/\* USER CODE BEGIN Header \*/

/\*\*

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

\* @file : main.c

\* @brief : Main program body

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

\* @attention

\*

\* Copyright (c) 2022 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"

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

/\* USER CODE BEGIN Includes \*/

uint32\_t butonkontrol = 0;

uint32\_t kontrol = 0;

uint32\_t kontrol1 = 0;

uint32\_t sayac = 0;

uint32\_t saniye = 0;

uint32\_t dakika=0;

uint32\_t i = 0;

**int** dizi[10]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7C,0x07,0x7F,0x6F};

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

TIM\_HandleTypeDef htim4;

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

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

**void** **SystemClock\_Config**(**void**);

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

**static** **void** **MX\_TIM4\_Init**(**void**);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

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

/\* USER CODE BEGIN 0 \*/

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

{

**if** (butonkontrol == 1)

{

++sayac;

**if** (sayac%1000==0)

{

sayac = 0;

++saniye;

**if**(saniye%60==0)

{

++dakika;saniye=0;

}

}

}

**else** **if**(butonkontrol == 0)

{

saniye=0;dakika=0;

}

**if** (kontrol1 == 1)

++kontrol;

// ++sayac;

// if(sayac%1000==0)

// {

// ++saniye;

// if(saniye%60==0)

// {

// ++dakika;

// saniye=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\_TIM4\_Init();

/\* USER CODE BEGIN 2 \*/

HAL\_TIM\_Base\_Start\_IT(&htim4);

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

**if** (butonkontrol == 0) // Kronometreyi baslatma

{

**if** (HAL\_GPIO\_ReadPin(GPIOA, GPIO\_PIN\_0))

{

kontrol1 = 1;

**if** (kontrol >= 400)

{

**if** (HAL\_GPIO\_ReadPin(GPIOA, GPIO\_PIN\_0))

{

butonkontrol = 1;

kontrol = 0;

kontrol1 = 0;

} **else**

{

butonkontrol = 0;

kontrol = 0;

kontrol1 = 0;

}

}

}

}

**else** **if** (butonkontrol == 1) // Kronometriyi durdurma

{

**if** (HAL\_GPIO\_ReadPin(GPIOA, GPIO\_PIN\_0))

{

kontrol1 = 1;

**if** (kontrol >= 400) {

**if** (HAL\_GPIO\_ReadPin(GPIOA, GPIO\_PIN\_0))

{

butonkontrol = 2;

kontrol = 0;

kontrol1 = 0;

}

**else**

{

butonkontrol = 1;

kontrol = 0;

kontrol1 = 0;

}

}

}

}

**else** **if** (butonkontrol == 2) // Kronometriyi sıfırlama

{

**if** (HAL\_GPIO\_ReadPin(GPIOA, GPIO\_PIN\_0)) {

kontrol1 = 1;

**if** (kontrol >= 400) {

**if** (HAL\_GPIO\_ReadPin(GPIOA, GPIO\_PIN\_0)) {

butonkontrol = 0;

kontrol = 0;

kontrol1 = 0;

sayac = 0;

} **else**

{

butonkontrol = 2;

kontrol = 0;

kontrol1 = 0;

}

}

}

// HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_13, 0);

// HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_14, 0);

// HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_12, 0);

// HAL\_Gecikme(1000);

// HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_15, 1);

// HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_13, 1);

// HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_14, 1);

// HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_12, 1);

// HAL\_Gecikme(1000);

/\* KULLANICI KODU END 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 = 4;

**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 TIM4 Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_TIM4\_Init**(**void**)

{

/\* USER CODE BEGIN TIM4\_Init 0 \*/

/\* USER CODE END TIM4\_Init 0 \*/

TIM\_ClockConfigTypeDef sClockSourceConfig = {0};

TIM\_MasterConfigTypeDef sMasterConfig = {0};

/\* USER CODE BEGIN TIM4\_Init 1 \*/

/\* USER CODE END TIM4\_Init 1 \*/

htim4.Instance = TIM4;

htim4.Init.Prescaler = 84-1;

htim4.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim4.Init.Period = 1000-1;

htim4.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim4.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

**if** (HAL\_TIM\_Base\_Init(&htim4) != *HAL\_OK*)

{

Error\_Handler();

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

**if** (HAL\_TIM\_ConfigClockSource(&htim4, &sClockSourceConfig) != *HAL\_OK*)

{

Error\_Handler();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

**if** (HAL\_TIMEx\_MasterConfigSynchronization(&htim4, &sMasterConfig) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM4\_Init 2 \*/

/\* USER CODE END TIM4\_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\_GPIOH\_CLK\_ENABLE();

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

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

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOD, LED1\_Pin|LED2\_Pin|LED3\_Pin|LED4\_Pin, *GPIO\_PIN\_RESET*);

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

GPIO\_InitStruct.Pin = BUTON\_Pin;

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

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

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

/\*Configure GPIO pins : LED1\_Pin LED2\_Pin LED3\_Pin LED4\_Pin \*/

GPIO\_InitStruct.Pin = LED1\_Pin|LED2\_Pin|LED3\_Pin|LED4\_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);

}

/\* 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 \*/