摘要

从古代的滴漏更鼓到近代的机械钟，从电子表到目前的数字时钟，人类的生活和工作均离不开时钟。为了准确的测量和记录时间，人们一直在努力改进着计时工具。钟表的数字化，大力推动了计时的精确性和可靠性其中最具代表性的计时产品就是数字时钟，从原有传统指针计时的方式发展为人们日常更为熟悉的背光数字显示方式，直观明了，并增加了全自动日期、星期、以及其他日常附属信息的显示功能，更能符合消费者的生活需求。

本文所设计的数字时钟是基于ARMCortex™-M4处理器的，该处理器是由ARM专门开发的最新嵌入式处理器，在M3的基础上强化了运算能力，新加了浮点、DSP、并行计算等，用以满足需要有效且易于使用的控制和信号处理功能混合的数字信号控制市场。其高效的信号处理功能与Cortex-M处理器系列的低功耗、低成本和易于使用的优点的组合[2]。此数字时钟具有读取方便、显示直观、功能多样、电路简洁、成本低廉等诸多优点，符合电子仪器仪表的发展趋势，具有广阔的市场前景。值得我们进行深入的研究和了解。

**关键字**：数字时钟，ARMCortex™-M4，触摸屏

ABSTRACT

From the ancient drip to the modern mechanical clock, from the electronic watch to the current digital clock, human life and work are inseparable from the clock. In order to accurately measure and record time, people have been trying to improve the timing tool. Digital watches and clocks, and vigorously promote the timing of the accuracy and reliability. One of the most representative timing products is the digital clock, from the original traditional way of timing the development of people are more familiar with the daily background of the digital display, intuitive and clear, and increase the automatic date, week, and other daily subsidiary information Of the display function, more in line with the needs of consumers living.

The digital clock designed in this paper is based on the ARMCortex ™ -M4 processor, which is the latest embedded processor specifically developed by ARM, which enhances computing power on the basis of M3, adding floating point, DSP, and parallel Calculation, etc., to meet the needs of effective and easy to use control and signal processing functions mixed digital signal control market. The combination of its high-efficiency signal processing capabilities and the low power, low cost and easy-to-use advantages [2]. This digital clock has the advantages of easy reading, intuitive display, versatile, simple circuit, low cost and so on. It is in line with the development trend of electronic instrumentation and has broad market prospect. Worthy of our in-depth study and understanding.

**Key Words:** Digital clock, ARMCortex ™ -M4, touch screen

目录

[第1章 课题整体框架 1](#_Toc482299010)

[1.1 课题任务 1](#_Toc482299011)

[1.2 课题要求 1](#_Toc482299012)

[1.3 研究意义 1](#_Toc482299013)

[第2章 设计方案 2](#_Toc482299014)

[2.1 硬件部分 2](#_Toc482299015)

[2.1.1 电源系统模块 2](#_Toc482299016)

[2.1.2 开机启动模块 2](#_Toc482299017)

[2.2 MCU模块 2](#_Toc482299018)

[2.3 SPI-flash存储模块 3](#_Toc482299019)

[2.4 LCD液晶显示屏模块 6](#_Toc482299020)

[2.5 电容触摸屏模块 8](#_Toc482299021)

[2.1.7模块流程框图 9](#_Toc482299022)

[2.2实现功能描述 11](#_Toc482299023)

[2.2.1时间的显示功能 11](#_Toc482299024)

[2.2.2 虚拟时钟 11](#_Toc482299025)

[2.2.3 闹钟功能 11](#_Toc482299026)

[2.2.4 秒表功能 11](#_Toc482299027)

[2.2.5 图形界面的实现 12](#_Toc482299028)

[2.3电路设计 13](#_Toc482299029)

[2.3.1 时钟电路 13](#_Toc482299030)

[2.3.2 延时启动电路 13](#_Toc482299031)

[2.3.3 TF卡接口 14](#_Toc482299032)

[2.3.4 SPI-flash存储电路 14](#_Toc482299033)

[2.3.5 电容触摸屏电路 15](#_Toc482299034)

[2.3.6 液晶显示屏 15](#_Toc482299035)

[2.3.7 电压转换电路 16](#_Toc482299036)

[2.3.8 cortex-M4芯片 16](#_Toc482299037)

[2.4软件设计 17](#_Toc482299038)

[2.4.1软件设计思路 17](#_Toc482299039)

[第3章 调试与实现 20](#_Toc482299040)

[3.1 调试中遇到的重点与难点 20](#_Toc482299041)

[3.2 解决方案 20](#_Toc482299042)

[3.3 实现展示（附上仿真图或实物照片） 21](#_Toc482299043)

[3.4 实现过程 25](#_Toc482299044)

[3.4.1 电路的安装 25](#_Toc482299045)

[3.4.2 电路的焊接 25](#_Toc482299046)

[3.4.3 插件原件焊接的步骤 25](#_Toc482299047)

[3.4.4 焊接要素 25](#_Toc482299048)

[3.4.5硬件测试 26](#_Toc482299049)

[第4章 总结 27](#_Toc482299050)

[参考文献 28](#_Toc482299051)

[致谢 29](#_Toc482299052)

[附录 30](#_Toc482299053)

[附录一：硬件设计原理图 30](#_Toc482299054)

[附录二：数字时钟函数实现 30](#_Toc482299055)

[附录三：图形界面绘图代码 31](#_Toc482299056)

[附录四：cortex-M4数据手册 32](#_Toc482299057)

1. 课题整体框架
   1. 课题任务

设计一个基于cortex-M4的数字手表系统：实现时间如年、月、日、时、分、秒的显示；实现秒表功能；实现虚拟时钟界面。

* 1. 课题要求

根据要求，设计初步思路：本设计该设计硬件由电源系统、开机启动模块、MCU处理器模块、SPI-flash存储模块、LCD液晶显示屏模块、电容式触摸屏模块6个部分组成。锂电池作为电源为整个工作系统供电，经过电源系统处理保证整个系统的供电电压恒定为3.3V。开机启动电路实现软开机，避免瞬间电流冲击。Cortex-M4是整个系统的控制核心，数据的显示、输入、输出都由它控制。SPI-flash存储器主要完成数据的存储和读取，闹钟定时及秒表功能的时钟数据的记忆和保存。LCD液晶显示屏用来显示图形界面以及年、月、日、时、分、秒等时间内容。电容式触摸屏实现触摸控制和参数设置、日期调整功能的实现。

* 1. 研究意义

加入世贸组织后，中国将面临激烈的竞争。这场比赛将是一场科技实力，管理水平和人才素质的较量，风险和机遇并存，及电子产品的发展变化迅速，不仅在通信技术的数字替代模拟信号，甚至在我们日常的生活让数字化取缔.相比模拟钟能给人一种一目了然的感觉，它不仅可以显示在同一时间，时、分和秒，并且可以完成准确的校正。同时，数字时钟可以准确的时间，你的时间精确到报时的声音，提醒你在这个时候，需要做的事情。它比老式时钟更适合现代生活。一个数字时钟振荡器，计数器，译码器和显示器电路精确时间“小时”“分”“秒”与数字显示，并需要校正电路，使其准确的工作，也可有定时和计时功能。数字钟及扩大其应用，有着非常现实的意义。

1. 设计方案
   1. 硬件部分
      1. 电源系统模块

电源系统模块主要由锂电池电源及电源系统构成。锂电池为整个电路供电，电源首先输送至电源系统，由电源系统处理后为电路提供稳定的3.3V电压，从而保证电路的正常工作。

* + 1. 开机启动模块

开机启动过后，会产生瞬间强电流，开机启动模块主要为实现软开机，避免瞬间强电流对电路冲击，从而实现对电路的保护。

* 1. MCU模块

MCU模块各引脚如图，其中DB8-DB0为数据传输引脚，可以同时传输九位二进制数据，高达512字节。Cortex-M4共有2种数据总线模式：8Bit、9Bit。12种数据格式：8Bit（GRB4-4-4）、8Bit（GRB5-6-5）、8Bit（GRB6-6-6）、9Bit（GRB4-4-4）、9Bit（GRB5-6-5）、9Bit（GRB6-6-6）、16Bit（GRB4-4-4）、16Bit（GRB5-6-5）、16Bit（GRB6-6-6）、18Bit（GRB4-4-4）、18Bit（GRB5-6-5）、18Bit（GRB6-6-6）。本文采用的是9Bit8080总线模式RGB6-6-6数据格式。

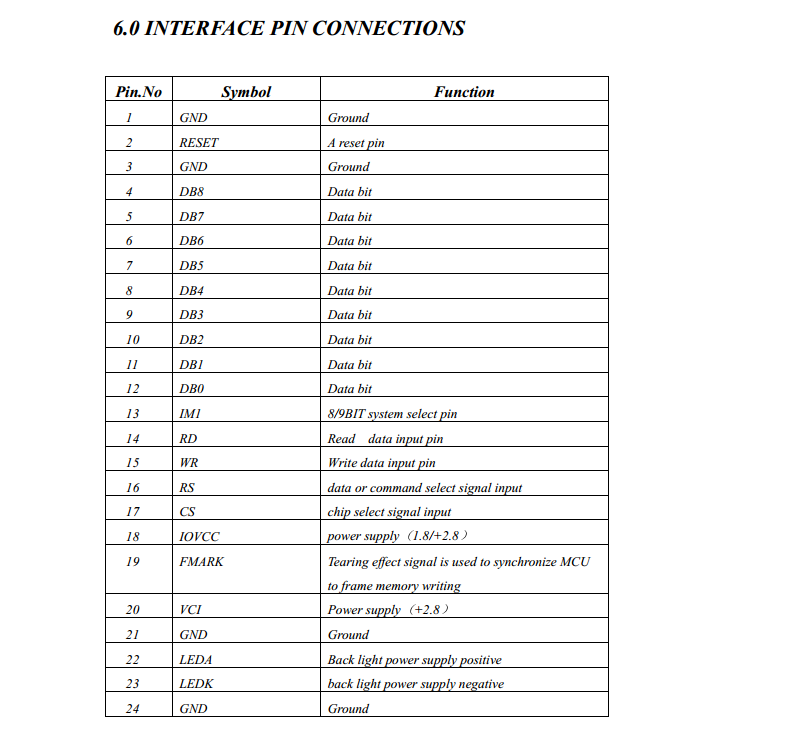


图2-1 LCD液晶屏引脚功能图

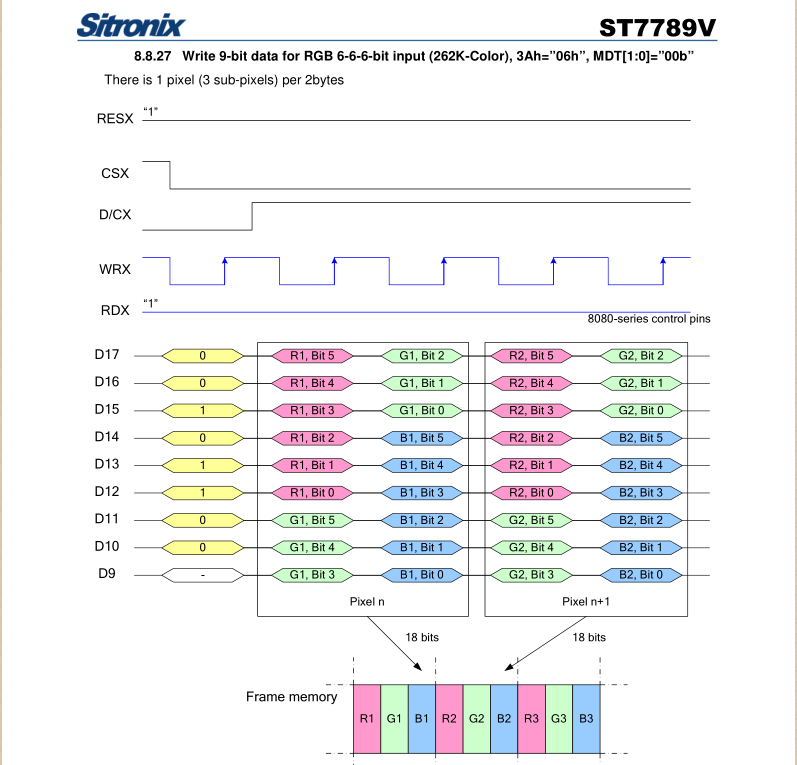


图2-2 9Bit总线下RGB6-6-6数据传输图

* 1. SPI-flash存储模块

串行flash 存储器W25Q128BV为那些对空间大小，引脚数，功耗有限制的系统提供了一个存储解决方案。25Q系列的灵活性和性能比一般的串行flash 设备要高。利用串行FLASH 可以实现代码映射到RAM，直接通过DUAL/QUAD SPI方式来执行代码，存储声音，文本，数据。

W25Q128BV 供电范围为2.7——3.6V，在激活状态下电流功耗低到4MA，睡眠状态下则降低到1UA。所有的25Q 系列都提供节省空间的封装。

W25Q128BV由65536可编程的页组成的，每页有256个字节。一次最多可以写256个字节。可以一次擦除16页（4KB sector erase），128页（32KB block erase）,256页(64KB block erase),或者擦除一整片。W25Q128BV 有4096个可擦除的扇区，256可擦除的块。4KB的扇区对于数据和参数存储有更高的灵活性。

W25Q128BV 支持标准SPI 接口，以及更高性能的DUAL/QUAD SPI ，对应的管脚为时钟，片选，（I/O0）DI，（I/O1）DO，I/O2（/WP），I/O3（/HOLD）。SPI 时钟可以达到104MHz，在DUAL使用快速读时就相当于208MHz，在QUAD 使用快速读时相当于320MHz。这个传输速率比一般的异步8位，16位并行FLASH 存储器要快。连续读模式访问存储器的效率很高，只要8个时钟的指令开销就可以读24位地址的数据，这样就可以实现XIP，HOLD，WP管脚，可编程的写保护，可分为顶部，底部，整个存储器，这些提供了更灵活的控制。一般地，W25Q128BV支持JEDEC标准，一个64位的独立串行数字，包含制造商和芯片ID。

2.1.4.1写使能（06h）

写使能指可以设置状态寄存器中的WEL位置1 。在页写、QUAD页写、扇区擦除、块擦除、片擦除、写状态寄存器、擦写安全寄存器指令之前，必须先将WEL 位置1 。写使能指令是以/CS拉低开始的，将06H 通过DI 在时钟的上升沿锁存，然后/CS 拉高来结束指令。

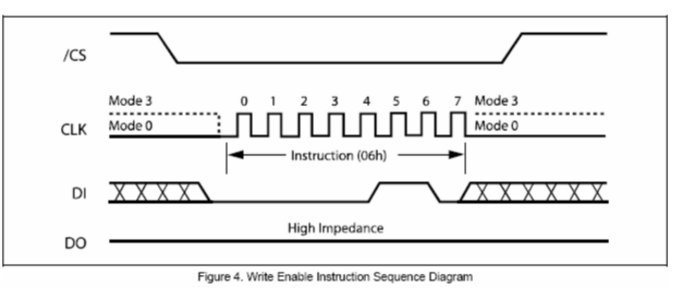


图2-3 写使能状态下的时序图

2.1.4.2读状态寄存器1（05H）和读状态寄存器2（35H）指令

读状态寄存器指令允许读8位状态寄存器位。这条指令是以/CS 拉低开始，然后通过DI 在时钟的上升沿传输指令代码05H（读寄存器1指令）或者是35H（读寄存器2指令），然后状态寄存器的相应位通过DO在时钟的下降沿从高位到低位依次传出。最后以/CS拉高结束。读状态寄存指令可以任何时间使用，在擦写，写状态寄存器指令周期中依然可以。这样就可以随时检查BUSY位，检查相应的指令周期有没有结束，芯片是不是可以接受新的指令。状态寄存器可以连续读出来。如图

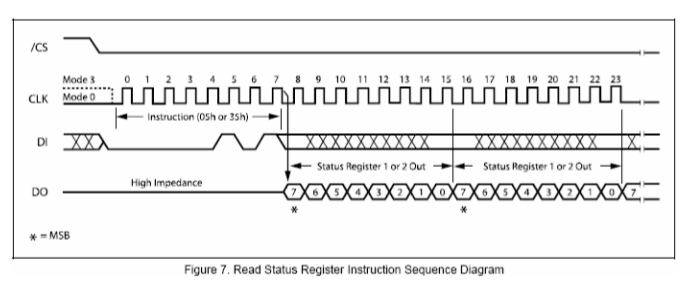


图2-4 读状态寄存器状态下的时序图

2.1.4.3读数据（03H）

读数据指令允许从存储器读一个字节和连续多个字节。该指令是以/CS 拉低开始，然后通DI 在时钟的上升沿来传输指令代码（03H）和24位地址。当芯片接受完地址位后，相应地址处的值将会，在时钟的下降沿，以高位在前低位在后的方式，在D0上传输。如果连续的读多个字节的话，地址是自动加1的。这意味着可以一次读出整个芯片。该指令也是以/CS拉高来结束的。如果当BUSY=1时执行该指令，该指令将被忽略，并且对正在执行的其他指令不会有任何影响。读数据指令的时钟可以从DC 到最大的fR快速读指令（0BH）。

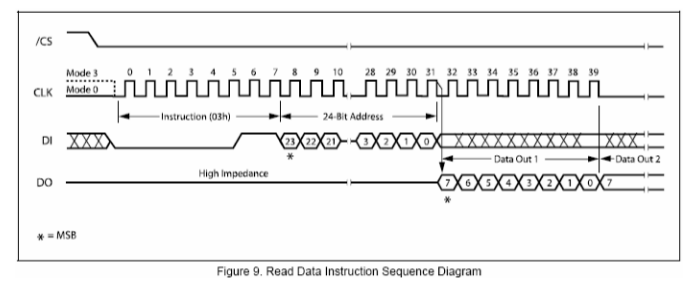


图2-5 读数据状态下的时序图

2.1.4.4快速读指令（0BH）

快速读指令与读数据指令比较相似，差别在于快速读数据指令可以以最高的频率读。只是在传输完24位地址后，另加8位无关的数据。在传输8位无关的数据时间内，芯片用来建立初始地址，这时DO上的数据也是无关的。

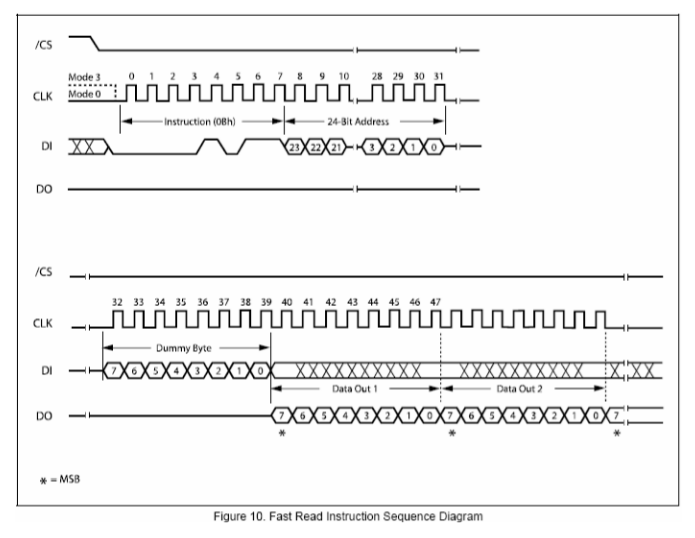


图2-6 快速读指令状态下的时序图

* 1. LCD液晶显示屏模块

目前液晶显示技术大多以TN、STN、TFT三种技术为主轴，因此我们就这从这三种技术来探讨它们的运作原理。TN型的液晶显示技术可说是液晶显示器中最基本的，而之后其它种类的液晶显示器也可说是以TN型为原点来加以改良。同样的，它的运作原理也较其它技术来的简单，请读者参照下方的图片。图中所表示的是TN型液晶显示器的简易构造图，包括了垂直方向与水平方向的偏光板，具有细纹沟槽的配向膜，液晶材料以及导电的玻璃基板。 其显像原理是将液晶材料置于两片贴附光轴垂直偏光板之透明导电玻璃间，液晶分子会依配向膜的细沟槽方向依序旋转排列，如果电场未形成，光线会顺利的从偏光板射入，依液晶分子旋转其行进方向，然后从另一边射出。如果在两片导电玻璃通电之后，两片玻璃间会造成电场，进而影响其间液晶分子的排列，使其分子棒进行扭转，光线便无法穿透，进而遮住光源。这样所得到光暗对比的现象，叫做扭转式向列场效应，简称TNFE（twisted nematic field effect）。在电子产品中所用的液晶显示器，几乎都是用扭转式向列场效应原理所制成。STN型的显示原理也似类似，不同的是TN扭转式向列场效应的液晶分子是将入射光旋转90度，而STN超扭转式向列场效应是将入射光旋转180~270度。要在这边说明的是，单纯的TN液晶显示器本身只有明暗两种情形（或称黑白），并没有办法做到色彩的变化。而STN液晶显示器牵涉液晶材料的关系，以及光线的干涉现象，因此显示的色调都以淡绿色与橘色为主。但如果在传统单色STN液晶显示器加上一彩色滤光片（color filter），并将单色显示矩阵之任一像素（pixel）分成三个子像素（sub-pixel），分别透过彩色滤光片显示红、绿、蓝三原色，再经由三原色比例之调和，也可以显示出全彩模式的色彩。另外，TN型的液晶显示器如果显示屏幕做的越大，其屏幕对比度就会显得较差，不过藉由STN的改良技术，则可以弥补对比度不足的情况。

TFT型的液晶显示器较为复杂，主要的构成包括了，荧光管、导光板、偏光板、滤光板、玻璃基板、配向膜、液晶材料、薄模式晶体管等等。首先液晶显示器必须先利用背光源，也就是荧光灯管投射出光源，这些光源会先经过一个偏光板然后再经过液晶，这时液晶分子的排列方式进而改变穿透液晶的光线角度。然后这些光线接下来还必须经过前方的彩色的滤光膜与另一块偏光板。因此我们只要改变刺激液晶的电压值就可以控制最后出现的光线强度与色彩，并进而能在液晶面板上变化出有不同深浅的颜色组合了。

该部分主要用于显示年、月、日、时、分、秒等基本内容，以及显示虚拟时钟界面、设置界面、菜单界面，秒表界面等。由MCU向其传输数据（9Bit总线模式下GRB6-6-6数据传输模式），控制其显示效果。

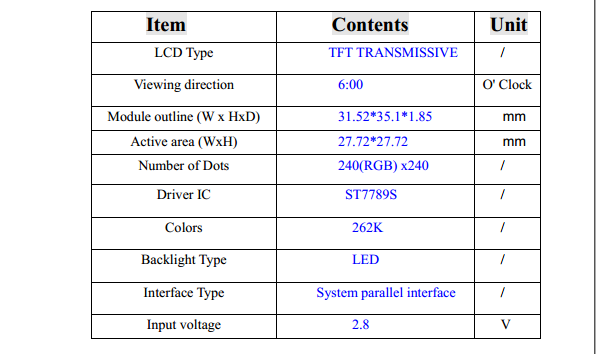


图2-7 LCD规格参数

* 1. 电容触摸屏模块

电容式触摸屏的构造主要是在玻璃屏幕上镀一层透明的薄膜导体层ITO(目前的透明导电材料——氧化金属),再在导体层外加上一块非常薄的坚硬的保护玻璃(一块四层复合玻璃屏)，这层玻璃显然是不导电的，直流导电是不行了，改用高频交流信号，靠人的手指头（隔着薄玻璃）与工作面形成的耦合电容来吸走一个交流电流，这就是电容屏“电容”名字的由来：靠耦合电容来工作。双玻璃设计能彻底保护导体层及感应器。电容式触摸屏在触摸屏四边均镀上狭长的电极，在导电体内形成一个低电压交流电场。在触摸屏幕时，由于人体电场，手指与导体层间会形成一个耦合电容，四边电极发出的电流会流向触点，而电流强弱与手指到电极的距离成正比，位于触摸屏幕后的控制器便会计算电流的比例及强弱，准确算出触摸点的位置。电容触摸屏的双玻璃不但能保护导体及感应器，更有效地防止外在环境因素对触摸屏造成影响，就算屏幕沾有污秽、尘埃或油渍，电容式触摸屏依然能准确算出触摸位置。

本模块在设计中主要用于控制整个系统的设置、虚拟时钟的切换、秒表的设置等。作为信号的输入端，电容触摸屏将接收到的电信号传输给MCU，由MCU作出判断后传输给存储器，同时控制LCD液晶显示屏的内容改变。

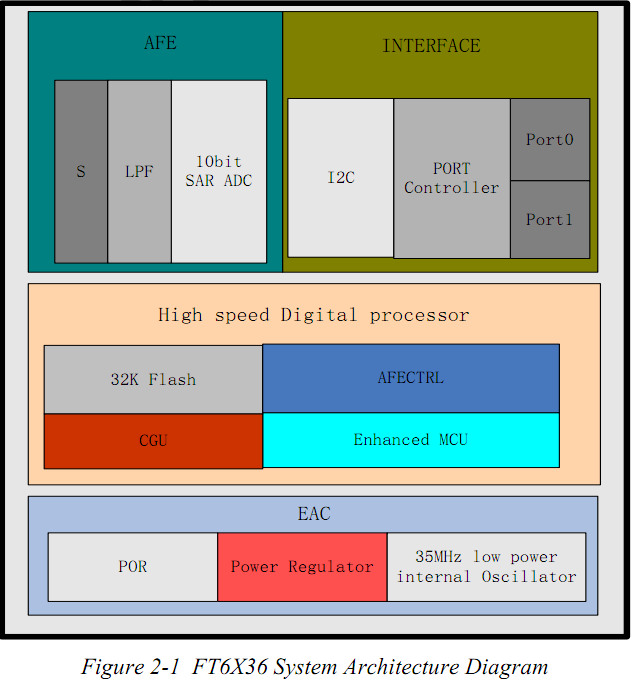


图2-8 FT6X36系统网络图解

### 2.1.7模块流程框图

该数码时钟有6个部分：电源系统、开机启动电路、MCU、SPI-flash存储器、LCD液晶显示屏、电容式触摸屏等六个部分组成。锂电池作为电源为整个工作系统供电，经过电源系统处理保证整个系统的供电电压恒定为3.3V。开机启动电路实现软开机，避免瞬间电流冲击。Cortex-M4是整个系统的控制核心，数据的显示、输入、输出都由它控制。SPI-flash存储器主要完成数据的存储和读取，闹钟定时及秒表功能的时钟数据的记忆和保存。LCD液晶显示屏用来显示图形界面以及年、月、日、时、分、秒等时间内容。电容式触摸屏实现触摸控制和参数设置、日期调整功能的实现。

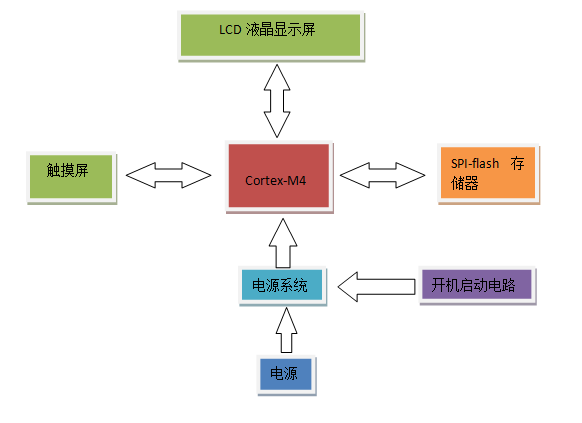


图2-9 系统电路原理框图

2.2实现功能描述

2.2.1时间的显示功能

在表盘上显示时间，即年、月、日、时、分、秒。是数码手表应该具有的基本功能。而这些功能的良好实现，离不开MCU的精确处理。

2.2.2 虚拟时钟

虚拟时钟是通过MCU控制液晶显示屏的显示，从而达到随意切换时钟界面的效果。通常，在彩色LCD面板中，每一个像素都是由三个液晶单元格构成，其中每一个单元格前面都分别有红色，绿色，或蓝色的过滤器。而不同的信号可以调制出不同的颜色配比，这样，通过不同单元格的光线就可以在屏幕上显示出不同的颜色。本设计是cortex-M4在9位数据总线模式下传输RGB6-6-6数据格式，通过18位数据传输模式，高效的向液晶屏传输数据，从而达到清晰的虚拟时钟效果。

2.2.3 闹钟功能

闹钟是数码手表的常用功能之一，影响单片机定时器准确性的原因 ：(1) 单片机电子时钟的计时脉冲基准是由外部晶振的频率经过12分频后提供，采用内部的定时/计数器来实现计时功能。所以，外接晶振频率精确度直接影响电子钟计时的准确性。 (2) 大家都知道从定时/计数器产生中断请求到响应中断需要3-8个机器周期，定时中断子程序中的数据入栈和重装定时/计数器的初值还需要占用数个机器周期，还有从中断入口转到中断子程序也要占用一定的机器周期。通过将定时数据写入SPI存储器，再设置中断，通过程序控制来实现闹钟功能。

2.2.4 秒表功能

常见的数码手表一般不支持秒表功能，而本款设计除了日常的时间显示以外，还加入了秒表功能，该功能可以实现精确定时。

2.2.5 图形界面的实现

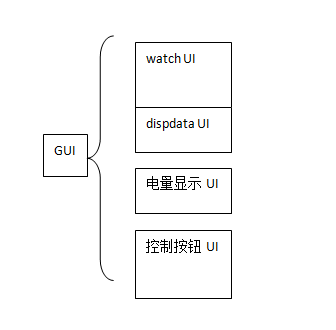
****

图2-10 图形界面的实现流程

图形界面和字体的实现是通过位图转换为二维数组实现的，而字体则是通过取模软件实现的。具体实现代码见附录。

2.3电路设计

2.3.1 时钟电路

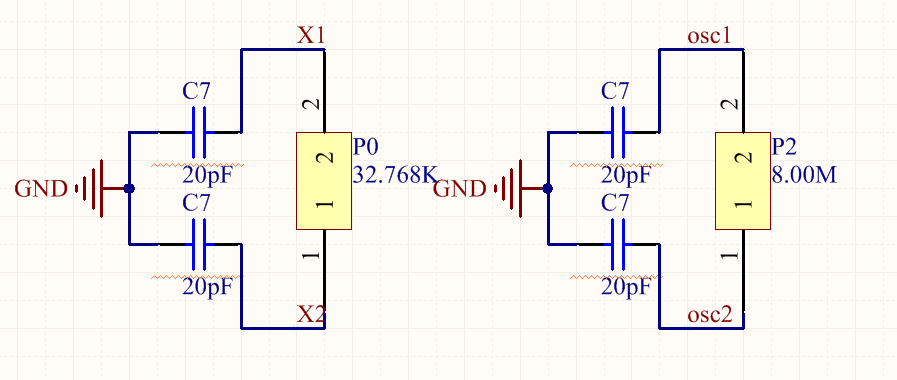


图 2-11 时钟电路

晶振的全称是叫晶体震荡器，在单片机系统里的作用非常大，它结合单片机内部的电路，产生单片机所必须的时钟频率，为系统提供基本的时钟信号。通常一个系统共用一个晶振，便于各部分保持同步。有些通讯系统的基频和射频使用不同的晶振，而通过电子调整频率的方法保持同步。单片机的一切指令的执行都是建立在这个基础上的，晶振所提供的时钟频率越高，那单片机的运行速度也就越快。晶振电路就是一个晶体振荡器，接在单片机内部的振荡电路上，两端所接电容是起振电容，频率越高，电容值应该越小。本电路采用20pF的电容作为起振电容，为电路提供准确而稳定的时序。

2.3.2 延时启动电路

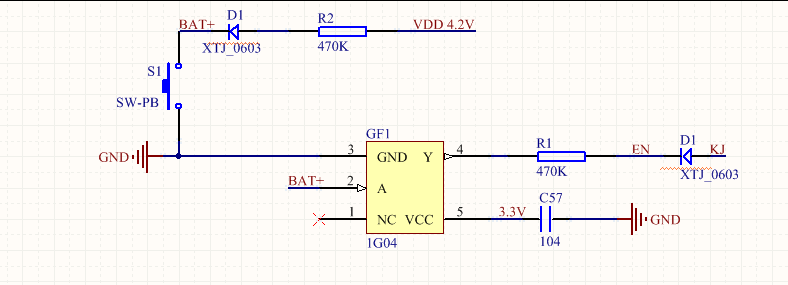


图 2-12 延时启动电路

延时启动电路是使电路在开关按下后并不马上启动电路，一方面是为了延迟一段时间再启动，其目的是为了避免开机时的过大电流对电路造成冲击，从而延长使用寿命。另一方面是为了因误操作本电路在开关机。按下后，4.2V电压经二极管提供一个稳定的压降，再通过反相器，反相器具有去抖动和滤波作用。

2.3.3 TF卡接口

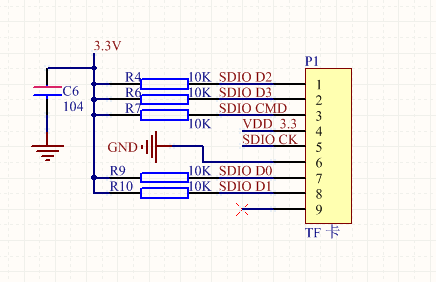


图 2-13 TF卡接口电路

TF卡座接口电路，因在外部存储器通讯协议中，SDIO协议通讯速度较快所以此处采用SDIO协议进行通讯。

2.3.4 SPI-flash存储电路

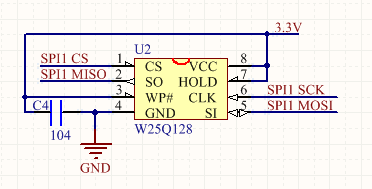


图2-14 SPI-flash存储电路

用于存储数据，其中数据包含汉字字库、位图、操作系统界面以及一些时钟的设置参数。以便在开机重启时重新调用这些参数。

### 2.3.5 电容触摸屏电路

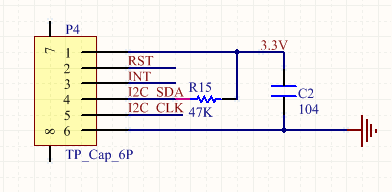


图2-15 电容触摸屏电路

电容触摸屏采用IIC通信协议，IIC总线只要求2条双向通信线路SDA(串行数据)和SCL(串行时钟)。每个器件都有一个唯一的地址,而且都可以作为一个发送器或接收器。如常用的存储器既可以接收又可以发送数据。除了发送器和接收器外器件在执行数据传输时也可以被看作是主机或从机。接到总线的器件都可以通过唯一的地址和一直存在的简单的主机从机关系软件设定地址主机可以作为主机发送器或主机接收器。

### 2.3.6 液晶显示屏

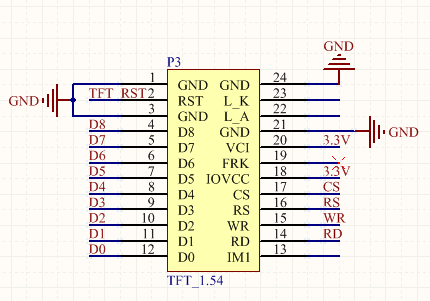


图2-16 液晶显示屏电路

液晶接口采用8080接口传输，其本质是并口数据形式，这样的好处是相比采用其他方式（SPI）速度更快，从而打达到更大的数据量传输。从而保证界面流畅运行。

### 2.3.7 电压转换电路

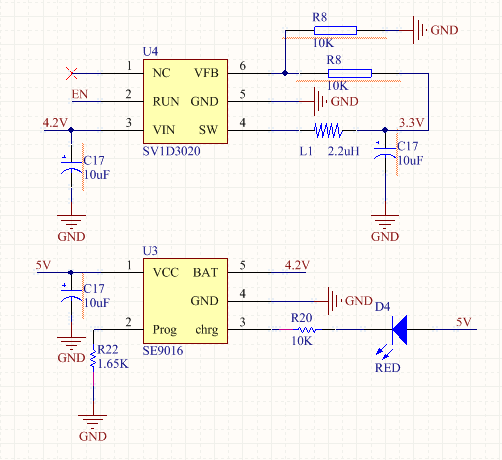
****

图2-17 电压转换电路

锂电池放电过程+整个供电系统需要提供3.3V供电电压，所以我们采用SV1D3020这颗DC-DC电压降压器将电压稳定到3.3V，并保证高达95%以上的效率和准确的电压值。SE9016是恒流/恒压线性充电芯片，主要面向单节锂电池充电应用。

### 2.3.8 cortex-M4芯片

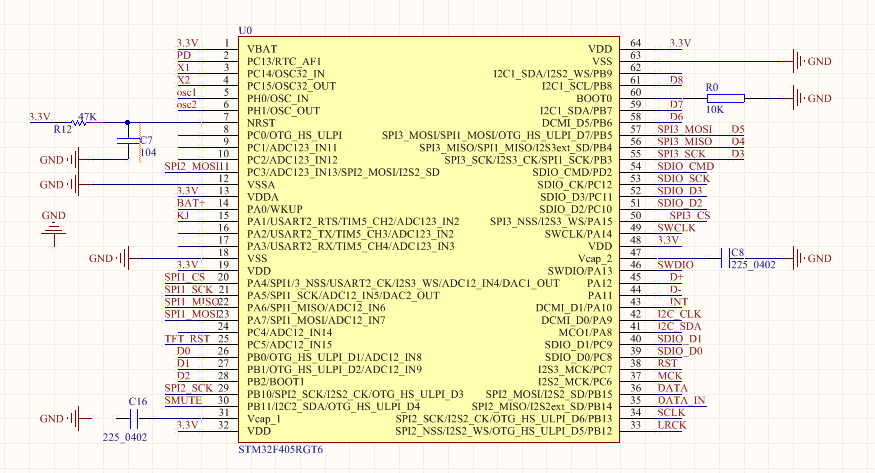


图2-18 cortex-M4芯片

Cortex-M4处理器采用一个扩展的单时钟周期乘法累加（MAC）单元、优化的单指令多数据（SIMD）指令、饱和运算指令和一个可选的单精度浮点单元（FPU)。这些功能以表现

ARMCortex-M系列处理器特征的创新技术为基础。包括RISC处理器内核，高性能32位CPU、具有确定性的运算、低延迟3阶段管道，可达1.25DMIPS/MHz；Thumb-2指令集，16/32位指令的最佳混合、小于8位设备3倍的代码大小、对性能没有负面影响，提供最佳的代码密度；低功耗模式，集成的睡眠状态支持、多电源域、基于架构的软件控制；嵌套矢量中断控制器（NVIC），低延迟、低抖动中断响应、不需要汇编编程、以纯C语言编写的中断服务例程，能完成出色的中断处理；工具和RTOS支持，广泛的第三方工具支持、Cortex微控制器软件接口标准（CMSIS）、最大限度地增加软件成果重用;CoreSight调试和跟踪，JTAG或2针串行线调试（SWD）连接、支持多处理器、支持实时跟踪。

此外，该处理器还提供了一个可选的内存保护单元（MPU），提供低成本的调试/追踪功能和集成的休眠状态，以增加灵活性。

2.4软件设计

2.4.1软件设计思路

Altium Designer 2014画图的具体步骤：

1．打开Altium Designer 2014，新建工程，再新建原理图库来创建一个电路原理图设计文件watch.SCHdoc。双击watch.SCHdoc进入原理图设计主界面。

2．在工作界面设计原理图

（1）熟悉工具

（2）熟悉快捷键的利用

（3）选择元器件，元件库里没有的元件自己编辑后放入原理图界面，编辑元件是从书上学的，下面是一些自己编辑的一些新元件。

（4）摆放元件

Space键：让元件作90°的旋转；

X键：元件左右对调。

Y键：元件上下对换。

（5）元件连线

当预拉线的指针移动到元件的引脚或其他电器特性线时，指针的中心将会出现一个黑点，她提示我们在当前状态下单击鼠标左键就会形成一个有效的电气连接。

（6）加入输入/输出端口

（7）更改元件属性

更改元件属性要注意元器件的封装号及footprint，设置元件的序号，设置元件的有效数值。

（8）DRC电气规则检查

如果有错误要检查原理图并改正其错误。

下图是用AltiumDesigner2014绘制的原理图，如图3-1所示：

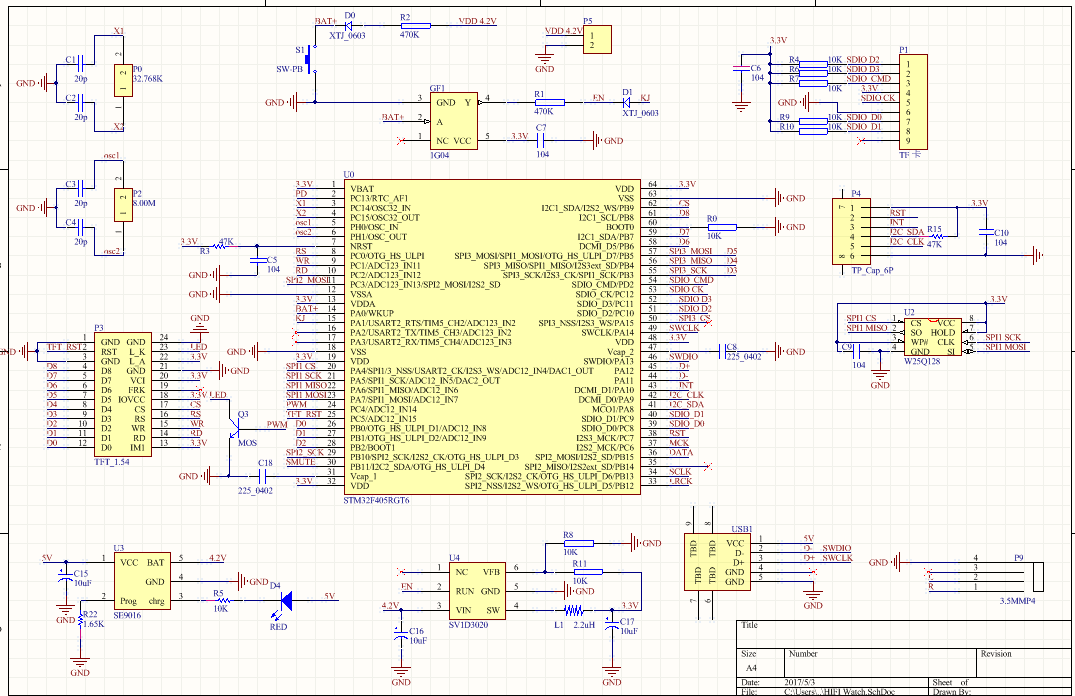


图2-19 AltiumDesigner2014绘制的原理图

AltiumDesigner2014制PCB板：

绘制一个完整的PCB电路板设计必需满足以下基本原则：

（1）电气连接正确。

（2）符合电路设计者意图。

（3）符合电路板安装的要求。

（4）元器件布局合理。

（5）电路板布线合理。

（6）便于安装和调试。

绘制一个完整的PCB电路板的基本步骤：

1．创建PCB设计文件

设置图纸区域的工作参数

2．先载入元器件封装库，再载入网络连接与元器件封装：在原理图编辑器中，执行菜单命令【新建】/【PCB】。

3．元器件布局

关键元器件预布局分成以下3个步骤：

（1）对所有的元器件进行分类筛选，找出电路板上的关键元器件。

（2）放置关键元器件。

（3）锁定关键元器件。

元器件自动布局包括以下两个基本步骤：

（1）设置元器件布局有关的设计规则。

（2）选择自动布局的方式，并进行自动布局的操作。

设置元器件布局有关的设计规则：

（1）设置元器件安全间距限制设计规则。

（2）设置元器件方向位置限制设计规则。

元器件自动布局

4．布线

设计布线规则：

（1）设置安全间距限制设计规则。

（2）设置短路限制设计规则。

（3）设置布线宽度限制设计规则。

预布线主要包括两个步骤：

（1）对重要的网络进行预布线。

（2）锁定预布线。在进行布线的过程中，有时候可能需要事先布置一些走线（如电源线和地线），以满足一些特殊要求，并有利于改善随后的自动布线结果。

选取菜单命令【工具】/【设计规则检查】，打开【运行DRC】对话框即可。结果如图。再进行设计规则检验。

1. 调试与实现
   1. 调试中遇到的重点与难点
2. 存在系统干扰。
3. 功耗问题

3、焊接问题

* 1. 解决方案

1、系统的抗干扰分析：

硬件抗干扰设计，在每个集成电路的电源和地之间加一个去藕电容，其作用一方面是作为集成电路的蓄能电容，提供和吸收该集成电路开、关瞬间的充放电能，另一方面是旁路掉该集成电路的高频噪声。在电源进入印刷版的地方加一个电容，具有较好的抑制噪声干扰作用。

软件抗干扰设计，在程序设计时我们通过设置软件陷阱和指令冗余的方式来抗干扰，这两种方式是提高软件可靠性，防止干扰造成误差，保证控制系统正常运行的有力措施。

2、功耗主要集中在TFT液晶屏背光功耗和处理器。可以通过降低背光或者降低处理器主频从而降低功耗。在待机状态下，可以关掉一些没用到的功能，提高供电系统的效率。

3、因为整个数字手表体积要求很小，所以器件选型应要求在0402及更小封装，所以给手工焊接带来很大难度。需要反复练习，尽可能熟悉过后再进行焊接。

* 1. 实现展示（附上仿真图或实物照片）

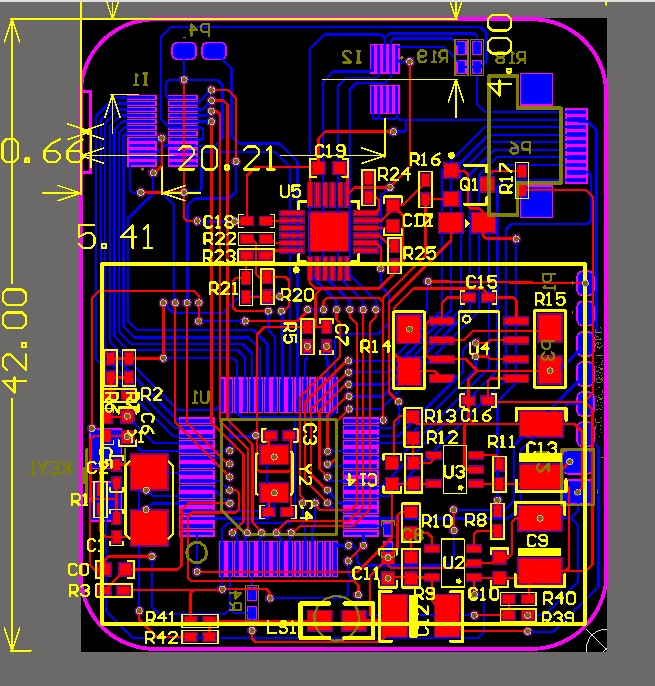


图3-1 AltiumDesigner2014绘制的PCB板

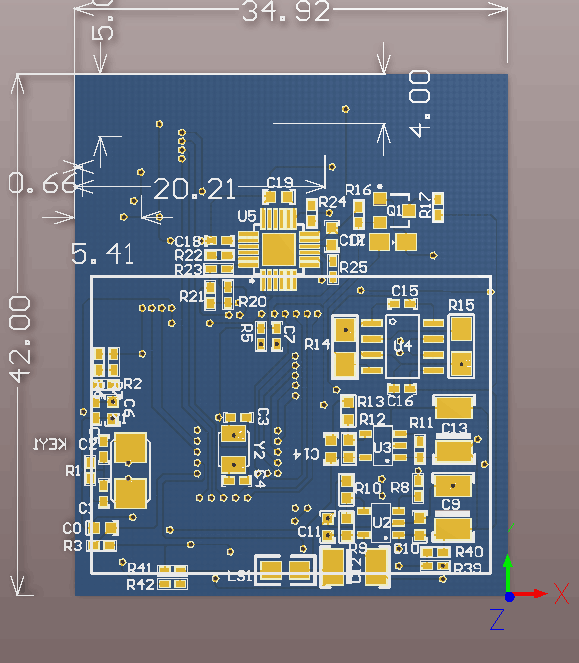


图3-2 AltiumDesigner2014绘制的PCB板3D试图正面

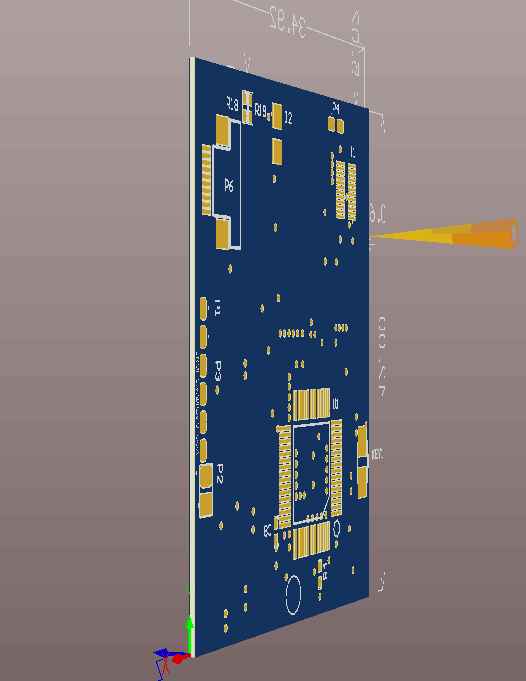


图3-3 AltiumDesigner2014绘制的PCB板3D试图侧面

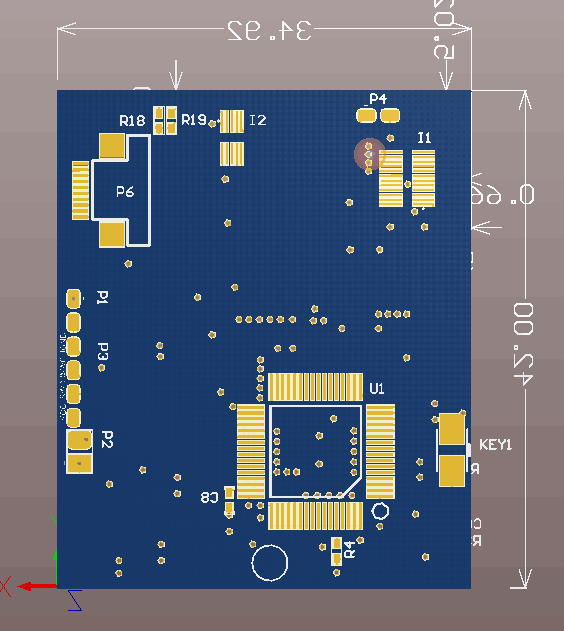


图3-4 AltiumDesigner2014绘制的PCB板3D试图底面

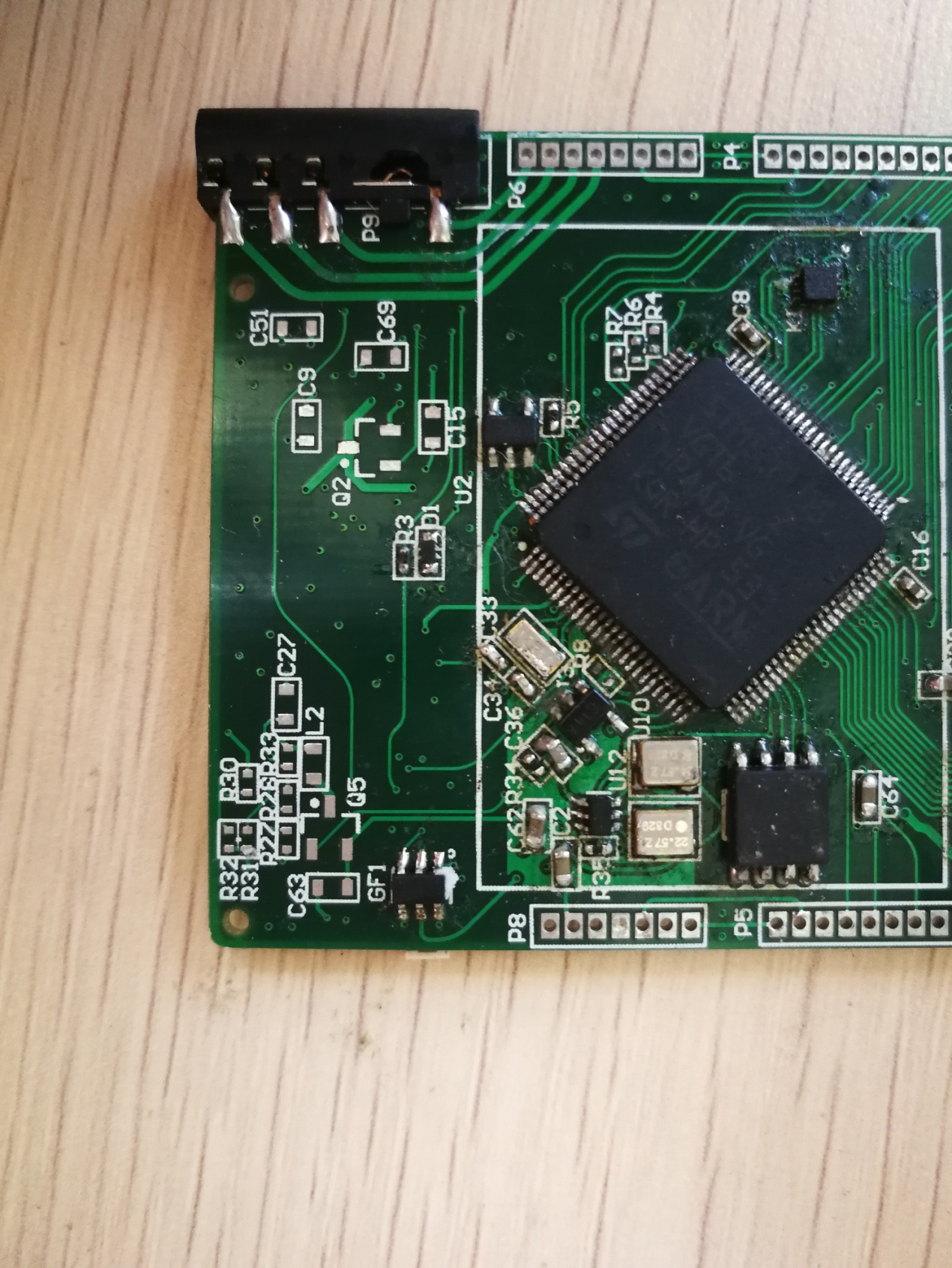


图3-5 PCB板实物图

`

图3-6 实物图



图3-7 实物图



图3-8 实物图

* 1. 实现过程
     1. 电路的安装

按照布局图或PCB图把元件逐一焊接在电路板上，对于二极管或电解电容等有极性器件要用仪器判断好后在焊接。

### 3.4.2 电路的焊接

焊接工具：电烙铁、所需元器件、导线、焊锡丝、松香、万用表、万能板。

主体思想：由于该电路直接为PCB板，有人也许认为手工焊接非常容易，没有技术含量，其实不然。正确手工焊接的方法，需要深入理解焊接要素和通过长期的练习，才能保证焊接的质量。正确的焊接方法：焊接时利用烙铁头的对元件引线和焊盘预热，烙铁头与焊盘的平面最好成45°夹角，等待焊金属上升至焊接温度时，再加焊锡丝。被焊金属未经预热，而将焊锡直接加在烙铁头上，使焊锡直接滴在焊接部位，这种焊接方法常常会导致虚焊。

### 3.4.3 插件原件焊接的步骤

1．插入将插件元件插入电路板标示位置过孔中，与电路板紧贴至无缝为止。如未与电路板贴紧，在重复焊接时焊盘高温易使焊盘损伤或脱落，物流过程中也可导致焊盘损伤或脱落。

2．预热烙铁与元件引脚、焊盘接触，同时预热焊盘与元件引脚，而不是仅仅预热元件，此过程约需1秒钟时间。

3．加焊锡加到焊盘上（而不是仅仅加在元件引脚上），待焊盘温度上升到使焊锡丝熔化的温度，焊锡就自动熔化。不能将焊锡直接加在烙铁上使其熔化，这样会造成冷焊。

4．加适量的焊锡，然后先拿开焊锡丝。

5．焊后加热拿开焊锡丝后，不要立即拿走烙铁，继续加热使焊锡完成润湿和扩散两个过程，直到焊点最明亮时再拿开烙铁，不应有毛刺和空隙。

6．冷却在冷却过程中不要移动插件元件。

### 3.4.4 焊接要素

1．焊接温度和时间焊锡的最佳温度为350ºC，温度太低易形成冷焊点，高于400ºC易使焊点质量变差，且容易导致焊盘（铜皮）变形或脱落。焊接时间：完成润湿和扩散两个过程需2-3S，1S仅完成润湿和扩散两个过程的35%。一般IC、三极管焊接时间小于3S，其他元件焊接时间为4-5S。

2．焊锡量适当焊点上焊锡过少，机械强度低。焊锡过多，会容易造成绝缘距离减小、焊点相碰或跳锡等现象。

电烙铁使用注意事项：

电烙铁温度升高后，首先应将烙铁尖点上薄薄的一层焊锡，避免烙铁的尖因氧化而不沾锡。使用过程中，烙铁尖表面应一直保持有薄薄的焊锡层，多余的焊锡可轻轻甩在烙铁架上，或用一块湿布（湿海绵）擦拭一下。暂时不用时，应将电烙铁温度调至最低。元件全部焊接完成后，在仔细检查几遍，确保器件连接正确后，方可通电测试。

### 3.4.5硬件测试

1．电路板焊好电子元件后,要仔细检查电路板有无焊焊接错误的地方,特别要注意有极性的电子元件,如电解电容,桥式整流堆,一旦极性器件焊反，就会有烧毁元器件的风险,应特别注意。

2．接上变压器,先检测MCU供电和电源管理部分是否有电源供给,要注意电路是否有短路的情况。

1. 总结

自选题到调试硬件电路成功，一路上，也算是付出了不少的努力。通过多次上图书馆查阅相关方面的资料和向指导老师请教，成功设计出了电路图。但是，理论和实际终归有些差距，调试的过程中有些参数很难达到，给设计带来了一定的困扰，最后通过和指导老师交流及自己的努力，才得以将硬件电路调试成功。

接着就是开题报告、外文翻译和文献综述的撰写，起初认为按照老师给的模板修改下即可，结果交给指导老师审查的时候，有很多地方都不符合要求。这让我认识到，做事情一定要踏踏实实，不能心存侥幸。

我的收获总体概括起来大致可分为以下三点：

1、课题设计前应仔细阅读资料文献，充分理解课题中所涉及的大小知识点，这样才不至于在设计中犯一些低级错误，并能有效的节省时间。

2、设计中所遇到的一些问题要尽量独立分析，摸根问底，这样才可以得到更深刻的体会。

3、电路设计是一个追求最适效果的过程，往往对某一项要求的达成会影响另一个功能的正常实现，所以在明确此电路所追求的功能的同时在电路或者元件选择中要做到正确的取舍。

参考文献

[1] 《模拟电子技术》,西安电子科技大学, 江小安 主编,西北大学出版社,2008. 45~96

[2] 《模拟电子技术基础》,武汉理工大学, 吴友宇 主编,清华大学出版社,2008. 50~150

[3]《模拟电子技术基础》,清华大学教研组, 童诗白 主著,高等教育出版社,2001. 45~96

[4]《电子技术基础》,华中工学院电子学教研室编, 康华光 主编,高等教育出版社,2005.30~80

[5]《电子技术工艺基础》,王天曦, 李洪儒 主编,清华大学出版社,2000.10~100

[6]《电子线路设计》,(第三版)华中科技大学,谢自美 主编, 华中科技大学出版社,2005.12~52

[7]《模拟电子技术》,程开明 主编, 重庆大学出版社,1992.50~100

[8]《TFT-LCD原理与设计》，北京，马群刚 著，电子工业出版社，2011.16~345

[9]《发光原理与发光材料》，电子科技大学，祁康成 主编，曹贵川 副主编，电子科技大学出版社，2012,52~93

[10]《现代光电子成像技术概论》，北京理工大学，向世明 主编，北京理工大学出版社，2014,76~101

[11]大学生电子设计网：http://www.uednet.com

[12]电子课程设计网：http://www.edatime.com

[13]电子发烧友网：http://www.elecfans.com

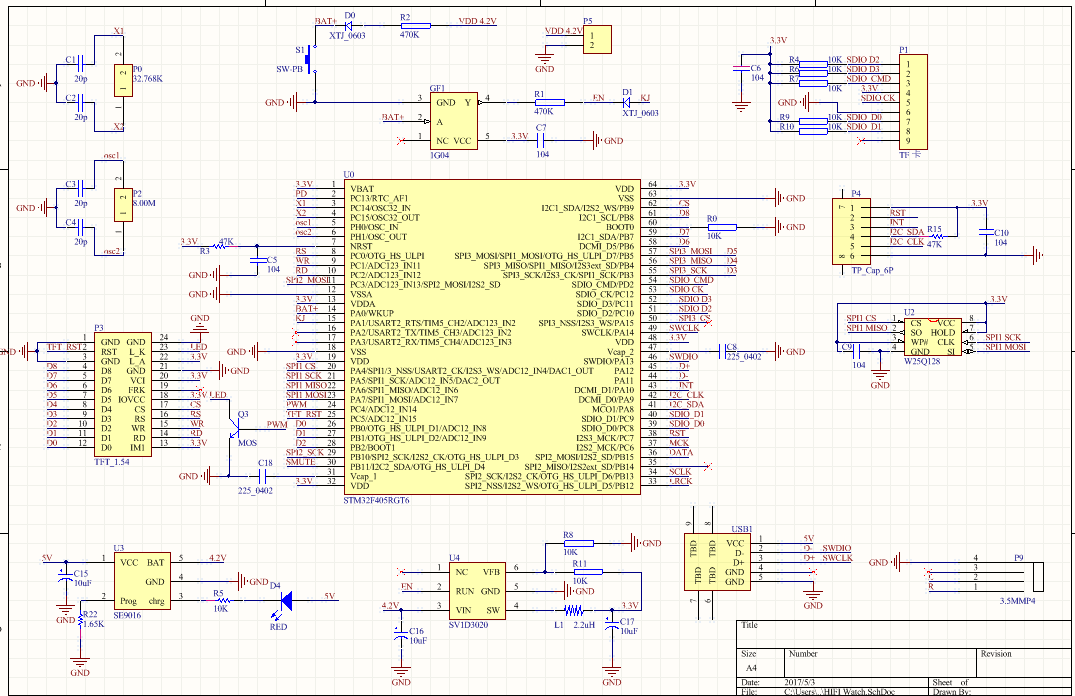
[14]百度百科：http://baike.baidu.com

致谢

我要首先感谢我的父母，辛苦养育我，送我上学。然后我才能来到这个学校，与大家一起度过这四年，留下余生中最美丽灿烂的忆记。其次，我要感谢我的老师们，在生活上和学习上给予我的帮助，尤其是因为生病而郁郁寡欢的时候，各位老师对我的关心和开导，犹记在心。本课题在选题及设计过程中得到李伟老师的悉心指导。李老师多次询问设计进程，并为我指点迷津，帮助我开拓研究思路，精心点拨、热忱鼓励。李老师无论是在理论上还是在实践中，都给了我很大的帮助，李老师一丝不苟的作风，严谨求实的态度，踏踏实实的精神，是我学习的好榜样。还有我的同学们，感谢吴利波同学在整个设计过程中给我的指导与帮助。大学四年，能有你们陪伴，一起从青涩走向成熟，从学校迈入社会，是我的荣幸。很庆幸在短暂的人生中与你们相遇，亦感激美好时光可以与你们一起共度。海阔凭鱼跃，天高任鸟飞，愿各位能比鲲鹏，不枉此生。

附录

## 附录一：硬件设计原理图



附图1 系统总框图

## 附录二：数字时钟函数实现

oid Disp\_Watch(void)

{

GUI\_AUTODEV AutoDev; // Object for banding memory device

PARAM Param; // Parameters for drawing routine

//Param.sec = My\_Clock.sec;

//Param.day = My\_Clock.date;

//Param.week = My\_Clock.week;

GUI\_MEMDEV\_CreateAuto(&AutoDev); //创建自动设备对象

//计算旋转角度

Param.MinAngle = (-My\_Clock.min\*6\*3.1412/180);

Param.HourAngle = (-(My\_Clock.hour\*5+My\_Clock.min/12.0)\*6\*3.1412/180);

GUI\_RotatePolygon(Param.MinPoints,\_aPointMin, 4, Param.MinAngle); //按照指定角度旋转多边形

GUI\_RotatePolygon(Param.HourPoints,\_aPointHour, 4, Param.HourAngle); //按照指定角度旋转多边形

//绘制

GUI\_MEMDEV\_DrawAuto(&AutoDev, &Param.AutoDevInfo, &\_Draw, &Param); //使用自动设备对象绘制。

GUI\_MEMDEV\_DeleteAuto(&AutoDev);//删除自动设备对象

}

## 附录三：图形界面绘图代码

GUI\_CONST\_STORAGE GUI\_BITMAP bmwatch = {

240, // xSize

240, // ySize

480, // BytesPerLine

16, // BitsPerPixel

(unsigned char \*)\_acwatch, // Pointer to picture data

NULL, // Pointer to palette

GUI\_DRAW\_BMP565

};

static void \_Draw(void \* p)

{

u8 x,y;

PARAM \* pParam;

pParam = (PARAM \*)p;

if (pParam->AutoDevInfo.DrawFixed)

{

GUI\_DrawBitmap(&bmwatch,0, 0);

}

GUI\_SetColor(GUI\_RED);

GUI\_AA\_FillPolygon(pParam->HourPoints, 4, MAG \* 120, MAG \* 120);

GUI\_AA\_FillPolygon(pParam->MinPoints, 4, MAG \* 120, MAG \* 120); x=120+(95\*cos((float)(3.1412/2-6\*(My\_Clock.sec)\*3.1412/180)));

y=120-(95\*sin((float)(3.1412/2-6\*(My\_Clock.sec)\*3.1412/180)));

GUI\_AA\_DrawLine(x\*2,y\*2,120\*2,120\*2);

if (pParam->AutoDevInfo.DrawFixed) {

GUI\_FillCircle(120,120,8);

GUI\_SetTextMode(GUI\_TM\_TRANS);

GUI\_SetColor(GUI\_YELLOW);

GUI\_SetFont(&GUI\_Font24\_ASCII);

GUI\_DispStringAt("2016",53,170);

if((My\_Clock.month)<10)

{

GUI\_DispDecAt(My\_Clock.month,123,170,1);

}

if((My\_Clock.date)<10)

{

GUI\_DispDecAt(My\_Clock.date,157,170,1);

}

else

{

GUI\_DispDecAt(My\_Clock.date,153,170,2);

}

//

GUI\_SetFont(&GUI\_Fontmytimefont);

GUI\_DispStringAt("\xe5\xb9\xb4",103,170);

GUI\_DispStringAt("\xe6\x9c\x88",133,170);

GUI\_DispStringAt("\xe6\x97\xa5",175,170);

GUI\_DispStringAt("\xe5\x91\xa8",160,100);

switch(My\_Clock.week)

{

case 1 :GUI\_DispStringAt("\xe4\xb8\x80",180,100); break;

case 2 :GUI\_DispStringAt("\xe4\xba\x8c",180,100); break;

case 3 :GUI\_DispStringAt("\xe4\xb8\x89",180,100); break;

case 4 :GUI\_DispStringAt("\xe5\x9b\x9b",180,100); break;

case 5 :GUI\_DispStringAt("\xe4\xba\x94",180,100); break;

case 6 :GUI\_DispStringAt("\xe5\x85\xad",180,100); break;

case 7 :GUI\_DispStringAt("\xe6\x97\xa5",180,100); break;

default: break;

}

}

}

## 附录四：cortex-M4数据手册

