Guideline for configuring the S2E as TCP Server by MCU

V1.1

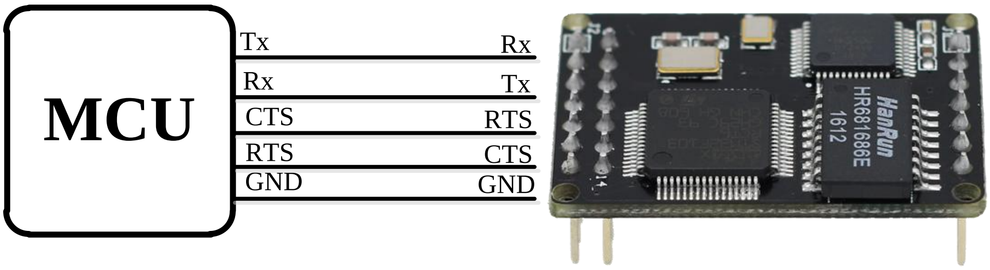
Document Revision History

Version	Date	Remark
V1.0	2017/11/09	Official Release
V1.1	2017/12/12	Add the third part code

Copyright notice

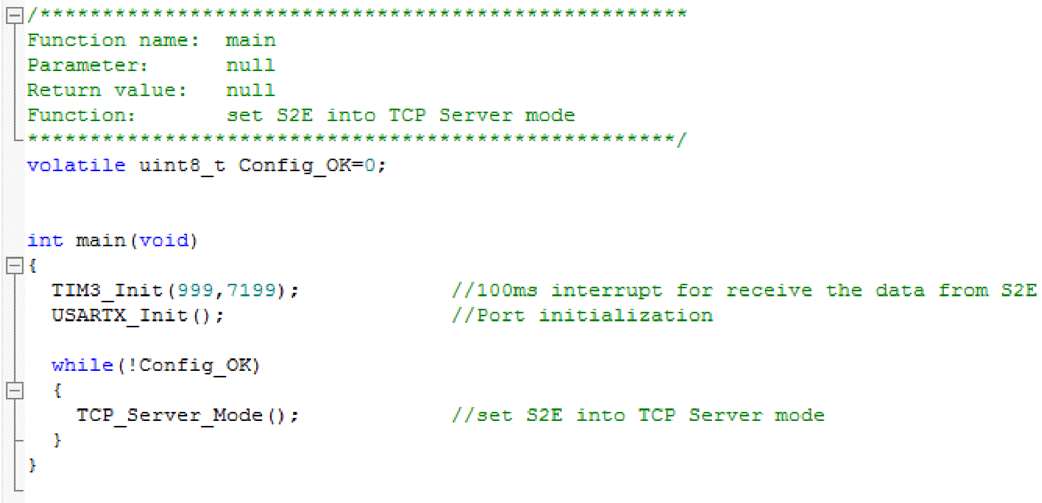
Contact E-mail: sales@wiznet.hk
For more information, please visit: www.wiznet.com.hk

1、 Hardware connection



2、 Example explanation

Open the ‘Guideline for configure the S2E as TCP Server by MCU ’ project，the first part of the main function TIM3_Init(); It set up an interrupt timer. After configuring the S2E by AT commands, it will send back data to the MCU. In this example, the MCU serial ports receive data by frame interrupt.

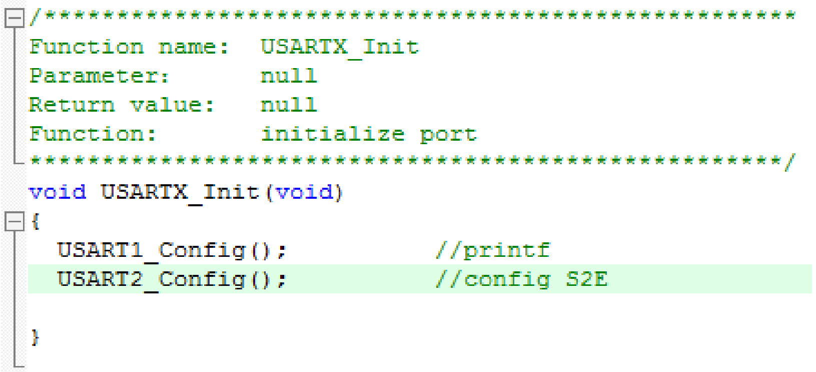


```
/*****************************************************
Function name:   main
Parameter:       null
Return value:    null
Function:        set S2E into TCP Server mode
*****************************************************/
volatile uint8_t Config_OK=0;


int main(void)
{
  TIM3_Init(999,7199);            //100ms interrupt for receive the data from S2E
  USARTX_Init();                  //Port initialization

  while(!Config_OK)
  {
    TCP_Server_Mode();            //set S2E into TCP Server mode
  }
}
```

The second function USARTX_Init(); It initializes the ports of the MCU by calling the USART1_Config() function; It is a printf function which is used to watch the debugging information. Function USART2_Config() is used to configure the S2E. Note: The configure parameters of the MCU serial port should be as same as the serial port configure parameters of the S2E, otherwise the configuration failed to initialize.



```
/*****************************************************
Function name:  USARTX_Init
Parameter:      null
Return value:   null
Function:       initialize port
*****************************************************/
void USARTX_Init(void)
{
  USART1_Config();            //printf
  USART2_Config();            //config S2E

}
```

The third function TCP_Server_Mode() in the main function is used to configure the S2E into TCP Server mode. The details of the S2E AT commands is available in the AT command chapter of each S2E model User manual。If successfully configure, the serial port will printout ‘TCP Server Config Success!’, otherwise it will print ‘TCP Server Config Fail!’.

```
/*****************************************************
Function name:   TCP_Server_Mode
Parameter:       null
Return value:    null
Function:        Send AT-command to configure S2E module via UART
*****************************************************/
volatile uint8_t SendFlag=0;

void TCP_Server_Mode(void)
{
  uint8_t RecvFlag=1;
   char *state;

  switch(SendFlag)
   {
      case 0:
    {
    Usart_Send(USART2,"AT\r\n");//Terminal check
      while(RecvFlag)
        {
       if(RX2_Point & FRAME_LEN)         //If receive data
          {
       state=strstr((char *)RecvBuff,"OK"); //check is there 'OK' in receive buffer
             if(state!=NULL)           //true
       {
                 RX2_Point=0;             //clear receive flag
            RecvFlag=0;               //clear receive state flage
         SendFlag=1;     //se send state flag
           printf("Recv:%s\r\n",RecvBuff);
              memset(RecvBuff,0,RECV_LEN);    //clear receive buffer
       }
       else{    //false
        SendFlag=100; //Failed to configure
        RecvFlag=0;
       }
          }
        }
      }break;
    case 1:
    {
    Usart_Send(USART2,"AT+ECHO=0\r\n");//(open --1/ close --0)    echo command
        RecvFlag=1;
      while(RecvFlag)
        {
       if(RX2_Point & FRAME_LEN) //If receive data
          {
       state=strstr((char *)RecvBuff,"OK");
             if(state!=NULL)
       {
                 RX2_Point=0;
            RecvFlag=0;   //clear receive state flage
         SendFlag=2;
           printf("Recv:%s\r\n",RecvBuff);
              memset(RecvBuff,0,RECV_LEN);
       }
             else{
        SendFlag=100;
        RecvFlag=0;
       }
           }
        }
      }break;
    case 2:
    {
    Usart_Send(USART2,"AT+C1_OP=0\r\n");//Set into TCP Server mode
      RecvFlag=1;
      while(RecvFlag)
        {
       if(RX2_Point & FRAME_LEN) //If receive data
           {
      state=strstr((char *)RecvBuff,"OK");
      if(state!=NULL)
      {
                 RX2_Point=0;
            RecvFlag=0; //clear receive state flage
          SendFlag=3;
          printf("Recv:%s\r\n",RecvBuff);
              memset(RecvBuff,0,RECV_LEN);
              }
        else{
        SendFlag=100;
        RecvFlag=0;
       }
      }
        }
      }break;
    case 3:
    {
    Usart_Send(USART2,"AT+IP_MODE=0\r\n");//set into static IP mode
    RecvFlag=1;
      while(RecvFlag)
        {
       if(RX2_Point & FRAME_LEN)//If receive data
           {
      state=strstr((char *)RecvBuff,"OK");
      if(state!=NULL)
      {
                 RX2_Point=0;
            RecvFlag=0;  //clear receive state flage
          SendFlag=4;
          printf("Recv:%s\r\n",RecvBuff);
              memset(RecvBuff,0,RECV_LEN);
              }
        else{
        SendFlag=100;
        RecvFlag=0;
       }
      }
        }
      }break;
    case 4:
    {
    Usart_Send(USART2,"AT+IP=192.168.1.88\r\n");    //configure locak IP address
    RecvFlag=1;
      while(RecvFlag)
        {
       if(RX2_Point & FRAME_LEN)//If receive data
           {
      state=strstr((char *)RecvBuff,"OK");
      if(state!=NULL)
```

```
    {
                    RX2_Point=0;
             RecvFlag=0;    //clear receive state flage
          SendFlag=5;
          printf("Recv:%s\r\n",RecvBuff);
               memset(RecvBuff,0,RECV_LEN);
               }
       else{
       SendFlag=100;
       RecvFlag=0;
      }
     }
       }
    }break;
 case 5:
 {
  Usart_Send(USART2,"AT+MARK=255.255.255.0\r\n");    //configure locak IP address
  RecvFlag=1;
     while(RecvFlag)
        {
      if(RX2_Point & FRAME_LEN)//If receive data
          {
    state=strstr((char *)RecvBuff,"OK");
    if(state!=NULL)
    {
                    RX2_Point=0;
             RecvFlag=0;   //clear receive state flage
          SendFlag=6;
          printf("Recv:%s\r\n",RecvBuff);
               memset(RecvBuff,0,RECV_LEN);
               }
       else{
       SendFlag=100;
       RecvFlag=0;
      }
     }
       }
    }break;
 case 6:
 {
  Usart_Send(USART2,"AT+GATEWAY=192.168.1.1\r\n");    //configure locak IP address
  RecvFlag=1;
     while(RecvFlag)
        {
      if(RX2_Point & FRAME_LEN)//If receive data
          {
    state=strstr((char *)RecvBuff,"OK");
    if(state!=NULL)
    {
                    RX2_Point=0;
             RecvFlag=0;   //clear receive state flage
          SendFlag=7;
          printf("Recv:%s\r\n",RecvBuff);
               memset(RecvBuff,0,RECV_LEN);
               }
       else{
       SendFlag=100;
       RecvFlag=0;
      }
     }
       }
    }break;
  case 7:
```

```
{
Usart_Send(USART2,"AT+C1_PORT=5000\r\n");//configure locak port number
RecvFlag=1;
  while(RecvFlag)
     {
   if(RX2_Point & FRAME_LEN)//If receive data
       {
  state=strstr((char *)RecvBuff,"OK");
  if(state!=NULL)
  {
             RX2_Point=0;
       RecvFlag=0;  //clear receive state flage
     SendFlag=8;
     printf("Recv:%s\r\n",RecvBuff);
         memset(RecvBuff,0,RECV_LEN);
         }
    else{
    SendFlag=100;
    RecvFlag=0;
   }
  }
     }
 }break;
case 8:
{
Usart_Send(USART2,"AT+START_MODE=0\r\n"); //configure start mode (0--AT mode £¬1--data mode)
RecvFlag=1;
  while(RecvFlag)
     {
   if(RX2_Point & FRAME_LEN)//If receive data
       {
  state=strstr((char *)RecvBuff,"OK");
  if(state!=NULL)
  {
             RX2_Point=0;
       RecvFlag=0;  //clear receive state flage
     SendFlag=9;
     printf("Recv:%s\r\n",RecvBuff);
         memset(RecvBuff,0,RECV_LEN);
         }
 else{
    SendFlag=100;
      RecvFlag=0;
   }
  }
     }
 }break;
case 9:
{
Usart_Send(USART2,"AT+EXIT\r\n");    //Save the configuration and set into data mode
     RecvFlag=1;
  while(RecvFlag)
     {
   if(RX2_Point & FRAME_LEN)//If receive data
       {
  state=strstr((char *)RecvBuff,"OK");
  if(state!=NULL)
  {
             RX2_Point=0;
       RecvFlag=0;  //clear receive state flage
     SendFlag=99;
     printf("Recv:%s\r\n",RecvBuff);
         memset(RecvBuff,0,RECV_LEN);
```

```
        }
                  else{
          SendFlag=100;
                        RecvFlag=0;
          }
         }
               }
           }break;
      case 99:
        {
      printf("TCP Server Config Success!\r\n");
      Config_OK=1;
        }
     default:
      RecvFlag=100;break;
      case 100:
     {
          printf("TCP Server Config Fail!\r\n");
        Config_OK=1;
     }break;
      }
  }
```