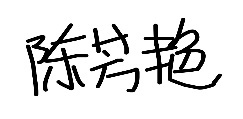
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 班 级： | | | | | | | | | | | | | | | | | 信工1601 | | |
| 北京化工大学标识-黑白学 号： | | | | | | | | | | | | | | | | | 2015014449 | | |
|  | | | | 北京化工大学 | | | | | | | | | | |  | | | | |
|  | | | 毕业设计(论文) | | | | | | | | | | | | |  | | | |
|  | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | |  | | | | | | | | | |
|  | 题 目 | | | | **基于STM32G071的电子钟设计与实现** | | | | | | | | | | | | |  | |
|  | |  | | | |  | | | | | | | | | | | |  | |
|  | | | | | | | | | | | | | | | | | | | |
|  | | | | 专 业 | | | **电子信息工程** | | | | | | | |  | | | | |
|  | | | | 学 生 | | | **陈芳艳** | | | | | | | |  | | | | |
|  | | | | 指导教师 | | | **何 宾** | | | | | | | |  | | | | |
|  | | | | | | | **2021** | **年** | **6** | | **月** | **20** | **日** |  | | | | |

**诚信声明**

本人声明：

本人所呈交的毕业设计（论文），是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

作者签名：  日期： 2021 年 6 月 20 日

**毕业设计(论文)任务书**

设计(论文)题目： 基于STM32G071的电子钟设计与实现

学院： 信息科学与技术学院 专业 电子信息工程 班级： 信工1601

学生： 陈芳艳 指导教师： 何宾（讲师） 专业负责人： 尹嫱

1．设计（论文）的主要任务及目标

本次课题的任务是利用STM32嵌入式芯片内部的RTC模块、带有I2C接口的OLED显示模块，实现电子时钟显示、闹钟响铃功能，目标是了解嵌入式工作原理和I2C传输协议、实时时钟和OLED显示原理，锻炼理论结合实践和动手的能力。

2．设计（论文）的主要内容及要求

该系统主要以STM32G071为核心微控机，结合OLED显示模块、扬声器模块、按键模块等，实现电子时钟显示、闹钟响铃等功能。

具体技术指标：

（1）能实现实时时钟的显示和修改；

（2）按键控制切换模式；

（3）能设置闹钟，实现闹钟响铃功能；

（4）能显示图片、表情包。

3．主要参考文献

[1]C语言程序设计 谭浩强 清华大学出版社2000.1

[2]电子设计从零开始 杨欣、王玉凤等 清华大学出版社 2005.10

[3]STM32固件库使用手册

[4]嵌入式系统设计（基于STM32F4） 徐灵飞等 电子工业出版社 2020.8

4．进度安排

|  |  |  |
| --- | --- | --- |
|  | 设计（论文）各阶段名称 | 起 止 日 期 |
| 1 | 掌握电子时钟模型和原理、OLED显示电路、按键模块、扬声器模块 | 2020-12-26—2-20 |
| 2 | 掌握单片机的工作原理和程序编写方法 | 2021-2-21—3-20 |
| 3 | 掌握STM32G0X1工作原理、RTC时钟、I2C串口通信协议 | 2021-3-21—4-20 |
| 4 | 程序编写、硬件实现、软件调试、系统测试 | 2021-4-21—5-15 |
| 5 | 撰写毕业论文,答辩材料编写 | 2021-5-16—6-5 |

**基于STM32G071的电子钟设计与实现**

摘 要

嵌入式系统的软件核心大多为实时操作系统。当中断发生，非实时系统需要考虑全局的任务调动、内存管理情况，最后需等待一定的时间来响应中断。而嵌入式的实时性体现在输入一个中断时，系统能暂停当下所有任务去处理中断，以最快的速度应答需求。在紧急情况下，实时操作系统有极大的应答优势。

当中断发生，实时操作系统STM32 MCU立即停止所有任务，但RTC实时时钟依然保持运行状态。本文使用STM32G071 MCU内集成的实时时钟模块以及带有I2C接口的OLED实现实时时钟的显示和修改、闹钟和响闹的设置，对实时时钟和I2C总线的原理和功能，以及实时时钟具体的实现过程进行了介绍。

**关键词：**STM32,I2C,RTC,OLED

**DESIGN AND IMPLEMENTATION OF ELECTRONIC CLOCK BASED ON STM32G071**

**ABSTRACT**

The core of embedded software is mostly a real-time operating system. When an interruption occurs, a non-real-time system needs to consider global task redeployment, memory management, and finally a certain amount of time to respond to the interruption. Embedded real-time performance is embodied in the input of an interrupt, the system can suspend all the current tasks to deal with the interrupt, respond immediately, and respond to the demand at the fastest speed. In emergency situations, real-time operating systems have great advantages in response.

When the interrupt program starts, the real-time operating system STM32 MCU immediately stops all tasks, but the RTC real-time clock remains running.In this paper, the real-time clock module integrated in STM32G071 MCU and OLED with I2C interface are used to display and modify the real-time clock. The principle and function of the real-time clock and the I2C bus, and the concrete realization process of the real-time clock are introduced.

**KEY WORDS:** STM32，I2C，RTC，OLED

目录

[**前 言** 1](#_Toc75680965)

[第1章 绪论 2](#_Toc75680966)

[1.1 课题研究的背景和意义 2](#_Toc75680967)

[1.2 平板显示技术研究现状 2](#_Toc75680968)

[1.2.1 OLED的发光原理 2](#_Toc75680969)

[1.2.2 OLED与LCD比较 3](#_Toc75680970)

[1.2.3 OLED的最新发展 3](#_Toc75680971)

[1.3 嵌入式发展现状 3](#_Toc75680972)

[1.4 论文主要工作和结构安排 4](#_Toc75680973)

[第2章 电子时钟的设计原理 6](#_Toc75680974)

[2.1 RTC工作原理 6](#_Toc75680975)

[2.1.1 可编程的报警 7](#_Toc75680976)

[2.1.2 RTC初始化和配置 7](#_Toc75680977)

[2.1.3 读日历 7](#_Toc75680978)

[2.1.4 复位RTC 7](#_Toc75680979)

[2.1.5 时间戳功能 8](#_Toc75680980)

[2.1.6 RTC低功耗模式 8](#_Toc75680981)

[2.1.7 RTC中断 8](#_Toc75680982)

[2.2 串行同步传输接口（I2S）原理 8](#_Toc75680983)

[2.2.1 I2C主设备时钟 10](#_Toc75680984)

[2.2.2 从停止模式唤醒 11](#_Toc75680985)

[2.2.3 数据传输的处理 12](#_Toc75680986)

[2.2.4 从设备模式 13](#_Toc75680987)

[2.2.5 中断和DMA 14](#_Toc75680988)

[2.2.6 低功耗模式 14](#_Toc75680989)

[2.3 OLED显示模块原理 15](#_Toc75680990)

[2.4 本章小结 16](#_Toc75680991)

[第3章 系统总体方案设计 17](#_Toc75680992)

[3.1 系统总体结构 17](#_Toc75680993)

[3.2 I2C模块 17](#_Toc75680994)

[3.2.1 通用I2C协议 17](#_Toc75680995)

[3.2.2 I2C驱动OLED 18](#_Toc75680996)

[3.2.3 I2C的写模式 18](#_Toc75680997)

[3.2.4 I2C的数据位传输 20](#_Toc75680998)

[3.2.5 GPIO模拟与I2C内置模块对比 20](#_Toc75680999)

[3.3 OLED模块 21](#_Toc75681000)

[3.4 RTC模块 22](#_Toc75681001)

[3.5 本章小结 22](#_Toc75681002)

[第4章 软件部分 23](#_Toc75681003)

[4.1 初始化 23](#_Toc75681004)

[4.1.1 STM32CubeMX初始化 23](#_Toc75681005)

[4.1.2 MDK-ARM参数初始化 24](#_Toc75681006)

[4.2 OLED初始化 24](#_Toc75681007)

[4.3 显示壁纸 24](#_Toc75681008)

[4.4 显示日期 25](#_Toc75681009)

[4.4.1 获取日期 25](#_Toc75681010)

[4.4.2 字符串函数 25](#_Toc75681011)

[4.4.3 字符库 25](#_Toc75681012)

[4.5 循环判断模式 26](#_Toc75681013)

[4.5.1 显示时钟 26](#_Toc75681014)

[4.5.2 修改日历和闹钟 26](#_Toc75681015)

[4.6 两个中断产生一个闹钟 27](#_Toc75681016)

[4.6.1 闹钟模式 27](#_Toc75681017)

[4.6.2 非闹钟模式 27](#_Toc75681018)

[4.7 本章小结 28](#_Toc75681019)

[第5章 硬件实现 29](#_Toc75681020)

[5.1 硬件接线 29](#_Toc75681021)

[5.2 实验结果 29](#_Toc75681022)

[结 论 32](#_Toc75681023)

[参考文献 33](#_Toc75681024)

[致 谢 35](#_Toc75681025)

前 言

在军事、航空、船舶、工业和医疗领域，对实时性要求高，以此确保安全和可靠性。PC机、电脑、手机是非实时操作系统，当按下一个按键系统不能立即做出应答，需要等待一定时间。当中断发生，非实时系统需要考虑全局的任务调动、内存管理情况，最后以一定的时间来响应中断。而嵌入式的实时性体现在输入一个中断时，系统能暂停当下所有任务去处理中断，以最快的速度应答需求。在紧急情况下，实时操作系统有极大的应答优势。

设备间通信离不开通信协议，无论是嵌入式系统，还是PC机。通信协议就是设备间信息能够互通需要遵循的规则。通信协议就像设备间的翻译，将数据信息以设备能识别的方式在设备间传递。比较主流的嵌入式通信协议有USART串口、SPI、I2C等。实时时钟是STM32内部的时钟逻辑，在初始化配置启动后，就可自主运行。且实时时钟功耗极低，机械表中一颗小小的纽扣电池就可续航相当长时间。

本文使用STM32G071 MCU内集成的实时时钟模块以及带有I2C接口的OLED实现实时时钟的显示和修改，介绍了实时时钟和I2C总线的原理和功能，以及实时时钟的具体实现过程。

# 绪论

## 1.1 课题研究的背景和意义

信息化产业离不开显示技术，显示屏在生活中十分常见。小到可穿戴电子产品，大到巨型屏幕，显示技术无时无刻不在传输图像和视频信息。嵌入式由于便携、实时性和专用性强，一直是融合平板显示技术的热门领域。嵌入式的发展促进了平板显示技术的实用化进程。嵌入式自带实时时钟，可利用嵌入式自带的实时时钟做一个电子日历和时间显示且能闹钟响铃的电子钟，应用到许多可穿戴设备和日常使用的屏幕显示设备中。

## 1.2 平板显示技术研究现状

市场上主流的几种显示技术：占有大块市场的液晶显示（LCD）成本较低,技术已相当成熟。有机发光二极管显示（OLED）和量子点显示（QLED）的亮点在于柔韧性好可弯曲展示。而激光显示技术（LD）攻坚全真色彩显示[1]。本节主要介绍OLED的研究现状和最新进展。

### 1.2.1 OLED的发光原理

加电压在AMOLED阴阳极两端，电路中流动的电子和空穴在有机发光层碰撞，引起激发态，能阶差转化为光和热释放，发光效率高，稳定性好。发出的光的颜色取决于有机发光材料的类别。最基本的是红绿蓝，再由此三原色产生各种颜色。缺陷在于易衰变的蓝光材料缩短了OLED的使用寿命[2]。

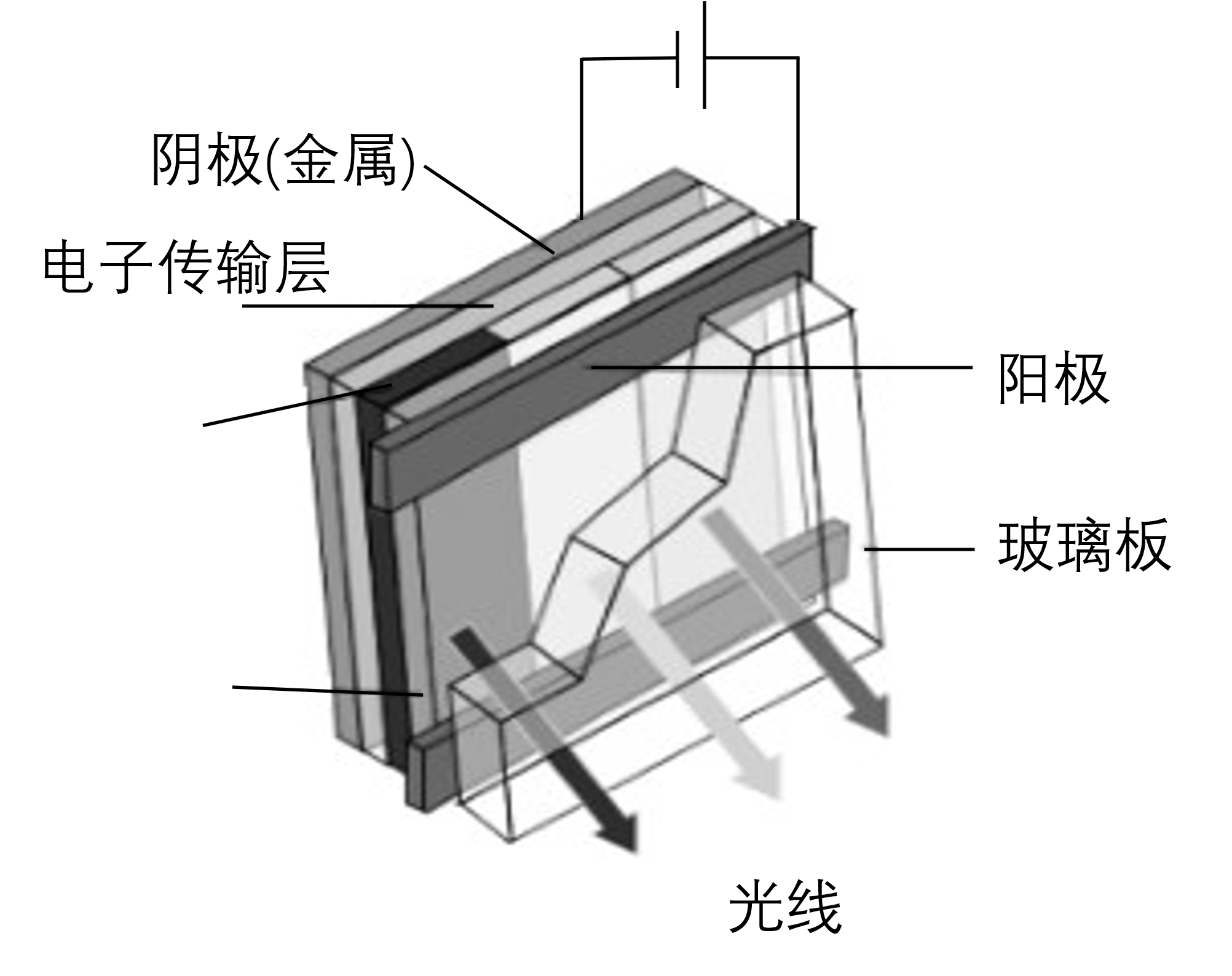


图 1‑1 AMOLED的基本结构

AMOLED的亮度和像素点是由驱动电路控制的，驱动电路上有电源信号、输入信号和TFT开关控制信号。AMOLED的基本结构如图 1‑1所示。

### 1.2.2 OLED与LCD比较

如表1-1和表1-2所示，将OLED显示技术与传统LED显示技术从结构、原理、性能等方面进行了对比。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 基板 | 发光层材料 | 偏光片 | 背光模组 | 彩色滤光片 | 扩散膜 | 导光板 | 发光原理 | 显示原理 |
| LCD | 玻璃基板 | LED/CCFL | 线偏光2层 | 有 | 有 | 有 | 有 | 背光源发光 | 非主动发光显示 |
| OLED | 玻璃/塑料基板 | 有机发光材料 | 圆偏光1层 | 无 | 无 | 无 | 无 | 有机发光层发光 | 主动发光显示 |

表 1‑1不同显示技术的结构与原理比较

表 1‑2不同显示技术的性能比较

|  | 对比度 | 可视度 | 响应时间 | 功耗 | 厚度 | 寿命 | 柔性 | 成本 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LCD | >10000:1 | <150º，有色偏 | ms，较慢 | 高 | >12mm | 高 | 不可弯曲 | 低 |
| OLED | >1000000:1 | 接近180º，无色偏 | μs，较快 | 较低 | <15mm | 低 | 可弯曲 | 高 |

由表 1‑1和表 1‑2可知，与LCD相比，OLED结构更简单，制作工序更少，质量更轻薄，量产成本更低。OLED丢弃了传统的背光源，转而使用自主发光的有机发光材料，不需要背光，比LCD的显示亮度更高。OLED响应速度快，柔韧性好可弯曲显示，显示色域更宽[3]。但OLED有机发光材料容易和空气中的氧气和水汽发生化学反应而使显示效果大打折扣，以及蓝光材料易衰变，是OLED显示技术亟待解决的瓶颈。

### 1.2.3 OLED的最新发展

近年来OLED显示技术的发展如火似荼，各大品牌商都在研制最新OLED显示产品。如创维、小米的OLED超薄电视。随着各大企业以及研究机构对柔性OLED开展了广泛的应用研究。柔性显示器实用化进程大大加速。2016年10月，荷兰用陶瓷材料基板做出了防潮性更好的OLED面板，提高了面板寿命。2016年12月，韩国两家公司合作宣布在薄膜封装中使用能阻止AMOLED透水透氧能力的原子层淀积（AOLD）技术，提升显示器寿命及性能[4]。

## 1.3 嵌入式发展现状

PC机、手机、电脑是非实时操作系统，当按下一个按键不能立即做出应答，需要等待一定时间。即使手机刷新速率越来越快，但它的非实时性特质不变。当用户端的中断发生，非实时系统需要考虑全局的任务调动、内存管理情况，最后以一定的时间来响应中断。而嵌入式的实时性体现在输入一个中断，系统能立即做出应答。在紧急情况下，实时操作系统有极大的应答优势。当中断发生，嵌入式能暂停当下所有任务去处理中断，以最快的速度应答需求。在军事、航空、船舶、工业和医疗领域，对实时性要求高，以此确保不会出现滞后反应，保证系统的安全和可靠性。

嵌入式较PC机的优势在于其体积小、功耗低、价格实惠、且具备便携性和可靠性。 意法半导体公司生产的STM32系列芯片由于覆盖范围广，且进入中国市场最早，应用最广泛、使用者众多。其中，Cortex系列不同于之前的ARM11、ARM9，做出了新的全系列的方案，囊括低性能的Cortex-M0到高端的Cortex-A8, Cortex-A15。Cortex-M系列主要应用于嵌入式开发，例如控制器、智能控制、电机控制、彩屏显示；Cortex-A8、Cortex-A9则更多应用于手机，具有更大彩屏更多应用，性能和速率更高。低功耗设备，如基于电池的智能手环，多基于Cortex-M0搭建而成。Cortex-M0针对低成本和简单的功能，如电池设备、电子钟。Cortex-M3针对性能的效率，Cortex-M4针对有效的数字信号控制。

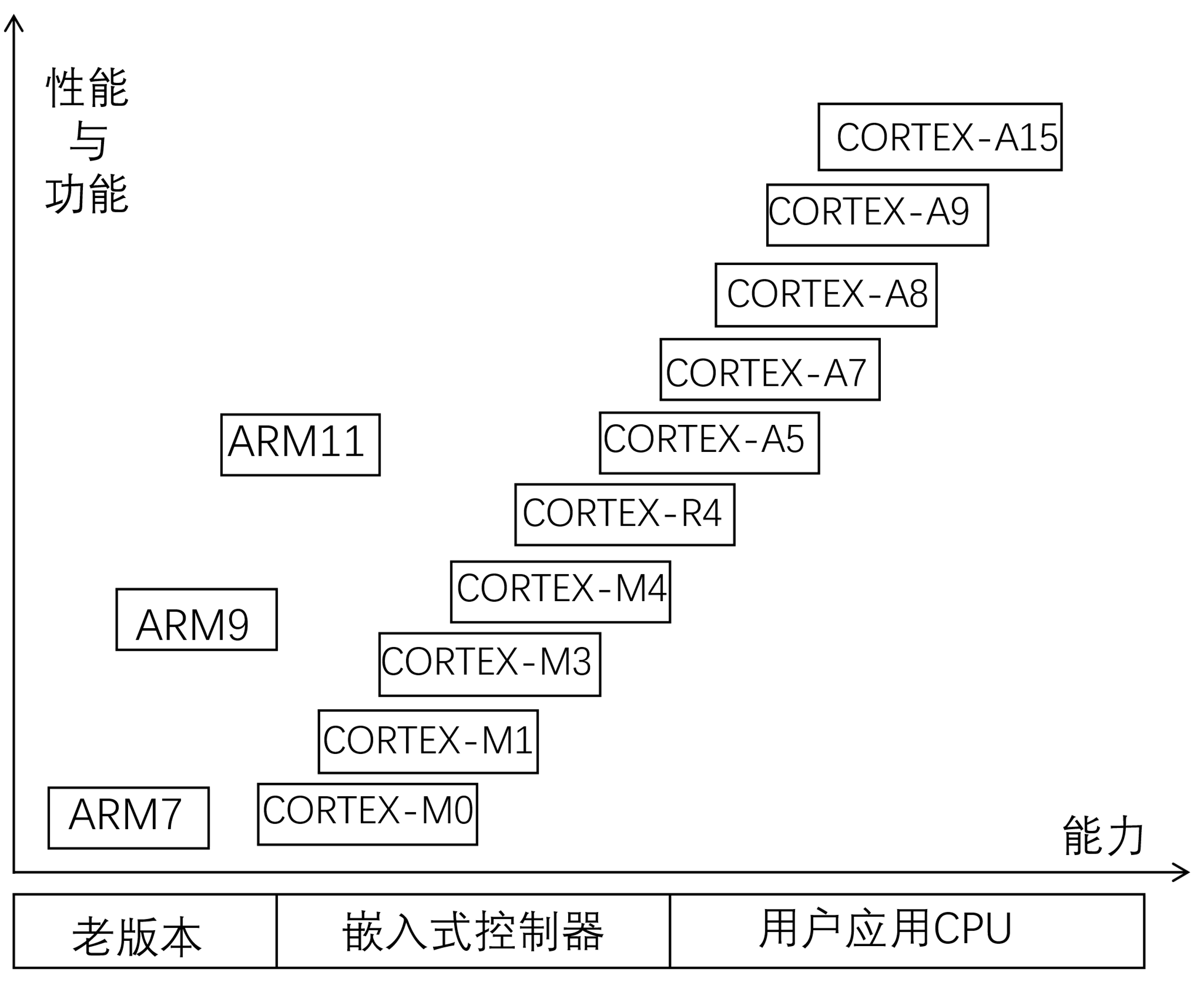


图 1‑2 嵌入式芯片的发展情况

嵌入式开发尽可能追求极致性价比，所用芯片并不要求价格越高功能越多，而需要适得其用。当需要某个功能时就选择该类芯片。本文为电子钟开发，功能简单，因此选用意法半导体公司的低成本的Cortex-M0系列芯片STM32G071。

## 1.4 论文主要工作和结构安排

本文从OLED入手，第一章先介绍了平板显示技术的发展现状，详细讲解了OLED，然后介绍嵌入式的发展现状。第二章从理论上分析RTC模块、I2C协议、OLED模块的工作原理。第三章从系统整体出发，设计系统整体工作流程框图。第四章从软件编程方面介绍运行电子时钟系统所用的软件和核心程序代码。第五章从硬件方面介绍制作电子时钟系统所用到的硬件。第六章测试电子时钟系统运行结果。第七章为结论。最后是致谢和参考文献。

# 电子时钟的设计原理

## 2.1 RTC工作原理

实时时钟（RTC）既可定时，也可计数。全天时钟/日历可以在RTC的基础上实现，而且该类时钟/日历允许报警中断，中断在时钟显示程序中编写[5]。在初始化配置启动后，RTC就可自主运行。且实时时钟功耗极低。

RTC的主要功能包括：

（1）以BCD（编码方式为二进制的十进制）的编码方式，生成具有亚秒、秒、分、时（12或24格式）、周、日期、月、年的日历。

（2）自动校正月份，包括大小月30、31天，平闰年二月的28、29天。

（3）两个可编程的警报，中断在时钟显示程序中编写。

（4）RTC时钟脉冲可以用来进行实时校正和同步。

（5）参考时钟重检测，日历的精确度可以得到进一步提高。

（6）校准电路分辨率高，石英晶体的误差可以得到一定程度的补偿。

（7）利用时间戳，对日历内容进行保存。

（8）17位自动重加载唤醒定时器（wakeup timer，WUT），用于具有可编程分辨率和周期的周期性事件。

RTC通过一个开关供电，该开关从芯片的VDD电源或VBAT引脚供电[6]。如图 2‑1所示，RTC的时钟源可以是：

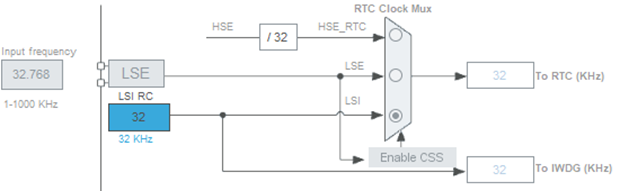
（1）32.768kHz外部晶体（LSE）；

图 2‑1 RTC的时钟源

（2）外部谐振器或振荡器（LSE）；

（3）内部低功耗RC振荡器（LSI，典型频率为32kHz）；

（4）高速外部时钟（HSE），除以RCC中的预分频器。

当由LSE驱动时，RTC可在VBAT模式和所有低功耗模式下工作。当由LSI驱动时，RTC在VBAT模式下不起作用，但在除断电模式外的所有低功耗模式下都起作用。

## 2.1.1 可编程的报警

RTC单元提供两个可编程报警：报警A和报警B。

若报警寄存器中的值与日历的亚秒、秒、分、时、天或日期能匹配，则将ALRAF设置为“1”。日历字段都可通过RTC\_ALRMAR寄存器中的MSKx字段和RTC\_ALRMASSR寄存器中的MASKSSx字段独立选择。

### 2.1.2 RTC初始化和配置

1）RTC寄存器写保护

写保护机制不受系统复位干扰。解锁写保护时，进行如下操作：

（1）将0xCA写入RTC\_WPR寄存器；

（2）将0x53写入RTC\_WPR寄存器。

若密钥错误，写保护会被重激活。

2）日历初始化和配置

（1）RTC\_ICSR寄存器初始化模式的INIT位更改为1，日历计数器不再计数，可手动更新；

（2）在两个RTCCLK时钟周期内反复询问RTC\_ICSR寄存器中的INITF位。

当初始化序列完成后，日历开始计数。

3）编程报警

必须遵循以下步骤来编程或更新可编程的报警。

（1）清除RTC\_CR中的ALRAE，以禁止报警A；

（2）清除报警A寄存器（RTC\_ALRMASSR/RTC\_ALRMAR）；

（3）在RTC\_CR寄存器中设置ALRAE，以再次使能报警A。

### 2.1.3 读日历

1）清除RTC\_CR寄存器中的BYPSHAD控制位；

2）在RTC\_CR寄存器（旁路影子寄存器）中设置BYPSHAD控制位。

### 2.1.4 复位RTC

当LSE为RTC提供时钟时，如果复位源与备份域复位源不同，则在系统复位时RTC仍然保持运行。当发生备份域复位时，停止RTC，并且所有RTC寄存器均复位到它们的复位值。

### 2.1.5 时间戳功能

1）设置TSE；

2）设置TAMPTS；

3）设置ITSE。

内部事件戳事件是由切换到电源产生的[7]。可选篡改事件可以导致记录时间戳[8]。

### 2.1.6 RTC低功耗模式

RTC低功耗模式如表 2‑1所示。可以看出RTC在低功耗模式中依然保持运行，RTC中断会使系统MCU退出低功耗模式。

表 2‑1低功耗模式对RTC的影响

|  |  |
| --- | --- |
| 模式 | 描述 |
| 休眠 | 没有效果  RTC中断引起器件退出休眠模式 |
| 停止 | 当RTC时钟源为LSE或LSI时，RTC保持活动状态。  RTC中断导致器件退出停止模式 |
| 待机 | 当RTC时钟源为LSE或LSI时，RTC保持活动状态。  RTC中断导致器件退出待机模式 |
| 断电 | 当RTC时钟源为LSE时，RTC保持活动状态。  RTC中断导致器件退出断电模式 |

### 2.1.7 RTC中断

中断事件及其描述，如表 2‑2所示。从表中看出，当日历值匹配报警设置时，即可触发中断，利用这一点可以设置闹钟响闹。当时钟与预设定闹钟时刻匹配时，系统进入中断程序，在中断程序里显示闹钟的相关设置，如显示图片、闹铃响起等。

表 2‑2中断事件及其描述

|  |  |
| --- | --- |
| 中断事件 | 描述 |
| 报警A | 当日历值匹配报警A的值时，设置 |
| 报警B | 当日历值匹配报警B的值时，设置 |
| 唤醒定时器 | 当唤醒自动重加载定时器到达0时，设置 |
| 时间戳 | 当发生时间戳事件时，设置 |

## 2.2 串行同步传输接口（I2S）原理

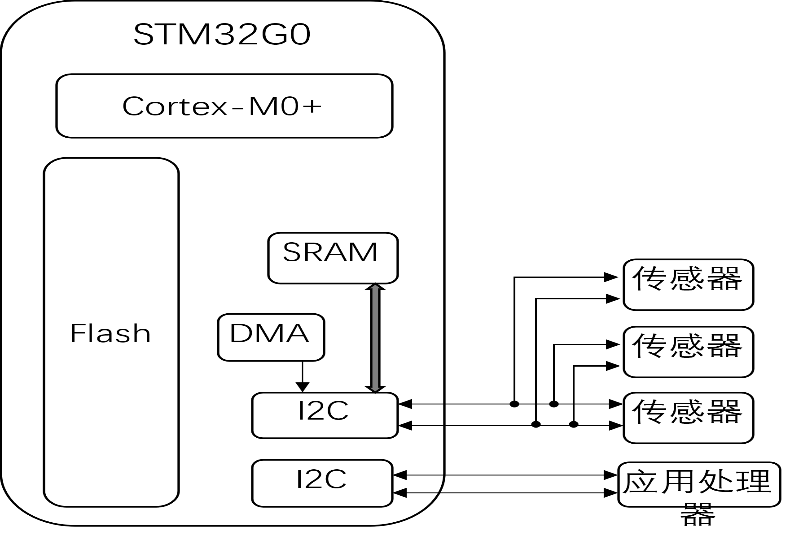
本节介绍I2C总线的功能和结构。I2C模块的典型应用如图 2‑2所示。DMA可不通过MCU直接控制数据在存储器和外设直接的传输，以此节省MCU。DMA发出控制命令，存储器的数据遵循I2C的通信协议与外设进行通信。

图 2‑2 STM32G0内I2C模块与外设的连接

STM32G071 MCU内集成的I2C模块的主要功能包括：

1）I2C总线规范rev03兼容性：

①从设备和主设备模式；

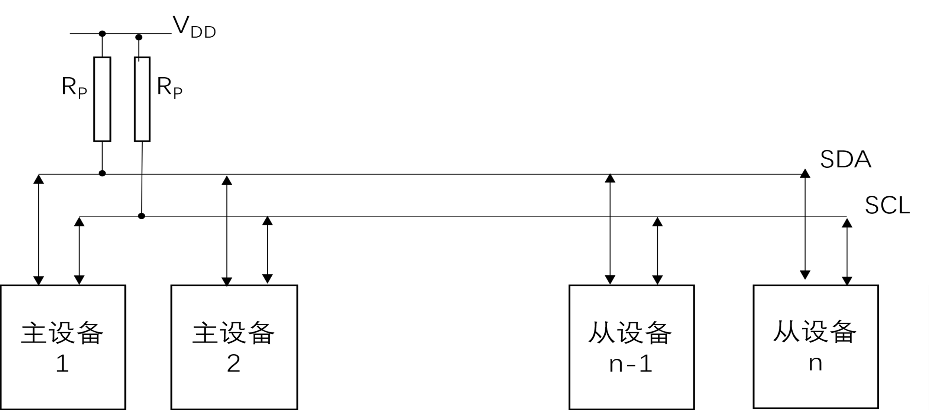
②多个主设备功能，连接关系如图 2‑3所示。

图 2‑3 I2C总线上的多个主设备和多个从设备

③在SM模式下，最高为100kHz；在FM模式下，最高为400kHz；在FM+模式下，最高为1MHz。

④7位和10位寻址模式；

⑤所有7位地址确认模式；

⑥一般呼叫；

⑦可编程的建立和保持时间；

⑧易于使用的事件管理；

⑨可选的时间延长；

⑩软件复位；

2）具有DMA功能的1字节缓冲区；

3）可编程的模拟和数字噪声滤波器

I2C模块引脚的定义，如表 2‑3所示。

表 2‑3 I2C模块引脚的定义

|  |  |  |
| --- | --- | --- |
| 引脚名字 | 信号类型 | 描述 |
| I2C\_SDA | 双向 | I2C数据 |
| I2C\_SCL | 双向 | I2C时钟 |
| I2C\_SMBA | 双向 | SMBus报警 |

I2C模块内部信号的定义，如表 2‑4所示。

表 2‑4 I2C内部输入/输出信号

|  |  |  |
| --- | --- | --- |
| 内部信号名字 | 信号类型 | 描述 |
| i2c\_ker\_ck | 输入 | I2C内核时钟，也称为I2CCLK |
| i2c\_pclk | 输入 | I2C APB时钟 |
| i2c\_it | 输出 | I2C中断 |
| i2c\_rx\_dma | 输出 | I2C接收数据DMA请求（I2C\_RX） |
| i2c\_tx\_dma | 输出 | I2C发送数据DMA请求（I2C\_TX） |

I2C1由独立的时钟源提供时钟，该时钟源允许I2C独立于PCLK频率工作。对于支持20mA输出电流驱动以实现FM+操作的I2C I/O，可通过系统配置控制器（SYSCFG）中的控制位来使能驱动能力[9]。

### 2.2.1 I2C主设备时钟

I2C主设备时钟生成和同步，如图 2‑4和图 2‑5所示。I2C主时钟的低电平和高电平持续时间由I2C时序寄存器中的软件配置。

从图中可知，在检测到SCL信号的跳变沿后，启动SCL低电平和高电平计数器。实现允许在多个主设备环境中支持主设备时钟同步机制以及从设备时钟延长功能。

因此，总的SCL周期大于计数器的和。这与由于SCL线边沿的内部检测而增加的延迟有关。这些延迟取决于SCL下降沿/上升沿、滤波器（包括模拟滤波器和数字滤波器）引起的输入延迟以及内部SCL与I2C时钟同步引起的延迟。

上升沿取决于上拉电阻和SCL线的电容。下降沿取决于数据手册中所定义的I/O端口参数。为了正确配置时钟速度，可以测量或计算这些边沿。为了在STM32CubeMX工具中正确配置I2C外设，就需要这些参数，然后在该工具中可以自动计算时序寄存器的设置。

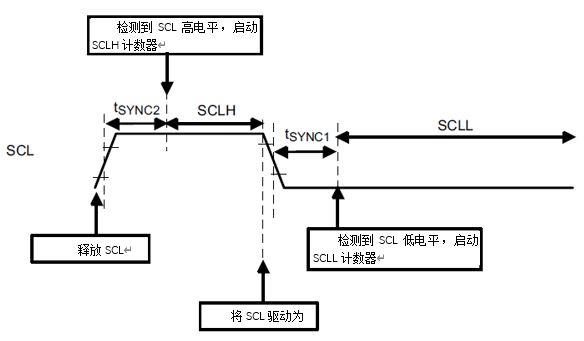
因此，总的SCL周期大于计数器的和。这与由于SCL线边沿的内部检测而增加的延迟有关。这些延迟取决于SCL下降沿/上升沿、滤波器（包括模拟滤波器和数字滤波器）引起的输入延迟以及内部SCL与I2C时钟同步引起的延迟。

图 2‑4 SCL主设备时钟生成

图 2‑5 SCL主设备时钟同步

上升沿取决于上拉电阻和SCL线的电容。下降沿取决于数据手册中所定义的I/O端口参数。为了正确配置时钟速度，可以测量或计算这些边沿。为了在STM32CubeMX工具中正确配置I2C外设，就需要这些参数，然后在该工具中可以自动计算时序寄存器的设置。

### 2.2.2 从停止模式唤醒

当地址匹配/不匹配时的唤醒情况，如图2-6和2-7所示。I2C外设支持在地址匹配时从停止模式唤醒。为此，必须将I2C外设时钟设置为HSI16振荡器。

当器件处于停止模式时，关闭内部的高速振荡器。当检测到启动条件时，I2C设备使能内部高速振荡器，该振荡器用于接收总线上的地址[10]。

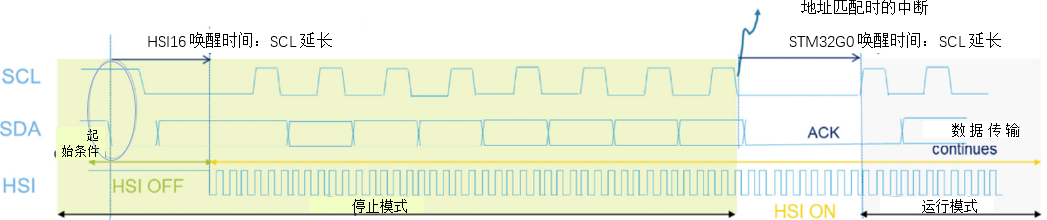
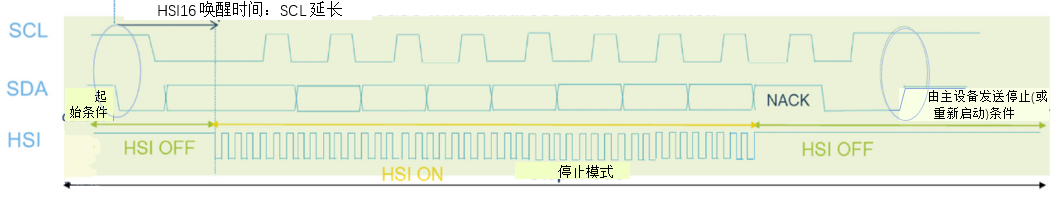
正在停止模式下接收到地址后，若能匹配从设备地址，则会产生唤醒中断，如图 2‑6所示。如果地址不匹配，则会关闭内部高速振荡器，不产生中断，器件保持停止模式，如图 2‑7所示。

图 2‑7 地址不匹配时的唤醒情况

图 2‑6 地址匹配时的唤醒情况

必须使能时钟延长，因为I2C外设在启动条件后将时钟线为低的时间延长，直到启动内部高速振荡器。在收到与编程的从设备地址匹配的地址后，I2C外设还将时钟线为低的时间延长，直到唤醒STM32G0器件[11]。

### 2.2.3 数据传输的处理

通过发送和接收数据寄存器和移位寄存器，管理数据传输。

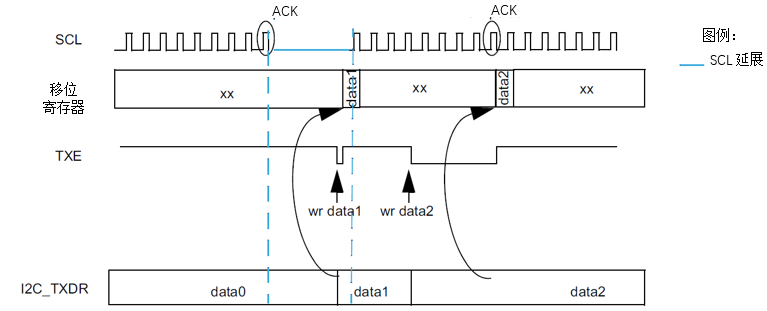
1. 接收

图 2‑8 接收数据

如图 2‑8所示，SDA输入填充寄存器。在第8个SCL脉冲之后（当接收到完整的数据字节时），如果接收寄存器I2C\_RXDR寄存器为空（RXNE=0），则拉低SCL线，直到读取I2C\_RXDR寄存器为止[12]。在第8个和第9个SCL脉冲之间插入延展（在ACK脉冲之前）

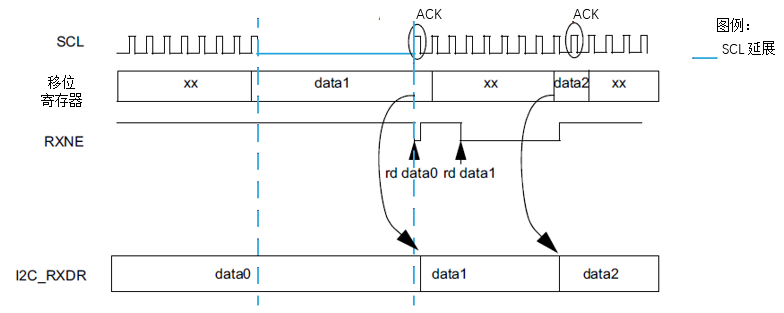
2．发送

图 2‑9 发送数据

如图 2‑9所示，如果I2C\_TXDR寄存器不为空（TXE=0），则在第9个SCL脉冲（ACK脉冲）之后复制I2C\_TXDR寄存器的内容到移位寄存器，在SDA线上移出移位寄存器的内容。如果TXE=1，意味着I2C\_TXDR中还没有数据写入，延长SCL线拉低的时间直到有数据写入I2C\_TXDR。在第9个脉冲之后，完成延展。

### 2.2.4 从设备模式

默认从设备使用它的时钟延展功能，这意味着它在需要时延展低电平时的SCL信号，以便执行软件操作。如果主设备不支持时钟延展，则在I2C\_CR1寄存器中将I2C配置为NOSTRETCH=1。

在收到ADDR中断后，如果使能了多个地址，则使用者必须在I2C\_ISR寄存器中读取ADDCODE[6:0]位，以检查匹配的地址。此外，还必须检查DIR标志以便知道传输方向。

1）从设备时间延长（NOSTRETCH=0）

当设置ADDR标志：接收到的地址与使能的从设备地址之一匹配。当通过软件设置ADDRCF位清除ADDR标志时，释放该延长。

2）从设备没有时间延长（NOSTRETCH=1）

当I2C\_CR1寄存器钟的NOSTRETCH=1时，I2C从设备不会延长SCL信号。

（1）当设置ADDR标志时，不会延长SCL时钟。

（2）在发送时，数据必须在与其传输对应的第一个SCL脉冲发生之前写入I2C\_TXDR寄存器。

（3）在接收时，必须在下一个数据字节的第9个SCL脉冲（ACK脉冲）出现之前从I2C\_RXDR寄存器中读取数据。

3）从设备字节控制模式

为了在从设备接收模式下允许字节ACK控制，必须通过设置I2C\_CR1寄存器中的SBC位来使能从设备字节控制模式。

（1）地址解析协议

为了提供一种机制来隔离每个设备以进行地址分配，每个设备继续实现唯一的设备标识符（unique device identifier，UDID）。这个128位数字由软件实现。

该外设支持地址解析协议（address resolution protocol，ARP）。通过设置I2C\_CR1寄存器中的SMBDEN位来使能SMBus设备默认地址（0b1100001）。ARP命令应使用软件实现[13]。

在从设备模式下也执行仲裁以支持ARP[14]。

（2）数据包错误检查

SMBus规范中引入了错误包检查机制，以提高可靠性和通信的稳健性。数据包错误检查时通过在每个消息传输的末尾附加一个包错误码（packet error checking，PEC）来实现。

外设嵌入了硬件PEC计算器，并允许在接收到的字节与硬件计算的PEC不匹配时自动发送非确认。

### 2.2.5 中断和DMA

I2C产生的中断可由以下事件触发：

（1）当接收缓冲区包含接收到的数据并准备好读取时，设置接收缓冲区非空标志。当发送缓冲区为空并准备写入时，设置停止检测标志。

（2）当设置RELOAD位并且已经传输了NBYTES中规定的字节数据时，设置传输完成重加载标志。

（3）当清除RELOAD和AUTOEND位并且已经传输了NBYTES中规定的字节数据时，设置传输完成标志。

（4）当接收到的从设备地址与使能的从设备地址之一匹配时，设置地址匹配标志。

（5）当在发送一个字节之后收到不确认（NACK）时，设置NACK接收标志。

当设置接收缓冲区非空或发送缓冲区为空标志时，产生DMA请求[15]。

### 2.2.6 低功耗模式

如表 2‑5所示，在低功耗模式下I2C的状态。在芯片休眠状态下，对I2C无影响，在停止状态保留I2C寄存器的内容，在待机状态退出后必须重新初始化。

表 2‑5 低功耗模式对I2C的影响

|  |  |
| --- | --- |
| 模式 | 功能 |
| 休眠 | 没有影响。I2C中断会导致器件退出休眠模式 |
| 停止 | 保留I2C寄存器的内容。如果WUPEN=1并且I2C由内部振荡器（HSI16）提供时钟：地址识别有效。I2C地址匹配条件导致器件退出停止模式。如果WUPEN=0：在进入停止模式之前必须禁止I2C |
| 待机 | I2C外设断电，退出待机模式后必须重新初始化 |

## 2.3 OLED显示模块原理

本文使用0.96寸有机发光半导体（OLED）。OLED丢弃了传统的背光源，转而使用自主发光的有机发光材料，不需要背光，比LCD的显示亮度更高。OLED响应速度快，柔韧性好可弯曲显示，显示的色域较宽。与LCD相比，OLED结构更简单，制作工序更少，质量更轻薄，量产成本低。[19]

OLED显示模块面积为0.96寸，且带有I2C/SPI接口。该模块的外观如图 2‑10所示。显示模块的主控是SSD1306芯片。在该模块中，提供了7针的接口，如表 2‑6所示。

图 2‑10 0.96寸OLED显示模块

表 2‑6 接口模式的设置（仅列出当前模块支持的模式）

|  |  |  |  |
| --- | --- | --- | --- |
| BS2 | BS1 | BS0 | 接口模式 |
| 0 | 1 | 0 | I2C接口 |
| 0 | 0 | 0 | 4线串行接口 |
| 0 | 0 | 1 | 3线串行接口 |

根据表 2‑6给出的模式设置，当：

（1）使用4线SPI接口时，焊接电阻R3和R4；

（2）使用3线SPI接口时，焊接电阻R2和R3;

（3）使用I2C接口时，焊接电阻R1、R4、R6、R7、R8。

（4）7针接口上的引脚VCC\_IN连接到开发板上的3.3V电源。

当设置为I2C接口时，连接器CON7上的引脚定义，如表 2‑7所示。

表 2‑7 连接器上CON7的引脚定义（I2C接口模式）

|  |  |  |
| --- | --- | --- |
| 引脚号 | 引脚名字 | 引脚功能 |
| 1 | GND | 地引脚，通过开发板上的连接器与地连接在一起 |
| 2 | VCC\_IN | 供电电源引脚，通过开发板上的连接器与3.3V电源连接在一起 |
| 3 | D0 | 对应于I2C接口的SCL信号 |
| 4 | D1 | 对应于I2C接口的SDA信号 |
| 5 | RES | 该引脚为复位信号输入。 |
| 6 | D/C | 在I2C模式下，该引脚作为从设备地址选择的SA0 |
| 7 | CS | 片选输入引脚。CS拉低使能芯片的MCU通信 |

1）从设备地址位（SA0）

SSD1306在通过I2C总线传输或接收任何信息之前必须识别从设备地址。

2）I2C总线数据信号（SDA）

SDA连接到上拉电阻。SDA充当发送器和接收器之间的通信信道。数据和确认信息通过SDA发送。在OLED模块中，将“SDAIN”和“SDAOU”绑定在一起作为SDA。“SDAIN”引脚必须连接以用作SDA。“SDAOUT”引脚可能断开。当“SDAOUT”引脚断开时，将忽略I2C总线中的确认信号。

3）I2C总线时钟信号（SCL）

SCL连接到上拉电阻。I2C总线中的信息传输遵循时钟信号SCL。每个数据位的传输发生在SCL的单个时钟周期内。

4）RES复位信号，使设备初始化。

## 2.4 本章小结

本章主要讲述电子时钟显示技术的理论基础。首先是RTC的工作原理。RTC有许多功能，有包括时钟和日期在内的可调日历，大小月和平闰年二月天数还可自动校正；有两个可报警中断提醒。RTC功耗低。接着介绍RTC传输数据到OLED选用的通信协议I2C，最后介绍OLED的原理。OLED有7个引脚，其中SCL时钟总线，SDA数据总线、RES复位信号引脚。

# 系统总体方案设计

## 3.1 系统总体结构

如图 3‑1所示，RTC有计时和闹钟中断功能，利用计时功能，即STM32内部时钟逻辑，将时钟信息读取到变量中，然后进行数字格式到字符串格式的转换，便可用I2C协议将字符串输出到OLED屏上。RTC计时到预设时刻，触发RTC中断，在中断程序中可设置OLED显示图片、响铃。在中断程序中，实时操作系统STM32 MCU立即停止所有任务，但RTC实时时钟依然保持运行状态[16]。

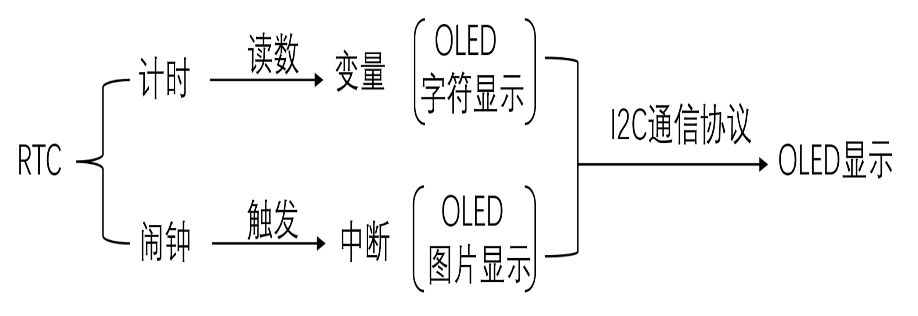


图 3‑1 系统总体结构图

## 3.2 I2C模块

### 3.2.1 通用I2C协议

通信协议就是设备间收发信息需要遵循的规则，如同摩尔斯密码规则，当一个设备向另一个设备发送信息，通信协议以设备能够识别的方式将信息转化，使得不同设备之间能够互相通信。设备间通信离不开通信协议，无论是嵌入式系统，还是PC机。比较主流的嵌入式通信协议有USART串口、SPI、I2C等。I2C每次传输的数据包有10bit和7bit之分，本文使用7bit模式，不涉及10bit模式。

设备通过通信协议传输，但I2C和SPI通信协议不是像导线那样的硬件设备。I2C是半双工工作模式[17]。

如图 3‑2所示，通用的I2C协议是两个主板之间的通信。主板可选主从机模式。当主机发送开始信号后，紧接着发送7bit地址码和1bit读写控制命令，从机的应答信号为1bit，当主机收到从机传输过来的1bit应答信号，表示一个传输周期完成。随后开始下一个传输周期。主机连续发送8位数据，理论上不限个数。除了第一个8bit数据是7bit地址码和1bit读写控制命令，后面传输的都是广义数据。每发送一个8bit数据，主机都留一个时钟周期给从机应答，0表示收到，1表示未收到或者无应答。当主机收到的应答信号不是“收到（0）”时，主机重复发送当前数据，直到从机收到。收到从机应答的主机接着传输下一个8bit数据。这就是I2C的阻塞传输模式[18]。

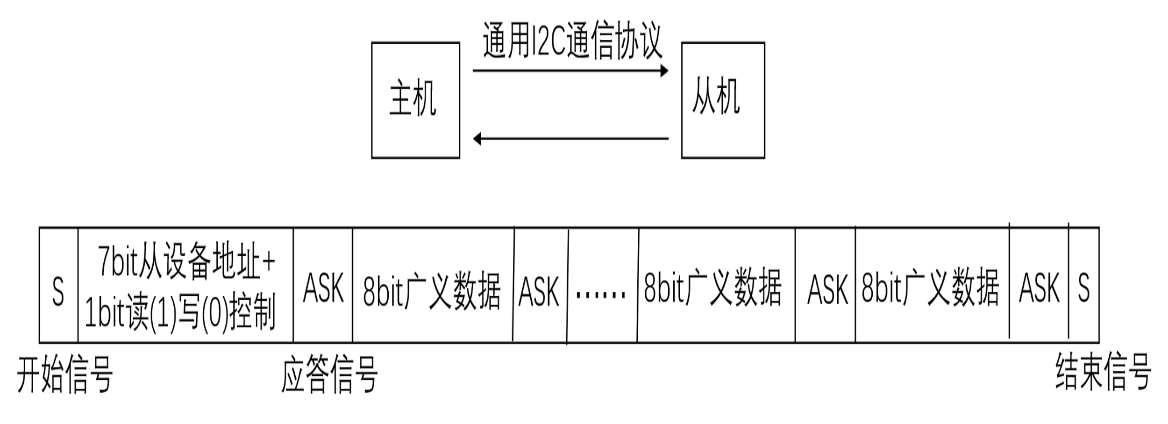


图 3‑2 主从机间的通用I2C通信协议

### 3.2.2 I2C驱动OLED

OLED接收到的广义数据包括命令和数据，命令如显示开启、显示关闭、设置光标的起始位置坐标等，数据包括图像（图片、字符串等）数据和每个像素点的亮灭信息。OLED自带的I2C串口对通用I2C协议做了进一步格式要求[19]。I2C总线接口允许将数据和命令写入设备。I2C总线的写模式按按时间顺序，在该I2C协议中，只能以如图 3‑3的格式传送数据。此种模式将广义数据分为命令和数据，开始信号和7bit从设备地址和1bit读(1)写(0)控制命令与通用I2C模式相同，收到应答0后主机发送8bit区分命令和数据的标识位，当该位为命令标识位时，主机传输8bit命令，从机根据命令执行操作；当该位为数据标识位时，主机传输8bit数据，从机将数据传输到存储器或外设中。

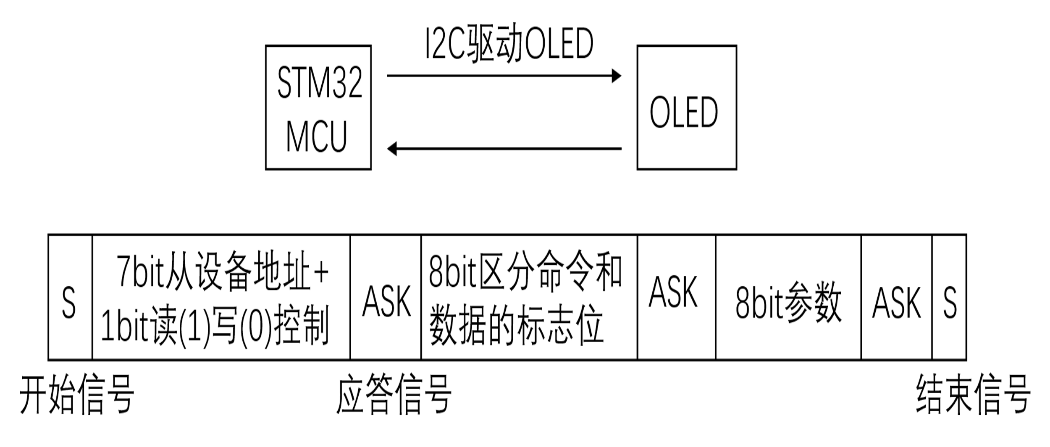
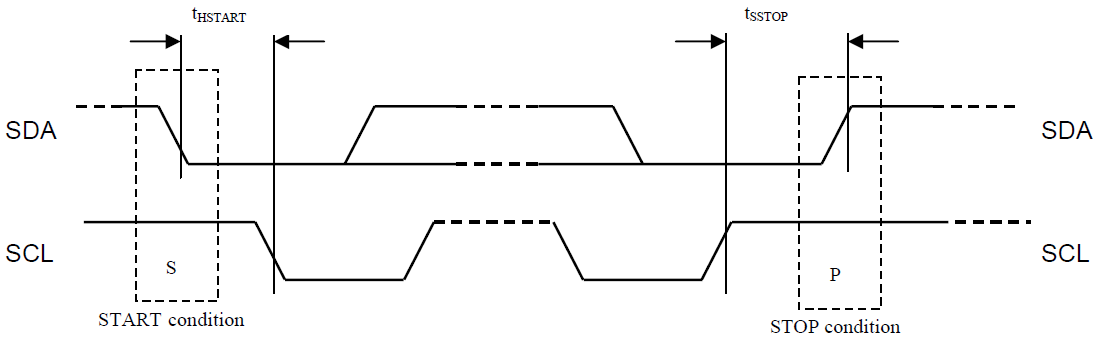


图 3‑3 I2C驱动OLED

3.2.3 I2C的写模式

1）主设备通过一个启动条件初始化数据通信。启动条件的定义如图 3‑4所示。通过将SDA从高电平拉到低电平而SCL保持高电平来建立启动条件。



START条件

STOP条件

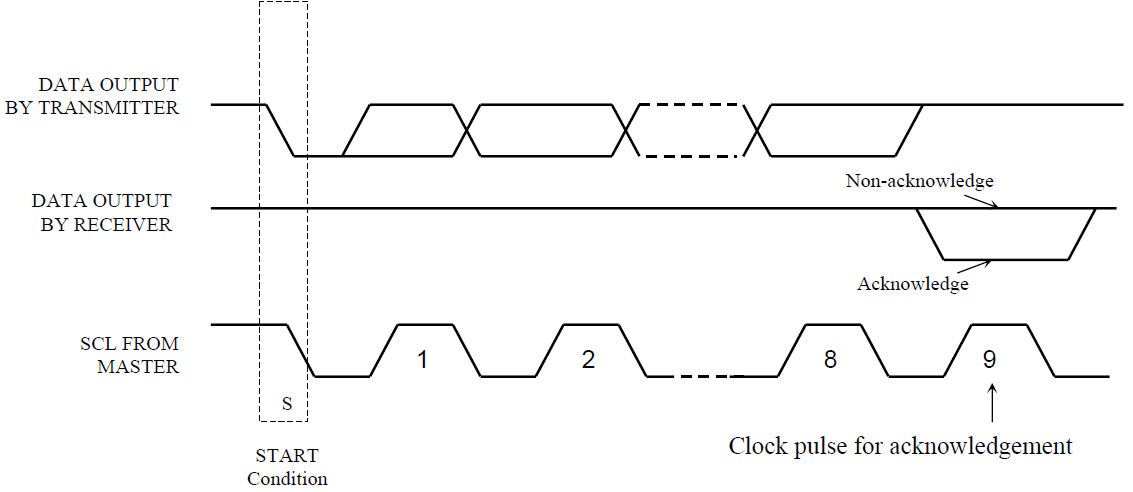
2）从设备地址跟随起始条件用于识别。对于SSD1306来说，通过将SA0更改为低/高位（D/C引脚充当SA0），从设备地址为“b0111100”或“b0111101”。

图 3‑4 起始条件和停止条件的定义

3）通过将R/W#位设置为逻辑“0”来建立写模式。

4）接收到一个字节的数据后会产生一个确认信号，包括从设备地址和R/W#位。确认信号如图 3‑5所示。确认位（ACK）定义为在确认相关时钟脉冲的高电平期间将SDA线拉低。

图 3‑5 确让条件的定义



发送器的数据输出

接送器的数据输出

来自主设备的SCL

起始条件

没确认

确认

用于确认的时钟脉冲

5）在传输从设备地址后，可以通过SDA线发送控制字节或数据字节。一个控制字节主要由Co和D/C#位构成，后面跟着六个“0”。

（1）如果将Co位设置为逻辑“0”，则随后的信息传输将仅包含数据字节。

（2）D/C#位决定下一个数据字节是作为命令还是数据。如果将D/C#位设置为逻辑“0”，则将随后的数据字节定义为命令；如果将D/C#位设置为逻辑“1”,则将随后的数据字节定义为将保存在GDDRAM中的数据。

当每次写入数据后，GDDRAM列地址指针会自动加1[20]。

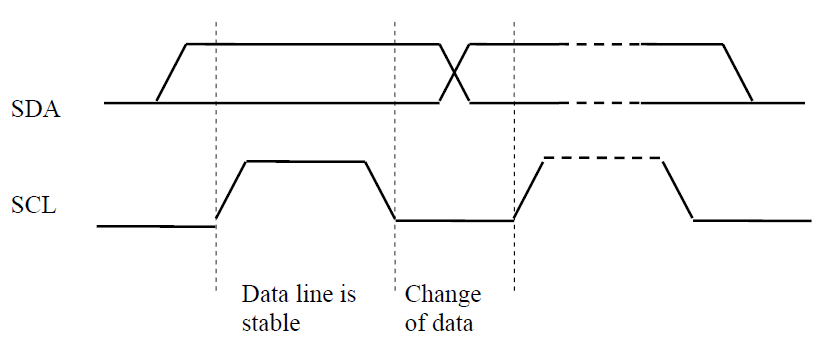
6）当接收到每个控制字节或数据字节后会产生确认位。

7）当应用停止条件时，将结束写入模式。在图15.16中定义了停止条件。在SCL保持高电平时，将SDA信号从低电平拉到高电平将建立停止条件。

### 3.2.4 I2C的数据位传输

需要注意，数据位的传输有一些限制。

1）在每个SCL脉冲期间传输的数据位必须在时钟脉冲的“高”周期内保持稳定状态，如图 3‑6所示。除了在启动或停止条件下，只有在SCL为低电平时才能切换数据线[21]。



数据线是稳定的

数据的

变化

图 3‑6 数据传输条件的定义

2）数据线（SDA）和时钟线（SCL）都要通过外部电阻上拉。

当STM32给OLED发送开始信号后，紧接着发送7bit地址码和1bit读写控制命令，这点和通用I2C协议相同。不同的是，当OLED回应后，STM32发送8bit数据，该数据为区分命令和数据的标志位。0x00表示命令模式，0x40表示数据模式。得到OLED应答反馈信号后，STM32向OLED发送8bit参数数据。在命令模式下，该参数说明具体命令；在数据模式下，参数代表像素点的亮灭信息。当光标的起始坐标确定时，每完成一次数据和命令的传送，起始光标的周围便有若干个像素点亮灭。此时OLED的一次I2C传输中的三类信息传输完毕，OLED应答后，STM32发送停止信号。在OLED的I2C传输模式中，阻塞原理同样适用。当OLED未应答或未收到，STM32持续发送当前信息，直到OLED应答为收到为止。

### 3.2.5 GPIO模拟与I2C内置模块对比

GPIO是通用输入输出串口，可用GPIO模拟实现I2C时序协议。如图所示，当SCL拉高，开始传输8bit数据，并留出一个时钟周期以便OLED应答，但STM32的GPIO端口接收不到应答信号。但这并不影响STM32通过GPIO继续向OLED发送下一个8bit数据。所以能够看出GPIO不同于I2C协议的关键一点在于GPIO是非阻塞传输模式。即无论OLED有无应答，GPIO只管连续发送后续的8bit数据。而OLED自带的I2C协议为阻塞传输模式，只有当OLED有应答且应答为收到时，STM32 MCU才向OLED发送下一个8bit数据，否则STM32会持续向OLED发送当前数据或命令，直到OLED回应为收到为止。

I2C内置模块每次传输的像素亮灭信息储存在一个大矩阵缓存区里。当I2C传输光标起始位置的控制命令后，系统从缓存区中调出图像的像素亮灭信息，经过一页一页的刷新，整幅图像便可以在OLED屏上显示出来。

另一个容易影响系统正常运行的点在于R8=0Ω，如图 3‑7中所示，即将OLED芯片上的R8位置两端焊接上导线。这是I2C协议中STM32能收到OLED应答信号的关键。

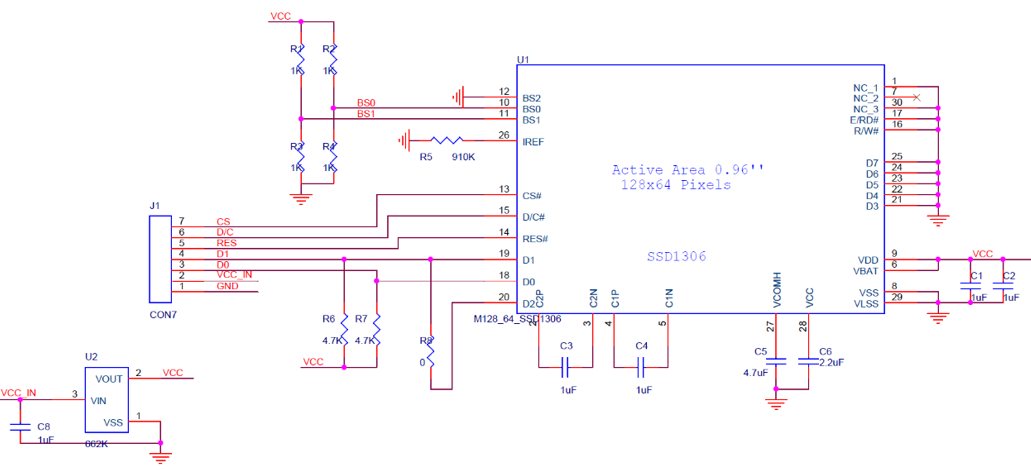


图 3‑7 0.96寸OLED显示模块原理

## 3.3 OLED模块

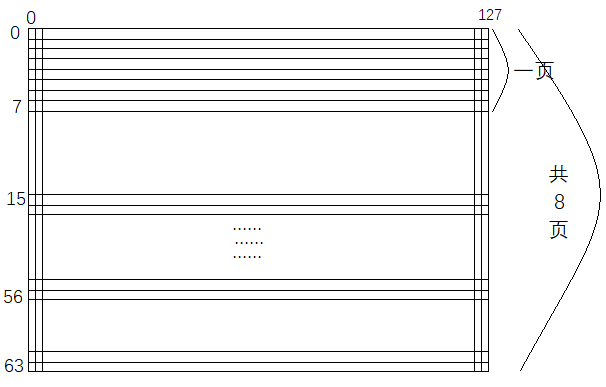
本文所使用0.96寸OLED自带图形显示数据RAM（graphic display data RAM，GDDRAM），保存着要显示的位图案。RAM的容量为128×64位，将RAM分为8页，从PAGE0到PAGE7，用于单色128×64位点阵显示[22]。SSD1306有三种不同的存储器寻址模式，页面寻址模式、水平寻址模式和垂直寻址模式[23]。本文选择水平寻址模式。128\*64个像素，竖排每8行为一组，共有8个组。一组横向竖向的像素面称为一页。

图 3‑8 OLED显示屏的像素坐标示意图

如图 3‑8所示，OLED共有8页。上一页刷新完紧接着刷新下一页。OLED的刷新速率快，不掉帧。每次刷新图片时，STM32向OLED发送光标的起始位置，一个光标刷新8个像素点（8bit数据）。当列地址未到最后一列时，光标的起始位置行地址不变，列地址自动增1，直到最后一列刷新完毕。当一个8行像素矩阵完成刷新后，需重新设置光标的起始坐标，从下一行的第一列刷新到最后一列。以此类推。如从（0，0）刷新到（0，127），设置光标命令后，从（8，0）刷新到（8，127）。

## 3.4 RTC模块

RTC是实时时钟，是STM32内部时钟逻辑。在完成初始化配置启动时钟后，RTC会自主运行，只需极低的功耗就可运行很长时间。当需要将RTC内部的亚秒、秒、分、时（12或24格式）、星期、日期、月份、年份输出显示时，先将它们读到变量里，如（h,m,s），然后将数字格式转换为字符串形式。OLED支持字符串显示。若要用数字显示，则需要先在数字库里将数字以像素码的形式转换，比直接转为字符串输出显示复杂。然后用I2C协议将字符串输出到OLED显示屏上。无论是显示日历还是显示时钟，STM32内部的时钟逻辑都可以通过将数字转换成字符串，然后通过I2C协议输出到OLED显示屏中。

OLED自带的I2C串口每次传送时都以开始信号、7bit地址1bit读写控制、1bit应答信号、8bit数据/命令标志位、1bit应答信号、8bit 参数信号、1bit应答信号、停止位的格式。且I2C协议的阻塞模式依然适用。当1bit应答信号不为0时，STM32 MCU持续通过I2C发送当前数据，直到应答信号为0，接下来便发送下一个8bit数据。

RTC允许闹钟中断。当RTC计时到预定时刻，就会触发RTC中断。在中断程序中，可设置响铃和起床图片、表情包显示。通过按键结束中断，或等待系统跳出中断，重新加载时钟和日历显示界面。

## 3.5 本章小结

本章先从总体结构介绍了实时时钟通过I2C协议在OLED上显示和计时产生中断使系统进入闹钟模式的流程。然后介绍了通用I2C协议和OLED自带的I2C驱动接口之间的区别。通用I2C协议适用于两个主板间的主从机通信，理论上可以一次传输尽可能多的广义数据，直至耗尽存储区。而OLED自带的I2C驱动串口对传送格式进行了更进一步的要求。以及将GPIO模拟以I2C内置模块对比。最后介绍RTC。RTC是STM32内部时钟逻辑，启动后可自主运行，既可计数也可计时，允许闹钟中断。利用RTC的计时功能可在OLED上显示日历和时钟，利用RTC的可报警中断设置闹钟。

# 软件部分

嵌入式包括两个部分：硬件部分和软件部分。在选取主控芯片和OLED显示面板方面有多种选择。通过参考数据手册，查看芯片有无时钟资源。本文使用STM32G071 MCU内集成的实时时钟模块以及带有I2C接口的OLED实现实时时钟的显示和修改，和闹钟功能。

## 4.1 初始化

使用编程软件向与OLED显示屏连接的STM32G071芯片传送控制命令和信息时，需要先对系统进行初始化。可以用图形编程工具STM32CubeMX初始化，也可由传统完整的开发环境MDK-ARM软件对系统的参数初始化。

### 4.1.1 STM32CubeMX初始化

STM32CubeMX是ST重点打造新推出的图形化配置工具，在图形向导下选择使用的STM32芯片能快速实现代码工程初始化。将STM32 G071的PA5引脚设置为GPIO\_OUTPUT模式用以作为闹钟提示灯。配置RCC和系统时钟后，开启RTC日历、时钟、闹钟A、闹钟B,如图 4‑1所示：

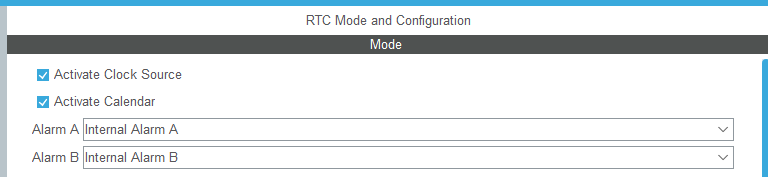


图 4‑1 开启RTC的时钟、日历和闹钟

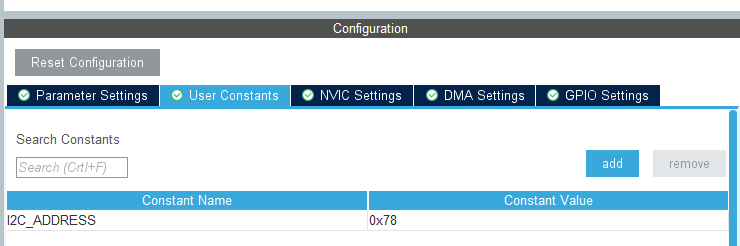
配置完RTC，选择I2C模式，并添加从设备为OLED显示屏地址标号，如所示。添加管脚映射：将I2C\_SDA映射在PB8引脚，将I2C\_SCL映射在PB6引脚。

图 4‑2 添加OLED从设备

### 4.1.2 MDK-ARM参数初始化

MDK-ARM的开发环境较完整。本文使用Keil uVision5进行编程。首先是对各种参数进行初始化，对硬件抽象层HAL库函数初始化，对系统时钟、通用串口GPIO、I2C、实时时钟RTC初始化，这些也可在STM32CubeMX中完成初始化。

HAL\_Init();

SystemClock\_Config();

MX\_GPIO\_Init();

MX\_I2C1\_Init();

MX\_RTC\_Init();

## 4.2 OLED初始化

然后对OLED屏初始化，初始化芯片ssd1306，多次调用Write\_I2C\_Command()函数，发送一系列控制命令执行如下操作：将传入的参数写入发射命令缓存区的第2位，命令缓存区的数据由HAL\_I2C\_Master\_Transmit()函数的调用，被输送到OLED屏以字符或图片的形式显示。

void Initial\_M096128x64\_ssd1306()

{

Write\_I2C\_Command(0xAE); //显示关闭命令

}

void Write\_I2C\_ Command (unsigned char I2C\_ Command) //传输命令

{

aTxBuffer\_Command [1]=I2C\_ Command;

HAL\_I2C\_Master\_Transmit(&hi2c1,(uint16\_t)I2C\_ADDRESS,(uint8\_t\*)aTxBuffer\_Data, TXBUFFERSIZE, 10000);

}

## 4.3 显示壁纸

调用两层for循环，将每个像素的亮灭信息依次发送给OLED。最外层循环为8页（8×8=64像素），内层循环为0~127列（列地址自加）。每更新完一页数据，都设置一次光标。写数据执行如下：

根据两层for循环的参数，寻址图片数据存储矩阵，将参数传输到Write\_I2C\_Data()函数中。将传入的参数写入数据命令缓存区的第2位，然后调用HAL\_I2C\_Master\_Transmit()函数，将数据缓存区的数据发送到OLED显示屏上。

void Write\_I2C\_Data(unsigned char I2C\_Data) //传输数据

{

aTxBuffer\_Data[1]=I2C\_Data;

HAL\_I2C\_Master\_Transmit(&hi2c1,(uint16\_t)I2C\_ADDRESS,(uint8\_t\*)aTxBuffer\_Data, TXBUFFERSIZE, 10000);

}

## 4.4 显示日期

日期可由RTC\_CalendarShow()函数获得，由OLED\_ShowString()函数在OLED显示。

### 4.4.1 获取日期

RTC\_CalendarShow()函数可以获取RTC实时时钟和日历信息。通过写字符串函数OLED\_ShowString()在OLED上显示日期。日期变化的慢，就直接显示在OLED屏上，不进入while循环里，而时间刷新的快，进入while循环，以判断是否处于闹钟模式，以显示不同的信息。

### 4.4.2 字符串函数

写字符串函数OLED\_ShowString()用来清空某几个格或者显示某些字符。OLED\_ShowString(x, y, char \*chr)表示要显示的起始坐标值为(x,y)，\*chr指向字符串的第1个字符。用遍历的方法，依次读取每一个不为回车字符的字符的起始位置，并访问该竖排8个像素点的亮灭信息，当未到最后一列，行地址不变，列地址自增1，直到最后一列，然后行地址加8，再次循环遍历，直到遇到回车符号。

当需要在特定区域显示字符时，先判断字符串是否为空。当为空时结束识别程序，不为空时调用OLED\_ShowChar(x,y,chr[i])函数读取起始坐标(x,y)和字符信息，将对应字符在屏幕上刷新显示。

### 4.4.3 字符库

先计算ASK码偏移量，便能确定为哪个字符，然后在字符库中查询该字符的亮灭信息矩阵，将该矩阵参数写入数据缓存区中。字符库如图 4‑3所示。将传入的参数写入数据命令缓存区的第2位，然后调用HAL\_I2C\_Master\_Transmit()函数，将数据缓存区的数据发送到OLED显示屏上。图像的信息可由取模软件运算出像素亮灭矩阵。

图 4‑3 字符库

## 4.5 循环判断模式

系统进入while循环，然后根据标志位判断是否处于闹钟模式。如果是则不执行任何内容，否则显示当前时间及日期。

### 4.5.1 显示时钟

时钟信息由RTC\_CalendarShow()函数获得，由OLED\_ShowString()输出显示。然后判断是否处于闹钟模式（flag=0）。是则不执行任何内容，否则显示当前时间及日期。

### 4.5.2 修改日历和闹钟

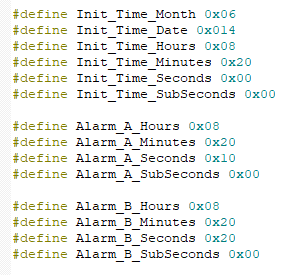
在图形化工具STM32CubeMX中初始化RTC时钟后，会自动生成一个函数，这个函数中年月日时分秒、亚秒以及闹钟时间，都是以分散的形式在函数中分布，不利于开发者识别和修改。因此可在函数外定义全局变量年月日时分秒亚秒和闹钟时间，若要修改初始值，只需修改全局变量即可。如图 4‑4所示。

图 4‑4 定义时间和闹钟全局变量

## 4.6 两个中断产生一个闹钟

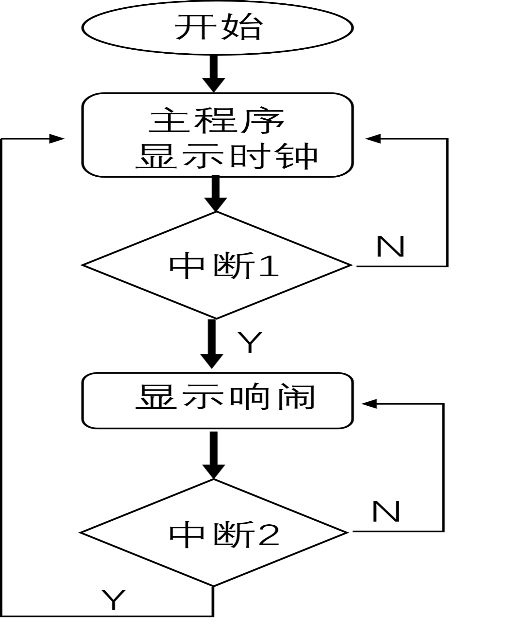
RTC时钟显示主程序，当时钟到达预设定的时间时，RTC产生中断1请求，系统进入中断服务程序，OLED显示表情包图像。中断标志位Flag\_Alarm置1。当中断2来临时，Flag\_Alarm置0，OLED显示壁纸。程序流程图如图 4‑5所示。

图 4‑5 程序流程图

### 4.6.1 闹钟模式

当时钟到达预设定的时间时，RTC产生中断1请求，系统进入中断服务程序，LED灯点亮，显示表情包，闹钟模式标志位Flag\_Alarm置1。

### 4.6.2 非闹钟模式

中断2来临时，关闭LED灯，OLED显示壁纸。字符串显示日期，闹钟模式标志位关闭，即Flag\_Alarm置0。

void HAL\_RTCEx\_AlarmBEventCallback(RTC\_HandleTypeDef \*hrtc)

{

HAL\_GPIO\_WritePin(GPIOA,GPIO\_PIN\_5,0); //\* 关闭LED灯 \*/

Picture(1); //显示壁纸

OLED\_ShowString(22,2," "); //清空原图片中间部分（显示日期与时间）

OLED\_ShowString(22,4," ");

OLED\_ShowString(22,2,aShowDate); //字符串显示日期

Flag\_Alarm=0; //闹钟模式标志位关闭

}

## 4.7 本章小结

本章从软件部分叙述了基于STM32G071的实时时钟产生时钟和闹钟并在OLED显示屏上显示的流程。先对系统的各类参数进行初始化。可以用图形化工具STM32CubeMX或者MDK-ARM软件进行初始化。然后在OLED上显示壁纸。用RTC\_CalendarShow()函数获取日期，再通过OLED\_ShowString()函数在OLED显示日期。RTC\_CalendarShow()函数还能获取时间。由于日期刷新得慢，直接在while循环之前显示。而实时时钟由于刷新得快，放在while循环里。进入while循环后，系统循环访问时钟信息，并通过闹钟模式标志位Flag\_Alarm判断系统是否处于闹钟模式。若是不执行操作，若否则显示当前时间和日期。闹钟是由两个中断产生的。当中断1发生，Flag\_Alarm置1，系统进入闹钟模式。当中断2发生，Flag\_Alarm置0，系统恢复时间日期显示。

# 硬件实现

## 5.1 硬件接线

由于I2C协议只有两根串行通信总线。故系统数据线只有两根，其余接电源和地。故系统数据线只有两根，其余接电源和地。如图 5‑1所示，STM32G071主控的引脚图中，SCL时钟总线引脚在D15,即PB8位置上，SDA数据总线在D14，即PB9位置上。

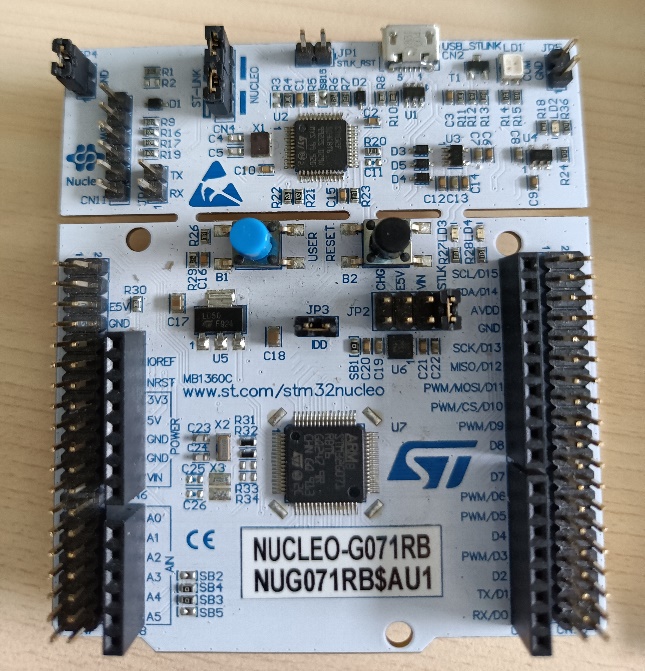
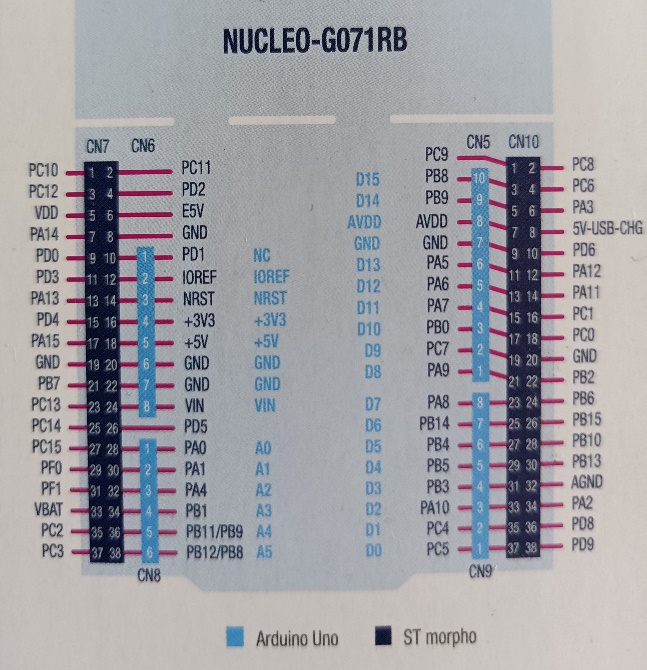


图 5‑1 STM32G071引脚图

则OLED的SCL时钟总线和SDA数据总线连接到主控STM32G071的PB8、PB9引脚，其余引脚根据OLED数据手册连接VCC、GND，对应关系如表 5‑1所示：

表 5‑1 硬件连接引脚对应关系

|  |  |  |  |
| --- | --- | --- | --- |
| OLED引脚 | 引脚功能 | 开发板引脚 | 软件/算法引脚 |
| GND | 地引脚 | GND | GND |
| VCC\_IN | 供电电源引脚，连接3.3V | 3.3V | 3.3V |
| D0 | 对应于I2C接口的SCL信号 | CN5-10 | PB8 |
| D1 | 对应于I2C接口的SDA信号 | CN5-9 | PB9 |
| RES | 该引脚为复位信号输入 | 3.3V | 3.3V |
| D/C | 在I2C模式下，从设备地址选择的SA0 | 0V | 0V |
| CS | 片选输入引脚。CS拉低使能芯片 | 0V | 0V |

## 5.2 实验结果

本节将对设计进行编译和链接，生成可以下载到STM32G071 MCU内Flash存储器的文件格式，并在OLED屏上显示实时时钟。

1）在Keil主界面主菜单中，选择Project->Build Target，对设计代码进行编译和链接，然后生成可以下载到STM32G071 MCU内Flash存储器的文件格式。

2）通过USB电缆，将计算机/笔记本电脑的USB接口连接到NUCLEO-G071RB开发板的USB接口。

3）在Keil主界面主菜单中，选择Flash->Download，将前面生成的Flash格式的文件下载到STM32G071 MCU内的Flash存储器中。按一下NUCLEO-G071RB开发板上标记为RESET的按键，使程序正常运行。

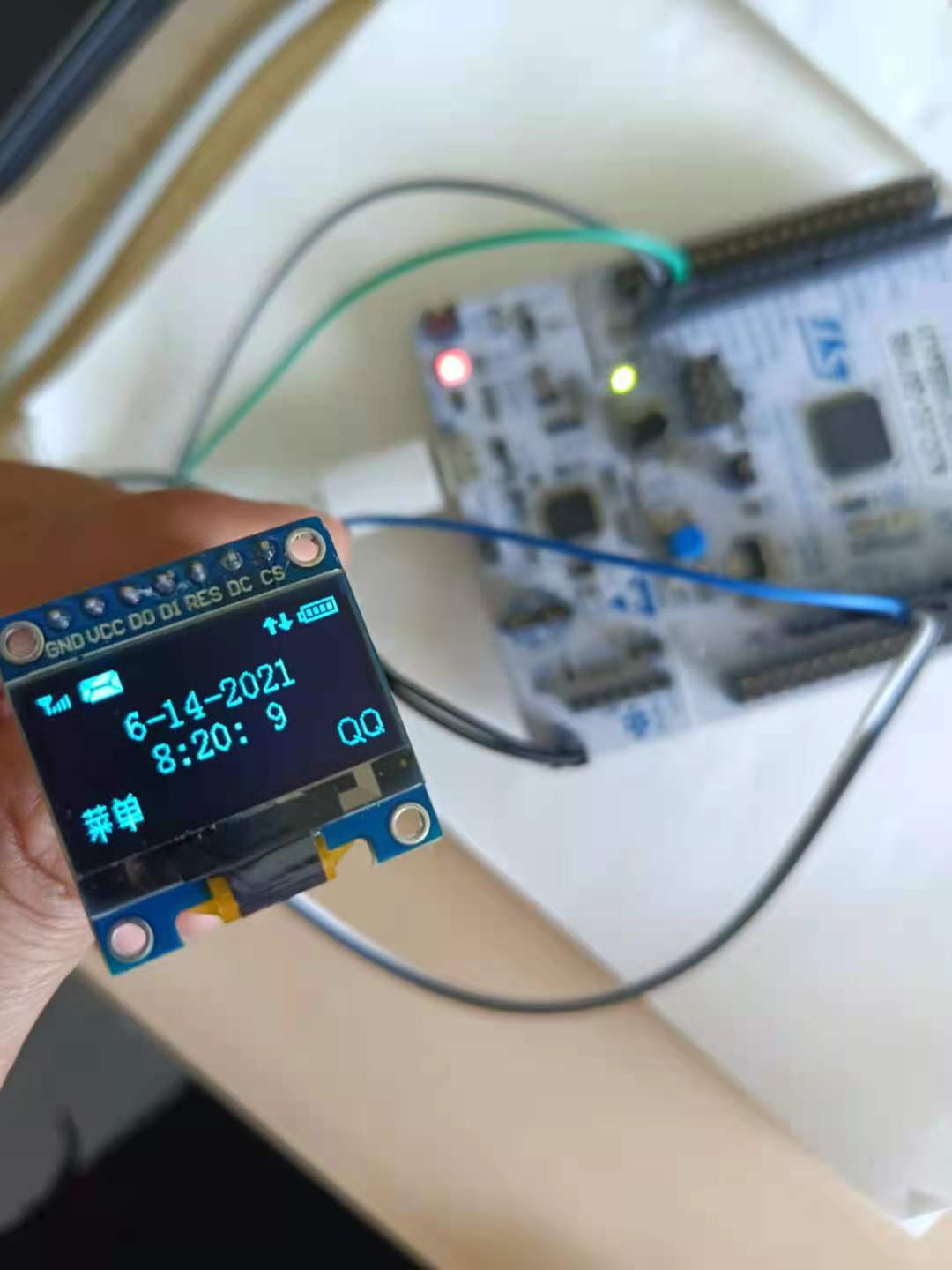
将程序烧录到连接OLED显示模块的STM32G071主板上后，由于刷新速率快，OLED显示屏立即加载出壁纸和日期时钟信息，且可看见显示屏上的时钟时间以秒变化。起始时间为设定的闹钟时间为6-14-2021，8：20：00. 实时时钟始终运行，经过9s后，OLED显示如图 5‑2所示。

图 5‑2 闹钟响之前的显示

设定闹钟时间为8：20：10，持续响闹10秒。响闹时，LED灯亮起，OLED显示板显示表情包。如图 5‑4所示。当响闹结束，系统恢复日期时钟显示，闹钟标志LED灯灭。如图 5‑3所示。

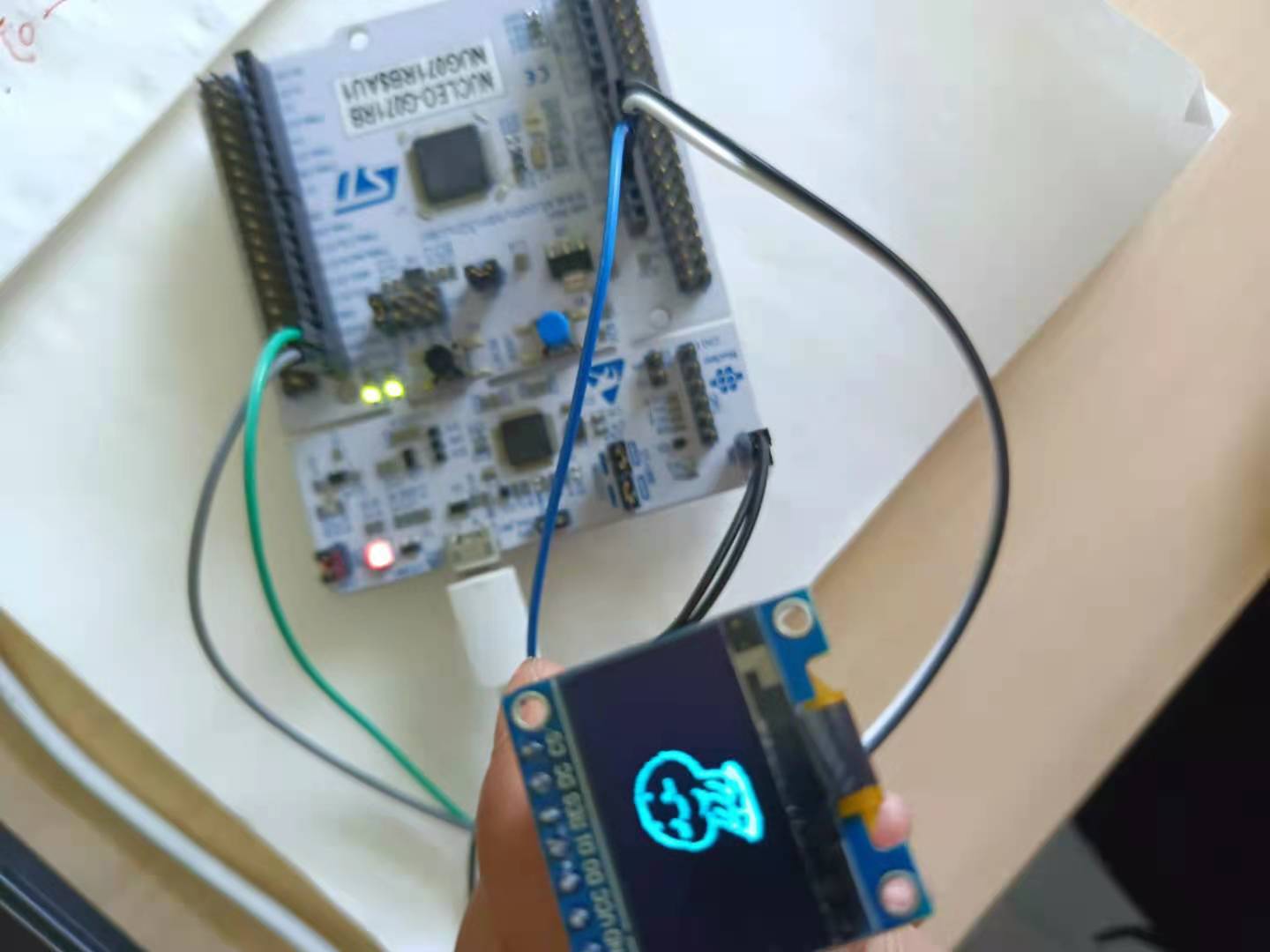
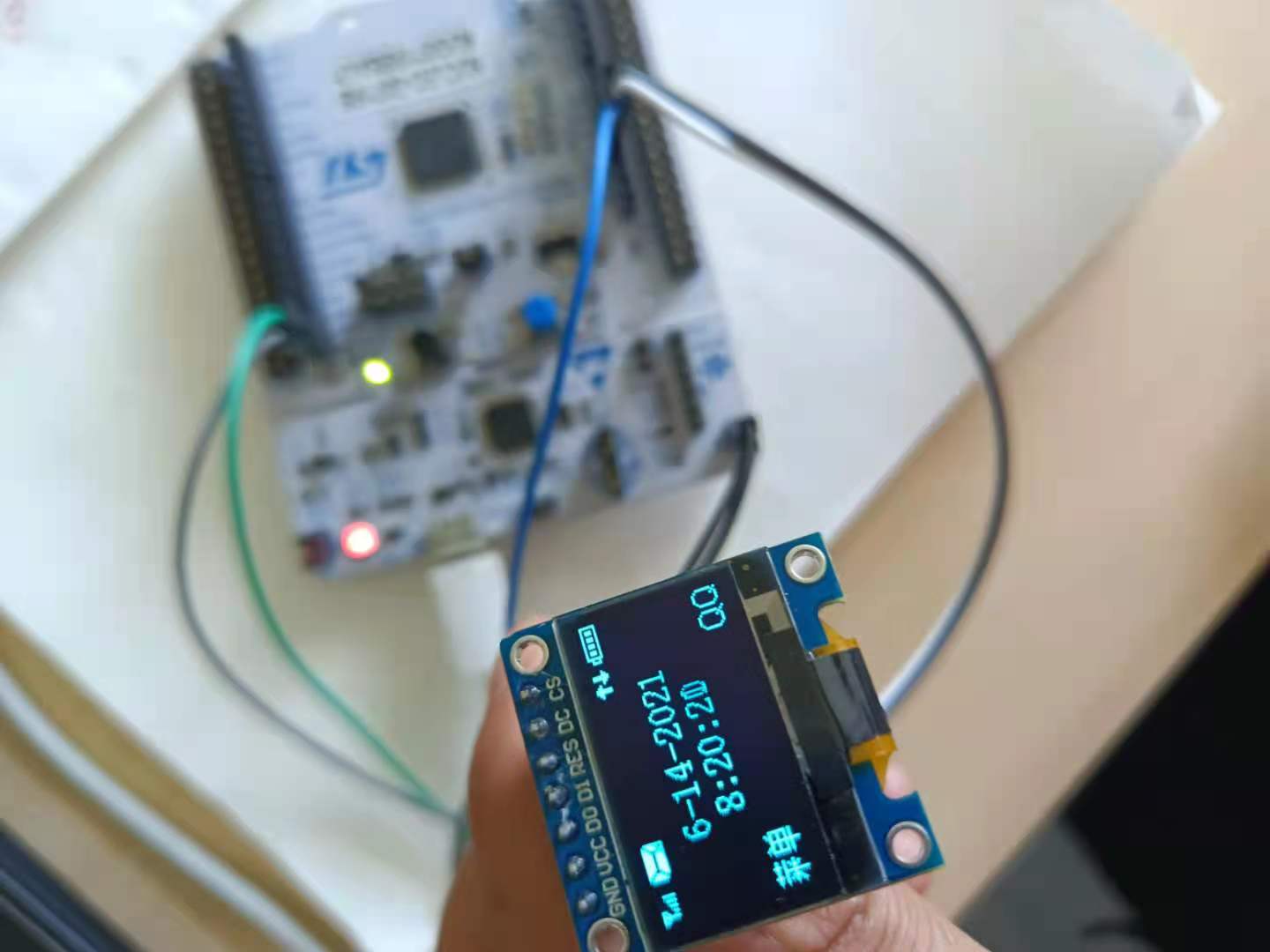


图 5‑3 非闹钟模式下显示日期时钟

图 5‑4 闹钟模式下显示表情包

结 论

通过本文STM32G071 MCU内集成的实时时钟模块以及带有I2C接口的OLED所构成的实时时钟系统，能够在LED上显示，且能够设置一个闹钟。当闹钟响起时，系统进入中断，OLED显示屏上显示表情包。当另一个中断发生，系统能从闹钟模式中退出，重新加载显示时钟和日期。STM32G071为实时操作系统，当中断发生，STM32 MCU立即暂停当前所有任务，以响应中断程序，这在军事、航空、船舶、工业和医疗等对实时性要求高的领域中十分必要，以此来确保安全和可靠性。但STM32 MCU暂停所有服务时，内部实时时钟RTC依然保持运行状态。

本文未实现的功能有闹钟响铃和按键控制切换模式功能。由于扬声器模块较为复杂，若能够加入语音识别和语音播报模块到系统中，实现语音定闹钟和语音报时，整个嵌入式低功耗时钟系统会更加智能。本系统是由两个中断来实现设置一个闹钟，理论上可实现设置多个闹钟，后期进一步探究。实时时钟和嵌入式系统应用十分广泛，后期可进一步研究，以求理论知识提升的同时提升编程并软硬件结合的能力。

参考文献

[1]白木,子荫.新一代OLED显示技术[J].现代显示,2002,33(3):14-17

[2]李继军,聂晓梦.平板显示技术比较及研究进展[J].中国光学,2018,11(5):605-710

[3]邱勇,万博泉.关于我国发展OLED技术和产业的思考[J].现代显示,2002,33(3):4-7

[4]庄衍竖.TFT-LCD与OLED图像显示降低功耗方法的研究与实现[D].广州:华南理工大学,2016

[5]Tang C W,VanSlyke S A.Organic electroluminescent diodes[J].Appl Phys Lett,1987,51(12):913-915

[6]庞斯棉.嵌入式电子时钟系统的设计与实现[J].电子测试,2015,6:31-33

[7]刘玉秀,郭建强.基于超低功耗单片机的RTC时钟的设计实现[J].信息技术,2016(7):156-160

[8]Kim T H,Choi K S,Ko S J.Backlight power reduction using efficient image compensation for mobile devices[J].Consumer Electronics IEEE Transactions,2010,56(3):1972-1978

[9]胡洪坡,钱宗锋.基于STM32F107VCT6平台的I2C总线使用技术研究[J].山西电子技术,2013,5:74-76

[10]Steve Logan.可控制多种外设的SPI\_I\_2C总线[J].世界电子元器件,2006,9:54-59

[11]赵晶.单片机控制OLED显示系统研究[D].重庆:重庆大学,2006

[12]Tang C W,An overview of organic electroluminescent materials and devices[J].Society of Information Display,1997,5(1):11-14

[13]基于STM32微控制器的OLED模块的应用装置[J].云南大学学报(自然科学版),2013,35(S2):130-134

[14]Paul,Burnetal L.Chemicaltuning of electronic properties of poly(p-phenylenevinylene) based copolymers[J].JAmChemSoc,1993,115:10117-10124

[15]付新虎,郑喜凤,丁铁夫.OLED驱动技术的发展与应用[J].电视技术,2005,9:36-39.

[16]焦石,王琛.基于STM32的OLED显示屏驱动设计[J].电子世界,2018,12:127-128

[17]Pope M,Kallmann H P,Magnante P.Electroluminescence in Organic Crystals[J].Chem Phys,1963,38(8):2042-2043

[18]刘正翔.基于STM32的OLED显示屏接口设计[J].电子技术2018,6:65

[19]Satoshi Miyaguchietal.Organic LED full-color passive-matrix display[J].Journal of the Society for Information Display,1999,7(3):221-226

[20]高锋,叶成彬.基于STM32的OLED音乐频谱显示器的设计[J].电子设计工程,2020,28(11):156-160

[21]Stas G,Lihi Z M,Ayellet T.Context-aware saliency detection[J].IEEE Transactions on Pattern Analysis & Machine Intelligence,2012,34(10):1915-1926

[22]陈海峰,陈盛闯.基于单片机的OLED显示系统设计与实现[J].电子设计工程,2015,23(20):137-140

[23]高丽,张步新.用液晶显示控制驱动器实现OLED的矩阵字符显示[J].光电子激光,2002,13(2):129-132

致 谢

这篇毕业论文得以最终完成所有的工作，非常感谢信息学院老师们对我的帮助！感谢父母在对我的教育上倾力付出，感谢外公外婆对我的关爱！感谢亲人朋友和所有帮助我的人。非常感激我的导师何宾老师，非常及时严厉指出我的问题，并提出解决方案。我也深刻意识到了我的错误。其次非常感谢何老师的弟子李天凌同学，在给我指导时不厌精细。每次我有问题，无论大小，总能得到指点和解答。

大学生活正式宣告结束，同学们也都已走入了人生下一个阶段。多么遗憾由于疫情没能和同学们一起穿上学士服拍照聚餐，许多人再也很难见面。还记得，老北区的小巧与炎热，通宵自习室和校门口的夜宵。还记得，南口校区的阔大和绿意盎然，湖、柳、鸭、亭、山和食堂。现代化的教室、图书馆、宿舍，上床下铺的快乐。记得东三环的车流声，母校之光下的雪人，雪地上的名字，图书馆的芝兰满室，操场上奔跑的身影。羽翼渐满终离巢，愿母校双一流建设蒸蒸日上！

万分感谢老师们的教导，感谢与同学们共度的时光，下个路口再相见！