|  |
| --- |
| Elvonzion.WX |
| 设备特性与应用 |
|  |
|  |
| **admin** |
| **2012/8/12** |

|  |
| --- |
|  |

**目 录**

[1 基础设备 1](#_Toc390159389)

[1.1 电平转换 1](#_Toc390159390)

[1.1.1 逻辑电平 1](#_Toc390159391)

[1.1.2 高速逻辑 2](#_Toc390159392)

[1.1.3 三极管 5](#_Toc390159393)

[1.1.4 IO 5](#_Toc390159394)

[1.2 ADC 6](#_Toc390159395)

[1.2.1 SAR-ADC 6](#_Toc390159396)

[1.2.2 问题收集 6](#_Toc390159397)

[1.3 电路规范 7](#_Toc390159398)

[1.3.1 接地 7](#_Toc390159399)

[1.3.2 电压标准 10](#_Toc390159400)

[1.3.3 ESD测试 11](#_Toc390159401)

[1.3.4 电路检测 11](#_Toc390159402)

[1.4 编码转换 14](#_Toc390159403)

[1.4.1 归零编码 14](#_Toc390159404)

[1.4.2 曼彻斯特编码 16](#_Toc390159405)

[2 处理器核心 17](#_Toc390159406)

[2.1 ARM 架构 17](#_Toc390159407)

[2.1.1 GIC 17](#_Toc390159408)

[2.1.2 AHB 17](#_Toc390159409)

[2.2 ARM指令 18](#_Toc390159410)

[2.2.1 程序状态状态寄存器(PSR) 18](#_Toc390159411)

[2.2.2 协处理器15 (CP15) 20](#_Toc390159412)

[2.2.3 向量处理模块 23](#_Toc390159413)

[2.2.4 协处理器指令 24](#_Toc390159414)

[2.2.5 向量处理指令 25](#_Toc390159415)

[2.2.6 多寄存器寻址 25](#_Toc390159416)

[2.2.7 汇编指令的特殊符号 25](#_Toc390159417)

[2.2.8 rsb指令 26](#_Toc390159418)

[2.2.9 svc与swi指令 26](#_Toc390159419)

[2.2.10 ldr 和 adr指令 27](#_Toc390159420)

[2.2.11 条件判断语句的实现 27](#_Toc390159421)

[2.3 DSP 29](#_Toc390159422)

[2.4 FPGA 29](#_Toc390159423)

[2.5 GPU 30](#_Toc390159424)

[2.5.1 显卡 30](#_Toc390159425)

[2.5.2 嵌入式 30](#_Toc390159426)

[3 存储设备 31](#_Toc390159427)

[3.1 DDR 32](#_Toc390159428)

[3.2 寄存器 33](#_Toc390159429)

[3.3 ROM 34](#_Toc390159430)

[3.4 MMU 34](#_Toc390159431)

[3.5 NandFlash 35](#_Toc390159432)

[3.5.1 存储特点 35](#_Toc390159433)

[3.5.2 常用命令 35](#_Toc390159434)

[3.5.3 存储空间编址 36](#_Toc390159435)

[3.5.4 ECC原理 38](#_Toc390159436)

[3.5.5 ECC的纠错能力 38](#_Toc390159437)

[3.5.6 ECC算法特征 39](#_Toc390159438)

[3.6 SD Host Controller 42](#_Toc390159439)

[3.6.1 Architecture 42](#_Toc390159440)

[3.6.2 DMA Transfer 45](#_Toc390159441)

[3.6.3 Buffer Control 48](#_Toc390159442)

[3.6.4 Interfaces Sync 50](#_Toc390159443)

[3.6.5 Buffer sync 51](#_Toc390159444)

[3.6.6 Interrupt Control 53](#_Toc390159445)

[3.6.7 Standard CMD 55](#_Toc390159446)

[3.6.8 Card Initialization and Identification 57](#_Toc390159447)

[3.7 eMMC 59](#_Toc390159448)

[3.7.1 设备特性 59](#_Toc390159449)

[3.7.2 IO规格 61](#_Toc390159450)

[3.7.3 Combo Socket 61](#_Toc390159451)

[4 图像设备 62](#_Toc390159452)

[4.1 图像标准 62](#_Toc390159453)

[4.1.1 YUV Color Space 62](#_Toc390159454)

[4.1.2 Resolution 63](#_Toc390159455)

[4.1.3 Definition 64](#_Toc390159456)

[4.2 图像输入 65](#_Toc390159457)

[4.2.1 Image Sensor属性 65](#_Toc390159458)

[4.3 图像输出 66](#_Toc390159459)

[4.3.1 显示设备 66](#_Toc390159460)

[4.4 MALI 68](#_Toc390159461)

[5 设备总线 69](#_Toc390159462)

[5.1 JTAG 69](#_Toc390159463)

[5.1.1 接口规范 69](#_Toc390159464)

[5.1.2 接口信号 73](#_Toc390159465)

[5.1.3 TAP原理 75](#_Toc390159466)

[5.1.4 TAP状态机 75](#_Toc390159467)

[5.1.5 接口指令 80](#_Toc390159468)

[5.2 IPC 82](#_Toc390159469)

[5.3 USB 83](#_Toc390159470)

[5.3.1 Connector 83](#_Toc390159471)

[5.3.2 Host Controller Interface 85](#_Toc390159472)

[5.4 IIC 87](#_Toc390159473)

[5.4.1 传输协议 87](#_Toc390159474)

[5.5 SPI 88](#_Toc390159475)

[5.6 UART 89](#_Toc390159476)

[5.6.1 RS232 89](#_Toc390159477)

[5.7 CAN 90](#_Toc390159478)

[6 有线网络 91](#_Toc390159479)

[6.1 网络连接 91](#_Toc390159480)

[6.1.1 以太网介绍 91](#_Toc390159481)

[6.1.2 双绞线 92](#_Toc390159482)

[6.1.3 RJ45接口 94](#_Toc390159483)

[6.1.4 光纤接口 97](#_Toc390159484)

[6.2 Ethernet PHY 98](#_Toc390159485)

[6.2.1 MII接口 99](#_Toc390159486)

[6.2.2 RTL8201介绍 103](#_Toc390159487)

[6.3 Ethernet MAC 105](#_Toc390159488)

[6.3.1 DWC 105](#_Toc390159489)

[6.3.2 GMAC-UNIV 105](#_Toc390159490)

[6.4 TCP/IP协议 107](#_Toc390159491)

[6.4.1 协议实现 107](#_Toc390159492)

[7 无线网络 108](#_Toc390159493)

[7.1 WIFI 108](#_Toc390159494)

[7.2 BlueTooth 109](#_Toc390159495)

[8 按键设备 110](#_Toc390159496)

[8.1 电容触摸屏 110](#_Toc390159497)

[8.1.1 感应原理 110](#_Toc390159498)

[8.1.2 感应材料 110](#_Toc390159499)

[8.1.3 布局结构 111](#_Toc390159500)

[8.1.4 采样方式 112](#_Toc390159501)

[8.1.5 鬼点消除 113](#_Toc390159502)

[9 感应设备 115](#_Toc390159503)

[9.1 技术发展 115](#_Toc390159504)

[9.2 惯性感应 116](#_Toc390159505)

[9.2.1 加速度计 116](#_Toc390159506)

[9.2.2 陀螺仪 116](#_Toc390159507)

[9.2.3 磁力计 116](#_Toc390159508)

[9.3 环境感应 117](#_Toc390159509)

[9.3.1 压力感测器 117](#_Toc390159510)

[9.3.2 湿度感测器 117](#_Toc390159511)

[9.3.3 温度感测器 117](#_Toc390159512)

[9.3.4 气体化学感测器 117](#_Toc390159513)

[9.4 其他 118](#_Toc390159514)

[9.4.1 红外线(IR)成像 118](#_Toc390159515)

# 基础设备

## 电平转换

### 逻辑电平

1. TTL

TTL集成电路的全名是晶体管-晶体管逻辑集成电路（Transistor-Transistor Logic)，

1. 电路特性

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 电路类型 | Vcc | VOH | VOL | VIH | VIL |
| TTL | 5V | >=2.4V | <=0.5V | >=2V | <=0.8V |
| LVTTL | 3.3V | >=2.4V | <=0.4V | >=2V | <=0.8V |
| 2.5V | >=2.0V | <=0.2V | >=1.7V | <=0.7V |

VIH:输入高电平，VOH:输出高电平，VIL:输入低电平，VOL:输出低电平。

因为TTL的2.4V与5V之间还有很大空闲，对改善噪声容限并没什么好处，又会白白增大系统功耗，还会影响速度。所以后来就把一部分“砍”掉变为LVTTL（Low Voltage TTL）。

1. 设计要求

TTL电路的电源VDD供电只允许在+5V±10%范围内；

TTL电平一般过冲都会比较严重，可能在始端串22欧或33欧电阻；

TTL电平输入脚悬空时内部认为是高电平，下拉需使用1k以下电阻；

TTL输出不能驱动CMOS输入。

1. COMS

COMS集成电路是互补对称金属氧化物半导体（Compiementary symmetry metal oxide semicoductor）集成电路的英文缩写，COMS电路的供电电压VDD范围比较广在+5--+15V均能正常工作，电压波动允许±10%，当输出电压高于VDD-0.5V时为逻辑1，输出电压低于VSS+0.5V(VSS为数字地)为逻辑0。

1. 电路特性

1. 设计要求

CMOS结构内部寄生有可控硅结构，当输入或输入管脚高于VCC一定值(比如一些芯片是0.7V)时，电流足够大的话，可能引起闩锁效应，导致芯片的烧毁。

1. TTL与CMOS电平使用区别：
2. 电平的上限和下限定义不一样，CMOS具有更大的抗噪区域。
3. 电流驱动能力不一样，ttl一般提供25毫安的驱动能力，而CMOS一般在10毫安左右。
4. 需要的电流输入大小也不一样，一般ttl需要2.5毫安左右，CMOS几乎不需要电流输入。
5. 电路互连

很多器件都是兼容ttl和CMOS的，datasheet会有说明。如果不考虑速度和性能，一般器件可以互换。

有时候负载效应可能引起电路工作不正常，因为有些TTL电路需要下一级的输入阻抗作为负载才能正常工作。因此，CMOS电路与TTL电路就有一个电平转换的问题，使两者电平域值能匹配。

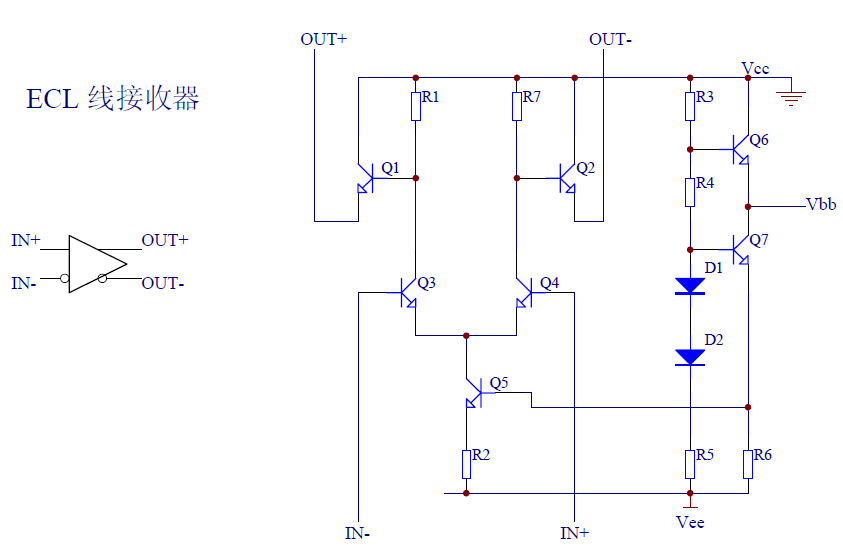
TTL（或CMOS）电平之间的连接不需要做电路匹配，因此两个TTL（或CMOS）电平之间可以直接互联，TTL（或CMOS）电平与PECL电平之间的转换是通过T-P、P-T转换芯片来实现的，这类型的芯片例如Micrel的SY100ELT系列等。

### 高速逻辑

高速IC芯片与高速光模块间互连通常有四种接口：PECL (Positive Emitter-Coupled Logic)、LVPECL (Low-Voltage Positive Emitter-Coupled Logic)、LVDS (Low-Voltage Differential Signals)、CML (Current ModeLogic)。为解决不同接口标准芯片与高速光模块间的互连这一问题，我们首先需要了解每一种接口标准的输入输出电路结构，由此可以知道如何进行直流偏置和终端匹配。

TTL电路已经无法适应越来越高的工作速率，最先由Motorola公司提出ECL标准ECL电路（即发射极耦合逻辑电路）是一种非饱和型的数字逻辑电路，电路内晶体管工作在线性区或截止区，速度不受少数载流子的存储时间的限制，所以它是现有各种逻辑电路中速度最快的一种ECL电路,能满足高达10Gbps工作速率.

1. ECL
2. 电路实例



1. 电路优点

􀁺速度快

晶体管工作时不进入饱和状态，只工作在线性区和截止区，没有少数载流子的存储现象，开关时间大为缩短；集电结电容大大减小，RC时间常数也相应减小，电路的传输延迟时间就很短；电路的逻辑电平摆幅小（单端小于850mV），在动态转换过程中各个结上的电压变化对结电容（包括寄生电容）的充放电时间很短。

􀁺 逻辑功能强

􀁺 扇出能力强

输入阻抗高（>10KΩ），输出阻抗低（约3～7 Ω ），因此它的直流扇出负载数可以高达92。

􀁺 噪声低

差分电路两臂交替工作，电源总电流基本恒定，电流尖峰很小；电压摆幅小，并且采用差分对或传输线传输信号，对外串扰和受外界干扰都减小了。

􀁺 便于数据传输

1. 电路缺点

ECL电路的直流功耗大，实际上，工作速率的提高是以牺牲功耗为代价换取来的。

1. PECL

PECL由ECL标准发展而来，在PECL电路中省去了负电源，较ECL电路更便于使用。PECL信号的摆幅相对ECL要小，这使得该逻辑更适合于高速

数据的串行或并行连接，由于ECL电路是采用-5.2V电源供电，Vcc是接地的，这样做虽有一些优点，但负电源还是很麻烦。

􀁺 PECL由ECL标准发展而来，采用+5V供电，可以和系统内其他电路共用一个正电源供电。PECL信号的摆幅相对ECL要略小些。

1. CML

CML电路即“电流模式逻辑”电路。CML是所有高速数据接口形式中最简单的一种

􀁺 CML电路与ECL电路类似，电路的核心部分同样是一个差分放大器，它的共发射极也是通过恒流源到地。

􀁺 CML电路提供的电压摆幅较小(典型值：单端400mV)，芯片内部已集成了输入、输出的匹配电阻（50Ω），所以，在它的输入、输出端不需要外加端接和偏置电阻。

􀁺 CML电路的功耗比ECL电路小。

1. LVDS

LVDS（low-voltage differential signaling）即低电压差分信号电路

它的优点是：

信号摆幅更小，使它具有更好的噪声性能，与ECL、CML电路相比功耗最低；

因为信号的摆幅小，使LVDS电路可在2.5V的低电源电压下工作；

允许输入共模电压范围宽，从0.2V到2.2V。

### 三极管

1. 放大原理

输出信号不是从基极来的，而是基极信号控制下的电源信号变化过来的。要使集电极上的电压随着基极信号变化而变化，就要切断电源电压与集电极之间的联系，最好的办法就是加个电阻（电阻可以分压），如果不加，则集电极上的电压会一直是电源电压，无法变化了。

上拉电阻应该放在OD引脚附近。如果两端(主设备及从设备)都为OD输出，则放在中间位置。

1. 上下拉

### IO

1. PIN description

DP/DG => Digital Power/Ground

AP/AG => Analog Power/Ground

OD => Open Drain Output

AI/AO => Analog Input/Output

AIO => Analog Input & Output

I/O => Input/Output

IO => Bi-directional

## ADC

### SAR-ADC

### 问题收集

1. 运放接线错误造成电压采样失真

一个交流采样系统, 交流信号由运放组成信号调理电路,并有运放做加法运算以后,进入stm32的ADC管脚.

结果测试的时候,发现采样很不准确,插交流有效值算法也没问题.

仔细检测,发现信号越小,测量越准!

这很违反尝试,用示波器检测adc输入管脚发现交流正弦信号底部失真,

仔细检查,发现原因是由于运放是正负电源供电,运放的输入端加了二极管保护,分别对+12和对地保护,

结果交流信号小于0的时候,底部就被削底了,发生了削底失真.

保护二极管正确的接法,应该是对+12V,-12保护,而不是对地保护.

.

## 电路规范

### 接地

除了正确进行接地设计、安装,还要正确进行各种不同信号 的接地处理。控制系统中，大致有以下几种地线：

(1)数字地：也叫逻辑地，是各种开关量(数字量)信号的零电位。

(2)模拟地：是各种模拟量信号的零电位。

(3)信号地：通常为传感器的地。

(4)交流地：交流供电电源的地线，这种地通常是产生噪声的地。

(5)直流地：直流供电电源的地。

(6)屏蔽地：也叫机壳地，为防止静电感应和磁场感应而设。

以上这些地线处理是系统设计、安装、调试中的一个重要问题。下面就接地问题提出一些看法：

(1)控制系统宜采用一点接地。一般情况下,高频电路应就近多点接地，低频电路应一点接地。在低频电路中，布线和元件间的电感并不是什么大问题，然而接地形成的环路的干扰影响很大，因此，常以一点作为接地点;但一点接地不适用于高频，因为高频时，地线上具有电感因而增加了地线阻抗，同时各地线之间又产生电感耦合。一般来说，频率在1MHz以下,可用一点接地;高于10MHz时，采用多点接地;在1～10MHz之间可用一点接地，也可用多点接地。

(2)交流地与信号地不能共用。由于在一段电源地线的两点间会有数mV甚至几V电压，对低电平信号电路来说，这是一个非常重要的干扰，因此必须加以隔离和防止。

(3)浮地与接地的比较。全机浮空即系统各个部分与大地浮置起来，这种方法简单，但整个系统与大地绝缘电阻不能小于50MΩ。这种方法具有一定的抗干扰能力，但一旦绝缘下降就会带来干扰。还有一种方法，就是将机壳接地，其余部分浮空。这种方法抗干扰能力强，安全可靠，但实现起来比较复杂。

(4)模拟地。模拟地的接法十分重要。为了提高抗共模干扰能力，对于模拟信号可采用屏蔽浮技术。对于具体模拟量信号的接地处理要严格按照操作手册上的要求设计。

(5)屏蔽地。在控制系统中为了减少信号中电容耦合噪声、准确检测和控制，对信号采用屏蔽措施是十分必要的。根据屏蔽目的不同，屏蔽地的接法也不一样。电场屏蔽解决分布电容问题，一般接大地;电磁场屏蔽主要避免雷达、电台等高频电磁场辐射干扰。利用低阻金属材料高导流而制成，可接大地。磁场屏蔽用以防磁铁、电机、变压器、线圈等磁感应，其屏蔽方法是用高导磁材料使磁路闭合，一般接大地为好。当信号电路是一点接地时，低频电缆的屏蔽层也应一点接地。如果电缆的屏蔽层地点有一个以上时，将产生噪声电流，形成噪声干扰源。当一个电路有一个不接地的信号源与系统中接地的放大器相连时，输入端的屏蔽应接至放大器的公共端;相反，当接地的信号源与系统中不接地的放大器相连时，放大器的输入端也应接到信号源的公共端。

对于电气系统的接地，要按接地的要求和目的分类，不能将不同类接地简单地、任意地连接在一起，而是要分成若干独立的接地子系统，每个子系统都有其共同的接地点或接地干线，最后才连接在一起，实行总接地。

Q1：为什么要接地?

Answer：接地技术的引入最初是为了防止电力或电子等设备遭雷击而采取的保护性措施，目的是把雷电产生的雷击电流通过避雷针引入到大地，从而起到保护建筑物的作用。同时，接地也是保护人身安全的一种有效手段，当某种原因引起的相线(如电线绝缘不良，线路老化等)和设备外壳碰触时，设备的外壳就会有危险 电压产生，由此生成的故障电流就会流经PE线到大地，从而起到保护作用。随着电子通信和其它数字领域的发展，在接地系统中只考虑防雷和安全已远远不能满足 要求了。比如在通信系统中，大量设备之间信号的互连要求各设备都要有一个基准‘地’作为信号的参考地。而且随着电子设备的复杂化，信号频率越来越高，因此，在接地设计中，信号之间的互扰等电磁兼容问题必须给予特别关注，否则，接地不当就会严重影响系统运行的可靠性和稳定性。最近，高速信号的信号回流技术 中也引入了“地”的概念。

Q2：接地的定义

Answer:在现代接地概念中、对于线路工程师来说，该术语的含义通常是‘线路电压的参考点’;对于系统设计师来说，它常常是机柜或机架;对电气工程师来说，它是绿色安全地线或接到大地的意思。一个比较通用的定义是“接地是电流返回其源的低阻抗通道”。注意要求是”低阻抗”和“通路”。

Q3：常见的接地符号

Answer:PE,PGND,FG-保护地或机壳;BGND或DC-RETURN-直流-48V(+24V)电源(电池)回流;GND-工作地;DGND-数字地;AGND-模拟地;LGND-防雷保护地 .GND在电路里常被定为电压参考基点。从电气意义上说，GND分为电源地和信号地。PG是Power Ground(电源地)的缩写。另一个是 Signal Ground(信号地)。实际上它们可能是连在一起的(不一定是混在一起哦!)。两个名称，主要是便于对电路进行分析。进一步说，还有因电路形式不同而必须区分的两种“地”：数字地，模拟地.数字地和模拟地都有信号地、电源地两种情况。数字地和模拟地之间，某些电路可以直接连接，有些电路要用电抗器连接，有些电路不可连接。

Q4：合适的接地方式

Answer:接地有多种方式，有单点接地，多点接地以及混合类型的接地。而单点接地又分为串联单点接地和并联单点接地。一般来说，单点接地用于简单电路，不同功能模块之间接地区分，以及低频(f10MHz)电路时就要采用多点接地了或者多层板(完整的地平面层)。

Q5：信号回流和跨分割的介绍

Answer：对于一个电子信号来说，它需要寻找一条最低阻抗的电流回流到地的途径，所以如何处理这个信号回流就变得非常的关键。

第一，根据公式可以知道，辐射强度是和回路面积成正比的，就是说回流需要走的路径越长，形成的环越大，它对外辐射的干扰也越大，所以，PCB布板的时候要 尽可能减小电源回路和信号回路面积。

第二，对于一个高速信号来说，提供有好的信号回流可以保证它的信号质量，这是因为PCB上传输线的特性阻抗一般是以地层(或电源层)为参考来计算的，如果 高速线附近有连续的地平面，这样这条线的阻抗就能保持连续，如果有段线附近没有了地参考，这样阻抗就会发生变化，不连续的阻抗从而会影响到信号的完整性。所以，布线的时候要把高速线分配到靠近地平面的层，或者高速线旁边并行走一两条地线，起到屏蔽和就近提供回流的功能。

第三，为什么说布线的时候尽量不要跨电源分割，这也是因为信号跨越了不同电源层后，它的回流途径就会很长了，容易受到干扰。当然，不是严格要求不能跨越电源分割，对于低速的信号是可以的，因为产生的干扰相比信号可以不予关心。对于高速信号就要认真检查，尽量不要跨越，可以通过调整电源部分的走线。(这是针对多层板多个电源供应情况说的)

Q6：为什么要将模拟地和数字地分开，如何分开?

Answer：模拟信号和数字信号都要回流到地，因为数字信号变化速度快，从而在数字地上引起的噪声就会很大，而模拟信号是需要一个干净的地参考工作的。如果模拟地和数字地混在一起，噪声就会影响到模拟信号。 一般来说，模拟地和数字地要分开处理，然后通过细的走线连在一起，或者单点接在一起。总的思想是尽量阻隔数字地上的噪声窜到模拟地上。当然这也不是非常严格的要求模拟地和数字地必须分开，如果模拟部分附近的数字地还是很干净的话可以合在一起。

Q7：单板上的信号如何接地?

Answer：对于一般器件来说，就近接地是最好的，采用了拥有完整地平面的多层板设计后，对于一般信号的接地就非常容易了，基本原则是保证走线的连续性，减少过孔数量;靠近地平面或者电源平面，等等。

Q8：单板的接口器件如何接地?

Answer：有些单板会有对外的输入输出接口，比如串口连接器，网口RJ45连接器等等，如果对它们的接地设计得不好也会影响到正常工作，例如网口互连有误码，丢包等，并且会成为对外的电磁干扰源，把板内的噪声向外发送。一般来说会单独分割出一块独立的接口地，与信号地的连接采用细的走线连接，可以串上 0欧姆或者小阻值的电阻。细的走线可以用来阻隔信号地上噪音过到接口地上来。同样的，对接口地和接口电源的滤波也要认真考虑。

Q9：带屏蔽层的电缆线的屏蔽层如何接地?

Answer：屏蔽电缆的屏蔽层都要接到单板的接口地上而不是信号地上，这是因为信号地上有各种的噪声，如果屏蔽层接到了信号地上，噪声电压会驱动共模电流沿屏蔽层向外干扰，所以设计不好的电缆线一般都是电磁干扰的最大噪声输出源。当然前提是接口地也要非常的干净混合电路里面。

### 电压标准

VCC表示模拟信号电源,GND表示模拟信号地,VDD表示数字信号电源,VSS表示数字电源地 。VCC主要表示Bipolar电路的电源，C表示Collector集电极,电源一般接在NPN的集电极(或PNP的发射极)，集成电路刚出现时只有NPN管，后来才有集成进去的PNP管。VDD/VSS一般表示MOS电路的电源和“地”，D/S分别表示MOS管的Drain(漏)/Source(源)。

一、解释

VCC：C=circuit 表示电路的意思, 即接入电路的电压;

VDD：D=device表示器件的意思, 即器件内部的工作电压;

VSS：S=series表示公共连接的意思，通常指电路公共接地端电压。

二、说明

1、对于数字电路来说，VCC是电路的供电电压,VDD是芯片的工作电压(通常Vcc>Vdd)，VSS是接地点。

2、有些IC既有VDD引脚又有VCC引脚，说明这种器件自身带有电压转换功能。

3、在场效应管(或COMS器件)中，VDD为漏极，VSS为源极，VDD和VSS指的是元件引脚，而不表示供电电压。

VDD:电源电压(单极器件);电源电压(4000系列数字电 路);漏极电压(场效应管)

VCC：电源电压(双极器件);电源电压(74系列数字电路);声控载波(Voice ControlledCarrier)

VSS:地或电源负极

VEE：负电压供电;场效应管的源极(S)

VPP：编程/擦除电压。

详解：

在电子电路中，VCC是电路的供电电压, VDD是芯片的工作电压：

VCC：C=circuit表示电路的意思, 即接入电路的电压， D=device 表示器件的意思, 即器件内部的工作电压，在普通的电子电路中，一般Vcc>Vdd !

VSS：S=series表示公共连接的意思，也就是负极。

有些IC 同时有VCC和VDD， 这种器件带有电压转换功能。

在“场效应”即COMS元件中，VDD乃CMOS的漏极引脚，VSS乃CMOS的源极引脚， 这是元件引脚符号，它没有“VCC”的名称，你的问题包含3个符号，VCC / VDD /VSS， 这显然是电路符号

### ESD测试

静电防护测试。

1. 主要测试项目

表面电阻或者体积电阻，接地电阻。

静电压和摩擦电压。

静电衰减时间。

静电荷静电放电电流或者波形。

### 电路检测

**1**. 检查所有的芯片封装图引脚是否有误

当然，我指的是自己画的芯片封装。我在项目中曾经把一个芯片的2个引脚画反了，导致最后制版出来后不得不跳线，这样就很难看了。

所以，检查与原理图前一定要从芯片的封装入手，坚决把错误的封装扼杀在摇篮中!

2. 使用protel的Tools->ERC电气规则检查，根据其生成的文件来排错

这个指的是protel99的ERC电气规则检查，DXP应该也会有相应的菜单可以完成这样一个检查。很有用，它可以帮你查找出很多错误，根据它生成的错误文件，对照着错误文件检查一下你的原理图，你应该会惊叹：“我这么仔细地画图，竟然还会有这么多错误啊?”

3. 检测所有的网络节点net是否都连接正确(重点)

一般容易出现的错误有：

(1)   
本来两个net是应该相连接的，却不小心标得不一致，例如我曾经把主芯片的DDR时钟脚标的是DDR\_CLK，而把DDR芯片对应的时钟脚标成了DDRCLK，由于名字不一致，其实这两个脚是没有连接在一起的。

(2) 有的net只标出了一个，该net的另一端在什么地方却忘记标出。

(3) 同一个net标号有多个地方重复使用，导致它们全部连接到了一起。

4. 检测各个芯片功能引脚是否都连接正确，检测所有的芯片是否有遗漏引脚，不连接的划X

芯片的功能引脚一定不要连错，例如我使用的音频处理芯片有LCLK、BCLK、MCLK三个时钟引脚，与主芯片的三个音频时钟引脚一定要一一对应，连反一个就不能工作了。

是否有遗漏引脚其实很容易排查，仔细观察各个芯片，看是否有没有遗漏没有连接出去的引脚，查查datasheet，看看该引脚什么功能，如果系统中不需要，就使用X把该引脚X掉。

5. 检测所有的外接电容、电感、电阻的取值是否有根据，而不是随意取值

其实新手在画原理图时，时常不清楚某些外围电阻、电容怎么取值，这时千万不要随意取值，往往这些外围电路电阻、电容的取值在芯片的datasheet上都有说明的，有的datasheet上也给出了典型参考电路，或者一些电阻电容的计算公式，只要你足够细心，大部分电阻电容的取值你都是可以找到依据的。偶尔实在找不到依据的，可以在网上搜搜其他人的设计案例或者典型连接，参考一下。总之，不要随意设置这些取值。

6. 检查所有芯片供电端是否加了电容滤波

电源端的电容滤波的重要性就不用我多说了，其实做过硬件的人都应该知道。一般情况下，电路电源输入端会引进一些纹波，为了防止这些纹波对芯片的逻辑造成太大的影响，往往需要在芯片供电端旁边加上一些0.1uf之类的电容，起到一些滤波效果，检查电路原理图时，你可以仔细观察一下是否在必要地芯片电源端加上了这样的滤波电路呢?

7. 检测系统所有的接口电路

接口电路一般包括系统的输入和输出，需要检查输入是否有应有的保护等，输出是否有足够的驱动能力等

输入保护一般有：反冲电流保护、光耦隔离、过压保护等等。

输出驱动能力不足的需要加上一些上拉电阻提高驱动能力。

8. 检查各个芯片是否有上电、复位的先后顺序要求，若有要求，则需要设计相应的时延电路

例如我项目中使用的DM6467芯片，对供电电压的上电有先后顺序要求，必须先给1.2V电源端供电，然后给1.8V电源端供电，最后给3.3V电源端供电。因此，我们将电源芯片产生的三种电压通过一个时延芯片的处理(其实也可以使用一个三极管，利用钳位电压)，然后再依次输送到主芯片中。

9. 检查各个芯片的地，该接模拟地的接模拟地，该接数字地的是否接的数字地，数字地与模拟地之间是否隔开

一般处理模拟信号的芯片有：传感器芯片、模拟信号采集芯片、AD转换芯片、功放芯片、滤波芯片、载波芯片、DA转换芯片、模拟信号输出芯片等等，往往只有当系统中存在这些处理模拟信号的芯片或者电路时才会涉及模拟地和数字地。

一般芯片的接地脚该连接模拟地还是数字地在芯片手册中都有说明，按照datasheet上连接就可以了。

10. 观察各个模块是否有更优的解决方案(可选)

其实，刚刚设计原理图初稿时，往往没有想那么多，当整个系统成型后，你往往会发现其实很多地方是可以改进可以优化的。我们项目中的电源模块前前后后改版了4次，每过一段时间往往又发现了更好的解决方案，现在的电源方案又简洁又实用，效果也高很多，我想这就是不断改进不断优化的好处吧!

## 编码转换

### 归零编码

像 UART、I2C、SPI 、USB这样的串行总线，通信时连续的0、1 信号只通过一根数据线发送给接受者。但是因为发送者和接收者运行的频率不一样，信号的同步就是个问题，比如，接受者接收到了一个持续一段时间的低电平，无法得知这究竟是代表了 5 个 0 还是 1000 个 0。

一个解决办法，就是在传输数据信号的同时，附加一个时钟信号线，用来同步两端的传输，接受者在时钟信号的辅助下对数据信号采样，就可以正确解析出发送的数据了，比如 I2C 就是这样做的，SDA 来传输数据，SCL 来传输同步时钟：



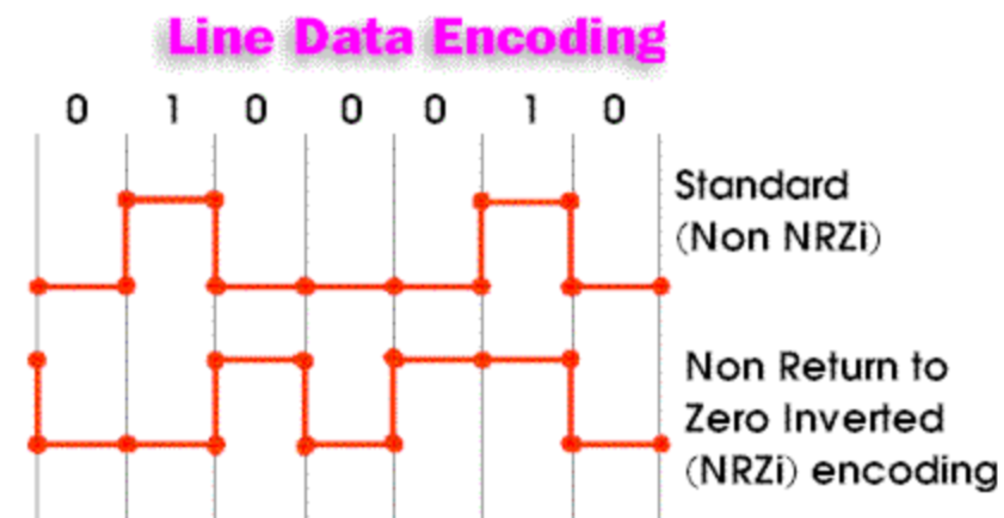
虽然这样解决了问题，但是却需要附加一根时钟信号线来传输时钟。有没有不需要附加的时钟信号，也能保持两端的同步呢？ 有的，这就是 RZ 编码（Return-to-zero Code），即归零编码。 在 RZ 编码中，正电平代表逻辑 1，负电平代表逻辑 0，并且，每传输完一位数据，信号返回到零电平，也就是说，信号线上会出现 3 种电平：正电平、负电平、零电平：



从图上就可以看出来，因为每位传输之后都要归零，所以接受者只要在信号归零后采样即可，这样就不在需要单独的时钟信号。实际上， RZ 编码就是相当于把时钟信号用归零编码在了数据之内。这样的信号也叫做自同步（self-clocking）信号。 这样虽然省了时钟数据线，但是还是有缺点的，因为在 RZ 编码中，大部分的数据带宽，都用来传输“归零”而浪费掉了。 那么，我们去掉这个归零步骤，NRZ 编码（Non-return-to-zero Code）就出现了，和 RZ 的区别就是 NRZ 是不需要归零的：



这样，浪费的带宽又回来了，不过又丧失宝贵的自同步特性了，貌似我们又回到了原点，其实这个问题也是可以解决的，不过待会儿再讲，先看看什么是 NRZI： NRZI 编码（Non-Return-to-Zero Inverted Code）和 NRZ 的区别就是 NRZI 用信号的翻转代表一个逻辑，信号保持不变代表另外一个逻辑。 USB 传输的编码就是 NRZI 格式，在 USB 中，电平翻转代表逻辑 0，电平不变代表逻辑1：



翻转的信号本身可以作为一种通知机制，而且可以看到，即使把 NRZI 的波形完全翻转，所代表的数据序列还是一样的，对于像 USB 这种通过差分线来传输的信号尤其方便~ 现在再回到那个同步问题： 的确，NRZ 和 NRZI 都没有自同步特性，但是可以用一些特殊的技巧解决。比如，先发送一个同步头，内容是 0101010 的方波，让接受者通过这个同步头计算出发送者的频率，然后再用这个频率来采样之后的数据信号，就可以了。 在 USB 中，每个 USB 数据包，最开始都有个同步域（SYNC），这个域固定为 0000 0001，这个域通过 NRZI 编码之后，就是一串方波（复习下前面：NRZI 遇 0 翻转遇 1 不变），接受者可以用这个 SYNC 域来同步之后的数据信号。

此外，因为在 USB 的 NRZI 编码下，逻辑 0 会造成电平翻转，所以接受者在接受数据的同时，根据接收到的翻转信号不断调整同步频率，保证数据传输正确。 但是，这样还是会有一个问题，就是虽然接受者可以主动和发送者的频率匹配，但是两者之间总会有误差。假如数据信号是 1000 个逻辑 1，经过 USB 的 NRZI 编码之后，就是很长一段没有变化的电平，在这种情况下，即使接受者的频率和发送者相差千分之一，就会造成把数据采样成 1001 个或者 999 个 1了。

USB 对这个问题的解决办法，就是强制插 0，也就是传说中的 bit-stuffing，如果要传输的数据中有 7 个连续的 1，发送前就会在第 6 个 1 后面强制插入一个 0，让发送的信号强制出现翻转，从而强制接受者进行频率调整。接受者只要删除 6 个连续 1 之后的 0，就可以恢复原始的数据了。

### 曼彻斯特编码

曼彻斯特（ Manchester ）码是一种双相码。用高电平到低电平的转换边表示 0 ，而用低电平到高高电平的转换边表示 1 。

# 处理器核心

## ARM 架构

### 官方文档

<http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dht0008a/CJAGCFAF.html>

### GIC

Generic Interrupt Controller

Distributor–detecting, disabling, prioritizing, directing interrupts to CPU(s).CPU interfaces–asserting an interrupt indication to the CPUSupport max to 1020 interrupts

 GIC interrupt handling sequence1. The GIC determines the interrupts that are enabled.2. For each pending interrupt, the GIC determines the targeted processor or processors.3. For each CPU interface, the Distributor forwards the highest priority pending interrupt that targets that interface.4. Each CPU interface determines whether to signal an interrupt request to its processor, and if required, does so.5. The processor acknowledges the interrupt, and the GIC returns the interrupt ID and updates the interrupt state.6. After processing the interrupt, the processor signals End of Interrupt (EOI) to the GIC.

### AHB

## ARM指令

### 程序状态状态寄存器(PSR)

PSR寄存器的结构如下：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| f | | | | | | s | x |  |  |  |  | c | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26:24 | 23:16 | 15:8 |  |  |  |  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| N | Z | C | V | Q |  |  |  |  |  |  |  | I | F | T | M4 | M3 | M2 | M1 | M0 |

1. CPSR：Current Program Status Register，只有一个。
2. SPSR：Saved Program Status Register，用来保存CPSR中的内容，在5种异常模式中各有一个。
3. 用户（user）模式和系统（System）模式不是异常模式，所以没有SPSR，当在这两种模式下访问SPSR，将产生不可预知的结果；
4. 用户模式下可读写CPSR的条件位PSR[31:27]，但控制位PSR[8:0]只能读取。
5. 当某个异常发生时，ARM会先将当前的CPSR保存到该异常对应模式专用的SPSR\_xxx寄存器，当准备退出中断前，需要由用户把SPSR\_xxx中保存的值恢复到CPSR。
6. 不同的中断可理解为CPU级别的多任务，其SPSR可以理解为多任务对应的程序控制块，CPSR可以理解为正在运行的任务控制块。
7. PSR的域分类

cpsr\_<xxx>和spsr\_<xxx>中的<xxx>指令只<xxx>中的位域有影响

c：PSR[7:0] (control field)

x：PSR[15:8] (extend field)

s：PSR[23:16] (status field)

f：PSR[31:24] (flag field)

1. PSR的条件标志位更新规则

条件标志用于表示ALU工作状态(NZCV)，以及DSP状态(Q)，具体意义如下：

1. 负数标志N(Nagative)

本位设置成当前指令运算结果的bit[31]位的值。当两个补码表示的有符号整数运算时，N=1表示运算的结果为负数；N=0表示结果为正数或0。

1. 零标志Z(Zero)

Z=1表示运算结果为0；Z=0表示运算结果不为0；对于CMP指令，Z=1表示两个数大小相等。

1. 进位标志C(Carry)

在加法指令中（包括比较指令CMP），当结果产生了进位，则C=1，表示无符号数运算发生上溢出，其他情况下C=0；

在减法指令中（包括比较指令CMP），当运算中发生借位，则C=0，表示无符号数运算发生下溢出，其他情况C=1；

对于包含移位操作的非加**/**减运算指令，C中包含最后一次被移出的位的数值；

对于其他非加**/**减运算指令，C位的值通常不受影响。

1. 溢出标志V(oVerFlow)

对于加**/**减运算指令，当操作数和运算结果为二进制的补码表示的带符号数时，V=1表示符号位溢出，通常其他指令不影响V标志位。

1. DSP溢出标志Q

在ARMv5的E系列处理器中，CPSR的bit[27]为Q标志位，用于指示增强的DSP指令是否发生了溢出。在ARMv5以前及非E系列的处理器中，Q标志位没有定义。

1. PSR的控制位功能
2. IRQ中断使能位I

I=1，禁止IRQ中断；I=0，使能IRQ中断；

1. FIQ中断使能位F

F=1，禁止FIQ中断；F=0，使能FIQ中断；

1. Thumb控制位T

ARMv3以及更低的版本和ARMv4的非T系列：

没有ARM状态和Thumb状态切换，T控制位应为0；

ARMv4以及更高版本的T系列：

T=0表示执行ARM指令，T=1表示执行Thumb指令；

ARMV5以及更高版本的非T系列：

T=0表示执行ARM指令，T=1表示强制下一条执行的指产生未定义指令中断。

1. 系统模式标志位M[4:0]

该标志表示系统当前CPU工作模式，即CPU当前使用的寄存器组资源。进入中断时CPSR的M[4:0]由CPU自动切换到相应模式，此时当用户修改CPSR[4:0]时，则CPU会主动切换到对应工作模式(寄存器组)。

1. ARM指令条件位和Cw xPSR的ALU状态位关系

当两者关系一致时，则指令执行，否则不执行。

|  |  |  |  |
| --- | --- | --- | --- |
| 指令中的条件码 | | | CPSR的ALU状态位 |
| 0000 | EQ | Equal | Z=1 |
| 0001 | NE | Not Equal | Z=0 |
| 0010 | CS**/** HS | Higher or Same(unsigned>=) | C=1 |
| 0011 | CC**/** LO | Lower(unsigned<) | C=0 |
| 0100 | MI | Negative | N=1 |
| 0101 | PL | Positive or Zero | N=0 |
| 0110 | VS | Overflow | V=1 |
| 0111 | VC | No Overflow | V=0 |
| 1111 | HI | Higher(unsigned>) | C=1&Z=0 |
| 1001 | LS | Lower or Same(unsigned<=) | C=0|Z=1 |
| 1010 | GE | Greater or Equal(signed>=) | N=V |
| 1011 | LT | Lower(signed<) | N!=V |
| 1100 | GT | Greater(signed>) | Z=0&N=V |
| 1101 | LE | Lower or Equal(signed<=) | Z=1&N!=V |
| 1110 | AL | Always | Any |
| 1111 | NV | Always(ARM>=v5) Undefined(ARM=v3**/**v4)  Never(ARM<v3) | Any |

其中：

1. cpu会根据CPSR中的条件标志位决定是否执行该指令，如果条件不满足，该指令相当于一条nop指令；
2. 当指令后有S后缀时，CPU会根据执行情况更新CPSR的条件标志位；
3. 标记为无符号数的条件表示该指令不会去判断正负号，即不判断CPSR的N标志位，也就是上个指令计算结果的最高位(bit[31])。

### 协处理器15 (CP15)

1. 寄存器列表

|  |  |  |  |
| --- | --- | --- | --- |
| CP15 Register List | | |  |
| Register | R/W | Function | |
| MMU(memory manage) | MPU(memory protect) |
| C0.0 | RO | ID Code (1) |  |
| C0.1 | RO | Catch type(1) |  |
| C1 | RW | Control |  |
| C2 | RW | Translation table base | Cachability control |
| C3 | RW | Domain access control | Bufferablity control |
| C4 |  | Unpredictable |  |
| C5 | RW | Fault status(2) | Access control |
| C6 | RW | Fault address | Protect area control |
| C7 | WO | Cache operations |  |
| C8 | WO | TLB operations |  |
| C9 | RW | Cache lockdown(2) |  |
| C10 | RW | TLB lock down(2) |  |
| C11 |  | Unpredictable |  |
| C12 |  | Unpredictable |  |
| C13 | RW | Process ID |  |
| C14 |  | Unpredictable |  |
| C15 | RW | Test configuration |  |

1. 数据交换方式

写操作：MCR p15，0，<Rd>,<CRn>,<CRm>{,<opcode\_2>}

读操作：MRC p15，0，<Rd>,<CRn>,<CRm>{,<opcode\_2>}

1. CRm/Crn寄存器的编号范围为C0~C15。
2. <opcode\_2>根据CRn的不同，可取值如下(部分)：

|  |  |  |
| --- | --- | --- |
| CRn | function | value |
| C9 | cache类型 | 1=指令 0= 数据 |
| C10 | TLB类型 | 1=指令 0= 数据 |

1. C0寄存器

|  |  |  |
| --- | --- | --- |
| CR0.0 ID Code Register | | |
| bits | Function | Value |
| [31:24] | Implementor | 0x41(‘A’,表示Arm公司) |
| [23:20] | Specification revision | 0x1 |
| [19:16] | Architecture version(4T) | 0x2(ARM体系版本4T) |
| [15:4] | Part number | 0x920 |
| [3:0] | Layout revision | 0x0 |

|  |  |  |
| --- | --- | --- |
| CR0.1 Cache type register | | |
| bits | Function | Value |
| [31:29] | Reserved | 000 |
| [28:25] | Cache type | 0110 |
| [24] | Harvard/Unified | 1(defines Harvard cache) |
| [23:21] | Reserved | 000 |
| [20:18] | Data Cache size | 101(defines 16KB) |
| [17:15] | Data Cache associativity | 110(defines 64 way) |
| [14] | Reserved | 0 |
| [13:12] | Data Cache words per line | 10(defines 8 words per line) |
| [11:9] | Reserved | 000 |
| [ 8:6] | Instruction Cache size | 101(defines 16KB) |
| [5:3] | Instruction Cache Associativity | 110(defines 64 way) |
| [2] | Reserved | 0 |
| [1:0] | Instruction Cache per line | 10(defines 8 words per line) |

1. C1寄存器

|  |  |  |
| --- | --- | --- |
| C1 Control register | | |
| bits | Function | Value |
| [31:30] | bus mod | 00=FastBus 01=Syn 11=Asyn |
| [29:15] | Reserved | Write(0) Read (Unpredictable) |
| [14] | replacement | 0 = Random 1 = Round robin |
| [13] | Base location of exception register | 0 = Low address 1 = High address |
| [12] | Instruction cache enable | 0 = disable 1 enable |
| [11:10] | Reserved | Write = Read = 00 |
| [9:8] | ROM protection | 见图1 |
| [8] | System protection | 见图1 |
| [7] | Big-endian/little-endian | 0 = Little-endian 1 = Big-endian |
| [6:3] | Reserved | Write = Read = 1111 |
| [2] | Data cache enable | 0 = disable 1 enable |
| [1] | Data address alignment fault checking | 0 = disable 1 enable |
| [0] | MMU enable | 0 = disable 1 enable |

1. C2寄存器

|  |  |
| --- | --- |
| C2 Translation Table Base(TTB) Register | |
| bits | Function |
| [31:14] | Pointer to first level translation table base (PGD Entry) |
| [13:0] | Reserved Read (Unpredictable) Write (0) |

1. C5寄存器

|  |  |
| --- | --- |
| C5 fault status | |
| bits | Function |
| [31:8] | UNP/SBZP |
| [7:4] | domain type |
| [3:0] | fault type |

其中fault type定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| C5[3:0] | | fault type | |
| Operation Type | address type |
| 0 | 0000 | Exception | Vector |
| 1 | 0001 | Not alignment | Byte |
| 3 | 0011 |
| 2 | 0010 | TLB Miss | PTE |
| 12 | 1100 | PDU |
| 14 | 1110 | PMU |
| 5 | 0101 | Translation | Section |
| 7 | 0111 | Page |
| 9 | 1001 | Domain Control Failed | Section |
| 13 | 1101 | Page |
| 15 | 1111 | permission control Failed | Section |
| 4 | 0100 | Page |
| 6 | 0110 | Cache Miss | Section |
| 8 | 1000 | Page |
| 10 | 1010 | Through Access Failed | Section |
| 11 | 1011 | Page |

1. C6寄存器

|  |  |
| --- | --- |
| C6 fault address | |
| bits | Function |
| [31:0] | fault address (virtual address) |

1. C13寄存器

|  |  |
| --- | --- |
| C13 OS Context Switch | |
| bits | Function |
| [31:25] | PID |
| [24:0] | VMA |

### 向量处理模块

Neon和VFP(Vector Floating-point Coprocessor for ARM)是ARM用于实现向量浮点计算的模块。

1. NEON / VFP
2. NEON和VFP共用一组32x64bit register file；
3. NEON和VFP 都是通过CP10 和CP15 协处理器完成；
4. NEON仅 支持向量计算，不支持双精度浮点运算，不支持某些复杂操作(比如：square root and divide)，其可用于SIMD引擎，类似DSP；
5. NEON和VFP处于同一个power domain， ARM CPU Core可以动态的Power on/Power off Neon/VFP，从而节省power.
6. SIMD指令集

SIMD(Single Instruction Multiple Data)，表示单指令多数据流，指能够复制多个操作数，并把它们打包在大型寄存器的一组指令集。比如3DNow!、SSE。

SIMD指令一般以同步方式运行，在同一时间内执行同一条指令，适合于多媒体应用等数据密集型运算。

1. ARM架构支持组合
2. No NEON or VFP
3. VFP only
4. NEON and VFP

### 协处理器指令

1. 指令格式：

MCR{<cond>} <p>，< opcode\_1>，<Rd>,<CRn>,<CRm>{,<opcode\_2>}

MRC{<cond>} <p>，< opcode\_1>，<Rd>,<CRn>,<CRm>{,<opcode\_2>}

1. <p>：协处理器(CP)的编号
2. <Rd>和<CRn>：需要交换数据的寄存器，<Rd>为ARM寄存器，<CRn>为CP寄存器。
3. <opcode\_1>，<CRm>和<opcode\_2>：通过这三个的组合最终确定CP的具体操作，其中<opcode\_1>为主操作码，CRm默认为C0寄存器，opcode\_2默认为0。
4. 功能：
5. MCR：数据交换后CP🡨 ARM， CP开始执行指令。
6. MRC：CP执行指令后开始交换数据ARM 🡨 CP。
7. MCR/MRC指令使用限制：

只能在CPU系统模式时执行，在用户模式下执行将产生Undefined异常。

1. 举例如下：

MCR p15, 0, R4, C1, C0, 0

表示C1🡨 R4，然后p15根据C0,0组合执行0编号的操作。

MRC p15, 0, R3, C2, C0, 0

表示p15根据C0,0组合执行0编号的操作，然后R3🡨C2。

### 向量处理指令

1. 指令格式：

VMRS{cond} Rd, extsysreg

VMSR{cond} extsysreg, Rd

1. 功能：在ARM寄存器(Rd)和向量模块寄存器(extsysreg)之间传输数据。
2. VMRS： ARM 🡨 VM。
3. VMSR： VM 🡨 ARM。
4. 举例如下：

VMRS APSR\_nzcv, FPSCR; 将FP 状态寄存器的值写入ARM 的APSR。

### 多寄存器寻址

1. 入栈指令🡨(STMFD/STMDB)

STMFD SP!,{R2-R4}

[SP-4] 🡨 R4

[SP-8] 🡨 R3

[SP-12] 🡨 R2

1. 出栈指令🡪(LDMFD/LDMIA）

LDMIA R0!,{R1-R3}

[R0] 🡪 R1

[R0+4] 🡪 R2

[R0+8] 🡪 R3

1. 列表顺序和存储位置

不论入栈和出栈的方式如何，还是寄存器列表中寄存器顺序如何，数据交换时，最小编号的寄存器对应当次寻址范围的最低地址。

### 汇编指令的特殊符号

1. “^”和“S”
2. “^”仅用于STM/LDM指令，添加在寄存器列表之后。
3. “S”用于STM/LDM和所有数据处理类的指令，添加在操作码之后。
4. “^”和“S”功能相同。
5. 该标志只在特权模式(非usr模式)下有效。
6. 对于TEQ，TST，CMP，CMN测试类型指令，其默认已经设置了“S”标志位，所以无需再添加到指令中。
7. 模式改变功能

当使用LDM和数据处理指令时，如果pc寄存器出现在目标寄存器(或寄存器列表)的位置，则表示数据交换后，同时完成SPSR\_<mode>🡪CPSR的复制。

1. 标志位设置功能或用户块传输(User Bank Transfer)功能

如果pc寄存器没有出现在目标寄存器(或寄存器列表)的位置，则根据实际ALU状态更新CPSR的标志位。如果同时操作码为STM/LDM，则还表示列表中的寄存器是user模式的寄存器，而不是当前模式的寄存器。

1. “！”

用于STM/LDM和LDR/STR指令，添加在<基址寄存器>后，表示数据交换后，将结果地址写回<基址寄存器>。

### rsb指令

1. 指令格式：RSB{cond}{S} Rd,Rn,operand2
2. 功能：逆向减法，用寄存器operand2 减法Rn，结果保存到Rd 中。
3. 举例如下：

RSB R3,R1,#0xFF00 ;R3=0xFF00-R1

RSBS R1,R2,R2,LSL #2 ;R1=(R2<<2)-R2即R2×3

RSB R0,R1,#0 ;R0=-R1

### svc与swi指令

1. 功能

SVC 的功能与 SWI 是相同的，其功能都是从当前模式切入到软件中断模式。

这两个指令的机器码是相同的，之所以有两个名称，是因为在 CM3中，异常处理模型已经“洗心革面”了，就故意把该指令也重命名，以强调它是在新生的系统中使用的。并且让程序员在把 ARM7代码移植到 CM3时，能充分注意到这个本质的不同（至少必须得改名，每次改名时都得到警示）。

1. 执行效果

指令执行后，处理器进入超级用户模式，CPSR 存储到 spsr\_svc 中，返回地址存储在 lr\_svc 中。

1. 使用注意

如果在超级用户模式下调用 SVC或者嵌套使用SVC，则必须存储lr\_svc和 spsr\_svc，保证旧的lr\_svc 和 spsr\_svc的可以还原。

1. 应用场合

SVC通常用于在操作系统上请求特权操作或访问系统资源。SVC指令中嵌入了一个数字，这个数字通常称为 SVC 编号，此编号一般用于指示要请求的服务。

### ldr 和 adr指令

1. ldr reg, lable @将 lable对应内存地址中的**值**加载到reg

lable的寻址方式: 使用编译时确定的链接地址;

1. ldr reg, =lable @将lable对应内存地址加载到reg

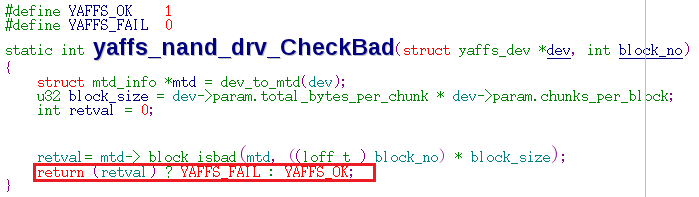
lable的寻址方式: 使用编译时确定的链接地址，lable被当作表示所在内存地址的一个**立即数**，所以当 lable的值可以用mov指令直接赋值时，则ldr指令被直接翻译为一条mov指令，否则会多出一个内存空间用于存储lable的内存地址，然后使用ldr加载这个内存的值(即lable的链接地址)，即间接寻址。

1. adr     reg, lable @将lable对应的内存**地址**加载到reg

lable的寻址方式: 使用链接地址的相对偏移值，然后被翻译为与运行时的PC值做add指令或sub指令。

### 条件判断语句的实现

以下C代码中的条件判断语句如红色方框内：



其反汇编代码如下：



其中：

1. 汇编运行流程
2. retval即初始的r0
3. 执行(1- retval)，将结果存入r0，并更新CPSR的条件标志位。
4. 如果为CPSR的C位等于0，则r0=0，否则为 1- retval。
5. 汇编可能运行结果
6. retval <1，1- retval不需要借位，故C=1，N=0，Z=0，则r0=不确定正数；
7. retval ==1，1- retval不需要借位，故C=1，N=0，Z=1，则r0=1-1=0；
8. retval >1，1- retval，需要借位，故C=0，N=1，Z=0，则r0=0；
9. 因为CC条件不检查N位，所以如果在retval <1的前提下，又满足retval<-1，则1-abs(retval)是需要借位的，这样C=0，继而导致r0=0，从而覆盖了其原来的“不确定正数”；这样retval<1中就只剩下retval=0，r0=1-0=1的情况和retval=-1，r0=1-abs(-1)=0的情况；
10. 结果分析
11. C代码中当retval为真时，结果为0；
12. 汇编中，最终r0=0的条件为retval>1，retval=1，retval=-1，retval<-1，即retval为不等于0的任意值；
13. 条件判断的本质

从结果可以看出，条件为真的本质是条件表达式的值为非0，条件为假的本质是条件表达式的值为0。

## DSP

## FPGA

## GPU

### 显卡

1. ATIAMD
2. NVIDIA

Intel

### 嵌入式

DSP

1. MALI

# 存储设备

## DDR

## 寄存器

CSR=Control and Status Register

## ROM

## MMU

## NandFlash

### 存储特点

NandFlash是一种不可靠的存储介质，写入的bit可能出现反转，为了提高可靠性，一般采取如下措施：

1. 写数据时，生成对应区域的校验信息和数据一起存储；读数据时，需要先读取对应位置的校验信息，然后和读出来的数据做校验处理。该校验信息一般使用的是ECC。
2. 读写功能由NandFlash自身提供命令接口实现，[ECC](#_ECC的使用)一般由外部控制器或软件校验实现。
3. 制作nandflash时，增加额外的容量用于存储校验信息，保证容量大小的可靠性。这个额外增加的容量一般习惯称为OOB区域：
4. 该区域与其他区域一样具有相同的物理特性；
5. 该区域的功能没有特殊规定，可以存储任意数据。通常情况下用于存储读写辅助信息，比如ECC，文件系统的额外信息等，具体内容由读写双方约定；
6. 该区域的位置以及大小由读写双方约定，可以是nandflash的任意区域。

### 常用命令

1. 读命令

读取操作类似内存的操作，最小单位为其数据总线的宽度。

1. 擦除命令
2. 因为nandflash的写入数据本质上是对非1 bit的翻转，所以在写入前必须先使用擦除命令将目标区域全部置为比特1。
3. 擦除命令以块单位(块大小可查看nandflash手册)进行，执行后对应区域将全部置为比特1。
4. 写命令
5. 写命令是以页为单位的。
6. 如果在写之前，对应区域的bit不全为1，则必须先擦除所在页的整个块。
7. 一般在写数据时，先擦除整个块，然后按照页的方式逐次写入数据，当一个块写完时，则擦除下个块继续按页写入。
8. 定位命令
9. nandflash物理上是一个字节连续的存储介质，所有的存储单元分布在一个二维空间，通过行地址和列地址来确定某个单元，即可以定位到任意字节偏移位置。
10. 在执行不同命令前，需要设置定位目标偏移，此时必须保证目标位置后续存储空间不小于对应命令的最小访问单位。

### 存储空间编址

一般NandFlash对其存储空间编址方法由其实际nandflash 页大小决定，编址方法不仅要方便主数据读写，也要方便定位NandFlash的OOB区。

1. 物理空间。

NandFlash在物理上是一块地址连续的存储介质，所有数据从低地址到高地址依次排序。

1. 访问空间。

因为OOB区是数据区的“额外空间”，所以一般在读写地址空间内是不会将其纳入编址范围，如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| pagesize = 2048，oobsize=64 | | | | | | |
| 存储布局 | data[0] | oob[0] | page[1] | oob[1] | page[2] | oob[2] |
| 物理空间(起始偏移) | 0 | 2048 | 2112 | 4160 | 4224 | 6272 |
| 访问空间(起始偏移) | 0 |  | 2048 |  | 4096 |  |

其中用户的访问空间如黄色区域。即：从访问者角度看，访问空间实际是隐藏了OOB区的存在。所以实际在读写时，需要进行地址转换，将用户的访问空间映射到对应物理空间。

1. 地址映射

一般NandFlash的定位命令会将其定位的行列按照其页大小和OOB大小规划如下：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| NandFlash参数：pagesize = 2048，oobsize=64 | | | | | | | | | | |
| 列地址 行地址 | 0 | 1 | 2 | …… | 2047 | 2048 | 2049 | 2050 | …… | 2111 |
| 0 | data[0] | | | | | oob[0] | | | | |
| 1 | data[1] | | | | | oob[1] | | | | |
| 2 | data[2] | | | | | oob[2] | | | | |
| …… | …… | | | | | …… | | | | |
| n-1 | data[n-1] | | | | | oob[n-1] | | | | |

其中：

* 1. 行地址即代表其页号，一行的单位即数据区和OOB的长度和。
  2. 行数据的布局需要读写双方约定为相同布局。
  3. MTD默认行数据布局规则是：将data区数据连续布局在0列的地址偏移处，OOB紧跟其后。

1. 访问方法。

OOB对应数据的访问只能通过MTD层的接口根据“主空间”的位置来映射。

有些NandFlash的驱动已经提供了OOB区的访问命令，MTD可以直接从驱动层访问到OOB。

无论哪种OOB区的访问只能通过特定接口获取，使用时只需给定OOB的缓冲信息和主数据位置即可。

### ECC原理

对数据的校验常用的有奇偶校验、CRC 校验等，而在NandFlash处理中，一般使用ECC (Error Checking and Correcting)。

ECC是一种用于memory器件(比如NandFlash或者内存)的差错检测和修正算法，一般由特定硬件或者软件算法完成ECC的生成和校验。

1. 基本流程如下：
2. 在数据写入时，通过输入的数据生成一个校验码，然后和数据一起存储起来；
3. 当数据读取时，再次生成一个校验码并和存储的校验码比较，来确定数据的有效性。
4. 如果校验错误，可以认为本次访问区域的介质已经出错，需要对该区域进行错误标记。为了简化坏块管理，一般是以block大小为单位，即发生一次错误，则整个block都将废弃，实际上其他区域的介质并不一定存在错误。
5. 实现方式：
6. ECC的算法由读写双方约定，实现的方法有多种。比如汉明码，BCH码。
7. ECC数据的读写方法和普通数据相同。
8. ECC的生成和校验一般分为硬件模块(通常是nandflash控制器，目前很少有nandflash自身支持ECC)实现，或者纯软件方式实现；
9. 使用硬件实现时，ECC算法对使用者而言已经封装，只能通过硬件手册查询其[纠错能力](#_ECC的纠错能力)，具体算法一般不会提供。此时nandflash会自动使用最近读取的数据作为ECC算法模块的源输入，使用者执行了nand读取或写入后可以直接读取硬件模块模块的ECC，该ECC即对应数据的校验码。
10. 使用软件实现时，需要自行确定算法以及算法的输入。

### ECC的纠错能力

NandFlash在出厂时因为所采用的工艺不同，对ECC模块的纠错能力要求也不同，一般情况下，越是容量大的Nand芯片出错的概率就越高，越需要更强的纠错算法。具体该需要多大的纠错能力主要由以下方面决定。

1. 算法限制

一般的汉明码修正1bit，检测2bit；BCH可以修正8bit，12bit，16bit。

1. SLC和MLC工艺差异

SLC 和MLC分别是是Single Layer Cell 单层单元和Multi-Level Cell多层单元的缩写。

1. SLC Nand的位翻转在1Bit **/** 512byte 内，需纠错能力至少1bit**/**512byte；
2. MLC Nand的位翻转在8Bit**/**512 byte内，需纠错能力至少8bit**/**512byte。
3. 厂家实现技术差异

即使相同工艺，因采用不同技术实现，不同厂家的NandFlash位翻转个数也是不一样的，这个需要从nandflash器件手册上查询，一般厂家会标明该nandflash对ECC纠错能力最低要求。

1. 系统配置不匹配

如果系统配置的ECC模块不能提供给nandflsh要求的最低ECC能力，则可能造成在nandflash出错后系统无法纠正，此时可如下处理：

1. 如果是硬件实现的ECC，可以将ECC算法改为软件方式，从而将ECC算法更换，或者直接更换ECC模块(一般就是换cpu)；
2. 更换nandflash；
3. 不使用ECC。

### ECC算法特征

这里说的ECC算法实现是指采用Hamming Code（汉明码）实现的ECC算法，其特征如下：

1. 算法流程
2. 将数据在二维空间内以行列的形式分割。
3. 计算CP(Column Parity)。

即在列的方向计算奇偶校验码，其方法如下图。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 位序字节序 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |
| …… |  |  |  |  |  |  |  |  |
| N-1 |  |  |  |  |  |  |  |  |
| CP位序 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 3 | | 2 | | 3 | | 2 | |
| 5 | | | | 4 | | | |

其中：

CP0=Bit0^Bit2^Bit4^Bit6，即数据区所有白色区域的bit进行异或。

CP2=Bit0^Bit1^Bit4^Bit5，即数据区所有蓝色边框内的bit进行异或。

CP4=Bit0^Bit1^Bit2^Bit3，即数据区所有斜线底纹的bit进行异或。

而CP1，CP3，CP5则分别是CP0，CP2，CP4未使用的bit异或。

其计算方法是将所有列按照二分法的规则构造出一个完全二叉树结构，CP序列即为搜索二叉树一个叶子结点所有路径的选择极性值，其中01表示右树，10表示左数。

当需要使用CP查找错误时，则从CP序列按照每次取2个bit的方式逐次扫描，当检测到相同值(00或11)的bit对时，则表示出错(因为二叉树的选择只有左树或右数，不可能都选择左树或右数)。

nandflash用于ECC校验的数据单位为8bit，构造的二叉树高度为4(此处根节点高度为1，部分文献中根节点高度为0)，则其列对应的奇偶校验码长度为 6，其计算公式如下：

1. 在行的方向计算校验位串。

一般称为RP(Row Parity)，即：行奇偶校验。RP串的长度是由总数据的长度决定。

其构造方法和列方法类似，即(假设为256字节)：

RP0=Byte0^Byte2^Byte4……^Byte252^Byte254

RP2=Byte0^Byte1^Byte4……^Byte252^Byte253

而RP1，RP3则分别是CP0，CP2未使用的bit异或。

1. 将CP串和RP串连接。

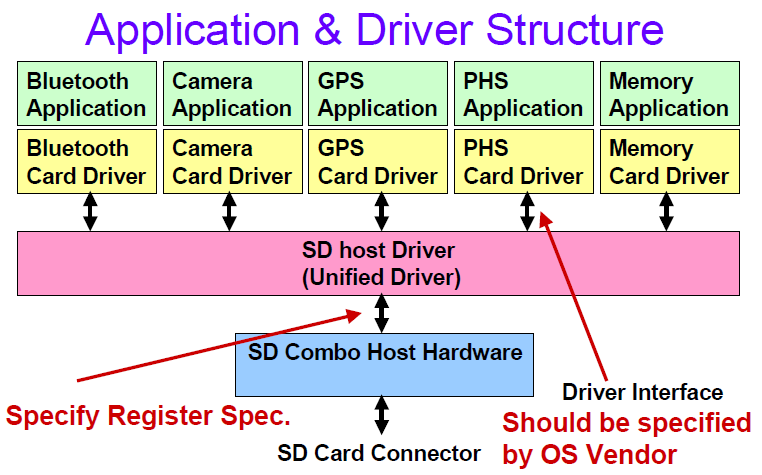
一般将RP作为ECC的低字节位，CP作为高位，其总长度为公式如下：

1. 软件实现和硬件实现的区别
2. 采用硬件实现时，一次可进行ECC的数据长度由其ECC纠错能力决定，一般可以从编程手册查询(一般是在CPU手册中的ECC模块章节)，查询s3c2440手册，可知其一次可以进行2048字节的校验。
3. 采用软件时，是使用固定长度数据(一般采用256字节)生成一个ECC串，如果目标数据长度大于一次ECC长度，则将数据分割为若干段，每个段单独进行ECC，然后简单的连接起来即该数据的ECC，校验时每个段的的数据只使用其对应的ECC段。
4. 与CRC的比较
5. ECC可以纠正1比特错误和检查出2比特错误(1bit correction & 2bit detection )，但对1比特以上的错误无法纠正，对2比特以上的错误不保证能检测。CRC只能检查出错误是否存在。
6. ECC校验数据时，数据位每增加1倍，检验位增加1bit。CRC校验时，每单位个数据都需要1bit。但如果数据量很大时，比如2048字节时，单位为8bit，则需要2048bit=256字节的校验信息，而ECC只需要28bit，不到4个字节。
7. ECC校验方法和CRC类似，也是采用位异或的方法生成校验位。
8. CRC是将所有数据在一维空间内分割为固定单位，每个单位生成一个校验位，且只用于该单位校验，各单位之间相互独立。
9. ECC是将数据在二维空间内按照固定单位分割，所有单位统一生成一串校验位串，用于所有数据的校验，各单位之间相互关联。

## SD Host Controller

### Architecture

1. AP & Driver Structure



It is the intention of this specification that by defining a standard SD Host Controller, Operating System(OS) Vendors can develop SD Host Drivers that will work with Host Controllers from any vendor.

Applications may in addition require the Card Drivers. The Card Drivers will communicate with the Host Driver using a driver interface specified by the OS.

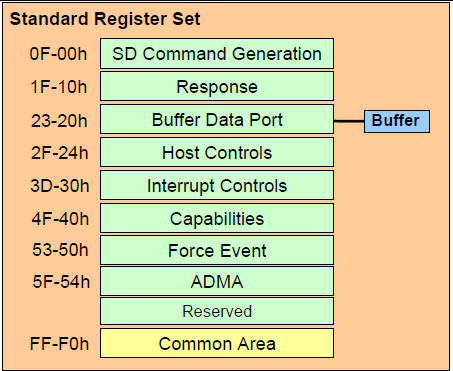
1. bus interfaces

This specification can be applied to any system bus interface.

The Host Controller has two bus interfaces, the System Bus Interface and the SD Bus Interface.



1. Classification of the Standard Register Map
2. Standard register set



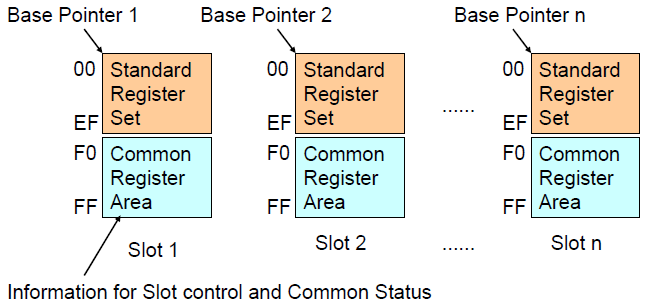
The standard register map is classified in 9 parts listed below.

|  |  |  |
| --- | --- | --- |
| addr | name | function |
| 0F-00h | SD command generation | Parameters to generate SD commands |
| 1F-10h | Response | Response value from the card |
| 23-20h | Buffer Data Port | Data access port to the internal buffer |
| 2F-24h | Host Controls | Present State, controls for the SD Bus, Host reset and so on. |
| 3D-30h | Interrupt Controls | Interrupt statuses and enables. |
| 4F-40h | Capabilities | Vendor specific host controller information |
| 53-50h | Force Event | Test register to generate events by software. |
| 5F-54h | ADMA | Advanced DMA registers. |
| … | Reserved |  |
| FF-F0h | Common Area | Common information area. |

The Host Controller shall support byte,word and double word accesses to these registers. Reserved bits in all registers shall be fixed to zero.The Host Controller shall ignore writes to reserved bits; however, the Host Driver should write them aszero to ensure compatibility with possible future revisions to this Specification.

1. Multiple Slot support

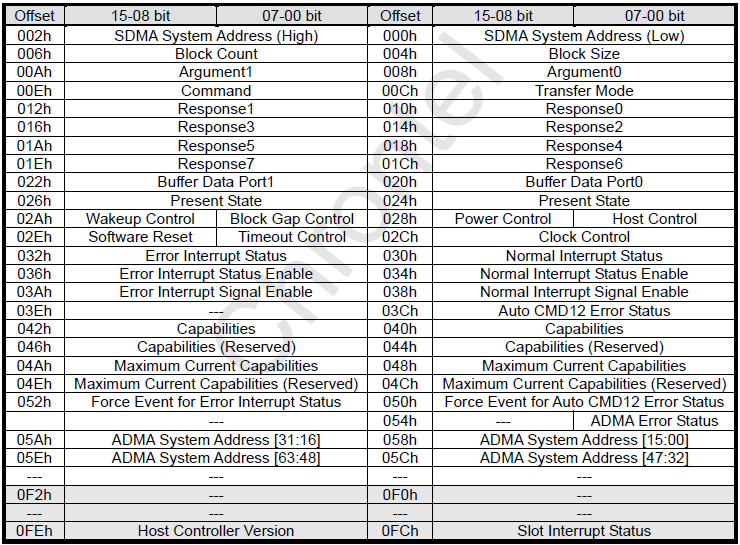
One Standard Register Set is defined for each slot. If the Host Controller has two slots, two register sets shall be required. Each slot is controlled independently. This enables support for combinations of Fullspeed and Low speed cards in regards to SD Clock frequencies.



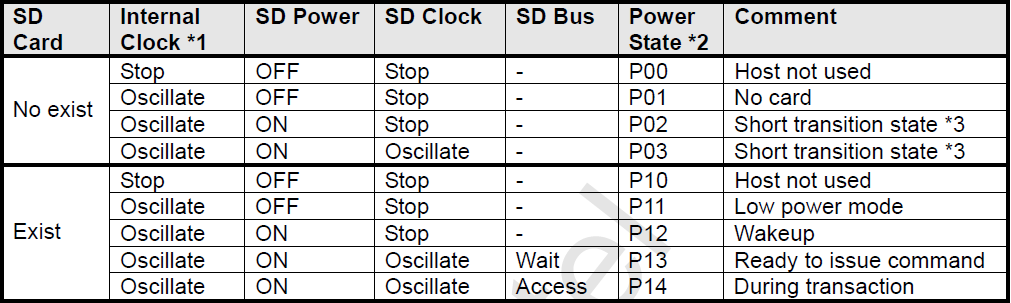
1. register spec

The Host Driver needs to determine the base address of the register set by a Host System specific method.

The register set is 256 bytes in size. For multiple slot controllers, one register set is assigned per each slot, but the registers at offsets 0F0h-0FFh are assigned as a common area. These registers contain the same values for each slot's register set.



1. Power state



States description

|  |  |  |
| --- | --- | --- |
| field | optional value | corresponding register**/**note |
| Internal clock | oscillate/stop | Internal Clock Enable in the **Clock Control register**  Should **be stopped** when the Host System does not use the Host Controller |
| SD Card | Exist/Not exist | Card Inserted in the **Present State register** |
| SD Power | ON/OFF | SD Bus Power in the **Power Control register**. |
| SD Clock | oscillate/stop | SD Clock Enable in the **Clock Control register**.  Shall **not be supplied** when card power is OFF. |
| SD Bus | access/wait (idle) | Command Inhibit (CMD) and Command Inhibit (DAT) in the **Present State register** |
| Power state | P00/P01/…P14 | Power states are not actually implemented in Host Controller. This label is for reference. |
| comment |  | Short transition state: Temporary power states. The Host Controller automatically goes to P01 when it detects No Card. |

### DMA Transfer

The Host Controller provides a "programmed I/O" method for the Host Driver to transfer data using theBuffer Data Port register. Optionally, Host Controller implementers may support data transfer using DMA, Support of SDMA and ADMA are optional for the Host Controller.

1. SDMA

The DMA algorithm defined in the SD Host Controller Standard Specification Version 1.00 is called SDMA(Single Operation DMA). Only one SD command transaction can be executed per a SDMA operation.Support of SDMA can be checked by the **SDMA Support** in the *Capabilities* register.

SDMA had disadvantage that DMA Interrupt generated at every page boundary disturbs CPU to reprogram the new system address. This SDMA algorithm forms a performance bottleneck by interruption at every page boundary.

1. ADMA

SD Host Controller Standard Specification Version 2.00 defines a new DMA transfer algorithm called ADMA (Advanced DMA). ADMA provides data transfer between system memory and SD card without interruption of CPU execution. Support of ADMA can be checked by the Capabilities register.

1. Features

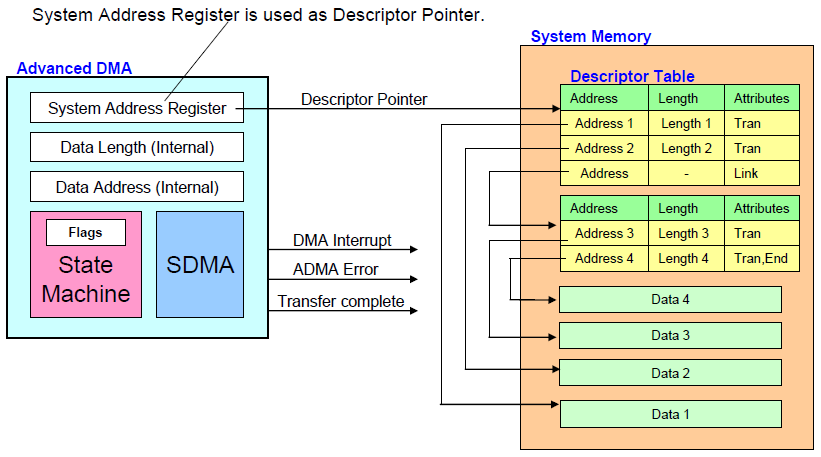
ADMA adopts scatter gather DMA algorithm so that higher data transfer speed is available. The Host Driver can program a list of data transfers between system memory and SD card to the Descriptor Table before executing ADMA.

It enables ADMA to operate without interrupting the Host Driver. Furthermore, ADMA can support not only 32-bit system memory addressing but also 64-bit system memory addressing. The 32-bit system memory addressing uses lower 32-bit field of 64-bit address registers.

There are two types of ADMA; ADMA1 and ADMA2. ADMA1 can support data transfer of only 4KB aligned data in system memory. ADMA2 improves the restriction so that data of any location and any size can be transferred in system memory.

1. ADMA2

The Host Controller Specification Ver2.00 defines ADMA2 as standard ADMA and recommends supporting ADMA2 rather than ADMA1:



The Descriptor Table is created in **system memory by the** **Host Driver**.

32-bit Address Descriptor Table is used for the system with 32-bit addressing and 64-bit Address Descriptor Table is used for the system with 64-bit addressing.

Each descriptor line (one executable unit) consists with address, length and attribute field. The attribute specifies operation of the descriptor line.

ADMA2 includes SDMA, State Machine and Registers circuits.

ADMA2 does not use 32-bit SDMA System Address Register (offset 0) but uses the 64-bit Advanced DMA System Address.

1. DMA Usage

Prior to using DMA, the Host Driver shall confirm that both the Host Controller and the system bus support it (PCI bus can support DMA).

1. transfer

DMA shall support both single block and multiple-block transfers but does not support infinite transfers.

Host Controller registers shall remain accessible for issuing non-DAT line commands during a DMA transfer execution.

The result of a DMA transfer shall be the same regardless of the system bus data transfer method.

1. stop/restart/abort

The Host Driver can stop and restart a DMA operation by the control bits in the *Block Gap Control* register.

By setting Stop At Block Gap Request, a DMA operation can be stopped at block gap. If an error occurs, DMA operation shall be stopped.

By setting Continue Request, DMA operation can be restarted.

To abort DMA transfer, Host driver shall reset the Host Controller by the **Software Reset For DAT Line** in the *Software Reset* register and issue CMD12 if multiple-block read / write command is executing.

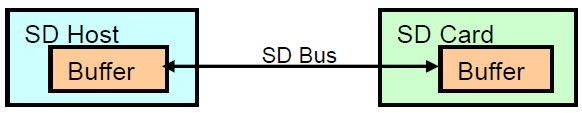
1. Transactions

Below table shows register settings (at offsets from 000h to 00Fh in the register set) necessary for three types of transactions: DMA generated transfers, CPU data transfers (using "programmed I/O") and non-DAT transfers.

|  |  |  |  |
| --- | --- | --- | --- |
|  | DMA command | CPU data transfer | Non-DAT transfer |
| SDMA System Address | Yes | No | No |
| Block Size | Yes | Yes | No (Protected) |
| Block Count | Yes | Yes | No (Protected) |
| Argument | Yes | Yes | Yes |
| Transfer Mode | Yes | Yes | No (Protected) |
| Command | Yes | Yes | Yes |

1. When initiating a transaction, the Host Driver should program these registers sequentially from **000**h to **00F**h.
2. The beginning register offset may be calculated based on the type of transaction.
3. The last written offset shall be always 00Fh because writing to the upper byte of the Command register shall trigger issuance of an SD command.
4. The Host Driver should not read the SDMA System Address, Block Size and Block Count registers during a data transaction unless the transfer is stopped or suspended because the value is changing and not stable.
5. To prevent destruction of registers using data transfer when issuing command, the Block Size, Block Count and Transfer Mode registers shall be write protected by the Host Controller while Command Inhibit (DAT) is set to 1 in the Present State register. (The SDMA System Address cannot be protected by this signal.)
6. The Host Driver shall not write the Argument and Command registers while Command Inhibit (CMD) is set to 1. The Host Driver shall protect the SDMA System Address, Argument and Command registers.

### Buffer Control



The Host Driver shall use the same data block length for both the Host Controller and the card. If the controller and card buffer sizes are different, the Host Driver shall use the smaller value.

1. Host Controller buffer size

The maximum Host Controller buffer size is defined by the **Max Block Length** field in the Capabilities register.

1. Card buffer size

The card buffer size is described as **maximum block length** in the **Card Specific Data** (CSD) register for **memory cards** (for cards compliant with the Physical Layer Specification, READ\_BL\_LEN and WRITE\_BL\_LEN shall be the same) and in the **Card Information Structure** (CIS) for **SDIO cards**.

1. memory card

Physical Layer Specification re-defines that **maximum block length** is only used to calculate capacity of memory card. Even though it indicates larger than 512 bytes, block length shall be set to 512 byte for data transfer. This is because 512 bytes block length is required to keep compatibility with 512 bytes data boundary.

1. SDIO card

If the SDIO card has multiple buffers, the block size in CIS indicates the size of the smallest buffer; this is the value the Host Driver should use when programming block size. Buffer information (for example, buffer port address) in the SDIO card is function specific.

1. Dividing large data transfer

The SDIO command CMD53 definition limits the maximum data size of data transfers according to the following formula:

Max data size = Block size x Block count

1. Dividing methods

If an application or Card Driver wants to transfer larger sizes of data, the Host Driver shall divide large data into multiple CMD53 blocks.

For example, Block size is specified by the buffer size as described in Capabilities register and the block count can be a maximum of 512 (9-bit count) as specified in the command argument for CMD53.

In the worst case,if the card has only a **1 byte buffer**, up to **512 bytes of data** can be transferred using CMD53 (Block Size =1, Block Count = 512).

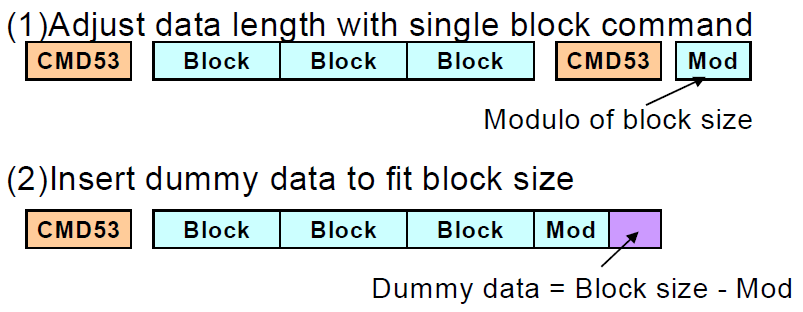
If the card does not support multiple-block mode, **only** **one byte** can be transferred in this case.

1. data length restriction

The length of a multiple block transfer needs to be in block size units. If the total data length cannot be divided evenly by the block size, there are two ways to transfer the data depending on the function and card design:

Option 1 is for the Card Driver to split the transaction. The remainder of block size data is transferred by using a single block command at the end.

Option 2 is to add dummy data in the last block to fill the block size. This is efficient but the card must manage or remove the dummy data.



1. Buffer Data Port Register (Offset 020h)

|  |  |  |
| --- | --- | --- |
| Location | Attrib | Register Field Explanation |
| 31-00 | RW | Buffer Data  The Host Controller buffer can be accessed through this 32-bit Data Port register. |

### Interfaces Sync

1. Interfaces clock

The **Host Controller** assumes that these interfaces are asynchronous (that is, are working on different clock frequencies). Host Controller shall synchronize signals to communicate between these interfaces.

1. system clock

The **Host Driver** is on system bus time (because it is software executed by the Host Controller CPU, on its system clock).

All status registers shall be synchronized by the system clock and maintain synchronization during output to the system interface (except, optionally, statuses for testing).

1. SDCLK

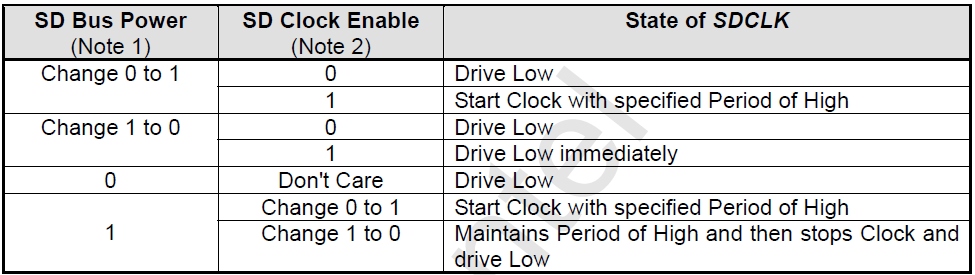
The **SD card** is on SD Bus time (that is, its operation is synchronized by SDCLK).

Control registers, which trigger SD Bus transactions, shall be synchronized by SDCLK. Therefore, there will be a timing delay when propagating signals between the two interfaces.

The Clock Period of SDCLK is specified by the **SDCLK Frequency Select** in the Clock Control register and the Base Clock Frequency For SD Clock in the Capabilities register.

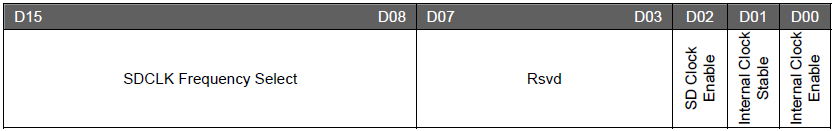
Because of the SD card may use both clock edges, the duty of SD clock should be average 50% (scattering within 45-55%) and the Period of High should be half of the Clock Period.

The oscillation of SDCLK starts from driving specified Period of High.



1. Clock Control Register (Offset 02Ch)

At the initialization of the Host Controller, the Host Driver shall set the SDCLK Frequency Select according to the Capabilities register.

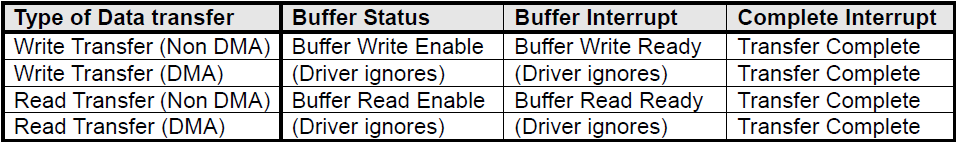


### Buffer sync

Blocks of data shall be synchronized at the buffer module. The Buffer Interface enables internal read and write buffers.

1. transfer sync

The Transfer Complete interrupt status indicates completion of the read / writes transfer for both DMA and non DMA transfers.

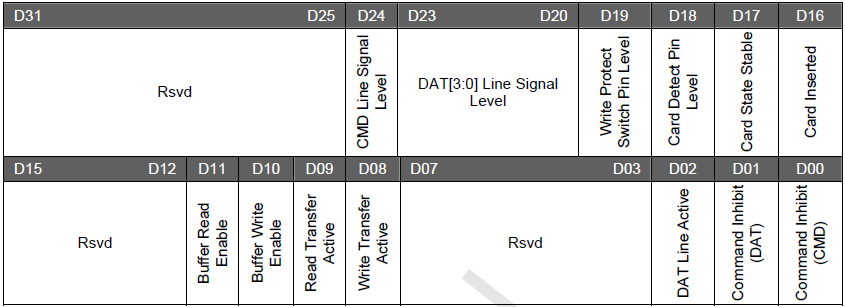


However, **the timing is different between reads and writes**.

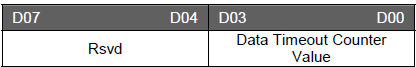
Read transfers shall be completed after all valid data have been transferred to the Host System and are **ready for the Host Driver to access**.

Write transfers shall be completed after all valid data have been transferred to the SD card and the **busy state is over**.

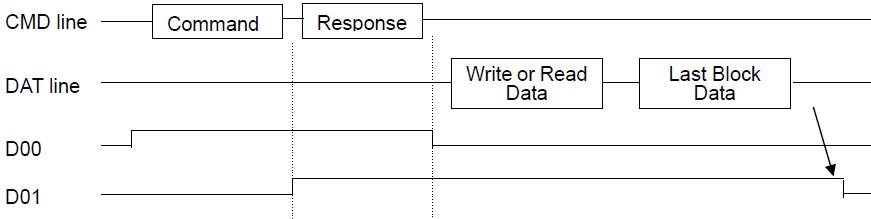
1. Present State Register (Offset 024h)



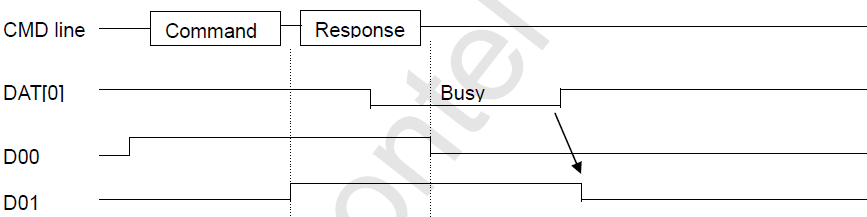
1. Timeout Control Register (Offset 02Eh)



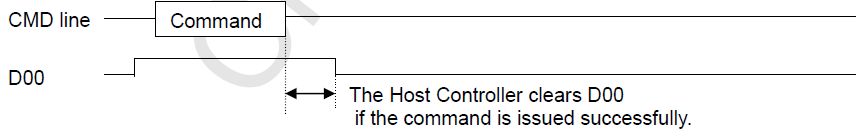
1. timing of setting and clearing the Command
2. Timing of Command(DAT and CMD) with data transfer



1. Timing of Command Inhibit (DAT) for the case of **response with busy**



1. Timing of Command Inhibit (CMD) for the case of **no response** command



### Interrupt Control

1. Interrupt source

The Host Controller implements a number of interrupt sources(for ap).



1. signal types

Interrupt sources can be enabled as **interrupts** or as **system wakeup signals**.

The **interrupt signal** and **wakeup signal** are logical Ored(相“或”) and shall be read from the ***Slot Interrupt Status* register**.

The ***Wakeup Control* register** enables **Card Interrupt**, **Card Insertion,** or **Card Removal** status changes to be configured to generate a system wakeup signal. These interrupts are enabled or masked inde*p*endently of the ***Normal Interrupt Signal Enable***register. The kind of wakeup event can be read from the *Normal Interrupt Status* register.

1. active interrupt

If the interrupt source's corresponding bit in the Normal Interrupt Status Enable or Error Interrupt Status Enable register is 1,the interrupt becomes active its active state is latched and made available to the Host Driver in the **Normal Interrupt Status** **register** or the **Error Interrupt Status register**;

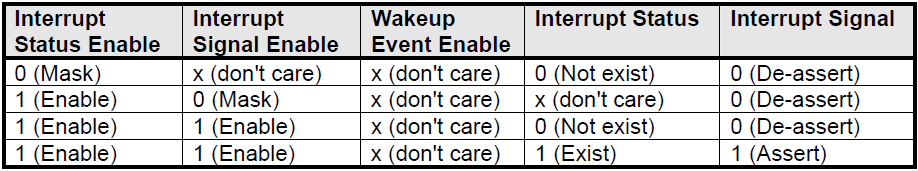
1. clear interrupt

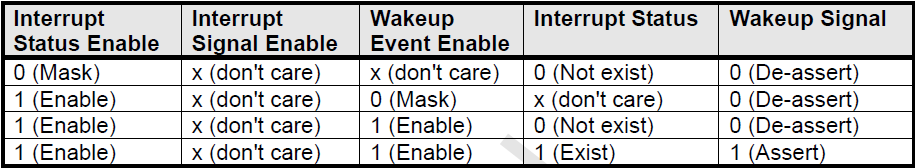
Interrupt Status shall be cleared when **Interrupt Status Enable** is cleared.

1. Interrupt process
2. Signal Assert

An interrupt source with its bit **set** in an **interrupt status register** shall **assert a** **system interrupt signal** if its corresponding bit is also set in the ***Normal Interrupt Signal Enable***register or the ***Error Interrupt Signal Enable***register.

Once signaled, most interrupts are **cleared** by writing a 1 to the associated bit in the **interrupt status register**.





1. Card interrupts Process

Card interrupts,howerver, must be cleared by the Card Driver.

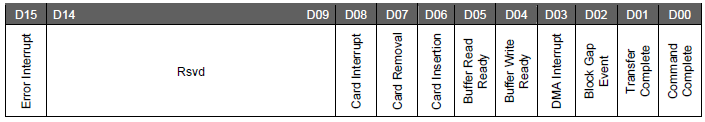
If the Card Interrupt is generated, the Host Driver shall clear **Card Interrupt Status Enable** before the Card Driver clears interrupt sources.

1. Assert others

After all interrupt sources are cleared, the Host Driver shall set **interrupt status register** again to enable another card interrupt.

1. register spec
2. Normal Interrupt Status Register (Offset 030h)

An interrupt is generated when the Normal Interrupt Signal Enable is enabled and at least one of the status bits is set to 1. For all bits except Card Interrupt and Error Interrupt, writing 1 to a bit clears it; writing to 0 keeps the bit unchanged. More than one status can be cleared with a single register write. The Card Interrupt is cleared when the card stops asserting the interrupt; that is, when the Card Driver services the interrupt condition.



1. Error Interrupt Status Register (Offset 032h)

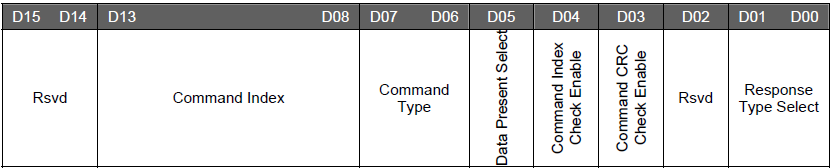
Signals defined in this register can be enabled by the Error Interrupt Status Enable register, but not by the Error Interrupt Signal Enable register. The interrupt is generated when the Error Interrupt Signal Enable is enabled and at least one of the statuses is set to 1. Writing to 1 clears the bit and writing to 0 keeps the bit unchanged. More than one status can be cleared at the one register write.



### Standard CMD

1. Command Register (Offset 00Eh)

The Host Driver shall check the **Command Inhibit (DAT)** bit and **Command Inhibit (CMD)** bit in the *Present State* register before writing to this register. Writing to the upper byte of this register triggers SD command generation. The Host Driver has the responsibility to write this register because the Host Controller does not protect for writing when **Command Inhibit (CMD)** is set.



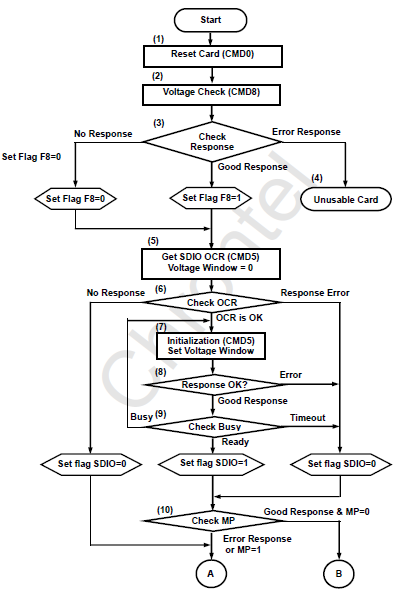
|  |  |  |
| --- | --- | --- |
| Location | Attrib | Register Field |
| 15-14 | Rsvd | Reserved |
| 13-08 | RW | Command Index These bits shall be set to the command number (CMD0-63, ACMD0-63) that is specified in bits 45-40 of the Command-Format in the Physical Layer Specification and SDIO Card Specification. |
| 07-06 | RW | |  |  |  | | --- | --- | --- | | 11b | Abort | CMD12, CMD52 for writing "I/O Abort" in CCCR | | 10b | Resume | CMD52 for writing "Function Select" in CCCR | | 01b | Suspend | CMD52 for writing "Bus Suspend" in CCCR | | 00b | Normal | Other commands | |
| 05 | RW | Data Present Select  This bit is set to 1 to indicate that data is present and shall be transferred using the DAT line. It is set to 0 for the following:  (1) Commands using only CMD line (ex. CMD52).  (2) Commands with no data transfer but using busy signal on DAT[0] line  (R1b or R5b ex. CMD38)  (3) Resume command |
| 04 | RW | Command Index Check Enable If this bit is set to 1, the Host Controller shall check the Index field in the response to see if it has the same value as the command index. If it is not, it is reported as a Command Index Error. If this bit is set to 0, the Index field is not checked. |
| 03 | RW | Command CRC Check Enable  If this bit is set to 1, the Host Controller shall check the CRC field in the  response. If an error is detected, it is reported as a Command CRC Error. If this bit is set to 0, the CRC field is not checked. The position of CRC field is  determined according to the length of the response. (Refer to definition in  D01-00 and Table 2-10 below.) |
| 02 | Rsvd | Reserved |
| 01-00 | RW | Respond Type Select   |  |  | | --- | --- | | 00 | No Response | | 01 | Response Length 136 | | 10 | Response Length 48 | | 10 | Response Length 48 check Busy after response | |

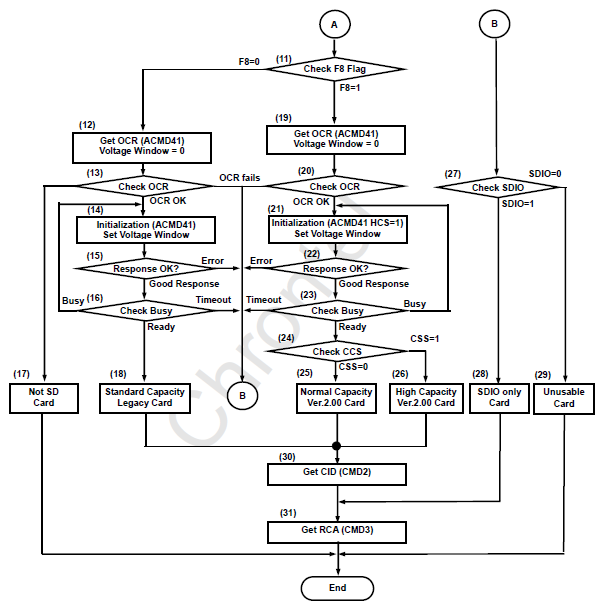
1. Auto CMD12

Multiple block transfers for SD memory require CMD12 to stop the transactions. The Host Controller automatically issues CMD12 when the last block transfer is completed. This feature of the Host Controller is called Auto CMD12. The Host Driver should set Auto CMD12 Enable in the Transfer Mode register when issuing a multiple block transfer command. Auto CMD12 timing synchronization with the last data block shall be done by hardware in the Host Controller. Commands that do not use the DAT line can be issued during multiple block transfers. These commands are referred to using the notation CMD\_wo\_DAT.

In order to prevent DAT line commands and CMD\_wo\_DAT commands from conflicting, the Host Controller shall arbitrate the timing by which each command is issued on the SD Bus. Therefore, a command might not immediately be issued after the Host Driver writes to the Command register. The command may be issued before or after Auto CMD12, depending on the timing. To be able to distinguish the responses of DAT line and CMD\_wo\_DAT commands, the Auto CMD12 response can be determined from the upper four bytes of the Response register (at offset 01Ch in the standard register set).

### Card Initialization and Identification

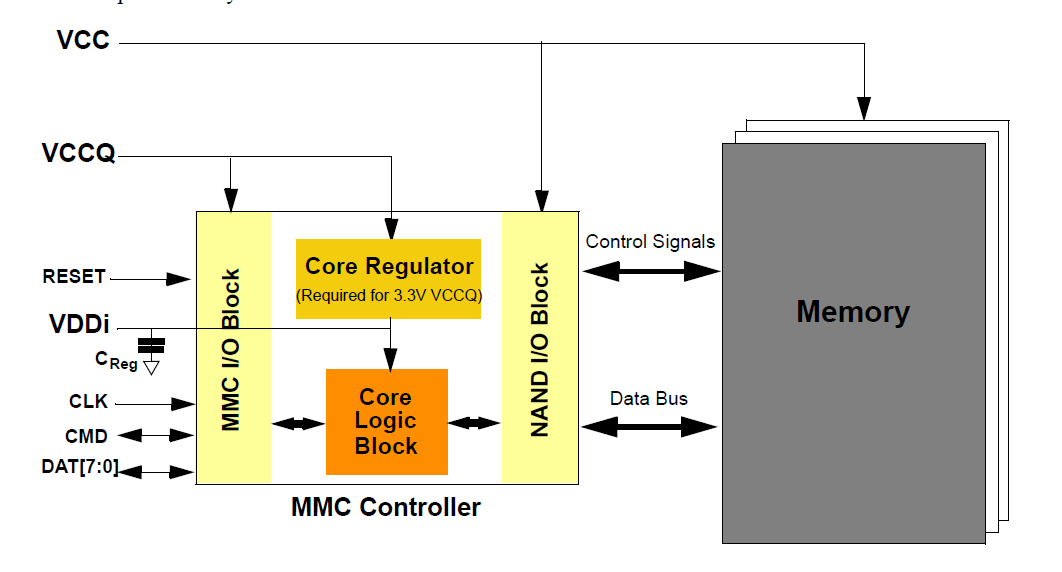




## eMMC

### 设备结构

eMMC = NAND Flash + Controller + Standard Interface

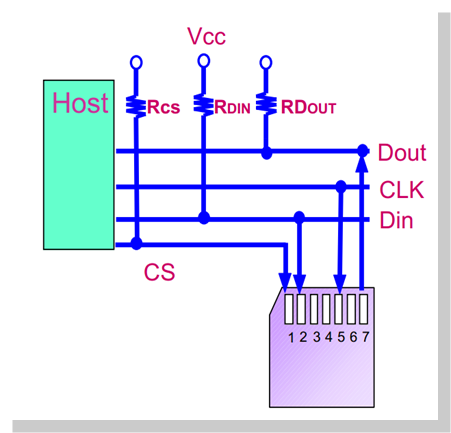
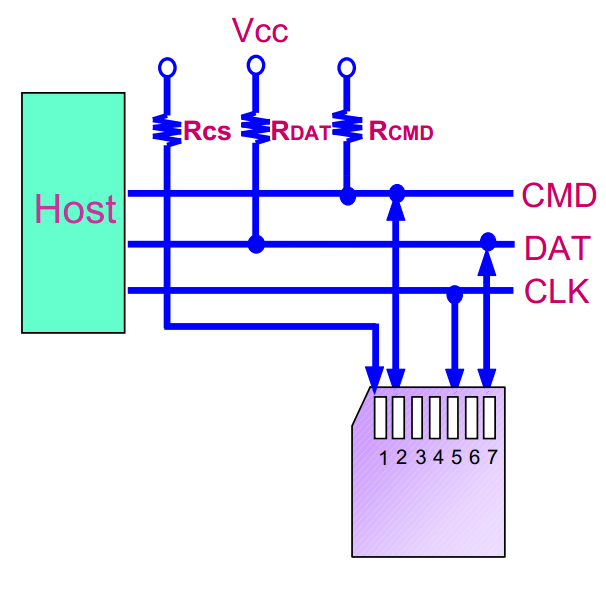


### 设备特性

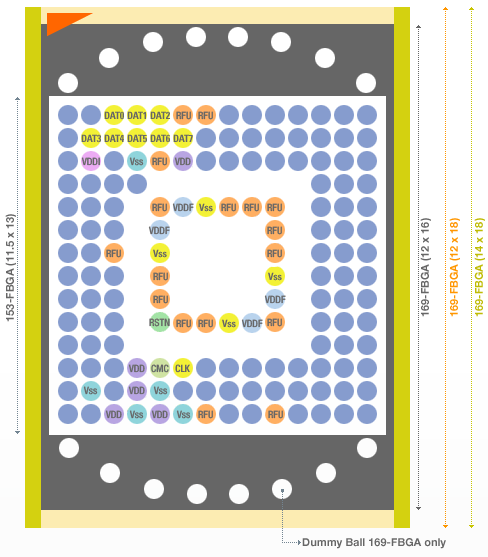
MMC指接口协议，eMMC(embeded MMC)是一种实现了MMC协议的芯片。

1. 设备接口

共有7个引脚，分为MMC和SPI两种工作模式。各模式对应的引脚定义如下：

1. 设备优点
2. 包括存储以及取代NOR Flash的开机功能
3. 不需处理NAND Flash相容性和管理问题
4. 缩短新产品品的上市周期和研发成本
5. 封装

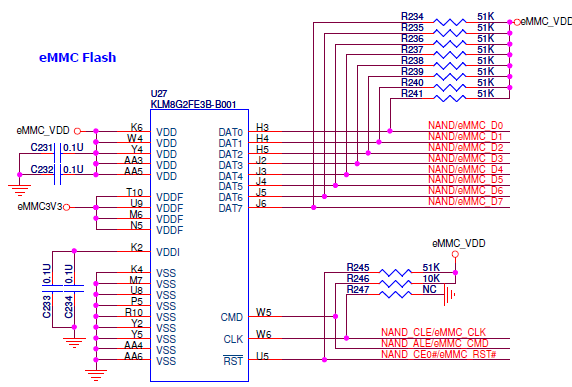


1. 设备厂家

|  |  |
| --- | --- |
| 厂家 | 主页 |
| 20090915111512390625[1] | www.samsung.com |
| 20090915111856343750[1] | www.sandisk.com |
| 20090915122348906250[1] | www.kingston.com/cn |
| 20090915111449468750[1] | ww.toshiba.com |
| 20090915111810687500[1] |  |
| 20090915111437625000[1] |  |
| 20090915111839328125[1] |  |

最高clock:52MHz

### IO规格



### Combo Socket

# 图像设备

## 图像标准

### YUV Color Space

1. 原理

The human eye was more sensitive to black-and-white picture information than color, YUV algorithm was key to simplifying and reducing the initial picture information sampled, saving valuable space.

The "Y" was luminance, or the detail of the video picture, in black-and-white; The "UV" was a math algorithm of R-Y (Red minus Luminance) and B-Y (Blue minus Luminance).



1. 模拟和数字的表示区别

Y'UV is often used as the term for YCbCr. However, they are different formats. Y'UV (Y'PbPr)is an analog system with scale factors different from the digital Y'CbCr system.

1. 常见的YUV格式

|  |  |  |  |
| --- | --- | --- | --- |
| sample | Byte/px | packed | planar |
| YUV444 | 3 | IYU2 | NV24 |
| YUV422 | 2 | UYVY ,IUYV | YV16,NV16 |
| YUV411 | 1.5 | IYU1,Y41P | Y41B |
| YUV420 | 1.5 |  | NV12,YV12,IM1,IYUV |

1. packed formats

将Y、U、V值储存成Macro Pixels阵列，和RGB的存放方式类似。对于YUV4:4:4格式，用紧缩格式很合适的，因此有UYVY、YUYV等。

1. planar formats

将Y、U、V的三个分量分别存放在3个不同的矩阵中。其中SP(Semi-Planar)格式指的是YUV分成2个平面。Y数据一个平面，UV数据合用一个平面。

### Resolution

解析度(分辨率) 用于衡量图像细节表现力，衡量标准为：一幅图像其水平方向和垂直方向所使用的点数。点越多即图像解析度越高，其质感就越强。图像Resolution的标准可以分为输入(捕获)和输出(显示)两类：

1. 设备分辨率

设备分辨率使用每英寸的成像点数DPI（Dots Per Inch）来衡量。比如电视显示器的分辨率。

1. DPI计算方法

显示器出厂时并不提供DPI参数，可以通过点距参数计算出DPI。

比如点距为0.28mm的显示器分辨率=25.3995mm/inch÷0.28mm/Dot≈90DPI（1 inch=25.3995mm）

1. 最高显示模式计算

最高显示模式指用一个点（Dot）表示一个像素（pixel）时设备的分辨率。通过显示器的大小和DPI可获得。

比如90DPI的14英寸(假设有效显示范围的对角线长度为11.5英寸) 显示器，根据勾股定理，可得，水平宽度=11.5÷5×4=9.2英寸，垂直高度为11.5÷5×3=6.8英寸。则最高显示模式约为：800（9.2×90）×600（6.8×90），

1. 常见设备分辨率

对于常见的输出设备电视, 其分辨率单位为TV Line，其值等价于垂直方向的点数，TV Line乘以宽高比(4/3或16/9)则为水平方向的点数。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **格式** | **等效格式** | **TV Line (可见/总数)** | **宽高 比例** | **分辨率** | **场频 (Hz)** | **行频 (kHz)** |
| D1@NTSC | 480i | 483/525 | 4:3,16:9 | 720x486 | 60 | 15.25 |
| D1@PAL/SECAM | 576i/625i | 576/625 | 4:3,16:9 | 720x576 | 50 | 15.625 |
| D2(逐行扫描DVD) | 480p | 480/525 | 4:3,16:9 | 640x480 | 60 | 31.5 |
| D3 | 1080i | 1080/1125 | 16:9 | 1920x1080 | 60 | 33.75 |
| D4 | 720p | 720/750 | 16:9 | 1280x720 | 60 | 45 |
| D5 | 1080p | 1080/1125 | 16:9 | 1920x1080 | 60 | 67.5 |

其中：

PAL(Phase Alteration Line)

NTSC(National Television Standards Committee)

不同分辨率对电视机处理能力(例如带宽)的要求：

480i<480p<1080i<720p。

1. 图像捕获分辨率

图像捕获分辨率使用每英寸捕获的像素数个数PPI（Pixels Per Inch）来衡量。比如DVR(Digital Video Recorder)的分辨率。

1. 常见图像分辨率

|  |  |  |
| --- | --- | --- |
| **Width [px]** | **Height [px]** | **Acronym** |
| 160 | 120 | QSIF |
| 176 | 144 | QCIF(Quarter CIF) |
| 320 | 240 | QVGA (also SIF) |
| 352 | 288 | CIF(Common Intermediate Format) |
| 640 | 480 | VGA(Video Graphics Array) |
| 1024 | 768 | XGA |
| 1280 | 960 | SXGA (4:3) |
| 1280 | 1024 | SXGA (5:4) |
| 1280 | 720 | HD(720p,High Definition) |
| 1920 | 1080 | FHD(1080p,Full HD) |
| 2560 | 1440 | QHD(2K,Quad HD) |
| 3840 | 2160 | QFHD,UHD(4K,Quad FHD,Ultra HD) |

1. 分辨率标准区别

|  |  |  |  |
| --- | --- | --- | --- |
| **格式** | **时间** | **标准化组织** | **应用领域** |
| D1 | 1986年 | 电影电视工程师协会(SMPTE) | 广播电视、电影、录像 |
| CIF | 1990年 | 国际电信联盟(ITU) | 电信、网络视频 |
| SIF | 1992年 | 动态图像专家组(MPEG) | 多媒体 |

### Definition

清晰度是指人眼宏观看到的图像的清晰程度，是图像输出系统的客观性能对图像接收者产生的图像感知效果。

1. Definition是边界锐度、层间微反差和质感的共同作用的结果。
2. Resolution只能影响图像的质感，但不是图像Definition的决定因素。
3. 同Resolution比较， Definition更象是一种心理学范畴。
4. 可以用黑白相间的线条的粗细来衡量Definition，使用标准的测试方法和测试图进行测量，其单位为TVL(电视线)。

## 图像输入

camera用于图像输入。

### Image Sensor属性

1. ISO (sensitivity)

ISO is the number indicating a digital camera sensors sensitivity to light. The higher the sensitivity, the less light is needed to make an exposure.

ISO数值越高就说明该感光元器件的感光能力越强。ISO的计算公式为 H\*S=0.8(S感光度，H为曝光量)，从公式中我们可以看出，感光度越高，对曝光量的要求就越少。变形公式:H=0.8/s，相同曝光量的前提下，iso50时的曝光时间为iso100时的曝光时间的两倍。常用的ISO值有50、 100 、200、400 、1000等，iso50,iso100在光线充足的情况使用，而高iso值在光线不足的情况下使用。

一般情况下，iso值越低，相片的质量越高，相片的细节表现的得越细腻，iso值越高，相片的亮度就越高，而相片的质量会随着iso值的升高而降低，噪点会变得越来越严重，但高iso值可以弥补光线的不足。

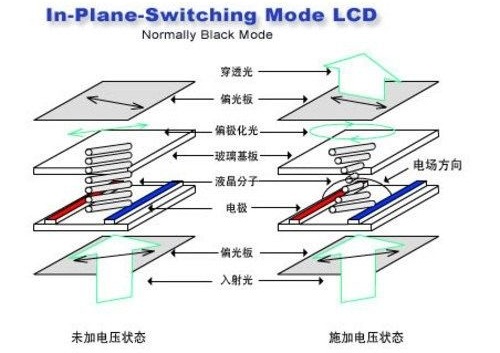
## 图像输出

### 显示设备

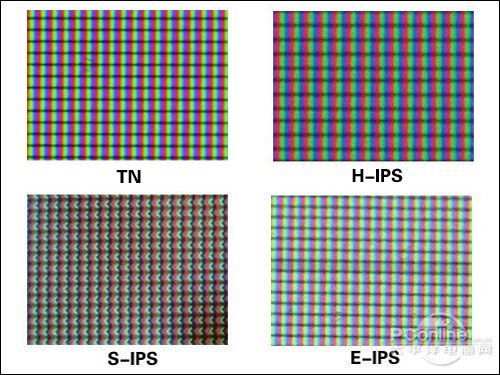
1. IPS 屏

IPS(In-Plane Switching，平面转换)技术是日立公司于2001推出的液晶面板技术，俗称“Super TFT”。

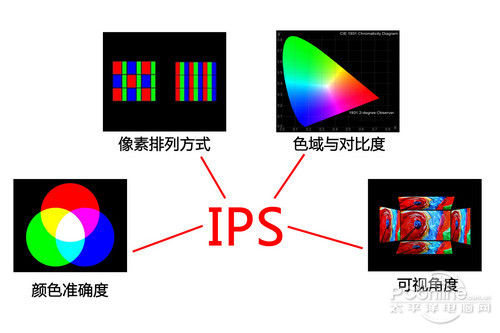
针对IPS屏幕是什么意思的问题，IPS屏幕的优势是可视角度高、响应速度快，色彩还原准确，是液晶面板里的高端产品。而且相比PVA面板，采用了IPS屏的LCD电视机动态清晰度能够达到780线。而静态清晰度方面，按照720线的高清标准要求仍能达到高清。该面板技术增强了LCD电视的动态显示效果，在观看体育赛事、动作片等运动速度较快的节目时能够获得更好的画质。和其他类型的面板相比，IPS面板用手轻轻划一下不容易出现水纹样变形，因此又有硬屏之称。仔细看屏幕时，如果看到是方向朝左的鱼鳞状象素，加上硬屏的话，那么就可以确定是IPS屏幕了。

[](http://www.pconline.com.cn/display_pic/1111/2579049_pic.html?imgsrc=http://img0.pconline.com.cn/pconline/1306/20/2579049_1928611.jpg&channel=8967)

IPS屏幕最大的特点就是它的两极都在同一个面上，而不象其它液晶模式的电极是在上下两面，立体排列。该技术把液晶分子的排列方式进行了优化，采取水平排列方式，当遇到外界压力时，分子结构向下稍微下陷，但是整体分子还呈水平状。在遇到外力时，硬屏液晶分子结构坚固性和稳定性远远优于软屏!所以不会产生画面失真和影响画面色彩，可以最大程度的保护画面效果不被损害。此外还有一种S-IPS面板属于IPS的改良型。

[](http://www.pconline.com.cn/display_pic/1111/2579049_pic.html?imgsrc=http://img0.pconline.com.cn/pconline/1306/20/2579049_137668449.jpg&channel=8967)

IPS硬屏技术的液晶面板不仅在耐用性和易清洁方面具备一定优势，更在功能参数和性能指标方面领先于VA软屏。响应时间一直是液晶电视的短板，但随着液晶技术的不断进步，液晶电视的画面拖尾问题在IPS面板上得到很好的解决。响应时间短的优势，使IPS屏幕能够在动态画面中显示出更加清晰的影像。另外，IPS屏幕还较VA软屏具有色彩还原准确、黑色的表现力突出等性能优点。

[](http://www.pconline.com.cn/display_pic/1111/2579049_pic.html?imgsrc=http://img0.pconline.com.cn/pconline/1306/20/2579049_20130613013025695.jpg&channel=8967)

## MALI

ARM® Mali™ 系列 GPU 为ARM公司设计的嵌入式图形 IP。其支持的 API 包括 Khronos™ OpenVG® 1.1、OpenGL® ES 1.1 和 2.0、OpenCL™ 以及 Microsoft® DirectX®。

# 设备总线

## JTAG

### 接口规范

JTAG是英文“Joint Test Action Group（联合测试行为组织）”的词头字母的简写，该组织成立于1985 年，是由几家主要的电子制造商发起制订的PCB 和IC 测试标准。JTAG 建议于1990 年被IEEE 批准为IEEE1149.1-1990 测试访问端口和边界扫描结构标准。该标准规定了进行边界扫描所需要的硬件和软件。自从1990 年批准后，IEEE 分别于1993 年和1995 年对该标准作了补充，形成了现在使用的IEEE1149.1a-1993 和IEEE1149.1b-1994。JTAG 主要应用于：电路的边界扫描测试和可编程芯片的在线系统编程。

JTAG最初是用来对芯片进行测试的,基本原理是在器件内部定义一个TAP（Test Access Port测试访问口）通过专用的JTAG测试工具对内部节点进行测试。JTAG测试允许多个器件通过JTAG接口串联在一起，形成一个JTAG链，能实现对各个器件分别测试。

在硬件结构上，JTAG 接口包括两部分：JTAG 端口和控制器。与JTAG 接口兼容的器件可以是微处理器（MPU）、微控制器（MCU）、PLD、CPL、FPGA、DSP、ASIC 或其它符合IEEE1149.1 规范的芯片。IEEE1149.1 标准中规定对应于数字集成电路芯片的每个引脚都设有一个移位寄存单元，称为边界扫描单元BSC。它将JTAG 电路与内核逻辑电路联系起来，同时隔离内核逻辑电路和芯片引脚。由集成电路的所有边界扫描单元构成边界扫描寄存器BSR。边界扫描寄存器电路仅在进行JTAG 测试时有效，在集成电路正常工作时无效，不影响集成电路的功能。

现今多数的高级器件都支持JTAG协议,如[DSP](http://baike.baidu.com/view/1192.htm)、FPGA、ARM、部分单片机器件等。标准的[JTAG接口](http://baike.baidu.com/view/1231341.htm)是4线:

[TMS](http://baike.baidu.com/view/1163589.htm)、[TCK](http://baike.baidu.com/view/160759.htm)、[TDI](http://baike.baidu.com/view/242148.htm)、[TDO](http://baike.baidu.com/view/3742112.htm),分别为模式选择、[时钟](http://baike.baidu.com/view/554890.htm)、[数据输入](http://baike.baidu.com/view/1309823.htm)和[数据输出](http://baike.baidu.com/view/324739.htm)线。 相关JTAG[引脚](http://baike.baidu.com/view/641241.htm)的定义为：TCK为测试时钟输入；TDI为测试数据输入，数据通过TDI引脚输入JTAG接口；TDO为测试[数据输出](http://baike.baidu.com/view/324739.htm)，数据通过TDO引脚从JTAG接口输出；TMS为测试模式选择，TMS用来设置JTAG接口处于某种特定的测试模式；TRST为测试复位，输入引脚，低电平有效

边界扫描原理：

IEEE 1149.1 标准规定了一个四线串行接口（第五条线是可选的），该接口称作测试访问端口（TAP），用于访问复杂的集成电路（IC），例如微处理器、DSP、ASIC和CPLD。除了TAP之外，混合IC也包含移位寄存器和状态机，以执行边界扫描功能。在TDI（测试数据输入）引线上输入到芯片中的数据存储在指令寄存器中或一个数据寄存器中。串行数据从TDO（测试数据输出）引线上离开芯片。边界扫描逻辑由TCK（测试时钟）上的信号计时，而且TMS（测试模式选择）信号驱动TAP控制器的状态。TRST（测试重置）是可选项。根据相关数据手册中的说明，TRST、TDI、TMS引脚上需要接一个10KΩ的上拉电阻，而TCK需要接一个10KΩ的下拉电阻。

在PCB上可串行互连多个可兼容扫描功能的IC，形成一个或多个扫描链，每一个链都由其自己的TAP。每一个扫描链提供电气访问，从串行TAP接口到作为链的一部分的每一个IC上的每一个引线。在正常的操作过程中，IC执行其预定功能，就好像边界扫描电路不存在。但是，当为了进行测试或在系统编程而激活设备的扫描逻辑时，数据可以传送到IC中，并且使用串行接口从IC中读取出来。这样数据可以用来激活设备核心，将信号从设备引线发送到PCB上，读出PCB的输入引线并读出设备输出。

1 JTAG(Joint Test Action Group;联合测试行动小组)是一种国际标准测试协议（IEEE 1149.1兼容），主要用于芯片内部测试。现在多数的高级器件都支持JTAG协议，如DSP、FPGA器件等。标准的JTAG接口是4线：TMS、 TCK、TDI、TDO，分别为模式选择、时钟、数据输入和数据输出线。

JTAG最初是用来对芯片进行测试的，JTAG的基本原理是在器件内部定义一个TAP（Test Access Port;测试访问口）通过专用的JTAG测试工具对进行内部节点进行测试。JTAG测试允许多个器件通过JTAG接口串联在一起，形成一个JTAG链，能实现对各个器件分别测试。现在，JTAG接口还常用于实现ISP（In-System Programmable&#0;在线编程），对FLASH等器件进行编程。

JTAG编程方式是在线编程，传统生产流程中先对芯片进行预编程现再装到板上因此而改变，简化的流程为先固定器件到电路板上，再用JTAG编程，从而大大加快工程进度。JTAG接口可对PSD芯片内部的所有部件进行编程

具有JTAG口的芯片都有如下JTAG引脚定义：

TCK——测试时钟输入；

TDI——测试数据输入，数据通过TDI输入JTAG口；

TDO——测试数据输出，数据通过TDO从JTAG口输出；

TMS——测试模式选择，TMS用来设置JTAG口处于某种特定的测试模式。

可选引脚TRST——测试复位，输入引脚，低电平有效。

含有JTAG口的芯片种类较多，如CPU、DSP、CPLD等。

JTAG内部有一个状态机，称为TAP控制器。TAP控制器的状态机通过TCK和TMS进行状态的改变，实现数据和指令的输入。图1为TAP控制器的状态机框图。

2 JTAG芯片的边界扫描寄存器

JTAG 标准定义了一个串行的移位寄存器。寄存器的每一个单元分配给IC芯片的相应引脚，每一个独立的单元称为BSC（Boundary-Scan Cell）边界扫描单元。这个串联的BSC在IC内部构成JTAG回路，所有的BSR（Boundary-Scan Register）边界扫描寄存器通过JTAG测试激活，平时这些引脚保持正常的IC功能。图2为具有JTAG口的IC内部BSR单元与引脚的关系。

3 JTAG在线写Flash的硬件电路设计和与PC的连接方式

以含JTAG接口的StrongARM SA1110为例，Flash为Intel 28F128J32 16MB容量。SA1110的JTAG的TCK、TDI、TMS、TDO分别接PC并口的2、3、4、11线上，通过程序将对JTAG口的控制指令和目标代码从PC的并口写入JTAG的BSR中。在设计PCB时，必须将SA1110的数据线和地址线及控制线与Flash的地线线、数据线和控制线相连。因 SA1110的数据线、地址线及控制线的引脚上都有其相应BSC，只要用JTAG指令将数据、地址及控制信号送到其BSC中，就可通过BSC对应的引脚将信号送给Flash，实现对Flash的操作。JTAG的系统板设计和连线关系如图3所示。

4 通过使用TAP状态机的指令实行对Flash的操作

通过TCK、TMS的设置，可将JTAG设置为接收指令或数据状态。JTAG常用指令如下：

SAMPLE/PRELOAD——用此指令采样BSC内容或将数据写入BSC单元；

EXTEST——当执行此指令时，BSC的内容通过引脚送到其连接的相应芯片的引脚；

BYPASS——此指令将一个一位寄存器轩于BSC的移位回路中，即仅有一个一位寄存器处于TDI和TDO之间。

在PCB电路设计好后，即可用程序先将对JTAG的控制指令，通过TDI送入JTAG控制器的指令寄存器中。再通过TDI将要写Flash的地址、数据及控制线信号入BSR中，并将数据锁存到BSC中，用EXTEST指令通过BSC将写入Flash。

jtag接口中，必要的信号是TMS, TCK, TDI, TDO， TRST和SRST是可选的信号，其中TRST用来复位TAP控制器，SRST用来复位目标CPU。

1. 上位机软件

OpenOCD(On chip Debug)是一个开源的JTAG上位机程序.

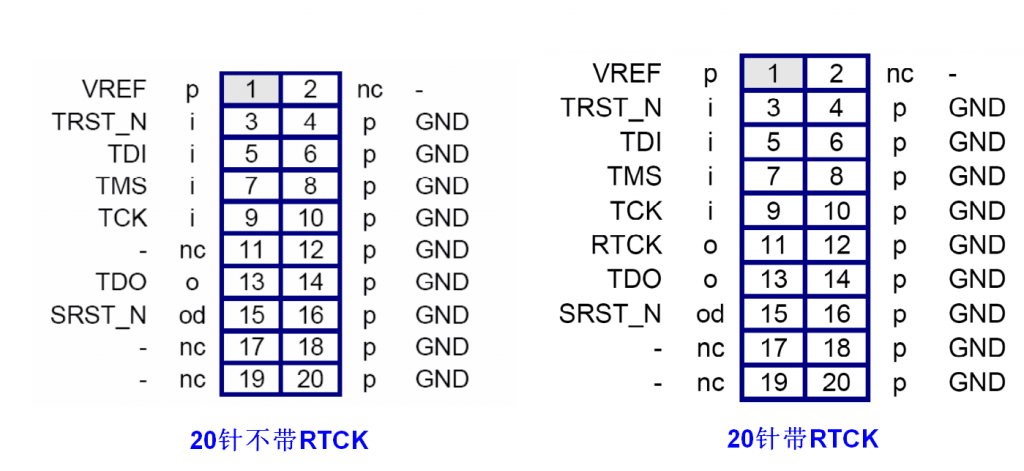
对openocd来说，最理想的情况是，它能够分别控制TRST和SRST两个信号。可以先设置SRST并保持住，让目标CPU保持在复位的状态，然后用TRST信号复位TAP控制器，最后再取消SRST信号，这样就可以让目标CPU复位后就处于debug状态。这样的前提是，目标CPU和jtag小板都要支持这两个信号。

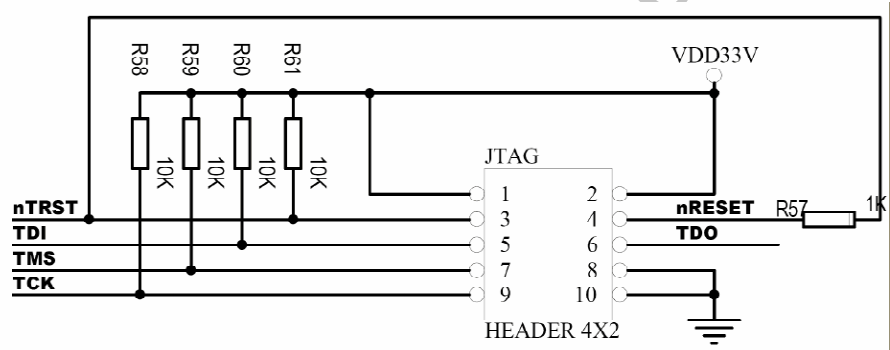
jlink使用一块可编程芯片实现了tap状态机。

JTAG有10pin的、14pin的和20pin的，尽管引脚数和引脚的排列顺序不同，但是其中有一些引脚是一样的，各个引脚的定义如下。

1. **引脚定义**

**20、**14**、10pin JTAG的引脚名称与序号对应关系**

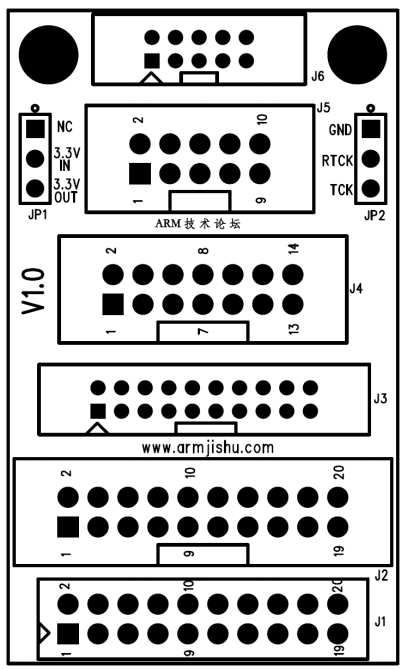
值得注意的是，不同的IC公司会自己定义自家产品专属的Jtag头，来下载或调试程序。嵌入式系统中常用的20、14、10pin JTAG的信号排列如下：



需要说明的是，上述Jtag头的管脚名称是对IC而言的。例如TDI脚，表示该脚应该与IC上的TDI脚相连，而不是表示数据从该脚进入download cable。

实际上10针的只需要接4根线，4号是自连回路，不需要接，1，2接的都是1管脚，而8，10接的是GND，也可以不接。

 附转接板电路：



### 接口信号

IEEE 1149.1标准定义了标准的JTAG接口的信号管脚：

IEEE 1149.1标准

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号 | | | | 功能 |
| 强制要求 | | | | |
| TCK | Test Clock Input | | TCK为TAP的操作提供了一个独立的、基本的时钟信号，TAP的所有操作都是通过这个时钟信号来驱动的 | |
| TMS | Test Mode Selection Input | | TMS信号在TCK的上升沿有效，用来控制TAP状态机的转换。实现TAP在不同的状态间相互转换。 | |
| TDI | Test Data Input | | 所有要输入到特定寄存器的数据都是通过TDI接口一位一位串行输入的（由TCK驱动）。 | |
| TDO | Test Data Output | | 所有要从特定的寄存器中输出的数据都是通过TDO接口一位一位串行输出的（由TCK驱动）。 | |
| VTREF | Voltage Reference | | 接口信号电平参考电压，一般直接连接Vsupply。用来确定ARM的JTAG接口使用的逻辑电平（比如3.3V还是5.0V） | |
| 非强制要求(可选项) | | | | |
| TRST | Test Reset Input | | TRST可以用来对TAPController进行复位（初始化）。因为通过TMS也可以对TAP Controll进行复位（初始化）。所以有四线JTAG与五线JTAG之分。 | |
| RTCK | Return Test Clock | | 目标端反馈给仿真器的时钟信号,用来同步TCK信号的产生,不使用时直接接地。 | |
| nSRST | System Reset | | 与目标板上的系统复位信号相连,可以直接对目标系统复位。同时可以检测目标系统的复位情况，为了防止误触发应在目标端加上适当的上拉电阻。 | |
| DBGRQ |  | | 目标板上工作状态的控制信号 | |
| 用户信号 | | | | |
| USER IN | | 用户自定义输入。可以接到一个IO上，用来接受上位机的控制 | | |
| USER OUT | | 用户自定义输出。可以接到一个IO上，用来向上位机的反馈一个状态 | | |

由于JTAG经常使用排线连接，为了增强抗干扰能力，在每条信号线间加上地线就出现了这种20针的接口。但事实上，**RTCK、USER IN、USER OUT**一般都不使用，于是还有一种14针的接口。对于实际开发应用来说，由于实验室电源稳定，电磁环境较好，干扰不大。

### TAP原理

是Test Access Port（测试访问端口）的缩写，是芯片内部一个通用的端口，通过TAP可以访问芯片提供的所有数据寄存器（DR）和指令寄存器（IR），对整个TAP的控制是通过TAP控制器（TAP Controller）完成的。

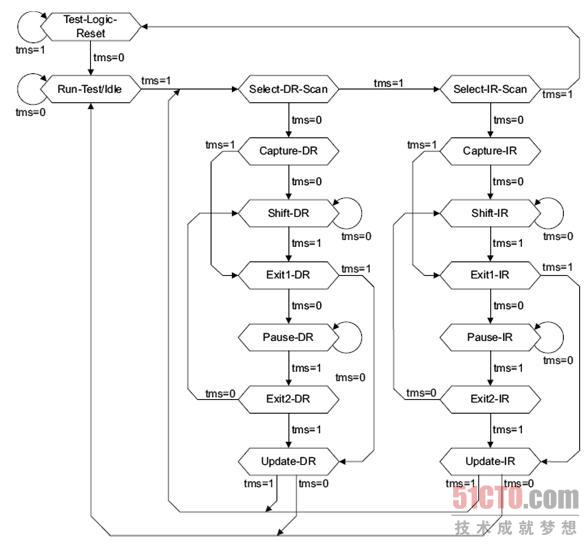
**边界扫描**

英文叫Boundary Scan，边界扫描的基本思想是在靠近芯片的输入输出管脚（PIN）上设置一个移位寄存器单元，也就是边界扫描寄存器（Boundary-Scan Register）。当芯片处于调试状态时，边界扫描寄存器可以将芯片和外部输入输出管脚隔离开来，通过边界扫描寄存器单元，可以实现对芯片外部输入输出管脚的观察和控制。对于芯片的输出管脚可以通过与之相连的边界扫描寄存器单元把信号（数据）加载到该引脚中去，对于芯片的输入管脚，也可以通过与之相连的边界扫描寄存器"捕获"该管脚上的输出信号。在正常的运行状态下，边界扫描寄存器对芯片来说是透明的，所以正常的运行不会受到任何影响，这样，边界扫描寄存器提供了一种便捷的途径用于观测和控制所需调试的芯片。另外，芯片管脚上的边界扫描（移位）寄存器单元可以相互连接起来，使芯片的周围形成一个边界扫描链（Boundary-Scan Chain），边界扫描链可以串行地输入和输出，通过相应的时钟信号和控制信号，就可以方便地观察和控制处在调试状态下的芯片。

### TAP状态机

TAP控制器有16个同步状态，控制器的下一个状态TMS信号决定，TMS信号在TCK的上升沿被采样生效。

图10-1列出了TAP控制器的16个同步状态转换机制。

[](http://new.51cto.com/files/uploadimg/20090226/111442122.jpg)

**Test-Logic-Reset测试逻辑复位状态**

处于这种状态下，测试逻辑被禁止以允许芯片正常操作，读IDCODE寄存器将禁止测试逻辑。

无论TAP控制器处于何种状态，只要将TMS信号在5个连续的TCK信号的上升沿保持高电平，TAP就将进入Test-Logic-Reset状态，如果TMS信号一直为高电平，那么TAP将保持在Test-Logic-Reset状态，另外TRST信号也可以强迫TAP进入Test- Logic-Reset状态。

处于Test-Logic-Reset状态的TAP，如果下一个TCK的上升沿时TMS信号处于低电平，那么TAP将被切换到Run-Test-Idle状态。

**Run-Test-Idle运行测试空闲状态**

Run-Test-Idle是TAP控制器扫描操作空闲状态，如果TMS信号一直处于低电平，那么TAP将保持在TRun-Test-Idle状态。当TMS信号在TCK上升沿处于高电平，TAP控制器将进入Select-DR-Scan状态。

**Select-DR-Scan选择数据寄存器扫描状态**

Select-DR-Scan是TAP控制器的一个临时状态，边界扫描寄存器BSR保持它们先前的状态。

当TMS信号在下一个TCK上升沿处于低电平，TAP控制器进入Capture-DR状态，一个边界扫描寄存器的扫描操作同时被初始化。

如果TMS信号在下一个TCK上升沿处于高电平，TAP控制器将进入Select-IR-Scan状态。

**Capture-DR捕获数据寄存器状态**

如果TAP控制器处于Capture-DR状态，且当前指令是SAMPLE/PRELOAD指令，那么边界扫描寄存器BSR在TCK信号的上升沿捕获输入管脚的数据。如果此时不是SAMPLE/PRELOAD指令，那么BSR保持它们先前的值，另外BSR的值被放入连接在TDI和TDO管脚之间的移位寄存器中。

处于Capture-DR状态时，指令不会被改变。

如果TMS信号在下一个TCK上升沿处于高电平，TAP进入Exit1-DR状态。如果TMS信号在下一个TCK上升沿处于低电平，则TAP进入Shift-DR状态。

**Shift-DR移位数据寄存器状态**

在Shift-DR状态下，在每个TCK的上升沿，TDI-移位寄存器-TDO串行通道向右移一位，TDI的数据移入移位寄存器，移位寄存器最靠近TDO的位移到TDO管脚上。

处于Shift-DR状态时，指令不会被改变。

如果TMS信号在下一个TCK上升沿处于高电平，TAP进入Exit1-DR状态。如果TMS信号处于低电平，则TAP一直进行移位操作。

**Exit1-DR退出数据寄存器状态1**

Exit1-DR是TAP控制器的一个临时状态，如果TMS信号在下一个TCK上升沿处于高电平，TAP进入Update-DR状态；如果TMS信号在下一个TCK上升沿处于低电平，则TAP进入Pause-DR状态。

处于Exit1-DR状态时，指令不会被改变。

**Pause-DR暂停数据寄存器状态**

Pause-DR状态允许TAP控制器暂时停止TDI-移位寄存器-TDO串行通道的移位操作。

处于Pause-DR状态时，指令不会被改变。

如果TMS信号在下一个TCK上升沿处于高电平，TAP进入Exit2-DR状态；如果TMS信号处于低电平，则TAP一直保持暂停状态。

**Exit2-DR退出数据寄存器状态2**

Exit2-DR也是TAP控制器的临时状态，如果TMS信号在下一个TCK上升沿处于高电平，TAP进入Update-DR状态，结束扫描操作；如果TMS信号在下一个TCK上升沿处于低电平，则TAP重新进入Shift-DR状态。

处于Exit2-D状态时，指令不会被改变。

**Update-DR更新数据寄存器状态**

在正常情况下，边界扫描寄存器BSR的值是被锁存在并行输出管脚中，以免在EXTEST或SAMPLE/PRELOAD命令下执行移位操作时改变 BSR的值。当处于Update-DR状态时选择的是BSR寄存器，那么移位寄存器中的值将在TCK的下降沿被锁存到BSR寄存器的并行输出管脚中去。

处于Update-DR状态时，指令不会被改变。

如果TMS信号在下一个TCK上升沿处于高电平，TAP进入Select-DR-Scan状态；如果TMS信号在下一个TCK上升沿处于低电平，则TAP进入Run-Test-Idle状态。

**Select-IR-Scan选择指令寄存器扫描状态**

Select-IR-Scan是TAP控制器的一个临时状态。

如果TMS信号在下一个TCK上升沿处于低电平，TAP控制器进入Capture-IR状态，一个对指令寄存器的扫描操作同时被初始化。

如果TMS信号在下一个TCK上升沿处于高电平，TAP控制器将进入Test-Logic-Reset状态。

处于Select-IR-Scan状态时，指令不会被改变。

**Capture-IR捕获指令寄存器状态**

处于Capture-IR状态时，指令寄存器中的值被固定设置成0b0000001，并将它放入连接在TDI与TDO之间的移位寄存器中。

处于Capture-DR状态时，指令不会被改变。

如果TMS信号在下一个TCK上升沿处于高电平，TAP进入Exit1-IR状态；如果TMS信号在下一个TCK上升沿处于低电平，则TAP进入Shift-IR状态。

**Shift-IR移位指令寄存器状态**

在Shift-IR状态下，在每个TCK的上升沿，TDI-移位寄存器-TDO串行通道向右移一位，JTAG指令从TDI管脚上被逐位移入移位寄存器，而移位寄存器中的0b0000001则被逐位从TDO管脚移出。

处于Shift-IR状态时，指令不会被改变。

如果TMS信号在下一个TCK上升沿处于高电平，TAP进入Exit1-IR状态；如果TMS信号处于低电平，则TAP一直进行移位操作。

**Exit1-IR退出指令寄存器状态1**

Exit1-IR是TAP控制器的一个临时状态，如果TMS信号在下一个TCK上升沿处于高电平，TAP进入Update-IR状态；如果TMS信号在下一个TCK上升沿处于低电平，则TAP进入Pause-IR状态。

处于Exit1-IR状态时，指令不会被改变。

**Pause-IR暂停指令寄存器状态**

Pause-IR状态允许TAP控制器暂时停止TDI-移位寄存器-TDO串行通道的移位操作。

处于Pause-IR状态时，指令不会被改变。

如果TMS信号在下一个TCK上升沿处于高电平，TAP进入Exit2-IR状态；如果TMS信号处于低电平，则TAP一直处于暂停状态。

**Exit2-IR退出指令寄存器状态2**

Exit2-IR也是TAP控制器的临时状态，如果TMS信号在下一个TCK上升沿处于高电平，TAP进入Update-IR状态，结束扫描操作；如果TMS信号在下一个TCK上升沿处于低电平，则TAP重新进入Shift-IR状态。

处于Exit2-D状态时，指令不会被改变。

**Update-IR更新指令寄存器状态**

处于Update-IR状态时，移位寄存器中的值将在TCK的下降沿被锁存到指令寄存器中，一旦锁存成功，新的指令将成为当前的指令。

如果TMS信号在下一个TCK上升沿处于高电平，TAP进入Select-DR-Scan状态；如果TMS信号在下一个TCK上升沿处于电平，则TAP进入Run-Test-Idle状态。

### 接口指令

JTAG接口指令集包含以下常用指令：

**EXTEST指令**

外部测试指令，必须全为0，TAP强制定义。该指令初始化外部电路测试，主要用于板级互连以及片外电路测试。

EXTEST指令在Shift-DR状态时将扫描寄存器BSR寄存器连接到TDI与TDO之间。在Capture-DR状态时，EXTEST指令将输入管脚的状态在TCK的上升沿装入BSR中。EXTEST指令从不使用移入BSR中的输入锁存器中的数据，而是直接从管脚上捕获数据。在Update- DR状态时，EXTEST指令将锁存在并行输出寄存器单元中的数据在TCK的下降沿驱动到对应的输出管脚上去。

**SAMPLE/PRELOAD指令**

采样/预装载指令，TAP强制定义。在Capture-DR状态下，SAMPLE/PRELOAD指令提供一个从管脚到片上系统逻辑的数据流快照，快照在TCK的上升沿提取。在Update-DR状态时，SAMPLE/PRELOAD指令将BSR寄存器单元中的数据锁存到并行输出寄存器单元中，然后由EXTEST指令将锁存在并行输出寄存器单元中的数据在TCK的下降沿驱动到对应的输出管脚上去。

**BYPASS指令**

旁路指令，必须全为1，TAP强制定义。BYPASS指令通过在TDI和TDO之间放置一个1位的旁通寄存器，这样移位操作时只经过1位的旁通寄存器而不是很多位（与管脚数量相当）的边界扫描寄存器BSR，从而使得对连接在同一JTAG链上主CPU之外的其他芯片进行测试时提高效率。

**IDCODE指令**

读取CPU ID号指令，TAP强制定义。该指令将处理器的ID号寄存器连接到TDI和TDO之间。

## IPC

Inter-Processor Communication

## USB

### Connector

1. USB2.0
2. 普通connector

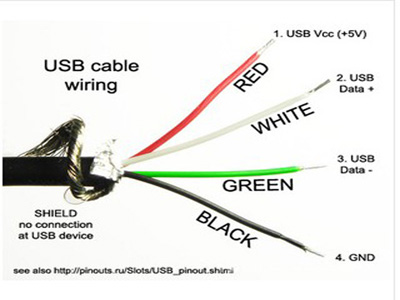


1. mini-usb connector

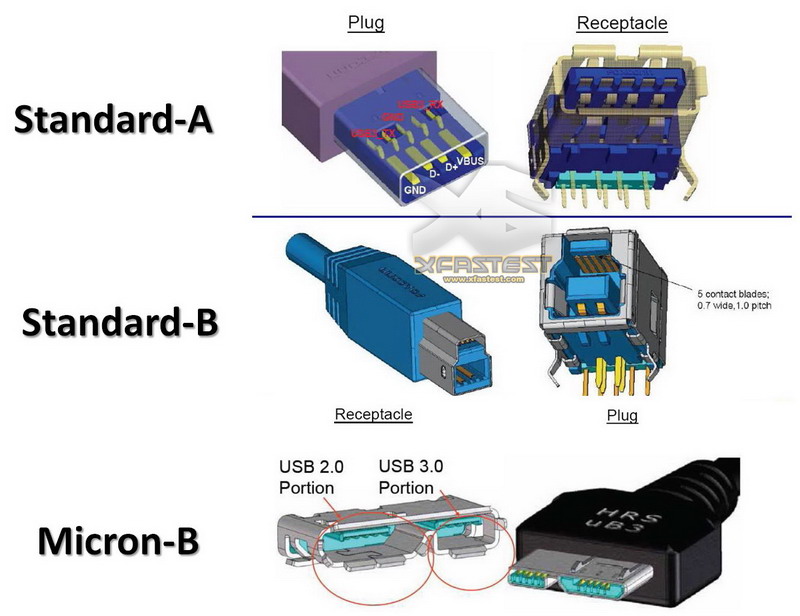


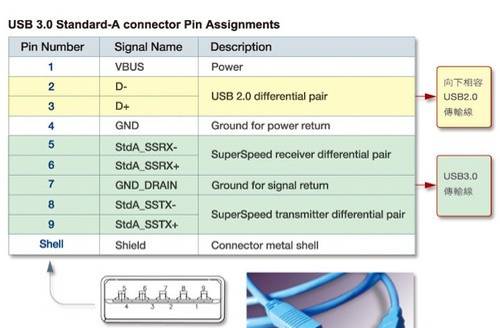
OTG(On-The-Go)用于在没有Host的情况下，实现从设备间的数据传送。如果ID脚是高电平，则是B接头插入，此时系统就做主模式(master mode)，如果ID为低电平，则是A接口插入，然后系统就会使用HNP对话协议来决定哪个做Master，哪个做Slave。

1. 传输线

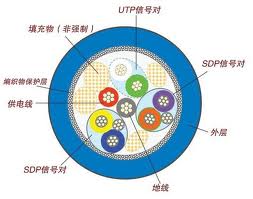


1. USB3.0
2. connector

****



1. 传输线



### Host Controller Interface

所有的USB设备的请求提交给usb host controller统一来调度，Host Controller Interface即USB设备和USB Controller的桥接器。

1. USB2.0 Full Speed(USB1.1): 12Mbps

UHCI(universal host controller interface): Intel芯片组上的usb 1.1主控制器(host controller).

OHCI(open host controller inferface)：遵循csr (configuration space register)标准。是其他厂商在设计usb host controller时遵循的标准，如via, nec, ali, 包括nvidia等。该标准不仅适用于usb主控制器接口标准，也包括IEEE1394 Interface Card(iLink/Firewire)等主控器接口标准。

1. USB2.0 High Speed: 480Mbps

EHCI (enhanced host controller interface)满足usb 2.0 specification里面对usb host controller (high speed)的硬件设计要求。

EHCI为了兼容USB 1.1，将OHCI或UHCI之一集成进自己的规范里 。这样会同时存在两个USB Host Controller，这种方式名叫Companion（伙伴）模式。

1. USB3.0: 4.8Gbps

XHCI（eXtensible Host Controller Interface）可扩展的主机控制器接口是英特尔公司开发的一个USB主机控制器接口。它目前主要是面向USB 3.0的，同时它也支持USB 2.0及以下的设备。

xHCI没有采用伙伴模式，它不再和以前的主机控制器规范兼容，一个规范支持所有的USB协议，即仅存在一个USB Host Controller。

## IIC

### 传输协议

I2C(Inter－Integrated Circuit)总线 I2C(Inter－Integrated Circuit)总线是一种由PHILIPS公司开发的两线式串行总线，用于连接微控制器及其外围设备。I2C总线产生于在80年代，最初为音频和视频设 备开发，如今主要在服务器管理中使用，其中包括单个组件状态的通信。例如管理员可对各个组件进行查询，以管理系统的配置或掌握组件的功能状态，如电源和系 统风扇。可随时监控内存、硬盘、网络、系统温度等多个参数，增加了系统的安全性，方便了管理。 参考资料：http://www.mcufan.com/xinshou/mcu-iic.htm I2C总线特点 I2C总线最主要的优点是其简单性和有效性。由于接口直接在组件之上，因此I2C总线占用的空间非常小，减少了电路板的空间和芯片管脚的数量，降低了互联成本。总线的长度可高达25英尺，并且能够以10Kbps的最大传输速率支持40个组件。I2C总线的另一个优点是，它支持多主控(multimastering)， 其中任何能够进行发送和接收的设备都可以成为主总线。一个主控能够控制信号的传输和时钟频率。当然，在任何时间点上只能有一个主控。 总线的构成及信号类型 I2C 总线是由数据线SDA和时钟SCL构成的串行总线，可发送和接收数据。

在CPU与被控IC之间、IC与IC之间进行双向传送，最高传送速率 100kbps。各种被控制电路均并联在这条总线上，但就像电话机一样只有拨通各自的号码才能工作，所以每个电路和模块都有唯一的地址，在信息的传输过程 中，I2C总线上并接的每一模块电路既是主控器（或被控器），又是发送器（或接收器），这取决于它所要完成的功能。CPU发出的控 制信号分为地址码和控制量两部分，地址码用来选址，即接通需要控制的电路，确定控制的种类；控制量决定该调整的类别（如对比度、亮度等）及需要调整的量。 这样，各控制电路虽然挂在同一条总线上，却彼此独立，互不相关。 I2C总线在传送数据过程中共有三种类型信号， 它们分别是：开始信号、结束信号和应答信号。 开始信号：SCL为高电平时，SDA由高电平向低电平跳变，开始传送数据。 结束信号：SCL为低电平时，SDA由低电平向高电平跳变，结束传送数据。 应答信号：接收数据的IC在接收到8bit数据后，向发送数据的IC发出特定的低电平脉冲，表示已收到数据。CPU向受控单元发出一个信号后，等待受控单 元发出一个应答信号，CPU接收到应答信号后，根据实际情况作出是否继续传递信号的判断。若未收到应答信号，由判断为受控单元出现故障

## SPI

## UART

### RS232

1. 流控制模式
2. DTR/DSR

硬件上要有对应接口，软件上实现对应协议，才能实现此流控制。一般和RTS/CTS一直搭配使用。

1. RTS/CTS

硬件上要有对应接口，软件上实现对应协议，才能实现此流控制。一般常见的也就是这种。

但是很多开发板用secureCRT连接开发板的时候，往往都是在Uboot阶段，那时候还没有实现对应的RTS/CTS的流控制，所以此处需要取消此选择，否则，secureCRT就会去一直检测对应的RTS（Request To Send）pin脚，发现一直是无效，所以就一直没有接受到数据，就“卡”在那了。同样，你要输入数据，就是secureCRT要发送数据，所以其先发了个RTS给开发板，然后就一直检测自己的CTS是否有效，直到自己的CTS有效后，才能发送数据，而此处由于开发板上的驱动没有实现RTS/CTS，所以secureCRT就一直检测CTS，就“卡”死了，即你无法输入数据。

1. XON/XOFF

软件上实现的流控制，硬件上无需像上面的RTS/CTS或DTR/DSR那样要增加对应的引脚。但是由于XON/XOFF分别对应的两个二进制的值，所以如果本身传输数据中包括该值，那么此软件的流控制就失效了。现在好像也很少用此种流控制了。

## CAN

# 有线网络

## 网络连接

### 以太网介绍

以太网(Ethernet)一种基带局域网规范，是当今局域网采用的最通用的通信协议标准。以太网络使用CSMA/CD（载波监听多路访问及冲突检测）技术，并以10M/S的速率运行在多种类型的电缆上。以太网与IEEE802.3系列标准相类似。

1. 局域网规范
2. 以太网（EtherNet)

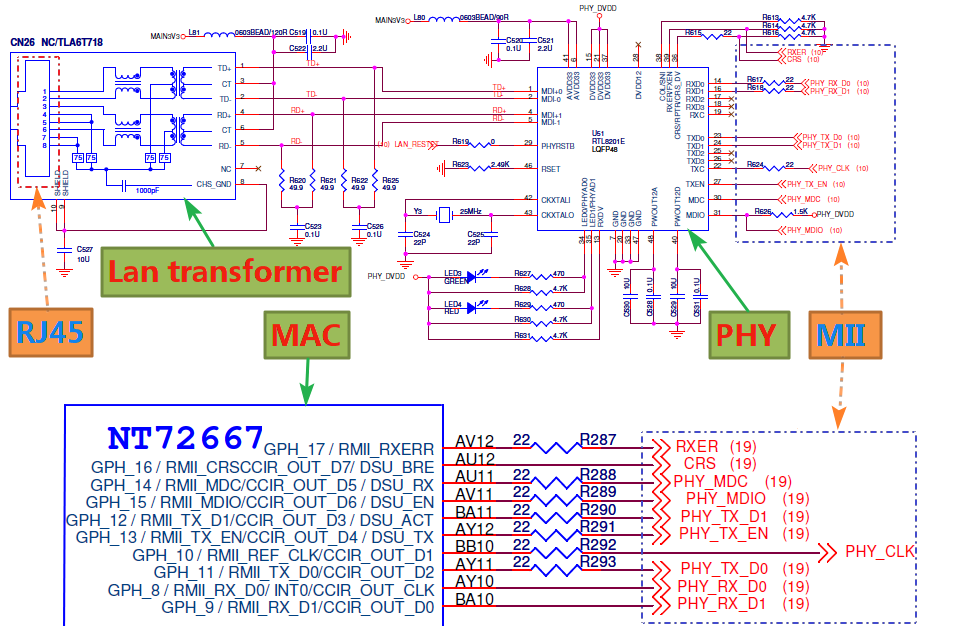
以太网最早由Xerox（施乐）公司创建，于1980年DEC、lntel和Xerox三家公司联合开发成为一个标准。以太网是应用最为广泛的局域网，包括标准的以太网（10Mbit/s)、快速以太网（100Mbit/s）和10G（10Gbit/s）以太网。它们都符合IEEE802.3。

1. IEEE802.3标准

IEEE802.3规定了包括物理层的连线、电信号和介质访问层协议的内容。以太网是当前应用最普遍的局域网技术，它很大程度上取代了其他局域网标准。如令牌环、FDDI和ARCNET。历经100M以太网在上世纪末的飞速发展后，千兆以太网甚至10G以太网正在国际组织和领导企业的推动下不断拓展应用范围。

|  |  |
| --- | --- |
| 常见的802.3应用 | |
| 应用类型 | 介质 |
| 10M | 铜线UTP模式 |
| 100M | 铜线UTP模式 |
| 100base-FX | 光纤线 |
| 1000M: 1000base-T | 铜线UTP模式 |

1. 设备连接



### 双绞线

1. 功能

双绞线对是电缆工程设计的一部分。当电线中有电流通过时，会产生电磁场，这会对电缆中的其他线对产生干扰。通过让线对中的每根电线的电流相反，并紧密排列可以使线对的电磁场相互抵消。电缆内部线对间的电磁干扰称为串扰。每个线对的双绞线密度不同，以使串扰降到最低。

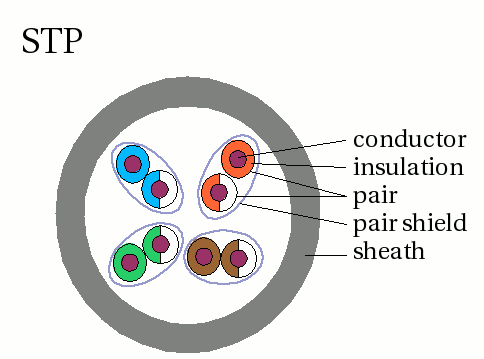
1. 分类

双绞线由四对不同颜色的传输线互相缠绕所组成，每对相同颜色的线传递著来回两方向的电脉冲，这样的设计是利用了电磁感应相互抵销的原理来屏蔽电磁干扰。

双绞线分为非屏蔽双绞线(Unshielded Twisted Pair，UTP)和屏蔽双绞线(Shielded Twisted Pair，STP)两大类。

1. STP

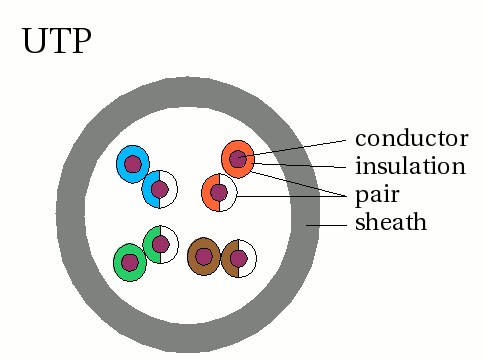
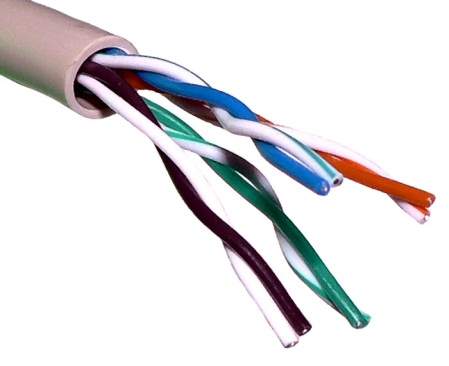
STP双绞线外的金属网（通常是铜质）可以进一步屏蔽传输线，使之不受外部电磁场干扰，以及减小对外辐射，防止信息被窃听，同时作为接地之用。但这种额外的保护结构降低了线材的弹性。这种线常用在以太网（局域网络）中，屏蔽双绞线额外的保护结构提高了此种线材的单位价格。外面由一层金属材料包裹，，同时具有较高的数据传输速率，但价格较高，安装也比较复杂；

[](http://upload.wikimedia.org/wikipedia/commons/0/09/STP-cable.png)

|  |  |
| --- | --- |
| STP规格 | 结构 |
| F/UTP | 总屏蔽层为铝箔屏蔽，没有线对屏蔽层的屏蔽双绞线 |
| U/FTP | 没有总屏蔽层，线对屏蔽为铝箔屏蔽的屏蔽双绞线 |
| SF/UTP | 总屏蔽层为丝网＋铝箔的双重屏蔽，线对没有屏蔽的双重屏蔽双绞线 |
| S/FTP | 总屏蔽层为丝网，线对屏蔽为铝箔屏蔽的多重屏蔽双绞线 |
| U/UTP | 即通常所说的UTP双绞线，非屏蔽双绞线 |

1. UTP

UTP无金属屏蔽材料，只有一层绝缘胶皮包裹，价格相对便宜，组网灵活，其线路优点是阻燃效果好，不容易引起火灾。

[](http://upload.wikimedia.org/wikipedia/commons/8/8a/UTP-cable.png) [](http://upload.wikimedia.org/wikipedia/commons/c/cb/UTP_cable.jpg)

conductor:导体，insulation:绝缘体，sheath:护套

非屏蔽双绞线电缆最早在1881年被用于贝尔发明的电话系统中。1900年美国的电话线网络亦主要由UTP所组成，它们由电话公司所拥有。

UTP被广泛用于电脑网络，但由于使用过长的UTP电缆传输数据会引致讯号衰减问题，惟其价格较光纤和同轴电缆低，因此UTP主要用作短途传输，一般不多于100米。

UTP电缆末端通常连接RJ-45，以便插入到与其相容的连接埠中。

1. 标准

1991年，美国电子协会EIA（Electronic Industries Association）和电信工业协会TIA联合发布了一个标准EIA/TIA-568，即“商用建筑物电信布线标准”（Commercial Building Telecommunications Cabling Standard）。这个标准规定了用于室内传送数据的无屏蔽双绞线和屏蔽双绞线的标准。随着局域网上数据传送速率的不断提高，EIA/TIA在1995年将布线标准更新为EIA/TIA-A，对于UTP此标准规定了5个种类的标准（从1类到5类线）。对传送数据来说，现在最常用的是5类线（Category 5 或CAT5）至今已经发展到超5类线。

### RJ45接口

RJ45 型网线插头又称水晶头，共有八芯做成，广泛应用于局域网和 ADSL 宽带上网用户的网络设备间网线（称作五类线或双绞线）的连接。

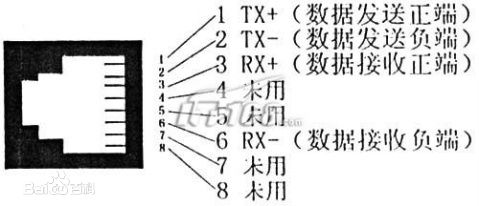
RJ这个名称代表已注册的插孔（Registered Jack），是来源于贝尔系统的USOC (Universal Service Ordering Codes，通用服务分类代码) 代码。USOC 是一系列已注册的插孔及其接线方式，是由贝尔系统开发的，用于将用户的设备连接到公共网络。FCC 规定控制着这一目的的应用。FCC（联邦通信委员会）代表美国政府发布了一个文档规定了RJ11。

1. 接口类型

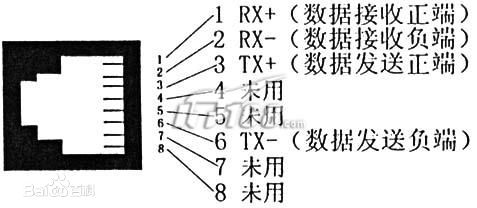
常见的RJ45接口有两类：用于以太网网卡、路由器以太网接口等的DTE类型，还有用于交换机等的DCE类型。

DTE可称作“数据终端设备”，DCE可以称作“数据通信设备”。从某种意义来说，DTE设备称为“主动通信设备”，DCE设备称为“被动通信设备”。

1. DTE类型引脚定义:



1. DCE类型引脚定义



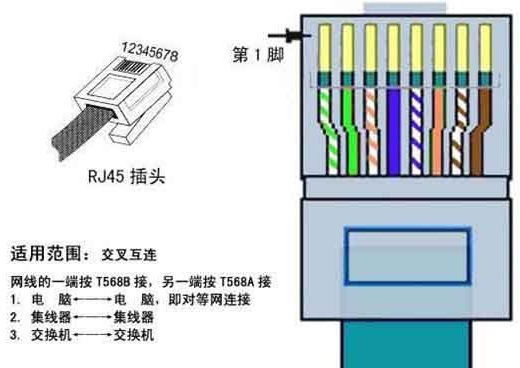
1. 线序

当两个类型一样的设备使用RJ45接口连接通信时，必须使用交叉线连接。

在具体应用时，RJ45 型插头和网线有两种连接方法（线序），分别称作 T568A 线序和T568B 线序。

RJ45 型网线插头引脚号的识别方法是：手拿插头，有 8 个小镀金片的一端向上，有网线装入的矩形大口的一端向下，同时将没有细长塑料卡销的那个面对着你的眼睛，从左边第一个小镀金片开始依次是第1 脚、第2 脚、…、第8 脚。

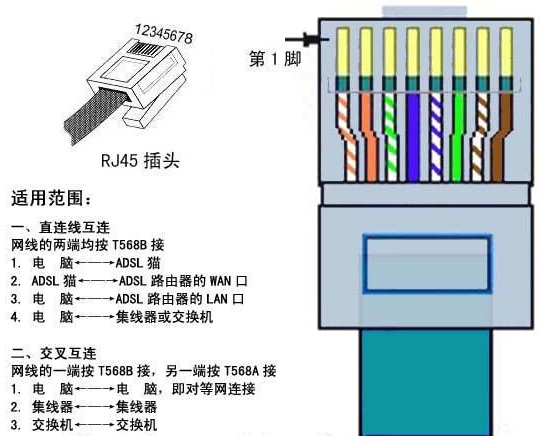
1. T568A 线序



这种接法用于网络设备需要交叉互连的场合，所谓交叉是指网线的一端和另一端与 RJ45 网线插头的接法不同，一端按 T568A 线序接，另一端按 T568B 线序接，即有几根网线在另一端是先做了交叉才接到 RJ45 插头上去的，适用的连接场合有：

|  |  |
| --- | --- |
| T568A线序 | T568B线序 |
| 电　脑 | 电 脑 |
| 集线器 | 集线器 |
| 交换机 | 交换机 |

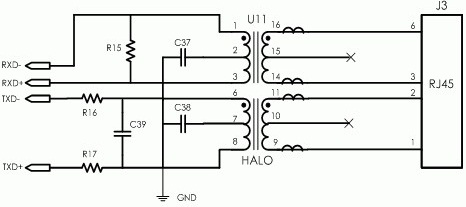
1. T568B 线序



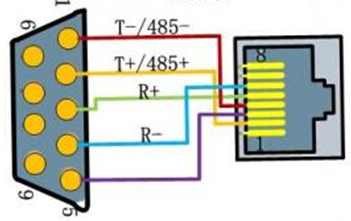
网线的两端均按 T568A或者T568B 接，直连线互连（直通线）适用的连接场合有：

|  |  |
| --- | --- |
| T568A线序 | T568A线序 |
| T568B线序 | T568B线序 |
| 电脑 | ADSL 猫 |
| ADSL 猫 | ADSL 路由器的 WAN 口 |
| 电脑 | ADSL 路由器的 LAN 口 |
| 电脑 | 集线器或交换机 |

1. 隔离电路

[](http://image.baidu.com/i?ct=503316480&z=&tn=baiduimagedetail&ipn=d&word=rj45%20%E9%9A%94%E7%A6%BB&step_word=&ie=utf-8&in=15156&cl=2&lm=-1&st=-1&pn=162&rn=1&di=25641830830&ln=1993&fr=&&fmq=1399281909911_R&ic=0&s=&se=1&sme=0&tab=&width=&height=&face=0&is=&istype=2&ist=&jit=&objurl=http://www.51kaifa.com/upload/2005/09/1128091717.gif)

1. 转接口
2. rs422转rj45



1. rs232转rj45

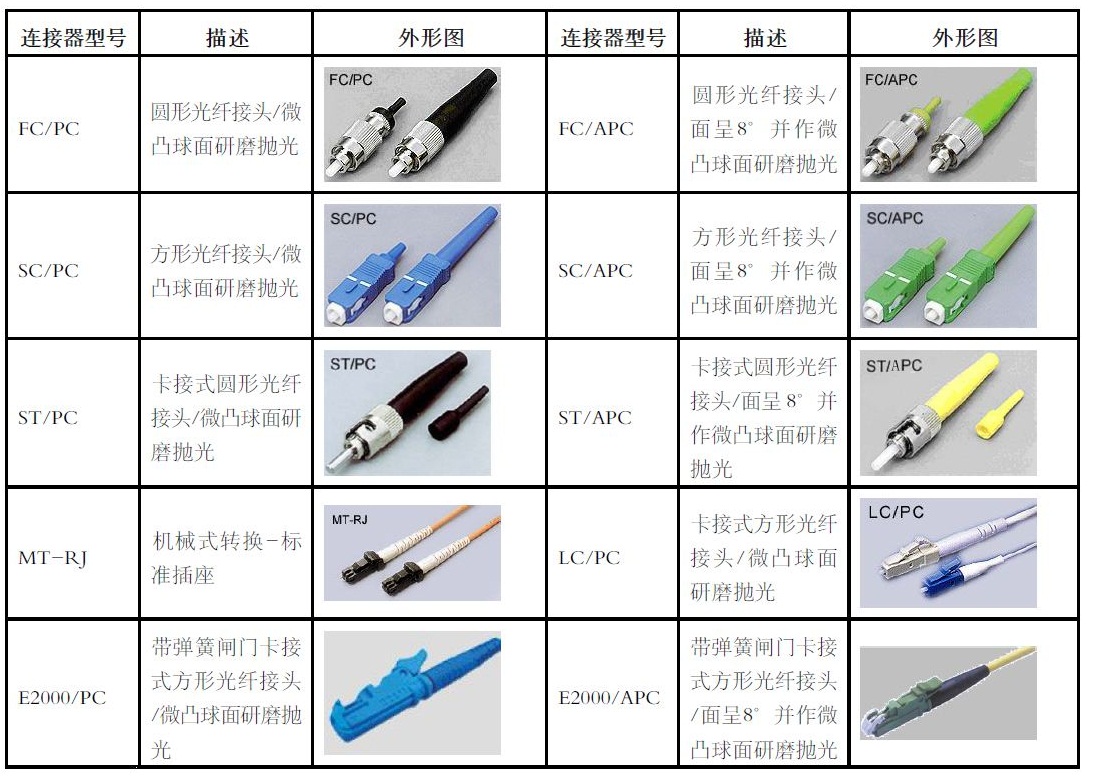


### 光纤接口

光纤提速目前能到到最好的是FTTH，一般通过光猫作为光转电的设备，然后与设备相连。

1. 接口类型

光纤连接线种类如下：



## Ethernet PHY

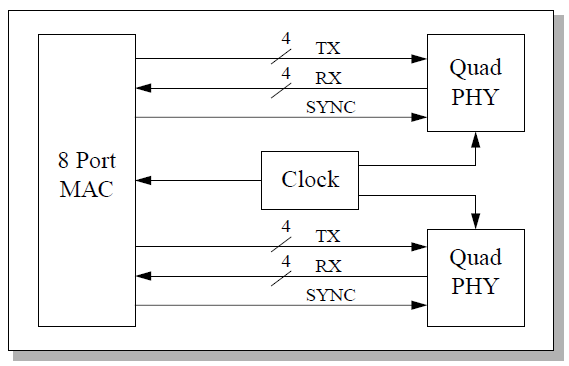
### MII接口

MII是英文Medium Independent Interface的缩写，翻译成中文是“介质无关(独立)接口”，一般应用于以太网硬件平台的MAC层和PHY层之间，MII接口的类型有很多，常用的有MII、RMII、SMII、SSMII、SSSMII、GMII、RGMII、SGMII、TBI、RTBI、XGMII、XAUI、XLAUI等。

1. MII接口框图

MII是IEEE-802.3定义的以太网行业标准，用于 Fast Ethernet MAC-block 与各种类型的PHY 连接。 它包括一个数据接口 ，以及一个MAC和PHY之间的管理接口。

The following figure shows a common SMII application。



其中：左边为MAC层芯片，其接收发送的为数字信号，右边为PHY层芯片，其用于将MAC层信号转换为物理信号。

数据接口包括分别用于发送器和接收器的两条独立信道。每条信道都有自己的数据、时钟和控制信号 。

管理接口是个双信号接口：一个是时钟信号，另一个是数据信号。通过管理接口，上层能监视和控制PHY。

与MII等效（指结构上的等效）的接口有：AUI(10M以太网)、GMII(Gigabit以太网)和XAUI(10-Gigabit以太网)。

1. MII接口类型表

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 类型 | 功能 | wire | signals(MAC 🡪 PHY，MAC🡨PHY，MAC🡨🡪PHY) | | | | | | | | | | | | | |
| MII | Medium Independent  Interface | 16 | TX | | | | | | | | RX | | | |  |  |
| **ER** | **EN** | | | **CLK** | | **D[3:0]** | | **ER** | **DV** | **CLK** | **D[3:0]** | **CRS** | **COL** |
| CRS | | Carrier Sense，当PHY在半双工模式下有效 | | | | | | | | | | | |
| COL | | Collision Detectd，当PHY在半双工模式下有效 | | | | | | | | | | | |
| CLK | | 由PHY芯片提供给MAC芯片。  100Mbps速率下，CLK=25MHz;  10Mbps速率下，CLK=2.5MHz。 | | | | | | | | | | | |
| EN | | TX\_EN(Transmit Enable)：发送使能 | | | | | | | | | | | |
| DV | | RX\_DV(Reveive Data Valid)：接收有效 | | | | | | | | | | | |
|  |  |  | ER | | 在EN/DV有效时，D是否错误，10Mbps下没有意义。 | | | | | | | | | | | |
| RMII | Reduced MII | 8 | TX | | | | | | | | RX | | | |  |  |
|  | **EN** | | |  | | **D[1:0]** | | **ER** |  |  | **D[1:0]** | **CRS\_DV** | **CLK\_REF** |
| CLK\_REF | | | | | TX和RX的时钟由PHY芯片提供给MAC芯片。 | | | | | | | | |
| CRS\_DV | | | | | 此信号是由MII接口中的RX\_DV和CRS两个信号合并而成 | | | | | | | | |
| SMII | Serial MII  (MAC/PHY芯片在接收到数据后会进行串/并转换) | 4 | TX | | | | | | | | RX | | | |  |  |
|  |  | | |  | | **D** | |  |  |  | **D** | **SYNC** | **CLK\_REF** |
| SYNC | | | 每10个时钟周期置1次高电平实现同步指示，表示一组TXD/RXD数据的开始。  对于TX：其发送数据依次为D[7:0]🡪 TX\_EN🡪TX\_ER。  对于RX：其接收数据依次为：D[7:0]🡪  RX\_DV🡪CRS。 | | | | | | | | | | |
| RXD数据格式 | | | D7 | | | | 1 | | | | | | |
| D6 | | | | False Carrier Detected | | | | | | |
| D5 | | | | Upper Nibble： 0=invalid 1=valid | | | | | | |
| D4 | | | | Jabber:：0 = OK,1 = Error | | | | | | |
| D3 | | | | Link：0 = Down,1 = Up | | | | | | |
| D2 | | | | Duplex：0 = Half，1 = Full | | | | | | |
| D1 | | | | Speed(bps)：0 = 10M，1 = 100M | | | | | | |
| D0 | | | | RX\_ER from previous frame | | | | | | |
| RX\_DV | | | | Receive Data Valid | | | | | | |
| CRS | | | | 载波侦测 | | | | | | |
| SSMII | Serial Sync MII | 6 | TX | | | | | | | | RX | | | |  |  |
| **SYNC** |  | | | **CLK** | | **D** | | **SYNC** |  | **CLK** | **D** |  |  |
|  |  |  | SSMII的TX/RX参考时钟和同步时钟都是由PHY芯片提供. | | | | | | | | | | | | | |
| SSSMII | Source Sync Serial MII | 6 | TX | | | | | | | | RX | | | |  |  |
| **SYNC** |  | | | **CLK** | | **D** | | **SYNC** |  | **CLK** | **D** |  |  |
| SSSMII的TX参考时钟和同步时钟是由MAC芯片提供，RX参考时钟和同步时钟是由PHY芯片提供 | | | | | | | | | | | | | |
| GMII | Gigabit MII | 24 | TX | | | | | | | | RX | | | |  |  |
| **ER** | **EN** | | | **CLK** | | **D[7:0]** | | **ER** | **DV** | **CLK** | **D[7:0]** | **CRS** | **COL** |
| 与MII接口类似，区别：数据宽度由4位变为8位;TX的CLK由MAC芯片提供给PHY芯片。 | | | | | | | | | | | | | |
| RGMII | Reduced GMII | 14 | TX | | | | | | | | RX | | | |  |  |
|  | **EN** | | | **CLK** | | **D[3:0]** | |  | **DV** | **CLK** | **D[3:0]** | **CRS** | **COL** |
| 在参考时钟的上升沿传输GMII接口中的D[3:0]，在参考时钟的下降沿传输GMII接口中的D[7:4] | | | | | | | | | | | | | |
| SGMII | Serial GMII | 3 | TX | | | | | | | | RX | | | |  |  |
|  |  | | |  | | **D** | |  |  | **CLK** | **D** |  |  |
| 收发各一对差分信号线，时钟频率625MHz，在时钟信号的上升沿和下降沿均采样，参考时钟RX\_CLK由PHY提供，是可选的，主要用于MAC侧没有时钟的情况，一般情况下，RX\_CLK不使用。收发都可以从数据中恢复出时钟。 | | | | | | | | | | | | | |
| TBI | Ten Bit Interface | 26 | TX | | | | | | | | RX | | | |  |  |
| **ER** | **EN** | | | **CLK** | | **D[9:0]** | | **ER** | **DV** | **CLK+/-** | **D[9:0]** | **CRS** | **COL** |
| 与GMII类似，区别：多出来的2位数据主要是因为在TBI接口下，MAC芯片在将数据发给PHY芯片之前进行了8B/10B变换(变换的主要作用是扰码，让信号中不出现过长的连“0”和连“1”情况，影响时钟信息的提取)；CLK+/-叫伪差分信号，由两个有180度的相位差的独立的信号组成。 | | | | | | | | | | | | | |
| RTBI | Reduced TBI | 14 | TX | | | | | | | | RX | | | |  |  |
|  | **EN** | | | **CLK** | | **D[4:0]** | |  | **DV** | **CLK** | **D[4:0]** |  |  |
| 接口数据位宽为5bit，时钟频率为125MHz，在时钟的上升沿和下降沿都采样数据，同RGMII接口一样，TX\_EN线上会传送TX\_EN和TX\_ER两种信息，在时钟的上升沿传TX\_EN，下降沿传TX\_ER；RX\_DV线上传送RX\_DV和RX\_ER两种信息，在RX\_CLK上升沿传RX\_DV，下降沿传RX\_ER。 | | | | | | | | | | | | | |
| XGMII | 10 Gigabit MII | 74 | TX | | | | | | | | RX | | | |  |  |
|  | **C[3:0]** | | | **CLK** | | **D[31:0]** | |  | **C[3:0]** | **CLK** | **D[31:0]** |  |  |
| PCB走线最大传输距离仅有7cm | | | | | | | | | | | | | |
| 单端信号，采用HSTL/SSTL\_2逻辑，端口电压1.5V/2.5V，由于SSTL\_2的端口电压高，功耗大，现在已很少使用。HSTL即High Speed Transceiver Logic，高速发送逻辑的意思。SSTL，即Stub Series Terminated Logic，短路终止逻辑，主要用于高速内存接口，SSTL目前存在两种标准，SSTL\_3是3.3V标准；SSTL\_2是2.5V标准。 | | | | | | | | | | | | | |
| C | | 通道控制信号，C=0时，表示D上传输的是数据；C=1时，表示D上传输的是控制字符 | | | | | | | | | | | |
| XAUI | 10 Gigabit attachment  unit interface |  | TX | | | | | | | | RX | | | |  |  |
|  | **C[3:0]** | | | **CLK** | | **D[31:0]** | |  | **C[3:0]** | **CLK** | **D[31:0]** |  |  |
| PCB走线的传输距离增加到50cm | | | | | | | | | | | | | |
| XAUI接口采用差分线，收发各四对，CML逻辑，AC耦合方式，耦合电容在10nF~100nF之间。 | | | | | | | | | | | | | |
| 有些芯片不支持XAUI接口，只支持XGMII接口，这时可以用专门的芯片进行XGMII🡪XAUI接口转换，如BCM8011等 | | | | | | | | | | | | | |

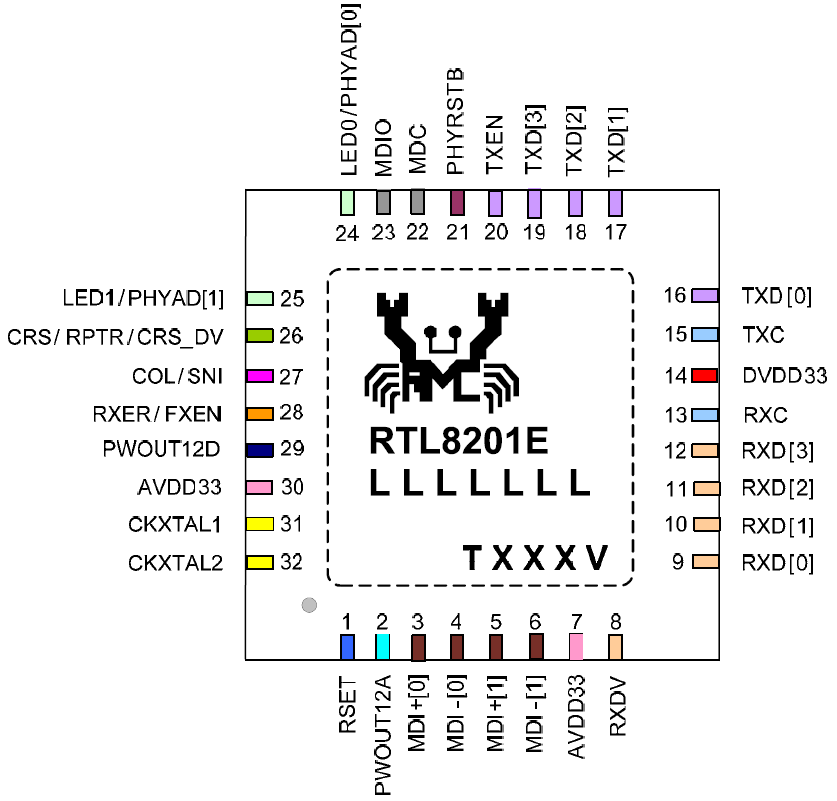
1. 接口芯片

常见接口芯片介绍

|  |  |
| --- | --- |
| 接口芯片 | 介绍 |
| Realtek 8201BL | 是一种常见的主板集成网络芯片（又称为PHY网络芯片）。PHY芯片是指将网络控制芯片的运算部分交由处理器或南桥芯片处理，以简化线路设计 |
| Realtek 8139C/D | 主要增加了电源管理功能，其他则基本上与8139C芯片无异。支持10M/100Mbps |
| lntel Pro/100VE | lntel公司的入门级网络芯片 |
| nForce MCP NVIDIA/3Com | 内置了两组网络芯片功能：Realtek 8210BL PHY网络芯片和Broabcom AC101L PHY网络芯片 |
| 3Com 905C | 支持10/100Mbps速度 |
| SiS900 | 已经集成到南桥芯片中。支持100Mbps |
| RC82545EM | Intel推出的千兆系列网卡芯片中的一种，可以支持普通的网络设备以及五类、六类双绞线 |
| DM9102HEP | Davicom的网卡芯片，PCI接口10/100Mbps以太网控制器，适用于主芯片带PCI总线的嵌入式应用 |
| DM9000AEP/CEP | Local Bus总线接口10/100Mbps以太网控制器，适用于用ARM、DSP等开发的各种带网络功能的产品 |

### RTL8201介绍

1. RTL8201E芯片



## Ethernet MAC

### DWC

Synopsyss DesignWare Cores Ethernet MAC Universal, release 3.42a. This product enables a host to transmit and receive data over Ethernet in compliance with the IEEE 802.3 specification. Ethernet MAC Universal (or GMAC-UNIV for short) is provided as two separate licenses, as follows:

❖ DWC Ether MAC 10/100/1000 Universal is the Media Access Controller for 10/100/1000 Ethernet.

❖ DWC Ether MAC 10/100 Universal is the Media Access Controller for 10/100 Ethernet.

Ethernet MAC Universal corresponds to either DWC Ether MAC 10/100/1000 Universal or DWC Ether MAC 10/100 Universal in the SolvNet database.

1. configurations

The DWC Ether MAC 10/100/1000 Universal, commonly referred to as GMAC-UNIV in this document, enables a host to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2002 standard. The GMAC-UNIV can have four major configurations:

GMAC core only with native interface (GMAC-CORE);

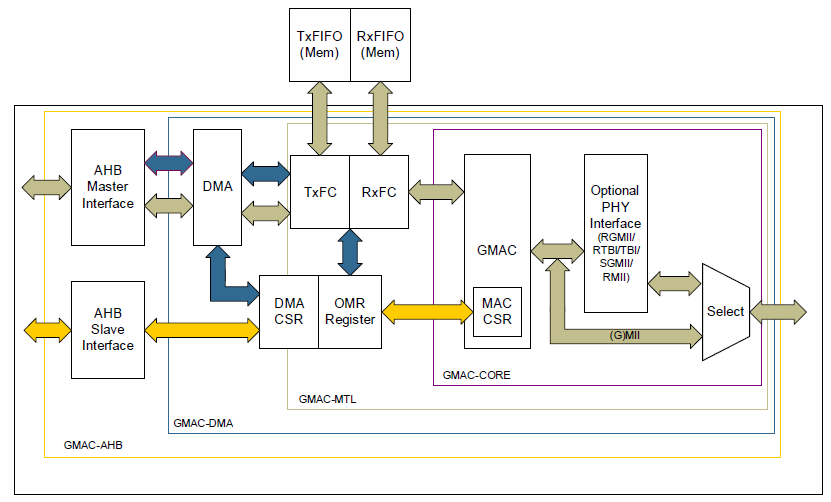
GMAC with transaction layer (GMAC-MTL);

GMAC with native DMA (GMAC-DMA);

GMAC with AHB-interfaced DMA (GMAC-AHB).

### GMAC-UNIV

1. 系统框图



The GMAC-AHB transfers data to system memory through the AHB master interface. This interface can be removed for a non-AHB system, and the subsystem will have a direct native FIFO-type interface. The host CPU uses the default 32-bit AHB Slave interface to access the GMAC subsystem’s Control and Status registers (CSRs). There is an option to select an APB port for CSR access instead of the AHB Slave port. For non-AHB systems, the AHB/APB slave modules can be removed and the native 32-bit Read/Write bus is provided for CPU accesses. The GMAC-UNIV supports any one or a combination of the following PHY interfaces:

❖ Gigabit Media Independent Interface (GMII)/Media Independent Interface (MII) [Default]

❖ Reduced GMII (RGMII)

❖ Serial GMII (SGMII)

## TCP/IP协议

### 协议实现

1. LwIP

轻量型IP（LightWeightIP，LwIP）协议，是TCP/IP协议栈的一种实现。它可以被移植到各种操作系统中，也可以无操作系统的支持独立运行。LwIP实现的重点是在保持TCP/IP协议主要功能的基础上，减少存储器占用量和代码尺寸。LwIP实现了较为完备的IP，CMP，UDP，TCP协议。具有超时时间估计、快速恢复和重发、窗口调整等功能。该协议栈提供了一组API 函数供应用程序调用，编程方便。

# 无线网络

## WIFI

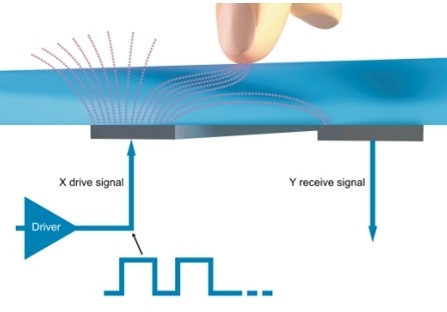
## BlueTooth

# 按键设备

## 电容触摸屏

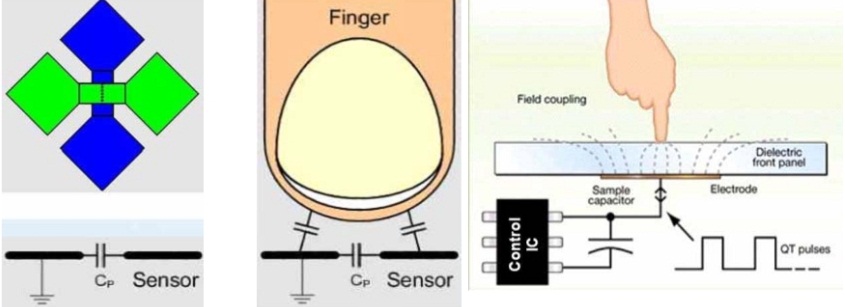
### 感应原理

电容屏通过任何持有电荷的物体包括人体皮肤工作。（人体所带的电荷）电容式触摸屏是由诸如合金或是銦錫氧化物（ITO）这样的材料构成，电荷存储在微型静电网中。当手指点击屏幕，会从接触点吸收小量电流，造成角落电极的压降，利用感应人体微弱电流的方式来达到触控的目的。（这是为什么当你带上手套触摸屏幕时，没有反应的原因），如下图。



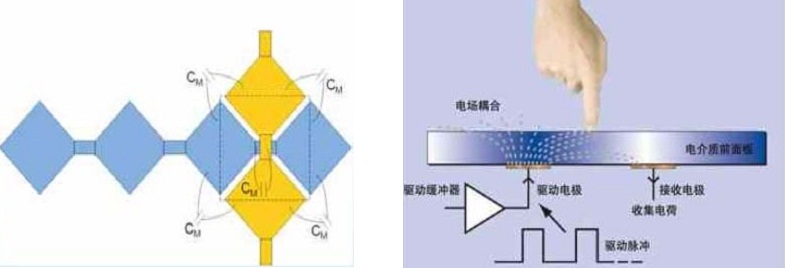
### 感应材料

1. 自生电容式触摸屏



手指触摸时，皮肤表面和sensor之间会产生寄生电容，则通过检测Cp电容的变化量,确定手指触摸的位置。

1. 互电容式触摸屏

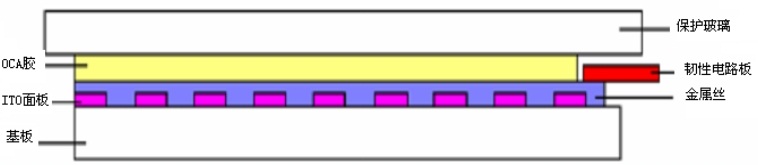


手指触摸时耦合电容CM减小 ，则通过检测CM变化量,确定手指触摸的位置。

### 布局结构

根据电容材料在触摸屏的布局可分为如下：

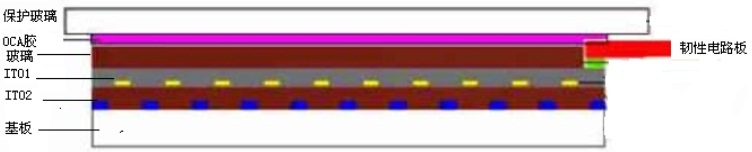
1. 单层ITO



优点:成本低,透过率

高缺点: 抗干扰能力 差

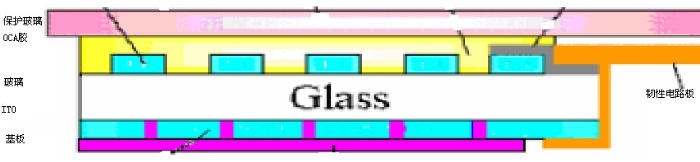
1. 单面双层ITO



优点:性能好,良率高

缺点:成本较高

1. 双面单层ITO



优点:性能好,抗静电能力强

缺点:抗干扰能力差

### 采样方式

根据触摸信号采集方式分为两种：

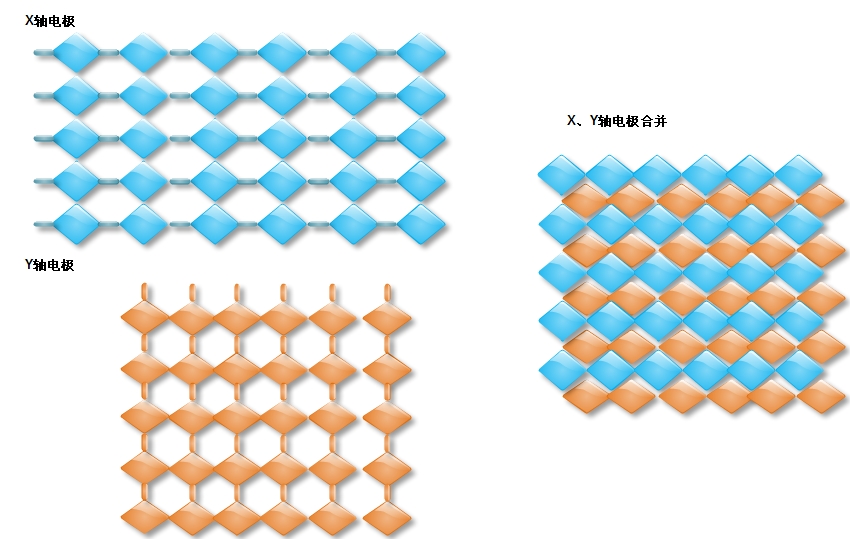
1. 表面电容式

表面电容式利用位于四个角落的传感器以及均匀分布整个表面的薄膜，有一个普通的ITO层和一个金属边框,当一根手 指触摸屏幕时,从板面上放出电荷,感应在触 屏 的四角完成,不需要复杂的ITO图案；

1. 投射式电容

采用一个或多个被蚀烛的ITO,这些 ITO层通过蛀蚀形成多个水平和垂直电极，采用成行/列交错同时带有传感功能的独立芯片。现在平板电脑、手机、车载等多用投射式电容。

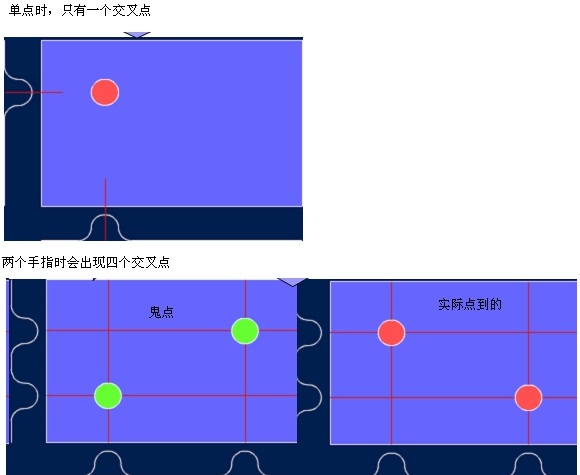
投射电容的轴坐标式感应单元以两个交叉的滑条实现 X轴滑条 Y轴滑条检测每一格感应单元的电容变化。（示意图中电容，实际为透明的）



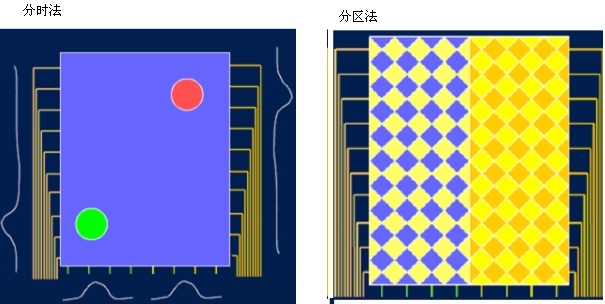
### 鬼点消除

1. 鬼点现象

当一个手指按下时，X、Y轴只有一个交叉点，两个同时按下时就会出现4个交叉点，如下图所示，我们不期望得到的点就是所说的鬼点。



1. 消除方法



1. 分时法

基于时间的多点触摸,假设多点触摸 分时进行,操作间 隔续集毫秒；

1. 分区法

将整个触屏物理上分割几个区域 通过判断触摸进入推出相应区域,从而分出鬼点中分出真实点。

# 感应设备

## 技术发展

在大多数的手机中大约有7种感测器类型，分别是麦克风、加速度计、陀螺仪、体声波(BAW)、磁力计、压力与湿度感测器。加速度计/陀螺仪/磁力计被封装在一起形成惯性测量单元(IMU)，而压力、湿度与温度感测器则链接在一起形成环境感测单元。

感测器中枢微控制器(MCU)是近来的一项发展趋势。各种运算全部都在应用处理器(AP)进行。现在，包括博世(Bosch)、意法半导体(ST)、应美盛(InvenSense)等公司都已经推出这种感测器中枢了。」

## 惯性感应

### 加速度计

### 陀螺仪

### 磁力计

## 环境感应

### 压力感测器

### 湿度感测器

### 温度感测器

### 气体化学感测器

## 其他

### 红外线(IR)成像

红外线是一种非常有趣的发展成果，因为它支援手势与脸部辨识，也支援夜视功能