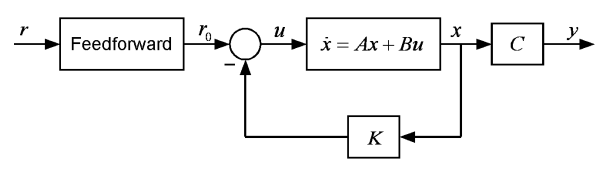
|  |  |
| --- | --- |
| 년도-학기 | 2021년 2학기 |
| 과목명 | 자동화프로그래밍 |

|  |  |
| --- | --- |
| **번호** | **실험 제목** |
| LAB 6 | State Feedback Controller |

|  |  |
| --- | --- |
| 실험 일자 | 2021년 10월 19일 |
| 제출자 이름 | 강\*\* |
| 제출자 학번 | 201803\*\*\*\* |
| 팀원 이름 |  |
|  |  |

**Chapter 1. 관련 이론(Theoretical Background)**

<Full-state Feedback>



Full-state Feedback controller는 시스템의 상태방정식이 일 때 상태변수인 모든 x를 제어입력 u에 반영하여 피드백 하는 것을 의미한다.

따라서 가 되고, 가 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

일 때 이고 이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

따라서 로 나타낼 수 있다.

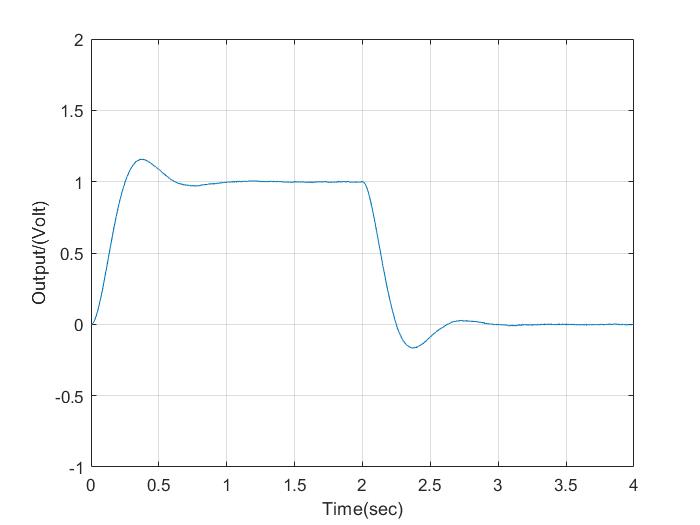
K는 정수로 이루어진 (1xn) 벡터이고 K 값에 따라 state feedback의 성능을 결정한다. K를 구하기 위해선 Ackermann’s formula 를 이용한다.

식으로 구할 수 있고, , 으로 나타낸다.

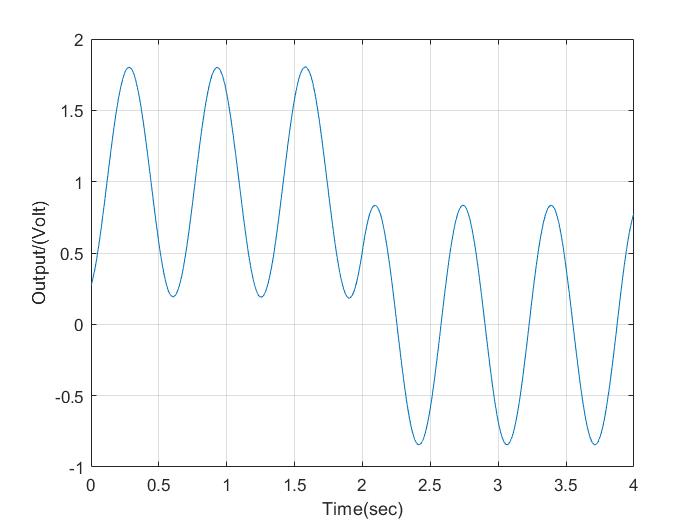
**Chapter 2. 실험 결과(Experimental Results)**

[Exercise 1]

일 때 Kp=1.0, Kd=0인 P제어기를 실행한 결과는 다음과 같다. 약간의 오버슛이 생기지만 1에 수렴한다.



또한, SW1의 1번 스위치를 OFF하여 일 때 응답을 확인해본 결과는 다음과 같았다. 모든 pole이 허수축에 존재하여 시스템이 진동하는 것을 볼 수 있다.



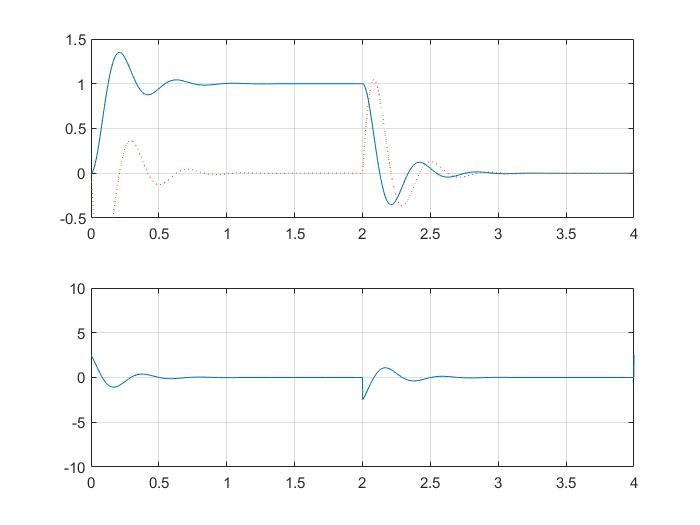
[Exercise 2]

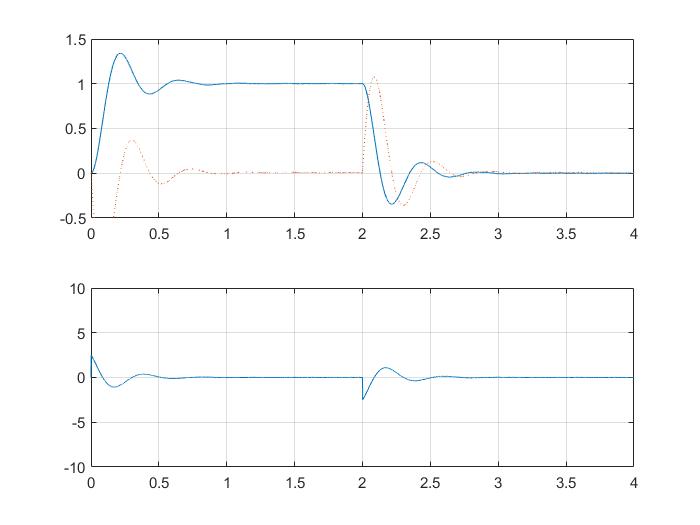
SW1의 1번 스위치를 OFF한 상태에서 full-state feedback controller를 구현하는 실험을 수행하였다. 이고, 다음 두 가지 경우로 pole을 선택하였다.

(1) 허수축에 가깝고, 실수축에선 먼 경우

이 때의 pole은 -5+j\*15, -5-j\*15 로 설정하였고 K1 = 2.5, k2 = -1 이 나왔다.

시뮬레이션 결과는 다음과 같다. Underdamped 상태이고 오버슛이 존재하지만 속도가 빠르다.

  
설계한 제어기를 이용해 실행한 결과를 plot 해 본 결과는 다음과 같다.

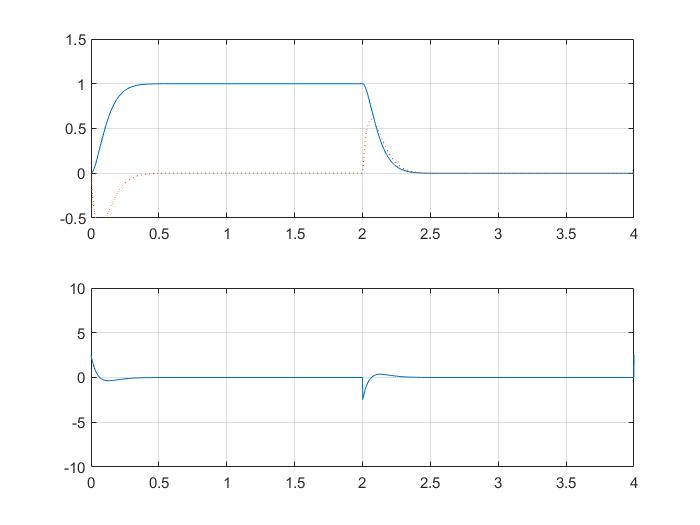


시뮬레이션과 실험결과가 같음을 볼 수 있다.

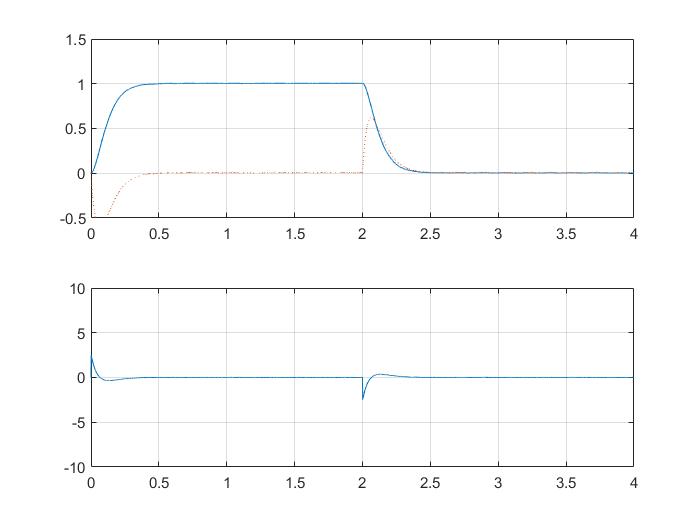
(2) 허수축에서 멀고, 실수축에 가까운 경우

이 때의 pole은 -15+j\*5, -15-j\*5 로 설정하였고 K1 = 2.5, k2 = -3 이 나왔다.

시뮬레이션 결과는 다음과 같다. Overdamped 상황으로 오버슛이 존재하지 않지만 속도가 느리다.



설계한 제어기를 이용해 실행한 결과를 plot 해 본 결과는 다음과 같다.



시뮬레이션과 실험결과가 같음을 볼 수 있다.

**Chapter 3. 결론 및 Discussion**

이번 실습은 특별히 어려운 점은 없었다. 원하는 pole의 위치를 정하면 그 특성에 맞는 따른 결과가 나타났다. 시뮬레이션 결과와 실험 결과를 비교해보았을 때도 일치하였다. Pole이 허수축에 가깝고, 실수축에선 먼 경우엔 damping ratio와 wn이 작아 오버슛이 크고 속도가 느리지만, pole이 허수축에서 멀고, 실수축에 가까운 경우에는 damping raio와 wn이 커서 오버슛이 작거나 없고, 속도가 빠른 것을 볼 수 있었다.

이번 실습은 이전 실습과 달리 원하는 pole의 위치를 먼저 정하고 이에 따라 제어기를 설계할 수 있다는 점에서 가장 달랐다. 이로 인해 원하는 제어 특성을 가지는 제어기를 설계할 수 있다는 장점을 가지는 것 같다. 하지만 실제로는 모든 변수를 피드백 한다는 것이 측정 자체가 불가능할 수 있다는 근본적 문제와 실용적이지 않고 비경제적이라는 문제로 state feedback을 잘 사용하지 않는다고 한다.

**Appendix:**

**K 값에 대해서만 수정이 필요했기 때문에 별도 파일은 첨부하지 않았습니다.**

**아래는 Exercise 2 (1)의 코드입니다.**

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.

\* All rights reserved.</center></h2>

\*

\* This software component is licensed by ST under Ultimate Liberty license

\* SLA0044, the "License"; You may not use this file except in compliance with

\* the License. You may obtain a copy of the License at:

\* www.st.com/SLA0044

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

#include "usb\_host.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

#include "stdio.h"

/\* 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 ---------------------------------------------------------\*/

ADC\_HandleTypeDef hadc1;

ADC\_HandleTypeDef hadc3;

CRC\_HandleTypeDef hcrc;

DAC\_HandleTypeDef hdac;

DMA2D\_HandleTypeDef hdma2d;

I2C\_HandleTypeDef hi2c3;

LTDC\_HandleTypeDef hltdc;

SPI\_HandleTypeDef hspi5;

TIM\_HandleTypeDef htim1;

TIM\_HandleTypeDef htim10;

UART\_HandleTypeDef huart1;

SDRAM\_HandleTypeDef hsdram1;

/\* USER CODE BEGIN PV \*/

float K1,K2,control;

volatile int32\_t x1,x2,ref,interrupt\_counter,sampling\_frequency,data\_counter;

int16\_t data[4000],data2[4000],data3[4000];

volatile uint8\_t data\_flag=0,data\_done=0;

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

static void MX\_CRC\_Init(void);

static void MX\_DMA2D\_Init(void);

static void MX\_FMC\_Init(void);

static void MX\_I2C3\_Init(void);

static void MX\_LTDC\_Init(void);

static void MX\_SPI5\_Init(void);

static void MX\_TIM1\_Init(void);

static void MX\_USART1\_UART\_Init(void);

static void MX\_ADC1\_Init(void);

static void MX\_ADC3\_Init(void);

static void MX\_DAC\_Init(void);

static void MX\_TIM10\_Init(void);

void MX\_USB\_HOST\_Process(void);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

#ifdef \_\_GNUC\_\_

#define PUTCHAR\_PROTOTYPE int \_\_io\_putchar(int ch)

#else

#define PUTCHAR\_PROTOTYPE int fputc(int ch, FILE \*f)

#endif /\* \_\_GNUC\_\_ \*/

PUTCHAR\_PROTOTYPE

{

HAL\_UART\_Transmit(&huart1, (uint8\_t \*)&ch, 1, 0xFFFF);

return ch;

}

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

int main(void)

{

/\* USER CODE BEGIN 1 \*/

K1=2.5;K2=-1.0;

sampling\_frequency=1000;

/\* 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\_CRC\_Init();

MX\_DMA2D\_Init();

MX\_FMC\_Init();

MX\_I2C3\_Init();

MX\_LTDC\_Init();

MX\_SPI5\_Init();

MX\_TIM1\_Init();

MX\_USART1\_UART\_Init();

MX\_USB\_HOST\_Init();

MX\_ADC1\_Init();

MX\_ADC3\_Init();

MX\_DAC\_Init();

MX\_TIM10\_Init();

/\* USER CODE BEGIN 2 \*/

HAL\_TIM\_Base\_Start\_IT(&htim10);

HAL\_DAC\_Start(&hdac, DAC\_CHANNEL\_2);

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

if(data\_done == 1) {

for (int i=0; i < sampling\_frequency\*4 ;i++){

printf("%d %d %d %d\r\n",i,data[i],data2[i],data3[i]);

}

data\_flag=0;

data\_done=0;

HAL\_GPIO\_TogglePin(GPIOG, GPIO\_PIN\_14);

}

/\* USER CODE END WHILE \*/

MX\_USB\_HOST\_Process();

/\* 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\_SCALE1);

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;

RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;

RCC\_OscInitStruct.PLL.PLLM = 4;

RCC\_OscInitStruct.PLL.PLLN = 168;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2;

RCC\_OscInitStruct.PLL.PLLQ = 7;

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\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV4;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV2;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_5) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief ADC1 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_ADC1\_Init(void)

{

/\* USER CODE BEGIN ADC1\_Init 0 \*/

/\* USER CODE END ADC1\_Init 0 \*/

ADC\_ChannelConfTypeDef sConfig = {0};

/\* USER CODE BEGIN ADC1\_Init 1 \*/

/\* USER CODE END ADC1\_Init 1 \*/

/\*\* Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of conversion)

\*/

hadc1.Instance = ADC1;

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

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

hadc1.Init.ScanConvMode = 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;

if (HAL\_ADC\_Init(&hadc1) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.

\*/

sConfig.Channel = ADC\_CHANNEL\_13;

sConfig.Rank = 1;

sConfig.SamplingTime = ADC\_SAMPLETIME\_3CYCLES;

if (HAL\_ADC\_ConfigChannel(&hadc1, &sConfig) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN ADC1\_Init 2 \*/

/\* USER CODE END ADC1\_Init 2 \*/

}

/\*\*

\* @brief ADC3 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_ADC3\_Init(void)

{

/\* USER CODE BEGIN ADC3\_Init 0 \*/

/\* USER CODE END ADC3\_Init 0 \*/

ADC\_ChannelConfTypeDef sConfig = {0};

/\* USER CODE BEGIN ADC3\_Init 1 \*/

/\* USER CODE END ADC3\_Init 1 \*/

/\*\* Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of conversion)

\*/

hadc3.Instance = ADC3;

hadc3.Init.ClockPrescaler = ADC\_CLOCK\_SYNC\_PCLK\_DIV4;

hadc3.Init.Resolution = ADC\_RESOLUTION\_12B;

hadc3.Init.ScanConvMode = DISABLE;

hadc3.Init.ContinuousConvMode = DISABLE;

hadc3.Init.DiscontinuousConvMode = DISABLE;

hadc3.Init.ExternalTrigConvEdge = ADC\_EXTERNALTRIGCONVEDGE\_NONE;

hadc3.Init.ExternalTrigConv = ADC\_SOFTWARE\_START;

hadc3.Init.DataAlign = ADC\_DATAALIGN\_RIGHT;

hadc3.Init.NbrOfConversion = 1;

hadc3.Init.DMAContinuousRequests = DISABLE;

hadc3.Init.EOCSelection = ADC\_EOC\_SINGLE\_CONV;

if (HAL\_ADC\_Init(&hadc3) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.

\*/

sConfig.Channel = ADC\_CHANNEL\_4;

sConfig.Rank = 1;

sConfig.SamplingTime = ADC\_SAMPLETIME\_3CYCLES;

if (HAL\_ADC\_ConfigChannel(&hadc3, &sConfig) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN ADC3\_Init 2 \*/

/\* USER CODE END ADC3\_Init 2 \*/

}

/\*\*

\* @brief CRC Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_CRC\_Init(void)

{

/\* USER CODE BEGIN CRC\_Init 0 \*/

/\* USER CODE END CRC\_Init 0 \*/

/\* USER CODE BEGIN CRC\_Init 1 \*/

/\* USER CODE END CRC\_Init 1 \*/

hcrc.Instance = CRC;

if (HAL\_CRC\_Init(&hcrc) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN CRC\_Init 2 \*/

/\* USER CODE END CRC\_Init 2 \*/

}

/\*\*

\* @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 OUT2 config

\*/

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

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

if (HAL\_DAC\_ConfigChannel(&hdac, &sConfig, DAC\_CHANNEL\_2) != 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 I2C3 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_I2C3\_Init(void)

{

/\* USER CODE BEGIN I2C3\_Init 0 \*/

/\* USER CODE END I2C3\_Init 0 \*/

/\* USER CODE BEGIN I2C3\_Init 1 \*/

/\* USER CODE END I2C3\_Init 1 \*/

hi2c3.Instance = I2C3;

hi2c3.Init.ClockSpeed = 100000;

hi2c3.Init.DutyCycle = I2C\_DUTYCYCLE\_2;

hi2c3.Init.OwnAddress1 = 0;

hi2c3.Init.AddressingMode = I2C\_ADDRESSINGMODE\_7BIT;

hi2c3.Init.DualAddressMode = I2C\_DUALADDRESS\_DISABLE;

hi2c3.Init.OwnAddress2 = 0;

hi2c3.Init.GeneralCallMode = I2C\_GENERALCALL\_DISABLE;

hi2c3.Init.NoStretchMode = I2C\_NOSTRETCH\_DISABLE;

if (HAL\_I2C\_Init(&hi2c3) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Configure Analogue filter

\*/

if (HAL\_I2CEx\_ConfigAnalogFilter(&hi2c3, I2C\_ANALOGFILTER\_ENABLE) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Configure Digital filter

\*/

if (HAL\_I2CEx\_ConfigDigitalFilter(&hi2c3, 0) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN I2C3\_Init 2 \*/

/\* USER CODE END I2C3\_Init 2 \*/

}

/\*\*

\* @brief LTDC Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_LTDC\_Init(void)

{

/\* USER CODE BEGIN LTDC\_Init 0 \*/

/\* USER CODE END LTDC\_Init 0 \*/

LTDC\_LayerCfgTypeDef pLayerCfg = {0};

/\* USER CODE BEGIN LTDC\_Init 1 \*/

/\* USER CODE END LTDC\_Init 1 \*/

hltdc.Instance = LTDC;

hltdc.Init.HSPolarity = LTDC\_HSPOLARITY\_AL;

hltdc.Init.VSPolarity = LTDC\_VSPOLARITY\_AL;

hltdc.Init.DEPolarity = LTDC\_DEPOLARITY\_AL;

hltdc.Init.PCPolarity = LTDC\_PCPOLARITY\_IPC;

hltdc.Init.HorizontalSync = 9;

hltdc.Init.VerticalSync = 1;

hltdc.Init.AccumulatedHBP = 29;

hltdc.Init.AccumulatedVBP = 3;

hltdc.Init.AccumulatedActiveW = 269;

hltdc.Init.AccumulatedActiveH = 323;

hltdc.Init.TotalWidth = 279;

hltdc.Init.TotalHeigh = 327;

hltdc.Init.Backcolor.Blue = 0;

hltdc.Init.Backcolor.Green = 0;

hltdc.Init.Backcolor.Red = 0;

if (HAL\_LTDC\_Init(&hltdc) != HAL\_OK)

{

Error\_Handler();

}

pLayerCfg.WindowX0 = 0;

pLayerCfg.WindowX1 = 240;

pLayerCfg.WindowY0 = 0;

pLayerCfg.WindowY1 = 320;

pLayerCfg.PixelFormat = LTDC\_PIXEL\_FORMAT\_RGB565;

pLayerCfg.Alpha = 255;

pLayerCfg.Alpha0 = 0;

pLayerCfg.BlendingFactor1 = LTDC\_BLENDING\_FACTOR1\_PAxCA;

pLayerCfg.BlendingFactor2 = LTDC\_BLENDING\_FACTOR2\_PAxCA;

pLayerCfg.FBStartAdress = 0xD0000000;

pLayerCfg.ImageWidth = 240;

pLayerCfg.ImageHeight = 320;

pLayerCfg.Backcolor.Blue = 0;

pLayerCfg.Backcolor.Green = 0;

pLayerCfg.Backcolor.Red = 0;

if (HAL\_LTDC\_ConfigLayer(&hltdc, &pLayerCfg, 0) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN LTDC\_Init 2 \*/

/\* USER CODE END LTDC\_Init 2 \*/

}

/\*\*

\* @brief SPI5 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_SPI5\_Init(void)

{

/\* USER CODE BEGIN SPI5\_Init 0 \*/

/\* USER CODE END SPI5\_Init 0 \*/

/\* USER CODE BEGIN SPI5\_Init 1 \*/

/\* USER CODE END SPI5\_Init 1 \*/

/\* SPI5 parameter configuration\*/

hspi5.Instance = SPI5;

hspi5.Init.Mode = SPI\_MODE\_MASTER;

hspi5.Init.Direction = SPI\_DIRECTION\_2LINES;

hspi5.Init.DataSize = SPI\_DATASIZE\_8BIT;

hspi5.Init.CLKPolarity = SPI\_POLARITY\_LOW;

hspi5.Init.CLKPhase = SPI\_PHASE\_1EDGE;

hspi5.Init.NSS = SPI\_NSS\_SOFT;

hspi5.Init.BaudRatePrescaler = SPI\_BAUDRATEPRESCALER\_16;

hspi5.Init.FirstBit = SPI\_FIRSTBIT\_MSB;

hspi5.Init.TIMode = SPI\_TIMODE\_DISABLE;

hspi5.Init.CRCCalculation = SPI\_CRCCALCULATION\_DISABLE;

hspi5.Init.CRCPolynomial = 10;

if (HAL\_SPI\_Init(&hspi5) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN SPI5\_Init 2 \*/

/\* USER CODE END SPI5\_Init 2 \*/

}

/\*\*

\* @brief TIM1 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_TIM1\_Init(void)

{

/\* USER CODE BEGIN TIM1\_Init 0 \*/

/\* USER CODE END TIM1\_Init 0 \*/

TIM\_ClockConfigTypeDef sClockSourceConfig = {0};

TIM\_MasterConfigTypeDef sMasterConfig = {0};

/\* USER CODE BEGIN TIM1\_Init 1 \*/

/\* USER CODE END TIM1\_Init 1 \*/

htim1.Instance = TIM1;

htim1.Init.Prescaler = 0;

htim1.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim1.Init.Period = 65535;

htim1.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim1.Init.RepetitionCounter = 0;

htim1.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

if (HAL\_TIM\_Base\_Init(&htim1) != HAL\_OK)

{

Error\_Handler();

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

if (HAL\_TIM\_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL\_OK)

{

Error\_Handler();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

if (HAL\_TIMEx\_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM1\_Init 2 \*/

/\* USER CODE END TIM1\_Init 2 \*/

}

/\*\*

\* @brief TIM10 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_TIM10\_Init(void)

{

/\* USER CODE BEGIN TIM10\_Init 0 \*/

/\* USER CODE END TIM10\_Init 0 \*/

/\* USER CODE BEGIN TIM10\_Init 1 \*/

/\* USER CODE END TIM10\_Init 1 \*/

htim10.Instance = TIM10;

htim10.Init.Prescaler = 839;

htim10.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim10.Init.Period = 199;

htim10.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim10.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

if (HAL\_TIM\_Base\_Init(&htim10) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM10\_Init 2 \*/

/\* USER CODE END TIM10\_Init 2 \*/

}

/\*\*

\* @brief USART1 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_USART1\_UART\_Init(void)

{

/\* USER CODE BEGIN USART1\_Init 0 \*/

/\* USER CODE END USART1\_Init 0 \*/

/\* USER CODE BEGIN USART1\_Init 1 \*/

/\* USER CODE END USART1\_Init 1 \*/

huart1.Instance = USART1;

huart1.Init.BaudRate = 115200;

huart1.Init.WordLength = UART\_WORDLENGTH\_8B;

huart1.Init.StopBits = UART\_STOPBITS\_1;

huart1.Init.Parity = UART\_PARITY\_NONE;

huart1.Init.Mode = UART\_MODE\_TX\_RX;

huart1.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

huart1.Init.OverSampling = UART\_OVERSAMPLING\_16;

if (HAL\_UART\_Init(&huart1) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN USART1\_Init 2 \*/

/\* USER CODE END USART1\_Init 2 \*/

}

/\* FMC initialization function \*/

static void MX\_FMC\_Init(void)

{

/\* USER CODE BEGIN FMC\_Init 0 \*/

/\* USER CODE END FMC\_Init 0 \*/

FMC\_SDRAM\_TimingTypeDef SdramTiming = {0};

/\* USER CODE BEGIN FMC\_Init 1 \*/

/\* USER CODE END FMC\_Init 1 \*/

/\*\* Perform the SDRAM1 memory initialization sequence

\*/

hsdram1.Instance = FMC\_SDRAM\_DEVICE;

/\* hsdram1.Init \*/

hsdram1.Init.SDBank = FMC\_SDRAM\_BANK2;

hsdram1.Init.ColumnBitsNumber = FMC\_SDRAM\_COLUMN\_BITS\_NUM\_8;

hsdram1.Init.RowBitsNumber = FMC\_SDRAM\_ROW\_BITS\_NUM\_12;

hsdram1.Init.MemoryDataWidth = FMC\_SDRAM\_MEM\_BUS\_WIDTH\_16;

hsdram1.Init.InternalBankNumber = FMC\_SDRAM\_INTERN\_BANKS\_NUM\_4;

hsdram1.Init.CASLatency = FMC\_SDRAM\_CAS\_LATENCY\_3;

hsdram1.Init.WriteProtection = FMC\_SDRAM\_WRITE\_PROTECTION\_DISABLE;

hsdram1.Init.SDClockPeriod = FMC\_SDRAM\_CLOCK\_PERIOD\_2;

hsdram1.Init.ReadBurst = FMC\_SDRAM\_RBURST\_DISABLE;

hsdram1.Init.ReadPipeDelay = FMC\_SDRAM\_RPIPE\_DELAY\_1;

/\* SdramTiming \*/

SdramTiming.LoadToActiveDelay = 2;

SdramTiming.ExitSelfRefreshDelay = 7;

SdramTiming.SelfRefreshTime = 4;

SdramTiming.RowCycleDelay = 7;

SdramTiming.WriteRecoveryTime = 3;

SdramTiming.RPDelay = 2;

SdramTiming.RCDDelay = 2;

if (HAL\_SDRAM\_Init(&hsdram1, &SdramTiming) != HAL\_OK)

{

Error\_Handler( );

}

/\* USER CODE BEGIN FMC\_Init 2 \*/

/\* USER CODE END FMC\_Init 2 \*/

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* GPIO Ports Clock Enable \*/

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

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

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

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

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

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

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

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

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOC, NCS\_MEMS\_SPI\_Pin|CSX\_Pin|OTG\_FS\_PSO\_Pin, GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(ACP\_RST\_GPIO\_Port, ACP\_RST\_Pin, GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOD, RDX\_Pin|WRX\_DCX\_Pin, GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOG, LD3\_Pin|LD4\_Pin, GPIO\_PIN\_RESET);

/\*Configure GPIO pins : NCS\_MEMS\_SPI\_Pin CSX\_Pin OTG\_FS\_PSO\_Pin \*/

GPIO\_InitStruct.Pin = NCS\_MEMS\_SPI\_Pin|CSX\_Pin|OTG\_FS\_PSO\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

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

/\*Configure GPIO pin : B1\_Pin \*/

GPIO\_InitStruct.Pin = B1\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(B1\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pins : MEMS\_INT1\_Pin MEMS\_INT2\_Pin TP\_INT1\_Pin \*/

GPIO\_InitStruct.Pin = MEMS\_INT1\_Pin|MEMS\_INT2\_Pin|TP\_INT1\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_EVT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

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

/\*Configure GPIO pin : ACP\_RST\_Pin \*/

GPIO\_InitStruct.Pin = ACP\_RST\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(ACP\_RST\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : OTG\_FS\_OC\_Pin \*/

GPIO\_InitStruct.Pin = OTG\_FS\_OC\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_EVT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(OTG\_FS\_OC\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : BOOT1\_Pin \*/

GPIO\_InitStruct.Pin = BOOT1\_Pin;

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

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(BOOT1\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : TE\_Pin \*/

GPIO\_InitStruct.Pin = TE\_Pin;

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

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(TE\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pins : RDX\_Pin WRX\_DCX\_Pin \*/

GPIO\_InitStruct.Pin = RDX\_Pin|WRX\_DCX\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

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

/\*Configure GPIO pins : LD3\_Pin LD4\_Pin \*/

GPIO\_InitStruct.Pin = LD3\_Pin|LD4\_Pin;

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);

/\* EXTI interrupt init\*/

HAL\_NVIC\_SetPriority(EXTI0\_IRQn, 0, 0);

HAL\_NVIC\_EnableIRQ(EXTI0\_IRQn);

}

/\* USER CODE BEGIN 4 \*/

void HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin)

{

HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_14, GPIO\_PIN\_SET);

data\_flag=1;

}

/\* USER CODE END 4 \*/

/\*\*

\* @brief Period elapsed callback in non blocking mode

\* @note This function is called when TIM6 interrupt took place, inside

\* HAL\_TIM\_IRQHandler(). It makes a direct call to HAL\_IncTick() to increment

\* a global variable "uwTick" used as application time base.

\* @param htim : TIM handle

\* @retval None

\*/

void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim)

{

/\* USER CODE BEGIN Callback 0 \*/

int32\_t da\_value,ad\_value,sum;

if (htim->Instance == TIM10) {

sum=0;

for (int i=0; i<20 ; i++) {

HAL\_ADC\_Start(&hadc1);

if (HAL\_ADC\_PollForConversion(&hadc1, 10000) == HAL\_OK) {

ad\_value = HAL\_ADC\_GetValue(&hadc1);

sum += ad\_value;

}

}

x1 = sum/20 - 2048;

sum=0;

for (int i=0; i<20 ; i++) {

HAL\_ADC\_Start(&hadc3);

if (HAL\_ADC\_PollForConversion(&hadc3, 10000) == HAL\_OK) {

ad\_value = HAL\_ADC\_GetValue(&hadc3);

sum += ad\_value;

}

}

x2 = sum/20 - 2048;

interrupt\_counter++;

if (interrupt\_counter >= sampling\_frequency\*4) {

interrupt\_counter=0;

if (data\_flag==1) {

data\_counter=0;

data\_flag=2;

}

ref=205;

}

if (interrupt\_counter >= sampling\_frequency\*2) {

ref=0;

}

if (data\_flag==2) {

if (data\_counter<sampling\_frequency\*4) {

data[data\_counter]=(int16\_t)x1;

data2[data\_counter]=(int16\_t)x2;

data3[data\_counter]=(int16\_t)control;

data\_counter++;

}

else {

data\_done=1;

}

}

control = K1\*(float)(ref-x1)-K2\*(float)x2;

if (control > 2047) control = 2047;

if (control < -2048) control = -2048;

da\_value = control + 2048;

HAL\_DAC\_SetValue(&hdac, DAC\_CHANNEL\_2, DAC\_ALIGN\_12B\_R, (uint32\_t)(da\_value));

}

/\* USER CODE END Callback 0 \*/

if (htim->Instance == TIM6) {

HAL\_IncTick();

}

/\* USER CODE BEGIN Callback 1 \*/

/\* USER CODE END Callback 1 \*/

}

/\*\*

\* @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 \*/

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