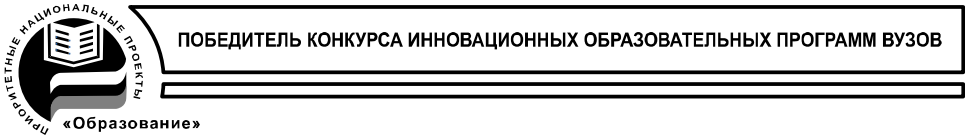
17.04.15
Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ



В.И. Скорубский

Управление вводом-выводом в mcs51

Часть II

Санкт-Петербург

2015

Скорубский В.И. : **Управление вводом-выводом в mcs51**/пособие к лабораторным работам . – СПб: СПбГУ ИТМО, 2014г. – с.

Пособие содержит описание и примеры выполнения лабораторных работ с встроенными средствами управления вводом-выводом

Продолжение изучения программных моделей совместимых с MCS51 архитектур с различными встроенными средствами управления вводом -выводом

Работы выполняются на двух уровнях – алгоритмическом с использованием языка С51 и уровне Макроассемблера А51. Используются эффективные и наглядные средства отладки в системе Keil на всех уровнях, в частности, графика Логического Анализатора для вывода и имитатор внешних событий в виде Сигнальных функций .

Пособие предназначено для студентов по курсу «Организация ЭВМ», Микропроцессорные системы, Программное обеспечение встроенных МПС для специальностей 230100 «Информатика и вычислительная техника», 230101 «Вычислительные машины, комплексы, системы и сети», 210202 «Проектирование, программирование и эксплуатация ИВС», 230104 «Системы автоматизации проектирования».

Рекомендовано Советом факультета Компьютерных технологий и управления _____ 2014 г., протокол №_____

СПбГУ ИТМО стал победителем конкурса инновационных образовательных программ вузов России на 2007-2008 годы и успешно реализовал инновационную образовательную программу «Инновационная система подготовки специалистов нового поколения в области информационных и оптических технологий», что позволило выйти на качественно новый уровень подготовки выпускников и удовлетворять возрастающий спрос на специалистов в информационной, оптической и других высокотехнологичных отраслях науки. Реализация этой программы создала основу формирования программы дальнейшего развития вуза до 2015 года, включая внедрение современной модели образования.

Санкт-Петербургский государственный университет информационных технологий, механики и оптики, 2012

Содержание

стр.

Введение.

Рассматривается системная организация ЭВМ (система на кристалле SoC), в которой интегрированы схемы, относящиеся к процессору, средства управления внешними интерфейсами для подключения внешней памяти и периферии, контроля и управления системой.

Традиционные модули SoC

- **программируемый** микроконтроллер **(** в данном случае mcs51)
- **таймеры** позволяют организовать измерение временных интервалов, контроль временных событий в системе, управление процессами,
- **подсистема прерываний (МПП)** обеспечивает контроль асинхронных системных событий ,
- **цифровые параллельные и последовательные внешние интерфейсы**
- **аналого-цифровые преобразователи**

На рис.1.1 приведена структура компьютера на основе mcs51.

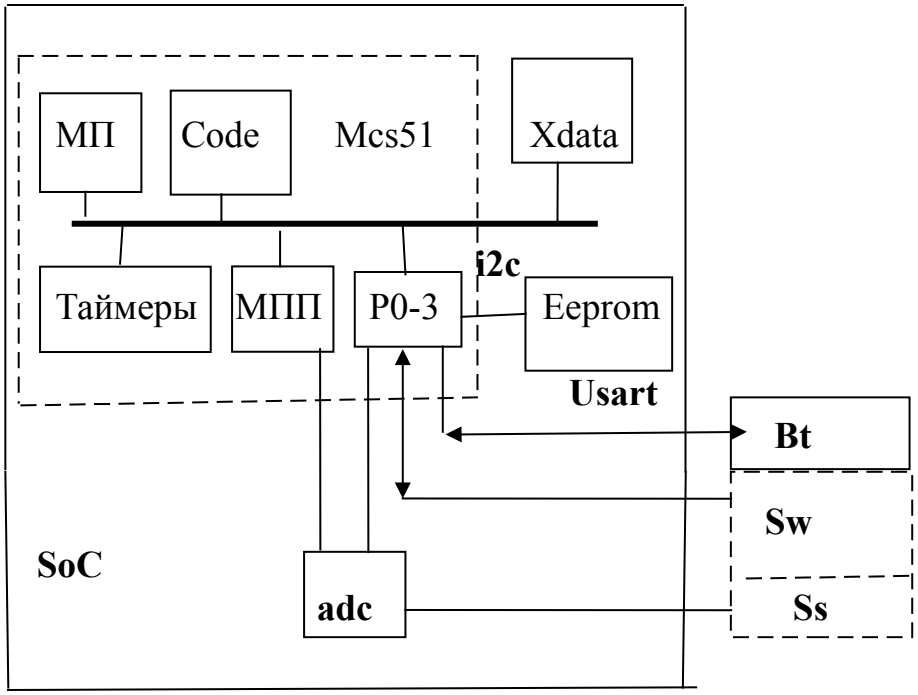


Рис.1.1.Структура компьютерной системы

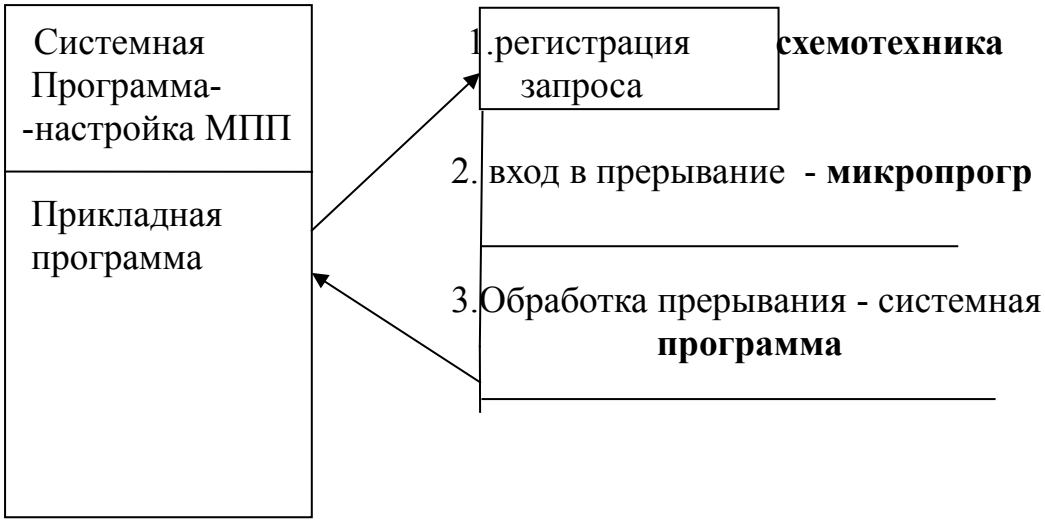
МП – микропроцессор в кристалле **mcs51**
I2c, Usart – последовательные интерфейсы
Eeprom – энергонезависимая память с последовательным интерфейсом **i2c**
Возможная внешняя периферия

Sw –переключатели, клавиатура
Ss – датчики, сенсоры
Bt – преобразователи цифровых проводных интерфейсов в беспроводные

I. Система прерывания mcs51.

Система обеспечивает контроль “случайных” внутренних и внешних событий, возникающих при исполнении “основной” программы, и представленные сигналами (запросами) прерывания.

Многоуровневая организация работы системы прерывания



1)Схемы контроля на аппаратном уровне фиксирует сигналы (**запросы прерываний**), обозначающие события , на регистрах запросов

2) Если разрешено прерывание (бит EA=1 и разрешены масками конкретные запросы) на микропрограммном уровне в начале следующей команды схемы МПП выбирают наиболее приоритетный запрос. Формируется **адрес-вектор** прерывания с сохранением адреса возврата в стеке.

3) В подпрограмме обработки прерывания сохраняется состояние **(контекст)** программы (ACC, PSW, используемые в подпрограмме регистры), выполняется некоторая функция, восстанавливается состояние и происходит возврат в прерванную программу.

Различаем внутренние и внешние сигналы прерывания (события)

1.1. Внутренние сигналы прерывания формируются схемами микропроцессора mcs51

К внутренним сигналам и событиям можно отнести сигналы , формируемые таймерами и последовательным интерфейсом при завершенииотсчетов временных интервалов и завершении приема и/или передачи байта данных в USART.

Параметры внутренних сигналов для mcs51 приведены в таблице

Таблица 2.1.

1	2	3	4	5	6
Tm0		TF0	ET0	1	0bh
Usart		T1vR1	ES	4	23h
Tm1		TF1	ET1	3	1bh

.

1)Источники запросов прерываний -таймеры `Tmi и Usart`

3) биты регистра запросов прерываний . Бит сбрасывается автоматически при входе в прерывание.
4) Маски разрешения прерывания (ET0=1 разрешено).
5) Номер (приоритет) прерывания – чем меньше номер, тем выше приоритет.
6) Адрес-вектор в таблице векторов прерываний в памяти Code.

1.2. Внешние прерывания mcs51 .

Внешние прерывания формируются сигналами (событиями) , связанными с асинхронными процессами в периферии на входах (контактах) цифровых портов.

Например,

1) при вводе с клавиатуры – сигналы нажатия клавиш,
2) временные диаграммы протоколов обмена данными через порты в последовательных интерфейсах

Внешние события в MCU MCS51 представлены сигналами-запросами прерывания на входных портах P3.2=INT0 и P3.3=INT1.

Таблица 2.2.

1	2	3	4	5	6
\p3.2 **INT0**	IT0	IE0	EX0	0	03
\p3.3 **INT1**	IT1	IE1	EX1	2	13h

1) входные сигналы прерываний.
2 **)** тип прерываний IT0=1 – выбирается H/L фронт входного сигнала.
3 **)** бит регистра запросов прерываний (IE0=1 запрос). Бит сбрасывается при входе в прерывание.

4) Маски разрешения прерывания (EX0=1 разрешено).

5)– Номер (приоритет) прерывания

6) – Адрес-вектор в таблице векторов прерываний в памяти Code.

Регистр признаков внешних прерываний

TCON = . . . ie1.**it1**.ie0.**it0.**

7 3 2 1 0

Виртуальные внутренние сигналы прерывания (**или программные прерывания)** могут быть сформированы программным переключением pins(контактов) портов P3.2 или P3.1, программируемых на ввод.

Управление прерываниями:

1) Установить маску прерывания и бит разрешения прерываний **EA**=1.

2) Сформировать вектор прерывания в таблице векторов по фиксированному адресу – (**jmp** на программу обработки в Ассемблере).

3) Подготовить программу обработки, которая завершается командой возврата из прерывания (**reti** в Ассемблере A51).

4) Основная программа и программа обработки прерываний должны быть смещены в памяти Code на размер используемой таблицы векторов прерывания (в опциях C51 проекта Кейл и в сегментах для Ассемблера).

5) В программе на Ассемблере резервировать Стек для прерываний и подпрограмм (по умолчанию в C51 назначается **SP=07**).

6) В программе на Ассемблере предусматривается сохранение состояния (контекста) программы и восстановление. Для сохранения и восстановления активных регистров **r0-r7** могут быть использованы два варианта – сохранение регистров в Стеке и сохранение переключением регистрового банка.

Обращение к подпрограмме в C51

```
void tm0(void) interrupt 1 using 1
 {                          }
```

tm0- имя подпрограммы обработки прерывания.

Interrupt – служебное слово-признак функции и 1-номер прерывания.

Using 1 – регистровый банк 1 используется в подпрограмме.

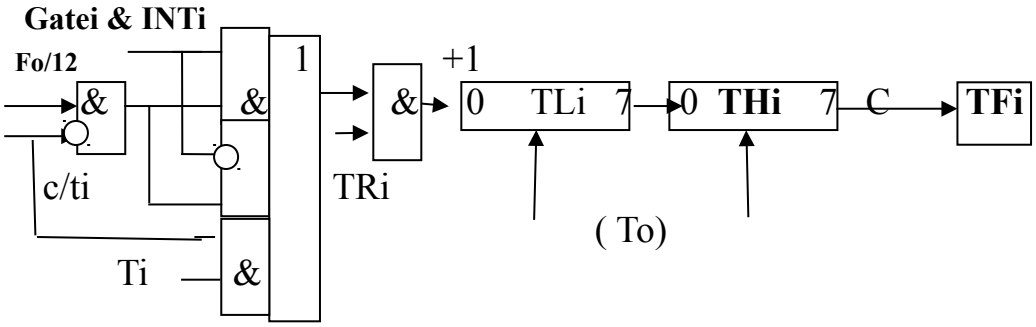
Указывая номер банка регистров, автоматически сохраняем **контекст** программы из регистров общего назначения текущего банка. При этом в программе обработки прерывания используется новый регистровый банк. При выходе из программы прерывания автоматически восстанавливается регистровый банк (регистровый контекст) на момент прерывания.

В языке C51 шаги 3-6 выполняются автоматически, в Ассемблере – программируются в системе команд

.

1.3. Таймеры. [ ]

Таймеры – счетчики реального времени, задаваемого стабильным кварцевым генератором частоты.
В MCS51 основной цикл выполнения команды 12 тактов генератора частоты. Команды выполняются за 1-3 цикла. Один цикл является единицей времени, отсчитываемой таймером. Если установить частоту генератора Fo=12 Мгц, то частота отсчета 1 Мгц и единица времени 1 мкс.

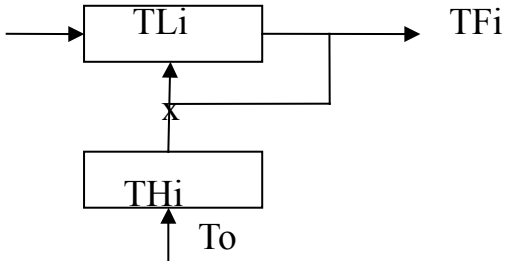


THi.TLi инкрементный 16-разрядный счетчик калиброванной **частоты fo/12 (цикл)** или счетчик **внешних событий** на входе Ti, представленных L/H фронтами.
Интервал измерения 16-битного таймера 2^{16} ~ 65 000 мкс = 0.065 с
To – отрицательная константа пересчета
Управляющие регистры таймеров

TCON = TF1.TR1,TF0.TR0. . . . .
7 6 5 4 0

TFi =C - бит переноса из старшего разряда таймера используется как сигнал запроса прерывания.
Для **расширения диапазона** используется программный счетчик , который инкрементируется по сигналу TFi.
TRi - бит разрешения счетчика
TMOD =gate1.c/t1.m1.m0.gate0.c/t0.m1.m0
7 6 5 4 3 2 1 0
m1m0=0x01 // режим TM0(TM1)- 16 бит счетчик

m1m0=0x10



8-битовый счетчик TLi с автозагрузкой THi по переполнению, другие режимы не актуальны

C/Ti - выбор синхронизации (0→f0/12, 1→Ti)

Возможны следующие **режимы измерения реального времени**

1) Формирование Задержки - загружается значение (–To), устанавливается режим 16-разрядного счетчика, разрешается работа счетчика Tri=1 (начало отсчета). При завершении заданного интервала To счетчик сбрасывается и формируется сигнал запроса прерывания TFi. To – единица программного отсчета реального времени в секундах, минутах.

2) Метод измерения **захватом (Capture**).

Начало и завершение временного интервала контролируется внешними прерываниями INT0,INT1 по фронтам H/L входного сигнала. Сохраняя t_i и t_{i+1} состояния таймера, определяем интервал (например, период) прямоугольного сигнала

$$T = (t_{i+1} - t_i).$$

3) Измерение **в режиме Gate** длительности положительного прямоугольного сигнала– отсчет TM0/TM1, пока входной сигнал INT0/INT1=H

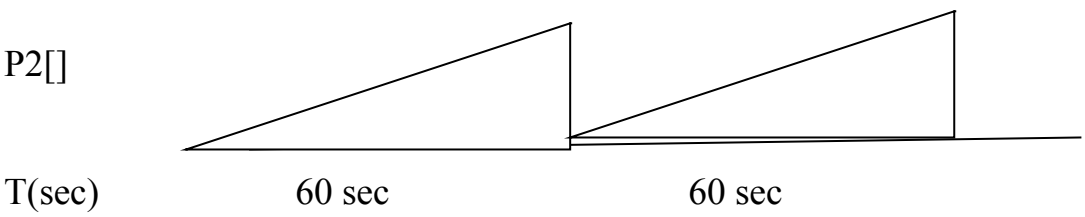
1. Измерения реального времени.

1) Таймеры по переполнению через постоянный интервал времени **To** вызывают прерывания. Счетчик событий используется для расчета текущего времени с учетом длительности интервала в секундах и минутах в портах P1,P2. Точность измерения не менее 0.1 сек на интервале 1 минута.

```
#include <reg51.h>

unsigned    char sec,min;
unsigned   char count=0;
intt0() interrupt 1  //счетчик
{
 TH0, TL0=To;                    //константа пересчета
  count++;  счетчик переполнений
  if(count== Tc ) {sec++;count=0;}
  if(sec== Tm ){min++; sec=0;}
   }
main()
{
  TMOD=1;
```

```
  ET0=1;
  TR0=1;
  EA=1;
  while(1)
    { P1=sec;
        P2=min; }
}
```

Real Time diagram in **Logic Analyzer** in Keil



Задание 1. Программы измерения в C51 и A51

Выбрать константу To. Построить временные диаграммы, оценить задержки и погрешность измерения времени. Исследовать погрешности в программах с другими способами измерения времени – непрерывный режим измерений с автозагрузкой.

2. Измерение частоты и скважности периодического сигнала .

Система KEIL позволяет формировать и запускать для параллельного исполнения скомпилированную прикладную программу в C51 или в A51 и программу имитации работы внешнего оборудования в режиме интерпретации в языке с синтаксисом C.

Внешние программы имитации **signal.inc** (имя и расширение выбираются) загружаются и запускаются на исполнение перед запуском основной программы. Для выделения времени на исполнение сигнальных функций в цикле формирования периодического сигнала применяются операторы задержки **twatch(100) -**интервал задержки в циклах **Fo/12.**

В языке доступны все внешние порты MCU, которые могут быть обозначены виртуальными именами (см. Keil/ Help).

time - имя временного параметра, используемого в в функции

Для выполнения сигнальных функций:

1. Перейти в режим Загрузки(Debug);
2. Выбрать виртуальные и реальные порты в графическом анализаторе командами **La P2, La Port3,..**
2. В окне COMMAND выполнить команду загрузки сигнального файл **>include fnt.inc** (файл fnt.inc должен быть в одном каталоге с объектным).

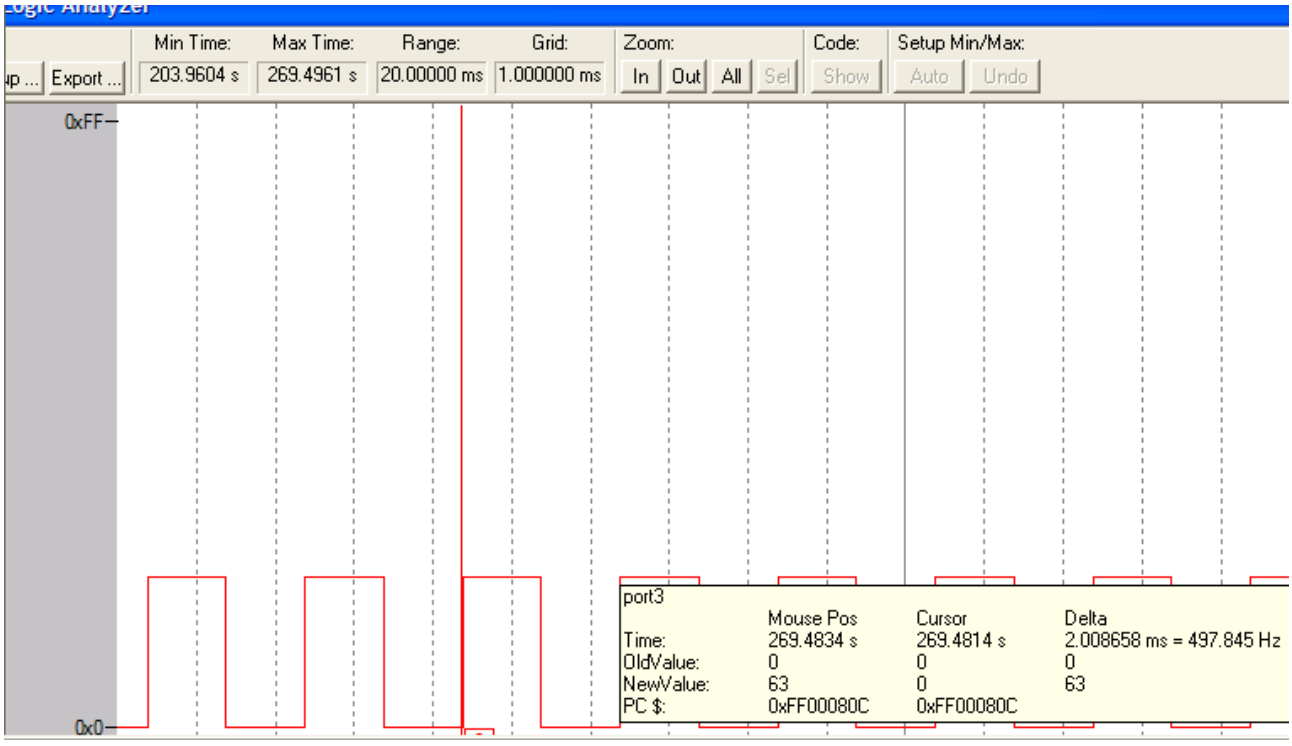
3. Запустить прикладную программу

Пример файла **func.inc.**

// Функции формирования входного периодического прямоугольного сигнала на входе P3.2- INT0)

```
SIGNAL void Signa (unsigned int Time) {
twatch (Time); //задержка Time  мкс при частоте f0=12мГц
while (1) {
PORT3 = 0x3f;
twatch (Time);
PORT3 = 0;
twatch (2*Time);
} }
```

Signa(1000) //запуск функции
La PORT3 //вывод сигнала в Анализаторе



Задание 2.

Сформировать временную диаграмму внешних сигналов на входе INT0 сигнальной функцией. В C51 и A51 в режиме **Gate** измерить длительность положительного интервала, в режиме **Capture** измерить период.

Оценить точность измерений временных параметров и сдвиг по фазе

3. Широтно-импульсная модуляция (ШИМ)

Метод кодирования информации скважностью прямоугольного сигнала называют ШИМ.

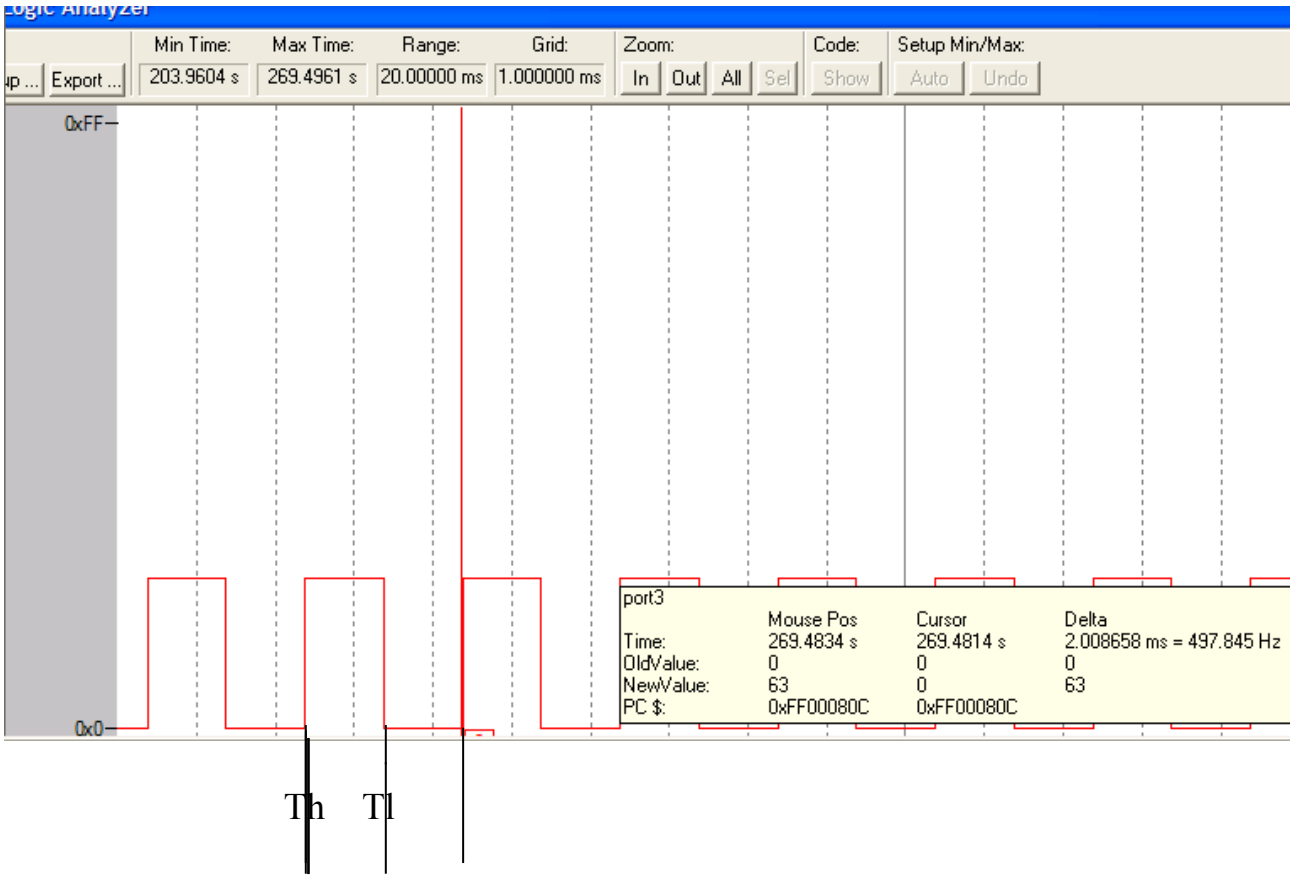
Применения ШИМ [2]

- передача данных
- управление двигателями постоянного тока
- управление двигателями переменного тока
- управление уровнем напряжения питания

Представим двоичное кодирование следующими временными интервалами положительного прямоугольного сигнала

1 ~ Th=t1, 0 ~ Th=to

Требуется сформировать временную ШИМ-диаграмму передачи двоичного кода на линии порта и прочитать код.



Скважность Th/(Th + Tl) или заполнение периода (Th +Tl)/Th.

Th + Tl = const

Задание 3.

Организовать кодирование байта данных и передачу кода . Построить и проверить в С51 временную диаграмму обмена данными.

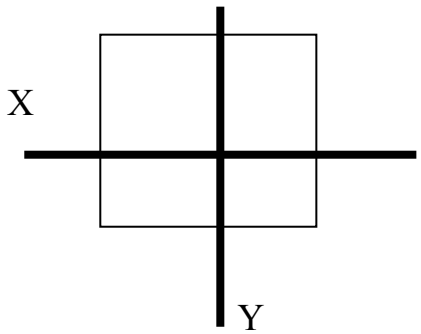
II. Ввод данных с переключателей и клавиатуры [2].

2.1. Сканирование клавиатуры

Ручное управление и ввод, контроль состояния ЭВМ и объекта управления, оперативное программирование осуществляется с **локальных пультов**. Типичный состав пульта – двузначные переключатели (Включение, Выключение, Режимы, ...), клавиатуры, светодиодные и ЖКИ-индикаторы.

Состояния объектов управления контролируются двоичными концевыми переключателями.

N переключателей для сокращения числа используемых цифровых линий при подключении к MCU объединяются в матрицу x*y.

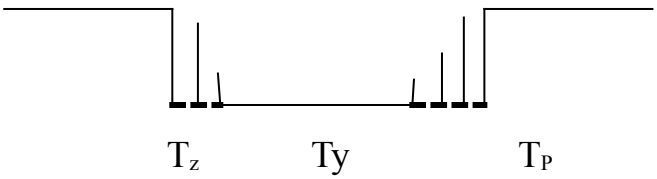


X линий используются для последовательного сканирования матрицы и являются выходами MCU, Y линий считываются при вводе.

Сканирование в MCS51 выполняется сигналами низкого уровня построчно (по умолчанию, состояние линии порта H) – унитарные коды Cx, при замыкании переключателя считывается унитарный код Cy.

Если в MCU используются m=X+Y выводов для сканирования и распознавания замкнутых переключателей, то матрица позволяет коммутировать N=X*Y=X*(m-X) переключателей.

Временная диаграмма переключения



Tz, Tp – время замыкания и размыкания с дребезгом около 0.04 с

Ty – время удержания в замкнутом состоянии. 0.1-0.2 с

Алгоритм сканирования

1. Формирование кода сканирования Cx

2. Ввод и контроль Cy - контроль нажатия клавиши

3. Повторить 1, если переключатель в строке X не замкнут

4. Идентифицировать замкнутый переключатель и выполнить соответствующую функцию

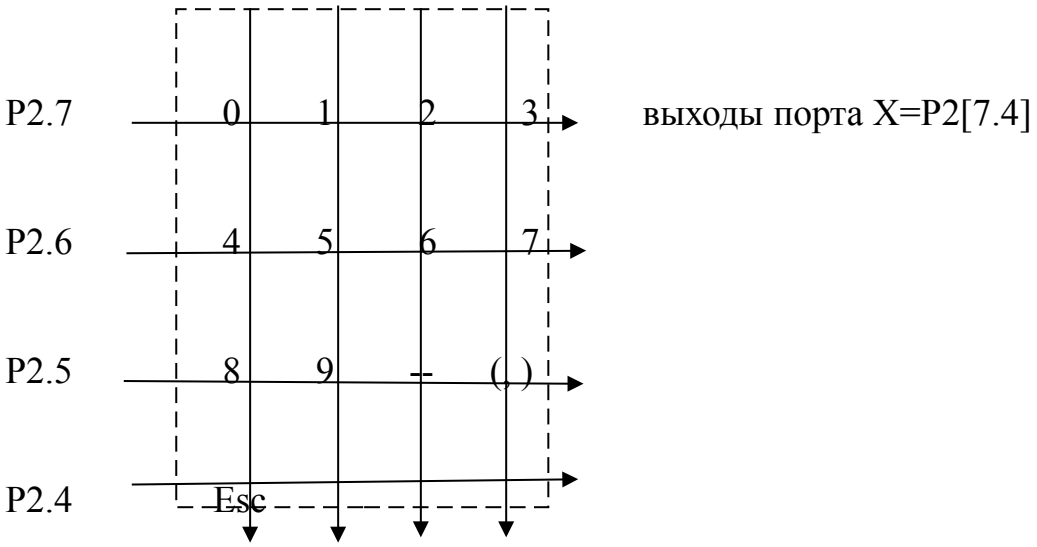
5. Задержка дребезга tz

6. Ожидание размыкания – считыванием и контролем кода Cy

7. При размыкании задержка размыкания tp и возврат к сканированию п.1.

Клавишам присваиваются в определенном порядке символы алфавита, цифры и специальные функциональные обозначения их размещения (**раскладка**)

Раскладки клавиатуры и схема включения с портом P3 mcs51.



0	1	2	3
4	5	6	7
8	9	(-)	(,)
(e)	.	.	.

Рис. 2.1. Раскладка клавиатуры.

Предполагается, что в нормальном состоянии (при включении питания) контакты порта P3[7.0] в высоком состоянии H и уровень поддерживается Pull_up резистором. В процессе сканирования в строки клавиатуры последовательно подается низкий уровень L. Если замкнут контакт одной из клавиш, L-уровень поступает на входные контакты порта P2[3.0]. Считываемый код с порта P2[7.0] идентифицирует нажатую клавишу.

В режиме ожидания контролируется состояние линий ввода (P2[3-0]!=1111_2)

1

При сканировании, например, строки P2.7=0 и нажатой клавише ‘1’ низкий уровень(L) поступает на вход P2.2 порта P2. Считываемое состояние порта P2 - **Код сканирования**.

	0	1	1	1	1	0	1	1
P2 [	7			4	3			0]
	вывод Cx				**ввод Cy**			

Числа вводятся с естественной запятой и преобразуются в машинный формат с плавающей точкой. Идентифицировать замкнутый переключатель

- завершение ввода - esc
- функциональные клавиши (-), (,)

При обнаружении нажатия клавиши программа сканирования сохраняет значение, пока не будет отпущена клавиша. Нажатие клавиши – случайное событие по отношению к синхронному событию сканирования. Можно представить, что кроме дребезга, возможно неуверенное считывание кода сканирования в конце или в начале импульса сканирования.

2.2. Программа ввода численных данных

Число знаковое вводится с естественной запятой и преобразуется в машинный формат с плавающей.
В системе Кейл нет адекватной модели клавиатуры со сканированием, предлагается заменить сканирование клавиатуры вводом кодов сканирования по прерыванию.

```
#include <reg51.h>
#include <intrins.h>  //расширение библиотеки C51 специальными функциями
unsigned char x,y,i,w,sign,
    digit,    //ASCII цифра, символ, вводимые с клавиатуры
    m=0 ;    //масштаб – 10 (номер цифры после запятой)
char mas[10] ={ 0x7e,0x7b, 0x7d,0xde, 0x7d, 0xbb,0xed   };  //контрольный массив введенных символов
float numb;   //формируемое при вводе  числа с естественной запятой

Delay(int t)   //задержка
  { while(t--); }

  pushing( ) interrupt 0
    { P2=mas[i++];}

char scan( );  //предопределение функции сканирования
```

```
main ( )
{  EX0=EA=IT0=1;
 x=0xEF;   //начальное значение кода сканирования
while (1)    //цикл сканирования
    {
   x= _crol_(x,1);   //код сканирования циклическим сдвигом (_crol( )-в intrins.h)
   if((x&0xF0)!=0xF0)
       { w=P2;    //порт-сканирование и ввод w=P2[7.4].pin P2[3.0]
    if((w&0x0f)!=0x0f)  //P2pin
        {Delay(100);  //противодребезговая задержка нажатия клавиши
           scan();       //функция ввода
              P2=0xff;
    while((P2&0x0f)!=0x0f)
           Delay(100);  //противо дребезговая задержка нажатия клавиши
         }
       }
    }
}

char scan(void)
{               if(i==0)    //начало ввода
              {numb=0; //сброс значения вводимого числа
               sign=0;    //признак знака (-)
               m=0;}     //нет масштаба
   switch(w)
        {     case 0xe7:  digit='0'; break;
              case 0x77: digit='1'; break;
              case 0x7b: digit='2'; break;
              case 0x7d: digit='3'; break;
              case 0xb7: digit='4'; break;
              case 0xbb: digit='5'; break;
              case 0xbd: digit='6'; break;
              case 0xd7: digit='7'; break;
              case 0xdb: digit='8'; break;
             case 0xdd: digit='9'; break;
             case 0xde: digit=','; break;
             case 0xed: digit='e'; break;
               case 0x7e: digit='-'; break;
                          default: digit=0xff;
            }
if(digit==0xff) goto exit;  //ошибочный код сканирования
  if(digit=='-') {sign=digit;goto exit;}
  if(digit==',') { m=1;goto exit;}  //начало формирования масштаба
  if(digit=='e')    //конец ввода числа
```

```
                                                    1
        { if(m) { numb=numb/m;                  i=0;m=0;  //завершение ввода
          if(sign) numb=-numb;          goto exit; }
  else {numb=numb*10 + (digit&0xf);
                        if(m) m*=10; } // ввод цифры и формирование масштаба
     exit: return digit;
   }
```

Задание 4. Выбрать коды для индивидуального варианта раскладки клавиатуры

1)

1	2	3	4
5	6	7	8
9	0	(,)	.
(-)	.	.	e

3)

0	1	2	3	4
5	6	7	8	9
(,)	(-)	.	.	e

5)

1	2	3	(-)
4	5	6	.
7	8	9	.
0	(,)	e	.

2)

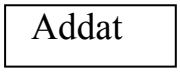
8	9	.	.
4	5	6	7
0	1	2	3
(-)	(,)	.	e

4)

1	2	3	.
4	5	6	.
7	8	9	e
0	(,)	(-)	.

III.Аналого-цифровые(ADC) преобразователи ЭВМ SAB515/535. [2]

Внешние сигналы, параметры которых передают информацию, в большинстве случаев имеют непрерывную аналоговую форму.
Элементы, преобразующие физические воздействия внешней среды в сигналы, параметры которых содержат информацию об этих воздействиях , называют ***сенсорами (датчиками)***.
Обзор датчиков [ ] показывает, что преобладает электрический выходной сигнал в виде напряжения различной формы. Входные данные для компьютеров в измерительных и управляющих системах, в медицинских приборах, в системах искусственного климата формируются преобразованием **аналоговых электронных сигналов в цифровую** форму .
ADC- преобразование может быть представлено как внешними, так и встроенными модулями в MCU и является одним из распространенных внешних интерфейсов ЭВМ.
Для демонстрации используется MCU SAB515/535 – расширение с ядром mcs51 [. Приложение]
Схема встроенного модуля ввода аналоговых значений в SAB515 и управления аналого-цифровым преобразованием



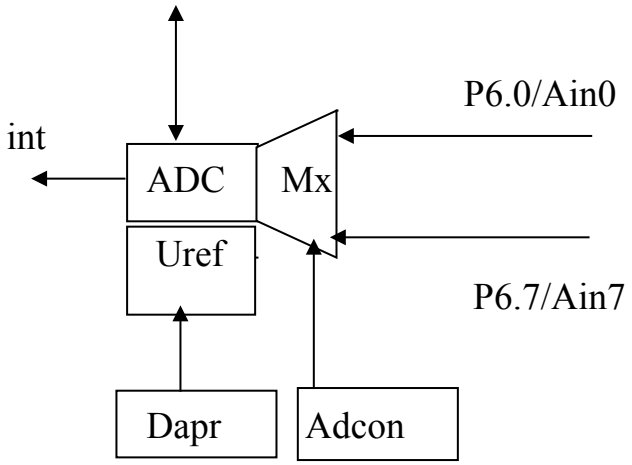


Рис. Схема модуля ADC в SAB515

Входные аналоговые сигналы подключаются к входам аналогового мультиплексора Mx и выбираются **mx[2.0]** полем регистра **ADCON -** 8 адресуемых аналоговых входа **Ain7-0** порта P6.
Результат преобразования – 8-разрядное целое двоичное число S в регистре **ADDAT**
Регистр управления **ADCON= ---.bsy.adm.mx[2.0]**

bsy=1 – признак незаконченного преобразования .
adm= 0/1 – одиночное/многократное преобразование.

8-разрядное ADC-преобразование **S(Ux)** последовательного приближения
Регистр масштаба **DAPR** определяет опорное напряжения **Uоп** преобразования. При DAPR =0 Uоп=5v, что соответствует максимальному значению уровня входного сигнала. Управляя **Uоп,** можно повысить точность преобразования на два бита [ ].
Цифровое значение аналогового сигнала – дробное число **$U=U_{оп}/2^8 *S$,**
где **$U_{оп}/2^8$** – цена бита для 8-разрядного преобразования;
S-двоичный код – результат преобразования;
U=Uоп для S ~ 0xFF.

Для целых вычислений необходимо перевести U с использованием масштабирования в целое

1) **Ввод аналогового сигнала** .
Сигнальной функцией формируется на входе синусоидальный аналоговый сигнал, выполняется ADC-преобразование.

#include <reg515.h> //в **опциях Project** выбрать микросхему **SAB515** //фирмы **Infenion**
delay(char t) //задержка преобразования
{ while(t--); }

```
Adc( ){  DAPR=0;   //запуск              преобразования с опорным
напряжением 5в
                While(BSY);  //задержка для завершения преобразования
                  P2=ADDAT;  //чтение результата и подтверждение его в P3 для
Анализатора
                  }
       main()
       {  char i;
        while(1){
            for(i=0; i<100; i++)  adc();
       }
```

Сигнальная функция для работы с Логическим анализатором (файл adc.inc)

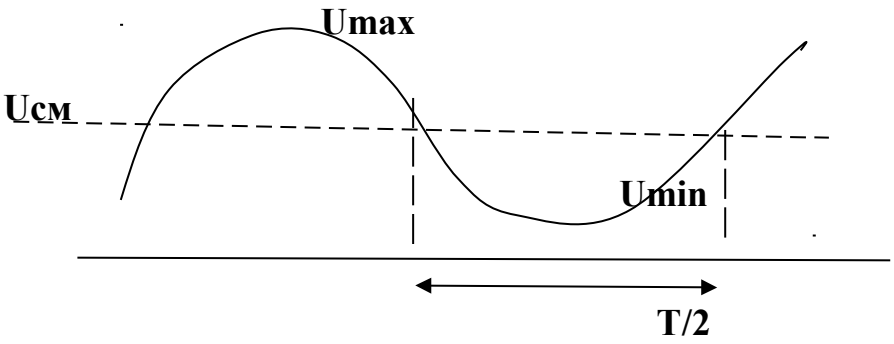
```
SIGNAL void Signa (void) {   //Сигнальная функция
float x;
char  i;
while (1) {    //__sin(x) –формирование аналоговой (float) величины  с учетом
опорного (и максимального по умолчанию) напряжения  0xff ~5в
 for(i=0;i<100;i++){
  AIN0 =__sin(x)*2 + 2,0;   //масштабирование и смещение в положительной
области на 2 вольта
    twatch (100);   //задержка    100  мкс при частоте f0=12мГц
    x=x+0.062;
        }}}
Signa()    //запуск сигнальной функции  -
LA AIN0     // контроль аналогового сигнала на входе
LA PORT2  //чтение целой величины –результата преобразования через порт P3
в программе измерений.
include adc.inc   - загрузка функции (командный режим)
```

Задание 5.1. Выполнить программу измерений с плавающей точкой и в целых в C51, в A51 – в целых,
Измерить задержку преобразования
В A51 можно отменить стандартные режимы

```
$nomod51    ;отмена стандартных режимов MCS51
 $include  (reg515.inc)  ;загрузка файла определения регистров SAB515
```

Или определить отсутствующие в mcs51 регистры управления SAB515(см Приложение)
Для желающих можно предложить другие модели преобразования в модулях **Aduc812-824, AT, Ti**

2) **Измерение параметров синусоидального сигнала – Umax, Uvin, Uсм, T**



Реальные измерения выполняются в условиях помех и случайных воздействий внешней (производственной) среды. При этом значение **min** и **max sin-**сигнала измеряются с погрешностью и практически при многократных измерениях не совпадают. В динамике выбираем метод измерения периода между двумя переходами через 0(среднюю линию).

```
#include <reg515.h>

int mm;
unsigned int T;
unsigned char max,min;

unsigned char adc(void); //предопределение функции
        main()
        {
          max=0; min=0x70;
                          while(INT0)    //ожидание INT0=P3.2pin==0
            { if (adc()>max) max=P2;
                  if (adc()<min) min=P2;
             }
            mm=(max+min)/2; //среднее значение
            TMOD=1;
            TH0=TL0=0;
            TR0=0;
           while(adc()>=mm);
               while(adc()<=mm);      //ожидание L/H  средней линии
              TR0=1;
          while(adc()>=mm);         //ожидание H/L  средней линии
            TR0=0;
            T=(TH0<<8)+TL0;
                              while(1);  //динамический останов
        }

 unsigned char adc(void)
  {  unsigned char x;
```

```
DAPR=0;
 while(BSY);     //ожидание завершения преобразования
return P2=x=ADDAT;
            }
```

Сигнальная функция из предыдущего примера.

Задание 5.2. В С51выполнить программу измерений амплитуды и среднего значения в вольтах с точностью 2 знака после запятой. Измерить среднее время измерений и период сигнала, время ввода 1000 отсчетов по прерыванию)- оценить погрешности реального времени (задержка по фазе, среднее время преобразования)

IV. Последовательные интерфейсы [2].

Последовательные каналы (интерфейсы) с минимальным числом линий связи (2-3) широко применяются для подключения к ЭВМ программируемого удаленного внешнего оборудования и дополнительных встроенных модулей преобразования интерфейсов.

Интерфейсы различаются как синхронные и асинхронные..

4.1. Универсальный синхронно-асинхронный приемник-передатчик (USART)

Модуль USART в mcs51 совмещает асинхронный (ART) и синхронный (SRT) обмен данными

Асинхронный интерфейс используется как внешний, так и внутренний – межмодульный и мультимашинный.

Для связи с удаленным периферийным оборудованием или HOST-машиной, требуются помехоустойчивость, согласование уровней и электрическая развязка по питанию.

Согласование обеспечивается специальными схемами преобразования сигналов в **стандартах RS232 или RS485.**

RS232 (1962 г EIA) – физический интерфейс NRZ (точка-точка), расстояние до 15 м, скорость до 19.2 Кбод (в настоящее время- 115 Кбод). Уровни сигналов от -12 до -5в – значение единицы и от +5 до +12в – значение нуля, минимальное число линий 3 **(Tx,Rx,Gnd)**.

Асинхронный протокол (ART) обмена данными по внешним каналам передачи данных 8-9 бит включает 8 бит двоичного кода данных и, в частном случае, бит контроля четности, Для выбора байта в последовательности битов

используется *старт-бит* низкого уровня и завершающий *стоп-бит* высокого уровня.

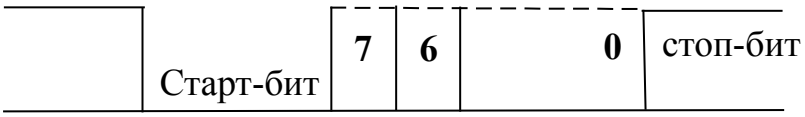


Рис. 4.2. Протокол передачи байта в RS232.

Синхронизация битов обеспечивается:

- многократным 1:16 и более сканированием бита,
- одинаковой скоростью передачи на стороне датчика и приема на стороне приемника.

При этом допускается некоторое рассогласование синхронизации по фазе, что и характеризует такой интерфейс как асинхронный.

RS232 (COM-порт) используется в качестве внешнего интерфейса ПЭВМ для подключения периферии.

Большинство MCU включают в качестве внешнего интерфейса модуль подобный UART – один или несколько. В частности, MCS51 включает совмещенный модуль USART синхронного и асинхронного интерфейса. [Сташин В]

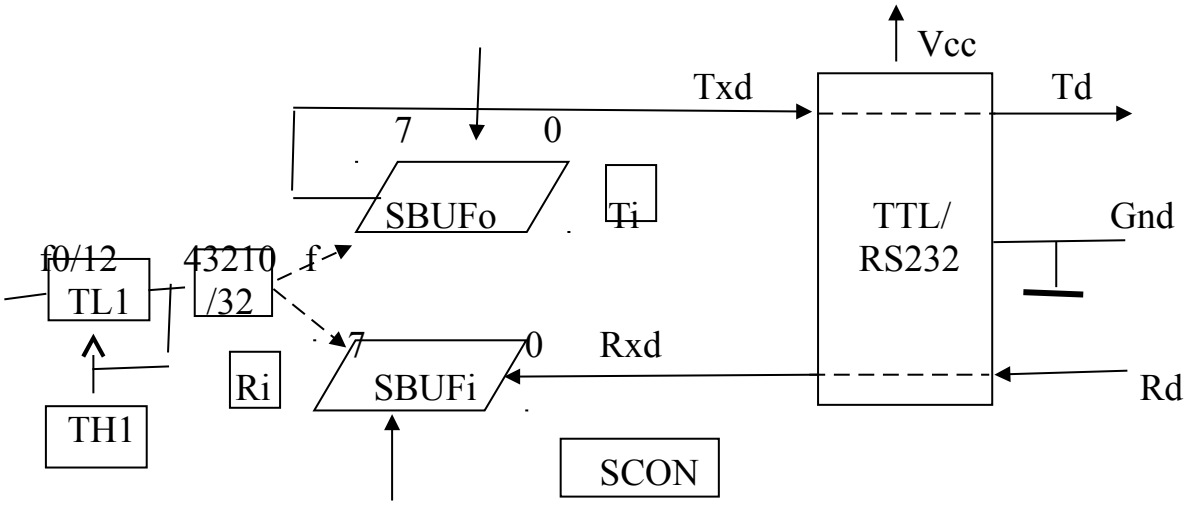


Рис.4.1. Схема работы модуля UART с интерфейсом RS232

USART реализует четыре режима:

0– синхронный режим для межсхемного использования полудуплексный двухпроводный с двумя линиями – двунаправленная линия данных и однонаправленная линия синхронизации, скорость передачи 1 Мбод. =f0/12

1 – асинхронный 8-битовый дуплексный (линия Td- передачи и линия Rd-приема) с программируемой таймером **Тм1** (во втором режиме) переменной

скоростью обмена (стандартный ряд скоростей – 480 бод, 960, ..4800, 9600, 19.200,). 8-разрядный таймер позволяет получить необходимую скорость до 19.2 Кбод с точностью не менее 3% Этот режим используется чаще других и согласуется с **COM-портом PC**.

2–**асинхронный 9-битовый** дуплексный (9 бит четности), фиксированная скорость

3 – асинхронный 9-битовый (9 –бит признак адреса) переменная скорость Модуль содержит два регистра данных **SBUFo –** буфер вывода и **SBUFi –** буфер ввода (обращение по одному адресу SBUF)

Управляющий регистр

SCON= SM[0.1].SM2.REN.TB8.RB8.Ti.Ri

Линии дуплексного обмена Txd(P3.1) и приема Rxd(P3.0) могут быть использованы для соединения модулей в мультимашинной системе. Для организации внешнего интерфейса в стандарте Rs232 используется внешняя схема преобразования уровней.

При **инициализации USART** в регистре **SCON** разрешается чтение **REN=**1, выбирается режим SM[0-1]={0 – синхронный(S) 8 бит,

1- асинхронный(A) 8 бит, 3-асинхронный точка-точка и мультимашинный Master/Slave} 9 бит.

Скорость передачи задается таймером Тм**1** (режим 2 – меандр, 8 – разрядный счетчик, частота задается константой в TH1). Скорость определяется частотой с делителя TL1 и TH1-константа автозагрузки по переполнению таймера TL1. /32-счетчик(делитель частоты) используемый для идентификации значения бита в интервале примерно 100мкс, когда передается бит. Например, f0=11 mGc, TH1=0xfd(-3), скорость (частота) передачи битов при тактовой частоте работы MCU f0=11.059 Мгц составляет f=(f0/12) /TL1/32 =f0/ (12*32*TL1) = 9.600 Гц(бод), что соответствует стандартной скорости обмена 9.600 бод

5- битный счетчик детектирования битов при вводе может контролировать значение бита тремя сигналами (разряды 3,2,1) по совпадению. Биты 0,4 пропускаются, что допускает сдвиг по фазе на 3%

Отрицательный перепад старт- бита синхронизирует начало детектирования битов в следующем интервале Детектирование завершается по стоп-биту после ввода 8 битов.

Завершение обмена контролируется битами готовности TI (завершение передачи), RI (завершение приема) или по прерыванию при установке этих битов TI v RI и разрешенной маской ES.

Передача **начинается** записью байта в регистр SBUFo=x, биты кода передаются в **линию Txd старшими разрядами вперед**, завершается передача установкой признака **Ti=1;**

В программе автоматического приема контролируется перед началом состояние Ti=1 (контроль завершения предыдущей передачи) –после чего флаг сбрасывается.

Если разрешен прием **(REN=1)**, то поступающая на **вход Rxd** последовательность бит сдвигается в регистр SBUFi и устанавливается признак завершения приема **Ri=1**.- в программе чтения флаг сбрасывается.

Типовая настройка – скорость обмена 9600 бод, 8-бит формат, 1 – стоп бит, нет бита контроля четности, управление обменом – программное, режим асинхронный SM=01.

Стандартные функции из библиотеки **studio.h** языка C51(**getkey, printf**,..) обмениваются данными с периферийным оборудованием через UART.

Функция **char getkey()** – ввести символ
Функция **printf("text %d\n", x)** – форматированный вывод в последовательный канал, результат отображается в окне **Serial** системы Keil.

Ввод по прерыванию

```
#include <reg51.h>
#include <stdio.h> //библиотека ввода-вывода C51

char  i,aa[6],s;
int x;

Seria() interrupt 4
{
   S=SBUF;
   aa[i++]=s;
   aa[i]=0;                    // конец строки
  x=x*10+(s&0x0f);             //двоичное число
    // printf("%s", s);        //работа дуплексная совмещенная ввода/вывода
                }
}

main()
{
        SCON=0x50; //режим асинхронный 8 бит, разрешение ввода ren=1
        TMOD=0x20; //таймер 1 - режим 2
         TH1=0xfd;   //константа автозагрузки - частота 9600 бод
         TR1=1;
           ES=1; //маска прерывания
         EA=1;
           TI=1;  //начальная установка готовности передачи
          While (i!=5) ; //ожидание прерывания и ввод
             printf("x= %4d\n", x);   // форматированный вывод  в USART
```

```
        i=0; x=0;
    while(1) ;
}
```

Прикладная программа выбирает режим USART (асинхронный, 8 бит, разрешение приема), скорость обмена данными (синхронизация осуществляется таймером TM1 в режиме меандра(TMOD=0x20), скорость обмена задается константой пересчета (TH1=0xfd — 9600 бод при частоте генератора синхросигналов 11.056 кГц) .

По прерыванию (завершение приема или передачи байта**)** читается символ **getkey()**, при вводе цифр числа формируется двоичное число и выводится **printf()** - отображается в окне SERIAL (меню VIEW).

Сигнальная функция формирует биты данных для ввода в последовательном интерфейсе.

В сигнальной функции используется переменная **SIN** для обозначения входа RxD последовательного канала (Симулятор поддерживает только функцию соответсвующих команд ввода-вывода и не отображает временную диаграмму протокола обмена на линиях порта P3)

Ниже приведен файл **serial.inc** сигнальной функции для ввода ASCII -кодов символов цифр

```
signal void serial(void){
char s;
for(s=0; s<5;s++){
  twatch(1000);
  SIN=s+0x30;  //ввод  ASCII-кода  цифры
  }
  }
serial()  //исполнение функции
```

1/ Загрузка программы и ввод сигнального файла
2/ Запуск программы и однократный запуск функции serial() – ввод числа и отображение результата **printf( )** в окне **Serial #0** , в окне **Watch** отображается массив цифр.
3) Для повторного ввода – сбросить программу Reset и повторить пункт 2.

Задание 6:
1. ввести и контролировать текстовую строку
2.ввести десятичное число с естественной запятой и знаком ,

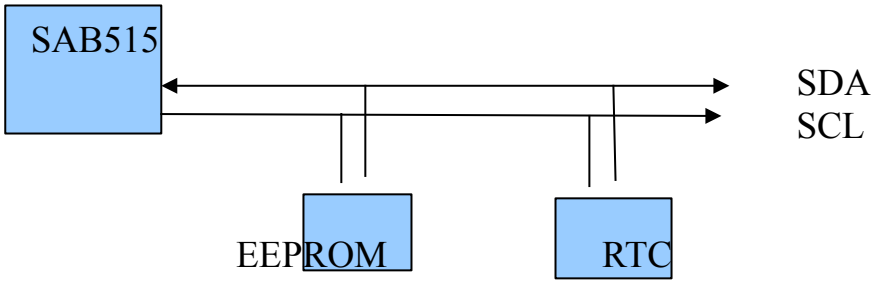
преобразовать в двоичное с плавающей точкой и контролировать ввод при печати

4.2.Синхронный Интерфейс I2C.

Синхронные Интерфейсы используются для внутрисхемного применения (на печатной плате между модулями) в режиме Ведущий(Master)-Ведомый(Slave)**.** В явном виде используется линия жесткой **побитовой синхронизации**

Физический интерфейс - дуплексная (двунаправленная) линия данных SDA и синхросигнал SCL, который формируется Ведущим.

Для организации удаленного обмена требуется линия Gnd. К линиям SDA ,SCL с открытым коллектором могут подключаться несколько ведущих(Masters) и до 127 ведомых(Slaves)



Протокол обмена данными может быть представлен временной диаграммой, зависит от направления обмена байтами (чтение или запись со стороны Ведущего). Микросхемы Slave аппаратно поддерживают протокол.

Элементы протокола:

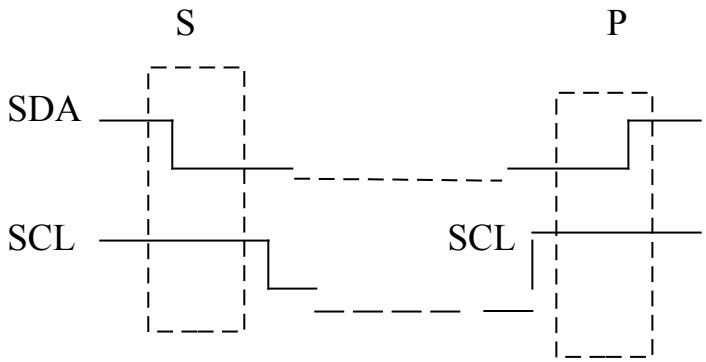
Ведущий MCU формирует **синхросерию** на линии SCL

Ведущий формирует **Старт-условие S(разрешение обмена)** и **Пост-условие P(завершение обмена):**

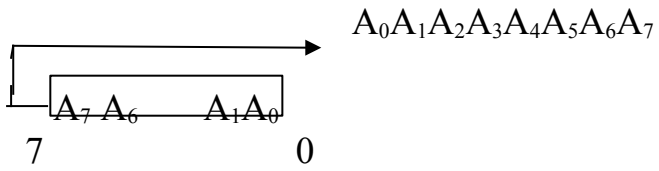
Первый байт протокола – адрес Slave и признак записи байта.

Второй байт **может быть** адрес байта в Slave.

Далее последовательно передаются байты и **пост-условие P(завершение обмена):**

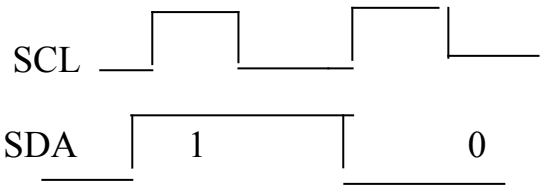


Информация передается байтами, приемник подтверждает прием байта битом подтверждения (SDA=ACK=0), отсутствие бита (SDA=ACK=1) обозначает задержку-ошибку чтения в модуле, который в данный момент является приемником.
Первый байт протокола содержит адрес Slave, зависящий от типа микросхемы Slave, следующие биты либо содержат адрес одного из однотипных элементов Slave или часть адреса ячейки или регистра в Slave, последний бит 0 обозначает запись. Биты байта передаются **старшими разрядами вперед**.



На временной диаграмме t,t+1,t+2, ….t+7 видим биты в зеркальном отражении.

Далее в зависимости от типа элемента и выполняемой операции (запись или чтение) формируются байты протокола записи или чтения и последовательно передаются байты данных.
Биты данных при чтении фиксируются по положительному фронту SLC.



Программное управление интерфейсом I2C

Простой синхронный протокол I2C может быть реализован полностью программно без аппаратной поддержки со стороны Master. Во многих MCU предполагается аппаратная поддержка формирования соответствующей временной диаграммы, что позволяет увеличить скорость обмена.

//Протокол записи в Slave по произвольному адресу формируется ведущим
/*
S. Ad.0.A.W_адрес.A. W_data.A.W_data.A.....P
S-старт-условие,
Ad.0- 7-1 биты адреса slave-приемника , последний 0-бит равен 0-признак передачи байта

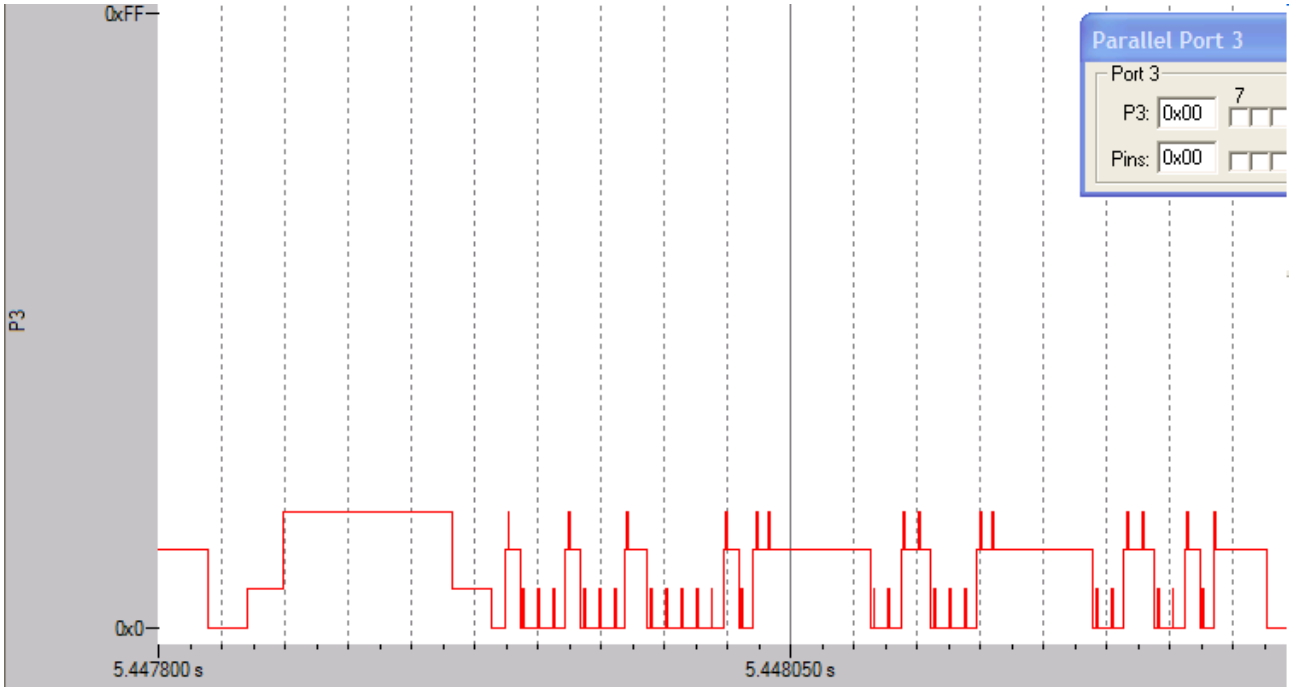
```
         A- Ack - подтверждение чтения приемником
        W_адрес.- байт адреса в модуле
       W_data - байты данных
         P- пост-условие
       */

//Протокол чтения из RAM -_Slave  по произвольному адресу формируется ведущим
        /*
         S.Ad.0.A.W_адрес.A.S.Ad.1.data.A....data.1.P
       S-старт-условие,
         Ad.0- 7 бит адрес slave-приемника , последний бит 0-признак передачи байта
        Ad.1- 7-1 биты адреса slave-передатчика , последний 0-бит равен 1-признак переключения ведущего на чтение
        W_адрес.- байт адреса в модуле
         A-   Ack - подтверждение чтения приема байта Maste
         1-   бит подтверждения приема
        P- пост-условие
              */

#include <reg515.h>
unsigned char ad_i2c=0x88; //адрес Slave-модуля в интерфейсе    [A6    A1A00]

char bdata byte;         //байт с побитовым доступом     7 … 0
sbit b0=byte^0;
sbit b7=byte^7;
char tt[ ]="1234";
// ---------------интерфейс I2C
wait(unsigned int x)
{ while(x--); }

sbit scl = P3^4;   //физический интерфейс I2C
sbit sda = P3^5;

#define clk { scl=1; scl=0;}   //формирование синхросигнала ведущим
#define s_cond { sda=scl=1; sda=0;wait(1); scl=0;} //старт-условие формируется ведущим

#define p_cond { sda=0;wait(1); scl=1;wait(1); sda=1;}       //пост-условие формируется ведущим

#define Ack   { sda=1; sda=1;sda=0;}   // бит Ack=1 – нормальное состояние линии с открытым коллектором, на SDA формируется Ack=0 читающим модулем  для подтверждения завершения операции чтения и записи байта
```

```
#define w_bit { sda=b7; clk; byte<<=1; }   //вывод старшего бита в линию
#define r_bit { sda=scl=1; byte<<=1;  b0=sda; scl=0; }  //чтение старшего бита с
линии

// ----------------------функции интерфейса
w_byte(char y)     /* синхронная запись байта  ведущим МКК */
{ unsigned char i;
  byte=y;
  for (i=8;i;i--)  w_bit;
  sda=1;clk;
} //читать ACK не обязательно, но синхросигнал для чтения обязательно

 r_byte()     /* синхронное чтение байта  ведущим МКК */
{ unsigned char i;
  for (i=8;i;i--) r_bit;
  Ack; }


 W_text(char x,char *y)   // запись байта y по адресу x
{                    char i=0;
          s_cond;
          w_byte(ad_i2c); // адрес модуля-запись

           w_byte(x); // адрес ячейки EEPROM
                 while(1)
                 {if(y[i]==0) break;
           w_byte(y[i++]); // байт данных
                  }
           p_cond;
                 }

char R_text(char x)        // чтение байта по адресу
{     s_cond;
      w_byte(ad_i2c);    // адрес модуля - запись
      w_byte(x);    // адрес ячейки EEPROM
      s_cond;          //переключение с записи на чтение
           w_byte(ad_i2c|1);   // адрес модуля и далее чтение
      r_byte();   // байт данных
      p_cond;
           return(byte);
  }

main()
```

```
{
 while(1)
   {P3=0x30;
    wait(1);
    W_text(5,&tt[0]);
         wait(5);
         }
}
```

Совмещенная в P3 временная диаграмма сигналов SDA, CLK, S, P при записи в Slave.

S-старт-условие,
Ad.0- адрес slave 1000100w w-последний бит 0-признак записи
A- Ack - подтверждение чтения приемником
W_адрес.- байт адреса в модуле 0x05
W_data - байт данных 0x31
W_data - байт данных 0x32
P- пост-условие



P S 10001000a00000101a 00110001a 00110010a

Задание 7.

1. Выполнить программу вывода(записи) символьной строки в протоколе i2c .
2. Показать и комментировать временную диаграмму работы интерфейса в Анализаторе
3. Измерить скорость передачи байтов

4. С использованием сигнальной функции организовать чтение slave-контроллером данных , записанных в линию ведущим-сигнальная функция

Литература.

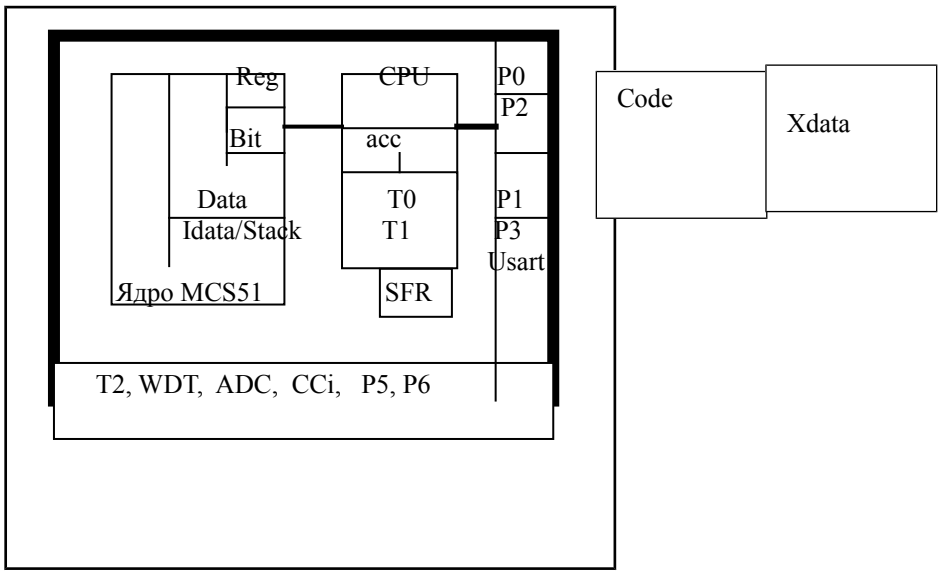
1. ЛабОРгЭВМ. Mcs51_1.
2. ЛабОРгЭВМ/ Mcs51_2./периферия
3. ЛабОРгЭВМ/ Mcs51_2./ SAB515/517.PDF

Расширение mcs51 в SAB515

-16 бит таймер T2 работает в режиме 16-бит счетчика и используется как опорный для каналов Захвата-Сравнения (CCC)

- программируемый сторожевой таймер WDT, при включении и отсутствии программного сброса в течение 40 мс обеспечивает принудительный сброс микроконтроллера.
- 8/10 разрядный 8 канальный АЦП(ADC) с внутренним программируемым опорным напряжением , формируемым из 5 в питания
- 6 канальный модуль захвата-сравнения CCi
- 8-разрядные универсальные параллельные порты P4, P5 и порт ввода аналоговых сигналов P6

Диаграмма ресурсов микроконтроллера



SFR	Adcon	D8
	Addat	D9
	P4	E8
	P5	F8
	P6	PB

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

О кафедре

Кафедра ВТ СПбГУ ИТМО создана в 1937 году и является одной из старейших и авторитетнейших научно-педагогических школ России.

Первоначально кафедра называлась кафедрой математических и счетно-решающих приборов и устройств и занималась разработкой электромеханических вычислительных устройств и приборов управления. Свое нынешнее название кафедра получила в 1963 году.

Кафедра вычислительной техники является одной из крупнейших в университете, на которой работают высококвалифицированные специалисты, в том числе 8 профессоров и 15 доцентов, обучающие около 500 студентов и 30 аспирантов.

Кафедра имеет 4 компьютерных класса, объединяющих более 70 компьютеров в локальную вычислительную сеть кафедры и обеспечивающих доступ студентов ко всем информационным ресурсам кафедры и выход в Интернет. Кроме того, на кафедре имеются учебные и научно-исследовательские лаборатории по вычислительной технике, в которых работают студенты кафедры.

Чему мы учим

Традиционно на кафедре ВТ основной упор в подготовке специалистов делается на фундаментальную базовую подготовку в рамках общепрофессиональных и специальных дисциплин, охватывающих наиболее важные разделы вычислительной техники: функциональная схемотехника и микропроцессорная техника, алгоритмизация и программирование, информационные системы и базы данных, мультимедиатехнологии, вычислительные сети и средства телекоммуникации, защита информации и информационная безопасность. В то же время, кафедра предоставляет студентам старших курсов возможность специализироваться в более узких профессиональных областях в соответствии с их интересами.

Специализации на выбор

Кафедра ВТ ИТМО предлагает в рамках инженерной и магистерской подготовки студентам на выбор по 3 специализации.

1. Специализация в области информационно-управляющих систем направлена на подготовку специалистов, умеющих проектировать и разрабатывать управляющие системы реального времени на основе средств

микропроцессорной техники. При этом студентам, обучающимся по этой специализации, предоставляется уникальная возможность участвовать в конкретных разработках реального оборудования, изучая все этапы проектирования и производства, вплоть до получения конечного продукта. Дня этого на кафедре организована специальная учебно-производственная лаборатория, оснащенная самым современным оборудованием. Следует отметить, что в последнее время, в связи с подъемом отечественной промышленности, специалисты в области разработки и проектирования информационно-управляющих систем становятся все более востребованными, причем не только в России, но и за рубежом.

2. Кафедра вычислительной техники - одна из первых, начавшая в свое время подготовку специалистов в области открытых информационно-вычислительных систем. Сегодня студентам, специализирующимся а этой области, предоставляется уникальная возможность изучать и осваивать одно из самых мощных средств создания больших информационных систем - систему управления базами данных Oracle. При этом повышенные требования, предъявляемые к вычислительным ресурсам, с помощью которых реализуются базы данных в среде Oracle, удовлетворяются за счет организации на кафедре специализированного компьютерного класса, оснащенного мощными компьютерами фирмы SUN, связанными в локальную сеть кафедры. В то же время, студенты, специализирующиеся в данной области, получают хорошую базовую подготовку в области информационных систем, что позволяет им по завершению обучения успешно разрабатывать базы данных и знаний не только в среде Oracle, но и на основе любых других систем управления базами данных.

3. И, конечно же, кафедра не могла остаться в стороне от бурного натиска вычислительных сетей и средств телекоммуникаций в сфере компьютерных технологий. Наличие высокопрофессиональных кадров в данной области и соответствующей технической базы на кафедре (две локальные вычислительные сети, объединяющие около 80 компьютеров и предоставляющие возможность работы в разных операционных средах - Windows, Unix, Solaris), позволило организовать подготовку специалистов по данному направлению, включая изучение вопросов компьютерной безопасности, администрирования, оптимизации и проектирования вычислительных сетей.