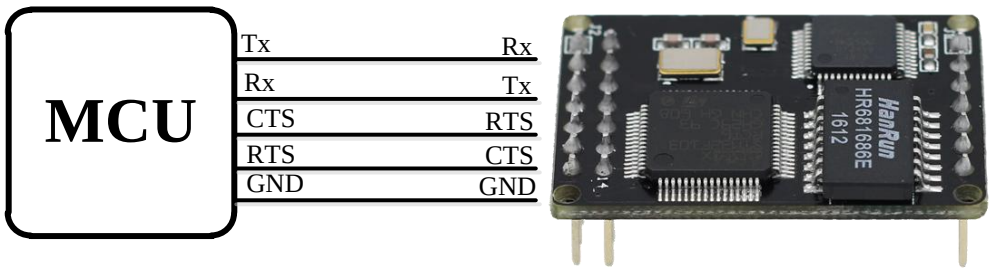
Guideline for configuring the S2E into UDP mode by MCU

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

1、 Hardware connection



2、 Example explanation

Open the 'Guideline for configuring the S2E into UDP mode by MCU' project. The first part of the main function is 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 UDP mode
*****************************************************/
volatile uint8_t Config_OK=0;


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

  while(!Config_OK)
  {
    UDP_Mode();             //set S2E into UDP mode
  }
}
```

The second function is 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 UDP_Mode() in the main function is used to configure the S2E into UDP 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 'UDP Config Success!', otherwise it will print 'UDP Config Fail!'.

```
/*************************************************
Function name:  UDP_Mode
Parameter:  null
Return value: null
Function:    Set S2E into UDP mode
*************************************************/
volatile uint8_t SendFlag=0;

void UDP_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 buffer pointer
          RecvFlag=0;               //clear receive state flag
       SendFlag=1;
         printf("Recv:%s\r\n",RecvBuff);
            memset(RecvBuff,0,RECV_LEN);   //clear receive buffer
    }
    else{
     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;
       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=2\r\n");//Set into UDP 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;    //If receive data
       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=1\r\n");//set into DHCP 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 flag
       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+C1_PORT=5000\r\n");    //Configure the local 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 flag
        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+C1_CLI_IP1=192.168.1.100\r\n");//Set the remote host 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 flag
        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+C1_CLI_PP1=5000\r\n");//Set the remote lcoal server port number
   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 flag
        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+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 flag
      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+EXIT\r\n");    //Save setting and enter the 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 flag
      SendFlag=99;
      printf("Recv:%s\r\n",RecvBuff);
          memset(RecvBuff,0,RECV_LEN);
  }
            else{
    SendFlag=100;
                  RecvFlag=0;
   }
  }
       }
    }break;
case 99:
  {
printf("UDP Config Success!\r\n");
Config_OK=1;
  }
default:
RecvFlag=100;break;
case 100:
{
    printf("UDP Config Fail!\r\n");
  Config_OK=1;
}break;
  }
}
```