USART 的使用

SAM4N 学习笔记

版本号：1.00

日期：2013/10/25

一、准备工作:

将上一节搭建的工程复制一份，命名为“4.usart”。这一节主要讲如何使用 SAM4N 的 USART 功能，实现串口的收发。

二、程序编写:

SAM4N 除了 4 个 UART，还提供了 3 个 USART，这 3 个 USART 可以配置成多种模式，支持 SPI 模式，流控模式，IrDA 红外模式，IS07816 模式，RS485 模式等，真的很强大，足以满足用户连接各种 Modem，射频卡等需求。

32.2 Embedded Characteristics

- Programmable Baud Rate Generator
- 5- to 9-bit Full-duplex Synchronous or Asynchronous Serial Communications
  - 1, 1.5 or 2 Stop Bits in Asynchronous Mode or 1 or 2 Stop Bits in Synchronous Mode
  - Parity Generation and Error Detection
  - Framing Error Detection, Overrun Error Detection
  - MSB- or LSB-first
  - Optional Break Generation and Detection
  - By 8 or by 16 Over-sampling Receiver Frequency
  - Optional Hardware Handshaking RTS-CTS
  - Receiver Time-out and Transmitter Timeguard
  - Optional Multidrop Mode with Address Generation and Detection
- RS485 with Driver Control Signal
- ISO7816, T = 0 or T = 1 Protocols for Interfacing with Smart Cards
  - NACK Handling, Error Counter with Repetition and Iteration Limit
- IrDA Modulation and Demodulation
  - Communication at up to 115.2 Kbps
- SPI Mode
  - Master or Slave
  - Serial Clock Programmable Phase and Polarity
  - SPI Serial Clock (SCK) Frequency up to Internal Clock Frequency MCK/6
- Test Modes
  - Remote Loopback, Local Loopback, Automatic Echo
- Supports Connection of:
  - Two Peripheral DMA Controller Channels (PDC)
- Offers Buffer Transfer without Processor Intervention

通过上面这段概述可以知道，USART 比上一节的 UART 要多了不少功能。

Table 32-3. I/O Lines

Instance	Signal	I/O Line	Peripheral
USART0	CTS0	PA8	A
USART0	RTS0	PA7	A
USART0	RXD0	PA5	A
USART0	SCK0	PA2	B
USART0	TXD0	PA6	A
USART1	CTS1	PA25	A
USART1	RTS1	PA24	A
USART1	RXD1	PA21	A
USART1	SCK1	PA23	A
USART1	TXD1	PA22	A
USART2	CTS2	PC17	A
USART2	RTS2	PC16	A
USART2	RXD2	PC9	A
USART2	SCK2	PC14	A
USART2	TXD2	PC10	A

由上面的表格可以看出 PA5 和 PA6 分别为 USART0 的 RXD 和 TXD，都是用的是外设功能 A，和上一节讲的 UART0 很类似。

32.7.1.1 Baud Rate in Asynchronous Mode

If the USART is programmed to operate in asynchronous mode, the selected clock is first divided by CD, which is field programmed in the Baud Rate Generator Register (US_BRGR). The resulting clock is provided to the receiver as a sampling clock and then divided by 16 or 8, depending on the programming of the OVER bit in US_MR.

If OVER is set to 1, the receiver sampling is 8 times higher than the baud rate clock. If OVER is cleared, the sampling is performed at 16 times the baud rate clock.

The following formula performs the calculation of the Baud Rate.

$$Baudrate = \frac{SelectedClock}{(8(2-Over)CD)}$$

This gives a maximum baud rate of MCK divided by 8, assuming that MCK is the highest possible clock and that OVER is programmed to 1.

从上面的描述可以看出，计算波特率的方式和 UART 有些不同。

总体上来看，USART 和 UART 有很多相似的地方，委员不同的是波特率的配置和模式配置，多了很多模式。

下面开始编写 USART0 的初始化代码。

```
/*************************************************************
* 函数名：USART0_Init()
* 参数  ：uint32_t buadrate 波特率
* 返回值：void
* 描述  ：USART0 初始化函数，在使用 USART0 前先调用
*************************************************************/
void USART0_Init(uint32_t baudrate)
{
```

```
    /*禁止外设管理控制寄存器(PMC)写保护*/
  PMC->PMC_WPMR = 0x504D4300;
  /*使能 USART1 和 PIOA 时钟*/
  PMC->PMC_PCER0 = ((1UL << ID_PIOA) |
                  (1UL << ID_USART0) );
    /*使能外设管理控制寄存器(PMC)写保护*/
  PMC->PMC_WPMR = 0x504D4301;
  /*配置 PA5 为 USART0 的 RXD，PA6 为 USART0 的 TXD*/
    PIOA->PIO_IDR=(PIO_PA5A_RXD0|PIO_PA6A_TXD0);
    PIOA->PIO_PUDR=(PIO_PA5A_RXD0|PIO_PA6A_TXD0);
    PIOA->PIO_ABCDSR[0]&=~(PIO_PA5A_RXD0|PIO_PA6A_TXD0);
    PIOA->PIO_ABCDSR[1]&=~(PIO_PA5A_RXD0|PIO_PA6A_TXD0);
    PIOA->PIO_PDR=(PIO_PA5A_RXD0|PIO_PA6A_TXD0);
           /* 复位并禁止 USART 的发送和接收*/
    USART0->US_CR = US_CR_RSTRX | US_CR_RSTTX
                  | US_CR_RXDIS | US_CR_TXDIS;
    /*配置 USART0 的波特率*/
    USART0->US_BRGR=BAUD(baudrate);
    /*定义数据位为 8bit，停止位为 1，校验位为 NONE*/
    USART0->US_MR = US_MR_USART_MODE_NORMAL|   //普通模式
                    US_MR_CHRL_8_BIT|         //数据位为 8 位
                    US_MR_NBSTOP_1_BIT|       //停止位为 1 位
                    US_MR_PAR_NO|             //校验位为 NONE
                    US_MR_CHMODE_NORMAL;      //普通通道模式
    /*禁止 DMA 通道 */
    USART0->US_PTCR = US_PTCR_RXTDIS | US_PTCR_TXTDIS;
    /*使能 USART 接收和发送*/
    USART0->US_CR = US_CR_RXEN | US_CR_TXEN;
    /*使能接收中断*/
    USART0->US_IER=US_IER_RXRDY;
  /*配置 USART0 的先占优先级为 1，从优先级为 1*/
  NVIC_SetPriority(USART0_IRQn, ((0x01<<3)|0x01));
    /*使能 USART0 的中断通道*/
   NVIC_EnableIRQ(USART0_IRQn);
}
```

从代码上来看，和 UART 的配置一样，就是寄存器的名字变了一点，模式配置那里多了数据位，停止位的配置，其他基本就是一样的。

下面是接收和发送代码：

```
/*************************************************************
* 函数名：USART0_Handler()
* 参数  ：void
* 返回值：void
* 描述  ：USART0 中断服务函数
*************************************************************/
void USART0_Handler(void)
 {
     uint8_t temp;
     if((USART0->US_CSR& US_CSR_RXRDY)==1)
     {    //接收数据中断
      temp= USART0->US_RHR&0xff;         //接收一个字节
      USART0_SendByte(temp);             //将接收的数据发回
   }
 }

/*************************************************************
* 函数名：USART0_SendByte()
* 参数  ：uint8_t c  要发送字符
* 返回值：void
* 描述  ：USART0 发送一个字符函数
*************************************************************/
void USART0_SendByte(uint8_t c)
{   /*等待发送缓冲器为空*/
   while((USART0->US_CSR & US_CSR_TXEMPTY) == 0);
   USART0->US_THR=c;        //将发送字符写入发送保持寄存器
}

/*************************************************************
* 函数名：USART0_SendString()
* 参数  ：uint8_t *s  指向字符串的指针
* 返回值：void
* 描述  ：USART0 发送字符串函数
*************************************************************/
void USART0_SendString(uint8_t *s)
{
  while(*s)
  {
   USART0_SendByte(*s);    //发送指针当前所指的字节
   s++;
```

```
  }
}
```

在 main.c 中写个简单的测试程序，如下：

```
int main(void)
{
 systick_hw_init();
 led_hw_init();
 USART0_Init(115200);
 USART0_SendString("hello,this is a usart demo!\r\n");
 while(1){
USART0_SendString("hello,I am SAM4N!\r\n");
 led_hw_on();
 delay_ms(500);
 led_hw_off();
 delay_ms(500);
 }
}
```

找一条串口线接到 PA5 和 PA6 上就可以看到输出了，效果和上一节一样。注意串口线要用 TTL 电平的哦。