```c
/* 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"
#include "tim.h"
#include "usart.h"
#include "gpio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "led.h"
#include "interrupt.h"
#include "lcd.h"
#include "stdio.h"
#include "string.h"


/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
extern struct keys key[4];
extern uint PA15_freq,PA15_duty;
extern uint PA15_rise,PA15_fall;
extern uint PB4_freq,PB4_duty;
extern uint PB4_rise,PB4_fall;
extern char rx_arry[50];
extern char rx_data;
extern char rx_pointer;

/* 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 ---------------------------------------------------------*/

/* USER CODE BEGIN PV */
__IO uint32_t led_uwTick;
__IO uint32_t lcd_uwTick;
__IO uint32_t freq_uwTick;
__IO uint32_t freq_uwTick_2;

char lcd_arry[50];
char lcd_view;
uint PD_value = 1000;
uint PH_value = 5000;
int PX_value = 0;//有符号
```

```
uint NDA_value;
uint NDB_value;
uint NHA_value;
uint NHB_value;

uint para_choose_state;//0,PD;1,PH;2,PX
uint freq_or_period_flag;//0,频率显示;1, 周期显示
uint RECD_clear_flag;//1,清零

int A_freq,B_freq;//A:PA15;B:PB4

int A_freq_reg;
int B_freq_reg;
uint A_freq_max = 0;
uint A_freq_min = 100000;
uint B_freq_max = 0;
uint B_freq_min = 100000;

uint A_freq_null_flag,B_freq_null_flag;

char led_num =0x00;
uint time_3s_windows_flag;

/* USER CODE END PV */

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

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void rx_proc();
void key_proc();
void led_proc();
void lcd_proc();
void freq_proc();
/* 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_TIM3_Init();
  MX_TIM6_Init();
  MX_TIM8_Init();
```

```
  MX_USART1_UART_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
    HAL_TIM_Base_Start_IT(&htim6);
    HAL_UART_Receive_IT(&huart1, (uint8_t *)&rx_data, 1);
    HAL_TIM_IC_Start_IT(&htim8,TIM_CHANNEL_1);
    HAL_TIM_IC_Start_IT(&htim8,TIM_CHANNEL_2);
    HAL_TIM_IC_Start_IT(&htim3,TIM_CHANNEL_1);
    HAL_TIM_IC_Start_IT(&htim3,TIM_CHANNEL_2);

    LCD_Init();
    LCD_Clear(Black);
    LCD_SetBackColor(Black);
    LCD_SetTextColor(White);


  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
      rx_proc();
      key_proc();
      led_proc();
      lcd_proc();
      freq_proc();

  }
  /* 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_PWREx_ControlVoltageScaling(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 = RCC_PLLM_DIV3;
  RCC_OscInitStruct.PLL.PLLN = 20;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
  RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
  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;
```

```
214   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
215   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
216
217   if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
218   {
219     Error_Handler();
220   }
221 }
222
223 /* USER CODE BEGIN 4 */
224 void freq_proc()
225 {
226     if(uwTick - freq_uwTick_2 <100)
227         return;
228     freq_uwTick_2 = uwTick;
229
230
231     A_freq = PA15_freq + PX_value;
232     B_freq = PB4_freq + PX_value;
233
234     if(A_freq_reg <= PH_value && A_freq_reg > 0)
235     {
236         if(A_freq >PH_value && A_freq>0)
237             NHA_value++;
238     }
239
240     if(B_freq_reg <= PH_value && B_freq_reg>0)
241     {
242         if(B_freq >PH_value && B_freq>0)
243             NHB_value++;
244     }
245
246     A_freq_reg = A_freq;
247     B_freq_reg = B_freq;
248
249     if(A_freq > A_freq_max && A_freq >0)
250         A_freq_max = A_freq;
251     else if(A_freq < A_freq_min && A_freq >0)
252         A_freq_min = A_freq;
253
254     if(B_freq > B_freq_max && B_freq >0)
255         B_freq_max = B_freq;
256     else if(B_freq < B_freq_min && B_freq >0)
257         B_freq_min = B_freq;
258
259     if(time_3s_windows_flag == 1)//3s内
260     {
261         time_3s_windows_flag = 0;
262
263         if((A_freq_max - A_freq_min) > PD_value)
264             NDA_value++;
265
266         if((B_freq_max - B_freq_min) > PD_value)
267             NDB_value++;
268
269         B_freq_min = 100000;B_freq_max = 0;
270         A_freq_min = 100000;A_freq_max = 0;
271     }
272
273     if(uwTick - freq_uwTick >3000)
274     {
275         freq_uwTick = uwTick;
276         time_3s_windows_flag = 1;
277     }
278
279     if(RECD_clear_flag == 1)
280     {
281         RECD_clear_flag = 0;
282         NDA_value = 0;
283         NDB_value = 0;
284         NHA_value = 0;
```

```
285             NHB_value = 0;
286         }
287 
288 }
289 void led_proc()
290 {
291     led_num = 0x00;
292     if(lcd_view == 0)
293         led_num = led_num|0x01;
294     else led_num = led_num&0xfe;
295 
296     if(A_freq >PH_value && A_freq >0)
297         led_num = led_num|0x02;
298     else led_num = led_num&0xfd;
299 
300     if(B_freq >PH_value && B_freq >0)
301         led_num = led_num|0x04;
302     else led_num = led_num&0xfb;
303 
304     if(NDA_value >=3 ||NDB_value >=3)
305         led_num = led_num|0x80;
306     else led_num = led_num&0x7f;
307 
308     led_disp(led_num);
309 }
310 void lcd_proc()
311 {
312     if(uwTick - lcd_uwTick < 100)
313         return;
314     lcd_uwTick = uwTick;
315 //  LCD_Clear(Black);
316 
317     if(lcd_view == 0)//数据界面
318     {
319         sprintf(rx_arry,"        DATA");
320         LCD_DisplayStringLine(Line1,(u8 *)rx_arry);
321 
322         if(freq_or_period_flag == 0)//周期模式
323         {
324             if(A_freq>=1000)
325             {
326                 sprintf(rx_arry,"     A=%-5.2fKHz",A_freq/1000.0);
327                 LCD_DisplayStringLine(Line3,(u8 *)rx_arry);
328             }
329             else if(A_freq<1000 && A_freq>=0)
330             {
331                 sprintf(rx_arry,"     A=   Hz   ");
332                 LCD_DisplayStringLine(Line3,(u8 *)rx_arry);
333                 sprintf(rx_arry,"     A=%-3dHz",A_freq);
334                 LCD_DisplayStringLine(Line3,(u8 *)rx_arry);
335             }
336             else
337             {
338                 sprintf(rx_arry,"     A=NULL  ");
339                 LCD_DisplayStringLine(Line3,(u8 *)rx_arry);
340             }
341 
342             if(B_freq>=1000)
343             {
344                 sprintf(rx_arry,"     B=%-5.2fKHz",B_freq/1000.0);
345                 LCD_DisplayStringLine(Line4,(u8 *)rx_arry);
346             }
347             else if(B_freq<1000 && B_freq>=0)
348             {
349                 sprintf(rx_arry,"     B=   Hz   ");
350                 LCD_DisplayStringLine(Line4,(u8 *)rx_arry);
351                 sprintf(rx_arry,"     B=%-3dHz",B_freq);
352                 LCD_DisplayStringLine(Line4,(u8 *)rx_arry);
353             }
354             else
355             {
```

```
                sprintf(rx_arry,"      B=NULL   ");
                LCD_DisplayStringLine(Line4,(u8 *)rx_arry);
            }
        }
        else
        {
            if(A_freq<1000 && A_freq>=0)//周期小于1000hz，周期大于1ms，大于1000us
            {
                sprintf(rx_arry,"      A=%-4.2fmS",1000.0/A_freq);
                LCD_DisplayStringLine(Line3,(u8 *)rx_arry);
            }
            else if(A_freq >=1000)
            {
                sprintf(rx_arry,"      A=    uS ");
                LCD_DisplayStringLine(Line3,(u8 *)rx_arry);
                sprintf(rx_arry,"      A=%-3duS",1000000/A_freq);
                LCD_DisplayStringLine(Line3,(u8 *)rx_arry);
            }
            else
            {
                sprintf(rx_arry,"      A=NULL");
                LCD_DisplayStringLine(Line3,(u8 *)rx_arry);
            }

            if(B_freq<1000 && B_freq>=0)
            {
                sprintf(rx_arry,"      B=%-4.2fmS",1000.0/B_freq);
                LCD_DisplayStringLine(Line4,(u8 *)rx_arry);
            }
            else if(B_freq >=1000)
            {
                sprintf(rx_arry,"      B=    uS ");
                LCD_DisplayStringLine(Line4,(u8 *)rx_arry);
                sprintf(rx_arry,"      B=%-3duS",1000000/B_freq);
                LCD_DisplayStringLine(Line4,(u8 *)rx_arry);
            }
            else
            {
                sprintf(rx_arry,"      B=NULL");
                LCD_DisplayStringLine(Line4,(u8 *)rx_arry);
            }
        }
    }
    else if(lcd_view == 1)//参数界面
    {
        sprintf(rx_arry,"         PARA");
        LCD_DisplayStringLine(Line1,(u8 *)rx_arry);

        sprintf(rx_arry,"      PD=%-4dHz",PD_value);
        LCD_DisplayStringLine(Line3,(u8 *)rx_arry);
        sprintf(rx_arry,"      PH=%-4dHz",PH_value);
        LCD_DisplayStringLine(Line4,(u8 *)rx_arry);
        sprintf(rx_arry,"      PX=%dHz",PX_value);
        LCD_DisplayStringLine(Line5,(u8 *)rx_arry);
    }
    else if(lcd_view == 2)//统计界面
    {
        sprintf(rx_arry,"         RECD");
        LCD_DisplayStringLine(Line1,(u8 *)rx_arry);

        sprintf(rx_arry,"      NDA=%d",NDA_value);
        LCD_DisplayStringLine(Line3,(u8 *)rx_arry);
        sprintf(rx_arry,"      NDB=%d",NDB_value);
        LCD_DisplayStringLine(Line4,(u8 *)rx_arry);
        sprintf(rx_arry,"      NHA=%d",NHA_value);
        LCD_DisplayStringLine(Line5,(u8 *)rx_arry);
        sprintf(rx_arry,"      NHB=%d",NHB_value);
        LCD_DisplayStringLine(Line6,(u8 *)rx_arry);
    }

}
```

```
427  void key_proc()
428  {
429      for(int i=0;i<4;i++)
430          if((key[i].short_flag == 1) || (key[i].long_flag == 1))
431              LCD_Clear(Black);
432
433      if(key[0].short_flag == 1)
434      {
435          key[0].short_flag = 0;
436          switch(para_choose_state)
437          {
438              case 0:
439                  if(PD_value>=100 && PD_value<1000)
440                      PD_value+=100;
441                  break;
442              case 1:
443                  if(PH_value>=1000 && PH_value<10000)
444                      PH_value+=100;
445                  break;
446              case 2:
447                  if(PX_value>=-1000 && PX_value<1000)
448                      PX_value+=100;
449                  break;
450          }
451      }
452
453      if(key[1].short_flag == 1)
454      {
455          key[1].short_flag = 0;
456          switch(para_choose_state)
457          {
458              case 0:
459                  if(PD_value>100 && PD_value<=1000)
460                      PD_value-=100;
461                  break;
462              case 1:
463                  if(PH_value>1000 && PH_value<=10000)
464                      PH_value-=100;
465                  break;
466              case 2:
467                  if(PX_value>-1000 && PX_value<=1000)
468                      PX_value-=100;
469                  break;
470          }
471      }
472
473      if(key[2].short_flag == 1)
474      {
475          key[2].short_flag = 0;
476          if(lcd_view == 1)//参数界面
477          {
478              para_choose_state++;
479              if(para_choose_state>=3)
480                  para_choose_state = 0;
481          }
482          else if(lcd_view == 0) //数据界面
483          {
484              freq_or_period_flag = !freq_or_period_flag;
485          }
486      }
487
488      if(key[2].long_flag == 1)
489      {
490          key[2].long_flag = 0;
491          if(lcd_view == 2)//记录界面
492          {
493              RECD_clear_flag = 1;
494          }
495      }
496
497      if(key[3].short_flag == 1)
```

```
498         {
499             key[3].short_flag = 0;
500             lcd_view++;
501             if(lcd_view == 3)
502                 lcd_view = 0;
503         }
504 
505         if(lcd_view == 0)
506             para_choose_state = 0;
507         else if(lcd_view == 1)
508             freq_or_period_flag = 0;
509 
510 
511 }
512 void rx_proc()
513 {
514     if(rx_pointer!=0)
515     {
516         int temp = rx_pointer;
517         HAL_Delay(1);
518         if(temp == rx_pointer)
519         {
520             //...
521             printf("ok\r\n");
522             printf("%d\r\n",PA15_freq);
523             printf("%d\r\n",PB4_freq);
524             rx_pointer = 0;memset(rx_arry,0,50);
525         }
526     }
527 }
528 int fputc(int ch, FILE *f)
529 {
530   HAL_UART_Transmit(&huart1, (const uint8_t *)&ch, 1, 20);
531   return ch;
532 }
533 
534 /* USER CODE END 4 */
535 
536 /**
537   * @brief  This function is executed in case of error occurrence.
538   * @retval None
539   */
540 void Error_Handler(void)
541 {
542   /* USER CODE BEGIN Error_Handler_Debug */
543   /* User can add his own implementation to report the HAL error return state */
544   __disable_irq();
545   while (1)
546   {
547   }
548   /* USER CODE END Error_Handler_Debug */
549 }
550 
551 #ifdef  USE_FULL_ASSERT
552 /**
553   * @brief  Reports the name of the source file and the source line number
554   *         where the assert_param error has occurred.
555   * @param  file: pointer to the source file name
556   * @param  line: assert_param error line source number
557   * @retval None
558   */
559 void assert_failed(uint8_t *file, uint32_t line)
560 {
561   /* USER CODE BEGIN 6 */
562   /* User can add his own implementation to report the file name and line number,
563      ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
564   /* USER CODE END 6 */
565 }
566 #endif /* USE_FULL_ASSERT */
567 
```