迅通科技

nRF24LE1 无线监控通信开发文档

第一章 项目背景概述

本无线通信设计应用于家庭安防监控系统。如图 1.1 所示，该系统以主机为中心，多个传感器及摄像头等设备与主机构成一个星形的网络结构。他们每个节点都配有一个无线收发模块 nRF24LE1，主机也有无线收发模块，因此主机与各个设备之间都是无线通讯的，不需要布线来进行连接，系统维护和管理都不需要太多的外部干预，这样使得系统更加简便安全。

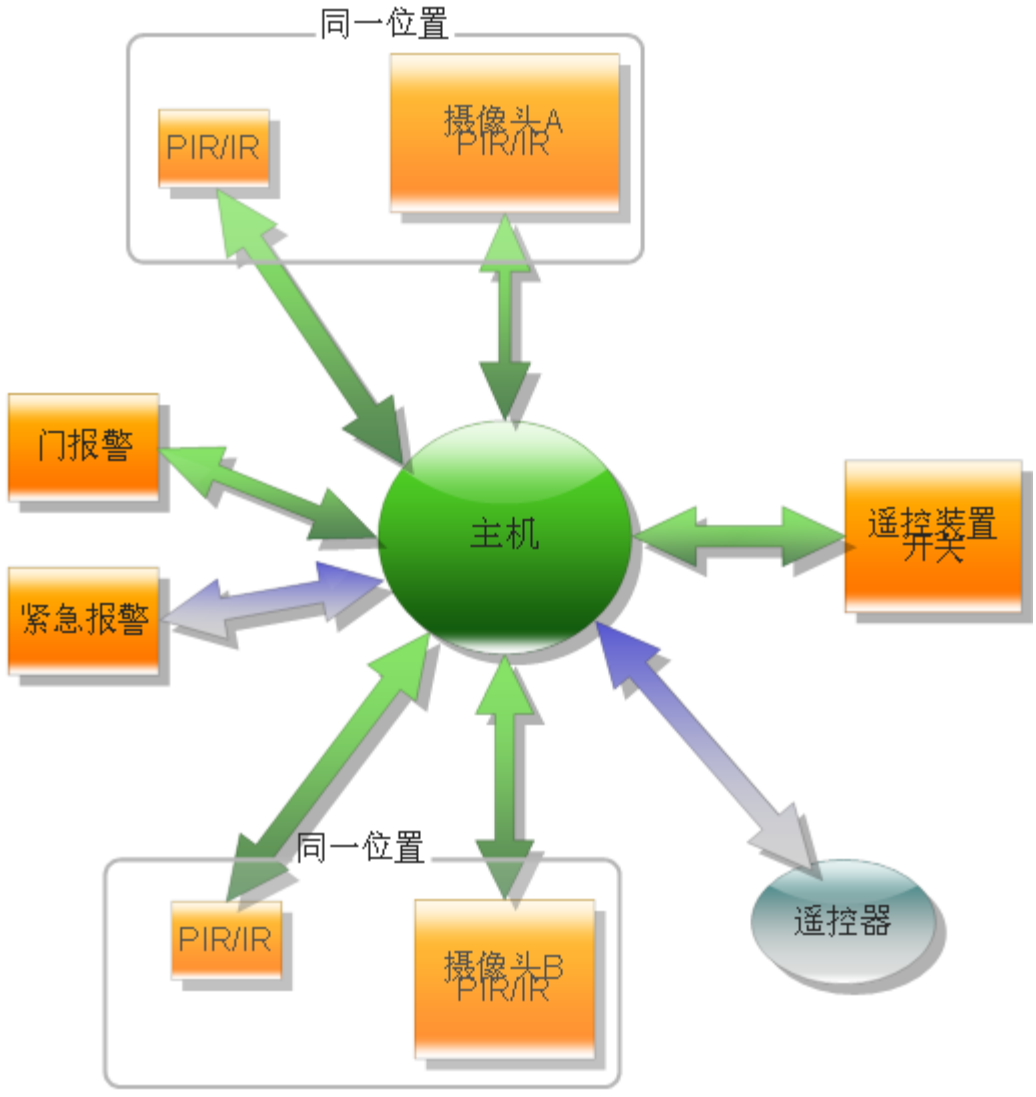

图 1.1 无线通信系统框图

各模块之间的通信关系：

- 主机与摄像头之间： ①向主机申请加入网络，主机收到加入申请后，回复加入网络成功信号；②主机发送拍摄照片命令，摄像头收到指令后按指令要求向主机发送图片数据；③当传感器被触发后，向主机发送触发信号；④定时的网络维持。
- 主机和 PIR/IR 之间：①向主机申请加入网络，主机收到加入申请后，回复加入网络成功信号；②当传感器被触发后，向主机发送触发信号；③接受到主机命令，工作于闪光（5s）或长亮或关的模式；④定时的网络维持。
- 主机和门警报之间： ①向主机申请加入网络，主机收到加入申请后，回复加入网络成功信号；②当被触发后，向主机发送触发信号；③定时的网络维持。
- 主机和遥控开关之间：①向主机申请加入网络，主机收到加入申请后，回复加入网络成功信号；②接受主机的命令，控制 GPIO 的变化或者发送给主机自己的 GPIO 状态；③定时的网络维持。
- 主机和遥控器之间：①第一次，手动长按遥控器一按钮和主机进行配对，把自己的 MAC 地址发送给主机以供绑定；②发送对主机的几条控制命令：布防；撤防。
- 主机和紧急报警之间：①第一次，手动长按紧急报警一按钮和主机进行配对，把自己的 MAC 地址发送给主机以供绑定；②发送对主机的报警命令。

第二章 nRF24LE1 背景知识

2.1 nRF24LE1 的硬件架构介绍

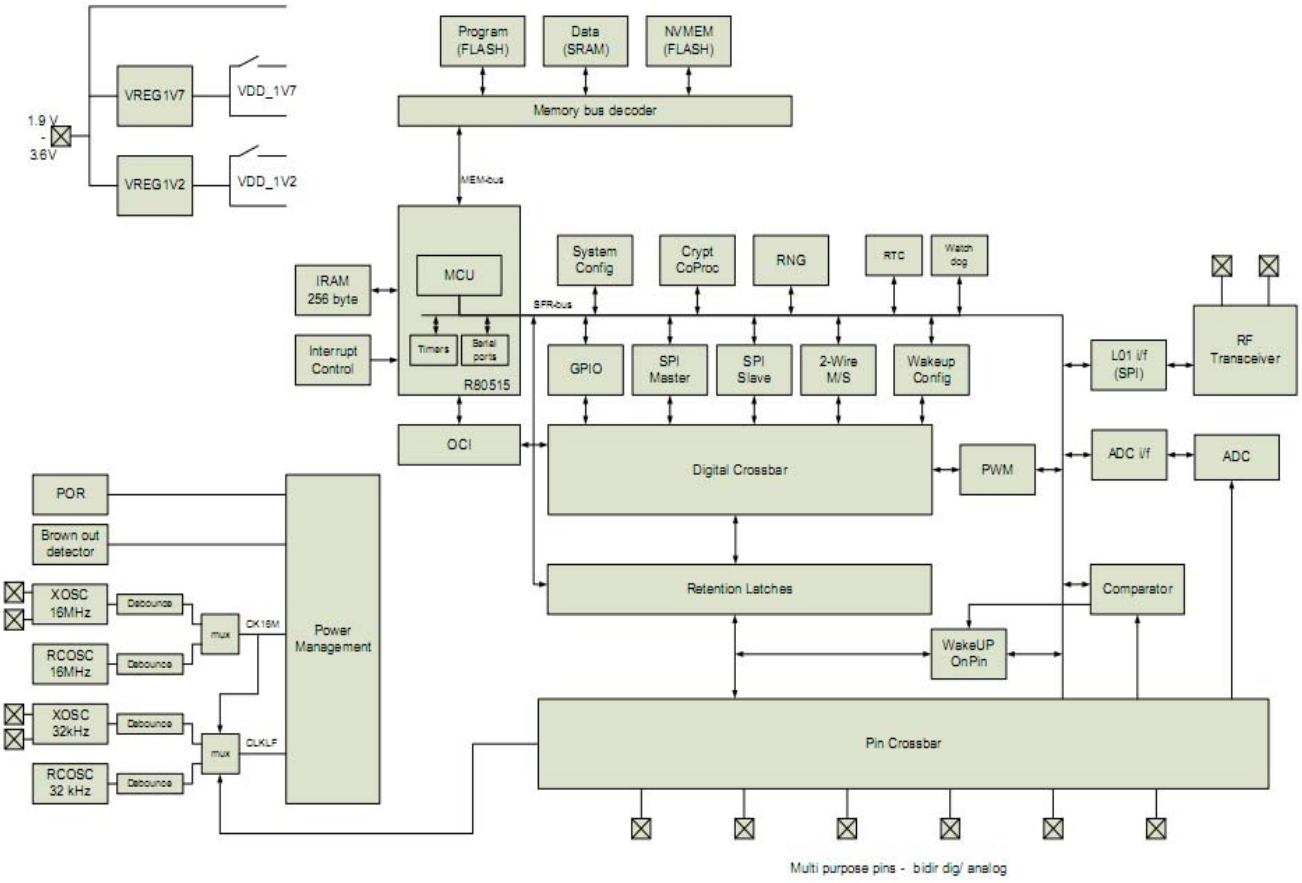

图 2.1 nRF24LE1 硬件架构

如图 2.1 所示，即为 nRF24LE1 的硬件架构。从功能模块上，可以大体分为 2 个部分：一是增强型的 8051MCU；二是 nRF24L01+2.4G 射频收发器。两个部分通过 SPI 接口进行通信。

nRF24LE1 提供三种不同的封装：4mm*4mm 24 引脚 QFN 封装（7 个通用 I/O）；5mm*5mm 32 引脚 QFN 封装（15 个通用的 I/O）；7mm*7mm 48 引脚的 QFN 封装（31 个通用的 I/O）。不同引脚的封装，除了 IO 口的数量不同外，

在功能上也有一定的区别。本设计采用的为 5mm*5mm 32 引脚 QFN 封装（15 个通用的 I/O）。

2.2 nRF24L01+2.4G 射频收发器介绍

2.2.1 射频收发器架构介绍

射频收发器工作与国际 ISM 频段 2.400~2.4835GHz。射频收发内核的配置通过射频收发器的寄存器映像进行，寄存器由 MCU 通过双向的片内 SPI 接口来访问，并可在各种节能模式下工作。

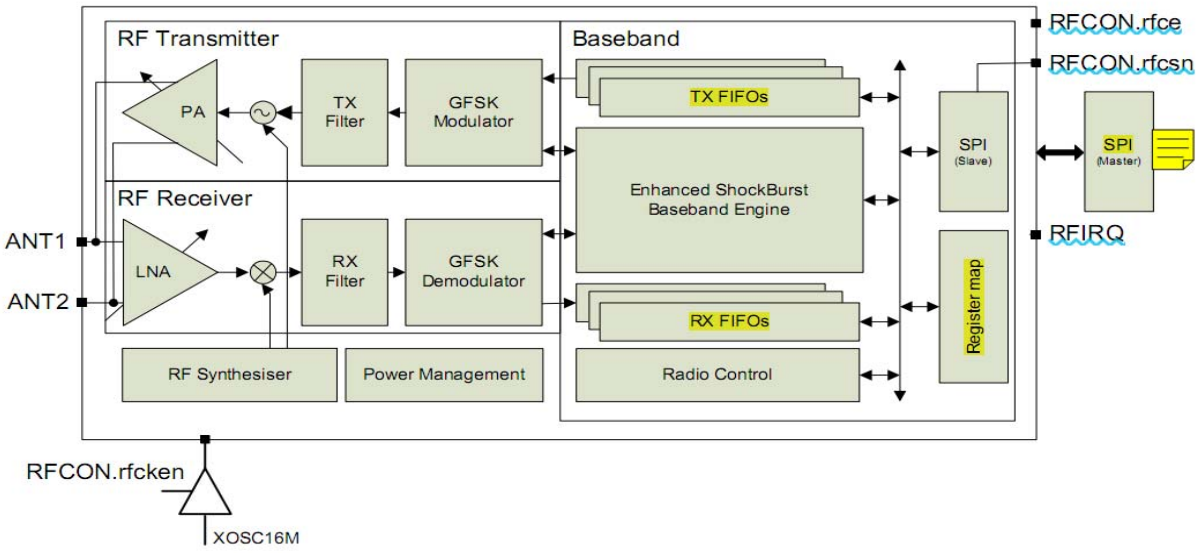

图 2.2 射频收发器框图

如图 2.2 所示，即为射频收发器内部框图。射频收发器通过 SPI 接口与 MCU 通信；MCU 通过三个接口（RFCON.rfce，RFCON.rfcsn，RFIRQ）对射频收发器进行控制；register map 为寄存器映射，用于保存 MCU 对于射频收发的配置；TX FIFOs、RX FIFOs 分别用于存储待发送和接收到的数据包。

2.2.2 射频收发功能说明

2.2.2.1 工作模式

射频收发器可配置为四种工作模式：掉电模式；待机模式；接收模式；发射模式。通过配置 CONFIG 寄存器的 PWR_UP 位、PRIM_RX，rfce，rfcsn，可以改变射频收发器的工作模式。具体配置可参见表 2.1。

Mode	PWR_UP register	PRIM_RX register	rfce	FIFO state
RX mode	1	1	1	-
TX mode	1	0	1	Data in TX FIFO. Will empty all levels in TX FIFO[a].
TX mode	1	0	Minimum 10µs high pulse	Data in TX FIFO.Will empty one level in TX FIFO[b].
Standby-II	1	0	1	TX FIFO empty
Standby-I	1	-	0	No ongoing packet transmission
Power Down	0	-	-	-

表 2.1 射频收发器工作模式

2.2.2.2 空中速率

空中速率指的是在发射和接收时，射频收发器使用的已调制的信号速率。对于 nRF24LE1,通过设置 RF_SETUP 寄存器中的 RF_DR，可将空中速率设定为 250kbps，1Mbps 或 2Mbps。使用高速率可以获得较低平均电流从而减少空中受干扰和碰撞机会；使用较低的速率将会获得更好的接收灵敏度。接收方和发送方必须设定为同一速率方可互相通信。

2.2.2.3 射频频道频率的设定

射频频道的频率决定射频收发器所使用频道的中心频率。在速率为 250kbps 或 1Mbps 时，频道占有的带宽小于 1M；在 2Mbps 时，所占带宽小于 2M。射频收发器工作的频率范围在 2.400~2.525GHz。无线频道设置的频率分辨率为

1MHz。

无线频率由 RF_CH 寄存器的内容确定，可由以下公式计算得出：

F0=（2400+RF_CH）MHz

通过改变频率，可以实现跳频等功能。为确保相互通信，发射器和接收器须在同一个频率。

2.2.2.4 接收功率检波测量

接收功率检波（RPD），位于寄存器 09 的位 0，在当前无线频道上接收到的功率电平高于-64dBm 就会置该位为 RPD=1；否则，RPD=0。

在接收模式下，RPD 可以随时被读出。只要收到一个包或者 rfce 为低，RPD 将清零。

通过 RPD 检测，可以实现载波监听等功能。

2.2.3 增强型 ShockBurst

增强型 ShockBurst 是由 ShockBurst 模式发展而来的。增强型 ShockBurst 模式是一个以包为基础的数据链路层，功能包括包的自动设定装配和装配时间，自动应答和自动重发。增强型 ShockBurst 能够完成低功耗和高性能的通信，能有效改善双向和非双向系统无线通信的能源效率，而无需在微处理器端进行复杂的操作。

增强型 ShockBurst 重要作用在于使得一个双向数据链路的可靠通信变得更加容易实现。一个增强型 ShockBurst 的包处理实际上是在两个射频收发器之间进行的包的交换，一个射频收发器作为主接收（PRX），另一个射频收发器作为主发送(PTX)。包的自动处理过程如下：

1 先从 PTX 发送一个数据包到 PRX，增强型 ShockBurst 将设置 PTX 为接收并等待 ACK 包。

2 如果数据包被 PRX 所接收，增强型 ShockBurst 将自动装配并发送一个应

答包（ACK 包）给 PTX 然后返回接收模式。

3 假设 PTX 没有即时收到 ACK 包，增强型 ShockBurst 将会在一个可编程的延迟时隔后自动重发此数据包，然后将 PTX 设置为接收模式等待 ACK 包。

在增强型 ShockBurst 中可以设定重发的次数和重发的间隔参数，而后所有的工作均由增强型 ShockBurst 自动完成而无需 MCU 的干预。

增强型 ShockBurst 包格式如表 2.2 所示。

前置域 1 byte	地址域 3~5	包控制域 9 bits	载荷 0~32	CRC 1~2

表 2.2 增强型 ShockBurst 包格式

其中，前置域为确保足够时间来稳定接收机处理；地址域为接收机地址；包控制域包含数据载荷长度、包标识符、无应答标志；载荷为用户所定义的包的内容；CRC 为数据包使用的错误检测机制。

采用增强型 ShockBurst，设置最大重发次数，可以减少 MCU 的相关操作，并提高数据发送成功率。

ShockBurst 除了有自动应答和自动重发功能外，在包格式中没有 9 位的控制域。

2.3 存储器相关

如图 2.3 和图 2.4 所示。MCU 各有 64kb 代码存储器（Data Space）和数据存储器（Code space）、256 字节内部数据存储器（IRAM）和 128 字节特殊功能寄存器(SFR)。

nRF24LE1 包括 16kb 可擦写存储器 Flash、1kb 数据存储器 SRAM、两块 Flash 非易失存储器（1kb 标准擦写循环周期和 512 字节更长擦写受命周期）。

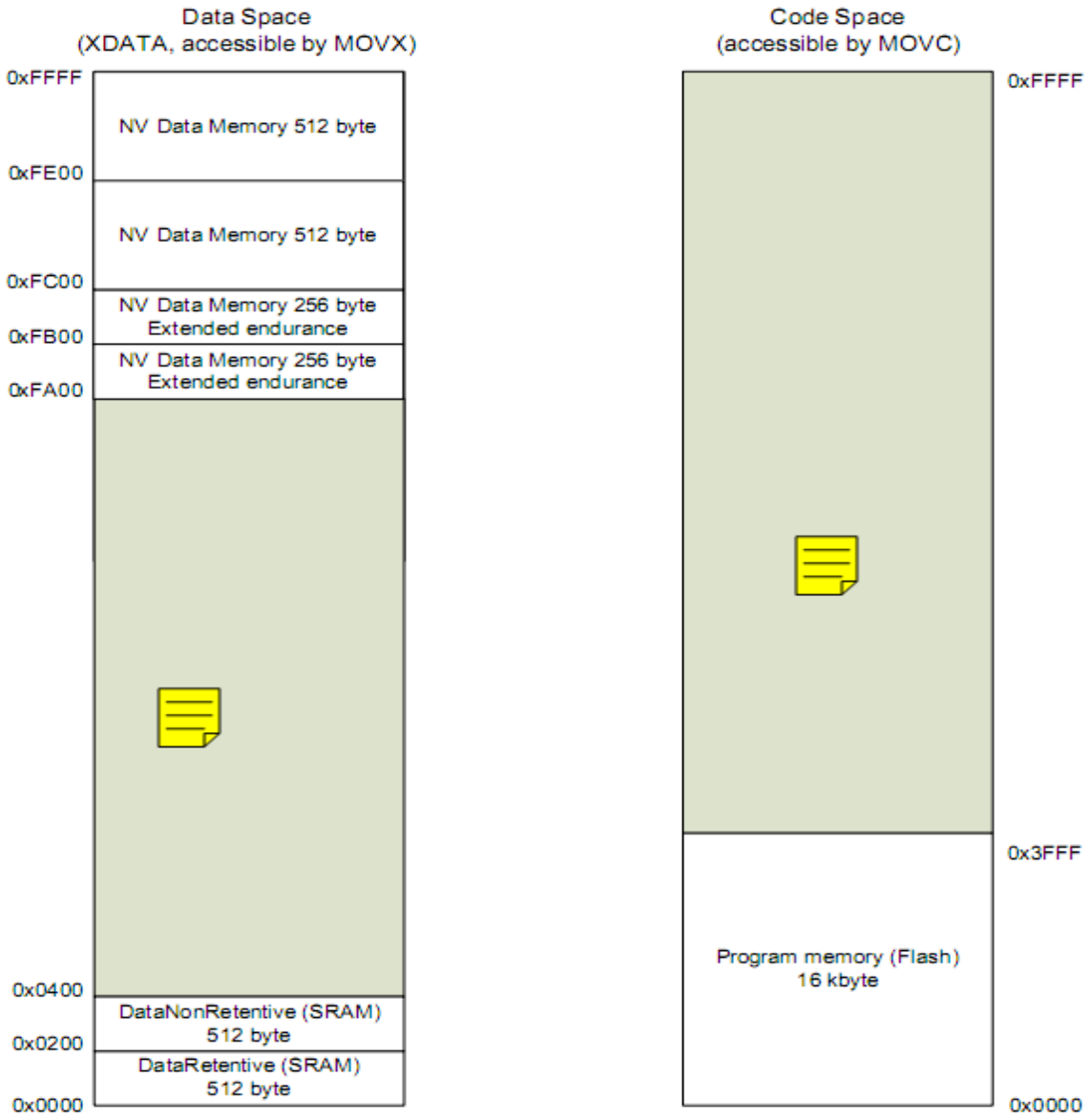

图 2.3 数据存储器和代码存储器

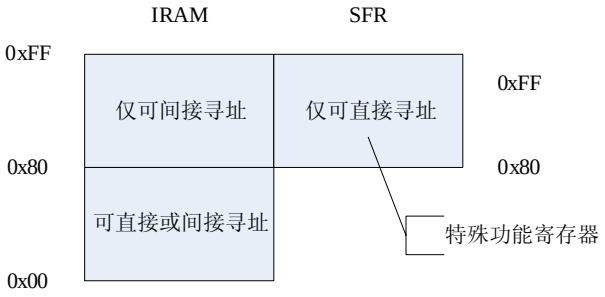

图 2.4 内部数据存储器和特殊功能寄存器

第三章 软硬件开发平台搭建

3.1 硬件平台介绍

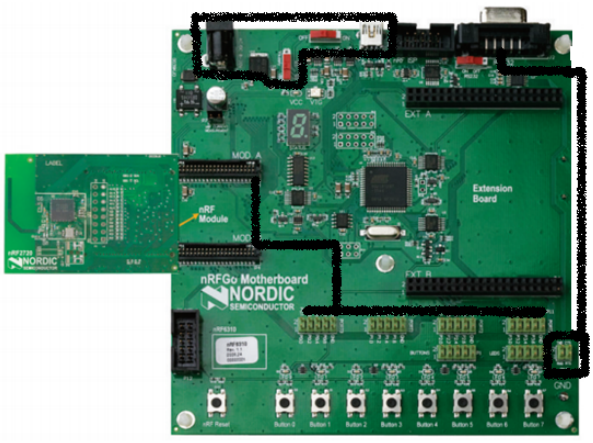

图 3.1 nRF24LE1 nRFgo DK

开发平台采用如图 3.1 所示的 nordic 公司 nRF24LE1 nRFgo DK。主要由两部分组成：具有 PCB 天线的 nRF24LE1 开发模块 nRF2722，提供电源和扩展连接功能的母板 nRF6310。本项目开发中，共使用了 2 个 nRF2722 和 2 个 nRF6310。

左侧 nRF2722 通过插槽与母板 nRF6310 相连接；母板 nRF6310 下部的倒数第二排引脚与 nRF24LE1 的 IO 口对应相连；母板 nRF6310 右下角的四个引脚接到上部的串口（在子模块部分会详细介绍）；最下边八个为按键，与引脚最下排左边八个对应相连；按键上部为八个 LED 灯，与引脚最下排中间八个对应相连；上部所框为电源接口部分，母板有三种供电方式：一是通过 USB 供电，同时也是下载线；二是板子背部的三节七号干电池供电；三是通过直流供电，此方式须切换 S8 开关。对于供电方式的建议：开发时采用 USB 供电，在烧写程序的同时还可以供电；在需较远距离测试时，可采用高容量干电池供电。

3.2 软件平台介绍

在搭建好硬件开发平台后，还要建立好开发应用的软件平台。软件部分可分

为三部分，其中 SDK 和 nRFgo Studio 是 nordic 公司的配套软件，Keil C51 uVision3 为第三方软件。

3.2.1 软件开发包 SDK

Software Development Kit (SDK)是 nordic 公司配套的，包括关于芯片开发的参考代码以及文档，安装以后源代码在相应的目录下，文档说明以超文本方式提供。

nRFgo SDK：

- 包括 1nRF SOC 器件的通用软件开发平台；
- 包含器件操作、协议堆栈和应用函数的软件代码库。

本设计中采用了 v2.0 版本安装，如图 3.3 所示。

在安装后的文件夹中，Documentation 文件夹里 nrfgo_sdk_help.chm 文件为帮助文件，如图 3.5 所示。

图 3.4 SDK 安装后文件夹

在文件夹 Source_Code 中的可以找到相应的源代码，在其中的 projects 文件夹下可以找到相应的 demo 例程的 keil C51 工程。

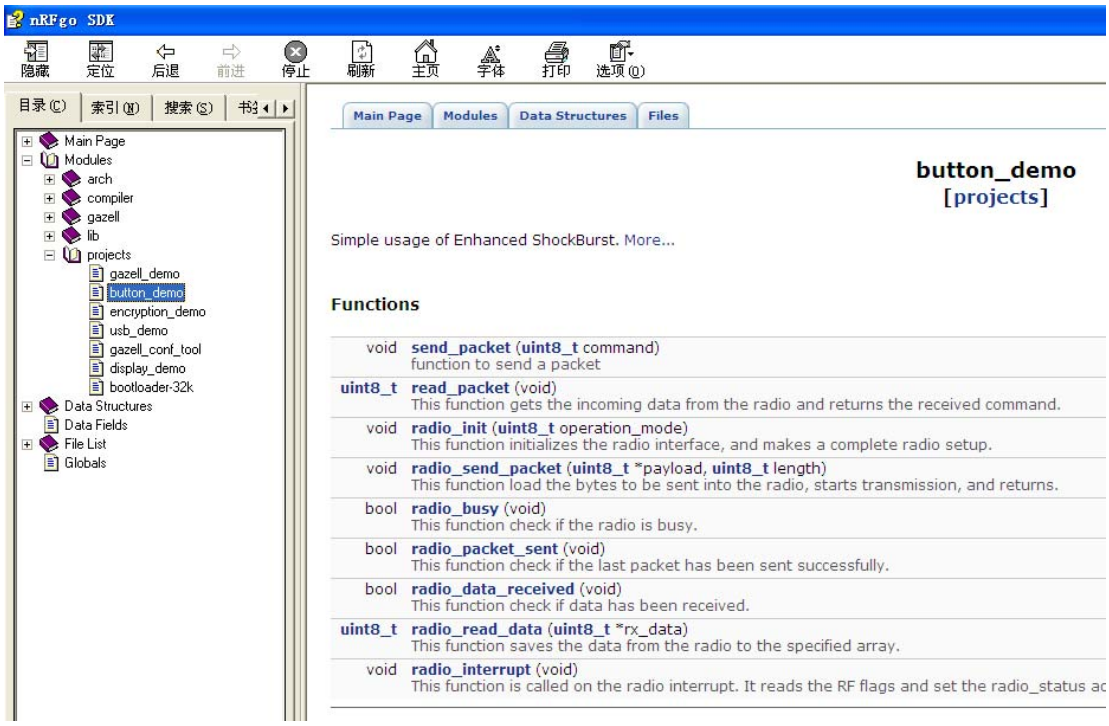

图 3.5 nrfgo_sdk_help.chm

3.2.2 nRF 软件综合环境

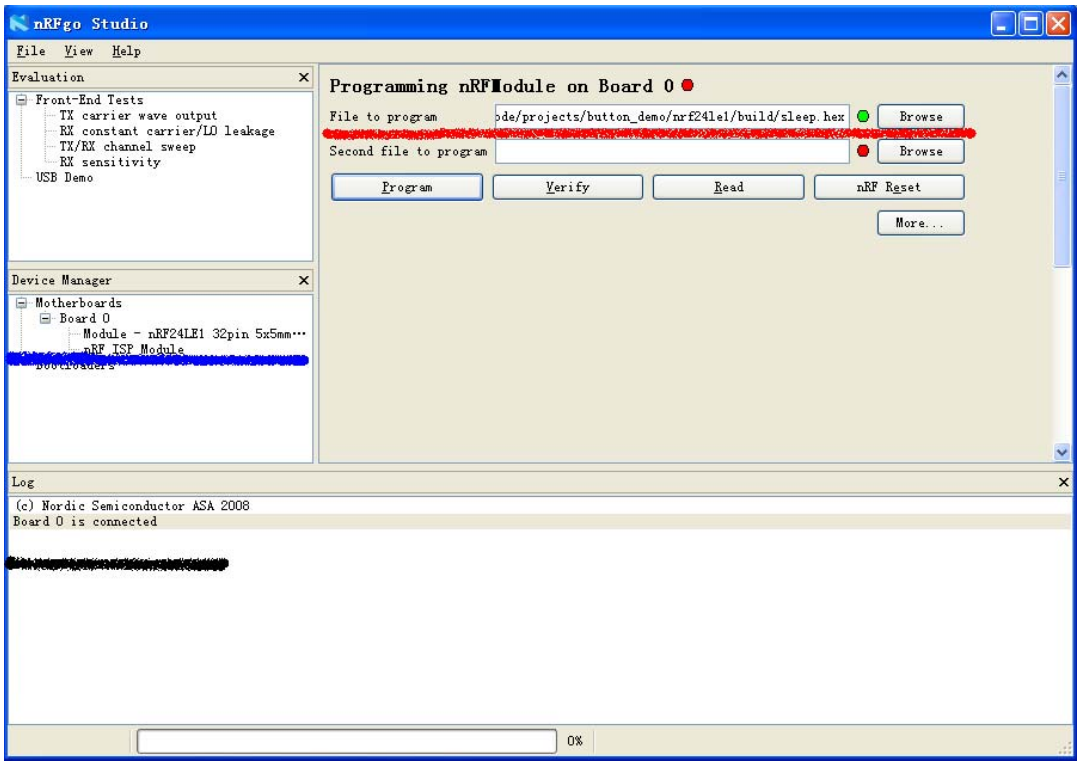

图 3.6　nRFgo Studio 操作界面

nRFgo Studio 是一个 PC 端的应用程序，用来控制及管理开发平台，具有评估及测试功能，并可进行 Flash 在线编程。

如图 3.6 所示为 nRFgo Studio 软件操作界面，红色部分为选择要下载的.Hex 文件路径；蓝色部分为设备识别控制部分；黑色部分为信息显示区，可以显示连接或断开设备，验证、下载成功之类信息。

通过 USB 连接开发板与电脑，打开 nRFgo Studio，即可以在 Decice Manager 部分发现到设备 boards 0，同时，开发板上的 LED 也会显示 0，同理，在接入第二个母板时，会发现设备 boards 1 等。

3.2.3 nRFprobe——在线仿真软件调试工具

nRFprobe 具有一下特征：

- 为 nRF SOC 设计的 Flash 编程和硬件调试的综合工具；
- 与 keil uVision 完全无缝连接；
- 集成于 nRFgo 平台；
- 包含在 nRFgo 开发目标板套件中。

若不需要硬件调试不必安装此软件。

3.2.4 软件开发平台

本设计使用的软件开发平台为 Keil C51。Keil C51 是美国 Keil Software 公司出品的 51 系列兼容单片机 C 语言软件开发系统。

本设计开发使用的 keil C51 版本为 uVision3 V3.30。如图 3.7，即为 keil C51 uVision3 的操作界面。

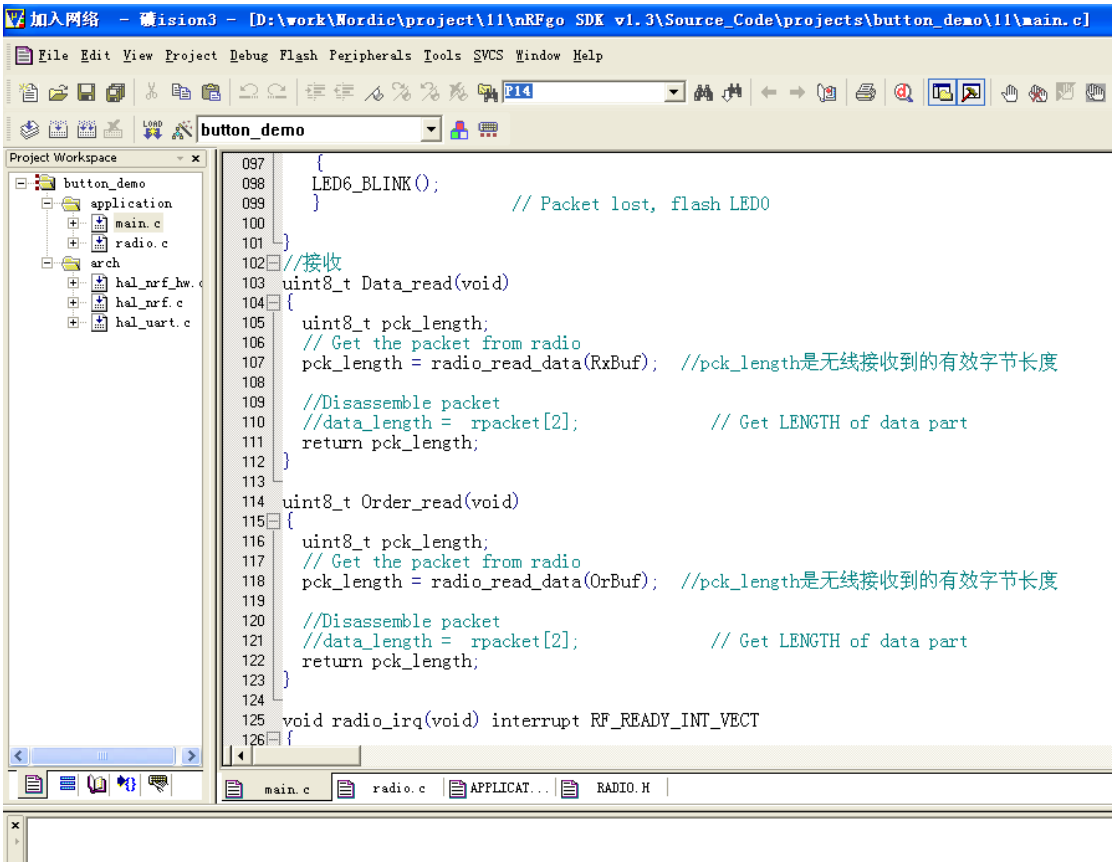

图 3.7 keil C51 uVision3 操作界面

注意：要确保 stdint.h 和 stdbool.h 头文件已经在..\Keil\C51\INC 及

reg24le1.h 在..\Keil\C51\INC\Nordic。否则工程将链接出错。在 Keil C V8.47 及其以后版本中，nRF24LE1 已经列入其型号表，这些文件应该在相应的目录中。

3.3 nRF24LE1 DK Getting Started Guide

第四章 子模块代码解析

本设计采用可读性好，移植容易并普遍使用的 C 语言进行开发。nRF24LE1 内置增强型的 8051，因此，大部分针对 8051 的 C 代码可以加以利用。

本设计采用的为 5mm*5mm 32 引脚 QFN 封装（15 个通用的 I/O），因此以此型号为例进行介绍。本章节代码均在 keil C51 下编译通过，并在 nRF24LE1 上验证成功。

4.1 I/O 口的使用

nRF24LE1 的每个 I/O 引脚具有 MCU 通用的 I/O 引脚功能包括以下功能(x 取 0、1、2、3，为四组 I/O)：

- 数字或模拟
- 配置方向：寄存器 PxDIR 做输入和输出方向控制
- 配置驱动能力，配置上拉或者下拉：寄存器 PxCON 控制每个引脚的驱动能力，以及上拉和下拉电阻。

5mm*5mm 32 引脚具有 15 个通用的 I/O，如表 4.1 所示。注意：不同封装的芯片的管脚功能有一定的区别。

管脚复用：

- 对于默认连接：例如表 4.1 中 P0.4~P1.0 的每个引脚列出有两个系统输入，这意味着该引脚若已配置为输入则将控制两个模块的输入，即这两

个功能通过一个与门结合同时起作用。

1) 对于动态使能连接：端口交叉模块可以根据系统需要实时修改与芯片的外设模块（如 SPI，I2C 等）的动态连接，在选用较小封装时，引脚的分配可能会发送冲突，这可以通过设置每个外设模块的优先级来解决。

pin	Default connections		Dynamically enabled connections																	
	Inputs	Outputs	XOSC32K		SPI Master		Slave/Flash SPI		PWM		ADC/COMP		HW Debug		2-Wire					
			priority 1		priority 2		priority 3		priority 4		priority 5		priority 6		priority 7					
P1.6	p1Di 6	p1Do 6			MMISO	in														
P1.5	p1Di 5	p1Do 5			MMOSI	out														
P1.4	p1Di 4	p1Do 4			MSCK	out														
P1.3	p1Di 3	p1Do 3											OCITO	out						
P1.2	p1Di 2	p1Do 2									AIN10	ana	OCITDO	out						
P1.1	p1Di 1	p1Do 1					SCSN	in			AIN 9	ana	OCITDI	in						
							FCSN[a]	in												
P1.0	p1Di 0	p1Do 0					SMISO	out			AIN 8	ana	OCITMS	in						
	TIMER1						FMISO[a]	out												
P0.7	p0Di 7	p0Do 7					SMOSI	in			AIN 7	ana	OCITCK	in						
	T0						FMOSI[a]	in												
P0.6	p0Di 6	p0Do 6									AIN 6	ana								
	GPINT1																			
P0.5	p0Di 5	p0Do 5					SSCK	in			AIN 5	ana			W2SDA	ino ut				
	GPINT0						FSCK[a]	in												
P0.4	p0Di 4	p0Do 4									AIN 4	ana			W2SCL	ino ut				
	UART/ RXD																			
P0.3	p0Di 3	p0Do 3							PWM1	out	AIN 3	ana								
		UART/ TXD																		
P0.2	p0Di 2	p0Do 2							PWM0	out	AIN 2	ana								
P0.1	p0Di 1	p0Do 1	CLKLF [b]								AIN1	ana								
P0.0	p0Di 0	p0Do 0	CLKLF [c]	ana							AIN0	ana								

表 4.1　32 引脚映射表

例如令 P0DIR = 0x17，按表 4.1，可以得出设置结果为：P0.7、P0.6、P0.5、P0.3 设置为输出；P0.4、P0.2、P0.1、P0.0 设置为输入。其中，P0.4 为输入，P0.3 为输出，则默认连接 UART/RXD 和 UART/TXD 起作用，接通相应电路后，就可以用做串口通信。

4.2 UART 子模块

串口由寄存器 S0CON 控制，而实际的数据传送可读/写 SBUF 寄存器。数据传送速率（波特率）S0RELL、S0RELH 和 ADCON 寄存器来选择。

实现串口通信步骤：

✓ 1 要在母板上相应的线连接起来，如图 4.1 所示，并确定上部的串口开

关置于开启状态。

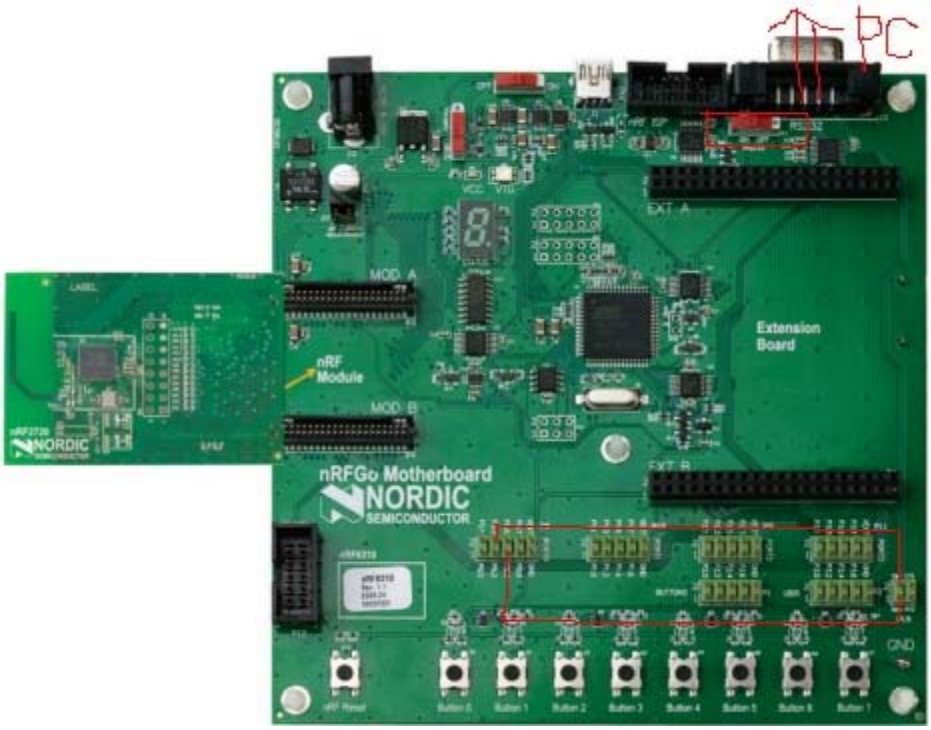

图 4.1 串口硬件连接图

✓ 2 配置相应的 I/O 口。配置 P0.4 为输入，P0.3 为输出，即可。

✓ 3 通过调用 SDK 中的 hal_uart.c 文件，可以容易的实现串口通信。首先调用函数 hal_uart_init(hal_uart_baudrate_t baud)初始化串口，然后就可以通过 hal_uart_getchar()和 hal_uart_putchar(uint8_t ch)进行串口通信了。

代码示例：

```
#include "hal_uart.h"
Main()
{
…
P0DIR = 0x17;
…
hal_uart_init(4);   //采用方式 1,波特率为 9600
```

```
…
hal_uart_getchar(); //串口接收一个字节
hal_uart_putchar(0xFF);//串口输出一个十六进制 FF
}
```

4.3 实时钟 RTC

RTC2 包含两个寄存器用来捕获定时器的值，一个由 32.768kHz 时钟的下降沿驱动；另一个由 MCU 的时钟驱动来获得更好的分辨率。两个寄存器均可由外部事件的结果来更新。当定时器和比较器寄存器预定义周期的值相同时，RTC2 产生一个中断，RTC2 确保唤醒的功能优于中断。通过 RTC 定时，可以用于芯片掉电休眠的周期性唤醒源。

例程

```
//*************************************
//功能：RTC2 定时器初始化子程序
//*************************************
void rtc2_init(void)
{
    CLKLFCTRL=0x01;                    // 使用 32K RC 时钟
    RTC2CMP0=0xff;                     // 定时时间 2 秒
    RTC2CMP1=0xff;
    RTC2CON=0x07;                      // 比较模式，启动 RTC2
    WUIRQ=1;                           // 使能 TICK 中断
}
//*************************************
//功能：RTC2 中断服务子程序
//*************************************
void RTC2_IRQ(void) interrupt INTERRUPT_TICK
```

```
{
        LED0= ！ LED0; //需要实现的 RTC 中断功能
}
```

或者调用 SDK 中的 hal_rtc.c 文件。

4.4 跳频子函数

在跳频通信过程中主要是如何实现接收端和发送端在改变频道的过程中频道的统一，而且在频道转换过程中应当尽可能地少花费时间。nRF24LE1 的频道转换时间可以满足这个要求。下面来如何从软件上实现跳频。

首先，要编辑一个随机数表：

```
const uint8_t code
FreBuf[12]={0x28,0x49,0x38,0x07,0x19,0x17,0x2f,0x1f,0x14,0x16,0x06,0x33};
```

这是一个数组，称为跳频频道表，而更换频道就是用这个数组中的每个值作为一个频道，通过迅速的改变这些值实现跳频通信。跳频频道表一般不采用 1, 2, 3, 4……的顺序安排，因为如果一个信道受到干扰，那么与之相邻的信道很有可能也受到干扰，考虑到 W-LAN 的信道带宽为 22MHz，可以用公式 i+j×22 (j=1, 2, 3, 4, 5; i=1, 2, 3) 的顺序安排跳频频道表。

```
//***********************************
//功能：发送端改变频道
//***********************************
void Fswitchchannel(uint8_t fre)
{
 CE_LOW();
 hal_nrf_set_rf_channel(fre);
 //CE_HIGH();
 hal_nrf_set_power_mode(HAL_NRF_PWR_UP);
 radio_status = RF_IDLE;
}
```

```
//***********************************
//功能：接收端改变频道
//***********************************
void Jswitchchannel(uint8_t fre)
{
 CE_LOW();
 hal_nrf_set_rf_channel(fre);
 CE_HIGH();
 radio_status = RF_IDLE;
}
```

在这里只用到了 12 个频道。其实在实际应用中可以适当的增加或减少所用的频道。

4.5 发送和接收子模块

发送和接收模块是 nRF24LE1 的一个重要功能模块。本模块可通过调用 SDK 中的 radio.c 文件来实现。

示例代码如下：

```
#include "radio.h"
//***********************************
//功能：发送一个数据包
//***********************************
void send_packet(uint8_t command)
{
bool packet_sent;
uint8_t data_length;
// Assemble packet
packet[0] = CMD_SEND_DATA;     //相当于帧头吧  Add TYPE (data =
                                0x00) //CMD_SEND_DATA=0:send a
                                new command
```

```
                                        //可以把第一帧设为地址（本机 ID 号）
 packet[1] = data_length = 1;          // Add LENGTH (1 byte)
 packet[DATA_POS] = command;     //Add DATA to
                                        send;DATA_POS=7 : Defines //the
                                        start-position of the data in packet
                                        //可以加和校验位和尾帧
 // Send packet with radio
radio_send_packet(packet, data_length+DATA_POS);
//void radio_send_packet(uint8_t *payload（待发数组的地址）, uint8_t
//length(总长度))
// Wait for respons from radio, TX_DS or MAX_RT
while(radio_busy());          //等到 nRF 产生中断，改变 radio_status 的值，
                                   才会//跳出 while
packet_sent = radio_packet_sent(); //改变 radio_status 为 RF_IDLE
    if(packet_sent)
    {
       LED1_BLINK();                          // Packet sent, flash LED1
    }
    else
    {
       LED0_BLINK();                          // Packet lost, flash LED0
    }
 }
 //************************************
 //功能：接收一个数据包，并放入指定的数组中
 //************************************
 uint8_t read_packet(void)
```

```
{
    uint8_t nrf_data, data_length,type, pck_length;
    // Get the packet  from radio
    pck_length = radio_read_data(packet);   //这是接受到的有效位的
长度
    // Disassemble packet
    type          =   packet[0];              // Get TYPE of message
    data_length =   packet[1];              // Get LENGTH of data
part
    // .... Room for decryption of message ....
    nrf_data = packet[DATA_POS];            // Get the received
nrf_data
    return nrf_data;
}
Main()
{
…
RFCKEN = 1;                          // enable L01 clock RF 部分的 CLK
使能
RF = 1;                             // enable RF interrupt
EA = 1;                             // Global interrupt enable
…
radio_init(x); //射频收发器初始化，x=0 则为接收模式；x=1 则为发送模式
//*************************************
//功能：若设为发送模式，发送一个数据包
//*************************************
while(radio_busy());     // Wait until radio ready
```

```
send_packet(CMD1);  //发送数据包
//*************************************
//功能：若设为接收模式时，检测到完整数据包后接收数据
//*************************************
if(radio_data_received())  //若收到完整的数据包
command = read_packet(); //读取数据包至指定数组
}
```

下面重点讲解一下 radio.c 文件。

```
//*************************************
//功能：射频收发器初始化
//*************************************
void radio_init(uint8_t operation_mode)
{
CE_LOW();                                   // Disable radio

hal_nrf_enable_ack_payload(true);          // Enable dynamic ack
hal_nrf_enable_dynamic_payload(true);     // Enable dynamic payload
hal_nrf_setup_dynamic_payload(0xFF);     // All pipes uses dynamic ack.

hal_nrf_close_pipe(HAL_NRF_ALL);          // First close all radio pipes
                                          // Pipe 0 and 1 open by default
hal_nrf_open_pipe(HAL_NRF_PIPE0, true);   // Open pipe0, with autoack

hal_nrf_set_crc_mode(HAL_NRF_CRC_16BIT); // Operates in 16bits
```

```
CRC mode
    hal_nrf_set_auto_retr(15, 250);              // 250   delay, 15
retransmits

    hal_nrf_set_address_width(HAL_NRF_AW_5BYTES); // 5 bytes address
width
    hal_nrf_set_address(HAL_NRF_TX, address);       // Set device's
addresses
    hal_nrf_set_address(HAL_NRF_PIPE0, address);   // Pipe0 used for
auto ACK
    //hal_nrf_set_datarate(hal_nrf_datarate_t datarate)//调用来设置传
                                                    输速率，默//认为 2Mbps
    if(operation_mode == 0)                  // Mode depentant settings
      {
      hal_nrf_set_operation_mode(HAL_NRF_PRX);      // Enter RX mode
      CE_HIGH();                                    // Enable radio
      }
      else                                        // Mode depentant
settings
      {
      hal_nrf_set_operation_mode(HAL_NRF_PTX);      // Enter TX mode
      }

    hal_nrf_set_rf_channel(40);              // Using channel 40 (2440MHz)
    hal_nrf_set_power_mode(HAL_NRF_PWR_UP);       // Power up
device
```

```
    radio_status = RF_IDLE;                         // Radio ready, i.e.
RF_IDLE
    }
    //****************************************
    //功能：射频收发器发送数据包子函数
    //****************************************
    void radio_send_packet(uint8_t *payload, uint8_t length)
    {
    hal_nrf_write_tx_payload(payload, length);      // Load message
into radio
    CE_PULSE();                                     // Send packet
    radio_status = RF_BUSY;                     // Trans. in progress;
RF_BUSY
    }
    bool radio_busy(void)
    {
      return(radio_status == RF_BUSY);
    }
    bool radio_packet_sent(void)
    {
      switch(radio_status)
      {
    case HAL_NRF_TX_DS:                         // Packet sent
        radio_status = RF_IDLE;                     // Return to
radio:RF_IDLE
        return true;
        break;
```

```
    case HAL_NRF_MAX_RT:                    // Packet lost
        radio_status = RF_IDLE;                 // Return to
radio:RF_IDLE
        return false;
        break;

    default:
        return false;
        break;
      }
    }
    //*****************************************
    //功能：射频收发器读出数据包子函数
    //*****************************************
    uint8_t radio_read_data(uint8_t *rx_data)     //This function saves
                                        the data //from the radio to the
                                        specified array.
    {
    uint8_t length = LSB(hal_nrf_read_rx_payload(rx_data));        // Get
                        received //data,pipe number (MSB byte) and
                        packet length (LSB byte)
                        // and bytes received
      if (hal_nrf_get_rx_fifo_status() == FIFO_EMPTY)          // Fifo is
empty?
      {
        radio_status = RF_IDLE;
```

```
    }
    return length;
}
```

第五章 无线抗干扰技术

5.1 2.4GHz ISM 频段分析

2.4GHz ISM 频段是全球开放频段，许多系统如 W- LAN、蓝牙等都共用这一频段，如图 5.1 所示。 nRF24LE1 的工作环境也许会是一个干扰很多的环境，系统往往在受控的实验室环境下工作得很好，但在现场却会由于受到其他工作在 2.4GHz 产品的影响而使性能显著下降。在应用 nRF24LE1 时如何处理其他设备的干扰是必须考虑的问题。本系统应用于家庭安防，因此系统的抗干扰能力尤为重要。

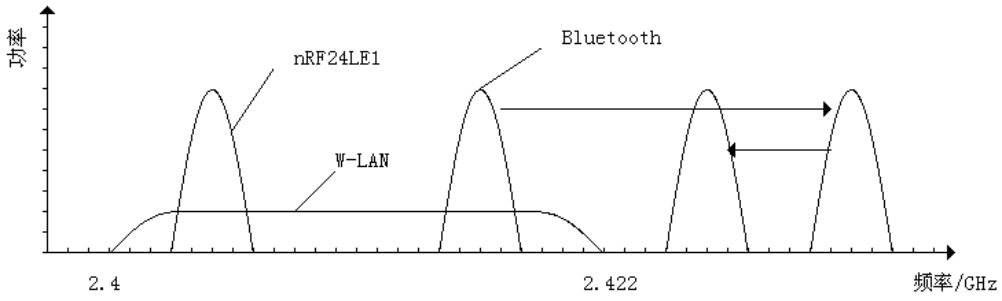

图 5.1 nRF24LE1、W-LAN和蓝牙的频谱分布图[4]

（注：选取 W-LAN 第一个信道的频谱）

工作在 2.4GHz 频段的无线设备的频道使用情况主要分为两种，一种是频率分布相对稳定的系统如 W-LAN 以及恶意同频干扰,另一种是跳频系统如蓝牙。

W- LAN 采用 DSSS (直接序列扩频)，其每信道带宽为 22MHz，故允许使用 3 个分布式信道而不会相互重叠,它将原信号 “ 1” 或 “ 0” 利用 10 个以上的 chips 代表 “ 1” 或 “ 0” ，使得原来较高功率、较窄频率变成具有

较宽频的低功率，因此 W- LAN 对其他设备而言产生的是在某些频率段相对稳定的干扰。

蓝牙技术采用 FHSS (跳频扩频) 并将 2.4GHz ISM 频段划分成 79 个 1MHz 的信道。蓝牙设备以伪随机码方式在这 79 个信道间每秒钟跳 1600 次，它同时接受两端以特定型式的窄频载波来传送讯号。对于一个非特定的接收端，FHSS 所产生的跳动讯号对它而言，只能算是脉冲噪声而已。

nRF24LE1 的信道带宽同样是 1MHz,它将整个 2.4GHz ISM 频段分为 125 个有效信道。频率稳定的系统在一定的频段工作，而跳频系统在整个 2.4GHz 频段范围都会产生影响。在 2.4GHz 频段工作的系统其数据发送都是基于数据包的。如果跳频系统在某个时刻占用了某频道，那它在其他时间占用整个频段的任何其他频道的概率是相同的，也就是说发生冲突的概率是相同的。因此 nRF24LE1 与跳频系统的工作发生冲突时没有必要改变自己的工作频道;而干扰来自频率稳定的系统时，需要跳转到另一个与该系统发生冲突概率较小的频道。只有当受到持续的干扰时才跳转到另一个信道，这便是频率捷变技术。

基于以上的分析，可以归纳出如下的跳频规则:

1) 监测到当前信道的持续干扰；
2) 跳转到受到自同一干扰源的干扰概率较小的信道；
3) 如果干扰来自其他的跳频系统，则不进行跳频。

5.2 无线抗干扰设计

nRF24LE1 集成了载波监测功能，可以准确地监测当前工作信道是否有干扰，保证了在 W- LAN 环境下可靠地工作。其 SPI 接口与外接微控制器的通信速率可达 10Mbps，具有高速度和独特的切换时间，减少了与跳频系统如蓝牙出现时碰撞的可能。

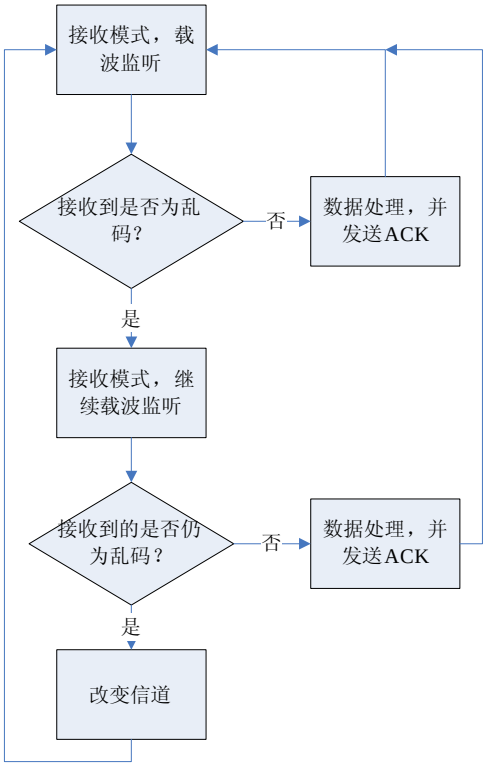

图 5.2 主节点跳频流程

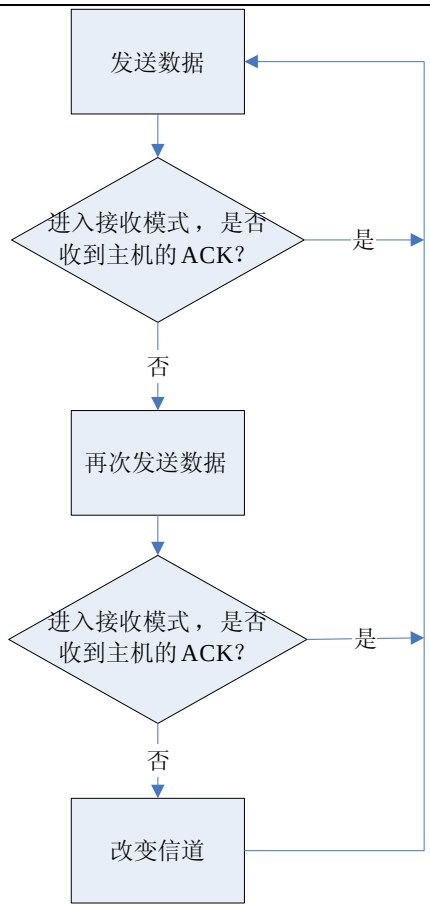

图 5.3 从节点跳频流程

本系统采用 nRF24LE1 的 Enhanced ShockBurst 模式进行通信，主节点接收和从节点发送方案如下:

对于主机节点，如图 5.2 所示，始终处于载波检测状态，当仅收到一次乱码干扰的时候，可能是与蓝牙系统发送冲突，主机不必改变信道；当持续收到当前频率的乱码干扰时，说明收到的不是脉冲干扰，而是稳定的干扰，这时 nRF24LE1 需要按照已设定的信道列表跳转到另一个信道。

对于从机节点，如图 5.3 所示，发送完数据后等待主机的 ACK，如果没有收到 ACK，表示发送失败，则在相同信道上重发三次。由于蓝牙系统在每个信道上停留的时间为 650us ，而 nRF24LE1 一次动作 (即发送数据并等待接收对方 ACK 的时间) 大约为 1ms，因此如果第一次发送失败是由于与蓝牙系统发生冲突，那么第二次发送一般可以顺利到达接收方。如果三次发送均失败，说明受到的不是脉冲干扰，而是稳定的干扰，这时 nRF24LE1 需要按照已设定的信道

列表跳转到另一个信道。事先将所有想要使用的信道做成列表，在需要跳频时查表即可。

第六章 无线通信协议

6.1 介质访问控制协议设计

为了实现在同一范围内多点间通信，需要考虑防止数据包在大气中传输时相互碰撞。为了建立可靠的无线传输通路，必须采用各种方法。无线通信中避免多节点无线通信冲突常用的办法有频分多址（FDMA）、时分多址（TDMA）、 跳频（FHSS）、载波监听（CSMA）等。

频分多址（FDMA）技术将可用的频率带宽拆分为具有较窄带宽的子信道，这样每个子信道均独立于其他子信道，从而可被分配给各个发送器。FDMA 因为系统有自己的专门频道，所以非常低的系统同步要求，软件控制比较简单，实现相对容易。但是，子信道之间必须间隔一定的距离以防止干扰，造成频带利用率不高；当用户处于空闲状态时，会导致带宽的浪费。对于本系统星型网络而言，频分多址可以避免通信冲突问题，但是，随着节点的增多，单一的主机节点必须增加监听的信道数目，从而使通信效率下降。

时分多址（TDMA）是在同一信道，把不同地址发送的信号按照时间间隔的方法进行传输的一种无线通信方式。TDMA 通信质量高，保密性较好，系统容量大；节点在不需要进行数据收发时间段，相关无线节点可以关闭，以降低节点电池消耗；可以使用多个时间片，提高数据传输速度；频道资源利用率大大高于 FDMA。但是, TDMA 必须有精确的定时和同步以保证发送端和接收端间正常通信，技术上比较复杂；在突发性传送时，需要信号处理技术和高通信余量。对于本星型网络而言，时分多址在避免通信冲突并关闭非时段的节点降低功耗，但是，其需要精确的定时和同步和高通信余量，都会增加系统的复杂性。

跳频（FHSS）是收发双方传输信号的载波频率按照预定规律进行离散变化，也就是说,通信中使用的载波频率受伪随机变化码的控制而随机跳变。与定制通信相比，跳频通信比较隐蔽且难以被截获——只要对方不清楚载频跳变的规律,就很难截获我方的通信内容。FHSS 不是抑制干扰，而是容忍干扰。每个数据包都会被校对，如果它被破坏，这个数据包将不能再用，随后使用下一个传输频率的通信会提供正确的数据包。因而 FHSS 传输速度较慢却稳定可靠。对于本系统而言，跳频可以提高通信的安全性和可靠性，但是，跳频通信需要通信双方都一直在线，这样不利于系统低功耗实现。

载波监听（CSMA）比起前面的 FDMA、TDMA、FHSS，能更好的利用资源，因为这种通信方法在发送数据之前，一直在检测大气中是否存在相同频率的载波。如果在当前时间中有相同频率的载波，就不发送数据；如果大气中没有相同频率的载波，表明现在空间资源没有被占用, 可以发送数据。这样不仅提高了空间资源的利用效率，也提高了通信的可靠性。CSMA 既是一种争用型的介质访问控制协议，又是一种分布式介质访问控制协议，网络中各个节点都能独立的决定数据帧的发送与接收。每个节点在发送数据之前，首先要进行载波监听，只有介质空闲时，才允许发送数据帧。如果两个以上的节点同时监听到介质空闲并发送帧，则会产生冲突现象，发送的帧为无效帧，应停止发送，以免介质带宽因传送无效帧而被白白浪费；然后根据节点的 ID 号延时一段时间后，再重新争用介质，重发数据。

相对来说，CSMA 原理比较简单，技术实现容易，网络中各工作站处于平等低位，不需集中控制, 不需提供优先权。因此，如图 6.1 所示，本系统采用载波监听和据 ID 号延时的方法来实现一对多的无线数据传输。

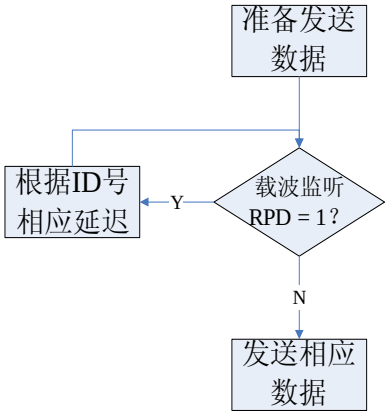

图 6.1 载波监听流程图

本系统采用载波监听具有诸多优点。比较简单，技术容易实现，可以提高开发效率；无用节点可以睡眠，达到低功耗功能。

6.2 通信协议设计

首先，要设定表 2.2 增强型 ShockBurst 包格式中载荷的数据结构，如表 6.1 所示。

帧头 1 byte	类 ID 1 byte	设备 ID 1 byte	数据长度 1 byte	数据 0~26 byte	总帧数 1 byte	帧号 1 byte

表 6.1 载荷的数据结构

- 帧头：通过帧头来判断数据的类型，有：主机申请数据；传感器触发；加入网络；退出网络；广播信号。根据不同的帧头，确定对载荷采取不同的处理。
- 类 ID：划分不同类型的设备。
- 设备 ID：划分同类型的设备。
- 数据长度：为数据的长度，取值为 0~26。
- 数据：0~26 byte。
- 总帧数：一个数据包总帧的个数。
- 帧号：表示现在传输的是第几帧数据。

帧头，ID 号，数据长度，数据都由 ARM 处理器部分通过 SPI 发过来；单片机再通过 SPI 传给 RF 部分，RF 部分再自动的把它组装成无线传输的数据格式，通过无线传输；总帧数和帧号，可以保证断点续传。

如表 6.2 所示，为针对此监控项目所设计的通信协议。

序号	同步时段	广播时段	加入网络时段	从机申请数据上传（申请数据和网络维持时段）	网络维持（申请数据和网络维持时段）	主机索取数据（点到点数据交换时段）
主机从机发送接收状态	主机发送；从机接收	主机发送；从机接收	主机接收；从机发送	主机接收；从机发送	主机接收；从机发送	主机先发送后接收；从机先接收后发送
1（帧头）	0xDD	0xFF	0xEE	0x88	0xB0	0x80
2（类 ID）	对于广播信号，类 ID 和设备 ID 全为 oxFF，则全部节点都可接收到此帧；再判断 Data1 和 Data2 是否和自己的匹配，若匹配，则执行 Data3 的命令，并在从机申请数据长传时段把主机所需状态信息传给主机。（Integrated Camera Unit 在从机申请数据上传时段，上传自己的准备状态，在点到点数据交换时段才进行图片数据的交换） Integrated Camera Unit 0x11；Remote Sensor IR Unit 0x22；Door / Panic Alarm 0x33；Remote Equipment Switch 0x44；remote controller 0x55.					
3（设备 ID）	从机的设备 ID 由主机指定。在加入网络申请帧里，从机的此值为 0xFF；当主机给申请的从机确认信息帧里，包含给其分配的设备 ID。					

4（有效数据长度）	0x04	0x04	0x04	0x04	0x04	0<L<27
5 Data（1）	Th（T 计时器相关）	preserved	类ID	详见（注 3）	是否正常 正常（0xFF） 问题（0x00）	——
6 Data（2）	Tl（T 计时器相关）	preserved	设备 ID		出现问题（注 4）	——
7Data（3）	TTemp （T 计数器）	preserved	命令（注 2）		preserved	——
8 Data（4）	preserved	preserved	preserved	preserved	preserved	——
9Data other	XX					——
10 总帧数	0x01					——
11 帧号	0xFF					——

表 6.2 通信协议

对于遥控器，其命令帧头为 0xAA，无论主机何时收到次命令都会优先传输给 ARM 主机进行处理。初次同 ARM 主机进行配对，配对完成后，绑定其 MAC 地址。其命令格式另外定义。

点到点数据交换时段仅为 Integrated Camera Unit 向主机传输图像数据时段。

✧ （注释 1）在点对点数据交换时段，总帧数最大为 255，最多传输 6885Byte。所以，可以通过控制帧头以达到传输多次的作用。主机索取数据帧头为 0x80。但是，要传输数据的节点，当数据<6885B 时，从机传输的帧头仍为 0x80；当>6885B 时，传输给主机的帧头为以 0x60 开头，到 0x6F，共可传输 16*FF 个 Btye。（？？）

✧ （注释 2）命令类型：

0x11：报告从机自己的基本状态；

0x22：向 Integrated Camera Unit 索取图像命令；

0x33：对 Remote Equipment Switch 的具体控制；

✧ （注释 4）从机申请数据上传时，一方面，除了 Integrated Camera Unit 外的所有节点分机，把自己的信息状态发送给主机；另一方面，Integrated

Camera Unit 只是上传自己的准备状态，在点到点数据交换时段才进行图片数据的交换。

✧ （注释 5）问题包括：0x11：传感器被触发； 0x22：门传感被触发； 0x33：紧急呼叫； 0x44：电量不足

参考文献

[1] 谭晖.nRF 无线 SOC 单片机原理与高级应用[M].北京:北京航空航天大学出版社,2009.

[2] 曹勇，杨涛，冯月晖. 基于 nRF24L01 的超低功耗无线传感器网络节点设计[J]. 电子技术应用,2008,(7):45-48.

[3] nRF24LE1_Product_Spec_v1_2.pdf. [EB/OL]. 2009.3. http://www.nordicsemi.com/index_popup.cfm?obj=product&act=displayWhitepapers&pro=95&con=data_sheets

[4] 李晓东，卫建华.nRF24xx 频道避撞及频率捷变技术的实现[J].西安工程科技学院学报,2007,(2).226-229

[5] 喻金钱，喻斌.短距离无线通信详解—基于单片机控制[M].北京:北京航空航天大学出版社,2009.

[6] 李文仲，段朝玉.短距离无线数据通信入门与实战[M].北京:北京航空航天大学出版社,2008.