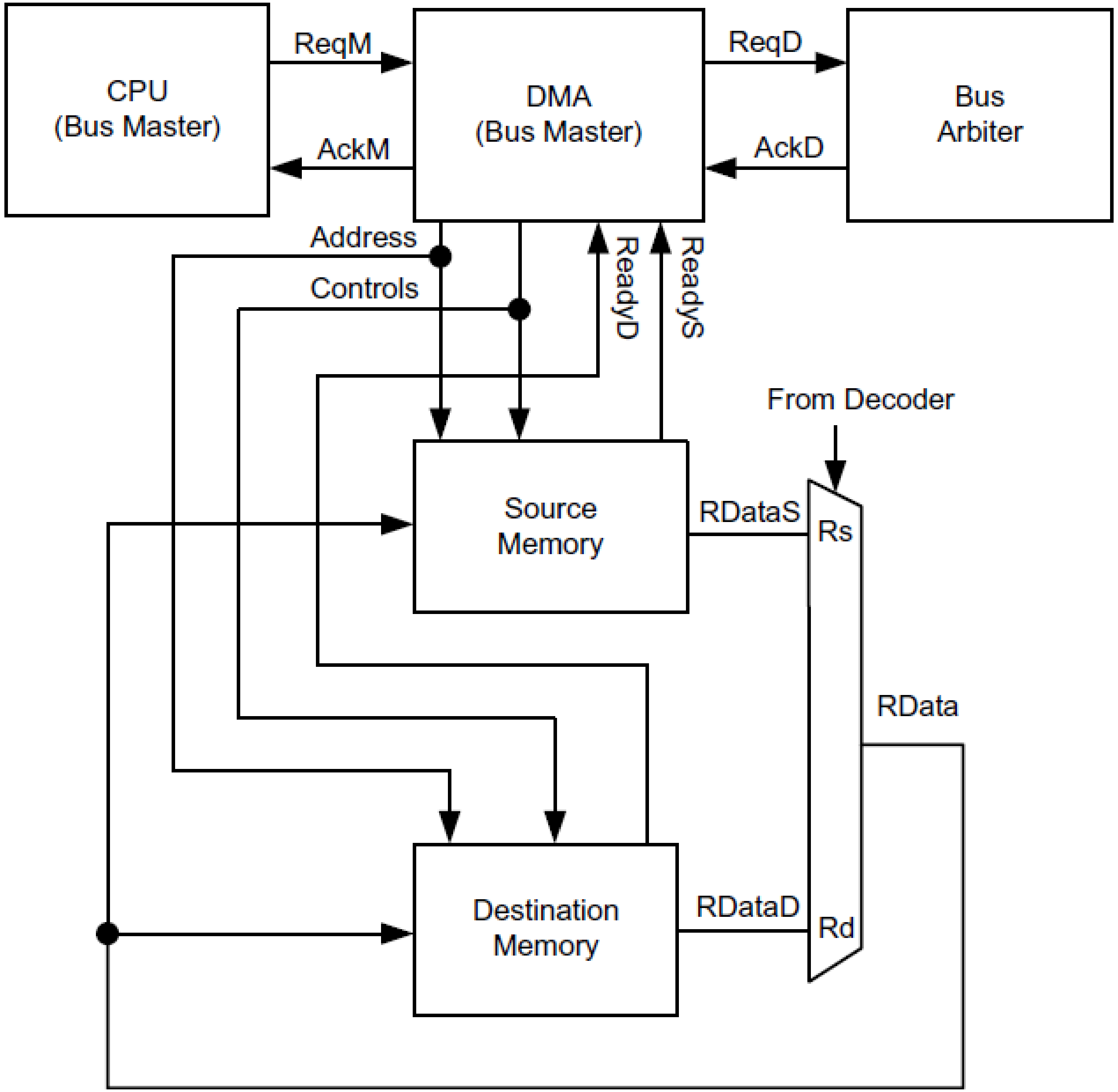
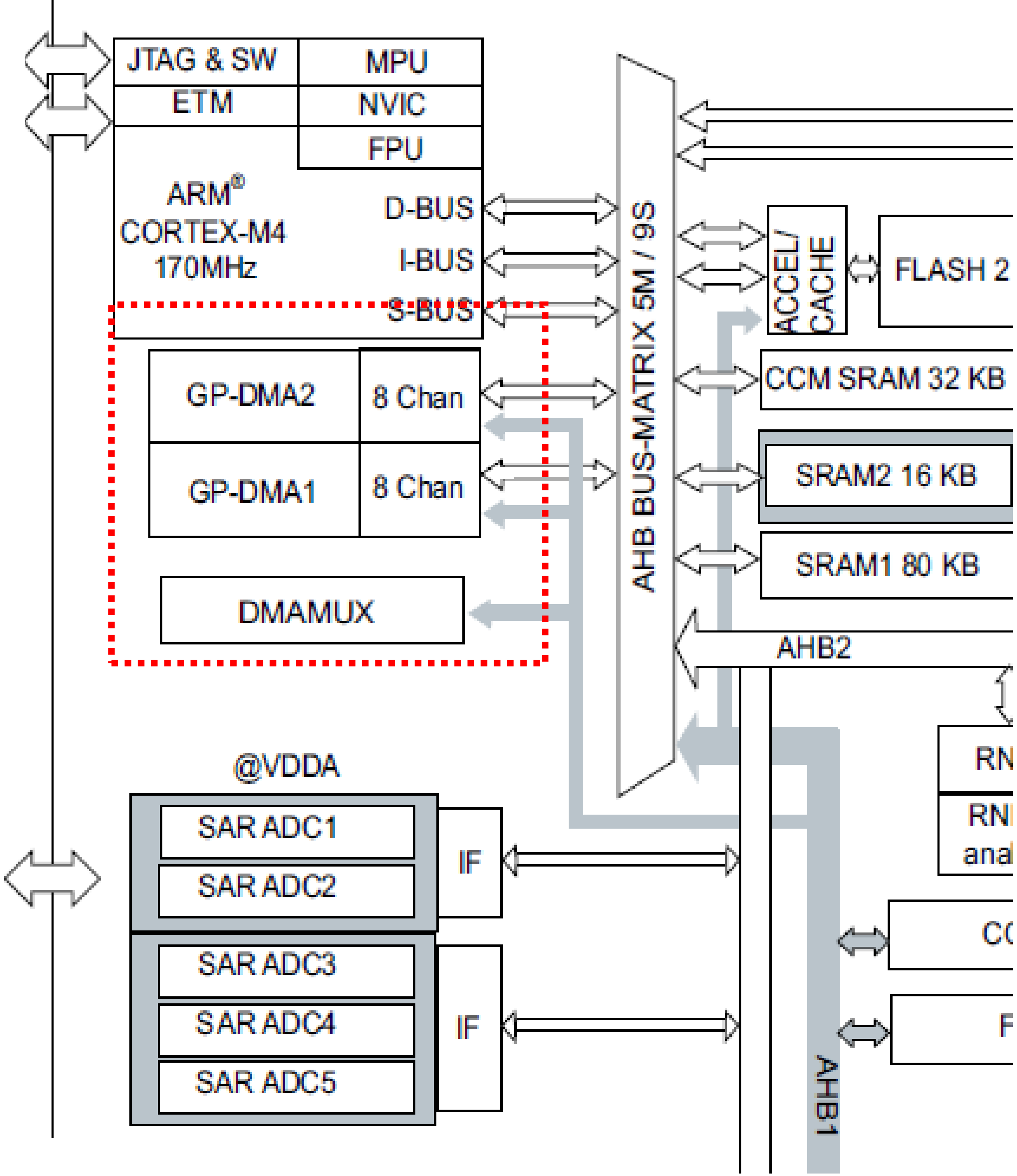
第9讲

DMA

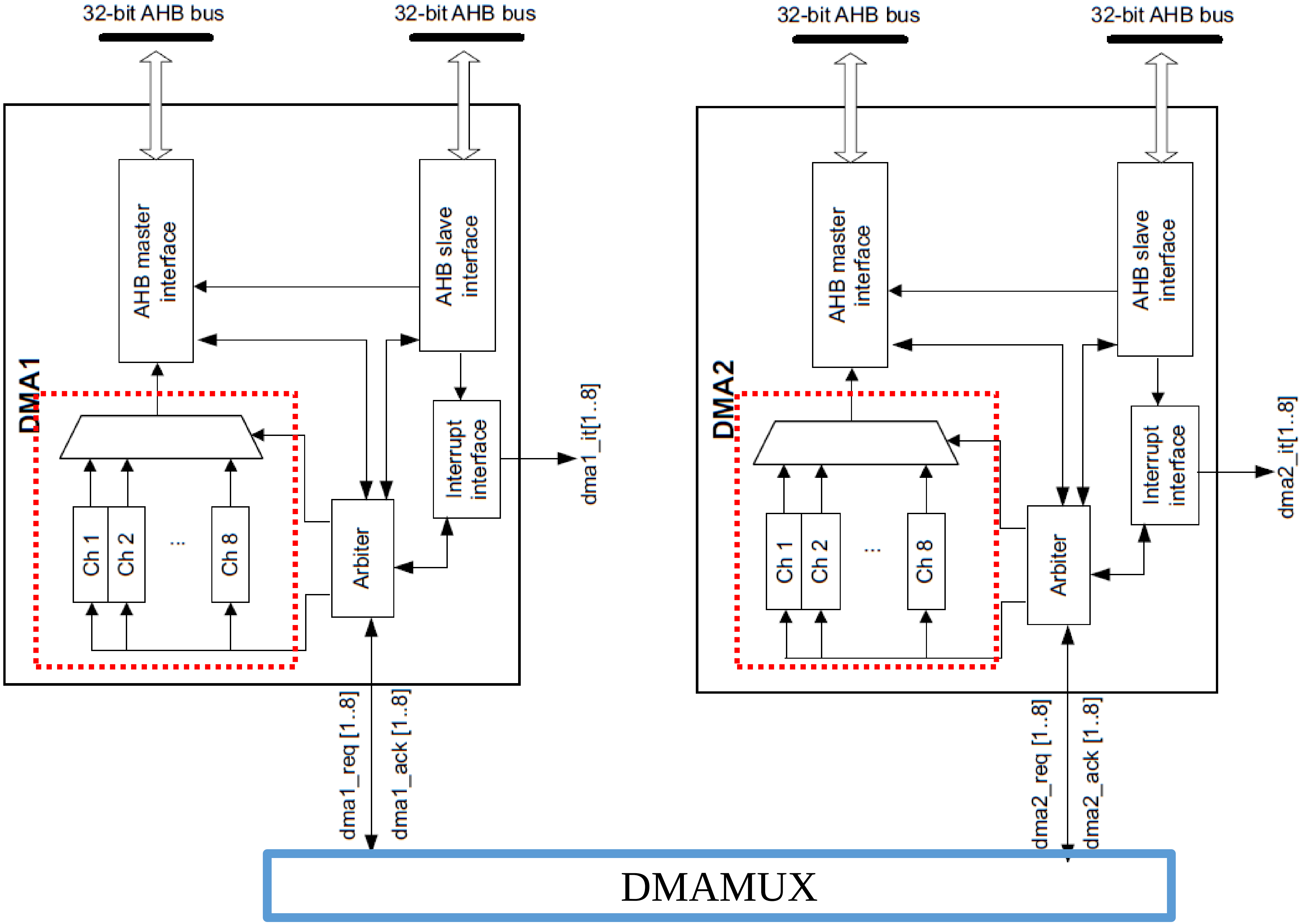
主要内容

- STM32G4的DMA
- 用DMA实现DAC输出
- 用DMA实现ADC采样
- 动手练习9


CPU
(Bus Master)
ReqM
AckM
DMA
(Bus Master)
ReqD
AckD
Bus
Arbiter
Address
Controls
ReadyD
ReadyS
From Decoder
Source
Memory
RDataS
Rs
RData
Destination
Memory
RDataD
Rd



DMAMUX:
DMA request multiplexer


32-bit AHB bus
32-bit AHB bus
AHB master interface
AHB slave interface
DMA1
Ch 1
Ch 2
...
Ch 8
Arbiter
Interrupt interface
dma1_it[1..8]
dma1_req [1..8]
dma1_ack [1..8]
32-bit AHB bus
32-bit AHB bus
AHB master interface
AHB slave interface
DMA2
Ch 1
Ch 2
...
Ch 8
Arbiter
Interrupt interface
dma2_it[1..8]
dma2_req [1..8]
dma2_ack [1..8]
DMAMUX

用DMA实现DAC输出

- 利用DMA控制器实现一种数据的传递工作
- 用DMA的方式将位于存储器（通过数组形式访问）中的数据传递给DAC的数据输出寄存器
- 放到存储器（数组）中的数据，可以是一段波形数据（正弦波、锯齿波等），譬如一个周期的数据，利用DMA周期性地将该数据传递到DAC，就可以实现周期信号的输出

建立新工程

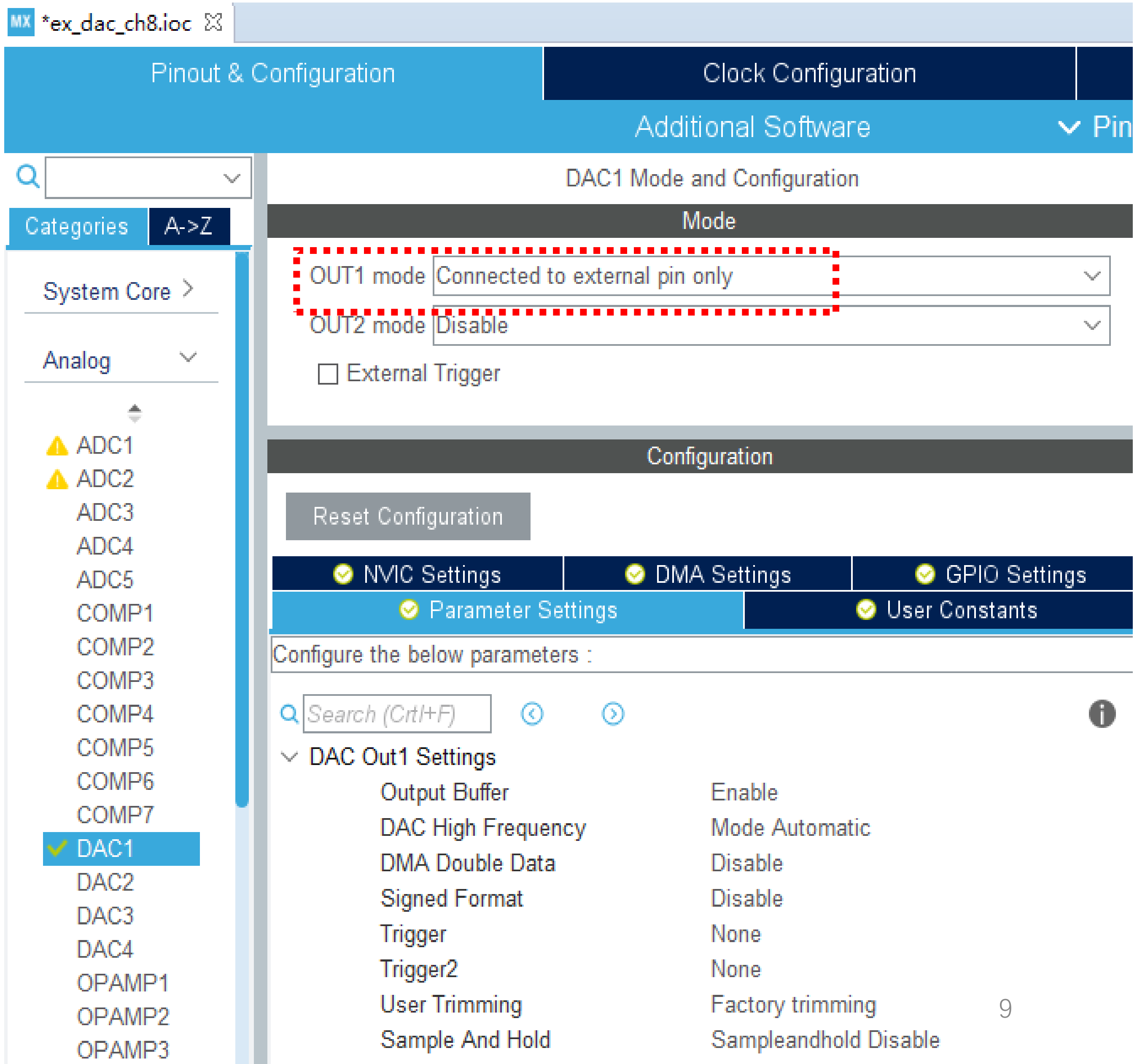
时钟源与Debug模式配置

■ 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

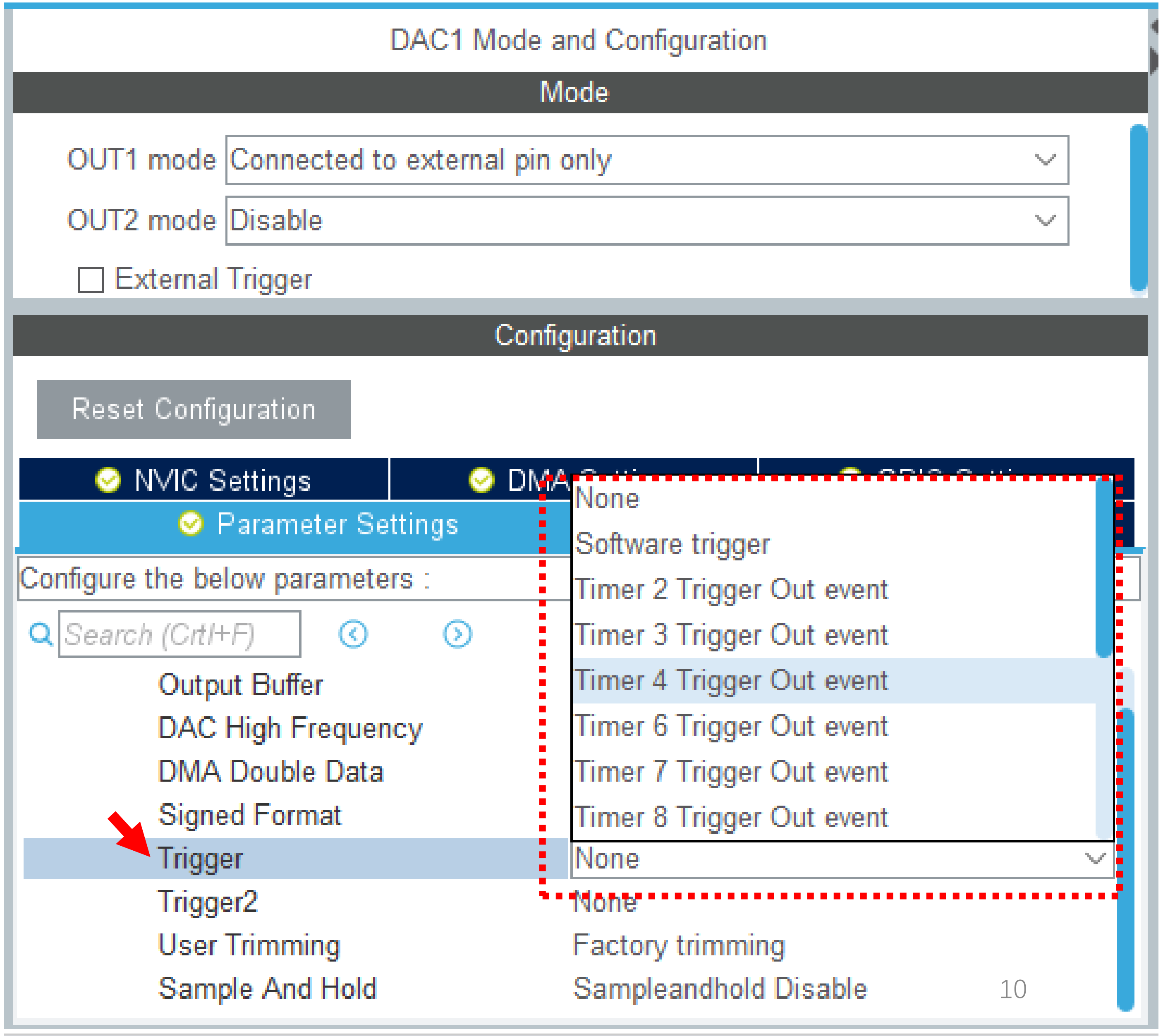
配置DAC(1)

- 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
- OUT2暂不使用，保持为Disable
- 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



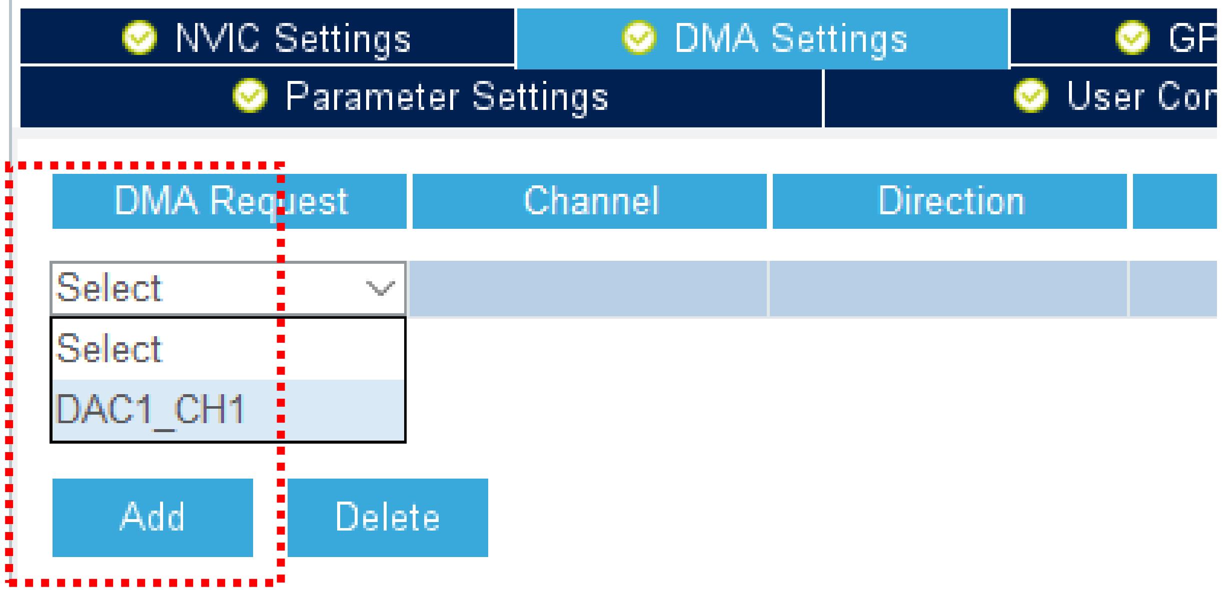
配置DAC(2)

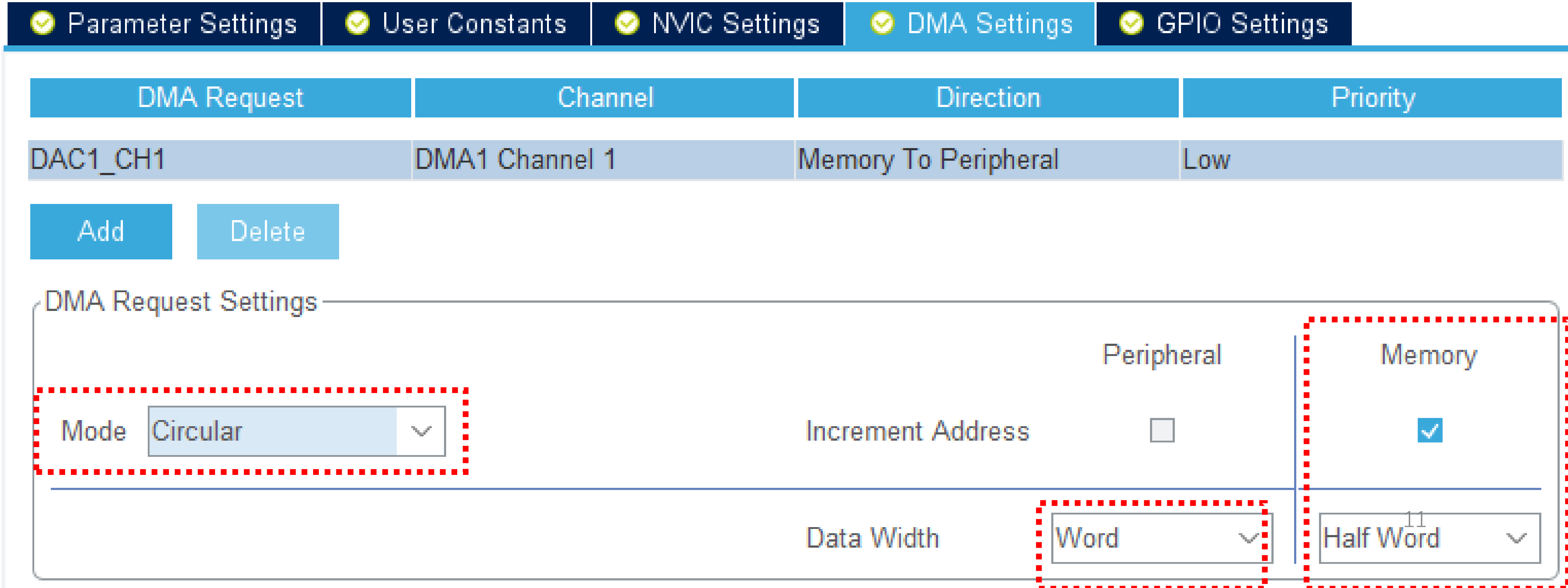
- 用定时器来触发DAC，所以将其中的Trigger，选择为Timer 4 Trigger Out event



配置DAC(3)

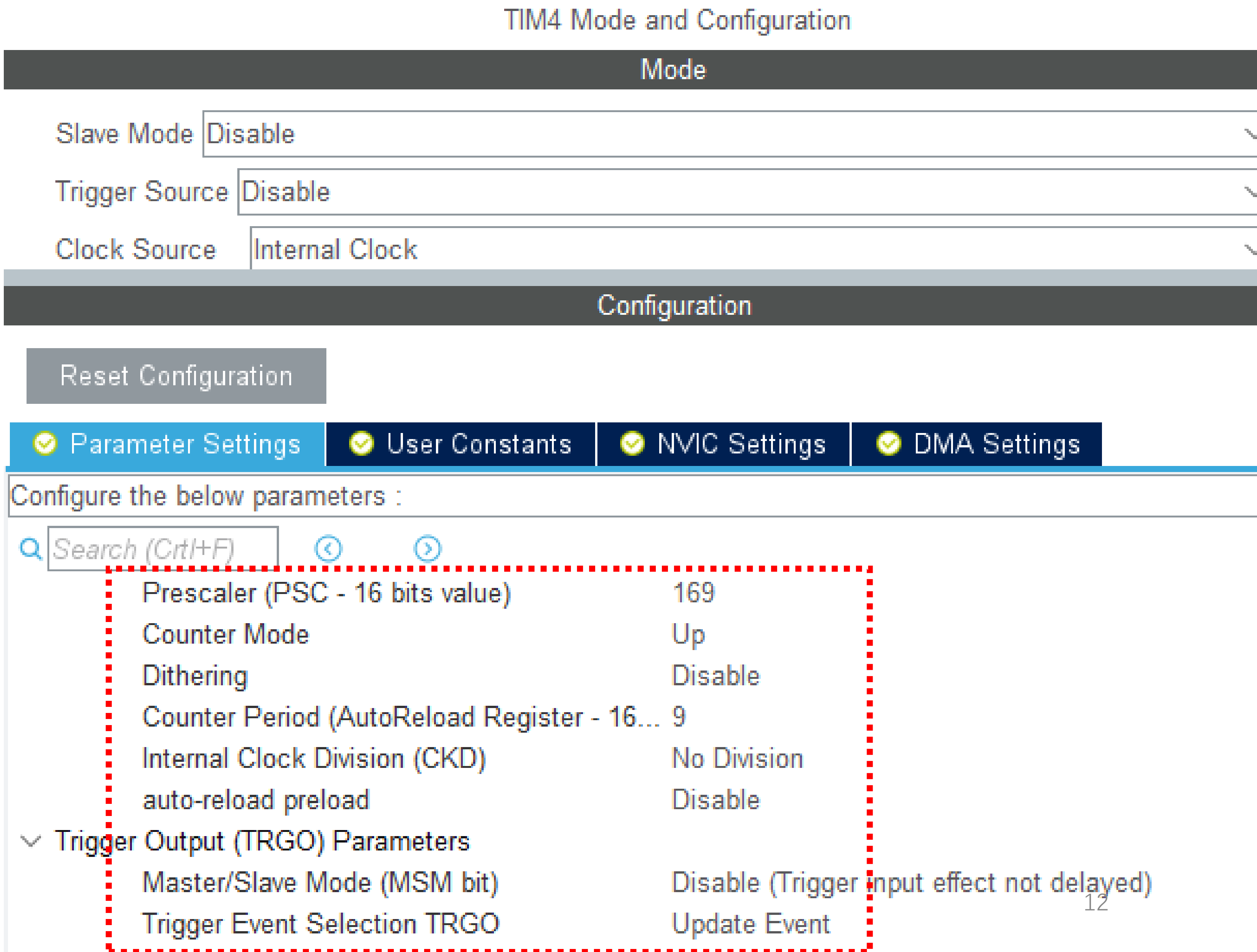
- 配置DAC的DMA





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
- Trigger Event Selection TRGO，选Update Event
- 不配置中断

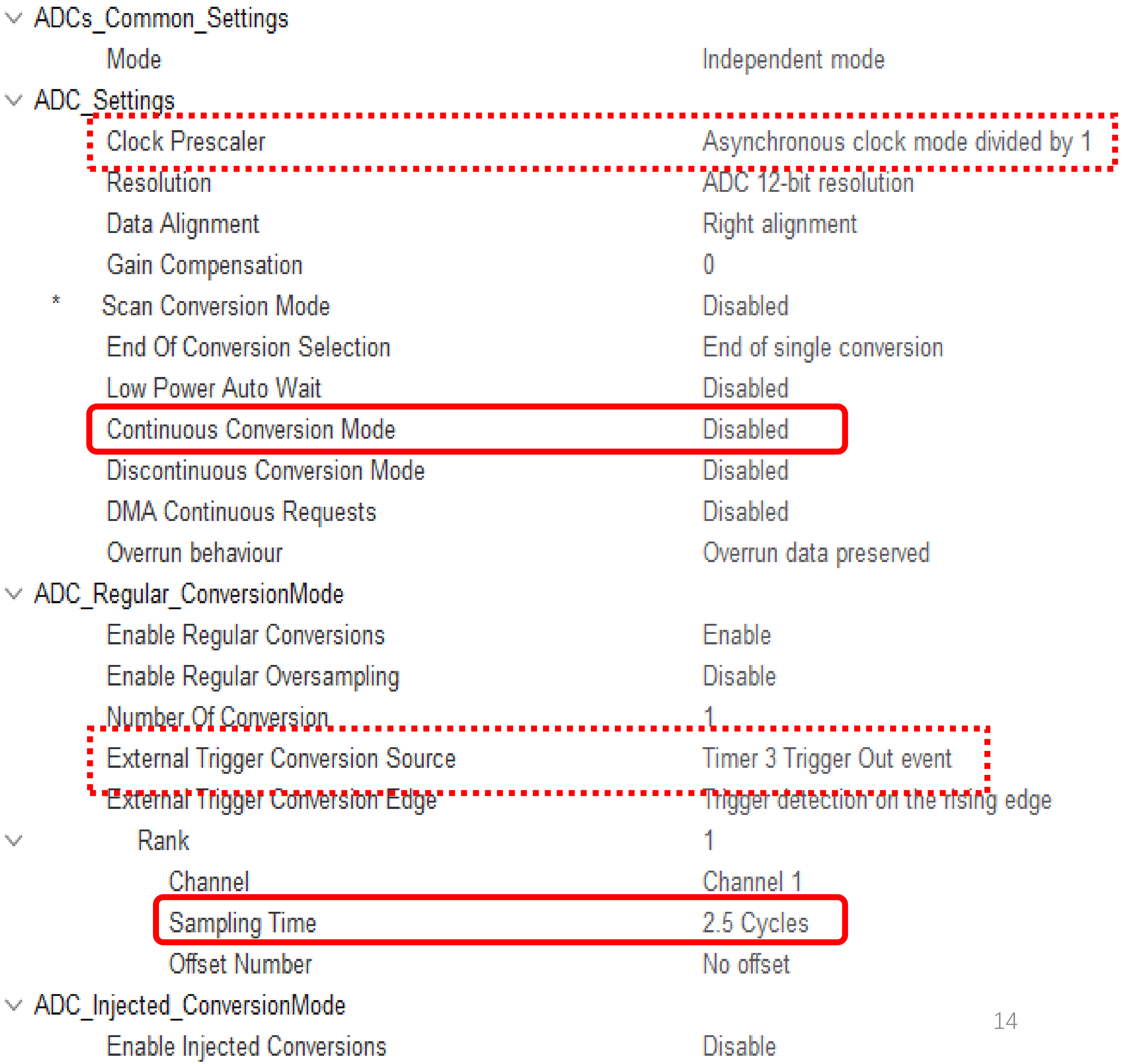


配置串口

- 选择“Connectivity”中的USART2，在其模式（Mode）栏选“异步”（Asynchronous），其他参数设置均保持默认（波特率115200），不开启中断
- 将USART2的两个引脚PA2和PA3均设置为上拉（Pull-up）

配置ADC

- 选择Analog中的ADC1
- 选择IN1为IN1 Single-ended
- 将ADC的Clock Prescaler选择为：Asynchronous clock mode divided by 1
- 将ADC_Settings参数栏中Continuous Conversion Mode选择为Disabled
- 在ADC_Regular_Conversion Mode栏，将External Trigger Conversion Source选择为：Timer 3 Trigger Out event
- 将位于Rank下的采样时间选择为2.5周期。这个参数决定着ADC的转换时间

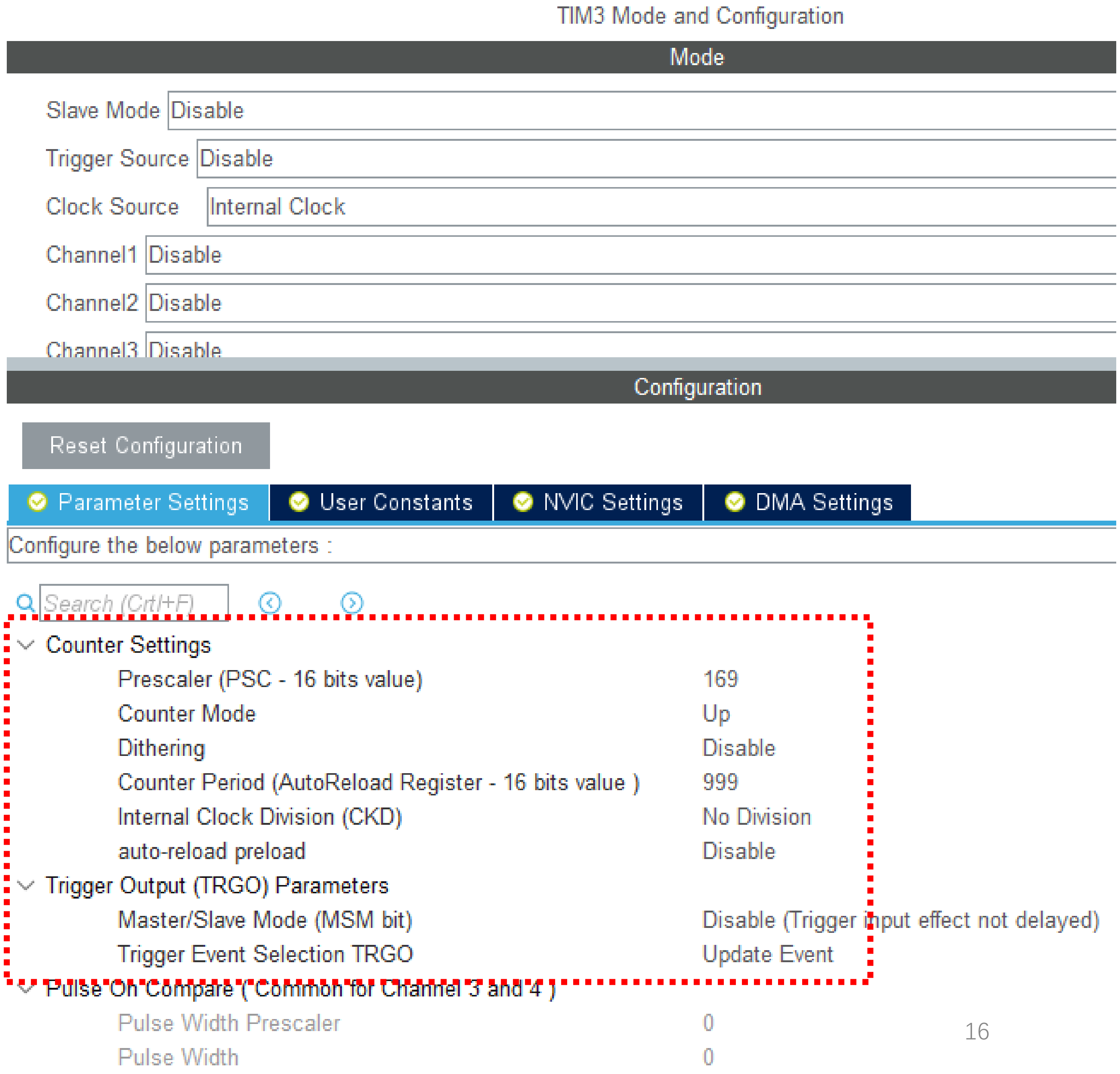


配置中断

- 打开ADC配置界面中的NVIC设置（NVIC Settings），使能ADC1的中断（ADC1 and AD2 global interrupt，ADC1与ADC2共用一个中断类型）
- 在“System Core”中的NVIC中，将ADC1中断的优先级设置为1。因只有一个中断，也可以保持为0

配置定时器TIM3

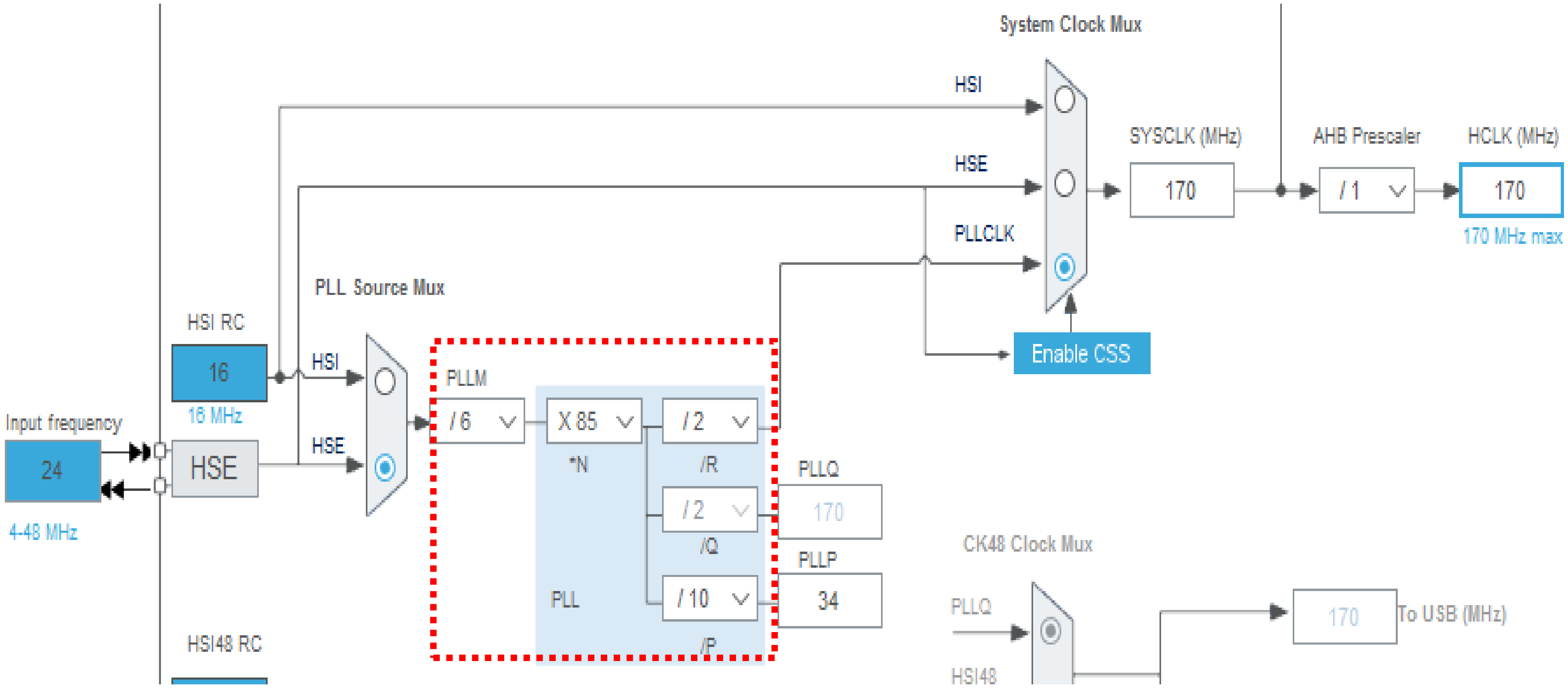
- 在TIM3的Mode栏，只需选择Clock Source为Internal Clock
- 在下面的配置栏，设置计数器的预分频因子为169
- 计数器周期设置为999
- 在Trigger Output参数栏，将Trigger Event Selection TRGO选为Update Event



时钟配置

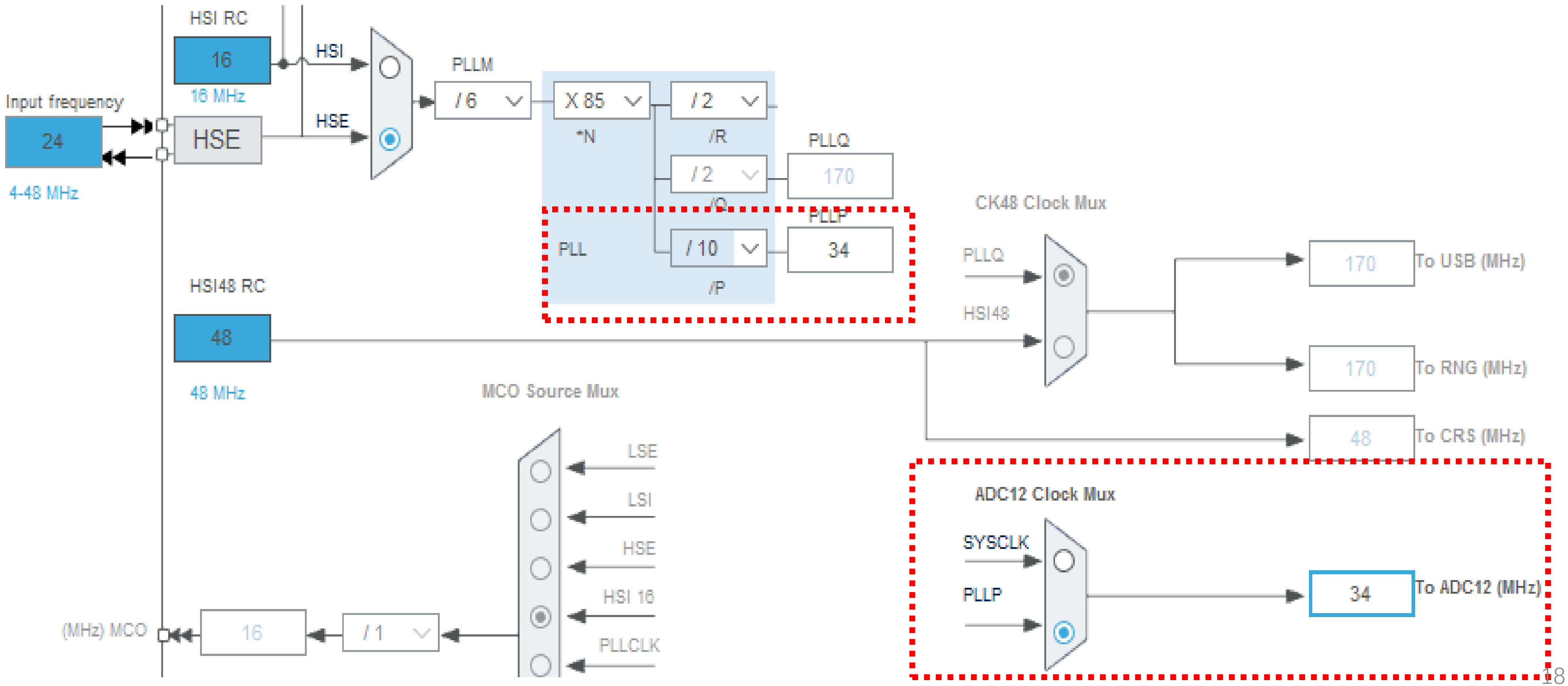
■配置系统时钟

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



ADC时钟设置

■ 配置ADC的时钟为34MHz



修改代码

- 保存硬件配置界面（*.ioc），启动代码生成
- 打开main.c

启动定时器ADC、DAC和DMA

■ TIM3、TIM4，DAC，ADC的初始化

```
/* USER CODE BEGIN 2 */
 HAL_TIM_Base_Start(&htim3);
 HAL_TIM_Base_Start(&htim4);
 HAL_DAC_Start_DMA(&hdac1, DAC_CHANNEL_1,(uint32_t *)
                   SineWaveData,DAC_BUFFER_SIZE,DAC_ALIGN_12B_R);
 HAL_ADCEx_Calibration_Start(&hadc1, ADC_SINGLE_ENDED);
 HAL_ADC_Start_IT(&hadc1);
 /* USER CODE END 2 */
```

重新定义ADC回调函数

■ 重写回调函数HAL_ADC_ConvCpltCallback()读取AD采样值，并通过串口发送

```
/* USER CODE BEGIN 4 */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc)
{
   ADC1ConvertedValue = HAL_ADC_GetValue(&hadc1);
   HAL_UART_Transmit(&huart2, (uint8_t *)&ADC1ConvertedValue, 2, 0xFFFF);
}
/* USER CODE END 4 */
```

变量定义

- 在主程序中定义AD采样数据变量和DAC波形数据变量（放到注释对中）：

```
/* USER CODE BEGIN PV */
uint16_t ADC1ConvertedValue = 0;
uint16_t SineWaveData[DAC_BUFFER_SIZE] = {
2047,2304,2557,2801,3034,3251,3449,3625,3776,3900,3994,4058,4090,4090,4058,
3994,3900,3776,3625,3449,3251,3034,2801,2557,2304,2048,1791,1538,1294,1061,
844,646,470,319,195,101,37,5,5,37,101,195,319,470,646,844,1061,1294,1538, 1791
 };
/* USER CODE END PV */
```

- DAC数组长度**DAC_BUFFER_SIZE**，定义到main.h中：

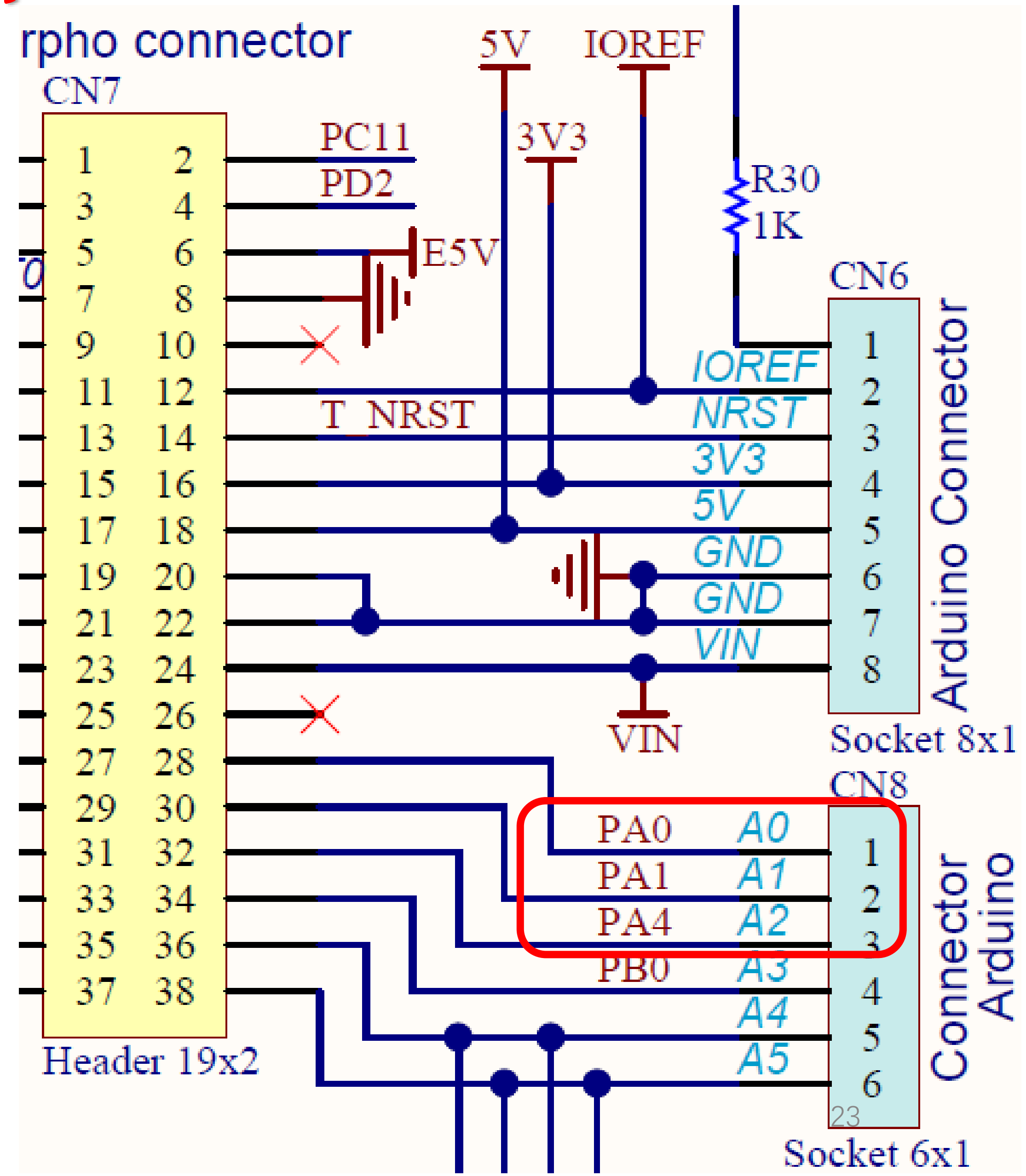
```
/* USER CODE BEGIN Private defines */
#define DAC_BUFFER_SIZE (uint16_t) 50
/* USER CODE END Private defines */
```

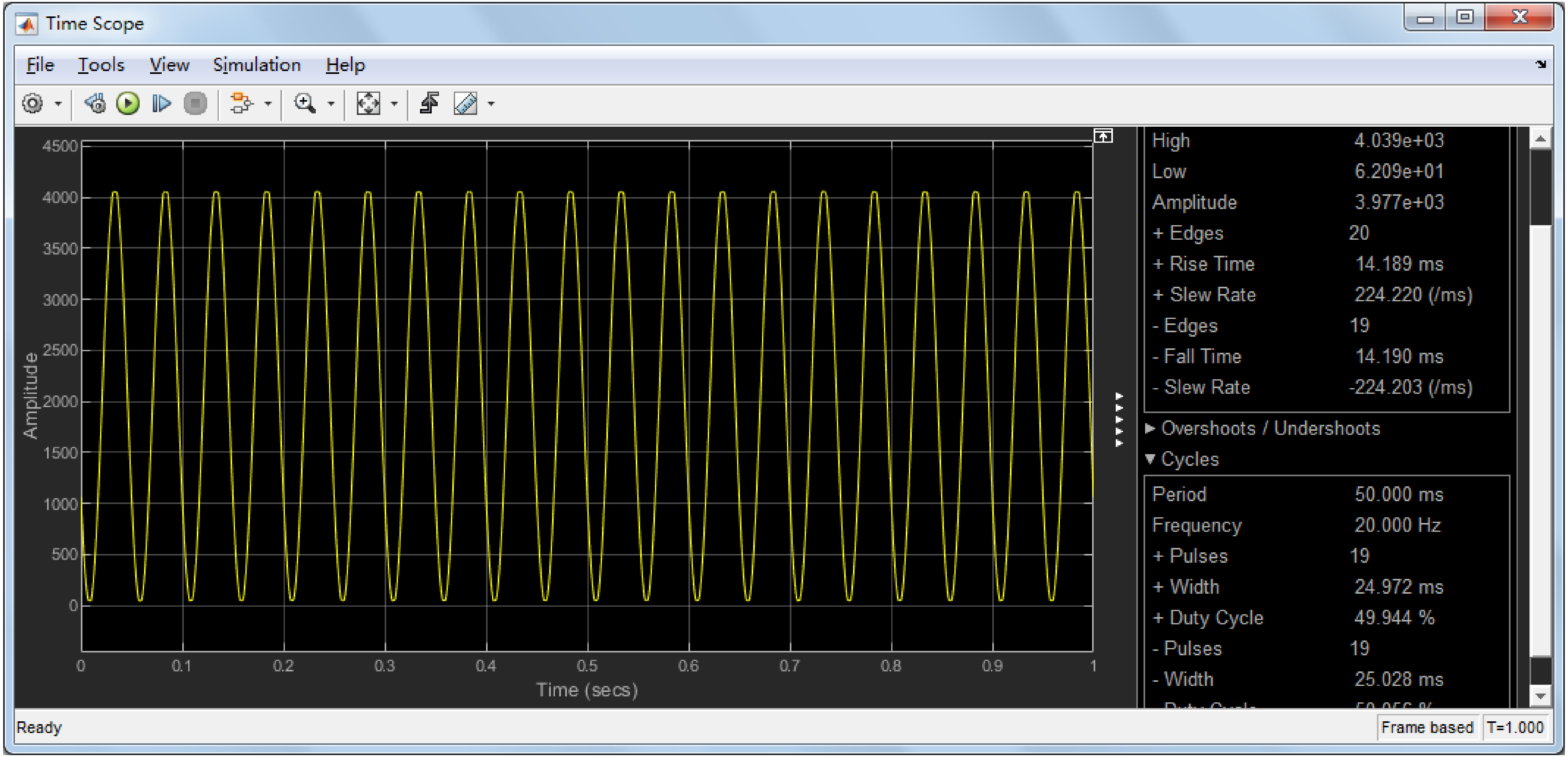
计算一下DAC输出信号频率和ADC采样频率

查看结果

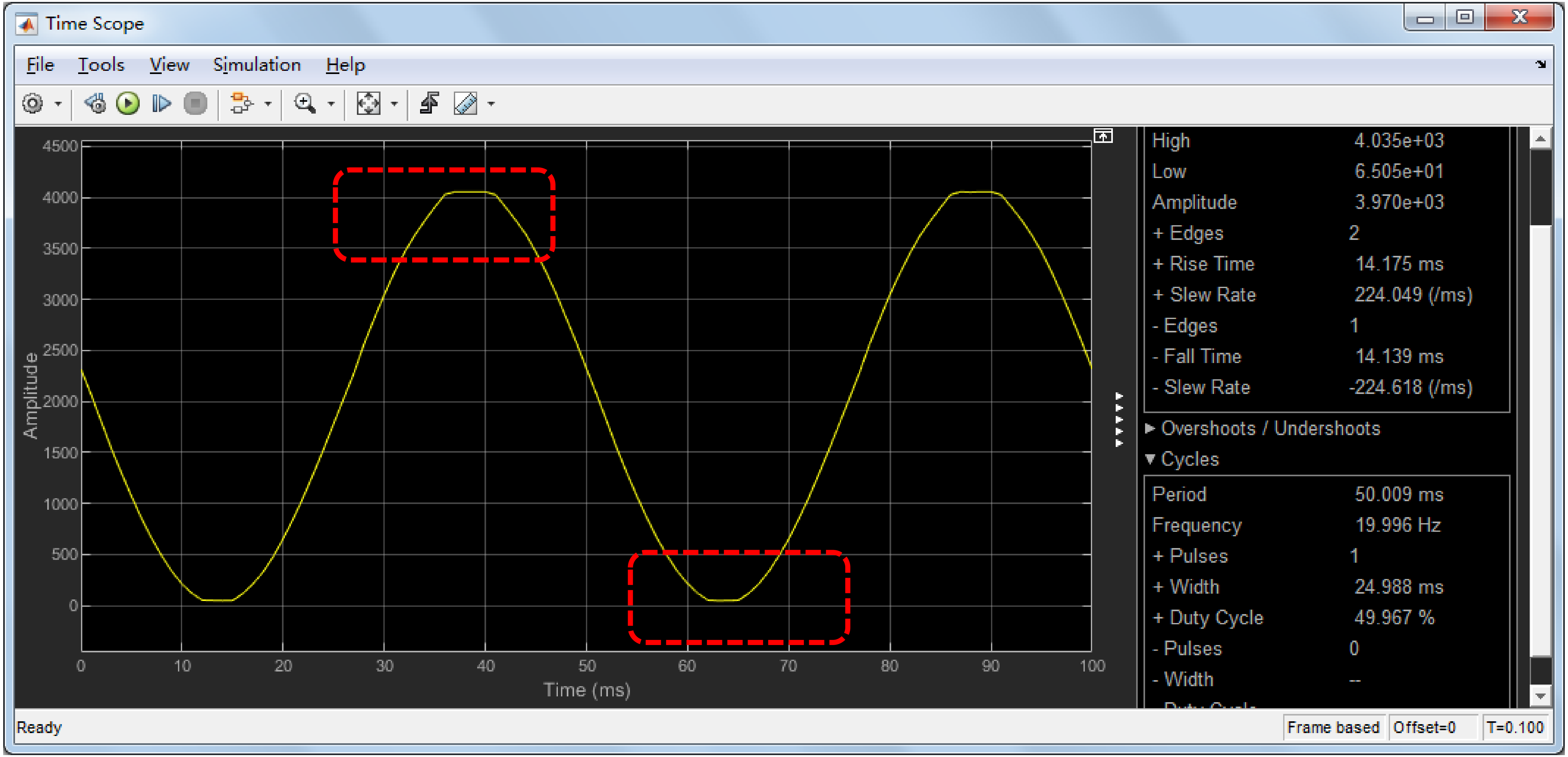
■ 编译工程，下载代码到硬件中，运行。将DAC输出PA4连接到ADC输入端PA0上

■ 打开simulink模型，查看结果

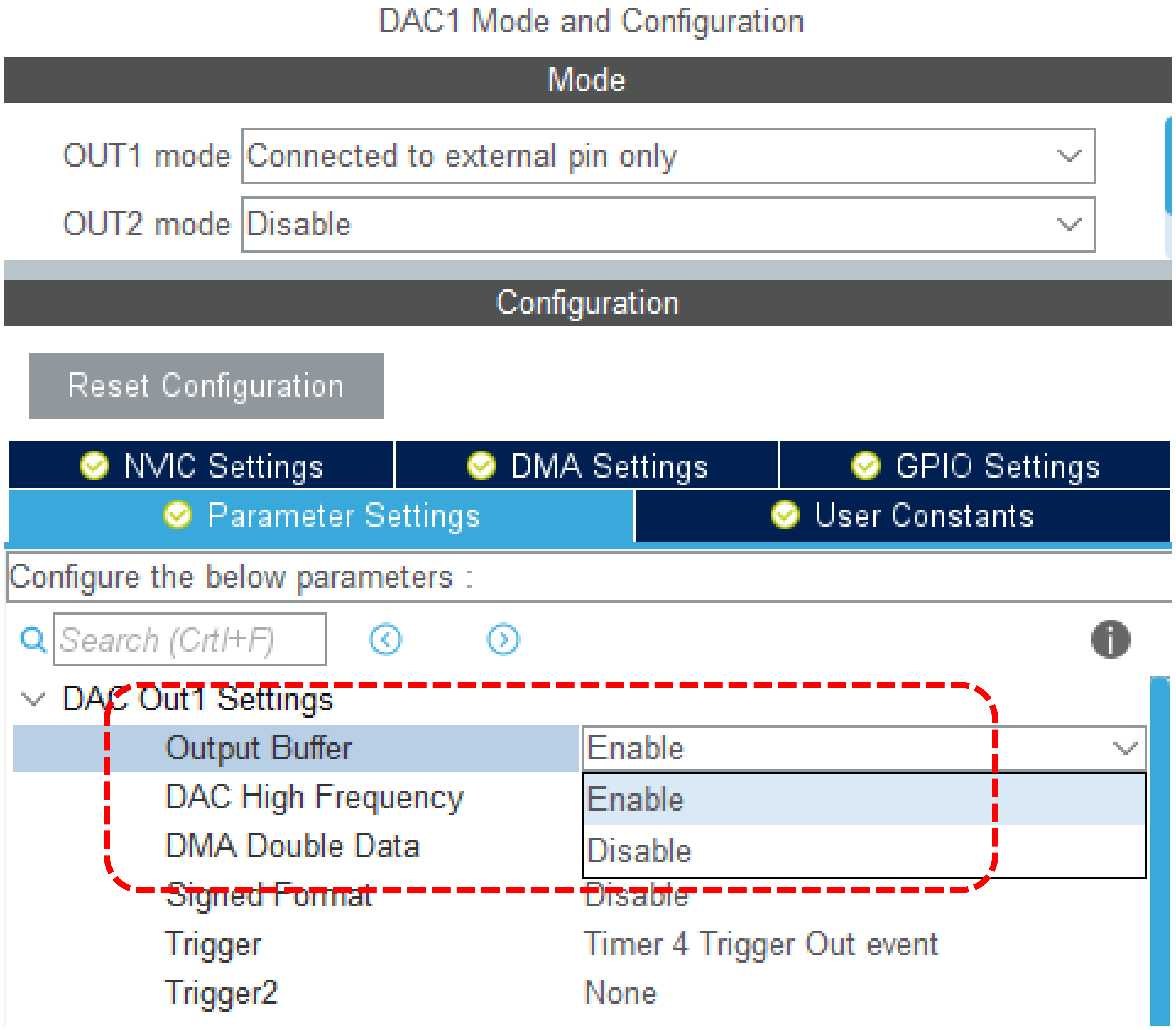


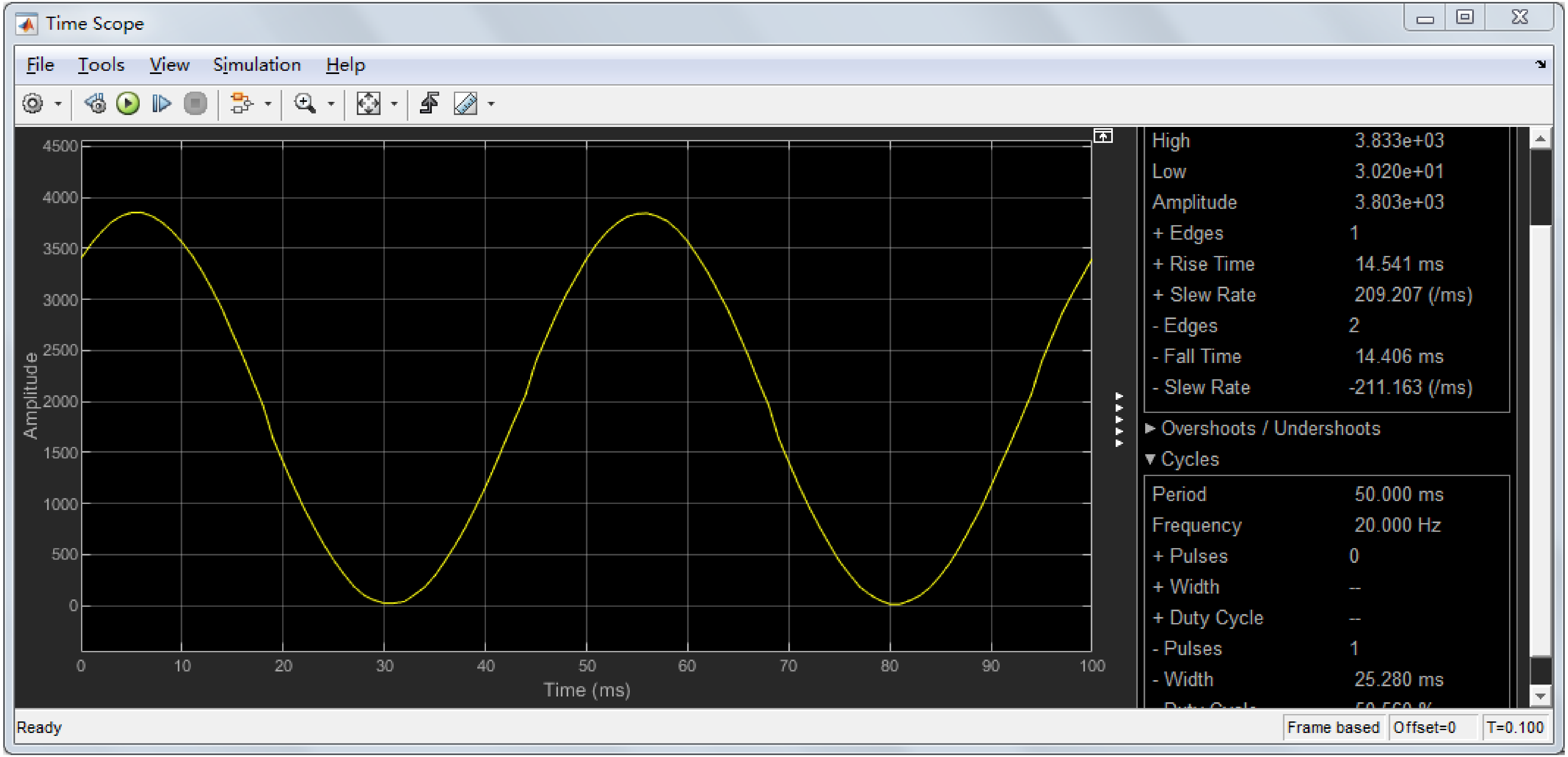


Time Scope
File Tools View Simulation Help
4500
4000
3500
3000
2500
2000
1500
1000
500
0
Amplitude
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Time (secs)
High 4.039e+03
Low 6.209e+01
Amplitude 3.977e+03
+ Edges 20
+ Rise Time 14.189 ms
+ Slew Rate 224.220 (/ms)
- Edges 19
- Fall Time 14.190 ms
- Slew Rate -224.203 (/ms)
Overshoots / Undershoots
Cycles
Period 50.000 ms
Frequency 20.000 Hz
+ Pulses 19
+ Width 24.972 ms
+ Duty Cycle 49.944 %
- Pulses 19
- Width 25.028 ms
Ready
Frame based
T=1.000



Time Scope
File Tools View Simulation Help
Amplitude
Time (ms)
High 4.035e+03
Low 6.505e+01
Amplitude 3.970e+03
+ Edges 2
+ Rise Time 14.175 ms
+ Slew Rate 224.049 (/ms)
- Edges 1
- Fall Time 14.139 ms
- Slew Rate -224.618 (/ms)
Overshoots / Undershoots
Cycles
Period 50.009 ms
Frequency 19.996 Hz
+ Pulses 1
+ Width 24.988 ms
+ Duty Cycle 49.967 %
- Pulses 0
- Width --
Ready
Frame based
Offset=0
T=0.100


DAC1 Mode and Configuration
Mode
OUT1 mode
Connected to external pin only
OUT2 mode
Disable
Configuration
Reset Configuration
NVIC Settings
DMA Settings
GPIO Settings
Parameter Settings
User Constants
Configure the below parameters :
Search (Crtl+F)
DAC Out1 Settings
Output Buffer
Enable
DAC High Frequency
Enable
DMA Double Data
Disable
Signed Format
Disable
Trigger
Timer 4 Trigger Out event
Trigger2
None



Time Scope
File Tools View Simulation Help
Amplitude
Time (ms)
High 3.833e+03
Low 3.020e+01
Amplitude 3.803e+03
+ Edges 1
+ Rise Time 14.541 ms
+ Slew Rate 209.207 (/ms)
- Edges 2
- Fall Time 14.406 ms
- Slew Rate -211.163 (/ms)
Overshoots / Undershoots
Cycles
Period 50.000 ms
Frequency 20.000 Hz
+ Pulses 0
+ Width --
+ Duty Cycle --
- Pulses 1
- Width 25.280 ms
Ready
Frame based
Offset=0
T=0.100

练习9：用DMA实现DAC输出

任务9.1、实现上述ADC（中断）+DAC（DMA）方式的数据测量系统。利用simulink模型，查看结果。

用DMA实现ADC采样

用DMA实现ADC采样

- 上面的例子中，用中断方式实现了AD采样；实际中，还可以用DMA方式实现AD采样
- 下面以重新建立新工程为例，

介绍构建基于ADC（DMA）+DAC（DMA）方式的数据测量系统的实现过程。也可以在前面练习的基础上进行修改。

建立新工程

时钟源与Debug模式配置

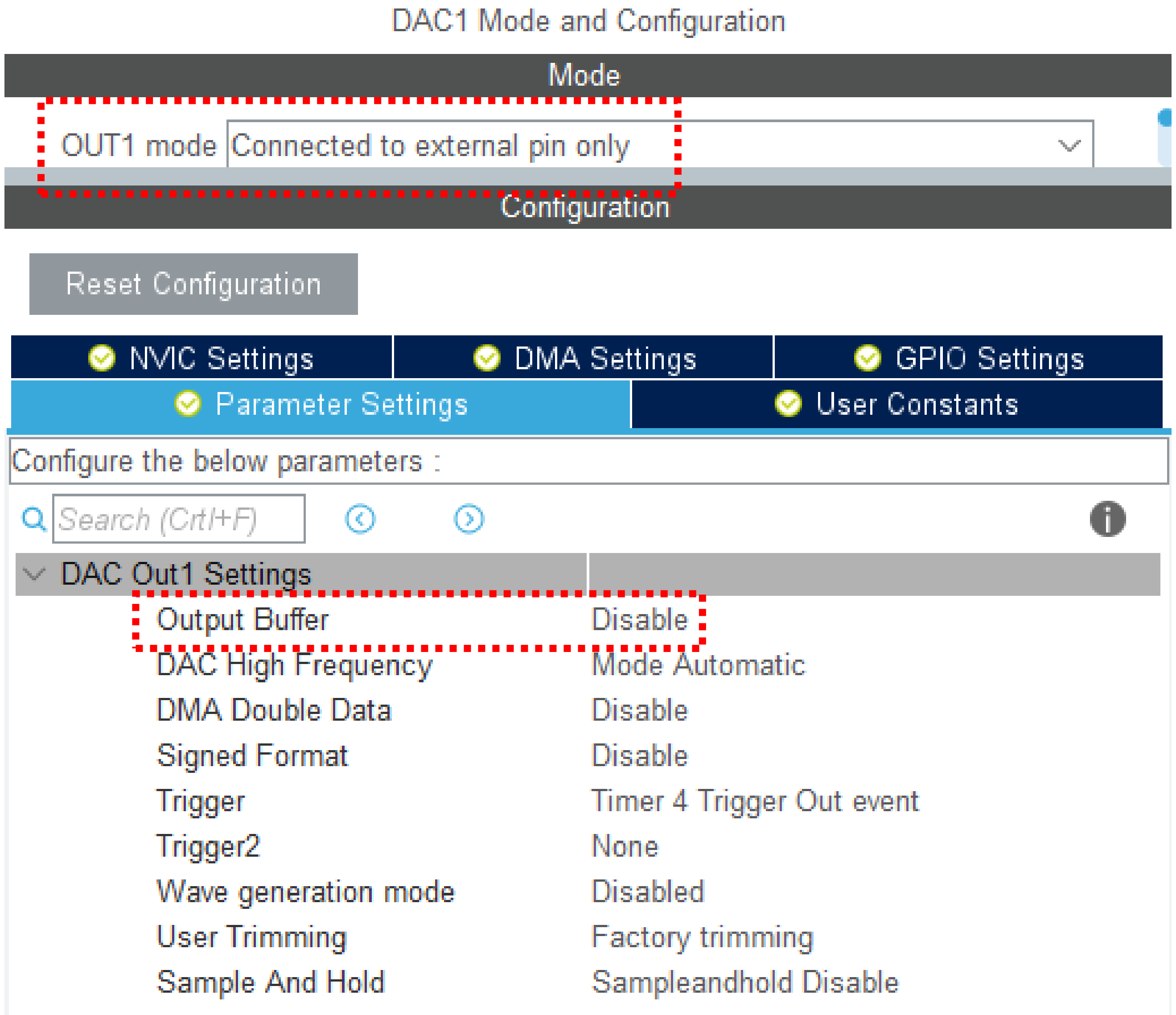
■ 选择时钟源和Debug模式

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

✓ SYS->Debug选择为Serial Wire

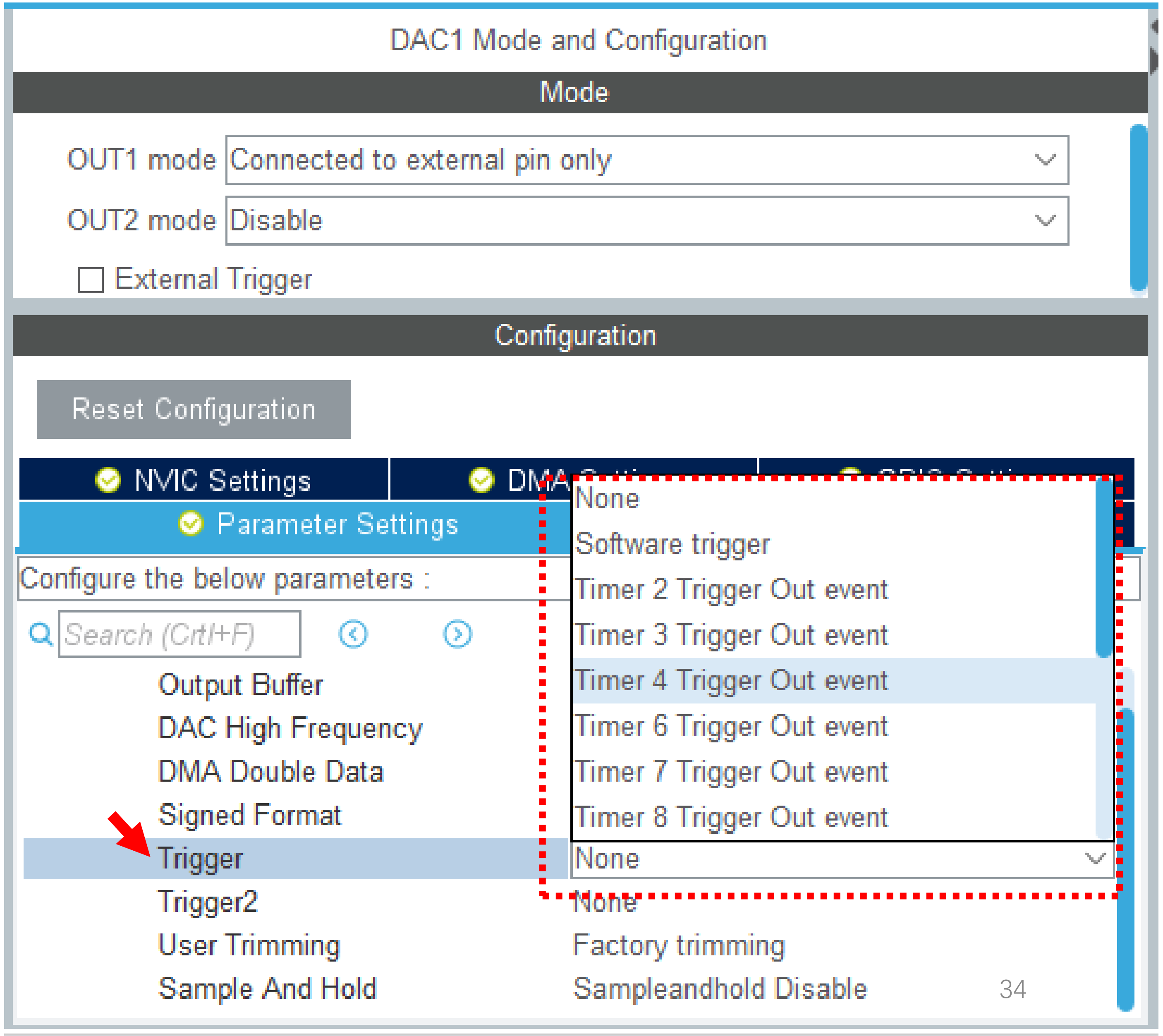
配置DAC(1)

- 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
- OUT2暂不使用，保持为Disable
- 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，**选为Disable**



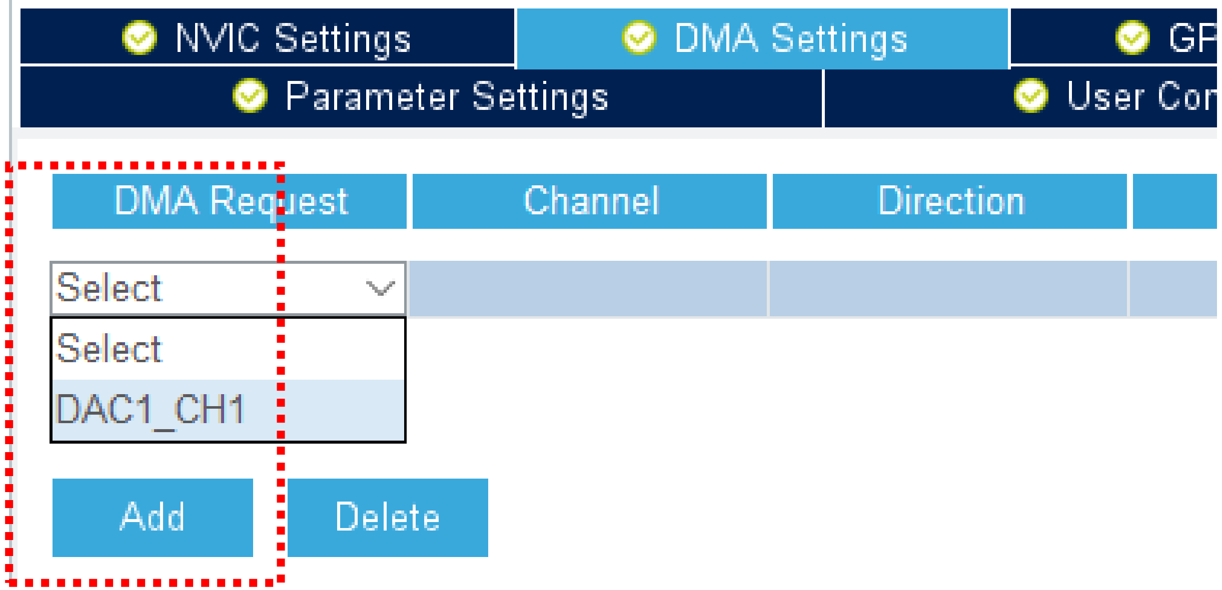
配置DAC(2)

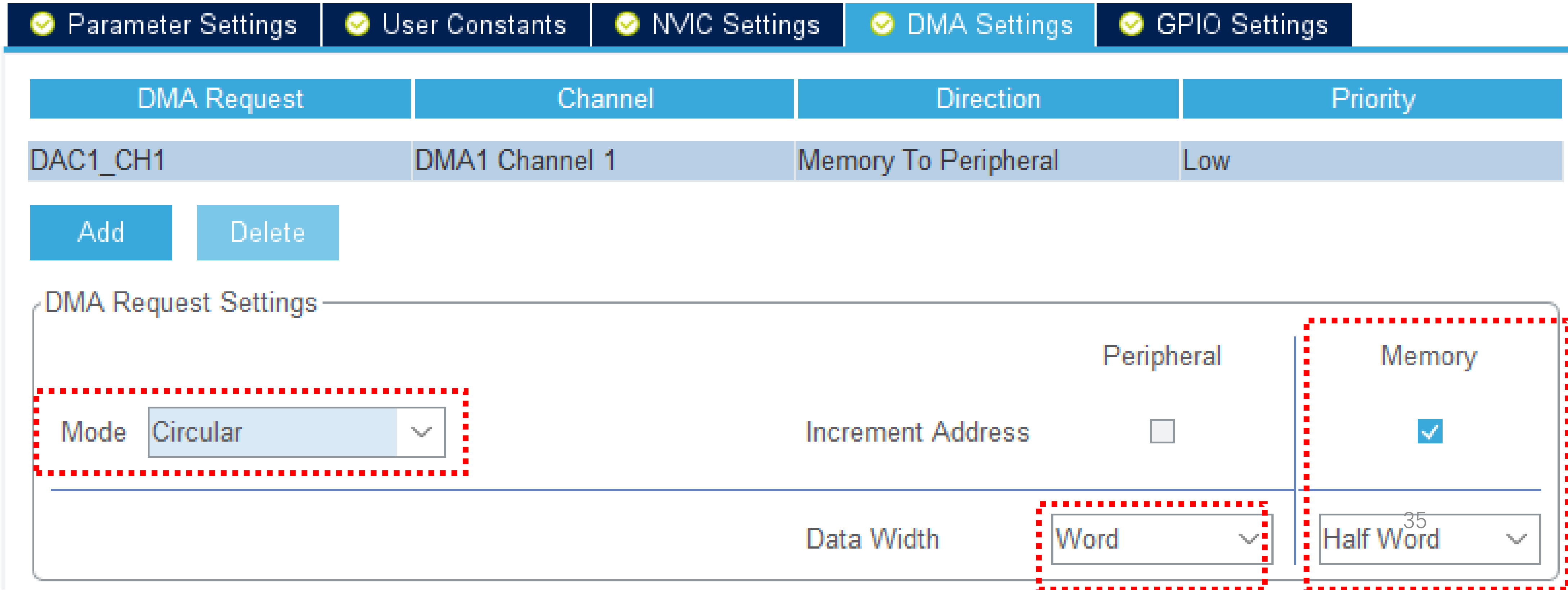
■ 用定时器来触发DAC，所以将其中的Trigger，选择为Timer 4 Trigger Out event



配置DAC(3)

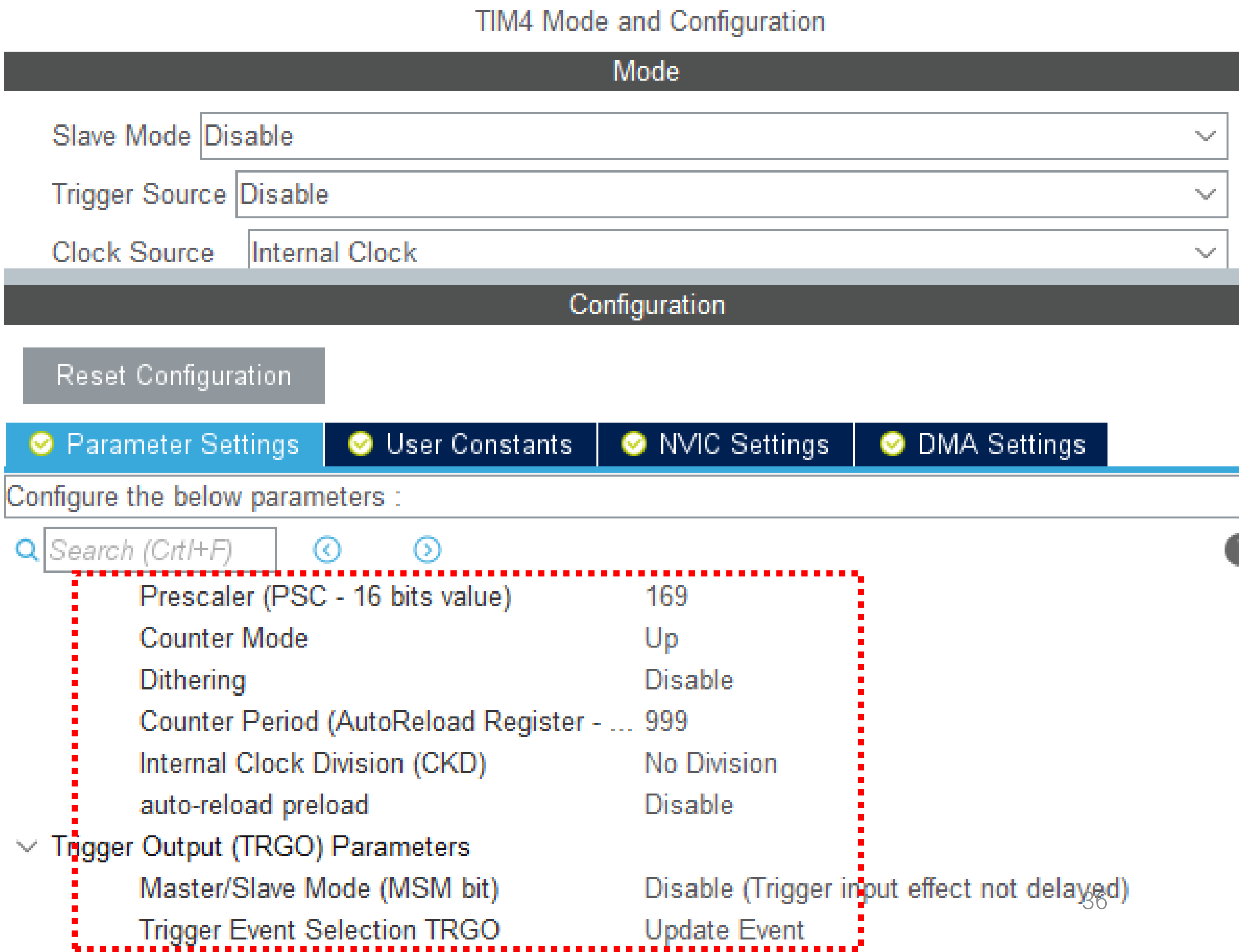
- 配置DAC的DMA





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999
- Trigger Event Selection TRGO，选Update Event
- 不用配置中断



配置串口

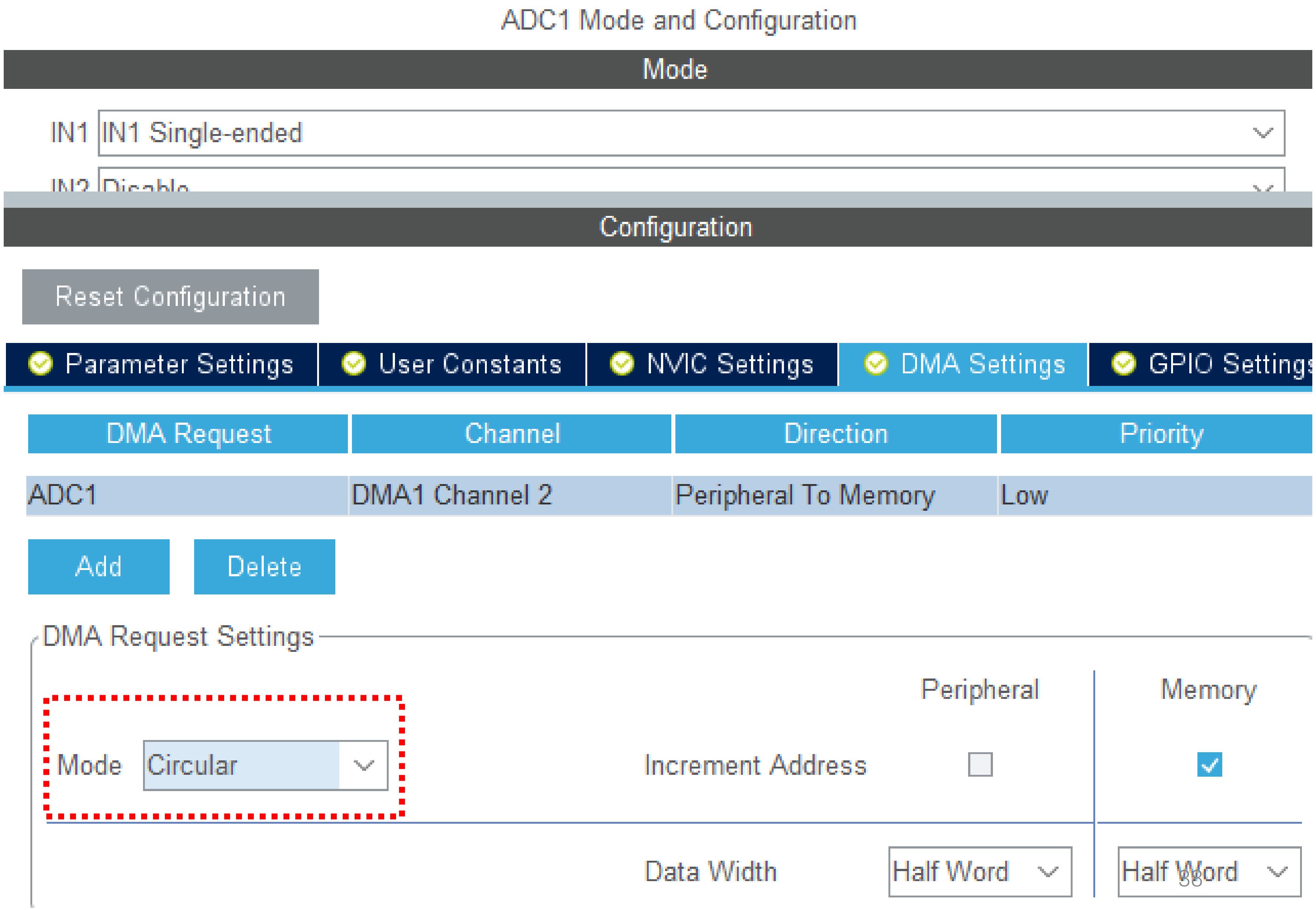
- 选择“Connectivity”中的USART2，在其模式（Mode）栏选“异步”（Asynchronous），其他参数设置均保持默认（波特率115200），不开启中断
- 将USART2的两个引脚PA2和PA3均设置为上拉（Pull-up）

配置ADC(1)

- 选择Analog中的ADC1，选择IN1为IN1 Single-ended
- 打开DMA Settings栏，添加ADC1的DMA通道

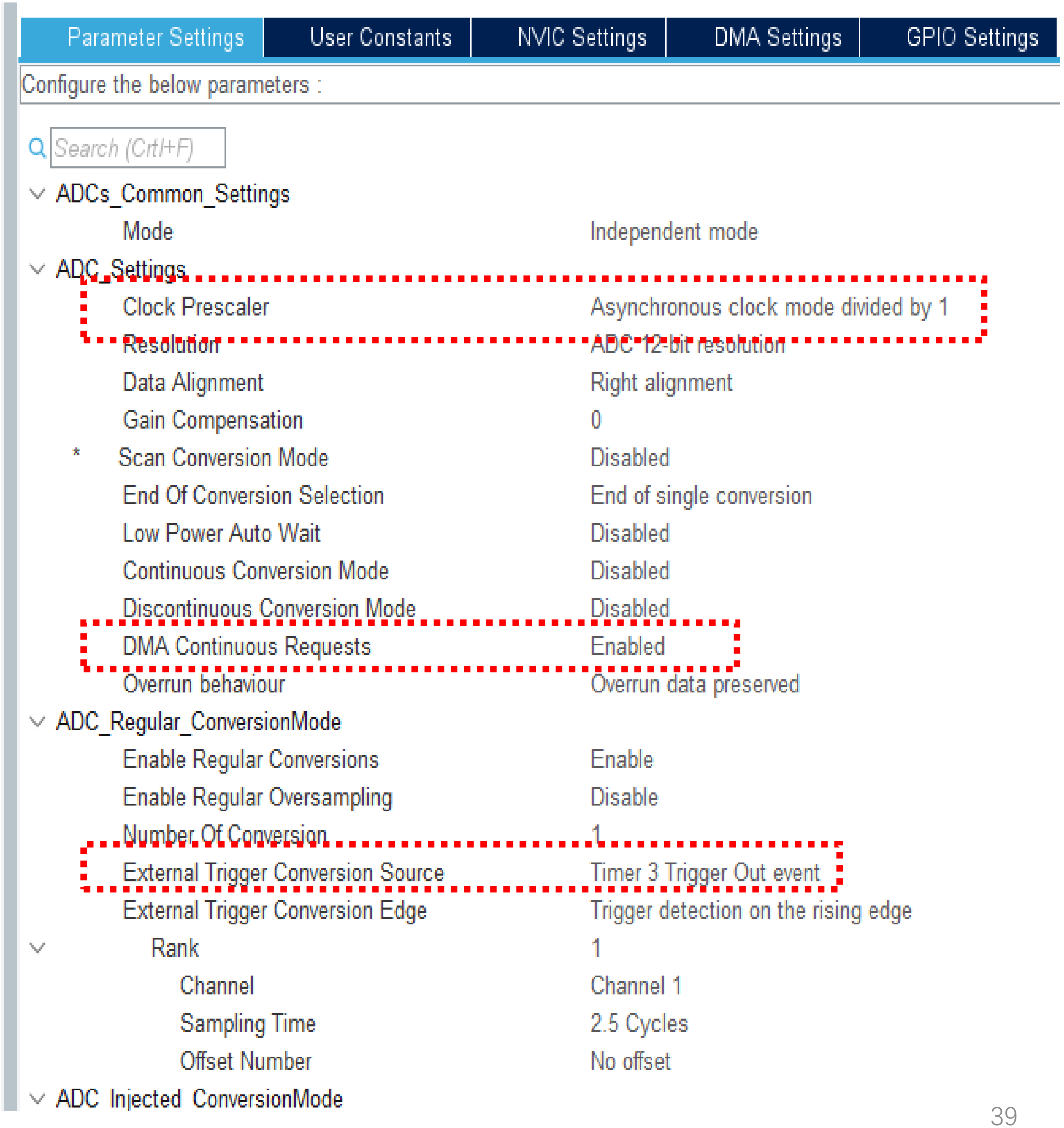
✓ Mode选Circular

✓ 其他参数用默认值



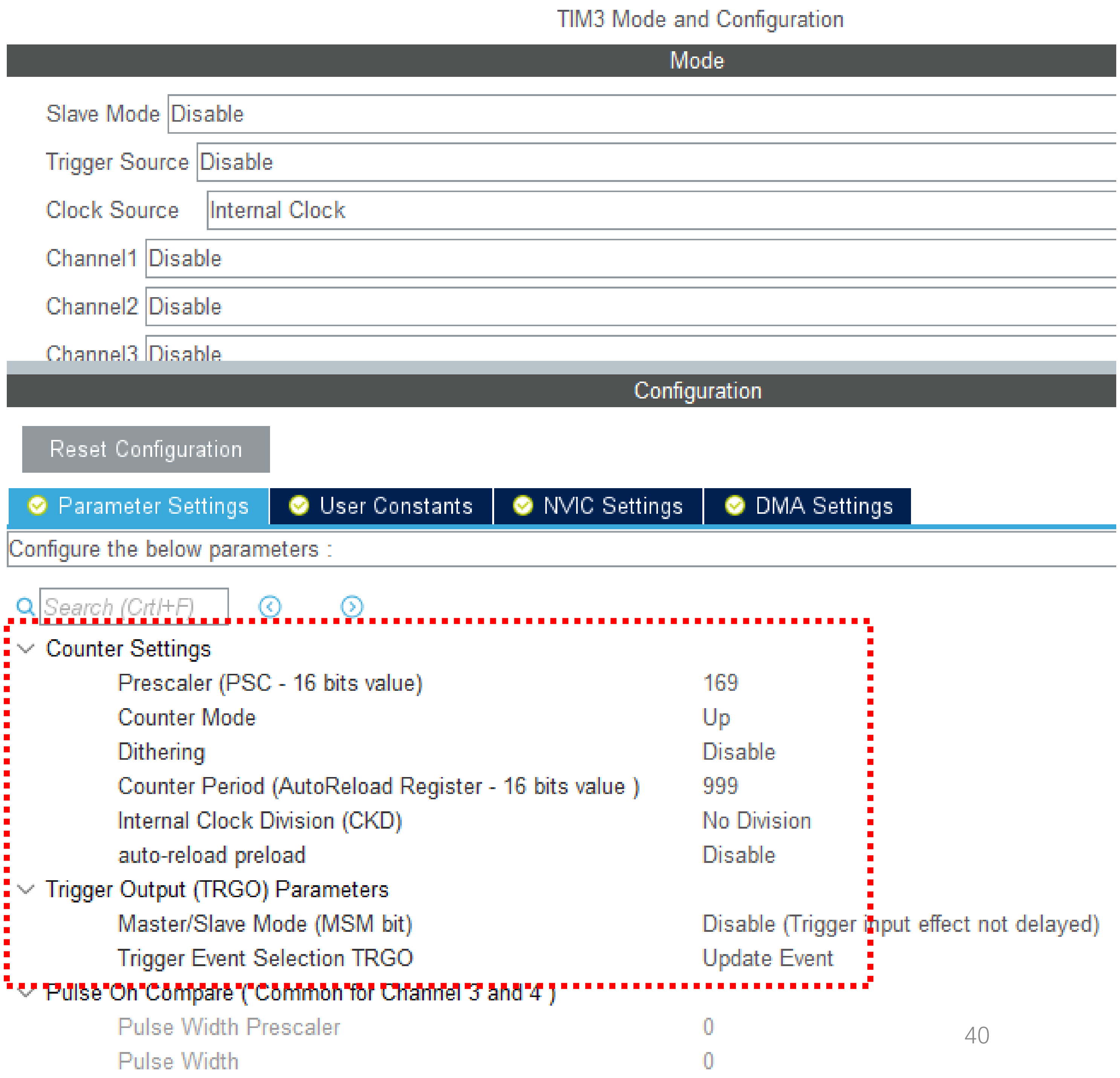
配置ADC(2)

- 将ADC的Clock Prescaler选择为：Asynchronous clock mode divided by 1
- 将ADC_Settings参数栏中Continuous Conversion Mode选择为Disabled
- **DAM Continuous Requests参数选Enable**
- 在ADC_Regular_Conversion Mode栏，将External Trigger Conversion Source选择为：Timer 3 Trigger Out event
- 将位于Rank下的采样时间选择为2.5周期。这个参数决定着ADC的转换时间
- **不设置ADC中断**



配置定时器TIM3

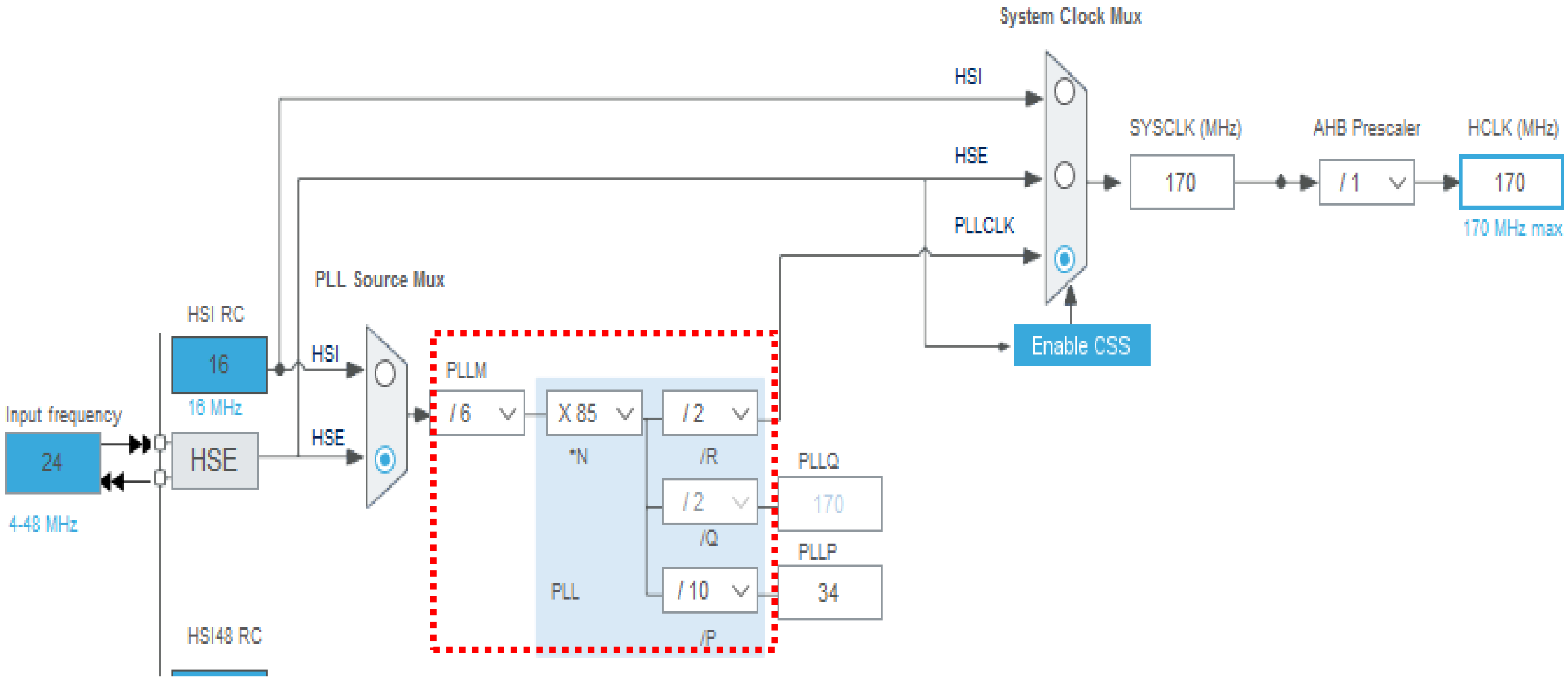
- 在TIM3的Mode栏，只需选择Clock Source为Internal Clock
- 在下面的配置栏，设置计数器的预分频因子为169
- 计数器周期设置为999
- 在Trigger Output参数栏，将Trigger Event Selection TRGO选为Update Event



时钟配置

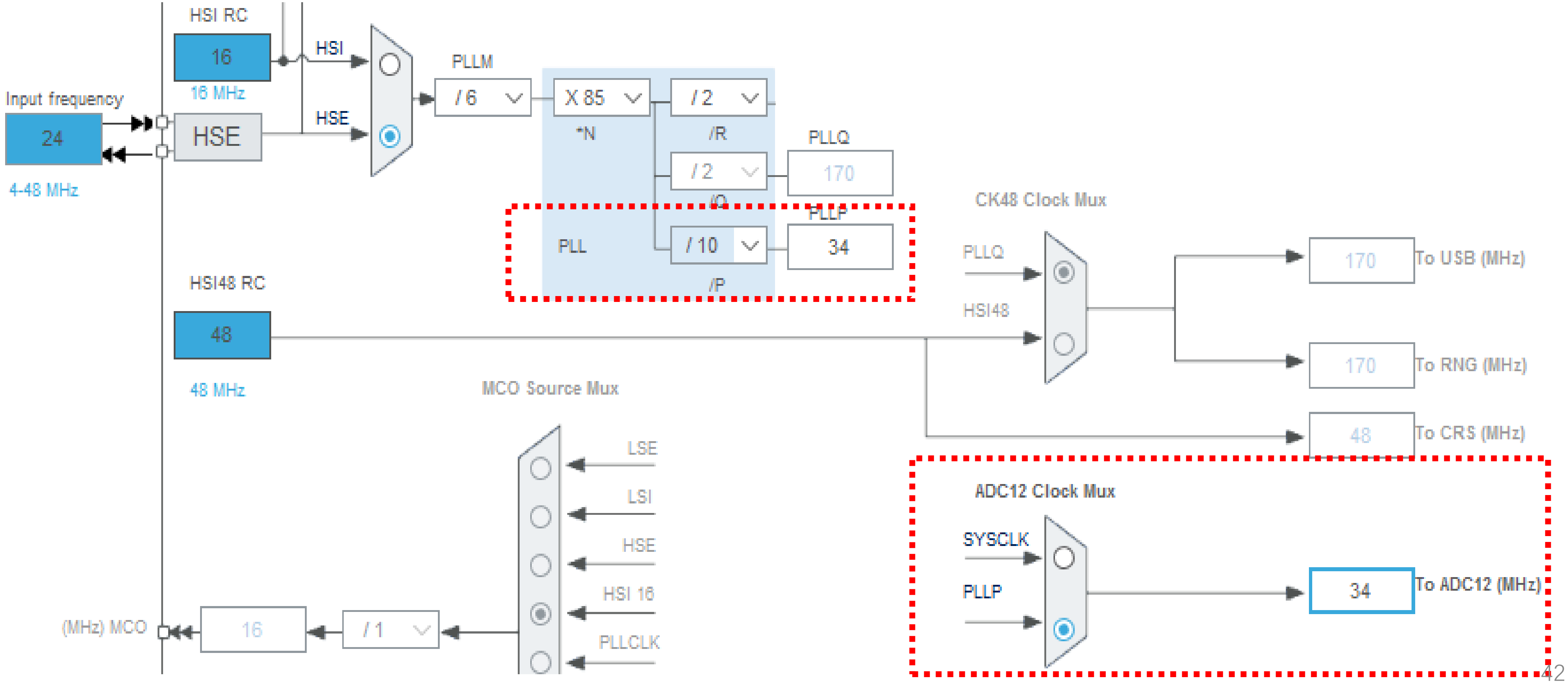
■配置系统时钟

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



ADC时钟设置

■ 配置ADC的时钟为34MHz



修改代码

- 保存硬件配置界面（*.ioc），启动代码生成
- 打开main.c

启动定时器ADC、DAC和DMA

- TIM3、TIM4，DAC，ADC的初始化

```
/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start(&htim3);
HAL_TIM_Base_Start(&htim4);
HAL_DAC_Start_DMA(&hdac1, DAC_CHANNEL_1,(uint32_t *)SineWaveData,DAC_BUFFER_SIZE,DAC_ALIGN_12B_R);
HAL_ADCEx_Calibration_Start(&hadc1, ADC_SINGLE_ENDED);
HAL_ADC_Start_DMA(&hadc1,(uint32_t *)&ADC1ConvertedData,ADC_CONVERTED_DATA_BUFFER_SIZE);
 /* USER CODE END 2 */
```

重新定义ADC回调函数

■ 由于DMA传递完规定数目的ADC采样值后，会触发回调函数HAL_ADC_ConvCpltCallback()的执行，所以可以在该回调函数中，将ADC的采样值通过串口发送出来。

```
/* USER CODE BEGIN 4 */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* AdcHandle)
{
        HAL_UART_Transmit(&huart2, (uint8_t *)&FrameHeader,2, 0xFFFF);
        HAL_UART_Transmit(&huart2, (uint8_t *)&ADC1ConvertedData,
                ADC_CONVERTED_DATA_BUFFER_SIZE*2, 0xFFFF);
        HAL_UART_Transmit(&huart2, (uint8_t *)&FrameTerm,2, 0xFFFF);
}
/* USER CODE END 4 */
```

变量定义

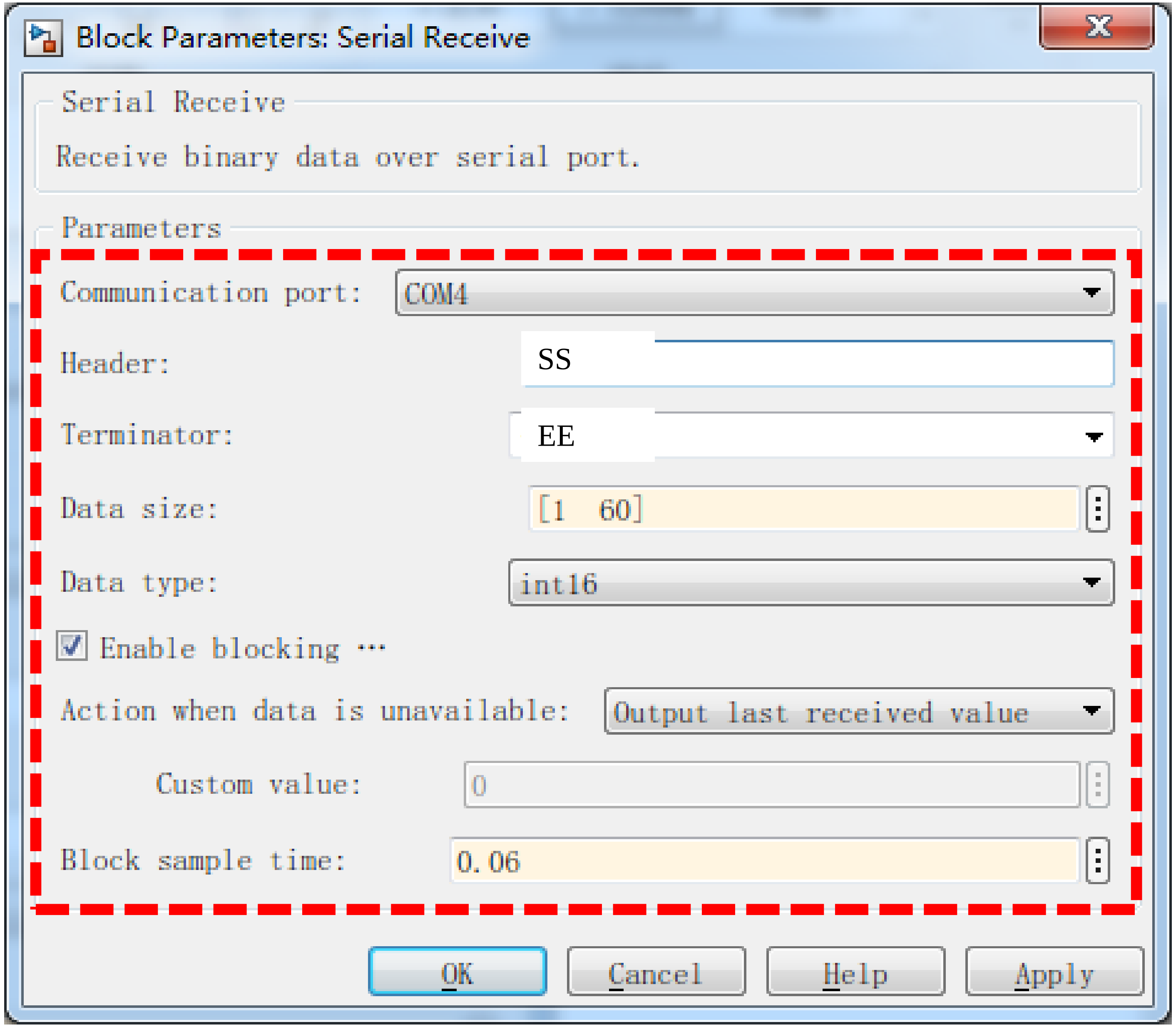
- 在主程序中定义AD采样数据变量和DAC波形数据变量（放到注释对中）：

```
/* USER CODE BEGIN PV */
uint16_t SineWaveData[DAC_BUFFER_SIZE] = {
2047,2304,2557,2801,3034,3251,3449,3625,3776,3900,3994,4058,4090,4090,4058,3994,3900,3776,3625,3449,3251,3034,2801,2557,2304,2048,1791,1538,1294,1061,844,646,470,319,195,101,37,5,5,37,101,195,319,470,646,844,1061,1294,1538,1791
 };
uint16_t ADC1ConvertedData[ADC_CONVERTED_DATA_BUFFER_SIZE];
uint16_t FrameHeader = 0x5353;
uint16_t FrameTerm = 0x4545;
/* USER CODE END PV */
```

- DAC和ADC数据数据长度，定义到main.h中：

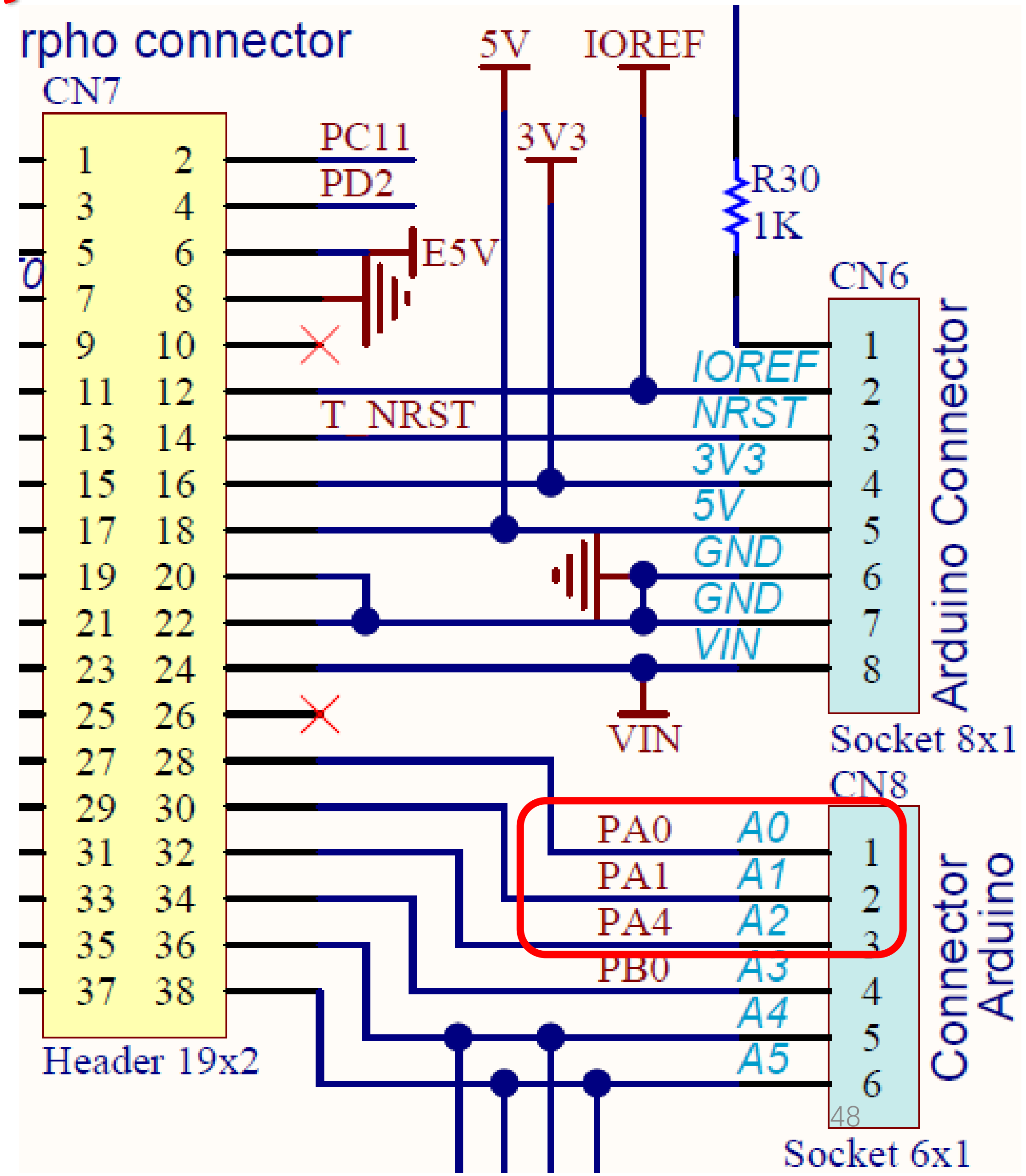
```
/* USER CODE BEGIN Private defines */
#define ADC_CONVERTED_DATA_BUFFER_SIZE (uint16_t) 60
#define DAC_BUFFER_SIZE (uint16_t) 50
/* USER CODE END Private defines */
```

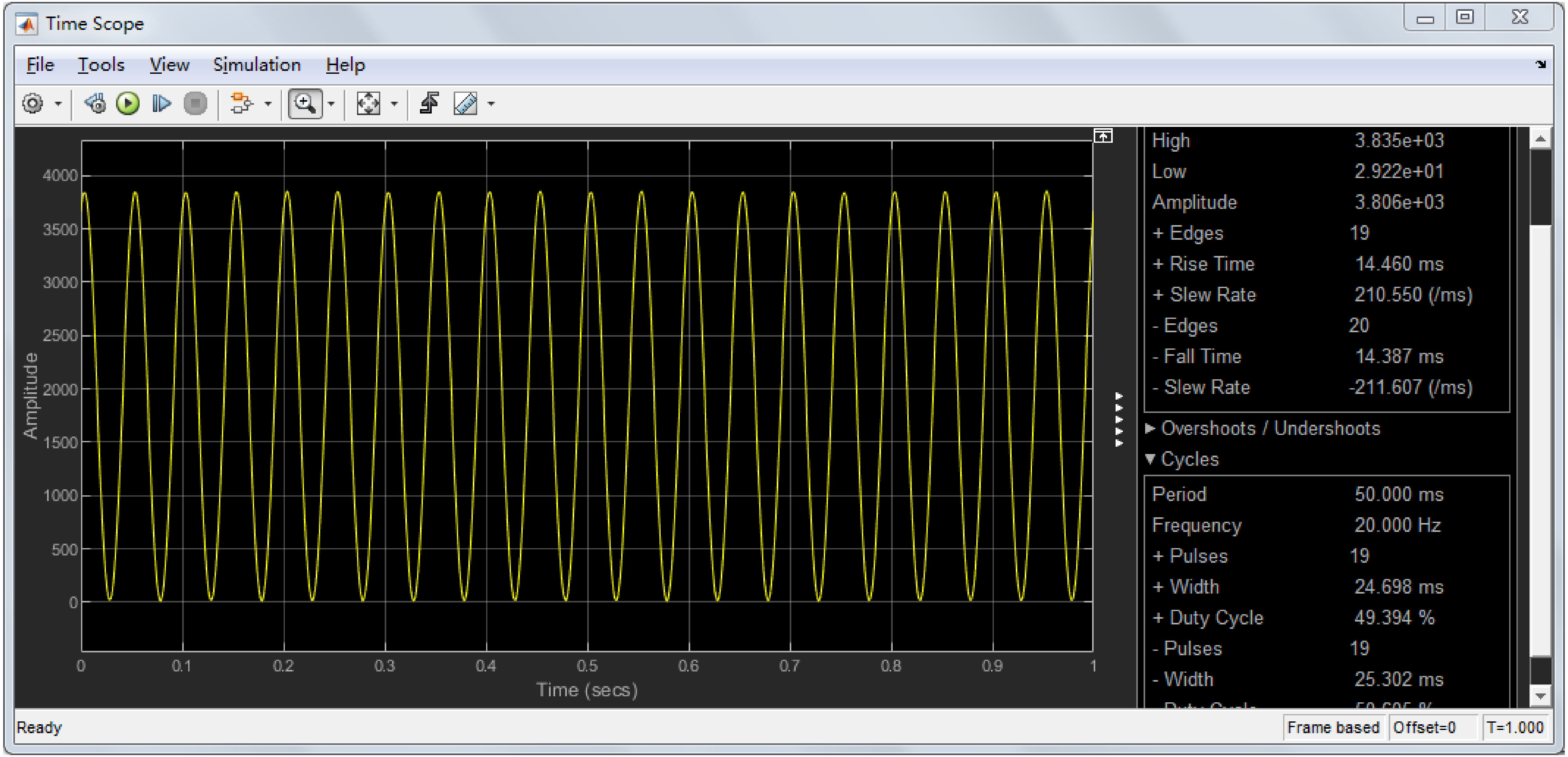
修改Serial Receive模块



查看结果

- 编译工程，下载代码到硬件中，运行。将DAC输出PA4连接到ADC输入端PA0上
- 打开simulink模型，查看结果




Time Scope
File Tools View Simulation Help
4000
3500
3000
2500
2000
1500
1000
500
0
Amplitude
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Time (secs)
High 3.835e+03
Low 2.922e+01
Amplitude 3.806e+03
+ Edges 19
+ Rise Time 14.460 ms
+ Slew Rate 210.550 (/ms)
- Edges 20
- Fall Time 14.387 ms
- Slew Rate -211.607 (/ms)
Overshoots / Undershoots
Cycles
Period 50.000 ms
Frequency 20.000 Hz
+ Pulses 19
+ Width 24.698 ms
+ Duty Cycle 49.394 %
- Pulses 19
- Width 25.302 ms
Ready
Frame based
Offset=0
T=1.000

练习9：ADC+DAC+DMA

任务9.2、实现上述ADC（DMA）+DAC（DMA）方式的数据测量系统。利用simulink模型，查看结果。

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！