MFRC522 IC 卡模块使用手册 V1.0

嵌达科技，快乐生活

欢迎访问：http://shop109772519.taobao.com/shop/view_shop.htm?tracelog=twddp

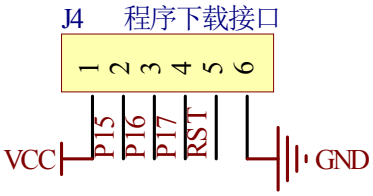
程序功能：

当 IC 卡放在模块上时，先把预先赋值给 Write_Data[16]这个数组中的数据写到 IC 卡中，然后马上把写入的数据读出来，显示在 PC 机的串口调试助手上。

操作步骤：

1、 下载程序到单片机中。

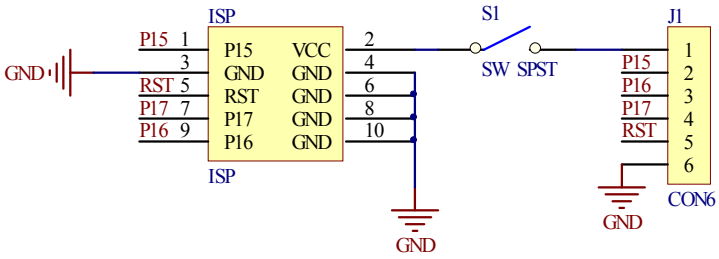
我们设计的电路板使用的是 AT89S52，因此只能通过 ISP 下载，为了布线方便，我只引出了必须的 6 根线供下载，图如下：



程序下载接口

我们都知道 ISP 下载器都是 10 脚的，从 10 脚转变成 6 脚就必须有所转换，一般有两种解决方法：1）通过杜邦线一对一连，这种方法的前提是你必须知道自己手上的 ISP 下载器引脚定义；2）通过另一块转接板把 10 脚转换为 6 脚，并附加上开关功能，这种方法比较方便，如下图：

由于此转换电路板制作费用较小，如果您买了 ISP 下载器或者买了读写卡



模块，我们将免费赠送。

2、 准备工作

1 将读写卡模块插在电路板上；

2 将串口线连接电路板和电脑，保证可以实现串口通信的硬件连接；

3 打开 PC 机上的串口调试助手，并打开对应端口，开启 16 进制显示，准备接收数据。

3、 实现读写卡

1 给电路板通电；

2 把卡放在读卡器模块上，当绿灯亮时表示读写卡结束，此时串口调试助

手会显示出卡对应数据块中的数据，如下图：

1、 工程文件说明：

我们的程序只用了四个文件，分别为：读写卡.c、read_card.h、read_card.c 和 rc522.h。下面我来介绍一下这四个文件的作用，各位亲，这部份仔细看哦，尤其是那些还执着于在一个文件中实现所的函数编写的朋友们，下面我们所展示的文件架构可以适用于小中型工程的实现，这会让您的程序看上去更有条理性和逻辑性，更容易调试和修改程序，这对于您自身编程能力的提升是无往不利的。如果您要实现大型的工程，就需要考虑更多的文件存放各种类别的程序了。

读写卡.c：这个文件是我们的 main()函数所在的文件，是我们整个工程的入口点，这个文件中包含了串口通信初始化函数的调用和读写卡函数的调用，进而牵动所有函数的调用。

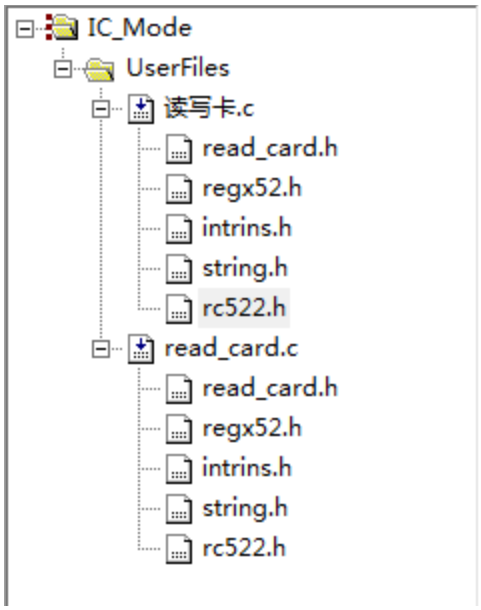
Read_card.h：这个文件是我们除 main()函数之外所有的函数定义的地方，只有在这里定义函数，这些函数才能在别的文件中被使用。另外，这个文件中统一定义了本工程所要用到的所有系统文件，比如说：regx52.h、intrins.h 等。

Read_card.c：如果说 read_card.h 这个文件实现了所有函数的定义，read_card.c 这个文件则实现了所有函数的编写。从很简单的延时函数到复杂的寄存器操作函数，从单一功能的实现函数到完整读写功能的实现函数，全都会体现在这里。

Rc522.h：MFRC522 读写模块上有一个芯片实现了对读写卡功能的控制，这个芯片和我们日常使用的 51、stm、stc 之类的芯片是一样的，都有内部控制寄存器，只有对这些寄存器实现程序中的定义才能进行使用，rc522.h 这个文件就是实现了这个功能，这有点类似于 51 单片机的 regx51.h 这个文件。当然，我们对单片机引脚的定义也可以在这个文件中实现。

当然，我们在实际使用中会看到所有用到的文件，很多都是系统自带的文件，架构如下图，很多人看到这么多的文件会感到很头疼，不知道如何理清思路，各位亲，千万不要被表象所迷惑哦，我们真实所编辑的只有 4 个文件而已，

而且你一旦掌握了这个编写程序的方式，可以使用到任何工程中，使你的作品看上去层次分明，你将会永远告别在一个文件中查找、修改子函数的痛苦。



2、硬件说明：

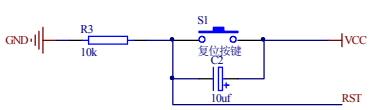
程序是依赖硬件编写的，这是嵌入式系统的一大特色，只有熟悉硬件的工作方式才能继续编写程序。因此，在讲解程序之前很有必要先讲讲硬件的组成。

我们设计的硬件电路分类很多，本次讲解的程序所依赖的硬件很简单，实现的功能只是对模块实现读写控制，并把卡里的数据显示在串口调试助手上。

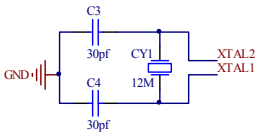
按照功能来分，我们把硬件分为 6 个部分：

1　51 单片机最小系统

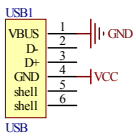
这个是所有嵌入式系统的基础配置，是供单片机工作的最小资源，分为电源电路、时钟电路和复位电路，这些自然不必多说，电路设计如下图：



复位电路



时钟电路

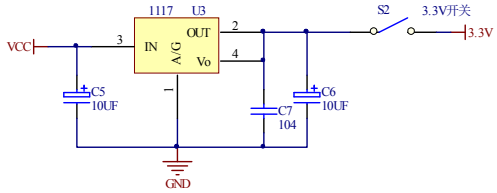


USB供电

2　供电模块

我们单片机用的是 5V 电压供电，这部分电源我们可以用 220V 交流转 5V 直流适配器来实现，但最常见的是用电脑的 USB 口来供电，方便快捷。但是我们还需要考虑到另外一个问题，那就是 MFRC522 模块是 3.3V 供电，这就要用到电压转换电路来实现了。我们常用到的电路如下图：

这个电路图摘自 ASM1117-3.3 这个电压转换芯片的芯片说明文档中，这个文档中根据各种不同的应用给出了不同的使用电路，已经能够满足我们一般的项目所需。



5V转3.3V

3　程序下载接口

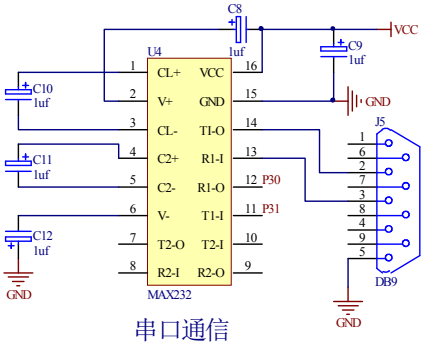
我们设计的板子是用 AT89S52 当作主控芯片的，而且为了减少制版面积，我们采用 TFQP 贴片式封装的芯片，如下图：

针对 AT89S52 系列的单片机，我们通常用的都是 ISP 下载器给芯片下载程序。下载接口的说明在上面操作方法中已经讲过，这里就不再赘述了。

4　串口通信电路

我们要给串口调试助手发送数据，就必须要先设计串口通信电路，使用的最广泛的就是以电平转换芯片 MAX232 为中心的 RS232 通信电路。废话不多说，电路如下：

这个电路摘自 MAX232 芯片使用文档中，也在实践中被证明可以工作的很

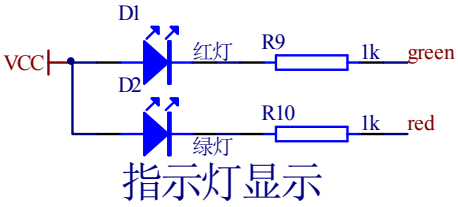


串口通信

好，其中 C12、C8 不要认为极性接反了哦。偷偷地告诉你，虽然在 MAX232 官方文档里面要求使用 1uf 的电解电容，但经过实践证明，这个电路中所有的电容都可以用 0.1uf 的瓷片电容代替，对电路板面积有严格要求的亲们可要注意了哦，嘻嘻。

5　调试和运行状态接口

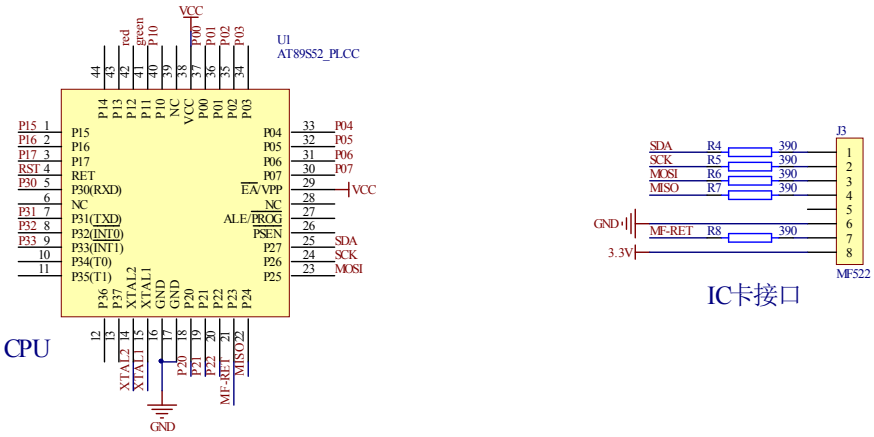
当我们在调试程序的过程中需要时时监控程序运行的状态，当在程序加上某个条件后，我们往往不能确定程序会不会进入某个函数中，这时候我们就要借助于 led 灯这种简单的程序运行状态指示器了。在电路中插上一两个 led 灯有莫大的好处，它不仅可以监控程序的运行状态，还可以为我们提供执行某个程序的信号，所以我非常建议朋友们在自己的电路中安排一两个 led 灯，我们电路中的 led 灯电路如下：



6　MFRC522 传感器模块接口

这部分电路是我们在设计程序中需要参考的，先看看电路图：

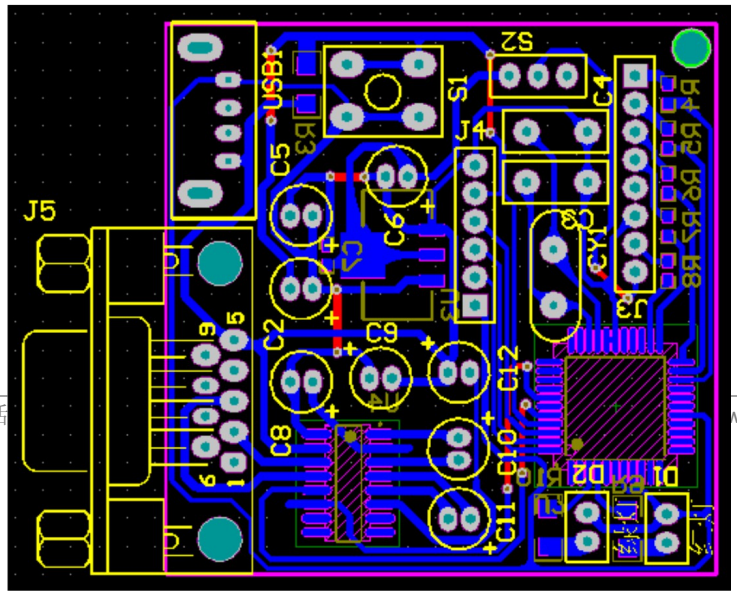
在程序中我们需要通过单片机控制 MFRC522 模块，因此我们必须知道两者

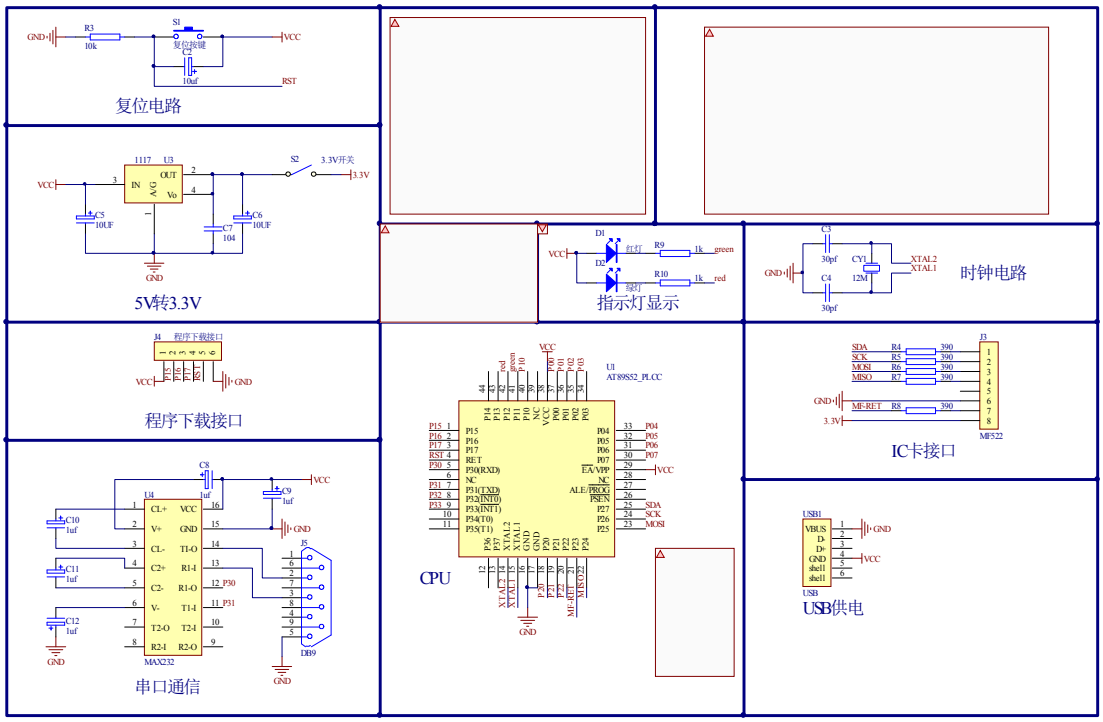


之间的硬件连接，由上图我们知道了控制模块锁所用到的引脚为：P2^3、P2^4、P2^5、P2^6、P2^7 这 5 个 IO 口，这部份接口定义在 rc522.h 中的最下面，至于如何实现控制，我们在程序讲解中聊哈。

硬件部分到这里就已经讲解得差不多了，但是各位亲们想必对整个硬件环境还没有一个整体的概念，下面我把整个电路图放出来，给大家一个总体的概念，当然，如果是嵌入式高手，经过我们上面的描述，一定对我们所用的硬件环境了如指掌了，因为这些电路实在太简单了。

这是我们的原理图，灰色的部分是编译屏蔽，您可以取消这些屏蔽，使得增加蜂鸣器和 Lcd1602 液晶显示功能。





这是我们画的 PCB 图，我们有按照这个电路图做出来的现成的电路板，所开发的程序都在这个电路板上调试出来的，这足以证明这个电路是可以使用的。

电路图获取：http://item.taobao.com/item.htm?id=38736244749&spm=2014.21279357.0.0

3、程序解剖

终于要讲到程序了，这可是我们嵌入式系统的精华部份哦。虽然我们硬件设计的很简单，但如果在程序上下一番功夫的话，这个小系统也将会有大作用，别忘了我们可是可以和电脑实现数据交互的，通过上位机程序编写，我们可以把这个系统用到方方面面。当然，我们得从最基础的程序开始学起，上位机的编写是以后的事。

我们先来看看我们所要用到的所有函数：

```
void delay_ns(unsigned int data ns); //纳秒级延时
unsigned char SPIReadByte(void);  // 读 SPI 数据
void SPIWriteByte(unsigned char data SPIData); // 写 SPI 数据
unsigned char ReadRawRC(unsigned char data Address);      //功能：读 RC632 寄存器
void WriteRawRC(unsigned char data Address, unsigned char data value);//功能：写 RC632 寄存器
void ClearBitMask(unsigned char data reg,unsigned char data mask);//功能：清 RC522 寄存器位
void SetBitMask(unsigned char data reg,unsigned char data mask); //功能：置 RC522 寄存器位
char PcdComMF522(unsigned char data Command,unsigned char *pInData,  unsigned char data InLenByte,
                 unsigned char *pOutData,unsigned int *pOutLenBit); //功能：通过 RC522 和 ISO14443 卡通讯
char PcdRequest(unsigned char data req_code,unsigned char *pTagType);//功能：寻卡
char PcdAnticoll(unsigned char *pSnr);   //功能：防冲撞
void CalulateCRC(unsigned char *pIndata,unsigned char data len,unsigned char *pOutData);//用 MF522 计算 CRC16 函数
char PcdSelect(unsigned char *pSnr)     ; //功能：选定卡片
char PcdAuthState(unsigned char data auth_mode,unsigned char data addr,unsigned char *pKey,unsigned char *pSnr);
//功      能：验证卡片密码
```

```
char PcdRead(unsigned char data addr,unsigned char *pData);//功能：读取 M1 卡一块数据
char PcdWrite(unsigned char data addr,unsigned char *pData);//功能：写数据到 M1 卡一块
char PcdHalt(void)    ;//功能：命令卡片进入休眠状态
char PcdReset(void);//功能：复位 RC522
unsigned char Write_to_Card(unsigned char data KuaiN,unsigned char *pData);
unsigned char Read_from_Card(unsigned char data KuaiN,unsigned char *pData);
//串口通讯
void init();
void delay_welcome(int ms);
void Serial_Rs232();
```

函数很多对不对？脑袋变大了对不对？如果我告诉你只需要知道一个函数的用法你就能把这些函数嵌入到你自己的系统中去，你会不会好一点。或许你不太相信我说的话，那么，让我们拭目以待吧。

有开发经验的朋友们都知道，从淘宝网上淘来的 MFRC522 会给一些例程，但并没有把程序所依赖的硬件描述的很清楚，使得我们拿到资料后无法轻易使用，庞大的代码使得我们很难嵌套入自己的系统中，这时我们该怎么办呢？一步一步分解代码吗？直到裁剪出适合自己的嵌入式系统的代码？这个工作量是巨大的，尤其是对那些开发经验不足的亲们，面对这种尴尬的情况，我们“嵌达科技”给大家提供了解决方法：给大家最简单的程序实现最基础的功能。告别裁剪，快捷嵌套。附件 1 是我们程序的函数关系，一起去看看吧。（附件 1）

从附件 1 中可以看出 Write_to_Card(…)和 Read_from_Card(…)可谓是所有函数的终点，而 SPIWriteByte(…)则是最底层对 MFRC522 模块进行操作的函数，所有函数都是为了 Write_to_Card(…)和 Read_from_Card(…)而产生的。因此，我们只需要调用这两个函数就可以完成 IC 卡的读写，同时，这两个函数的参数和使用方法都是一样的，是不是很简单！

虽然说可以直接把我们的程序拿来用应一下急，但作为嵌入式开发者，我们建议还是需要了解这些程序是怎么来的，下面我们就来讲讲程序的组成架构和函数的功能原理。在讲程序之前，我们需要将 IC 卡的结构给大家讲清楚，看下图：

扇区	块	描述
15	63	第 15 扇区尾块
	62	数据块
	61	数据块
	60	数据块
14	59	第 14 扇区尾块
	58	数据块
	57	数据块
	56	数据块
	⋮	
1	7	第 1 扇区尾块
	6	数据块
	5	数据块
	4	数据块
0	3	第 0 扇区尾块
	2	数据块
	1	数据块
	0	厂商标志块

上图是 IC 卡中的存储区结构。Mifare 卡片的存储容量为 8192×1 位字长（即 1K ＊ 8 位字长），采用 E2PROM 作为存储介质。整个结构划分为 16 个扇区，

编为扇区 0～15。每个扇区有 4 个块（Block），分别为块 0, 块 1, 块 2 和块 3。每个块有 16 个字节。一个扇区共有 16Byte * 4 = 64Byte。每个扇区的块 3（即第四块）也称作尾块，包含了该扇区的密码 A(6 个字节)、存取控制(4 个字节)、密码 B(6 个字节)。其余三个块是一般的数据块。

每个扇区的块 3（即第四块）也称作尾块，包含了该扇区的密码 A(6 个字节)、存取控制(4 个字节)、密码 B(6 个字节)。其余三个块是一般的数据块。扇区 0 的块 0 是特殊的块,包含了厂商代码信息，在生产卡片时写入，不可改写。其中：第 0～4 个字节为卡片的序列号，第 5 个字节为序列号的校验码；第 6 个字节为卡片的容量“SIZE”字节；第 7、8 个字节为卡片的类型号字节，即 Tagtype 字节；其他字节由厂商另加定义。

综上所述，我们可以用来存储数据的空间为第一扇区的第 1、2 块，第二扇区到第 63 扇区的第 0、1、2 块，简而言之就是我们上图中的“数据块”。

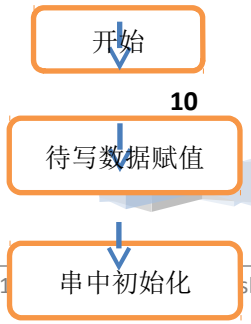
因此，只要做出一个接口，即设计一个函数，让我们指定要在哪个块上输入什么数据就好了，块的数据范围是 0 到 63，写入的数据的格式为 16 个字节，如：ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff，我们设定一个数组来存储这个数据：Write_Data[16] = {0x00};

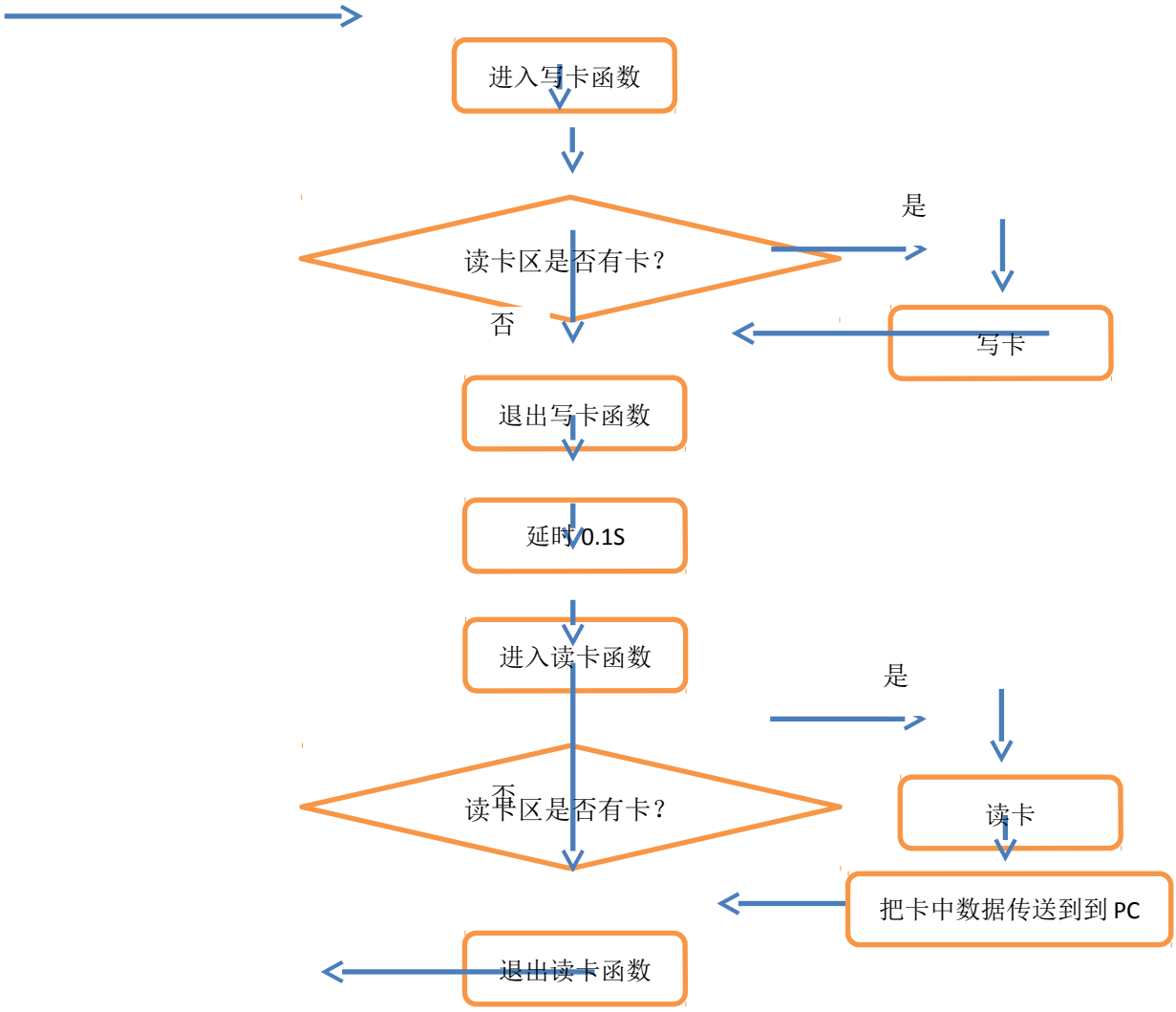
我们设计了一个函数：Write_to_Card(kuai,Write_Data)；其中，kuai 就是指定是哪个块，Write_Data 是指要写入的数据，就是上面所说数组的起始地址，只要事先对这两个变量进行赋值，就可以轻松向 IC 卡中写入数据了，我们设计了另一个函数 Read_from_Card(kuai,Read_Data)；kuai 是指要操作的块，Read_Data 就是从这个块中读取出来的数据的存放地，有了数据，不管是在 lcd1602 还是 lcd12864 或者是 tft 液晶显示都没什么问题了。

现在让我们看看 Write_to_Card(…)和 Read_from_Card(…)的使用方法：

```
void main()
{
    Write_Data[0] = 0x33;                    //需要写入 IC 卡中的数据赋值
    init();                    //串口通信初始化
    while(1)
    {
        Write_to_Card(0x01,Write_Data);  //向 IC 卡中的 0x01 块存储区中写数据
        delay_welcome(100);
        Read_from_Card(0x01,Read_Data);  //从 IC 卡中的 0x01 块存储区中读数据
    }
}
```

这是 Mian 函数程序，简单吧，但只是这么几句还不能让人理解其工作原理，下面我们来看看程序流程图。





读写卡的时间是短暂的，而人刷卡时，卡在读卡器读卡范围内的时间相对较长，这就涉及到一次读卡和重复读卡的问题了，我们的程序设计的是一次读卡，也就是说你不管把卡放在读卡器上多长时间，系统只读写一次卡，串口调试助手不会重复显示一样的数据，这个功能可不是所有淘宝店可以提供的哦。下面我们来看看程序：

```
//向 IC 卡中写数据
unsigned char Write_to_Card(unsigned char data KuaiN,unsigned char *pData)
{
    char data status;
    unsigned char idata RevBuffer[4];
    unsigned char data MLastSelectedSnr[4];
    unsigned char data PassWd[6]={0xff,0xff,0xff,0xff,0xff,0xff};
    unsigned char data place = 0x00;
    PcdReset();
    //寻天线区内未进入休眠状态的卡，返回卡片类型 2 字节
    status=PcdRequest(PICC_REQIDL,&RevBuffer[0]);
    if(status!=MI_OK)
    {
        flag = 0;
```

```
        green = 1;
        return 0x00;
    }
    //判断当操作卡完成后就停止重复读写卡
    if(flag == 1)
    {
        return 0x00;
    }
    //防冲撞，返回卡的序列号 4 字节
    status=PcdAnticoll(&RevBuffer[2]);
    if(status!=MI_OK)
    {
        return 0x00;
    }
    memcpy(MLastSelectedSnr,&RevBuffer[2],4); //拷贝序列号到变量 MLastSelectedSnr
中
    //选卡，卡号为 MLastSelectedSnr
    status=PcdSelect(MLastSelectedSnr);
    if(status!=MI_OK)
    {
        return 0;
    }
    //验证卡片密码，空白卡都是 0xff,0xff,0xff,0xff,0xff,0xff
    status=PcdAuthState(PICC_AUTHENT1A,KuaiN,PassWd,MLastSelectedSnr);
    if(status!=MI_OK)
    {
        return 0x00;
    }
    //向卡中指定块中写数据
    status=PcdWrite(KuaiN,pData);
    if(status!=MI_OK)
    {
        return 0;
    }
    //命令卡片进入休眠状态
    PcdHalt();
    flag = 1;
    green = 0;
    return 1;
}
//从 IC 卡中读数据
unsigned char Read_from_Card (unsigned char data KuaiN,unsigned char *pData)
{
    char data status;
```