

第8讲

DAC

主要内容

- STM32G4的DAC
- 单通道DAC输出
- 动手练习8

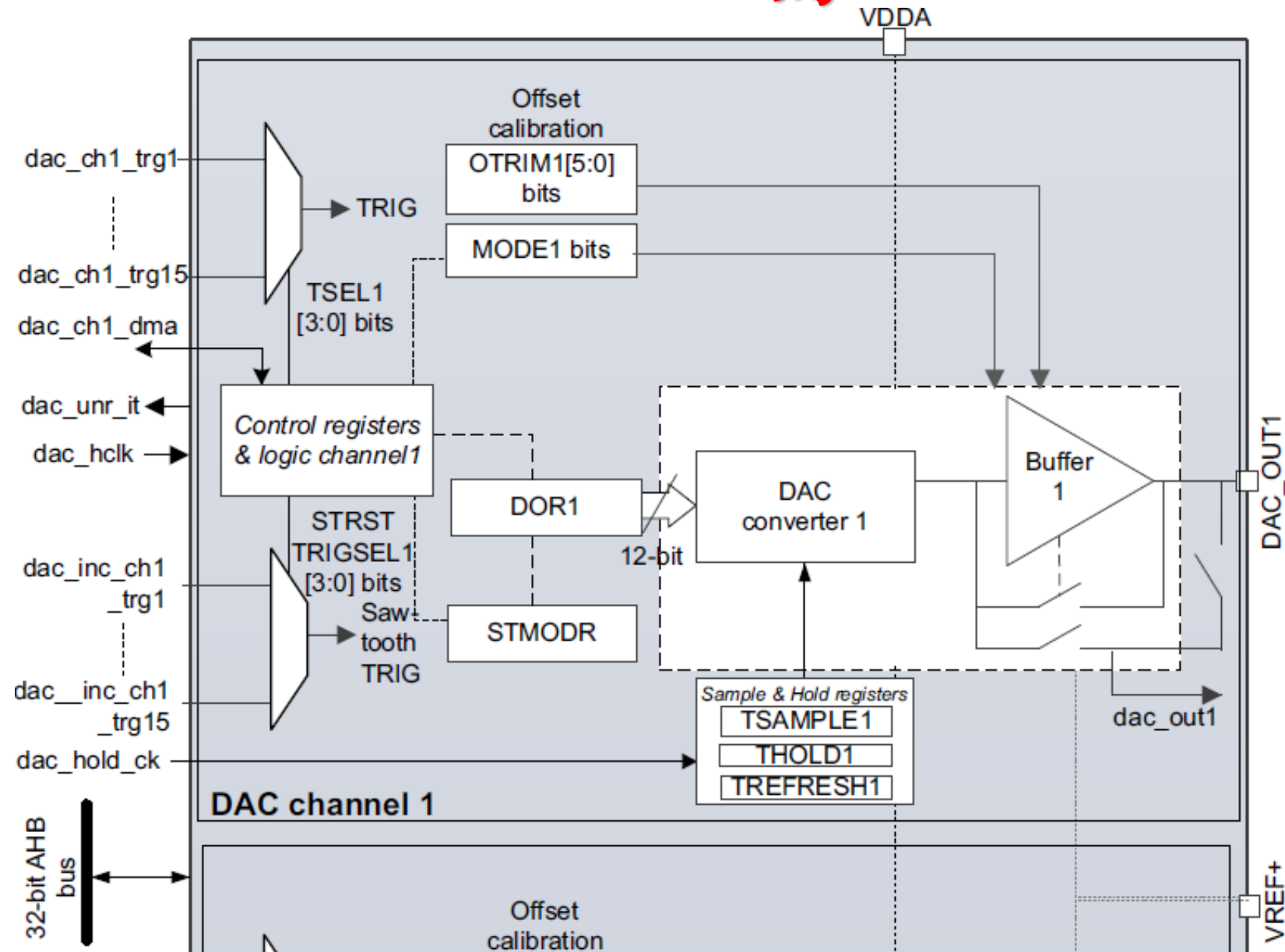
STM32G4的DAC

■ STM32G431RB有2路可以引出到GPIO引脚的12位DAC：DAC1_OUT1/DAC1_OUT2。

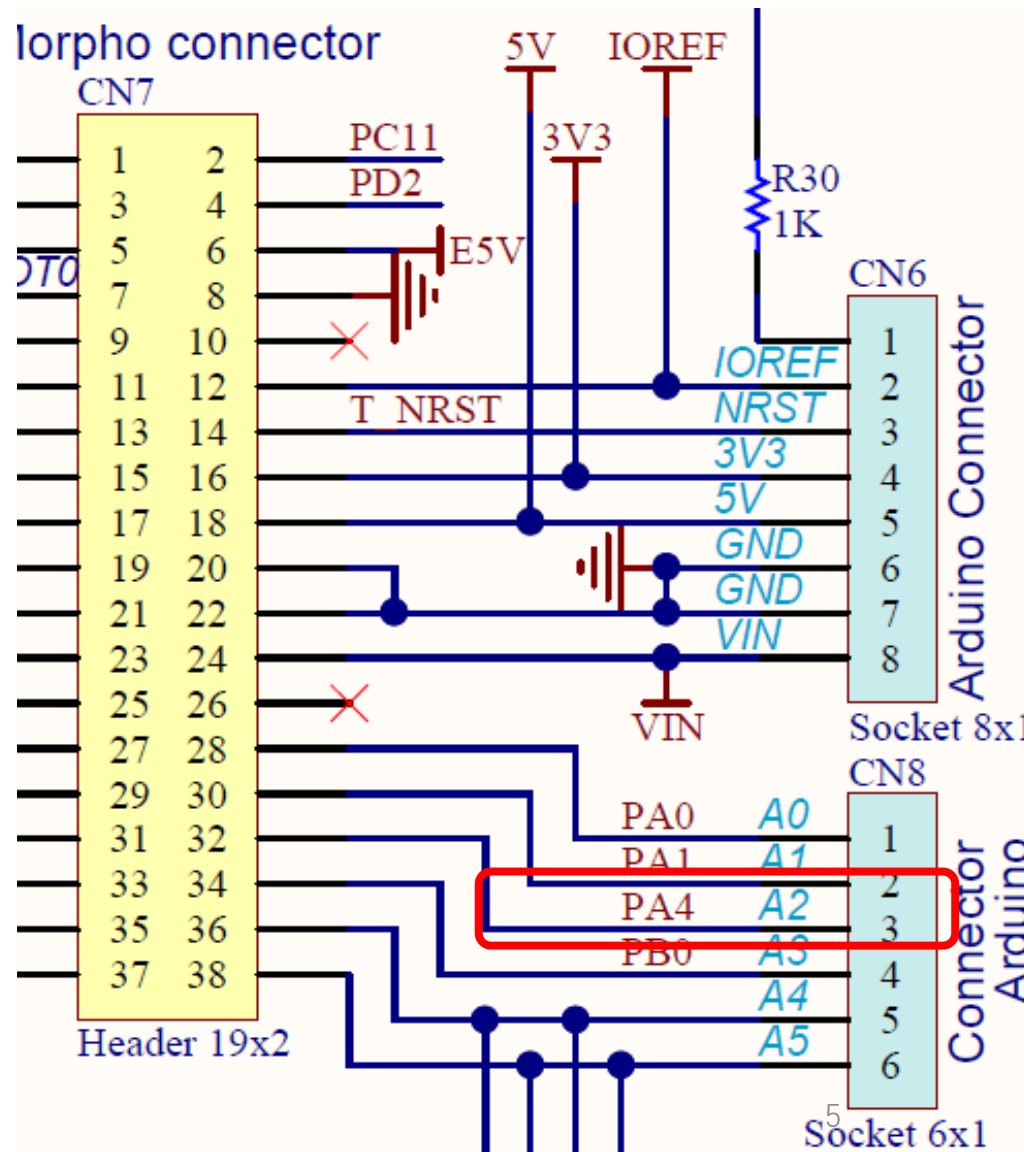
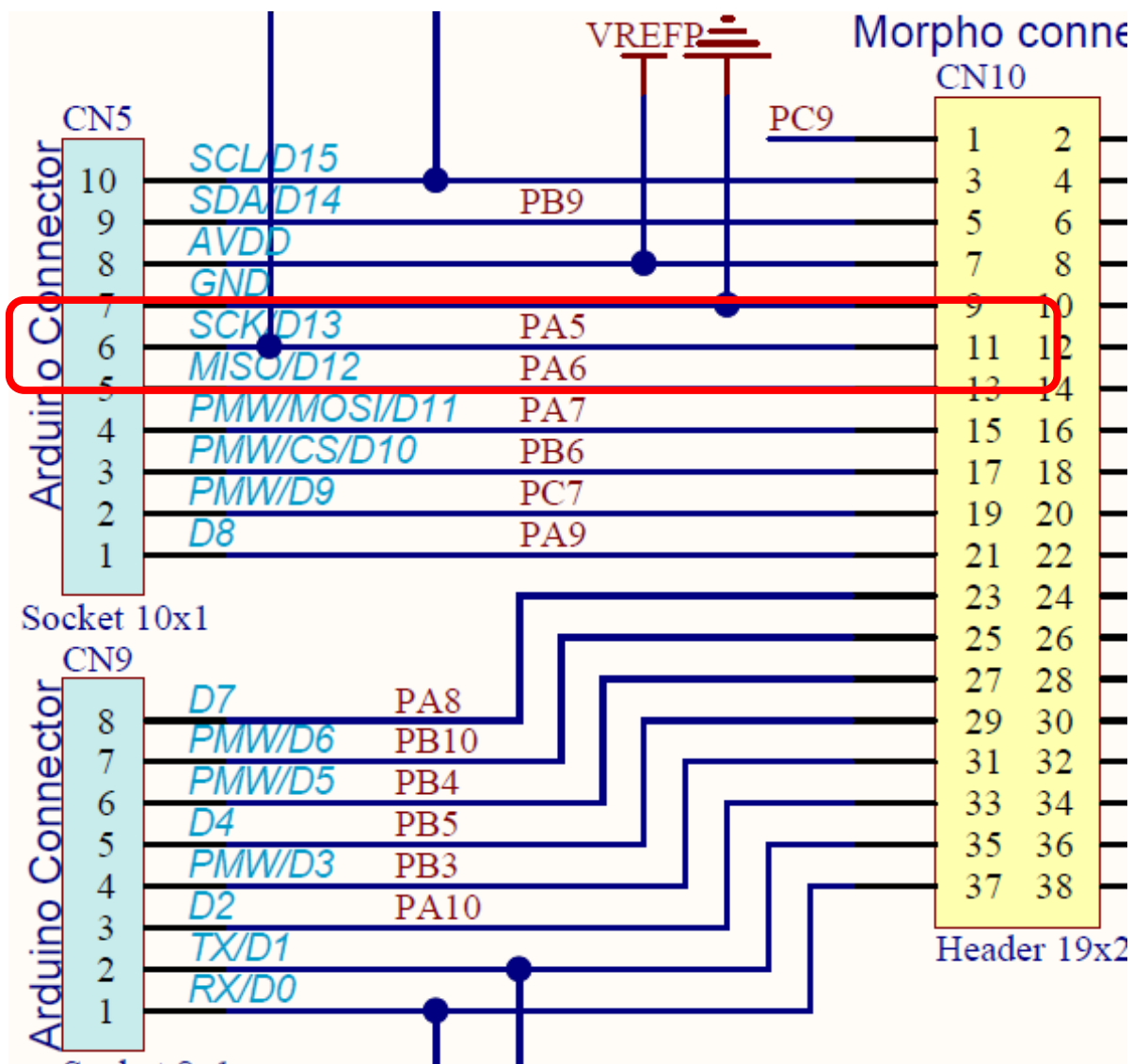
DAC features	DAC1	DAC2	DAC3	DAC4
Dual channel	X	-	X	X
Output buffer	X	X	-	-
I/O connection	DAC1_OUT1 on PA4 DAC1_OUT2 on PA5	DAC2_OUT1 on PA6	No connection to a GPIO	
Maximum sampling time	1MSPS		15MSPS	
Autonomous mode	-			

DAC implementation

STM32G4的DAC



2路DAC对应的引脚分别为：DAC1_OUT1对应PA4； DAC1_OUT2对应PA5



DAC输出

- 这2路DAC均有输出缓冲。所谓缓冲，是指信号是经过运算放大器电路送出的。在STM32G431RB中，该运放集成在MCU内部。在参数配置的时候，可以选择是否能缓冲。
- DAC模块从MCU引脚上最终输出的电压，与输入到它的数据输出寄存器中的数值有关。由于DAC为12位，当数据寄存器中的值为4095时，DAC会输出最大值。但这个最大值具体是几伏，还与MCU的参考电压（VREF）有关。具体公式如下

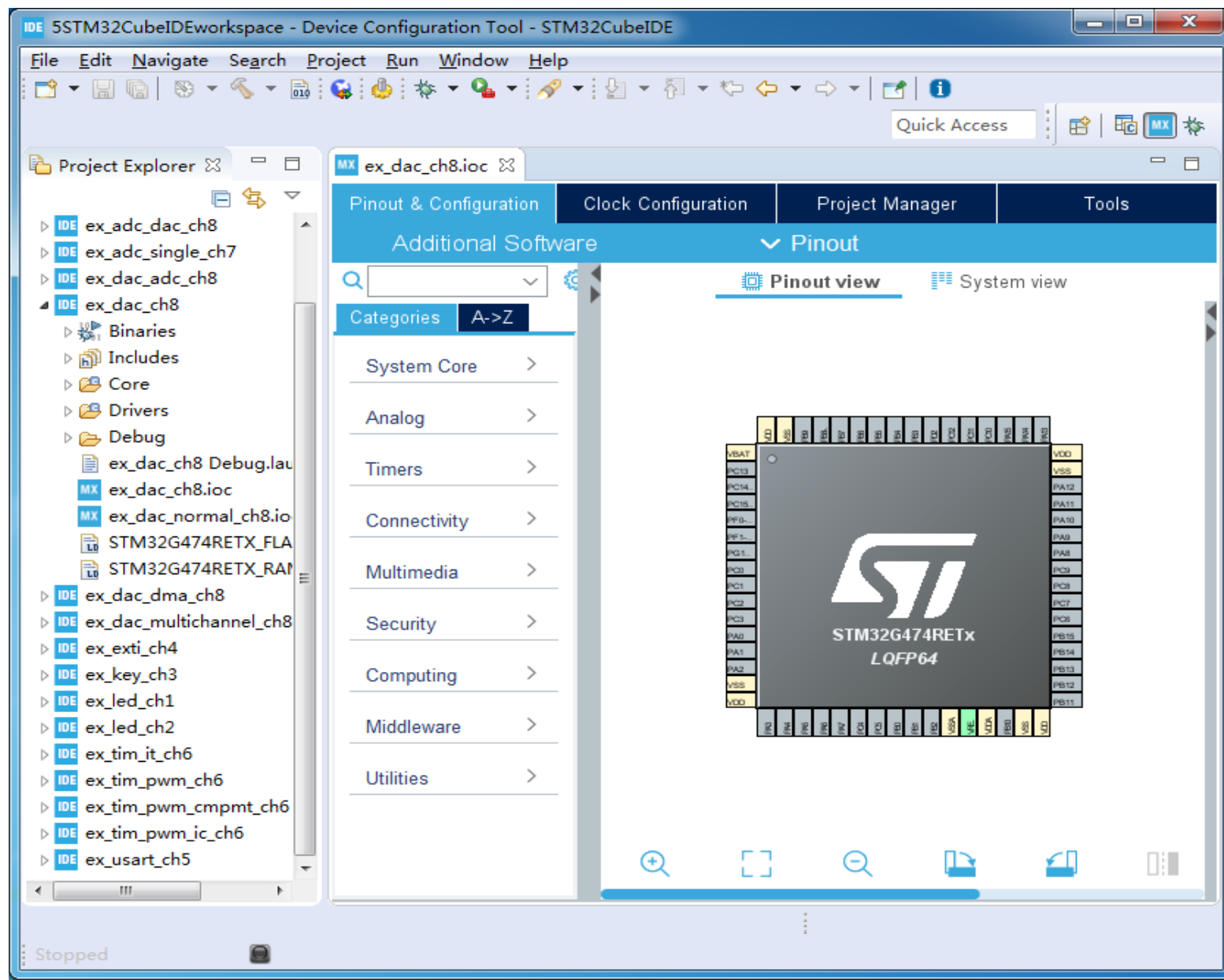
$$DAC_{output} = V_{REF} \times \frac{DOR}{4096}$$

DOR: data output register

练习8：单路DAC输出

- 配置DAC模块，输出模拟信号。

建立新工程



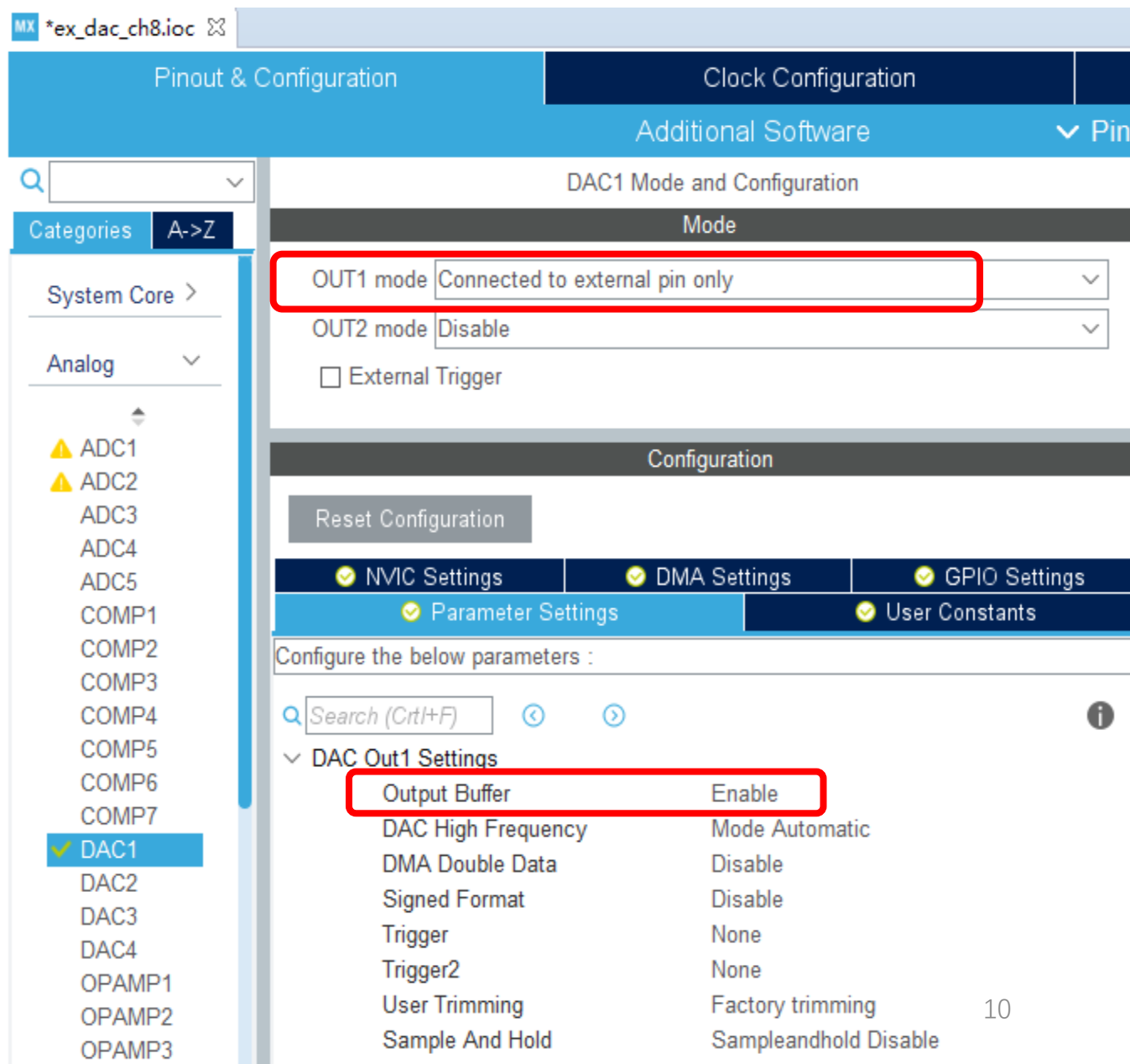
时钟源和Debug模式配置

■ 选择时钟源和Debug模式

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

配置DAC

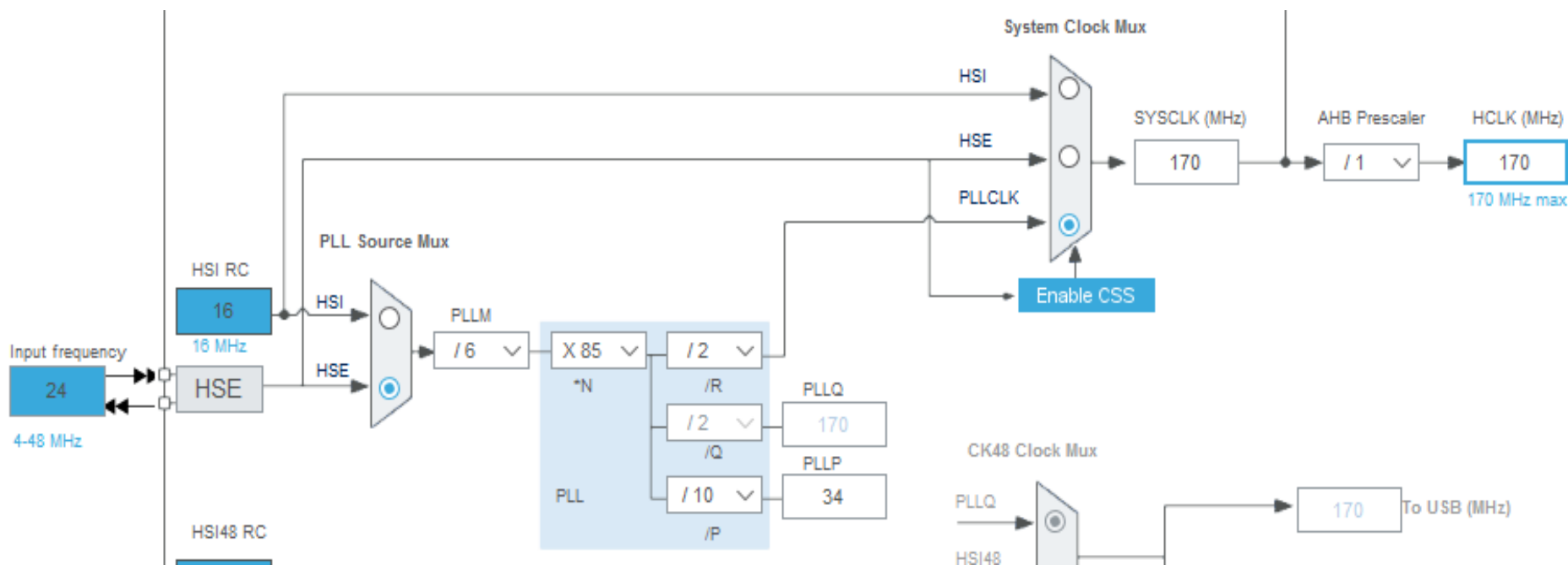
- 配置DAC1的OUT1
- 将OUT1 mode选择为: Connected to external pin only, 将OUT1连接到外部引脚PA4
- Configuration栏中, 是DAC的一些参数, 其中第一个参数就是是否使能Output Buffer, 该参数在默认情况下是Enable的
- 其他参数, 可以暂时均保持默认值



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

代码修改

- Core->Src, 打开main.c
- 启动DAC: 库函数HAL_DAC_Start()实现这个功能。可将它放到main函数中, while(1)前、MX_DAC1_Init()函数之后的注释对中:

```
/* USER CODE BEGIN 2 */  
HAL_DAC_Start(&hdac1, DAC_CHANNEL_1);  
/* USER CODE END 2 */
```

DAC数据输出寄存器

- DAC启动之后，可以在while(1)中给它的数据寄存器赋值，这样就可以通过它输出所需要的模拟电压信号.
- 给DAC的数据寄存器赋值，可以使用库函数：HAL_DAC_SetValue().
- 在while(1)中添加如下代码：

```
while (1)
{
    /* USER CODE BEGIN 3 */
        DACIndex++;
        if (DACIndex == 4096)
            DACIndex = 0;
        HAL_DAC_SetValue(&hdac1, DAC_CHANNEL_1, DAC_ALIGN_12B_R, DACIndex);
        HAL_Delay(10);
    }
    /* USER CODE END 3 */
```

DAC数据输出寄存器

- 变量DACIndex，该变量逐步增加，到4095后，再从0开始。当然，需要在main函数中声明该变量。此处，将其声明为全局变量，放到main函数前面的注释对中：

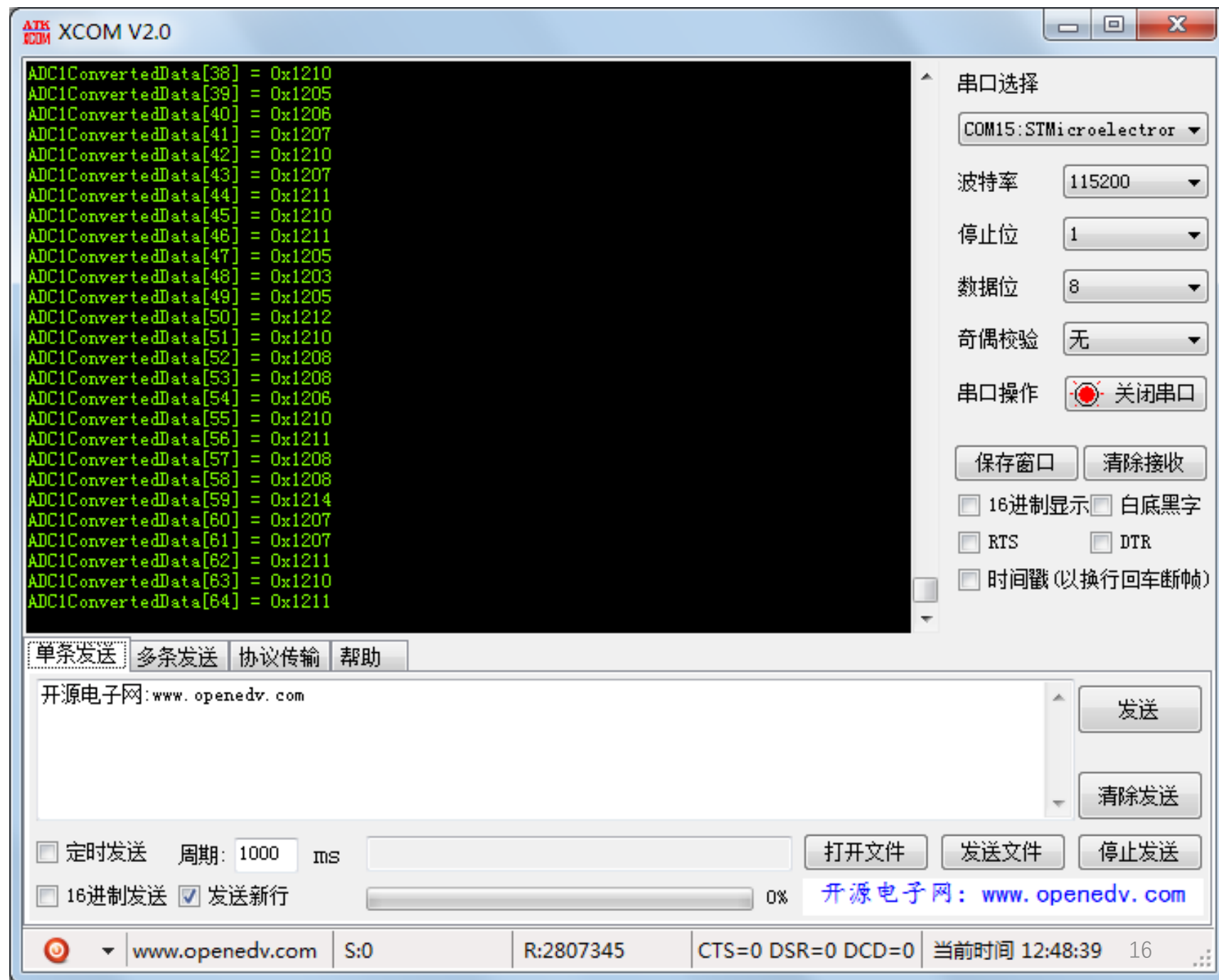
```
/* USER CODE BEGIN PV */  
  
uint16_t DACIndex = 0;  
  
/* USER CODE END PV */
```

编译、下载，运行

- 编译工程，并下载到硬件中运行
- 可以通过示波器或万用表来测量PA4引脚上的电压。
- 在NUCLEO-G431RB板上，PA4通过CN7端子的第32引脚或CN8的第3引脚引出

没有测量设备，如何测量？

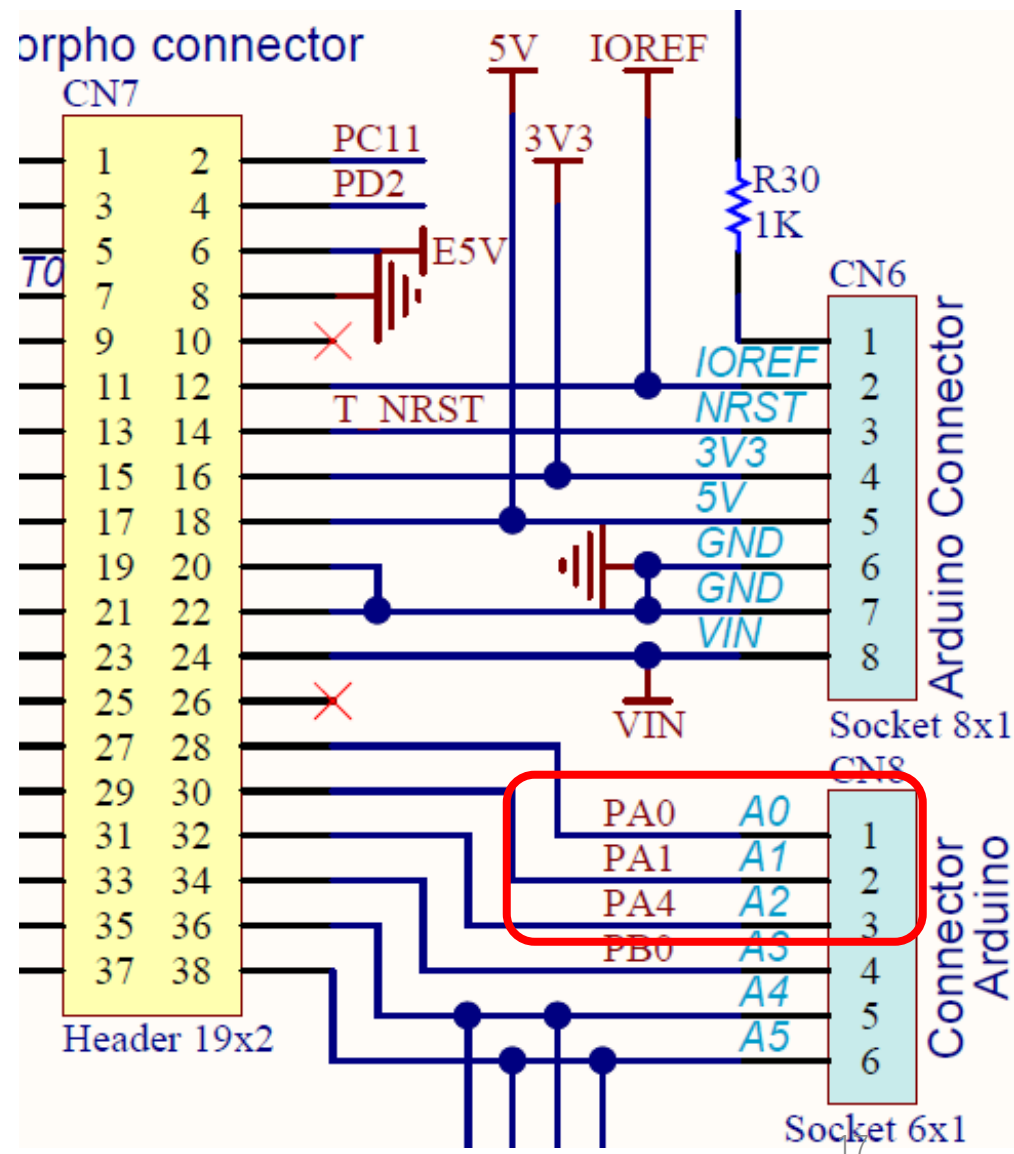
ADC采样 串口送出数据



练习8: DAC

任务8.1、在**任务7.3**的基础上，加入DAC1。利用ADC采样DAC输出的模拟信号，通过串口显示结果。

DAC1_OUT1对应PA4



用定时器控制DAC输出

用定时器控制DAC输出

- 上面的例子中，是在while(1)循环中实现了DAC的输出，并且让DAC输出逐步变化
- DAC的使用比较简单。在硬件配置完成后，在代码中，实际上需要做的只有两步：
 - ✓ 一是用库函数HAL_DAC_Start()启动DAC
 - ✓ 二是用函数HAL_DAC_SetValue()给DAC的数据寄存器赋值
- 下面对上述代码进行修改。依然是在初始化阶段启动DAC，即函数HAL_DAC_Start()放置的位置不变，而将给DAC的数据寄存器赋值的语句放置到定时器的中断中。这里用定时器TIM4中断。

配置定时器TIM4

- 在TIM4的Mode栏，只需选择Clock Source为Internal Clock
- 在下面的配置栏，设置计数器的预分频因子为169
- 计数器周期设置为9

TIM4 Mode and Configuration

Mode	
Slave Mode	Disable
Trigger Source	Disable
Clock Source	Internal Clock

Configuration

Reset Configuration

Parameter Settings User Constants NVIC Settings DMA Settings

Configure the below parameters :

Search (Ctrl+F)

Counter Settings

Prescaler (PSC - 16 bits value)	169
Counter Mode	Up
Dithering	Disable
Counter Period (AutoReload Regis...	9
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable

Trigger Output (TRGO) Parameters

配置定时器TIM4

- 打开 TIM4 的 NVIC Settings，使能中断

TIM4 Mode and Configuration

Mode

Slave Mode

Trigger Source

Clock Source

Configuration

☒ Parameter Settings ☒ User Constants ☒ NVIC Settings ☒ DMA Settings

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
TIM4 global interrupt	<input checked="" type="checkbox"/>	0	0

Software Packs

Pinout



Categories

A->Z

DMA

GPIO

IWDG

NVIC

✓ RCC

⚠ SYS

WWDG

Analog



Timers



HRTIM1

LPTIM1

RTC

TIM1

TIM2

✓ TIM3

✓ TIM4

TIM5

NVIC Mode and Configuration

Configuration

✓ NVIC

✓ Code generation

Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Prefetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0	0
PVD/PVM1/PVM2/PVM3/PVM4 interrupts through EXTI lines 16/38/39/40/...	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
ADC1 and ADC2 global interrupt	<input checked="" type="checkbox"/>	1	0
TIM3 global interrupt	<input type="checkbox"/>	0	0
TIM4 global interrupt	<input checked="" type="checkbox"/>	0	0
USART2 global interrupt / USART2 wake-up interrupt through EXTI line 26	<input type="checkbox"/>	0	0
TIM6 global interrupt, DAC1 and DAC3 channel underrun error interrupts	<input type="checkbox"/>	0	0
FPU global interrupt	<input type="checkbox"/>	0	0

修改代码

- 保存硬件配置界面 (*.ioc)，启动代码生成
- 打开main.c
- 加入初始化TIM4中断语句

```
/* USER CODE BEGIN 2 */  
HAL_ADCEx_Calibration_Start(&hadc1, ADC_SINGLE_ENDED);  
HAL_ADC_Start_IT(&hadc1);  
HAL_TIM_Base_Start(&htim3);  
HAL_TIM_Base_Start_IT(&htim4);  
HAL_DAC_Start(&hdac1, DAC_CHANNEL_1);  
/* USER CODE END 2 */
```

重新定义中断回调函数

- 在主程序中重写回调函数HAL_TIM_PeriodElapsedCallback()，放到main.c中的注释对中

```
/* USER CODE BEGIN 4 */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc)
{
    .....
}
int __io_putchar(int ch)
{
    .....
}
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if (htim == (&htim4))
    {
        DACIndex++;
        if (DACIndex == 4096)
            DACIndex = 0;
        HAL_DAC_SetValue(&hdac1, DAC_CHANNEL_1, DAC_ALIGN_12B_R, DACIndex);
    }
}
/* USER CODE END 4 */
```


此时输出波形的频率是多少？

- ☒ A 约24.4Hz
- ☐ B 约2.44Hz
- ☐ C 约244Hz
- ☐ D 其他

TIM4 Mode and Configuration

Counter Settings

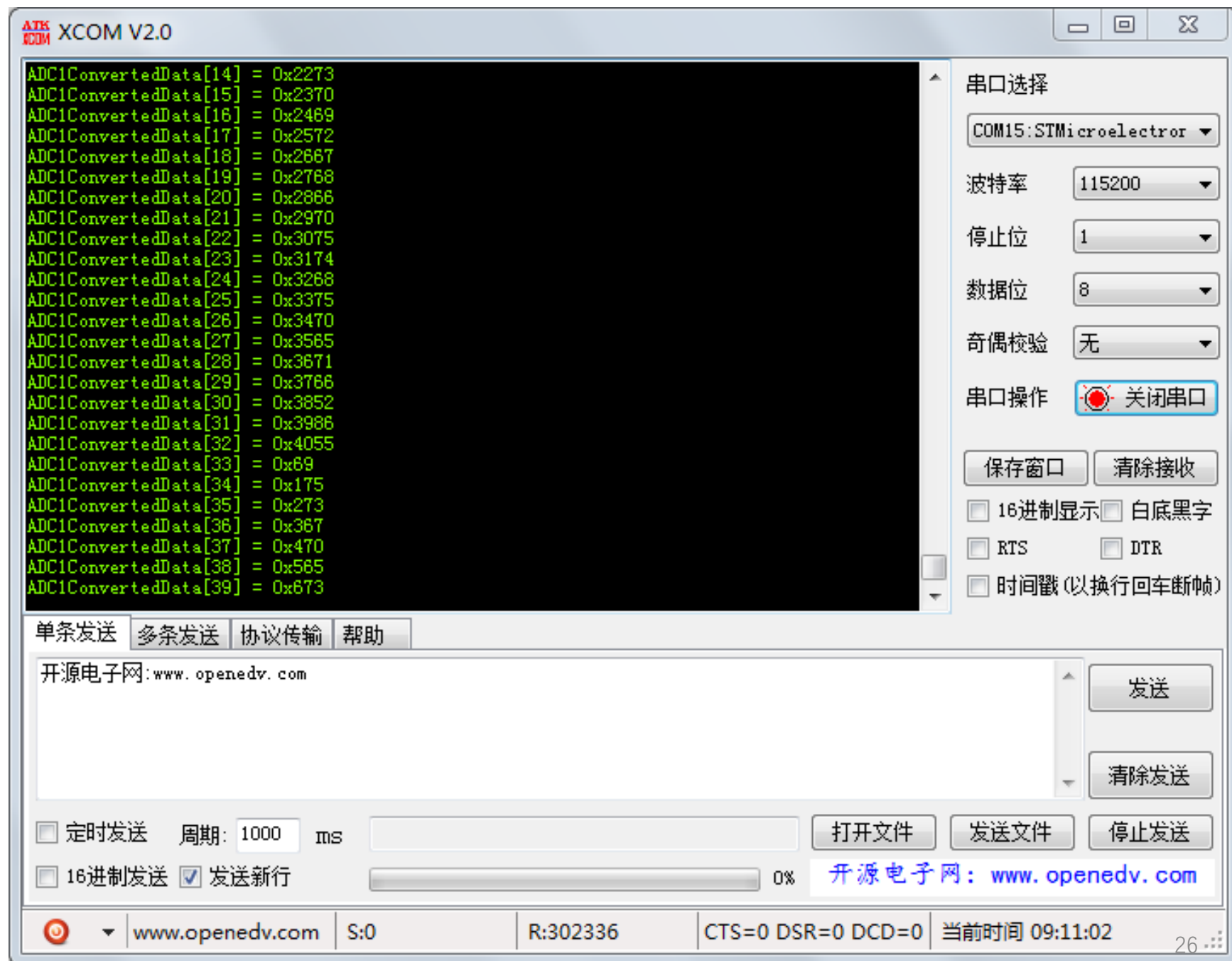
Prescaler (PSC - 16 bits value)	169
Counter Mode	Up
Dithering	Disable
Counter Period (AutoReload Regis...	9
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable

Trigger Output (TRGO) Parameters

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if (htim == (&htim4))
    {
        DACIndex++;
        if (DACIndex == 4096)
            DACIndex = 0;
        HAL_DAC_SetValue(&hdac1, DAC_CHANNEL_1, DAC_ALIGN_12B_R, DACIndex);
    }
}
```

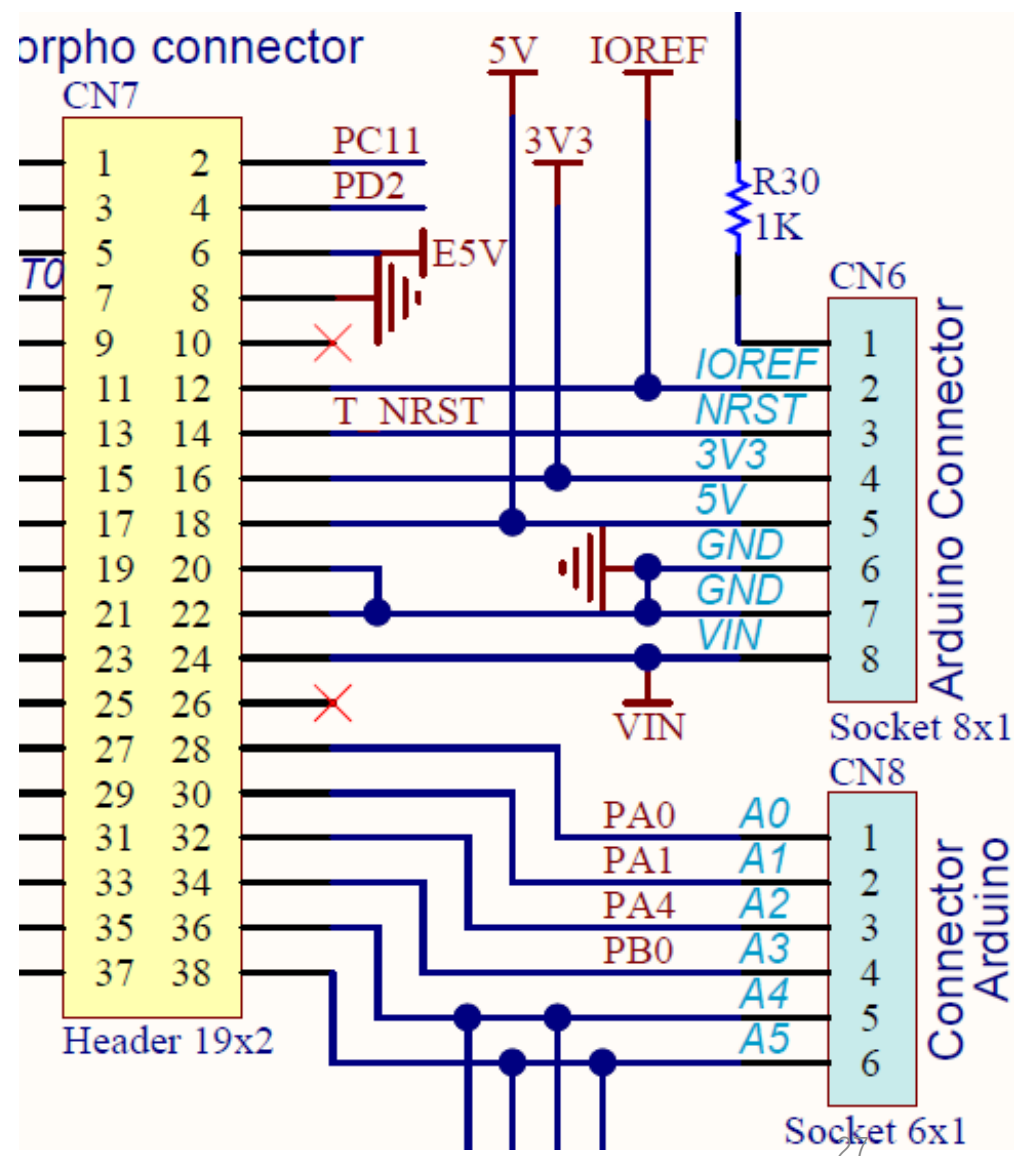
提交

ADC采样 串口送出数据



练习8: DAC

任务8.2、在**任务8.1**的基础上，加入TIM4。利用ADC采样DAC输出的模拟信号，通过串口显示结果。



如何看波形?

DIY示波器

构建包含ADC和DAC的测量系统

构建包含ADC和DAC的测量系统

- 利用STM32G431片上的ADC和DAC构建测量系统
- DAC输出作为ADC的输入信号
- 结合DAC、ADC和串口等模块，用ADC采集DAC的输出信号，并将结果通过串口送至PC机，在PC机上用Simulink来显示采集信号的波形

修改串口发送数据代码

- 将在while循环中发送串口数据的方式，修改为放到ADC回调函数中

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc)
{
    ADC1ConvertedValue = HAL_ADC_GetValue(&hadc1);
    HAL_UART_Transmit(&huart2, (uint8_t *)&ADC1ConvertedValue, 2, 0xFFFF);
}
```

- 定义变量：uint16_t ADC1ConvertedValue = 0
- 功能：直接将AD采样值通过串口发送出来
- 注意：在HAL_UART_Transmit()函数中，将发送的字节数改为2
- 注释掉while(1)循环中的代码，即while循环中不做什么事
- 编译、下载，运行

构建simulink模型

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



MATLAB®

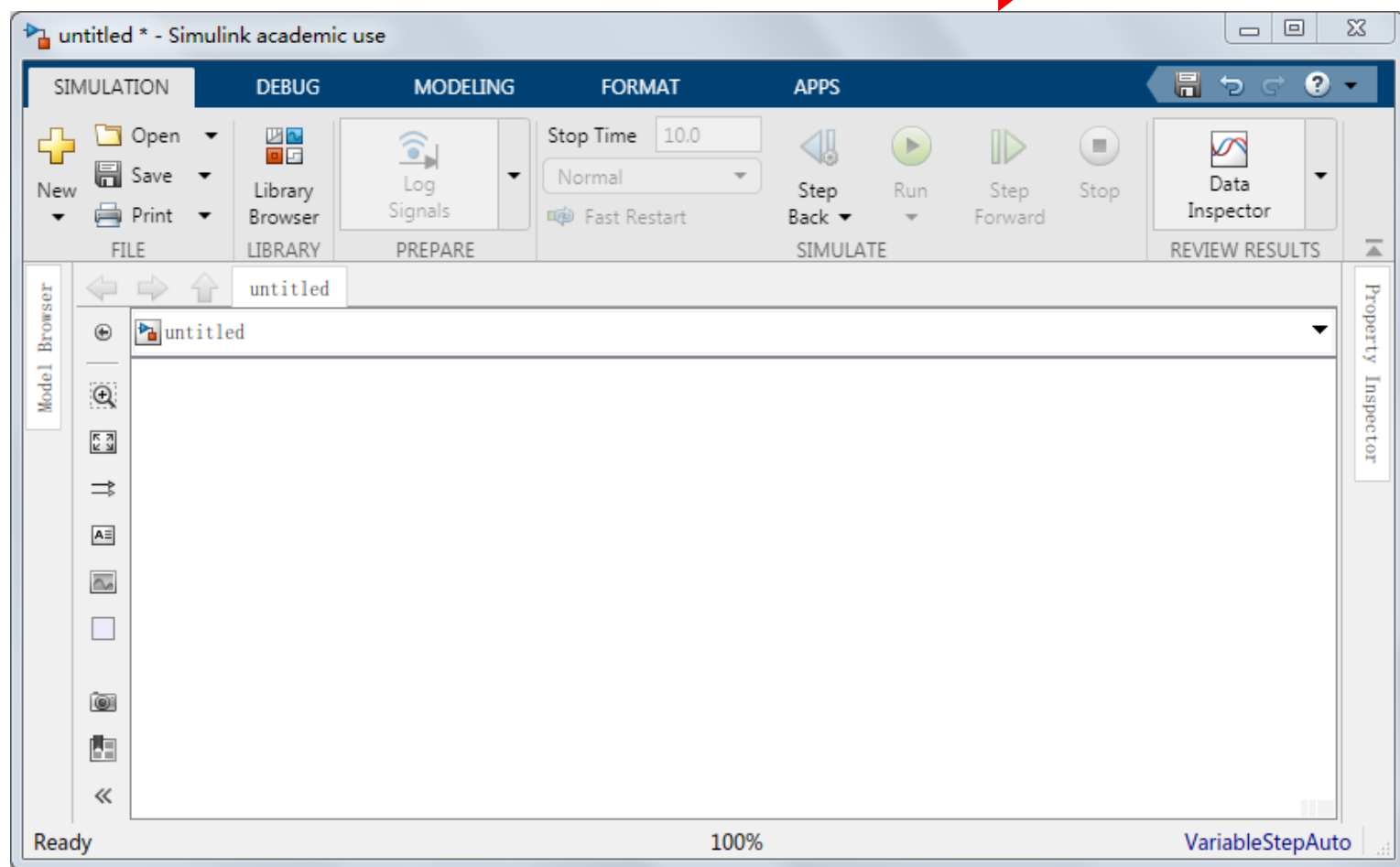
Academic License - for use in teaching, academic research, and meeting course requirements at degree granting institutions only. Not for government, commercial, or other organizational use.

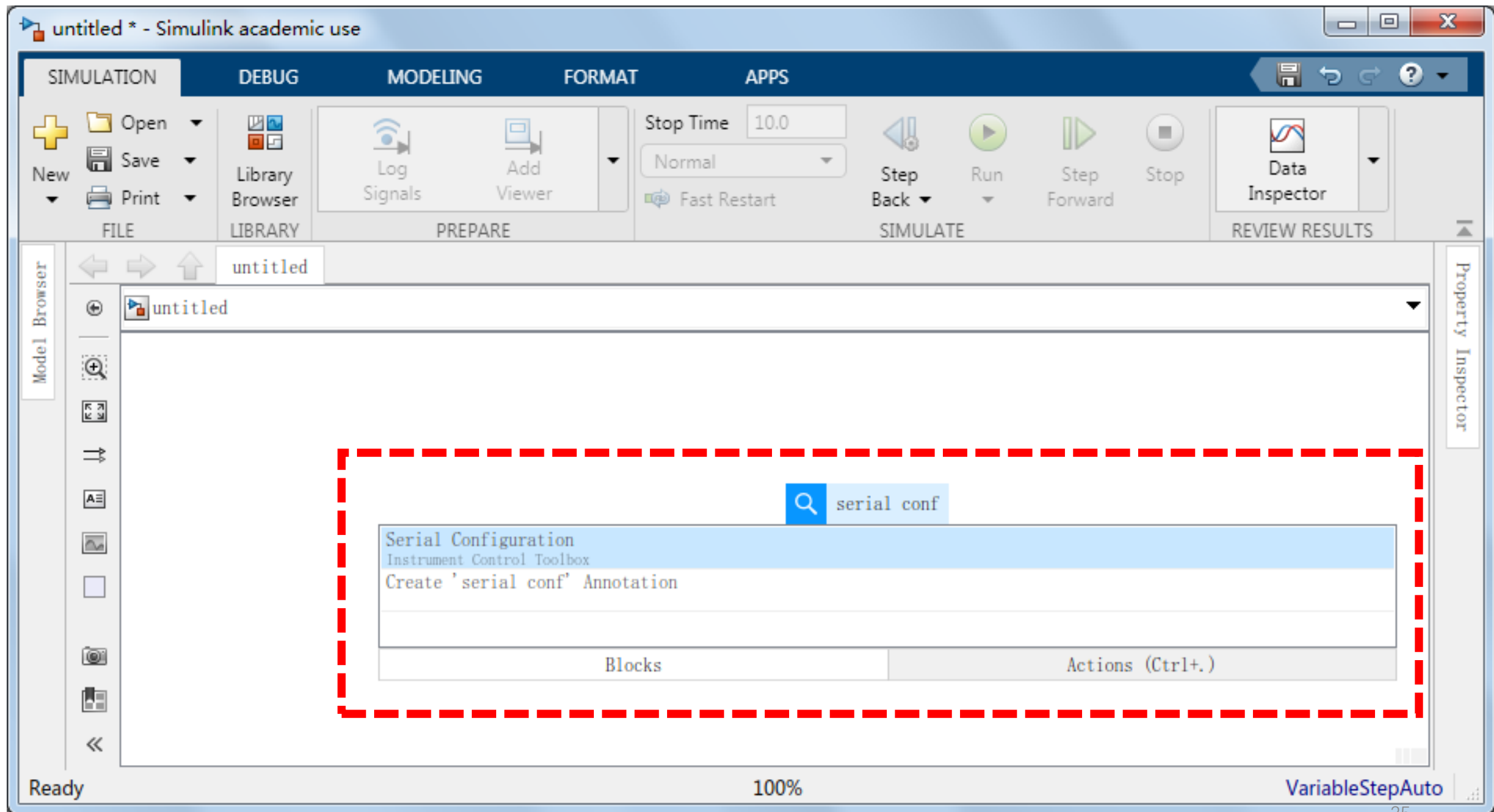
© 1984-2020 The MathWorks, Inc. Protected by U.S and international patents. See mathworks.com/patents. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks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



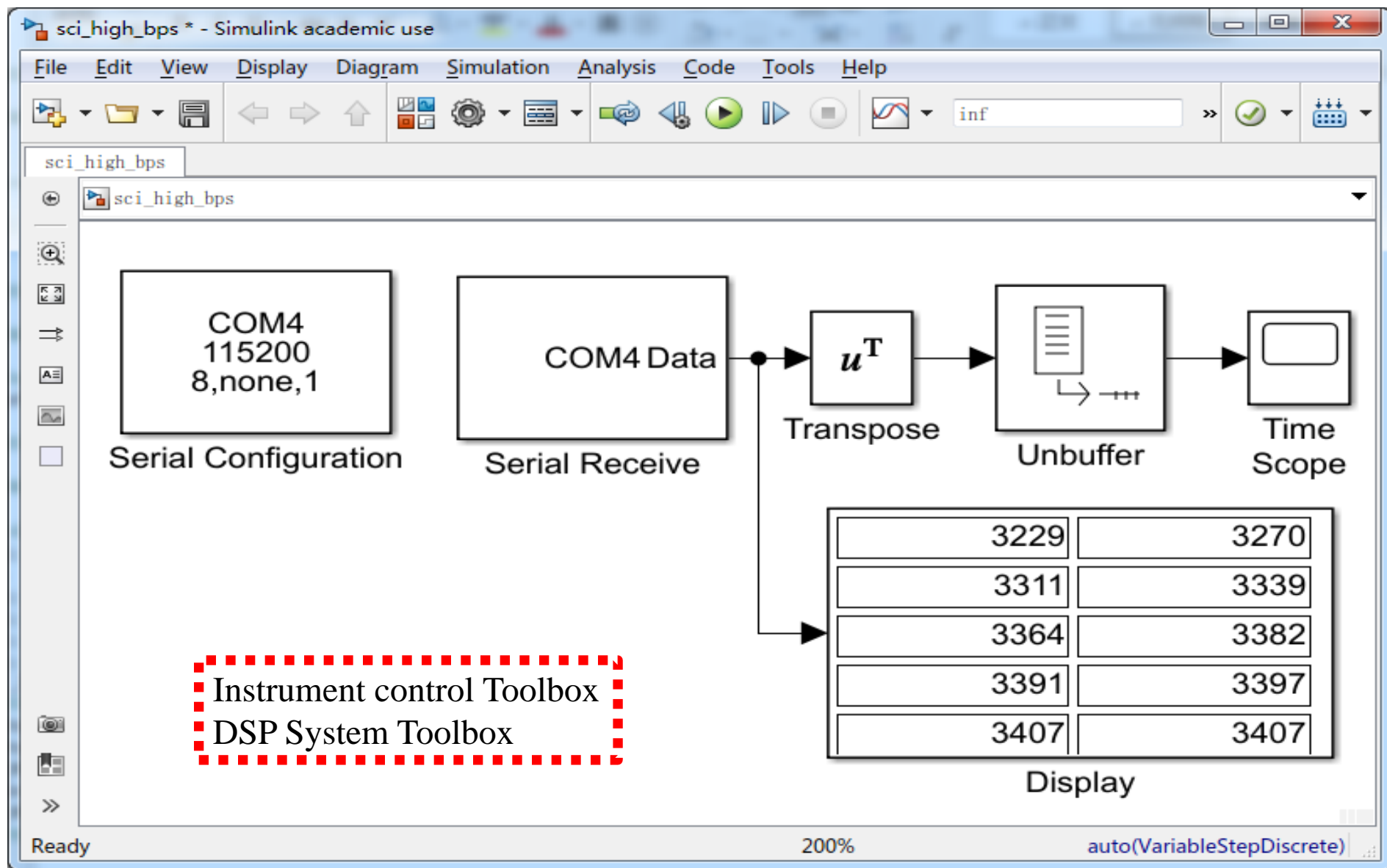
R2020a

别的版本也没问题

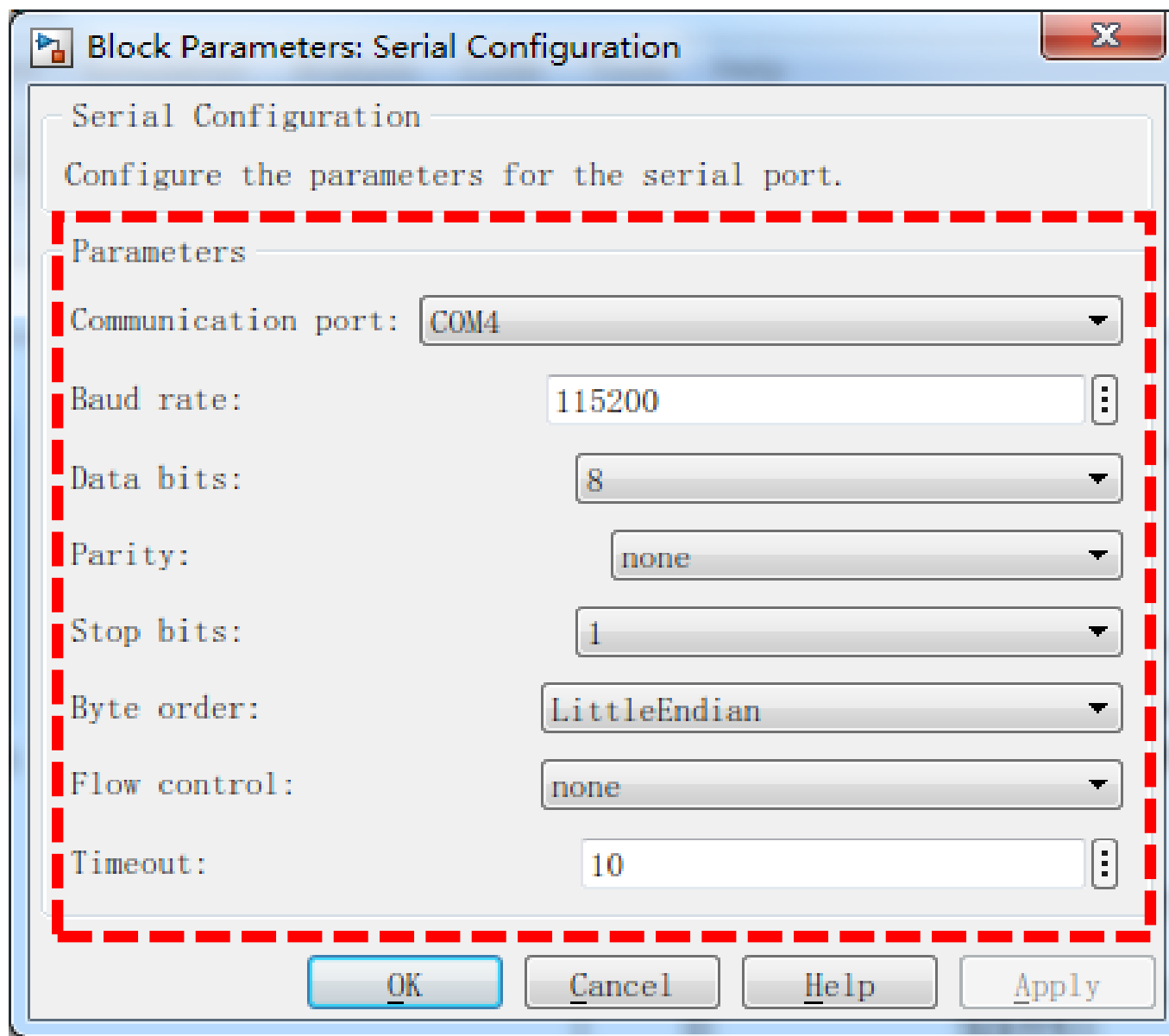




构建simulink模型



配置Serial Configuration模块



The image shows a Windows-style dialog box titled "Block Parameters: Serial Configuration". It contains a "Serial Configuration" section with the instruction "Configure the parameters for the serial port." Below this is a "Parameters" section, which is highlighted by a red dashed border. This section includes several configuration options: "Communication port" (set to COM4), "Baud rate" (set to 115200), "Data bits" (set to 8), "Parity" (set to none), "Stop bits" (set to 1), "Byte order" (set to LittleEndian), "Flow control" (set to none), and "Timeout" (set to 10). At the bottom of the dialog are four buttons: "OK", "Cancel", "Help", and "Apply".

Block Parameters: Serial Configuration

Serial Configuration

Configure the parameters for the serial port.

Parameters

Communication port: COM4

Baud rate: 115200

Data bits: 8

Parity: none

Stop bits: 1

Byte order: LittleEndian

Flow control: none

Timeout: 10

OK Cancel Help Apply

配置Serial Receive模块

- 数据类型选为int16
- 在Data size栏，方括号中
 - ✓ 第1个数为有几组数据
 - ✓ 第2个数为每组数据的长度
- 由于只有一路数据送上来，所以第1个数写为1，第2个数设置为60
- 最后一个采样时间设置为0.06s。

Block Parameters: Serial Receive

Serial Receive
Receive binary data over serial port.

Parameters

Communication port: COM4

Header:

Terminator: <none>

Data size: [1 60]

Data type: int16

☒ Enable blocking ...

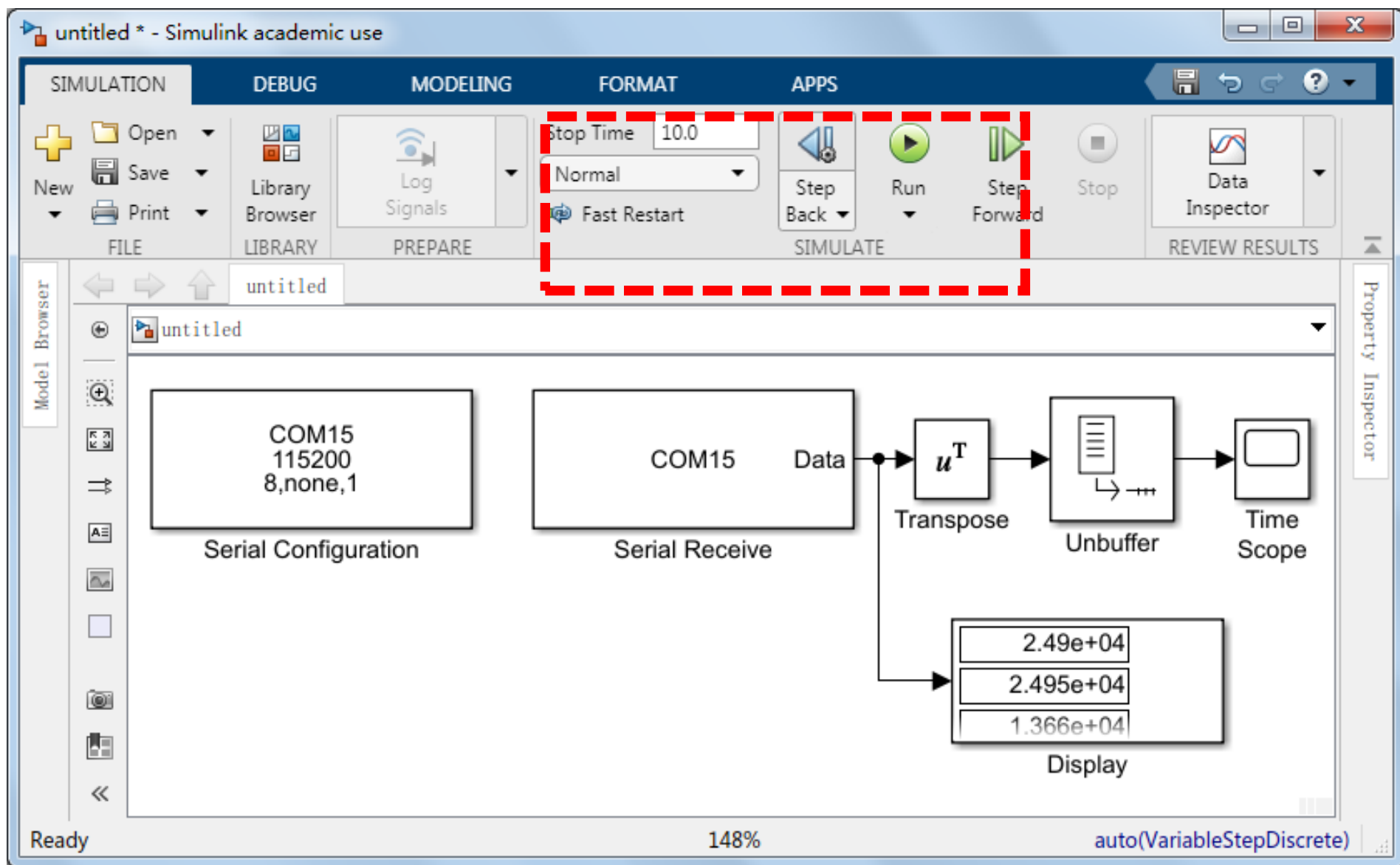
Action when data is unavailable: Output last received value

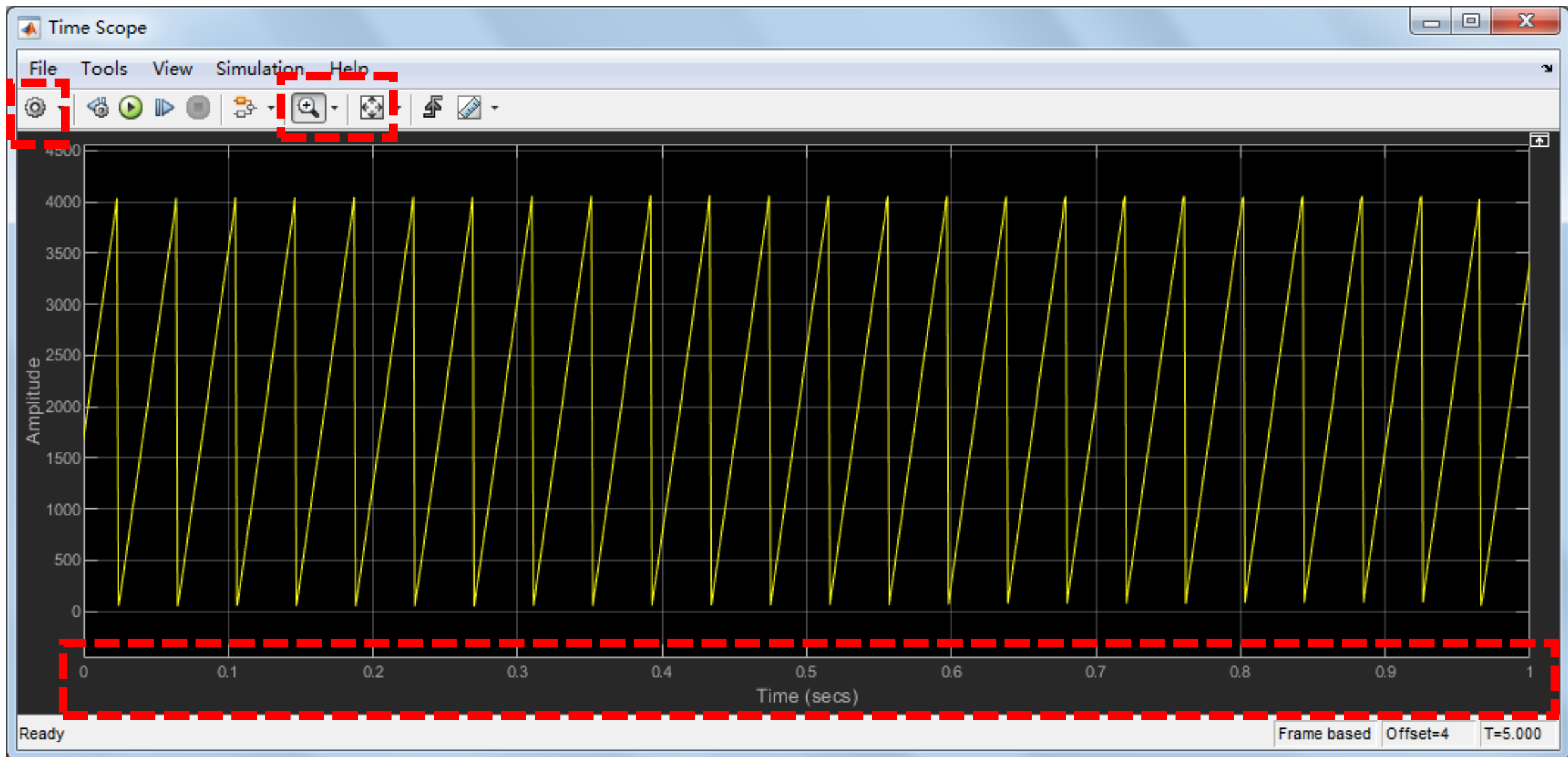
Custom value: 0

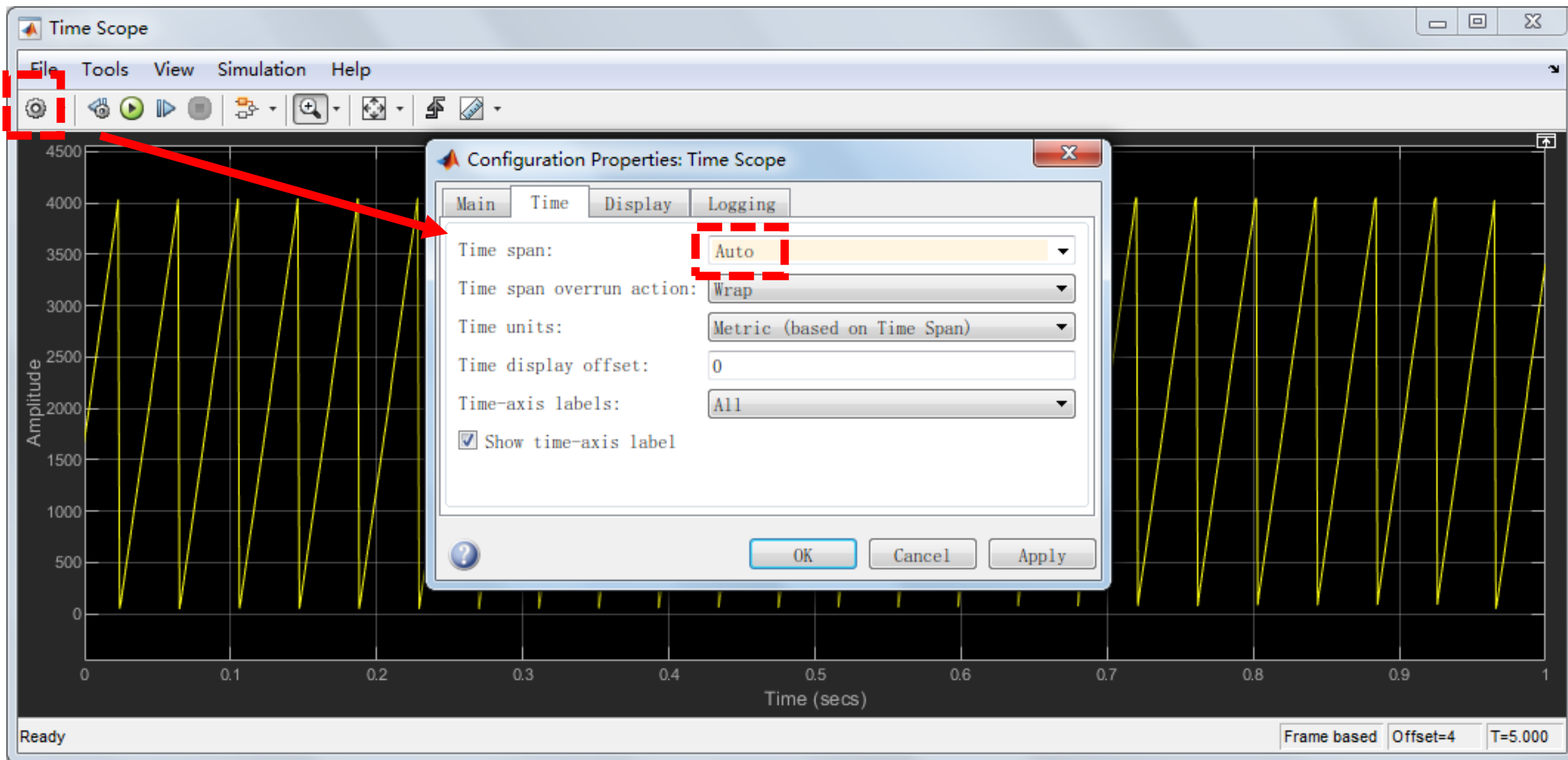
Block sample time: 0.06

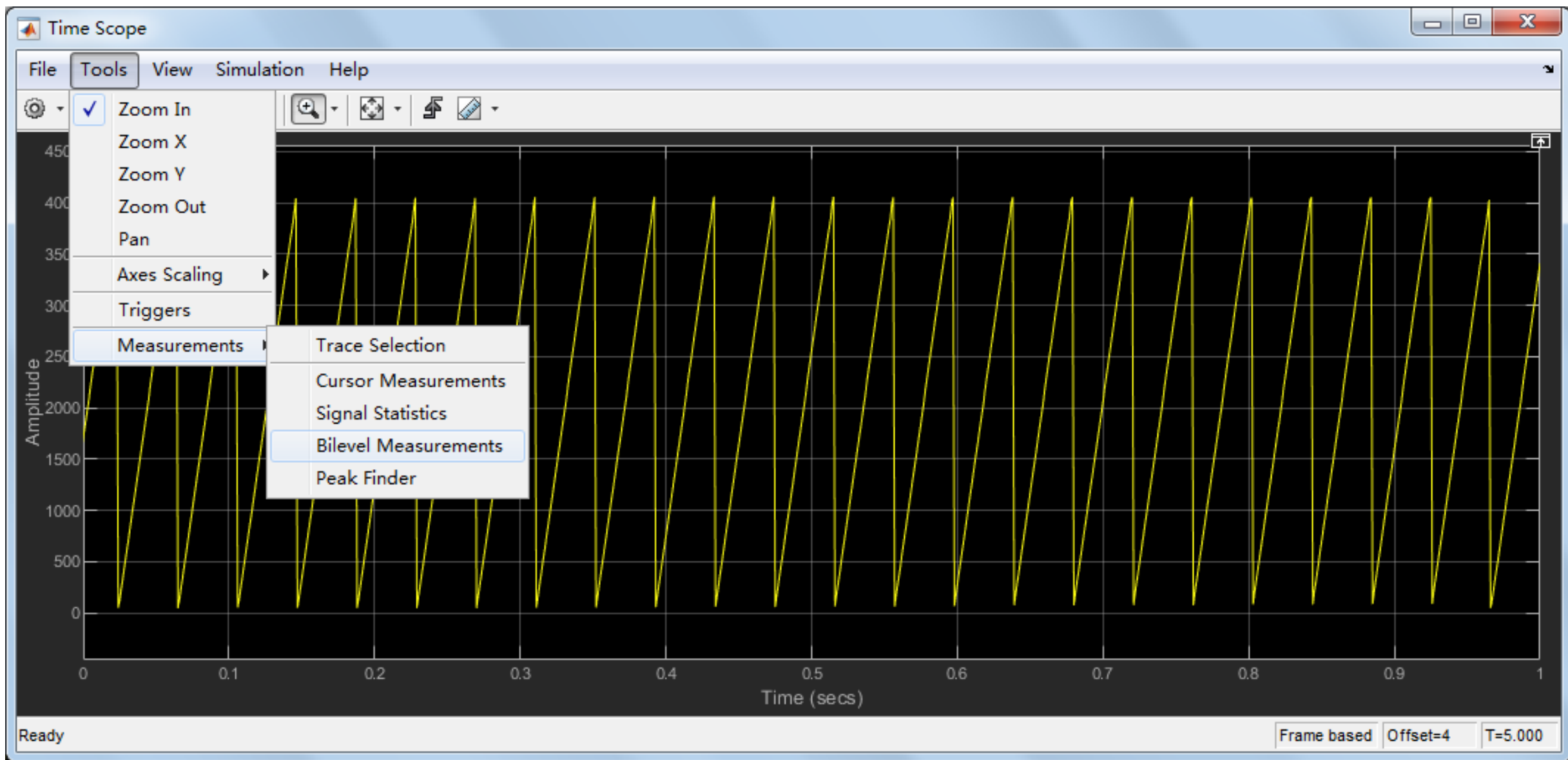
OK Cancel Help Apply

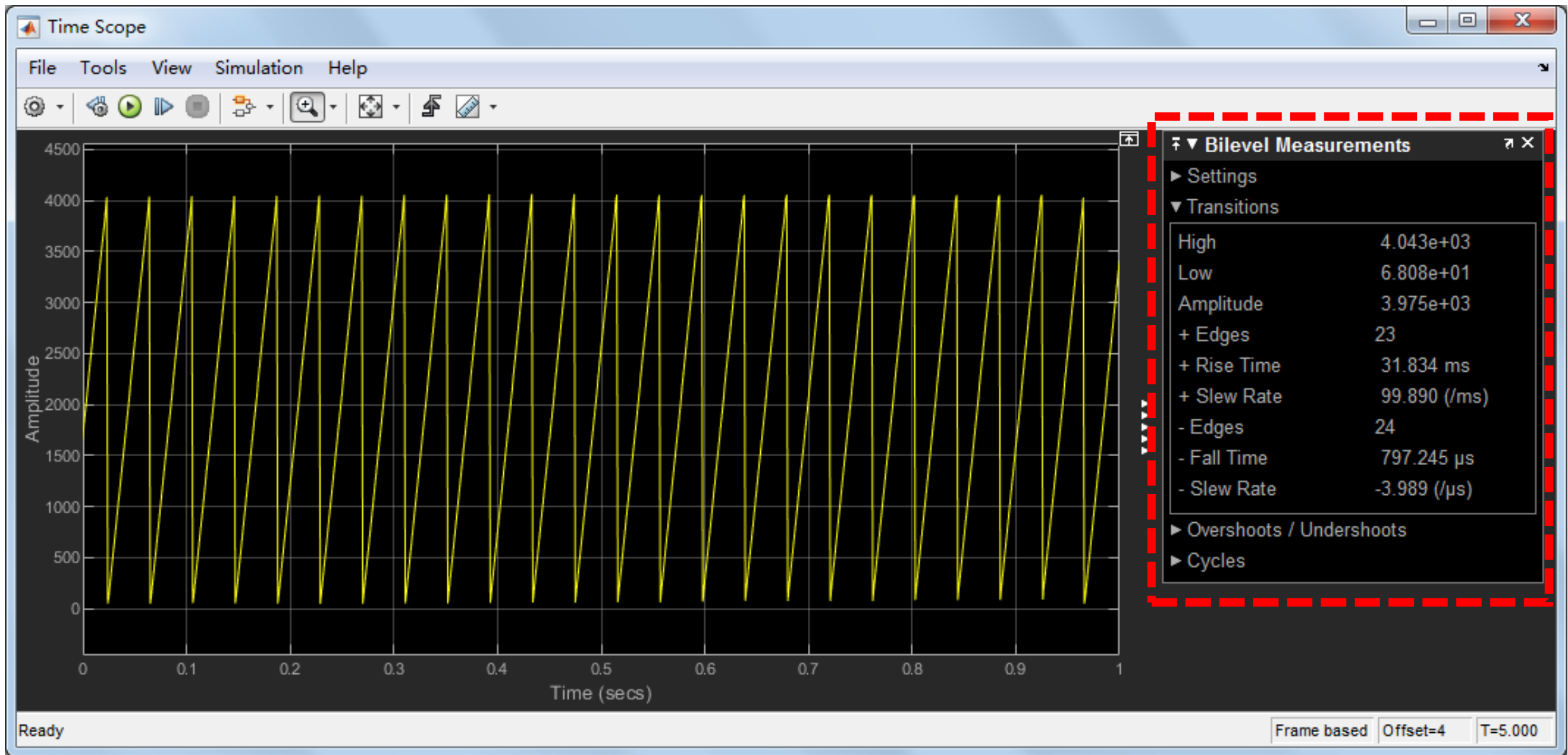
运行simulink模型

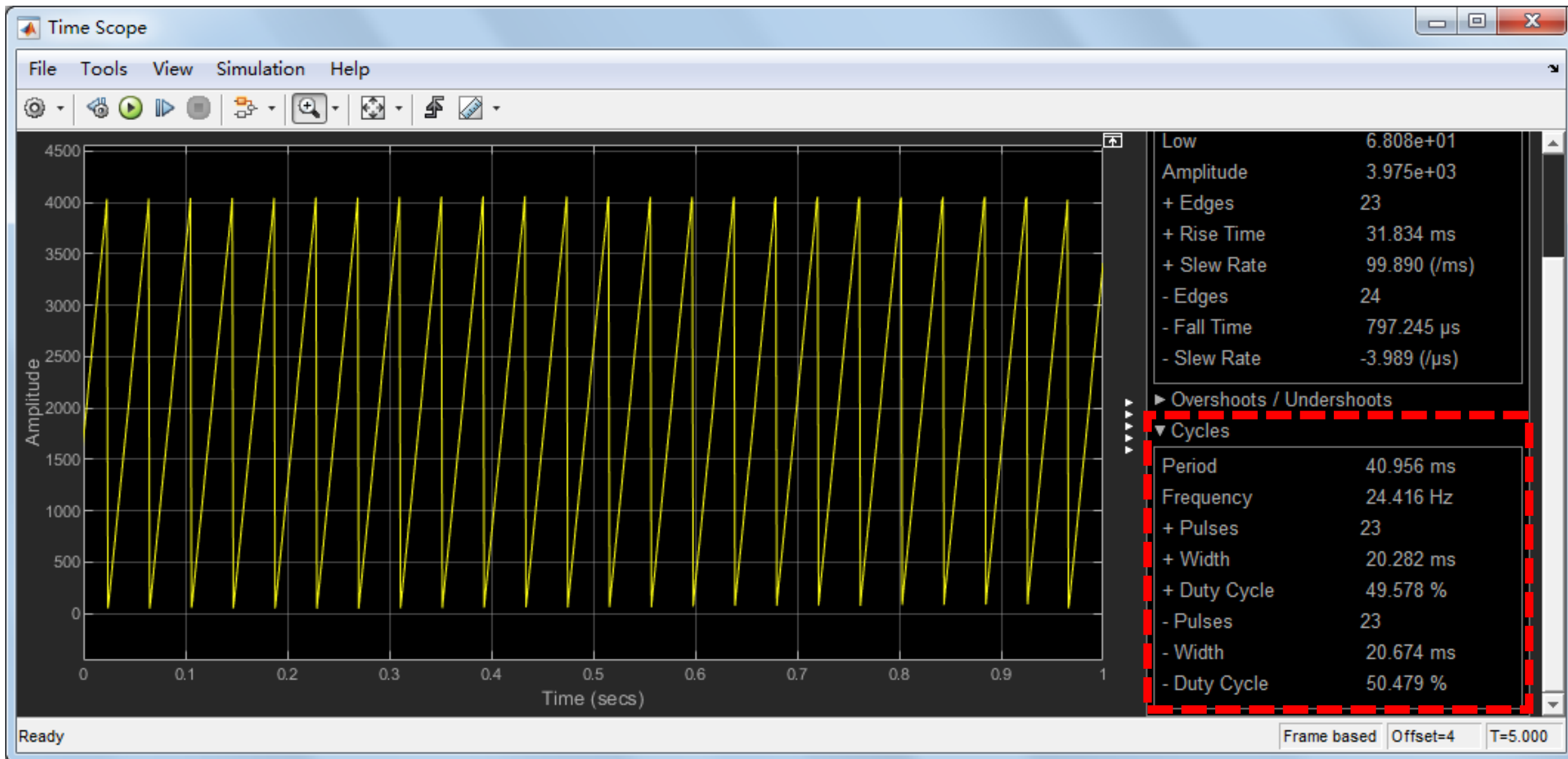












▼ Counter Settings

Prescaler (PSC - 16 bits value)	169
Counter Mode	Up
Dithering	Disable
Counter Period (AutoReload R...	9

▼ Cycles

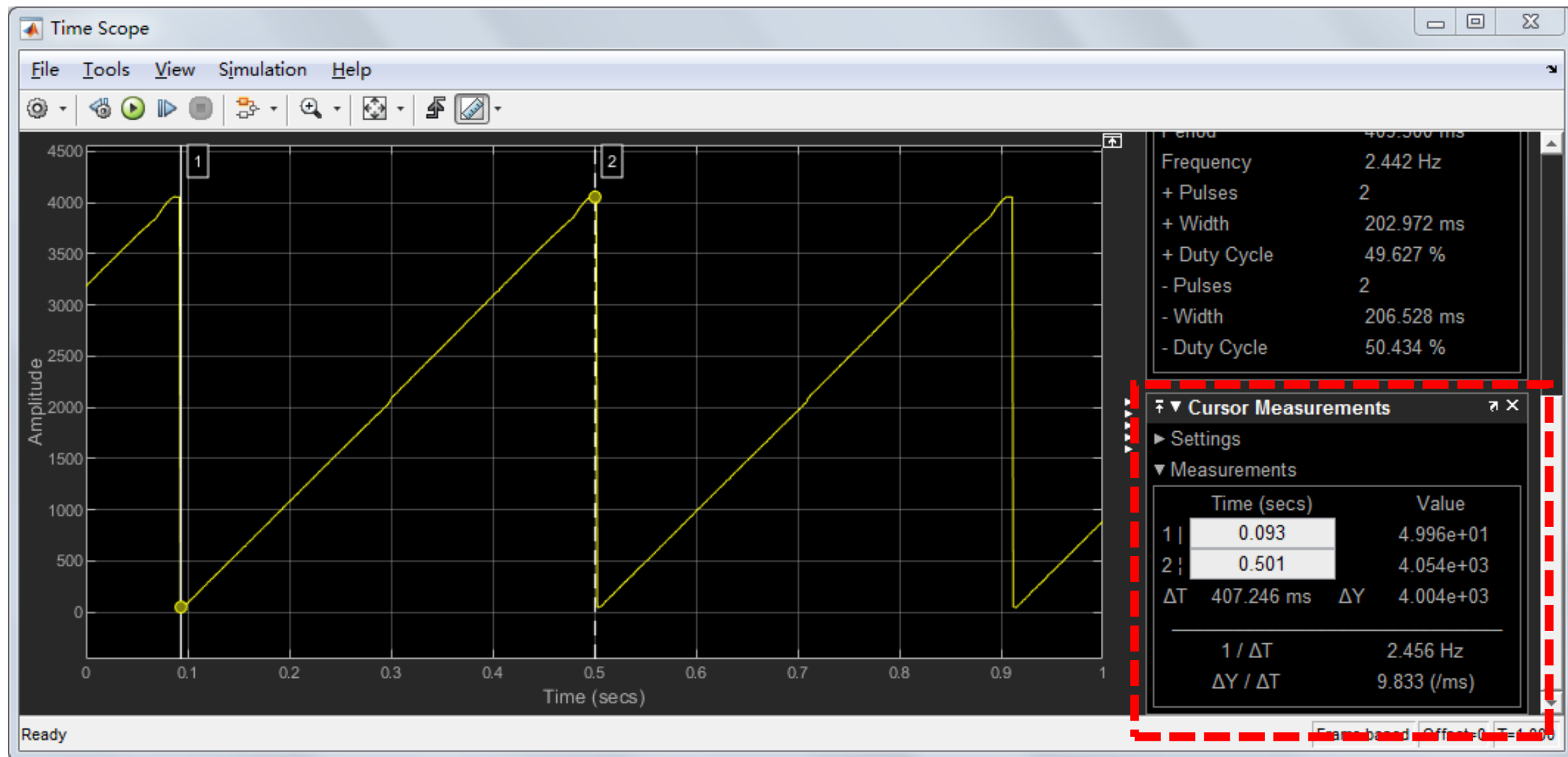
Period	40.956 ms
Frequency	24.416 Hz
+ Pulses	23
+ Width	20.282 ms
+ Duty Cycle	49.578 %
- Pulses	23
- Width	20.674 ms
- Duty Cycle	50.479 %

▼ Counter Settings

Prescaler (PSC - 16 bits value)	169
Counter Mode	Up
Dithering	Disable
Counter Period (AutoReload R...	99

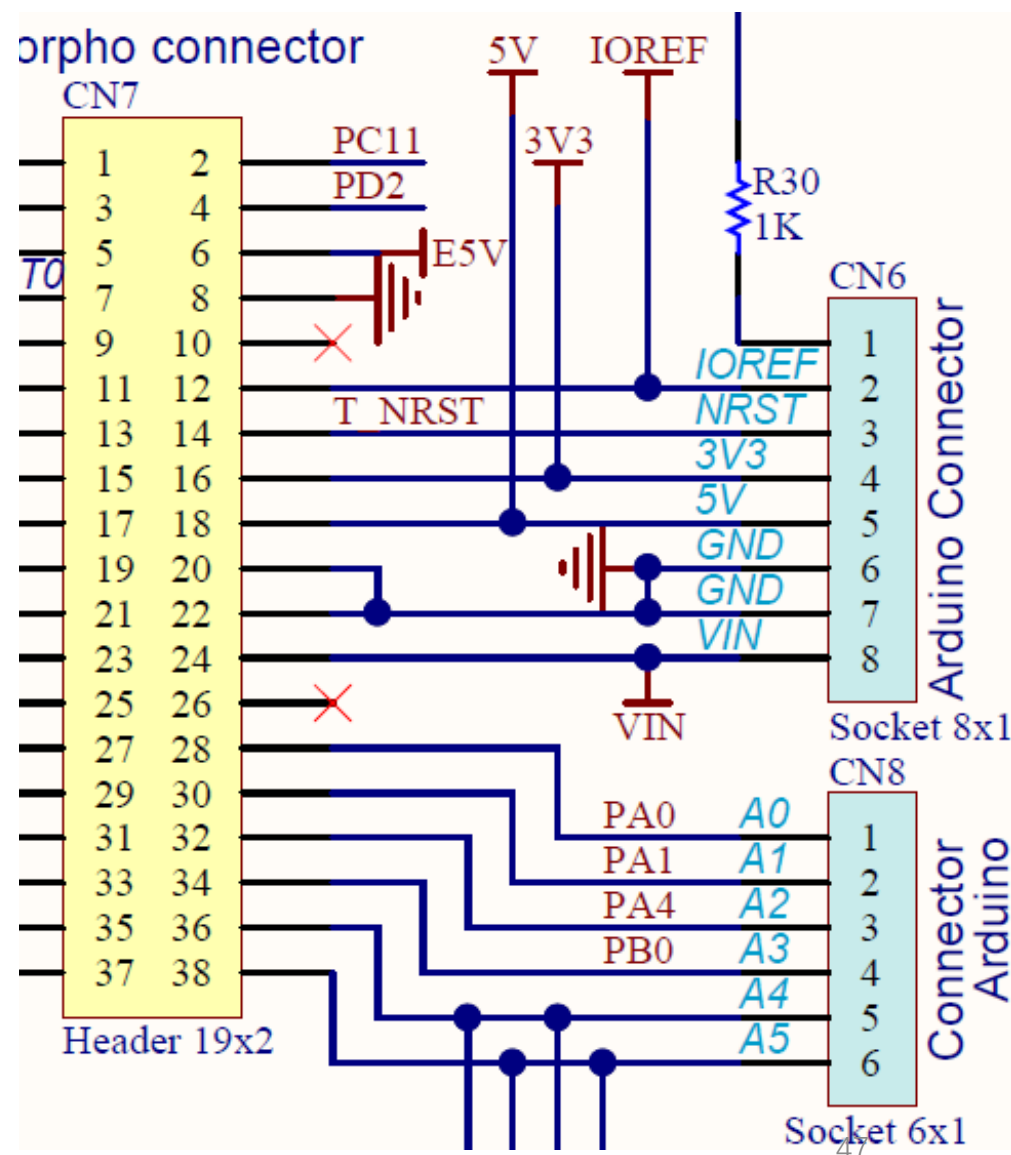
▼ Cycles

Period	409.500 ms
Frequency	2.442 Hz
+ Pulses	2
+ Width	202.972 ms
+ Duty Cycle	49.627 %
- Pulses	2
- Width	206.528 ms
- Duty Cycle	50.434 %



练习8: DAC

任务8.3、参照上述步骤，搭建
simulink模型，实现对DAC输出波形
的测量。通过修改TIM4参数，改变
输出信号频率为学号的后两位（如
果小于10，放大10倍）。查看结果。



中断优先级

Categories

A->Z

System Core

DMA

GPIO

IWDG

NVIC

✓ RCC

▲ SYS

WWDG

Analog

Timers

Connectivity

Multimedia

Security

NVIC Mode and Configuration

Configuration

✓ NVIC

✓ Code generation

NVIC Interrupt Table	Enabled	Preemption Prio...	Sub Priority
Non maskable interrupt	✓	0	0
Hard fault interrupt	✓	0	0
Memory management fault	✓	0	0
Prefetch fault, memory access fault	✓	0	0
Undefined instruction or illegal state	✓	0	0
System service call via SWI instruction	✓	0	0
Debug monitor	✓	0	0
Pendable request for system service	✓	0	0
Time base: System tick timer	✓	0	0
PVD/PVM1/PVM2/PVM3/PVM4 interrupts thro...	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
ADC1 and ADC2 global interrupt	<input checked="" type="checkbox"/>	1	0
TIM3 global interrupt	<input type="checkbox"/>	0	0
TIM4 global interrupt	<input checked="" type="checkbox"/>	0	0
USART2 global interrupt / USART2 wake-up int...	<input type="checkbox"/>	0	0
TIM6 global interrupt, DAC1 and DAC3 channel...	<input type="checkbox"/>	0	0
FPU global interrupt	<input type="checkbox"/>	0	0

如果TIM4的
中断优先级
比ADC低，
会有何结果？

练习8：DAC

任务8.4、在8.3的基础上，改变TIM4或ADC1中断的优先级，通过波形，分析其对结果的影响。

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

谢 谢!