齐 齐 哈 尔 大 学

毕业设计（论文）

题 目 基于ARM的伐木场安全监控系统设计

学 院 通信与电子工程学院

专业班级 通信工程152班

学生姓名 邓子成

指导教师 朱恒军

成 绩

2019年4月15日

# 摘 要

传统伐木场大多只生产原木，员工生活环境简陋，随着时代的发展，传统伐木场的生存空间越来越小，而现代伐木场在生产原木的基础上，对原木进行深度加工，所以，伐木场的体量和产品深度都在增加，相应的也会产生一下问题以及对社会提出一些需求。

现代伐木场一般面临作业园区的防火安全隐患、厂区库存量的多少、人员流动登记问题以及园区环境的基本状态问题等。本系统基于以上问题设计系统，系统能完成厂区24小时无缝火警检测、厂区所处环境的基本物理参数采集及显示功能、厂区库存量的自动检测以及厂区人员流动登记系统，以上信息都在设计的上位机管理系统中查看。

**关键字**：伐木场；防火安全；库存量；人员流动登记；上位机

# **Abstract**

Most of the traditional logging farms only produce logs, and the living environment of their employees is simple and crude. With the development of the times, the living space of the traditional logging farms is getting smaller and smaller, while the modern logging farms process logs deeply on the basis of the production of logs. Therefore, the volume and depth of products of the logging farms are increasing, which will also cause some problems and put forward some demands on the society.

Modern logging yards are generally faced with fire safety hazards in the operation park, the amount of stock in the factory, the registration of personnel flow and the basic state of the park environment. The system is designed based on the above problems. The system can complete 24-hour seamless fire detection in the factory area, collection and display of basic physical parameters of the environment in the factory area, automatic detection of inventory in the factory area and personnel flow registration system in the factory area. The above information is checked in the designed upper computer management system.

**Keywords** **Urban Scale;** Woodcutting grounds; Fire safety; Inventory; Registration of movement of personnel; Upper computer

目 录

[摘 要 I](#_Toc6985817)

[**Abstract** II](#_Toc6985818)

[第1章 绪 论 1](#_Toc6985819)

[1.1 课题研究背景 1](#_Toc6985820)

[1.2 国内外研究现状 1](#_Toc6985821)

[1.3 课题研究内容 1](#_Toc6985822)

[1.4 文本结构 2](#_Toc6985823)

[第2章 总体设计方案 2](#_Toc6985824)

[2.1 总体结构设计 2](#_Toc6985825)

[2.2 硬件结构选择 3](#_Toc6985826)

[2.2.1 主控选择 3](#_Toc6985827)

[2.2.2 环境监测传感器的选择 4](#_Toc6985828)

[2.2.3 RFID门禁模块选择 7](#_Toc6985829)

[2.3 软件结构设计 9](#_Toc6985830)

[2.3.1 硬件电路程序设计软件选择 9](#_Toc6985831)

[2.3.2 上位机设计软件选择 10](#_Toc6985832)

[第3章 系统硬件设计 11](#_Toc6985833)

[3.1 STM32ZET6最小系统电路设计 11](#_Toc6985834)

[3.2 供电电路设计 12](#_Toc6985835)

[第4章 系统软件设计 14](#_Toc6985836)

[4.1 主控程序设计 14](#_Toc6985837)

[4.2 通信程序设计 18](#_Toc6985838)

[4.3 传感器通信程序设计 20](#_Toc6985839)

[4.3.1 门禁控制程序设计 20](#_Toc6985840)

[4.3.2 土壤温湿度、大气温湿度及雨滴检测程序设计 22](#_Toc6985841)

# 第1章 绪 论

## 1.1 课题研究背景

在七届二中全会中，党中央提出“科学发展观”走可持续发展的道路，因为林业资源生长周期慢，所以传统伐木场已经不再适合这个时代，在建设现代伐木场的同时，无论是伐木场本身的需求还是消费市场的需要，现代伐木场都需要采用自动化控制，相应的就对社会提出了伐木场监控系统的需求。

首先作为伐木场主，其需要实时的了解厂区的情况，在安全生产的需求中，作业园区的防火异常重要，如果指派安全员监测，既浪费劳动力，又不能保证安全员能实时监测整个厂区。其次，随着中国公路网的建立，伐木场的市场也会越来越大，在进行对外销售的同时，能及时的知道厂区的货物库存量也至关重要，另外，随着伐木场体量的增大，人员流动也会增加，所以，作为厂主，其需要一个系统，在保证实时监测厂区情况的前提下，能在情况异常时报警，并且还能对库存进行实时监测，以及对厂区大量流动人员的监测和登记。

## 1.2 国内外研究现状

安全生产是现代工业生产的基本前提，西方发达国家在上世纪60年代就建立了安全生产信息化的系统，至今已有四代产品，比如美国的伐木场，现在都配套有相应的安全管理系统，伐木场的系统连接当地警局及消防局，在伐木场出现安全问题后，第一时间停止政府机构。并且每个伐木场的信息管理系统还可以排除隐患，将隐患消灭在萌芽中。此外，信息管理系统还可以帮助厂主合理的分配劳动力，大幅提高生产效率。

国内相应的伐木场安全检测系统起步于上世纪80年代，如今已经得到广泛的运用，随着新技术产生、新型传感器的问世已经电子通信的发展，现代伐木场安全监控系统已经有了长足的发展，在获得高稳定性、高可靠性、高精度已经高分辨率及实用性的基础上，已单片机及微处理器为核心，结合多种传感器的检测系统在获得软件的支持上，已经深入现代伐木场生产中的方方面面。

## 1.3 课题研究内容

本文主要基于现代伐木场的基本需求设计系统，主要解决如下问题：

1：作业区防火监控及视频监控

2：上位机综合监控系统

3：园区环境监控系统

4：报警及自动控制电路系统设计

5：门禁登录系统设计

这其中涉及到AD转换的知识、单片机定时器及中断的设计和使用问题、摄像头控制的问题、传感器配置的问题以及上位机的设计问题等。

## 1.4 文本结构

论文共6章，第一章介绍研究课题的价值和主要研究内容，第二章主要介绍系统的总体设计方案，第三四章主要介绍系统的硬件电路设计及软件设计的具体设计思路及方案，第五章为系统测试，介绍系统运行中碰到的问题及运行结果，最后第六章为结论、参考文献及致谢。

# 第2章 总体设计方案

## 2.1 总体结构设计

本设计基于STM32的各种内置功能，通过C语言的语法及逻辑语句，进行传感器的无线组网布置，传感器的实时监测配置，门禁管理功能设计以及人机交互电路系统设计。

主控

上位机

人员登记

串口通信

火源检测

环境检测

图2-1 基于ARM的伐木场安全监控系统设计基本框架

## 2.2 硬件结构选择

### 2.2.1 主控选择

本设计的主控模块为STM32ZET6最小系统板，STM32是意法半导体公司专为要求高性能、低成本、低功耗的嵌入式应用设计的主控芯片，内核为ARM Cortex®-M0，M0+，M3, M4和M7内核。其中意法半导体已经推出STM32基本型系列、增强型系列、USB基本型系列、互补型系列；新系列产品沿用增强型系列的72MHz处理频率。内存包括64KB到256KB闪存和 20KB到64KB嵌入式SRAM。新系列采用LQFP64、LQFP100和LFBGA100三种封装，不同的封装保持引脚排列一致性，结合STM32平台的设计理念，开发人员通过选择产品可重新优化功能、[存储器](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E5%99%A8)、性能和[引脚](https://baike.baidu.com/item/%E5%BC%95%E8%84%9A)数量，以最小的硬件变化来满足个性化的应用需求。

本系统采用的系统板扩展的基本功能如下表：

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **功能** | **名称** | **功能** |
| **芯片型号** | STM32ZET6 | **12路DAC** | 2路 |
| **主频** | 72MHz | **定时器** | 8个 |
| **内核** | Cortex®-M3 | **CAN接口** | 1个 |
| **工作电压** | 2.7V-3.6V | **SPI接口** | 3个 |
| **工作温度** | -40℃-105℃ | **IIC接口** | 2个 |
| **SRAM** | 64KB | **IIS接口** | 2个 |
| **内部flash** | 512KB | **USART** | 3个 |
| **12位ADC** | 21路 | **UART** | 2个 |
| **USB设备** | 支持 | **RTC** | 支持 |

由表易知，本系统采用的开发板基本上将STM32ZET6的可调用引脚都已引出供开发者使用，STM32内部提供了8个定时器，开发者可以使用这些定时器对外部脉冲进行计数以及可以通过配置预分频寄存器和配置自动重装载寄存器来输出PWM等等。另外STM32ZET6内部还集成了12路的ADC和DAC，开发者在对外部模拟输入和对外输出模拟值时不再需要设计AD转换电路和DA转换路，大大减少了系统开发的繁杂程度。同时STM32ZET6还已经集成了多种通信接口，例如CAN通信接口，它的设计目标是，以最小的CPU负荷来高效处理大量收到的报文。它也支持报文发送的优先级要求，优先级特效可以通过软件进行配置、还有SPI通信接口，SPI接口全称为串行外设接口，允许STM32与外部设备以半/全双工、同步、串行方式通信，另外此接口还可以被配置为主模式，并为外部从设备提供通信时钟SCK等等。

本系统使用的开发板实物图如下图：



图2-2 本系统采用的STM32ZET6最小系统开发板

### 2.2.2 环境监测传感器的选择

根据系统预设的功能，环境监测的主要参数有：

1. 伐木场温湿度监测
2. 伐木场下雨监测
3. 伐木场火警监测

1：伐木场温湿度监测

本系统对蔬菜生长环境的温湿度监测主要采用DHT11传感器，DHT11是一款有已校准数字信号输出的温湿度传感器。[数字温湿度传感器](https://baike.baidu.com/item/%E6%95%B0%E5%AD%97%E6%B8%A9%E6%B9%BF%E5%BA%A6%E4%BC%A0%E6%84%9F%E5%99%A8)是一款含有已校准数字信号输出的温湿度复合传感器，它应用专用的数字模块采集技术和温湿度传感技术，确保产品具有极高的可靠性和卓越的长期稳定性。传感器包括一个电阻式感湿元件和一个NTC测温元件，并与一个高性能8位单片机相连接。因此该产品具有品质卓越、超快响应、抗干扰能力强、性价比极高等优点。每个DHT11传感器都在极为精确的湿度校验室中进行校准。校准系数以程序的形式存在OTP内存中，传感器内部在检测信号的处理过程中要调用这些校准系数。单线制串行接口，使系统集成变得简易快捷。超小的体积、极低的功耗，使其成为该类应用中，在苛刻应用场合的最佳选择。产品为4针单排引脚封装，连接方便。

产品主要参数：

1. 湿度精度：+-5%RH
2. 温度精度：+-2℃
3. 湿度量程：20-90%RH
4. 温度量程：0~50℃

其实物图如下图所示：



图2-5 DHT11温湿度传感器实物图

其工作电路图如下图所示：

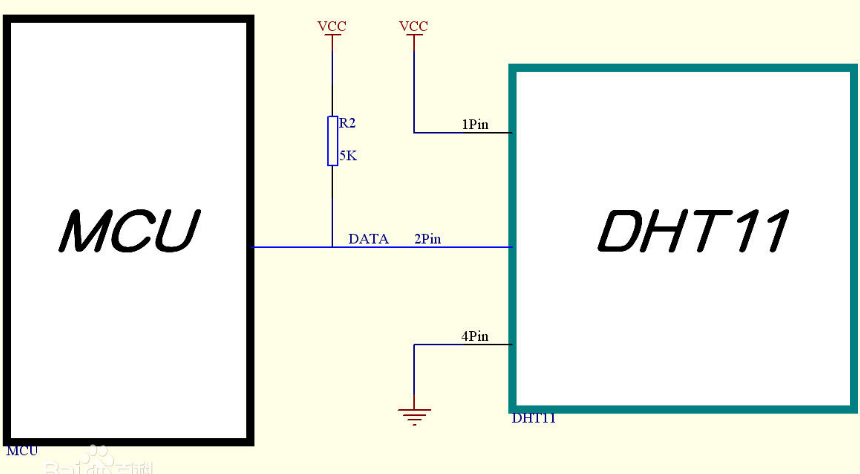


图2-6 DHT11与单片机连接电路图

由电路图易知，DHT11本身引脚不多，只有4个引脚，其中两个为电源端，剩下两个引脚，其中一个引脚悬空不接，另外一个引脚为数据输出端，直接接单片机的IO口，通过对单片机进行程序的编写，来实现单片机读取DHT11发送的数据，并通过一定的算法对读取的数据进行运算，从而得到我们需要的数据。

2：伐木场下雨监测

本系统对雨滴检测主要通过高灵敏的雨水检测模块来实现，Water Sensor是一个利用湿度方式检测有没有水分的传感器，可用于天气状况的监测，可以检测有没有下雨。将传感器放置在室外，当有雨水滴到检测面上时，湿度将增大，随之电压升高，并转成数字信号DO和模拟信号AO输出。除了检测雨水外，雾水和水蒸气也都能检测。传感器采用高品质FR-04双面材料，大面积5.0\*4.0CM,并用镀镍处理表面，具有对抗氧化，导电性，及寿命方面更优越的性能。使用宽电压LM393比较器，信号干净，波形好，驱动能力强，超过15mA。配电位器调节灵敏度。设有固定螺栓孔，方便安装。

其工作参数为：

1. 工作电压：3.3V-5V DC

其实物图如下图所示：



图2-7 雨滴检测传感器实物图

使用时，将其接上5V电源，电源指示灯亮，感应板上没有水滴时，DO 输出为高电平，开关指示灯灭 ，滴上一滴水，DO 输出为低电平，开关指示灯亮，刷掉上面的水滴，又恢复到输出高电平状态。AO 模拟输出，可以连接单片机AD口检测滴在上面的雨量大小。DO TTL数字输出也可以连接单片机检测是否有雨。TTL输出有效信号为低电平.驱动能力100MA左右，可直接驱动继电器，蜂鸣器，小风扇等。

其电路原理图如下图所示：

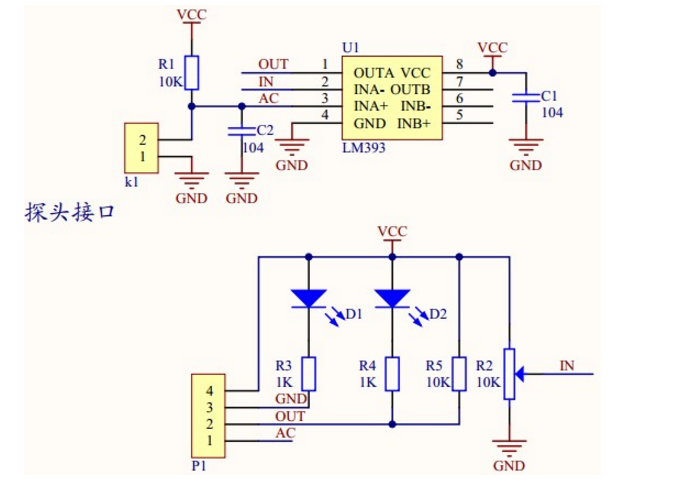


图2-8 雨滴检测传感器电路原理图

3：伐木场火警监测

本系统检测火警主要采用Risym火源探测模块，其工作原理主要通过红外检测，当接收到火焰的特殊波长后，将检测信号发给单片机，来实现火灾报警的作用。

其主要工作参数如下：

1. 检测距离：0-100CM
2. 触发光线波长：760nm-1100nm
3. 输出：比较输出，使用LM393比较器，灵敏度可调，输出数字量
4. 工作电压：3.3V-5V

当传感器检测到火源后，发给单片机高电平，然后单片机开始启动自动控制电路和报警电路。

其实物图及实物连接图如下：

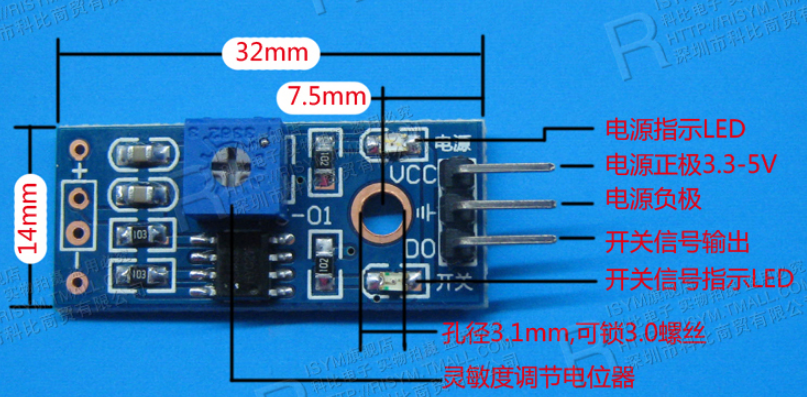


图2-9 火源检测传感器实物图

其电路原理图如下：

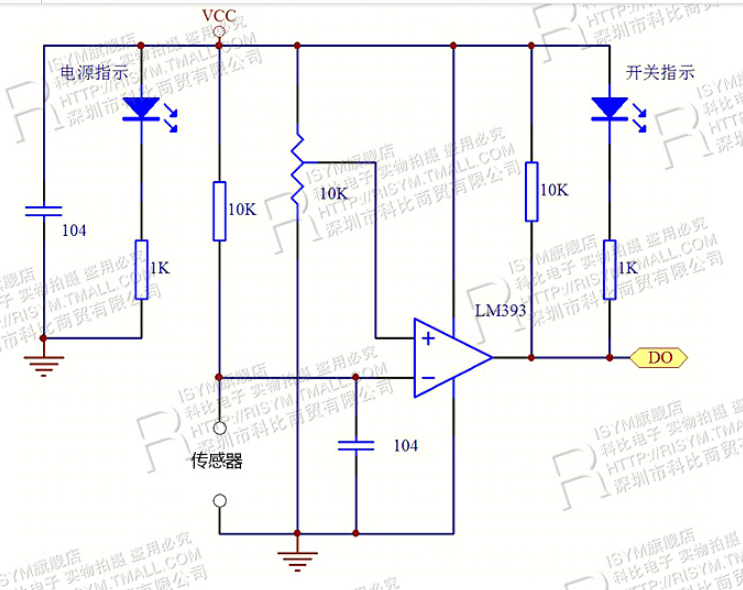


图2-10 火源检测传感器实物图

### 2.2.3 RFID门禁模块选择

本系统登录信息采集主要采用MF RC522模块，MF RC522是应用于13.56MHz非接触式通信中高集成度的读写卡芯片，是NXP公司针对“三表”应用推出的一款低电压、低成本、体积小的非接触式读写卡芯片，是智能仪表和便携式手持设备研发的较好选择。 MF RC522利用了先进的调制和解调概念，完全集成了在13.56MHz下所有类型的被动非接触式通信方式和协议。支持14443A兼容应答器信号。数字部分处理ISO14443A帧和错误检测。此外，还支持快速CRYPTO1加密算法，用语验证MIFARE系列产品。MFRC522支持MIFARE系列更高速的非接触式通信，双向数据传输速率高达424kbit/s。 作为13.56MHz高集成度读写卡系列芯片家族的新成员，MF RC522与MF RC500和MF RC530有不少相似之处，同时也具备许多特点和差异。它与主机间通信采用SPI模式，有利于减少连线，缩小PCB板体积，降低成本。

另外MF522-AN模块采用Philips MFRC522芯片设计读卡电路，使用方便，成本低廉，适用于设备开发、读卡器开发等高级应用的用户、需要进行射频卡终端设计/生产的用户。本模块可直接装入各种读卡器模具。模块采用电压为3.3V,通过SPI接口简单的几条线就可以直接与用户任何CPU主板相连接通信,可以保证模块稳定可靠的工作、读卡距离远.

其主要工作参数如下：

1. 工作电流：13—26mA/直流3.3V
2. 空闲电流：10-13mA/直流3.3V
3. 休眠电流：80uA
4. 峰值电流：30mA
5. 工作频率：13.56MHz
6. 数据传输速率：≤10Mbit/s
7. 环境相对湿度：相对湿度5%—95%
8. 支持的卡类型：mifare1 S50、mifare1 S70、mifare UltraLight、mifare Pro
9. 产品物理特性：尺寸：40mm×60mm
10. 环境工作温度：摄氏-20—80度
11. 环境储存温度：摄氏-40—85度

其示意图如下图所示：



图2-12 门禁识别模块实物图

## 2.3 软件结构设计

### 2.3.1 硬件电路程序设计软件选择

本系统使用C语言开放，主要使用Keil软件进行编程，Keil C51是美国Keil Software公司出品的51系列兼容[单片机](https://baike.baidu.com/item/%E5%8D%95%E7%89%87%E6%9C%BA/102396)C语言软件开发系统，与汇编相比，C语言在功能上、结构性、可读性、可维护性上有明显的优势，因而易学易用。Keil提供了包括C[编译器](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%99%A8/8853067)、宏汇编、链接器、库管理和一个功能强大的仿真调试器等在内的完整开发方案，通过一个[集成开发环境](https://baike.baidu.com/item/%E9%9B%86%E6%88%90%E5%BC%80%E5%8F%91%E7%8E%AF%E5%A2%83/298524)（μVision）将这些部分组合在一起。运行Keil软件需要WIN98、NT、WIN2000、WINXP等操作系统。如果你使用C语言编程，那么Keil几乎就是你的不二之选，即使不使用C语言而仅用汇编语言编程，其方便易用的集成环境、强大的软件仿真调试工具也会令你事半功倍。2013年10月，Keil正式发布了Keil μVision5 IDE。

使用Keil软件进行程序设计主要有以下两大优点：

1. Keil C51生成的目标代码效率非常之高，多数语句生成的汇编代码很紧凑，容易理解。在开发大型软件时更能体现高级语言的优势。
2. 与汇编相比，C语言在功能上、结构性、可读性、可维护性上有明显的优势，因而易学易用。用过汇编语言后再使用C来开发，体会更加深刻。

Keil软件图标如下图所示：



图2-13 Keil软件图标

其次，本系统主要使用的编程语言为C语言，[C语言](https://baike.baidu.com/item/C%E8%AF%AD%E8%A8%80/105958)是一门面向过程、抽象化的通用程序设计语言，广泛应用于底层开发。C语言能以简易的方式[编译](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91/1258343)、处理低级[存储器](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E5%99%A8/1583185)。C语言是仅产生少量的[机器语言](https://baike.baidu.com/item/%E6%9C%BA%E5%99%A8%E8%AF%AD%E8%A8%80/2019225)以及不需要任何运行环境支持便能运行的高效率程序设计语言。尽管C语言提供了许多低级处理的[功能](https://baike.baidu.com/item/%E5%8A%9F%E8%83%BD/10346898)，但仍然保持着跨平台的特性，以一个标准规格写出的C语言程序可在包括一些类似[嵌入式处理器](https://baike.baidu.com/item/%E5%B5%8C%E5%85%A5%E5%BC%8F%E5%A4%84%E7%90%86%E5%99%A8/9242667)以及超级计算机等作业平台的许多[计算机平台](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA%E5%B9%B3%E5%8F%B0/2606037)上进行编译。

### 2.3.2 上位机设计软件选择

上位机的设计主要使用VS，Microsoft Visual Studio是VS的全称。VS是美国[微软公司](https://baike.baidu.com/item/%E5%BE%AE%E8%BD%AF%E5%85%AC%E5%8F%B8)的开发工具包系列产品。VS是一个基本完整的开发工具集，它包括了整个[软件生命周期](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E7%94%9F%E5%91%BD%E5%91%A8%E6%9C%9F/861455)中所需要的大部分工具，如UML工具、代码管控工具、[集成开发环境](https://baike.baidu.com/item/%E9%9B%86%E6%88%90%E5%BC%80%E5%8F%91%E7%8E%AF%E5%A2%83/298524)(IDE)等等。所写的[目标代码](https://baike.baidu.com/item/%E7%9B%AE%E6%A0%87%E4%BB%A3%E7%A0%81/9407934)适用于微软支持的所有平台，包括[Microsoft Windows](https://baike.baidu.com/item/Microsoft%20Windows)、[Windows Mobile](https://baike.baidu.com/item/Windows%20Mobile)、[Windows CE](https://baike.baidu.com/item/Windows%20CE)、[.NET Framework](https://baike.baidu.com/item/.NET%20Framework)、[.Net Core](https://baike.baidu.com/item/.Net%20Core)、[.NET Compact Framework](https://baike.baidu.com/item/.NET%20Compact%20Framework)和Microsoft [Silverlight](https://baike.baidu.com/item/Silverlight) 及[Windows Phone](https://baike.baidu.com/item/Windows%20Phone)。

该款软件的六大优点如下：

**1、**VS2012和VS2010相比，最大的新特性莫过于对Windows 8 Metro开发的支持。Metro天生为云+端而生，简洁、数字化、内容优于形式、强调交互的设计已经成为未来的趋势。不过对于开发者而言，要想使用这项新功能，必须要安装Windows 8 RP版。该版本中包含了新的[Metro](https://baike.baidu.com/item/Metro/1344949)应用程序模板，增加了JavaScript功能、一个新的动画库，并提升了使用[XAML](https://baike.baidu.com/item/XAML/6123952)的Metro应用程序的性能。

**2、**VS2012 RC在界面上，比Beta版更容易使用，彩色的图标和按照开发、运行、调试等环境区分的颜色方案让人爱不释手。

**3、**VS2012集成了ASP。NET MVC 4，全面支持移动和HTML5，WF 4.5相比WF 4，更加成熟，期待已久的状态极工作流回来了，更棒的是，它的设计器已经支持C#表达式(之前只能用VB。NET)。

**4、**[VS2012](https://baike.baidu.com/item/VS2012/5005728)支持.NET 4.5，和.NET 4.0相比，4.5更多的是完善和改进，4.5也是Windows RT被提出来的首个框架库，.NET获得了和Windows API同等的待遇。

**5、**VS2012+TFS2012实现了更好的生命周期管理，可以这么说，VS2012不仅是开发工具，也是团队的管理信息系统。

**6、**VS2012对系统资源的消耗并不大，不过需要Windows 7/8的支持。

VS 2013:微软打破了[Visual Studio](https://baike.baidu.com/item/Visual%20Studio)两年升级一次的传统，Visual Studio 2012发布还不足一年，微软就计划发布了Visual Studio 2013了。在2013的[TechEd](https://baike.baidu.com/item/TechEd)大会上，微软宣布，2013年6月底的BUILD全球开发者大会上，大家不仅能见到[Windows 8.1](https://baike.baidu.com/item/Windows%208.1)的预览版，还能下载到Visual Studio 2013和Team Foundation Server 2013的预览版。

Visual Studio 2013新增了代码信息指示（Code information indicators）、团队工作室（Team Room）、身份识别、.NET内存转储分析仪、[敏捷开发](https://baike.baidu.com/item/%E6%95%8F%E6%8D%B7%E5%BC%80%E5%8F%91)项目模板、Git支持以及更强力的单元测试支持。

# 第3章 系统硬件设计

## 3.1 STM32ZET6最小系统电路设计

最小系统电路主要由竞争电路、复位电路、若干电阻电容、若干排针已经若干LED灯组成，开发板主控芯片为STM32ZET6，其主要特性如下：

1. 64KB SRAM
2. 512KB FLASH
3. 2 个 基本定时器
4. 4 个 通用定时器
5. 2 个 高级定时器
6. 2 个 DMA 控制器（共 12 个通道）
7. 3 个 SPI
8. 2 个 IIC
9. 5 个 串口
10. 1 个 USB
11. 1 个 CAN
12. 3 个 12 位 ADC
13. 1 个 12 位 DAC
14. 1 个 SDIO 接口
15. 1 个 FSMC 接口
16. 112 个通用 IO 口

STM32ZET6原理图如下图：

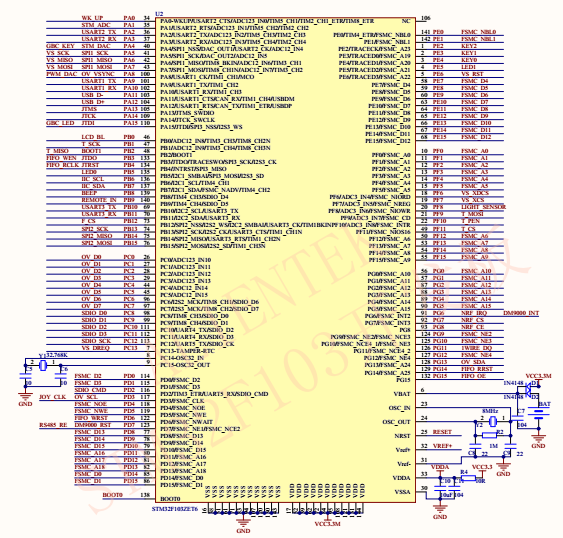


图3-1 STM32ZET6原理图

## 3.2 供电电路设计

主控电路板需要2.0V-3.6VDC电压，其他传感器需要3.3V-5V DC电压，故本系统的电源系统电路采用DC-005电源转接板模块，通过电源线接220V插排对系统供电。在通过连接线将直流5V或12V电接入主控板后，主控板通过MP2359稳压芯片将5V或12V直流电降压为3.3V。

MP2359是基于CMOS工艺开关内置DC/DC转换器。内置上端开关的导通电阻为0.35欧（典型值），能提供最大为1.2A的电流。这款芯片由振荡电路、PWM控制电路、基准电压源、误差放大器、相位补偿电路、斜坡电路、软启动电路、保护电路、内部电压调节器、开关等组成，外部器件用到电感、电阻、二极管、电容，所有这些构成完整的降压DC/DC转换器。

其中MP2359工作特性如下：

1. 工作电压范围：4.5V-24V
2. 内置N型驱动：典型值Ron=0.36欧
3. 反馈电压：0.8V 误差范围±1.5%
4. 尖峰电流限制：典型值2.0安培
5. 开关频率：1.4MHz
6. 封装：TSOT23-6

主控板供电电路如下图所示：

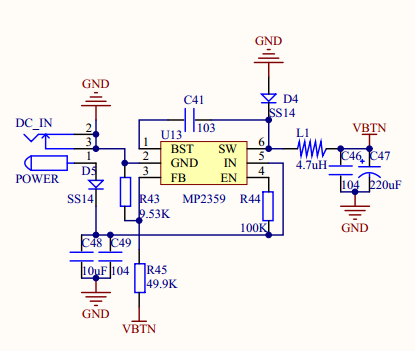


图3-2 主控板供电电路图

另外主控板还可以通过USB进行供电，电路原理图如下图所示：

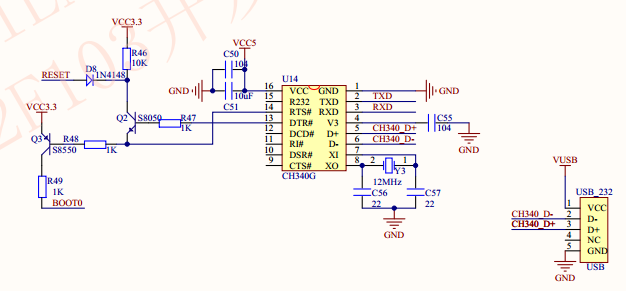


图3-3 主控板USB供电电路图

# 第4章 系统软件设计

## 4.1 主控程序设计

void LCD\_Init(void)

{

RCC->AHBENR|=1<<8; //使能FSMC时钟

RCC->APB2ENR|=1<<3; //使能PORTB时钟

RCC->APB2ENR|=1<<5; //使能PORTD时钟

RCC->APB2ENR|=1<<6; //使能PORTE时钟

RCC->APB2ENR|=1<<8; //使能PORTG时钟

GPIOB->CRL&=0XFFFFFFF0; //PB0 推挽输出 背光

GPIOB->CRL|=0X00000003;

//PORTD复用推挽输出

GPIOD->CRH&=0X00FFF000;

GPIOD->CRH|=0XBB000BBB;

GPIOD->CRL&=0XFF00FF00;

GPIOD->CRL|=0X00BB00BB;

//PORTE复用推挽输出

GPIOE->CRH&=0X00000000;

GPIOE->CRH|=0XBBBBBBBB;

GPIOE->CRL&=0X0FFFFFFF;

GPIOE->CRL|=0XB0000000;

//PORTG12复用推挽输出

GPIOG->CRH&=0XFFF0FFFF;

GPIOG->CRH|=0X000B0000;

GPIOG->CRL&=0XFFFFFFF0;//PG0->RS

GPIOG->CRL|=0X0000000B;

//寄存器清零

//bank1有NE1~4,每一个有一个BCR+TCR，所以总共八个寄存器。

//这里我们使用NE4 ，也就对应BTCR[6],[7]。

FSMC\_Bank1->BTCR[6]=0X00000000;

FSMC\_Bank1->BTCR[7]=0X00000000;

FSMC\_Bank1E->BWTR[6]=0X00000000;

//操作BCR寄存器 使用异步模式

FSMC\_Bank1->BTCR[6]|=1<<12; //存储器写使能

FSMC\_Bank1->BTCR[6]|=1<<14; //读写使用不同的时序

FSMC\_Bank1->BTCR[6]|=1<<4; //存储器数据宽度为16bit

//操作BTR寄存器

//读时序控制寄存器

FSMC\_Bank1->BTCR[7]|=0<<28; //模式A

FSMC\_Bank1->BTCR[7]|=1<<0; //地址建立时间（ADDSET）为2个HCLK 1/36M=27ns(实际>200ns)

//因为液晶驱动IC的读数据的时候，速度不能太快，尤其对1289这个IC。

FSMC\_Bank1->BTCR[7]|=0XF<<8; //数据保存时间为16个HCLK

//写时序控制寄存器

FSMC\_Bank1E->BWTR[6]|=0<<28; //模式A

FSMC\_Bank1E->BWTR[6]|=0<<0; //地址建立时间（ADDSET）为1个HCLK

//4个HCLK（HCLK=72M）因为液晶驱动IC的写信号脉宽，最少也得50ns。72M/4=24M=55ns

FSMC\_Bank1E->BWTR[6]|=3<<8; //数据保存时间为4个HCLK

//使能BANK1,区域4

FSMC\_Bank1->BTCR[6]|=1<<0; //使能BANK1，区域4

delay\_ms(50); // delay 50 ms

lcddev.id=LCD\_ReadReg(0x0000); //读ID（9320/9325/9328/4531/4535等IC）

if(lcddev.id<0XFF||lcddev.id==0XFFFF||lcddev.id==0X9300)//读到ID不正确,新增lcddev.id==0X9300判断，因为9341在未被复位的情况下会被读成9300

{

//尝试9341 ID的读取

LCD\_WR\_REG(0XD3);

lcddev.id=LCD\_RD\_DATA(); //dummy read

lcddev.id=LCD\_RD\_DATA(); //读到0X00

lcddev.id=LCD\_RD\_DATA(); //读取93

lcddev.id<<=8;

lcddev.id|=LCD\_RD\_DATA(); //读取41

}

// printf(" LCD ID:%x\r\n",lcddev.id); //打印LCD ID

if(lcddev.id==0X9341) //9341初始化

{

LCD\_WR\_REG(0xCF);

LCD\_WR\_DATA(0x00);

LCD\_WR\_DATA(0xC1);

LCD\_WR\_DATA(0X30);

LCD\_WR\_REG(0xED);

LCD\_WR\_DATA(0x64);

LCD\_WR\_DATA(0x03);

LCD\_WR\_DATA(0X12);

LCD\_WR\_DATA(0X81);

LCD\_WR\_REG(0xE8);

LCD\_WR\_DATA(0x85);

LCD\_WR\_DATA(0x10);

LCD\_WR\_DATA(0x7A);

LCD\_WR\_REG(0xCB);

LCD\_WR\_DATA(0x39);

LCD\_WR\_DATA(0x2C);

LCD\_WR\_DATA(0x00);

LCD\_WR\_DATA(0x34);

LCD\_WR\_DATA(0x02);

LCD\_WR\_REG(0xF7);

LCD\_WR\_DATA(0x20);

LCD\_WR\_REG(0xEA);

LCD\_WR\_DATA(0x00);

LCD\_WR\_DATA(0x00);

LCD\_WR\_REG(0xC0); //Power control

LCD\_WR\_DATA(0x1B); //VRH[5:0]

LCD\_WR\_REG(0xC1); //Power control

LCD\_WR\_DATA(0x01); //SAP[2:0];BT[3:0]

LCD\_WR\_REG(0xC5); //VCM control

LCD\_WR\_DATA(0x30); //3F

LCD\_WR\_DATA(0x30); //3C

LCD\_WR\_REG(0xC7); //VCM control2

LCD\_WR\_DATA(0XB7);

LCD\_WR\_REG(0x36); // Memory Access Control

LCD\_WR\_DATA(0x48);

LCD\_WR\_REG(0x3A);

LCD\_WR\_DATA(0x55);

LCD\_WR\_REG(0xB1);

LCD\_WR\_DATA(0x00);

LCD\_WR\_DATA(0x1A);

LCD\_WR\_REG(0xB6); // Display Function Control

LCD\_WR\_DATA(0x0A);

LCD\_WR\_DATA(0xA2);

LCD\_WR\_REG(0xF2); // 3Gamma Function Disable

LCD\_WR\_DATA(0x00);

LCD\_WR\_REG(0x26); //Gamma curve selected

LCD\_WR\_DATA(0x01);

LCD\_WR\_REG(0xE0); //Set Gamma

LCD\_WR\_DATA(0x0F);

LCD\_WR\_DATA(0x2A);

LCD\_WR\_DATA(0x28);

LCD\_WR\_DATA(0x08);

LCD\_WR\_DATA(0x0E);

LCD\_WR\_DATA(0x08);

LCD\_WR\_DATA(0x54);

LCD\_WR\_DATA(0XA9);

LCD\_WR\_DATA(0x43);

LCD\_WR\_DATA(0x0A);

LCD\_WR\_DATA(0x0F);

LCD\_WR\_DATA(0x00);

LCD\_WR\_DATA(0x00);

LCD\_WR\_DATA(0x00);

LCD\_WR\_DATA(0x00);

LCD\_WR\_REG(0XE1); //Set Gamma

LCD\_WR\_DATA(0x00);

LCD\_WR\_DATA(0x15);

LCD\_WR\_DATA(0x17);

LCD\_WR\_DATA(0x07);

LCD\_WR\_DATA(0x11);

LCD\_WR\_DATA(0x06);

LCD\_WR\_DATA(0x2B);

LCD\_WR\_DATA(0x56);

LCD\_WR\_DATA(0x3C);

LCD\_WR\_DATA(0x05);

LCD\_WR\_DATA(0x10);

LCD\_WR\_DATA(0x0F);

LCD\_WR\_DATA(0x3F);

LCD\_WR\_DATA(0x3F);

LCD\_WR\_DATA(0x0F);

LCD\_WR\_REG(0x2B);

LCD\_WR\_DATA(0x00);

LCD\_WR\_DATA(0x00);

LCD\_WR\_DATA(0x01);

LCD\_WR\_DATA(0x3f);

LCD\_WR\_REG(0x2A);

LCD\_WR\_DATA(0x00);

LCD\_WR\_DATA(0x00);

LCD\_WR\_DATA(0x00);

LCD\_WR\_DATA(0xef);

LCD\_WR\_REG(0x11); //Exit Sleep

delay\_ms(120);

LCD\_WR\_REG(0x29); //display on

}

LCD\_Display\_Dir(0); //默认为竖屏

LCD\_LED=1; //点亮背光

LCD\_Clear(WHITE);

}

## 4.2 通信程序设计

其中部分从机通信代码如下：

const u8 TX\_ADDRESS[TX\_ADR\_WIDTH]={0x34,0x43,0x10,0x10,0x01}; //发送地址

const u8 RX\_ADDRESS[RX\_ADR\_WIDTH]={0x34,0x43,0x10,0x10,0x01}; //发送地址

//初始化24L01的IO口

void NRF24L01\_Init(void)

{

RCC->APB2ENR|=1<<3; //使能PORTB时钟

RCC->APB2ENR|=1<<8; //使能PORTG时钟

GPIOB->CRH&=0XFFF0FFFF;

GPIOB->CRH|=0X00030000; //PB12 推挽

GPIOB->ODR|=1<<12; //PB12上拉 防止W25Qxx的干扰

GPIOG->CRL&=0X00FFFFFF;

GPIOG->CRL|=0X38000000; //PG6输入,PG7推挽输出

GPIOG->CRH&=0XFFFFFFF0;

GPIOG->CRH|=0X00000003; //PG8 推挽输出

GPIOG->ODR|=7<<6; //PG6 7 8 上拉

SPI2\_Init(); //初始化SPI

//针对NRF的特点修改SPI的设置

SPI2->CR1&=~(1<<6); //SPI设备失能

SPI2->CR1&=~(1<<1); //空闲模式下SCK为0 CPOL=0

SPI2->CR1&=~(1<<0); //数据采样从第1个时间边沿开始,CPHA=0

SPI2->CR1|=1<<6; //SPI设备使能

NRF24L01\_CE=0; //使能24L01

NRF24L01\_CSN=1; //SPI片选取消

}

//检测24L01是否存在

//返回值:0，成功;1，失败

u8 NRF24L01\_Check(void)

{

u8 buf[5]={0XA5,0XA5,0XA5,0XA5,0XA5};

u8 i;

SPI2\_SetSpeed(SPI\_SPEED\_4); //spi速度为9Mhz（24L01的最大SPI时钟为10Mhz）

NRF24L01\_Write\_Buf(NRF\_WRITE\_REG+TX\_ADDR,buf,5);//写入5个字节的地址.

NRF24L01\_Read\_Buf(TX\_ADDR,buf,5); //读出写入的地址

for(i=0;i<5;i++)if(buf[i]!=0XA5)break;

if(i!=5)return 1;//检测24L01错误

return 0; //检测到24L01

}

//SPI写寄存器

//reg:指定寄存器地址

//value:写入的值

u8 NRF24L01\_Write\_Reg(u8 reg,u8 value)

{

u8 status;

NRF24L01\_CSN=0; //使能SPI传输

status =SPI2\_ReadWriteByte(reg);//发送寄存器号

SPI2\_ReadWriteByte(value); //写入寄存器的值

NRF24L01\_CSN=1; //禁止SPI传输

return(status); //返回状态值

}

//读取SPI寄存器值

//reg:要读的寄存器

u8 NRF24L01\_Read\_Reg(u8 reg)

{

u8 reg\_val;

NRF24L01\_CSN = 0; //使能SPI传输

SPI2\_ReadWriteByte(reg); //发送寄存器号

reg\_val=SPI2\_ReadWriteByte(0XFF);//读取寄存器内容

NRF24L01\_CSN = 1; //禁止SPI传输

return(reg\_val); //返回状态值

}

//在指定位置读出指定长度的数据

//reg:寄存器(位置)

//\*pBuf:数据指针

//len:数据长度

//返回值,此次读到的状态寄存器值

u8 NRF24L01\_Read\_Buf(u8 reg,u8 \*pBuf,u8 len)

{

u8 status,u8\_ctr;

NRF24L01\_CSN = 0; //使能SPI传输

status=SPI2\_ReadWriteByte(reg);//发送寄存器值(位置),并读取状态值

for(u8\_ctr=0;u8\_ctr<len;u8\_ctr++)pBuf[u8\_ctr]=SPI2\_ReadWriteByte(0XFF);//读出数据

NRF24L01\_CSN=1; //关闭SPI传输

return status; //返回读到的状态值

}

## 4.3 传感器通信程序设计

根据预设功能选择多种传感器，传感器通过数据线和单片机的IO口进行连接，之间通过二进制电信号进行通信，不同种的传感器通过自身材料的特性，通电之后会产生相应的电信号，单片机在接收到这些电信号之后通过内置的AD、DA已经其他功能，将这些电信号进行处理和加工，得到我们需要的数据。

### 4.3.1 门禁控制程序设计

门禁模块主要也是通过SPI通信给单片机传输数据，可以将卡的基本信息传递给单片机，单片机通过检测卡的ID来进行判断、识别和控制功能。

其中部分门禁控制代码如下：

u8 SPIWriteByte(u8 Byte)

{

while((SPI1->SR&0X02)==0); //等待发送区空

SPI1->DR=Byte; //发送一个byte

while((SPI1->SR&0X01)==0); //等待接收完一个byte

return SPI1->DR; //返回收到的数据

}

//配置SPI2

void SPI1\_Init(void)

{

RCC->APB2ENR|=1<<0;//复用时钟使能

RCC->APB2ENR|=1<<2;//PORTA时钟使能

RCC->APB2ENR|=1<<12;//SPI3时钟使能

GPIOA->CRL&=0X00000FFF; //PA.3 复位

GPIOA->CRL|=0XB8B33000; //PA.4 片选 推挽输出

SPI1->CR1|=0<<10;//全双工模式

SPI1->CR1|=1<<9; //软件nss管理

SPI1->CR1|=1<<8;

SPI1->CR1|=1<<2; //SPI主机

SPI1->CR1|=0<<11;//8bit数据格式

SPI1->CR1&=~0x02;//空闲模式下SCK为0 CPOL=0

SPI1->CR1&=~0x01;//数据采样从第一个时间边沿开始,CPHA=0

SPI1->CR1|=7<<3; //Fsck=Fcpu/4

SPI1->CR1|=0<<7; //MSBfirst

SPI1->CR1|=1<<6; //SPI设备使能

}

//RFID信息读取函数 被调用的函数

void RFID\_read(void)

{

unsigned char status;

status = PcdRequest(PICC\_REQALL,CT);/\*尋卡\*/

if(status==MI\_OK)//尋卡成功

{

status=MI\_ERR;

status = PcdAnticoll(SN);/\*防冲撞\*/

}

if(status==MI\_OK)

{

status=MI\_ERR;

status =PcdSelect(SN);

}

if(status==MI\_OK)//選卡成功

{

status=MI\_ERR;

status =PcdAuthState(0x61,0x06,KEY,SN);

}

if(status==MI\_OK)//驗證成功

{

status=MI\_ERR;

LCD\_Clear(WHITE);

page=1;

delay\_ms(10);

}

}

//RFID刷卡进入显示函数

void RFID\_display(void)

{

POINT\_COLOR=BLUE;

Show\_Str(48,5,240,24,"蔬菜生长检测系统",24,0);

POINT\_COLOR=BLACK;

if(seat==0)

{

Show\_Str(24,40,192,24,"欢迎使用本系统",24,0);

Show\_Str(5,70,144,24,"用户信息为：",24,0);

LCD\_ShowNum(10,100,SN[0],3,16);

LCD\_ShowNum(58,100,SN[1],3,16);

LCD\_ShowNum(106,100,SN[2],3,16);

LCD\_ShowNum(154,100,SN[3],3,16);

}

### 4.3.2 土壤温湿度、大气温湿度及雨滴检测程序设计

雨滴检测传感器主要通过本身材料的特殊性，因为不纯净的水具有导电性，当雨滴传感器表面沾有导电液体时，其回路电流值会发生相应变化，然后通过数据线将电信号传给单片机，单片机再通过传输回来的电信号进行判断。

土壤温湿度主要通过土壤温湿度检测模块，也是通过传感器的材料特性，当土壤温湿度不同时，对土壤温湿度回路电流的影响不同，单片机通过传回的电信号进行状态判断。

大气温湿度主要通过DHT11进行检查。

其中检测部分代码如下：

void LED\_Init(void)

{

RCC->APB2ENR|=1<<2; //使能PORTa时钟

GPIOA->CRH&=0XFFF00FF0;

GPIOA->CRH|=0X00033003;

GPIOA->ODR|=1<<8; //PB.5 输出高

GPIOA->ODR|=1<<11; //PB.5 输出高

GPIOA->ODR|=1<<12; //PB.5 输出高

}

void LCD\_page0\_display(void) //主页面设计 (被调用的函数)

{

num=3;

POINT\_COLOR=BLUE;

Show\_Str(48,5,240,24,"车位检测系统",24,0);

POINT\_COLOR=BLACK;

Show\_Str(5,35,108,24,"环境温度：",24,0);

POINT\_COLOR=BLACK;

temp(); //读取温度

LCD\_ShowNum(120,39,temperature/10,2,16);

LCD\_ShowString(155,39,8,16,16,".");

LCD\_ShowNum(165,39,temperature%10,1,16);

Show\_Str(5,65,108,24,"是否下雨：",24,0);

adcx=Get\_Adc\_Average(ADC\_CH1,5);

if(adcx>4000)

{

LCD\_Fill(120,65,96,24,WHITE);

Show\_Str(120,65,48,24,"无雨",24,0);

}

else if(adcx<4000)

{

LCD\_Fill(120,65,96,24,WHITE);

Show\_Str(120,65,48,24,"有雨",24,0);

}

/初始化ADC1

//这里我们仅以规则通道为例

//我们默认仅开启通道1

void Adc\_Init(void)

{

//先初始化IO口

RCC->APB2ENR|=1<<2; //使能PORTA口时钟

GPIOA->CRL&=0XFFFFFF0F;//PA1 anolog输入

RCC->APB2ENR|=1<<9; //ADC1时钟使能

RCC->APB2RSTR|=1<<9; //ADC1复位

RCC->APB2RSTR&=~(1<<9);//复位结束

RCC->CFGR&=~(3<<14); //分频因子清零

//SYSCLK/DIV2=12M ADC时钟设置为12M,ADC最大时钟不能超过14M!

//否则将导致ADC准确度下降!

RCC->CFGR|=2<<14;

ADC1->CR1&=0XF0FFFF; //工作模式清零

ADC1->CR1|=0<<16; //独立工作模式

ADC1->CR1&=~(1<<8); //非扫描模式

ADC1->CR2&=~(1<<1); //单次转换模式

ADC1->CR2&=~(7<<17);

ADC1->CR2|=7<<17; //软件控制转换

ADC1->CR2|=1<<20; //使用用外部触发(SWSTART)!!! 必须使用一个事件来触发

ADC1->CR2&=~(1<<11); //右对齐

ADC1->SQR1&=~(0XF<<20);

ADC1->SQR1|=0<<20; //1个转换在规则序列中 也就是只转换规则序列1

//设置通道1的采样时间

ADC1->SMPR2&=~(3\*1); //通道1采样时间清空

ADC1->SMPR2|=7<<(3\*1); //通道1 239.5周期,提高采样时间可以提高精确度

ADC1->CR2|=1<<0; //开启AD转换器

ADC1->CR2|=1<<3; //使能复位校准

while(ADC1->CR2&1<<3); //等待校准结束

//该位由软件设置并由硬件清除。在校准寄存器被初始化后该位将被清除。

ADC1->CR2|=1<<2; //开启AD校准

while(ADC1->CR2&1<<2); //等待校准结束

//该位由软件设置以开始校准，并在校准结束时由硬件清除

}

//获得ADC1某个通道的值

//ch:通道值 0~16

//返回值:转换结果

u16 Get\_Adc(u8 ch)

{

//设置转换序列

ADC1->SQR3&=0XFFFFFFE0;//规则序列1 通道ch

ADC1->SQR3|=ch;

ADC1->CR2|=1<<22; //启动规则转换通道

while(!(ADC1->SR&1<<1));//等待转换结束

return ADC1->DR; //返回adc值

}

//获取通道ch的转换值，取times次,然后平均

//ch:通道编号

//times:获取次数

//返回值:通道ch的times次转换结果平均值

u16 Get\_Adc\_Average(u8 ch,u8 times)

{

u32 temp\_val=0;

u8 t;

for(t=0;t<times;t++)

{

temp\_val+=Get\_Adc(ch);

delay\_ms(5);

}

return temp\_val/times;

}