

ADI NOTE: CONJESTED HOPPING

144026, 2020-11

目录

1 通信协议设计	1
1.1 数据包字段	1
1.2 通信流程图	2
1.3 通信算法	3
2 协议实现	3
2.1 Utilities	3

1 通信协议设计

1.1 数据包字段

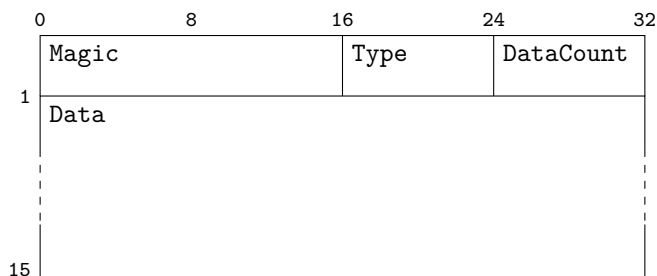


图 1: 数据包结构

各字段作用如下:

- (1) Magic: 2 个字节, 值依次为 0xad 和 0x13。接收方用于判断解调、解码出的数据包是否有效。
- (2) Type: 1 个字节, 表示数据包的类型, 其值可以为:
 - (a) 0x0: Hopping SYN(跳频同步), 通知接收方 (B 机) 跳至下一个频率。
 - (b) 0x1: Hopping ACK(跳频确认), 通知发送方 (A 机) 跳频已完成。

- (c) 0x2: File, 表示 Data 段包含的内容是文件 (例如 TXT 和 WAV) 数据。B 机收到后, 应该打开一个新的文件, 向其中复制数据。
- (d) 0x3: File Secondary, 表示 Data 段包含的是当前打开文件的后续内容,
- (3) DataCount: 仅当 Type 为 0x2 和 0x3 时有效。1 个字节, 表示数据段中, 有效数据的长度 (以字节为单位)。
- (4) Data: 56 字节的数据段, 仅当 Type 为 0x2 和 0x3 时有效, 有效载荷分别为:
 - (a) Type 为 0x2: TotalDataCount, 小端序 uint32。
 - (b) Type 为 0x3: 文件比特流。

1.2 通信流程图

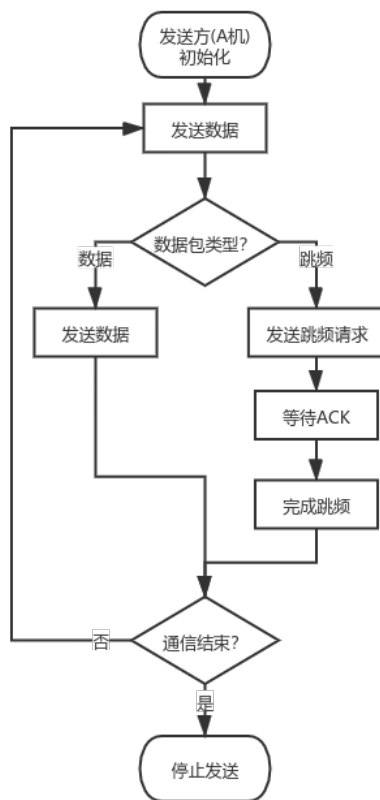


图 2: 发送方流程

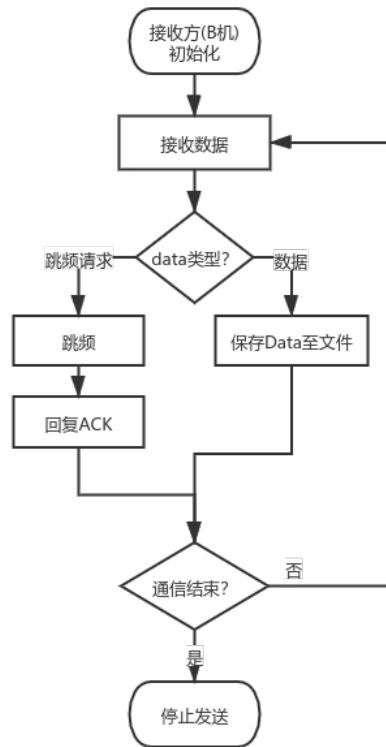


图 3: 接收方流程

1.3 通信算法

发送文件的程序算法如 **Algorithm 1**所示。

2 协议实现

2.1 Utilities

Algorithm 1 发送文件

```
1: 长度为 5 的跳频序列, 保存在数组 freqs[5] 中;
2:
3: 打开一个文件;
4: 读取 116 字节, 令 DataCount= 实际读到的字节数;
5: 令 Type=2, 构造数据包 (不足的 Data 补 0);
6: bpsk_send();
7:
8: 重新令 Type=0, 再次调用 bpsk_send();
9: while true do
10:   bpsk_receive() 接收 120 字节数据, 存入 rcv[120];
11:   if rcv[0] = 0xad && rcv[1] = 0x13 && rcv[2] = 1 then
12:     将 TX 和 RX 频率设置为下一个值;
13:     break;
14:   end if
15: end while
16:
17:
18: while DataCount ≥ 16 do
19:   继续读取 116 字节, 令 DataCount= 实际读到的字节数;
20:   令 Type=3, 构造数据包 (不足的 Data 补 0);
21:   bpsk_send();
22:
23:   重新令 Type=0, 再次调用 bpsk_send();
24:   while true do
25:     bpsk_receive() 接收 120 字节数据, 存入 rcv[120];
26:     if rcv[0] = 0xad && rcv[1] = 0x13 && rcv[2] = 1 then
27:       将 TX 和 RX 频率设置为下一个值;
28:       break;
29:     end if
30:   end while
31: end while
32:
33: 停止通信;
```
