XN297L 跳频重传方案设计 V1.0

Panchip Microelectronics

www.panchip.com

目录

第1章 背景介绍

本文档主要介绍了一种应用于 XN297L 的跳频重传方案。

XN297L 的工作频段为 2.4GHz，也因此容易受到 WiFi、蓝牙等无线信号的干扰。而对窄带通信系统而言，使用跳频技术不失为一种较为有效的抗干扰手段。

本文档所介绍的跳频方案主要针对于以下应用场景：

1、对丢包率要求较为严格（如语音和视频等流媒体信号）；

2、发送端的发送间隔恒定（或基本固定，如遵循某种变化规律）；

3、发送端的发送间隔为 T ，每次发送时间为 t ，为确保通信质量，建议至少保证 $T \geq 6t$。

第2章 重传机制设计

2.1 重传机制简述

在引入跳频机制之前，先简单介绍一下 XN297L 的重传机制。

在当前的应用场景下，假设发送端每包数据的发送间隔为 T ，XN297L 的一次完整发送流程所需时间为 t_{send}。为了提高通信质量，一种很直接的方法是使用重传机制，而 XN297L 本身就具备自动重传功能[1]。

XN297L 增强模式自动重传的时序图如下图所示：

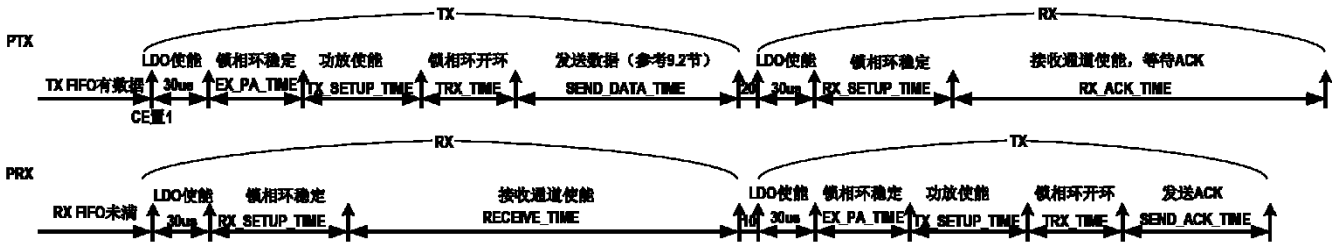

图 2.1 增强模式下的 PTX 和 PRX 的时序图（发送成功）

在增强模式下，接收端一旦收到数据，会切换至发送模式并发回一个 ACK。发送端在单次发送完毕后，会切换至接收模式并保持 RX_ACK_TIME 时间，等待 ACK。如果在时间内等到 ACK，则判定此次发送成功，结束发送流程；如果等待超时，则会再切换回发送模式，循环之前的操作，直到达到最大重发次数。

自动重传的优点是易于实现：收发模式切换、重传、ACK 等操作全部交由硬件自动完成。

但是缺点是耗时：在丢包环境下，一次发送完毕后，假如出现了丢包，发端也必须在 RX_ACK_TIME 超时后，才会开始下一次的发送流程。重传机制就是为了在丢包环境下，牺牲部分有效带宽降低丢包率，但自动重传机制下，丢包会造成时间的严重浪费，这是自相矛盾的。

2.2 手动重传模式

目前使用的策略是在普通模式下，使用 REUSE_TX_PL 指令，手动控制重传。

对于每一包数据，首次发送需要时间 $t_{send} = t_{SPI} + t_{RF}$；之后的每次重发过程不再需要写 TX_FIFO，需要时间 $t_{resend} = t_{RF}$。故最大重传次数 $N_{resend} = \left\lfloor \frac{T - t_{send}}{t_{resend}} \right\rfloor$。

在发送端，连续地进行 1 次直接发送和 N_{resend} 次重发，期间不做状态切换和等待；在收端对数据进行判断，如果非重发数据则保留并进行下一步处理，重发数据直接舍弃。整个过程中收端不发回 ACK。

[1] 带自动重传带 ACK 的模式也被称为“增强模式”，具体信息请参考 XN297L 的相关文档。

2.3 跳频和重传的结合

对于每一包数据，最大可能的发送次数为 $N = N_{resend} + 1$。

为了表述方便，将这 N 次发送的数据统称为一“包”，每次发送或重发的数据称为一“帧”。但不会区分丢包率和丢帧率的概念。

易知，随着 N 的增大，每包数据的丢包率会降低。但同时这种提升效果的边际效应递减十分明显。因此，有必要引入跳频机制，进一步提升通信质量。

对发送端的 N 帧数据来说，无非有两种跳频模式：N 帧数据在多个频点分别发送，或是在某一特定频点上进行发送。这一般被称为快跳频和慢跳频。

其中，慢跳频需要在收端进行较为严格的时间同步，以保证和发端的跳频速率一致；然而在丢包环境下，关键的同步信号的丢失可能会导致之后一段时间内收发端不同频（即失联），产生额外的丢包。

因此，目前使用的发送方案是基于快跳频的：选取 n 个频点，将 $N = n \cdot k$ 帧数据在每个频点各发送 k 次。

第3章 跳频机制设计

3.1 跳频机制简述

结合上文所述，目前使用的是基于快跳频的方案。

收发两端使用相同的跳频表。对于每一包数据，在 T 时间内，发送端将在 n 个频点上发送总计 N 次。接收端在每 T 时间内，停留在其中某一频点上接收。

假设$\{n = 3, k = 2\}$，即发送端对于每包数据在 3 个频点上各发 2 包，共计发送 6 包。而收端的频点随机跳动，则收发两端频点随时间变化图如下所示：

图中，ABC 分别表示频点，数字表示包号。对于第一包数据而言，依次发送了 A1, B1, C1, A1, B1, C1 这六次。而此时接收端在频点 B 上，共收到了两个 B1 包。

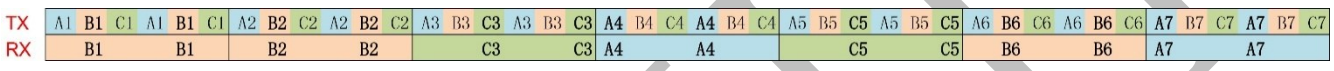

图 3.1 { n = 3, k = 2}、收端随机跳频，频点时序示意图

对于这种跳频方案，需要注意以下三个要点：

1、n 和 k 的取值，以及 n 个频点的选择；

2、收端的时间同步（周期为 T 的相位同步）；

3、收端的频点切换策略（基于频点质量）。

3.2 跳频参数的确定

无论跳频方案如何设计，对于每一包数据，最大可能的发送次数 N 是固定的。而$N = n \cdot k$，其中 n 为跳频使用的频点个数，k 为每个频点上的发送次数。

为了保证通信质量，建议至少保证 $k \geq 2, n \geq 3$。

目前的方案中，使用的是固定的跳频表。2.4G 主要的干扰源是 WiFi 和蓝牙，其中蓝牙由于其跳频特性，其信号基本可视作平均分布在 2.4G 频段内；而 WiFi 是宽带定频传输，为了尽量降低 WiFi 的干扰，需要将跳频表内的频点设置地尽量分散。例如选取至少 1 个低频频点、1 个高频频点，和 1~2 个中频频点，如{2.404, 2.436, 2.470}。

而对于每一包数据，收端在某一频点上接收，最多只可能收到 k 帧。在无干扰的情况下，跳频发送 nk 次的通信质量等价于定频发送 k 次。因此 k = 1 时，通信质量很难得到保证，建议降低 n 值，甚至放弃跳频，使用定频重传的方式进行通信。一般地，建议 k 值至少取 2 或 3。

3.3 跳频时点的选择

在发送端，跳频的时点很好确定，每一次发送/重发之前或之后进行跳频即可。由于是连续发送的（在收到 RF 发送完毕中断后立即开始下一次发送），之前和之后并没有本质上的差别。

3.3.1 时间同步

时间同步，即基于发端的信号进行频率和相位同步，确定出适合进行跳频的时点。

这里不妨先假设收端在每包数据之间都会进行跳频。图 3.1 中，收端选择的均是较为理想的跳频时点。

由于发送端的发送间隔已知且固定，因此收端可以直接在本地配置相同的的跳频间隔 T。那么时间同步主要就是相位同步。理想状态下，要求收发两端的相差 $\Delta T = 0$。

假设忽略发送和重发的时间差，令 $t = t_{resend} \approx t_{send}$，此时 $T = nkt$ 。事实上，$\Delta T = nt$或其整数倍均是可以接受的。如下图所示：

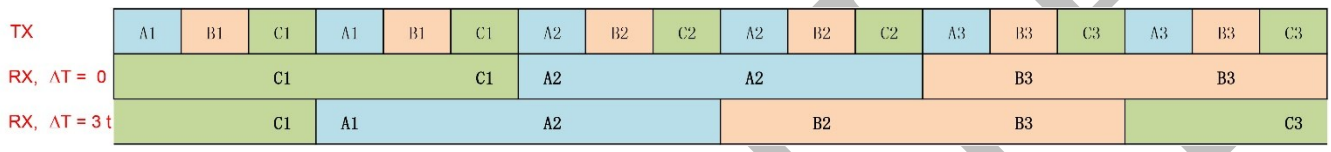

图 3.2 { n = 3, k = 2}、收端跳频周期为 T、相差为ΔT，频点时序示意图

因此，可以以每次收到数据（非重复帧）为基准进行相位同步，计算出下一个跳频的时点。具体的计算方法，需要对写 TX_FIFO 的时间、发送时间、读 RX_FIFO 的时间进行实测，在发送间隔、重传次数、Payload 长度固定时，这些时间都是比较稳定的。

而且，只要保持通信，频繁的相位同步也可以修正收发两端的频率误差，避免频率误差的累积。

3.3.2 频点切换策略

频点切换策略，即判断满足怎样的条件时，会选择 3.3.1 中的时点进行跳频。

如果收端在每包数据之间都进行跳频，则总体丢包率约等于各个频点上丢包率的均值。如果在某些频点出现了干扰，平均通信质量也会因此而下降，降幅会被平均化。

显然，一种更优的策略是使总体丢包率接近各个频点上丢包率的最小值，即尽可能地让通信建立并保持在质量最好的某个频点上。如 3.2 所述，受限于时间资源，并没有加入频点扫描机制。同样地，在频点的切换策略上，也没有进行过于复杂的设计。

目前的方案是：在收端进行一段时间内的丢包率统计。（为保证时效性，时间窗口不宜取得太大）一旦丢包率大于某阈值，则说明该频点质量不够好，此时进行频点的切换；（准确地说，是在下一个满足 3.3.1 中条件的时点进行跳频）否则保持在当前的频点不变。

目前使用的是固定的阈值。假设阈值为 30%，这意味着：

1、 如果某一段时间内，跳频表中存在一个或以上频点满足丢包率 $< 30\%$，则通信可能建立并保持在其中任意一个频点上，总体丢包率 $< 30\%$。

2、 如果某一段时间内，跳频表中不存在任意频点满足丢包率 $< 30\%$，则通信无法保持在任何一个频点上，收端会一直跳频，总体丢包率约等于各频点上丢包率的均值。$(> 30\%)$

以音频和视频传输为例，可以将维持流畅播放所能承受的最高丢包率设为阈值。

3.4 可能的优化方向

如前文所述，由于受限于时间 T，有很多设计存在待优化的空间。

3.4.1 XN297L 的增强模式

如 2.1 所述，XN297L 的增强模式会延长每次发送所需的时间。如果时间宽裕，则增强模式也不失为一个可选的方案。

手动重传模式下，发端将无视收端是否有收到数据，每一包均发送 N 次。而增强模式下，如果收端收到数据，则可以提前结束发送流程。在发送间隔不严格相等的情况下，这部分时间的节约依然可能是有意义的。

此外，增强模式允许收端发回带 Payload 的 ACK，可以发挥一定的作用。例如可以将各频点丢包率的统计结果写入 Payload 发回等等。

3.4.2 频点扫描

如 3.2 所述，目前并没有进行 2.4G 范围内的频点扫描。

由于使用了固定的跳频表，且选择的频点有限，因此可能会出现表内所有频点或多或少均受到干扰的情况。即使选取了其中最优的频点进行通信，总体通信质量依然会受限于跳频表的选择。

频点扫描可以在一定程度上解决这种问题。通过遍历 2.4G 范围内的若干频点，可以挑选出这段时间内，质量最好的几个频点，生成动态的跳频表。由于频点的使用情况随时间波动较明显，需要周期性地重复此操作。

但是频点扫描需要在收发两端进行多次频点切换，在多个频点分别进行通信，收端还需要将统计的数据回传，最终才能确定动态的跳频表。这将对通信协议和时序设计提出比较高的要求。

3.4.3 动态丢包率阈值

如 3.3.2 所述，目前的频点切换策略是比较简单的。如果阈值取得过低，当所有频点的丢包率均大于该阈值时，会导致收端无法稳定在一个其中较好的频点上。

结合频点扫描机制，可以利用发送的间隙，对各个频点进行评判、并制定动态的丢包率门限。

关于这部分，也可以参考一些自适应跳频的设计思路，在此不过多展开。

第4章 版本信息

版本	日	内容
1.0	2017-10-28	新建