Функция приема очередного символа передаваемого персональным
компьютером через RS-232   */
void get()
{
m2:
if(RI!=1)goto m2;/*ожидаем установки флага RI в единицу – прием очередного
байта завершен*/
else{cod=SBUF;//забираем принятый код из SBUF в  переменную cod
RI=0;}//сбрасываем флаг RI
}
/*  Главная функция программы  */
void main()
{
inic();// инициализация UART
m3:
get();//вначале принимаем код от персонального компьютера
put();//после приема символа переходим на передачу его обратно
goto m3;
}
```

Содержание отчета:
Отчет должен содержать:
1. Текст программы тестирования портов;
2. Ответы на поставленные вопросы.

ВОПРОСЫ

1. Почему каждому символу предшествует стартовый бит?
2. Приведите два важных преимущества использования напряжения противоположной полярности для передачи логических состояний, а не обычных TTL-уровней 5 и 0в.
3. Какое  логическое  состояние передается между символами?
4. За какое время будет передан 7-разрядный символ с дополнительным битом четности при скорости передачи 1000бод?
5. Каким образом принимающее устройство узнает о том, что передающее устройство не подключено, если используется трехпроводная система, в которой соединены выводы 2, 3 и 7?

## Лабораторная работа  №8_2

Тема:Изучение принципов работы и управления последовательным портом

Цель работы:  приобретение навыков управления последовательным портом ввода-вывода МК51.

```
/*  Программа управления UART микроконтроллера МК51  */
```

**Пример программы обслуживания десятичной цифровой кавиатуры и индикации**

Цифры нажатой клавиши приведен ниже: (цифровой индикатор подключен к порту P1)

```
#include<reg51.h>
#include<stdio.h>
        #include<stdlib.h>
        #define rel96 0xfd //константа таймера для скорости 9600 б.
        unsigned char cod;
#define s1s7 0x6f //код сканирования кнопок 1,4,7,*
#define s2s0 0x5f //код сканирования кнопок 2,5,8,0
#define s3s9 0x3f //код сканирования кнопок 3,6,9,#
unsigned char klav;
unsigned char klav1;
unsigned char sim;
/* Функция задержки, необходимая для устранения дребезга контактов при нажатии и
отпускании клавишей */
/*  функция инициализации UART и таймера CT1  */
        void inic()
        {
        IE=0x00; // выключаем систему прерываний микроконтроллера
        TMOD=0x20; устанавливаем таймер CT1 в режим 2
        TH1=rel96; //устанавливаем канстанты в таймер CT1 для скорости 9600
        TL1=rel96; /* контроллер должен иметь кварцевый резонатор на 11.059мгц */
        PCON&=0x7f;
        SCON=0x50;//UART будет работать в реж.1. Разрешаем прием данных
        TR1=1; Приемник готов к приему очередного символа
        }
void delay()
{
char i,j;
for(i=0x7f;i>=0;i--){
for(j=0x7f;j>=0;j--);}
return;
}
        /*  Функция передачи принятого кода приемником UART на персональный
        компьютер через RS- 232  */
        void put()
        {
        SBUF=sim;/* инициируем передачу загружая в SBUF принятый код  *.
        M4:
        if(TI!=1)goto m4; else TI=0;/* ожидаем окончание передачи символа и переходим на
        режим приема (TI=0) следующего кода  */
}
        /*  Функция приема очередного символа передаваемого вторым МК через RS-232
        */
        void get()
        {
        m5:
```

```
        if(RI!=1)goto m5;/*ожидаем установки флага RI в единицу – прием очередного
        байта завершен*/
        else{cod=SBUF;//забираем принятый код из SBUF в  переменную cod
        RI=0;}//сбрасываем флаг RI
        }
void main ()
{
void inic();
m1: P2=s1s7;//Сканирование кнопок 147*
P2|=0x0f;
klav=P2;
klav1=klav|0xf0;
if(klav1==0xff)goto m2; else delay();
klav=P2;
klav1=klav|0xf0;
if(klav1==0xfe){sim=0x01; goto z1;}//Символ "1"
else if(klav1==0xfd){sim=0x04;goto z1;}//Символ"4"
else if(klav1==0xfb){sim=0x07;goto z1;}//Символ "7"
else goto m1;
z1:
// Определение факта отпускания клавиши
while(klav1!=0xff){P2|=0x0f; klav=P2;klav1=klav|0xf0;}
delay();
        /*  Функция передачи  кода sim на UART МК через RS- 232  */
        put();
        /*  Функция приема очередного символа передаваемого вторым МК через RS-232
        */

        get();
        P1=cod;
        goto m1;
m2:
P2=s2s0;сканирование кнопок 2580
P2|=0x0f;
klav=P2;
klav1=klav|0xf0;
if(klav1==0xff)goto m3; else delay();
if(klav1==0xfe){sim=0x02;goto z1;}//Символ "2"
if(klav1==0xfd){sim=0x05;goto z1;}//Символ"5"
if(klav1==0xfb){sim=0x08;goto z1;}//символ"8"
if(klav1==0xf7){sim=0x00;goto  z1;}//Символ"0"
else goto m1;
m3:
P2=s3s9;
P2|=0x0F;
```

```
klav=P2;
klav1=klav|0xf0;//сканирование кнопок 369#
if(klav1==0xff)goto m1; else delay();
if(klav1==0xfe){sim=0x03;goto z1;}//Символ"3"
if(klav1==0xfd){sim=0x06;goto z1;}//Символ"6"
if(klav1==0xfb){sim=0x09;goto z1;}//Символ"9"
else goto m1;
}
    //*****************************************************************

    /* Программа обслуживания второго МК, подключенного к первому МК через UART
    Lb8_2 */

#include<reg51.h>
#include<stdio.h>
        #include<stdlib.h>
        #define rel96 0xfd //константа таймера для скорости 9600 б.
        unsigned char cod;

        /*  функция инициализации UART и таймера CT1  */
        void inic()
        {
        IE=0x00; // выключаем систему прерываний микроконтроллера
        TMOD=0x20; устанавливаем таймер CT1 в режим 2
        TH1=rel96; //устанавливаем канстанты в таймер CT1 для скорости 9600
        TL1=rel96; /* контроллер должен иметь кварцевый резонатор на 11.059мгц */
        PCON&=0x7f;
        SCON=0x50;//UART будет работать в реж.1. Разрешаем прием данных
        //TI=0; Приемник готов к приему очередного символа
        }

        /*  Функция передачи принятого кода приемником UART второго МК через
         RS- 232 в МК1  */
        void put()
        {
        SBUF=cod;/* инициируем передачу загружая в SBUF принятый код  *.
        M4:
        if(TI!=1)goto m4; else TI=0;/* ожидаем окончание передачи символа и переходим на
        режим приема (TI=0) следующего кода  */
}
        /*  Функция приема очередного символа передаваемого первым МК через RS-232
        */
        void get()
        {
        m5:
        if(RI!=1)goto m5;/*ожидаем установки флага RI в единицу – прием очередного
        байта завершен*/
        else{cod=SBUF;//забираем принятый код из SBUF в  переменную cod
        P1=cod;
```

```
        RI=0;}//сбрасываем флаг RI
        }
void main ()
{
 inic();
h1:
get();
put();
goto h1;
    }
```

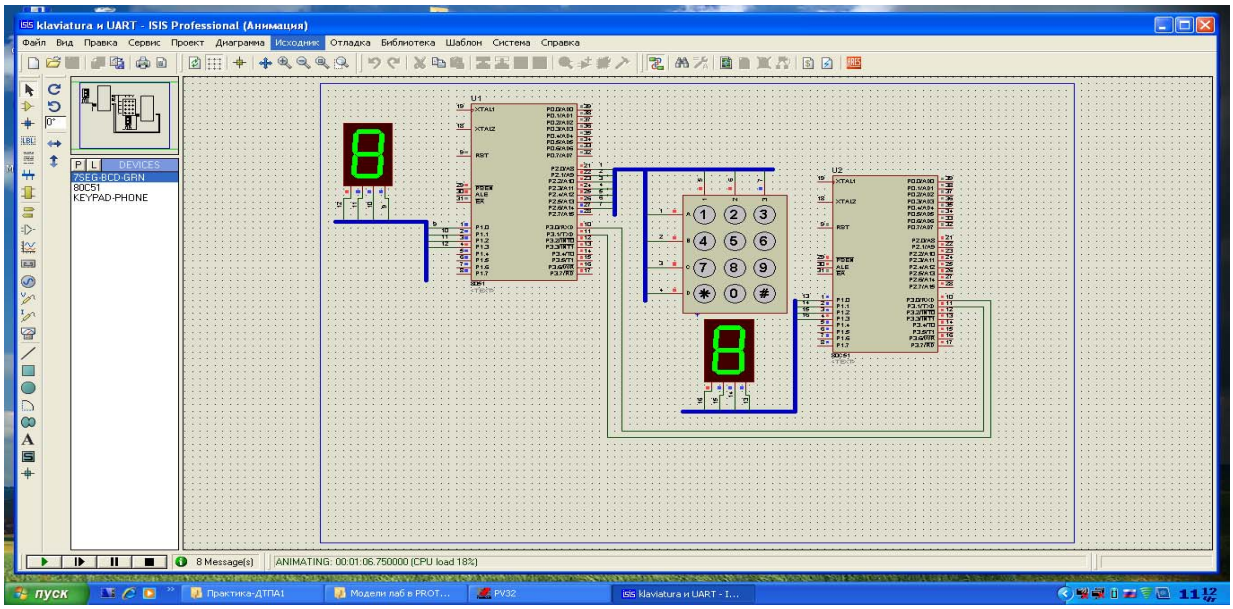



Рисунок 1. Модель схемы сопряжения микроконтроллеровМК1 и МК2по последовательному каналу в PROTEUS.

**Задание для практической работы:**

1. Разработать программу, обслуживающую клавиатуру с набором символов (ASC2 кодов), достаточным для ввода фамилии студента.
2. Выполнить ввод фамилии и вывод ее на UART на заданной скорости.

Таблица вариантов заданий.

| №вар | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| скоро сть | 12 00 | 24 00 | 48 00 | 96 00 | 12 00 | 24 00 | 48 00 | 96 00 | 12 00 | 24 00 |

**Содержание отчета.**

Отчет по лабораторной работе должен содержать:

1. Титульный лист.
2. Цель и задачи работы.
3. Схему включения клавиатуры и цифрового индикатора
4. Тексты индивидуального задания и программы согласно задания на языке СИ.
5. Выводы по работе.

## Список рекомендованной литературы


1. Однокристалльные МикроЭВМ. Справочник /под ред. Боборыкина, Липовецкого и др./ М.:Наука и техника, 1995. – 340с.
2. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах. М.:Наука и техника, 1991. – 245с
3. Белов А.В. Самоучитель по микропроцессорной технике. СПб.:Наука и техника,2003. – 224с.
4. Белов А.В. Конструирование устройств на микроконтроллерах. СПб.:Наука и техника,2005. – 255с
5. Фрунзе А.В. Микроконтроллеры? Это же просто! М.:ООО <ИД СКИМЕН>2002.(Т.1,2,3).
6. Микушин А. Занимательно о МИКРОКОНТРОЛЛЕРАХ. СПб.:<БХВ-ПЕТЕРБУРГ> 2006.- 424с
7. Каспер Эрни Программирование на языке АССЕМБЛЕРА для микроконтроллеров семейства I8051. М.: Горячая линия-Телеком, 2003.-191с
8. Программирование на языке С для AVR и PIC микроконтроллеров./ Сост. Ю.А.Шпак – К.: “МК-ПРЕСС”,2006-400с.,ил.
9. Методические указания к выполнению лабораторных работ по курсу “Архитектура компьютеров”. Составители: Башков Е.А., Мальчева Р.В., Красичков А.А. Донецк: ДОННТУ,2006г.-50с