摘 要

具体地讲就是研究工作的主要对象和范围， 采用的手段和方法， 得出的结果和重要的结论， 有时也包括具有情报价值的其它重要的信息。 一般字数在200-400字之间

https://www.zhihu.com/question/20028088

本部分正文结束。

**关键字：**关键字1，关键字2，关键字3

ABSTRACT

英文摘要正文

英文摘要结束.

**Key Words:** 英文关键字1,英文关键字2,英文关键字3

目录

[第1章 课题整体框架 1](#_Toc29413)

[1.1 课题任务 1](#_Toc19877)

[1.2 课题要求 1](#_Toc20859)

[1.3 研究意义 1](#_Toc14954)

[第2章 硬件部分和软件部分 2](#_Toc20533)

[2.1 硬件部分 2](#_Toc21308)

[2.1.1 电源模块 2](#_Toc6751)

[2.1.2 最小系统开发板 2](#_Toc15363)

[2.1.3 寻迹红外模块 2](#_Toc31454)

[2.1.4 前轮舵机 2](#_Toc30759)

[2.1.5 TB6612FNG模块说明 5](#_Toc9006)

[2.1.6 0.96OLED显示屏 6](#_Toc2078)

[2.1.7 方式 8](#_Toc31538)

[2.1.8 规范化发 8](#_Toc123)

[2.1.9 规范化 8](#_Toc18558)

[2.2 软件部分 9](#_Toc20110)

[2.2.1 软件工具keil 9](#_Toc12552)

[2.2.2 Oled显示屏专用取模工具 9](#_Toc16072)

[2.2.3 Cubmax 10](#_Toc31852)

[2.2.4 Git 10](#_Toc15237)

[2.2.5 模块流程图 10](#_Toc20506)

[第3章 实现功能与电路设计 11](#_Toc12014)

[3.1 实现功能与设计 11](#_Toc17961)

[3.1.1 实现功能描述 11](#_Toc2311)

[3.2 电路设计 11](#_Toc31660)

[3.2.1 最小系统板电路 11](#_Toc13105)

[3.2.2 红外检测模块电路 12](#_Toc23222)

[3.2.3 TB6612带稳压模块板原理图 13](#_Toc17566)

[3.2.4 0.96OLED显示屏原理图 14](#_Toc17775)

[3.3 软件设计 15](#_Toc10127)

[3.3.1 软件设计思路 15](#_Toc13035)

[第4章 调试难点和问题 16](#_Toc7504)

[4.1 调试中遇到的重点与难点 16](#_Toc5028)

[4.2 解决方案 16](#_Toc16208)

[4.3 实现展示（附上仿真图或实物照片） 16](#_Toc17086)

[第5章 总结 17](#_Toc19918)

[参考文献 18](#_Toc12932)

[想对老师说的话 19](#_Toc24800)

[附录 20](#_Toc13762)

[附录一：06年11月CD电信新华营业厅日缴费顾客半小时到达数据 20](#_Toc30446)

[（略） 20](#_Toc5344)

[附录二：顾客调查问卷 20](#_Toc29658)

[（略） 20](#_Toc21742)

1. 课题整体框架
   1. 课题任务

用STM32f103c8t6作为微控制器，设计一款智能小车。小车能实现自动巡航，可前进后退，转弯。小车具有手动控制模式，能使用远程设备解除智能小车的自动巡航。实现手动控制小车的行进。

* 1. 课题要求

自动巡航的跑道采用黑色线纸组成的一圈半径为25cm环形跑道，小车在自动巡航的时候无需外部控制，可完成一圈巡航。手动控制器与小车分离，且采用合适的无线通信技术跟小车进行通信。小车可以测速，并通过速度反馈控制电机的驱动能力，使速度恒定。小车的转向模拟真实的车辆前轮左右转向。手动控制器需要屏幕显示。且输入方式不可以使用按键输入。

* 1. 研究意义

自动巡航和无人驾驶技术一直是热门的研究技术，前人已经提出了许多优秀的检测方法和算法去实现小车的智能驾驶。但是总体的解决方案价格都是价格高昂的。在某些运用场景上，这些价格高昂的技术就显得十分的鸡肋。所以，本课题在于研究如何利用更低廉的单片机芯片去实现些简易，普通的自动巡航技术在简单赛道上的运用。并结合手动控制来完善自动巡航驾驶的不足，解决无人驾驶潜在的风险和威胁。

1. 硬件部分和软件部分
   1. 硬件部分
      1. 电源模块

小车的电源模块，采用了可充放电的12v锂电池供电，并采用LM2596S DC-DC直流可调的降压稳压电源模块板进行降压处理成3.3v。手柄的电源模块采用了可充放电的3.7v锂电池供电，同样配置了降压模块进行降压。

* + 1. 最小系统开发板

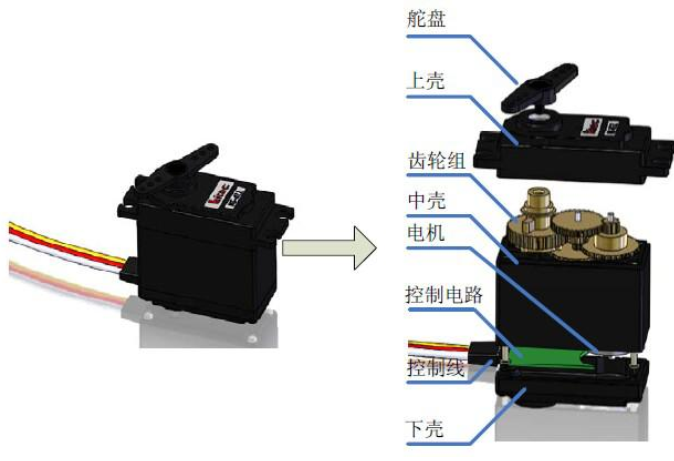
小车和手柄的核心控制处理器采用了stm32f103c8t6的最小系统控制板。基于ARM Cortex-M 内核STM32系列的三十二位的微控制器嵌入式操作系统，拥有从六万四千或十二万八千字节的闪存flash程序存储器，高达两万字节的SRAM。工作电压两伏到三点六伏，温度区间为负四十摄氏度到八十五摄氏度。拥有最高七十二MHz的工作频率，在存储器的等待周期访问时可达五分之四DMips/MHz。拥有两个I方C接口(支持SMBus和PMBus)。三个USART串口通信接口。两个个SPI通信接口(十八兆比特每秒)。

* + 1. 寻迹红外模块

在小车的前端配置了左右两个红外传感器模块。该红外传感器模块对环境光线的抗干扰能力强，拥有一对红外线发射装置与接收装置，发射装置发射出某一波段的红外线，当发射红外线方向遇到拥有反射红外线能力的反射面时，红外线被反射回来，被接收设备接收，然后通过比较器LM393电路进行信号处理之后，接受信号的LED灯会亮起。同时，输出接口OUT引脚输出一个数字低电平信号。拥有电位器旋钮调节按钮，方便调节红外检测距离，有效距离范围是二到三十厘米，工作电压区间为三点三伏到五伏。因为传感器是主动发射红外线进行红外反射探测,因此探测目标的红外反射率和形状是影响红外探测距离的重要因素。其中黑色对红外线的吸收能力最强。红外探测距离因此会减小。相反白色是反射能力最强的。小面积物体的探测距离小,大面积物体的探测距离大。输出端口OUT引脚可直接与MCU的IO口引脚连接，也可以直接驱动一个五伏的继电器；连接方式：VCC-VCC;GND-GND;OUT-IO。电路板尺寸为3.2CM\*1.4CM。

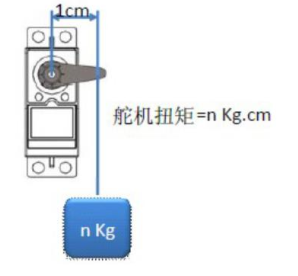
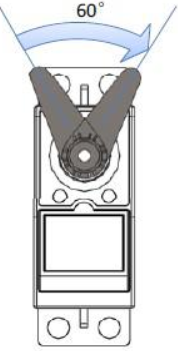
* + 1. 前轮舵机

舵机根据其控制电机的方式可以称为微型伺服马达。因为它早期在模型上主要用于控制模型的舵面，所以市面上又被称为舵机。舵机能够接收一个简易的操作命令，便能够自动旋转到操作命令对应的较为精准的位置，所以十分方便在关节型结构机器产品中运用。舵机通俗来说便是组合了直流电机、微型电机控制器和减速装置等，并组装在方便安装的外壳里的随动系统。可以使用简单的输入信号进行精确度不错的旋转到给定角度的电机系统。舵机安装了电位器进行检测输出轴的转动角度，并通过控制器与电位器的信息判断。可以较为准确的操控和维持输出轴的转动角度。直流电机这样的控制方式又叫做闭环控制，所以舵机更准确的来说，应该叫伺服马达，英文名servo。



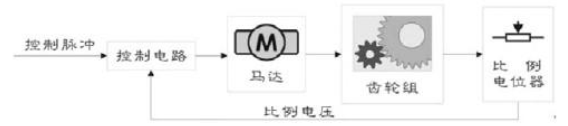
舵机的结构

舵机的主体结构如图 2-1 所示，主要有几个部分：外壳、减速齿轮组、电机、电位器、 控制电路。简单的工作原理是控制电路接收信号源的控制信号，并驱动电机转动；齿轮组将电机的速度成大倍数缩小，并将电机的输出扭矩放大响应倍数，然后输出；电位器和齿轮组的末级一起转动，测量舵机轴转动角度；电路板检测并根据电位器判断舵机转动角度，然后控制舵机转动到目标角度或保持在目标角度。舵机的外壳一般是塑料的，特殊的舵机可能会有金属铝合金外壳。金属外壳能够提供更好的散热，可以让舵机里面的电机运行在更高功率下，以提供更高的扭矩输出。金属外壳也可以提供更牢固的固定位置。舵机的齿轮箱有塑料齿轮、混合齿轮、金属齿轮的差别。塑料齿轮成本底，噪音小，但强度较低；金属齿轮强度高，但成本高，在装配精度一般的情况下会有很大的噪音。小扭矩舵机、微舵、扭矩大但功率密度小的舵机一般都用塑料齿轮，如 Futaba 3003，辉盛的 9g微舵金属齿轮一般用于功率密度较高的舵机上，比如辉盛的 MG995 舵机，在和3003一样体积的情况下却能提供13KG的扭矩。Hitec 甚至用钛合金作为齿轮材料，其高强度能保证3003大小的舵机能提供20几公斤的扭矩。混合齿轮在金属齿轮和塑料齿轮间做了折中，在电机输出减速箱扭矩不大的部位，用塑料齿轮。舵机的规格和选型：1舵机转速，转速由舵机无负载的情况下转过 60°角所需时间来衡量，常见舵机的速度一般在0.11s/60°-0.21s/60°之间。2舵机转矩，舵机扭矩的单位是KG·CM，这是一个扭矩单位。可以理解为在舵盘上距舵机轴中心水平距离1CM处，舵机能够带动的物体重量。



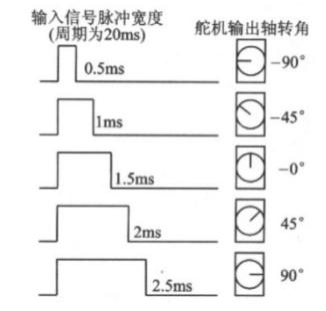
舵机的转速和转矩

舵机的工作原理：舵机是一个微型的伺服控制系统，具体的控制原理可以用下图表示。



舵机的控制原理

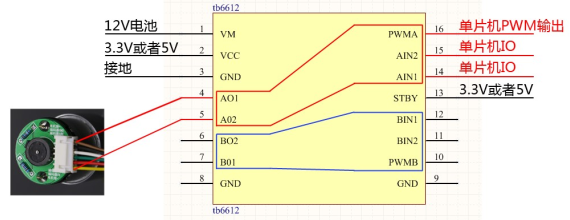
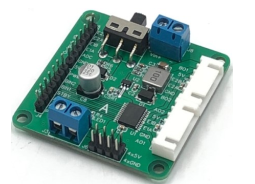
工作原理是控制电路接收信号源的控制脉冲，并驱动电机转动；齿轮组将电机的速度成大倍数缩小，并将电机的输出扭矩放大响应倍数，然后输出；电位器和齿轮组的末级一起转动，测量舵机轴转动角度；电路板检测并根据电位器判断舵机转动角度，然后控制舵机转动到目标角度或保持在目标角度。模拟舵机需要一个外部控制器（遥控器的接收机或者单片机）产生脉宽调制信号来告诉舵机转动角度，脉冲宽度是舵机控制器所需的编码信息。舵机的控制脉冲周期 20ms，脉宽从0.5ms-2.5ms，分别对应-90度到+90度的位置(对于180°舵机)。如下图所示：



舵机信号脉冲示意图

* + 1. TB6612FNG模块说明

TB6612FNG是东芝半导体公司生产的一款直流电机驱动器件，它具有大电流 MOSFET-H桥结构，双通道电路输出，可同时驱动2个电机。相比 L298N的热耗性和外围二极管续流电路，它无需外加散热片，外围电路简单，只需外接电源滤波电容就可以直接驱动电机，利于减小系统尺寸。对于 PWM信号输入频率范围，高达 100 kHz的频率更是足以满足我们大部分的需求了。对于 PWM信号输入频率范围，高达 100 kHz的频率更是足以满足我们大部分的需求了。以下是 TB6612FNG的主要参数：最大输入电压：VM = 15V。最大输出电流：Iout = 1.2A（平均）/3.2A（峰值）。正反转/短路刹车/停机功能模式，内置过热保护和低压检测电路。带稳压模块版说明：新增 5V稳压电路，支持 5V 2A的输出。共有 4个这样的引脚可以对外部供电。增加电压测量电路，通过串联一个 10k和一个 1k的电阻，对输入电源进行 1/11的分压后，可以通过 ADC进行采集并计算得到电源电压进行监控。引出电机标准的 6PIN接口，可以直接用排线连接，AB相编码器的信号接到单独的引脚输出。新增电源输出电路，可输出与输入的电源保持一致的电源。含有电源开关，可以对板子供电进行开启和关闭的操作。引脚说明:要实现的调试和换向功能，我们可以使用单片机实现的，但是单片机IO的带负载能力较弱，而直流电机是大电流感性负载，所以我们需要功率放大器件，在这里，我们选择了TB6612FNG。



TB6612FNG实物图和引脚连接示意图

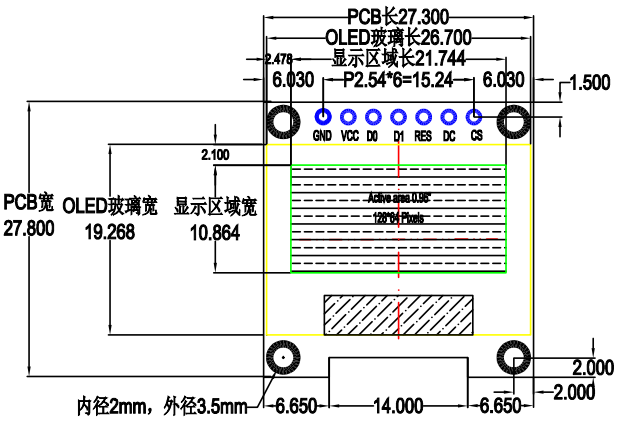
必须要有 PWM输入才有AO1和AO2的信号，只接AIN1和AIN2不会产生AO1和AO2的信号。VM直接接电池即可，VCC是内部的逻辑供电，一般给 3.3或者5V都行，模块的GND建议一个接电源地，一个接单片机地，STBY置高模块才能正常工作。完成上面的接线之后，我们就可以开始控制电机了，上图中红色部分的5个引脚控制一路电机，蓝色部分的控制另外一路电机，这里只讲其中的 A路，B路的使用是一样的。AO1和AO2分别接到电机的+和-。然后通过PWMA、AIN2、AIN1控制电机。其中 PWMA接到单片机的PWM引脚，一般10Khz的 PWM即可，并通过改变占空比来调节电机的速度。下面是真值表：

TB6612FNG输出真值表

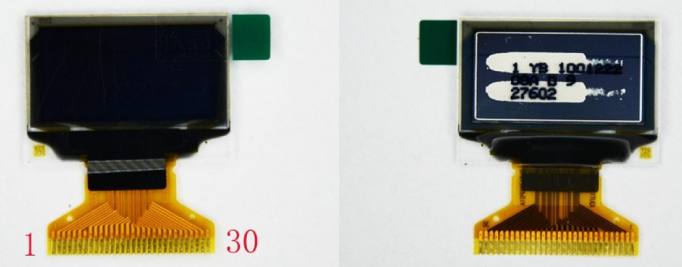
|  |  |  |
| --- | --- | --- |
| AIN1 | 0 | 0 |
| AIN2 | 0 | 1 |
|  | 停止 | 反转 |

* + 1. 0.96OLED显示屏

OLED，即有机发光二极管（ Organic Light Emitting Diode）。OLED 由于同时具备自发光，不需背光源、对比度高、厚度薄、视角广、反应速度快、可用于挠曲性面板、使用温度范围广、构造及制程较简单等优异之特性，被认为是下一代的平面显示器新兴应用技术。LCD 都需要背光，而 OLED 不需要，因为它是自发光的。这样同样的显示OLED效果要来得好一些。以目前的技术，OLED 的尺寸还难以大型化，但是分辨率确可以做到很高。在此我们使用的是中景园电子的0.96寸OLED显示屏，该屏有以下特点：0.96寸OLED有黄蓝，白，蓝三种颜色可选；其中黄蓝是屏上 1/4 部分为黄光，下 3/4 为蓝；而且是固定区域显示固定颜色，颜色和显示区域均不能修改；白光则为纯白，也就是黑底白字；蓝色则为纯蓝，也就是黑底蓝字。分辨率为 128\*64。多种接口方式；OLED 裸屏总共种接口包括：6800、8080 两种并行接口方式、3 线或 4 线的串行 SPI 接口方式、IIC 接口方式（只需要 2 根线就可以控制 OLED 了！），这五种接口是通过屏上的 BS0~BS2 来配置的。

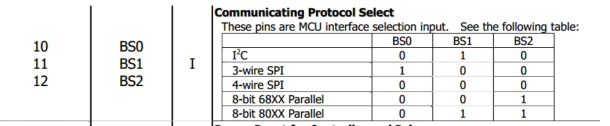


裸屏为 30pin，从屏正面看左下角为 1，右下角为 30；如下图所示：



0.96寸OLED裸屏外观

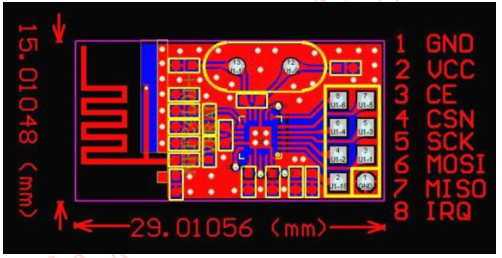
0.96寸OLED驱动IC为 SSD1306；其具有内部升压功能；所以在设计的时候不需要再专一设计升压电路；当然了本屏也可以选用外部升压，具体的请详查数据手册。SSD1306 的每页包含了128 个字节，总共 8 页，这样刚好是 128\*64 的点阵大小。七针 SPI/IIC OLED 模块共有七个管脚，1～7 分别为 GDN、VCC、D0、D1、RES、DC、CS此模块支持四线 SPI、三线 SPI、IIC 接口；由 OLED 的数据手册我们可以知道 0.96 寸 OLED 裸屏是支持四种五种不同接口的；除了前面的三种还有 6800、8080 并口方式；由于这两种接口占用数据 线 比 较 多 ； 而 且 不 太 常 用 ， 所 以 模 块 在 设 计 的 时 候 没 有 引 出 来 。



通信接口是通过 BS0,BS1,BS2 三个管脚来配置的。

* + 1. NRF24L01 无线模块简介

NRF24L01无线模块，采用的芯片是NRF24L01，该芯片的主要特点如下：2.4G全球开放的ISM频段，免许可证使用。最高工作速率 2Mbps，高校的 GFSK调制，抗干扰能力强。125个可选的频道，满足多点通信和调频通信的需要。内置 CRC 检错和点对多点的通信地址控制。低工作电压（1.9-3.6V）。可设置自动应答，确保数据可靠传输。该芯片通过 SPI 与外部 MCU 通信， 最大的 SPI 速度可以达到 10Mhz，所以在后面软件编程的时候 SPI 速度不能高于这个最大值。 本章我们用到的模块是深圳云佳科技生产的 NRF24L01， 该模块已经被很多公司大量使用， 成熟度和稳定性都是相当不错的。 该模块的外形和引脚图如图 2-8 所示：



NRF24L01 模块外观引脚图

模块 VCC 脚的电压范围为 1.9-3.6V，建议不要超过 3.6V，否则可能烧坏模块，一般用 3.3V 电压比较合适。除了VCC和GND脚，其他引脚都可以和 5V单片机的 IO 口直连，正是因为其兼容 5V 单片机的 IO，故使用上具有很大优势。nRF24L01可以设置为以下几种主要的模式：

nRF24L01的模式设置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模式 | PWR\_UP | PRIM\_RX | CE | FIFO寄存器状态 |
| 接受模式 | 1 | 1 | 1 | - |
| 发送模式 | 1 | 0 | 1 | 数据在TX FIFO寄存器中 |
| 发送模式 | 1 | 0 | 1-0 | 停留在发送模式，直到数据发送完 |
| 待机模式2 | 1 | 0 | 1 | TX FIFO 为空 |
| 待机模式1 | 1 | - | 0 | 无数据传输 |
| 掉电模式 | 0 | - | - | - |

nRF24L01在不同模式下的引脚功能

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 引脚名称 | 方向 | 发送模式 | 接受模式 | 待机模式 | 掉电模式 |
| CE | 输入 | 高电平>10us | 高电平 | 低电平 | - |
| CSN | 输入 | SPI片选使能，低电平使能 | | | |
| SCK | 输入 | SPI时钟 | | | |
| MOSI | 输入 | SPI串行输入 | | | |
| MISO | 三态输出 | SPI串行输出 | | | |
| IRQ | 输出 | 中断，低电平使能 | | | |

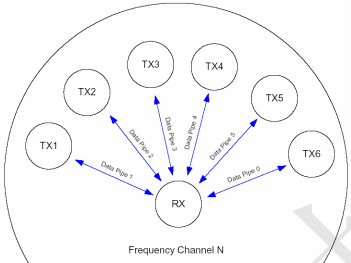
待机模式：待机模式 I 在保证快速启动的同时减少系统平均消耗电流。在待机模式 I 下，晶振正常工作。在待机模式 II 下部分时钟缓冲器处在工作模式。 当发送端 TX FIFO 寄存器为空并且 CE 为高电平时进入待机模式II。在待机模式期间，寄存器配置字内容保持不变。

掉电模式：在掉电模式下,nRF24L01各功能关闭，保持电流消耗最小。进入掉电模式后，nRF24L01 停止工作，但寄存器内容保持不变。启动时间见表格 13。掉电模式由寄存器中 PWR\_UP 位来控制。

nRF24L01 有如下几种数据包处理方式：1.ShockBurstTM（与 nRF2401， nRF24E1，nRF2402，nRF24E2数据传输率为1Mbps 时相同）。2.增强型 ShockBurstTM 模式。

ShockBurstTM 模式：ShockBurst 模式下 nRF24L01 可以与成本较低的低速 MCU 相连。高速信号处理是由芯片内部的射频协议处理的， nRF24L01 提供 SPI 接口，数据率取决于单片机本身接口速度。 ShockBurst 模式通过允许与单片机低速通信而无线部分高速通信，减小了通信的平均消耗电流。在 ShockBurstTM 接收模式下，当接收到有效的地址和数据时 IRQ 通知 MCU，随后 MCU 可将接收到的数据从 RX FIFO 寄存器中读出。在 ShockBurstTM 发送模式下， nRF24L01 自动生成前导码及 CRC 校验，参见表格 12。数据发送完毕后 IRQ 通知 MCU。减少了 MCU 的查询时间，也就意味着减少了 MCU 的工作量同时减少了软件的开发时间。 nRF24L01 内部有三个不同的 RX FIFO 寄存器（ 6 个通道共享此寄存器）和三个不同的 TX FIFO 寄存器。在掉电模式下、待机模式下和数据传输的过程中 MCU 可以随时访问 FIFO 寄存器。这就允许 SPI接口可以以低速进行数据传送，并且可以应用于 MCU 硬件上没有 SPI 接口的情况下。

增强型的 ShockBurstTM 模式：增强型 ShockBurstTM 模式可以使得双向链接协议执行起来更为容易、有效。典型的双向链接为：发送方要求终端设备在接收到数据后有应答信号，以便于发送方检测有无数据丢失。一旦数据丢失，则通过重新发送功能将丢失的数据恢复。 增强型的 ShockBurstTM 模式可以同时控制应答及重发功能而无需增加 MCU工作量。



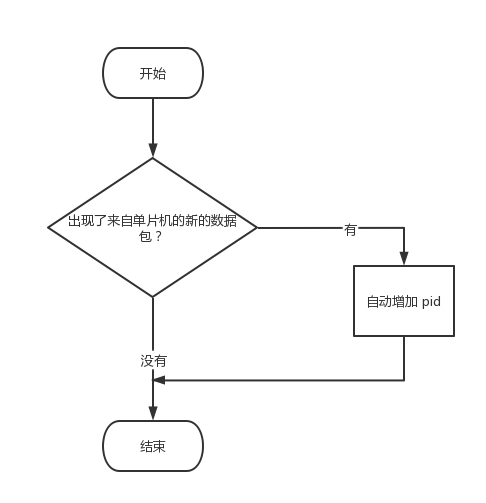
nRF2402在星形网络中的结构图

nRF24L01 在接收模式下可以接收 6 路不同通道的数据，见图 4。每一个数据通道使用不同的地址，但是共用相同的频道。也就是说 6 个不同的 nRF24L01 设置为发送模式后可以与同一个设置为接收模式的nRF24L01 进行通讯，而设置为接收模式的 nRF24L01 可以对这 6 个发射端进行识别。数据通道 0 是唯一的一个可以配置为 40 位自身地址的数据通道。 1~5 数据通道都为 8 位自身地址和 32 位公用地址。所有的数据通道都可以设置为增强型 ShockBurst 模式。nRF24L01 在确认收到数据后记录地址，并以此地址为目标地址发送应答信号。在发送端，数据通道0被用做接收应答信号，因此，数据通道 0 的接收地址要与发送端地址相等以确保接收到正确的应答信号。nRF24L01 配置为增强型的 ShockBurstTM 发送模式下时，只要 MCU 有数据要发送， nRF24L01 就会启动 ShockBurstTM 模式来发送数据。在发送完数据后 nRF24L01 转到接收模式并等待终端的应答信号。如果没 有 收 到 应 答 信 号 ， nRF24L01 将 重 发 相 同 的 数 据 包 ， 直 到 收 到 应 答 信 号 或 重 发 次 数 超 过SETUP\_RETR\_ARC 寄存器中设置的值为止，如果重发次数超过了设定值，则产生 MAX\_RT 中断。只要收到确认信号， nRF24L01 就认为最后一包数据已经发送成功（接收方已经收到数据），把 TX FIFO中的数据清除掉并产生 TX\_DS 中断（ IRQ 引脚置高）。在增强型 ShockBurstTM 模式下， nRF24L01 有如下的特征：当工作在应答模式时，快速的空中传输及启动时间，极大的降低了电流消耗。低成本。 nRF24L01 集成了所有高速链路层操作，比如：重发丢失数据包和产生应答信号。无需单片机硬件上一定有 SPI 口与其相连。 SPI 接口可以利用单片机通用 I/O 口进行模拟。由于空中传输时间很短，极大的降低了无线传输中的碰撞现象由于链路层完全集成在芯片上，非常便于软硬件的开发。

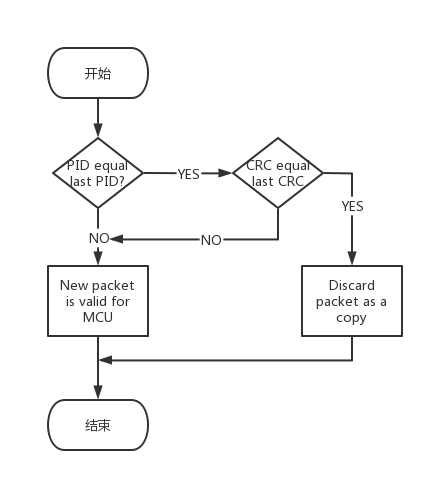
增强型 ShockBurstTM 发送模式：1、 配置寄存器位 PRIM\_RX 为低。2、 当 MCU 有数据要发送时，接收节点地址（ TX\_ADDR）和有效数据(TX\_PLD)通过 SPI 接口写入nRF24L01。 发送数据的长度以字节计数从 MCU 写入 TX FIFO。当 CSN 为低时数据被不断的写入。发送端发送完数据后，将通道 0 设置为接收模式来接收应答信号，其接收地址(RX\_ADDR\_P0)与接收端地址(TX\_ADDR)相同。例：在图 5 中数据通道 5 的发送端(TX5)及接收端(RX)地址设置TX5：TX\_ADDR=0xB3B4B5B605。TX5：RX\_ADDR\_P0=0xB3B4B5B605。RX： RX\_ADDR\_P5=0xB3B4B5B605。3、 设置 CE 为高，启动发射。 CE 高电平持续时间最小为 10 us。4、 nRF24L01 ShockBurstTM 模式：无线系统上电，启动内部 16MHz 时钟，无线发送数据打包（见数据包描述），高速发送数据（由 MCU 设定为 1Mbps 或 2Mbps）。5、 如果启动了自动应答模式（自动重发计数器不等于 0， ENAA\_P0=1），无线芯片立即进入接收模式。如果在有效应答时间范围内收到应答信号，则认为数据成功发送到了接收端，此时状态寄存器的TX\_DS 位置高并把数据从 TX FIFO 中清除掉。如果在设定时间范围内没有接收到应答信号，则重新发送数据。如果自动重发计数器（ ARC\_CNT）溢出（超过了编程设定的值），则状态寄存器的MAX\_RT 位置高。不清除 TX FIFO 中的数据。当 MAX\_RT 或 TX\_DS 为高电平时 IRQ 引脚产生中断。 IRQ 中断通过写状态寄存器来复位（见中断章节）。如果重发次数在达到设定的最大重发次数时还没有收到应答信号的话，在 MAX\_RX 中断清除之前不会重发数据包。数据包丢失计数器(PLOS\_CNT)在每次产生 MAX\_RT 中断后加一。也就是说：重发计数器 ARC\_CNT 计算重发数据包次数， PLOS\_CNT 计算在达到最大允许重发次数时仍没有发送成功的数据包个数。如果 CE 置低，则系统进入待机模式 I。如果不设置 CE 为低，则系统会发送 TX FIFO 寄存器中下一包数据。如果 TX FIFO 寄存器为空并且 CE 为高则系统进入待机模式 II.7、 如果系统在待机模式 II，当 CE 置低后系统立即进入待机模式 I.

增强型 ShockBurstTM 接收模式：1、 ShockBurstTM 接收模式是通过设置寄存器中 PRIM\_RX 位为高来选择的。准备接收数据的通道必须被使能（ EN\_RXADDR 寄存器），所有工作在增强型 ShockBurstTM 模式下的数据通道的自动应答功能是由(EN\_AA 寄存器)来使能的，有效数据宽度是由 RX\_PW\_Px 寄存器来设置的。地址的建立过程见增强型 ShockBurstTM 发送章节。2、 接收模式由设置 CE 为高来启动。3、 130us 后 nRF24L01 开始检测空中信息。4、 接收到有效的数据包后（地址匹配、 CRC 检验正确），数据存储在 RX\_FIFO 中，同时 RX\_DR 位置高，并产生中断。状态寄存器中 RX\_P\_NO 位显示数据是由哪个通道接收到的。5、 如果使能自动确认信号，则发送确认信号。6、 MCU 设置 CE 脚为低，进入待机模式 I（低功耗模式）。7、 MCU 将数据以合适的速率通过 SPI 口将数据读出。8、 芯片准备好进入发送模式、接收模式或掉电模式。两种数据双方向的通讯方式如果想要数据在双方向上通讯,PRIM\_RX 寄存器必须紧随芯片工作模式的变化而变化。处理器必须保证 PTX 和 PRX 端的同步性。在 RX\_FIFO 和 TX\_FIFO 寄存器中可能同时存有数据。自动应答（ RX）：自动应答功能减少了外部 MCU 的工作量，并且在鼠标/键盘等应用中也可以不要求硬件一定有 SPI 接口，因此降低成本减少电流消耗。自动重应答功能可以通过 SPI 口对不同的数据通道分别进行配置。在自动应答模式使能的情况下，收到有效的数据包后，系统将进入发送模式并发送确认信号。发送完确认信号后，系统进入正常工作模式（工作模式由 PRIM\_RX 位和 CE 引脚决定）。自动重发功能（ ART） (TX)：自动重发功能是针对自动应答系统的发送方。 SETUP\_RETR 寄存器设置：启动重发数据的时间长度。在每次发送结束后系统都会进入接收模式并在设定的时间范围内等待应答信号。接收到应答信号后，系统转入正常发送模式。如果 TX FIFO 中没有待发送的数据且 CE 脚电平为低，则系统将进入待机模式 I。如果没有收到确认信号，则系统返回到发送模式并重发数据直到收到确认信号或重发次数超过设定值（达到最大的重发次数）。有新的数据发送或 PRIM\_RX 寄存器配置改变时丢包计数器复位。

数据包识别和 CRC 校验应用于增强型 ShockBurstTM 模式下：每一包数据都包括两位的 PID（数据包识别）来识别接收的数据是新数据包还是重发的数据包。 PID识别可以防止接收端同一数据包多次送入 MCU。在发送方每从 MCU 取得一包新数据后 PID 值加一。 PID和 CRC 校验应用在接收方识别接收的数据是重发的数据包还是新数据包。如果在链接中有一些数据丢失了，则 PID 值与上一包数据的 PID 值相同。如果一包数据拥有与上一包数据相同的 PID 值， nRF24L01 将对两包数据的 CRC 值进行比较。 如果 CRC 值也相同的话就认为后面一包是前一包的重发数据包而被舍弃。1：接收方：接收方对新接收数据包的 PID 值与上一包进行比较。如果 PID 值不同，则认为接收的数据包是新数据包。如果 PID 值与上一包相同，则新接收的数据包有可能与前一包相同。接收方必须确认 CRC值是否相等，如果 CRC 值与前一包数据的 CRC 值相等，则认为是同一包数据并将其舍弃。2：发送方：每发送一包新的数据则发送方的 PID 值加一。

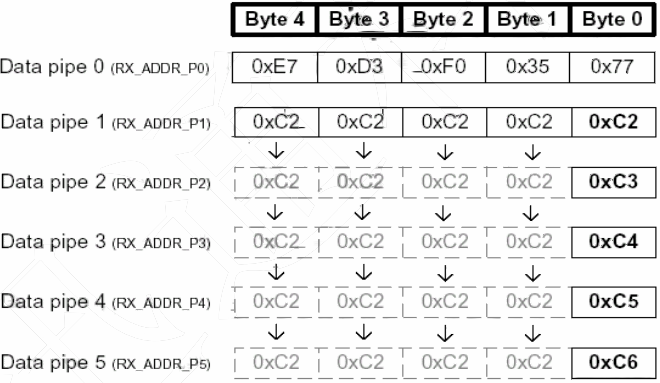


PID的生成



PID的检测

CRC 校验的长度是通过 SPI 接口进行配置的。一定要注意 CRC 计算范围包括整个数据包：地址、 PID和有效数据等。若 CRC 校验错误则不会接收数据包，这一点是接收数据包的附加要求，在上图没有说明。载波检测—CD：当接收端检测到射频范围内的信号时将 CD 置高，否则 CD 为低。内部的 CD 信号在写入寄存器之前是经过滤波的，内部 CD 高电平状态至少保持 128us 以上。在增强型ShockBurstTM模式中只有当发送模块没有成功发送数据时，推荐使用CD检测功能。如果发送端PLOS\_CNT显示数据包丢失率太高时，可将其设置位接收模式检测CD值，如果CD为高（说明通道出现了拥挤现象），需要更改通信频道；如果CD为低电平状态（距离超出通信范围），可保持原有通信频道，但需作其它调整。数据通道：nRF24L01 配置为接收模式时可以接收 6 路不同地址相同频率的数据。每个数据通道拥有自己的地址并且可以通过寄存器来进行分别配置。数据通道是通过寄存器 EN\_RXADDR 来设置的，默认状态下只有数据通道 0 和数据通道 1 是开启状态的。每一个数据通道的地址是通过寄存器 RX\_ADDR\_Px 来配置的。通常情况下不允许不同的数据通道设置完全相同的地址。数据通道 0 有 40 位可配置地址。数据通道 1~5 的地址为： 32 位共用地址+各自的地址（最低字节）。



通道0-5的地址设置

图 2-12 所示的是数据通道 1~5 的地址设置方法举例。所有数据通道可以设置为多达 40 位，但是 1~5 数据通道的最低位必须不同。当从一个数据通道中接收到数据，并且此数据通道设置为应答方式的话，则 nRF24L01 在收到数据后产生应答信号，此应答信号的目标地址为接收通道地址。寄存器配置有些是针对所有数据通道的， 有些则是针对个别的。 如下设置举例是针对所有数据通道的：CRC 使能/禁止, CRC 计算,接收地址宽度,频道设置,无线数据通信速率,LNA 增益,射频输出功率。

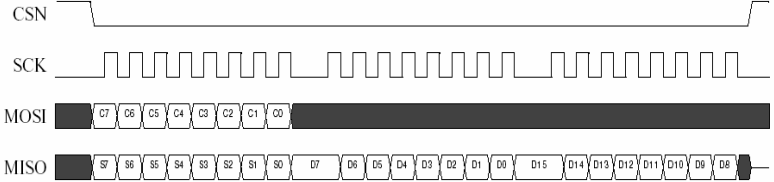
寄存器配置：nRF24L01 所有配置都在配置寄存器中，所有寄存器都是通过 SPI 口进行配置的。SPI 接口：SPI 接口是标准的 SPI 接口，其最大的数据传输率为 10Mbps。大多数寄存器是可读的。SPI 指令设置：SPI 接口可能用到的指令在下面有所说明。 CSN 为低后 SPI 接口等待执行指令。每一条指令的执行都必须通过一次 CSN 由高到低的变化。SPI 指令格式：<命令字：由高位到低位（每字节）><数据字节：低字节到高字节，每一字节高位在前>

nRF24L01 SPI串行口指令设置

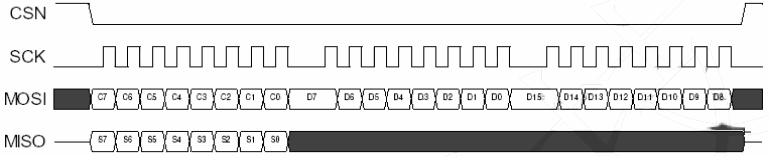
|  |  |  |
| --- | --- | --- |
| 指令名称 | 指令格式 | 操作 |
| R\_REGISTER | 000A AAAA | 读配置寄存器。 AAAAA 指出读操作的寄存器地址 |
| W\_REGISTER | 001A AAAA | 写配置寄存器。 AAAAA 指出写操作的寄存器地址。只有在掉电模式和待机模式下可操作。 |
| R\_RX\_PAYLOAD | 0110 0001 | 读 RX 有效数据： 1-32 字节。读操作全部从字节 0 开始。当读 RX有效数据完成后， FIFO 寄存器中有效数据被清除。应用于接收模式下。 |
| W\_RX\_PAYLOAD | 1010 0000 | 写 TX 有效数据： 1-32 字节。写操作从字节 0 开始。应用于发射模式下。 |
| FLUSH\_TX | 1110 0001 | 清除 TX FIFO 寄存器，应用于发射模式下。 |
| FLUSH\_RX | 1110 0010 | 清除 RX FIFO 寄存器，应用于接收模式下。在传输应答信号过程中不应执行此指令。也就是说，若传输应答信号过程中执行此指令的话将使得应答信号不能被完整的传输。 |
| REUSE\_TX\_PL | 1110 0011 | 重新使用上一包有效数据。当 CE 为高过程中，数据包被不断的重新发射。在发射数据包过程中必须禁止数据包重利用功能。 |
| NOP | 1111 1111 | 空操作。可以用来读状态寄存器。 |

R\_REGISTER 和 W\_REGISTER 寄存器可能操作单字节或多字节寄存器。当访问多字节寄存器时首先要读/写的是最低字节的高位。在所有多字节寄存器被写完之前可以结束写 SPI 操作，在这种情况下没有写完的高字节保持原有内容不变。例如： RX\_ADDR\_P0 寄存器的最低字节可以通过写一个字节给寄存器RX\_ADDR\_P0 来改变。在 CSN 状态由高变低后可以通过 MISO 来读取状态寄存器的内容。中断：nRF24L01 的中断引脚（ IRQ）为低电平触发，当状态寄存器中 TX\_DS、 RX\_DR 或 MAX\_RT 为高时触发中断。当 MCU 给中断源写‘ 1’时，中断引脚被禁止。可屏蔽中断可以被 IRQ 中断屏蔽。通过设置可屏蔽中断位为高，则中断响应被禁止。默认状态下所有的中断源是被禁止的。

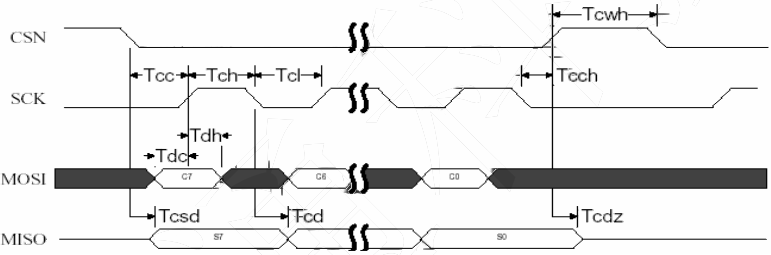
SPI 时序：Cn-SPI 指令位。Sn-状态寄存器位。Dn-数据位（备注：由低字节到高字节，每个字节中高位在前）。



SPI读操作



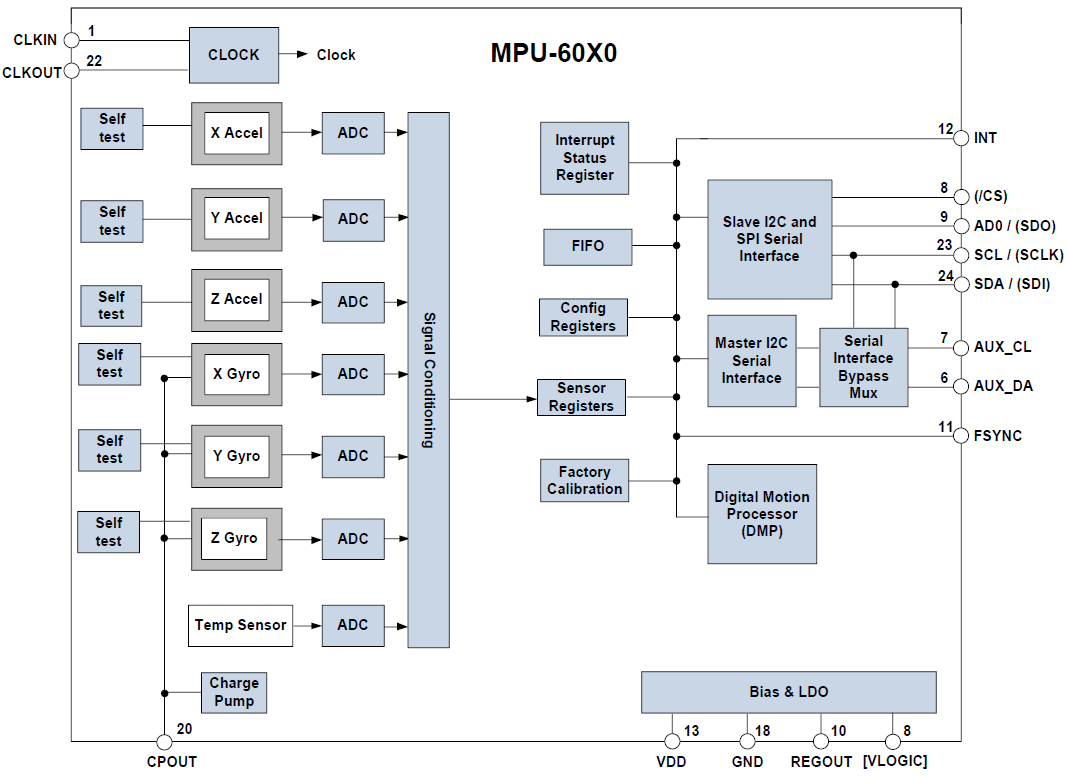
SPI写操作



SPI NOP操作时序图

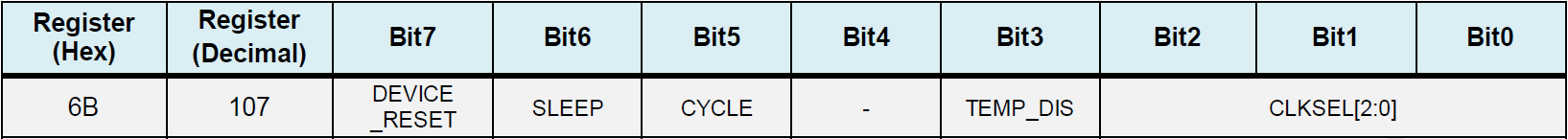
* + 1. MPU6050简介规范化

MPU6050是InvenSense公司推出的全球首款整合性6轴运动处理组件，内带3轴陀螺仪和3轴加速度传感器，并且含有一个第二IIC接口，可用于连接外部磁力传感器，利用自带数字运动处理器（DMP: Digital Motion Processor）硬件加速引擎，通过主IIC接口，可以向应用端输出完整的9轴姿态融合演算数据。有了DMP，我们可以使用InvenSense公司提供的运动处理资料库，非常方便的实现姿态解算，降低了运动处理运算对操作系统的负荷，同时大大降低了开发难度 。①自带数字运动处理（ DMP: Digital Motion Processing ），可以输出6轴或9轴（需外接磁传感器）姿态解算数据。②集成可程序控制，测量范围为±250、±500、±1000与±2000°/sec 的3轴角速度感测器(陀螺仪)。③集成可程序控制，范围为±2g、±4g、±8g和±16g的3轴加速度传感器。④自带数字温度传感器 。⑤可输出中断(interrupt)，支持姿势识别、摇摄、画面放大缩小、滚动、快速下降中断、high-G中断、零动作感应、触击感应、摇动感应功能。⑥自带1024字节FIFO，有助于降低系统功耗。⑦高达400Khz的IIC通信接口。⑧超小封装尺寸：4x4x0.9mm（QFN）。



MPU6050框图

AD0=0地址=0X68；AD0=1地址=0X69；



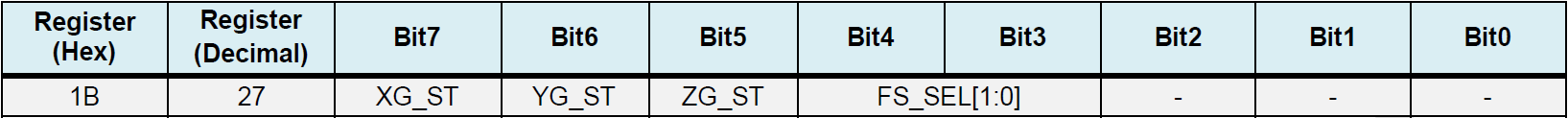
电源管理寄存器1（0X6B）

DEVICE\_RESE=1，复位MPU6050，复位完成后，自动清零。SLEEP=1，进入睡眠模式；SLEEP=0，正常工作模式。TEMP\_DIS，用于设置是否使能温度传感器，设置为0，则使能CLKSEL[2:0]，用于选择系统时钟源，如下表所示：

系统时钟源选择表

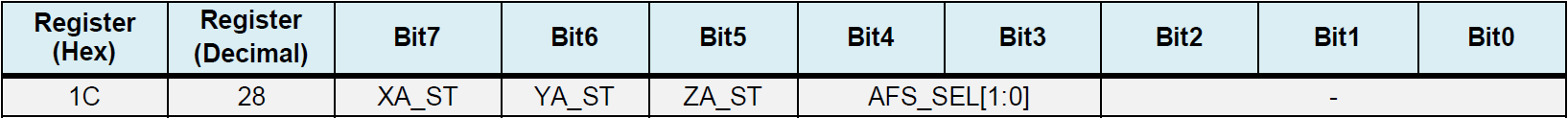
|  |  |
| --- | --- |
| CLKSEL[2:0] | 时钟源 |
| 000 | 内部8M RC晶振 |
| 001 | PLL，使用X轴陀螺作为参考 |
| 010 | PLL，使用Y轴陀螺作为参考 |
| 011 | PLL，使用Z轴陀螺作为参考 |
| 100 | PLL，使用外部32.768Khz作为参考 |
| 101 | PLL，使用外部19.2Mhz作为参考 |
| 110 | 保留 |
| 111 | 关闭时钟，保持时序产生电路复位状态 |

陀螺仪配置寄存器，FS\_SEL[1:0]这两个位，用于设置陀螺仪的满量程范围：0，±250°/S；1，±500°/S；2，±1000°/S；3，±2000°/S；一般设置为3，即±2000°/S，因为陀螺仪的ADC为16位分辨率，所以得到灵敏度为：65536/4000=16.4LSB/(°/S)。



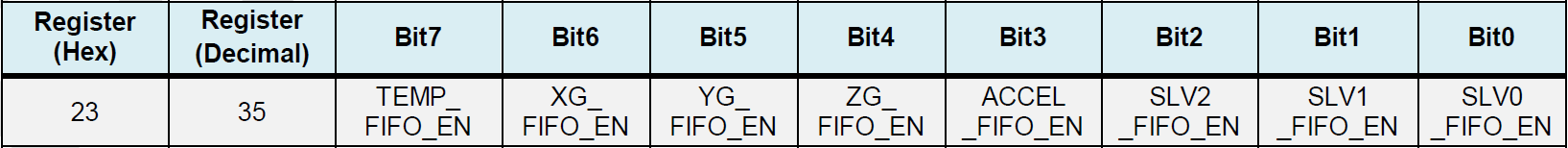
陀螺仪配置寄存器（0X1B）

加速度传感器配置寄存器，AFS\_SEL[1:0]这两个位，用于设置加速度传感器的满量程范围：0，±2g；1，±4g；2，±8g；3，±16g；一般设置为0，即±2g，因为加速度传感器的ADC也是16位，所以得到灵敏度为：65536/4=16384LSB/g。



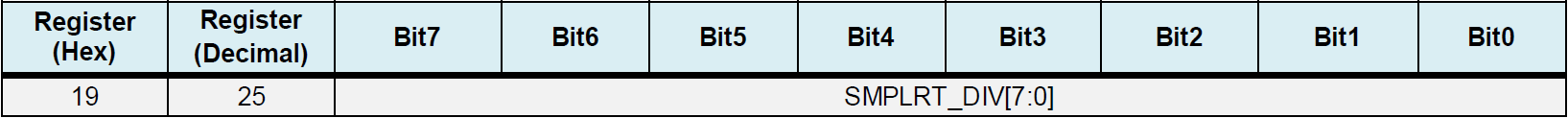
加速度传感器配置寄存器（0X1C）

FIFO使能寄存器用于控制FIFO使能，在简单读取传感器数据的时候，可以不用FIFO，设置对应位为：0，即可禁止FIFO，设置为1，则使能FIFO。注意：加速度传感器的3个轴，全由1个位（ACCEL\_FIFO\_EN）控制，只要该位置1，则加速度传感器的三个通道都开启FIFO了。

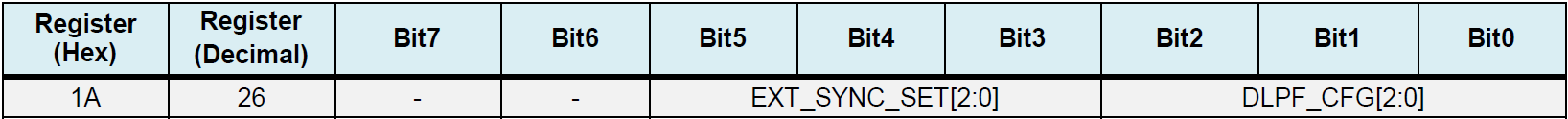


FIFO使能寄存器（0X23）

该寄存器用于设置MPU6050的陀螺仪采样频率，计算公式为：采样频率 = 陀螺仪输出频率 / (1+SMPLRT\_DIV)这里陀螺仪的输出频率，是1Khz或者8Khz，与数字低通滤波器（DLPF）的设置有关，当DLPF\_CFG=0/7的时候，频率为8Khz，其他情况是1Khz。而且DLPF滤波频率一般设置为采样率的一半。采样率，我们假定设置为50Hz，那么：SMPLRT\_DIV=1000/50-1=19。



陀螺仪采样率分频寄存器（0X19）



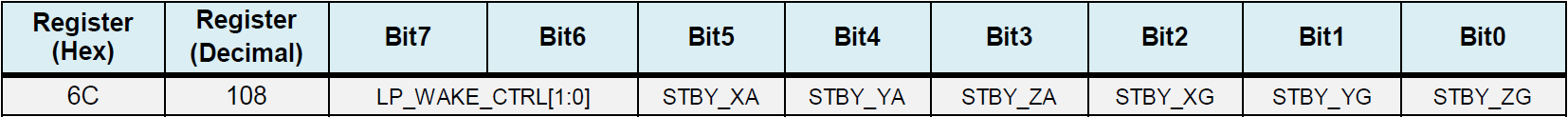
配置寄存器（0X1A）

重点看数字低通滤波器（DLPF）的设置位，即：DLPF\_CFG[2:0]，加速度计和陀螺仪，都是根据这三个位的配置进行过滤的，如下表：

数字低通滤波器（DLPF）的设置位

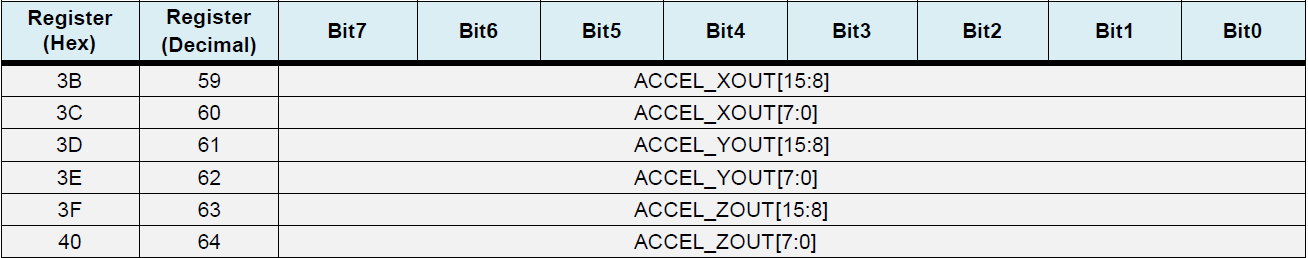
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DLPF\_CFG[2:0] | 加速度传感器Fs=1Khz | | 角速度传感器（陀螺仪） | | |
| 带宽(Hz) | 延迟（ms） | 带宽(Hz) | 延迟（ms） | Fs(Khz) |
| 000 | 260 | 0 | 256 | 0.98 | 8 |
| 001 | 184 | 2.0 | 188 | 1.9 | 1 |
| 010 | 94 | 3.0 | 98 | 2.8 | 1 |
| 011 | 44 | 4.9 | 42 | 4.8 | 1 |
| 100 | 21 | 8.5 | 20 | 8.3 | 1 |
| 101 | 10 | 13.8 | 10 | 13.4 | 1 |
| 110 | 5 | 19.0 | 5 | 18.6 | 1 |
| 111 | 保留 | | 保留 | | 8 |

该寄存器的LP\_WAKE\_CTRL用于控制低功耗时的唤醒频率。剩下的6位，分别控制加速度和陀螺仪的x/y/z轴是否进入待机模式，这里我们全部都不进入待机模式，所以全部设置为：0 ，即可。



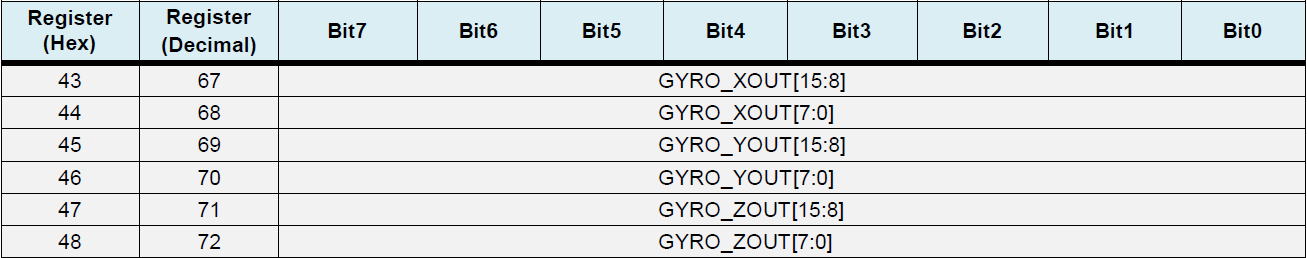
电源管理寄存器2（0X6C）

加速度传感器数据输出寄存器总共由6个寄存器组成，输出X/Y/Z三个轴的加速度传感器值，高字节在前，低字节在后。



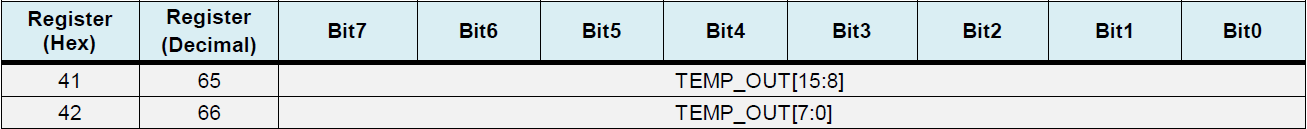
加速度传感器数据输出寄存器（0X3B~0X40）

陀螺仪数据输出寄存器总共由6个寄存器组成，输出X/Y/Z三个轴的陀螺仪传感器数据，高字节在前，低字节在后。



陀螺仪数据输出寄存器（0X43~0X48）

通过读取0X41（高8位）和0X42（低8位）寄存器得到，温度换算公式为：Temperature = 36.53 + regval/340。其中，Temperature为计算得到的温度值，单位为℃，regval为从0X41和0X42读到的温度传感器值。



温度传感器数据输出寄存器（0X41~0X42）

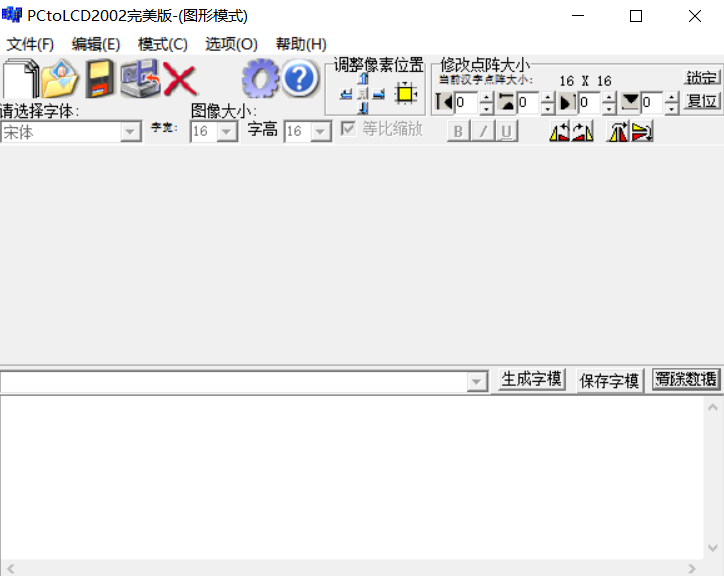
通过前面配置，可以正常读取MPU6050的加速度传感器、陀螺仪和温度传感器的数据，但是实际使用的时候（比如做四轴），我们更希望得到姿态数据，即欧拉角：航向角（yaw）、横滚角（roll）和俯仰角（pitch）。要得到欧拉角数据，就得利用我们的原始数据，进行姿态融合解算，这个比较复杂，知识点比较多，初学者不易掌握。而MPU6050自带了数字运动处理器，即DMP，并且，InvenSense提供了一个MPU6050的嵌入式运动驱动库，结合MPU6050的DMP，可以将我们的原始数据，直接转换成四元数输出，而得到四元数之后，就可以很方便的计算出欧拉角，从而得到yaw、roll和pitch。使用内置的DMP，可以大大简化代码设计，MCU不用进行姿态解算过程，大大降低了MCU的负担，从而有更多的时间去处理其他事件，提高系统实时性。InvenSense提供的MPU6050运动驱动库是基于MSP430的，我们需要将其移植一下，才可以用到STM32上面。官方DMP驱动库移植，主要是实现这4个函数：i2c\_write、i2c\_read、delay\_ms和get\_ms。

* 1. 软件部分
     1. 软件工具MDK5 简介

MDK 源自德国的 KEIL 公司，是 RealView MDK 的简称。在全球 MDK 被超过 10 万的嵌入式开发工程师使用。目前最新版本为： MDK5.14，该版本使用 uVision5 IDE 集成开发环境，是目前针对 ARM 处理器，尤其是 Cortex M 内核处理器的最佳开发工具。MDK5 向后兼容 MDK4 和 MDK3 等，以前的项目同样可以在 MDK5 上进行开发(但是头文件方面得全部自己添加)， MDK5 同时加强了针对 Cortex-M 微控制器开发的支持，并且对传统的开发模式和界面进行升级， MDK5 由两个部分组成： MDK Core 和 Software Packs。其中，Software Packs 可以独立于工具链进行新芯片支持和中间库的升级。

* + 1. Oled显示屏专用取模工具

该软件可生成中英文数字混合的字符串的字模数据.可选择字体，大小，并且可独立调整文字的长和宽，生成任意形状的字符。各种旋转，翻转文字功能。任意调整输出点阵大小，并任意调整字符在点阵中的位置。字模数据输出可自定义各种格式，系统预设了C语言和汇编语言两种格式，并且可自己定义出新的数据输出格式；每行输出数据个数可调。支持四种取模方式：逐行（就是横向逐行取点），逐列（纵向逐列取点），行列（先横向取第一行的8个点作为第一个字节，然后纵向取第二行的8个点作为第二个字节……），列行（先纵向取第一列的前8个点作为第一个字节，然后横向取第二列的前8个点作为第二个字节……）支持阴码（亮点为1），阳码（亮点为0）取模。支持纵向（第一位为低位）（倒向第一位为高位）取模。输出数制可选16进制或10进制。可生成索引文件，用于在生成的大量字库中可快速检索到需要的汉字。动态液晶面板彷真，可调节彷真面板象素点大小和颜色。图形模式下可任意用鼠标作画，左键画图，右键擦图。旋转，翻转，平移等字符模式下的功能也可用与对BMP图象的处理。



软件界面展示

* + 1. CubeMX

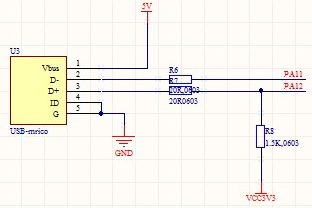
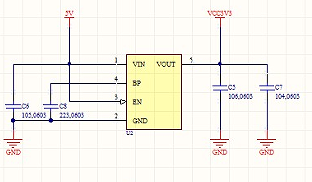
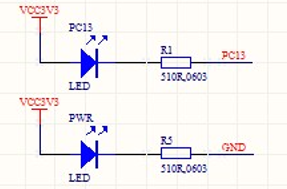
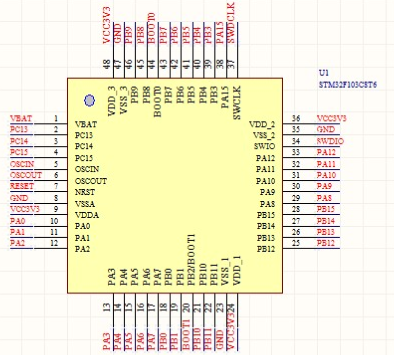
STM32Cube 是一项意法半导体的原创活动, 通过减少开发工作、时间和成本, 使开发者的开发工作更轻松。 STM32Cube 是一个全面的软件平台，包括了 ST 产品的每个系列。 (如， STM32CubeF4是针对 STM32F4 系列)。平台包括了 STM32Cube 硬件抽象层和一套的中间件组件 (RTOS， USB，FS， TCP/IP， Graphics，等等)。在Cube工具还没出来之前，在ST的MCU开发都是用标准固件库，标准库自推出以来受到ST的使用者的推崇，现在很多公司也都在使用。但是ST官方在2013年后就没有更新版本，ST官方也全力推HAL（Hardware Abstraction Layer）库。它的存在是为了确保 STM32 系列最大的移植性。HAL 位于操作系统内核与硬件电路之间的接口层，其目的在于将硬件抽象化。CubeMX软件可以直接根据开发人员所需的功能勾选，以HAL库文件基础自动产生代码。开发者就可以轻松应用每一个外设。因此，使用 HAL 库可以大大减少用户的程序编写时间，进而降低开发成本。每个外设驱动都由一组函数组成，这组函数覆盖了该外设所有功能。每个器件的开发都由一个通用 API (application programming interface 应用编程界面)驱动，API 对该驱动程序的结构，函数和参数名称都进行了标准化。在 ST 官方的声明中，HAL 库是大势所趋，在 ST 公司最新开发的部分芯片中，只有 HAL 库而没有标准库，从这点便可以说明，以后的战略目标是逐渐的转向HAL 库 。相对于标准库来说，在使用 CubeMX 生成代码后，工程项目和初始化代码已经完成。cubeMX直接生产的工程支持IAR，keil，TrueSudio。

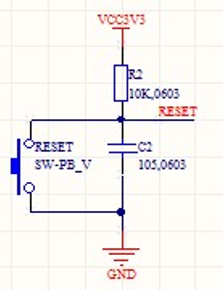
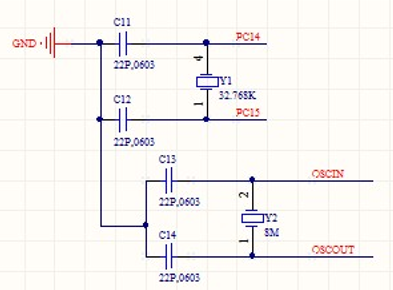
* + 1. Git

Git --- The stupid content tracker, 傻瓜内容跟踪器。Git 是用于 Linux内核开发的版本控制工具。与常用的版本控制工具 CVS, Subversion 等不同，它采用了分布式版本库的方式，不必服务器端软件支持（wingeddevil注：这得分是用什么样的服务端，使用http协议或者git协议等不太一样。并且在push和pull的时候和服务器端还是有交互的。），使源代码的发布和交流极其方便。 Git 的速度很快，这对于诸如 Linux kernel 这样的大项目来说自然很重要。 Git 最为出色的是它的合并跟踪（merge tracing）能力。实际上内核开发团队决定开始开发和使用 Git 来作为内核开发的版本控制系统的时候，世界开源社群的反对声音不少，最大的理由是 Git 太艰涩难懂，从 Git 的内部工作机制来说，的确是这样。但是随着开发的深入，Git 的正常使用都由一些友好的脚本命令来执行，使 Git 变得非常好用，即使是用来管理我们自己的开发项目，Git 都是一个友好，有力的工具。现在，越来越多的著名项目采用 Git 来管理项目开发。

* + 1. 模块流程图

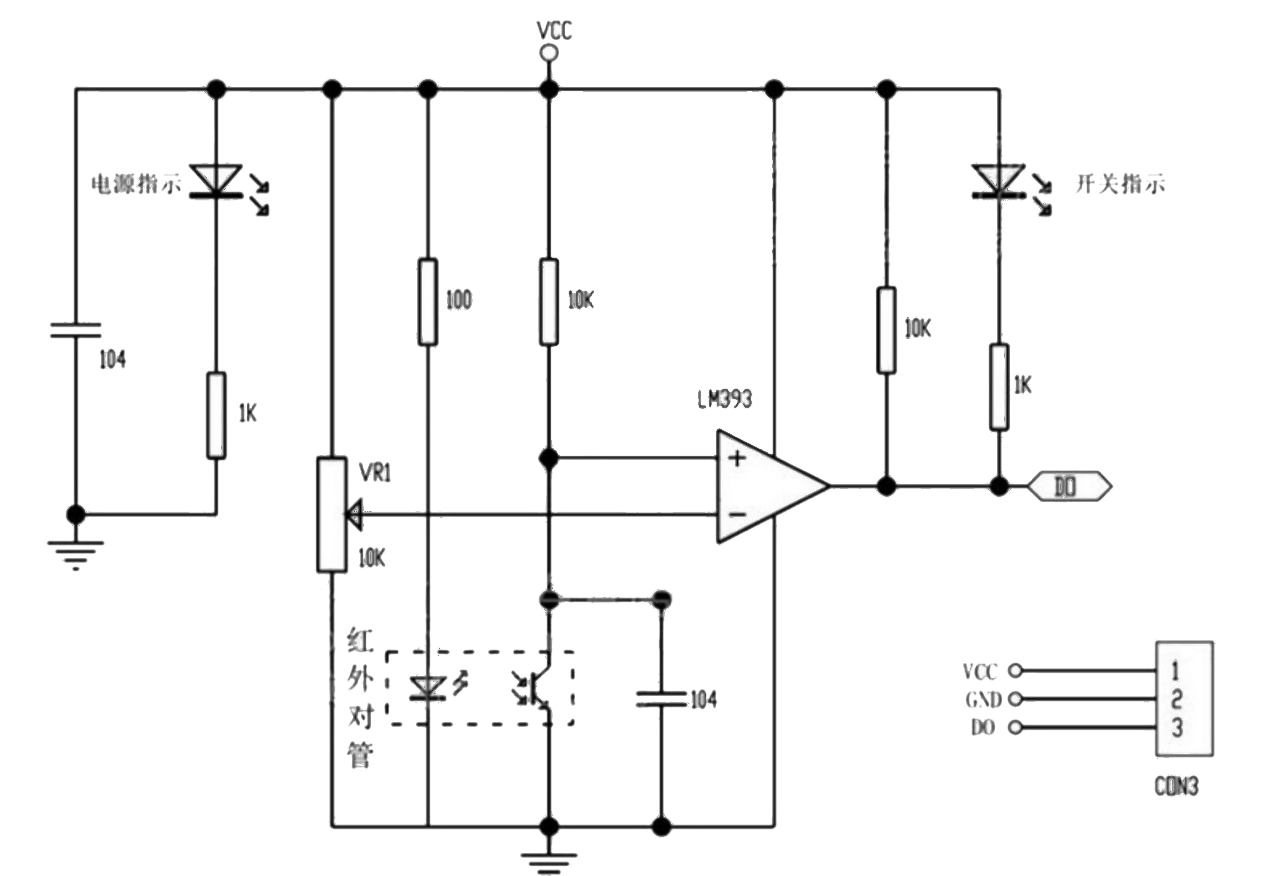
1. 实现功能与电路设计
   1. 实现功能与设计
      1. 实现功能描述
   2. 电路设计
      1. 最小系统板电路





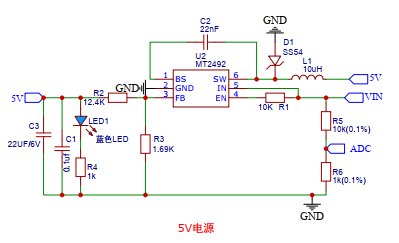
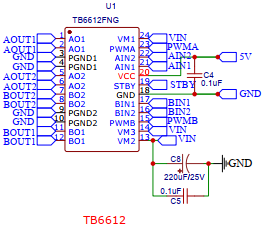
最小系统板C8T6主要部分电路图

* + 1. 红外检测模块电路

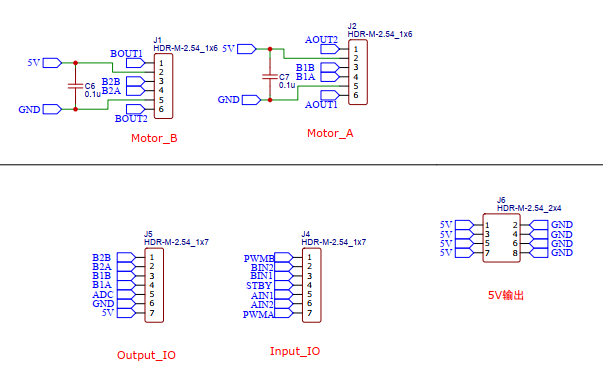


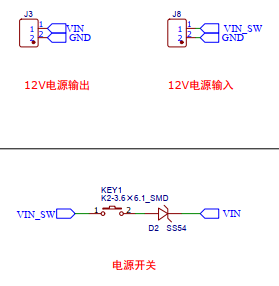
红外检测模块电路图

* + 1. TB6612带稳压模块板原理图



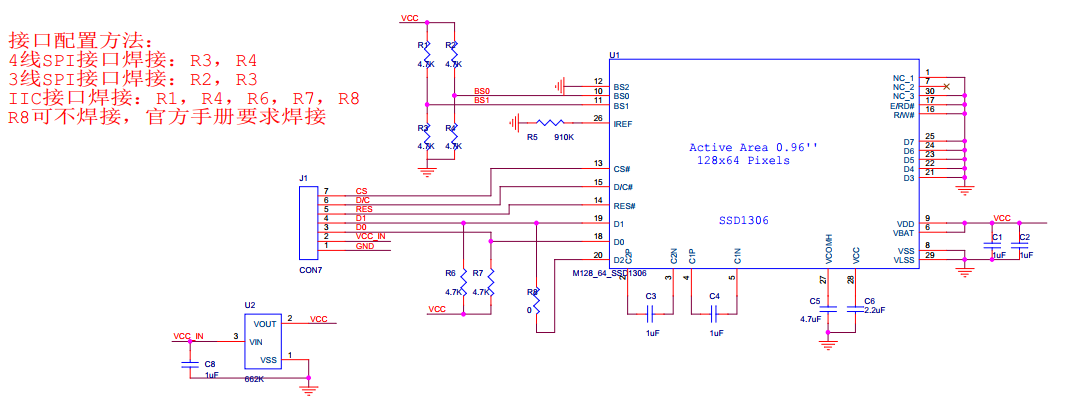
TB6612芯片和电源部分



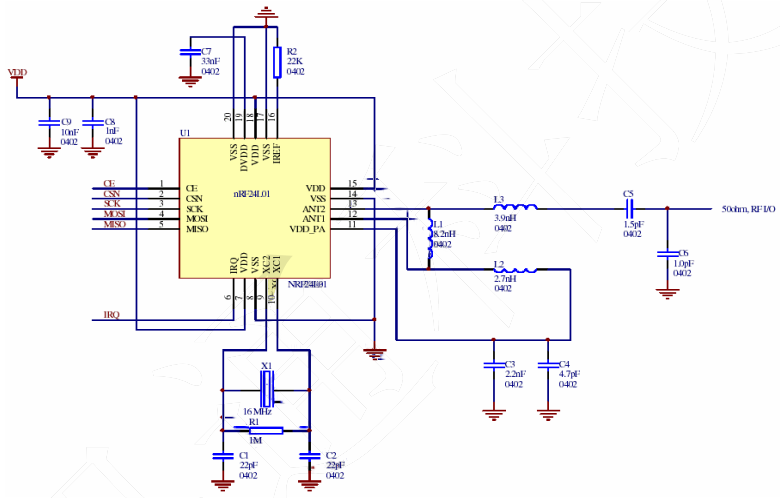


TB6612外接端口部分

* + 1. 0.96OLED显示屏原理图



* + 1. nRF24L01 单端匹配网络：晶振、偏置电阻、去藕电容部分电路



* 1. 软件设计
     1. MPU6050初始化

①初始化IIC接口。②复位MPU6050。由电源管理寄存器1(0X6B)控制。③设置角速度传感器和加速度传感器的满量程范围。由陀螺仪配置寄存器(0X1B)和加速度传感器配置寄存器(0X1C)设置 。④设置其他参数。配置中断，由中断使能寄存器(0X38)控制；设置AUX IIC接口，由户控制寄存器(0X6A)控制；设置FIFO，由FIFO使能寄存器(0X23)控制；陀螺仪采样率 ，由采样率分频寄存器(0X19)控制；设置数字低通滤波器，由配置寄存器(0X1A)控制。⑤设置系统时钟。由电源管理寄存器1(0X6B)控制。一般选择x轴陀螺PLL作为时钟源，以获得更高精度的时钟。⑥使能角速度传感器(陀螺仪)和加速度传感器。由电源管理寄存器2(0X6C)控制。

* + 1. DMP使用

MPU6050 DMP输出的是姿态解算后的四元数，采用q30格式，也就是放大了2的30次方，我们要得到欧拉角，就得做一个转换，代码如下：q0=quat[0] / q30;q1=quat[1] / q30;q2=quat[2] / q30;q3=quat[3] / q30; pitch=asin(-2 \* q1 \* q3 + 2 \* q0\* q2)\* 57.3; roll=atan2(2 \* q2 \* q3 + 2 \* q0 \* q1, -2 \* q1 \* q1 - 2 \* q2\* q2 + 1)\* 57.3;yaw=atan2(2\*(q1\*q2+q0\*q3),q0\*q0+q1\*q1-q2\*q2-q3\*q3)\*57.3;quat[0]~quat[3]：是MPU6050的DMP解算后的四元数，q30格式。q30：是一个常量：1073741824，即2的30次方。

57.3：是弧度转换为角度，即180/π，这样结果就是以度（°）为单位的。

* + 1. 寻迹逻辑程序设计
    2. 电机驱动部分程序设计
    3. 通信部分程序设计
    4. 手动控制部分程序设计
    5. CubeMX对底层BSP(板级支持包)的初始化配置

1. 调试难点和问题
   1. 调试中遇到的重点与难点
      1. MPU6050的DMP调试出现FIFO溢出问题。

源代码：

while(1)

{

delay\_ms(1000);

if(mpu\_dmp\_get\_data(&pitch,&roll,&yaw)==0)

{

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

temp=MPU\_Read\_Temp();

MPU\_Read\_Accel(&ax,&ay,&az);

MPU\_Read\_Gyro(&gx,&gy,&gz);

mpu6050\_send\_data(ax,ay,az,gx,gy,gz); usart1\_report\_imu(ax,ay,az,gx,gy,gz,(int)(roll\*100),(int)(pitch\*100),(int)(yaw\*10));

}

}

通过打印信息分析是没有进入if语句，进入mpu\_dmp\_get\_data发现是卡在dmp\_read\_fifo函数if(dmp\_read\_fifo(gyro, accel, quat, &sensor\_timestamp, &sensors,&more))return 1;再进入dmp\_read\_fifo函数是卡在mpu\_read\_fifo\_stream函数，最终定位为如下函数的return语句：if (fifo\_count > (st.hw->max\_fifo >> 1)) { /\* FIFO is 50% full, better check overflow bit. \*/ if (i2c\_read(st.hw->addr, st.reg->int\_status, 1, tmp)) return -1; // printf("0x%02x, ",tmp[0]); if (tmp[0] & BIT\_FIFO\_OVERFLOW) { mpu\_reset\_fifo(); return -2; } }打印读到的int\_status的值为0x13,即为FIFO溢出产生中断。从语句中看出当FIFO的数量大于50%就会进入中断Check状态，计数寄存器的数据过多，所以溢出可能是因为读取速度过慢导致，回看MPU6050采样率设置为50Hz，即为20ms采样一次，while语句中设置了1s的延时，1s的延时期间导致采样数据过多而溢出，所以把延时改为100ms，问题解决。

* + 1. 红外检测过慢，导致小车错过检测跑道时机，而跑出跑道。
    2. Pwm过低，转弯的驱动能力不足。
  1. 解决方案
  2. 实现展示（附上仿真图或实物照片）

1. 总结

参考文献

[1]范书瑞,赵燕飞,高铁成. arm处理器C语言开发应用(第2版).北京:北京航空航天大学出版社,2001.78~80

[2]占跃华,主审,王明文.c语言程序设计(第2版).北京:北京邮电大学出版社,2003.128~132

[3]刘火良,杨森. FreeRTOS内核实现与应用开发实战指南(第1版).北京:机械工业出版社,2003.145~163

[4]

[5]

[6]

[7]

[8]

[9]

[10]

[11]

[12]

[13]

[14]

[15]

[16]

[17]

[18]

想对老师说的话

附录

附录一：06年11月CD电信新华营业厅日缴费顾客半小时到达数据

（略）

附录二：顾客调查问卷

（略）