Experiment 11

Title: DAC (Digital to Analog Converter).

Aim: To create a project in STM32Cube IDE to convert Digital values to Voltage output using DAC.

Tool used: Tool used for this assignment is **STM32CubeIDE**.

Procedure:

1. Create a new STM32 project with a suitable project name.

2. IOC UI will open in that configure desired pins as input/output.

3. Under system core select Serial-wire for Debug option.

4. In DAC mode configuration select OUT1 Configuration.

5. In parameter settings output buffer is enabled.

6. Select TIM2 and give Prescaler 90-1 and counter period 100.

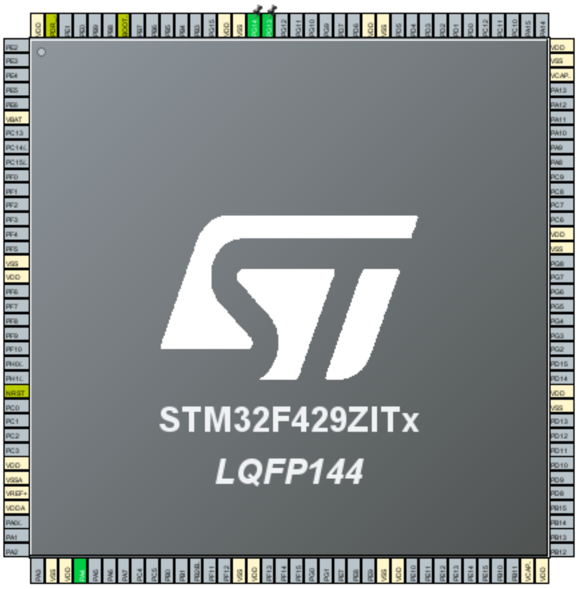
4. Press Ctrl+S to generate the code.

5. In the main.c file add the desired code.

6. Go to Project-> Build Project

7. Connect the discovery Board and go to Run-> Run.

CubeMx pin diagram:

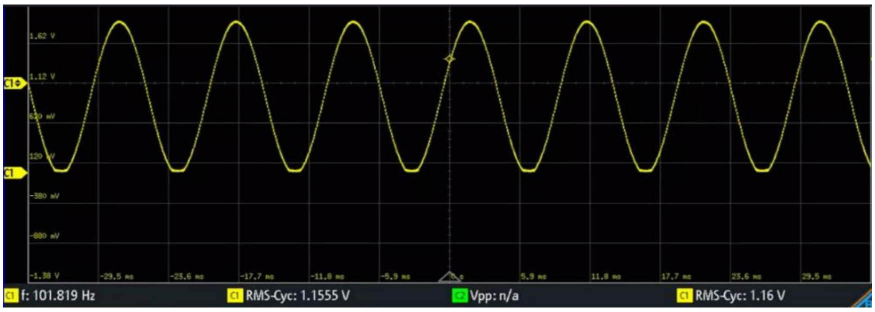


Code:

```
/* USER CODE BEGIN Header */
/**

***************************************************************************
***
  * @file           : main.c
  * @brief          : Main program body

***************************************************************************
***
  * @attention
  *
  * Copyright (c) 2023 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

***************************************************************************
***
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "math.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
uint32_t sine_val[100];

#define PI 3.1415926

void calcsin ()
{
        for (int i=0; i<100; i++)
        {
                sine_val[i] = ((sin(i*2*PI/100) + 1)*(4094/2));
        }
}
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
DAC_HandleTypeDef hdac;
DMA_HandleTypeDef hdma_dac1;

DMA2D_HandleTypeDef hdma2d;

TIM_HandleTypeDef htim2;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_DAC_Init(void);
static void MX_TIM2_Init(void);
static void MX_DMA2D_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();
  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_DAC_Init();
  MX_TIM2_Init();
  MX_DMA2D_Init();
  HAL_TIM_Base_Start(&htim2);
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */
  calcsin();
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  HAL_DAC_Start_DMA(&hdac, DAC1_CHANNEL_1, sine_val, 100,
DAC_ALIGN_12B_R);
  while (1)
  {
    /* USER CODE END WHILE */
	    HAL_GPIO_TogglePin(GPIOG, GPIO_PIN_14); //Toggle the state of pin PC9
	    HAL_Delay(100);
	    HAL_GPIO_TogglePin(GPIOG, GPIO_PIN_13); //Toggle the state of pin PC9
	    HAL_Delay(100);
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
```

```
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE
3);

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief DAC Initialization Function
  * @param None
  * @retval None
  */
static void MX_DAC_Init(void)
{

  /* USER CODE BEGIN DAC_Init 0 */

  /* USER CODE END DAC_Init 0 */

  DAC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN DAC_Init 1 */

  /* USER CODE END DAC_Init 1 */
  /** DAC Initialization
  */
  hdac.Instance = DAC;
  if (HAL_DAC_Init(&hdac) != HAL_OK)
  {
    Error_Handler();
  }

  /** DAC channel OUT1 config
  */
  sConfig.DAC_Trigger = DAC_TRIGGER_T2_TRGO;
  sConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_ENABLE;
  if (HAL_DAC_ConfigChannel(&hdac, &sConfig, DAC_CHANNEL_1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN DAC_Init 2 */

  /* USER CODE END DAC_Init 2 */

}

/**
  * @brief DMA2D Initialization Function
  * @param None
  * @retval None
  */
static void MX_DMA2D_Init(void)
{

  /* USER CODE BEGIN DMA2D_Init 0 */

  /* USER CODE END DMA2D_Init 0 */

  /* USER CODE BEGIN DMA2D_Init 1 */

  /* USER CODE END DMA2D_Init 1 */
  hdma2d.Instance = DMA2D;
  hdma2d.Init.Mode = DMA2D_M2M;
  hdma2d.Init.ColorMode = DMA2D_OUTPUT_ARGB8888;
  hdma2d.Init.OutputOffset = 0;
  hdma2d.LayerCfg[1].InputOffset = 0;
  hdma2d.LayerCfg[1].InputColorMode = DMA2D_INPUT_ARGB8888;
  hdma2d.LayerCfg[1].AlphaMode = DMA2D_NO_MODIF_ALPHA;
  hdma2d.LayerCfg[1].InputAlpha = 0;
  if (HAL_DMA2D_Init(&hdma2d) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_DMA2D_ConfigLayer(&hdma2d, 1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN DMA2D_Init 2 */

  /* USER CODE END DMA2D_Init 2 */

}

/**
  * @brief TIM2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM2_Init(void)
{

  /* USER CODE BEGIN TIM2_Init 0 */

  /* USER CODE END TIM2_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM2_Init 1 */

  /* USER CODE END TIM2_Init 1 */
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 16-1;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 100-1;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM2_Init 2 */
  /* USER CODE END TIM2_Init 2 */

}

/**
  * Enable DMA controller clock
  */
static void MX_DMA_Init(void)
{

  /* DMA controller clock enable */
  __HAL_RCC_DMA1_CLK_ENABLE();

  /* DMA interrupt init */
  /* DMA1_Stream5_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA1_Stream5_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(DMA1_Stream5_IRQn);

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOG_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13|GPIO_PIN_14, GPIO_PIN_RESET);

  /*Configure GPIO pins : PG13 PG14 */
  GPIO_InitStruct.Pin = GPIO_PIN_13|GPIO_PIN_14;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */
/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

Output:



Functions used:

```
HAL_GPIO_TogglePin(GPIOx, GPIO Pin Number);

HAL_TIM_Base_Start(&handle_variable_name);

HAL_DAC_Start_DMA(&hdac, DAC1_CHANNEL_1, sine_val, 100,
DAC_ALIGN_12B_R);
```

Result:

Basic STM32Cube project for converting Digital values to Voltage output using DAC was built using STM32CubeIDE.