/\*

 \* task3.c

 \*

 \*/

#include "init.h"

ADC\_HandleTypeDef hadc1;

GPIO\_InitTypeDef GPIO\_InitStruct;

DAC\_HandleTypeDef DACHandler;

DMA\_HandleTypeDef ADC\_DMA\_Handle;//Declare DMA handler struct for ADC module

unsigned int ADCresult1;

int X;

int X1 = 0;

int X2 = 0;

double Y;

int Y1 = 0;

char convComplete=0;//adc conversion complete flag

//Function Prototypes

void configureDAC();

void configureADC();

void configureDMA();

void filter();

int main(void)

{

    // Initialize the system

    Sys\_Init();

    configureDAC();

    configureADC();

    configureDMA();

    // Turn on DAC at Channel 1

    HAL\_DAC\_Start(&DACHandler, DAC\_CHANNEL\_1);

    // start ADC conversion

    HAL\_ADC\_Start\_DMA(&hadc1, &ADCresult1, 1);

    //infinite loop

    while(1){

        filter();

    }

}

void filter(){

    int DACValue;

    while(!convComplete){} // hold until the ADC reading is complete

    convComplete = 0;//set flag back to 0

    ADCresult1 = HAL\_ADC\_GetValue(&hadc1);//Get new ADC reading

    X = ADCresult1;

    // Filter

    Y = 0.312500\*X + 0.240385\*X1+ 0.312500\*X2+ 0.296875\*Y1;

    DACValue = (int)(Y);

    HAL\_DAC\_SetValue(&DACHandler, DAC\_CHANNEL\_1, DAC\_ALIGN\_12B\_R, DACValue);//Set DC valueA

    // Update value

    X2 = X1;

    X1 = X;

    Y1 = DACValue;

}

void configureDAC()

{

    // DAC part are the same as LAB 4

    \_\_HAL\_RCC\_DAC\_CLK\_ENABLE();

    DAC\_ChannelConfTypeDef ChannelConfig;

    ChannelConfig.DAC\_Trigger = DAC\_TRIGGER\_NONE;

    ChannelConfig.DAC\_OutputBuffer = DAC\_OUTPUTBUFFER\_ENABLE;

    DACHandler.Instance = DAC;

    HAL\_DAC\_Init(&DACHandler);

    // Configure the ADC channel

    HAL\_DAC\_ConfigChannel(&DACHandler, &ChannelConfig, DAC\_CHANNEL\_1);

}

void configureADC()

{

    ADC\_ChannelConfTypeDef ChannelConfig;

    \_\_HAL\_RCC\_ADC1\_CLK\_ENABLE();

    // USE ADC1

    hadc1.Instance = ADC1;

    hadc1.Init.ClockPrescaler = ADC\_CLOCK\_SYNC\_PCLK\_DIV4;

    hadc1.Init.Resolution = ADC\_RESOLUTION\_12B;

    hadc1.Init.ScanConvMode = DISABLE;

    hadc1.Init.ContinuousConvMode = ENABLE;

    hadc1.Init.DiscontinuousConvMode = DISABLE;

    hadc1.Init.DataAlign = ADC\_DATAALIGN\_RIGHT;

    hadc1.Init.NbrOfConversion = 1;

    // Enable DMA

    hadc1.Init.DMAContinuousRequests = ENABLE;

    hadc1.Init.EOCSelection = ADC\_EOC\_SEQ\_CONV;

    HAL\_ADC\_Init(&hadc1);

    ChannelConfig.Channel = ADC\_CHANNEL\_6;

    ChannelConfig.Rank = 1;

    // Choose sample time not too long

    ChannelConfig.SamplingTime = ADC\_SAMPLETIME\_56CYCLES;

    HAL\_ADC\_ConfigChannel(&hadc1, &ChannelConfig);

    /\* Available sampling times:

        ADC\_SAMPLETIME\_3CYCLES

        ADC\_SAMPLETIME\_15CYCLES

        ADC\_SAMPLETIME\_28CYCLES

        ADC\_SAMPLETIME\_56CYCLES

        ADC\_SAMPLETIME\_84CYCLES

        ADC\_SAMPLETIME\_112CYCLES

        ADC\_SAMPLETIME\_144CYCLES

        ADC\_SAMPLETIME\_480CYCLES

    \*/

}

void HAL\_ADC\_MspInit(ADC\_HandleTypeDef \*hadc)

{

    \_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

    \_\_HAL\_RCC\_GPIOF\_CLK\_ENABLE();

    fflush(stdout);

    // ADC GPIO Init Pin A0

    GPIO\_InitStruct.Pin = GPIO\_PIN\_6;

    GPIO\_InitStruct.Mode = GPIO\_MODE\_ANALOG;

    GPIO\_InitStruct.Pull = GPIO\_PULLUP;

    GPIO\_InitStruct.Speed = GPIO\_SPEED\_HIGH;

    HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

    // DAC GPIO init Pin A1

    GPIO\_InitTypeDef GPIOinit;

    GPIOinit.Pin = GPIO\_PIN\_4;          //Pin4 of GPIOA

    GPIOinit.Mode = GPIO\_MODE\_ANALOG;   //Set Analog mode

    GPIOinit.Pull = GPIO\_NOPULL;        //No pull up

    HAL\_GPIO\_Init(GPIOA,&GPIOinit);//Initialize GPIOA Pin4 as configured above

}

void configureDMA()

{

    // Enable DMA2 for ADC

    \_\_HAL\_RCC\_DMA2\_CLK\_ENABLE();//Enable DMA2 clock

    //ADC uses DMA2 Stream0 channel 0

    ADC\_DMA\_Handle.Instance = DMA2\_Stream0;

    ADC\_DMA\_Handle.Init.Channel = DMA\_CHANNEL\_0;

    //ADC peripheral to memory

    ADC\_DMA\_Handle.Init.Direction = DMA\_PERIPH\_TO\_MEMORY;

    // No need for increement

    ADC\_DMA\_Handle.Init.PeriphInc = DMA\_PINC\_DISABLE;

    ADC\_DMA\_Handle.Init.MemInc = DMA\_MINC\_DISABLE;

    // Data alignment is 1 byte

    ADC\_DMA\_Handle.Init.MemDataAlignment = DMA\_MDATAALIGN\_BYTE;

    ADC\_DMA\_Handle.Init.PeriphDataAlignment = DMA\_MDATAALIGN\_BYTE;

    ADC\_DMA\_Handle.Init.Mode = DMA\_CIRCULAR;//Circular Mode

    ADC\_DMA\_Handle.Init.Priority = DMA\_PRIORITY\_LOW;

    // No FIFO in this lab

    ADC\_DMA\_Handle.Init.FIFOMode = DMA\_FIFOMODE\_DISABLE;

    // Init DMA

    HAL\_DMA\_Init(&ADC\_DMA\_Handle);

    // Enable Interrupt

    HAL\_NVIC\_EnableIRQ(DMA2\_Stream0\_IRQn);

    // Enable link

    \_\_HAL\_LINKDMA(&hadc1, DMA\_Handle, ADC\_DMA\_Handle);

}

void DMA2\_Stream0\_IRQHandler(){

    HAL\_DMA\_IRQHandler(&ADC\_DMA\_Handle);

}

void HAL\_ADC\_ConvCpltCallback(ADC\_HandleTypeDef\* hadc){

    convComplete = 1;   // Set the flag once the ADC reading complete

}