Software UART

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	Simulate UART half-duplex communication transmission using GPIO and TIMER.
BSP Version	M051 Series BSP CMSIS v3.01.001
Hardware	NuMaker-EVB-M051 v3.0

1 Function Description

1.1 Introduction

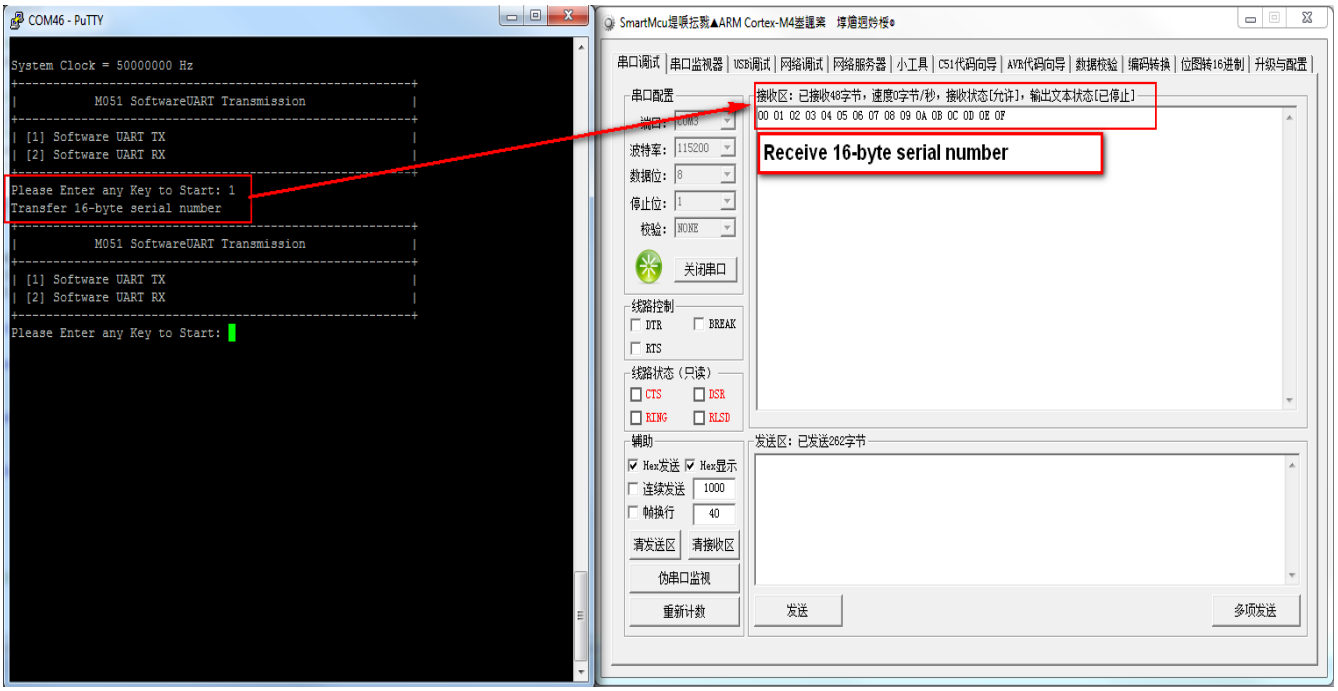
In this example, the IOs of the two GPIOs are set to output mode and input mode and are responsible for receiving or transmitting data. The half-duplex transmission function of UART_TX and UART_RX is simulated with two TIMER counting function generation cycles.

1.2 Principle

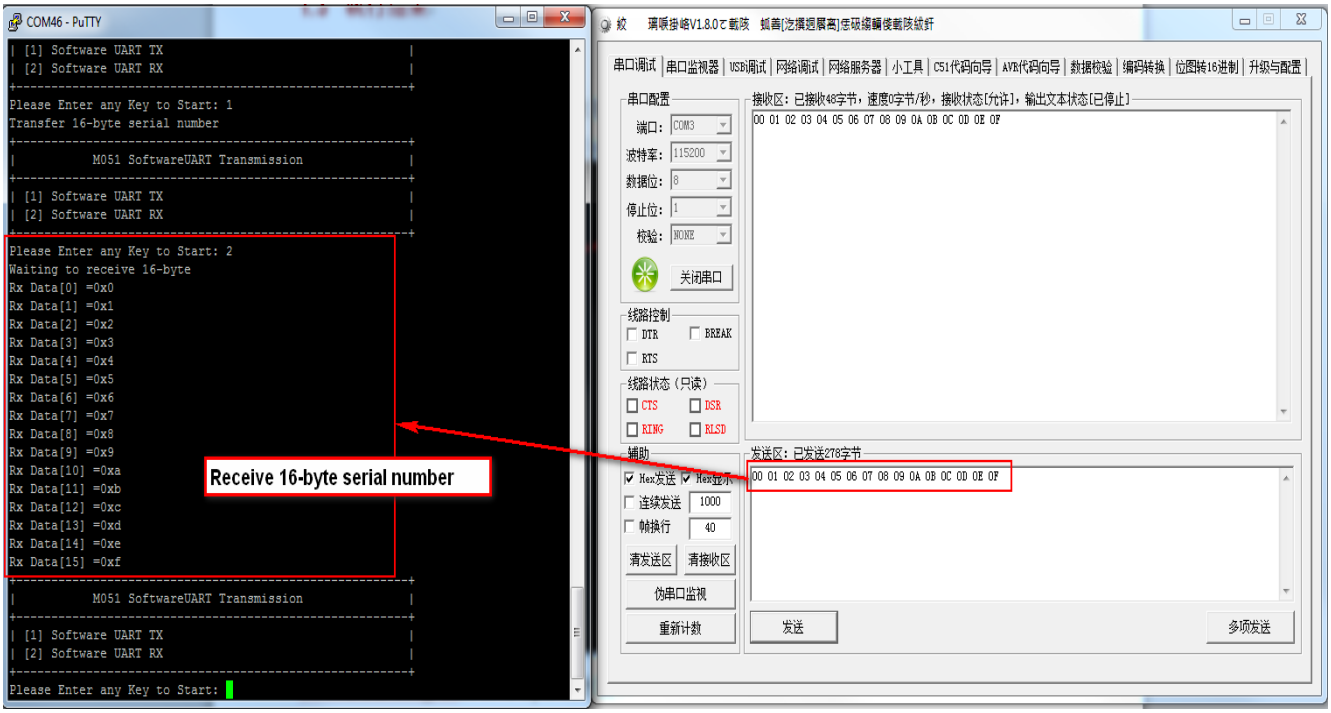
The two functions of this example are analog UART transmit data (TX) and receive data function (RX). The UART TX section first combines the data format to be transmitted into the number of bits to be transmitted (start bit (1) + data bit (5~8) + parity bit (0~1) + stop bit (1~2)). With the cycle count interrupt mode of TIMER1, the serial transmission rate of the data to be transmitted is generated. The number of bits to be transmitted is then changed through the GPIO output High or Low signal. A communication transmission mode for analog UART transmission data is achieved. The UART RX section first calculates the number of received characters using the communication format (start bit (1) + data bit (5~8) + parity bit (0~1) + stop bit (1~2)). With the TIMER0 cycle count interrupt mode, it generates twice the speed of the serial transmission rate of the data to be received. The number of data characters is converted by receiving the GPIO input signal High or Low. Error verification and data conversion will be performed on the number of characters received. A communication transmission mode for analog UART reception data is achieved. The double serial transmission rate is designed to make the input point of the input signal as close as possible to the center point.

1.3 Demo Result

1.3.1 Software UART Transfer Data Execution Result



1.3.2 Software UART Receive Data Execution Results



2 Code Description

2.1.1 Software UART Initialization Setting

```
/* Set to use GPIO emulation UART transfer */
#define GPIO_UART_TX       P27
#define GPIO_UART_RX       P26
```

```
void SoftwareUART_Init(sUART_Config *psConfig)
{
        /* Configure P2.6 as Input mode(Rx)*/
        GPIO_SetMode(P2, BIT6, GPIO_PMD_INPUT);
        /* Configure P2.7 as Output mode(Tx)*/
        GPIO_SetMode(P2, BIT7, GPIO_PMD_OUTPUT);
        /*Initial the SoftwareUART configuration setting*/
        SoftwareUART_ConfigInit(psConfig);
        /* Configure SoftwareUART and set SoftwareUART baudrate */
        SoftwareUART_Open(psConfig,115200);
}
```

2.1.2 Software UART Serial Transmission Rate Setting

```
void SoftwareUART_Open(sUART_Config *psConfig,uint32_t u32BaudRate)
{
       uint32_t u32TimePeriodic;
       psConfig->u32BaudRate = u32BaudRate;
       /*Setting the UART Rx bit rate*/
       u32TimePeriodic = psConfig->u32BaudRate*2;
       /* Open Timer0 frequency in periodic mode, and enable interrupt */
       TIMER_Open(TIMER0,TIMER_PERIODIC_MODE,u32TimePeriodic);
      /* Enable Timer0 INT */
       TIMER_EnableInt(TIMER0);
      /* Setting the UART Tx bit rate */
       u32TimePeriodic  =  psConfig->u32BaudRate;
       /* Open Timer0 frequency in periodic mode, and enable interrupt */
       TIMER_Open(TIMER1,TIMER_PERIODIC_MODE,u32TimePeriodic);
       /* Enable Timer0 INT */
       TIMER_EnableInt(TIMER1);
                                              .
                                              .
}
```

2.1.3 Software UART Data Format Setting

```
void SoftwareUART_DataFormartConfig(sUART_Config *psConfig)
{

     psConfig->u8DataBits = 8;                /* Setting data length is 8 bits */
     psConfig->u8Parity = PARITY_NONE;        /* No Parity */
     psConfig->u8StopBit = STOP_BIT_1;        /* 1 STOP Bit */
     psConfig->u8TxLatency = 2;               /* Transmission interval is 2 bits rate */
     psConfig->u32SendLen = 16;               /* Send 16 bytes (Tx) */
     psConfig->u32ReceiveLen = 16;            /* Receive 16 bytes (Rx) */
                                              .
                                              .
    /* Configure total communication bits length */
     psConfig->u8CommunBitsLen = psConfig->u8DataBits;

    if(psConfig->u8Pairty != PARITY_NONE)
       psConfig->u8CommunBitsLen += 1;     /* Parity bit */
     if(psConfig->u8StopBit != STOP_BIT_1)
       psConfig->u8CommunBitsLen += 3;     /* Start Bit + 2 Stop Bits */
     else
       psConfig->u8CommunBitsLen += 2;     /* Start Bit + 1 Stop Bit */

}
```

2.1.4 Software UART RX Receive Data

```
void SoftwareUART_ReadData(sUART_Config *psConfig)
{
                                              .
                                              .
       u8RxCount = 0;
      /* Use TIMER0 to emulate the bit rate of UART reception*/
      psConfig->u8RxTiming =0;
      /* Wait to Receive the start bit */
      while(GPIO_UART_RX == 1){};
      psConfig->u8RxStatus = eRxRcvData;
      do{
           while(psConfig->u8RxTiming == 0){};
           /* Receive data bit */
          if(GPIO_UART_RX == 0)
             psConfig->pu16RxBuff[u32Index] |= (0 << u8RxCount);
          else
             psConfig->pu16RxBuff[u32Index] |= (1 << u8RxCount);
           psConfig->u8RxTiming =0;
           while(psConfig->u8RxTiming == 0){};
           psConfig->u8RxTiming =0;
           u8RxCount++;
      /* Total bits Length (start bit(1)+data bits(5~8)+ parity bit(0-1)+stop bit(1~2))*/
     }while(u8RxCount < psConfig->u8CommunBitsLen);

      u8RxCount =0;
     /* End of reception */
     psConfig->u8RxStatus = eRxIdel;
     psConfig->u8RxTiming =0;

     }
                                              .
                                              .
}
```

```
void TMR0_IRQHandler(void)
{
    /* Clear Timer0 time-out interrupt flag */
    TIMER_ClearIntFlag(TIMER0);
        /* Receiving data time */
    if( g_psSoftwareUART_Config->u8RxStatus == eRxRcvData)
        g_psSoftwareUART_Config->u8RxTiming++;

}
```

2.1.5 Software UART TX Transfer Data

```
void SoftwareUART_SendData(sUART_Config *psConfig)
{
                                                .
                                                .
     psConfig->u8TxStatus = eTxIdel;
     psConfig->u8TxTiming =0;
                                                .
                                                .
    GPIO_UART_TX = 1;
    /* Wait the latency timing */
    u16SendData = psConfig->pu16TxBuff[u32Index];
    while( psConfig->u8TxStatus != eTxActive){};
    psConfig->u8TxStatus = eTxSendData;

   /* Use TIMER1 to emulate the bit rate of UART transmission */
   do{
             psConfig->u8TxTiming =0;
             if((u16SendData & 0x01) == 0x01)
                GPIO_UART_TX =1;
             else
                GPIO_UART_TX =0;
             u16SendData = u16SendData >> 1;
             while(psConfig->u8TxTiming == 0){};
     /* Total bits Length(start bit(1)+data bits(5~8)+parity bit(0~1)+stop bit(1~2)) */
   }while(psConfig->u8TxSendCount < (psConfig->u8CommunBitsLen+psConfig->u8TxLatency ));

    psConfig->u8TxStatus = eTxIdel;
    psConfig->u8TxSendCount =0;


                                                .
                                                .
}
```

```
void TMR1_IRQHandler(void)
{
    /* Clear Timer1 time-out interrupt flag */
     TIMER_ClearIntFlag(TIMER1);

     g_psSoftwareUART_Config->u8TxSendCount++;
     /*Wait for the Tx transfer start bit to complete*/
     if(g_psSoftwareUART_Config->u8TxStatus == eTxIdel)
     {
        if(g_psSoftwareUART_Config->u8TxSendCount == g_psSoftwareUART_Config->u8TxLatency)
           g_psSoftwareUART_Config->u8TxStatus = eTxActive;
     }/* Tx transmission time */
     else if(g_psSoftwareUART_Config->u8TxStatus == eTxSendData)
              g_psSoftwareUART_Config->u8TxTiming = 1;

}
```

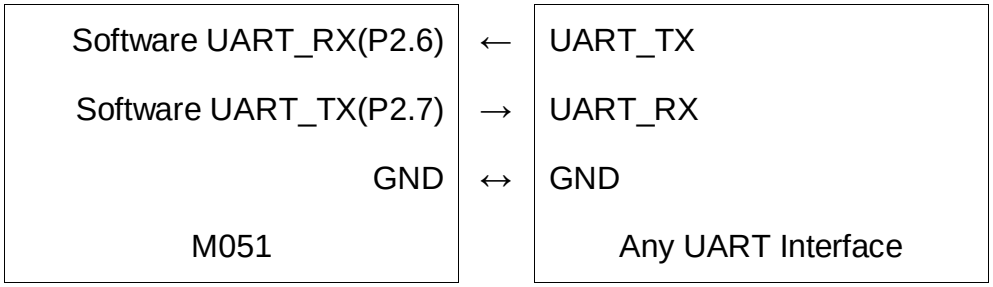
3 Software and Hardware Environment

Software Environment

- BSP version
  - M051 Series BSP CMSIS v3.01.001
- IDE version
  - Keil uVersion 4.70

Hardware Environment

- Circuit components
  - NuMaker-EVB-M051 v3.0
  - Any UART Interface
  - RS232 to TTL Module
- Diagram



4 Directory Information

- EC_M051_SoftwareUART_V1.00
  - Library — Sample code header and source files
    - CMSIS — Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
    - Device — CMSIS compliant device header file
    - StdDriver — All peripheral driver header and source files
  - SampleCode
    - ExampleCode — Source file of example code

5 How to Execute Example Code

1. Browsing into sample code folder by Directory Information (section 4) and double click SoftwareUART.uvproj.
2. Enter Keil compile mode
   a. Build
   b. Download
   c. Start/Stop debug session
3. Enter debug mode
   a. Run

6 Revision History

Date	Revision	Description
JUL, 2019	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.