

第6讲

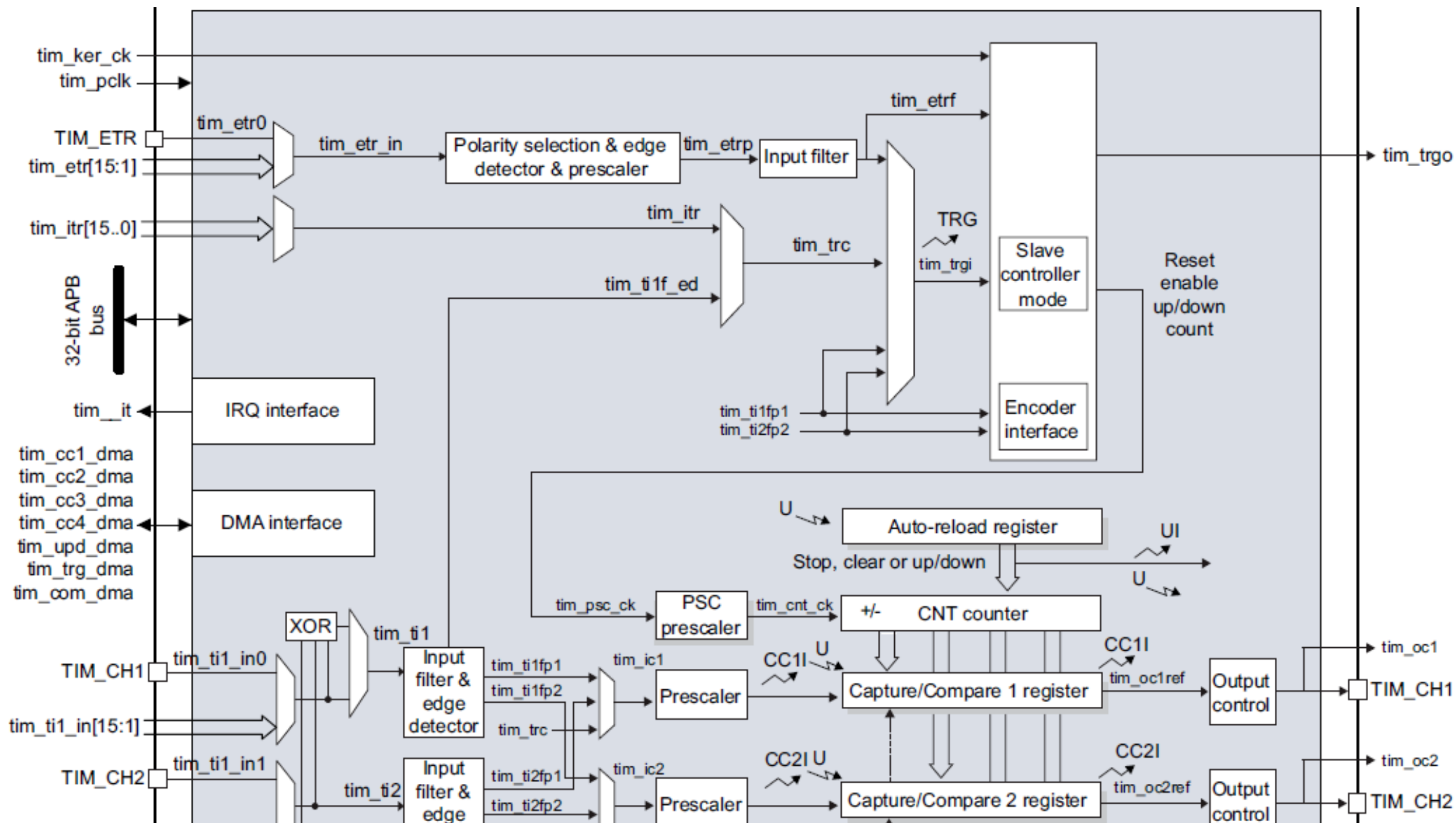
定时器

主要内容

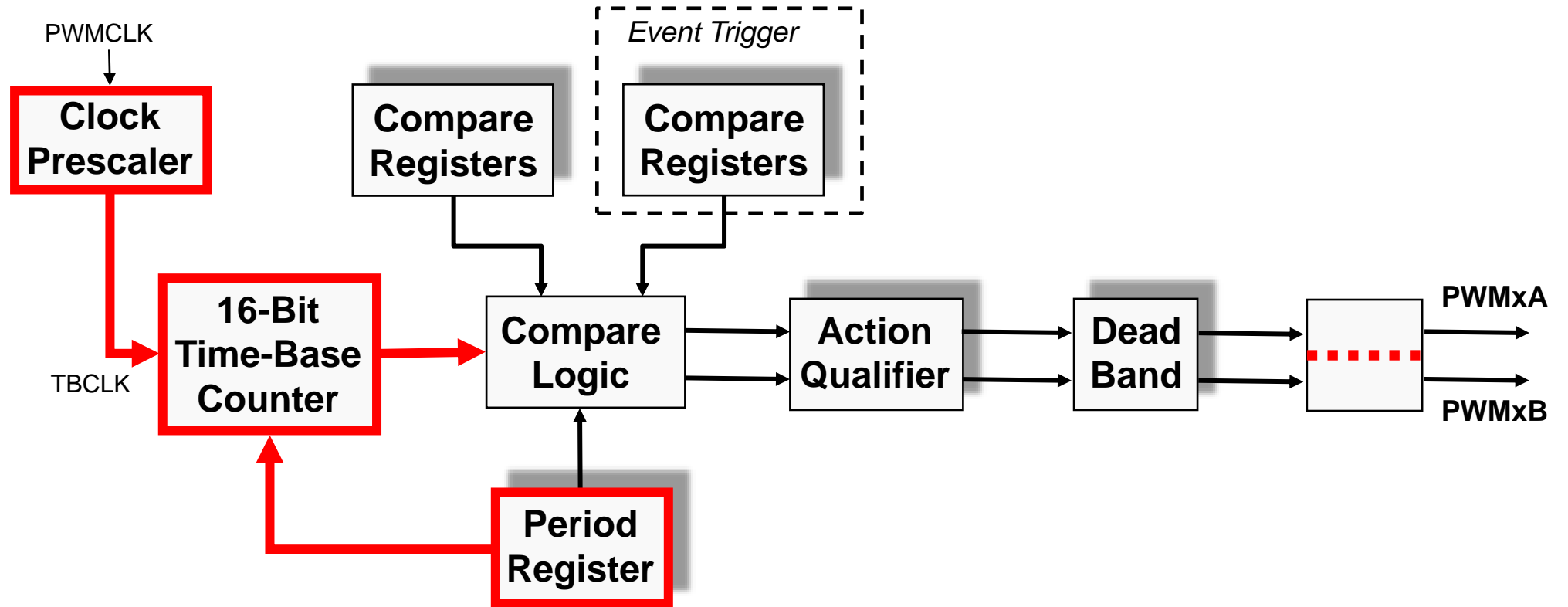
- STM32G4的定时器
- 定时器的编程实现方式
- ✓ 中断、PWM输出和输入捕捉
- 动手练习6

STM32G4的定时器

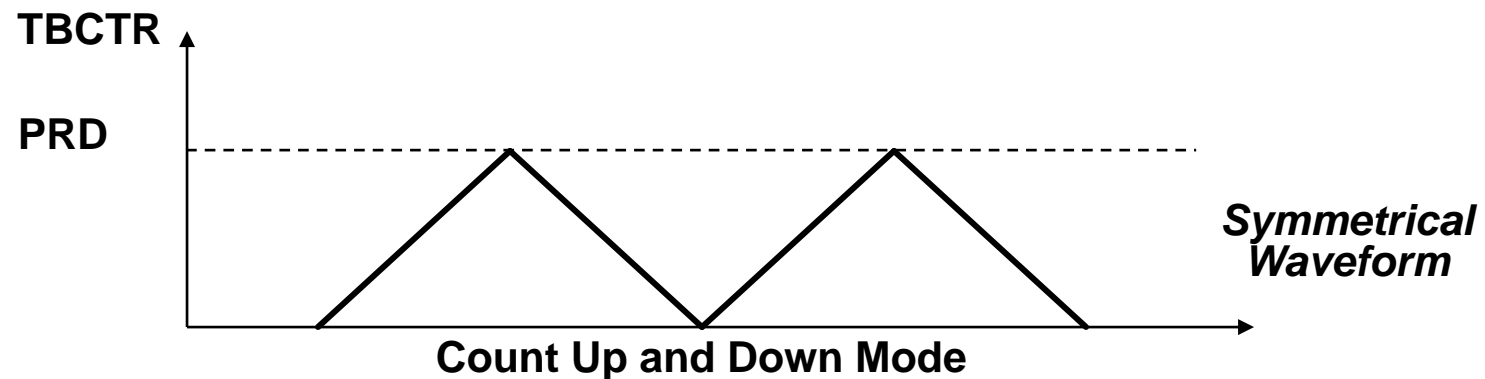
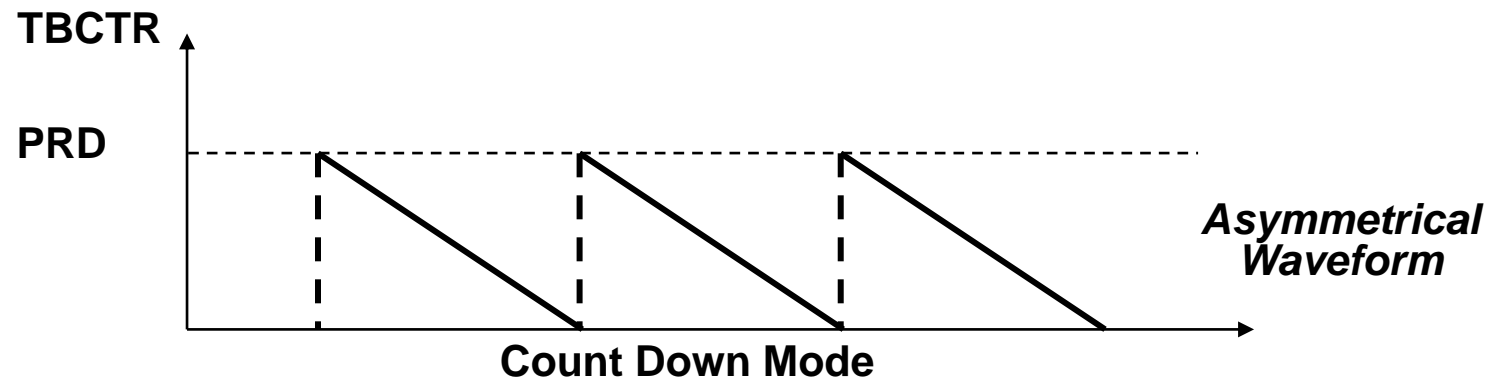
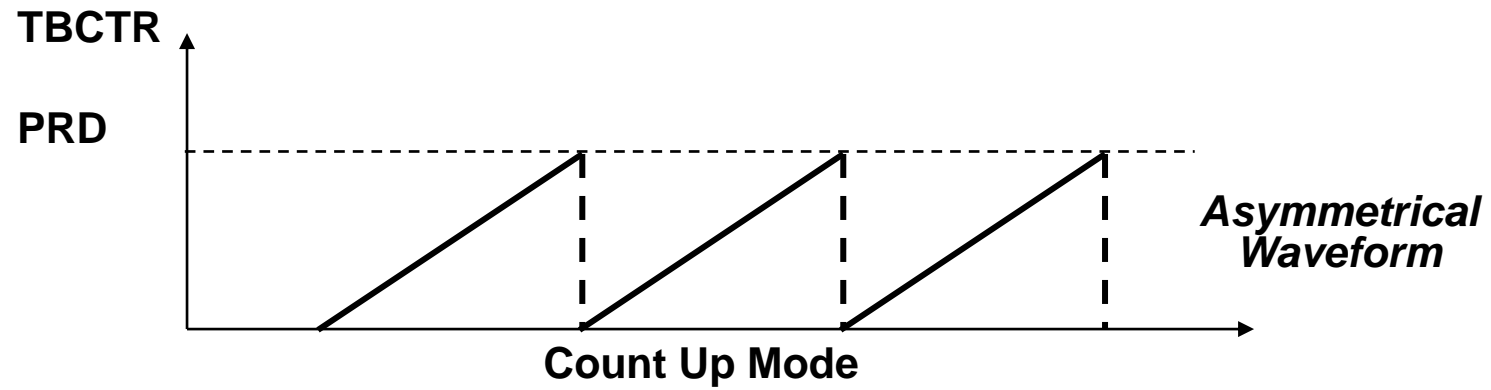
类型	定时器	计数器精度	计数器类型	预分频因子	DMA请求产生	捕捉/比较通道	互补输出
高精度定时器	HRTIM	16位	Up	/1/2/4(x2 x4 x8 x16 x32,带DLL)	是	12	有
高级控制	TIM1, TIM8,TIM20	16位	Up, down, Up/down	1~65536之间的整数	是	4	4
通用	TIM2,TIM5	32位	Up, down, Up/down	1~65536之间的整数	是	4	无
通用	TIM3,TIM4	16位	Up, down, Up/down	1~65536之间的整数	是	4	无
通用	TIM15	16位	Up	1~65536之间的整数	是	2	1
通用	TIM16,TIM17	16位	Up	1~65536之间的整数	是	1	1
基本	TIM6,TIM7	16位	Up	1~65536之间的整数	是	0	无



PWM Time-Base Sub-Module

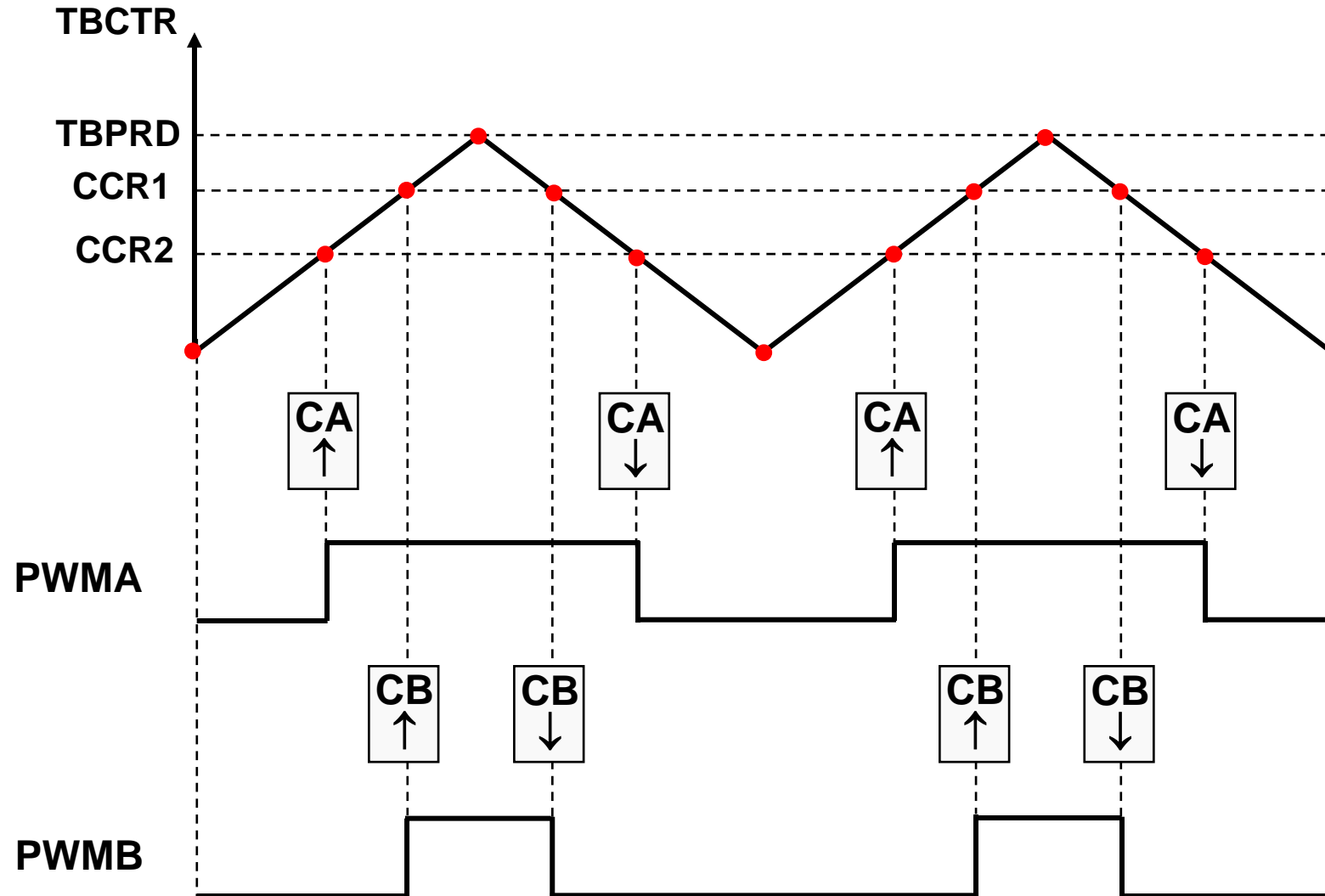


PWM Time-Base Count Modes



PWM Count Up-Down Symmetric Waveform

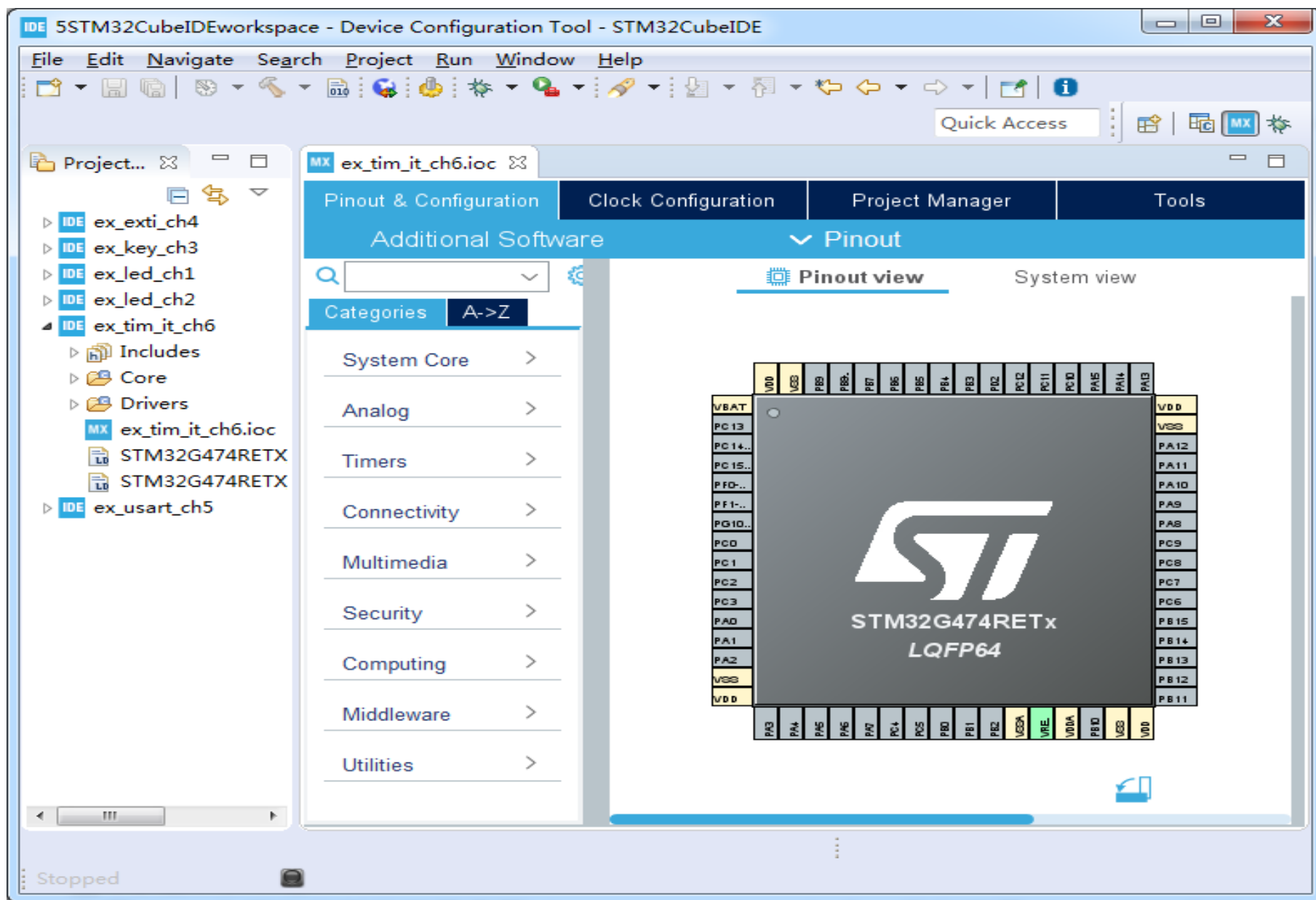
with Independent Modulation on EPWMA / B



练习6：定时器

- 利用STM32G431RB上的通用定时器，以定时器中断的方式控制NUCLEO-G431RB板上的发光二极管LD2以不同频率闪烁。

建立新工程



选择时钟源和Debug模

■ 选择时钟源和Debug模式

- ✓ System Core->RC->将高速时钟（HSE）选择为Crystal/Ceramic Resonator
- ✓ SYS->Debug选择为Serial Wire

配置GPIO

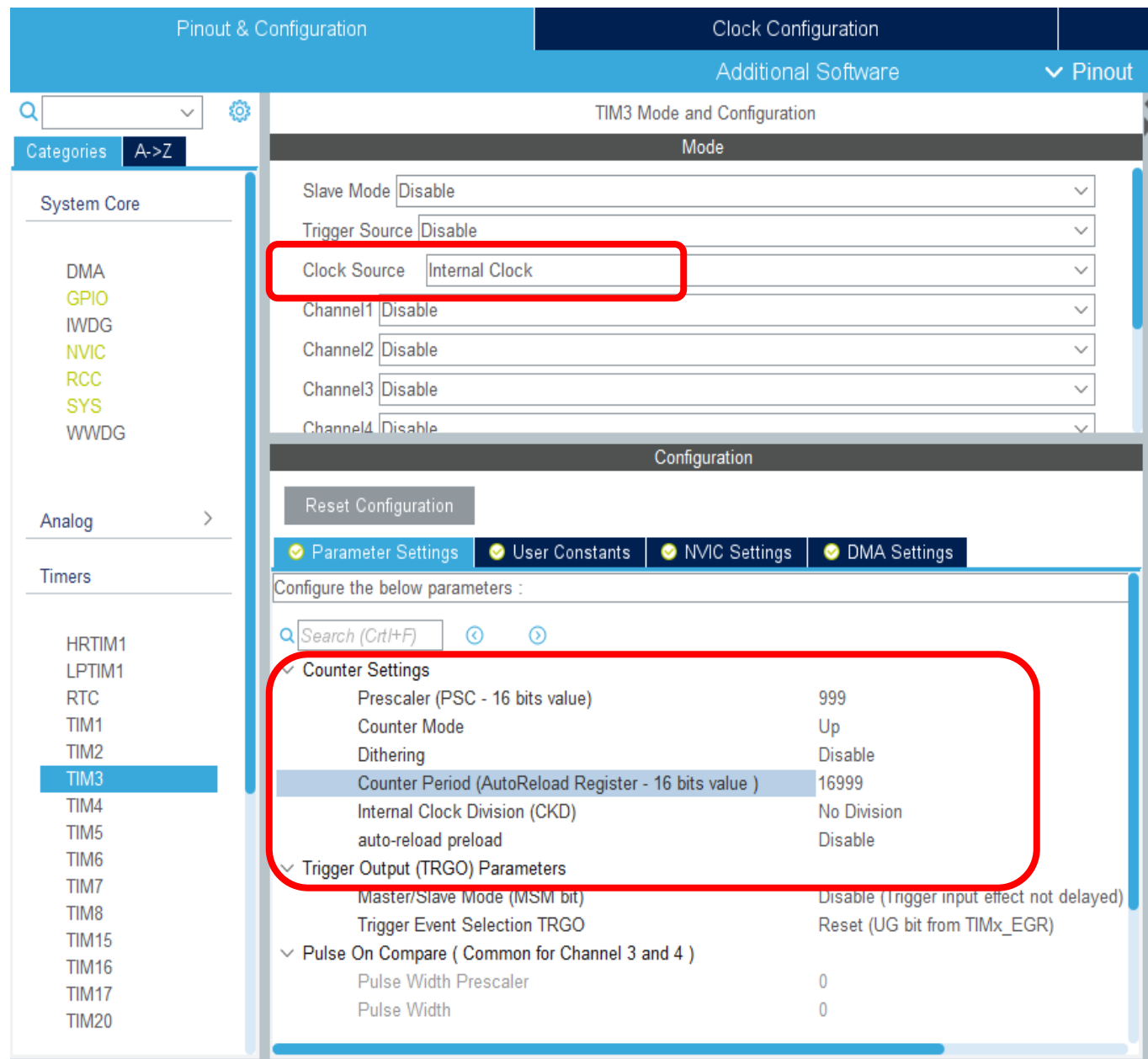
- 配置PA5为输出（GPIO_Output）。在NUCLEO-G431RB板上，PA5控制发光二极管LD2

PA5 Configuration :

GPIO output level	Low
GPIO mode	Output Push Pull
GPIO Pull-up/Pull-down	Pull-up
Maximum output speed	High
User Label	LED

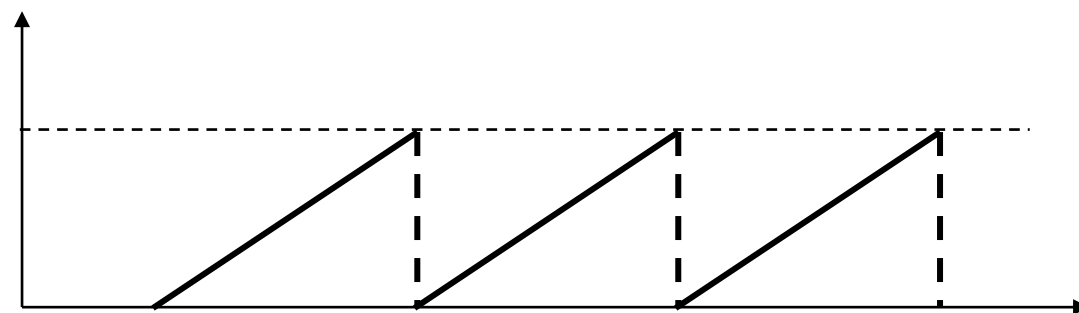
配置定时器

- 在模式（Mode）栏，将时钟源（Clock Source）选择为Internal Clock
- 把Parameter Settings中的预分频因子（Prescaler）和计数器周期（Counter Period）分别设置为999和16999
- 把Counter Mode设置为升模式（Up），并使能自动重载（auto-reload preload）



如果系统时钟频率为170MHz，TIM3定时器周期是多少？

- ☐ A 1ms
- ☐ B 10ms
- ☒ C 100ms
- ☐ D 1s



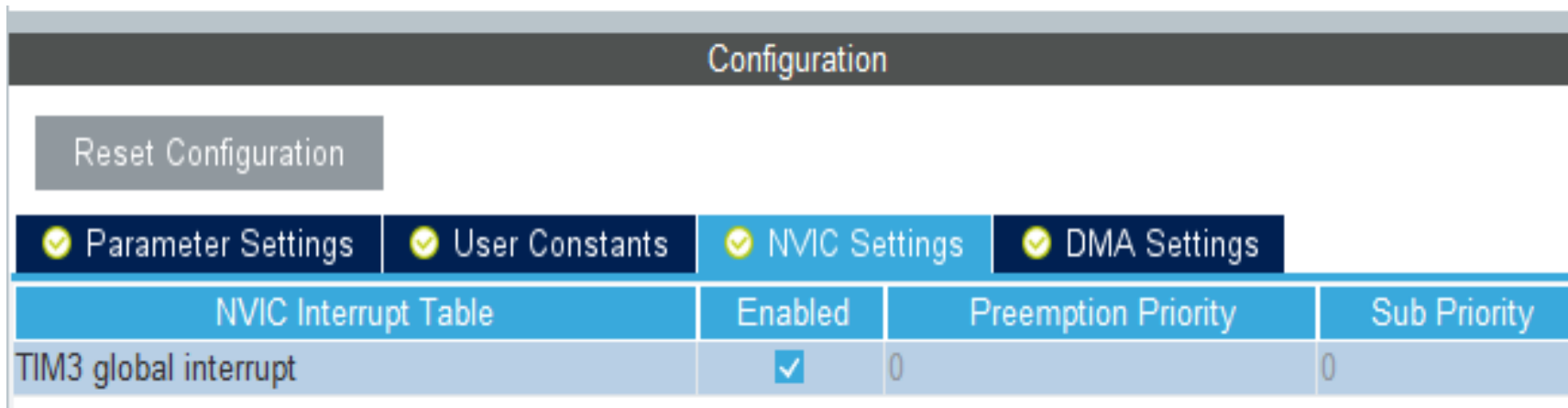
Counter Settings

Prescaler (PSC - 16 bits value)	999
Counter Mode	Up
Dithering	Disable
Counter Period (AutoReload Register - 16 bits value)	16999
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable

提交

配置中断

- 在TIM3的配置界面中，选中NVIC设置（NVIC Settings），使能TIM3的全局中断

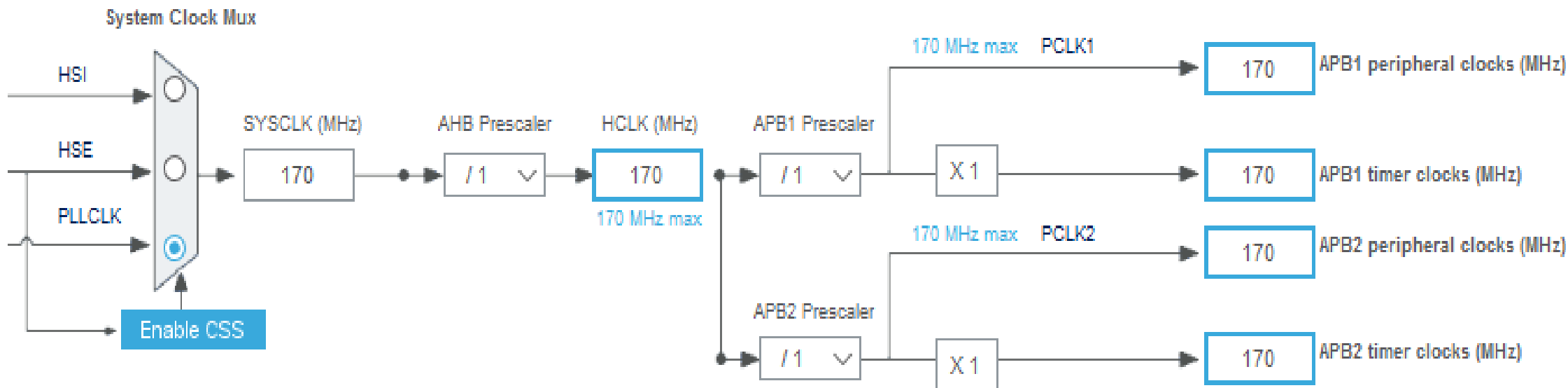


- 返回到System Core中的NVIC配置界面，将优先级组（Priority Group）选择4bits for pre-emption priority 0 bits for subpriority。同时，还会看到，TIM3 global interrupt已出现在中断表中，并且已使能；将它的抢占式优先级设为1，响应优先级为0。

时钟配置

■ 配置系统时钟

✓ 在“Clock Configuration”中，将系统时钟（SYSCLK）配置为170Mhz



■ 保存硬件配置界面 (*.ioc)，启动代码生成

生成代码分析

■ Core->Src, 打开main.c

■ MX_TIM3_Init(): 实现对定时器TIM3的初始化

```
static void MX_TIM3_Init(void)
{
    .....
    htim3.Instance = TIM3;
    htim3.Init.Prescaler = 999;
    htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
    htim3.Init.Period = 16999;
    htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
    htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
    if (HAL_TIM_Base_Init(&htim3) != HAL_OK)
    .....
}
```

main.c文件的最前面:

```
TIM_HandleTypeDef htim3;
```


生成代码分析

```
static void MX_TIM3_Init(void)
{
    .....
    htim3.Instance = TIM3;
    htim3.Init.Prescaler = 999;
    htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
    htim3.Init.Period = 16999;
    htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
    htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
    if (HAL_TIM_Base_Init(&htim3) != HAL_OK)
    .....
}
```



使能定时器中断

HAL_TIM_Base_Start_IT(TIM_HandleTypeDef *htim);


- 放到main函数中，位于while(1)循环前面的注释对中
- TIM3初始化函数MX_TIM3_Init()的后面

```
/* USER CODE BEGIN 2 */  
HAL_TIM_Base_Start_IT(&htim3);  
/* USER CODE END 2 */
```

定时器的中断服务函数

- 开启TIM3的中断后，当条件满足时，就会执行定时器中断服务函数

TIM3_IRQHandler()  stm32g4xx_it.c

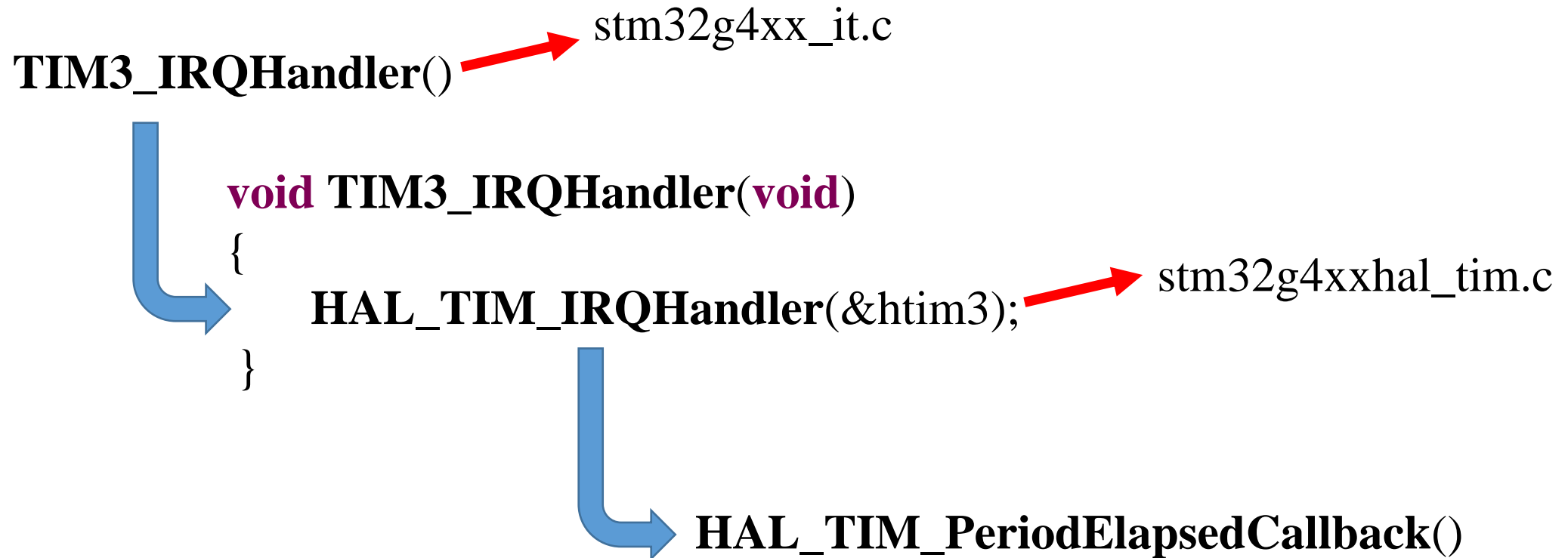
```
void TIM3_IRQHandler(void)
{
    HAL_TIM_IRQHandler(&htim3);  stm32g4xxhal_tim.c
}
```

TIM3_IRQHandler()  HAL_TIM_IRQHandler()  HAL_TIM_PeriodElapsedCallback()

弱函数

定时器的中断服务函数

- 开启TIM3的中断后，当条件满足时，就会执行定时器中断服务函数



弱函数

重定义定时器回调函数

- 在main.c中重新定义回调函数HAL_TIM_PeriodElapsedCallback(), 使PA5的输出状态翻转

```
/* USER CODE BEGIN 4 */  
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)  
{  
    HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);  
}  
/* USER CODE END 4 */
```

- 定时器的预分频因子和计数器周期分别设置为999和16999, 定时器的时钟频率为170MHz, 最终TIM3中断的周期为:
 $1000 \times 17000 / 170\text{MHz} = 0.1\text{s}$, 即10Hz

编译、下载，运行

- 编译工程，并下载到硬件中运行，LD2闪烁频率是多少？

练习6：定时器中断

任务6.1、修改定时器TIM3的预分频因子（Prescaler）和计数器周期（Counter Period），改变LD2的闪烁频率为1Hz、0.5Hz等，并下载到硬件上进行验证。

任务6.2、用定时器TIM3中断，控制LD2和蜂鸣器以9Hz频率闪烁/响。

任务6.3、设置TIM3中断频率为较高频率，譬如1kHz。配置串口模块USART2，实现通过串口助手发送数据，控制LD2的闪烁频率。要求LD2闪烁频率与TIM3中断有关。

任务6.4、编程实现：获取按键按下的时间，并通过串口输出结果。

任务6.5、用数码管实现秒表计数显示。

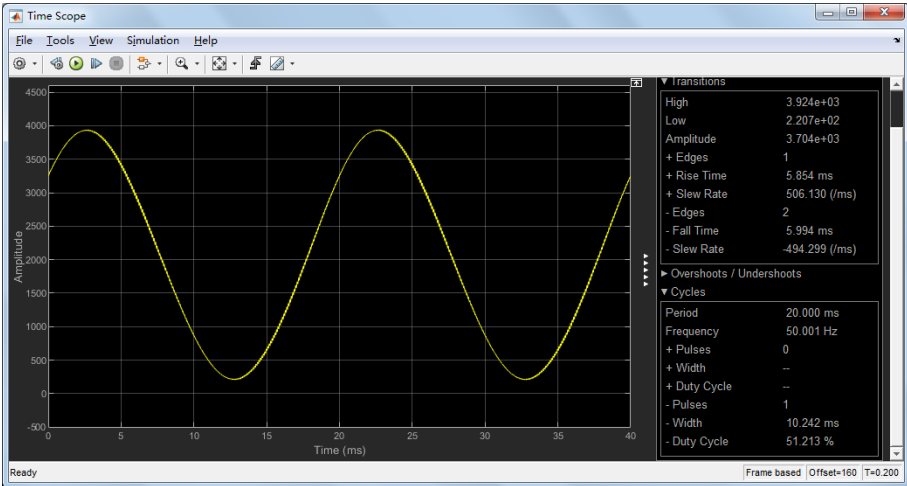
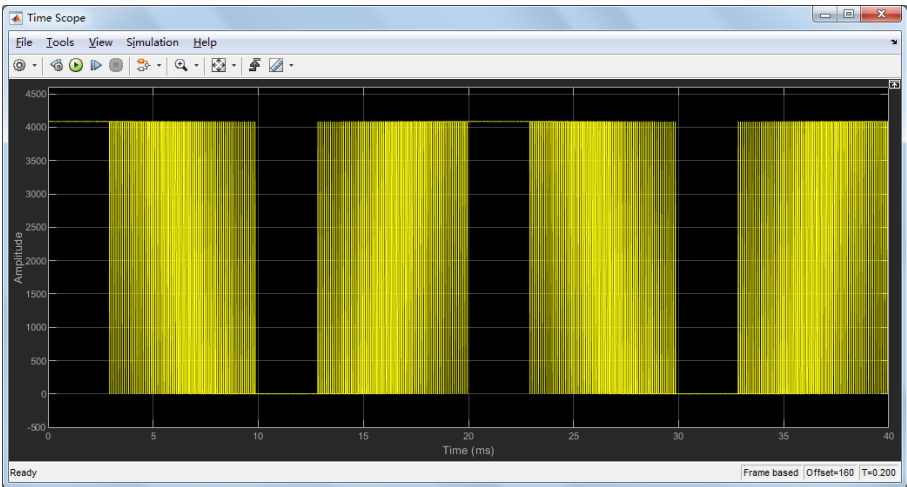
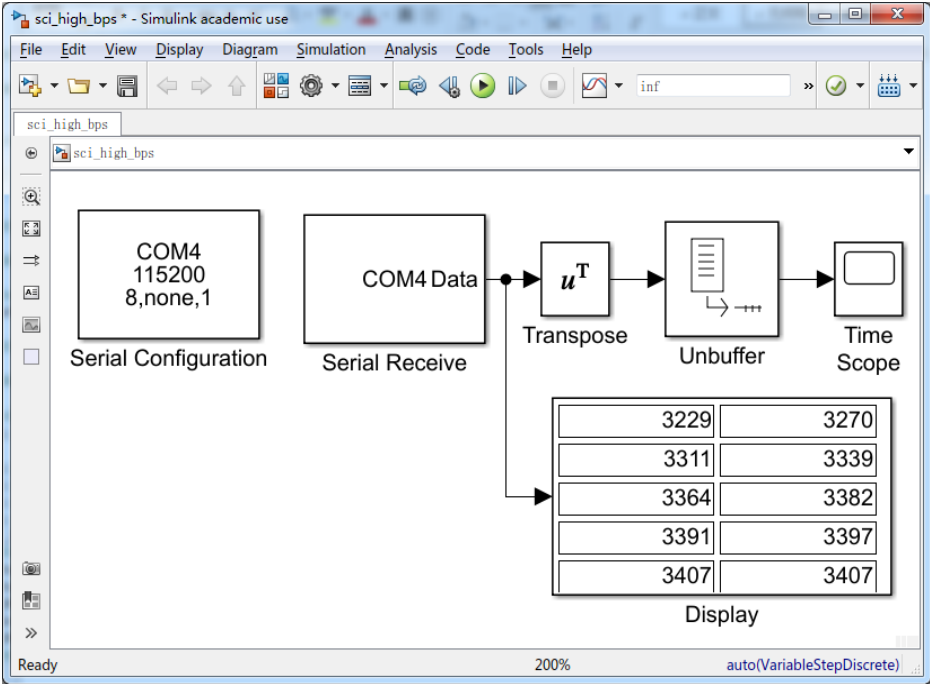
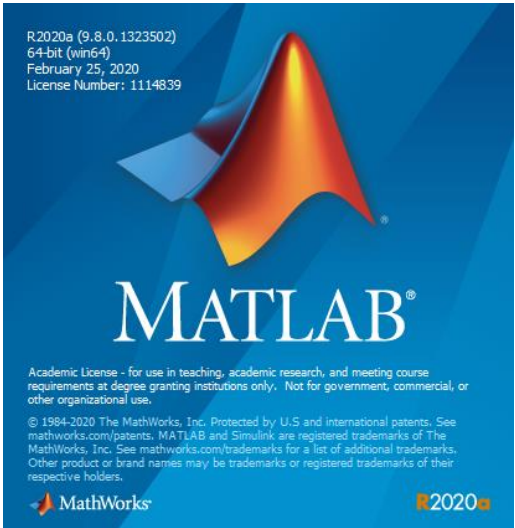
提交网络学堂：每个子任务的工程文件（压缩），代码有简单注释

- 熟悉开发系统
- C语言基础, GPIO, 输出
- GPIO, 输入
- 中断
- 串口
- 定时器: 中断
- ADC
- DAC+ADC
- DMA+ADC+DAC
- PWM波形输出

发光二极管 蜂鸣器

发光二极管 蜂鸣器
串口助手

? ? ?



R2020a (9.8.0.1323502)
64-bit (win64)
February 25, 2020
License Number: 1114839

The MATLAB logo is a 3D surface plot of a mathematical function, resembling a mountain peak. It is colored with a gradient from blue at the base to yellow and orange at the peak. The logo is positioned in the upper center of the slide.

MATLAB®

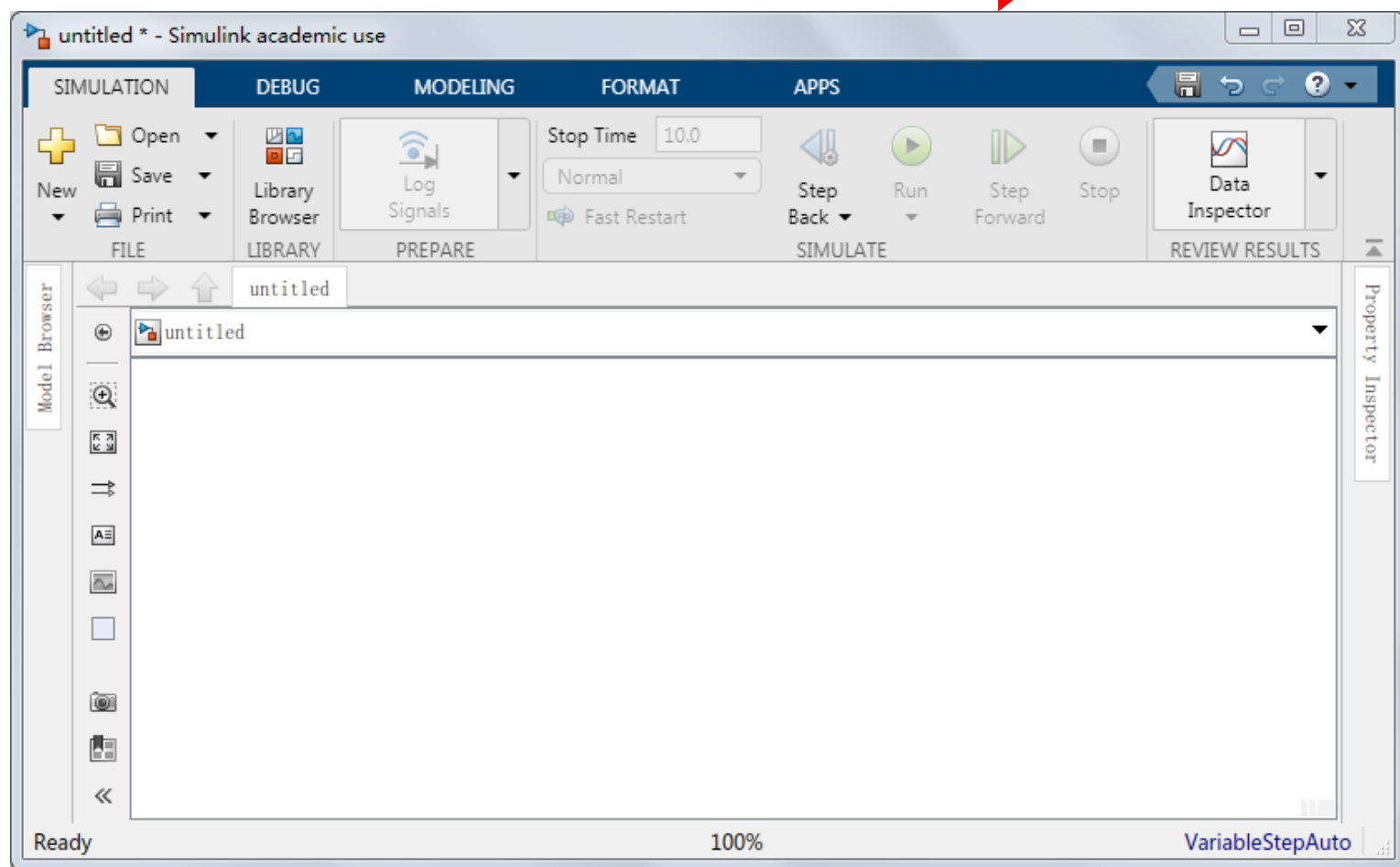
Academic License - for use in teaching, academic research, and meeting course requirements at degree granting institutions only. Not for government, commercial, or other organizational use.

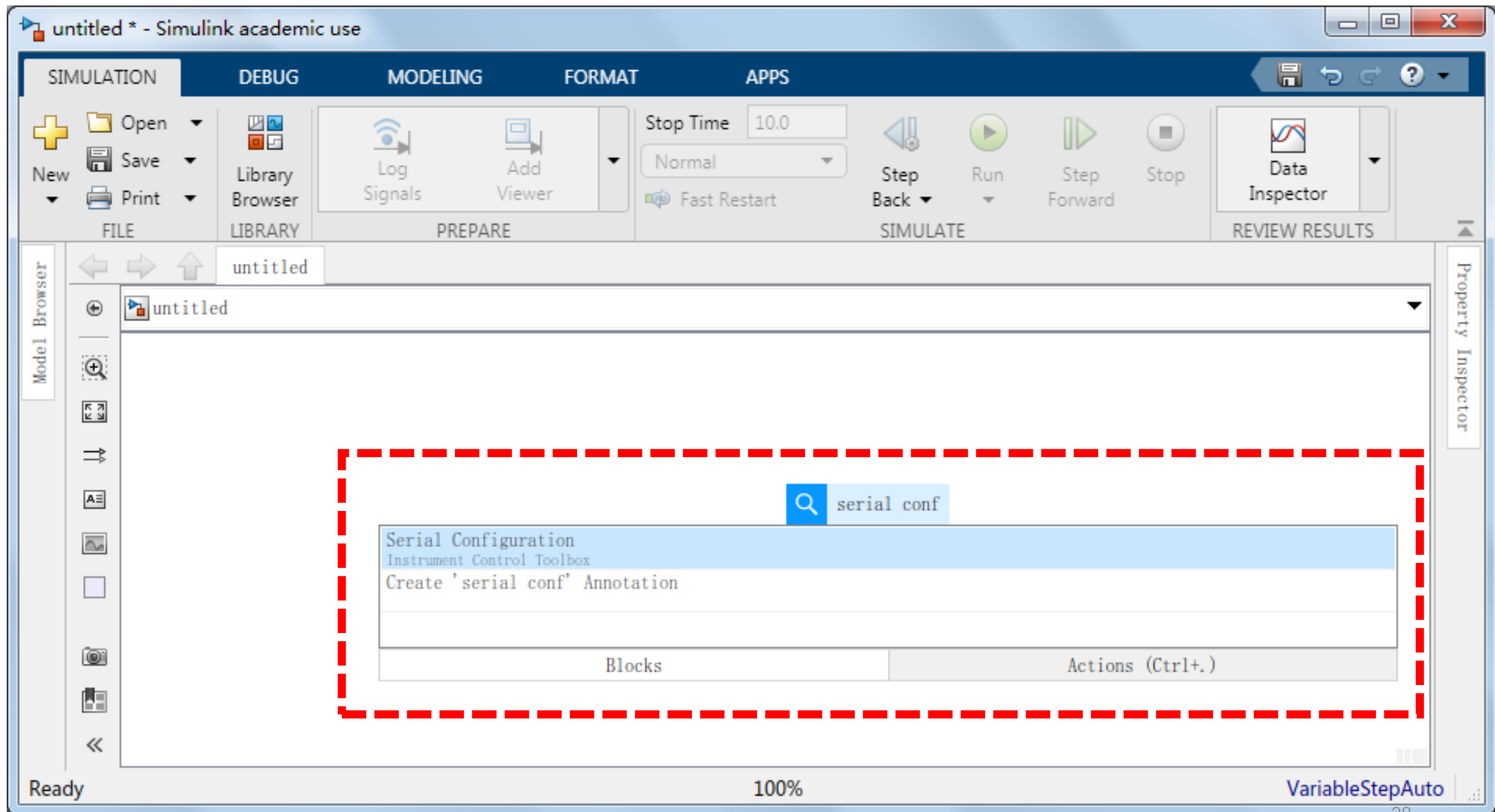
© 1984-2020 The MathWorks, Inc. Protected by U.S and international patents. See mathworks.com/patents. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.



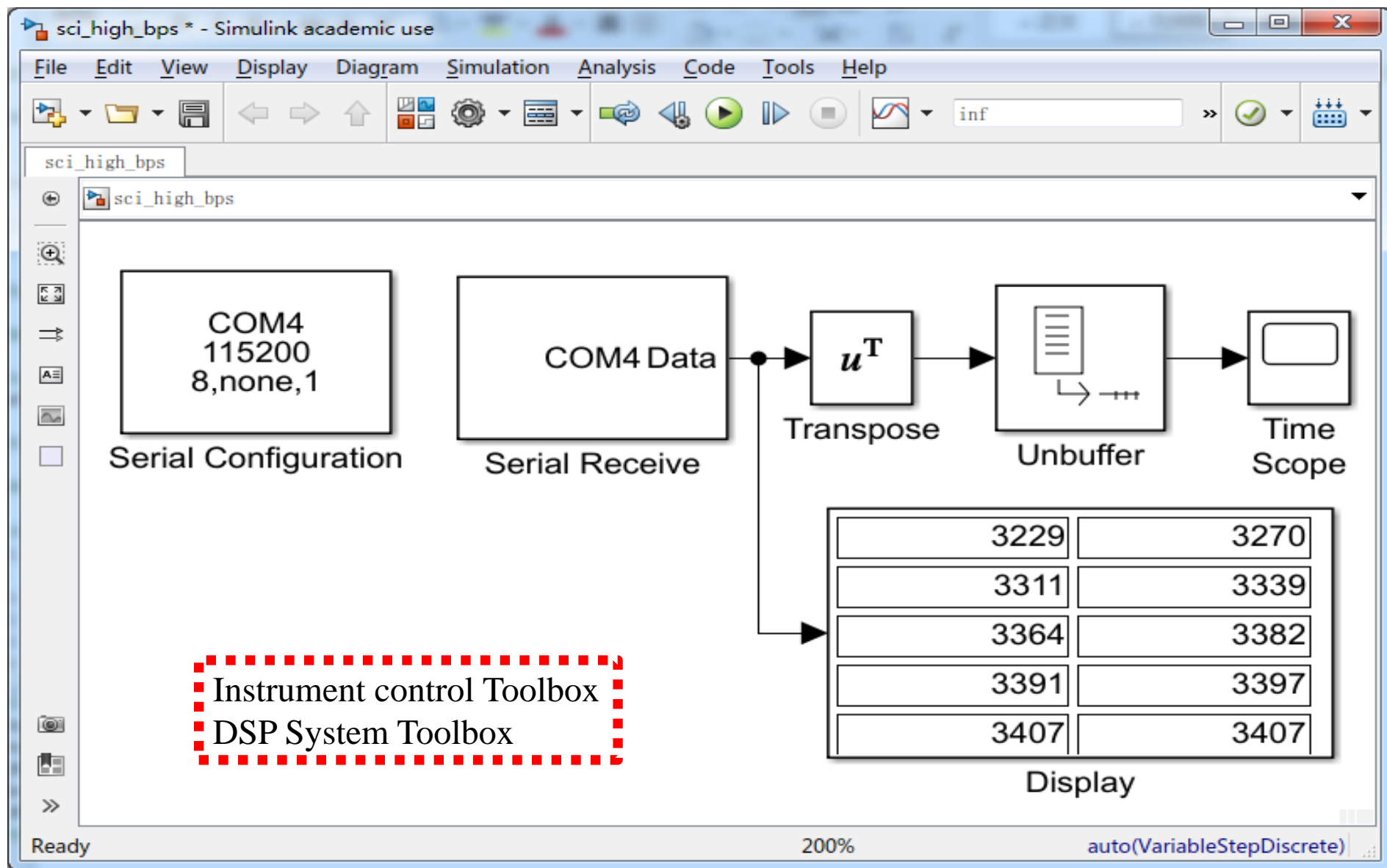
R2020a

别的版本也没问题





构建simulink模型



谢 谢!