**51单片机学习总结**

**目录**

**[一．51单片机介绍 1](#_Toc4999)**

**[二． 中断 5](#_Toc10918)**

**[三． 定时器 9](#_Toc2901)**

**[四． C总线 12](#_Toc12692)**

**[五． AD 18](#_Toc30368)**

**[六． DA 21](#_Toc3596)**

**[七． 红外遥控 24](#_Toc10825)**

**[八． 外围器件 28](#_Toc11138)**

**[1. LED 28](#_Toc28093)**

**[2. 独立按键 29](#_Toc15948)**

**[3. 数码管 30](#_Toc3088)**

**[4. 矩阵按键 33](#_Toc5343)**

**[5. LED点阵屏 34](#_Toc21197)**

**[6. 蜂鸣器 36](#_Toc17376)**

**[7. LCD1602液晶显示屏 38](#_Toc5190)**

**[8. 直流电机 41](#_Toc2985)**

**[9. 温度传感器（DS18B20） 42](#_Toc12653)**

**一．51单片机介绍**

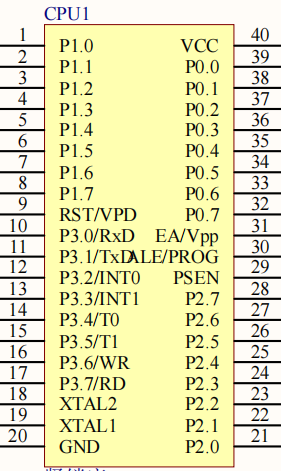
1. **定义：**单片机就是一块集成芯片，但这块集成芯片具有一些特殊的功能，而它的功能的实现要靠我们使用者自己来编程完成。我们编程的目的就是控制这块芯片的各个引脚在不同时间输出不同的电平，进而控制与单片机各个引脚相连接的外围电路的电气状态。单片机芯片自身不能单独运用于某项工程或产品上，它必须要靠外围数字器件或模拟器件的协调才可发挥其强大的功能。
2. **一台能够工作的计算机要有这样几个部份构成：**

①CPU：进行运算、控制 ②RAM：数据存储

③ROM：程序存储 ④输入/输出设备：串行口、并行输出口等

有一些单片机中除了上述部份外还集成了其它部份，如A/D，D/A等。

1. **引脚介绍：**（如图，共40个引脚）



①**VCC/GND**：接入电源，常压为5V,低压为3.3V。

②**XTAL1/XTAL2**：外接晶振，为单片机提供时钟。XTAL1为输入端，XTAL2为输出端。

③**RST/VPD：**RST：复位，单片机重头开始执行程序。

VPD：在VCC掉电时接备用电源。

④**PSEN**：读外部程序，低电平有效，目前单片机内部ROM充足，几乎不用。

⑤**ALE/PROG**：ALE：拓展外部RAM时，控制把P0口低8位地址锁存起来，实现数据与地址分离，下降沿发出地址信息。不拓展外部RAM时，以1/6周期震荡频率输出，故可用作外部时钟或外部定时脉冲。

PROG：编程脉冲输入端，在EPROM编程期间通过此端口将程序写入ROM。目前单片机多用串口直接写入程序，故不常用。

⑥**EA/VPP**：EA：高电平时读取内部存储器，拓展有外部ROM时，读完内部ROM接着读外部的ROM，低电平时直接读取外部ROM。目前单片机内都有ROM，故设计电路时该引脚始终高电平。

VPP：单片机内有EPPROM芯片，在EPROM编程期间施加编程电压VPP。

⑦**I/O口**：P0：双向8位三态I/O口，每个口可独立控制，内部没有上拉电阻，为高阻态，故使用时需外接10kΩ上拉电阻。

P1：准双向8位I/O口，每个口可独立控制，内部有上拉电阻。

P2：准双向8位I/O口，每个口可独立控制，内部有上拉电阻。

P3：准双向8位I/O口，每个口可独立控制，内部有上拉电阻。

⑧**P3的第二功能**：P3^0：RXD,串行输入口。

P3^1：TXD，串行输出口。

P3^2：INT0，外部中断0。

P3^3：INNT1，外部中断1。

P3^4：T0，定时器/计数器0外部中断输入端。

P3^5：T1，定时器/计数器1外部中断输入端。

P3^6：WR，外部数据存储器写脉冲。

P3^7：RD，外部数据存储器读脉冲。

注：①**三态是指**：高电平1、低电平0、高阻态Z

②**高阻态**：既不是高电平，也不是低电平，以高阻态对下级电路输出，对下级电路没有影响。高阻态的I/O口电平随下级电路一致。下级为高则为高，下级为低则为低，故可看成悬空状态。

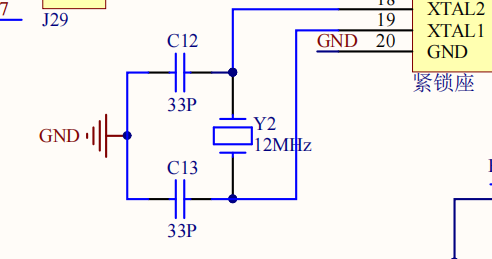
③**准双向I/O口**：数据输入时需要对其置1，否则若前一位为低电平，后一位输入的电平为高时，MOS管拉不起来导致出错。

**双向口：**数据传输时可直接输入输出，因为双向口有悬浮态。

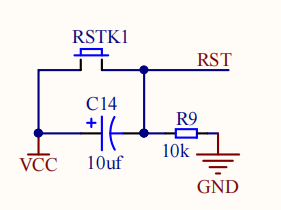
④若想提高I/O口驱动能力，可外接上拉电阻。

1. **最小系统：**①晶振电路 ②复位电路 ③电源电路 ④程序下载电路

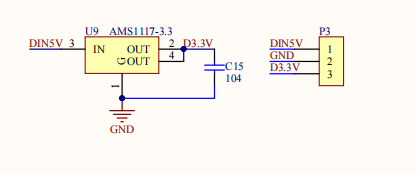
①**晶振电路**：若直接将晶振接入单片机晶振引脚，晶振起振的一瞬间会产生一些电感，导致系统工作不稳定。为了消除电感，在晶振两端分别加上一个无极性电容，电容另一端共地，大小一般选取33PF。



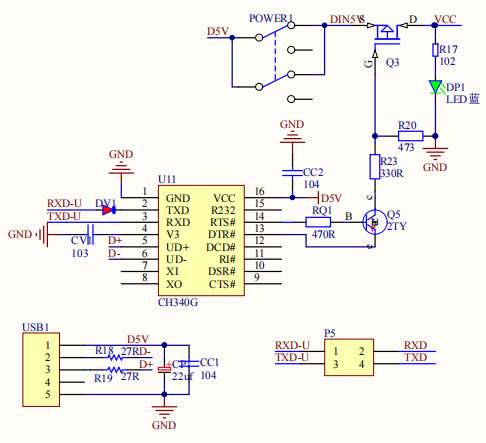
②**复位电路**：STC89CXX高电平复位。采用两种复位方式，即按键复位和上电自动复位。手动复位是通过一个按键及电容电阻组成，利用按键的开关功能实现复位。上电自动复位主要利用RC充放电功能达到复位效果。



③**电源电路**：STC89CXX工作电压为3.3V-5.5V。通常我们使用5V直流，将电源接入到个芯片电源引脚即可。P3端子可方便地为外部模块连接电源。

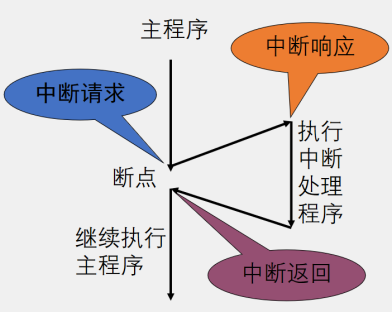
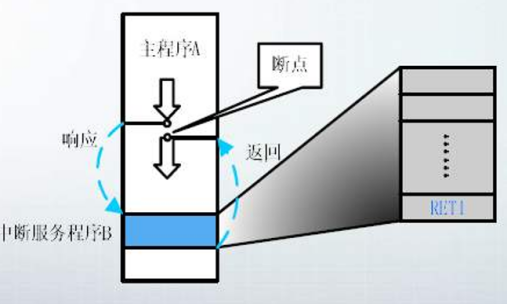


④**程序下载电路**：笔记本电脑没有RS232接口，所以要用USB转TTL串口电平芯片来建立PC机和单片机数据传输通路。通常用CH340G或者CH340C芯片来完成电平转换。



1. **中断**
2. **介绍**

为使单片机具有对外部或内部随机发生的事件实时处理而设置的，中断功能的存在，很大程度上提高了单片机处理外部或内部事件的能力。中断是指CPU在处理某一事件A时，发生了另一事件B，请求CPU迅速去处理(中断发生)；CPU暂时停止当前的工作(中断响应)，转去处理事件B(中断服务)；待CPU将事件B处理完毕后，再回到原来事件A被中断的地方继续处理事件A(中断返回)，这一过程称为中断。

1. **优点**

①**分时操作**：CPU可以分时为多个 I/O 设备服务，提高了计算机的利用率；

②**实时响应**：CPU能够及时处理应用系统的随机事件，系统的实时性大大增强；

③**可靠性高**：CPU具有处理设备故障及掉电等突发性事件能力，从而使系统可靠性提高。

1. **单片机内部配置**（51单片机内部共有5个中断请求源）

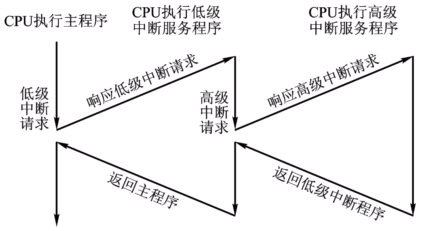
①外部中断0(INT0) ②外部中断1(INT1) ③定时器0中断

④定时器1中断 ⑤串口中断

**注**：52单片机内还额外有3个中断源：定时器2中断、外部中断2、外部中断3

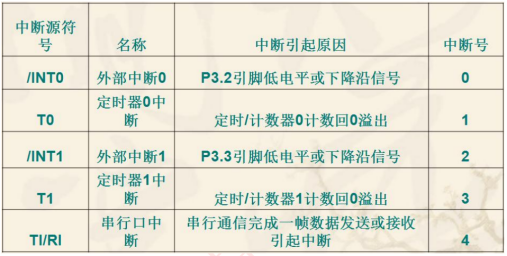
1. **中断嵌套**

如果单片机正在处理一个中断程序，此时又有一个中断现象发生，单片机将会停止当前的中断程序，而转去执行新的中断程序，新的中断程序处理完毕后再回到刚才停止的中断程序处继续执行，执行完之后中断再返回主程序继续执行主程序。



1. **中断优先级**

请示CPU中断的请求源称为中断源，单片机的中断系统一般允许多个中断源，当几个中断源同时向CPU请求中断，要求为它服务的时候，这就存在CPU优先响应哪一个中断源请求的问题。通常根据中断源的轻重缓急排队，优先处理最紧急事件的中断请求源，即规定每一个中断源有一个优先级别。CPU总是先响应优先级别最高的中断请求。

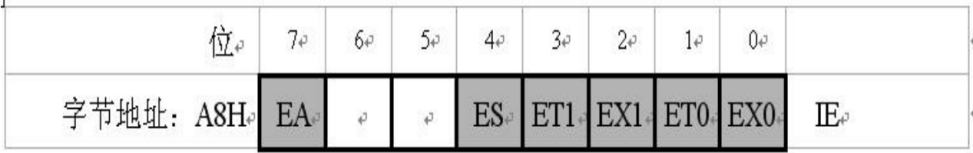
1. **配置优先级**

通过设置新增加的特殊功能寄存器IP中的相应位，可将中断优先级设为四级，如果只设置IP或TCON，那么中断优先级就只有两级。



1. **中断允许控制**

CPU对中断系统所有中断以及某个中断源的开放和屏蔽是由中断允许寄存器IE 控制的。



EX0：外部中断0允许位。

ET0：定时/计数器T0中断允许位。

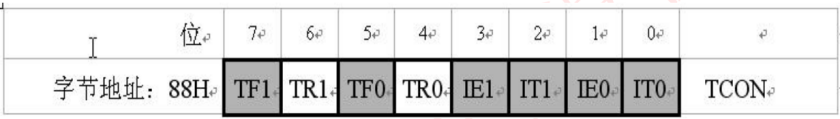
EX1：外部中断0允许位。

ET1：定时/计数器T1中断允许位。

ES：串行口中断允许位。

EA：CP中断允许（总允许）位。

1. **中断请求标志（TCON）**



**IT0**：外部中断0触发方式控制位。IT0=0：电平触发方式；IT0=1，为边沿触发方式(下降沿有效)。

**IE0**：外部中断0中断请求标志位。

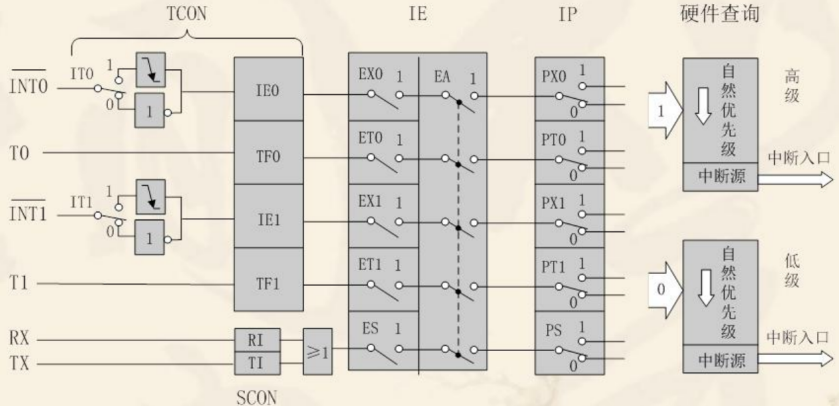
**IT1**：外部中断1触发方式控制位。

**IE1**：外部中断1中断请求标志位。

**TF0**：定时/计数器T0溢出中断请求标志位。

**TF1**：定时/计数器T1溢出中断请求标志位。

1. **单片机配置中断**



①INT0对应的是P3^2口的附加功能，可由IT0选择其为低电平有效还是下降沿有效。当CPU检测到P3^2引脚上出现有效的中断信号时，中断标志IE0置1，向CPU申请中断。

②INT1对应的是P3^3口的附加功能，可由IT1选择其为低电平有效还是下降沿有效。当CPU检测到P3^3引脚上出现有效的中断信号时，中断标志 IE1(TCON.3)置1，向CPU申请中断。

③T0对应的是P3^4口的附加功能，TF0,片内定时/计数器T0溢出中断请求标志。当定时/计数器T0发生溢出时，置位TF0，并向CPU申请中断。

④T1对应的是 P3^5口的附加功能，TF1，片内定时/计数器T1溢出中断请求标志。当定时/计数器T发生溢出时，置位TF1，并向CPU申请中断。

⑤RXD和TXD对应对应的是P3^0和P3^1口的附加功能，RI或TI，串行口中断请求标志。当串行口接收完一帧串行数据时置位RI或当串行口发送一帧串行数据时置位TI，向CP申请中断。

1. **中断响应条件：**以下三条同时满足时，CPU才有可能响应中断。

①中断源有中断请求

②此中断源的中断允许位为1

③CPU开中断（即 EA=1）

1. **代码范例**

以定时器0为例，用定时器工作方式1在51单片机上实现第一个LED灯以1s为周期改变亮灭状态。

#include <REGX52.H> //单片机头文件

sbit LED = P2^0;

int main()

{

TMOD &= 0xF0; //设置定时器模式

TMOD |= 0x01; //设置定时器模式

TL0 = 0x18; //设置定时初值

TH0 = 0xFC; //设置定时初值

TF0 = 0; //清除TF0标志

TR0 = 1; //定时器0开始计时

ET0 = 1; //定时/计数器T0中断允许

EA = 1; //打开总中断开关

PT0 = 0; //设置中断优先级

while(1); //程序停止在这里等待中断发生

return 0;

}

void Timer0\_Routine() interrupt 1 //interrupt：中断关键字 1：中断号

{

static unsigned int T0Count;

TL0 = 0x18; //设置定时初值

TH0 = 0xFC; //设置定时初值

T0Count++;

if(T0Count>=1000)

{

T0Count=0; //已到1秒钟，重新计数

LED = ~LED; //LED状态改变

}

}

1. **定时器**
2. **介绍：**定时器属于单片机的内部资源，其电路的连接和运转均在单片机内部完成。用于计时系统，可

实现软件计时，或者使程序每隔一固定时间完成一项操作。替代长时间的Delay，提高CPU的运行效率和处理速度等。有了定时器/计数器之后，可以增加单片机的效率，一些简单的重复加1的工作可以交给定时器/计数器处理。CPU转而处理一些复杂的事情。同时可以实现精确定时作用。

1. **单片机内部配置**
2. STC89C5X单片机内有两个可编程的定时/计数器T0、T1和一个特殊功能定时器T2。

因为既可以定时，又可以计数，故称之为定时器/计数器。通常我们使用的都是基本的2个定时器：定时器T0、T1。

1. 定时器/计数器和单片机的CPU是相互独立的。定时器/计数器工作的过程是自动完

成的，不需要CPU的参与。

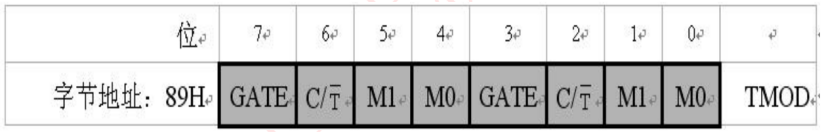
3）51单片机中的定时器/计数器是根据机器内部的时钟或者是外部的脉冲信号对寄存器中的数据加1。

1. **原理**

定时/计数器的实质是加1计数器（16位），由高8位和低8位两个寄存器THx和TLx组成。它随着计数器的输入脉冲进行自加1，也就是每来一个脉冲，计数器就自动加1，当加到计数器为全1时，再输入一个脉冲就使计数器回零，且计数器的溢出使相应的中断标志位置1，向CPU发出中断请求（定时/计数器中断允许时）。如果定时/计数器工作于定时模式，则表示定时时间已到；如果工作于计数模式，则表示计数值已满。可见，由溢出时计数器的值减去计数初值才是加1计数器的计数值。

1. **单片机配置定时器**（51单片机定时/计数器的工作由两个特殊功能寄存器控制）

① **TMOD**：定时/计数器的工作方式寄存器，确定工作方式和功能；低四位用于T0，高四位用于T1。



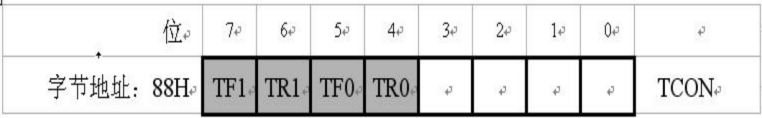
**GATE**：门控制位。为1时，使TR0或TR1为1，则启动定时/计数器； 为0时，使TR0或TR1为1，同时外部中断引脚INT0/1为高电平，则启动定时/计数器。

**C/T**：定时/计数模式选择。0为定时模式，1为计数模式。

**M1/M0**：工作方式设置。（定时器0为例）



② **TCON**：控制寄存器，控制T0、T1的启动和停止及设置溢出标志。高四位用于控制定时器/计数器的启动和中断申请。



**TF1**：T1溢出中断请求标志位。T1计数溢出时由硬件自动置TF1为1。CPU响应中断后TF1由硬件自动清0。T1工作时，CPU可随时查询TF1的状态。所以，TF1可用作查询测试的标志。TF1也可以用软件置1或清0，同硬件置1或清0的效果一样。

**TR1**：T1运行控制位。TR1置1时，T1开始工作；TR1置0时，T1停止工作。TR1由软件置1或清0。所以，用软件可控制定时/计数器的启动与停止。

**TF0**：T0溢出中断请求标志位，其功能与TF1类同。

**TR0**：T0运行控制位，其功能与TR1类同。

③ **工作方式介绍**

1） **方式0（13位）**：由TL0的低5位（高3位未用）和TH0的8位组成。TL0的低5位溢出时向

TH0进位，TH0溢出时，置位TCON中的TF0标志，向CPU发出中断请求。

2）**方式1（16位）**：TL0作为低8位，TH0作为高8位，组成了16位加1计数器。

1. **方式2（8位自动重装）**：TH0作为常数缓冲器，当TL0计数溢出时，在溢出标志TF0置1的同时，还自动地将TH0中的常数重新装入TL0中，TL0从初值开始重新计数，提高了定时精度。
2. **方式3**：只适用于定时/计数器 T0，定时器T1处于方式3时相当于TR1=0，停止计数。工作方式3将T0分成为两个独立的8位计数器TL0和TH0。

④**编写计时器程序步骤**

1. 对TMOD赋值，确定T0和T1的工作方式。
2. 计算初值，并将初值写入TH0，TL0或TH1，TL1中。
3. 如果使用中断，则对EA赋值，开放定时器中断。
4. 使TR0或TR1置位，启动定时器/计数器开始定时或计数。

⑤**计算初值**

51单片机内部时钟频率是外部时钟的12分频。比如外部用的是12MHZ的晶振，那么单片机内部的时钟频率就是 12/12MHZ，此时机器周期为1/1M=1us。如果想定时1ms的初值，1ms/1us=1000。也就是要计100个，初值=65535-1000+1=64536=FC18H，因为实际上计数器计数到66636（2的16次方）才溢出，所以后面要加1。所以初值即为 THx=0XFC，TLx=0X18。

⑥**代码范例**（定时器0为例，使用定时器1时只需把0改为1即可）

void Timer0Init()

{

TMOD|=0X01; //选择为定时器0模式，工作方式1，仅用TR0打开启动。

TH0=0XFC; //给定时器赋初值，定时1ms

TL0=0X18; //给定时器赋初值，定时1ms

ET0=1; //打开定时器0中断允许

EA=1; //打开总中断

TR0=1; //打开定时器

}

⑦**补充知识**

1）**振荡周期**：为单片机提供定时信号的振荡源的周期（晶振周期或外加振荡周期）。

2）**状态周期**：2个振荡周期为1个状态周期，用S表示。振荡周期又称S周期或时钟周期。

3）**机器周期**：1个机器周期含6个状态周期，12个振荡周期。

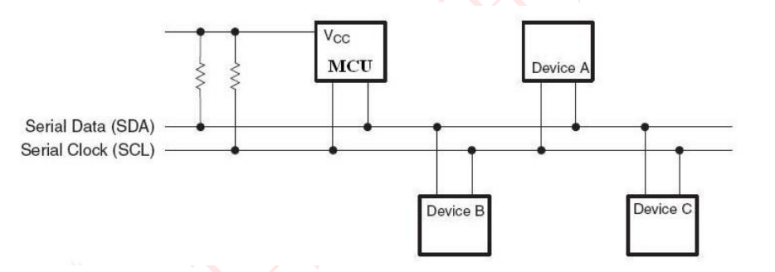
（机器周期 = 1 / 单片机的时钟频率）

4）**指令周期**：完成1条指令所占用的全部时间，它以机器周期为单位。

1. **C总线**
2. **介绍**

两线式串行总线，用于连接微控制器及其外围设备。是微电子通信控制领域广泛采用的一种总线标准。它是同步通信的一种特殊形式，具有接口线少，控制方式简单，器件封装形式小，通信速率较高等优点。I2C总线只有两根双向信号线，一根是数据线SDA，另一根是时钟线SCL。由于其管脚少，硬件实现简单，可扩展性强等特点，因此被广泛的使用在各大集成芯片内。

1. **外围设备连接方式**



①可连接多个通讯设备，支持多个主机和多个从机，数量受总线的最大电容400pF限制。每个设备有单独的地址，主机可通过地址访问不同设备，多个主机同时使用总线时，使用仲裁的方式选择主机来使用总线。

②SDA为双向串行数据线，用来表示数据。SCL为时钟线，用于数据收发同步。

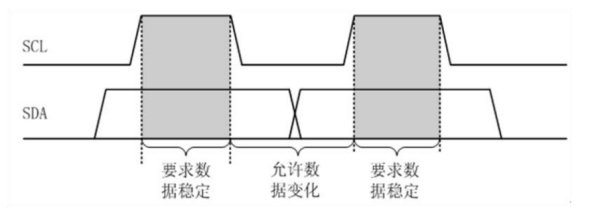
③I2C总线经上拉电阻接入电源。当设备空闲时输出高阻态，而当所有设备都空闲，都输出高阻态时，由上拉电阻把总线拉成高电平。

④**三种传输模式**：标准模式传输速率为100kbit/s，快速模式为400kbit/s，高速模式下可达3.4Mbit/s。

1. **使用方法**

①**数据发送**

时钟信号为高电平期间，数据线上的数据必须保持稳定，只有在时钟线上的信号为低电平期间数据线上的高电平或低电平状态才允许变化。每次数据传输都以字节为单位，每次传输的字节数不受限制。



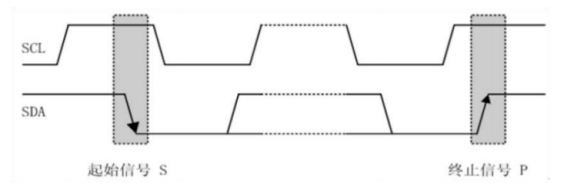
②**起始和终止信号**

**起始信号**：SCL为高电平期间，SDA由高电平向低电平的变化。

**终止信号**：SCL为高电平期间，SDA由低电平向高电平的变化。

起始和终止信号都是由主机发出的，在起始信号产生后，总线就处于被占用的状态；在终止信号

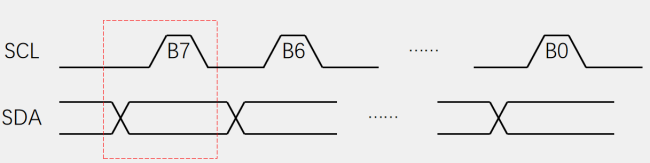
产生后，总线就处于空闲状态。



③**发送一个字节**

SCL低电平期间，主机将数据位依次放到SDA线上（高位在前），然后拉高SCL，从机将在SCL

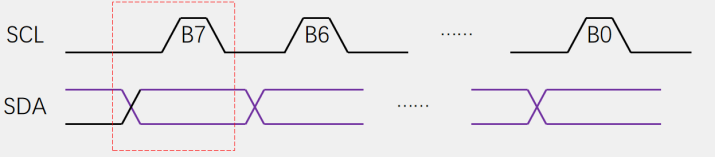
高电平期间读取数据位，所以SCL高电平期间SDA不允许有数据变化，依次循环上述过程8次，即可发送一个字节。



④**接收一个字节**

SCL低电平期间，从机将数据位依次放到SDA线上（高位在前），然后拉高SCL，主机将在SCL

高电平期间读取数据位，所以SCL高电平期间SDA不允许有数据变化，依次循环上述过程8次，即可接收一个字节（主机在接收之前，需要释放SDA）

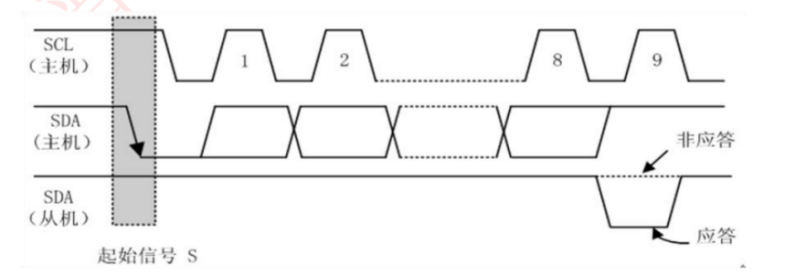


⑤**应答/非应答**

**应答**：接收端接收到一个字节数据或地址后，若希望对方继续发送数据，则需向对方发送应答信号(ACK)，即特定的低电平脉冲，发送方会继续发送下一个数据。

**非应答**：若接收端希望结束数据传输，则向对方发送“非应答(NACK)”信号，即特定的高电平脉冲，发送方接收到该信号后会产生一个停止信号，结束信号传输。

**注**：每一个字节必须保证是8位长度。数据传送时，先传送最高位（MSB），每一个被传送的字节后面都必须跟随一位应答位（即一帧共有9位）。



⑥**寻址**

当主机发送了一个地址后，总线上的每个器件都将头 7 位与它自己的地址比较，如果一样，器件

会判定它被主机寻址，其他地址不同的器件将被忽略后面的数据信号。至于是从机接收器还是从机发送器，都由R/W位决定的。

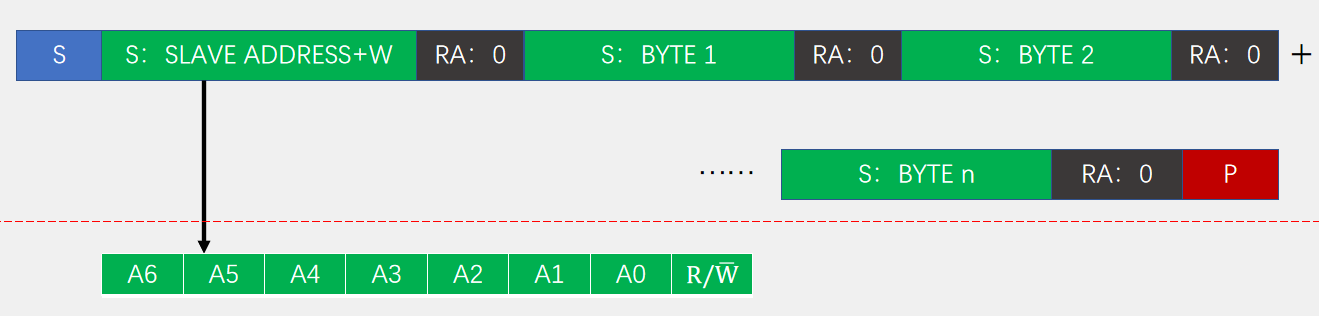


**D7～D1**：组成从机的地址

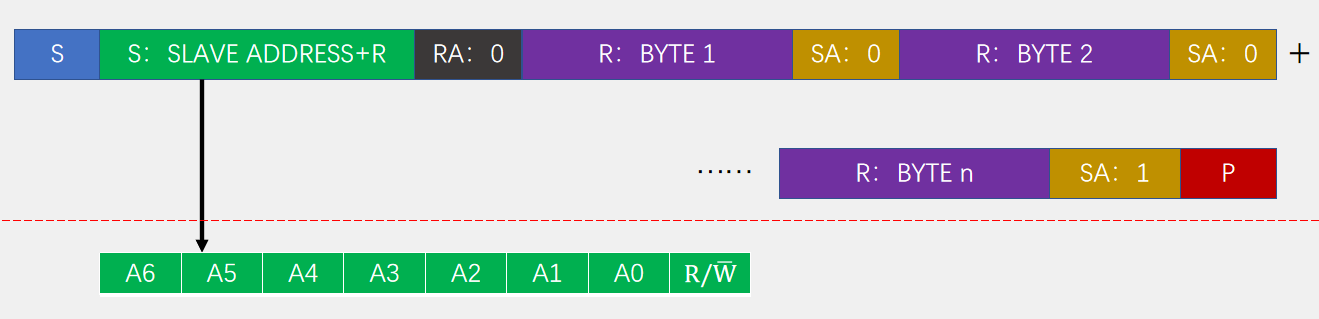
**D0**：数据传送方向位，0表示主机向从机写数据，1表示主机由从机读数据

**注**：从机的地址由固定部分和可编程部分组成。在一个系统中可能希望接入多个相同的从机，从机地址中可编程部分决定了可接入总线该类器件的最大数目。如一个从机的7 位寻址位有4位是固定位，3位是可编程位，这时仅能寻址8个同样的器件，即可以有8个同样的器件接入到该I2C总线系统中。

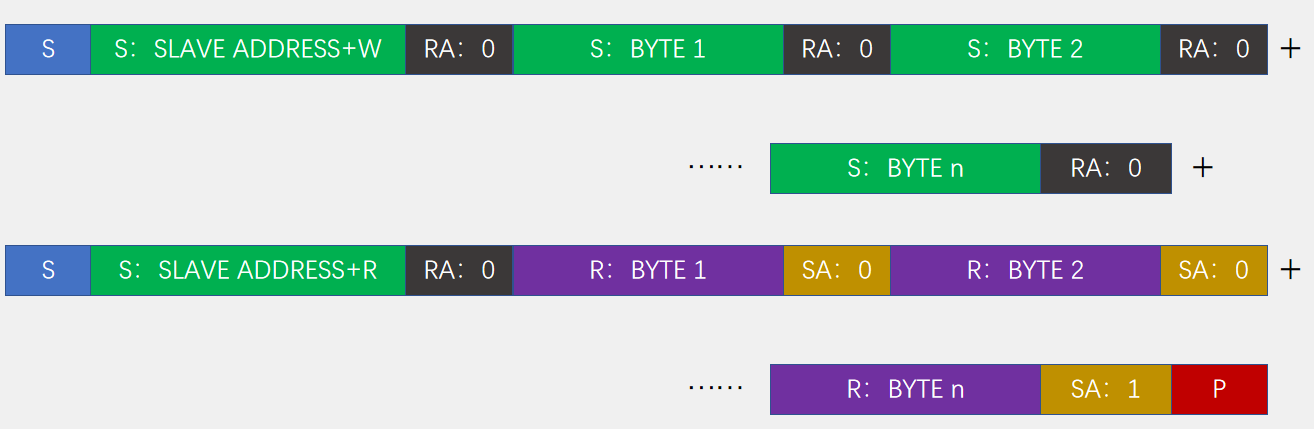
⑦**发送数据帧**



⑧**接收数据帧**



⑨**先发送再接收数据帧**



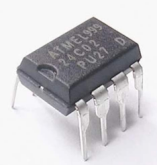
**注：**我们通常还是采用软件模拟I2C。主要原因是硬件I2C设计的比较复杂，而且稳定性不怎么好，程序移植比较麻烦，而用软件模拟I2C，最大的好处就是移植方便，同一个代码兼容所有单片机，任何一个单片机只要有IO口（不需要特定IO），都可以很快的移植过去。

1. **AT24C02芯片**

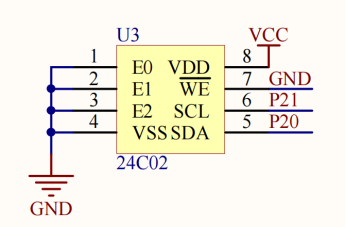
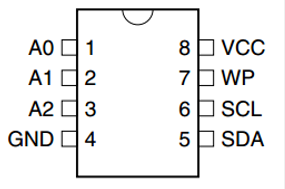
①**介绍**

一种可以实现掉电不丢失的存储器，可用于保存单片机运行时想要永久保存的数据信息。该器

件通过I2C总线接口进行操作，它有一个专门的写保护功能。存储介质为E2PROM。

②**开发板中AT24C02模块**

**VCC**：电源 **GND**：接地 **SCL**：时钟线

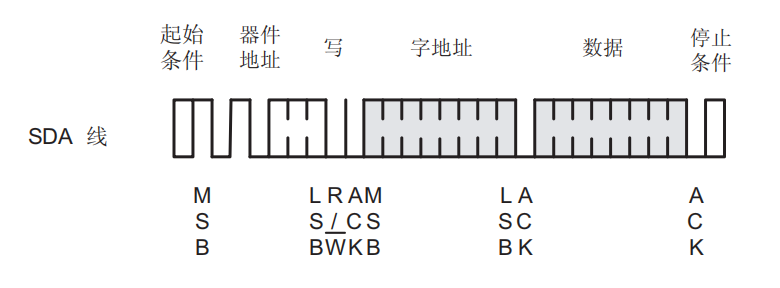
**WP**：写保护（高电平有效）  **SDA**：数据线

**A0,A1,A2**：输入硬件地址，故I2C总线上可同时级联8个AT24C02器件。

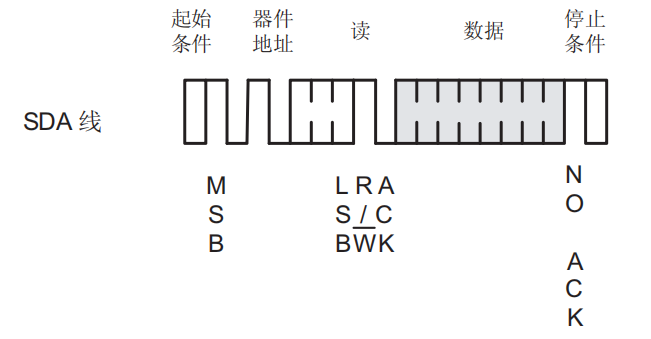
1. SDA为漏极开路，需外接上拉电阻到VCC，典型阻值为10KΩ。
2. AT24C02的固定地址为1010，可配置地址本开发板上为000，所以SLAVE ADDRESS+W

为0xA0，SLAVE ADDRESS+R为0xA1。

③**字节写数据帧**（在WORD ADDRESS处写入数据DATA）



④**字节读数据帧**（读出在WORD ADDRESS处的数据DATA）



1. **代码范例**

void I2C\_Start(void) //I2C开始

{

I2C\_SDA=1;

I2C\_SCL=1;

I2C\_SDA=0;

I2C\_SCL=0;

}

void I2C\_Stop(void) //I2C停止

{

I2C\_SDA=0;

I2C\_SCL=1;

I2C\_SDA=1;

}

void I2C\_SendByte(unsigned char Byte) //发送一个字节

{

unsigned char i;

for(i=0;i<8;i++)

{

I2C\_SDA=Byte&(0x80>>i);

I2C\_SCL=1;

I2C\_SCL=0;

}

}

unsigned char I2C\_ReceiveByte(void) //接收一个字节

{

unsigned char i,Byte=0x00;

I2C\_SDA=1;

for(i=0;i<8;i++)

{

I2C\_SCL=1;

if(I2C\_SDA){Byte|=(0x80>>i);}

I2C\_SCL=0;

}

return Byte;

}

void I2C\_SendAck(unsigned char AckBit) //发送应答

{

I2C\_SDA=AckBit;

I2C\_SCL=1;

I2C\_SCL=0;

}

unsigned char I2C\_ReceiveAck(void) //接收应答

{

unsigned char AckBit;

I2C\_SDA=1;

I2C\_SCL=1;

AckBit=I2C\_SDA;

I2C\_SCL=0;

return AckBit;

}

void AT24C02\_WriteByte(unsigned char WordAddress,Data) //AT24C02写入一个字节

{

I2C\_Start();

I2C\_SendByte(AT24C02\_ADDRESS);

I2C\_ReceiveAck();

I2C\_SendByte(WordAddress);

I2C\_ReceiveAck();

I2C\_SendByte(Data);

I2C\_ReceiveAck();

I2C\_Stop();

}

unsigned char AT24C02\_ReadByte(unsigned char WordAddress) //AT24C02读取一个字节

{

unsigned char Data;

I2C\_Start();

I2C\_SendByte(AT24C02\_ADDRESS);

I2C\_ReceiveAck();

I2C\_SendByte(WordAddress);

I2C\_ReceiveAck();

I2C\_Start();

I2C\_SendByte(AT24C02\_ADDRESS|0x01);

I2C\_ReceiveAck();

Data=I2C\_ReceiveByte();

I2C\_SendAck(1);

I2C\_Stop();

return Data;

}

1. **AD**
2. **AD**：模拟-数字转换，将模拟信号转换为计算机可操作的数字信号。
3. **ADC**：模数转换器，可将模拟信号转变为数字信号，采集模拟信号时通常需在前端加上A/D芯片。

**转换流程为**：取样保持->量化->编码

1. **转换指标**

①**分辨率**：对允许范围内的模拟信号进行转换，能输出离散数字值的个数。输出的信号通常用二进制存储，故分辨率单位通常为比特。

如：12位ADC的分辨率就是12位，或者说分辨率为满刻度的1/(2^12)。一个10V满刻度的12位ADC能分辨输入电压变化最小值是10V×1/(2^12 )=2.4mV。

②**转换误差**：实际输出的数字量和理论输出数字量之间的差距。常用最低有效位表示。

如：相对误差≤±LSB/2，这就表明实际输出的数字量和理论上应得到的输出数字量之间的误差小于最低位的半个字。

③**转换速率**：能够重复进行数据转换的速度，即每秒转换的次数。而完成一次A/D 转换所需的时间（包括稳定时间），则是转换速率的倒数。

1. **XPT2046芯片**

①**介绍**

一种典型的逐次逼近型模数转换器（SAR ADC），包含了采样/保持、模数转换、串口数据输出等

功能。内含12位分辨125KHz转换速率的A/D转换器，XPT2046采用微小的封装形式：TSSOP-16,QFN-16和VFBGA－48。与ADS7846、TSC2046、AK4182A完全兼容。

②**特性**

1）支持从1.5V到5.25V的低电压I/O接口，内置2.5V参考电压。

2）工作温度范围为-40℃～+85℃。

3）在2.7V的典型工作状态下，关闭参考电压，功耗可小于0.75mW。

4）可单电源供电，电压范围为2.7V~5.5V。

5）采用3线制SPI通信接口。

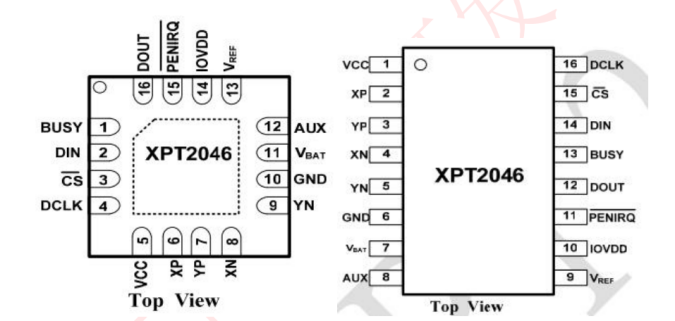
③**功能**

1）可通过执行两次A/D转换查出被按的屏幕位置。

2）可测量加在触摸屏上的压力、片内集成有一个温度传感器。

3）内部自带2.5V参考电压，可以作为辅助输入、温度测量和电池监测之用，电池监测的电压范围可以从0V到6V。

④**芯片图**



**VCC**：电源 GND：接地 **：**参考电压输入输出  **XP,YP,XN,YN**：输入端

：电池监视输入端 **IOVDD**：数字电源输入端 **PENIRQ**：笔接触中断引脚

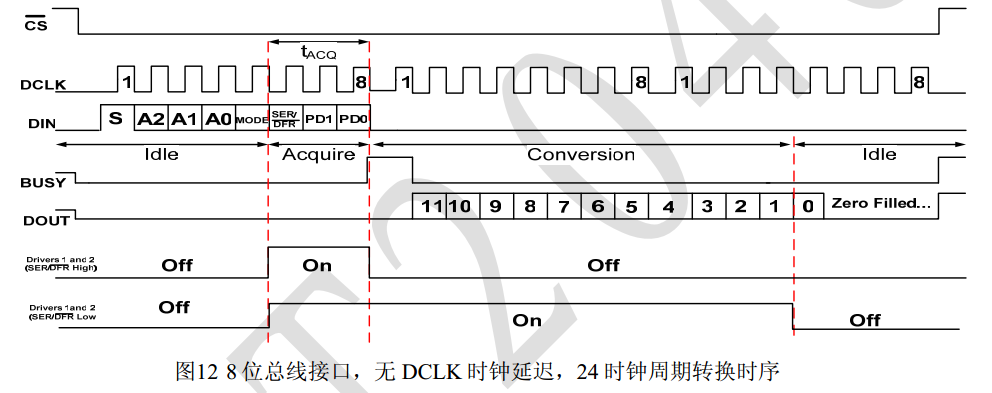
**AUX**：ADC辅助输入通道 **BUSY**：忙时信号线，CS为高电平时为高阻态

**CS**：控制转换时序和使能串行输入输出寄存器，高电平时ADC掉电 **DCLK**：外部时钟

**DIN**：串行数据输入端，CS为低电平时数据在DCLK上升沿锁存进来

**DOUT**：串行数据输出端，数据在DCLK的下降沿移出，CS为高电平时为高阻态

⑤**典型工作时序**



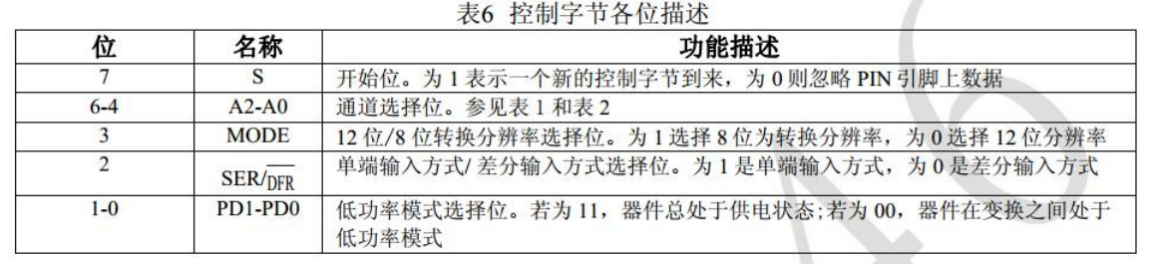
1）数据接口为串行接口。

2）处理器和转换器之间的通信需要8个时钟周期，可采用SPI、SSI和Microwire等同步串行接口。

3）一次完整的转换需要24个串行同步时钟（DCLK）来完成。前8个时钟用来通过DIN引脚输入控制字节。当转换器获取有关下一次转换的足够信息后，接着根据获得的信息设置输入多路选择器和参考源输入，并进入采样模式，如果需要，将启动触摸面板驱动器。3个多时钟周期后，控制字节设置完成，转换器进入转换状态。这时，输入采样－保持器进入保持状态，触摸面板驱动器停止工作（单端工作模式）。接着的12个时钟周期将完成真正的模数转换。如果是度量比率转换方式（SER/DFR＝0），驱动器在转换过程中将一直工作。第13个时钟将输出转换结果的最后一位。剩下的3个多时钟周期将用来完成被转换器忽略的最后字节（DOUT置低）。

⑥**命令字**







1. **DA**
2. DA：将数字信号转换为模拟信号。
3. **DAC**：数字模拟转换器，ADC把电压模拟信号转换成易于计算机存储、处理的数字编码，由计算机 处理完成后，再由DAC输出电压模拟信号，该电压模拟信号常常用来驱动某些执行器件，使人类易于感知。
4. **技术指标**

①**分辨率**：输入数字量的最低有效位（LSB）发生变化时，所对应的输出模拟量（电压或电流）的变化量，它反映了输出模拟量的最小变化值。

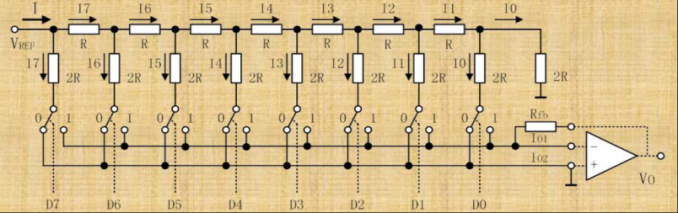
如：对于5V的满量程，采用８位的DAC时，分辨率为5V/256＝19.5mV；当采用12位的DAC时，分辨率则为5V/4096＝1.22mV。显然，位数越多分辨率就越高。

②**线性度**：实际转换特性曲线与理想直线特性之间的最大偏差。常以相对于满量程的百分数表示。如±１％是指实际输出值与理论值之差在满刻度的±１％以内。

③**精度**：在整个刻度范围内，任一输入数码所对应的模拟量实际输出值与理论值之间的最大误差。

④**建立时间**：输入的数字量发生满刻度变化时，输出模拟信号达到满刻度值的±1/2LSB所需的时间。是描述D/A转换速率的一个动态指标，根据建立时间 的长短，可以将DAC分成超高速 （＜1μS)、高速（10～1μS）、中速（100～10μS）、低速（≥100μS）几档。

1. **原理**（T型电阻网络）



输出电压计算公式：V = V’ \* Z / 256

**V**：输出电压 **V’**：参考电压 **Z**：单片机给的数字量 **256**：DAC精度位8位

DAC主要由数字寄存器、模拟电子开关、位权网络、求和运算放大器和基准电压源（或恒流源）

组成。用存于数字寄存器的数字量的各位数码，分别控制对应位的模拟电子开关，使数码为1的位在位权网络上产生与其位权成正比的电流值，再由运算放大器对各电流值求和，并转换成电压值。

1. **PWM**（出于成本考虑，在实际开发中，使用最多的是通过PWM来模拟DAC输出）

①**介绍**

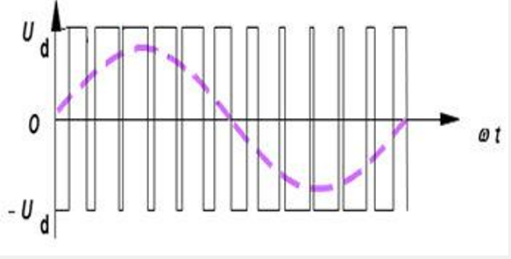
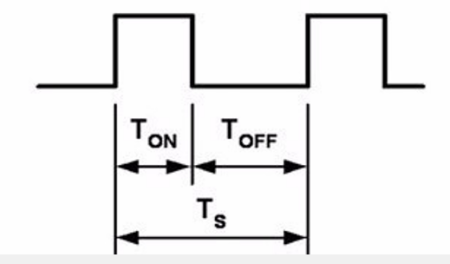
脉冲宽度调制，是利用微处理器的数字输出来对模拟电路进行控制的一种非常有效的技术。应用领域包括测量、通信、功率控制与变换，电动机控制、伺服控制、调光、开关电源，甚至某些音频放大器。

②**原理**

通过高分辨率计数器的使用，方波的占空比被调制用来对一个具体模拟信号的电平进行编码。

PWM信号仍然是数字的，因为在给定的任何时刻，满幅值的直流供电要么完全有(ON)，要么完全无(OFF)。电压或电流源是以一种通(ON)或断(OFF)的重复脉冲序列被加到模拟负载上去的。只要带宽足够，任何模拟值都可以使用PWM进行编码。

③**重要参数**

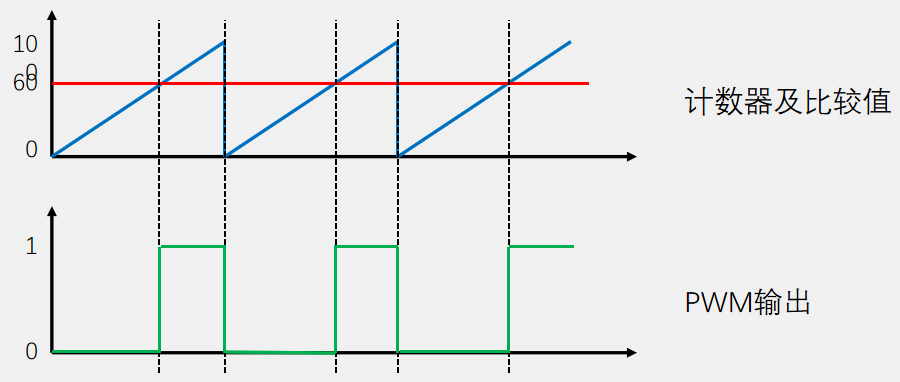
 

**频率** = 1 / TS **占空比** = TON / TS **精度** = 占空比变化步距

④**产生方法**

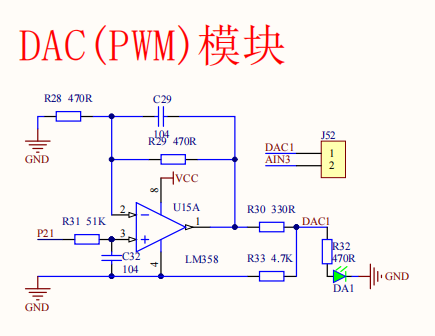


模型



波形

⑤**开发板中DAC（PWM）模块**



**P2^1**：控制引脚 **AIN3**：外部模拟信号输入

**DCA1**：信号输出端，将其连接一个LED，便可通过指示灯的状态直观反映PWM输出电压值的变化。

LM358芯片与这些电容电阻构成了一个跟随电路，即输入是多少，输出即为多大电压，输出电压范围是0-5V。如果需要检测PWM输出的电压大小，可以使用短接片将J52端子的DAC和AIN3短接，这样就可以使用XPT2046来检测模拟信号。

1. **代码**（LED呼吸灯）

sbit DA=P2^1;

unsigned char Counter,Compare; //计数值和比较值，用于输出PWM

unsigned char i;

Timer0\_Init(); //定时器0初始化，开始计数

while(1)

{

for(i=0;i<100;i++)

{

Compare=i; //设置比较值，改变PWM占空比

Delay(10);

}

for(i=100;i>0;i--)

{

Compare=i; //设置比较值，改变PWM占空比

Delay(10);

}

}

void Timer0\_Routine() interrupt 1

{

TL0 = 0x9C; //设置定时初值

TH0 = 0xFF; //设置定时初值

Counter++;

Counter%=100; //计数值变化范围限制在0~99

if(Counter<Compare) //计数值小于比较值

{

DA=1; //输出1

}

else //计数值大于比较值

{

DA=0; //输出0

}

}

1. **红外遥控**
2. **介绍**

红外遥控是一种无线、非接触控制技术，具有抗干扰能力强，信息传输可靠，功耗低，成本低，易实现等显著优点，被诸多电子设备特别是家用电器广泛采用，并越来越多的应用到计算机系统中。

1. **红外遥控系统**

①**红外发射装置**

红外遥控器，由键盘电路、红外编码电路、电源电路和红外发射电路组成。红外发射电路的主要元件为红外发光二极管。为了提高抗干扰性能和降低电源消耗，红外遥控器常用载波的方式传送二进制编码。通常的红外遥控器是将遥控信号（二进制脉冲码）调制在38KHz的载波上，经缓冲放大后送至红外发光二极管，转化为红外信号发射出去的。



二进制脉冲码的形式有多种，其中最为常用的是NEC Protocol的PWM码(脉冲宽度调制)和Philips RC-5 Protocol的PPM码(脉冲位置调制码，脉冲串之间的时间间隔来实现信号调制)。如果要开发红外接收设备，一定要知道红外遥控器的编码方式和载波频率，我们才可以选取一体化红外接收头和制定解码方案。我们配套的红外遥控器使用的是NEC协议。

②NEC协议特征

1）8位地址和8位指令长度；

2）地址和命令2次传输（确保可靠性）；

3）PWM脉冲位置调制，以发射红外载波的占空比代表“0”和“1”；

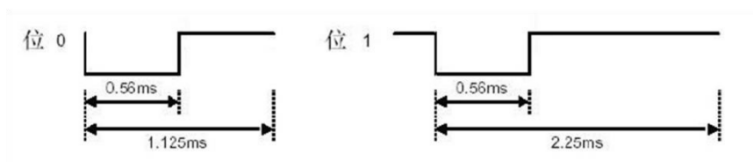
4）载波频率为38Khz；

5）位时间为1.125ms或2.25ms。

③NEC码（红外接收头在收到脉冲的时候为低电平，在没有脉冲的时候为高电平）

**逻辑1**：560us低+1680us高 **逻辑0**：560us低+560us高

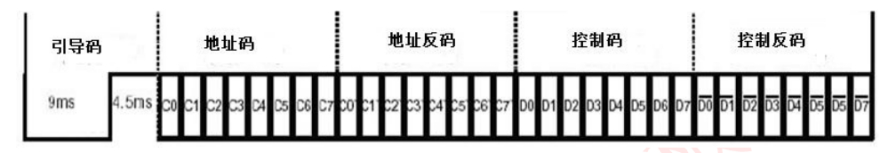
可以通过计算高电平时间判断接收到的数据是0还是1。时序图如下：



④**NEC遥控指令的数据格式**（引导码、地址码、地址反码、控制码、控制反码）

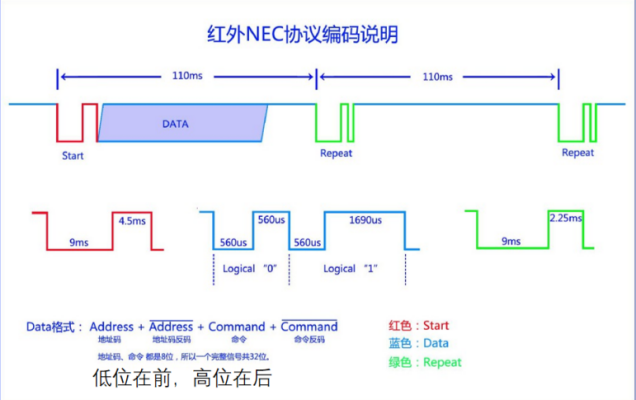
引导码由一个9ms的低电平和一个4.5ms的高电平组成，地址码、地址反码、控制码、控制反

码均是 8 位数据格式。按照低位在前，高位在后的顺序发送。采用反码是为了增加传输的可靠性（可用于校验）。数据格式如下：



NEC码还规定了连发码(由9ms低电平+2.5m高电平+0.56ms低电平+97.94ms高电平组成)，如

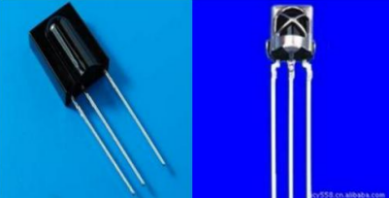
果在一帧数据发送完毕之后，红外遥控器按键仍然没有放开，则发射连发码，可以通过统计连发码的次数来标记按键按下的长短或次数。数据帧如下：



⑤**红外接收设备**

红外接收头，是由红外接收电路、红外解码、电源和应用电路组成。红外遥控接收器的主要作

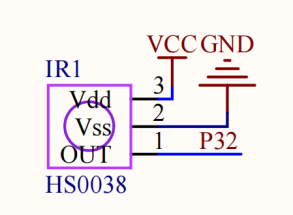
用是将遥控发射器发来的红外光信好转换成电信号，再放大、限幅、检波、整形，形成遥控指令脉冲，输出至遥控微处理器。通常采用成品红外接收头，封装大致有两种：一种采用铁皮屏蔽；一种是塑料封装。均有三只引脚，即电源正（VDD）、电源负（GND）和数据输出（VOUT）。其外观实物图如下图所示：



正对接收头的凸起处看，从左至右管脚依次是 VOUT、GND、VDD

由于红外接收头在没有脉冲时为高电平，当收到脉冲时为低电平，所以可以通过外部中断的下降沿触发中断，在中断内通过计算高电平时间来判断接收到的数据是0还是1。

⑥**开发板中红外遥控模块**（控制P3^2口即可控制红外遥控模块）



⑦**代码**

void Int0\_Init(void) //外部中断初始化

{

IT0=1;

IE0=0;

EX0=1;

EA=1;

PX0=1;

}

void Int0\_Routine(void) interrupt 0

{

if(IR\_State==0) //状态0，空闲状态

{

Timer0\_SetCounter(0); //定时计数器清0

Timer0\_Run(1); //定时器启动

IR\_State=1; //置状态为1

}

else if(IR\_State==1) //状态1，等待Start信号或Repeat信号

{

IR\_Time=Timer0\_GetCounter(); //获取上一次中断到此次中断的时间

Timer0\_SetCounter(0); //定时计数器清0

//若计时为13.5ms，则接收到Start信号(判定值在12MHz晶振下为13500，在11.0592MHz晶振下为12442)

if(IR\_Time>13500-500 && IR\_Time<13500+500)

{

IR\_State=2; //置状态为2

}

//如果计时为11.25ms，则接收到了Repeat信号（判定值在12MHz晶振下为11250，在11.0592MHz晶振下为10368）

else if(IR\_Time>11250-500 && IR\_Time<11250+500)

{

IR\_RepeatFlag=1; //置收到连发帧标志位为1

Timer0\_Run(0); //定时器停止

IR\_State=0; //置状态为0

}

else //接收出错

IR\_State=1; //置状态为1

}

else if(IR\_State==2) //状态2，接收数据

{

IR\_Time=Timer0\_GetCounter(); //获取上一次中断到此次中断的时间

Timer0\_SetCounter(0); //定时计数器清0

//如果计时为1120us，则接收到了数据0（判定值在12MHz晶振下为1120，在11.0592MHz晶振下为1032）

if(IR\_Time>1120-500 && IR\_Time<1120+500)

{

IR\_Data[IR\_pData/8]&=~(0x01<<(IR\_pData%8)); //数据对应位清0

IR\_pData++; //数据位置指针自增

}

//如果计时为2250us，则接收到了数据1（判定值在12MHz晶振下为2250，在11.0592MHz晶振下为2074）

else if(IR\_Time>2250-500 && IR\_Time<2250+500)

{

IR\_Data[IR\_pData/8]|=(0x01<<(IR\_pData%8)); //数据对应位置1

IR\_pData++; //数据位置指针自增

}

else //接收出错

{

IR\_pData=0; //数据位置指针清0

IR\_State=1; //置状态为1

}

if(IR\_pData>=32) //如果接收到了32位数据

{

IR\_pData=0; //数据位置指针清0

if((IR\_Data[0]==~IR\_Data[1]) && (IR\_Data[2]==~IR\_Data[3])) //数据验证

{

IR\_Address=IR\_Data[0]; //转存数据

IR\_Command=IR\_Data[2];

IR\_DataFlag=1; //置收到连发帧标志位为1

}

Timer0\_Run(0); //定时器停止

IR\_State=0; //置状态为0

}

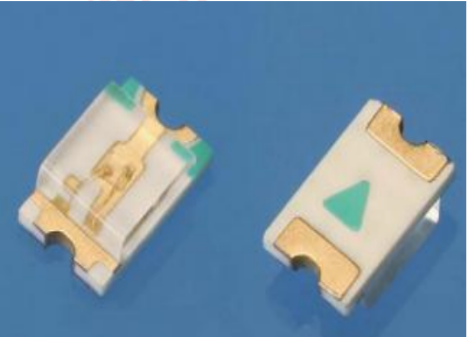
}

}

1. **外围器件**
2. **LED**

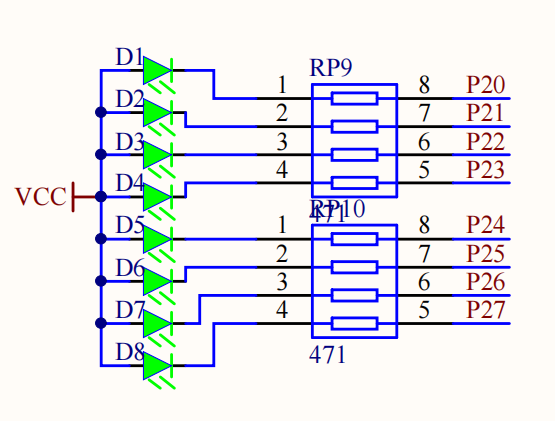
①**介绍**

发光二极管，具有单向导电性，通过5mA电流即可发光，电流越大，其亮度越强。但若电流过大，会烧毁二极管。通常我们会在LED管脚上串联一个电阻限制电流，即限流电阻。直插式二极管长脚为阳极，短脚为阴极。贴片式二极管通常有标记的一端为阴极。

** **

**发光二极管 贴片式二极管**

②**开发板内LED模块**



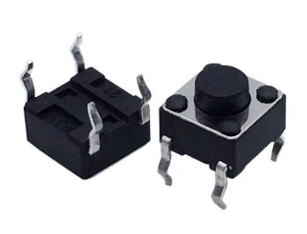
如图，LED采用共阳极接法，即所有LED阳极管脚接电源VCC，阴极管脚通过一个470欧的限流电阻接到P2口上。当I/O口P2为低电平时，二极管发光。

1. **独立按键**

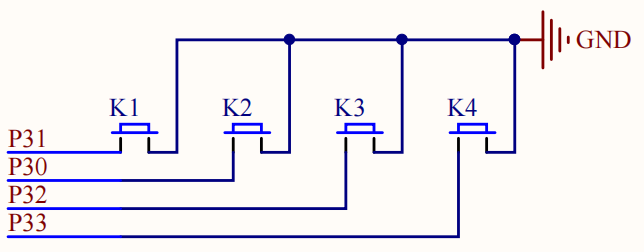
①**介绍**

轻触按键相当于是一种电子开关，按下时开关接通，松开时开关断开。实现原理是通过轻触按键

内部的金属弹片受力弹动来实现接通和断开。按键管脚两端距离长的表示默认是导通状态，距离短的默认是断开状态，如果按键按下，初始导通状态变为断开，初始断开状态变为导通。



②**开发板内独立按键模块**

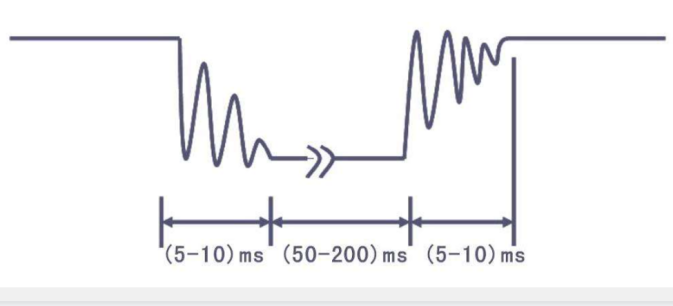


如图，4个独立按键的控制管脚直接连接到51单片机的I/O口P3^0~P3^3上。

③**按键消抖**

由于机械点的弹性作用，按键开关在闭合时不会马上稳定的接通，在断开时 也不会一下子断开，因而在闭合和断开的瞬间均伴随着一连串的抖动。抖动时间的长短由按键的机械特性决定的，一般为5ms到10ms。按键抖动会引起按 键被误读多次。为了确保CPU对按键的一次闭合仅作一次处理，必须进行消抖。

**消抖方式**：硬件消抖和软件消抖，通常采用软件消抖，即延时。

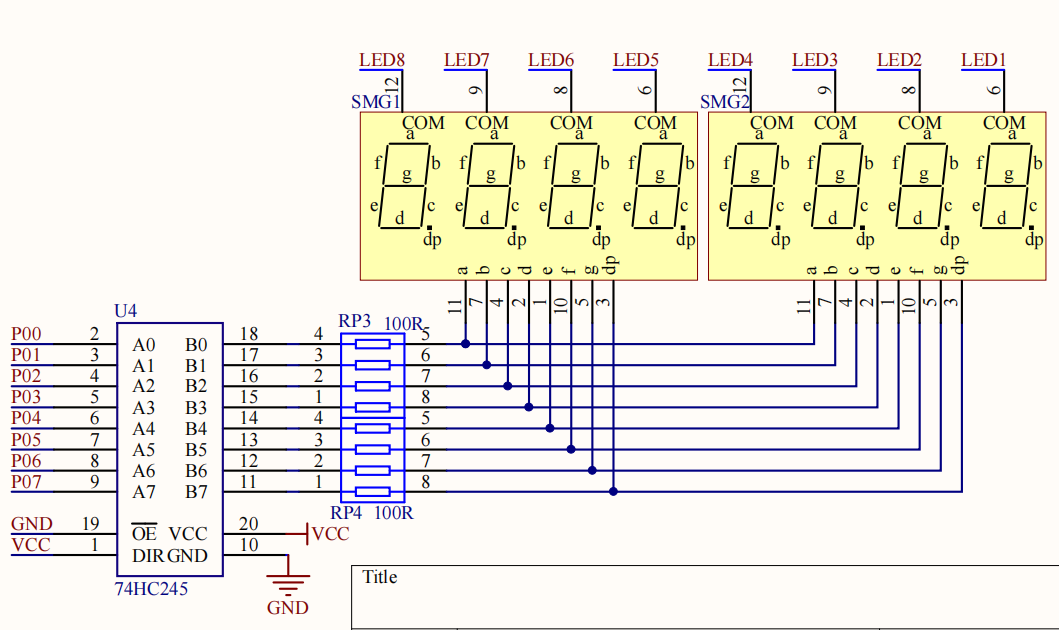


1. **数码管**

①**介绍**

一种简单、廉价的显示器，是由多个发光二极管封装在一起组成“8”字型的器件。多位数码管，即是两个或两个以上单个数码管并列集中在一起形成一体的数码管。当多位一体时，它们内部的公共端是独立的，而负责显示什么数字的段线全部是连接在一起的，独立的公共端可以控制多位一体中的哪一位数码管点亮，而连接在一起的段线可以控制这个能点亮数码管亮什么数字，通常我们把公共端叫做“位选线”，连接在一起的段线叫做“段选线”，有了这两个线后，通过单片机及外部驱动电路就可以控制任意的数码管显示任意的数字了。

②**开发板内数码管模块**



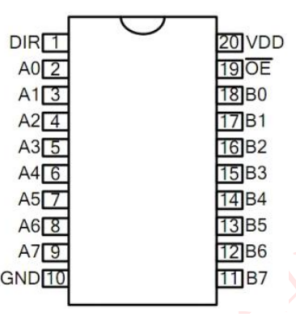
如图，动态数码管的控制管脚直接连接到51单片机的I/O上，段选端连接到P0上，位选端连接

到LED P2口上。段选端通过74HC245芯片驱动这两个共阴数码管的a-dp段。如果要想51单片机控制动态数码管，就必须通过单片机管脚输出低电平再有138译码器控制位选点亮数码管。

③**74HC245**

一种IO扩展芯片，三态输出、八路信号收发器。只需要很少的单片机IO口就可以扩展出8个控

制口，通过级联方式甚至可扩展出更多的控制口。



**管脚功能**：**A0-A7**：数据的输入输出 **B0-B7**：数据的输入输出 **OE**：输出使能

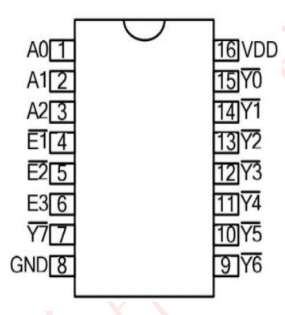
**DIR**：方向控制（1：A->B 0：B->A）

**VDD**：电源 **GND**：接地

**功能真值表**：



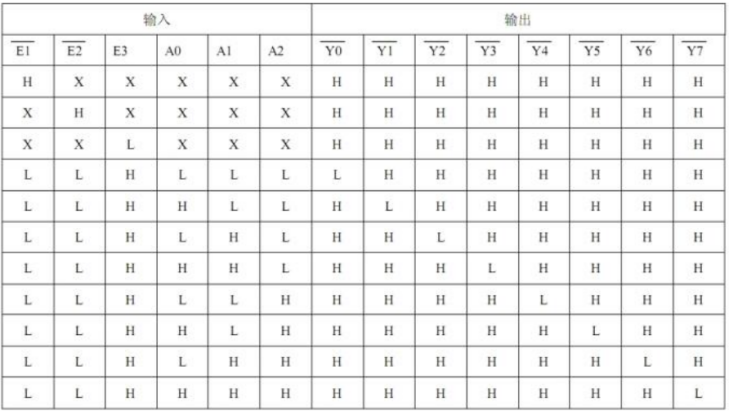
④**138译码器**（是一种三通道输入、八通道输出译码器）



**Y1-Y6,Y7**：数据输出 **A0-A2**：数据输入 **E1-E3**：使能控制

**VDD**：电源  **GND**：接地

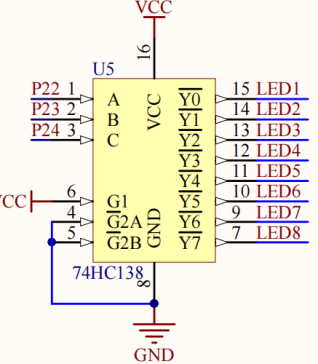
**真值表**：



观察真值表知，给 E1、E2 使能管脚低电平，E3 管脚为高电平，至于哪个管脚输出有效电平（低

电平），要看 A0，A1，A2 输入管脚的电平状态。A0、A1、A2输入就相当于3位2进制数，A0是低位，A1是次高位，A2是高位。而Y0-Y7具体哪一个输出有效电平，就看输入二进制对应的十进制数值。比如输入是101(A2，A1，A0)，其对应的十进制数是5，所以Y5输出有效电平(低电平)。

**开发板内138芯片模块**：



⑤**数码管动态显示**

利用减少段选线，分开位选线，利用位选线不同时选择通断，改变段选数据来实现的。比如在第

一次选中第一位数码管时，给段选数据0，下一次选中第二位数码管时显示1。为了在显示1的时候，0不会消失（ 当然实际上是消失了），必须在人肉眼观察不到的时间里再次点亮第一次点亮的0。而这时就需要记住，人的肉眼正常情况下只能分辨变化超过 24ms间隔的运动。也就是说，在下一次点亮0这个数字的时间差不得大于24ms。这时就会发现，数码管点亮是在向右或者向左一位一位点亮，形成了动态效果。

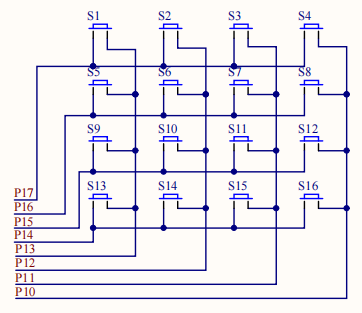
1. **矩阵按键**

①**介绍**

开发板上将16个按键排成4行4列，第一行将每个按键的一端连接在一起构成行线，第一列将每

个按键的另一端连接在一起构成列线，这样便一共有4行4列共8根线，我们将这8根线连接到单片机的8个I/O口上，通过程序扫描键盘就可检测16个键。

②**开发板中矩阵按键模块**



如图，矩阵按键引出的8根控制管脚直接连接到单片机的 P1口上。

③**检测扫描方式**

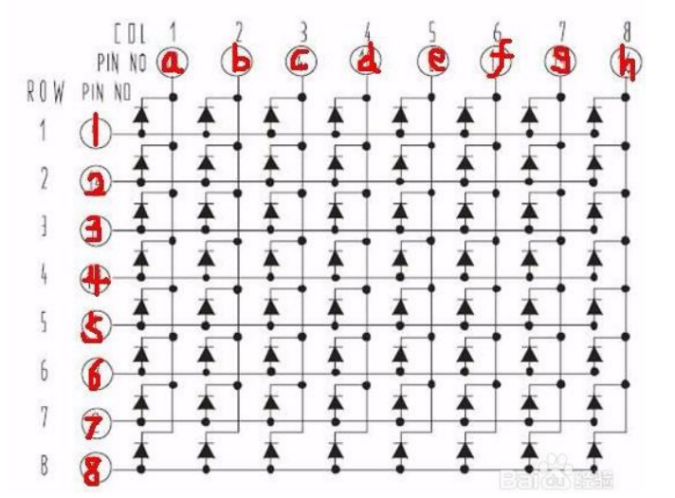
**行列扫描**：先送一列为低电平，其余几列全为高电平(此时我们确定了列数)，然后立即轮流检测一次各行是否有低电平，若检测到某一行为低电平(这时我们又确定了行数)，则我们便可确认当前被按下的键是哪一行哪一列的。用同样方法轮流送各列一次低电平，再轮流检测一次各行是否变为低电平，这样即可检测完所有的按键，当有键被按下时便可判断出按下的键是哪一个键。当然我们也可以将行线置低电平，扫描列是否有低电平。从而达到整个键盘的检测。

**线翻转法**：使所有行线为低电平时，检测所有列线是否有低电平。如果有，就记录列线值，然后再翻转，使所有列线都为低电平，检测所有行线的值。由于有按键按下，行线的值也会有变化，记录行线的值，从而就可以检测到全部按键。

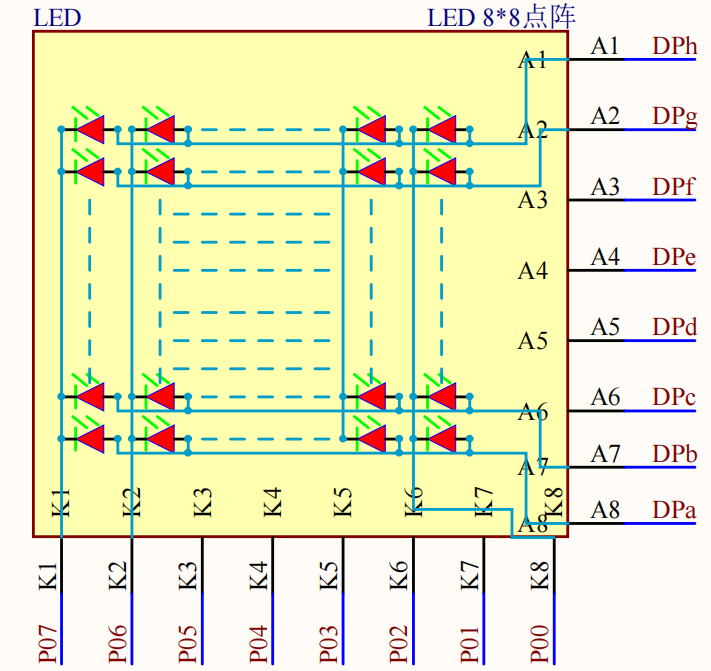
1. **LED点阵屏**

①**介绍**

由发光二极管排列组成的显示器件。8\*8点阵共由64个发光二极管组成，且每个发光二极管是放置在行线和列线的交叉点上。当对应的某一行置1电平，某一列置0电平，则相应的二极管就亮：例如要将第一个点点亮，则1脚接高电平，a脚接低电平，则第一个点就亮了；如果要将第一行点亮，则第1脚要接高电平，而（a、b、c、d、e、f、g、h ）这些引脚接低电平，那么第一行就会点亮；如要将第一列点亮，则第a脚接低电平，而（1、2、3、4、5、6、7、8）接高电平，那么第一列就会点亮。

②**开发板中LED点阵模块**



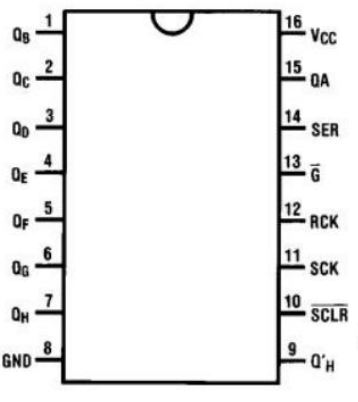
如图，如果点阵引脚直接与单片机相连，需要占用16个I/O口。为减少I/O口占用，借助74HC595

驱动点阵屏。

③**74HC595芯片**

一个8位串行输入、并行输出的位移缓存器，其中并行输出为三态输出（即高电平、低电平和高

阻抗）。数据在SCK的上升沿输入，在RCK的上升沿进入到存储器中。如果两个时钟连在一起，则移位寄存器总是比存储器早一个脉冲。移位寄存器有一个串行输入、一个串行输出和一个异步的低电平复位。存储寄存器有一个并行8位的、具有三态的总线输出。



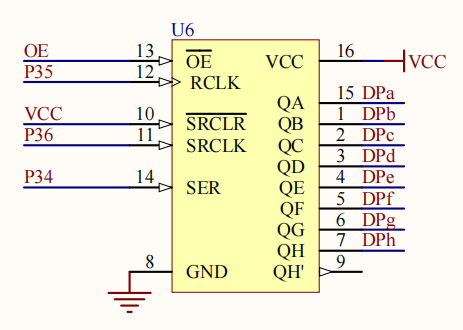
管脚功能：**SCK**：移位寄存器时钟输入  **SCLR**：低电平复位引脚。

**RCK**：存储寄存器时钟输入 **OE**：输出有效

**SER**：串行数据输入  **QH**：串行数据输出

**QA-QH**：并行数据输出 **VCC**：电源 **GND**：接地

④**开发板中的74HC595模块**



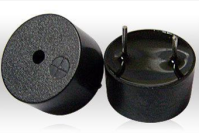
如图，输入SER、SRCLK、RCLK与单片机P3^4-P3^6相连，输出与LED点阵相连。SER串行输入数据；数据在SRCLK上升沿进入移位寄存器；在RCLK上升沿时数据寄存器的数据移位：QA-->QB-->QC-->...-->QH，并将数据输出到并行端口。如果两个时钟连在一起，则移位寄存器总是比存储寄存器早一个脉冲。

1. **蜂鸣器**

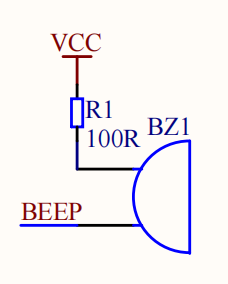
①**介绍**

一种一体化结构的电子讯响器，采用直流电压供电。蜂鸣器主要分为有源蜂鸣器（内部自带振荡

源）和无源蜂鸣器（内部没有振荡源）两种类型。想要无源蜂鸣器发声，需提供一定频率的脉冲信号；想要有源蜂鸣器发声，只需提供电源即可。开发板上使用的是无源蜂鸣器。



②**开发板中的蜂鸣器模块**



因为51单片机I/O口的驱动能力较弱（即使外接上拉电阻），而蜂鸣器驱动需要约 30mA，所以

非常困难。即使可以驱动，那对于整个芯片的其I/O剩下驱动能力就更加弱甚至无法工作。所以我们不会直接使用I/O口驱动蜂鸣器，而是通过三极管把电流放大后再驱动蜂鸣器，这样51单片机的I/O口只需要提供不到1mA的电流就可控制蜂鸣器。自己设计电路时，可用如下方式。



图中TP1 PNP三极管起到一个电流放大作用，从而可以驱动蜂鸣器。当J7端子有一个高电平进来

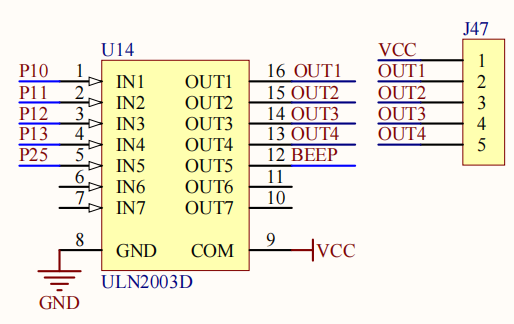
时，PNP三极管TP1截止，蜂鸣器不得电，当J7端子有一个低电平进来时，PNP三极管TP1导通，蜂鸣器得电。如果J7端子有一个一定频率的脉冲信号（高低电平不断翻转）时，无源蜂鸣器发出声音。

而有源蜂鸣器则只需电源即可发声，通过电阻R2和PNP三极管TP1进行电流放大，从而驱动蜂鸣器。电阻R1是一个上拉电阻，用来防止蜂鸣器误发声。当BEEP引脚输出低电平时，PNP三极管导通，蜂鸣器发声； 当BEEP引脚输出高电平时，PNP三极管截止，蜂鸣器停止发声。

③**ULN2003芯片**

一个单片高电压、高电流的达林顿晶体管阵列集成电路。它是由7对NPN达林顿管组成的，它的高电压输出特性和阴极箝位二极管可以转换感应负载。单个达林顿对的集电极电流是500mA。达林顿管并联可以承受更大的电流。此电路主要应用于继电器驱动器，字锤驱动器，灯驱动器，显示驱动器（LED气体放电），线路驱动器和逻辑缓冲器。

④**开发板中五线四项步进电机模块**



如图，蜂鸣器用ULN2003芯片驱动，另一端与单片机P2^5相连。由于开发板已将电路集成，所

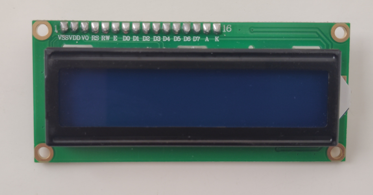
以控制单片机P2^5引脚即可控制蜂鸣器。

1. **LCD1602液晶显示屏**

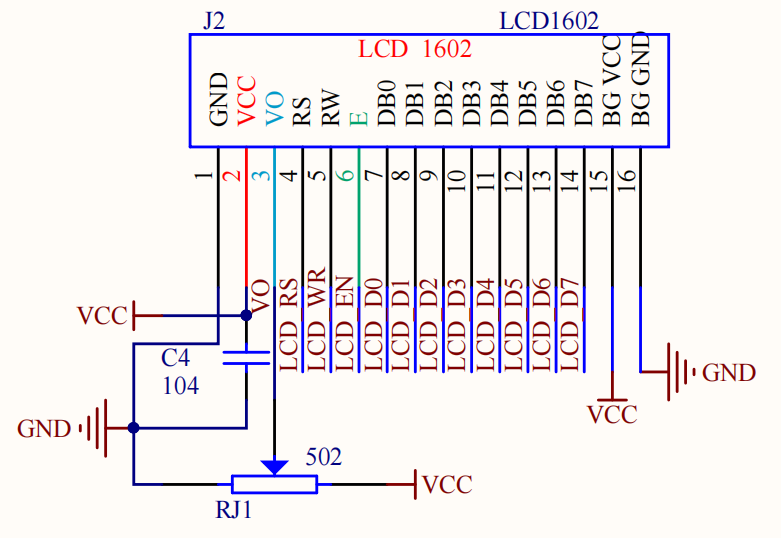
①**介绍**

也叫1602字符型液晶，它能显示2行字符信息，每行又能显示16个字符。它是一种专门用来显

示字母、数字、符号的点阵型液晶模块。它是由若干个5x7或者 5x10的点阵字符位组成，每个点阵字符位都可以用显示一个字符，每位之间有一个点距的间隔，每行之间也有间隔，起到了字符间距和行间距的作用，正因为如此，所以它不能很好的显示图片。

②**开发板中LCD1602模块**

LCD1602与单片机连接情况如上图，即可控制单片机对应引脚进而控制显示屏。

功能介绍：**VCC**：电源  **GND**：接地

**A**：背光灯电源正极 **K**：背光灯电源负极

**VO**：对比度调节电压  **RS**：数据/指令选择，1为数据，0为指令

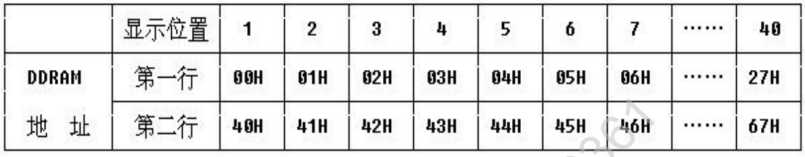
**RW**：读/写选择，1为读，0为写

**E：**使能，1为数据有效，下降沿执行命令

**D0-D7**：数据输入/输出

③**DDRAM**

显示数据RAM，用来寄存待显示的字符代码。共80个字节，其地址和屏幕的对应关系如下表：

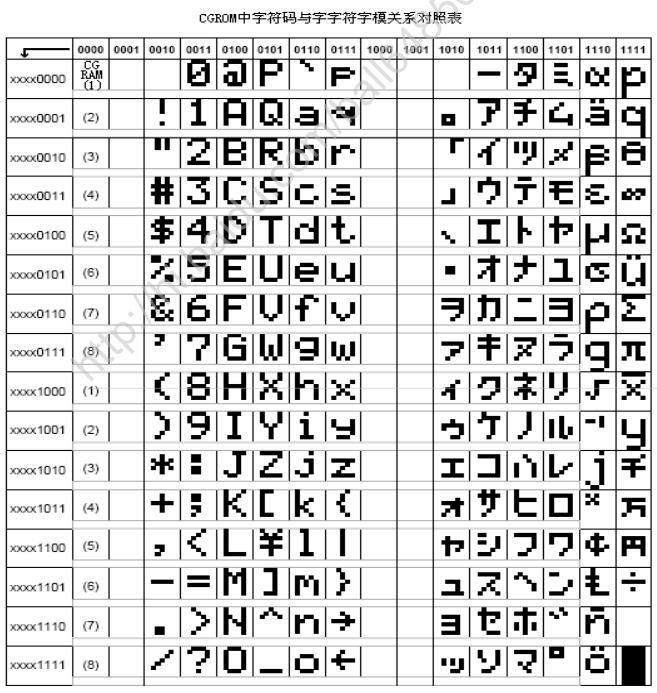


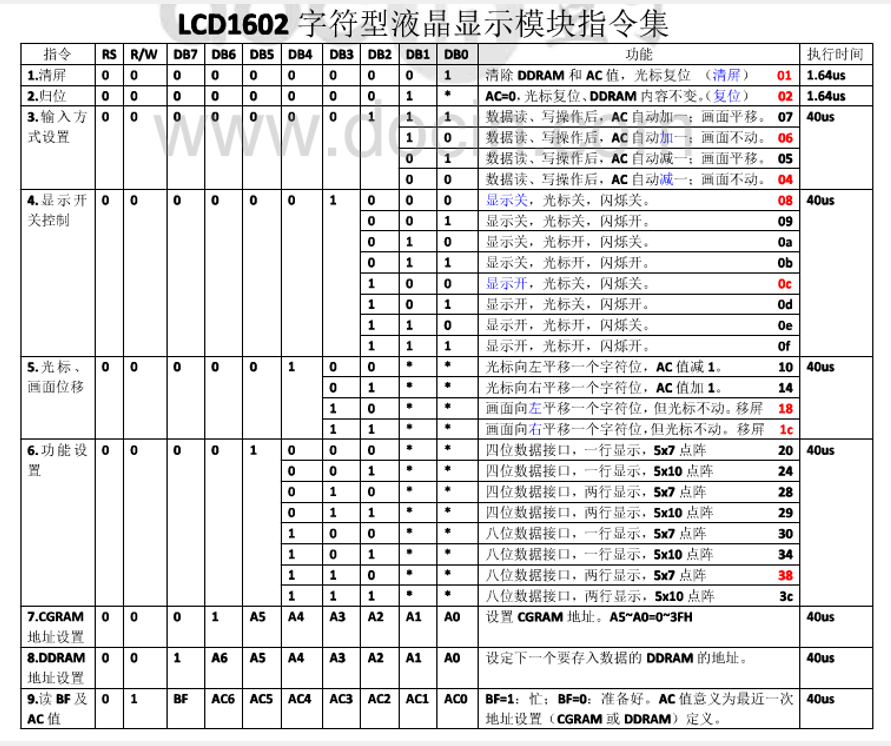
想要在LCD1602屏幕的第一行第一列显示一个“A”字,向DDRAM的00H地址写入A字的代码就行了。

④**CGROM和CGRAM**

在LCD模块上固化的字模存储器。HD44780内置了192个常用字符的字模，存于字符产生器

CGROM中，另外还有8个允许用户自定义的字符产生RAM，称为CGRAM。下图说明了CGROM和CGRAM与字符的对应关系。字模库如下：





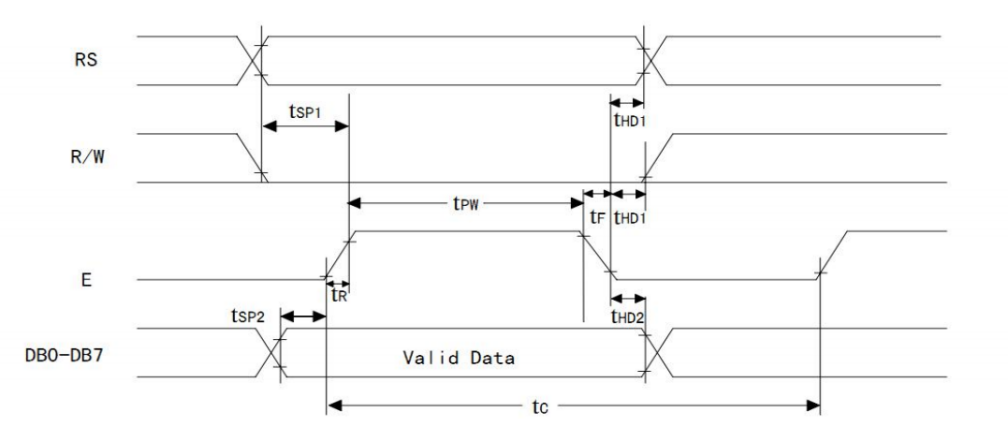
⑥**LCD1602的使用**

1. 写入特定命令，对其初始化
2. 写入特定命令，设置显示坐标
3. 将所要显示的数据发送到LCD的DDRAM

注：①当要写指令字，设置LCD1602的工作方式时：需要把RS置为低电平，RW置为低电平，然后将数据送到数据口D0~D7，最后E引脚一个高脉冲将数据写入。

②当要写入数据字，在1602上实现显示时：需要把RS置为高电平，RW置为低电平，然后将数据送到数据口D0~D7，最后E引脚一个高脉冲将数据写入。即写指令和写数据，差别仅仅在于RS的电平不一样而已。

⑦**时序图**





从上图可以看到，以上给的时间参数全部是ns级别的，而51单片机的机器周期是1us，指令周

期是2-4个机器周期，所以即便在程序里不加延时程序，也可以很好的配合LCD1602的时序要求了。

⑧**写命令时序**

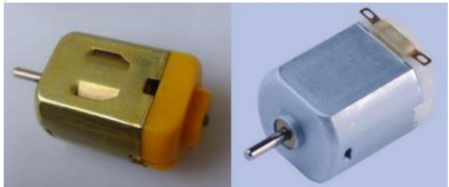
时间由左往右，RS变为低电平，R/W 变为低电平，注意看是RS的状态先变化完成。然后这时，DB0~DB7上数据进入有效阶段，接着E引脚有一个整脉冲的跳变，接着要维持时间最小值为tpw=400ns的E脉冲宽度。然后E引脚负跳变，RS电平变化，R/W电平变化。

1. **直流电机**

①**介绍**

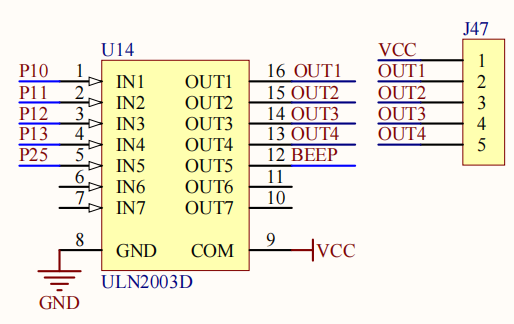
指能将直流电能转换成机械能（直流电动机）或将机械能转换成直流电能（直流发电机）的旋转

电机。当它作电动机运行时是直流电动机，将电能转换为机械能；作发电机运行时是直流发电机，将机械能转换为电能。直流电机的结构应由定子和转子两大部分组成。直流电机运行时静止不动的部分称为定子，定子的主要作用是产生磁场，由机座、主磁极、换向极、端盖、轴承和电刷装置等组成。运行时转动的部分称为转子，其主要作用是产生电磁转矩和感应电动势，是直流电机进行能量转换的枢纽，所以通常又称为电枢，由转轴、电枢铁心、电枢绕组、换向器和风扇等组成。直流电机没有正负之分，在两端加上直流电就能工作。需要知道直流电机的额定电压和额定功率，不能使之长时间超负荷运作。在交换接线后，可以形成正反转。



和蜂鸣器一样，直流电机不可由I/O口直接驱动，也需使用单片机控制ULN2003E进行驱动。

下图为步进电机芯片（ULN2003）引脚图：

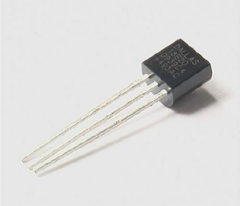
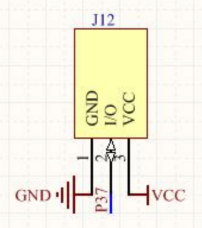


如图，将直流电机两个引脚与VCC和OUT1相连，控制P1^0即可控制直流电机，使用PWM可进行调速。

1. **温度传感器（DS18B20）**

①**介绍**

一种的“一线总线（单总线）”接口的温度传感器。与传统的热敏电阻等测温元件相比，它是一种新型的体积小、适用电压宽、与微处理器接口简单的数字化温度传感器。与微处理器连接时仅需要一条口线即可实现与微处理器的双向通讯。使用中不需要任何外围元件，全部传感元件及转换电路集成在形如一只三极管的集成电路内。支持多点组网功能，多个DS18B20可以并联在唯一的三线上，实现组网多点测温。测量结果直接输出数字温度信号，以"一根总线"串行传送给CPU，同时可传送CRC校验码，具有极强的抗干扰纠错能力。可编程的分辨率为9～12位，对应的可分辨温度分别为0.5℃、0.25℃、0.125℃和0.0625℃，可实现高精度测温。在9位分辨率时最多在93.75ms内把温度转换为数字，12位分辨率时最多在750ms内把温度值转换为数字，速度更快。单片机中DS18B20模块如下：

注：**电压范围**：3.0～5.5V，在寄生电源方式下可由数据线供电。

**温度范围**：－55℃～+125℃，在-10～+85℃时精度为±0.5℃。

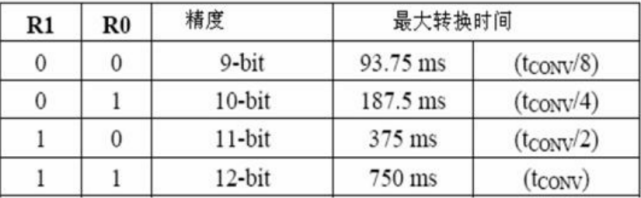
电源极性接反时，芯片不会因发热而烧毁，但不能正常工作。

②**芯片配置**

DS18B20温度传感器的内部存储器包括一个高速的暂存器RAM和一个非易失性的可电擦除的EEPROM,后者存放高温度和低温度触发器TH、TL和配置寄存器。配置寄存器是配置不同的位数来确定温度和数字的转化，配置寄存器结构如下：



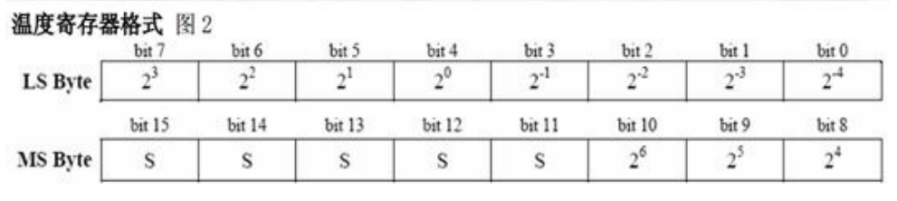
低五位一直都是"1"，TM是测试模式位，用于设置DS18B20在工作模式还是在测试模式。在DS18B20出厂时该位被设置为0，用户不需要去改动。R1和R0用来设置DS18B20的精度（分辨率），可设置为9，10，11或12位，对应的分辨率温度是 0.5℃，0.25℃，0.125℃和 0.0625℃。R0和R1配置如下图：（初始状态默认的精度是12位，即R0=1、 R1=1。）



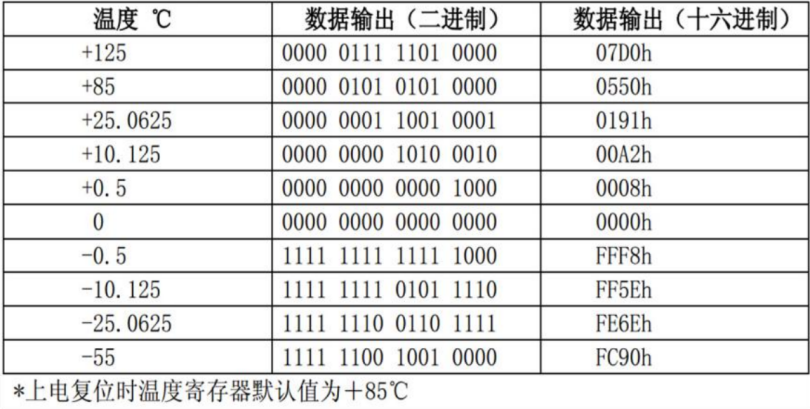
高速暂存存储器由9个字节组成，其分配如下：



当温度转换命令（44H）发布后，经转换所得的温度值以二字节补码形式存放在高速暂存存储器的第0和第1个字节。高字节的前5位是符号位S，单片机可通过单线接口读到该数据，读取时低位在前，高位在后，数据格式如下：



如果测得的温度大于0，这5位S为0，只要将测到的数值乘以0.0625即可得到实际温度；若温度小于0，这5位S为1，测到的数值需要取反加1再乘以0.0625即可得到实际温度。温度与数据对应关系如下：



③**计算温度**

比如：要计算+85度，数据输出十六进制是0X0550，因为高字节的高5位为0，表明检测的温度是正温度，0X0550对应的十进制为1360，将这个值乘以12位精度0.0625，所以可以得到+85度。

再如：要计算-10.125度，高五位为1，说明温度小于零。二进制数值1111 1111 0101 1110取反加一后为：0000 0000 1010 0010，其对应的十进制为162，乘以0.0625即为10.125。

④**读取温度**

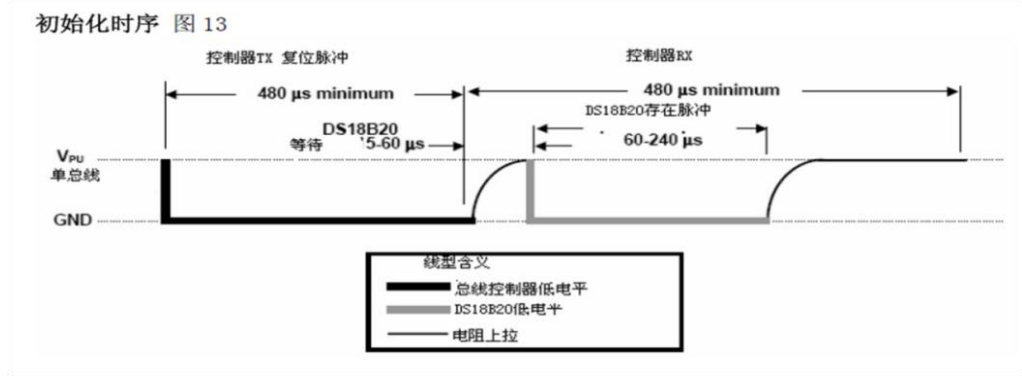
由于DS18B20是单总线器件，所有的单总线器件都要求采用严格的信号时序，以保证数据的完整性。DS18B20 时序包括如下几种：

1. 初始化时序：单总线上的所有通信都是以初始化序列开始。

Ⅰ：主机输出低电平，保持低电平480~960us以产生复位脉冲。

Ⅱ：主机释放总线，外部的上拉电阻将单总线拉高，延时15~60us进入接收模式。Ⅲ：DS18B20拉低总线60~240 us，以产生低电平应答脉冲，若为低电平，还要做延时，其延时的时间从外部上拉电阻将单总线拉高算起最少要480微妙。

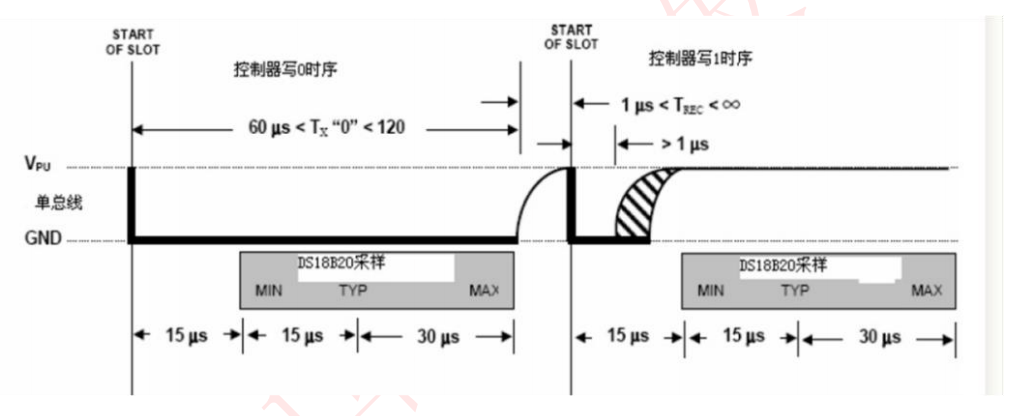
初始化时序图如下：



1. 写0和1时序

写时序包括写0时序和写1时序。所有写时序至少需要60us，且在两次独立的写时序之间至少需要1us的恢复时间，两种写时序均起始于主机拉低总线。写1时序：主机输出低电平，延时2us，然后释放总线，延时60us。写0时序：主机输出低电平，延时60us，然后释放总线，延时2us。

写时序图如下：

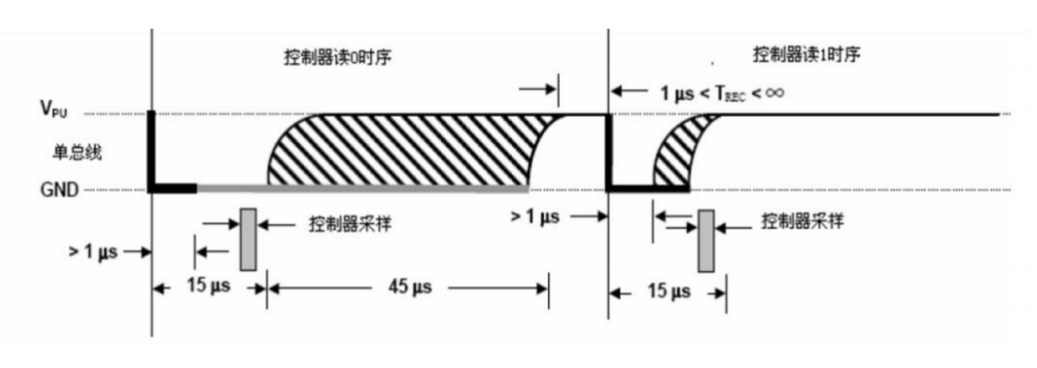


1. **读0和1时序**

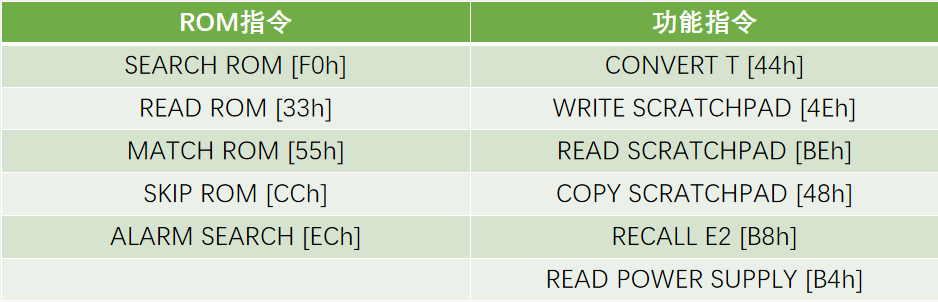
单总线器件在主机发出读时序时，才向主机传输数据，所以在主机发出读数据命令后，必须马上产生读时序，以便从机能够传输数据。所有读时序至少需要60us，且在两次独立的读时序之间至少需要1us的恢复时间。每个读时序都由主机发起，至少拉低总线1us。主机在读时序期间必须释放总线，并且在时序起始后的15us之内采样总线状态。

**读时序**：主机输出低电平延时2us，然后主机转入输入模式延时12us，随后读取单总线当前的电平，接着延时50us。

**读时序图如下**：



⑤**ROM指令**



⑥**读取过程**

复位→发SKIP ROM命令（0XCC）→发开始转换命令（0X44）→延时→复位

→发SKIP ROM命令（0XCC）→发读存储器命令（0XBE）→连续读出两个字节数据(即温度)→结束

**注**：DS18B20发送所有的命令和数据都是字节的低位在前。