文档版本	V1.0
发布日期	20210526

APT32S003 UART 应用指南

目录

1 概述

本文介绍了在APT32S003中使用UART的应用范例。

2. 适用的硬件

该例程使用于 APT32S003 系列学习板

3. 应用方案代码说明

3.1 UART 配置

基于 APT32S003 完整的库文件系统，可以对 UART 进行配置。

- **硬件配置：**

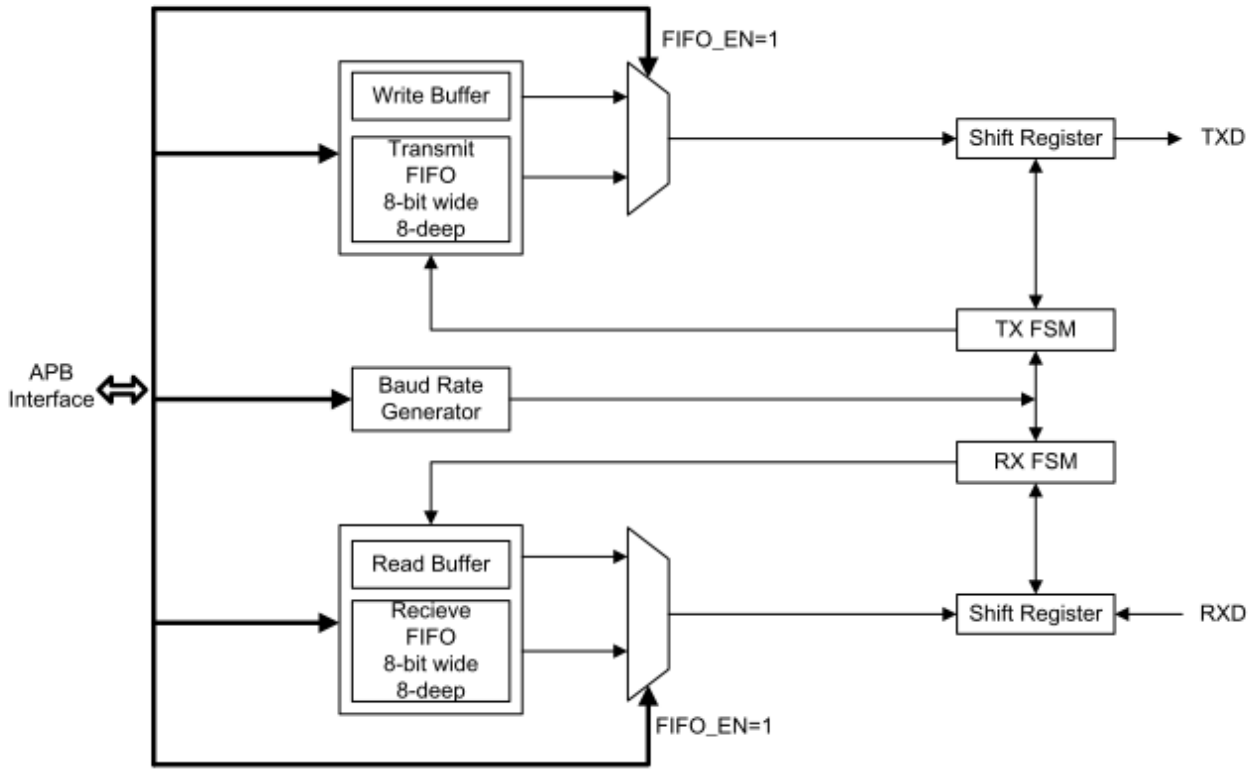

图 3.1.1 功能框图

功能引脚：

UART0	UART0_TX(O)	PA0.0/PA0.12
	UART0_RX(I)	PA0.1/PA0.5
UART2	UART2_TX(O)	PA0.1/PA0.6
	UART2_RX(I)	PA0.0/PA0.7

- **软件配置：**

  可在 apt32S003_initial.c 文件中 UART0_CONFIG()进行初始化的配置；

- **编程要点：**

  1. SYSCON_CONFIG();函数配置

  2.UART0_CONFIG();函数配置

  3.主循环代码

```
/*****************************************************************/
//UART0    CONFIG
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void UART0_CONFIG(void)
{
    UART0_DeInit();                                    //clear all UART Register
    UART_IO_Init(IO_UART0,0);                          //use PA0.1->RXD0, PA0.0->TXD0
    UARTInitRxTxIntEn(UART0,416,UART_PAR_NONE);        //baudrate=sysclock/416=115200
    //UART0_Int_Enable();
}
```

- **代码说明**:

  UART0_DeInit();----用于恢复默认寄存器

  UART_IO_Init();----用于配置 GPIO 口为 UART 功能

  UART0_Int_Enable();-----用于开启中断使能

  UARTInitRxTxIntEn();----用于设置 UART 使能控制和波特率设置

```
UARTInitRxTxIntEn(UART0,416,UART_PAR_NONE);
```

计算公式：

波特率 = PCLK / DIV

- **波特率设置示例**

PCLK	DIV	Baud Rate
48MHZ	5000	9600
	2500	19200
	1250	38400
	416	115200
24MHZ	2500	9600
	1250	19200
	625	38400
	208	115200
12MHZ	1250	9600
	625	19200
	312	38400
	104	115200
6MHZ	625	9600
	312	19200
	156	38400
	52	115200

3.2 UART 发送

```
int main(void)
{
   APT32S003_init();         //初始化

     while(1)
   {
          SYSCON_IWDCNT_Reload();
          UARTTxByte(UART0,0XAA);          //UART0    TX 发送0XAA
   }
```

- **代码说明：**

  UARTTxByte();--------用于发送一个字节

3.3 UART 中断发送

```
/**********************************************************************/
//UART0    CONFIG
//EntryParameter:NONE
//ReturnValue:NONE
/**********************************************************************/
void UART0_CONFIG(void)
{
        UART0_DeInit();                                                    //clear all UART Register
       UART_IO_Init(IO_UART0,0);                                          //use PA0.1->RXD0, PA0.0->TXD0
       UARTInitRxTxIntEn(UART0,48000000/115200,UART_PAR_NONE);          //baudrate=sysclock/46=115200
         UART0_Int_Enable();
}
/* defines -----------------------------------------------------------------*/
volatile U8_T bSendData[20] = {"     hello APT MCU!     "};
volatile U8_T bSendBytes;
volatile U8_T bReceiveData[5];
volatile U8_T bReceiveBytes;
/* externs Register---------------------------------------------------------------*/

/* externs function---------------------------------------------------------------*/
extern void delay_nms(unsigned int t);
extern void APT32S003_init(void);

/**********************************************************************/
//main
/**********************************************************************/
```

```
int main(void)
{
    APT32S003_init();
    bSendBytes = 0;
    bReceiveBytes = 0;
    UARTTxByte(UART0,bSendData[0]);
      while(1)
    {
            SYSCON_IWDCNT_Reload();
      }
}
/*****************************************************************/
//UART0 Interrupt
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void UART0IntHandler(void)
{
      // ISR content ...
    //Interupt
    if ((UART0->ISR&UART_RX_INT_S)==UART_RX_INT_S)
    {
            UART0->ISR=UART_RX_INT_S;
    }
    else if( (UART0->ISR&UART_TX_INT_S)==UART_TX_INT_S )
      {
            UART0->ISR=UART_TX_INT_S;
            TxDataFlag = TRUE;
            if(bSendBytes<20)
            {
                  UARTTxByte(UART0,bSendData[bSendBytes]);    //发送数据
                  if(++bSendBytes==20) bSendBytes = 0;             //数据累加
            }
    }
    else if ((UART0->ISR&UART_RX_IOV_S)==UART_RX_IOV_S)
    {
            UART0->ISR=UART_RX_IOV_S;
    }
    else if ((UART0->ISR&UART_TX_IOV_S)==UART_TX_IOV_S)
    {
            UART0->ISR=UART_TX_IOV_S;
    }
}
```

- **UART0-TX 发送**

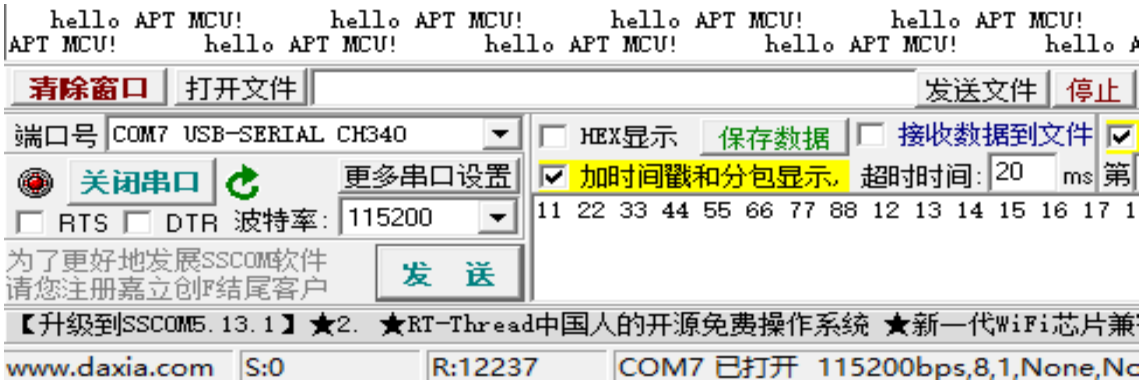

图 3.3.1　PC 串口工具接收

3.4 中断接收数据

```
/*****************************************************************/
//UART0 Interrupt
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void UART0IntHandler(void)
{
    // ISR content ...
  //Interupt
  if ((UART0->ISR&UART_RX_INT_S)==UART_RX_INT_S)
  {
        UART0->ISR=UART_RX_INT_S;
        RxDataFlag = TRUE;
        if(bReceiveBytes==0)
        {
            if((CSP_UART_GET_DATA(UART0)&0xff)==0xa5)      //&0xff 取低位数据，接收到0XA5
            {
                bReceiveBytes ++;
                bReceiveData[0] = 0xa5;
            }
        }
        else
        {
            bReceiveData[bReceiveBytes] = CSP_UART_GET_DATA(UART0)&0xff;
            if(++bReceiveBytes==5)
            {
                bReceiveBytes = 0;
            }
```

```
            }
        }
        else if( (UART0->ISR&UART_TX_INT_S)==UART_TX_INT_S )
          {
                UART0->ISR=UART_TX_INT_S;
                TxDataFlag = TRUE;
        }
        else if ((UART0->ISR&UART_RX_IOV_S)==UART_RX_IOV_S)
        {
                UART0->ISR=UART_RX_IOV_S;
        }
        else if ((UART0->ISR&UART_TX_IOV_S)==UART_TX_IOV_S)
        {
                UART0->ISR=UART_TX_IOV_S;
        }
    }
```

- **代码说明：**

  UART0IntHandler();----UART0 中断服务函数

  UART_RX_INT_S----接收一个字节中断状态

  UART_TX_INT_S----发送一个字节中断状态

  UART_RX_IOV_S----接收溢出中断标志

  UART_TX_IOV_S----发送溢出中断标志

- **UART0-RX 中断接收：**

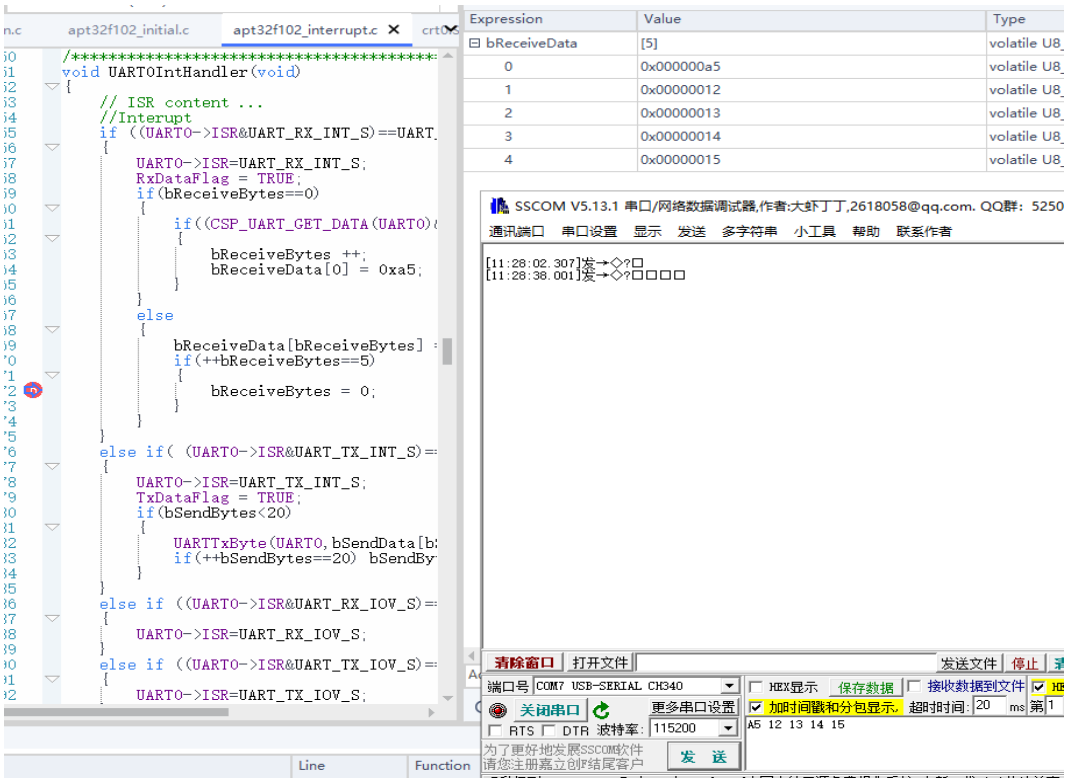

图 3.4.1　串口接收

4. 程序下载和运行

1．将目标板与仿真器连接，分别为 VDD SCLK SWIO GND

2. 主函数中发送数据，将选择的 TX 与 RX 连接串口工具。

3. 程序编译后仿真运行

4. 可以通过查看图 3.3.1、图 3.4.1 验证发送接收是否正确