```c
/*---------------------------------------------------------------------------
 * Name:    DAC.c
 * Purpose: Functions to initilise the DAC, and subsequently provide triangle wave
            functionality, noise generation, and supports arbitory function generation.
 * Note(s): Example code taken from STMicroElectronics Application Teams,
            DAC_SignalsGeneration example project
 *---------------------------------------------------------------------------
 *
 *---------------------------------------------------------------------------*/

#include "STM32F4xx.h"
#include "main_2.h"
#include "DAC.h"
#include "ArbitoryFunc.h"

// CMSIS data structure for DAC
DAC_InitTypeDef  DAC_InitStructure;

void DACs_Init(void)
{
   /* Preconfiguration before using DAC------*/
  GPIO_InitTypeDef GPIO_InitStructure;

  /* DMA1 clock and GPIOA clock enable (to be used with DAC) */
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_DMA1 | RCC_AHB1Periph_GPIOA, ENABLE);

  /* DAC Periph clock enable */
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_DAC, ENABLE);

  /* DAC channel 1 & 2 (DAC_OUT1 = PA.4)(DAC_OUT2 = PA.5) configuration */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4 | GPIO_Pin_5;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AN;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  /* TIM Configuration ----------*/
  TIM6_Config();
  TIM5_Config();

  /* Set DAC registers to default values */
  DAC_DeInit();
}

void TIM6_Config(void)
{
  TIM_TimeBaseInitTypeDef    TIM_TimeBaseStructure;

  /* TIM6 Periph clock enable */
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM6, ENABLE);

  /* Time base configuration */
  TIM_TimeBaseStructInit(&TIM_TimeBaseStructure);
  TIM_TimeBaseStructure.TIM_Period = 1;
  TIM_TimeBaseStructure.TIM_Prescaler = 0;
  TIM_TimeBaseStructure.TIM_ClockDivision = 0;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInit(TIM6, &TIM_TimeBaseStructure);

  /* TIM6 TRGO selection */
  TIM_SelectOutputTrigger(TIM6, TIM_TRGOSource_Update);

  /* TIM6 enable counter */
  TIM_Cmd(TIM6, ENABLE);
}

void DAC_Ch2_TriangleConfig(void)
{
 /* DAC channel2 Configuration */
  DAC_InitStructure.DAC_Trigger = DAC_Trigger_T6_TRGO;
  DAC_InitStructure.DAC_WaveGeneration = DAC_WaveGeneration_Triangle;
  DAC_InitStructure.DAC_LFSRUnmask_TriangleAmplitude = DAC_TriangleAmplitude_255;
  DAC_InitStructure.DAC_OutputBuffer = DAC_OutputBuffer_Enable;
  DAC_Init(DAC_Channel_2, &DAC_InitStructure);

  /* Set DAC channel2 DHR12RD register */
  DAC_SetChannel2Data(DAC_Align_12b_R, 0x100);
}

```

```c
void DAC_Ch1_NoiseConfig(void)
{
 /* DAC channel1 Configuration */
  DAC_InitStructure.DAC_Trigger = DAC_Trigger_T6_TRGO;
  DAC_InitStructure.DAC_WaveGeneration = DAC_WaveGeneration_Noise;
  DAC_InitStructure.DAC_LFSRUnmask_TriangleAmplitude = DAC_LFSRUnmask_Bits11_0;   // Max bits unmasked
  DAC_InitStructure.DAC_OutputBuffer = DAC_OutputBuffer_Enable;
  DAC_Init(DAC_Channel_1, &DAC_InitStructure);

  /* Set DAC Channel1 DHR12L register */
  DAC_SetChannel1Data(DAC_Align_12b_L, 0x7FF0);
}

void DAC_Noise_On(void)
{
  /* Enable DAC Channel1 */
  DAC_Cmd(DAC_Channel_1, ENABLE);
}

void DAC_Noise_Off(void)
{
  /* Disable DAC Channel1 */
  DAC_Cmd(DAC_Channel_1, DISABLE);
}

void DAC_Triangle_On(void) {
  /* Enable DAC Channel2 */
  DAC_Cmd(DAC_Channel_2, ENABLE);
}

void DAC_Traingle_Off(void) {
  /* Disable DAC Channel2 */
  DAC_Cmd(DAC_Channel_2, DISABLE);
}

```