STM32Cube高效开发教程（基础篇）

第15章　DAC

王维波
中国石油大学（华东）控制科学与工程学院

STM32Cube高效开发教程（基础篇）

作者：王维波，鄢志丹，王钊

人民邮电出版社

2021年9月出版

如果有读者需要本书课件的PPT版本用于备课，可以给作者发邮件免费获取，并可加入专门的教学和技术交流QQ群

邮箱：wangwb@upc. edu. cn

第15章　DAC

15.1 DAC功能概述

15.2 DAC的HAL驱动程序

15.3 示例1：软件触发DAC转换

15.4 示例2：输出三角波

15.1 DAC功能概述

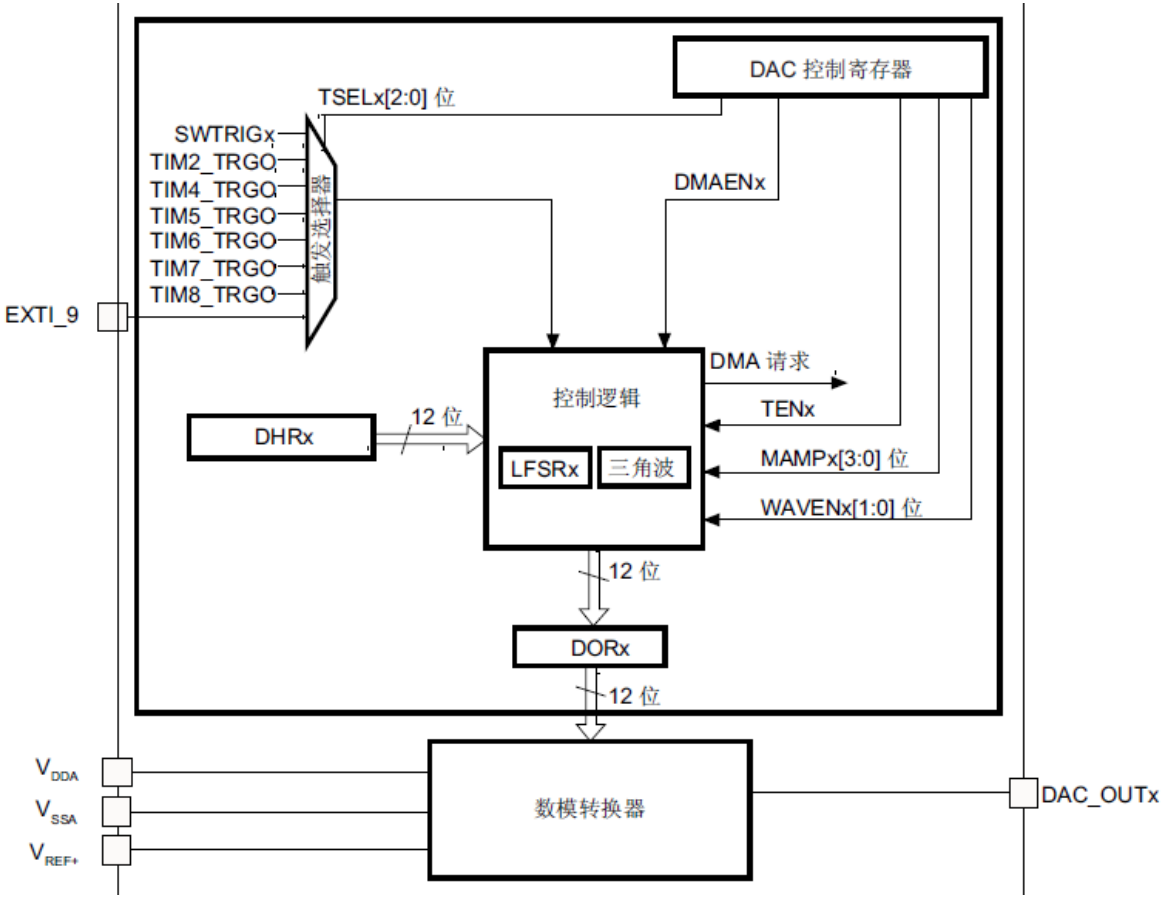
15.1.1 DAC的结构和特性

15.1.2 功能说明

15.1.1 DAC的结构和特性

STM32F407的DAC模块有2个DAC转换器，各对应一个输出通道。一个DAC的内部功能结构如图所示。

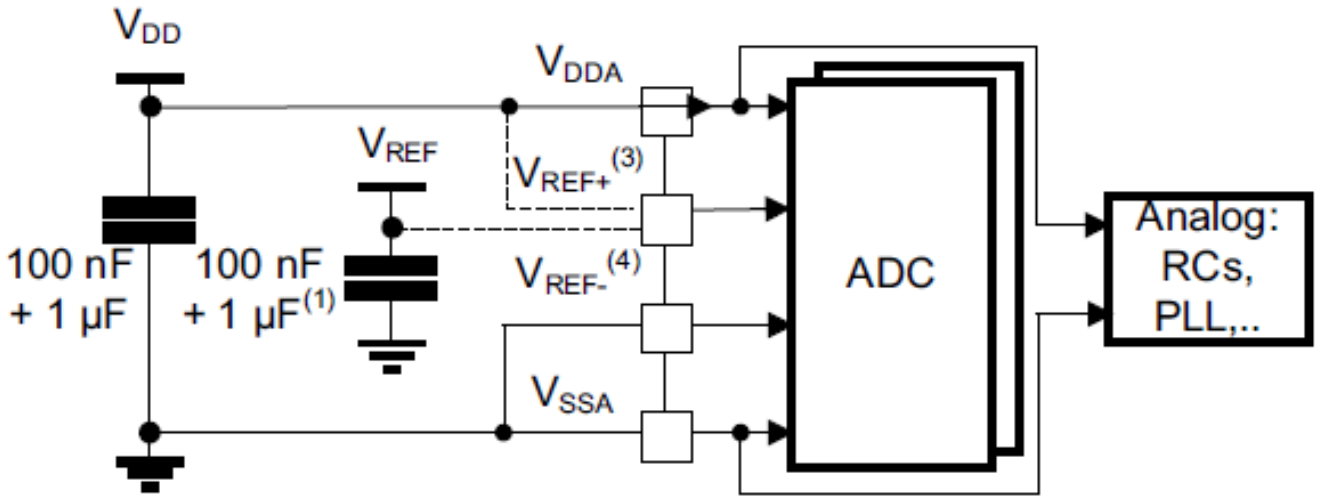
- DAC的核心是12位的数模转换器
- 数据输出寄存器DORx（x是1或2，表示通道1或通道2）
- 模拟电压输出到复用引脚DAC_OUTx
- 数据保持寄存器DHRx



DAC输出的模拟电压由寄存器DORx的数据值和参考电压V_{REF+}决定，输出电压计算公式是

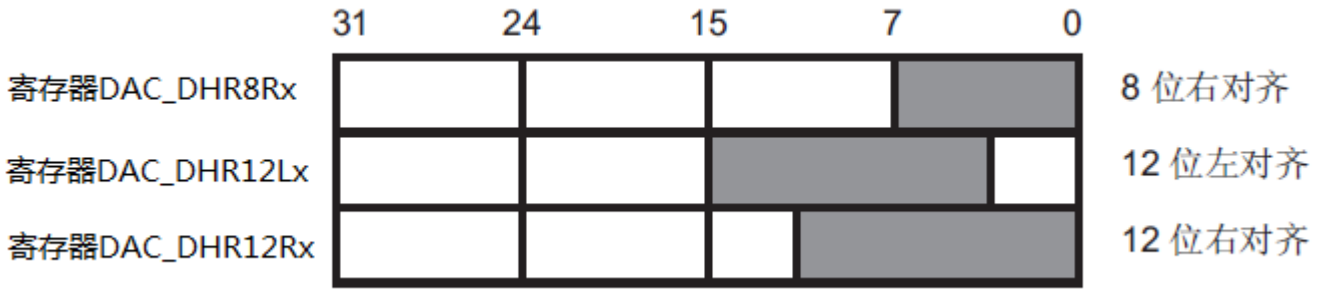
$$\text{DACoutput} = \frac{DOR_x}{4095} \times V_{REF+}$$

实际电路中一般采用数字电源V_{DD}作为V_{FRE+}，即3.3V



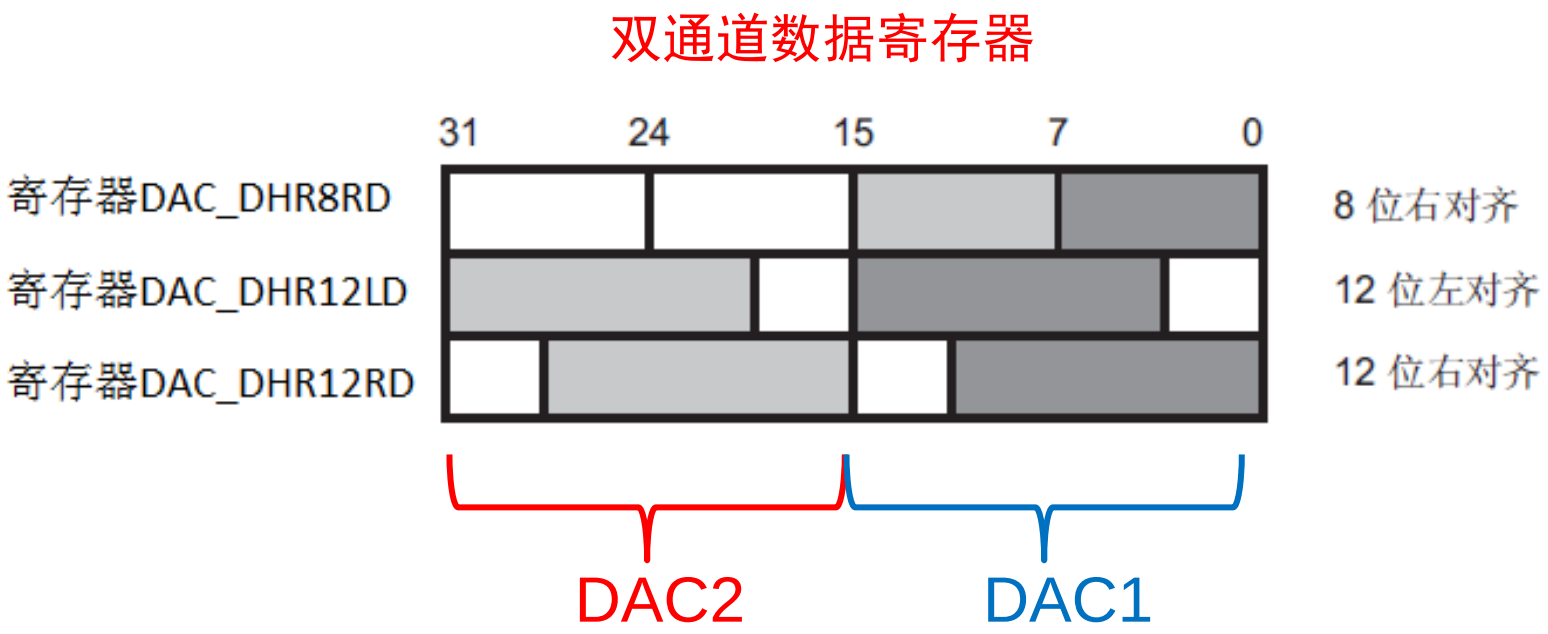
15.1.2 功能说明

1. DAC数据格式



单通道数据寄存器

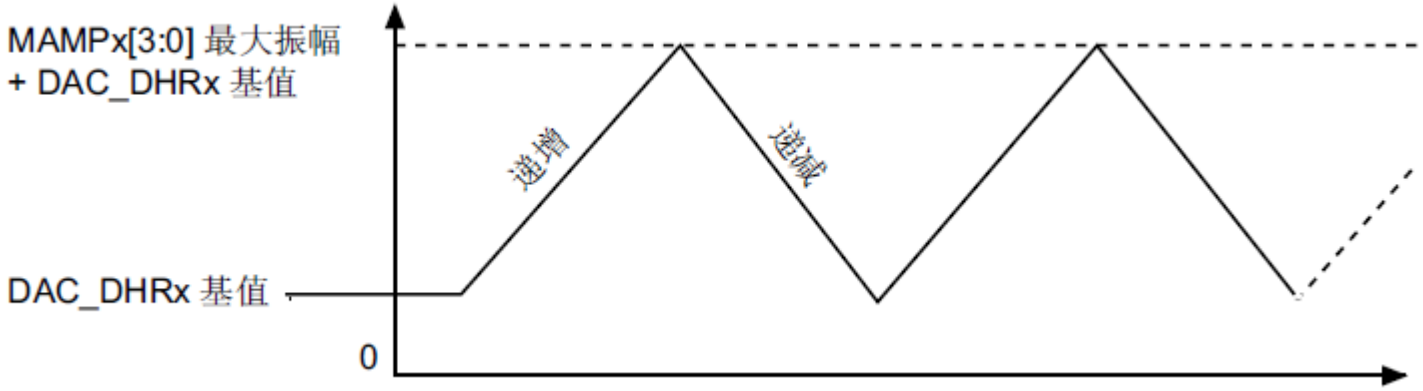
使用单通道独立输出时，向DAC写入数据有3种格式：8位右对齐，12位左对齐，12位右对齐。这3种格式的数据写入相应的对齐数据保持寄存器DAC_DHR8Rx、DAC_DHR12Lx或DAC_DHR12Rx



使用DAC双通道同步输出时，有3个专用的双通道寄存器用于向两个DAC通道同时写入数据，写入数据的格式有3种，其中高位是DAC2，低位是DAC1

2. 输出噪声波和三角波

DAC内部使用线性反馈移位寄存器LFSR生成变振幅的伪噪声。注意，要生成噪声波，必须使用外部触发。



可以在直流信号或慢变信号上叠加一个小幅三角波。在DAC控制寄存器DAC_CR的MAMPx[3:0]位设置一个参数用于表示三角波最大振幅，振幅从1到4095（非连续）

15.2 DAC的HAL驱动程序

15.2.1 DAC驱动宏函数

15.2.2 DAC驱动功能函数

15.2.1 DAC驱动宏函数

宏函数	功能描述
__HAL_DAC_DISABLE(__HANDLE__, __DAC_Channel__)	关闭DAC的某个通道
__HAL_DAC_ENABLE(__HANDLE__, __DAC_Channel__)	开启DAC的某个通道
__HAL_DAC_DISABLE_IT(__HANDLE__, __INTERRUPT__)	禁止DAC模块的某个中断事件源
__HAL_DAC_ENABLE_IT(__HANDLE__, __INTERRUPT__)	开启DAC模块的某个中断事件源
__HAL_DAC_GET_IT_SOURCE(__HANDLE__, __INTERRUPT__)	检查DAC模块的某个中断事件源是否开启
__HAL_DAC_GET_FLAG(__HANDLE__, __FLAG__)	获取某个事件的中断标志，检查事件是否发生
__HAL_DAC_CLEAR_FLAG(__HANDLE__, __FLAG__)	清除某个事件的中断标志

CubeMX自动生成的DAC外设初始化文件dac.c中会定义表示DAC的外设对象变量hdac。宏函数中的参数__HANDLE__是DAC外设对象指针，就可以用 &hdac。

```
DAC_HandleTypeDef  hdac;    //表示DAC的外设对象变量
```

DAC模块有2个DAC通道，用宏定义表示如下，可作为宏函数中参数__DAC_Channel__的取值。

```
#define DAC_CHANNEL_1    0x00000000U          //DAC通道1
#define DAC_CHANNEL_2    0x00000010U          //DAC通道2
```

15.2.2 DAC驱动功能函数

分组	函数名	功能描述
初始化和通道配置	HAL_DAC_Init()	DAC初始化
	HAL_DAC_MspInit()	DAC的MSP初始化函数
	HAL_DAC_ConfigChannel()	配置DAC通道1或通道2
	HAL_DAC_GetState()	返回DAC模块的状态
	HAL_DAC_GetError()	返回DAC模块的错误码
软件触发转换	HAL_DAC_Start()	启动某个DAC通道，可以是软件触发或外部触发
	HAL_DAC_Stop()	停止某个DAC通道
	HAL_DAC_GetValue()	返回某个DAC通道的输出值，就是返回数据输出寄存器的值
	HAL_DAC_SetValue()	设置某个DAC通道的输出值，就是设置数据保持寄存器的值
	HAL_DACEx_DualGetValue()	一次获取两个通道的输出值
	HAL_DACEx_DualSetValue()	同时为两个通道设置输出值
产生波形	HAL_DACEx_TriangleWaveGenerate()	在某个DAC通道上产生三角波
	HAL_DACEx_NoiseWaveGenerate()	在某个通道上产生随机信号，必须是外部触发

（续表）

分组	函数名	功能描述
DAC中断处理	HAL_DAC_IRQHandler()	DAC中断通用处理函数
	HAL_DAC_DMAUnderrunCallbackCh1	通道1出现DMA下溢事件中断的回调函数
	HAL_DACEx_DMAUnderrunCallbackCh2()	通道2出现DMA下溢事件中断的回调函数
DMA方式启动和停止	HAL_DAC_Start_DMA()	启动某个DAC通道的DMA方式传输，必须是外部触发
	HAL_DAC_Stop_DMA()	停止某个DAC通道的DMA方式传输
通道1的DMA流中断回调函数	HAL_DAC_ConvCpltCallbackCh1()	DMA传输完成事件中断的回调函数
	HAL_DAC_ConvHalfCpltCallbackCh1()	DMA传输半完成事件中断的回调函数
	HAL_DAC_ErrorCallbackCh1()	DMA传输错误事件的回调函数
通道2的DMA流中断回调函数	HAL_DACEx_ConvCpltCallbackCh2()	DMA传输完成事件中断的回调函数
	HAL_DACEx_ConvHalfCpltCallbackCh2()	DMA传输半完成事件中断的回调函数
	HAL_DACEx_ErrorCallbackCh2()	DMA传输错误事件中断的回调函数

1. 软件触发转换

以HAL_DAC_Start()启动的通道可以使用软件触发或外部触发。两个函数原型定义如下：

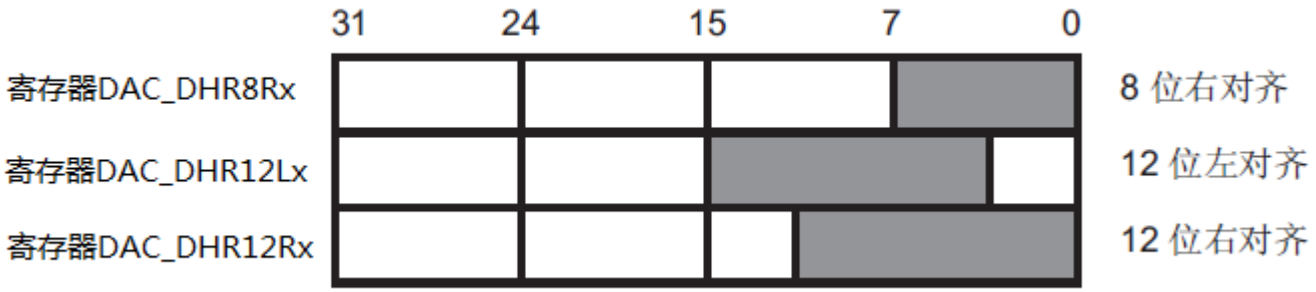
```
HAL_StatusTypeDef HAL_DAC_Start(DAC_HandleTypeDef* hdac,
    uint32_t Channel);

HAL_StatusTypeDef HAL_DAC_Stop(DAC_HandleTypeDef* hdac,
    uint32_t Channel);
```

函数HAL_DAC_SetValue()用于向一个DAC通道写入数据，实际就是将数据写入数据保持寄存器DHRx

```
HAL_StatusTypeDef HAL_DAC_SetValue(DAC_HandleTypeDef*
    hdac, uint32_t Channel, uint32_t Alignment, uint32_t Data);
```

其中，Alignment表示数据对齐格式，向单个DAC通道写入数据有3种对齐格式，可以从如下的3个宏定义中取值

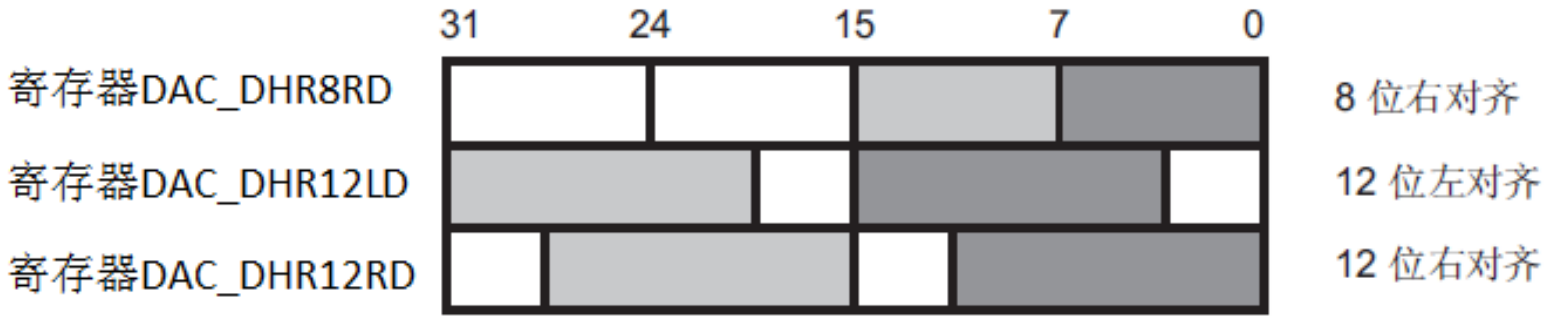
```
#define DAC_ALIGN_12B_R      0x00000000U  //12位右对齐
#define DAC_ALIGN_12B_L      0x00000004U  //12位左对齐
#define DAC_ALIGN_8B_R       0x00000008U  //8位右对齐
```



函数HAL_DACEx_DualSetValue()用于在双通道模式下向两个DAC通道同时写入数据，其函数原型定义如下：

```
HAL_StatusTypeDef HAL_DACEx_DualSetValue( DAC_HandleTypeDef*
    hdac,  uint32_t Alignment, uint32_t Data1, uint32_t Data2)
```

双通道模式写入数据的3种格式见图，参数Alignment的取值还是前面的对齐方式宏定义。



2. 产生波形

可以在输出信号上叠加一个三角波信号，需要在启动DAC通道前调用这个函数

```
HAL_StatusTypeDef HAL_DACEx_TriangleWaveGenerate
    ( DAC_HandleTypeDef* hdac,  uint32_t Channel,  uint32_t Amplitude);
```

其中，参数Amplitude是三角波最大幅度，用4位二进制数表示范围1到4095。

每次发生软件触发或外部触发时，三角波内部计数值就会加1。内部的三角波计数器就会递增或递减，在保障不溢出的情况下会和DHRx寄存器的值叠加后移送到DORx寄存器。

3. DMA方式传输

使用外部触发信号时，可以使用DMA方式启动DAC转换。

```
HAL_StatusTypeDef  HAL_DAC_Start_DMA(DAC_HandleTypeDef*
    hdac, uint32_t Channel, uint32_t* pData, uint32_t Length, uint32_t
    Alignment);
```

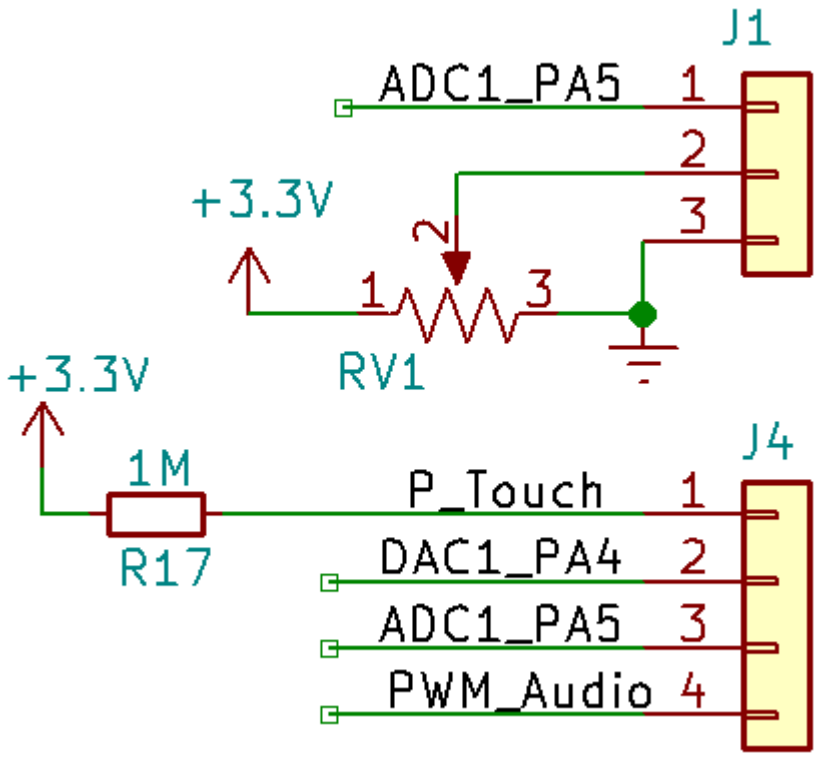
使用DMA方式传输时，每次外部信号触发时，就会将DMA缓存区的一个数据传输到DAC通道的数据输出寄存器DORx。设置存储器地址自增时，地址指针就会移动到DMA缓存区下一个数据点。

15.3 示例1：软件触发DAC转换

15.3.1 开发板上的DAC电路

要使用DAC，必须将J1上的1与2断开。J4上的Pin2是DAC1_OUT引脚，Pin3是DAC2_OUT引脚。

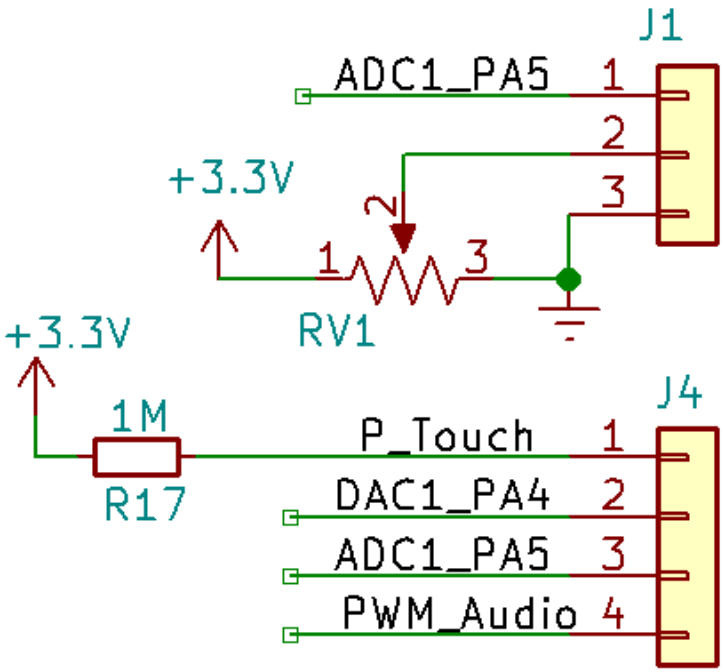
如果只使用DAC1，并且使用ADC1-IN5采集DAC1的输出，可以将J4的2和3用跳线帽短接。



15.3.2 示例功能和CubeMX项目设置

示例Demo15_1SoftTrig演示软件触发DAC转换。

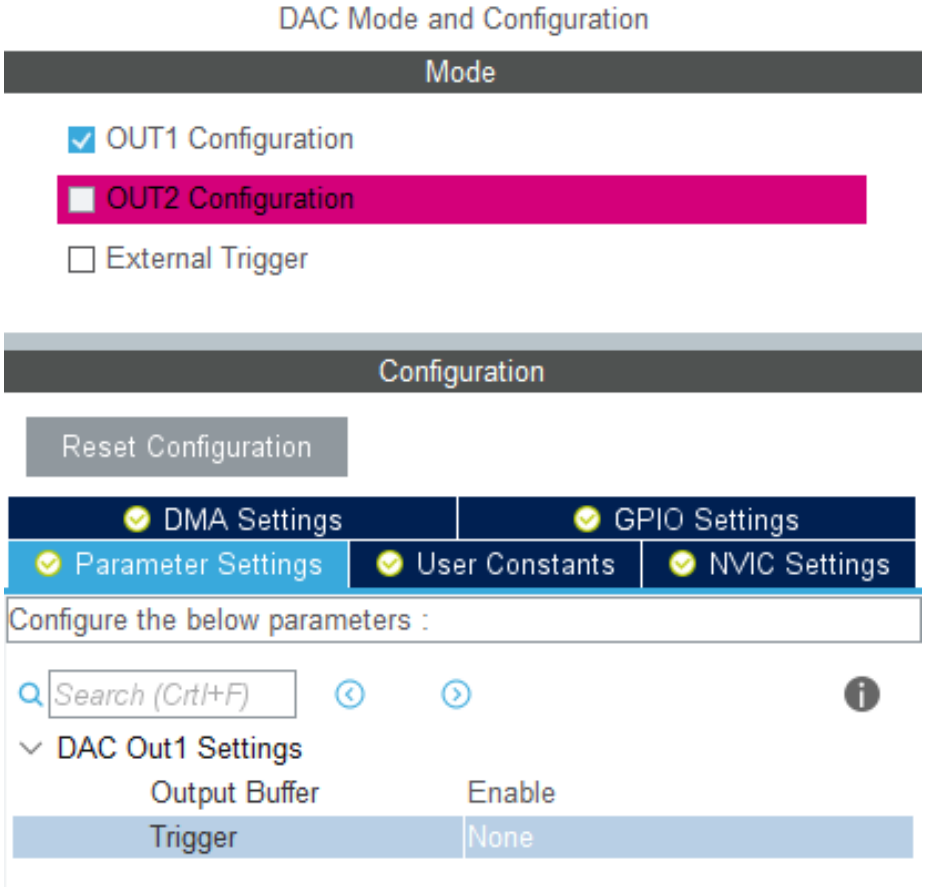
- 将跳线J4的2和3短接，DAC1的输出由ADC1-IN5采集
- ADC1-IN5输入，在定时器TIM3的TRGO信号触发下采集，TIM3定时周期500ms
- 通过按键KeyUp和KeyDown控制DAC1输出值的增减，用软件触发方式设置DAC1的输出值
- 在LCD上显示设定的DAC1输出值，以及ADC1-IN5采集的值



1. DAC的设置

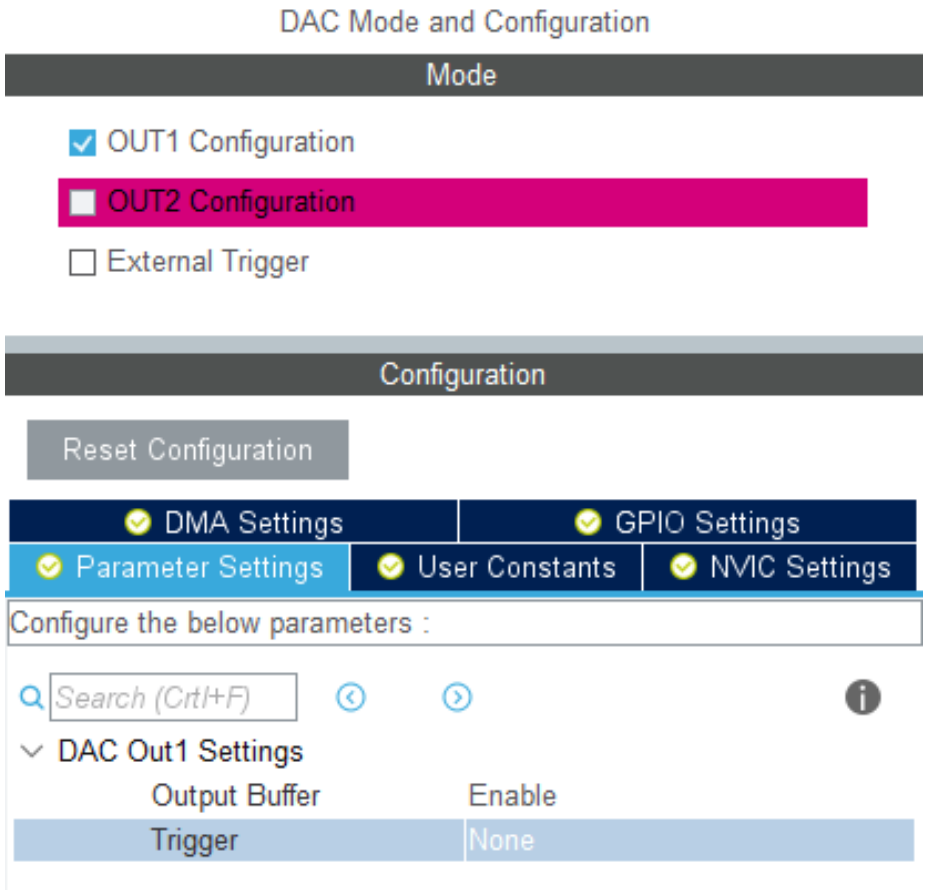
DAC的模式设置部分：

- OUT1 Configuration，启用DAC1
- OUT2 Configuration，启用DAC2
- External Trigger，是否使用外部触发。如果选择使用外部触发，会在参数设置部分出现外部触发信号源选项



DAC参数设置部分：

- Output Buffer，是否使用输出缓冲器。如果使用输出缓冲器，可以降低输出阻抗并提高输出的负载能力
- Trigger，外部触发信号源。触发信号源包括多个定时器的TRGO信号。本示例不使用触发信号，所以设置为None



2. ADC1和TIM3的设置

- 设置使用ADC1的IN5输入通道
- 使用TIM3的TRGO信号作为ADC1的外部触发信号源。开启ADC1的全局中断，设置其抢占优先级为1
- TIM3的定时周期设置为500ms，TRGO信号设置为UEV事件信号

15.3.2 程序功能实现

1. 主程序和ADC1中断处理

main()函数中的部分代码

打开示例源代码讲解

```
HAL_DAC_Start(&hdac,DAC_CHANNEL_1);   //启动DAC_OUT1
uint32_t  DacOutValue=1000;                    //DAC1输出设定值，范围0--4095
HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, DacOutValue);
LCD_ShowUintX(LcdX,LcdY,DacOutValue,4);  //显示设置的DAC1输出值
HAL_ADC_Start_IT(&hadc1);        //启动ADC中断方式输入
HAL_TIM_Base_Start(&htim3);      //启动TIM3，触发ADC定时采集
while (1)
{
    KEYS  curKey=ScanPressedKey(KEY_WAIT_ALWAYS);
    if (curKey==KEY_UP)
        DacOutValue += 50;
    else if  (curKey==KEY_DOWN)
        DacOutValue -= 50;

    HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, DacOutValue);
    LCD_ShowUintX(LcdX,LcdY,DacOutValue,4);    //显示设置的DAC1输出值
    HAL_Delay(300);          //消除按键抖动影响
}
```

ADC的中断回调函数

```
/* USER CODE BEGIN 4 */
/*   ADC转换完成中断回调函数   */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
      uint32_t val=HAL_ADC_GetValue(hadc);                  //读取ADC转换结果
      LCD_ShowUintX(LcdX,LcdY+LCD_SP20,val,5);              //显示原始值

      uint32_t Volt=3300*val;                   //单位：mV
      Volt=Volt>>12;                            //除以 2^12
      LCD_ShowUintX(LcdX,LcdY+2*LCD_SP20,Volt,4);     //电压值
}
/* USER CODE END 4 */
```

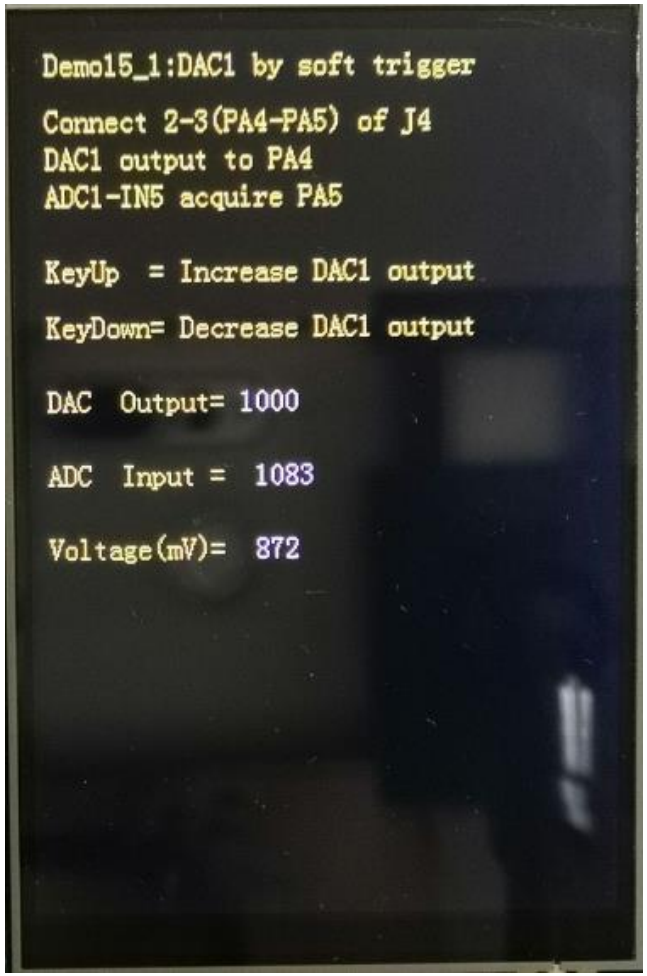
2. DAC初始化

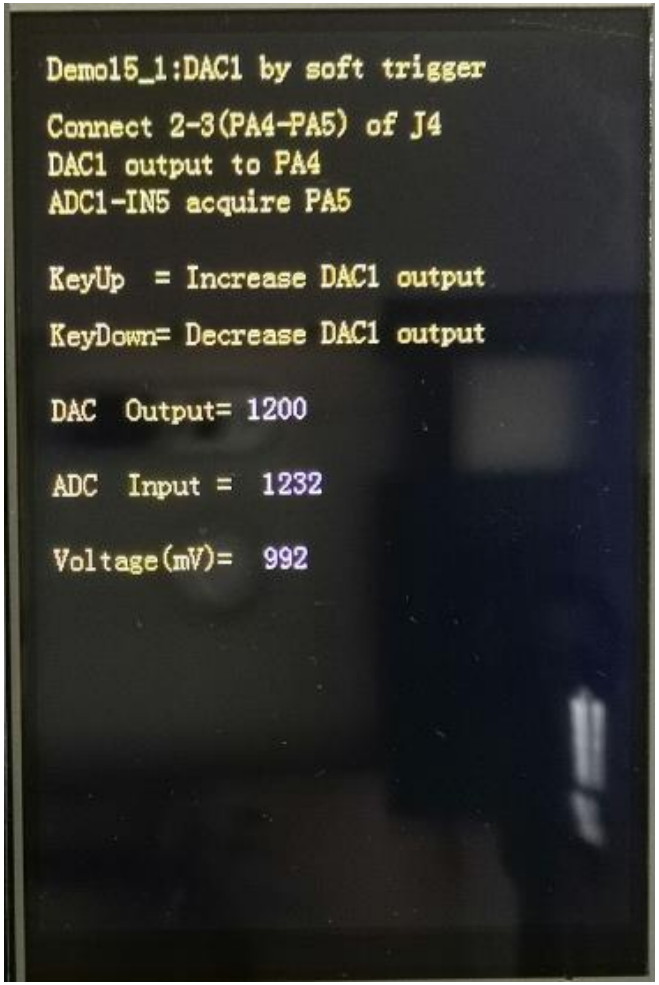
在文件dac.c中，有DAC的外设对象变量定义，和初始化函数MX_DAC_Init()

打开源代码讲解

```
DAC_HandleTypeDef  hdac;     //DAC外设对象变量
 void MX_DAC_Init(void)
{
      DAC_ChannelConfTypeDef sConfig = {0};
      hdac.Instance = DAC;        //DAC的寄存器基址
      if (HAL_DAC_Init(&hdac) != HAL_OK)      //DAC初始化
          Error_Handler();
      /** DAC OUT1 通道配置*/
      sConfig.DAC_Trigger = DAC_TRIGGER_NONE;//无外部触发
      sConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_ENABLE;  //输出缓冲器
      if (HAL_DAC_ConfigChannel(&hdac, &sConfig, DAC_CHANNEL_1) != HAL_OK)
          Error_Handler();
}
```

运行测试





DAC输出的模拟信号被ADC采集，按KeyUp和KeyDown键可以改变DAC的输出值

15.4 示例2：输出三角波

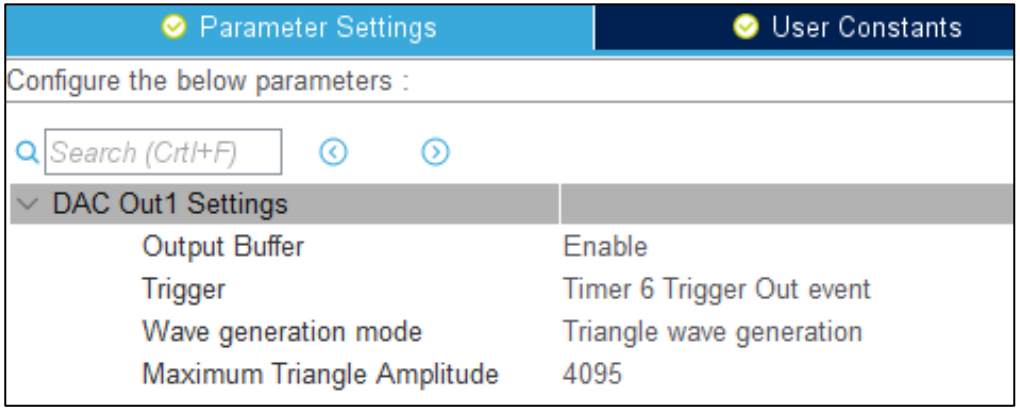
15.4.1 示例功能和CubeMX项目设置

示例Demo15_2TriangWave采用定时器TIM6的TRGO信号作为DAC1的触发信号，DAC1在触发信号驱动下输出三角波

1. DAC的设置

3个与外部触发信号源和生成三角波相关的参数

- Trigger，外部触发信号源
- Wave generation mode，波形生成模式。可选择三角波或噪声波

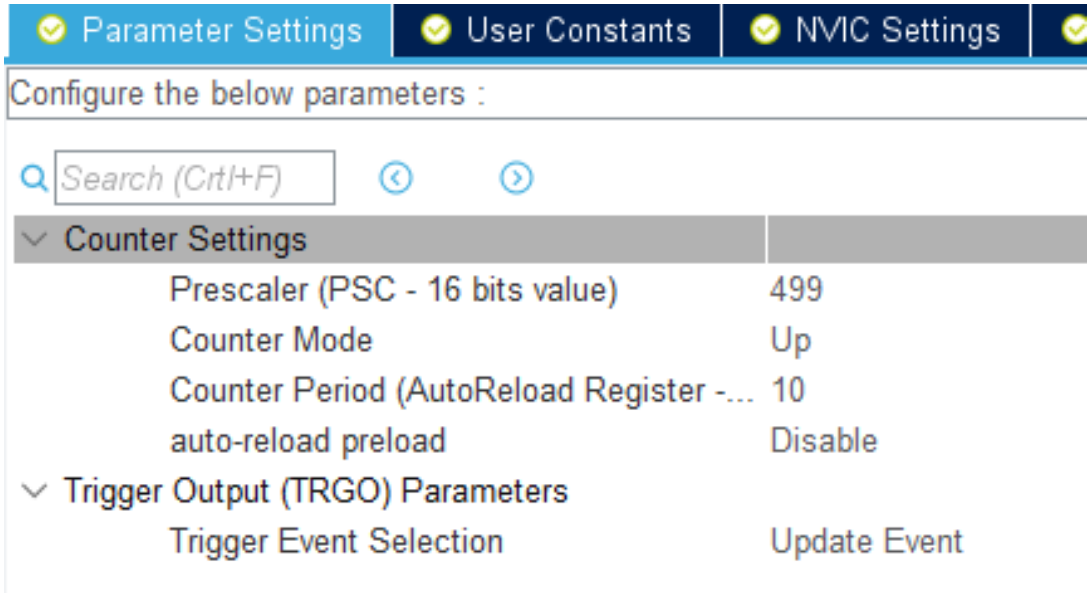


- Maximum Triangle Amplitude，三角波最大幅值。只能取1到4095之间某个固定的参数值

2. TIM6的设置

APB总线定时器时钟频率为50 MHz，500分频后计数器时钟信号是100 kHz，计数器周期（Counter Period）设置为10，所以TIM6的定时周期是0.1ms。

设置三角波的最大幅度为4095，DAC1在每次触发时使三角波幅度值加1（上行程）或减1（下行程），所以一个三角波的周期是819ms



15.4.2 程序功能实现

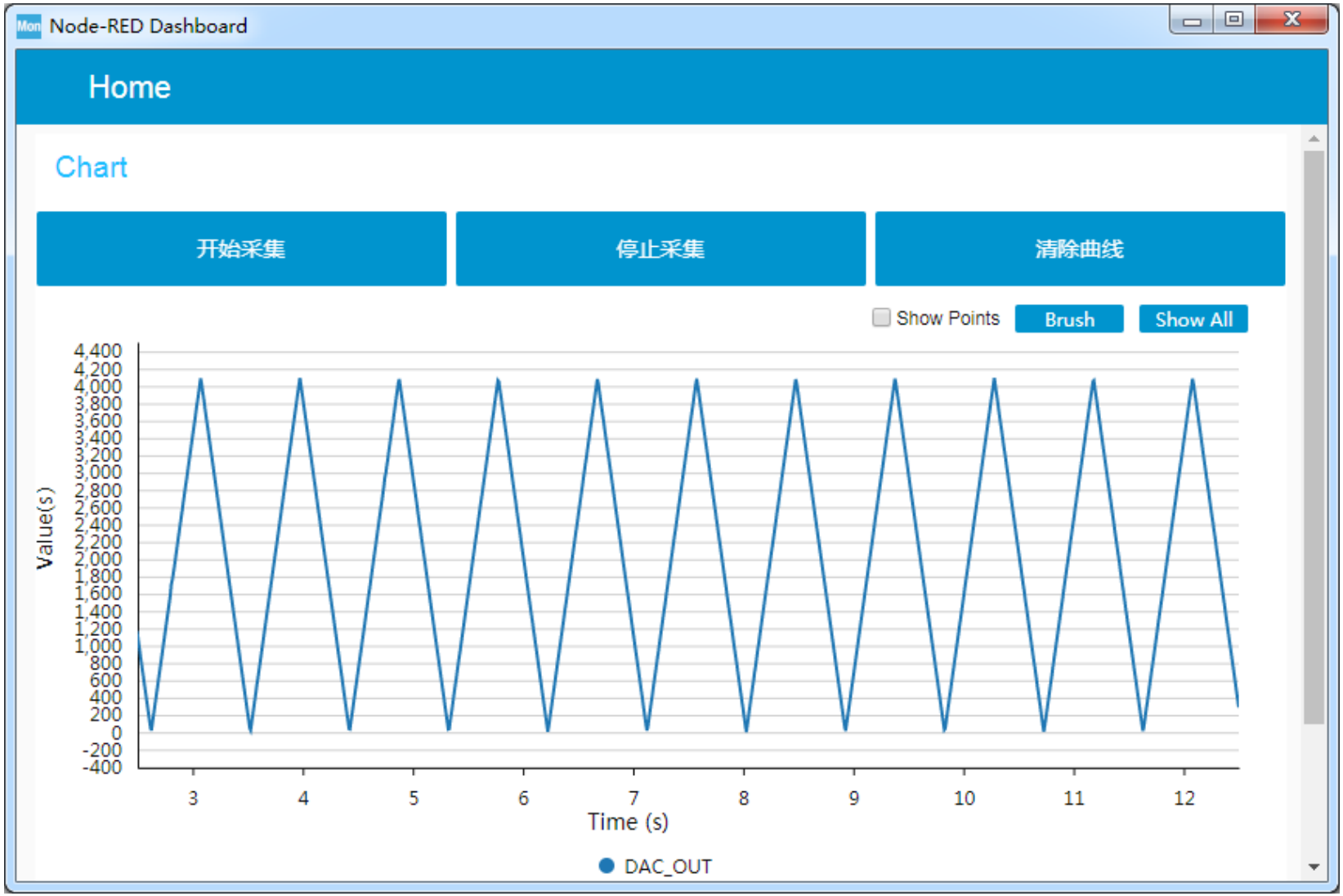
1. 主程序

启动DAC1和定时器TIM6后，DAC1就在TIM6的触发下输出三角波

```
HAL_DAC_Start(&hdac,DAC_CHANNEL_1); //启动DAC1
uint32_t DCValue=0;                 //12bits,直流分量
HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R,
DCValue);    //设置输出值
HAL_TIM_Base_Start(&htim6);     //启动TIM6，触发DAC1周期性输出
/* USER CODE END 2 */

/* Infinite loop */
while (1)
{
}
```

打开源代码讲解

可以使用CubeMonitor监测输出的三角波波形



练习任务

1. 自学教材的15.5节，掌握DAC使用DMA输出方式，输出自定义锯齿波的方法