```c
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2019 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  ******************************************************************************
  created by Tobias Jehle, 15.08.2019
  */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "stdio.h"
#include "stm32f1xx_it.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 ---------------------------------------------------------*/
ETH_HandleTypeDef heth;

/* USER CODE BEGIN PV */
ETH_HandleTypeDef hethcp;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ETH_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

```

```c
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 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_ETH_Init();
  /* USER CODE BEGIN 2 */

  union RegVal0 {
      uint32_t  RegValue;
      uint16_t  OutputValue;
  }RegValues0;

  RegValues0.RegValue  = 0;

  union RegVal3{
      uint32_t RegValue;
      uint16_t OutputValue;

  }RegValues3;

  RegValues3.RegValue = 0;

              HAL_GPIO_WritePin(GPIOC, GPIO_PIN_9, GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_SET);
              HAL_ETH_WritePHYRegister(&heth, 0x0, 0xA100);
              HAL_ETH_WritePHYRegister(&hethcp, 0x0, 0xA100);
              HAL_Delay(10);
              HAL_ETH_WritePHYRegister(&heth, 0x0, 0x2100);
              HAL_ETH_WritePHYRegister(&heth, 0x17, 0x40E0);
              HAL_ETH_WritePHYRegister(&hethcp, 0x0, 0x2100);
```

```
125               HAL_ETH_WritePHYRegister(&hethcp, 0x1f, 0x8180);
126             //TEST AUSGABE LATCH_IN Register BEGIN
127             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1F);
128             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0467);
129             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x401F);
130             HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
131             printf("Register 467 Value: 0x%04x\n", RegValues0.OutputValue);
132             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1F);
133             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x018B);
134             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x401F);
135             HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
136             printf("Register 18B Value: 0x%04x\n", RegValues0.OutputValue);
137             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1F);
138             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x018B);
139             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x401F);
140             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x005A);
141             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1F);
142             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x018B);
143             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x401F);
144             HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
145             printf("Register 18B Value: 0x%04x\n", RegValues0.OutputValue);
146             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1);
147             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x834);
148             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x4001);
149             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x8000);
150             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1);
151             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0834);
152             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x4001);
153             HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
154             printf("Register 834 Value: 0x%04x\n", RegValues0.OutputValue);
155             //ENDE
156
157
158               printf("Prototyp Slave\n");
159                           printf("PHY0\n");
160
161                                       if ( HAL_ETH_ReadPHYRegister(&heth, 0x0,
    &RegValues0.RegValue) != HAL_OK ) {
162
163                                               printf("ERROR  reading REG\n");
164
165                                           }
166                                           else {
167
168                                               printf("RegisterValue: 0x%04x\n",
    RegValues0.OutputValue);
169
170                                           }
171                                       if ( HAL_ETH_ReadPHYRegister(&heth, 0x17,
    &RegValues0.RegValue) != HAL_OK ) {
172
173                                               printf("ERROR  reading REG\n");
174
175                                       }
176                                       else {
177
178                                               printf("RegisterValue: 0x%04x\n",
    RegValues0.OutputValue);
179
180                                        }
181                                     printf("PHY3\n");
182
```

```
183                                             if ( HAL_ETH_ReadPHYRegister(&hethcp, 0x0,
    &RegValues3.RegValue) != HAL_OK ) {
184
185                                                 printf("ERROR  reading REG\n");
186
187                                             }
188                                             else {
189
190                                                 printf("RegisterValue: 0x%04x\n",
    RegValues3.OutputValue);
191
192                                             }
193                                         printf("PHY3\n");
194
195                                         if ( HAL_ETH_ReadPHYRegister(&hethcp, 0x1f,
    &RegValues3.RegValue) != HAL_OK ) {
196
197                                             printf("ERROR  reading REG\n");
198
199                                         }
200                                         else {
201
202                                             printf("RegisterValue: 0x%04x\n",
    RegValues3.OutputValue);
203                                         }
204
205                                     //TEST AUSGABE LATCH_IN Register BEGIN
206                                     HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1F);
207                                     HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0467);
208                                     HAL_ETH_WritePHYRegister(&heth, 0xD, 0x401F);
209                                     HAL_ETH_ReadPHYRegister(&heth, 0xE,
    &RegValues0.RegValue);
210                                     printf("Register467 Value: 0x%04x\n",
    RegValues0.OutputValue);
211                                     //ENDE
212
213         //Testmodes Slave
214         /*printf("PHY0:\n");
215         HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1);
216         HAL_ETH_WritePHYRegister(&heth, 0xE, 0x836);
217         HAL_ETH_WritePHYRegister(&heth, 0xD, 0x4001);
218         HAL_ETH_WritePHYRegister(&heth, 0xE, 0x8000);
219         HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1);
220         HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0836);
221         HAL_ETH_WritePHYRegister(&heth, 0xD, 0x4001);
222         HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
223         printf("Register 836 Value: 0x%04x\n", RegValues0.OutputValue);
224         printf("PHY im Testmodus 4!\n");*/
225
226   /* USER CODE END 2 */
227
228   /* Infinite loop */
229   /* USER CODE BEGIN WHILE */
230   while (1)
231   {
232     /* USER CODE END WHILE */
233
234     /* USER CODE BEGIN 3 */
235
236   }
237
238   /* USER CODE END 3 */
```

```c
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Initializes the CPU, AHB and APB busses clocks
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV5;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.Prediv1Source = RCC_PREDIV1_SOURCE_PLL2;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
  RCC_OscInitStruct.PLL2.PLL2State = RCC_PLL2_ON;
  RCC_OscInitStruct.PLL2.PLL2MUL = RCC_PLL2_MUL8;
  RCC_OscInitStruct.PLL2.HSEPrediv2Value = RCC_HSE_PREDIV2_DIV2;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB busses 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();
  }
  HAL_RCC_MCOConfig(RCC_MCO, RCC_MCO1SOURCE_PLL3CLK, RCC_MCODIV_1);
  /** Configure the Systick interrupt time
  */
  __HAL_RCC_HSE_PREDIV2_CONFIG(RCC_HSE_PREDIV2_DIV2);
  /** Configure the Systick interrupt time
  */
  __HAL_RCC_PLLI2S_CONFIG(RCC_PLLI2S_MUL10);
  /** Configure the Systick interrupt time
  */
  __HAL_RCC_PLLI2S_ENABLE();
}

/**
  * @brief ETH Initialization Function
  * @param None
  * @retval None
  */
static void MX_ETH_Init(void)
{

  /* USER CODE BEGIN ETH_Init 0 */
```

```
301
302   /* USER CODE END ETH_Init 0 */
303
304    uint8_t MACAddr[6] ;
305
306   /* USER CODE BEGIN ETH_Init 1 */
307    hethcp.Instance = ETH;
308    hethcp.Init.AutoNegotiation = ETH_AUTONEGOTIATION_DISABLE;
309    hethcp.Init.Speed = ETH_SPEED_100M;
310    hethcp.Init.DuplexMode = ETH_MODE_HALFDUPLEX;
311    hethcp.Init.PhyAddress = 3;
312    MACAddr[0] = 0x00;
313    MACAddr[1] = 0x80;
314    MACAddr[2] = 0xE1;
315    MACAddr[3] = 0x00;
316    MACAddr[4] = 0x00;
317    MACAddr[5] = 0x00;
318    hethcp.Init.MACAddr = &MACAddr[0];
319    hethcp.Init.RxMode = ETH_RXINTERRUPT_MODE;
320    hethcp.Init.ChecksumMode = ETH_CHECKSUM_BY_HARDWARE;
321    hethcp.Init.MediaInterface = ETH_MEDIA_INTERFACE_RMII;
322   /* USER CODE END ETH_Init 1 */
323   heth.Instance = ETH;
324   heth.Init.AutoNegotiation = ETH_AUTONEGOTIATION_DISABLE;
325   heth.Init.Speed = ETH_SPEED_100M;
326   heth.Init.DuplexMode = ETH_MODE_HALFDUPLEX;
327   heth.Init.PhyAddress = 0;
328   MACAddr[0] = 0x00;
329   MACAddr[1] = 0x80;
330   MACAddr[2] = 0xE1;
331   MACAddr[3] = 0x00;
332   MACAddr[4] = 0x00;
333   MACAddr[5] = 0x00;
334   heth.Init.MACAddr = &MACAddr[0];
335   heth.Init.RxMode = ETH_RXINTERRUPT_MODE;
336   heth.Init.ChecksumMode = ETH_CHECKSUM_BY_HARDWARE;
337   heth.Init.MediaInterface = ETH_MEDIA_INTERFACE_RMII;
338
339   /* USER CODE BEGIN MACADDRESS */
340
341   /* USER CODE END MACADDRESS */
342
343   if (HAL_ETH_Init(&heth) != HAL_OK)
344   {
345     Error_Handler();
346   }
347   /* USER CODE BEGIN ETH_Init 2 */
348
349   /* USER CODE END ETH_Init 2 */
350
351 }
352
353 /**
354   * @brief GPIO Initialization Function
355   * @param None
356   * @retval None
357   */
358 static void MX_GPIO_Init(void)
359 {
360   GPIO_InitTypeDef GPIO_InitStruct = {0};
361
362   /* GPIO Ports Clock Enable */
```

```
363   __HAL_RCC_GPIOD_CLK_ENABLE();
364   __HAL_RCC_GPIOC_CLK_ENABLE();
365   __HAL_RCC_GPIOA_CLK_ENABLE();
366   __HAL_RCC_GPIOB_CLK_ENABLE();
367
368   /*Configure GPIO pin Output Level */
369   HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_8|GPIO_PIN_9, GPIO_PIN_RESET);
370
371   /*Configure GPIO pins : PC6 PC7 PC8 PC9 */
372   GPIO_InitStruct.Pin = GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_8|GPIO_PIN_9;
373   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
374   GPIO_InitStruct.Pull = GPIO_NOPULL;
375   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
376   HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
377
378   /*Configure GPIO pin : MCO_Out_Pin */
379   GPIO_InitStruct.Pin = MCO_Out_Pin;
380   GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
381   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
382   HAL_GPIO_Init(MCO_Out_GPIO_Port, &GPIO_InitStruct);
383
384   /*Configure GPIO pin : PC10 */
385   GPIO_InitStruct.Pin = GPIO_PIN_10;
386   GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
387   GPIO_InitStruct.Pull = GPIO_NOPULL;
388   HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
389
390   /* EXTI interrupt init*/
391   HAL_NVIC_SetPriority(EXTI15_10_IRQn, 3, 0);
392   HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
393
394 }
395
396 /* USER CODE BEGIN 4 */
397
398 /* USER CODE END 4 */
399
400 /**
401   * @brief  This function is executed in case of error occurrence.
402   * @retval None
403   */
404 void Error_Handler(void)
405 {
406   /* USER CODE BEGIN Error_Handler_Debug */
407   /* User can add his own implementation to report the HAL error return state */
408
409   /* USER CODE END Error_Handler_Debug */
410 }
411
412 #ifdef  USE_FULL_ASSERT
413 /**
414   * @brief  Reports the name of the source file and the source line number
415   *         where the assert_param error has occurred.
416   * @param  file: pointer to the source file name
417   * @param  line: assert_param error line source number
418   * @retval None
419   */
420 void assert_failed(uint8_t *file, uint32_t line)
421 {
422   /* USER CODE BEGIN 6 */
423   /* User can add his own implementation to report the file name and line number,
424      tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
```

```
425   /* USER CODE END 6 */
426 }
427 #endif /* USE_FULL_ASSERT */
428
429 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
430
```