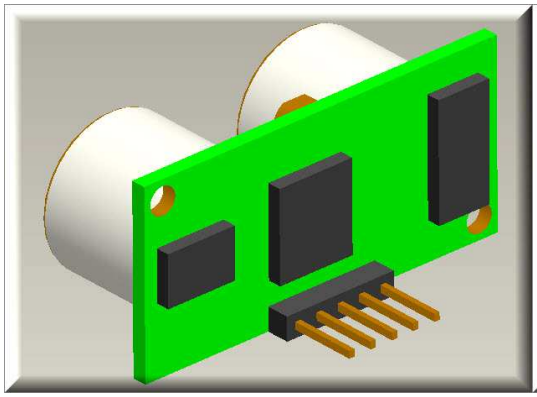
KS101B/KS103/KS103S 技术说明书

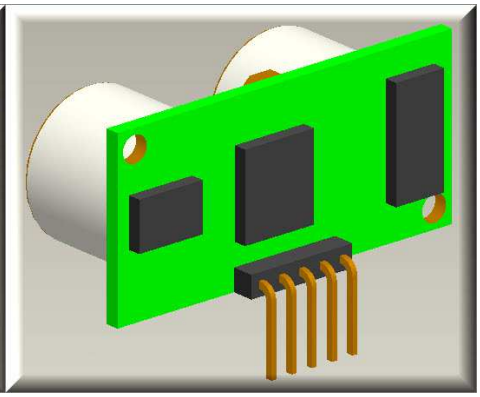
版本：Rev 3.58

日期：2011.03.10

Modify Date: 2014.05.23



KS103 VERTICAL(直针)



KS103 BENDING(弯针)

KS101B/KS103/KS103S 功能摘要：

- 包含实时温度补偿的距离探测，高探测精度
- 采用专利技术的探测模式，探测范围 1cm～800cm(见第 18 页)及 1cm～1000cm(10 米)
- 探测频率可达 500Hz，即每秒可探测 500 次
- 使用 ***I²C/串口***接口与主机通信，自动响应主机的 ***I²C/串口***控制指令
- 共 20 个可修改的 ***I²C/串口***地址，范围为 0xd0 ~ 0xfe (0xf0,0xf2,0xf4,0xf6 除外，8 位地址)
- I^2C 模式支持 0x00 广播地址(KS103/KS103S 不支持)
- 83ms 快速、高精度的温度探测，随时感知环境精确温度
- 5s 未收到 I^2C 控制指令自动进入 uA 级休眠，并可随时被主机 I^2C 控制指令唤醒
- 短距探测量程由 10cm、20cm、……、至 470cm，满足快速近距探测
- 1ms 快速光强探测，即时探测实时光强
- 使用工业级配置，工作温度 (-30℃~+85℃)
- 宽工作电压范围 (3.0V~5.5V)
- I^2C 模式通信速率 50~100kbit/s
- 采用独特的**可调**滤波降噪技术，电源电压受干扰或噪音较大时，仍可正常工作
- 环保无铅

三者区别为，KS101B 在 I^2C 模式时支持广播地址，支持温度修正的距离探测及温度探测；KS103 及 KS103S 均不支持广播地址，且 KS103 支持温度修正的距离探测及温度探测，而 KS103S 不支持温度相关的功能。

KS101B/KS103/KS103S 电性能参数：

工作电压：***3.0V~5.5V*** 直流电源

工作时瞬间最大电流：***10.6mA@5.0V, typical***

工作电流：***1.6-2.7 mA@5.0V, typical***

休眠时最大耗电量：***500uA@5.0V, typical*** （串口模式时不休眠）

功耗：使用纳瓦技术省电，5s 未收到 I^2C 控制指令自动进入 uA 级休眠，并可随时被主机 I^2C 控制指令唤醒。

在 KS101B/KS103/KS103S 上连线引脚上标识有：VCC、SDA/TX(简称 SDA)、SCL/RX(简称 SCL)、GND 及 MODE。MODE 引脚为 I²C 模式与 TTL 串口模式设置引脚，该引脚悬空时，KS101B/KS103/KS103S 工作于 I²C 模式；在上电之前 MODE 引脚接 0V 地时，KS101B/KS103/KS103S 工作于 TTL 串口模式。此处的 TTL 串口不是 232 串口，TTL 电平可以与单片机的 TXD/RXD 直接相连，但不能与 232 串口直接相连(直接连将烧坏本模块)，需要一个 MAX232 电平转换将 TTL 电平转换为 232 电平才可以。

I²C 模式

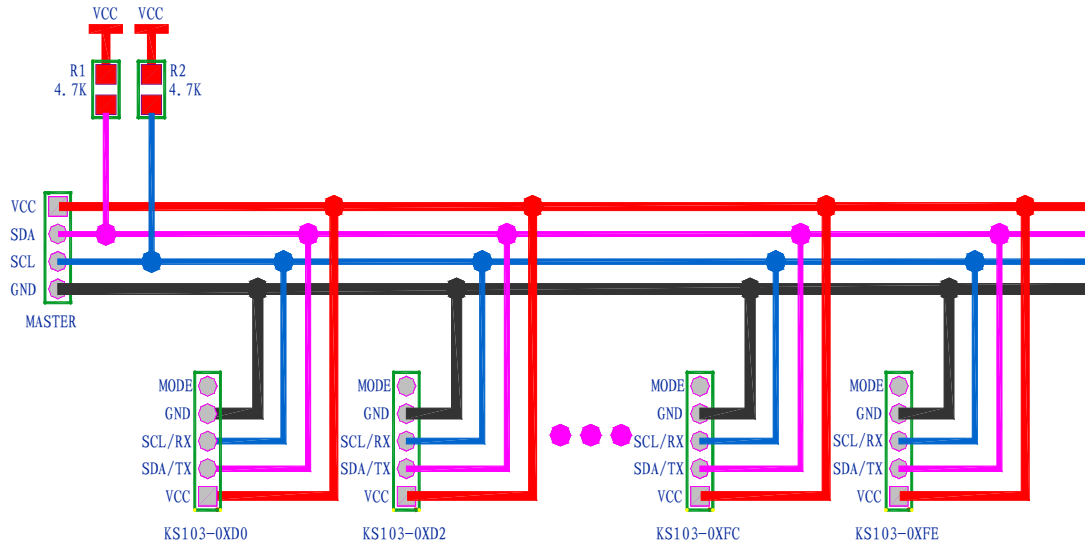
KS101B/KS103/KS103S 连线：

在 KS101B/KS103/KS103S 上连线引脚上标识有：VCC、SDA/TX(简称 SDA)、SCL/RX(简称 SCL)、GND 及 MODE。MODE 引脚悬空时，KS101B/KS103/KS103S 工作于 I²C 模式。

其中 VCC 用于连接+5V(3.0～5.5V 范围均可)电源(1)，GND 用于连接电源地，SDA/TX 是 I²C 通信的数据线，SCL/RX 引脚是 I²C 通信的时钟线。SCL 及 SDA 线均需要由主机接一个 4.7K(阻值 1～10K 均可)电阻到 VCC。KS101B/KS103/KS103S 的 I²C 通信速率建议不要高于 100kbit/s。

Note 1: 要达到最佳的工作状态推荐使用+5V 电源，低于 5V 的电压将影响测距量程。并且，严禁将 VCC 与 GND 接反，否则可能会损坏电路。超过 3 秒钟的电路反接将可能导致不可恢复的损坏。

具体连线如下图所示（20 个）：



KS101B/KS103/KS103S 默认地址为 0xe8，用户可以将地址修改为 20 种地址中的任何一个：0xd0,

0xd2, 0xd4, 0xd6, 0xd8, 0xda, 0xdc, 0xde, 0xe0, 0xe2, 0xe4, 0xe6, 0xe8, 0xea, 0xec, 0xee, 0xf8, 0xfa, 0xfc, 0xfe. (2)

Note 2: 请注意，以上地址并不包括 0xf0, 0xf2, 0xf4,0xf6，这 4 个地址保留用于 I²C 从机的 10 位地址。控制本模块的主机设备可能只支持 7 位的 I²C 从机地址，此时需要将 8 位地址右移 1 位作为地址来使用。例如，本模块默认地址 0xe8，对应 7 位的地址 0x74。

修改 I²C 地址时序：

地址	2	**0x9a**	延时 1ms	**地址**	2	**0x92**	延时 1ms	**地址**	2	**0x9e**	延时 1ms	**地址**	2	**新地址**	延时 100ms

修改 I²C 地址须严格按照时序来进行，时序中的延时时间为最小时间。对于 51 单片机主机，其可调用附件 3 所示的 change_i2c_address(addr_old,addr_new)函数来实现。

修改完毕后请给 KS101B/KS103/KS103S 重新上电，可观察到 LED 显示新地址。在修改 KS101B/KS103/KS103S 的 I²C 地址过程中，严禁突然给 KS101B/KS103/KS103S 断电。修改地址函数请不要放在 while(1)循环中，保证在程序中上电后只运行一次。

在 I²C 地址设置为不同之后，在主机的两根 I²C 总线上可以同时连接 20 个 KS101B/KS103/KS103S。主机在对其中一个 KS101B/KS103/KS103S 模块进行控制时，其他模块自动进入微瓦级功耗休眠模式，因此不必担心电流供应不足问题。

如果要修改多个地址不同的 KS101B，为降低工作量，可使用 0x00 来代替 KS101B 原地址。

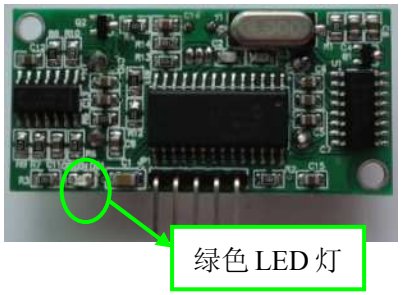
广播地址(0x00)接收(仅 KS101B 支持)：

KS101B 支持广播地址接收，如果不想获知其具体的 I²C 地址，可以使用 0x00 作为地址替代，亦可正常控制本模块。但是使用 0x00 广播地址仅能控制本模块，无法获得模块所探测的数据。要取得相应模块的数据，需要使用相应的地址。

KS101B/KS103/KS103S 工作流程：

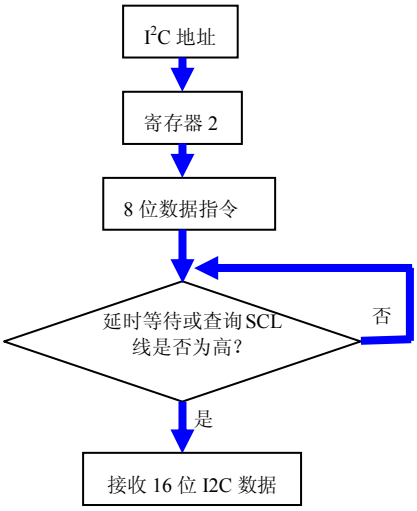
在 KS101B/KS103/KS103S 上电启动时，系统会开始自检，自检正常后其背面的 LED 会以二进制方式闪烁显示其 8 位 I²C 地址, 快闪两下代表“1”，慢闪一下代表“0”。例如显示 0xea 地址，其二进制数为 *0B*11101010，绿色 LED 渐亮→灭→**快闪两下**→灭→**快闪两下**→灭→**快闪两下**→灭→**慢闪一下**→灭→**快闪两下**→灭→**慢闪一下**→灭→**快闪两下**→灭→**慢闪一下**→灭。(3)

Note 3: LED 闪烁时的绿色亮光可能会刺激到眼睛，请尽量不要近距离直视工作中的 LED，可以使用眼睛的余光来观察其闪烁。



KS101B/KS103/KS103S 启动后如果收到主机的有效数据指令，LED 将立即停止闪烁显示。进入指令探测模式。

KS101B/KS103/KS103S 使用 I²C 接口与主机通信，自动响应主机的 I²C 控制指令。指令为 8 位数据，指令发送流程为：



多量程探测

探测指令从 0x01 到 0x2f，数值越大，信号增益越大。指令 0x01 对应量程约 100mm，0x02 对应量程约 200mm，……，依此类推，0x2f 对应量程约 4700mm。量程越小，探测速度越快。其探测时间约为超声波在量程范围内传输的时间的基础上再加约 1ms。注意探测时返回的是 us 值，是一个时间单位，其代表超声波从发出到遇到障碍物反射收回所经历的时间。

探测结束智能识别

KS101B/KS103/KS103S 在发送完探测指令后，需要等待一段时间方可以获取正确的 16 位 I^2C 数据。而用户只知道最大探测时间，但并不确知实际每次的探测时间。KS101B/KS103/KS103S 采用了探测结束智能识别技术。探测过程中 SCL 将一直保持为低电平，用户可以通过查询 SCL 线是否变为高电平即 while(!SCL)语句来等待，SCL 线变为高则表明探测完毕，可以开始通过 I^2C 总线接收到 KS101B/KS103/KS103S 探测到的 16 位数据。注意，发送完探测指令后，需要延时约 40us 以上再查询 SCL 线是否变高，所述 40us 为 KS101B/KS103/KS103S 响应延迟。由于最快的探测指令 0xa0 也需要 1ms 的时间，因此建议延时约 1ms 后再判断 SCL 线，这样做既不会打断正在进行的探测，也不会降低探测效率。也可以通过延时一段时间再开始接收 16 位 I^2C 数据。(4)

Note 4: 这种总线钳制探测方式可以为客户获得更大的探测速度及效率，而不是通过定时器延时或 delay 函数延时每次探测都要至少等待 65ms。换言之，用户大部分时候仅需要快速知晓 1m 范围内是否有障碍物。具体延时时间应大于表 1 所列各指令的最大探测时间。

如果不希望 SCL 线在探测时被拉低，可以通过发送指令 0xc3 指令，之后断电重启 KS101B/KS103/KS103S 后 SCL 线仍然不会拉低。如果想恢复 I^2C 钳制及 SCL 拉低功能，发送 0xc2 指令即可。

配置方法非常简单，向本模块发送指令时序：“I^2C 地址 + 寄存器 2 +0xc2/0xc3”即可，发送完成后请延时至少 2 秒，以让系统自动完成配置。并开始按照新配置工作。

以附件 3 所示程序为例，配置代码如下：

```
write_byte(0xe8,2,0xc2);
delayms(2000);
```

探测结束智能识别功能配置好之后会自动保存，并立即按照新配置工作。KS101B/KS103/KS103S 在重新上电后将按新配置运行。

探测指令

探测指令发送完成后，KS101B/KS103/KS103S 将依据探测指令进入相应探测模式，主机此

时须等待一段时间方可开始通过 I²C 总线查询探测结果，过早查询 I²C 总线将获得 0xff 值。注意，每一帧**探测指令**格式均为：

I²C 地址	寄存器 2	8 位数据

所有 **I²C 控制**指令汇总如下：

寄存器	命令	返回值范围（10 进制）	返回值范围（16 进制）	备注
0		0-255	0-0xff	制造年份
1		1-52	0x01-0x35	第几周生产
2	0x01	80-577μs	0x50-0x241μs	探测量程约为 100mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x02	66-1154μs	0x42-0x482μs	探测量程约为 200mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x03	66-1731μs	0x42-0x6c3μs	探测量程约为 300mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x04	80-2308μs	0x50-0x904μs	探测量程约为 400mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x05	82-2885μs	0x52-0xb05μs	探测量程约为 500mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x06	80-3462μs	0x50-0xd86μs	探测量程约为 600mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x07	68-4039μs	0x44-0xfc7μs	探测量程约为 700mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x08	80-4616μs	0x50-0x1208μs	探测量程约为 800mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x09	80-5193μs	0x50-0x1449μs	探测量程约为 900mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x0a	66-5770μs	0x42-0x168aμs	探测量程约为 1000mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x0b	93-6347μs	0x5d-0x18cbμs	探测量程约为 1100mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x0c	79-6924μs	0x4f-0x1b0cμs	探测量程约为 1200mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x0d	93-7501μs	0x5d-0x1d4dμs	探测量程约为 1300mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x0e	79-8078μs	0x4f-0x1f8eμs	探测量程约为 1400mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x0f	79-8655μs	0x4f-0x21cfμs	探测量程约为 1500mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x10	79-9232μs	0x4f-0x2410μs	探测量程约为 1600mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x11	77-9809μs	0x4d-0x2651μs	探测量程约为 1700mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x12	77-10386μs	0x4d-0x2892μs	探测量程约为 1800mm，返回μs。探测最大耗时＝返回最大值+1000μs

2	0x13	77-10963μs	0x4d-0x2ad3μs	探测量程约为 1900mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x14	79-11540μs	0x4f-0x2d14μs	探测量程约为 2000mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x15	63-12117μs	0x3f-0x2f55μs	探测量程约为 2100mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x16	79-12694μs	0x4f-0x3196μs	探测量程约为 2200mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x17	79-13271μs	0x4f-0x33d7μs	探测量程约为 2300mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x18	79-13848μs	0x4f-0x3618μs	探测量程约为 2400mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x19	77-14425μs	0x4d-0x3859μs	探测量程约为 2500mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x1a	93-15002μs	0x5d-0x3a9aμs	探测量程约为 2600mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x1b	63-15579μs	0x3f-0x3cdbμs	探测量程约为 2700mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x1c	79-16156μs	0x4f-0x3f1cμs	探测量程约为 2800mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x1d	79-16733μs	0x4f-0x415dμs	探测量程约为 2900mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x1e	79-17310μs	0x4f-0x439eμs	探测量程约为 3000mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x1f	77-17887μs	0x4d-0x45dfμs	探测量程约为 3100mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x20	91-18464μs	0x5b-0x4820μs	探测量程约为 3200mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x21	79-19041μs	0x4f-0x4a61μs	探测量程约为 3300mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x22	79-19618μs	0x4f-0x4ca2μs	探测量程约为 3400mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x23	79-20195μs	0x4f-0x4ee3μs	探测量程约为 3500mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x24	79-20772μs	0x4f-0x5124μs	探测量程约为 3600mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x25	77-21349μs	0x4d-0x5365μs	探测量程约为 3700mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x26	79-21926μs	0x4f-0x55a6μs	探测量程约为 3800mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x27	63-22503μs	0x3f-0x57e7μs	探测量程约为 3900mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x28	79-23080μs	0x4f-0x5a28μs	探测量程约为 4000mm，返回μs。探测最大耗时＝返回最大值+1000μs

2	0x29	63-23657μs	0x3f-0x5c69μs	探测量程约为 4100mm,返回μs。探测最大耗时＝返回最大值+1000μs
2	0x2a	79-24234μs	0x4f-0x5eaaμs	探测量程约为 4200mm,返回μs。探测最大耗时＝返回最大值+1000μs
2	0x2b	79-24811μs	0x4f-0x60ebμs	探测量程约为 4300mm,返回μs。探测最大耗时＝返回最大值+1000μs
2	0x2c	79-25388μs	0x4f-0x632cμs	探测量程约为 4400mm,返回μs。探测最大耗时＝返回最大值+1000μs
2	0x2d	77-25965μs	0x4d-0x656dμs	探测量程约为 4500mm,返回μs。探测最大耗时＝返回最大值+1000μs
2	0x2e	79-26542μs	0x4f-0x67aeμs	探测量程约为 4600mm,返回μs。探测最大耗时＝返回最大值+1000μs
2	0x2f	63-27119μs	0x3f-0x69efμs	探测量程约为 4700mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x70	无	无	第一级降噪，出厂默认设置，适用于电池供电
2	0x71	无	无	第二级降噪，适用于 USB 供电
2	0x72	无	无	第三级降噪，适用于较长距离 USB 供电
2	0x73	无	无	第四级降噪，适用于开关电源供电
2	0x74	无	无	第五级降噪，适用于噪音较大的开关电源供电
2	0x75	无	无	第六级降噪，适用于高噪音电源供电
2	0x8a	无	无	I^2C 通讯测试指令,指令发送完成后 LED 将显示相应指令二进制值
2	0x8b	无	无	
2	0x8c	无	无	
2	0x92	无	无	修改地址第二时序
2	0x9a	无	无	修改地址第一时序
2	0x9e	无	无	修改地址第三时序
2	0xa0	0-1023	0-0x3ff	光强探测指令，光线越强，数值越大，探测耗时约 1ms
2	0xb0	10-5200mm	0x0a-0x1450mm	0-5m 范围，普通距离(不带温度补偿)，返回 mm,探测最大耗时约 33ms
2	0xb2	79-30000μs	0x4f-0x7530μs	0-5m 范围，普通距离(不带温度补偿)，返回μs,探测最大耗时约 32ms
2	0xb4	10-5200mm	0x0a-0x1450mm	0-5m 范围，普通距离(**带温度补偿**)，返回 mm,探测最大耗时约 87ms(KS103S 不支持此命令)
2	0xb8	20-11280mm	0x14-0x2c10mm	0-11m 范围，普通距离(不带温度补偿)，返回 mm,探测最大耗时约 68ms
2	0xba	159-65278μs	0x9f-0xfefeμs	0-11m 范围，普通距离(不带温度补偿)，返回μs,探测最大耗时约 66ms
2	0xbc	20-11280mm	0x14-0x2c10mm	0-11m 范围，普通距离(**带温度补偿**)，返回 mm,探测最大耗时约 87ms(KS103S 不支持此命令)
2	0xbd	119-11280mm	0x77-0x2c10mm	12cm-11m 范围，0-10 米量程专用指令。普通距离(不带温度补偿)，返回 mm,探测最大耗时约 68ms
2	0xbe	720-65278μs	0x2d0-0xfefeμs	12cm-11m 范围，0-10 米量程专用指令。普通距离

				(不带温度补偿)，返回μs,探测最大耗时约 66ms
2	0xbf	120-11280mm	0x78-0x2c10mm	12cm-11m 范围，0-10 米量程专用指令。普通距离(**带温度补偿**)，返回 mm,探测最大耗时约 87ms(KS103S 不支持此命令)
2	0xc0	无	无	开 LED 探测显示，默认
2	0xc1	无	无	关 LED 探测显示
2	0xc2	无	无	探测时 I2C 的 SCL 线强制拉低，默认
2	0xc3	无	无	探测时 I2C 的 SCL 线不拉低
2	0xc4	无	无	5 秒休眠等待
2	0xc5	无	无	1 秒休眠等待
2	0xc9	0-255	0-0xff	返回 9 位精度的温度数据,按 DS18B20 格式,范围为-40℃- +125℃,探测耗时约 83ms(KS103S 不支持此命令)
2	0xca	0-255	0-0xff	返回 10 位精度的温度数据,按 DS18B20 格式,范围为-40℃- +125℃,探测耗时约 168ms(KS103S 不支持此命令)
2	0xcb	0-255	0-0xff	返回 11 位精度的温度数据,按 DS18B20 格式,范围为-40℃- +125℃,探测耗时约 315ms(KS103S 不支持此命令)
2	0xcc	0-255	0-0xff	返回 12 位精度的温度数据,按 DS18B20 格式,范围为-40℃- +125℃,探测耗时约 610ms(KS103S 不支持此命令)
3		0-255	0-0xff	寄存器 3 与寄存器 2 联合使用，寄存器 2 返回 16 位数据探测结果的高 8 位，寄存器 3 返回 16 位数据的低 8 位
4		0	0	保留供升级用
5		0	0	保留供升级用
6		0-255	0-0xff	程序版本
7-15		0	0	保留供升级用

表 1

距离探测(KS103S 不支持温度修正指令 0xb0 及 0xbc 及温度探测指令)

从 0x01 到 0x2f 共 47 个多量程探测指令，以及探测范围在 0～5m 的 0xb0/0xb2/0xb4 指令，探测范围在 0～11m 的 0xb8/0xba/0xbc 指令。通过“I^2C 地址 + 寄存器 2 + 距离探测指令”时序，延时或等待上表中所规定的相应时间后，再使用读取函数读寄存器 2 及寄存器 3 的值，即可取得 16 位的距离数据。指令 0xb0 及 0xb8 是按照 25℃标准通过实际探测时间换算而来的距离值；指令 0xb2 及 0xba 探测返回的均是一个时间单位(μs)，其代表超声波从发出到遇到障碍物反射收回所经历的时间。

对于 KS101B 及 KS103,要获得精准的距离探测值，请使用 0xb4 或 0xbc 命令，这两个命令自动使用高精度温度补偿技术，探测值更稳定更精确。也可以使用 ***0xb2/0xba(传输时间) + 0xc9/0xca/0xcb/0xcc(环境温度)*** 组合，探测出超声波在空气中的传输时间及相应环境温度，再通过声速换算出精确的距离值。使用经温度修正的 0xb4 指令，最高精度可达 1mm，误差为 0.152mm/17cm。随着环境与科技的变化与发展，KS101B/KS103 内部使用的声速计算公式可能并不准确。为获得精度达到毫米级别的距离，请通过超声波传输时间及环境温度并使用可能的最

新的声速计算公式来获取精确的距离值。

同时，在远距离探测时，如果电源噪音较大，KS103 将可能不能达到 1cm～800cm 最大量程，因此，如果使用噪音较大的电源（如从电脑 USB 口取电），请使用探测范围在 0～5m 的探测指令。

电源降噪指令

KS101B/KS103/KS103S 默认电源推荐使用电池供电。如果使用噪音较大的电源，测距值可能会出现不稳定的波动。用户可以通过发送 0x70，0x71，0x72，0x73，0x74，0x75 命令来配置 KS101B/KS103/KS103S 测距模块的杂波抑制功能。0x70 指令将使本模块配置为第一级降噪，适用于电池供电的场合，同时也是出厂默认设置。0x71 指令将使本模块配置为第二级降噪，适用于 USB 供电等有一定高频噪音的场合。0x72 指令将使本模块配置为第三级降噪，适用于较长距离 USB 供电的场合。0x73 指令将使本模块配置为第四级降噪，适用于开关电源供电的场合。0x74 指令将使本模块配置为第五级降噪，适用于噪音较大的开关电源供电场合。0x75 指令将使本模块配置为第六级降噪，适用于高噪音电源供电的场合。

应近可能选择比较小的值例如 0x70 以确保测量精度。降噪级别越高，有用波形被消除的概率越大，如此很可能会降低本模块的探测精度及探测量程。

同时，配置越高的降噪级别，本模块的波束角将越小，波束直线度越好，抗干扰性更高，但灵敏度将有所下降。

配置方法非常简单，向本模块发送指令时序：“I^2C 地址 + 寄存器 2 +0x70/0x71/0x72/0x73/0x74/0x75”即可，发送完成后请延时至少 2 秒，以让系统自动完成配置。并开始按照新配置工作。

以附件 3 所示程序为例，将本模块配置为二级降噪，配置代码如下：

```
write_byte(0xe8,2,0x71);
delayms(2000);
```

配置代码请放在程序的初始化函数中，即 while(1)循环之前，以保护模块。KS101B/KS103/KS103S 收到有效配置指令之后，LED 灯将长亮 5s，表明配置成功。

KS101B/KS103/KS103S 在重新上电后将按新配置运行。

温度探测（仅 KS101B/KS103 支持）

温度探测包括 0xc9，0xca，0xcb，0xcc 共 4 个探测指令，通过“I^2C 地址 + 寄存器 2 + 0xc9/0xca/0xcb/0xcc”时序，延时或等待上表中所规定的相应时间后，再使用读取函数读寄存器 2 及寄存器 3 的值，所取得的 16 位数据遵从 DS18B20 芯片的温度读数规则，具体请参阅 DS18B20 的芯片资料。以 0xcc 指令为例，其将获取共 16 位的探测数据。16 位数据中的前面 5 位是符号位，如果测得的温度大于 0，这 5 位为 0，只要将 16 位数据除以 16 或乘以 0.0625 即可获得精确到 0.0625 摄氏度的环境温度值。如果温度小于 0，这 5 位为 1，只需要将测到的 16 位数据按位取反然后加 1 再乘以 0.0625 即可得到实际负温度值。例如返回的 16 数据为 0xfe6a 时，0xfe6a 换成二进制是 0B1111 1110 0110 1010，最高位共 5 个 1，因此是负温度，按位取反后二进制值为 0B0000 0001 1001 0101，相应 10 进制值为 405，加 1 后为 406，406 乘以 0.0625 等于 25.375，则环境温度为-25.375℃。如果返回的 16 位数据为 0x1c6，其二进制值为 0B0000 0001 1100 0110，高 5 位为 0，因此直接乘以 0.0625 即 454 乘以 0.0625 等于 28.375℃。

光强探测

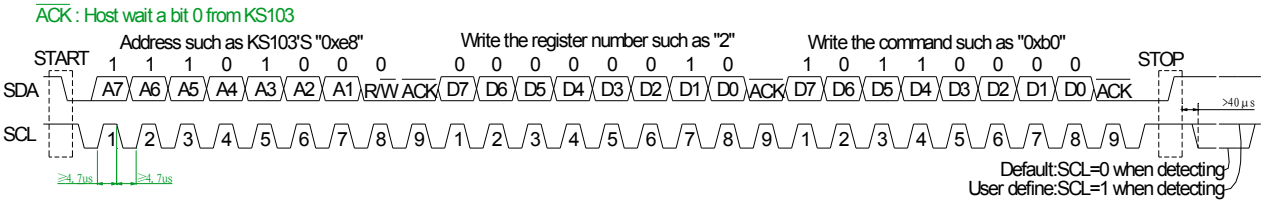
使用 0xa0 指令，通过“I^2C 地址 + 寄存器 2 + 0xa0”时序，延时或等待 1ms 后然后再使用读取函数读寄存器 2 及寄存器 3 的值，即可快速获得环境光的强度。光线越强时，返回数值越

大，返回值在 0～1023 之间。

时序图

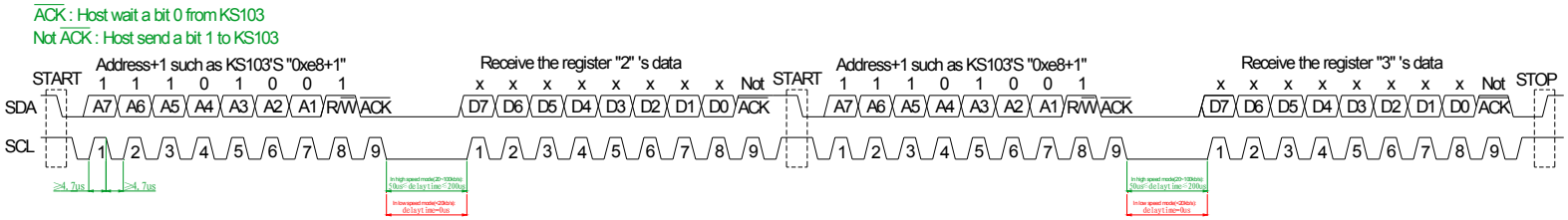
时序图 1：发送探测指令，指令格式为(Such as register 2)：

I²C 地址	寄存器 2	8 位数据指令



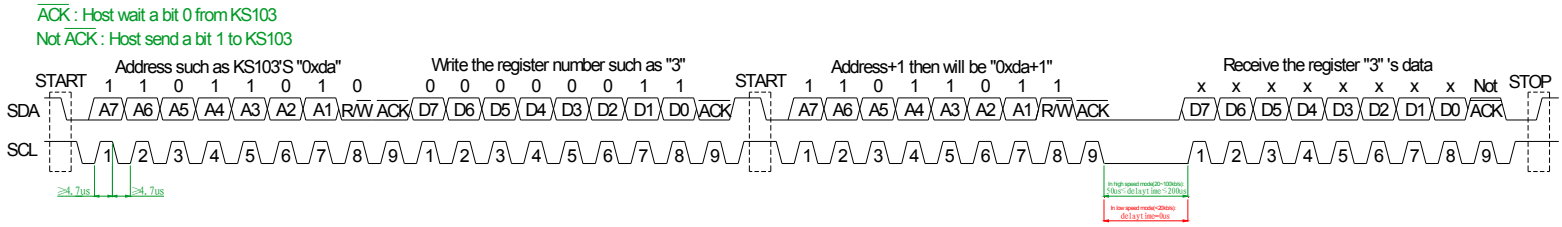
时序图 2：执行完时序图 1 后，等待 SCL 变高或延时 100ms 后接收 16 位数据，先高位后低位，指令格式为：

I²C 地址+1	读寄存器 2	I²C 地址+1	读寄存器 3



时序图 3：执行完时序图 1 后，等待 SCL 变高或延时 100ms 后接收寄存器 x 的数据(本例为寄存器 3)，读任意寄存器指令格式(Such as register 3)：(5)

I²C 地址	寄存器 3	I²C 地址+1	读寄存器 3



Note 5: 采用读任意寄存器指令时，如果读寄存器 2 及寄存器 3，必须先发送针对寄存器 2 的探测指令。注意，所有探测指令都储存在寄存器 2 中。例程中采用了先 *发送探测指令* 再 *读任意寄存器指令时序*（读寄存器 2 + 读寄存器 3）。向 KS103 写入“I²C 地址+1”后，在 20~100kb/s 的 I²C 通信速率时，不能立即去接收 8bit 的数据，要等待 ACK 低电平的有效回应，或再延时至少 50us(delaytime)，才可以接收到寄存器的数据。在写“I²C 地址+1”与“读寄存器 2/3”之间加一个至少 50us 延时(delaytime)的话，I²C 通信速率可以调大仍可以与 KS103 可靠通信。小于 20kb/s 的 I²C 通信速率时，可以不用前面所述至少 50us(delaytime)的延时。另外，小于 10cm 的距离探测，相隔时间建议大于 1ms，否则可能存在上次的超声波被下一次探测所接收到的问题。总之，**确保成功建立 I²C 通信的关键有两点**：第一，高低电平延时均应不小于 4.7us；第二，KS103 收到主机的有效探测数据绿色 LED 快闪但返回值不正确时，主机需要加上 delaytime 不小于 50us 的延时，即可获取正确数据。请遵从时序图 1~3 之规定。

进一步的省电措施

如果用户希望将省电进行到底，可以发送 0xc1 关 LED 探测显示，以降低电流消耗。发送 0xc0 可以恢复 LED 探测显示。

配置方法非常简单，向本模块发送指令时序：“I²C 地址 + 寄存器 2 +0xc0/0xc1”即可，

发送完成后请延时至少 2 秒，以让系统自动完成配置。并开始按照新配置工作。

以附件 3 所示程序为例，配置代码如下：

write_byte(0xe8,2,0xc0);

delayms(2000);

LED 探测显示配置好之后会自动保存，并立即按照新配置工作。KS101B/KS103/KS103S 在重新上电后将按新配置运行。

休眠等待时间设置

休眠模式默认为 5s 等待，5s 内未收到探测指令则自动进入休眠模式。另有 1s 模式可供用户选择。通过 I^2C 总线发送数据指令 0xc5 进入 1s 休眠模式；发送 0xc4 可以恢复 5s 休眠模式。

配置方法非常简单，向本模块发送指令时序：“I^2C 地址 + 寄存器 2 +0xc4/0xc5”即可，发送完成后请延时至少 2 秒，以让系统自动完成配置。并开始按照新配置工作。

以附件 3 所示程序为例，配置代码如下：

write_byte(0xe8,2,0xc4);

delayms(2000);

休眠等待时间设置好之后 KS101B/KS103/KS103S 会自动保存，并立即按照新配置工作。KS101B/KS103/KS103S 在重新上电后将按新配置运行。

TTL 串口模式

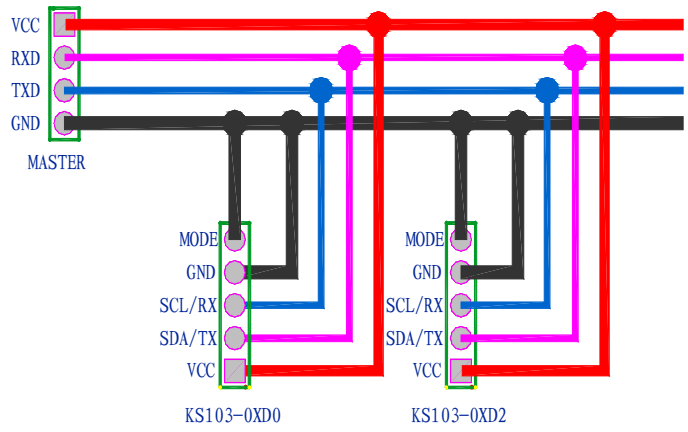
KS101B/KS103/KS103S 的 TTL 串口模式波特率为 9600bps，1 启动位，8 数据位，1 停止位，无校验位，TTL 电平。

KS101B/KS103/KS103S 连线：

在 KS101B/KS103/KS103S 上连线引脚上标识有：VCC、SDA/TX、SCL/RX、GND 及 MODE。本模块在上电之前，MODE 需要接 0V 地，上电后模块将工作于 TTL 串口模式。如果 KS101B/KS103/KS103S 在上电后再将 MODE 引脚接 0V 地，模块将仍然工作于 I^2C 模式。因此，TTL 串口模式时需要 5 根线来控制，其中 VCC 用于连接+5V(3.0～5.5V 范围均可)电源(1)，GND 用于连接电源地，SDA/TX 连接 MCU 或 USB 转 TTL 模块的 RXD，SCL/RX 引脚连接 MCU 或 USB 转 TTL 模块的 TXD。

Note 1: 要达到最佳的工作状态推荐使用+5V 电源，低于 5V 的电压将影响测距量程。并且，严禁将 VCC 与 GND 接反，否则可能会损坏电路。超过 3 秒钟的电路反接将可能导致不可恢复的损坏。

具体连线如下图所示（**最多接 2 个**）：



KS101B/KS103/KS103S 默认串口地址为 0xe8，用户可以将地址修改为 20 种地址中的任何一个：0xd0, 0xd2, 0xd4, 0xd6, 0xd8, 0xda, 0xdc, 0xde, 0xe0, 0xe2, 0xe4, 0xe6, 0xe8, 0xea, 0xec, 0xee, 0xf8,

0xfa, 0xfc, 0xfe. (2)

Note 2: 请注意，以上地址不包括 0xf0, 0xf2, 0xf4,0xf6，其与 I²C 版地址完全一致。此外，串口协议规定一对一，因此建议使用串口模式时，串口总线上最好只备有 1 台 KS101B/KS103/KS103S，**最多请不要超过 2 台**。

修改串口地址时序：

地址	2	**0x9a**	延时 1ms	**地址**	2	**0x92**	延时 1ms	**地址**	2	**0x9e**	延时 1ms	**地址**	2	**新地址**	延时 100ms

修改串口地址须严格按照时序来进行，时序中的延时时间为最小时间。

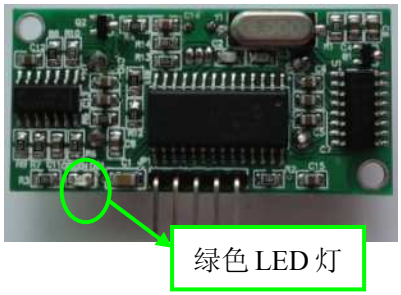
修改完毕后 LED 灯将长亮，给 KS101B/KS103/KS103S 重新上电，可观察到 LED 显示新地址。在修改 KS101B/KS103/KS103S 的串口地址过程中，严禁突然给 KS101B/KS103/KS103S 断电。修改地址函数请不要放在 while(1)循环中，保证在程序中上电后只运行一次。

在串口地址设置为不同之后，在主机的两根串口线上可以同时连接 20 个 KS101B/KS103/KS103S。主机在对其中一个 KS101B/KS103/KS103S 模块进行控制时，其他模块不会受到影响。

KS101B/KS103/KS103S 工作流程：

在 KS101B/KS103/KS103S 上电启动时，系统会开始自检，自检正常后其背面的 LED 会以二进制方式闪烁显示其 8 位串口地址, 快闪两下代表“1”，慢闪一下代表“0”。例如显示 0xea 地址，其二进制数为 *0B*11101010，绿色 LED 渐亮→灭→**快闪两下**→灭→**快闪两下**→灭→**快闪两下**→灭→**慢闪一下**→灭→**快闪两下**→灭→**慢闪一下**→灭→**快闪两下**→灭→**慢闪一下**→灭。(3)

Note 3: LED 闪烁时的绿色亮光可能会刺激到眼睛，请尽量不要近距离直视工作中的 LED，可以使用眼睛的余光来观察其闪烁。



KS101B/KS103/KS103S 启动后如果收到主机的有效数据指令，LED 将立即停止闪烁显示。进入指令探测模式。每探测一次 LED 灯会闪烁一次。

KS101B/KS103/KS103S 使用 TTL 串口接口与主机通信时，自动响应主机的控制指令。指令为 8 位数据，指令发送流程为：

串口地址(0xe8) —>延时 20～100us—> 寄存器(0x02) —>延时 20～100us—> 探测指令(0xbc) —> 通过串口接收 KS103 的探测数据高 8 位 —> 接收 KS103 的探测数据低 8 位

KS101B/KS103/KS103S 工作于串口模式时，只能写寄存器 0x02，写其他值将不响应。单片机接收 KS101B/KS103/KS103S 的探测结果时，可启用串口中断来接收 16 位探测结果，探测结果将先发高 8 位，再发低 8 位。接收到返回的 16 位探测结果之后才可以再发探测指令进行下一轮探测，否则串口将返回不正确值。

多量程探测

探测指令从 0x01 到 0x2f，数值越大，信号增益越大。指令 0x01 对应量程约 100mm，0x02 对应量程约 200mm，……，依此类推，0x2f 对应量程约 4700mm。量程越小，探测速度越快。其探测时间约为超声波在量程范围内传输的时间的基础上再加约 1ms。注意探测时返回的是 us 值，是一个时间单位，其代表超声波从发出到遇到障碍物反射收回所经历的时间。

探测结束智能识别

由于探测指令发出后 KS101B/KS103/KS103S 会自动通过串口返回 16 位探测结果，因此串口模式无此功能。

探测指令

探测指令发送完成后，KS101B/KS103/KS103S 将依据探测指令进入相应探测模式，主机此时开启串口中断，未接收到返回的探测结果不能又重新发探测指令。注意，每一帧**探测指令**格式均为：

TTL 串口地址	寄存器 2	8 位数据

所有串口控制指令汇总如下：

寄存器	命令	返回值范围（10 进制）	返回值范围（16 进制）	备注
2	0x01	80-577μs	0x50-0x241μs	探测量程约为 100mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x02	66-1154μs	0x42-0x482μs	探测量程约为 200mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x03	66-1731μs	0x42-0x6c3μs	探测量程约为 300mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x04	80-2308μs	0x50-0x904μs	探测量程约为 400mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x05	82-2885μs	0x52-0xb05μs	探测量程约为 500mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x06	80-3462μs	0x50-0xd86μs	探测量程约为 600mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x07	68-4039μs	0x44-0xfc7μs	探测量程约为 700mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x08	80-4616μs	0x50-0x1208μs	探测量程约为 800mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x09	80-5193μs	0x50-0x1449μs	探测量程约为 900mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x0a	66-5770μs	0x42-0x168aμs	探测量程约为 1000mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x0b	93-6347μs	0x5d-0x18cbμs	探测量程约为 1100mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x0c	79-6924μs	0x4f-0x1b0cμs	探测量程约为 1200mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x0d	93-7501μs	0x5d-0x1d4dμs	探测量程约为 1300mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x0e	79-8078μs	0x4f-0x1f8eμs	探测量程约为 1400mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x0f	79-8655μs	0x4f-0x21cfμs	探测量程约为 1500mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x10	79-9232μs	0x4f-0x2410μs	探测量程约为 1600mm，返回μs。探测最大耗时＝返回最大值+1000μs

2	0x11	77-9809μs	0x4d-0x2651μs	探测量程约为 1700mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x12	77-10386μs	0x4d-0x2892μs	探测量程约为 1800mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x13	77-10963μs	0x4d-0x2ad3μs	探测量程约为 1900mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x14	79-11540μs	0x4f-0x2d14μs	探测量程约为 2000mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x15	63-12117μs	0x3f-0x2f55μs	探测量程约为 2100mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x16	79-12694μs	0x4f-0x3196μs	探测量程约为 2200mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x17	79-13271μs	0x4f-0x33d7μs	探测量程约为 2300mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x18	79-13848μs	0x4f-0x3618μs	探测量程约为 2400mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x19	77-14425μs	0x4d-0x3859μs	探测量程约为 2500mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x1a	93-15002μs	0x5d-0x3a9aμs	探测量程约为 2600mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x1b	63-15579μs	0x3f-0x3cdbμs	探测量程约为 2700mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x1c	79-16156μs	0x4f-0x3f1cμs	探测量程约为 2800mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x1d	79-16733μs	0x4f-0x415dμs	探测量程约为 2900mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x1e	79-17310μs	0x4f-0x439eμs	探测量程约为 3000mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x1f	77-17887μs	0x4d-0x45dfμs	探测量程约为 3100mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x20	91-18464μs	0x5b-0x4820μs	探测量程约为 3200mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x21	79-19041μs	0x4f-0x4a61μs	探测量程约为 3300mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x22	79-19618μs	0x4f-0x4ca2μs	探测量程约为 3400mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x23	79-20195μs	0x4f-0x4ee3μs	探测量程约为 3500mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x24	79-20772μs	0x4f-0x5124μs	探测量程约为 3600mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x25	77-21349μs	0x4d-0x5365μs	探测量程约为 3700mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x26	79-21926μs	0x4f-0x55a6μs	探测量程约为 3800mm，返回μs。探测最大耗时＝返回最大值+1000μs

2	0x27	63-22503μs	0x3f-0x57e7μs	探测量程约为 3900mm,返回μs。探测最大耗时＝返回最大值+1000μs
2	0x28	79-23080μs	0x4f-0x5a28μs	探测量程约为 4000mm,返回μs。探测最大耗时＝返回最大值+1000μs
2	0x29	63-23657μs	0x3f-0x5c69μs	探测量程约为 4100mm,返回μs。探测最大耗时＝返回最大值+1000μs
2	0x2a	79-24234μs	0x4f-0x5eaaμs	探测量程约为 4200mm,返回μs。探测最大耗时＝返回最大值+1000μs
2	0x2b	79-24811μs	0x4f-0x60ebμs	探测量程约为 4300mm,返回μs。探测最大耗时＝返回最大值+1000μs
2	0x2c	79-25388μs	0x4f-0x632cμs	探测量程约为 4400mm,返回μs。探测最大耗时＝返回最大值+1000μs
2	0x2d	77-25965μs	0x4d-0x656dμs	探测量程约为 4500mm,返回μs。探测最大耗时＝返回最大值+1000μs
2	0x2e	79-26542μs	0x4f-0x67aeμs	探测量程约为 4600mm,返回μs。探测最大耗时＝返回最大值+1000μs
2	0x2f	63-27119μs	0x3f-0x69efμs	探测量程约为 4700mm，返回μs。探测最大耗时＝返回最大值+1000μs
2	0x70			第一级降噪，出厂默认设置，适用于电池供电
2	0x71			第二级降噪，适用于 USB 供电
2	0x72			第三级降噪，适用于较长距离 USB 供电
2	0x73			第四级降噪，适用于开关电源供电
2	0x74			第五级降噪，适用于噪音较大的开关电源供电
2	0x75			第六级降噪，适用于高噪音电源供电
2	0x8a			串口通讯测试指令,指令发送完成后 LED 将显示相应指令二进制值
2	0x8b			
2	0x8c			
2	0x92			修改地址第二时序
2	0x9a			修改地址第一时序
2	0x9e			修改地址第三时序
2	0xa0	0-1023	0-0x3ff	光强探测指令，光线越强，数值越大，探测耗时约 1ms
2	0xb0	10-5200mm	0x0a-0x1450mm	0-5m 范围，普通距离(不带温度补偿)，返回 mm,探测最大耗时约 33ms
2	0xb2	79-30000μs	0x4f-0x7530μs	0-5m 范围，普通距离(不带温度补偿)，返回μs,探测最大耗时约 32ms
2	0xb4	10-5200mm	0x0a-0x1450mm	0-5m 范围，普通距离(**带温度补偿**)，返回 mm,探测最大耗时约 87ms(KS103S 不支持此命令)
2	0xb8	20-11280mm	0x14-0x2c10mm	0-11m 范围，普通距离(不带温度补偿)，返回 mm,探测最大耗时约 68ms
2	0xba	159-65278μs	0x9f-0xfefeμs	0-11m 范围，普通距离(不带温度补偿)，返回μs,探测最大耗时约 66ms
2	0xbc	20-11280mm	0x14-0x2c10mm	0-11m 范围，普通距离(**带温度补偿**)，返回 mm,探

				测最大耗时约 87ms(KS103S 不支持此命令)
2	0xbd	119-11280mm	0x77-0x2c10mm	12cm-11m 范围，0-10 米量程专用指令。普通距离(不带温度补偿)，返回 mm,探测最大耗时约 68ms
2	0xbe	720-65278μs	0x2d0-0xfefeμs	12cm-11m 范围，0-10 米量程专用指令。普通距离(不带温度补偿)，返回μs,探测最大耗时约 66ms
2	0xbf	120-11280mm	0x78-0x2c10mm	12cm-11m 范围，0-10 米量程专用指令。普通距离(**带温度补偿**)，返回 mm,探测最大耗时约 87ms(KS103S 不支持此命令)
2	0xc0			开 LED 探测显示，默认
2	0xc1			关 LED 探测显示
2	0xc9	0-255	0-0xff	返回 9 位精度的温度数据,按 DS18B20 格式,范围为-40℃- +125℃,探测耗时约 83ms(KS103S 不支持此命令)
2	0xca	0-255	0-0xff	返回 10 位精度的温度数据,按 DS18B20 格式,范围为-40℃- +125℃,探测耗时约 168ms(KS103S 不支持此命令)
2	0xcb	0-255	0-0xff	返回 11 位精度的温度数据,按 DS18B20 格式,范围为-40℃- +125℃,探测耗时约 315ms(KS103S 不支持此命令)
2	0xcc	0-255	0-0xff	返回 12 位精度的温度数据,按 DS18B20 格式,范围为-40℃- +125℃,探测耗时约 610ms(KS103S 不支持此命令)

表 2

距离探测(KS103S 不支持温度修正指令 0xb0 及 0xbc 及温度探测指令)

从 0x01 到 0x2f 共 47 个多量程探测指令，以及探测范围在 0～5m 的 0xb0/0xb2/0xb4 指令，探测范围在 0～11m 的 0xb8/0xba/0xbc 指令。通过“TTL 串口地址 + 寄存器 2 + 距离探测指令”时序，再通过串口收取探测数据的高 8 位及低 8 位的值，即可取得 16 位的距离数据。指令 0xb0 及 0xb8 是按照 25℃标准通过实际探测时间换算而来的距离值;指令 0xb2 及 0xba 探测返回的均是一个时间单位(μs)，其代表超声波从发出到遇到障碍物反射收回所经历的时间。

对于 KS101B 及 KS103,要获得精准的距离探测值，请使用 0xb4 或 0xbc 命令，这两个命令自动使用高精度温度补偿技术，探测值更稳定更精确。也可以使用 ***0xb2/0xba(传输时间)*** + ***0xc9/0xca/0xcb/0xcc(环境温度)*** 组合，探测出超声波在空气中的传输时间及相应环境温度，再通过声速换算出精确的距离值。使用经温度修正的 0xb4 指令，最高精度可达 1mm，误差为 0.152mm/17cm。随着环境与科技的变化与发展，KS101B/KS103 内部使用的声速计算公式可能并不准确。为获得精度达到毫米级别的距离，请通过超声波传输时间及环境温度并使用可能的最新的声速计算公式来获取精确的距离值。

同时，在远距离探测时，如果电源噪音较大，KS103 将可能不能达到 1cm～800cm 最大量程，因此，如果使用噪音较大的电源（如从电脑 USB 口取电），请使用探测范围在 0～5m 的探测指令。

电源降噪指令

KS101B/KS103/KS103S 默认电源推荐使用电池供电。如果使用噪音较大的电源，测距值可能会出现不稳定的波动。用户可以通过发送 0x70，0x71，0x72，0x73，0x74，0x75 命令来配置

KS101B/KS103/KS103S 测距模块的杂波抑制功能。0x70 指令将使本模块配置为第一级降噪，适用于电池供电的场合，同时也是出厂默认设置。0x71 指令将使本模块配置为第二级降噪，适用于 USB 供电等有一定高频噪音的场合。0x72 指令将使本模块配置为第三级降噪，适用于较长距离 USB 供电的场合。0x73 指令将使本模块配置为第四级降噪，适用于开关电源供电的场合。0x74 指令将使本模块配置为第五级降噪，适用于噪音较大的开关电源供电场合。0x75 指令将使本模块配置为第六级降噪，适用于高噪音电源供电的场合。

应近可能选择比较小的值例如 0x70 以确保测量精度。降噪级别越高，有用波形被消除的概率越大，如此很可能会降低本模块的探测精度及探测量程。

同时，配置越高的降噪级别，本模块的波束角将越小，波束直线度越好，抗干扰性更高，但灵敏度将有所下降。

配置方法非常简单，向本模块发送指令时序：“TTL 串口地址 + 寄存器 2 +0x70/0x71/0x72/0x73/0x74/0x75”即可，发送完成后请延时至少 2 秒，以让系统自动完成配置。并开始按照新配置工作。

配置代码请放在程序的初始化函数中，即 while(1)循环之前，以保护模块。KS101B/KS103/KS103S 收到有效配置指令之后，LED 灯将长亮，表明配置成功。

KS101B/KS103/KS103S 在重新上电后将按新配置运行。

温度探测(仅 KS101B/KS103 支持)

温度探测包括 0xc9，0xca，0xcb，0xcc 共 4 个探测指令，通过“TTL 串口地址 + 寄存器 2 + 0xc9/0xca/0xcb/0xcc”时序，延时或等待上表中所规定的相应时间后，再使用读取函数读寄存器 2 及寄存器 3 的值，所取得的 16 位数据遵从 DS18B20 芯片的温度读数规则，具体请参阅 DS18B20 的芯片资料。以 0xcc 指令为例，其将获取共 16 位的探测数据。16 位数据中的前面 5 位是符号位，如果测得的温度大于 0，这 5 位为 0，只要将 16 位数据除以 16 或乘以 0.0625 即可获得精确到 0.0625 摄氏度的环境温度值。如果温度小于 0，这 5 位为 1，只需要将测到的 16 位数据按位取反然后加 1 再乘以 0.0625 即可得到实际负温度值。例如返回的 16 数据为 0xfe6a 时，0xfe6a 换成二进制是 0B1111 1110 0110 1010，最高位共 5 个 1，因此是负温度，按位取反后二进制值为 0B0000 0001 1001 0101，相应 10 进制值为 405，加 1 后为 406，406 乘以 0.0625 等于 25.375，则环境温度为-25.375℃。如果返回的 16 位数据为 0x1c6，其二进制值为 0B0000 0001 1100 0110，高 5 位为 0，因此直接乘以 0.0625 即 454 乘以 0.0625 等于 28.375℃。

光强探测

使用 0xa0 指令，通过“TTL 串口地址 + 寄存器 2 + 0xa0”时序，延时或等待 1ms 后然后再使用读取函数读寄存器 2 及寄存器 3 的值，即可快速获得环境光的强度。光线越强时，返回数值越大，返回值在 0～1023 之间。

时序图

发送探测指令，指令格式为(Only register 2)：

TTL 串口地址	延时 20～100us	寄存器 2	延时 20～100us	8 位数据指令

接收数据建议采用串口中断，这样单片机可以抽出时间做其他的事情。单片机采用模拟串口时请根据串口协议判断 SDA/TX 引脚的电平变化来接收数据，数据依次为：

探测结果高 8 位	探测结果低 8 位

接收完数据后方可以进行下一轮探测指令(例如：0xe8+0x02+0xbc)的发送。

进一步的省电措施

如果用户希望将省电进行到底，可以发送 0xc1 关 LED 探测显示，以降低电流消耗。发送

0xc0 可以恢复 LED 探测显示。

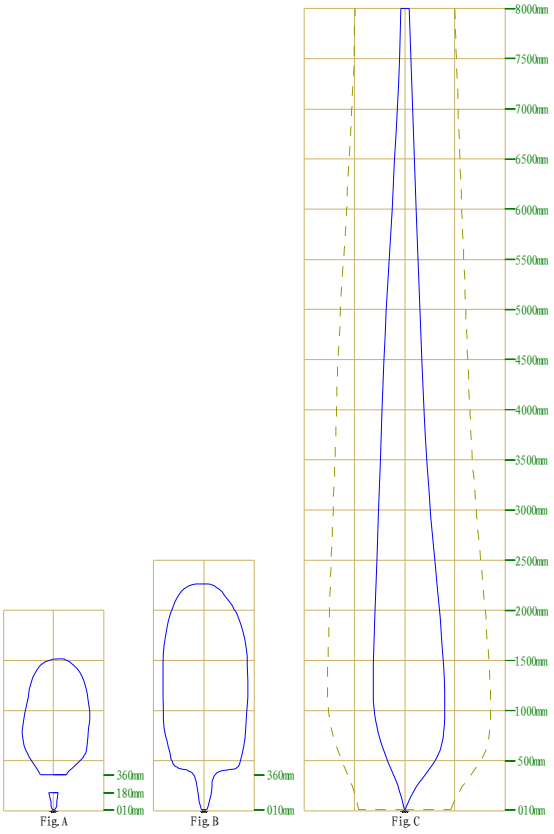
配置方法非常简单，向本模块发送指令时序：“TTL 串口地址 + 寄存器 2 +0xc0/0xc1”即可，发送完成后请延时至少 2 秒，以让系统自动完成配置。并开始按照新配置工作。

LED 探测显示配置好之后会自动保存，并立即按照新配置工作。KS101B/KS103/KS103S 在重新上电后将按新配置运行。

休眠等待时间设置

串口模式不进入休眠。

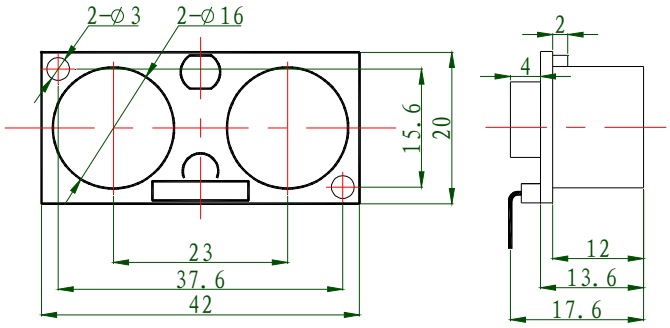
探测范围(40KHz 超声波)



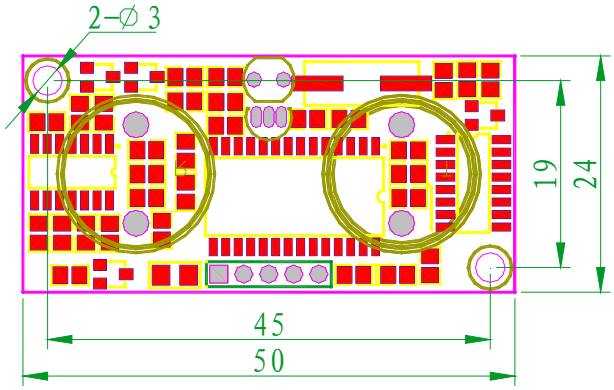
图号　条件	反射物大小	材料	探头离地高度	电压	降噪级别
Fig.A	直径 6mm	木材	62mm	5V	0x71
Fig.B	直径 15.6mm	304 不锈钢	62mm	5V	0x71
Fig.C	长 920mm 宽 860mm	2 坑瓦楞纸板	160mm	5V	0x71

备注：发射探头在右，接收探头在左，两探头与地面平行；电压 5V 与 3.3V 的测试效果无明显区别。Fig.C 中蓝色实线为纸箱边缘的极限探测范围；虚线对应为纸箱中心线极限位置。

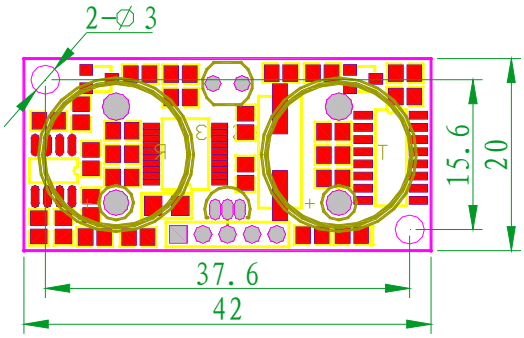
KS103/KS103S 安装尺寸(单位：毫米)：



KS101B PCB:



KS103/KS103S PCB:



可以使用 M3 螺钉及六角螺柱或白色间隔柱来固定。

包装

1) KS101B 产品尺寸：50mm×24mm×17mm;
2) KS103/KS103S 产品尺寸：42mm×20mm×17mm;
2) 产品净重：KS101B:11g; KS103/KS103S:9g.
3) 包装方式：KS101B:1PCS/盒，盒型为 1.5~2mm 厚瓦楞纸材硬纸箱。同时每件产品使用防静电袋包装，内装一包透明硅干燥剂(不可食用)
4) KS101B 包装尺寸： 85mm×80mm×32mm(1PCS/盒)
5) KS101B 包装后的毛重：75g 。

因产品改进需要，可能会对本资料进行修改，客户不能及时获得修改通知时，请在本公司网站 www.dauxi.com 获取最新产品资料。

附件：

1) PIC16F877A 主机采用硬件 I²C 通讯与 KS101B/KS103/KS103S 连接控制 C 代码
2) PIC16F877A 主机采用模拟 I²C 通讯与 KS101B/KS103/KS103S 连接控制 C 代码
3) 51 单片机主机模拟 I²C 通讯与 KS101B/KS103/KS103S 连接控制 C 代码
4) STM32 CORTEX-3 ARM 主机模拟 I²C 通讯与 KS101B/KS103/KS103S 连接控制 C 代码
5) KS10X 主要功能的演示视频(请复制到浏览器网页打开即可观看)：
http://v.youku.com/v_show/id_XMjYwMjUwNTg4.html
20 台 KS101B 共 I²C 总线工作演示视频(请复制到浏览器网页打开即可观看)：
http://v.youku.com/v_show/id_XMjYxMzMxNDE2.html

1) PIC16F877A 主机采用硬件 I²C 通讯与 KS101B/KS103/KS103S 连接控制 C 代码

/*电路连接方式：PIC16F877A 的 IO 口 SCL、SDA 与 KS101B/KS103/KS103S 的 SCL、SDA 连接，PIC16F877A 的 SCL、SDA 线均需个上拉一个 4.7K 的电阻到电源正极 VCC。*/

```
#include <pic.h>                          //4MHz 晶振
__CONFIG(0x3d76);                         //开看门狗
#define DELAY() delay(10)
#difine SCL RC3                  //  此引脚须上拉 4.7K 电阻至 VCC
#difine SDA RC4                  //  此引脚须上拉 4.7K 电阻至 VCC
void setup(void);
unsigned int detect_KS101B(unsigned char ADDRESS, unsigned char command);
void delay(unsigned int ms);
void change_address(unsigned addr_old,unsigned char addr_new);
void send_command(unsigned char cmd);
void display(unsigned int distance,unsigned int delay);     //显示函数请根据主机的实际接线编写
unsigned int distance;
void main(void)
{
    setup();
    //change_address(0xe8,0xe0);   //将默认地址 0xe8 改为 0xe0
    while(1)
    {
        CLRWDT();
        distance = detect_KS101B(0xe8,0xb0);  //Address:0xe8; command:0xb0.
                                              //Get detect result from KS101B/KS103, 16 bit data.
        display(distance,100);                //display function,you should apply it to the master
        delayms(200);
    }
}
void display(unsigned int distance,unsigned int delay);     //显示函数请根据主机的实际接线编写
{
    CLRWDT();
}
void change_address(unsigned addr_old,unsigned char addr_new)
{
    SEN = 1;                                          // send start bit to KS101B/KS103/KS103S
    while(SEN);                                       // wait for it to clear
    while(!SSPIF);                                    // wait for interrupt
    SSPIF = 0;                                        // then clear it.

    SSPBUF = addr_old;                          // KS101B/KS103/KS103S's I2C address
    while(!SSPIF);                              // wait for interrupt
    SSPIF = 0;                                  // then clear it.

    SSPBUF = 2;                                 // write the register number
    while(!SSPIF);                              // wait for interrupt
    SSPIF = 0;                                  // then clear it.

    SSPBUF = 0x9a;                        //command=0x9a, change I2C address, first sequence
```

```
        while(!SSPIF);
        SSPIF = 0;

        PEN = 1;                                        // send stop bit
        while(PEN);
DELAY();                                        // let KS101B/KS103/KS103S to break to do something

        SEN = 1;                                        // send start bit
        while(SEN);                                     // and wait for it to clear
        while(!SSPIF);
        SSPIF = 0;

        SSPBUF = addr_old;                              // KS101B/KS103/KS103S's I2C address
        while(!SSPIF);                                  // wait for interrupt
        SSPIF = 0;                                      // then clear it.

        SSPBUF = 2;                                     // address of register to write to
        while(!SSPIF);                                  //
        SSPIF = 0;

        SSPBUF = 0x92;                          //command=0x92, change I2C address, second sequence
        while(!SSPIF);                                  //
        SSPIF = 0;

        PEN = 1;                                        // send stop bit
        while(PEN);                                     //
DELAY();                                        // let KS101B/KS103/KS103S to break to do something
        SEN = 1;                                        // send start bit
        while(SEN);                                     // and wait for it to clear
        while(!SSPIF);
        SSPIF = 0;

        SSPBUF = addr_old;                              // KS101B/KS103/KS103S's I2C address
        while(!SSPIF);                                  // wait for interrupt
        SSPIF = 0;                                      // then clear it.

        SSPBUF = 2;                                     // address of register to write to
        while(!SSPIF);                                  //
        SSPIF = 0;

        SSPBUF = 0x9e;                          //command=0x9e,, change I2C address,third sequence
        while(!SSPIF);                                  // wait for interrupt
        SSPIF = 0;                                      // then clear it.

        PEN = 1;                                        // send stop bit
        while(PEN);                                     //
DELAY();                                        // let KS101B/KS103/KS103S to break to do something
        SEN = 1;                                        // send start bit
        while(SEN);                                     // and wait for it to clear
        while(!SSPIF);
        SSPIF = 0;

        SSPBUF = addr_old;                              // KS101B/KS103/KS103S's I2C address
        while(!SSPIF);                                  // wait for interrupt
        SSPIF = 0;                                      // then clear it.

        SSPBUF = 2;                                     // address of register to write to
        while(!SSPIF);                                  //
        SSPIF = 0;

        SSPBUF = addr_new;              //new address, it will be 0xd0~0xfe(without 0xf0,0xf2,0xf4,0xf6)
        while(!SSPIF);                                  //
        SSPIF = 0;
```

```
	PEN = 1;                                        // send stop bit
	while(PEN);                                     //
DELAY();                                    // let KS101B/KS103/KS103S to break to do something
}

unsigned int detect_KS101B(unsigned char ADDRESS, unsigned char command)
{
//ADDRESS will be KS101B/KS103/KS103S's address such as 0xb0, command will be the detect command such as 0xb0
unsigned int range=0;
	SEN = 1;                                        // send start bit
	while(SEN);                                     // and wait for it to clear
	while(!SSPIF);
	SSPIF = 0;
	SSPBUF = ADDRESS;                           // KS101B/KS103/KS103S's I2C address
	while(!SSPIF);                              // wait for interrupt
	SSPIF = 0;                                  // then clear it.
	SSPBUF = 2;                                     // address of register to write to
	while(!SSPIF);                              //
	SSPIF = 0;
	SSPBUF = command;
	while(!SSPIF);                              //
	SSPIF = 0;
	PEN = 1;                                        // send stop bit
	while(PEN);                                     //

	TMR1H = 0;                                  // delay while the KS101B/KS103/KS103S is ranging
	TMR1L = 0;
	T1CON = 0x31;                               //configuration of TIME1
	TMR1IF = 0;                                 //clean TIME1 interrupt flag
	while((!SCL) || (!TMR1IF))display(distance,100);        //要获得连续显示，这儿要加上显示函数
	TMR1ON = 0;                                     // stop timer
	// finally get the range result from KS101B/KS103/KS103S
	SEN = 1;                                        // send start bit
	while(SEN);                                     // and wait for it to clear
	ACKDT = 0;                                      // acknowledge bit
	SSPIF = 0;

	SSPBUF = ADDRESS;                           // KS101B/KS103/KS103S I2C address
	while(!SSPIF);                              // wait for interrupt
	SSPIF = 0;                                  // then clear it.

	SSPBUF = 2;                             // address of register to read from - high byte of result
	while(!SSPIF);                                  //
	SSPIF = 0;                                      //

	RSEN = 1;                                       // send repeated start bit
	while(RSEN);                                    // and wait for it to clear
	SSPIF = 0;                                      //
	SSPBUF = ADDRESS+1;             // KS101B/KS103/KS103S I2C address - the read bit is set this time
	while(!SSPIF);                                  // wait for interrupt
	SSPIF = 0;                                      // then clear it.
	RCEN = 1;                                       // start receiving
	while(!BF);                             // wait for high byte of range
	range = SSPBUF<<8;                              // and get it
	ACKEN = 1;                                      // start acknowledge sequence
	while(ACKEN);                                   // wait for ack. sequence to end
	RCEN = 1;                                       // start receiving
	while(!BF);                             // wait for low byte of range
	range += SSPBUF;                                // and get it
	ACKDT = 1;                                      // not acknowledge for last byte
	ACKEN = 1;                                      // start acknowledge sequence
	while(ACKEN);                                   // wait for ack. sequence to end
```

```
        PEN = 1;                                                    // send stop bit
        while(PEN);                                                 //
        return range;
}

void send_command(unsigned char command)     //向 KS101B/KS103/KS103S 发送一个 8 位数据指令
{
        SEN = 1;                                                    // send start bit
        while(SEN);                                                 // and wait for it to clear
        while(!SSPIF);
        SSPIF = 0;
        SSPBUF = ADDRESS;                                   // KS101B/KS103/KS103S I2C address
        while(!SSPIF);                                      // wait for interrupt
        SSPIF = 0;                                          // then clear it.
        SSPBUF = 2;                                                 // address of register to write to
        while(!SSPIF);                                      //
        SSPIF = 0;
        SSPBUF = command;
        while(!SSPIF);                                      //
        SSPIF = 0;
        PEN = 1;                                                    // send stop bit
        while(PEN);                                                 //
}

void setup(void)                                //PIC16F877A 硬件 I2C 初始化配置
{
        SSPSTAT = 0x80;
        SSPCON = 0x38;
        SSPCON2 = 0x00;
        SSPADD = 50;
        OPTION=0B10001111;//PSA = 1;切换到 1:128 分频给 WDT,即 32.64ms 之内必须清一次看门狗
        TRISC=0B00011000;
        PORTC=0x01;
        RBIE=0;
}

void delay(unsigned int ms)
{
 unsigned char i;
 unsigned int j;
 for(i=0;i<70;i++)
        for(j=0;j<ms;j++)CLRWDT();
}
```

2) PIC16F877A 主机采用模拟 I²C 通讯与 KS101B/KS103/KS103S 连接控制 C 代码

```
#include <pic.h>                //4MHz 晶振
__CONFIG(XT&WDTEN); //开看门狗
#define   SDA   RD6         //  此引脚须上拉 4.7K 电阻至 VCC
#define   SCL   RD5         //  此引脚须上拉 4.7K 电阻至 VCC
#define   SDAPORT   TRISD6 //
#define   SCLPORT   TRISD5    //引脚 RD6，RD5 可换为其他任何 I/O 脚
bit eepromdi;
bit eepromdo;

void delay(void)
{
        unsigned char k;
        for(k=0;k<180;k++)
```

```
        asm("CLRWDT");
}

void delayms(unsigned char ms)//ms 延时函数
{
    unsigned int i,j;
    for (i=0;i<ms;i++)
        for(j=0;j<110;j++)
        asm("CLRWDT");
}

void i2cstart(void)    // start the i2c bus
{
    SCLPORT=0;
    SDAPORT=0;
    SCL=1;
    asm("NOP");    asm("NOP");    asm("NOP");    asm("NOP");    asm("NOP");
    SDA=1;
    delay();
    SDA=0;
    delay();
    SCL=0;
    delay();
}

void i2cstop(void)    // stop the i2c bus
{
    SDA=0;
    SCLPORT=0;
    SDAPORT=0;
    SDA=0;
    asm("NOP");    asm("NOP");    asm("NOP");    asm("NOP");    asm("NOP");
    SCL=1;
    delay();
    SDA=1;
    delay();
}

void bitin(void)     //read a bit from i2c bus
{
    eepromdi=1;
    SCLPORT=0;
    SDAPORT=1;
    SCL=1;
    asm("NOP");    asm("NOP");    asm("NOP");    asm("NOP");    asm("NOP");
    eepromdi=SDA;
    asm("NOP");    asm("NOP");    asm("NOP");    asm("NOP");    asm("NOP");
    SCL=0;
    asm("NOP");    asm("NOP");    asm("NOP");    asm("NOP");    asm("NOP");
}

void bitout(void)    //write a bit to i2c bus
{
    SCLPORT=0;
    SDAPORT=0;
    SDA=eepromdo;
    asm("NOP");    asm("NOP");    asm("NOP");    asm("NOP");    asm("NOP");
    SCL=1;
    asm("NOP");    asm("NOP");    asm("NOP");    asm("NOP");    asm("NOP");
    SCL=0;
    asm("NOP");    asm("NOP");    asm("NOP");    asm("NOP");    asm("NOP");
}
```

```
void i2cwrite(unsigned char sedata)    //write a byte to i2c bus
{
        unsigned char k;
        for(k=0;k<8;k++)
        {
        if(sedata&0x80)
        {
                eepromdo=1;
        }
        else
        {
                eepromdo=0;
        }
        sedata=sedata<<1;
        bitout();
        }
        bitin();
}

unsigned char i2cread(void)          //read a byte from i2c bus
{
        unsigned char redata;
        unsigned char m;
        for(m=0;m<8;m++)
        {
        redata=redata<<1;
        bitin();
        if(eepromdi==1)
        {
                redata|=0x01;
        }
        else
        {
                redata&=0xfe;
        }
        asm("NOP");
        }
        eepromdo=1;
        bitout();
        return redata;
}

unsigned char KS101B_read(unsigned char address,unsigned char buffer)
//////////////read register: address + register ,there will be 0xe8 + 0x02/0x03
{
        unsigned char eebuf3;
//      unsigned int range;
        i2cstart();
        i2cwrite(address);
        i2cwrite(buffer);
        i2cstart();
        i2cwrite(address+1);
        i2cstart();
        eebuf3=i2cread();
        i2cstop();
        return eebuf3;
}

void KS101B_write(unsigned char address,unsigned char buffer,unsigned char command)
  //////////////////write a command: address + register + command,there will be 0xe8 + 0x02 + 0xb0
{
        i2cstart();
        i2cwrite(address);
```

```
        i2cwrite(buffer);
        i2cwrite(command);
        i2cstop();
}

void change_i2c_address(addr_old,addr_new) // addr_old is the address now, addr_new will be the new address
{                                           //that you want change to
    delayms(200);                           //Protect the eeprom,you can delete this
    KS101B_write(addr_old,2,0x9a);
    delayms(1);
    KS101B_write(addr_old,2,0x92);
    delayms(1);
    KS101B_write(addr_old,2,0x9e);
    delayms(1);
    KS101B_write(addr_old,2, addr_new);
    delayms(100);                           //Protect the eeprom,you can delete this
}

unsigned int detect_KS101B(unsigned char address, unsigned char command)
{
unsigned int range1;
        KS101B_write(address,2,command);
        delayms(1);                         //安全延时,如果显示不清晰可以将延时调大一些
        delayms(80);                        //如果是探测温度此处延时需延长，使用 while(!SCL)此处可删除
        //SCLPORT=1;while(!SCL);
        // delayms(80)也可换为 SCLPORT=1;while(!SCL);直接查询 SCL 线的等待时间将最短，探测速度最快
        range1 = KS101B_read(address,2);
        range1 =(range1<<8) + KS101B_read(address,3);
        delayms(5);
        return range1;
}

void main(void)
{
   unsigned int range;
        //change_i2c_address(0xe8,0xfe);     ////将默认地址 0xe8 改为 0xfe
        delayms(200);
        while(1)
        {
                asm("CLRWDT");
                range = detect_KS101B(0xe8,0xb0);    //you just need the only one sentence to get the range.
                delayms(200);
        }
}
```

3) 51 单片机主机模拟 I²C 通讯与 KS101B/KS103/KS103S 连接控制 C 代码

```
#include <reg51.h>
#include <intrins.h>
sbit SDA=P3^6;                          //  此引脚须上拉 4.7K 电阻至 VCC
sbit SCL=P3^7;                          //  此引脚须上拉 4.7K 电阻至 VCC
unsigned int range;

void display(unsigned int range)
{
        //input your display function, please.
}
```

```
void delay(void)          //short delay  使用速度较快的单片机时，I²C 通讯可能不正常，在此函数中多加 4～8 个_nop_();即可
 {
      _nop_(); _nop_(); _nop_(); _nop_();
      _nop_(); _nop_(); _nop_(); _nop_();
      _nop_(); _nop_(); _nop_(); _nop_();
      _nop_(); _nop_(); _nop_(); _nop_();
 }

 void start(void)                  //I2C start
 {
       SDA = 1;
      delay();
      SCL = 1;
      delay();
      SDA = 0;
      delay();
 }

 void stop(void)                    //I2C stop
 {
      SDA = 0;
      delay();
      SCL = 1;
      delay();
      SDA = 1;
      delay();
 }

 void ack(void)                     //ack
 {
      unsigned char i;
      SCL = 1;
      delay();
      while(SDA == 1 && i < 200)
      {
           i++;
      }
      SCL = 0;
      delay();
 }

 void no_ack()                      //not ack
{
      SDA = 1;
      delay();
      SCL = 1;
      delay();
      SCL = 0;
      delay();
}

void i2c_write_byte(unsigned char dat)          //write a byte
 {
      unsigned char i;
      SCL = 0;
      for(i = 0; i < 8; i++)
      {
           if(dat & 0x80)
           {
                SDA = 1;
           }
           else
```

```
            {
                SDA = 0;
            }
            dat = dat << 1;
            delay();
            SCL = 1;
            delay();
            SCL = 0;
            delay();
        }
        SDA = 1;
        delay();
}

unsigned char i2c_read_byte(void)       //read a byte
{
        unsigned char i,dat;
        SCL = 0;
        delay();
        SDA = 1;
        delay();
        for(i = 0; i < 8; i++)
        {
            SCL = 1;
            delay();
            dat = dat << 1;
            if(SDA == 1)
            {
                dat++;
            }
            SCL = 0;
            delay();
        }
        return dat;
}

void init_i2c(void)                     //i2c init
{
        SDA = 1;
        SCL = 1;
}

void write_byte(unsigned char address,unsigned char reg,unsigned char command) //address+register+command
{
        init_i2c();
        start();
        i2c_write_byte(address);
        ack();
        i2c_write_byte(reg);
        ack();
        i2c_write_byte(command);
        ack();
        stop();
}

unsigned char read_byte(unsigned char address,unsigned char reg)    //address(with bit 0 set) + register
{
    unsigned char dat;
    init_i2c();
    start();
    i2c_write_byte(address);
    ack();
    i2c_write_byte(reg);
```

```
    ack();
    start();
    i2c_write_byte(address+1);
    ack();
    delay();delay();delay();delay();delay();      //此处延时对于 STC89C 系列单片机，可以删除，如果对于快速单
//片机，需要加至少 50us 的延时，才可以可靠读到数据
    dat = i2c_read_byte();
    no_ack();
    stop();
    return dat;
 }

void delayms(unsigned int ms)       //delay ms
{
     unsigned char i;
     unsigned int j;
     for(i=0;i<110;i++)
          for(j=0;j<ms;j++);
}

void change_i2c_address(unsigned char addr_old, unsigned char addr_new)
// addr_old is the address now, addr_new will be the new address
{                                                  //that you want change to
   delayms(2000);                                   // Protect the eeprom ,you can delete this sentence
   write_byte(addr_old,2,0x9a);
   delayms(1);
   write_byte(addr_old,2,0x92);
   delayms(1);
   write_byte(addr_old,2,0x9e);
   delayms(1);
   write_byte(addr_old,2, addr_new);
   delayms(500);                                  //Protect the eeprom, you can delete this sentence
}

unsigned int detect(unsigned char address,unsigned char command)   //0xe8(address) + 0xb0(command)
{
     unsigned int distance,count;
     write_byte(address,2,command);                    //use command "0xb0" to detect the distance
     delayms(1);                                       //安全延时,如果显示不清晰可以将延时调大一些
     //delayms(80);                                    //如果是探测温度此处延时需根据表 1 所列时间相应延长
     count=800;
     while(--count || !SCL)                             //等待探测结束，count 值调小将减小探测等待时间
     {
          ;                                            // 空语句
          /////display(range);                              //显示语句，可根据需要保留或删除
     }
//   while(!SCL)display(range);                        //you can delete “display(range)”
//通过查询 SCL 线来智能识别探测是否结束，使用本语句可删除上条语句(count=800;while…)以节省探测时间
     distance=read_byte(address,2);
     distance <<= 8;
     distance += read_byte(address,3);
     return distance;                    //return 16 bit distance in millimeter
}

void main(void)
{
     //change_i2c_address(0xe8,0xfe);   //change default address 0xe8 to 0xfe
     while(1)
     {
          range = detect(0xe8,0xb0);
          //0xe8 is the address; 0xb0 is the command.you just need the only one sentence to get the range.
          display(range);
          delayms(200);
```

```
    }
}
```

4) STM32 CORTEX-3 ARM 主机模拟 I²C 通讯与 KS101B/KS103/KS103S 连接控制 C 代码

//单片机型号：STM32F103RBT //本程序未示出所有系统配置函数

```
#include <stm32f10x_lib.h>
#include "sys.h"
#include "usart.h"
#include "delay.h"

u8 KS103_ReadOneByte(u8 address, u8 reg)
{
    u8 temp=0;

   IIC_Start();
   IIC_Send_Byte(address);     //发送低地址
    IIC_Wait_Ack();
    IIC_Send_Byte(reg);     //发送低地址
    IIC_Wait_Ack();
    IIC_Start();
    IIC_Send_Byte(address + 1);                  //进入接收模式
    IIC_Wait_Ack();

    delay_us(50);          //增加此代码通信成功!!!
   temp=IIC_Read_Byte(0);         //读寄存器 3
   IIC_Stop();//产生一个停止条件
    return temp;
}


void KS103_WriteOneByte(u8 address,u8 reg,u8 command)
{

   IIC_Start();
    IIC_Send_Byte(address);            //发送写命令
    IIC_Wait_Ack();
    IIC_Send_Byte(reg);//发送高地址
    IIC_Wait_Ack();
   IIC_Send_Byte(command);     //发送低地址
    IIC_Wait_Ack();
   IIC_Stop();//产生一个停止条件
}



void IIC_Init(void)
{
    RCC->APB2ENR|=1<<4;//先使能外设 IO PORTC 时钟
    GPIOC->CRH&=0XFFF00FFF;//PC11/12 推挽输出
    GPIOC->CRH|=0X00033000;
    GPIOC->ODR|=3<<11;        //PC11,12 输出高
}
//产生 IIC 起始信号
void IIC_Start(void)
{
    SDA_OUT();        //sda 线输出
    IIC_SDA=1;
    IIC_SCL=1;
```

```
	delay_us(10);
	IIC_SDA=0;//START:when CLK is high,DATA change form high to low
	delay_us(10);
	IIC_SCL=0;//钳住 I2C 总线，准备发送或接收数据
}
//产生 IIC 停止信号
void IIC_Stop(void)
{
	SDA_OUT();//sda 线输出
	IIC_SCL=0;
	IIC_SDA=0;//STOP:when CLK is high DATA change form low to high
	delay_us(10);
	IIC_SCL=1;
	IIC_SDA=1;//发送 I2C 总线结束信号
	delay_us(10);
}
//等待应答信号到来
//返回值：1，接收应答失败
//		0，接收应答成功
u8 IIC_Wait_Ack(void)
{
	u8 ucErrTime=0;
	SDA_IN();		//SDA 设置为输入
	IIC_SDA=1;delay_us(6);
	IIC_SCL=1;delay_us(6);
	while(READ_SDA)
	{
		ucErrTime++;
		if(ucErrTime>250)
		{
			IIC_Stop();
			return 1;
		}
	}
	IIC_SCL=0;//时钟输出 0
	return 0;
}
//产生 ACK 应答
void IIC_Ack(void)
{
	IIC_SCL=0;
	SDA_OUT();
	IIC_SDA=0;
	delay_us(10);
	IIC_SCL=1;
	delay_us(10);
	IIC_SCL=0;
}
//不产生 ACK 应答
void IIC_NAck(void)
{
	IIC_SCL=0;
	SDA_OUT();
	IIC_SDA=1;
	delay_us(10);
	IIC_SCL=1;
	delay_us(10);
	IIC_SCL=0;
}
//IIC 发送一个字节
//返回从机有无应答
//1，有应答
//0，无应答
```

```
void IIC_Send_Byte(u8 txd)
{
    u8 t;
     SDA_OUT();
    IIC_SCL=0;//拉低时钟开始数据传输
    for(t=0;t<8;t++)
    {
        IIC_SDA=(txd&0x80)>>7;
        txd<<=1;
          delay_us(10);
          IIC_SCL=1;
          delay_us(10);
          IIC_SCL=0;
          delay_us(10);
    }
}
//读 1 个字节，ack=1 时，发送 ACK，ack=0，发送 nACK
u8 IIC_Read_Byte(unsigned char ack)
{
     unsigned char i,receive=0;
     SDA_IN();//SDA 设置为输入
    for(i=0;i<8;i++ )
     {
        IIC_SCL=0;
        delay_us(10);
          IIC_SCL=1;
        receive<<=1;
        if(READ_SDA)receive++;
          delay_us(5);
    }
    if (!ack)
        IIC_NAck();//发送 nACK
    else
        IIC_Ack(); //发送 ACK
    return receive;
}

int main(void)
{
     u16 range;
     Stm32_Clock_Init(9);//系统时钟设置
     delay_init(72);            //延时初始化
     uart_init(72,9600); //串口 1 初始化
     while(1)
     {
                KS103_WriteOneByte(0XE8,0X02,0XB0);
                delay_ms(80);
                range = KS103_ReadOneByte(0xe8, 0x02);
                range <<= 8;
                range += KS103_ReadOneByte(0xe8, 0x03);
     }
}
```