实验 4 - 有源 RFID 演示系统实验

1. 实验目的

掌握 NRF24LE1 作为有源电子标签和接收器的应用。

- 有源电子标签的工作流程；
- 休眠在有源电子标签中的作用和应用；
- 欠压指示在有源电子标签中的作用和应用；
- 接收器的工作流程、数据校验、数据打包上传。

2. 实验内容

一块开发板作为接收器（Reader），另一块开发板作为有源电子标签（Tag），用多个 Tag 测试效果更好。

每个 Tag 分配一个专属的 ID 号用于表示自己的身份。Tag 采用间歇式的发射方式，即每秒发射一次自身的 ID 和电压采样值，发射完成后休眠 1 秒，以达到节电的目的。

Reader 接收 Tag 发射的无线信息，并对接收到的信息进行校验，之后打包通过串口上传给计算机，计算机上的测试软件会对 Reader 上传的信息进行解析，并实时显示 Tag 的信息和上报的次数。

3. 实验设备

表 1：实验设备

硬件	
1.	IK-24LE1DK 开发板　至少 2 块
2.	USB MINI 数据线　至少 2 根
3.	nRFPRO 下载器　1 个
4.	10 芯排线，间距 2.54mm　1 根
软件	
1.	Keil for 51 集成开发环境
2.	有源 RFID 演示系统上位机软件 ActRFID.exe

4. 原理说明

4.1. 系统架构示意图

有源 RFID 演示系统的结构如图 1 所示，系统中包含的设备和作用如下：

- 计算机：提供图形界面，用于显示 Reader 上传的 Tag 的信息；
- 阅读器 Reader：通过无线的方式接收 Tag 的信息，并对信息进行校验，将接收到的数据整理打包后上传至计算机；

- Tag：定时采集自身供电的电池电压，每隔 1 秒发射一次自身的 ID 和电压采样值。

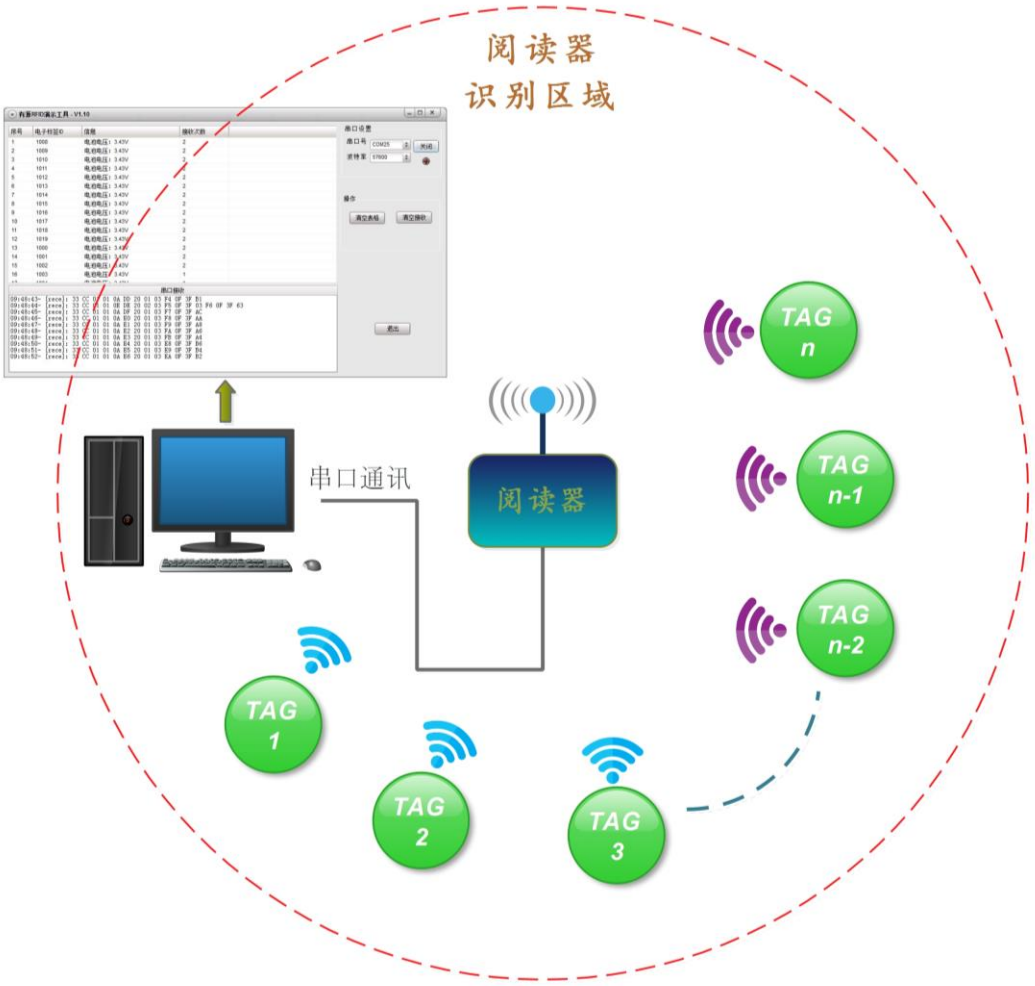
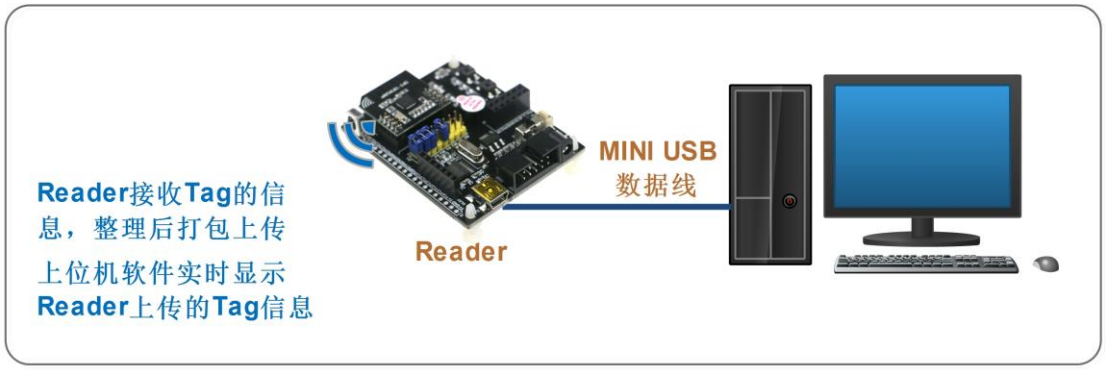


图 1：有源 RFID 系统结构示意图

4.2. 系统架构示意图

在本演示实验中，使用 ZT-24LE1 开发板作为 Reader 和 Tag。系统搭建如下：



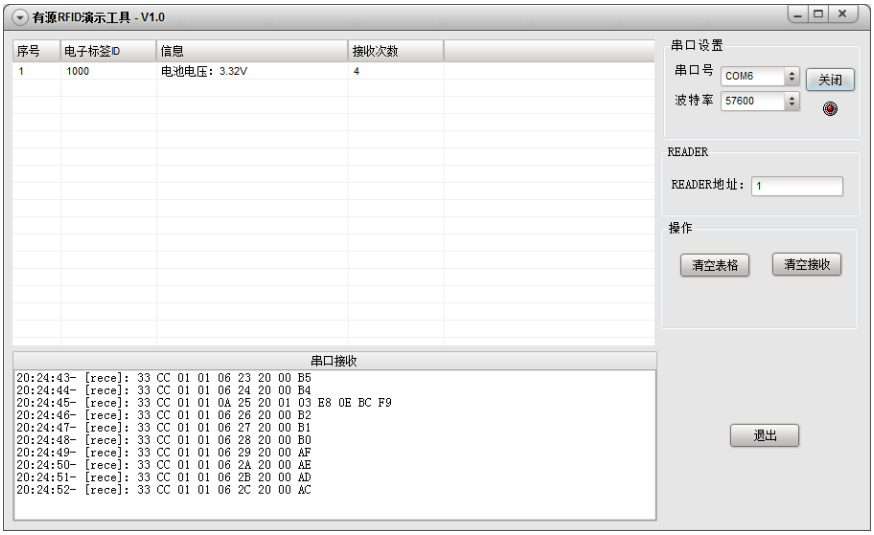


图 2：有源 RFID 系统搭建

如上图所示，系统工作时，Reader 接收 TAG 定时上报的信息，接收到 TAG 后信息，将 TAG 信息写入缓存，并对每个 TAG 进行计时，相同 ID 的 TAG 第一次接收到会立即上报，之后若继续接收到，则每 10s 上报一次，这样做的目的是为了降低数据流量，在系统比较庞大时，会有效减轻系统的负荷。

Reader 无论有无 TAG 信息时，每秒都会向 PC 机上报一次数据(有 TAG 信息时上报信息，没有 TAG 信息时上报联络包)。

有源 RFID 演示系统的无线参数配置如下：

表 2：有源 RFID 演示系统无线参数配置

序号	参数	TAG	READER	备注
1	通信频率	信道:50 频段:2450MHz；		
2	通信速率	250Kbps		
3	发射功率	0dBm		0dBm
4	自动应答	无		无
5	自动重发	无		无
7	地址宽度	5 个字节		
8	CRC 校验	2 个字节		
11	休眠时间	800ms		无

5. 通讯协议

5.1. Tag 和 Reader 之间的通讯协议

表 3：Tag→Reader 的数据格式

字节 1	字节 2	字节 3	字节 4	字节 5	字节 6
命令(CMD)	ID 高字节(IDH)	ID 低字节(IDL)	电池电压高字节（VH）	电池电压低字节（VL）	校验:累加和

- **命令 CMD**

表 4：发射器→接收器命令表

命令内容	命令字(HEX)	意义
上报 ID	1E	向接收器上报自身的 ID 号和电压采样值。

- TAG ID

编码型式：16 位二进制，编码范围：1～65535。如 ID 号 1000，对应的 IDH=0x03，IDL=0xE8。

- **校验(verification)**

采用双重校验，即同时采用累加和检验和 CRC16 的方式，其中，累加和检验由软件完成，CRC16 校验由硬件自动完成。

累加和校验 = (256 -（CMD + IDH + IDL + VH + VL）) %256

5.2. Reader 和计算机之间的通讯协议

表 5：接收器上报信息的数据格式

33 CC	DA	SA	IL	SN	数据包裹（Package）	verification
帧头	目的地址	源地址	数据长度	流水号	数据	累加和校验

- **帧头(Head)**
  表示一个数据帧开始传输，固定为：0x33, 0xCC。

- **目的地址(Dest address)**
  接收方地址，本演示系统中指的是计算机的地址。

- **源地址(Source address)**
  发送方地址，即 Reader 的地址。

- **帧长度(Frame length)**
  发送方地址。
  IL = 数据包裹长度+4

- **流水号(Serial number)**
  从 1 开始，每发送一次加 1，加到 0xFF 时返回到 1。

- **校验(verification)**
  采用累加和检验。
  累加和校验 = (256 -（DA + SA + IL + SN + Package）)%256

- **数据包裹（Package）**
  数据包裹（Package）由命令和 TAG 信息组成.

表 6：Reader→PC 数据包裹（Package）格式

数据包裹（Package）格式				
命令	TAG 数量	TAG 1	…	TAG n
1 字节	1 字节	5 字节	…	5 字节

6. 实验步骤

注意：本实验最少需要两块 IK-24LE1 开发板。

- 在 Keil uVision4 中打开工程“reader.uvproj”工程；

- 编译工程，注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止；

- 将编译生成的 HEX 文件“reader.hex” (该文件位于工程目录下的“Object”文件夹中)通过编程器下载到第 1 块开发板中运行；
- 在 Keil uVision4 中打开工程“tag.uvproj”工程；
- 编译工程，注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止；
- 将编译生成的 HEX 文件“tag.hex” (该文件位于工程目录下的“Object”文件夹中)通过编程器下载到第 2 块开发板中运行；
- 打开“有源 RFID 演示工具”，选择串口号，设置波特率为 57600，根据程序中的设定的 Reader 地址在软件中 Reader 地址，打开串口。

- 观察 “有源 RFID 演示工具”中上报的 TAG 信息。

7. 实验程序

7.1. 程序流程

1. Tag 程序流程图

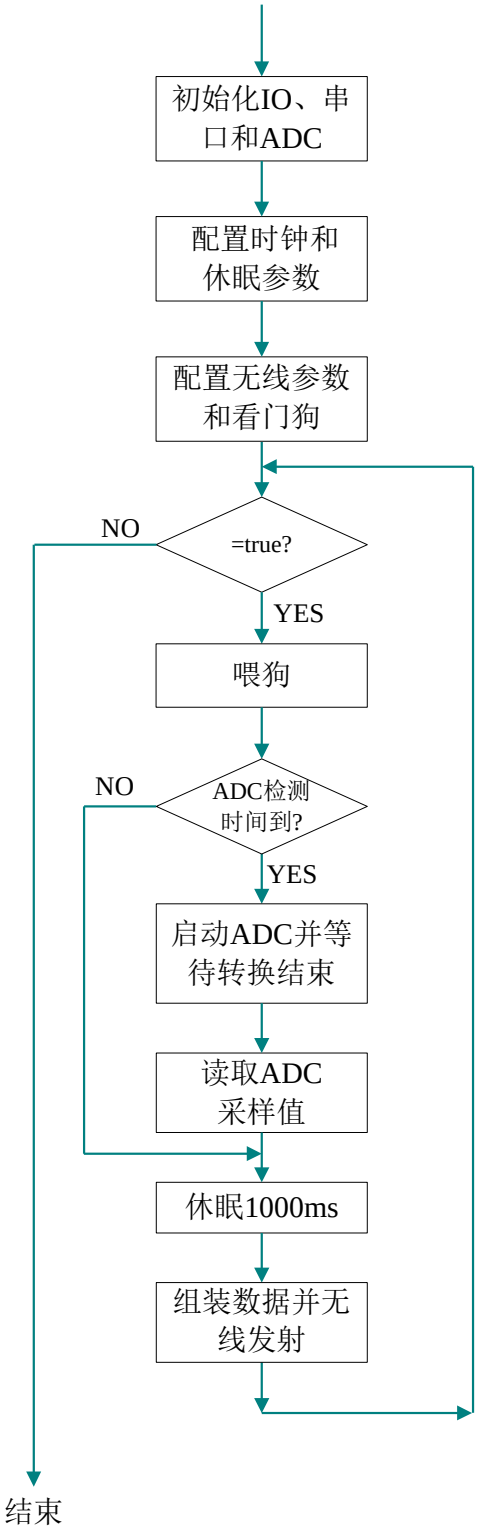


图 3：TAG 工作流程图

2. Reader 程序流程图

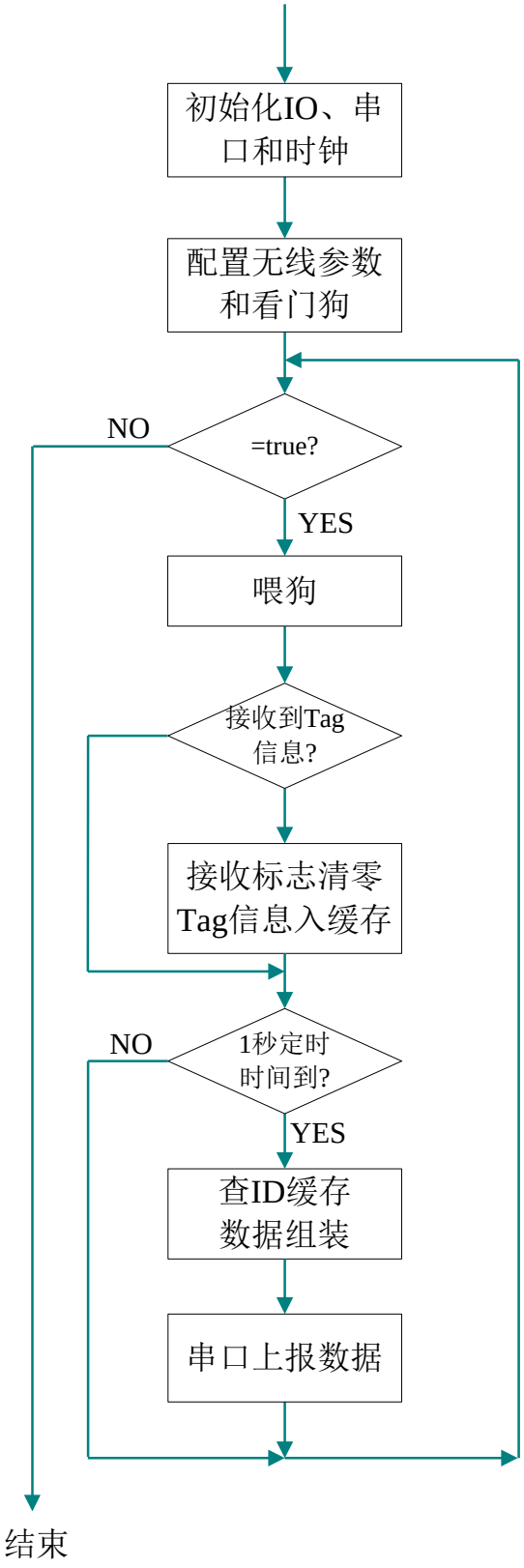


图 4：Reader 工作流程图

7.2. 程序清单

7.2.1. Tag

```
/*********************************************************************
* 描  述 : 初始化 IO
* 入  参 : 无
* 返回值 : 无
*********************************************************************/
void IoInit(void)
{
    P0DIR = 0x00;
    P1DIR = 0x00;
}

/*********************************************************************
** 描  述:  adc 初始化
** 入  参:  NONE
** 返回值:  NONE
*********************************************************************/
void adc_init(void)
{
  hal_adc_set_input_channel(HAL_INP_VDD1_3);  //设置通道 检测 1/3 VDD 电压
  hal_adc_set_reference(HAL_ADC_REF_INT);      //设置参考电压 内部 1.22V
  hal_adc_set_input_mode(HAL_ADC_SINGLE);            //单端输入
  hal_adc_set_conversion_mode(HAL_ADC_SINGLE_STEP); //单次采样模式
  hal_adc_set_sampling_rate(HAL_ADC_2KSPS);          //采样速率为 2ksps
  hal_adc_set_resolution(HAL_ADC_RES_12BIT);         //12 位 ADC
  hal_adc_set_data_just(HAL_ADC_JUST_RIGHT);         //数据右对齐
}

/*********************************************************************
 * 描  述 : 设置休眠时间，最长时间 2 秒(65536)，如参数输入错误，采用默认值 32768
 * 入  参 : period:休眠时间，范围 10~65536
 * 返回值 : 无
*********************************************************************/
void set_timer_period(uint16_t period)
{
    if((period<10) && (period>65536))period = 32768;

    hal_rtc_start(false);
    hal_rtc_start(true);
    hal_rtc_set_compare_value(period - 1);
}
```

```
/*****************************************************************
 * 描  述 : 配置无线参数
 * 入  参 : 无
 * 返回值 : 无
*****************************************************************/
void RfCofig(void)
{
  RFCKEN = 1;        //使能 RF 时钟

  hal_nrf_close_pipe(HAL_NRF_ALL);           //先关闭所有的通道.
  hal_nrf_open_pipe(HAL_NRF_PIPE0,false);    //再打开通道 0.

  hal_nrf_set_operation_mode(HAL_NRF_PTX);    // 模式：发送
  hal_nrf_set_rf_channel(TAG_CH); //RF 信道 50。接收和发送必须处于同一信道
  hal_nrf_set_datarate(HAL_NRF_250KBPS);       // RF 速率：250KBPS
  hal_nrf_set_output_power(HAL_NRF_0DBM);      // 功率：0DBM
  hal_nrf_set_crc_mode(HAL_NRF_CRC_16BIT);    //设置 CRC 校验：16 位 CRC。必
                                                须和接收设备一致。
  hal_nrf_set_address(HAL_NRF_TX,TX_ADDRESS); //设置发射机地址
  hal_nrf_set_auto_retr(0,1500);                //自动重发:0

  hal_nrf_set_power_mode(HAL_NRF_PWR_UP);       //发射机上电
  RF = 1;       //使能无线中断
  EA = 1;        // 使能全局中断
}

/*****************************************************************
 * 描  述 : 时钟和 RTC 唤醒配置
 * 入  参 : 无
 * 返回值 : 无
*****************************************************************/
void mcu_init(void)
{
    hal_rtc_start(false); //关闭 32.768KHz 时钟
    //32.768KHz 的时钟源为内部 RC
    hal_clklf_set_source(HAL_CLKLF_RCOSC32K);

    hal_rtc_set_compare_mode(HAL_RTC_COMPARE_MODE_0);//32 KHz 模式 0
    set_timer_period(TAG_TIME);                 //设置休眠时间
    hal_clk_set_16m_source(HAL_CLK_XOSC16M); //使用外部 16MHz 晶振
    hal_clk_regret_xosc16m_on(0); //在寄存器维持低功耗模式下关闭 16MHz 时钟

    hal_rtc_start(true); //启动 32kHz 时钟
```

```
  while((CLKLFCTRL&0x80)==0x80);    //等待时钟启动完成
  while((CLKLFCTRL&0x80)!=0x80);
}

/*********************************************************************
 * 描  述 : 组装数据
 * 入  参 : 无
 * 返回值 : 无
*********************************************************************/
void Assemble_Data(void)
{
    xdata uint8_t fcs = 0,i;

    TxPayload[0] = 0x1E; //上报信息命令：ID 和电压采样值
    TxPayload[1] = TagInfo.id.id8[0];  //IDL
    TxPayload[2] = TagInfo.id.id8[1];  //IDH
    TxPayload[3] = TagInfo.CellVoltageH; //电压采样值高字节
    TxPayload[4] = TagInfo.CellVoltageL; //电压采样值低字节

    //计算累加和
    for(i=0;i<(MAX_TX_PAYLOAD-1);i++)fcs += TxPayload[i];
    TxPayload[MAX_TX_PAYLOAD - 1] = (256 - fcs)%256;
}

/*********************************************************************
 * 描  述 : 主函数
 * 入  参 : 无
 * 返回值 : 无
*********************************************************************/
void main()
{
    uint8_t RfReceLen;
    xdata   uint32_t  loopCount = ADC_TIME-1;

    TagInfo.id.id16 = TAG_ID;

    IoInit();
    mcu_init();

    #ifdef DEBUG_UART
    hal_uart_init(UART_BAUD_57K6);  //初始化 UART，波特率 57600
    while(hal_clk_get_16m_source() != HAL_CLK_XOSC16M) //等待时钟稳定
      ;
    #endif
```

```
    adc_init();
    RfCofig();

    #ifdef  USE_WDT
    hal_wdog_init(WDT_TIME);//配置看门狗超时时间 2s，使能看门狗
    #endif

    while(1)
    {
       loopCount++;

       #ifdef  USE_WDT
       hal_wdog_restart(); //喂狗
       #endif

       //启动后执行一次 AD 检测，以后，每 2 小时检测一次
       if(loopCount == ADC_TIME)
       {
          hal_adc_start();          //启动 ADC
          while( hal_adc_busy())     //等待 ADC 转换结束
            ;
          TagInfo.CellVoltageH = hal_adc_read_MSB(); //读取 ADC 采样值
          TagInfo.CellVoltageL = hal_adc_read_LSB();
          loopCount=0;
        }

        #ifdef DEBUG_LED
        D1 = ~D1;
        #endif

        PWRDWN = 0x04;    // 进入寄存器维持低功耗模式
        PWRDWN = 0x00;

        Assemble_Data();  // 数据打包
        hal_nrf_write_tx_payload(TxPayload,MAX_TX_PAYLOAD);

        CE_PULSE();            //无线发射数据
        radio_busy = true;
        while(radio_busy)          //等待操作完成
         ;
    }
}
```

7.2.2. Reader

```
/*********************************************************************
 * 描  述 : Timer0 初始化
 * 入  参 : 无
 * 返回值 : 无
*********************************************************************/
void Timer0Init(void)
{
  TMOD = 0x01;                    //16 位定时器
  TH0  = (65536-TIMER0_VALUE)/256;   //写入初值
  TL0  = (65536-TIMER0_VALUE)%256;
  ET0  = 1;      //使能 Timer0 溢出中断
  TR0  = 1;      //使能 Timer0 中断
}

/*********************************************************************
 * 描  述 : 标签 ID 入缓存
 * 入  参 : 无
 * 返回值 : 无
 *********************************************************************/
void ID_Inbuf(void)
{
   uint8_t i,idx;
   xdata uint8_t iswrite,lastcnt = 0,row = 0;

   idx = 0xFF;

   if(ID_BufTmp[0] == CMD_TAG_REPID)
   {
      iswrite = true;

      for(i=0;i<MAX_TAG_BUFNUM;i++)
      {
        if(ID_Buf[i][0] != 0)
        {
          if((ID_Buf[i][1] == ID_BufTmp[1]) && (ID_Buf[i][2] ==
              ID_BufTmp[2]))//标签 ID 存在
          {
              //未超过 20 秒收到相同的标签号不上报
              if(ID_Buf[i][0] < ID_OVERTIME)
              {
                 iswrite = false;
```

```
                    break;
                }
                else
                {
                    ID_Buf[i][0] = TAG_NEED_REP;
                    iswrite = false;
                    break;
                }
            }
            else
            {
                if(ID_Buf[i][0]>lastcnt)
                {
                    lastcnt = ID_Buf[i][0];
                    row = i;
                }
            }
        }
        else
        {
            if(idx == 0xFF)idx = i;
        }
    }
    if(iswrite == true) //ID 需要写入缓存
    {
        if(idx < MAX_TAG_BUFNUM)
        {
            for(i=0;i<(RX_PAYLOAD_LEN-2);i++)ID_Buf[idx][i+1] =
            ID_BufTmp[i+1];
            ID_Buf[idx][0] = TAG_NEED_REP;
        }
        else
        {
          if(row < MAX_TAG_BUFNUM)
          {
            for(i=0;i<(RX_PAYLOAD_LEN-2);i++)ID_Buf[row][i+1] =
            ID_BufTmp[i+1];
            ID_Buf[row][0] = TAG_NEED_REP;
          }
        }
    }
  }
}
```

```
/*****************************************************************************
 * 描  述 : 主函数
 * 入  参 : 无
 * 返回值 : 无
*****************************************************************************/
void main()
{
  uint8_t i;

  IoInit();

  ClockInit();
  RfCofig();

  hal_uart_init(UART_BAUD_57K6);  //初始化 UART，波特率 57600

  while(hal_clk_get_16m_source() != HAL_CLK_XOSC16M) //等待时钟稳定
    ;
  Timer0Init();

  for(i=0;i<MAX_TAG_BUFNUM;i++)ID_Buf[i][0] = 0;
  #ifdef  USE_WDT
  hal_wdog_init(WDT_TIME);
  #endif


  while(1)
  {
     #ifdef  USE_WDT
     hal_wdog_restart();
     #endif

     if(RF_Recv_Flag == true)//接收到 Tag 信息
     {
        #ifdef DEBUG_LED
        D1 = ~D1;
        #endif

        RF_Recv_Flag = false;//接收有效标志清零
        ID_Inbuf();          //ID 信息写入缓存
     }

     if(SecondFlag == true)//1 秒定时时间到
     {
```

```
            SecondFlag = false; //清零秒定时标志
            Uart_PackAndRepDat();//串口上报数据
        }
    }
}
```