/\* USER CODE BEGIN Header \*/

/\*\*

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

\* @file : main.c

\* @brief : Main program body

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

\* @attention

\*

\* Copyright (c) 2024 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 \*/

**#include** "CO\_app\_STM32.h"

**#include** "../CANopenNode\_STM32/OD.h"

**#include** "../CANopenNode/301/CO\_driver.h"

**#include** "../CANopenNode/301/CO\_SDOserver.h"

**#include** "../CANopenNode/301/CO\_SDOclient.h"

**#include** "../CANopenNode/301/CO\_NMT\_Heartbeat.h"

**#include** "../CANopenNode/301/CO\_PDO.h"

**#include** "../CANopenNode/301/CO\_HBconsumer.h"

**#include** "../CANopenNode/CANopen.h"

**#include** "string.h"

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

CAN\_HandleTypeDef hcan1;

TIM\_HandleTypeDef htim14;

UART\_HandleTypeDef huart2;

/\* USER CODE BEGIN PV \*/

**extern** CO\_t \*CO; // Объявляем переменную как extern

CO\_HBconsumer\_t HBcons;

CO\_HBconsNode\_t monitoredNodes[1]; // Массив узлов, за которыми будем следить

CO\_EM\_t em;

OD\_entry\_t OD\_1016\_HBcons;

CO\_CANmodule\_t CANmodule; // Определяем переменную CANmodule

uint8\_t nodeId = 66; // Node ID нашего мастера

uint8\_t monitoredNodeId = 16; // Node ID нашего слейва (motor ID)

CO\_CANrx\_t CAN\_rxArray[32]; // Массив для обработки принятых сообщений

CO\_CANtx\_t CAN\_txArray[32]; // Массив для обработки отправляемых сообщений

/\* USER CODE END PV \*/

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

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

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

**static** **void** **MX\_CAN1\_Init**(**void**);

**static** **void** **MX\_TIM14\_Init**(**void**);

**static** **void** **MX\_USART2\_UART\_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** (htim == &htim14) {

canopen\_app\_interrupt();

}

}

**void** **disable\_CAN\_transmission**(CO\_CANmodule\_t \*CANmodule) {

CANmodule->CANerrorStatus |= CO\_CAN\_ERRTX\_BUS\_OFF;

// Дополнительно можно вызвать функцию, которая обработает это состояние

CO\_CANmodule\_disable(CANmodule);

}

**void** **checkHeartbeat**(CO\_HBconsumer\_t \*HBcons) {

**for** (uint8\_t i = 0; i < HBcons->numberOfMonitoredNodes; i++) {

**char** msg[100];

**snprintf**(msg, **sizeof**(msg), "Node %d NMT state: %d\r\n", i, HBcons->monitoredNodes[i].NMTstate);

HAL\_UART\_Transmit(&huart2, (uint8\_t\*)msg, **strlen**(msg), HAL\_MAX\_DELAY);

**if** (HBcons->monitoredNodes[i].NMTstate == *CO\_NMT\_OPERATIONAL*) {

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_9, *GPIO\_PIN\_SET*); // Включаем светодиод LED2

} **else** {

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_9, *GPIO\_PIN\_RESET*); // Выключаем светодиод LED2

}

}

}

**void** **read\_OD\_and\_send\_uart**(OD\_t \*od, uint16\_t index, uint8\_t subindex) {

OD\_entry\_t \*entry;

OD\_IO\_t io;

**char** msg[100];

OD\_size\_t bytesRd;

ODR\_t odRet;

/\* Find the OD entry \*/

entry = OD\_find(od, index);

**if** (entry == NULL) {

**snprintf**(msg, **sizeof**(msg), "OD entry 0x%X not found\r\n", index);

HAL\_UART\_Transmit(&huart2, (uint8\_t\*)msg, **strlen**(msg), HAL\_MAX\_DELAY);

**return**;

}

/\* Get the subindex \*/

odRet = OD\_getSub(entry, subindex, &io, false);

**if** (odRet != *ODR\_OK*) {

**snprintf**(msg, **sizeof**(msg), "Subindex 0x%X not found in 0x%X\r\n", subindex, index);

HAL\_UART\_Transmit(&huart2, (uint8\_t\*)msg, **strlen**(msg), HAL\_MAX\_DELAY);

**return**;

}

/\* Read the value \*/

CO\_LOCK\_OD(&CANmodule);

odRet = io.read(&io.stream, &msg[0], **sizeof**(msg), &bytesRd);

CO\_UNLOCK\_OD(&CANmodule);

**if** (odRet == *ODR\_OK*) {

HAL\_UART\_Transmit(&huart2, (uint8\_t\*)msg, bytesRd, HAL\_MAX\_DELAY);

} **else** {

**snprintf**(msg, **sizeof**(msg), "Error reading OD 0x%X:%d\r\n", index, subindex);

HAL\_UART\_Transmit(&huart2, (uint8\_t\*)msg, **strlen**(msg), HAL\_MAX\_DELAY);

}

}

/\* 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\_CAN1\_Init();

MX\_TIM14\_Init();

MX\_USART2\_UART\_Init();

/\* USER CODE BEGIN 2 \*/

HAL\_UART\_Transmit(&huart2, (uint8\_t\*)"Starting initialization\r\n", **strlen**("Starting initialization\r\n"), HAL\_MAX\_DELAY);

CANopenNodeSTM32 canOpenNodeSTM32;

canOpenNodeSTM32.CANHandle = &hcan1;

canOpenNodeSTM32.HWInitFunction = MX\_CAN1\_Init;

canOpenNodeSTM32.timerHandle = &htim14;

canOpenNodeSTM32.desiredNodeID = nodeId;

canOpenNodeSTM32.baudrate = 500;

canopen\_app\_init(&canOpenNodeSTM32);

HAL\_UART\_Transmit(&huart2, (uint8\_t\*)"CAN module initialization\r\n", **strlen**("CAN module initialization\r\n"), HAL\_MAX\_DELAY);

// Инициализация CAN модуля

CO\_ReturnError\_t err = CO\_CANmodule\_init(&CANmodule, &hcan1, CAN\_rxArray, **sizeof**(CAN\_rxArray)/**sizeof**(CAN\_rxArray[0]),

CAN\_txArray, **sizeof**(CAN\_txArray)/**sizeof**(CAN\_txArray[0]), 500);

**if** (err != *CO\_ERROR\_NO*) {

HAL\_UART\_Transmit(&huart2, (uint8\_t\*)"Error initializing CAN module\r\n", **strlen**("Error initializing CAN module\r\n"), HAL\_MAX\_DELAY);

**while** (1); // Обработка ошибки

}

HAL\_UART\_Transmit(&huart2, (uint8\_t\*)"Heartbeat consumer initialization\r\n", **strlen**("Heartbeat consumer initialization\r\n"), HAL\_MAX\_DELAY);

// Инициализация Heartbeat Consumer

err = CO\_HBconsumer\_init(&HBcons, &em, monitoredNodes, 1, &OD->list[11], &CANmodule, 0, NULL);

**if** (err != *CO\_ERROR\_NO*) {

HAL\_UART\_Transmit(&huart2, (uint8\_t\*)"Error initializing Heartbeat consumer\r\n", **strlen**("Error initializing Heartbeat consumer\r\n"), HAL\_MAX\_DELAY);

**while** (1); // Обработка ошибки

}

HAL\_UART\_Transmit(&huart2, (uint8\_t\*)"Initialization complete\r\n", **strlen**("Initialization complete\r\n"), HAL\_MAX\_DELAY);

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

canopen\_app\_process();

disable\_CAN\_transmission(&CANmodule);

uint32\_t timeDifference\_us = 2000; // Пример, нужно обновлять в зависимости от текущего времени

uint32\_t timerNext\_us;

// Обработка Heartbeat Consumer

CO\_HBconsumer\_process(&HBcons, true, timeDifference\_us, &timerNext\_us);

// Проверка состояния Heartbeat и управление светодиодом

checkHeartbeat(&HBcons);

// Пример чтения значения из словаря объектов и отправки через UART

read\_OD\_and\_send\_uart(OD, 0x1008, 0x00); // Чтение "Manufacturer Device Name"

/\* 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};

/\*\* 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\_HSI;

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

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

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

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

RCC\_OscInitStruct.PLL.PLLM = 8;

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 CAN1 Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_CAN1\_Init**(**void**)

{

/\* USER CODE BEGIN CAN1\_Init 0 \*/

/\* USER CODE END CAN1\_Init 0 \*/

/\* USER CODE BEGIN CAN1\_Init 1 \*/

/\* USER CODE END CAN1\_Init 1 \*/

hcan1.Instance = CAN1;

hcan1.Init.Prescaler = 21;

hcan1.Init.Mode = CAN\_MODE\_NORMAL;

hcan1.Init.SyncJumpWidth = CAN\_SJW\_1TQ;

hcan1.Init.TimeSeg1 = CAN\_BS1\_2TQ;

hcan1.Init.TimeSeg2 = CAN\_BS2\_1TQ;

hcan1.Init.TimeTriggeredMode = *DISABLE*;

hcan1.Init.AutoBusOff = *DISABLE*;

hcan1.Init.AutoWakeUp = *DISABLE*;

hcan1.Init.AutoRetransmission = *DISABLE*;

hcan1.Init.ReceiveFifoLocked = *DISABLE*;

hcan1.Init.TransmitFifoPriority = *DISABLE*;

**if** (HAL\_CAN\_Init(&hcan1) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN CAN1\_Init 2 \*/

/\* USER CODE END CAN1\_Init 2 \*/

}

/\*\*

\* @brief TIM14 Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_TIM14\_Init**(**void**)

{

/\* USER CODE BEGIN TIM14\_Init 0 \*/

/\* USER CODE END TIM14\_Init 0 \*/

/\* USER CODE BEGIN TIM14\_Init 1 \*/

/\* USER CODE END TIM14\_Init 1 \*/

htim14.Instance = TIM14;

htim14.Init.Prescaler = 63;

htim14.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim14.Init.Period = 1000;

htim14.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim14.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

**if** (HAL\_TIM\_Base\_Init(&htim14) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM14\_Init 2 \*/

/\* USER CODE END TIM14\_Init 2 \*/

}

/\*\*

\* @brief USART2 Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_USART2\_UART\_Init**(**void**)

{

/\* USER CODE BEGIN USART2\_Init 0 \*/

/\* USER CODE END USART2\_Init 0 \*/

/\* USER CODE BEGIN USART2\_Init 1 \*/

/\* USER CODE END USART2\_Init 1 \*/

huart2.Instance = USART2;

huart2.Init.BaudRate = 115200;

huart2.Init.WordLength = UART\_WORDLENGTH\_8B;

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;

**if** (HAL\_UART\_Init(&huart2) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN USART2\_Init 2 \*/

/\* USER CODE END USART2\_Init 2 \*/

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

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

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* USER CODE BEGIN MX\_GPIO\_Init\_1 \*/

/\* USER CODE END MX\_GPIO\_Init\_1 \*/

/\* GPIO Ports Clock Enable \*/

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

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

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

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

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOF, led2\_Pin|led1\_Pin, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pins : led2\_Pin led1\_Pin \*/

GPIO\_InitStruct.Pin = led2\_Pin|led1\_Pin;

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

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

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

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

/\* USER CODE BEGIN MX\_GPIO\_Init\_2 \*/

/\* USER CODE END MX\_GPIO\_Init\_2 \*/

}

/\* 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 \*/