DSP_Exemple_2\Core\Src\main.c

```
/**
 *
 * Processamento Digital de Sinais - 2023/1
 *
 * Aline Nunes de Souza
 * Davi de Souza Leão Schmitz
 *
 *
 *
 *
 ******************************************************************************
 * @file           : main.c
 * @brief          : Main program body
 ******************************************************************************
 ** This notice applies to any and all portions of this file
 * that are not between comment pairs USER CODE BEGIN and
 * USER CODE END. Other portions of this file, whether
 * inserted by the user or by software development tools
 * are owned by their respective copyright owners.
 *
 * COPYRIGHT(c) 2018 STMicroelectronics
 *
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
 *   1. Redistributions of source code must retain the above copyright notice,
 *      this list of conditions and the following disclaimer.
 *   2. Redistributions in binary form must reproduce the above copyright notice,
 *      this list of conditions and the following disclaimer in the documentation
 *      and/or other materials provided with the distribution.
 *   3. Neither the name of STMicroelectronics nor the names of its contributors
 *      may be used to endorse or promote products derived from this software
 *      without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 ******************************************************************************
 */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f3xx_hal.h"
#include "stdlib.h"

/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private variables ---------------------------------------------------------*/
```

```c
ADC_HandleTypeDef hadc1;

DAC_HandleTypeDef hdac1;

TIM_HandleTypeDef htim3;

UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/
//Vari◆vel com o valor lido do ADC.
volatile uint32_t g_ADCValue, g_DACValue;
uint8_t buffer[50];

// Coeficientes
float b0;
float b1;
float b2;

float a1;
float a2;

// Variaveis de estado
uint32_t xh1 = 0;
uint32_t xh2 = 0;

uint32_t yh1 = 0;
uint32_t yh2 = 0;

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_DAC1_Init(void);
static void MX_ADC1_Init(void);
static void MX_TIM3_Init(void);
static void MX_NVIC_Init(void);

/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------*/

/* USER CODE END PFP */

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
 * @brief  The application entry point.
 *
 * @retval None
 */
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_USART2_UART_Init();
  MX_DAC1_Init();
  MX_ADC1_Init();
  MX_TIM3_Init();

  /* Initialize interrupts */
  MX_NVIC_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */

  HAL_TIM_Base_Start_IT(&htim3);
  HAL_DAC_Start(&hdac1, DAC_CHANNEL_1);

    float Fs = 5000;  // taxa de amostragem

    float fc = 500;  //frequencia de corte

    float zeta = 0.707;
    float pi = 3.14159265358979323846;
    float C = 1/(tan(pi*fc/Fs));

    b0 = 1/(1+2*zeta*C+C*C);
    b1 = 2*b0;
    b2 = b0;

    a1 = 2*b0*(1-C*C);
    a2 = b0*(1-2*zeta*C+C*C);

    while (1)
    {
      /* USER CODE END WHILE */

      /* USER CODE BEGIN 3 */

    }

  /* USER CODE END 3 */
```

```
}

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{

RCC_OscInitTypeDef RCC_OscInitStruct;
RCC_ClkInitTypeDef RCC_ClkInitStruct;
RCC_PeriphCLKInitTypeDef PeriphClkInit;

/**Initializes the CPU, AHB and APB busses clocks
*/
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
RCC_OscInitStruct.HSIState = RCC_HSI_ON;
RCC_OscInitStruct.HSICalibrationValue = 16;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL16;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

/**Initializes the CPU, AHB and APB busses clocks
*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK

|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_ADC12;
PeriphClkInit.Adc12ClockSelection = RCC_ADC12PLLCLK_DIV1;
if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

/**Configure the Systick interrupt time
*/
HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);

/**Configure the Systick
*/
HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);

/* SysTick_IRQn interrupt configuration */
HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
}
```

```c
/**
 * @brief NVIC Configuration.
 * @retval None
 */
static void MX_NVIC_Init(void)
{
/* ADC1_2_IRQn interrupt configuration */
HAL_NVIC_SetPriority(ADC1_2_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(ADC1_2_IRQn);
/* TIM3_IRQn interrupt configuration */
HAL_NVIC_SetPriority(TIM3_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(TIM3_IRQn);
}

/* ADC1 init function */
static void MX_ADC1_Init(void)
{

ADC_MultiModeTypeDef multimode;
ADC_ChannelConfTypeDef sConfig;

/**Common config
*/
hadc1.Instance = ADC1;
hadc1.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV1;
hadc1.Init.Resolution = ADC_RESOLUTION_12B;
hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
hadc1.Init.ContinuousConvMode = DISABLE;
hadc1.Init.DiscontinuousConvMode = DISABLE;
hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
hadc1.Init.NbrOfConversion = 1;
hadc1.Init.DMAContinuousRequests = DISABLE;
hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
hadc1.Init.LowPowerAutoWait = DISABLE;
hadc1.Init.Overrun = ADC_OVR_DATA_OVERWRITTEN;
if (HAL_ADC_Init(&hadc1) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

/**Configure the ADC multi-mode
*/
multimode.Mode = ADC_MODE_INDEPENDENT;
if (HAL_ADCEx_MultiModeConfigChannel(&hadc1, &multimode) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

/**Configure Regular Channel
*/
sConfig.Channel = ADC_CHANNEL_1;
sConfig.Rank = ADC_REGULAR_RANK_1;
sConfig.SingleDiff = ADC_SINGLE_ENDED;
sConfig.SamplingTime = ADC_SAMPLETIME_1CYCLE_5;
sConfig.OffsetNumber = ADC_OFFSET_NONE;
sConfig.Offset = 0;
if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
{
```

```
_Error_Handler(__FILE__, __LINE__);
}

}

/* DAC1 init function */
static void MX_DAC1_Init(void)
{

DAC_ChannelConfTypeDef sConfig;

/**DAC Initialization
*/
hdac1.Instance = DAC1;
if (HAL_DAC_Init(&hdac1) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

/**DAC channel OUT1 config
*/
sConfig.DAC_Trigger = DAC_TRIGGER_NONE;
sConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_ENABLE;
if (HAL_DAC_ConfigChannel(&hdac1, &sConfig, DAC_CHANNEL_1) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

}

/* TIM3 init function */
static void MX_TIM3_Init(void)
{

TIM_MasterConfigTypeDef sMasterConfig;
TIM_OC_InitTypeDef sConfigOC;

htim3.Instance = TIM3;
htim3.Init.Prescaler = 100;
htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
htim3.Init.Period = 64;
htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
if (HAL_TIM_OC_Init(&htim3) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

sConfigOC.OCMode = TIM_OCMODE_TIMING;
sConfigOC.Pulse = 0;
sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
if (HAL_TIM_OC_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
```

```
  {
  _Error_Handler(__FILE__, __LINE__);
  }

  }

  /* USART2 init function */
  static void MX_USART2_UART_Init(void)
  {

  huart2.Instance = USART2;
  huart2.Init.BaudRate = 38400;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
  _Error_Handler(__FILE__, __LINE__);
  }

  }

  /** Configure pins as
   * Analog
   * Input
   * Output
   * EVENT_OUT
   * EXTI
   */
  static void MX_GPIO_Init(void)
  {

  GPIO_InitTypeDef GPIO_InitStruct;

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
```

```
}

/* USER CODE BEGIN 4 */

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
HAL_ADC_Start_IT(&hadc1);
}

void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
//HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);

/* Faz a leitura do valor analógico convertido */
g_ADCValue = HAL_ADC_GetValue(&hadc1);
//g_DACValue = g_ADCValue;

g_DACValue = b0*g_ADCValue + b1*xh1  + b2*xh2 - a1*yh1 - a2*yh2;

xh1 = xh1;
xh1 = g_ADCValue;

yh2 = yh1;
yh1 = g_DACValue;

HAL_DAC_SetValue(&hdac1, DAC_CHANNEL_1,DAC_ALIGN_12B_R,g_DACValue);
}

/* USER CODE END 4 */

/**
 * @brief  This function is executed in case of error occurrence.
 * @param  file: The file name as string.
 * @param  line: The line in file as a number.
 * @retval None
 */
void _Error_Handler(char *file, int line)
{
/* USER CODE BEGIN Error_Handler_Debug */
/* User can add his own implementation to report the HAL error return state */
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,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
/* USER CODE END 6 */
```

```
}
#endif /* USE_FULL_ASSERT */

/**
 * @}
 */

/**
 * @}
 */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```