KQ-130F power line carrier data transceiver module

· KQ-130F is a small high-performance single-pin 9 through the zero carrier data transceiver module. Is designed for 220V AC on the strong interference, strong attenuation, the distance requirements of the environment, the reliable transmission of data and in particular the design and development of high cost carrier module. Is used for metering, street, intelligent home, fire, and other building control applications need to transmit data to the power line.

A, KQ-130F Series Performance: ·

1. KQ-330F integrated module and the carrier plate peripheral circuits, no need of other coupling elements, are directly connected to AC 220V use. Dimensions of 53 × 38 × 17 mm (L × D × H), single lead pin (see below) 2 feet to 220V AC power non-directional (1 foot, 2 feet spacing 2X0.1 inches), 2 feet, 3 feet spacing of 1.1 inches, 0.1 inches spacing between each of the remaining pins.
2. The operating frequency of 120 ~ 135KHZ, the interface baud rate 9600bps. The actual baud rate 100bps, 250 byte buffers.
3. Temperature range: -25 ℃ ~ + 70 ℃ Humidity ≤90%
4. Power supply: ≤11mA when transmitting:: DC + 5V when receiving ≤300mA

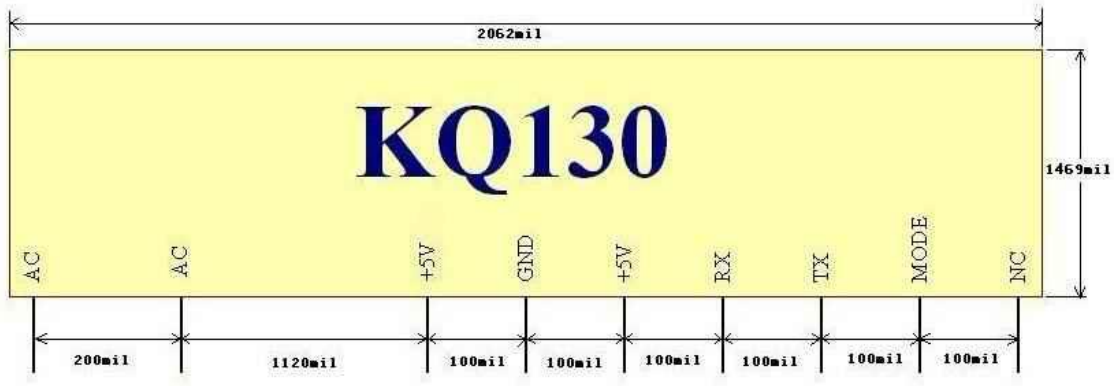
Second, the specifications and model: ·

KQ-130F: ·

After the first letter 130 is defined as:·

F: zero-crossing transmission type ·

Three, KQ-330 Pin Description ·: From left to right front foot 1 to 9:



1P-AC: Firewire (or neutral) AC voltage 220V
2P-AC: AC 220V line voltage zero (or FireWire)
3P- + 5V:+ 5V power transmission (260mA), data can be received if a single vacant reduce power consumption
4P-GND: Ground digital circuit

5P- + 5V: + 5V working power 11mA
6P-RX: TTL level into a data carrier, then MCU TXD,**Hi-Z input can not be suspended**
7P-TX: TTL level, a data carrier, then the microcontroller RXD
8P-MODE: mode selection, floating or connected to 5V is high, low ground
9P-NC / RST: reset pin (active low) mode is used only when switching frequently at work. No need for this feature, the pin should be left unconnected

Four, KQ130F series modules programming considerations ·

This module interface baud 9600bps, please use the user module 9600BPS asynchronous communication, the format of a start bit, 8 data bits, 1 stop bit format.

This module uses a transparent mode (high level) by the control module MODE pin, or a custom operating mode (low level). MODE high level (floating) transparent mode, a low level (ground) for the custom work mode.

In the transparent mode of operation: (MODE = 1 i.e. floating or connected to 5V MODE recommendation vacant.) No need to initialize the module programming, RS-485, and normal communication time similar manner. However, since more power line load, electrical harmonics generated will unavoidably coupled to the power line, this module is a high sensitivity carrier module, the carrier module are all in the receiving state, the power line will all be electrical harmonics generated is covered, then the module will demodulated data output from the TX terminal noise. Therefore, transmitting and receiving data to be introduced into the preamble to distinguish the real data transfer.

note: The transmit buffer (253 bytes) in the module no longer receive new data is full. I.e. a transmitted bytes is smaller than 253 bytes. A user data to send uninterrupted module, if the pause time exceeds the module has finished sending all the data time (buffer empty, the last byte has been completely sent), the receiver module may be inserted noisy data.

The continuously transmitted to the RX side: 5A 5A 5A 34 56 78 12 45 67 may output other receiving module

FE FD EF 5A 5A 5A 34 56 78 12 45 67 85 DE EF. Darkened modules when all bytes have not transmit data, receive data received noise module.

9600BPS received data or the use of asynchronous format of one start bit, 8 data bits, 1 stop bit format, sent from the TX, it takes about 0.09 seconds approximately every transmission time. Is also equivalent when the custom mode.

When custom work mode: (MODE = 0 MODE i.e., ground) User defined data transfer according to the company, a data transfer is defined as follows:

The first byte: the number of bytes to be transferred in a 0-250 (excluding the first byte)

The second byte to n + 1 bytes: byte user data for transfer

note: When a module is not finished sending data, the data will not receive the next frame.

Receiving and sending data equivalent.

The transmitted to the RX side:02 AE 87 outputs the other modules TX02 AE 87

02Is the byte length, which indicates that the following 2 bytes of data.

The transmitted to the RX side: 09 01 0,203,040,506,070,809

In other module outputs TX09 01 0,203,040,506,070,809

09Is the byte length, which indicates that the following nine bytes of data. The maximum length of 253 bytes.

The transmitted to the RX side: FD 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E ... FD

In other module outputs TXFD 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E ... FD

Valid data to be 253.

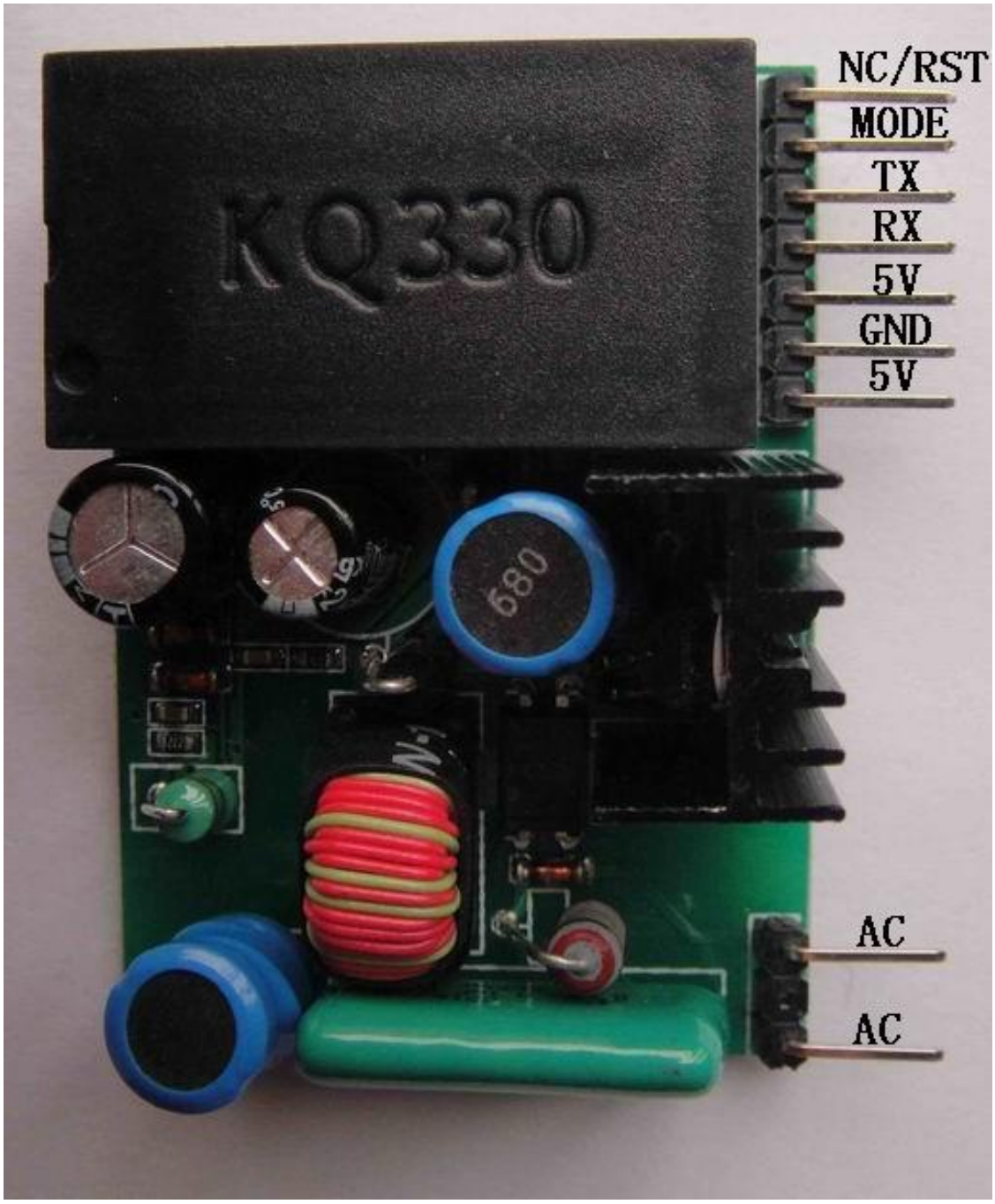
KQ-130 series modules differences:

KQ-130F is designed for AC 220V / 110V, 50HZ / 60HZ strong interfering AC module design based on the zero carrier transmission scheme, the transmission power upper city with good effect, distance and so on. Data to be transmitted in the case of AC power, the carrier rate 50HZ / 100BPS, 60HZ / 120BPS our optimization 9 BIT software can be transferred by one byte. KQ-130E is complete carrier module, and transmitting data independent of the zero point. After carrier demodulation digital filtering done to improve its anti-interference ability of the data carrier, the lower the rate the better. KQ-130E can pay at 0V-220V DC voltage for carrier communication, such as: 220V, 110V, 80V, 48V, 36V, 24V, 12V DC voltage, and the like in the communication carrier power failure. KQ-130E is the highest rate 400BPS. Communication can choose 100BPS improve anti-jamming capability, but also 100BPS of KQ-100E communication effect on AC power than a lot of KQ-130F poor. KQ-130K of communication and KQ-130E same, the only difference is the actual carrier rates up can be done 1200BPS. Suffix with + sign, KQ-130F +, KQ-130E +, KQ-130K +, can support the transmission power voltage is continuously transmitting data for a long time in the case of 12V.

KQ-130 series modules in common:

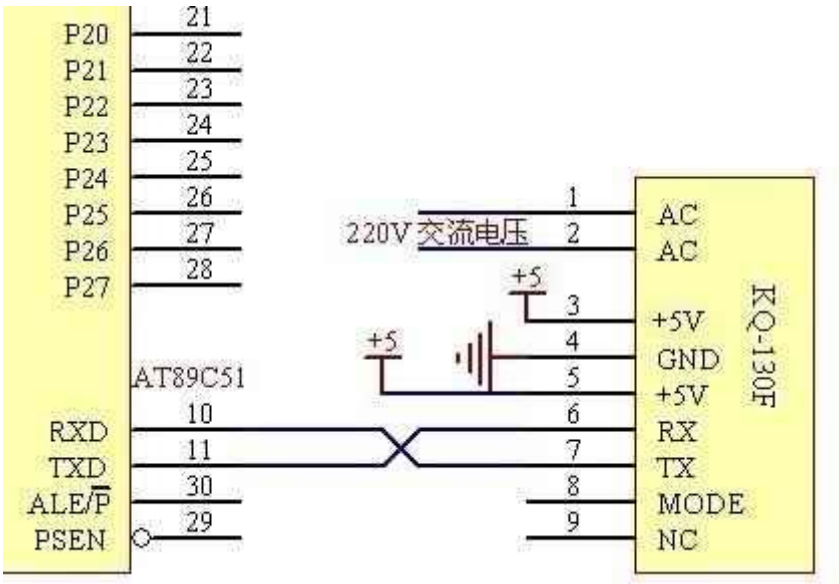
The interface is the same for all modules and KQ-130 Series microcontroller / microcomputer, same communication. And microcontroller / microcomputer serial interface rate is 9600BPS, a start bit, eight data bits and one stop bit. Of the same transparent operating mode or custom work mode.

KQ-130F power line carrier transceiver module image data:



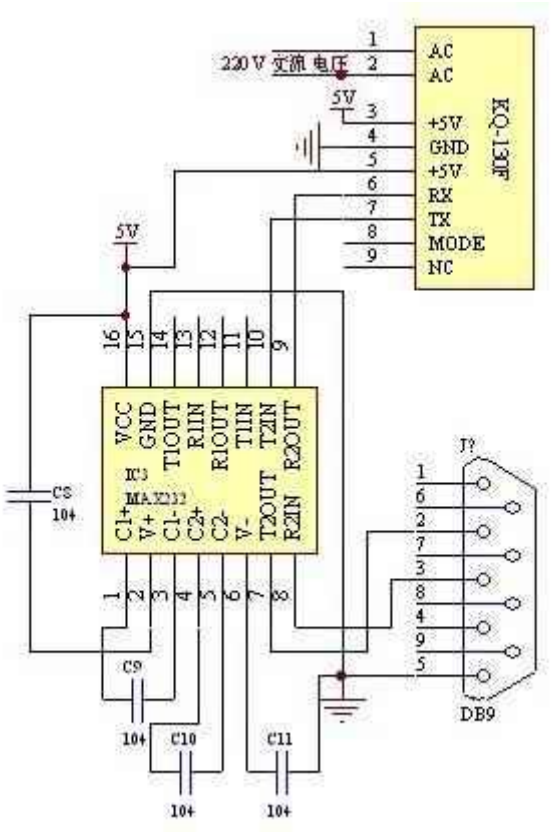
KQ-130F power line carrier data transceiver module

connected with the microcontroller FIG:



KQ-130F power line carrier data transceiver module and the microcomputer 9 pin RS232 port connection FIG:

DB9 2 feet to the microcomputer RXD, 3 TXD pin is connected to the microcomputer

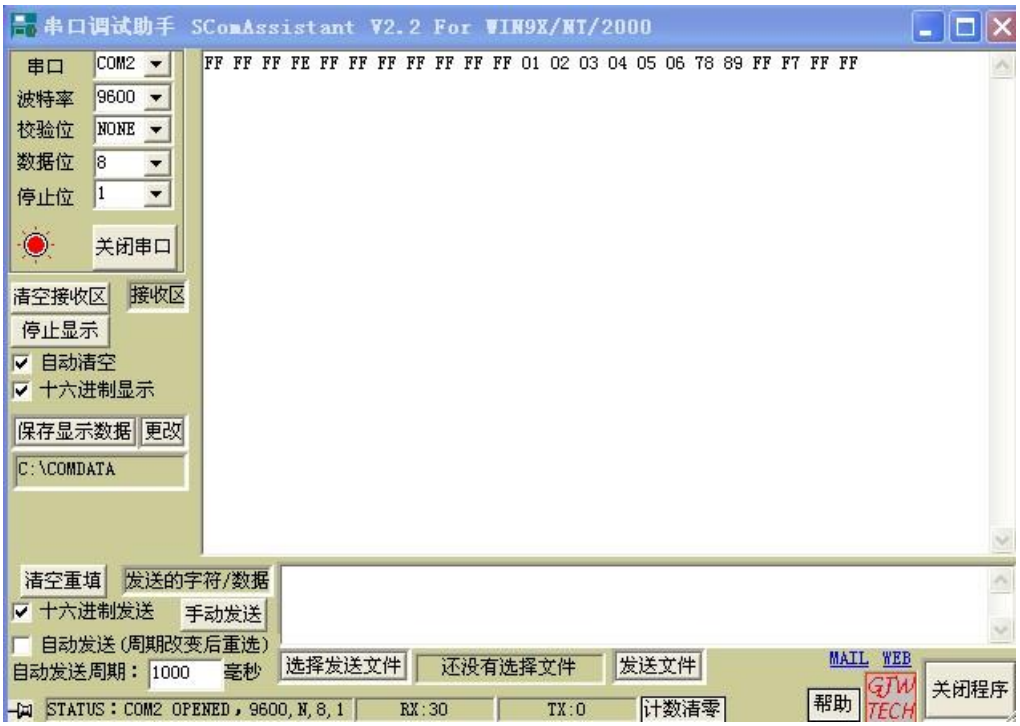


KQ-130F power line carrier data transceiver module after the microcomputer of FIG. 9 pin RS232 serial debugging assistant connection test pattern:

When a high level MODE =:

Transmitting 0,102,030,405,067,889 data, up to 255 bytes may be sent continuously. If a transmission of data exceeds 255 bytes, 250 bytes after the send delay of 20 seconds, and then send back data.

Received another computer to:



0,102,030,405,067,889 after a lot of noise. But it is completely transparent transmission, without adding any data.

MODE = 0 the low level:

Send 08 0,102,030,405,067,889

The first byte is a format requirements: the number of bytes to be sent,

there are eight, up to 250 (0xF0)

Received another computer to:

No noise, but we must newsletter format, and will increase the number of preamble.

Appendix 1:

Our articles about the module reference:

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

("Application of carrier communication module in a remote meter reading system", citedJune 21, 2006Queries related to electronic network user KQ-100 series of products using the experience)

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

("GPS-based bus electronic stop sign system design," quoted Wang optical teacher August 18, 2011 relating to the user KQ-100 series of products using the experience)

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

("Application KQ-100K module 220V power line carrier communication system" "Electronic Component Applications" 2008 9)

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

("Application of power carrier wave power grid control system in the rural areas", "agricultural research" September 2010)

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

("FSK- KQ100 module based on Power Line Carrier Communication Circuit Design", "Shenyang Institute of Engineering (Natural Science))," January 2006)

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

("Remote control using power line carrier module of the control and protection scheme switched"**"Low-voltage electrical"**2010 23)

Annex 2:

KQ-130 module as a slave device, MODE grounded. Return to the PC through the 89C2051 Microcontroller C Programming Language serial debugging assistant to send data :(interrupt transmission and reception mode)

HEX file download address (after downloading the modified .h suffix .hex)

www.kq100.com/kq_test.h

C language source file Download:

www.kq100.com/kq_test.c

```
/ *
  MODE = 0 the host computer transmits 0412345678
   89C2051 return the same data after receiving the host
  computer.
  Use of crystal 11.0592M
  We have the test.
* /
#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 ++;
}
```

```
    }
     }


   }
```

Continuous transmission AT89C51 C language program reference:

```
// AT89C51 Series transmitters demonstration program, 11.0592M
crystal, MODE pin to ground

char b [] = {0x03,0x12,0x34,0x56}; // 03 representative of the length
of the data to be transmitted
void delay_ms (int t);
void main ()
{

 int max;
 PCON = 0X80; // Step 1, initialization of the serial 10-bit mode, a
stop bit
 SCON = 0X70;
 TMOD = 0x20; // Step 2, Timer 1 Mode 2
 TH1 = 0XFA; // Step 3, the initial value of the timer 1 is assigned,
is arranged to 9600BPS
 TL1 = 0XFA;

 TR1 = 1; // Step 4, an open timer


 while (1)
 {


 for (max = 0; max <4; max ++)
{
 SBUF = b [max]; // 0x03,0x12,0x34,0x56 // independent transmission
and reception buffer
```

```
 while (TI == 0); // Transmit Interrupt Flag
 TI = 0;
}
 delay_ms (800); // add 4 800mS delay preamble, a total of 8 bytes, to
800mS

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

// AT89C51 series receivers demonstration program, 11.0592M crystal, MODE pin to ground

```
char b [20]; // data if it receives three significant byte is
transmitted from the above
void main ()
{

 int max;
 PCON = 0X80; // Step 1, initialization of the serial 10-bit mode, a
stop bit
 SCON = 0X70;
 TMOD = 0x20; // Step 2, Timer 1 Mode 2
 TH1 = 0XFA; // Step 3, the initial value of the timer 1 is assigned,
is arranged to 9600BPS
 TL1 = 0XFA;

 TR1 = 1; // Step 4, an open timer


 while (1)
 {

 RI = 0;
 for (max = 0; max <4; max ++)
{
```

```
 while (RI == 0); // Transmit Interrupt Flag
 RI = 0;
 b [max] = SBUF; // 0x03,0x12,0x34,0x56 // independent reception
buffer



}
 // do not need to delay reception
 }
}
```

/ *

* STM32 source thanks ibeerbear Selfless dedication

* Description: USART1 use in communication with the KQ-130F STM32 has debugging

* Wire connection: STM32 / PA9 / TX -> KQ130F / RX, STM32 / PA10 / RX -> KQ130F / TX, KQ130F / MODE ground, KQ130F / NC vacant, KQ130F bis + 5V power supply, AC feet to a household outlet

* In addition: 1, KQ130F operating voltage can be adjusted to 3.3v; 2, there must be a time interval between transmission frames

```
 * /

// configure STM32 RCC, GPIO, etc.

// reconfiguration USART1
void USART1_Config (void) {
   // Do not forget the RCC configuration USART1

   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) {
            // Add code here for processing data
            LED_Toggle (GPIO_LED_PORT, GPIO_LED_PIN);
         } Else {
            Here // add error handling code data
         }
      }
   #elif defined _CLIENT_
      Index = Index% 8;
      if (Index == 0) {
         // Delay function using 2 seconds delay accurate
systick
         Delay (2 * 1000 * 1000);
      }
      USART_SendData (USART1, Buffer [Index ++]);
      while (USART_GetFlagStatus (USART1, USART_FLAG_TXE) ==
RESET);
```

```
#endif

}
```