

# UART 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“3. uart”。这一节主要讲如何使用 SAM4N 的 UART 功能，实现串口的收发。

二、程序编写：

细心看数据手册的朋友也许已经发现了，SAM4N 有 4 个 UART，还有 3 个 USART 哦，如果都配置成串口，那就足足有 7 个可用的串口了。也许很多人就疑惑了，UART 和 USART 有啥区别啊？其实细节上我也不太懂有多少区别，看了数据手册，大概就明白 USART 可用工作在 SPI 模式，可用使用硬件流控，可用设置不同数据位和停止位，功能比 UART 要强很多，UART 不支持硬件流控，不支持 SPI 模式，不支持数据位和停止位编程。好了，下面咱就去试试 UART 是怎么用的吧。

打开原理图，可用看到有如下电路：



是的，这里说明板子上的 USB CDC 虚拟串口设备连接在了 SAM4N 的 PA10 和 PA9 上，而这两个正是 UART0 的 UTXD 和 URXD。

Table 31-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
UART0	URXD0	PA9	A
UART0	UTXD0	PA10	A
UART1	URXD1	PB2	A
UART1	UTXD1	PB3	A
UART2	URXD2	PA16	A
UART2	UTXD2	PA15	A
UART3	URXD3	PB10	B
UART3	UTXD3	PB11	B

通过上面这个表可用看出，UART0 的 URXD0 是 PA9 的外设 A 功能，UTXD0 是 PA10 的外设 A 功能。

要使用 UART0，首先需要对它进行初始化配置，代码如下：

```
/*
*****
* 函数名: UART0_Init()
* 参数   : uint32_t baudrate 波特率
* 返回值: void
* 描述   : UART0 初始化函数，在使用 UART0 前先调用
*****
*/
```

```

void UART0_Init(uint32_t baudrate)
{

    uint32_t Fdiv =0;
    /*禁止外设管理控制寄存器(PMC)写保护*/
    PMC->PMC_WPMR = 0x504D4300;
    /*使能 UART1 和 PIOA 时钟*/
    PMC->PMC_PCER0 = ((1UL << ID_PIOA) |
                      (1UL << ID_UART0));
    /*使能外设管理控制寄存器(PMC)写保护*/
    PMC->PMC_WPMR = 0x504D4301;
    /*配置 PA9 为 UART0 的 RXD， PA10 为 UART0 的 TXD*/
    PIOA->PIO_IDR=(PIO_PA9A_URXD0|PIO_PA10A_UTXD0);
    PIOA->PIO_PUDR=(PIO_PA9A_URXD0|PIO_PA10A_UTXD0);
    PIOA->PIO_ABCDSR[0]&=~(PIO_PA9A_URXD0|PIO_PA10A_UTXD0);
    PIOA->PIO_ABCDSR[1]&=~(PIO_PA9A_URXD0|PIO_PA10A_UTXD0);
    PIOA->PIO_PDR=(PIO_PA9A_URXD0|PIO_PA10A_UTXD0);
    /* 复位并禁止 UART 的发送和接收*/
    UART0->UART_CR = UART_CR_RSTRX | UART_CR_RSTTX
                  | UART_CR_RXDIS | UART_CR_TXDIS;
    /*配置 UART0 的波特率*/
    Fdiv = (SystemCoreClock/ baudrate) /UART_MCK_DIV ;
    if (Fdiv < UART_MCK_DIV_MIN_FACTOR || Fdiv > UART_MCK_DIV_MAX_FACTOR)
        return ;
    UART0->UART_BRGR=Fdiv;
    /*定义数据位为 8bit，停止位为 1，校验位为 NONE*/
    UART0->UART_MR = UART_MR_PAR_NO|UART_MR_CHMODE_NORMAL;
    /*禁止 DMA 通道 */
    UART0->UART_PTCR = UART_PTCR_RXTDIS | UART_PTCR_TXTDIS;
    /*使能 UART 接收和发送*/
    UART0->UART_CR = UART_CR_RXEN | UART_CR_TXEN;
    /*使能接收中断*/
    UART0->UART_IER=UART_IER_RXRDY;
    /*配置 UART0 的先占优先级为 1，从优先级为 1*/
    NVIC_SetPriority(UART0_IRQn, ((0x01<<3)|0x01));
    /*使能 UART0 的中断通道*/
    NVIC_EnableIRQ(UART0_IRQn);
}

```

第一步就是使能 PIOA 和 UART0 的时钟，然后配置 PIOA9 和 PIOA10 连接到外设 A 功能，也就是 URXD0 和 UTXD0 功能，然后禁止 PIOA9 和 PIOA10 的 GPIO 功能，这样 PIOA9 和 PIOA10 就配置成了 URXD0 和 UTXD0。

接下来配置 UART，首先复位 UART0 并禁止接收和发送，接着计算波特率分频值，数据手册给出如下计算公式，按照这个计算就好。

$$\text{Baud Rate} = \frac{\text{MCK}}{16 \times \text{CD}}$$

接下来配置 UART 的模式，这里选择普通模式，校验位为 NONE，然后禁止 DMA 功能，因为没有用 DMA 传输。

最后使能 UART 的收发，使能中断，这样算是完成了 UART 的配置了。

完成配置还不行，还需要一个接收和发送函数，代码如下：

```

/*****
* 函数名: UART0_Handler()
* 参数   : void
* 返回值: void
* 描述   : UART0 中断服务函数
*****/
void UART0_Handler(void)
{
    uint8_t temp;
    if((UART0->UART_SR & UART_SR_RXRDY) == 1)
    {
        temp = UART0->UART_RHR & 0xff;           //接收数据中断
        UART0_SendByte(temp);                     //接收一个字节
                                                //将接收的数据发回
    }
}

/*****
* 函数名: UART0_SendByte()
* 参数   : uint8_t c 要发送字符
* 返回值: void
* 描述   : UART0 发送一个字符函数
*****/
void UART0_SendByte(uint8_t c)
{
    while((UART0->UART_SR & UART_SR_TXEMPTY) == 0); //等待发送缓冲器为空
    UART0->UART_THR = c;                             //将发送字符写入发送保持寄
存器
}

/*****
* 函数名: UART0_SendString()
* 参数   : uint8_t *s 指向字符串的指针
* 返回值: void
* 描述   : UART0 发送字符串函数
*****/
void UART0_SendString(uint8_t *s)
{

```

```

        while(*s)                                //判断是否到字符串
末尾
    {
        UART0_SendByte(*s);                      //发送指针当前所指的字节
        s++;                                      //地址加 1
    }

}

```

这里我们使用的中断接收方式，因为接收的数据只是用来实验是否有接收到，所有这里是收到后再发回去。接收时要读取 UART 状态寄存器 UART\_SR 的值，看 UART\_SR\_RXRDY 位是否被置位，如果置位说明有数据接收。发送时要先检测 UART 的发送数据寄存器 UART\_THR 是不是为空。

好了，这样就完成 UART0 的驱动了，接下来按这样的方法去配置其他 UART 就都可以用了。

接下来还需要去实现 int fputc(int ch, FILE \*f) 这个函数，试 printf 的打印输出到 UART0 去，这样就可以在程序中使用 printf() 函数打印消息了，代码如下：

```

int fputc(int ch, FILE *f)
{
    UART0_SendByte((uint8_t)ch);                //发送 1 个字节
    return ch;
    //返回 ch
}

```

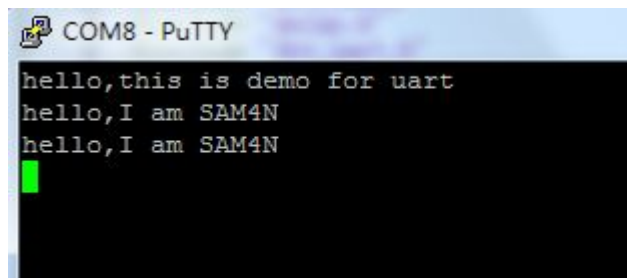
在 main.c 中写个简单的测试程序，如下：

```

int main(void)
{
    systick_hw_init();
    led_hw_init();
    UART0_Init(115200);
    UART0_SendString("hello,this is demo for uart\r\n");
    while(1){
        printf("hello,I am SAM4N\r\n");
        led_hw_on();
        delay_ms(500);
        led_hw_off();
        delay_ms(500);
    }
}

```

下载程序，打开串口调试工具，就可以看到板子的串口输出了，如下图：



```
COM8 - PuTTY
hello,this is demo for uart
hello,I am SAM4N
hello,I am SAM4N
█
```

注意：使用 printf 需要把 MicroLIB 勾上，如下图：

