第12章 STC单片机串行异步收发器原理及实现

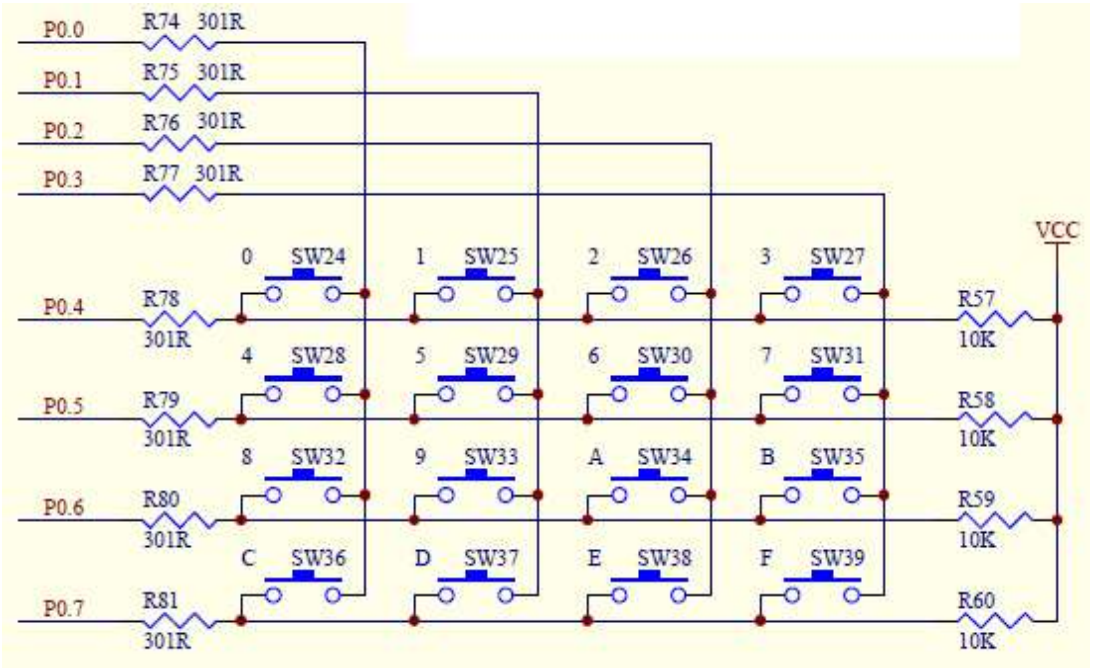
何宾

2019.05

串口1通信实例2
---矩阵按键结构及检测原理

矩阵按键结构及检测原理

- 在STC学习板上提供了16个按键，这16个按键按4×4形式排列，即：4行和4列形式。



串口1通信实例2
---矩阵按键结构及检测原理

由图可以判断出在实际中P0.0~P0.3应该为输出，或者逻辑高电平、或者逻辑低电平；而P0.7~0.4为输入，也就是读取P0.7~P0.4引脚的状态。

- 首先如何判断有按键被按下，方法是将P0.0~P0.3引脚拉低，也就是驱动P0.0~P0.3为低。
- 如果16个按键中，没有按下按键，则P0.4、P0.5、P0.6或者P0.7仍然处于上拉状态，即：逻辑高/逻辑1，此时如果读取这四个端口，读取的值应该是1111，分别对应于P0.7、P0.6、P0.5、P0.4引脚。
- 只要有一个按键按下，P0.4、P0.5、P0.6或者P0.7就有引脚被拉低，也就是读P0.4、P0.5 、P0.6、 P0.7引脚，它们组合的值一定不等于1111。因此，就可以判断是否有按键被按下。

串口1通信实例2
---矩阵按键结构及检测原理

- 驱动P0.3引脚为低/逻辑0，驱动P0.2、P0.1和P0.0引脚为逻辑1，即它们值的组合为0111，十六进制数7。
  - 当按下标号为0、1、2、4、5、6、8、9、A、C、D、E的按键时，P0.4~P0.7引脚的状态不会发生任何的变化。
  - 如果按下3号按键，则P0.4引脚被拉低，即：变化到逻辑状态0。而其他引脚状态仍然为逻辑高。此时如果读取这四个端口，读取的值应该是1110，十六进制数E分别对应于P0.7、P0.6、P0.5、P0.4引脚。
  - 如果按下7号按键，则P0.5引脚被拉低，即：变化到逻辑状态0。而其他引脚状态仍然为逻辑高。此时如果读取这四个端口，读取的值应该是1101，十六进制数D分别对应于P0.7、P0.6、P0.5、P0.4引脚。

串口1通信实例2
---矩阵按键结构及检测原理

- 如果按下11（B）号按键，则P0.6引脚被拉低，即：变化到逻辑状态0。而其他引脚状态仍然为逻辑高。此时如果读取这四个端口，读取的值应该是1011，十六进制数B分别对应于P0.7、P0.6、P0.5、P0.4引脚。
- 如果按下15（F）号按键，则P0.7引脚被拉低，即：变化到逻辑状态0。而其他引脚状态仍然为逻辑高。此时如果读取这四个端口，读取的值应该是0111，十六进制数7分别对应于P0.7、P0.6、P0.5、P0.4引脚。

串口1通信实例2
---矩阵按键结构及检测原理

- 驱动P0.2引脚为低/逻辑0，驱动P0.3、P0.1和P0.0引脚为1，即它们值的组合为1011，十六进制数B。
  - 当按下标号为0、1、3、4、5、7、8、9、B、C、D、F的按键时，P0.4~P0.7引脚的状态不会发生任何的变化。
  - 如果按下2号按键，则P0.4引脚被拉低，即：变化到逻辑状态0。而其他引脚状态仍然为逻辑高。此时如果读取这四个端口，读取的值应该是1110，十六进制数E分别对应于P0.7、P0.6、P0.5、P0.4引脚。
  - 如果按下6号按键，则P0.5引脚被拉低，即：变化到逻辑状态0。而其他引脚状态仍然为逻辑高。此时如果读取这四个端口，读取的值应该是1101，十六进制数D分别对应于P0.7、P0.6、P0.5、P0.4引脚。

串口1通信实例2
---矩阵按键结构及检测原理

- 如果按下10（A）号按键，则P0.6引脚被拉低，即：变化到逻辑状态0。而其他引脚状态仍然为逻辑高。此时如果读取这四个端口，读取的值应该是1011，十六进制数B分别对应于P0.7、P0.6、P0.5、P0.4引脚。
- 如果按下14（E）号按键，则P0.7引脚被拉低，即：变化到逻辑状态0。而其他引脚状态仍然为逻辑高。此时如果读取这四个端口，读取的值应该是0111，十六进制数7分别对应于P0.7、P0.6、P0.5、P0.4引脚。

串口1通信实例2
---矩阵按键结构及检测原理

- 驱动P0.1引脚为低/逻辑0，驱动P0.3、P0.2和P0.0引脚为1，即它们值的组合为1101，十六进制数D。
  - 当按下标号为0、2、3、4、6、7、8、A、B、C、E、F的按键时，P0.4~P0.7引脚的状态不会发生任何的变化。
  - 如果按下1号按键，则P0.4引脚被拉低，即：变化到逻辑状态0。而其他引脚状态仍然为逻辑高。此时如果读取这四个端口，读取的值应该是1110，十六进制数E分别对应于P0.7、P0.6、P0.5、P0.4引脚。
  - 如果按下5号按键，则P0.5引脚被拉低，即：变化到逻辑状态0。而其他引脚状态仍然为逻辑高。此时如果读取这四个端口，读取的值应该是1101，十六进制数D分别对应于P0.7、P0.6、P0.5、P0.4引脚。

串口1通信实例2
---矩阵按键结构及检测原理

- 如果按下9号按键，则P0.6引脚被拉低，即：变化到逻辑状态0。而其他引脚状态仍然为逻辑高。此时如果读取这四个端口，读取的值应该是1011，十六进制数B分别对应于P0.7、P0.6、P0.5、P0.4引脚。
- 如果按下13（D）号按键，则P0.7引脚被拉低，即：变化到逻辑状态0。而其他引脚状态仍然为逻辑高。此时如果读取这四个端口，读取的值应该是0111，十六进制数7分别对应于P0.7、P0.6、P0.5、P0.4引脚。

串口1通信实例2
---矩阵按键结构及检测原理

- 驱动P0.0引脚为低/逻辑0，驱动P0.3、P0.2和P0.1引脚为1，即它们值的组合为1110，十六进制数E。
  - 当按下标号为1、2、3、5、6、7、9、A、B、D、E、F的按键时，P0.4~P0.7引脚的状态不会发生任何的变化。
  - 如果按下0号按键，则P0.4引脚被拉低，即：变化到逻辑状态0。而其他引脚状态仍然为逻辑高。此时如果读取这四个端口，读取的值应该是1110，十六进制数E分别对应于P0.7、P0.6、P0.5、P0.4引脚。
  - 如果按下4号按键，则P0.5引脚被拉低，即：变化到逻辑状态0。而其他引脚状态仍然为逻辑高。此时如果读取这四个端口，读取的值应该是1101，十六进制数“D”分别对应于P0.7、P0.6、P0.5、P0.4引脚。

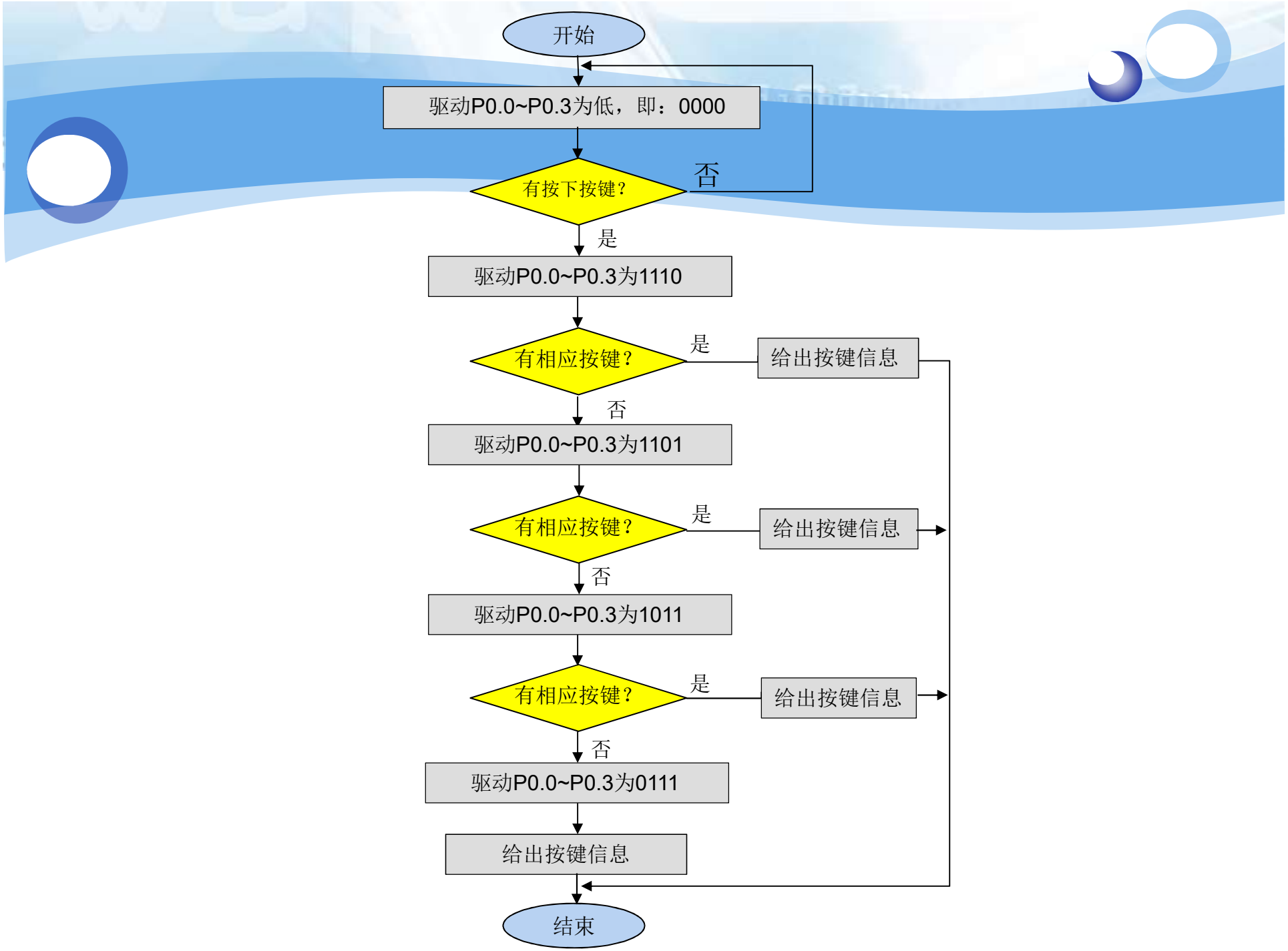
串口1通信实例2
---矩阵按键结构及检测原理

- 如果按下8号按键，则P0.6引脚被拉低，即：变化到逻辑状态0。而其他引脚状态仍然为逻辑高。此时如果读取这四个端口，读取的值应该是1011，十六进制数B分别对应于P0.7、P0.6、P0.5、P0.4引脚。
- 如果按下12（C）号按键，则P0.7引脚被拉低，即：变化到逻辑状态0。而其他引脚状态仍然为逻辑高。此时如果读取这四个端口，读取的值应该是0111，十六进制数7分别对应于P0.7、P0.6、P0.5、P0.4引脚。

串口1通信实例2
---矩阵按键结构及检测原理

所谓的扫描就是让P0.0、P0.1、P0.2和P0.3的驱动值快速的在0111、1011、1101、1110之间进行变化

- 这样就能在按下按键的时候，知道按下那个按键。


开始
驱动P0.0~P0.3为低，即：0000
有按下按键？
否
是
驱动P0.0~P0.3为1110
有相应按键？
是
给出按键信息
否
驱动P0.0~P0.3为1101
有相应按键？
是
给出按键信息
否
驱动P0.0~P0.3为1011
有相应按键？
是
给出按键信息
否
驱动P0.0~P0.3为0111
给出按键信息
结束

串口1通信实例2
---串口1参数设置

串口1参数设置

- 在该设计中串口1工作在模式1下，使用定时器1模式0（16位自动重加载）作为串口1的波特率发生器。

串口1通信实例2
--设计代码和分析

【例】 STC学习板上按键通过串口显示在主机上C语言描述的例子。

```
#include "reg51.h"
#define FOSC  18432000L                    //声明当前单片机主时钟频率
#define BAUD 115200                        //声明波特率常数115200
sfr AUXR  =0x8E;                           //声明AUXR寄存器的地址
bit busy=0;                                //声明bit型变量
xdata char menu[]={"\r\n--Display Press buttons information--\r\n"};
                                           //声明字符数组menu
```

串口1通信实例2
--设计代码和分析

```
void IO_KeyDelay(void)                          //声明IO_KeyDelay子函数，延迟
{
        unsigned char i;
        i = 60;
        while(--i)          ;
}
void SendData(unsigned char dat)                //声明SendData子函数
{
        while(busy);                            //判断是否发送完，没有则等待
        SBUF=dat;                               //否则，将数据dat写入SBUF寄存器
        busy=1;                                 //将busy置1
}
```

串口1通信实例2
--设计代码和分析

```
void SendString(char *s)                    //声明SendString子函数
{
        while(*s!='\0')                      //判断是否是字符串的结尾
        SendData(*s++);                      //如果没有结束，调用SendData发送数据
}
void uart1() interrupt 4                    //声明uart串口1中断服务程序
{
        if(RI)                               //通过RI标志，判断是否接收到数据
                RI=0;                        //如果RI为1，则软件清零RI
        if(TI)                               //通过TI标志，判断是否发送完数据
                TI=0;                        //如果TI为1，则软件清零TI
         busy=0;                             //将busy标志清零
}
```

串口1通信实例2
--设计代码和分析

```
void main()
{
    unsigned char c1_new,c1_old=0,c1;        //声明字符型变量
    SCON=0x50;                               //串口1模式1，使能串行接收
    AUXR=0x40;                               //定时器1不分频，作为串口1波特率时钟
    TL1=(65536-((FOSC/4)/BAUD));             //定时器1初值计数器低8位
    TH1=(65536-((FOSC/4)/BAUD))>>8;          //定时器1初值计数器高8位
    TR1=1;                                   //使能定时器1工作
    ES=1;                                    //允许串口1中断
   EA=1;                                     //CPU允许响应中断请求
```

串口1通信实例2
--设计代码和分析

```
SendString(&menu);              //在串口调试界面中打印字符串信息
while(1)
 {                              //无限循环
      P0=0xF0;                  //将P0.0~P0.3拉低，在读P0.4~P0.7前，发‘F’
      IO_KeyDelay();            //延迟读
      c1_new=P0&0xF0;           //得到矩阵按键的信息
```

串口1通信实例2
--设计代码和分析

```
if(c1_new!=c1_old)                  //如果新按键和旧按键状态不一样，则继续
    {
        c1_old=c1_new;              //把新按键的状态变量保存作为旧的按键
        if(c1_new!=0xF0)            //如果有按键按下，继续
        {
            P0=0xFE;                //将P0[3-0]置“1110”，在读P0.4~P0.7前，发‘F’
            IO_KeyDelay();          //延迟读
            c1_new=P0;              //获取P0端口的值
            switch (c1_new)
              {
```

串口1通信实例2
--设计代码和分析

```
case 0xee: c1=0;  break;      //如果值为0xee，则表示按下0号按键
case 0xde:  c1=4;  break;     //如果值为0xde，则表示按下4号按键
case 0xbe:  c1=8;  break;     //如果值为0xbe，则表示按下8号按键
case 0x7e:  c1=12;  break;    //如果值为0x7e，则表示按下12号按键
default  : ;
}
P0=0xFD;                  //将P0[3-0]置“1101”，在读P0.4~P0.7前，发‘F’
IO_KeyDelay();                //延迟读
c1_new=P0;                    //获取P0端口的值
switch (c1_new)
 {
```

串口1通信实例2
--设计代码和分析

```
switch (c1_new)
 {
  case 0xed: c1=1; break;        //如果值为0xed，则表示按下1号按键
   case 0xdd: c1=5; break;       //如果值为0xdd，则表示按下5号按键
   case 0xbd: c1=9; break;       //如果值为0xbd，则表示按下9号按键
   case 0x7d: c1=13; break;      //如果值为0x7d，则表示按下13号按键
   default  : ;
 }
 P0=0xFB;                  //将P0[3-0]置“1011”，在读P0.4~P0.7前，发‘F’
 IO_KeyDelay();                  //延迟读
 c1_new=P0;                      //获取P0端口的值
```

串口1通信实例2
--设计代码和分析

```
switch (c1_new)
 {
  case 0xeb: c1=2; break;       //如果值为0xeb，则表示按下2号按键
  case 0xdb: c1=6; break;       //如果值为0xdb，则表示按下6号按键
  case 0xbb: c1=10; break;      //如果值为0xbb，则表示按下10号按键
  case 0x7b: c1=14; break;      //如果值为0x7b，则表示按下14号按键
  default  : ;
 }
P0=0xF7;                  //将P0[3-0]置“0111”，在读P0.4~P0.7前，发‘F’
 IO_KeyDelay();                 //延迟读
 c1_new=P0;                     //获取P0端口的值
```

串口1通信实例2
--设计代码和分析

```
switch (c1_new)
 {
  case 0xe7: c1=3; break;      //如果值为0xe7，则表示按下3号按键
  case 0xd7: c1=7; break;      //如果值为0xd7，则表示按下7号按键
  case 0xb7: c1=11;break;      //如果值为0xb7，则表示按下11号按键
  case 0x77: c1=15; break;     //如果值为0x77，则表示按下15号按键
   default  : ;
 }
 SendString("\r\n press #"); //发送字符串信息
```

串口1通信实例2
--设计代码和分析

```
 if(c1<10)                          //如果按键变量小于10，即：0~9
      SendData(c1+0x30);    //转换为对应的ASCII，调用SendData发送
else if(c1==10)                    //如果按键值为10
      SendString(“10”);     //调用SendString函数，发送字符串“10”
else if(c1==11)                    //如果按键值为11
      SendString(“11”);     //调用SendString函数，发送字符串“11”
else if(c1==12)                    //如果按键值为12
     SendString(“12”);      //调用SendString函数，发送字符串“12”
else if(c1==13)                    //如果按键值为13
    SendString(“13”);       //调用SendString函数，发送字符串“13”
else if(c1==14)      //如果按键值为14
```

串口1通信实例2
--设计代码和分析

```
                SendString(“14”);      //调用SendString函数，发送字符串“14”
            else if(c1==15)                    //如果按键值为15
                SendString(“15”);      //调用SendString函数，发送字符串“15”
                SendString(" button\r\n"); //调用SendString函数，发送字符串
        }
    }
}
}
```

串口1通信实例2
--设计代码和分析

下面说明该代码的设计原理和验证方法，步骤包括：

- 使用T1定时器，根据前面给出的IRC的时钟频率为18.432MHz，波特率为115200，由于，T1的溢出率和波特率存在下面的关系，即：

  串口1的波特率=SYSclk/(65535-[RL_TH1,RL_TL1]/4

  因此，[RL_TH1,RL_TL1]=65536-SYSclk/(串口1波特率×4)

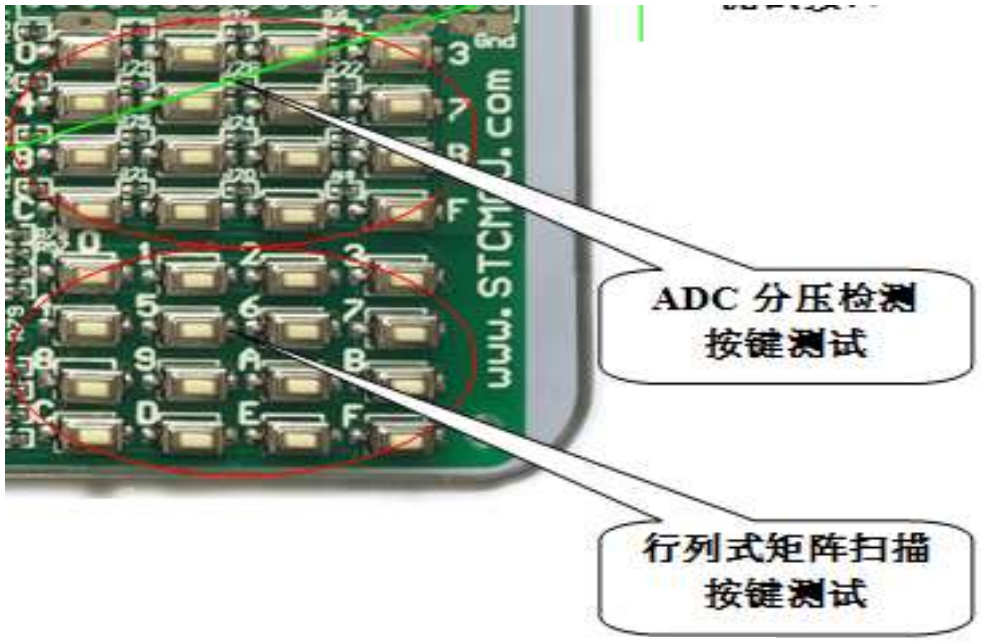
- 打开STC-ISP软件，在该界面内，选择硬件选项。将“输入用户程序运行时的IRC频率设置为18.432MHz。
- 单击下载/编程按钮，按前面的方法下载设计到STC单片机。

串口1通信实例2
--设计代码和分析

- 在STC-ISP软件右侧串口中，选择串口助手标签。在该标签串口界面下，按下面设置参数：
  - 串口：COM3（读者根据自己电脑识别出来的COM端口号进行设置）
  - 波特率：115200。
  - 校验位：无校验。
  - 停止位：1位。
- 单击打开串口按钮。
- 在STC学习板上，找到并按一下SW19按键，重新运行程序。可以看到在上面的接收窗口中，显示出提示信息“—Display Press buttons information—。

串口1通信实例2
--设计代码和分析

- 在STC学习板上右下角的位置，找到矩阵按键。



STC学习板上矩阵按键的位置

串口1通信实例2
--设计代码和分析

- 每次按下一个矩阵键盘中的一个按键，可以看到串口调试助手上显示按键信息。