

DMA任务

知识学习

- 为什么要用DMA? 有什么优势?

DMA是用于外设与存储器或存储器与存储器之间的高速数据传输。传输过程中由DMA控制器来控制，没有CPU参与，节省CPU资源，提高时间效率。

- 如何配置串口DMA接收?

The screenshot displays the STM32CubeMX software interface for configuring a USART2 peripheral. The top navigation bar includes 'Pinout & Configuration' and 'Clock Configuration'. The left sidebar shows a list of peripherals, with USART2 selected. The main area shows the 'USART2 Mode and Configuration' window, where the 'DMA Settings' tab is active. The table below shows the configuration for USART2_RX and USART2_TX. The bottom panel shows the 'Project Manager' tab.

DMA Request	Stream	Direction	Priority
USART2_RX	DMA1 Stream 5	Peripheral To Mem...	Low

DMA Request	Stream	Direction	Priority
USART2_RX	DMA1 Stream 5	Peripheral To Memory	Low
USART2_TX	DMA1 Stream 6	Memory To Peripheral	Low

DMA Request Settings

Peripheral		Memory
Mode	Normal	Increment Address <input checked="" type="checkbox"/>
Use Fifo <input type="checkbox"/>	Threshold	Data Width: Byte
		Burst Size

对串口2的RX和TX进行相应参数的配置，配置参数有传输模式选为普通模式，地址递增选为外设不递增，内存器递增，FIFO使用为不是能，数据宽度选为字节。

- 配置DMA时各个参数的意义？
 - Mode选为Normal是DMA的传输模式选为普通模式，普通模式是在某次数据传输结束后，DMA通道就关闭，在下次需要进行数据传输时再重新启用DMA
 - 地址模式选为外设不递增，内存器递增，此处未找到相关解释
 - FIFO使用为不使能
 - 数据宽度选为字节，由于我们发送的数据均是以字节为单位，故选择字节

实现串口不定长收发

串口空闲中断+DMA接收

1.按照上面配置DMA的方式配置串口2

2.代码编写

```
#define LENGTH 100 //接收缓冲区大小
/* USER CODE END PM */
uint8_t RxBuffer[LENGTH]; //接收缓冲区
uint8_t RecCount = 0; //接受数据个数
uint8_t flag = 0; //接收完成标志：0表示未接受完成，1表示接收完成
```

首先对于传输数据需要的几个变量进行宏定义

```
void USART2_IRQHandler(void)
{
    USER CODE BEGIN USART2_IRQn 1 */
    if( __HAL_UART_GET_FLAG(&huart2, UART_FLAG_IDLE) != RESET) //判断是否发生IDLE
    {
        __HAL_UART_CLEAR_IDLEFLAG(&huart2); //清除IDLE中断标志
        HAL_UART_IdleCpltCallback(&huart2); //中断回调函数
    }
}
```

在 `USART2_IRQHandler(void)` 函数中添加对IDLE中断，即空闲中断的判断，判断空闲中断发生后，调用中断回调函数

```
void HAL_UART_IdleCpltCallback(UART_HandleTypeDef *huart2)
{
    flag = 1; //设置接收标志位
}
```

回调函数中将flag数据接收标志位设为1，表示接收到数据

```
if(flag == 1) //判断数据是否完成接受
{
    flag = 0; //清除标志位
    RecCount = LENGTH - __HAL_DMA_GET_COUNTER(&hdma_usart2_rx); //已接收数据等于
    数据总量减去DMA数据流中待接收数据
```

```

        HAL_UART_Transmit_DMA(&huart2, (uint8_t*)RxBuffer, RecCount); //采用DMA方式将
接受的数据原样发回PC
        RecCount = 0;
        __HAL_DMA_DISABLE(&hdma_usart2_rx); //设置DISABLE，触发DMA中断回调函数来重启一
下下一次的DMA接受
    }

```

在mian函数中算出接收到的数据并原样发回PC，同时调用DMA中断调用函数来重新出发DMA以准备下一次的数据传输

```

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart2)
{
    if(huart2->Instance == USART2)
    {
        HAL_UART_Receive_DMA(huart2, (uint8_t*)RxBuffer, LENGTH); //重新启动DMA接收，准备下
一次数据传输
    }
}

```

该函数为DMA中断调用回调函数，由于重新启动DMA。

定时器TIM

知识学习

- 定时器的分类以及区别

根据定时器所处位置可以分为外设定定时器和内核定时器，内核定时器指的是系统节拍定时器，而外设定定时器根据定时器到的功能又可以分为常规定时器与专用定时器，常规定时器则有基本计数器，通用定时器，高级定时器3种定时器，而专门定时器由看门狗定时器，实时时钟和低功耗定时器。

常规定时器的几种定时器的区别：

基本定时器：几乎没有任何输入输出通道，常用作时基，实现基本的定时和计数功能

通用定时器：具有多路捕获和比较通道，可以完成定时，计数，输入捕获，输出比较等功能

高级定时器：功能最多，在有通用定时器功能的1基础上，还有带死区控制的互补信号输出、紧急刹车关断输入等功能

- 自己板子中的基本定时器，通用定时器和高级定时器分别有哪些？

我使用的是STM32F411CEU6，有5个16位通用定时器，5个32位通用定时器，为TIM2/5、TIM3/4、TIM9，TIM10/TIM11,1个16位高级定时器，为TIM1

- 解释时钟树结构：

- 系统时钟SYSCLK可以有哪些时钟源提供

SYSCLK时钟源有三个来源：HSI RC、HSE OSC、PLL

- 如何由系统时钟SYSCLK 得到总线时钟HCLK？

系统时钟 SYSCLK 经过 AHB 预分频器分频之后得到时钟叫 APB 总线时钟，即 HCLK

- STM32内部时钟有哪些时钟走总线？每条总线上挂载那些外设？

- 解释概念

- 定时器时钟源

时钟信号的产生之处，有内部时钟CK_INT,外部输入引脚CHx，外部触发信号ETR和内部触发信号ITRx，当时钟源选为内部时钟时，定时器工作为定时模式，选为两个外部时钟时，工作在计时模式，选为内部触发模式时，工作为从模式。

- 定时器时钟

当定时器时钟源选择为内部时钟时，定时器时钟等于预分频时钟CK_PSC，当定时器的工作模式为定时模式，此时内部时钟CK_INT就是定时器时钟TIM_CLK。

- 计数器

计时器是对周期不确定的脉冲信号进行计数，如MCU的I/O引脚的外部信号，定时器是特殊的计时器

- 自动重载寄存器

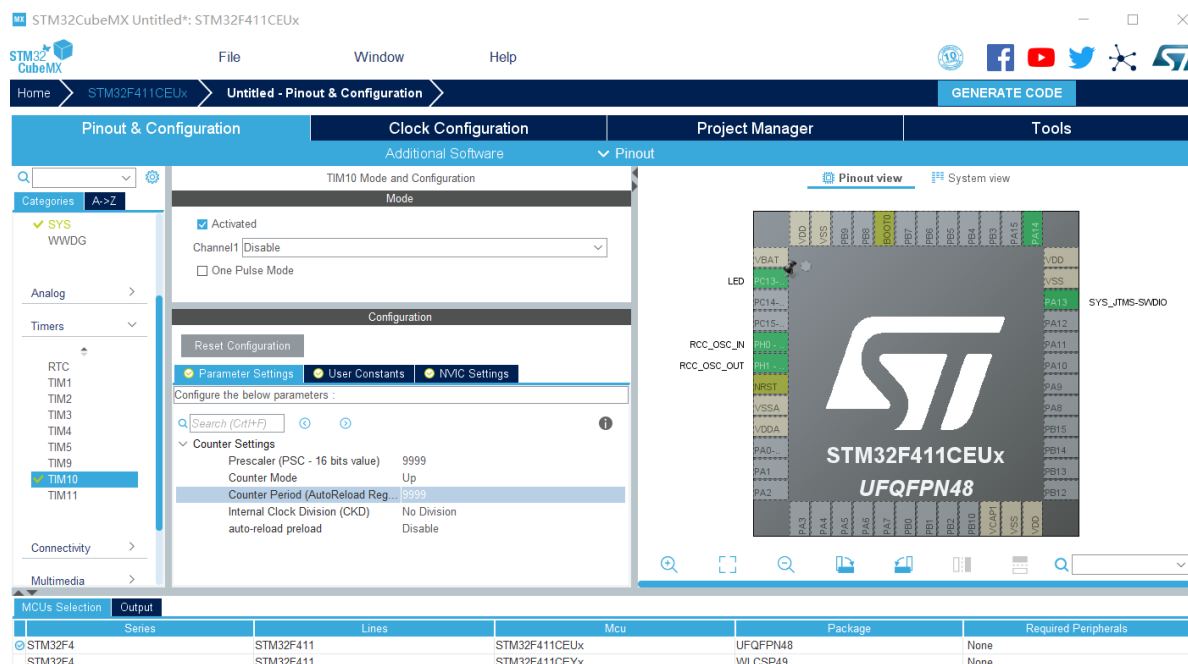
自动重载寄存器组成自动重载模块，由于记录核心寄存器的起始或终止值，确定计数的溢出值，当计数模式为递增时，寄存器中的值作为核心寄存器的计数终值，当计数模式为递减时，寄存器中的值作为核心寄存器的计数初值。

- 如何计算定时时间？

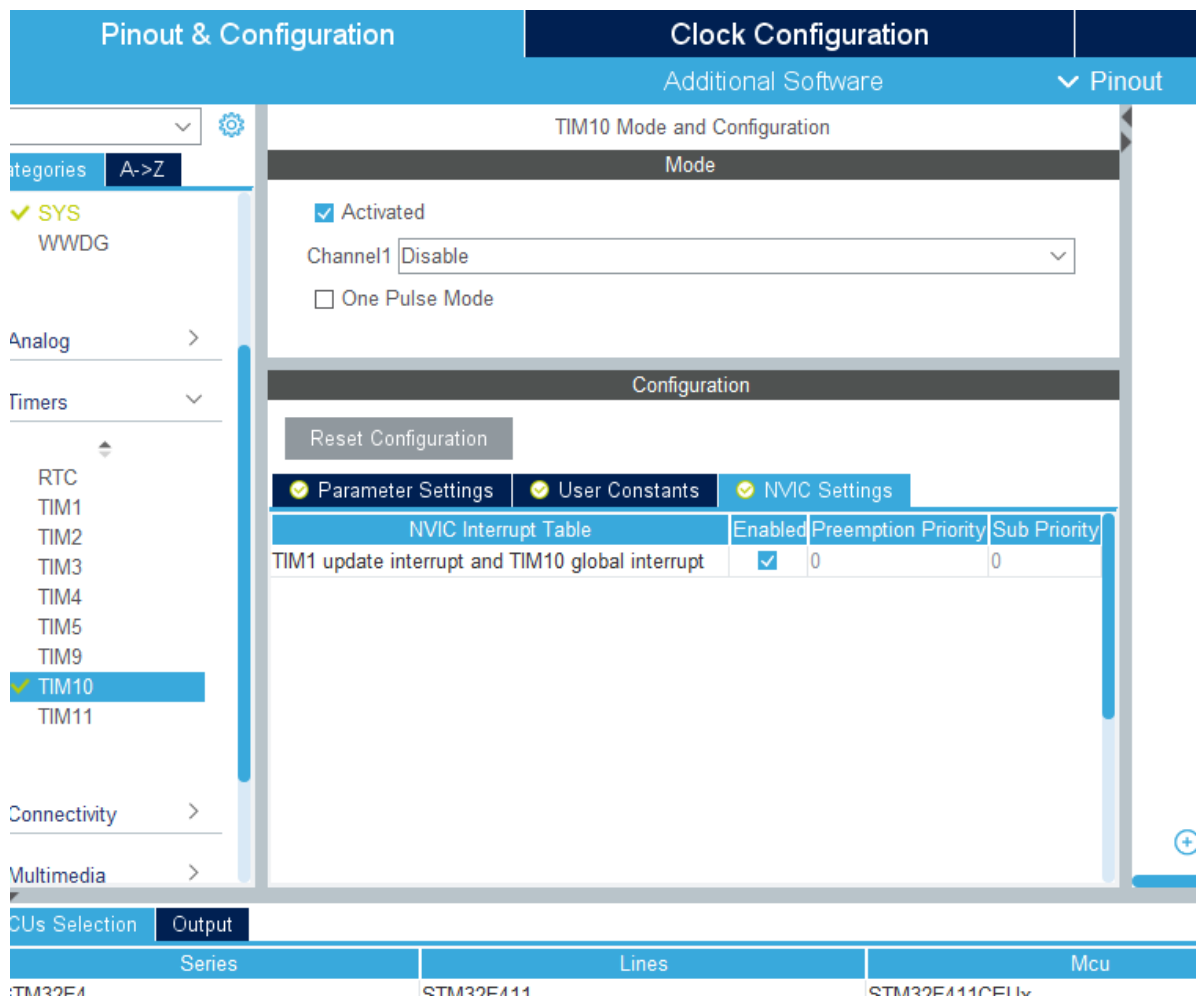
定时时间=计数值x计数周期，通过此公式来计算时间，也可以通过 定时时间=计数值/时钟频率来计算时间。计数值等于自动重载值ARR+1，时钟频率为计数频率，等于预分频时钟CK_PSC除以分频系数PSC+1，分别得到计数值和时钟频率后即可计算得到定时时间。

实验

1.配置时钟TIM10



MCUs Selection	Output			
Series	Lines	McU	Package	Required Peripherals
STM32F4	STM32F411	STM32F411CEUx	UFQFPN48	None
STM32F4	STM32F411	STM32F411CEYx	WLCSP49	None



时钟频率我设为100MHz，根据前面的时间计算公式可知，要计时1s的时间，就需要让 $(PSC+1) \times (ARR+1)$ 等于100M，故这里设置PSC=9999，ARR=9999.需要注意的是在设置PSC和ARR时要注意它们的最大取值。

2.代码分析

```
static void MX_TIM10_Init(void); //定时器10的初始化
```

先对TIM10初始化

```
__HAL_TIM_CLEAR_IT(&htim10, TIM_IT_UPDATE); //清除定时器初始化过程中的更新中断标志，避免定时器一起动就进入中断
```

```
HAL_TIM_Base_Start_IT(&htim10); //使能定时器10更新中断并启动定时器
```

为了防止时钟一初始化就马上触发中断，要清除更新中断标志，同时启动定时器

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if( htim->Instance == TIM10) // 判断发生更新中断的定时器
    {
        HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin); //翻转LED
    }
}
```

中断回调函数里面是对LED进行状态翻转。

PWM

知识学习

- 什么是脉冲宽度调制(PWM)?

PWM是一种对模拟信号电平进行数字编码的方法。PWM信号是通过对一系列脉冲的宽度进行调制，等效出所需要的波形，包含形状以及幅值，对模拟信号电平进行数字编码，也就是说通过调节占空比的变化来调节信号、能量等的变化。

- 定时器在PWM输出模式下是如何产生PWM波的

PWM输出主要与3个寄存器有关，自动重载寄存器ARR,捕获/比较寄存器CCR和计数寄存器CNT，PWM通过通道CHx输出波形，初始值为高电平，CNT为递增计数，当CNT的值大于CRR时，输出波形为低电平，当CNT的值到达ARR时，CNT的值归零，输出波形为高电平，如此反复，输出整个的PWM。其中ARR控制PWM信号的周期，而CNT控制PWM的占空比。

- 如何产生特定频率，占空比的PWM波? (列出PWM波输出频率和占空比的计算公式)

通过控制ARR和CNT的值来控制PWM信号的周期与占空比。

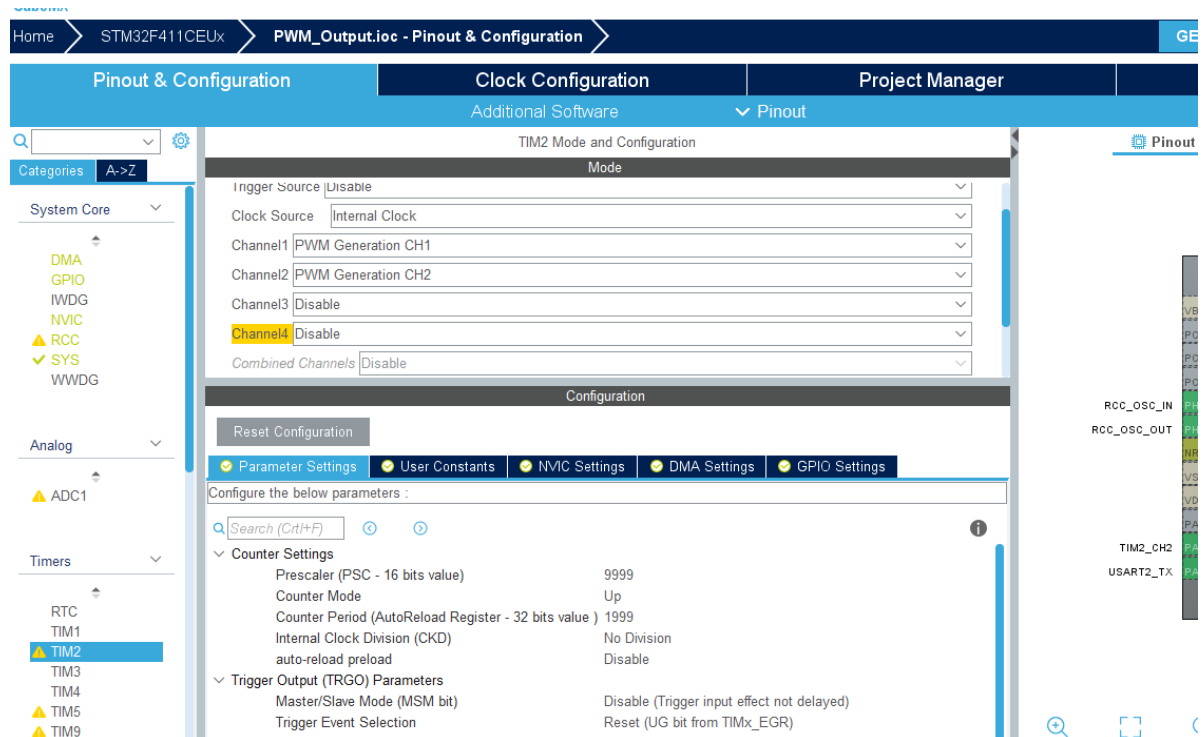
周期计算公式: $T = (ARR + 1) \times (PSC + 1) / TIM_CLK$

占空比计算公式: $D = (CRR / (ARR + 1)) \times 100\%$

实验

实验分析: 首先通过前面的不定长传输来接受所要求的指令，此处为了方便，将输入内容定为'100 40'此类形式，其中100位频率，单位为Hz，40指占空比40%，其中通过一个空格来区分。先通过DMA方式来实现不定长接收信息，在成功接收信息后，对接收的字符串进行判断处理，得到所要求的频率数值和占空比数值，然后输出相应的PWM波形。

1.PWM配置



配置时钟TIM2，设置好相关参数，并选择好通道1与通道2的PWM的输出。

DMA的配置与上面任务所写的步骤一致。

2.代码

```
uint8_t RxBuffer[LENGTH];//接收缓冲区
```

```
uint8_t RecCount = 0;//接受数据人数
```

```
uint8_t flag = 0;//接收完成标志：0表示未接受完成，1表示接收完成
```

```
uint8_t crr =0;//CRR改变占空比
```

此处crr为捕获/比较寄存器所储存的值，通过识别占空比并根据前面的计算公式计算出crr

```
if(flag == 1) //判断数据是否完成接受
```

```
{
```

```
    int i;
```

```
    flag = 0; //清除标志位
```

```
    RecCount = LENGTH - __HAL_DMA_GET_COUNTER(&hdma_usart2_rx);//已接收数据等于
```

数据总量减去DMA数据流中待接收数据

```
    HAL_UART_Transmit_DMA(&huart2, (uint8_t*)RxBuffer, RecCount);//采用DMA方式将
```

接受的数据原样发回PC

```
    RecCount = 0;
```

```
    int fre = 0;//频率
```

```
    int duty = 0;//占空比
```

```
    for(i = 0; RxBuffer[i] != ' '; i++)//识别出频率
```

```
{
```

```
        fre = 10*arr + (int)RxBuffer[i];
```

```
}
```

```
    arr = 10000/fre -1;
```

```
    for(i++; i < RecCount; i++)//识别出占空比
```

```
{
```

```
        duty = 10*duty + (int)RxBuffer[i];
```

```
}
```

```
    crr = duty*(arr+1)/100;
```

```
    __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, crr);//改通道1的占空比
```

```
    __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_2, crr);//改通道2的占空比
```

```
    HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);//启动定时器2的通道1输出PWM信号
```

```
    HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_2);//启动定时器2的通道2输出PWM信号
```

```
    __HAL_DMA_DISABLE(&hdma_usart2_rx);//设置DISABLE，触发DMA中断回调函数来重启一
```

下下一次的DMA接受

```
}
```

此处为代码主要内容，首先得到接收数据长度RecCount，接着通过根据' '位置用一个for循环得到频率fre，然后根据公式 $T = (ARR + 1) \times (PSC + 1) / TIM_CLK$ （其中PSC=9999，TIM_CLK为100M），可得 $arr = 10000 / fre - 1$ 。同理，得到占空比duty，根据公式 $D = (CRR / (ARR + 1)) \times 100\%$ 可得 $crr = duty \times (arr + 1) / 100$ 。用 __HAL_TIM_SET_COMPARE 函数修改好占空比，然后 HAL_TIM_PWM_Start 函数将PWM信号输出。

其余中断内容均与DMA数据不定长接收任务的一致。