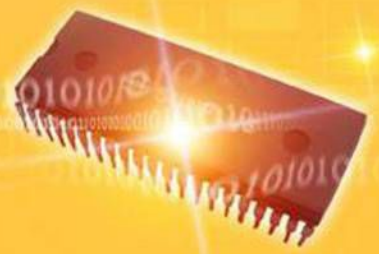
嵌入式系统工程师



电容屏项目

大纲

- 项目简介
- 芯片介绍
- 工作模式与通信时序
- 驱动框架

大纲

- 项目简介
- 芯片介绍
- 工作模式与通信时序
- 驱动框架

项目介绍

➢ 项目介绍

多点电容式触摸屏控制芯片Guitar，采用投射式电容检测原理，由15个驱动通道与10个感应通道组成触摸检测网络，通过内置模拟放大电路、数字运算模块，及高性能MPU得到实时准确的触摸信息，并通过I2C传输给主控芯片。实现“所点即所得”的非凡用户体验。

Guitar可同时识别5个触摸点位的实时准确位置，移动轨迹及触摸力度。并可根据主控需要，读取相应点数的触摸信息。

项目介绍

- 本项目主要通过i2c子系统及input子系统完成驱动与硬件的命令和信息交互，掌握常见触摸屏的工作原理
- 通过此项目的练习来掌握i2c子系统，并能熟练应用

大纲

- 项目简介
- 芯片介绍
- 工作模式与通信时序
- 驱动框架

芯片介绍

- 芯片介绍

guitar触摸屏：

内置电容检测电路及高性能MPU

采用投射式电容检测方式

电容检测频率：60Hz

触摸点坐标实时输出

支持配置固定触摸按键位置

统一软件版本适用于多种尺寸的电容屏

芯片介绍

- 环境适应性能
  - 初始化自动校准
  - 自动温漂补偿
  - 工作温度：-40℃~+85℃，湿度：≦95%RH
  - 储存温度：-60℃~+150℃，湿度：≦95%RH
- 通讯接口
  - 标准I2C通讯接口
  - 从设备工作模式

芯片介绍

- 响应时间

  Green mode: <48ms

  Sleep mode: <642.5ms

  Initialization: <642.5ms

- 电源电压：

  数字部分DVDD: 2.5V~3.6V

  模拟部分AVDD: 2.5V~3.6V

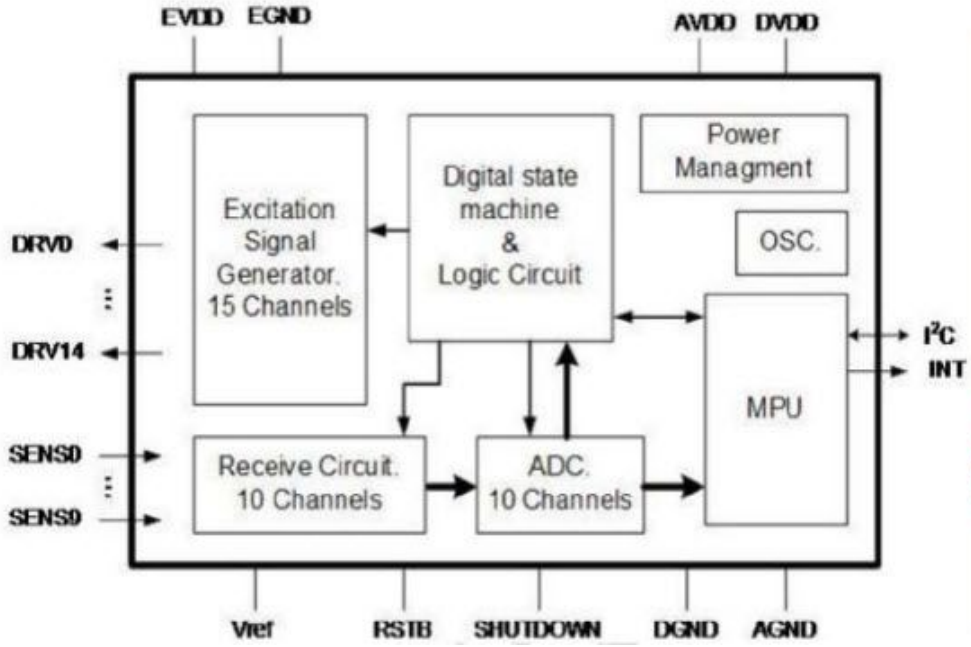
  触控驱动EVDD: 2.7V~35V

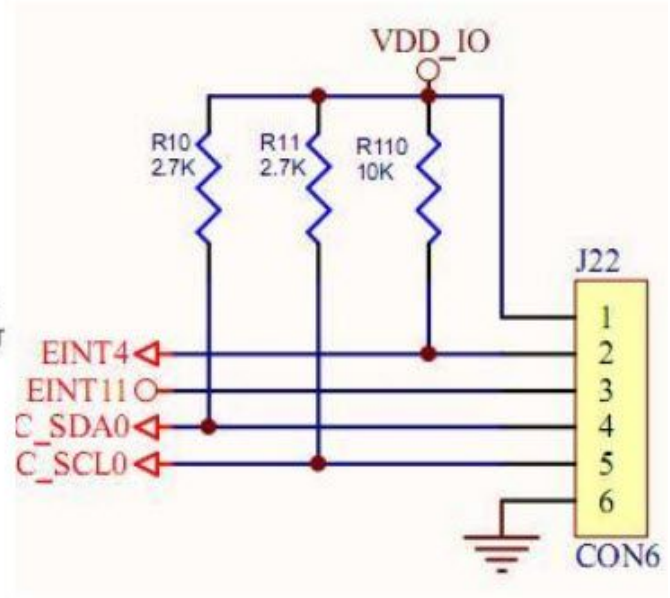
芯片介绍

- 应用领域
  - 智能手机
  - 移动上网设备
  - 便携式/平板电脑
  - GPS/数码相机/游戏机等

芯片介绍

➢ 连接电路图



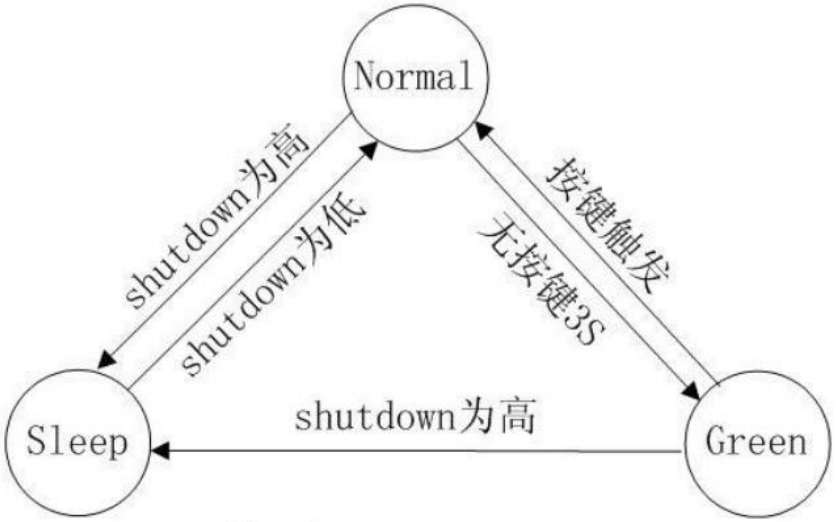


大纲

- 项目简介
- 芯片介绍
- 工作模式与通信时序
- 驱动框架

工作模式与通信时序

➢工作模式



a) Normal mode

Guitar在Normal mode时，屏幕扫描周期为16ms，输出坐标的更新频率为60Hz。若3s内未检测到新的触摸动作，芯片自动转入Green mode。

工作模式与通信时序

b) Green mode

在Green mode下，屏幕扫描周期为48ms，若检测到有触摸动作发生，自动

进入Normal mode。

c) Sleep mode

主CPU可通过对SHUTDOWN口状态的设置，使Guitar进入或退出Sleep mode。“1”表示进入Sleep mode ； “0”表示退出。当Guitar退出Sleep mode 时，直接进入

Normal mode。

工作模式与通信时序

- 触摸检测——脉冲方式呼叫

  为有效减轻主CPU负担，Guitar仅在输出信息有变化时，才会通知主CPU读取坐标信息。由INT 口输出脉冲信号。主CPU可通过对Guitar配置寄存器的设定，来选择适合自己的脉冲信号极性。

  我们可以将此管脚的中断设为中断信号监听管脚

工作模式与通信时序

➢ 睡眠模式

为保证Guitar内部电容检测电路的可靠性，Guitar专门规划了一个独立的SHUTDOWN输入口，用来切换芯片的工作状态。

当显示屏熄灭时或在其他不需要操作触摸屏的状态下，可以通过将SHUTDOWN口置“1”，使Guitar进入Sleep mode以降低功耗。

当需要Guitar正常工作时，将SHUTDOWN口置“0”即可。

在退出Sleep mode时，Guitar从初始化地址开始工作，因此SHUTDOWN还有复位的功能。

工作模式与通信时序

I2C通讯时序

- Guitar提供标准的I2C通讯接口，由SCL和SDA 与主CPU进行通讯。 在系统中Guitar 始终作为从设备，所有通讯都是由主CPU发起，其最高通讯速度为250K bps。
- Guitar 的从设备地址为0b1010 101x. 主CPU寻址Guitar时，同时还要发送读写控制位，读写控制位是附在从设备地址后，“0”表示写操作，“1”表示读操作，从而与设备地址组成一个字节。即：0xAA——对Guitar进行写操作；0xAB——对Guitar进行读操作。

工作模式与通信时序

a） 数据传输

通讯总是由主CPU发起，有效的起始信号为：在SCL保持为“1”时，SDA 上发生由“1”到“0”的跳变。地址信息或数据流均在起始信号之后传输。

所有连接在I2C总线上的从设备，都要检测总线上起始信号之后所发送的8位地址信息，并做出正确反应。在收到与自己相匹配的地址信息时，Guitar在第9个时钟周期，将SDA改为输出口，并置“0”，作为应答信号。若收到不与自己匹配的地址信息，即非0XAA或0XAB，Guitar将保持闲置状态。

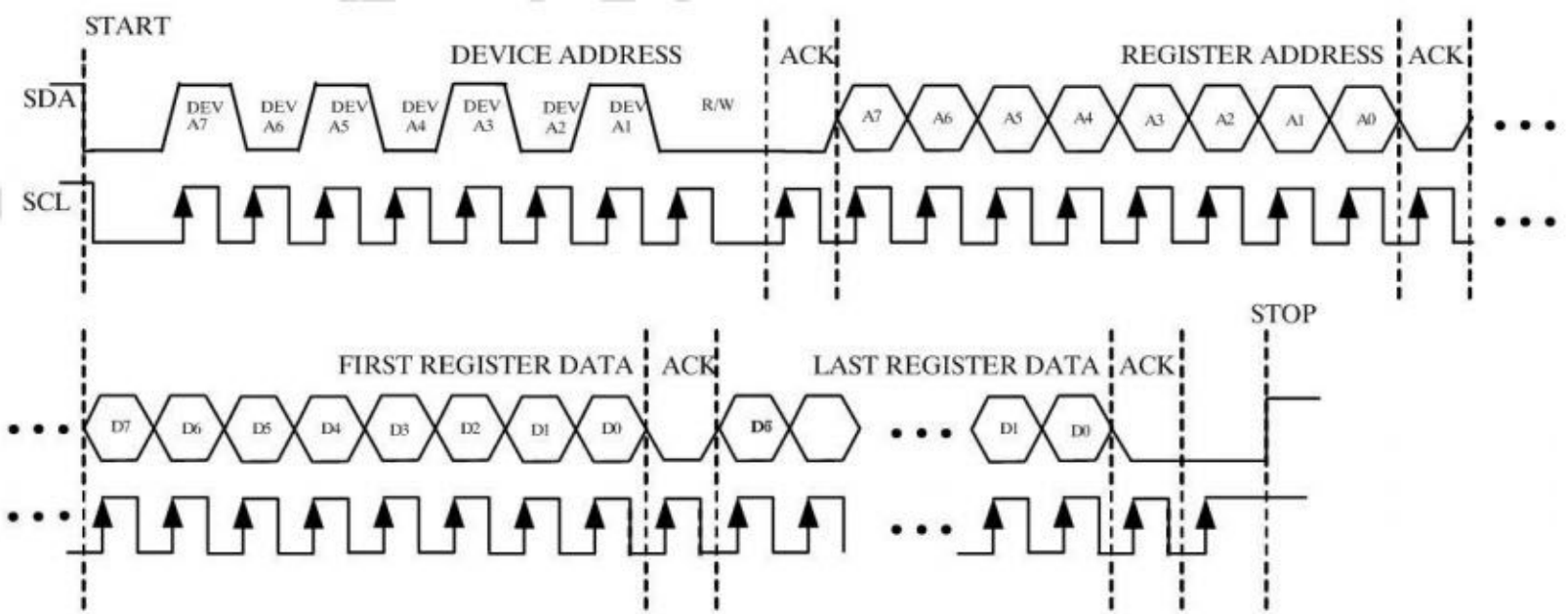
工作模式与通信时序

- SDA口上的数据按9个时钟周期串行发送9位数据：8位有效数据+1位接收方发送的应答信号ACK或非应答信号NACK。数据传输在SCL为“1”时有效。

  当通讯完成时，由主CPU发送停止信号。停止信号是当SCL为“1”时，SDA状态由“0”到“1”的跳变。接收到一个有效的停止信号后，Guitar内部I2C的指针变为0x00。

工作模式与通信时序

b) 对Guitar写操作



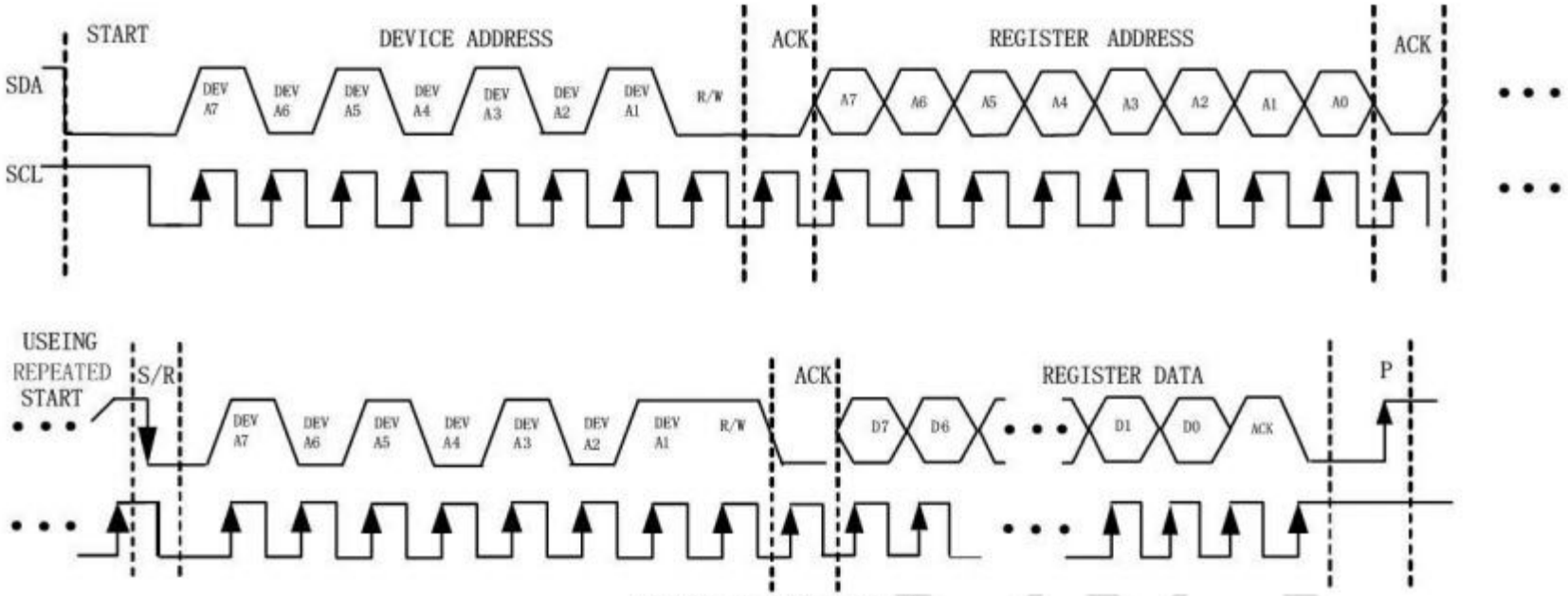
写操作流程图

工作模式与通信时序

- 上图为主CPU对Guitar进行的写操作流程图。首先主CPU产生一个起始信号，然后发送地址信息及读写位信息“0”表示写操作:0XAA。
- 在收到应答后，主CPU发送寄存器的8位地址，随后是8位要写入到寄存器的数据内容。Guitar 寄存器的地址指针会在写操作后自动加1，所以当主CPU需要对地址的寄存器进行连续写操作时，可以在一次写操作中连续写入。写操作完成，主CPU发送停止信号结束当前写操作。

工作模式与通信时序

c） 对Guitar读操作



读操作流程图

工作模式与通信时序

- 上图为主CPU对Guitar进行的读操作流程图。首先主CPU产生一个起始信号，然后发送设备地址信息及读写位信息“0”表示写操作：0XAA。
- 在收到应答后，主CPU发送首寄存器的8位地址信息，设置要读取的寄存器地址。在收到应答后，主CPU重新发送一次起始信号（不要发送停止信号，否则寄存器地址变回为0X00），发送读操作：0XAB。收到应答后，主CPU开始读取数据。

工作模式与通信时序

- Guitar同样支持连续的读操作，默认为连续读取数据。主CPU在每收到一个Byte数据后需发送一个应答信号表示成功接收。在接收到所需的最后一个Byte数据后，主CPU发送“非应答信号NACK”，然后再发送停止信号结束通讯。

工作模式与通信时序

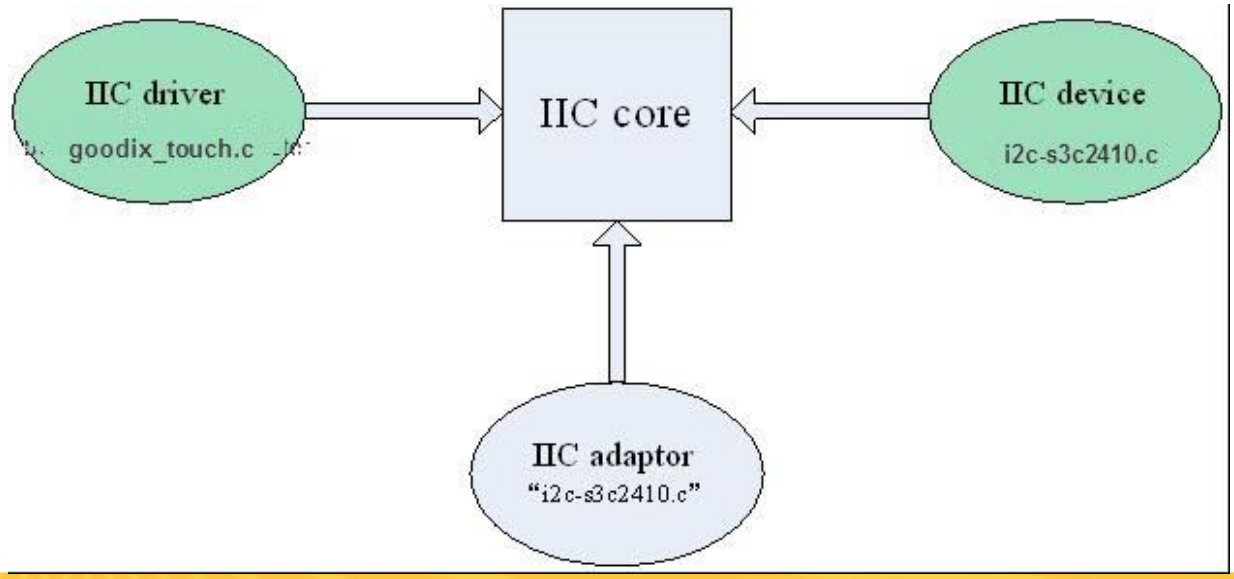
- 注意：
- 0、构建i2c_client结构体
- 1、填充 i2c_driver，进入probe
- 2、申请并填充input_dev
- 3、申请中断
- 4、初始化goodix芯片(写54个命令字)
- 5、触发中断，进入irq服务程序，queue_work调度下半部
- 5、在work中，读取35个字节的数据(注意芯片手册的读写过程)
- 6、调用给好的接口进行数据处理，得到存放x/y坐标的二维数组pos
- 7、input_report_abs上报坐标， input_sync同步操作
- ps:初始化时个别命令可能失败，多次写入即可解决

大纲

- 项目简介
- 芯片介绍
- 工作模式与通信时序
- 驱动框架

驱动框架

- 该项目采用了linux内核的i2c子系统，触摸屏芯片连接在s5pv210的i2c控制器0上，芯片地址为0b1010 101x。
- 下图为触摸屏驱动框架



驱动框架

- IIC device部分完成i2c设备的注册，主要是在platform总线端，先实现platform_driver（i2c-s3c2410.c）与 platform_device（控制器相关信息）的匹配，匹配成功跳到platform_driver的probe函数里面，完成i2c_adapter（控制器）的初始化与注册。然后从全局的__i2c_board_list上找到触摸屏的board info并创建i2c_client注册在i2c_bus上，等待i2c_driver去匹配
- IIC driver部分完成触摸屏芯片的驱动任务，一系列复杂的操作都由该部分完成。

数据传输

- 适配器支持两种传输方法
  - master_xfer（控制器传输）
    - master_xfer为子系统利用当前平台的i2c控制器完成i2c通信方法
    - 利用master传输只需要填充i2c_msg，然后通过i2c_transfer函数完成读写操作：

```
int i2c_transfer(struct i2c_adapter *adap, struct
    i2c_msg *msgs, int num)
```

数据传输

- master_xfer---写

  注：利用i2c_transfer发送数据只需填充一个i2c_msg即可

```
msg[0].addr = addr;//器件地址
msg[0].flags = !I2C_M_RD; //读写标记
msg[0].len = 1;//下面的buf大小
msg[0].buf = data;//一般有两个字节组成，一个目标单元，另一个是要写的数据
```

数据传输

master_xfer---读

注：利用i2c_transfer读取数据需填充两个个i2c_msg

```
msg[0].addr = chip_addr;//器件地址
msg[0].flags = !I2C_M_RD; //写标记
msg[0].len = 1;//下面的buf大小
msg[0].buf = &addr;//器件单元地址
msg[1].addr = chip_addr;//器件地址
msg[1].flags = I2C_M_RD;//读标记
msg[1].len = count;//下面的buf大小
msg[1].buf = buf;//读取到的数据
```

数据传输

- smbus_xfer（模拟时序传输）
  - smbus_xfer为子系统模拟的i2c通信方法,可以调用i2c_smbus_write_byte_data/i2c_smbus_read_byte_data直接发送和接收数据：

```
s32 i2c_smbus_read_byte_data(struct i2c_client *client, u8 command);
s32 i2c_smbus_write_byte_data(struct i2c_client *client, u8 command, u8 value);
```

凌阳教育官方微信：Sunplusedu

Tel: 400-705-9680 , BBS: www.51develop.net , QQ群：241275518