```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 */
```

```
 63
 64 /* USER CODE END PFP */
 65
 66 /* Private user code ---------------------------------------------------------*/
 67 /* USER CODE BEGIN 0 */
 68
 69 /* USER CODE END 0 */
 70
 71 /**
 72   * @brief  The application entry point.
 73   * @retval int
 74   */
 75 int main(void)
 76 {
 77   /* USER CODE BEGIN 1 */
 78
 79   /* USER CODE END 1 */
 80
 81
 82   /* MCU Configuration--------------------------------------------------------*/
 83
 84   /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
 85   HAL_Init();
 86
 87   /* USER CODE BEGIN Init */
 88
 89   /* USER CODE END Init */
 90
 91   /* Configure the system clock */
 92   SystemClock_Config();
 93
 94   /* USER CODE BEGIN SysInit */
 95
 96   /* USER CODE END SysInit */
 97
 98   /* Initialize all configured peripherals */
 99   MX_GPIO_Init();
100   MX_ETH_Init();
101   /* USER CODE BEGIN 2 */
102
103   union RegVal0 {
104       uint32_t  RegValue;
105       uint16_t  OutputValue;
106   }RegValues0;
107
108   RegValues0.RegValue  = 0;
109
110   union RegVal3{
111       uint32_t RegValue;
112       uint16_t OutputValue;
113
114   }RegValues3;
115
116   RegValues3.RegValue = 0;
117
118
119               HAL_GPIO_WritePin(GPIOC, GPIO_PIN_9, GPIO_PIN_RESET);
120               HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_RESET);
121               HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_RESET);
122               HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_RESET);
123               HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_SET);
124               HAL_ETH_WritePHYRegister(&heth, 0x0, 0xA100);
```

```
125             HAL_ETH_WritePHYRegister(&hethcp, 0x0, 0xA100);
126             HAL_Delay(10);
127             HAL_ETH_WritePHYRegister(&heth, 0x0, 0x2100);
128             HAL_ETH_WritePHYRegister(&heth, 0x17, 0x4060);
129             HAL_ETH_WritePHYRegister(&hethcp, 0x0, 0x2100);
130             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1);
131             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0834);
132             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x4001);
133             HAL_ETH_WritePHYRegister(&heth, 0xE, 0xC000);
134
135             HAL_ETH_WritePHYRegister(&hethcp, 0x16, 0x0002);
136             HAL_ETH_WritePHYRegister(&hethcp, 0x1f, 0x8100);
137             HAL_ETH_ReadPHYRegister(&hethcp, 0x1F, &RegValues3.RegValue);
138             printf("3 Register 1F Value: 0x%04x\n", RegValues3.OutputValue);
139             //TEST AUSGABE LATCH_IN Register BEGIN
140             //HAL_ETH_WritePHYRegister(&heth, 0x0, 0x6100);
141             HAL_ETH_ReadPHYRegister(&heth, 0x1F, &RegValues0.RegValue);
142             printf("Register 1F Value: 0x%04x\n", RegValues0.OutputValue);
143             HAL_ETH_ReadPHYRegister(&heth, 0x1, &RegValues0.RegValue);
144             printf("Register 1 Value: 0x%04x\n", RegValues0.OutputValue);
145             HAL_ETH_WritePHYRegister(&hethcp, 0x0, 0x2100);
146             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1F);
147             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0463);
148             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x401F);
149             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0000);
150             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1F);
151             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x04E0);
152             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x401F);
153             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0003);
154             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x0001);
155             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0836);
156             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x4001);
157             HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
158             printf("Register 836 Value: 0x%04x\n", RegValues0.OutputValue);
159             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1F);
160             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x018B);
161             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x401F);
162             HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
163             printf("Register 18B Value: 0x%04x\n", RegValues0.OutputValue);
164             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1F);
165             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x018B);
166             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x401F);
167             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x005A);
168             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1F);
169             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x018B);
170             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x401F);
171             HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
172             printf("Register 18B Value: 0x%04x\n", RegValues0.OutputValue);
173             printf("Register Test:\n");
174             HAL_ETH_ReadPHYRegister(&hethcp, 0x1F, &RegValues3.RegValue);
175             printf("Register 1F Value: 0x%04x\n", RegValues3.OutputValue);
176             HAL_ETH_ReadPHYRegister(&hethcp, 0x1, &RegValues3.RegValue);
177             printf("Register 1 Value: 0x%04x\n", RegValues3.OutputValue);
178             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1);
179             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x834);
180             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x4001);
181             HAL_ETH_WritePHYRegister(&heth, 0xE, 0xC000);
182             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1);
183             HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0834);
184             HAL_ETH_WritePHYRegister(&heth, 0xD, 0x4001);
185             HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
186             printf("Register 834 Value: 0x%04x\n", RegValues0.OutputValue);
```

```
187             //ENDE
188
189             printf("Prototyp Master\n");
190                         printf("PHY0\n");
191
192                             if ( HAL_ETH_ReadPHYRegister(&heth, 0x0,
    &RegValues0.RegValue) != HAL_OK ) {
193
194                                 printf("ERROR  reading REG\n");
195
196                             }
197                             else {
198
199                                 printf("RegisterValue 0: 0x%04x\n",
    RegValues0.OutputValue);
200
201                             }
202                         if ( HAL_ETH_ReadPHYRegister(&heth, 0x17, &RegValues0.RegValue) !
    = HAL_OK ) {
203
204                                                                     printf("ERROR
    reading REG\n");
205
206                                                                 }
207                                                                 else {
208
209
    printf("RegisterValue 17: 0x%04x\n", RegValues0.OutputValue);
210
211                                                                  }
212                           printf("PHY3\n");
213
214                                 if ( HAL_ETH_ReadPHYRegister(&hethcp, 0x0,
    &RegValues3.RegValue) != HAL_OK ) {
215
216                                     printf("ERROR  reading REG\n");
217
218                                 }
219                                 else {
220
221
222                                     printf("RegisterValue: 0x%04x\n",
    RegValues3.OutputValue);
223
224                                 }
225                             if ( HAL_ETH_ReadPHYRegister(&hethcp, 0x17,
    &RegValues3.RegValue) != HAL_OK ) {
226
227                                     printf("ERROR  reading REG\n");
228                                 }
229                             else {
230                                     printf("RegisterValue 17: 0x%04x\n",
    RegValues3.OutputValue);
231                             }
232                         printf("PHY0\n");
233                         HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1);
234                         HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0834);
235                         HAL_ETH_WritePHYRegister(&heth, 0xD, 0x4001);
236                         HAL_ETH_ReadPHYRegister(&heth, 0xD, &RegValues0.RegValue);
237                         printf("RegisterValue: 0x%04x\n", RegValues0.OutputValue);
238
239                         HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
```

```
240                         printf("RegisterValue: 0x%04x\n", RegValues0.OutputValue);
241
242                       //TEST AUSGABE LATCH_IN Register BEGIN
243                       HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1F);
244                       HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0467);
245                       HAL_ETH_WritePHYRegister(&heth, 0xD, 0x401F);
246                       HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
247                       printf("Register467 Value: 0x%04x\n", RegValues0.OutputValue);
248                       //ENDE
249                     //Testmodes
250                       /*printf("PHY0:\n");
251                       HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1);
252                       HAL_ETH_WritePHYRegister(&heth, 0xE, 0x836);
253                       HAL_ETH_WritePHYRegister(&heth, 0xD, 0x4001);
254                       HAL_ETH_WritePHYRegister(&heth, 0xE, 0xA000);
255                       HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1);
256                       HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0836);
257                       HAL_ETH_WritePHYRegister(&heth, 0xD, 0x4001);
258                       HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
259                       printf("Register 836 Value: 0x%04x\n", RegValues0.OutputValue);
260                       printf("PHY im Testmodus 5!\n");*/
261                       /*printf("PHY0:\n");
262                       HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1);
263                       HAL_ETH_WritePHYRegister(&heth, 0xE, 0x836);
264                       HAL_ETH_WritePHYRegister(&heth, 0xD, 0x4001);
265                       HAL_ETH_WritePHYRegister(&heth, 0xE, 0x8000);
266                       HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1);
267                       HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0836);
268                       HAL_ETH_WritePHYRegister(&heth, 0xD, 0x4001);
269                       HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
270                       printf("Register 836 Value: 0x%04x\n", RegValues0.OutputValue);
271                       printf("PHY im Testmodus 4!\n");*/
272                       /*printf("PHY0:\n");
273                       HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1);
274                       HAL_ETH_WritePHYRegister(&heth, 0xE, 0x836);
275                       HAL_ETH_WritePHYRegister(&heth, 0xD, 0x4001);
276                       HAL_ETH_WritePHYRegister(&heth, 0xE, 0x4001);
277                       HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1);
278                       HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0836);
279                       HAL_ETH_WritePHYRegister(&heth, 0xD, 0x4001);
280                       HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
281                       printf("Register 836 Value: 0x%04x\n", RegValues0.OutputValue);
282                       printf("PHY im Testmodus 2!\n");*/
283                       /*HAL_ETH_WritePHYRegister(&heth, 0xE, 0x2001);
284                       HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1);
285                       HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0836);
286                       HAL_ETH_WritePHYRegister(&heth, 0xD, 0x4001);
287                       HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
288                       printf("Register 836 Value: 0x%04x\n", RegValues0.OutputValue);
289                       printf("PHY im Testmodus 1!\n");*/
290
291     //Register 400  Line Driver Series Termination
292     /*HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1F);
293     HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0400);
294     HAL_ETH_WritePHYRegister(&heth, 0xD, 0x401F);
295     HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0000);*/
296
297   /* USER CODE END 2 */
298
299   /* Infinite loop */
300   /* USER CODE BEGIN WHILE */
301   while (1)
```

```
302  {
303    /* USER CODE END WHILE */
304
305    /* USER CODE BEGIN 3 */
306      HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1F);
307      HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0197);
308      HAL_ETH_WritePHYRegister(&heth, 0xD, 0x401F);
309      HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
310      printf("Register 197 SNR Value: 0x%04x\n", RegValues0.OutputValue);
311
312      HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1F);
313      HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0198);
314      HAL_ETH_WritePHYRegister(&heth, 0xD, 0x401F);
315      HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
316      printf("Register 198 SQI Value: 0x%04x\n", RegValues0.OutputValue);
317
318      HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1F);
319      HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0133);
320      HAL_ETH_WritePHYRegister(&heth, 0xD, 0x401F);
321      HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
322      printf("Register 133 Link Value: 0x%04x\n", RegValues0.OutputValue);
323
324      HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1F);
325      HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0400);
326      HAL_ETH_WritePHYRegister(&heth, 0xD, 0x401F);
327      HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
328      printf("Register 400 Value: 0x%04x\n", RegValues0.OutputValue);
329
330      HAL_ETH_WritePHYRegister(&heth, 0xD, 0x1F);
331      HAL_ETH_WritePHYRegister(&heth, 0xE, 0x0448);
332      HAL_ETH_WritePHYRegister(&heth, 0xD, 0x401F);
333      HAL_ETH_ReadPHYRegister(&heth, 0xE, &RegValues0.RegValue);
334      printf("Register 448 ESDS Value: 0x%04x\n", RegValues0.OutputValue);
335  }
336
337  /* USER CODE END 3 */
338}
339
340/**
341  * @brief System Clock Configuration
342  * @retval None
343  */
344void SystemClock_Config(void)
345{
346  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
347  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
348
349  /** Initializes the CPU, AHB and APB busses clocks
350  */
351  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
352  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
353  RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV5;
354  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
355  RCC_OscInitStruct.Prediv1Source = RCC_PREDIV1_SOURCE_PLL2;
356  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
357  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
358  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
359  RCC_OscInitStruct.PLL2.PLL2State = RCC_PLL2_ON;
360  RCC_OscInitStruct.PLL2.PLL2MUL = RCC_PLL2_MUL8;
361  RCC_OscInitStruct.PLL2.HSEPrediv2Value = RCC_HSE_PREDIV2_DIV2;
362  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
363  {
```

```
364     Error_Handler();
365   }
366   /** Initializes the CPU, AHB and APB busses clocks
367   */
368   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
369                                 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
370   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
371   RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
372   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
373   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
374
375   if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
376   {
377     Error_Handler();
378   }
379   HAL_RCC_MCOConfig(RCC_MCO, RCC_MCO1SOURCE_PLL3CLK_DIV2, RCC_MCODIV_1);
380   /** Configure the Systick interrupt time
381   */
382   __HAL_RCC_HSE_PREDIV2_CONFIG(RCC_HSE_PREDIV2_DIV2);
383   /** Configure the Systick interrupt time
384   */
385   __HAL_RCC_PLLI2S_CONFIG(RCC_PLLI2S_MUL10);
386   /** Configure the Systick interrupt time
387   */
388   __HAL_RCC_PLLI2S_ENABLE();
389 }
390
391 /**
392   * @brief ETH Initialization Function
393   * @param None
394   * @retval None
395   */
396 static void MX_ETH_Init(void)
397 {
398
399   /* USER CODE BEGIN ETH_Init 0 */
400
401   /* USER CODE END ETH_Init 0 */
402
403    uint8_t MACAddr[6] ;
404
405   /* USER CODE BEGIN ETH_Init 1 */
406    hethcp.Instance = ETH;
407    hethcp.Init.AutoNegotiation = ETH_AUTONEGOTIATION_DISABLE;
408    hethcp.Init.Speed = ETH_SPEED_100M;
409    hethcp.Init.DuplexMode = ETH_MODE_HALFDUPLEX;
410    hethcp.Init.PhyAddress = 0x3U;
411    MACAddr[0] = 0x00;
412    MACAddr[1] = 0x80;
413    MACAddr[2] = 0xE1;
414    MACAddr[3] = 0x00;
415    MACAddr[4] = 0x00;
416    MACAddr[5] = 0x00;
417    hethcp.Init.MACAddr = &MACAddr[0];
418    hethcp.Init.RxMode = ETH_RXINTERRUPT_MODE;
419    hethcp.Init.ChecksumMode = ETH_CHECKSUM_BY_HARDWARE;
420    hethcp.Init.MediaInterface = ETH_MEDIA_INTERFACE_RMII;
421   /* USER CODE END ETH_Init 1 */
422   heth.Instance = ETH;
423   heth.Init.AutoNegotiation = ETH_AUTONEGOTIATION_DISABLE;
424   heth.Init.Speed = ETH_SPEED_100M;
425   heth.Init.DuplexMode = ETH_MODE_HALFDUPLEX;
```

```
426   heth.Init.PhyAddress = LAN8742A_PHY_ADDRESS;
427   MACAddr[0] = 0x00;
428   MACAddr[1] = 0x80;
429   MACAddr[2] = 0xE1;
430   MACAddr[3] = 0x00;
431   MACAddr[4] = 0x00;
432   MACAddr[5] = 0x00;
433   heth.Init.MACAddr = &MACAddr[0];
434   heth.Init.RxMode = ETH_RXINTERRUPT_MODE;
435   heth.Init.ChecksumMode = ETH_CHECKSUM_BY_HARDWARE;
436   heth.Init.MediaInterface = ETH_MEDIA_INTERFACE_RMII;
437
438   /* USER CODE BEGIN MACADDRESS */
439
440   /* USER CODE END MACADDRESS */
441
442   if (HAL_ETH_Init(&heth) != HAL_OK)
443   {
444     Error_Handler();
445   }
446   /* USER CODE BEGIN ETH_Init 2 */
447
448   /* USER CODE END ETH_Init 2 */
449
450 }
451
452 /**
453   * @brief GPIO Initialization Function
454   * @param None
455   * @retval None
456   */
457 static void MX_GPIO_Init(void)
458 {
459   GPIO_InitTypeDef GPIO_InitStruct = {0};
460
461   /* GPIO Ports Clock Enable */
462   __HAL_RCC_GPIOD_CLK_ENABLE();
463   __HAL_RCC_GPIOC_CLK_ENABLE();
464   __HAL_RCC_GPIOA_CLK_ENABLE();
465   __HAL_RCC_GPIOB_CLK_ENABLE();
466
467   /*Configure GPIO pin Output Level */
468   HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_8|GPIO_PIN_9, GPIO_PIN_RESET);
469
470   /*Configure GPIO pins : PC6 PC7 PC8 PC9 */
471   GPIO_InitStruct.Pin = GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_8|GPIO_PIN_9;
472   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
473   GPIO_InitStruct.Pull = GPIO_NOPULL;
474   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
475   HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
476
477   /*Configure GPIO pin : MCO_Out_Pin */
478   GPIO_InitStruct.Pin = MCO_Out_Pin;
479   GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
480   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
481   HAL_GPIO_Init(MCO_Out_GPIO_Port, &GPIO_InitStruct);
482
483   /*Configure GPIO pin : PC10 */
484   GPIO_InitStruct.Pin = GPIO_PIN_10;
485   GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
486   GPIO_InitStruct.Pull = GPIO_NOPULL;
487   HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
```

```
488
489   /* EXTI interrupt init*/
490   HAL_NVIC_SetPriority(EXTI15_10_IRQn, 3, 0);
491   HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
492
493 }
494
495 /* USER CODE BEGIN 4 */
496
497 /* USER CODE END 4 */
498
499 /**
500   * @brief  This function is executed in case of error occurrence.
501   * @retval None
502   */
503 void Error_Handler(void)
504 {
505   /* USER CODE BEGIN Error_Handler_Debug */
506   /* User can add his own implementation to report the HAL error return state */
507
508   /* USER CODE END Error_Handler_Debug */
509 }
510
511 #ifdef  USE_FULL_ASSERT
512 /**
513   * @brief  Reports the name of the source file and the source line number
514   *         where the assert_param error has occurred.
515   * @param  file: pointer to the source file name
516   * @param  line: assert_param error line source number
517   * @retval None
518   */
519 void assert_failed(uint8_t *file, uint32_t line)
520 {
521   /* USER CODE BEGIN 6 */
522   /* User can add his own implementation to report the file name and line number,
523      tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
524   /* USER CODE END 6 */
525 }
526 #endif /* USE_FULL_ASSERT */
527
528 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
529
```