**定时器：PWM产生与捕获的实验**

1. **实验相关的硬件说明**

本实验，我们将使用pixhawk飞控板上的协处理器（STM32F100C8T6）完成。利用两个通道的定时器，一个定时器通道产生PWM波，另一个定时器通道捕获产生的PWM波的频率和占空比。

pixhawk飞控板协处理器引出了多路定时器，用来产生PWM波驱动舵机、动力电机等外设。相关的实物图如图1. 所示。

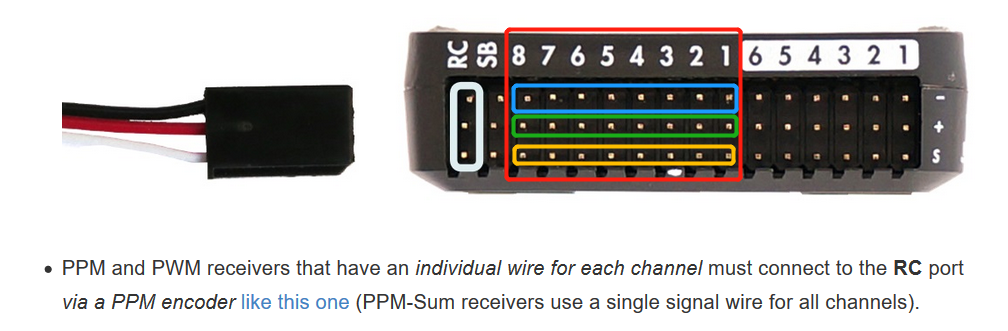


图1. 协处理器引出的定时器端口

红框框出来的就是协处理器引出的定时器端口，总共有8路。图中蓝框圈出来的全部是GND关口，绿框圈出来的是VCC端口，黄框圈出来的是定时器端口。下面查询原理图，和PCB工程图，可以找到这8路定时器端口，分别对应的是哪个定时器的哪个通道。如图2. 所示。

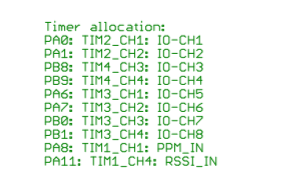
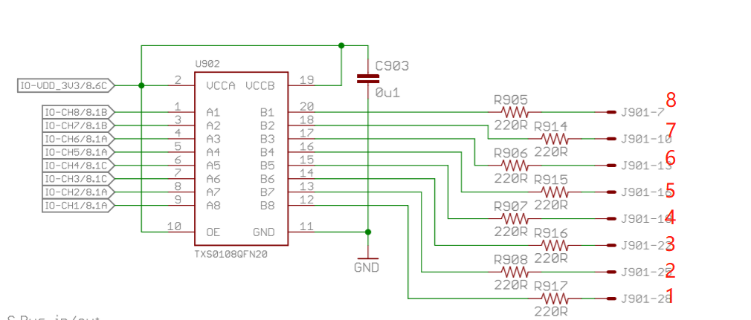


图2. 端口对应的定时器

图2中左图中的序号分别对应图1. 中的8路定时器。图2右图为飞控板原理图中的注释，这样我们就可以确切地知道，8路定时器端口分别对应的是哪个定时器的那个通道。例如：第8路端口，对应的就是TIM3\_CH4，即定时器3的第4通道；第7路端口，对应的就是TIM3\_CH3，即为定时器3的第3通道。

本实验使用定时器3的4通道产生PWM波，使用定时器4的1,2通道来捕获PWM波。

1. **工程配置**
2. **利用定时器产生PWM波**
   1. **原理说明**

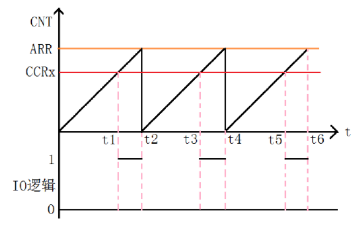


图3. PWM原理说明

如图3. 所示，IO逻辑对应的就是PWM波，CCRx为计数寄存器的值，ARR为自动重装载寄存器的值。图中所示即为定时器向上计数的PWM波的产生方式，在定时器的计数器向上计数到CCRx之前，IO口一直输出低电平，从到达CCRx后，开始输出高电平。再到ARR的值后，产生溢出，重新计数，如此循环，产生周期性交替的高低电平，即为PWM波。

所以显然，如果改变CCRx的值得大小，就可以改变PWM波形的占空比；改变ARR的值，就可以改变PWM波形的频率。

* 1. **配置说明**

首先是时钟配置，如图4. 所示：

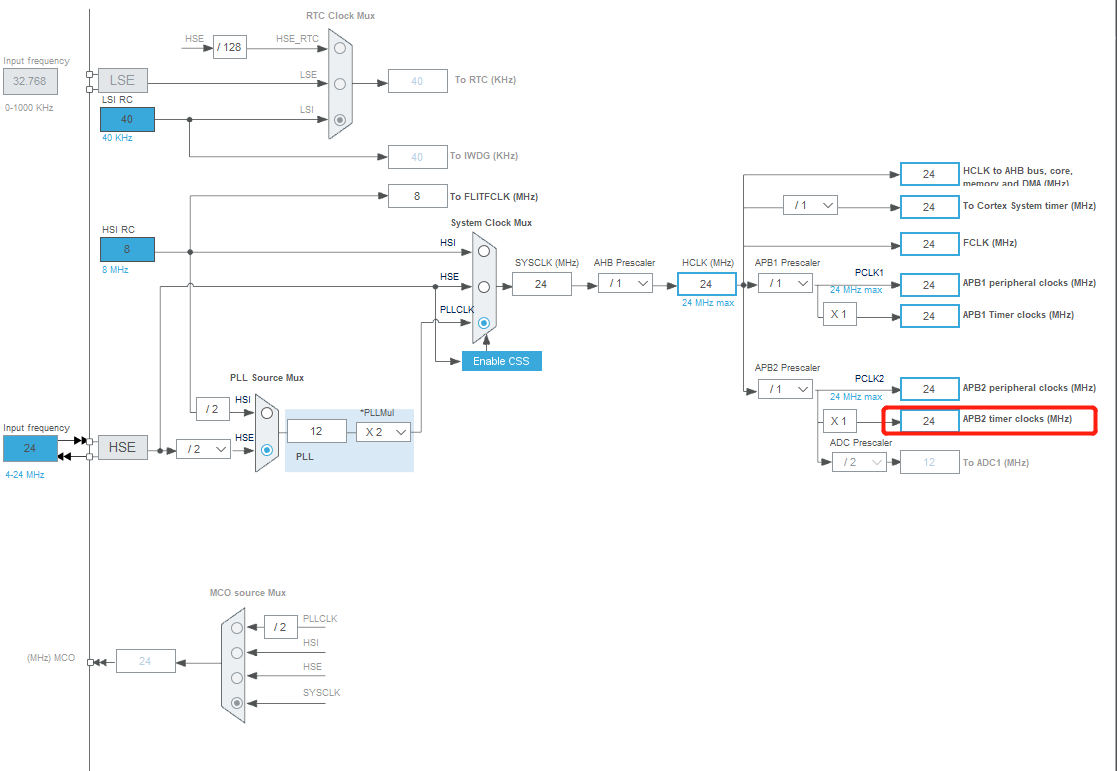


图4. 时钟配置

要注意图4. 中红框中的内容，我们使用的是通用定时器TIM3，挂载在APB2总线上，总线时钟为24MHz。

之后是TIM3\_CH4的配置，如图5. 所示：

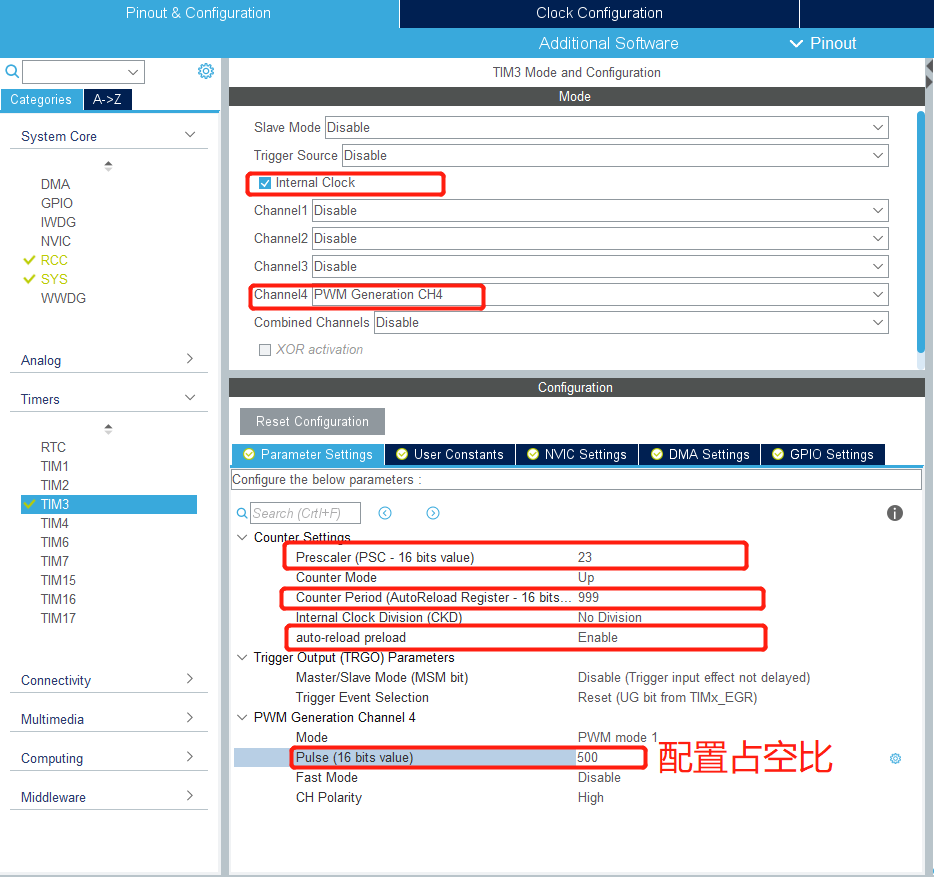


图5. TIM3\_CH4的配置

其中占空比的计算公式为：

在主函数之前，添加以下代码:

|  |
| --- |
| HAL\_TIM\_PWM\_Start(&htim3,TIM\_CHANNEL\_4);//start PWM output |

1. **利用定时器捕获PWM波**
   1. **原理说明**

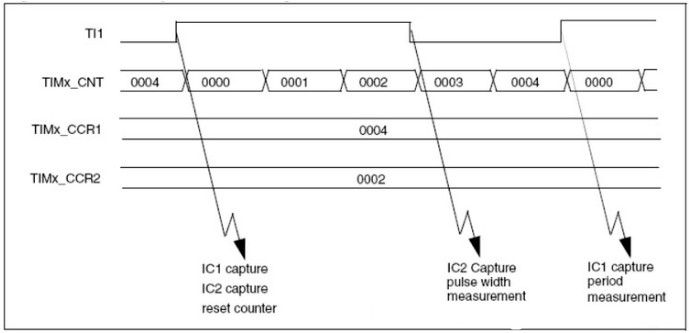
****

图6. 捕获PWM波的原理

PWM输入捕获可以获取到频率和占空比，对于我们使用的stm32F100芯片的TIM3来说，通道1,2共同用来完成PWM的输入捕获。

捕获的基本步骤：

●为了实现PWM输入捕获，TIM4占用了2个通道。第2通道的电平变化会被第一通道和第二通道引脚检测到，其中第一通道被设置为从机模式（规则如下：如果设置的是第二通道作为PWM输入捕获，则剩余的第一通道都为从机模式，反之亦然，**在我们的实验中，TIM4的通道1作为从机模式，通道2作为输入捕获**）。

●假设输入的PWM从低电平开始跳变，在第一个上升沿到来时，1,2通道同时检测到上升沿。而从机设置为复位模式，所以将TIM4的计数值复位至0，此时不会产生一个中断请求。

●下一个到来的电平是下降沿，此时通道1发生捕获事件，将计数值存入通道1的CCR1里。

●然后是第二个上升沿到来后，此时通道2发生捕获事件，将此时的计数值存入通道2的CCR2里。复位模式此时又将TIM4计数值复位至0，等待第二个下降沿到来。

一次捕获过程完成，则PWM的频率和占空比的计算公式如下所示：

* 1. **配置说明**

配置TIM4\_CH2为PWM的输入捕获端口，可以发现，在完成配置之后，TIM4\_CH1也跟着变成了disable，这是说明TIM4\_CH1被自动配置成为了从机模式。具体操作如图7. 所示：

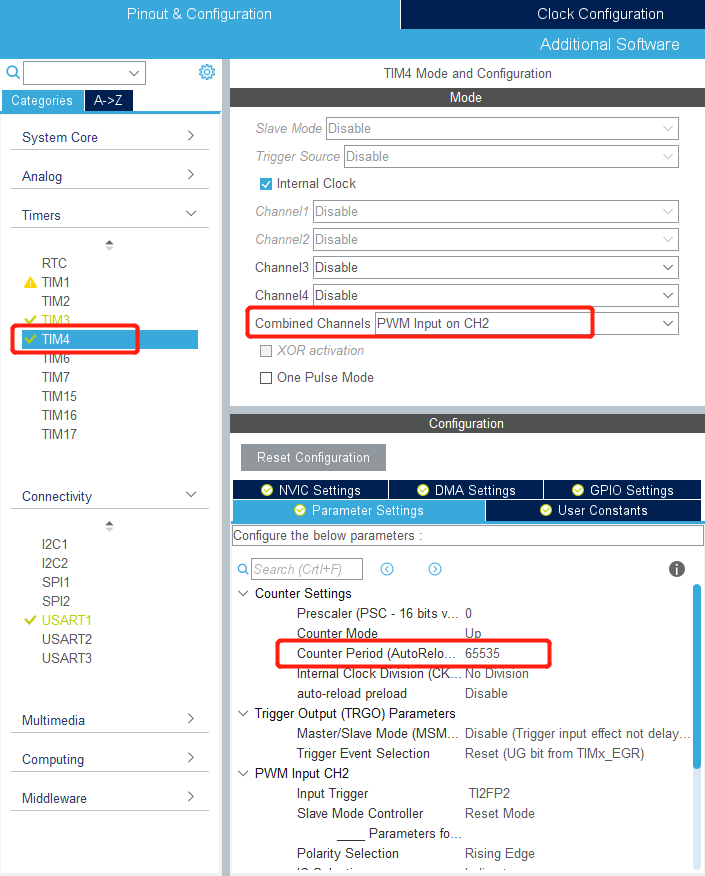
****

图7.TIM4的第1，第2通道的配置

完成上述配置之后，需要打开TIM4的中断。

1. **修改mian.c中的代码**

添加全局变量：

|  |
| --- |
| **volatile** **static** uint32\_t tmp1 = 0, tmp2 = 0;//volatile 该变量随时可能发生变化 |

初始化换函数：

|  |
| --- |
| HAL\_TIM\_PWM\_Start(&htim3,TIM\_CHANNEL\_4);//start PWM output  HAL\_TIM\_IC\_Start\_IT(&htim4, TIM\_CHANNEL\_1);  HAL\_TIM\_IC\_Start\_IT(&htim4, TIM\_CHANNEL\_2); |

While（1）循环中的代码：

|  |
| --- |
| **while** (1)  {  /\* USER CODE END WHILE \*/  **double** duty=0;  **double** freq=0;  uint8\_t pDuty[]="Duty is:";  uint8\_t pFreq[]="Freq is:";  **if** (tmp1 == 0)  {  duty = 0;  freq = 0;  }  **else**  {  duty = tmp2 / tmp1;  freq = 24000000.0f / tmp1;  }  uint8\_t pData[4]="";  **sprintf**(pData,"%.1f",duty);  **HAL\_UART\_Transmit**(&huart1,(uint8\_t\*)pDuty,**sizeof**(pDuty),100);//  **HAL\_UART\_Transmit**(&huart1,(uint8\_t\*)pData,**sizeof**(pData),100);//  **sprintf**(pData,"%.1f",freq);  **HAL\_UART\_Transmit**(&huart1,(uint8\_t\*)pFreq,**sizeof**(pFreq),100);//  **HAL\_UART\_Transmit**(&huart1,(uint8\_t\*)pData,**sizeof**(pData),100);//  /\* USER CODE BEGIN 3 \*/  } |

完成配置之后生成代码，在主函数下面添加定时器中断回调函数：

|  |
| --- |
| **void** **HAL\_TIM\_IC\_CaptureCallback**(TIM\_HandleTypeDef \*htim)  {  **if** (htim->Channel == *HAL\_TIM\_ACTIVE\_CHANNEL\_1*)  {  tmp1 = HAL\_TIM\_ReadCapturedValue(&htim4, TIM\_CHANNEL\_2);//周期  }  **else** **if** (htim->Channel == *HAL\_TIM\_ACTIVE\_CHANNEL\_2*)  {  tmp2 = HAL\_TIM\_ReadCapturedValue(&htim4, TIM\_CHANNEL\_1);//占空比  }  } |

1. **延伸**

自行搜集资料，尝试用PWM波驱动航模电机。