KS101 技术说明书

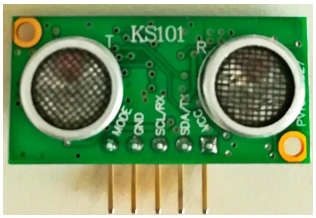
版本：Ver. 1.00

创新技术源自导向技术

深圳市导向机电技术有限公司

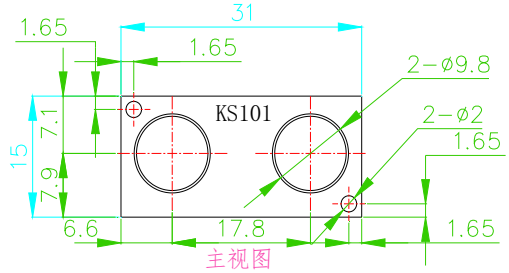
Modify Date	Content	Edit	Revsion	Note
Aug. 21,2017	Initial release.	X.Q.	1.00	

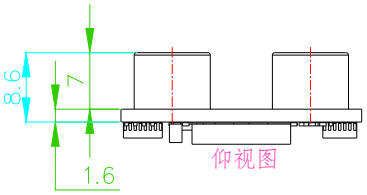
：



KS101-BENDING

KS101-NULL

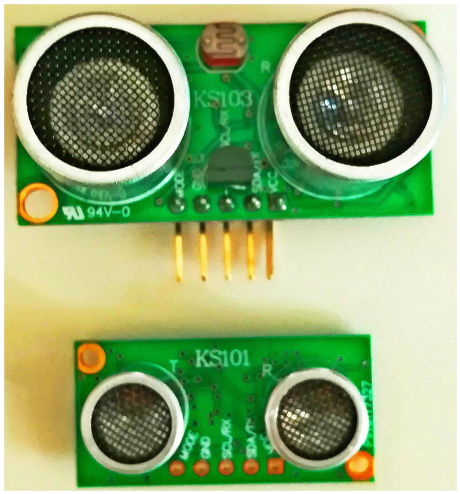




KS101 功能摘要：

- **收发分体**式设计，质量仅 5g；
- 支持 2 台 KS101 一发一收或远距离对射；
- 探测范围：**1cm-5.5m**，精度±1cm；
- 探测频率最高可达 50Hz，即每秒可探测 50 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，兼容 KS103 协议；
- 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)；
- 5s 未收到 I²C 控制指令自动进入 uA 级休眠，并可随时被主机 I²C 控制指令唤醒；
- 使用工业级配置，工作温度 (-30℃~+85℃)；
- 宽工作电压范围 (3.0V~5.5V)；
- I²C 模式通信速率 50~100kbit/s；串口通信速率默认 9600bps；客户可修改为 57600、115200bps；
- 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
- 上电自检，自检正常与异常上报错误代码，并智能修复；
- 环保无铅

KS101 与 KS103 外观对比：



KS101 电性能参数：

工作电压：***3.0V~5.5V*** 直流电源，推荐 ***5.0V。***

工作时瞬间最大电流：***50mA@5.0V, typical***。

工作电流：***10.6mA@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 I²C 控制指令自动进入 uA 级休眠，并可随时被主机 I²C 控制指令唤醒。

在 KS101 上连线引脚上标识有：VCC、SDA/TX(简称 SDA)、SCL/RX(简称 SCL)、GND、MODE 及 AN/SW 引脚。MODE 引脚为 I²C 模式与 TTL 串口模式设置引脚，该引脚悬空时，KS101 工作于 I²C 模式；在上电之前 MODE 引脚接 0V 地时，KS101 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

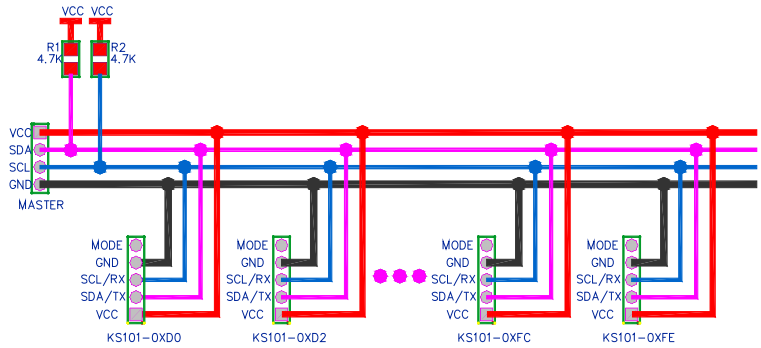
KS101 连线：

KS101 上依次引脚上标识有：VCC、SDA/TX(简称 SDA)、SCL/RX(简称 SCL)、GND、MODE 及 AN/SW。MODE 引脚悬空时，KS101 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。KS101 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 的电压将可能影响测距量程。并且，**严禁将 VCC 与 GND 接反，严禁 VCC 电压超过 6V**。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）：



KS101 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 KS101 重新上电，可观察到 LED 显示新地址。在修改 KS101 的 I²C 地址过程中，严禁突然给 KS101 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。

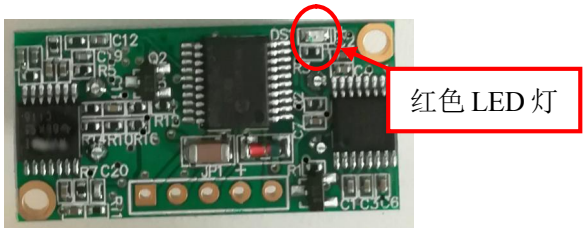
在 I²C 地址设置为不同之后，在主机的两根 I²C 总线上可以同时连接 20 个 KS101。主机在对其中一个 KS101 模块进行控制时，其他模块自动进入微瓦级功耗休眠模式，因此不必担心电流供应不足问题。

控制本模块的主机设备如果只支持 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。但是，要改出的新地址必须是 8 位的。否则改地址函数将不被 KS101 响应。

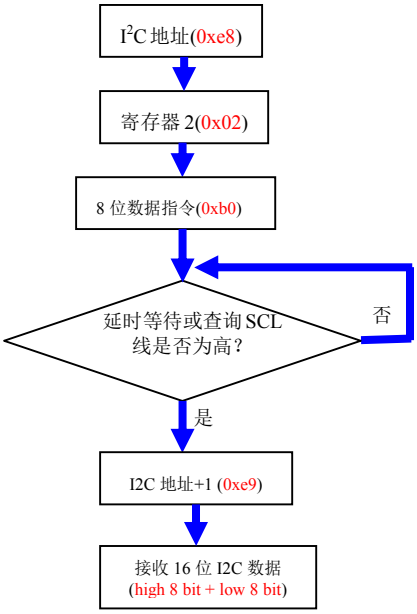
KS101 工作流程：

在 KS101 上电后，红色 LED 灯会立即快闪一下，提示用户接线正确。如果 LED 灯未亮代表正负接反，请立即断电。接线正确后系统会开始自检，自检正常后报错误代码 0xe0 代表自检初始化成功。自检之后其背面的 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，红色 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 闪烁时的红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。



KS101 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。KS101 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：



探测结束智能识别

KS101 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 I²C 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。KS101 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 I²C 总线接收到 KS101 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 KS101 响应延迟。由于最快的探测指令 0x01 也需要 10ms 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 I²C 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。

实际使用时，发现部分客户会使 KS101 工作在极高的频率之下。这将导致相邻探测的结果可能受影响。因此，KS101 的各探测指令均有加时保护。即 KS101 每次探测最短时间被故意设

计为 10ms，以防相邻探测之干扰。

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 KS101 后 SCL 线仍然不会拉低。如果想恢复 I²C 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：“I²C 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。KS101 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，KS101 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**指令及寄存器储存值汇总如下：

寄存器	命令	返回值范围（10 进制）	返回值范围（16 进制）	备注
0		1-254	0x01-0xff	程序版本标识及厂家标识。
1		1-252	0x01-0xfc	制造日期标识。16 位数据的高 8 位为制造年份，低 8 位为制造月份。11 年开始制造标识为 1；12 年开始制造标识为 2；……；25 年开始制造标识为 F；26 年开始制造标识为 0；27 年开始制造标识为 1。月份：1 月份标识为 1；10 月份标识为 A；12 月份对应 C。
2	0x01			作为发射探头用，发射一束 8 个 8.5mm 波长超声波
2	0xb0	130-11280mm	0x0d-0x2c10mm	1cm-5m 范围，普通距离，返回 mm，探测最大耗时约 68ms。波束角可调。默认模式 18cm-11m。
2	0xb1	130-11280mm	0x0d-0x2c10mm	作为接收探头用，1cm-11m 范围，普通距离，返回 mm，探测最大耗时约 68ms。波束角可调。
2	0xb2	768-65278μs	0xc8-0xfefeμs	768μs(13cm）-65278μs（11m)范围，普通距离，返回 μs，探测最大耗时约 66ms。波束角可调。
2	0xb3	0-65278μs	0-0xfefeμs	作为接收探头用，0μs(0cm）-65278μs（11m)范围，普通距离，返回 μs，探测最大耗时约 66ms。波束角不可调。
2	0xb4	270-11280mm	0x1b-0x2c10mm	1cm-5m 范围，普通距离，返回 mm，探测最大耗时约 68ms。波束角可调。默认模式 18cm-11m。
2	0xb5	10-11280mm	0x0a-0x2c10mm	作为接收探头用，1cm-11m 范围，普通距离，返回 mm，探测最大耗时约 68ms。波束角不可调。
2	0xb8	130-11280mm	0x0d-0x2c10mm	1cm-11m 范围，普通距离，返回 mm，探测最大耗时约 68ms。波束角可调。
2	0xba	60-65278μs	0x3c-0xfefeμs	60μs(1cm）-65278μs（11m)范围，普通距离，返

				回 μs, 探测最大耗时约 66ms。波束角可调。
2	0xbc	60-11280mm	0x3c-0x2c10mm	1cm-11m 范围，普通距离，返回 mm, 探测最大耗时约 68ms。波束角不可调。
2	0x71	无	无	第一级降噪。 所有指令指令将工作于第一级降噪。出厂默认设置。适用于电池供电
2	0x72	无	无	第二级降噪。 所有指令指令将工作于第一级降噪。适用于 USB 供电。
2	0x73	无	无	第三级降噪。 所有指令指令将工作于第一级降噪。适用于较长距离 USB 供电。
2	0x74	无	无	第四级降噪。 所有指令指令将工作于第一级降噪。适用于开关电源供电
2	0x76	无	无	将串口通信波特率配置为 1200bps
2	0x77	无	无	将串口通信波特率配置为 9600bps，出厂默认设置
2	0x78	无	无	将串口通信波特率配置为 57600bps
2	0x79	无	无	将串口通信波特率配置为 115200bps
2	0x7a	无	无	约 50° 波束角，盲区 18cm；本配置仅仅对 0xb0、0xb2、0xb4 三个指令有效，出厂默认设置
2	0x7b	无	无	约 45° 波束角；本配置仅仅对 0xb0、0xb2、0xb4 三个指令有效
2	0x7c	无	无	约 50° 波束角，盲区 13cm；本配置仅仅对 0xb0、0xb2、0xb4 三个指令有效
2	0x7d	无	无	约 60° 波束角；本配置仅仅对 0xb0、0xb2、0xb4 三个指令有效
2	0x95	无	无	0x71~0x7d 参数配置第二时序
2	0x98	无	无	0x71~0x7d 参数配置第三时序
2	0x9c	无	无	0x71~0x7d 参数配置第一时序
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	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
3		0-255	0-0xff	读数据时寄存器 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			0x76 或 0x77 或 0x78 或 0x79	本寄存器存储的是串口通信波特率 0x76~0x79，供查询用。0x76 对应波特率 2400bps；0x77 对应波特率 9600bps；0x78 对应波特率 57600bps；0x79

				对应波特率 115200bps
5			0xd0~0xfe	本寄存器存储的是 20 个 I²C 或串口地址，不包括 0xf0, 0xf2, 0xf4,0xf6，供查询用。
6			0x71~0x74	本寄存器存储的是降噪级别 0x71~0x74，供查询用
7			0x7a~0x7d	本寄存器存储的是波束角大小 0x7a~0x7d，供查询用
8			0xe0	初始化正常
			0xe1	硬件错误
			0xe2	电源噪音过大或附近有较大干扰，或者有导体掉落到了 PCB 板上并与元件有电接触
			0xe3	通信错误或通信丢 BIT
			0xe4	地址错误或 I2C 接线无上拉电阻或线路接反接错。此时 LED 灯若不亮，需马上断电！
9			0xe5	发生 0xe2 错误自动修复不成功，请检查硬件或干扰源
			0xe6	发生 0xe2 错误自动修复成功，恢复正常
			0	忽略
10-15		0	0	保留供升级用

表 1

距离探测

通过“I²C 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（I²C 地址 +1，具体参考附件例程），即可取得 16 位的距离数据。返回 mm 距离值是按照 25℃标准通过实际探测时间换算而来的距离值；返回 μs 值代表超声波从发出到遇到障碍物反射收回所经历的时间。

对射功能及 0 盲区功能实现

通过使用 2 台 KS101 协同工作，可以实现对射功能或 0 盲区功能。1 台 KS101 通过发送 0x01 指令发出一束 8.5mm 波长超声波，另 1 台 KS101 通过发送 0xb1 或 0xb3 或 0xb5 指令获取返回的结果。请注意：获取的结果需加上通信时间，声速可按 340m/s 即 340mm/ms 计算。如果无法计算通信时间，可以直接通过米尺校准，校准一次后续无需再次校准。

例如使用 0x01+0xb5 组合指令，两台 KS101 对射式安装，返回的是 5000mm，通讯时间假设程序设计为 20ms，那么实际距离为 5000mm*2+340mm/ms*20ms = 16800mm。
如果两台 KS101 反射式安装，即同一朝向。返回的是 5000mm，通讯时间假设程序设计为 2ms，那么实际距离为 5000+340mm/ms*2ms/2 = 5340mm.

电源降噪指令(0x71, 0x72, 0x73, 0x74)及波束角配置指令(0x7a-0x7d)

KS101 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 0x71，0x72，0x73，0x74 命令来配置 KS101 测距模块的杂波抑制功能。0x71 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。0x72 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，同时也是出厂默认设置。0x73 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。0x74 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。

用户可以通过发送 0x7a，0x7b，0x7c，0x7d 来根据实际障碍物情况选择指令。参见表 1.

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 + 0x9c; I²C 地址 + 寄存器 2 + 0x95; I²C 地址 + 寄存器 2 + 0x98; I²C 地址 + 寄存器 2 + 0x71/0x72/0x73/0x74/0x7a/0x7b/0x7c/0x7d”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：

config_0x71_0x7d(0xe8,0x72); //如果 I²C 地址为 0xe8

delayms(2000);

将本模块配置为 50°波束角，配置代码如下：

config_0x71_0x7d(0xe8,0x7b); //如果 I²C 地址为 0xe8

delayms(2000);

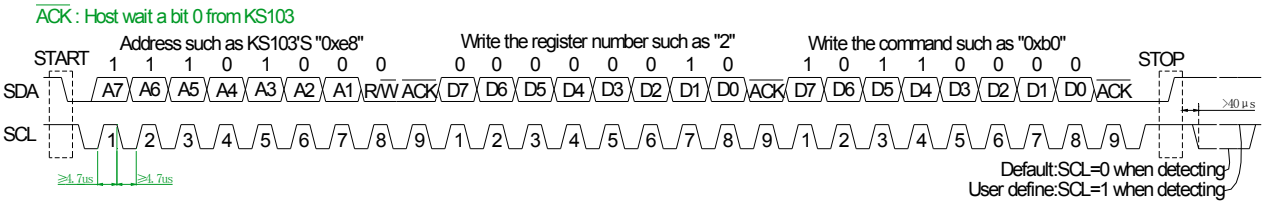
配置代码请放在程序的初始化函数中，即 while(1)循环之前，以保护模块。KS101 收到有效配置指令之后，LED 灯将长亮 5s，表明配置成功。

KS101 在重新上电后将按新配置运行。

时序图

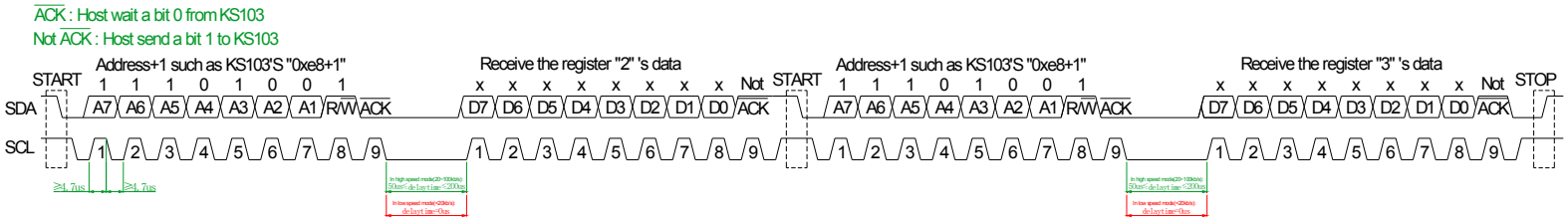
时序图 1：发送探测指令，指令格式为(Such as register 2)：

I²C 地址	寄存器 2	8 位数据指令



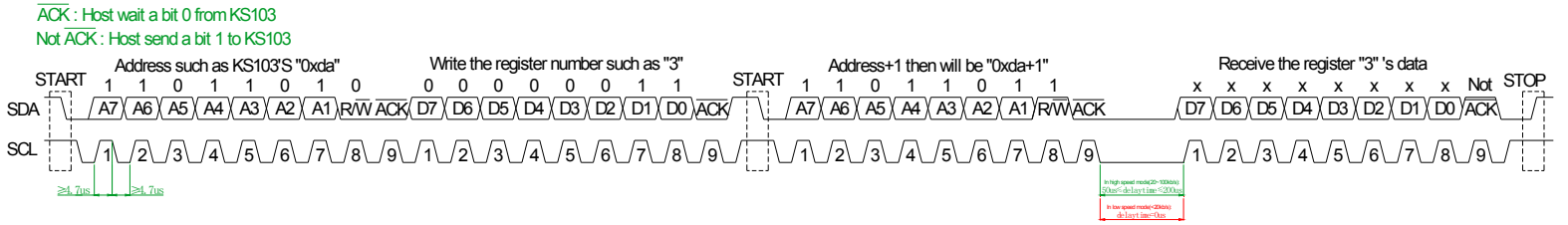
时序图 2：执行完时序图 1 后，等待 SCL 变高或延时 100ms 后接收 16 位数据，先高位后低位，指令格式为：

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 KS101 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 KS101 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，KS101 收到主机的有效探测数据红色 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。

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 I²C 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：“I²C 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 KS101 会自动保存，并立即按照新配置工作。KS101 在重新上电后将按新配置运行。

TTL 串口模式

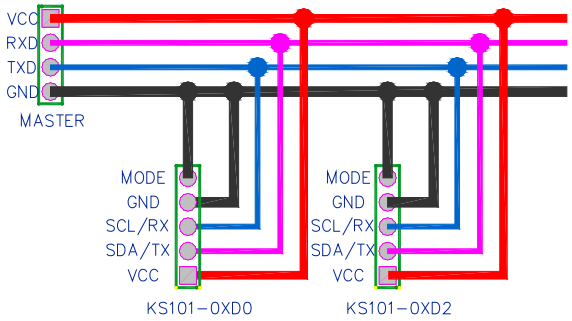
KS101 的 TTL 串口模式波特率为 9600bps，1 启动位，8 数据位，1 停止位，无校验位，TTL 电平。波特率 9600bps 可通过发送指令修改为 2400bps，57600bps，115200bps 波特率。

KS101 连线：

在 KS101 上连线引脚上标识有：VCC、SDA/TX、SCL/RX、GND、MODE 及 AN/SW。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 KS101 在上电后再将 MODE 引脚接 0V 地，模块将仍然工作于 I²C 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 的电压将可能影响测距量程。并且，**严禁将 VCC 与 GND 接反，严禁 VCC 电压超过 6V**。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**）：



KS101 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 KS101，**最多请不要超过 2 台**。 如果需要接多台，请加一个 485 转 TTL 芯片。

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。

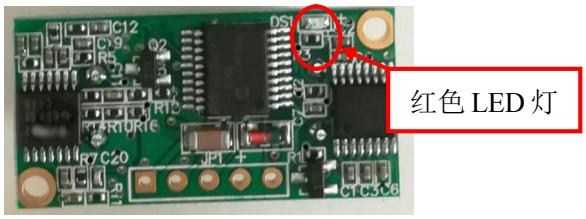
修改完毕后 LED 灯将长亮，给 KS101 重新上电，可观察到 LED 显示新地址。在修改 KS101 的串口地址过程中，严禁突然给 KS101 断电。修改地址函数请不要放在 while(1)循环中，保证在程序中上电后只运行一次。

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 2 个 KS101。主机在对其中一个 KS101 模块进行控制时，其他模块不会受到影响。

KS101 工作流程：

在 KS101 上电后，红色 LED 灯会立即快闪一下，提示用户接线正确。如果 LED 灯未亮代表正负接反，请立即断电。接线正确后系统会开始自检，自检正常后报错误代码 0xe0 代表自检初始化成功。自检之后其背面的 LED 会以二进制方式闪烁显示其 8 位 I^2C 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，红色 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 闪烁时的红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。



KS101 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。

上段所述，KS101 上电后报错误代码 0xe0 代表自检初始化成功。实际上，KS101 将向上位机发送如下十六进制代码：

67 78 77 E8 71 7A E0 00 0A 64 61 75 78 69 2E 63 6F 6D 0A 67 75 69 64 2E 74 61 6F 62 61 6F 2E 63 6F 6D 0A

其中依次地，0x67 为程序版本，存储在寄存器 0 中；0x78 为制造日期标识存储在寄存器 1 中；0x77 为串口通信波特率，存储在寄存器 4 中；0xE8 为 I2C 或串口地址，存储在寄存器 5 中；0x71 为降噪级别，存储在寄存器 6 中；0x7A 为波束角大小，存储在寄存器 7 中；0xE0 为错误代码，存储在寄存器 8 中；0x00 为错误代码，存储在寄存器 9 中。各十六进制值说明请参考本说明书第 4 至 6 页的表 1. 再往后返回的是 0x0A，此为换行标识。后面的返回值请转为字符格式观察，其返回的是制造公司网站等信息。

当寄存器 8 返回的错误代码为 0xE2 时，KS101 将尝试自我修复。修复成功通过寄存器 9 返回 0xE6；修复失败返回 0xE5.

KS101 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 KS101 的探测数据高 8 位 —> 接收 KS101 的探测数据低 8 位

KS101 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 KS101 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。

探测结束智能识别

由于探测指令发出后 KS101 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，KS101 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			作为发射探头用，发射一束 8 个 8.5mm 波长超声波
2	0xb0	130-11280mm	0x0d-0x2c10mm	1cm-5m 范围，普通距离，返回 mm, 探测最大耗时约 68ms。波束角可调。默认模式 18cm-11m。
2	0xb1	130-11280mm	0x0d-0x2c10mm	作为接收探头用，1cm-11m 范围，普通距离，返回 mm, 探测最大耗时约 68ms。波束角可调。
2	0xb2	768-65278μs	0xc8-0xfefeμs	768μs(13cm）-65278μs（11m)范围，普通距离，返回 μs, 探测最大耗时约 66ms。波束角可调。
2	0xb3	0-65278μs	0-0xfefeμs	作为接收探头用，0μs(0cm）-65278μs（11m)范围，普通距离，返回 μs, 探测最大耗时约 66ms。波束角不可调。
2	0xb4	270-11280mm	0x1b-0x2c10mm	1cm-5m 范围，普通距离，返回 mm, 探测最大耗时约 68ms。波束角可调。默认模式 18cm-11m。
2	0xb5	10-11280mm	0x0a-0x2c10mm	作为接收探头用，1cm-11m 范围，普通距离，返回 mm, 探测最大耗时约 68ms。波束角不可调。
2	0xb8	130-11280mm	0x0d-0x2c10mm	1cm-11m 范围，普通距离，返回 mm, 探测最大耗时约 68ms。波束角可调。
2	0xba	60-65278μs	0x3c-0xfefeμs	60μs(1cm）-65278μs（11m)范围，普通距离，返回 μs, 探测最大耗时约 66ms。波束角可调。
2	0xbc	60-11280mm	0x3c-0x2c10mm	1cm-11m 范围，普通距离，返回 mm, 探测最大耗时约 68ms。波束角不可调。
2	0x71	无	无	第一级降噪。 所有指令指令将工作于第一级降噪。出厂默认设置。适用于电池供电
2	0x72	无	无	第二级降噪。 所有指令指令将工作于第一级降噪。适用于 USB 供电。
2	0x73	无	无	第三级降噪。 所有指令指令将工作于第一级降噪。适用于较长距离 USB 供电。
2	0x74	无	无	第四级降噪。

				所有指令指令将工作于第一级降噪。适用于开关电源供电
2	0x76	无	无	将串口通信波特率配置为 1200bps
2	0x77	无	无	将串口通信波特率配置为 9600bps，出厂默认设置
2	0x78	无	无	将串口通信波特率配置为 57600bps
2	0x79	无	无	将串口通信波特率配置为 115200bps
2	0x7a	无	无	约 50° 波束角，盲区 18cm；本配置仅仅对 0xb0、0xb2、0xb4 三个指令有效，出厂默认设置
2	0x7b	无	无	约 45° 波束角；本配置仅仅对 0xb0、0xb2、0xb4 三个指令有效
2	0x7c	无	无	约 50° 波束角，盲区 13cm；本配置仅仅对 0xb0、0xb2、0xb4 三个指令有效
2	0x7d	无	无	约 60° 波束角；本配置仅仅对 0xb0、0xb2、0xb4 三个指令有效
2	0x95	无	无	0x71~0x7d 参数配置第二时序
2	0x98	无	无	0x71~0x7d 参数配置第三时序
2	0x9c	无	无	0x71~0x7d 参数配置第一时序
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	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

表 2

距离探测

通过“串口地址 + 寄存器 2 + 距离探测指令”时序，主机此时开启串口中断，探测完毕后，即可通过串口中断获取读寄存器 2 及寄存器 3 的值，即可取得 16 位的距离数据。返回 mm 距离值是按照 25℃标准通过实际探测时间换算而来的距离值；返回 μs 值代表超声波从发出到遇到障碍物反射收回所经历的时间。

对射功能及 0 盲区功能实现

通过使用 2 台 KS101 协同工作，可以实现对射功能或 0 盲区功能。1 台 KS101 通过发送 0x01 指令发出一束 8.5mm 波长超声波，另 1 台 KS101 通过发送 0xb1 或 0xb3 或 0xb5 指令获取返回的结果。请注意：获取的结果需加上通信时间，声速可按 340m/s 即 340mm/ms 计算。如果无法计算通信时间，可以直接通过米尺校准，校准一次后续无需再次校准。

例如使用 0x01+0xb5 组合指令，两台 KS101 对射式安装，返回的是 5000mm，通讯时间假设程序设计为 20ms，那么实际距离为 5000mm*2+340mm/ms*20ms = 16800mm。

如果两台 KS101 反射式安装，即同一朝向。返回的是 5000mm，通讯时间假设程序设计为 2ms，那么实际距离为 5000+340mm/ms*2ms/2 = 5340mm.

电源降噪指令(0x71, 0x72, 0x73, 0x74)及不同物体探测配置指令(0x7a, 0x7b, 0x7c, 0x7d)

KS101 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 0x71，0x72，0x73，0x74 命令来配置 KS101 测距模块的杂波抑制

功能。0x71 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。0x72 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。0x73 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。0x74 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。

用户可以通过发送 0x7a,0x7b,0x7c,0x7d 来根据实际障碍物情况选择指令。参见表 1.

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 + 0x9c; TTL 串口地址 + 寄存器 2 + 0x95; TTL 串口地址+寄存器 2 + 0x98; TTL 串口地址 + 寄存器 2 + 0x71/0x72/0x73/0x74/0x7a/0x7b/0x7c/0x7d”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。KS101 收到有效配置指令之后，LED 灯将长亮 5s，表明配置成功。

KS101 在重新上电后将按新配置运行。

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：

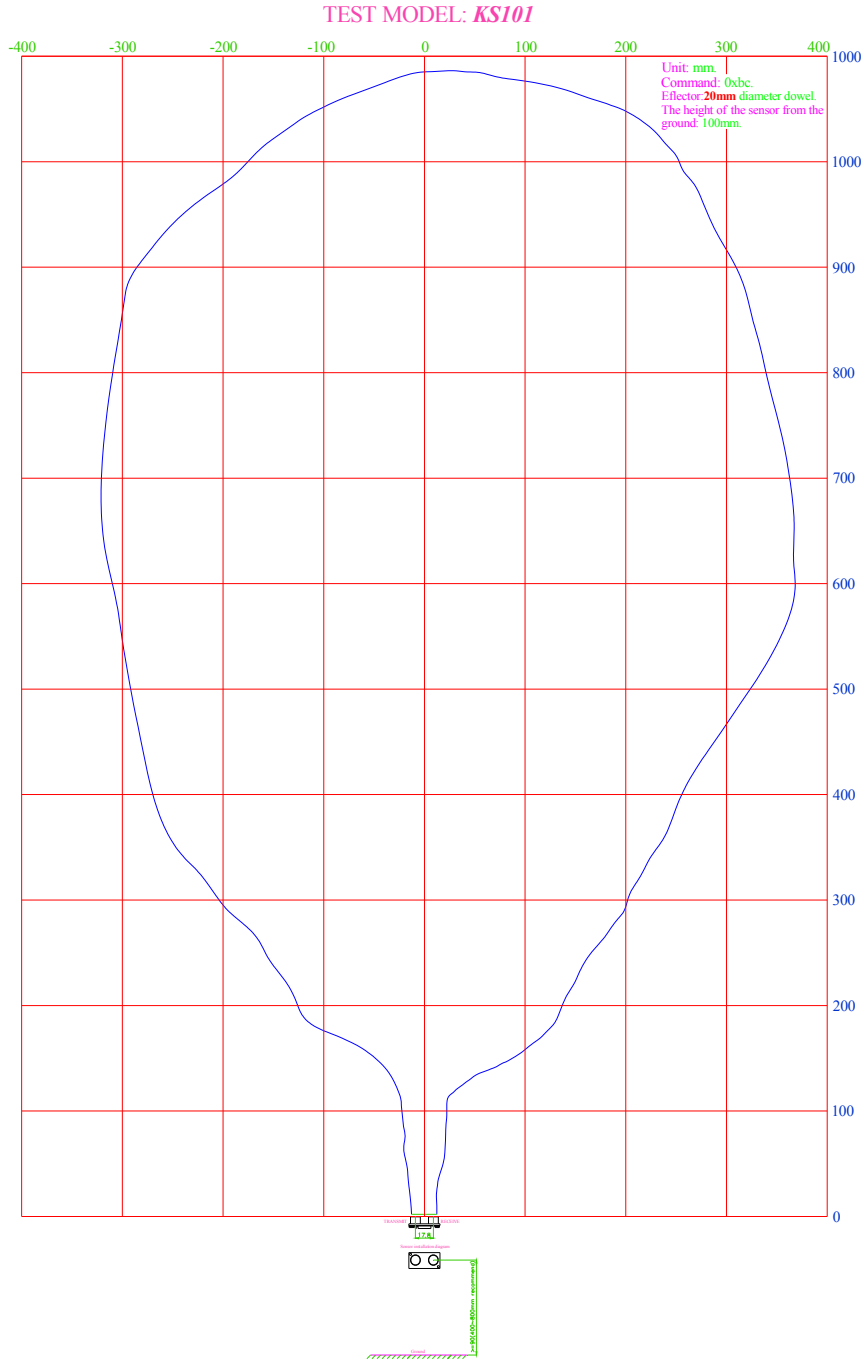
探测结果高 8 位	探测结果低 8 位

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。

休眠等待时间设置

串口模式不进入休眠。

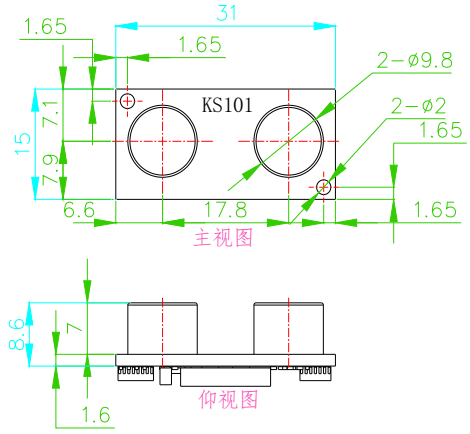
探测范围(40KHz 超声波)



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	PVC 塑料管	N/A	N/A	N/A
Fig.B	直径 20mm	ABS 塑料管	90mm	5V	0x71
Fig.C	长 920mm 宽 860mm	2 坑瓦楞纸板	N/A	N/A	N/A

备注：本波束角测试图为 KS101 测试结果,仅供参考用。KS101 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。

KS101 安装尺寸(单位：毫米)：



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 KS101 连接控制 C 代码
2) PIC16F877A 主机采用模拟 I²C 通讯与 KS101 连接控制 C 代码
3) 51 单片机主机模拟 I²C 通讯与 KS101 连接控制 C 代码
4) STM32 CORTEX-3 ARM 主机模拟 I²C 通讯与 KS101 连接控制 C 代码

1) PIC16F877A 主机采用硬件 I²C 通讯与 KS101 连接控制 C 代码

/*电路连接方式：PIC16F877A 的 IO 口 SCL、SDA 与 KS101 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_KS101(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_KS101(0xe8,0x30);    //Address:0xe8; command:0x30.
                                               //Get detect result from KS101, 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 KS101
    while(SEN);                                                 // wait for it to clear
    while(!SSPIF);                                              // wait for interrupt
    SSPIF = 0;                                                  // then clear it.

    SSPBUF = addr_old;                                      // KS101'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 KS101 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;                          // KS101'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 KS101 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;                          // KS101'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 KS101 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;                          // KS101'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 KS101 to break to do something
```

```
}

unsigned int detect_KS101(unsigned char ADDRESS, unsigned char command)
{
//ADDRESS will be KS101's address such as 0x30, command will be the detect command such as 0x30
unsigned int range=0;
        SEN = 1;                                        // send start bit
        while(SEN);                                     // and wait for it to clear
        while(!SSPIF);
        SSPIF = 0;
        SSPBUF = ADDRESS;                               // KS101'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 KS101 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 KS101
        SEN = 1;                                            // send start bit
        while(SEN);                                         // and wait for it to clear
        ACKDT = 0;                                          // acknowledge bit
        SSPIF = 0;

        SSPBUF = ADDRESS;                               // KS101 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;             // KS101 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 KS101 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;                                     // KS101 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 KS101 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 KS101_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 KS101_write(unsigned char address,unsigned char buffer,unsigned char command)
  ////////////////////write a command: address + register + command,there will be 0xe8 + 0x02 + 0x30
{
        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
   KS101_write(addr_old,2,0x9a);
   delayms(1);
   KS101_write(addr_old,2,0x92);
   delayms(1);
   KS101_write(addr_old,2,0x9e);
   delayms(1);
   KS101_write(addr_old,2, addr_new);
   delayms(100);                            //Protect the eeprom,you can delete this
}

unsigned int detect_KS101(unsigned char address, unsigned char command)
{
unsigned int range1;
      KS101_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 = KS101_read(address,2);
      range1 =(range1<<8) + KS101_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_KS101(0xe8,0x30);    //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 KS101 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
}

void config_0x71_0x7d(unsigned char addr_old, unsigned char flag)
//flag will be 0x71,0x72,0x73,0x74,0x7a,0x7b,0x7c,0x7d
{                                                  //that you want change to
   delayms(2000);                                   // Protect the eeprom ,you can delete this sentence
   write_byte(addr_old,2,0x9c);
   delayms(1);
   write_byte(addr_old,2,0x95);
   delayms(1);
   write_byte(addr_old,2,0x98);
   delayms(1);
   write_byte(addr_old,2, flag);
   delayms(500);                                  //Protect the eeprom, you can delete this sentence
}

unsigned int detect(unsigned char address,unsigned char command)   //0xe8(address) + 0x30(command)
{
     unsigned int distance,count;
     write_byte(address,2,command);                    //use command "0x30" 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(0xe8,0x30);
            //0xe8 is the address; 0x30 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 KS101 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 KS101_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 KS101_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)
	 {
                KS101_WriteOneByte(0XE8,0X02,0x30);
                delay_ms(80);
                range = KS101_ReadOneByte(0xe8, 0x02);
                range <<= 8;
                range += KS101_ReadOneByte(0xe8, 0x03);
	 }
}
```