KS136 技术说明书

版本：Ver. 1.00

创新技术源自导向技术

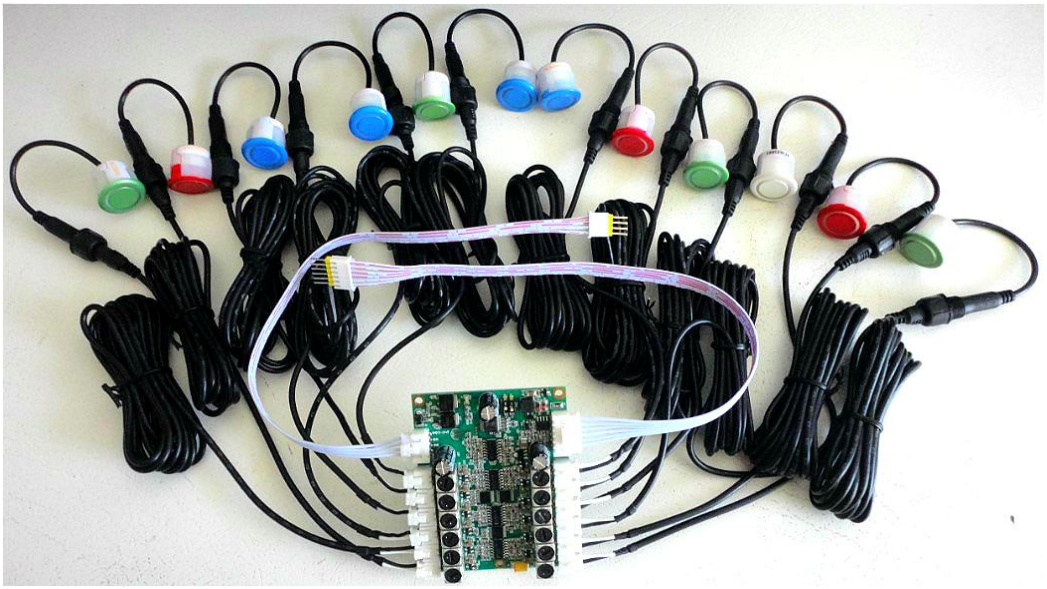
深圳市导向机电技术有限公司

Dauxi Technologies Co., Ltd. All rights reserved.

Modify Date	Content	Edit	Revision	Note
Nov. 4, 2016	Initial release.	*X.Q.*	1.00	
Jan. 16, 2017	Instruction detail description of command “0x01~0x06”	*X.Q.*	1.01	

：

：



KS136 功能摘要：

- **收发一体**式设计，1 个主控板接 12 个收发一体式防水探头，每个探头独立工作；
- 防水探头颜色**红**色，白色，**蓝**色，**绿**色，**黑**色，**黄**色可选，其他颜色可订制；
- 探测范围-单探头：**13cm-450cm**，探测范围-双探头：**1cm-450cm**。测人可到 **150cm**；
- 探测频率可达 50Hz，即每秒可探测 50 次；
- 支持 ***I²C/TTL*** 串口接口，兼容 KS103 协议；支持 ***485*** 接口；
- 共 20 个可修改的 ***I²C/TTL/485*** 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；***TTL/485*** 串口通信速率默认 9600bps；用户可修改为 115200bps 等；
- 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

KS136 电性能参数：

I^2C/TTL 工作电压：***3.0V~5.5V；***或 ***12V~24V*** 直流电源(推荐 ***12V***)。

使用 ***485*** 接口工作电压：***4.5V~5.5V；***或 ***12V~24V*** 直流电源(推荐 ***12V***)。

使用 ***24V*** 电压时请注意通风散热。

注意：已连 ***3.0V~5.5V*** 则建议 ***12V~24V*** 电源断开；已连 ***12V~24V*** 则建议 ***3.0V~5.5V*** 电源断开。

工作时瞬间最大电流：***200mA@5.0V, typical***。

工作电流：***100mA@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。

备注：使用电压为 ***3.0V~5.5V 时，***请确保电源是电池直接供电或线性稳压芯片输出的电源；或者建议电源噪音 VPP < 120mV。 因 KS136 尺寸受限，第 1 号探头离 ***3.0V~5.5V*** 电源较近，如果电源品质存在 120mV 以上噪音，可能出现第 1 号探头测超过 1200mm 距离时受干扰的可能。因此需要对 VPP > 120mV 的 ***3.0V~5.5V*** 加以留意。当无法提供符合基本要求（VPP < 120mV）的 ***3.0V~5.5V*** 电源时，建议使用 ***12V*** 电源供电以确保可靠性；或者第 1 号探头位悬空（如果使用 11 路或 11 路以下探头时）。

接线及模式切换说明

如下图 1 所示，在 KS136 上连线引脚上标识有：VCC(3-5.5V)、SDA/TX(简称 SDA)、SCL/RX(简称 SCL)、GND、485B、485A；CON1；CON2；CON3；CON4；CON5；CON6；CON7；CON8；CON9；CON10；CON11；CON12； VIN(12-24V)、GND、GND、VIN(12-24V)。

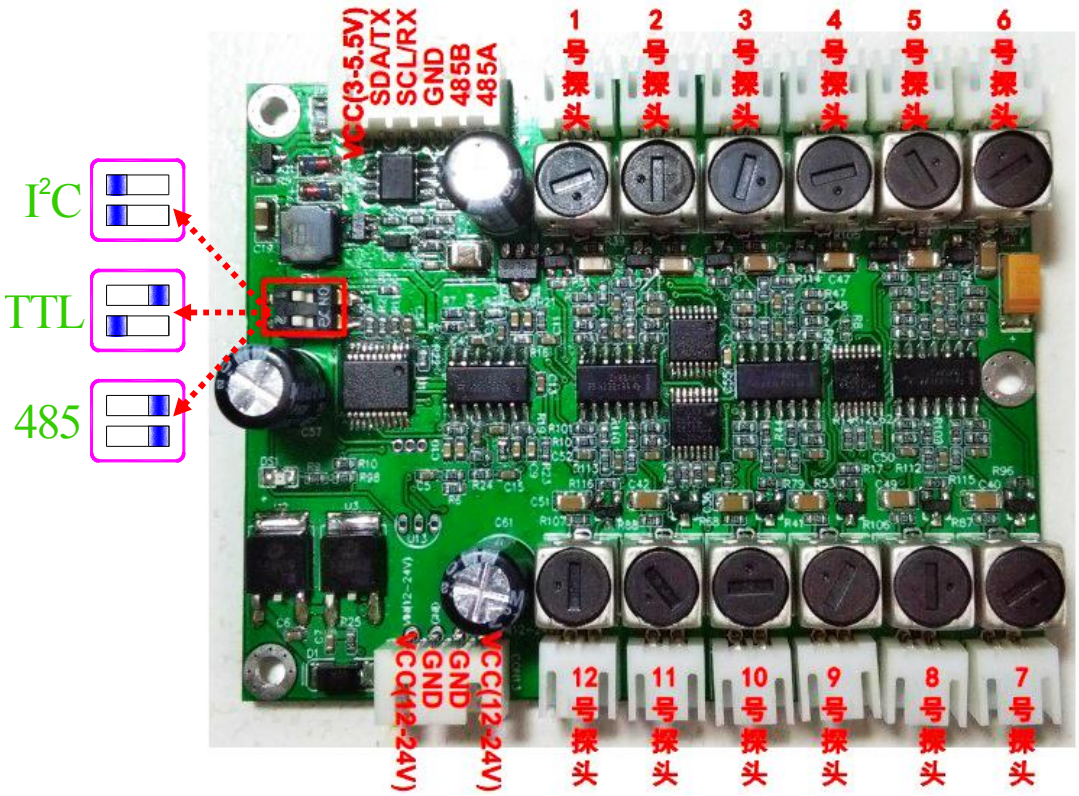


图 1

如图 1 所示，拨码开关 SW1 全左为 I²C 模式；拨码开关 SW1 上右下左为 TTL 串口模式；拨码开关 SW1 全右为 485 模式。

I²C 模式信号线接法为：SCL/RX 接上位机的 SCL；SDA/TX 接上位机的 SDA。

TTL 串口模式信号线接法为：SDA/TX 接上位机的 RXD;SCL/RX 接上位机的 TXD(1)。

Note (1):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。

485 串口模式时信号线接法为：485A 接 485A;485B 接 485B(2)。

Note (2):KS136-V101 的丝印采用了交叉丝印法，即在 485A 的位置印刷的是 485B，485B 的位置印刷的是 485A，接线时需注意。

以上三种模式的电源接法为：

方法 1：六脚插座的 VCC(3-5.5V)接电压范围为 3-5.5V 电源的正极，相邻 GND 接负极；

方法 2：四脚插座的 VCC(12-24V)接电压范围为 12-24V 电源的正极，相邻 GND 接负极。

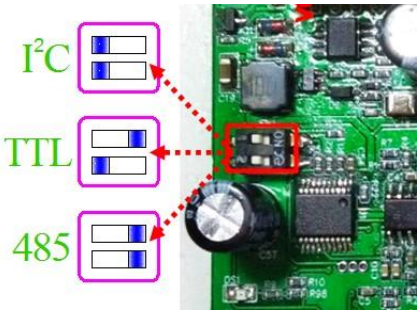
推荐使用 3-5.5V 电源(VPP<120mV)供电，如果只有 12-24V 电源，推荐使用 12V 电源。如果使用 24V 电源请注意通风及确保良好散热。KS136 为了追求高电源品质高抗干扰性，没有使用开关芯片 PWM 降压，而是采用了线性稳压器，以便获得更好的抗干扰体验。

接了 3-5.5V 电源则保持 12-24V 电源悬空；接了 12-24V 电源则保持 3-5.5V 电源悬空。

当使用 485 接口时，电源电压建议不要低于 4.5V。

以下分别详细介绍 I2C 模式、TTL 串口模式和 485 串口模式。

I²C 模式



KS136 连线：

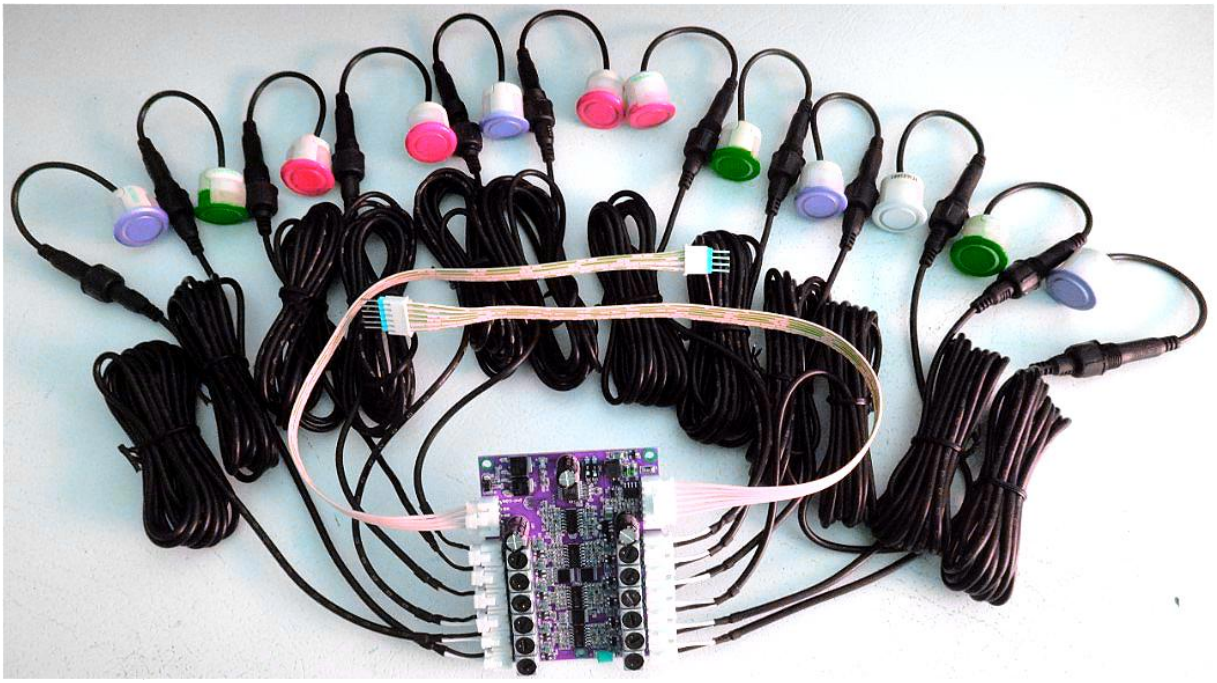


图 2

如图 2 所示，KS136 包括主控板一个，与主控板通过自锁 2PIN 插座连接（探头 1，探头 2，……，探头 12 依次对应自锁 2PIN 插座的 CON1，CON2，…，CON12）的 2.5 米线 12 根(其他长度可订制)，与 2.5 米线以防水螺纹连接的防水探头 12 个。在 KS136 上的自锁 6PIN 插座为外接数据及电源接口，依次引脚上标识有：VCC(3-5.5V)、SDA/TX(简称 SDA)、SCL/RX(简称 SCL)、GND、485B、485A；另一扩展 12-24V 电源接口依次引脚为：VIN(12-24V)、GND、GND、VIN(12-24V)。

6 脚插座中 VCC 用于连接+5V(3.0～5.5V 范围均可)电源(3)，SDA/TX 是 I²C 通信的数据线，SCL/RX 引脚是 I²C 通信的时钟线，GND 用于连接电源地。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。KS136 的 I²C 通信速率建议不要高于 100kbit/s。

Note (3): 在 485 通讯模式时建议使用 4.5~5.5V 电源。电压输入增加了防反接保护。如果指示灯不亮，请检查电源线是否接反。

4 脚插座中两个 VIN(12-24V)实际上内部是短接的，两个 GND 内部也是短接的。因此测试时可以只接其中一脚。量产时建议全接以减小接触电阻。VIN(12-24V)电压建议使用 12V。如果使用 24V 电源请注意通风及确保良好散热。KS136 为了追求高电源品质高抗干扰性，没有使用开关芯片 PWM 降压，而是采用了线性稳压器，以便获得更好的抗干扰效果。

探头与线之间连接如下图 3 所示，采用防水螺纹连接。插上后请注意旋紧以达防水需要。

图 3

I²C 模式具体连线如下图 4 所示（最多 20 个）：

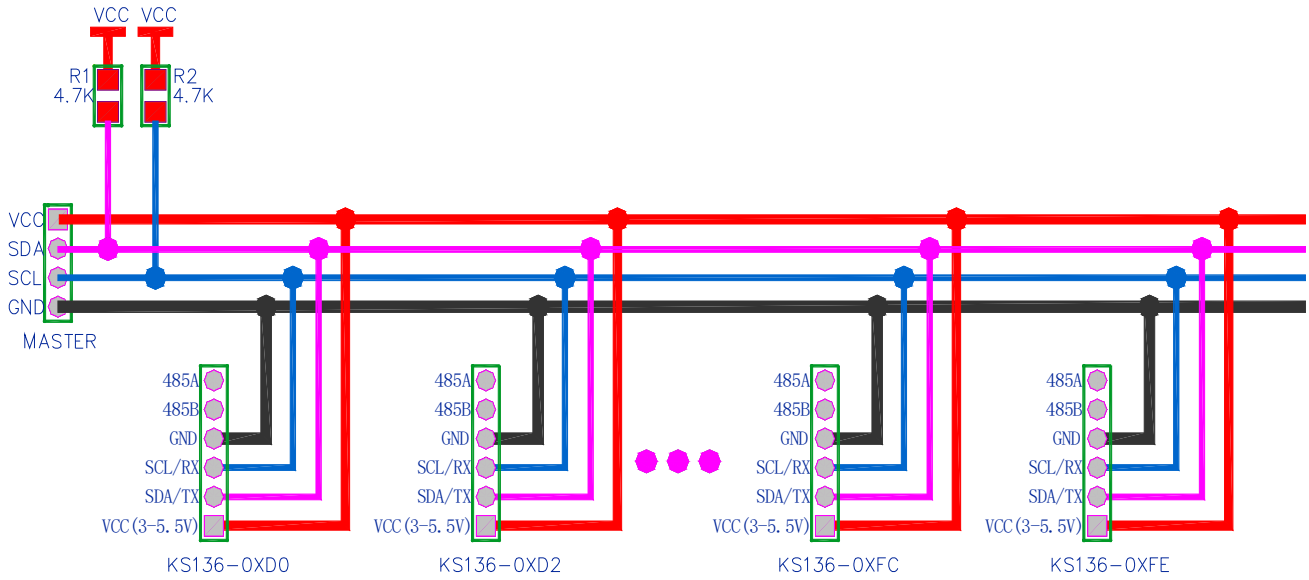


图 4

KS136 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. (4)

Note (4): 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 KS136 重新上电，可观察到 LED 显示新地址。在修改 KS136 的 I²C 地址过程中，严禁突然给 KS136 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 KS136。主机在

对其中一个 KS136 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。

KS136 工作流程：

在 KS136 上电启动时，系统会首先开始自检，自检需要约 1200ms。在此自检过程中，KS136 将会检测各路探头是否有正常插上，检测各配置是否正常。有异常会自动将探头故障位置上报。上报错误代码详情可参考表 1。I^2C 模式时可以通过读相应寄存器值获得错误代码。

自检完毕后图 5 所示 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，绿色 LED **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。(5)**

Note (5): 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。

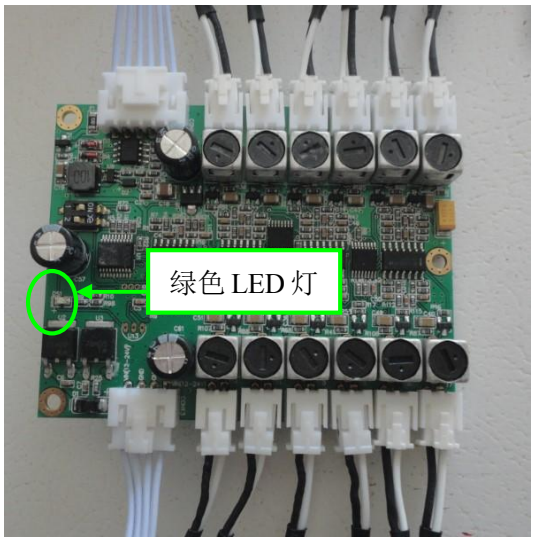
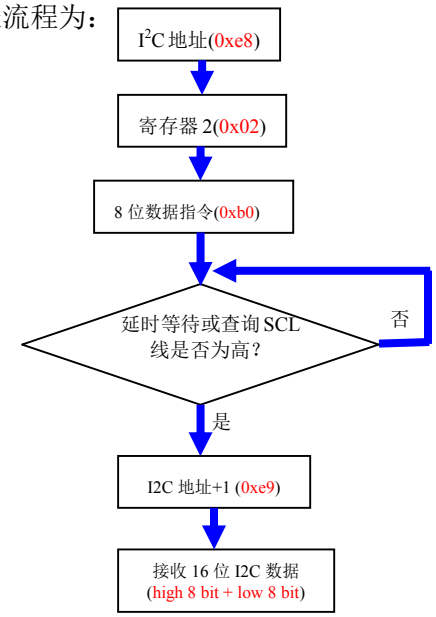


图 5

KS136 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。

KS136 使用 I^2C 接口与主机通信，自动响应主机的 I^2C 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

KS136 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。KS136 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 KS136 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 KS136 响应延迟。不同于 KS103，由于 KS136 采用了探测保护，每一次探测因此建议延时约 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。(6)

Note (6): 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 2m 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 KS136 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。KS136 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，KS136 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**指令及可读寄存器汇总如下：

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。 可参考附件 3 示例函数，返回值=read_byte(0xe8,0);
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；以此类推，10 月份标识为 A；12 月份对应 C。 可参考附件 3 示例函数，返回值=read_byte(0xe8,1);
2	0x01	34~32639us	0x22~0x7f7fus	探头 1 发射探头 12 接收，返回 us 值，是超声波从探头 1 发出到探头 12 接收到的飞行时间
2	0x02	34~32639us	0x22~0x7f7fus	探头 2 发射探头 11 接收，返回 us 值，是超声波从探头 2 发出到探头 11 接收到的飞行时间
2	0x03	34~32639us	0x22~0x7f7fus	探头 3 发射探头 10 接收，返回 us 值，是超声波

				从探头 3 发出到探头 10 接收到的飞行时间
2	0x04	34~32639us	0x22~0x7f7fus	探头 4 发射探头 9 接收，返回 us 值，是超声波从探头 4 发出到探头 9 接收到的飞行时间
2	0x05	34~32639us	0x22~0x7f7fus	探头 5 发射探头 8 接收，返回 us 值，是超声波从探头 5 发出到探头 8 接收到的飞行时间
2	0x06	34~32639us	0x22~0x7f7fus	探头 6 发射探头 7 接收，返回 us 值，是超声波从探头 6 发出到探头 7 接收到的飞行时间
2	**0x10**	**138~5267mm**	**0x88~0x1493mm**	**探头 1 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值**
2	**0x12**	**800~32639us**	**0x320~0x7f7fus**	**探头 1 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值**
2	**0x14**	**138~3015mm**	**0x88~0xbc7mm**	**探头 1 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值**
2	**0x16**	**182~5267mm**	**0xb6~0x1493mm**	**探头 1 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值**
2	**0x17**	**1055~65278us**	**0x41f~0x7f7fus**	**探头 1 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值**
2	**0x18**	**138~5267mm**	**0x88~0x1493mm**	**探头 2 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值**
2	**0x1a**	**800~32639us**	**0x320~0x7f7fus**	**探头 2 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值**
2	**0x1c**	**138~3015mm**	**0x88~0xbc7mm**	**探头 2 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值**
2	**0x1e**	**182~5267mm**	**0xb6~0x1493mm**	**探头 2 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值**
2	**0x1f**	**1055~65278us**	**0x41f~0x7f7fus**	**探头 2 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值**
2	**0x20**	**138~5267mm**	**0x88~0x1493mm**	**探头 3 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值**
2	**0x22**	**800~32639us**	**0x320~0x7f7fus**	**探头 3 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值**
2	**0x24**	**138~3015mm**	**0x88~0xbc7mm**	**探头 3 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值**
2	**0x26**	**182~5267mm**	**0xb6~0x1493mm**	**探头 3 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值**
2	**0x27**	**1055~65278us**	**0x41f~0x7f7fus**	**探头 3 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值**
2	**0x28**	**138~5267mm**	**0x88~0x1493mm**	**探头 4 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值**
2	**0x2a**	**800~32639us**	**0x320~0x7f7fus**	**探头 4 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值**
2	**0x2c**	**138~3015mm**	**0x88~0xbc7mm**	**探头 4 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值**
2	**0x2e**	**182~5267mm**	**0xb6~0x1493mm**	**探头 4 收发一体独立工作，有效探测范围**

				20cm~5.5m，返回 mm 值
2	0x2f	1055~65278us	0x41f~0x7f7fus	探头 4 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值
2	0x30	138~5267mm	0x88~0x1493mm	探头 5 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值
2	0x32	800~32639us	0x320~0x7f7fus	探头 5 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值
2	0x34	138~3015mm	0x88~0xbc7mm	探头 5 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值
2	0x36	182~5267mm	0xb6~0x1493mm	探头 5 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值
2	0x37	1055~65278us	0x41f~0x7f7fus	探头 5 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值
2	0x38	138~5267mm	0x88~0x1493mm	探头 6 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值
2	0x3a	800~32639us	0x320~0x7f7fus	探头 6 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值
2	0x3c	138~3015mm	0x88~0xbc7mm	探头 6 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值
2	0x3e	182~5267mm	0xb6~0x1493mm	探头 6 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值
2	0x3f	1055~65278us	0x41f~0x7f7fus	探头 6 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值
2	0x40	138~5267mm	0x88~0x1493mm	探头 7 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值
2	0x42	800~32639us	0x320~0x7f7fus	探头 7 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值
2	0x44	138~3015mm	0x88~0xbc7mm	探头 7 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值
2	0x46	182~5267mm	0xb6~0x1493mm	探头 7 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值
2	0x47	1055~65278us	0x41f~0x7f7fus	探头 7 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值
2	0x48	138~5267mm	0x88~0x1493mm	探头 8 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值
2	0x4a	800~32639us	0x320~0x7f7fus	探头 8 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值
2	0x4c	138~3015mm	0x88~0xbc7mm	探头 8 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值
2	0x4e	182~5267mm	0xb6~0x1493mm	探头 8 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值
2	0x4f	1055~65278us	0x41f~0x7f7fus	探头 8 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值
2	0x50	138~5267mm	0x88~0x1493mm	探头 9 收发一体独立工作，有效探测范围

				13cm~4.5m。返回 mm 值
2	0x52	800~32639us	0x320~0x7f7fus	探头 9 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值
2	0x54	138~3015mm	0x88~0xbc7mm	探头 9 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值
2	0x56	182~5267mm	0xb6~0x1493mm	探头 9 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值
2	0x57	1055~65278us	0x41f~0x7f7fus	探头 9 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值
2	0x58	138~5267mm	0x88~0x1493mm	探头 10 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值
2	0x5a	800~32639us	0x320~0x7f7fus	探头 10 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值
2	0x5c	138~3015mm	0x88~0xbc7mm	探头 10 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值
2	0x5e	182~5267mm	0xb6~0x1493mm	探头 10 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值
2	0x5f	1055~65278us	0x41f~0x7f7fus	探头 10 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值
2	0x60	138~5267mm	0x88~0x1493mm	探头 11 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值
2	0x62	800~32639us	0x320~0x7f7fus	探头 11 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值
2	0x64	138~3015mm	0x88~0xbc7mm	探头 11 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值
2	0x66	182~5267mm	0xb6~0x1493mm	探头 11 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值
2	0x67	1055~65278us	0x41f~0x7f7fus	探头 11 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值
2	0x68	138~5267mm	0x88~0x1493mm	探头 12 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值
2	0x6a	800~32639us	0x320~0x7f7fus	探头 12 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值
2	0x6c	138~3015mm	0x88~0xbc7mm	探头 12 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值
2	0x6e	182~5267mm	0xb6~0x1493mm	探头 12 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值
2	0x6f	1055~65278us	0x41f~0x7f7fus	探头 12 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值
2	0x70	无	无	第一级降噪。 所有指令指令将工作于第一级降噪。适用于电池供电
2	0x71	无	无	第二级降噪。 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	第三级降噪。 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	第四级降噪。 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	第五级降噪。 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	0x75	无	无	第六级降噪。 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	竖直方向上约 65° 波束角，水平方向上 120° 波束角(探头背后的“UP 箭头”朝向竖直向上)
2	0x7b	无	无	出厂默认设置，竖直方向上约 60° 波束角，水平方向上 115° 波束角(探头背后的“UP 箭头”朝向竖直向上)
2	0x7c	无	无	竖直方向上约 55° 波束角，水平方向上 110° 波束角(探头背后的“UP 箭头”朝向竖直向上)
2	0x7d	无	无	竖直方向上约 50° 波束角，水平方向上 105° 波束角(探头背后的“UP 箭头”朝向竖直向上)
2	0x7e	无	无	竖直方向上约 50° 波束角，水平方向上 100° 波束角(探头背后的“UP 箭头”朝向竖直向上)
2	0x95	无	无	0x70~0x7f 参数配置第二时序
2	0x98	无	无	0x70~0x7f 参数配置第三时序
2	0x9c	无	无	0x70~0x7f 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
2		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。 必须在发送完***地址+寄存器 2+探测指令***后方可查询本寄存器返回值。 可参考附件 3 示例函数，返回值=read_byte(0xe8,2);

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。 必须在发送完***地址+寄存器 2+探测指令***后方可查询本寄存器返回值。 可参考附件 3 示例函数，返回值=read_byte(0xe8,3);
4			0x76~0x79	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。 可参考附件 3 示例函数，返回值=read_byte(0xe8,4);
5			0xd0~0xfe	本寄存器存储的是 20 个 I^2C 或串口地址，不包括 0xf0, 0xf2, 0xf4,0xf6，供查询用。 可参考附件 3 示例函数，返回值=read_byte(0xe8,5);
6			0x70~0x75	本寄存器存储的是降噪级别 0x70~0x75，供查询用。 可参考附件 3 示例函数，返回值=read_byte(0xe8,6);
7			0x7a~0x7e	本寄存器存储的是波束角大小 0x7a~0x7e，供查询用。 可参考附件 3 示例函数，返回值=read_byte(0xe8,7);
8			0xe0	初始化正常。 可参考附件 3 示例函数，返回值=read_byte(0xe8,8);
			0xe1	硬件错误。
9			0xcf	初始化进行中
			0	初始化结束标志。 可参考附件 3 示例函数，返回值=read_byte(0xe8,9);
10		0~15	0~0x0f	本寄存器储存的是 12, 11, 10, 9 四个探头工作状态。本寄存器转为二进制数为 0B0000XXXX，自左向右依次存放 12, 11, 10, 9 四个探头工作状态;“1”代表正常，“0”代表未接探头或不正常。 可参考附件 3 示例函数，返回值=read_byte(0xe8,10);
11		0~255	0~0xff	本寄存器储存的是 8, 7, 6, 5, 4, 3, 2, 1 八个探头工作状态。本寄存器转为二进制数为 0B XXXX XXXX，自左向右依次存放 8, 7, 6, 5, 4, 3, 2, 1 四个探头工作状态；“1”代表正常，“0”代表未接探头或不正常。 可参考附件 3 示例函数，返回值=read_byte(0xe8,11);
12		0~12	0~0x0c	本寄存器储存的是上电后 KS136 实际正常工作的探头数量。 可参考附件 3 示例函数，返回值=read_byte(0xe8,12);
13~36				保留供升级用

表 1

距离探测-单探头模式

单探头模式探测范围 13cm-4.5 米，具体参数及控制指令请参见上表 1。

图 6

安装时注意探头的背面，即上图 6 所示，探头背面的箭头朝向竖直向上。探头箭头方向（即竖直方向）波束角出厂设置为约 60° 波束角，与探头箭头方向垂直的方向（即水平方向）115° 波束角。

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，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 us 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。

例如使用“0x30” 探测指令，将控制探头 1 独立探测；利用“0x38” 探测指令，将控制探头 2 独立探测；使用“0x40” 探测指令，将控制探头 3 独立探测；使用“0x48” 探测指令，将控制探头 4 独立探测。

距离探测-双探头模式

双探头模式可以对射式安装，也可以反射式安装。

双探头模式控制指令为 0x01-0x06，具体指令说明如下：

指令	返回值范围（10 进制）	返回值范围（16 进制）	说明
0x01	34~32639us	0x22~0x7f7fus	探头 1 发射探头 12 接收，返回 us 值，是超声波从探头 1 发出到探头 12 接收到的飞行时间
0x02	34~32639us	0x22~0x7f7fus	探头 2 发射探头 11 接收，返回 us 值，是超声波从探头 2 发出到探头 11 接收到的飞行时间
0x03	34~32639us	0x22~0x7f7fus	探头 3 发射探头 10 接收，返回 us 值，是超声波从探头 3 发出到探头 10 接收到的飞行时间
0x04	34~32639us	0x22~0x7f7fus	探头 4 发射探头 9 接收，返回 us 值，是超声波从探头 4 发出到探头 9 接收到的飞行时间
0x05	34~32639us	0x22~0x7f7fus	探头 5 发射探头 8 接收，返回 us 值，是超声波从探头 5 发出到探头 8 接收到的飞行时间
0x06	34~32639us	0x22~0x7f7fus	探头 6 发射探头 7 接收，返回 us 值，是超声波从探头 6 发出到探头 7 接收到的飞行时间

注意双探头探测返回的是 us 时间值。如果需要转换为 mm 距离值，请除以 5.8。双探头反射式安装时，安装图请参考如下：

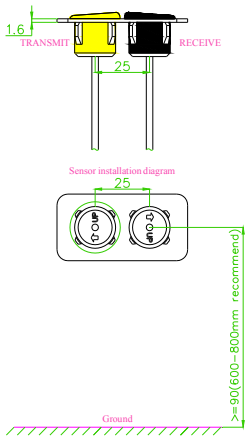


图 7

安装时在 1.6mm 厚度平板上留两个间距 25mm 直径 18.9mm 圆孔，探头按图按压装配到位。

以 0x01 指令，供电电压 12v 为例。如上图 7 所示，1 号探头在左，12 号探头在右，探头背面的箭头朝向如图 7 所示，即 1 号发射探头的箭头水平向左，12 号接收探头的箭头水平向右。这种方式可以实现和 KS103 近似的盲区和波束角。具体测试波束角图如下图 8 所示：

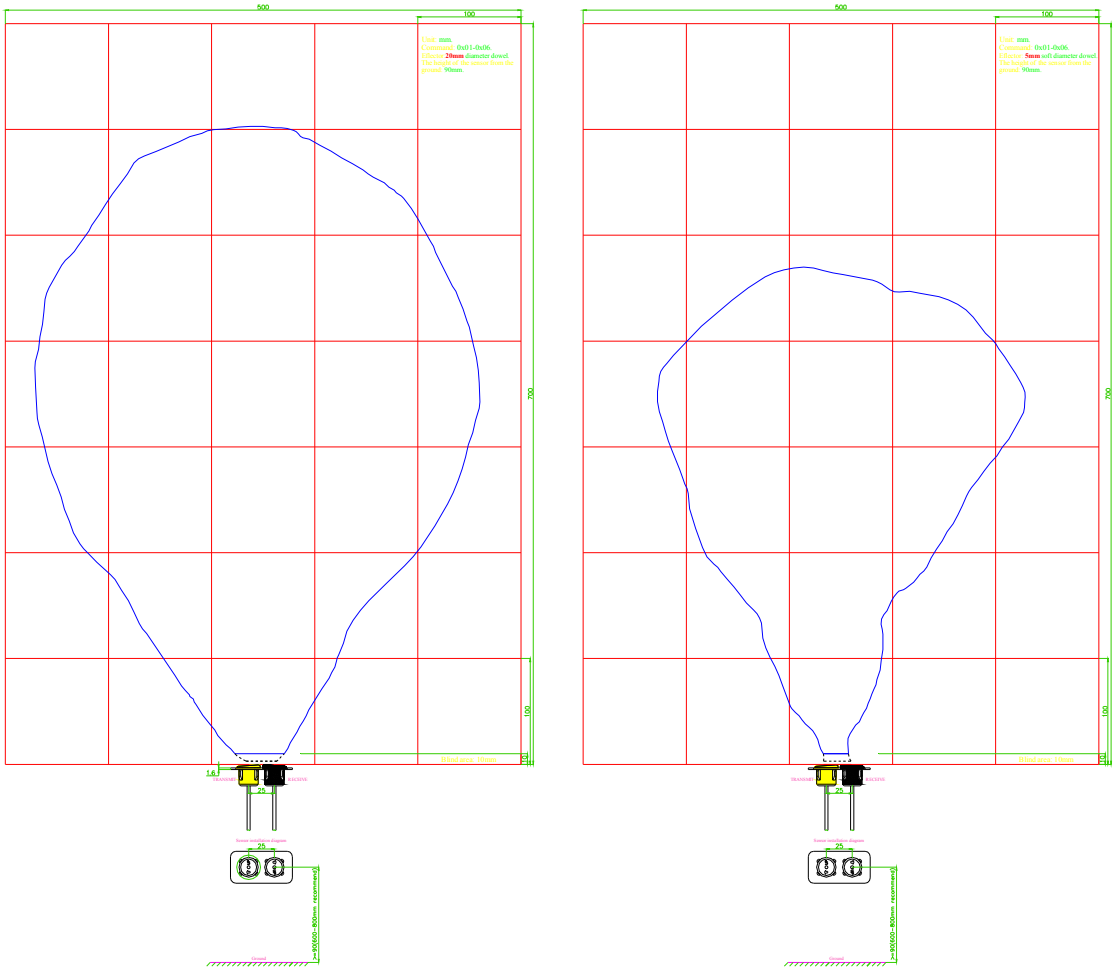


图 8

图 8 中两个图表右上角均标出了反射物的大小，当检测人时，最大可以检测到 1.5 米。

电源降噪指令(0x70, 0x71, 0x72, 0x73, 0x74, 0x75)**及不同物体探测配置指令**(0x7a-0x7e)

KS136 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 KS136 测距模块的杂波抑制功能。0x70 为出厂测试用级别，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，0x7e 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：“I^2C 地址 + 寄存器 2 + 0x9c; I^2C 地址 + 寄存器 2 + 0x95; I^2C 地址 + 寄存器 2 + 0x98; I^2C 地址 + 寄存器 2 + 0x70/0x71/0x72/0x73/0x74/0x75/0x7a/0x7b/0x7c/0x7d/0x7e”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 I^2C 地址为 0xe8

delayms(2000);

将本模块配置为最大波束角，配置代码如下：

config_0x71_0x7d(0xe8,0x7a); //如果 I^2C 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。KS136 收到有效配置指令之后，LED 灯将长亮 5s，表明配置成功。

KS136 在重新上电后将永久性按新配置运行。无须再次配置。

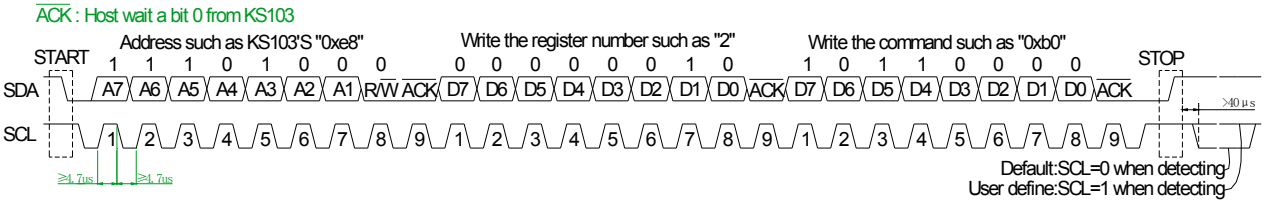
温度探测(KS136 暂不支持)

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℃。

时序图

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

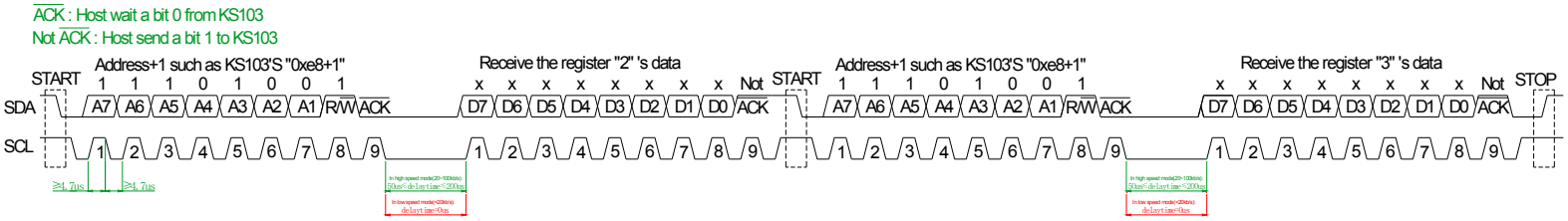
I^2C 地址	寄存器 2	8 位数据指令



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

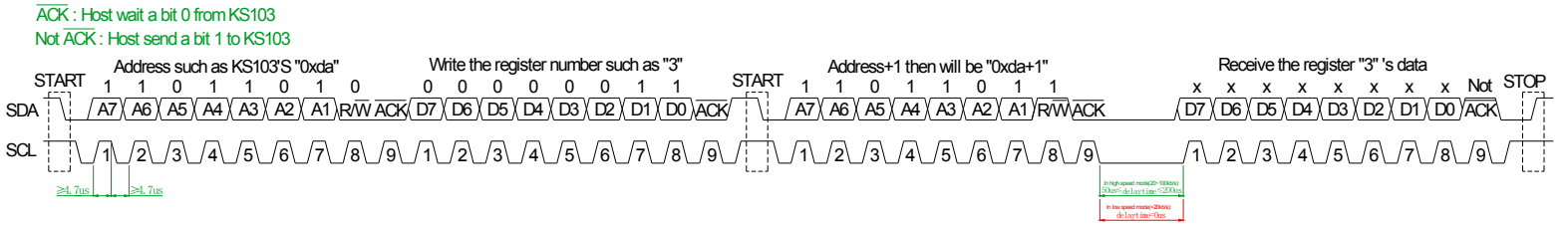
位，指令格式为：

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)：(7)

I²C 地址	寄存器 3	I²C 地址+1	读寄存器 3



Note (7): 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 KS136 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 KS136 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，KS136 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。

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。

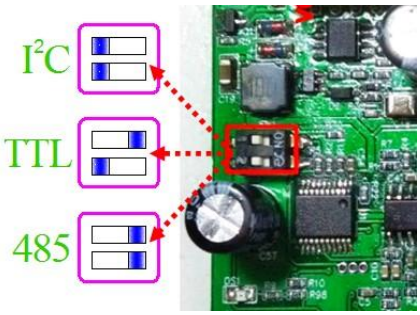
配置方法非常简单，向本模块发送指令时序：“I²C 地址 + 寄存器 2 +0xc4/0xc5”即可，发送完成后请延时至少 2 秒，以让系统自动完成配置。并开始按照新配置工作。

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 KS136 会自动保存，并立即按照新配置工作。KS136 在重新上电后将按新配置运行。

TTL 串口及 485 串口模式



KS136 的 TTL 或 485 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 可修改为 115200 等其他波特率。

KS136 连线：

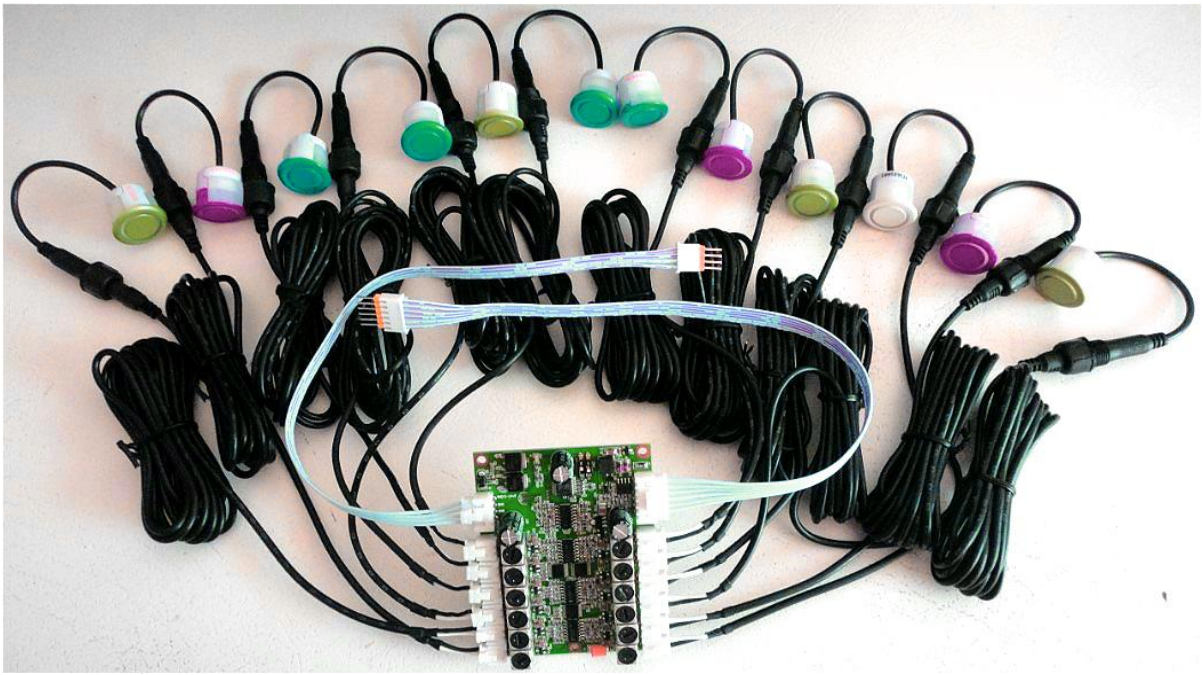


图 9

如图 9 所示，KS136 包括主控板一个，与主控板通过自锁 2PIN 插座连接（探头 1，探头 2，……，探头 12 依次对应自锁 2PIN 插座的 CON1，CON2，…，CON12）的 2.5 米线 12 根(其他长度可订制)，与 2.5 米线以防水螺纹连接的防水探头 12 个。在 KS136 上的自锁 6PIN 插座为外接数据及电源接口，依次引脚上标识有：VCC(3-5.5V)、SDA/TX(简称 SDA)、SCL/RX(简称 SCL)、GND、485B、485A；另一扩展 12-24V 电源接口依次引脚为：VIN(12-24V)、GND、GND、VIN(12-24V)。

4 脚插座中两个 VIN(12-24V)实际上内部是短接的，两个 GND 内部也是短接的。因此测试时可以只接其中一脚。量产时建议全接以减小接触电阻。VIN(12-24V)电压建议使用 12V。如果使用 24V 电源请注意通风及确保良好散热。KS136 为了追求高电源品质高抗干扰性，没有使用开关芯片 PWM 降压，而是采用了线性稳压器，以便获得更好的抗干扰效果。

探头与线之间连接如下图 10 所示，采用防水螺纹连接。插上后请注意旋紧以达防水需要。

图 10

TTL 串口模式连线依次为：六脚插座的 VCC(3-5.5V)接电压范围为 3-5.5V 电源的正极，相邻 GND 接负极；SDA/TX 接上位机的 RXD;SCL/RX 接上位机的 TXD。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。

TTL 串口模式具体连线如下图 11 所示（**最多接 2 个**）：

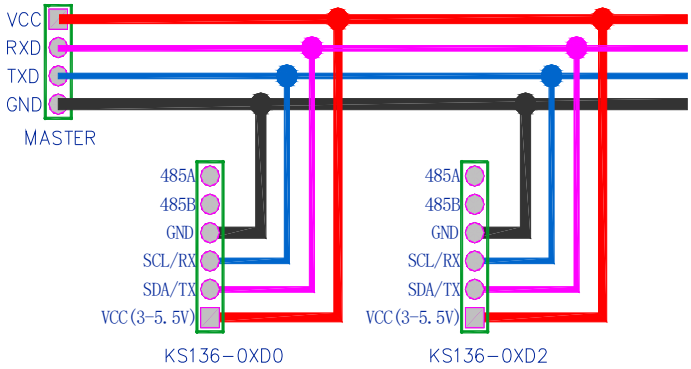


图 11

使用+5V 电源给 KS136 供电时，485 串口模式时信号线接法为：六脚插座的 VCC(3-5.5V)

接电压范围为 3-5.5V 电源的正极，相邻 GND 接负极；485A 接 485A;485B 接 485B。KS136-V101 的丝印采用了交叉丝印法，即在 485A 的位置印刷的是 485B，485B 的位置印刷的是 485A，接线时需注意。

使用+5V 电源 485 串口模式具体连线如下图 12 所示（**最多接 20 个**）：

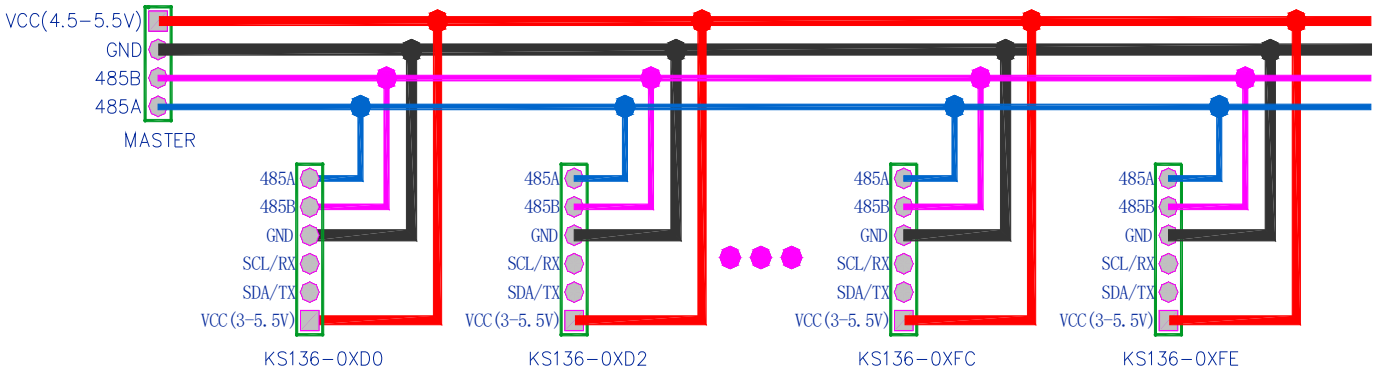


图 12

使用 12V 电源时，四脚插座的 VCC(12-24V)接电压范围为 12-24V 电源的正极，相邻 GND 接负极。

推荐使用 3-5.5V 电源，如果只有 12-24V 电源，推荐使用 12V 电源。如果使用 24V 电源请注意通风及确保良好散热。KS136 为了追求高电源品质高抗干扰性，没有使用开关芯片 PWM 降压，而是采用了线性稳压器，以便获得更好的抗干扰体验。

接了 3-5.5V 电源则保持 12-24V 电源悬空；接了 12-24V 电源则保持 3-5.5V 电源悬空。

当使用 485 接口时，电源电压建议不要低于 4.5V。在 485 通讯模式时建议使用 4.5~5.5V 电源。电压输入增加了防反接保护。如果指示灯不亮，请检查电源线是否接反。

使用 12V 电源供电时 485 串口模式具体连线如下图 13 所示（**最多接 20 个**）：

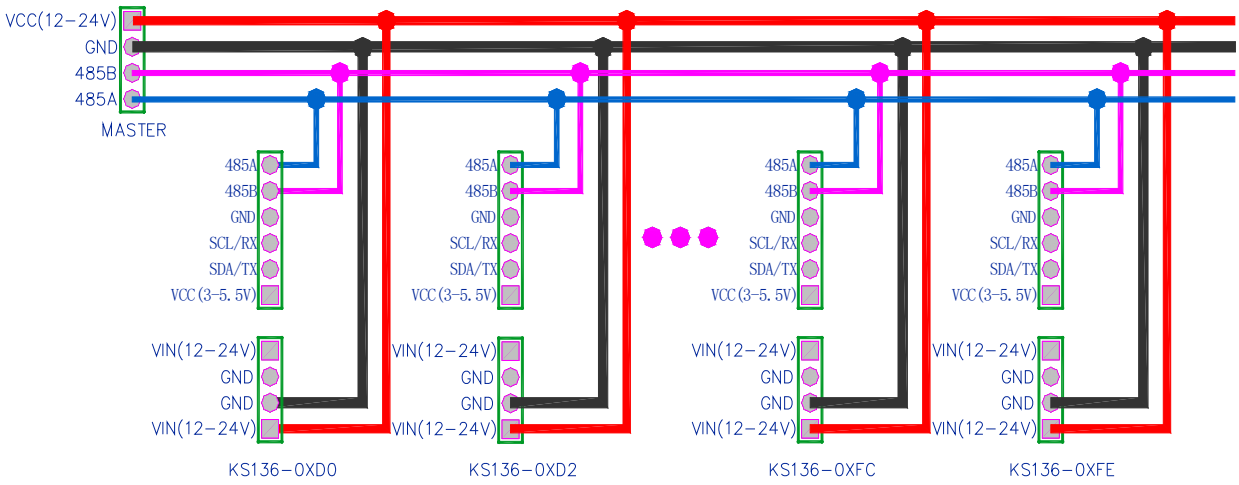


图 13

KS136 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. **(8)**

Note (8): 请注意，以上地址不包括 0xf0, 0xf2, 0xf4,0xf6，其与 I^2C 版地址完全一致。此外，TTL 串口协议规定一对一，因此建议使用 TTL 串口模式时，TTL 串口总线上最好只备有 1 台 KS136，**最多请不要超过 2 台**。使用 485 串口模式时，**485 串口总线上则可以接最多 20 台 KS136**。

485 串口与 TTL 串口除了接线上有所区别，控制代码上是完全一致的，以下关于“串口”的描述等同于“485 串口或 TTL 串口”。

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 KS136 重新上电，可观察到 LED 显示新地址。在修改 KS136 的串口地址过程中，严禁突然给 KS136 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 KS136（485 模式）；或同时连接 2 个 KS136（TTL 模式）。主机在对其中一个 KS136 模块进行控制时，其他模块不会受到影响。

KS136 工作流程：

在 KS136 上电启动时，系统会首先开始自检，自检需要约 1200ms。在此自检过程中，KS136 将会检测各路探头是否有正常插上，检测各配置是否正常。有异常会自动将探头故障位置上报。

初始化完毕 KS136 将通过串口自动向上位机发送如下十六进制代码：

65 6B 77 E8 71 7B E0 00 0F FF 0C 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，

0x**65**：程序版本，存储在寄存器 0 中；

0x**6B**：制造日期标识存储在寄存器 1 中；

0x**77**：串口通信波特率，存储在寄存器 4 中；

0x**E8**：I2C 或串口地址，存储在寄存器 5 中；

0x**71**：为降噪级别，存储在寄存器 6 中；

0x**7B**：为波束角大小，存储在寄存器 7 中；

0x**E0**：错误代码，存储在寄存器 8 中；

0x00：初始化结束标志，存储在寄存器 9 中；

0x**0F**：12~9 号探头自检标志位；正常该位为“1”，异常或未接为“0”。

0x**FF**：8~1 号探头自检标志位；正常该位为“1”，异常或未接为“0”。

0x**0C**：本寄存器储存的是上电后 KS136 实际正常工作的探头数量。

各十六进制值说明请参考如下表 2。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。

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；以此类推，10 月份标识为 A；12 月份对应 C。
4			0x76~0x79	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 I^2C 或串口地址，不包括 0xf0, 0xf2, 0xf4,0xf6，供查询用。
6			0x70~0x75	本寄存器存储的是降噪级别 0x70~0x75，供查询用。
7			0x7a~0x7e	本寄存器存储的是波束角大小 0x7a~0x7e，供查询用。
8			0xe0	初始化正常。
			0xe1	硬件错误。
9			0xcf	初始化进行中
			0	初始化结束标志。
10		0~15	0~0x0f	本寄存器储存的是 12, 11, 10, 9 四个探头工作状态。本寄存器转为二进制数为 0B0000XXXX，自左

				向右依次存放 12, 11, 10, 9 四个探头工作状态;“1”代表正常，“0”代表未接探头或不正常。
11		0~255	0~0xff	本寄存器储存的是 8, 7, 6, 5, 4, 3, 2, 1 八个探头工作状态。本寄存器转为二进制数为 0B XXXX XXXX，自左向右依次存放 8, 7, 6, 5, 4, 3, 2, 1 四个探头工作状态；“1”代表正常，“0”代表未接探头或不正常。
12		0~12	0~0x0c	本寄存器储存的是上电后 KS136 实际正常工作的探头数量。

表 2

自检初始化完毕后图 14 所示 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 **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。**(9)

Note (9): 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。

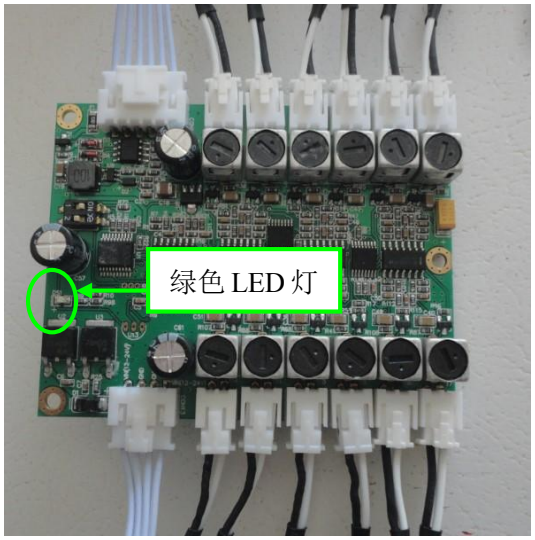


图 14

KS136 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。

KS136 使用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，指令发送及接收探测结果流程为：

串口地址(0xe8) —>延时 20～100us—> 寄存器(0x02) —>延时 20～100us—> 探测指令(0x30) —> 通过串口接收 KS136 的探测数据高 8 位 —> 接收 KS136 的探测数据低 8 位

KS136 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 KS136 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 KS136 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，KS136 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	34~32639us	0x22~0x7f7fus	探头 1 发射探头 12 接收，返回 us 值，是超声波从探头 1 发出到探头 12 接收到的飞行时间
2	0x02	34~32639us	0x22~0x7f7fus	探头 2 发射探头 11 接收，返回 us 值，是超声波从探头 2 发出到探头 11 接收到的飞行时间
2	0x03	34~32639us	0x22~0x7f7fus	探头 3 发射探头 10 接收，返回 us 值，是超声波从探头 3 发出到探头 10 接收到的飞行时间
2	0x04	34~32639us	0x22~0x7f7fus	探头 4 发射探头 9 接收，返回 us 值，是超声波从探头 4 发出到探头 9 接收到的飞行时间
2	0x05	34~32639us	0x22~0x7f7fus	探头 5 发射探头 8 接收，返回 us 值，是超声波从探头 5 发出到探头 8 接收到的飞行时间
2	0x06	34~32639us	0x22~0x7f7fus	探头 6 发射探头 7 接收，返回 us 值，是超声波从探头 6 发出到探头 7 接收到的飞行时间
2	**0x10**	**138~5267mm**	**0x88~0x1493mm**	**探头 1 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值**
2	**0x12**	**800~32639us**	**0x320~0x7f7fus**	**探头 1 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值**
2	**0x14**	**138~3015mm**	**0x88~0xbc7mm**	**探头 1 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值**
2	**0x16**	**182~5267mm**	**0xb6~0x1493mm**	**探头 1 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值**
2	**0x17**	**1055~65278us**	**0x41f~0x7f7fus**	**探头 1 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值**
2	**0x18**	**138~5267mm**	**0x88~0x1493mm**	**探头 2 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值**
2	**0x1a**	**800~32639us**	**0x320~0x7f7fus**	**探头 2 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值**
2	**0x1c**	**138~3015mm**	**0x88~0xbc7mm**	**探头 2 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值**
2	**0x1e**	**182~5267mm**	**0xb6~0x1493mm**	**探头 2 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值**
2	**0x1f**	**1055~65278us**	**0x41f~0x7f7fus**	**探头 2 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值**
2	**0x20**	**138~5267mm**	**0x88~0x1493mm**	**探头 3 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值**
2	**0x22**	**800~32639us**	**0x320~0x7f7fus**	**探头 3 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值**
2	**0x24**	**138~3015mm**	**0x88~0xbc7mm**	**探头 3 收发一体独立工作，3 米量程指令，有效探**

				测范围 13cm~3m。返回 mm 值
2	0x26	182~5267mm	0xb6~0x1493mm	探头 3 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值
2	0x27	1055~65278us	0x41f~0x7f7fus	探头 3 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值
2	0x28	138~5267mm	0x88~0x1493mm	探头 4 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值
2	0x2a	800~32639us	0x320~0x7f7fus	探头 4 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值
2	0x2c	138~3015mm	0x88~0xbc7mm	探头 4 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值
2	0x2e	182~5267mm	0xb6~0x1493mm	探头 4 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值
2	0x2f	1055~65278us	0x41f~0x7f7fus	探头 4 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值
2	0x30	138~5267mm	0x88~0x1493mm	探头 5 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值
2	0x32	800~32639us	0x320~0x7f7fus	探头 5 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值
2	0x34	138~3015mm	0x88~0xbc7mm	探头 5 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值
2	0x36	182~5267mm	0xb6~0x1493mm	探头 5 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值
2	0x37	1055~65278us	0x41f~0x7f7fus	探头 5 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值
2	0x38	138~5267mm	0x88~0x1493mm	探头 6 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值
2	0x3a	800~32639us	0x320~0x7f7fus	探头 6 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值
2	0x3c	138~3015mm	0x88~0xbc7mm	探头 6 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值
2	0x3e	182~5267mm	0xb6~0x1493mm	探头 6 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值
2	0x3f	1055~65278us	0x41f~0x7f7fus	探头 6 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值
2	0x40	138~5267mm	0x88~0x1493mm	探头 7 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值
2	0x42	800~32639us	0x320~0x7f7fus	探头 7 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值
2	0x44	138~3015mm	0x88~0xbc7mm	探头 7 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值
2	0x46	182~5267mm	0xb6~0x1493mm	探头 7 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值
2	0x47	1055~65278us	0x41f~0x7f7fus	探头 7 收发一体独立工作，有效探测范围

				20cm~5.5m，返回 us 值
2	0x48	138~5267mm	0x88~0x1493mm	探头 8 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值
2	0x4a	800~32639us	0x320~0x7f7fus	探头 8 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值
2	0x4c	138~3015mm	0x88~0xbc7mm	探头 8 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值
2	0x4e	182~5267mm	0xb6~0x1493mm	探头 8 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值
2	0x4f	1055~65278us	0x41f~0x7f7fus	探头 8 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值
2	0x50	138~5267mm	0x88~0x1493mm	探头 9 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值
2	0x52	800~32639us	0x320~0x7f7fus	探头 9 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值
2	0x54	138~3015mm	0x88~0xbc7mm	探头 9 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值
2	0x56	182~5267mm	0xb6~0x1493mm	探头 9 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值
2	0x57	1055~65278us	0x41f~0x7f7fus	探头 9 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值
2	0x58	138~5267mm	0x88~0x1493mm	探头 10 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值
2	0x5a	800~32639us	0x320~0x7f7fus	探头 10 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值
2	0x5c	138~3015mm	0x88~0xbc7mm	探头 10 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值
2	0x5e	182~5267mm	0xb6~0x1493mm	探头 10 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值
2	0x5f	1055~65278us	0x41f~0x7f7fus	探头 10 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值
2	0x60	138~5267mm	0x88~0x1493mm	探头 11 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值
2	0x62	800~32639us	0x320~0x7f7fus	探头 11 收发一体独立工作，有效探测范围 13cm~4.5m。返回 us 值
2	0x64	138~3015mm	0x88~0xbc7mm	探头 11 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值
2	0x66	182~5267mm	0xb6~0x1493mm	探头 11 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值
2	0x67	1055~65278us	0x41f~0x7f7fus	探头 11 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值
2	0x68	138~5267mm	0x88~0x1493mm	探头 12 收发一体独立工作，有效探测范围 13cm~4.5m。返回 mm 值
2	0x6a	800~32639us	0x320~0x7f7fus	探头 12 收发一体独立工作，有效探测范围

				13cm~4.5m。返回 us 值
2	0x6c	138~3015mm	0x88~0xbc7mm	探头 12 收发一体独立工作，3 米量程指令，有效探测范围 13cm~3m。返回 mm 值
2	0x6e	182~5267mm	0xb6~0x1493mm	探头 12 收发一体独立工作，有效探测范围 20cm~5.5m，返回 mm 值
2	0x6f	1055~65278us	0x41f~0x7f7fus	探头 12 收发一体独立工作，有效探测范围 20cm~5.5m，返回 us 值
2	0x70	无	无	第一级降噪。 所有指令指令将工作于第一级降噪。适用于电池供电
2	0x71	无	无	第二级降噪。 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	第三级降噪。 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	第四级降噪。 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	第五级降噪。 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	0x75	无	无	第六级降噪。 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	竖直方向上约 65° 波束角，水平方向上 120° 波束角(探头背后的“UP 箭头”朝向竖直向上)
2	0x7b	无	无	出厂默认设置，竖直方向上约 60° 波束角，水平方向上 115° 波束角(探头背后的“UP 箭头”朝向竖直向上)
2	0x7c	无	无	竖直方向上约 55° 波束角，水平方向上 110° 波束角(探头背后的“UP 箭头”朝向竖直向上)
2	0x7d	无	无	竖直方向上约 50° 波束角，水平方向上 105° 波束角(探头背后的“UP 箭头”朝向竖直向上)
2	0x7e	无	无	竖直方向上约 50° 波束角，水平方向上 100° 波束角(探头背后的“UP 箭头”朝向竖直向上)
2	0x95	无	无	0x70~0x7f 参数配置第二时序
2	0x98	无	无	0x70~0x7f 参数配置第三时序
2	0x9c	无	无	0x70~0x7f 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 3

距离探测-单探头模式

单探头模式探测范围 13cm-4.5 米，具体参数及控制指令请参见上表 3。

安装时注意探头的背面，即上图 6 所示，探头背面的箭头朝向竖直向上。探头箭头方向（即竖直方向）波束角出厂设置为约 60°波束角，与探头箭头方向垂直的方向（即水平方向）115°波束角。

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 us 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。

例如使用“0x30” 探测指令，将控制探头 1 独立探测；利用“0x38” 探测指令，将控制探头 2 独立探测；使用“0x40” 探测指令，将控制探头 3 独立探测；使用“0x48” 探测指令，将控制探头 4 独立探测。

例如使用“0x30” 探测指令，将控制探头 1 独立探测；利用“0x38” 探测指令，将控制探头 2 独立探测；使用“0x40” 探测指令，将控制探头 3 独立探测；使用“0x48” 探测指令，将控制探头 4 独立探测。

距离探测-双探头模式

双探头模式控制指令为 0x01-0x06.具体返回值如下：

指令	返回值范围（10 进制）	返回值范围（16 进制）	说明
0x01	34~32639us	0x22~0x7f7fus	探头 1 发射探头 12 接收，返回 us 值，是超声波从探头 1 发出到探头 12 接收到的飞行时间
0x02	34~32639us	0x22~0x7f7fus	探头 2 发射探头 11 接收，返回 us 值，是超声波从探头 2 发出到探头 11 接收到的飞行时间
0x03	34~32639us	0x22~0x7f7fus	探头 3 发射探头 10 接收，返回 us 值，是超声波从探头 3 发出到探头 10 接收到的飞行时间
0x04	34~32639us	0x22~0x7f7fus	探头 4 发射探头 9 接收，返回 us 值，是超声波从探头 4 发出到探头 9 接收到的飞行时间
0x05	34~32639us	0x22~0x7f7fus	探头 5 发射探头 8 接收，返回 us 值，是超声波从探头 5 发出到探头 8 接收到的飞行时间
0x06	34~32639us	0x22~0x7f7fus	探头 6 发射探头 7 接收，返回 us 值，是超声波从探头 6 发出到探头 7 接收到的飞行时间

注意双探头探测返回的是 us 时间值。如果需要转换为 mm 距离值，请除以 5.8。双探头反射式安装时，双探头安装图请参考图 7。

安装时在 1.6mm 厚度平板上留两个间距 25mm 直径 18.9mm 圆孔，探头按图按压装配到位。

以 0x01 指令，供电电压 12v 为例。如上图 7 所示，1 号探头在左，12 号探头在右，探头背面的箭头朝向如图 7 所示，即 1 号发射探头的箭头水平向左，12 号接收探头的箭头水平向右。这种方式可以实现和 KS103 近似的盲区和波束角。具体测试波束角图如图 8 所示：

图 8 中两个图表右上角均标出了反射物的大小，当检测人时，最大可以检测到 1.5 米。

电源降噪指令(0x70, 0x71, 0x72, 0x73, 0x74, 0x75)及不同物体探测配置指令(0x7a-0x7e)

KS136 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 KS136 测距模块的杂波抑制功能。0x70 为出厂测试用级别，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，0x7e 来根据实际障碍物情况选择指令。参见表 3.

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。KS136 收到有效配置指令之后，LED 灯将长亮 5s，表明配置成功。

KS136 在重新上电后将永久性按新配置运行。无须再次配置。

温度探测(KS136 暂不支持)

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℃。

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：

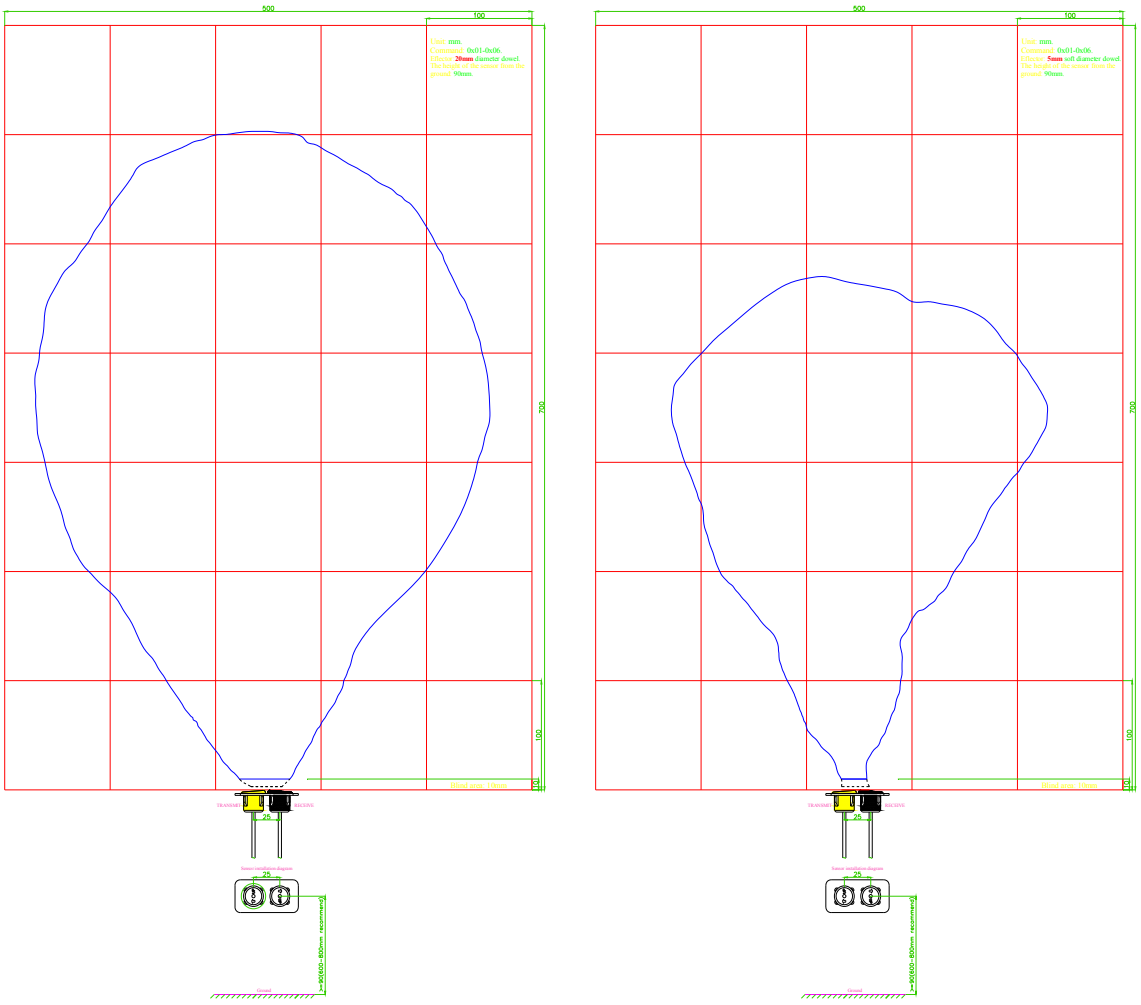
探测结果高 8 位	探测结果低 8 位

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。

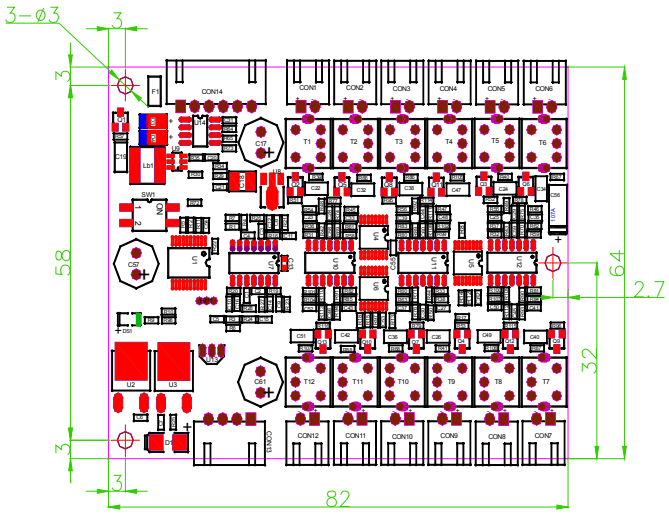
探测范围(40KHz 超声波)



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 5mm	PVC 管	90mm	5V	0x72
Fig.B	直径 20mm	304 不锈钢	90mm	5V	0x72

备注： KS136 发射探头在左，接收探头在右，两探头与地面平行；电压 5V 与 12V 的测试效果无明显区别。

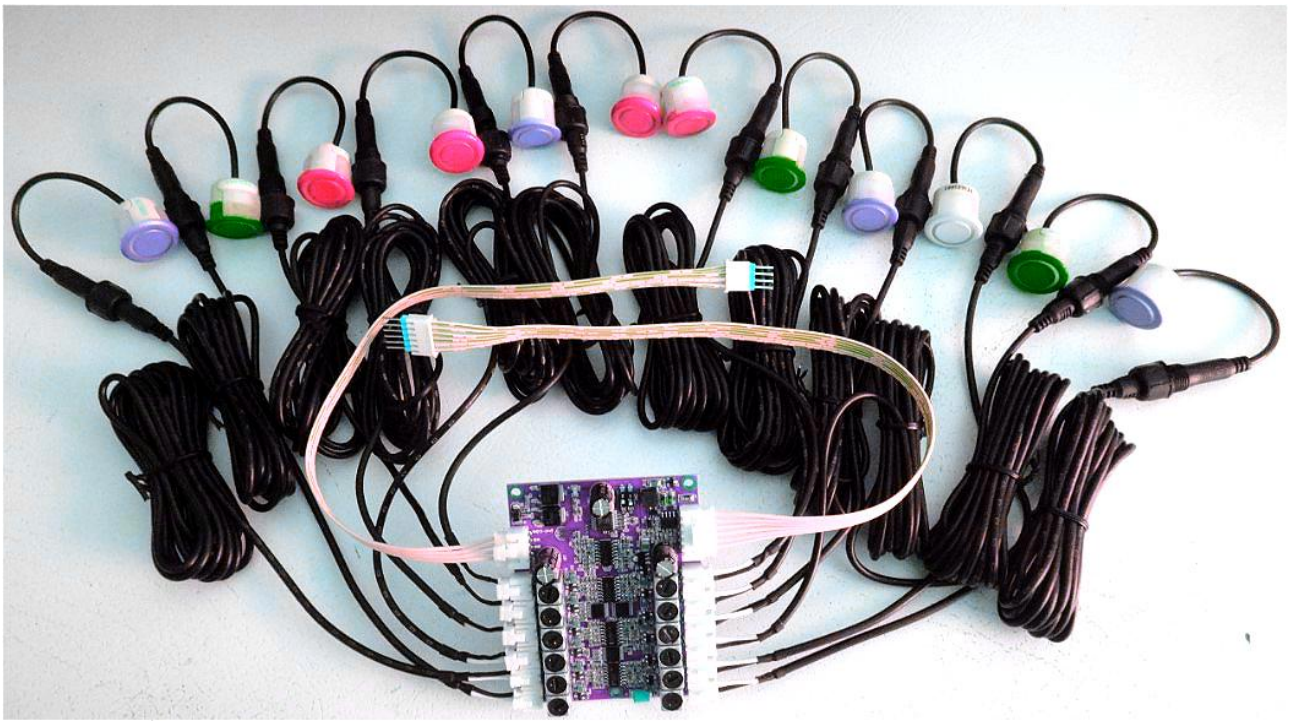
KS136 主控板安装尺寸(单位：毫米)：



KS136 探头安装方式：

探头为防水雷达探头，设计时在机器人表面预留直径18.8~18.9mm，厚度小于4mm圆孔即可。安装时PCB主控板固定在机器人内部，探头线长2500mm，探头由外向内插入18.8~18.9mm圆孔即可卡紧。

包装(完整 1 套 KS136 如下图)



发货清单如下：KS136 主控模块:1PCS/盒；2.5 米长探头线 4 根 3 包；探头 4PCS/盒共 3 盒。赠送 KS136 专用数据线。

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 KS136 连接控制 C 代码
2) PIC16F877A 主机采用模拟 I²C 通讯与 KS136 连接控制 C 代码
3) 51 单片机主机模拟 I²C 通讯与 KS136 连接控制 C 代码
4) STM32 CORTEX-3 ARM 主机模拟 I²C 通讯与 KS136 连接控制 C 代码

1) PIC16F877A 主机采用硬件 I²C 通讯与 KS136 连接控制 C 代码

/*电路连接方式：PIC16F877A 的 IO 口 SCL、SDA 与 KS136 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(0xe8,0x30);  //Address:0xe8; command:0x30.
                                              //Get detect result from KS136, 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 KS136
    while(SEN);                                               // wait for it to clear
    while(!SSPIF);                                            // wait for interrupt
    SSPIF = 0;                                                // then clear it.

    SSPBUF = addr_old;                                  // KS136'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 KS136 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;                              // KS136'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 KS136 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;                              // KS136'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 KS136 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;                              // KS136'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 KS136 to break to do something
}

unsigned int detect_KS101B(unsigned char ADDRESS, unsigned char command)
{
//ADDRESS will be KS136'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;                                       // KS136'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 KS136 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 KS136
        SEN = 1;                                                        // send start bit
        while(SEN);                                                     // and wait for it to clear
        ACKDT = 0;                                                      // acknowledge bit
        SSPIF = 0;

        SSPBUF = ADDRESS;                                       // KS136 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;             // KS136 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 KS136 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;                                 // KS136 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 KS136 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 + 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
  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(0xe8,0x30);    //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 KS136 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 KS136 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 KS136_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 KS136_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)
     {
                KS136_WriteOneByte(0XE8,0X02,0x30);
                delay_ms(80);
                range = KS136_ReadOneByte(0xe8, 0x02);
                range <<= 8;
                range += KS136_ReadOneByte(0xe8, 0x03);
     }
```

```
}
```