```c
#include "interrupt.h"
//按键
//UART
//捕获
//eeprom读写

#define PA15_FREQ 1000000
#define PB4_FREQ 1000000

struct keys key[4] = {0};

char rx_arry[50];
char rx_data;
char rx_pointer;

uint PA15_freq,PA15_duty,PA15_rise,PA15_fall;
uint PB4_freq,PB4_duty,PB4_rise,PB4_fall;
uint PB4 flag = 0;

void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
    if(htim->Instance == TIM3)
    {
        if(htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)
        {
            PA15_rise = __HAL_TIM_GetCounter(htim);
            __HAL_TIM_SetCounter(htim,0);
            PA15_freq = PA15_FREQ/PA15_rise;
            PA15 duty = PA15 fall*100/PA15 rise;
        }
        if(htim->Channel == HAL_TIM_ACTIVE_CHANNEL_2)
        {
            PA15_fall = __HAL_TIM_GetCounter(htim);
        }
    }
    if(htim->Instance == TIM8)
    {
        if(PB4_flag == 0)
        {
            PB4 rise =   HAL TIM GetCounter(htim);
            __HAL_TIM_SetCounter(htim,0);
            PB4_freq = PB4_FREQ/PB4_rise;
            PB4_duty = PB4_fall*100/PB4_rise;
            __HAL_TIM_SET_CAPTUREPOLARITY(htim,TIM_CHANNEL_1,TIM_INPUTCHANNELPOLARITY_FALLING);
        }
        else
        {
            PB4_fall = __HAL_TIM_GetCounter(htim);
              HAL TIM SET CAPTUREPOLARITY(htim,TIM CHANNEL 1,TIM INPUTCHANNELPOLARITY RISING);
        }

        PB4_flag = !PB4_flag;
    }
}
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if(huart->Instance == USART1)
    {
        rx_arry[rx_pointer++] = rx_data;
        HAL UART Receive IT(huart, (uint8 t *)&rx data, 1);
    }
}
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if(htim->Instance == TIM6)
    {
        key[0].value = HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_0);
        key[1].value = HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_1);
        key[2].value = HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_2);
        key[3].value = HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_0);

```

```
72          for(int i=0;i<4;i++)
73          {
74              switch(key[i].state)
75              {
76                  case 0:
77                      if(key[i].value == 0) key[i].state = 1;
78                      break;
79                  case 1:
80                      if(key[i].value == 0)
81                      {
82                          key[i].state = 2;
83                          key[i].click_time = 0;
84                      }
85                      else key[i].state = 0;
86                      break;
87  //                  case 2:
88  //                      if(key[i].value == 0)
89  //                          key[i].click time++;
90  //                      else if(key[i].click_time >70)
91  //                      {
92  //                          key[i].long_flag = 1;
93  //                          key[i].state = 0;
94  //                      }
95  //                      else
96  //                      {
97  //                          switch(key[i].double_state)
98  //                          {
99  //                              case 0:
100 //                                  key[i].double state = 1;
101 //                                  break;
102 //                              case 1:
103 //                                  key[i].double_flag = 1;
104 //                                  key[i].double_state = 0;
105 //                                  break;
106 //                          }
107 //
108 //                          key[i].double_time = 0;
109 //                          key[i].state = 0;
110 //                      }
111 //
112 //                      break;
113
114                     case 2:
115                         if(key[i].value == 0)
116                             key[i].click_time++;
117                         else
118                         {
119                             key[i].short_flag = 1;
120                             key[i].state = 0;
121                         }
122                         break;
123             }
124
125 //          if(key[i].double_state == 1)
126 //          {
127 //              key[i].double_time++;
128 //              if(key[i].double_time>30)
129 //              {
130 //                  key[i].short_flag = 1;
131 //                  key[i].double state = 0;
132 //              }
133 //          }
134         }
135     }
136 }
137
138
```