STM32F030-UART1_DMA 使用提示

前言：

今天把 STM32F030C8T6 的串口 DMA 学习了一下，为了加快各位研发人员的开发进度，避免浪费大量的时间在硬件平台上，写出个人代码调试的经验。个人水平有限，如有错误，还请指正 *mr.li.ming@qq.com*。

提示：使用的内部 RC 时钟，最大速度 48MHz;使用 USART1-PA9/PA10.

第一步：初始化端口

```
/*******************************************************************************
  * @brief   串口 1 端口初始化
  * @param   None
  * @retval None
*************************************************************Author:Liming**/
void USART1_GPIO_Init(void)
{
    GPIO_InitTypeDef      GPIO_Initstructure;

    RCC_AHBPeriphClockCmd(USART1_GPIO_CLK,ENABLE);

    /* Connect pin to Periph */
    GPIO_PinAFConfig(GPIOA, GPIO_PinSource9, GPIO_AF_1); // 注意这里是 GPIO_PinSource9
    GPIO_PinAFConfig(GPIOA, GPIO_PinSource10, GPIO_AF_1);

    GPIO_Initstructure.GPIO_Pin=USART1_TX_PIN;
    GPIO_Initstructure.GPIO_Mode=GPIO_Mode_AF;
    GPIO_Initstructure.GPIO_OType=GPIO_OType_PP;   // 推挽输出
    GPIO_Initstructure.GPIO_PuPd=GPIO_PuPd_UP;
    GPIO_Initstructure.GPIO_Speed=GPIO_Speed_50MHz;
    GPIO_Init(USART1_GPIO_PORT,&GPIO_Initstructure);

    GPIO_Initstructure.GPIO_Pin = USART1_RX_PIN;   // 浮空输入
    GPIO_Init(USART1_GPIO_PORT,&GPIO_Initstructure);
}
```

第二步：初始化 UART1

```
/*******************************************************************************
  * @brief  串口 1 初始化
  * @param  None
  * @retval None
*************************************************************Author:Liming**/
void USART1_Init(uint32_t BaudRate)
{
    USART_InitTypeDef   USART_Initstructure;

    RCC_APB2PeriphClockCmd(USART1_CLK,ENABLE);

    USART1_GPIO_Init(); // 调用了上面的端口初始化，故主函数里调用此函数即可。

    USART_Initstructure.USART_BaudRate = BaudRate;
    USART_Initstructure.USART_Parity    =USART_Parity_No;
    USART_Initstructure.USART_WordLength =USART_WordLength_8b;
    USART_Initstructure.USART_StopBits  =USART_StopBits_1;
    USART_Initstructure.USART_Mode      = USART_Mode_Rx|USART_Mode_Tx;
    USART_Initstructure.USART_HardwareFlowControl =USART_HardwareFlowControl_None;
    USART_Init(USART1,&USART_Initstructure);

    USART_ClearFlag(USART1,USART_FLAG_TC);
    USART_DMACmd(USART1,USART_DMAReq_Tx,ENABLE);
    USART_Cmd(USART1,ENABLE);          // 使能串口
}
```

第三步：DMA1 中断配置

```
/*******************************************************************************
  * @brief   DMA1 中断配置
  * @param   None
  * @retval None
*******************************************************************Author:Liming**/
void DMA1_NVIC_Init(void)
{
    NVIC_InitTypeDef       NVIC_InitStructure;

    NVIC_InitStructure.NVIC_IRQChannel = DMA1_Channel2_3_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_InitStructure.NVIC_IRQChannelPriority = 2;

    NVIC_Init(&NVIC_InitStructure);
}
```

注意事项：

1. **USART1** 发送数据 使用的是 **DMA1** 的第二通道。查表可知，为什么还有第四通道呢，那是给 **USART1** 端口重映射了之后使用的。

Table 1. Peripherals served by DMA1 and channel allocation

Peripherals		CH1	CH2	CH3	CH4	CH5
ADC	ADC1	ADC1	ADC1			
SPI	SPI1		SPI1_RX	SPI1_TX		
	SPI2				SPI2_RX	SPI2_TX
USART	USART1		USART1_TX	USART1_RX	USART1_TX	USART1_RX
	USART2				USART2_TX	USART2_RX
I^2C	I^2C1		I2C1_TX	I2C1_RX		
	I^2C2				I2C2_TX	I2C2_RX
TIM	TIM1		TIM1_CH1	TIM1_CH2	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP TIM1_CH3
	TIM2	TIM2_CH3	TIM2_UP	TIM2_CH2	TIM2_CH4	TIM2_CH1
	TIM3		TIM3_CH3	TIM3_CH4 TIM3_UP	TIM3_CH1 TIM3_TRIG	
	TIM6/DAC			TIM6_UP DAC		
	TIM15					TIM15_CH1 TIM15_UP TIM15_TRIG TIM15_COM
	TIM16			TIM16_CH1 TIM16_UP	TIM16_CH1 TIM16_UP	
	TIM17	TIM17_CH1 TIM17_UP	TIM17_CH1 TIM17_UP			

第四步：DMA1 配置

```
/*******************************************************************************
  * @brief   DMA1 配置
  * @param   None
  * @retval None
*************************************************************Author:Liming**/
void DMA1_Init(void)
{
    DMA_InitTypeDef DMA_InitStructure;
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1,ENABLE);

    DMA_InitStructure.DMA_BufferSize = 12;   // 缓存大小
    DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;      // 内存到内存关闭
    DMA_InitStructure.DMA_Mode = DMA_Mode_Normal;     // 普通模式
    DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralDST;   // 内存到外设
    DMA_InitStructure.DMA_Priority = DMA_Priority_High; // DMA 通道优先级
    DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;// 内存地址递增
    DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)&USART1->TDR;     // 外设地址
    DMA_InitStructure.DMA_PeripheralInc =    DMA_PeripheralInc_Disable;// 外设地址不变
    DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte; // 内存数据长度
    DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t)UART1_TXBUFFER;     // 定义内存基地址
    DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte;//外设数据长度

    DMA_Init(DMA1_Channel2,&DMA_InitStructure);
    DMA_ClearITPendingBit(DMA1_IT_TC2); // 清除一次 DMA 中断标志
    DMA_ITConfig(DMA1_Channel2,DMA_IT_TC,ENABLE);// 使能 DMA 传输完成中断

    DMA1_NVIC_Init(); // 调用了上面的中断配置，故主函数里调用此函数即可
    DMA_Cmd(DMA1_Channel2,ENABLE);
    }
```

注意事项：

1. 缓存大小：就是你一次要发送多长的数据。
2. **DMA** 方向：因为是串口发送数据，所以是从内存到外设，**USART1** 对于单片机来讲是个外设。定义的发送数组是内存。
3. 内存地址递增：其实不难理解，从发送数组的 **UART1_TXBUFFER[0]-UART1_TXBUFFER[n]**肯定是递增的。
4. 外设地址不递增：所有的数据都是通过串口发送寄存器发出去，所以外设地址不变。
5. 内存/外设数据长度：串口发送的数据都是字节为单位，所以长度是 **Byte**
6. **DMA_ITConfig(DMA1_Channel2,DMA_IT_TC,ENABLE);**注意这一句不要写错。

第五步：DMA1 的中断处理函数

```
/**
  * @brief   DMA1_Channel1 中断服务函数
  * @param   无
  * @retval 无
  */
void DMA1_Channel2_3_IRQHandler(void)
{
    /*判断 DMA 传输完成中断*/
    if(DMA_GetITStatus(DMA1_IT_TC2) != RESET)
    {
        UART1_STATE = 2;// send over
    }
    /*清除 DMA 中断标志位*/
    DMA_ClearITPendingBit(DMA1_IT_TC2);
}
```

这里使用了一个变量 UART1_STATE 作为标志位

第六步：使用 DMA1 发送串口数据

```
USART1_Init(115200);
DMA1_Init();

while(1)
{
    if(UART1_STATE==2)
    {
        UART1_STATE = 1;
        DMA_Cmd(DMA1_Channel2,DISABLE); //  发送完成先关掉 DMA 通道
        DMA_SetCurrDataCounter(DMA1_Channel2,12); //  设置需要发送的长度
        DMA_Cmd(DMA1_Channel2,ENABLE); //  再打开 DMA 通道
    }
    GPIO_SetBits(GPIOA,GPIO_Pin_4);Delay(500);
    GPIO_SetBits(GPIOA,GPIO_Pin_3);Delay(500);
    GPIO_ResetBits(GPIOA,GPIO_Pin_4);Delay(500);
    GPIO_ResetBits(GPIOA,GPIO_Pin_3);Delay(500);

}
```

注意事项：

1. 一定要注意，DMA 的传输有个长度计数器，DMA 传输完成后，计数器里的值就变成了 0；数据是不传了，但是通道并没有关闭。所以想要再次传输就需要修改这个长度计数器的值，但是这个值的修改必须要关闭通道后修改。所以就有了上面的步骤，关闭通道—修改计数值—打开通道

希望对各位看官有所帮助，并能触类旁通，对于外设到内存啊，内存到内存啊，ADC 与 DMA 啊，SPI 与 DMA 都能轻松的应用。