升级版 KQ-130F 电力载波数据收发模块

KQ-130F 是单列 9 针小体积高性能过零载波数据收发模块。是专门为在 220V 交流上，强干扰，强衰减，远距离要求的环境下，可靠的传送数据而特别设计和开发的性价比很高的载波模块。适用于抄表，路灯，智能家居，消防，楼宇控制以及需要电力线传送数据的其它应用领域。升级版 KQ-130F 增加了 1 个过零点传输 1 个字节方式以及内部带有 16 级深度的中继功能！

一、KQ-130F 系列性能：

1. 集成了 KQ-330F 模块及外围电路的载波板，毋需其他的耦合元件，直接连接 220V 的交流电使用。外型尺寸为 53×38×17 毫米(L×D×H)，单列排针引出（见下图）1、2 脚接 220V 交流电源无方向（1 脚,2 脚间距 2X0.1 英寸），2 脚,3 脚间距 1.1 英寸，其余各脚之间间距 0.1 英寸。

2. 工作频率 120～135KHZ，接口波特率 9600bps。实际波特率 1000bps/100bps，235 个字节缓存器。

3. 温度范围：-25℃～+70℃ 湿度≤90%

4. 供电电源：DC +5V　接收时：≤11mA　发送时：≤300mA

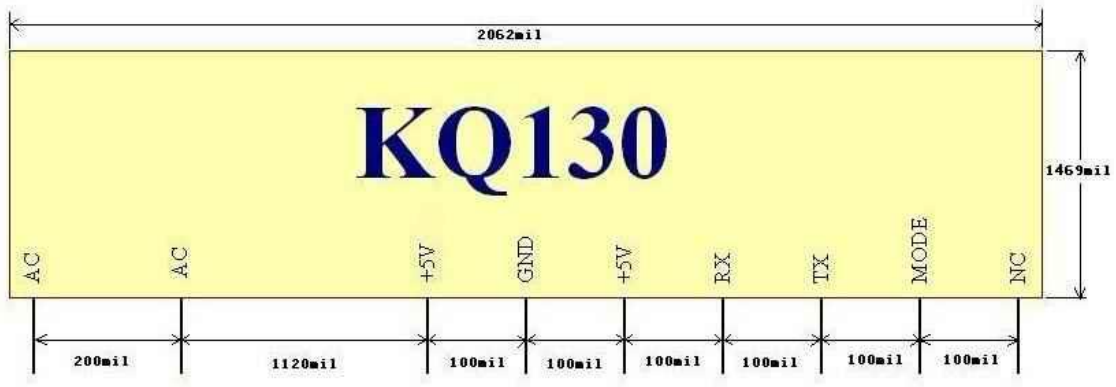
二、规格及型号：

KQ－130F：

130 后第一个字母定义为：

F：过零传送型

三、KQ-330 引脚说明　：　　正面从左至右为 1～9 脚：



1P—AC：220V 交流电压的火线（或零线）

2P—AC：220V 交流电压的零线（或火线）

3P—+5V：+5V 发送电源（260mA），如果单收数据可以悬空降低功耗

4P—GND：数字电路地线

5P—+5V：+5V 工作电源 11mA

6P—RX：TTL 电平，（5V 电压下高电平需要 3.5V 以上）载波数据入，接单片机的 TXD，高阻输入不能悬空

7P—TX：TTL 电平，载波数据出，接单片机的 RXD

8P—MODE：模式选择，悬空或接 5V 为高电平，接地为低电平。

高电平 1000bps　速度快（一个过零点传一个字节零点附近 2ms）

低电平 100bps 抗干扰更好（一个过零点传一个位零点附近 500us）

9P—RST ：复位脚（低电平有效）低脉冲复位模块。

四、KQ130F 系列模块编程注意事项

本模块接口波特率 9600bps，用户与模块通讯请采用 9600BPS 异步方式，格式为 1 个起始位，8 个数据位 1 个停止位格式。

本模块全部是按透明方式工作，MODE 脚控制模块过零传输方式：一个零点是 1 个字节（高电平），还是 1 个零点一个位（低电平）。MODE 高电平（悬空）时为字节高速方式，低电平（接地）时为低速一个位方式。由于一个过零点传输一个位，所以低速方式效果更好，传输距离更远，如果用户更要求单点传输距离可以采用这种方案。

编程时毋需对模块初始化，通讯时和普通 RS-485 方式类同。只是用户注意实际载波速率带来的延迟。整个模块收发过程，模块以 960bps0 收完用户数据（高电平时间持续 10ms 以上判断用户这包数据结束），启动载波发送，按照载波的高速或者低速（MODE 脚决定）发送出去，接收模块在收完整包数据后，解包把用户的原始数据以 9600bps 发送給接收方。所以整体传输时间延迟是 2 包 9600bps 加载波传输的时间。

注意：在模块发送缓存器（235 字节）满后不再接收新的数据。也就是一帧发送字节小于 235 个字节。用户的一帧数据请连续不间断的发送到模块，如果停顿时间超过 10ms 对方可能不能正确接收到数据。

如向 RX 端连续发送： 5A 5A 5A 34 56 78 12 45 67 85

接收模块连续输出： 5A 5A 5A 34 56 78 12 45 67 85

数据完全一致。

接收数据还是采用 9600BPS 异步方式，格式为 1 个起始位，8 个数据位 1 个停止位格式。

五、KQ130F 系列模块调试方法：

1.正确连接 KQ-130 的各个引脚接线,TX,RX 与单片机（用户系统）的 TXD,RXD 交叉连接。5V 电源由用户系统提供，GND 脚与用户系统的 GND 脚接在一起。MODE 根据用户实际使用选择。

2.可以连接电脑的 USB 转 TTL 模块，TXD,RXD 交叉连接模块，GND 与模块 GND 接好模块 5V 电源接上用户提供的 5V 电源

3.接口当地连接调试：模块 AC 必须悬空，MODE 必须悬空，按 1,2 步骤连接好模块。重新上电。

用串口助手测试模块，选择 9600，N,8,1，发送 AT 命令，注意发送 AT 命令必须换行。

换成 HEX 码发送如下：41 54 0D 0A

发送 AT 命令后返回

AT

OK

本地连接模块整个通讯，电源等成功!!

接好 AC 脚和发送电源就可以直接透明传输数据！

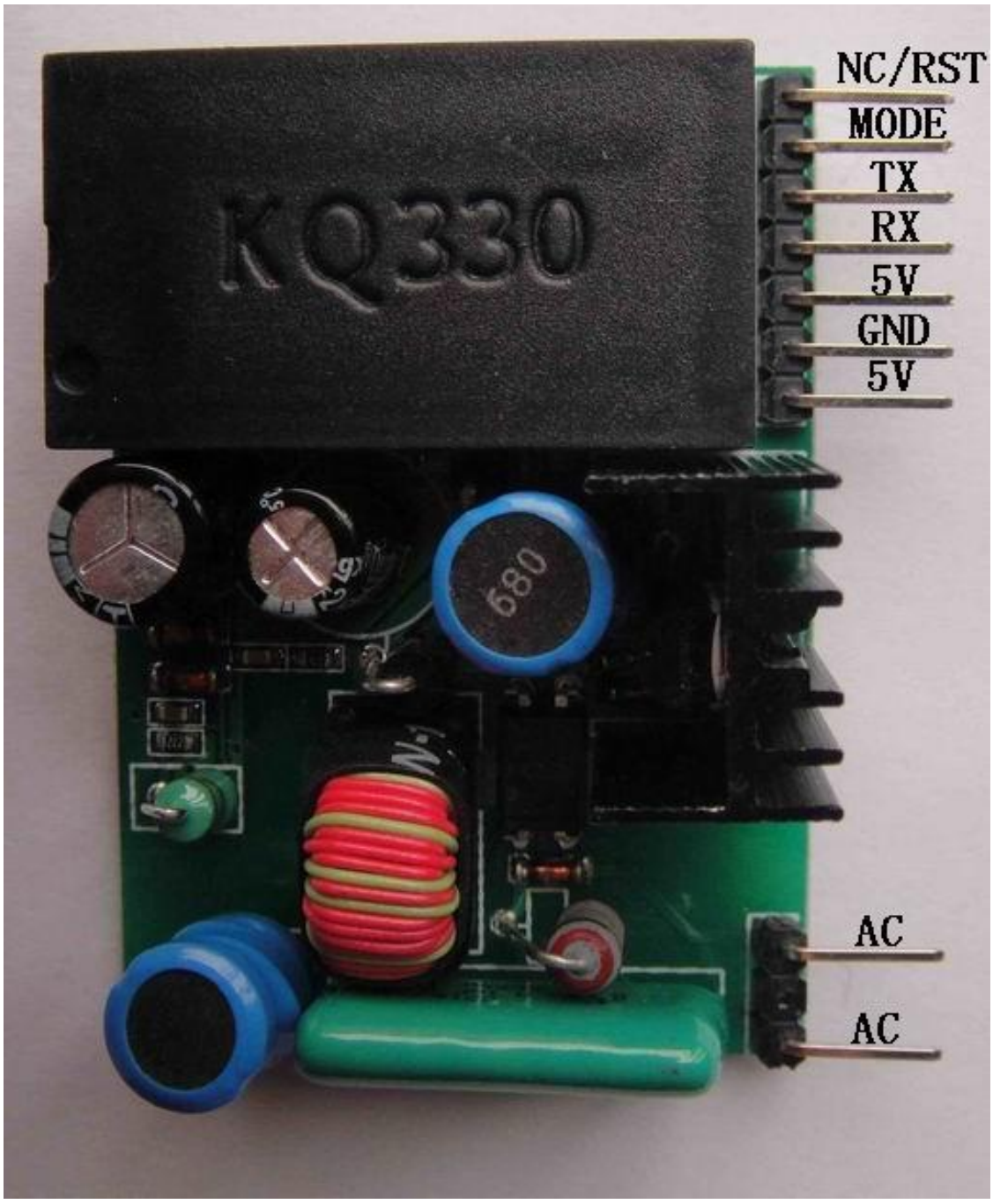
KQ-130 系列模块的区别：

KQ-130F 是专门针对交流 220V/110V，50HZ/60HZ 强干扰设计的基于交流零点传送方案的载波模块，在市电上面具有传送效果好，距离远等特点。必须在有交流电源的情况下才能传送数据，载波速率是 50HZ/100BPS，60HZ/120BPS 我们通过软件优化 9 个 BIT 可以传送 1 个字节。KQ-130E 是完全的载波模块，传送数据与零点无关。在载波解调后做了数字滤波，提高其载波数据的抗干扰能力，速率越低效果越好。KQ-130E 可以在 0V-220V 交直流电压下进行载波通讯，如：220V，110V，80V，48V，36V，24V，12V 等交直流电压以及停电情况下的载波通讯。KQ-130E 最高速率是 400BPS。可以选择 100BPS 通讯提高抗干扰能力，但是同样是 100BPS 的 KQ-100E 在交流电源上的通讯效果比 KQ-130F 差不少。KQ-130K 的通讯方式和 KQ-130E 相同，唯一的区别就是实际的载波速率最高可以做到 1200BPS。尾缀带＋号的，KQ-130F+，KQ-130E+，KQ-130K+，可以支持发送功率电压在 12V 的情况下连续的长时间发送数据。

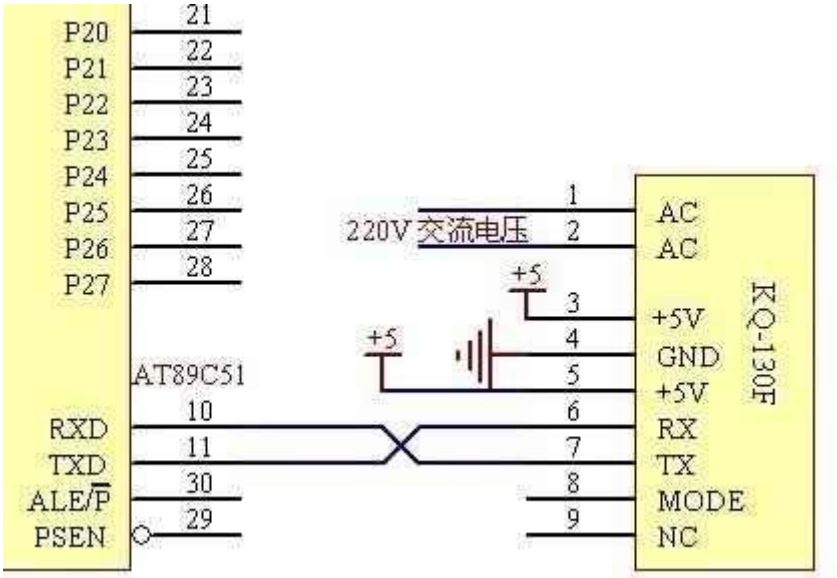
KQ-130 系列模块的共同点：

所有 KQ-130 系列模块与单片机/微机的接口都是一样的，通讯方式也相同。与单片机/微机的串行接口速率都是 9600BPS，一个起始位，八个数据位和一个停止位。相同的透明工作模式或自定义工作模式。

KQ-130F 电力载波数据收发模块图片：

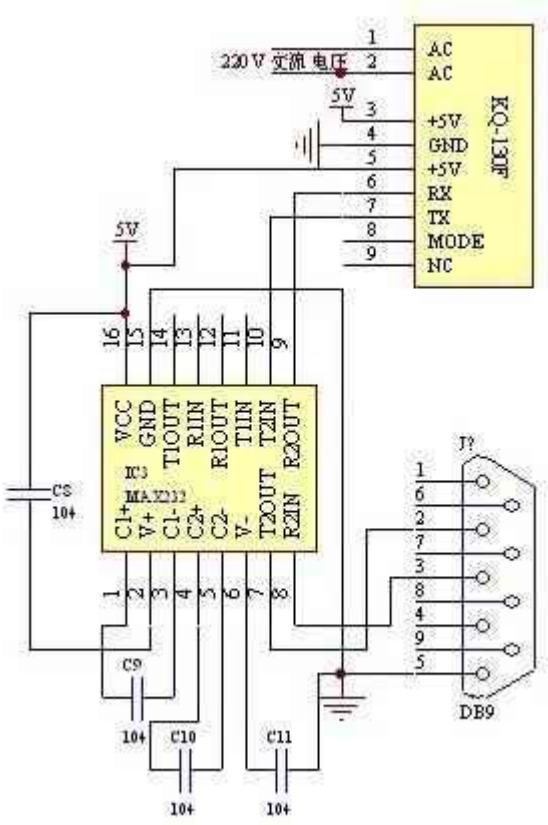


KQ-130F 电力载波数据收发模块与单片机连接图：



KQ-130F 电力载波数据收发模块与微机 9 针 RS232 口连接图：

DB9 的 2 脚接微机的 RXD，3 脚接微机的 TXD



KQ-130F 电力载波数据收发模块经上图与微机 9 针 RS232 连接后用串口调试助手测试图：

发送 08 01 02 03 04 05 06 78 89

另一台电脑接收到的：

附 1：

有关我公司模块的文章参考：

http://auto.yidaba.com/jsqy/40081.shtml

("载波通信模块在远程抄表系统中的应用"，引自 2006 年 6 月 21 日电子查询网有关 KQ-100 系列产品的用户使用经验）

http://www.dzsc.com/data/html/2011-8-18/92862_2.html

（"基于 GPS 公交车电子站牌系统的设计",引自 2011 年 8 月 18 日王光学老师有关 KQ-100 系列产品的用户使用经验）

http://www.docin.com/p-64949479.html

("KQ-100K 模块在 220V 电力线载波通信系统中的应用"《电子元器件应用》2008 年第 9 期)

http://www.kq100.com/kq100l.pdf

("电力载波在农村电网控制系统中的应用" 《农业科学研究》 2010 年 9 月）

http://www.kq100.com/kq100.pdf

("基于 FSK- KQ100 模块的低压电力载波通讯电路设计"《沈阳工程学院学报(自然科学版))》2006 年 1 月)

http://www.kq100.com/kq100e1.pdf

("采用电力载波模块对控制与保护开关的远程控制方案" 《低压电器》 2010 年 23 期)

附 2：

KQ-130 模块作为从设备，返回微机通过串口调试助手发送数据的 89C2051 单片机 C 语言程序：（中断接收发送方式）

HEX 文件下载地址（下载后把尾缀.h 修改成.hex）

www.kq100.com/kq_test.h

C 语言源文件下载地址：

www.kq100.com/kq_test.c

```
/*
   上位机发送 04 12 34 56 78
    89C2051 收到上位机后返回同样的数据。
   使用 11.0592M 的晶振
   我们已经测试通过。
*/
#include <reg51.h>
#include <string.h>
#include <absacc.h>
#include <math.h>
bit PTT;
unsigned char  trbuf[64];
main()
{
register unsigned int Dcn;

PCON=0X80;
    TMOD=0X21;
```

```
    TR0=1;
    IP=0X10;
    SCON=0X70;
    TH1=0XFA;
    TR1=1;
      IE=0X90;
      TR1=1;
      PTT=1;
SCON=0x70;
PCON=0x80;
P1=0X0FF;
P3=0X0FF;
while(1)
{
}
}
void estr0() interrupt 4 using 2{
static unsigned char len=0,max=0,i;
  unsigned char j,k;
    if (RI)
    {
    RI=0;
```

```
        if (PTT)
        {
        k=SBUF;
        trbuf[max]=k;
        if (len==0)
        {
          max=0;
          trbuf[0]=k;
          len=k;
          max++;}
        else if (max==len)
        { SBUF=trbuf[0];
          TI=0;
           PTT=0;
           max=1;
        }
        else
        {max++;
        }
         }
    }
      if(TI)
```

```
    { TI=0;
            if(!PTT)
            {
            SBUF=trbuf[max];
            if (max==len)
            {
             PTT=1;
             max=0;
             len=0;
            }
            else
            {
            max++;
            }
            }
    }

}
```

AT89C51 C 语言连续发送程序参考：

```
//AT89C51 系列发送示范程序，11.0592M 的晶振,MODE 脚接地

char b[]={0x03,0x12,0x34,0x56}; //03 代表要发送数据的长度
void delay_ms(int t);
```

```
void main()
{

   int max;
   PCON=0X80;       //第 1 步，初始化串口，10 位方式，一个停止位
   SCON=0X70;
   TMOD=0x20;       // 第 2 步，定时器 1 工作方式 2
   TH1=0XFA;        // 第 3 步，赋定时器 1 的初值，设置成 9600BPS
   TL1=0XFA;

   TR1=1;           // 第 4 步，开定时器 1


   while(1)
   {


         for(max=0;max<4;max++)
{
         SBUF=b[max];//0x03,0x12,0x34,0x56 //独立接收发送缓冲器


         while(TI==0);//发送中断标志
         TI=0;
}
   delay_ms(800);//延迟 800mS 加 4 个前导码，共 8 字节，要 800mS

   }
}
void delay_ms(int t)
{
   int a;
   while(t--)
   {
   for(a=300;a>0;a--);
   }
}
```

//AT89C51 系列接收示范程序，11.0592M 的晶振,MODE 脚接地

```
char b[20]; //在这假如接收 3 个有效字节，就是上面的发送的数据
void main()
{
```

```
    int max;
    PCON=0X80;          //第 1 步，初始化串口，10 位方式，一个停止位
    SCON=0X70;
    TMOD=0x20;          // 第 2 步，定时器 1 工作方式 2
    TH1=0XFA;           // 第 3 步，赋定时器 1 的初值，设置成 9600BPS
    TL1=0XFA;

    TR1=1;              // 第 4 步，开定时器 1


    while(1)
    {

          RI=0;
          for(max=0;max<4;max++)
{
          while(RI==0);//发送中断标志
          RI=0;
          b[max]=SBUF;//0x03,0x12,0x34,0x56 //独立接收缓冲器



}
 //接收不需要延迟
    }
}
```

```
    /*
     *STM32 源程序要感谢 ibeerbear 无私的奉献
     * 说明：利用 STM32 的 USART1 与 KQ-130F 通信，已调试通过
     * 线接法：STM32/PA9/TX -> KQ130F/RX，STM32/PA10/RX ->
```

KQ130F/TX，KQ130F/RST 悬空，KQ130F 双 +5V 供电，AC 脚接家用插座

* 另外：1，KQ130F 工作电压可以调整为 3.3v；2，发送帧之间一定要有时间间隔

*/

```
// 先配置 STM32 RCC、GPIO 等

// 再配置 USART1
void USART1_Config(void) {
    // 别忘记配置 USART1 的 RCC

    GPIO_InitTypeDef GPIO_InitStructure;
    USART_InitTypeDef USART_InitStructure;

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
```

```
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    USART_InitStructure.USART_BaudRate = 9600;
    USART_InitStructure.USART_WordLength = USART_WordLength_9b;
    USART_InitStructure.USART_StopBits = USART_StopBits_1;
    USART_InitStructure.USART_Parity = USART_Parity_No ;
    USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;

    USART_Cmd(USART1, DISABLE);
    USART_Init(USART1, &USART_InitStructure);
    USART_Cmd(USART1, ENABLE);
}

#ifdef _SERVER_
unsigned char  data;
#elif defined _CLIENT_
```

```
    unsigned char Buffer[] = { 7,0,1,2,3,4,5,6 };
    unsigned int Index = 0;
    #endif


    while (1) {
    #ifdef_SERVER_
        if(USART_GetFlagStatus(USART1,    USART_FLAG_RXNE)    !=
RESET) {
            data = USART_ReceiveData(USART1);
            if(data <= 7) {
                // 在这里添加处理数据的代码
                LED_Toggle(GPIO_LED_PORT, GPIO_LED_PIN);
            } else {
                // 在这里添加错误数据处理代码
            }
        }
    #elif defined _CLIENT_
        Index = Index % 8;
        if (Index == 0) {
            // Delay 函数利用 systick 精确延迟 2 秒
            Delay(2 * 1000 * 1000);
        }
```

```
        USART_SendData(USART1, Buffer[Index++]);
        while(USART_GetFlagStatus(USART1,  USART_FLAG_TXE)  ==
RESET);
    #endif
    }
```