main.c

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

**#include** "main.h"

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

/\* USER CODE BEGIN Includes \*/

/\* 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;

DMA\_HandleTypeDef hdma\_adc1;

/\* 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\_ADC1\_Init**(**void**);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

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

/\* USER CODE BEGIN 0 \*/

uint32\_t adc\_deger[3];

/\* 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\_ADC1\_Init();

/\* USER CODE BEGIN 2 \*/

HAL\_ADC\_Start\_DMA(&hadc1,adc\_deger,3);

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**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 = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

RCC\_PeriphCLKInitTypeDef PeriphClkInit = {0};

/\*\* 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.HSEPredivValue = RCC\_HSE\_PREDIV\_DIV1;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

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

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

RCC\_OscInitStruct.PLL.PLLMUL = RCC\_PLL\_MUL9;

**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\_DIV2;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

**if** (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_2) != *HAL\_OK*)

{

Error\_Handler();

}

PeriphClkInit.PeriphClockSelection = RCC\_PERIPHCLK\_ADC;

PeriphClkInit.AdcClockSelection = RCC\_ADCPCLK2\_DIV6;

**if** (HAL\_RCCEx\_PeriphCLKConfig(&PeriphClkInit) != *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 \*/

/\*\* Common config

\*/

hadc1.Instance = ADC1;

hadc1.Init.ScanConvMode = ADC\_SCAN\_ENABLE;

hadc1.Init.ContinuousConvMode = *ENABLE*;

hadc1.Init.DiscontinuousConvMode = *DISABLE*;

hadc1.Init.ExternalTrigConv = ADC\_SOFTWARE\_START;

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

hadc1.Init.NbrOfConversion = 3;

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

{

Error\_Handler();

}

/\*\* Configure Regular Channel

\*/

sConfig.Channel = ADC\_CHANNEL\_0;

sConfig.Rank = ADC\_REGULAR\_RANK\_1;

sConfig.SamplingTime = ADC\_SAMPLETIME\_1CYCLE\_5;

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

{

Error\_Handler();

}

/\*\* Configure Regular Channel

\*/

sConfig.Channel = ADC\_CHANNEL\_1;

sConfig.Rank = ADC\_REGULAR\_RANK\_2;

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

{

Error\_Handler();

}

/\*\* Configure Regular Channel

\*/

sConfig.Channel = ADC\_CHANNEL\_2;

sConfig.Rank = ADC\_REGULAR\_RANK\_3;

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

{

Error\_Handler();

}

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

/\* USER CODE END ADC1\_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\_Channel1\_IRQn interrupt configuration \*/

HAL\_NVIC\_SetPriority(*DMA1\_Channel1\_IRQn*, 0, 0);

HAL\_NVIC\_EnableIRQ(*DMA1\_Channel1\_IRQn*);

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_GPIO\_Init**(**void**)

{

/\* GPIO Ports Clock Enable \*/

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

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

}

/\* 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 \*/

}