XN297L 接收蓝牙数据配置方案

Panchip Microelectronics

www.panchip.com

目录

第1章 数据包格式和匹配机制

XN297L 普通模式的数据包格式如下所示：

前导码	地址	数据	CRC 校验
3 Bytes	3 – 5 Bytes	（Payload）	0 / 2 Byte(s)

其中，前导码固定为 0x710F55，地址的长度（3-5 字节）和数据可配，CRC 为硬件生成。

BLE 广播包的数据格式如下所示：

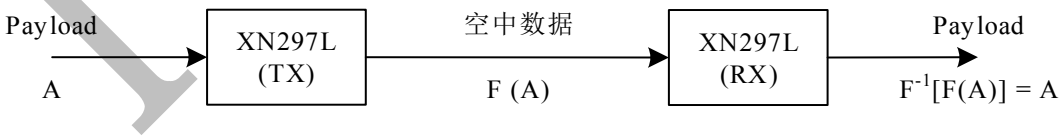
前导码	接入地址	数据	CRC 校验
1 Byte	4 Bytes	（PDU）	3 Bytes

XN297L 在匹配数据时，需要使前导码、地址和 CRC 的匹配全部通过。

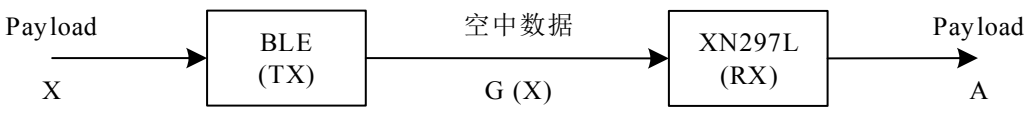
为了使 XN297L 能匹配上蓝牙发送的数据包，需要使 XN297L 数据包包含于 BLE 包内，即 BLE 发送的数据包应满足如下格式：

前导码	接入地址	数据头	XN297L 数据包				CRC
1	4	BLE Header	前导码(3)	地址(3-5)	Payload	CRC(2)	3

对 XN297L 的发送过程进行如下抽象：XN297L 的 Payload 为 A，XN297L (TX)可以等效为映射关系 F，XN297L(RX)等效为映射关系 F^{-1}。下图为抽象后的 XN297L 收发流程。



现将发送端替换为 BLE (TX)，将 BLE Payload 记为 X。其映射关系记为 G。



为使收端能成功匹配并收到数据，需要满足条件：一段完整的 F(A)被包含于 G(X)之中。

在这种匹配机制下，不需要再对接收端的 XN297L 进行额外的配置，只保持扰码和 CRC 功能开启，并将接收频点设置为和 BLE 广播包频点一致即可。

在 BLE 发送端（如手机 App 上），主要的工作是由 F、G、A 逆推反解出 X。

第2章 XN297L的空中数据格式

本章将介绍 XN297L 的数据转化成空中数据的过程。

这里将 XN297L 地址配置为 {0xC1, 0xC2, 0xC3, 0xC4, 0xC5}（长度为 5），Payload 配置为 {0x55}（长度为 1）。F(A)的包结构如下：

Preamble (前导码) （3 字节）	Address （5 字节）	Payload （1 字节）	CRC （2 字节）

XN297L(TX) 流程：

1、写 Address、Payload

Address 这里被设置为 5 字节，需要注意的是寄存器写入顺序为低字节在前，但空中数据为高字节在前。实际写入数据为：{0xC5, 0xC4, 0xC3, 0xC2, 0xC1}。

Payload 在这里被设置为 1 字节。多字节时，寄存器的写入顺序和空中数据相同，不需要反转端序。实际写入数据为：{0x55}。

	Address					Payload
index	0	1	2	3	4	5
value	0xC5	0xC4	0xC3	0xC2	0xC1	0x55

2、写 Preamble、CRC

Preamble 长度固定为 3 字节，写入数据为 {0x71, 0x0F, 0x55}。

CRC 长度为 2 字节（CRC16），作用范围为 Address 和 Payload。具体的实现方法（初始值、异或值、CRC 多项式等）见第 4 章。

这里直接给出结果：CRC16（0xC5, 0xC4, 0xC3, 0xC2, 0xC1, 0x55）= 0x52E7。组包时需要反转端序。实际写入数据为{0xE7, 0x52}。

	Preamble			Address					Payload	CRC	
index	0	1	2	3	4	5	6	7	8	9	10
value	0x71	0x0F	0x55	0xC5	0xC4	0xC3	0xC2	0xC1	0x55	0xE7	0x52

3、比特序反转

Preamble 和 Address 不变，将 Payload 和 CRC 的每个字节内部，进行比特序的高低反转（invert bit order）。

	Preamble			Address					Payload	CRC	
index	0	1	2	3	4	5	6	7	8	9	10
value	0x71	0x0F	0x55	0xC5	0xC4	0xC3	0xC2	0xC1	0x55	0xE7	0x4A

4、加扰/白化

对 Address、Payload、CRC 三部分做扰码。加扰前后数据长度不变，具体的扰码算法见第 4 章。这里直接给出加扰后的结果：

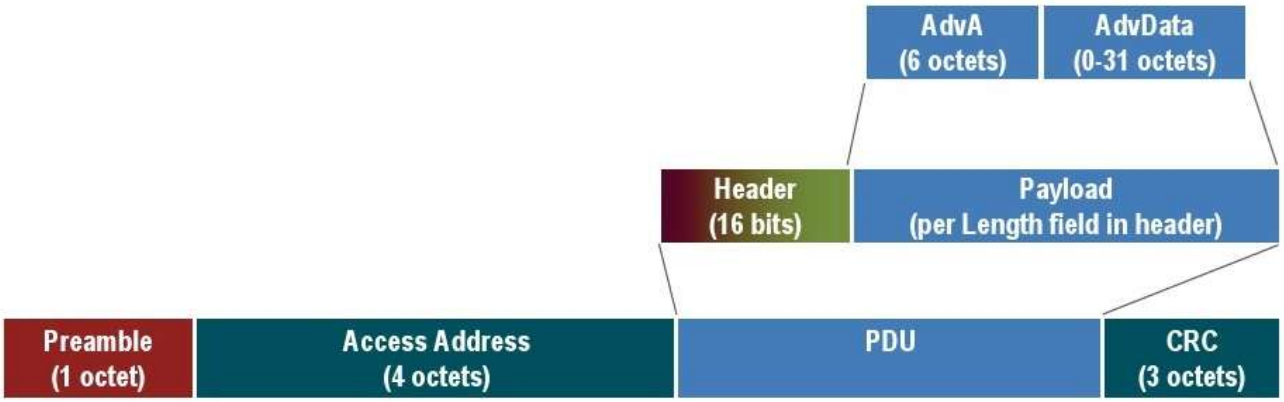
空中数据：

	Preamble			Address					Payload	CRC	
index	0	1	2	3	4	5	6	7	8	9	10
value	0x71	0x0F	0x55	0x02	0x49	0x11	0x95	0x60	0x97	0x40	0x2C

第3章 BLE的空中数据格式

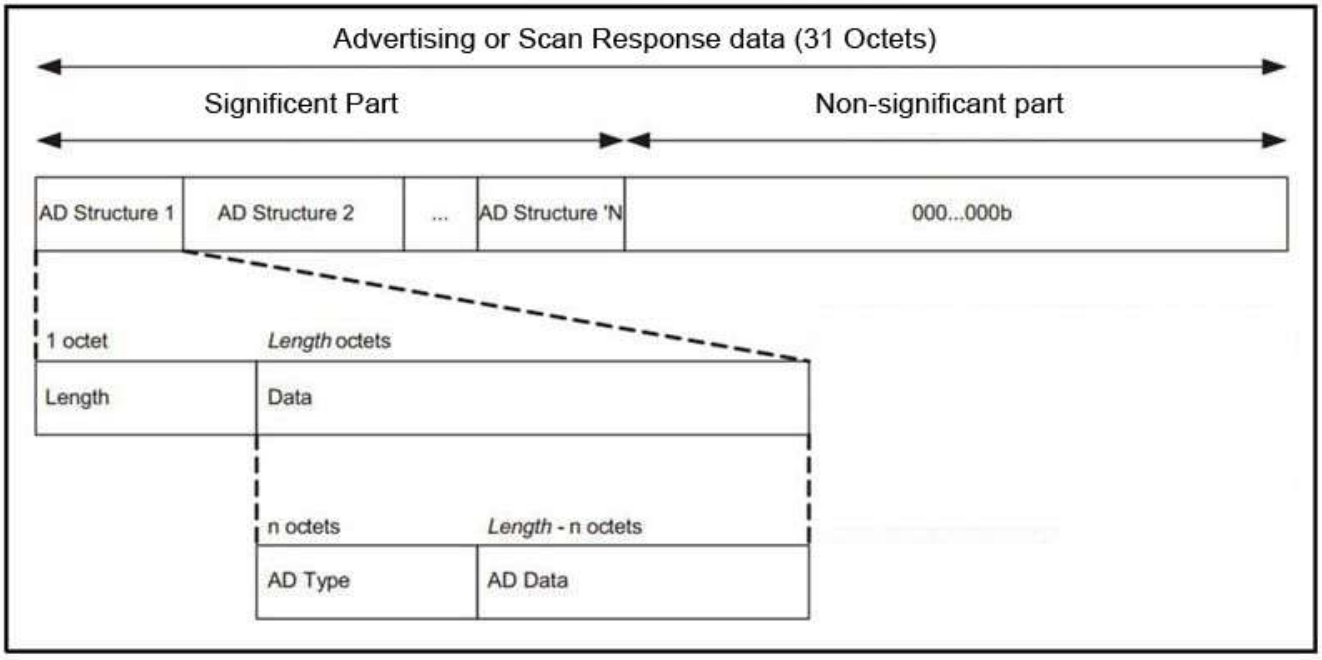
本章将介绍 BLE 的数据转化成空中数据的过程。

BLE 的广播包包结构如下所示：



Preamble	Access Address	PDU			CRC
1 Byte	4 Bytes	PDU Header	Adv Address	Adv Data	3 Bytes
		2 Bytes	6 Bytes	0 - 31 Bytes	

其中，Adv Data 由多个 AD Structure 组成，每个 AD Structure 需要满足特定格式：



在手机 App 端，会发送 2 个 AD Structure：

1、Length = 2，AD Type = 0x01 (Flags) 。这部分是由硬件生成的，总长度为 Length + 1 = 3。

2、Length = 14，AD Type = 0xFF (Manufacturer Specific)。在 App 内由用户输入的数据将被保存在该 AD Structure 内的 Additional Data 中。这也是用来构造 XN297L 的 Payload 的部分。

Manufacturer Specific：

Length	AD Type	Manufacturer ID	Additional Data
0x0E	0xFF	0xFFF0	11 Bytes

BLE(TX)流程：

1、组包

为表示方便，将 PDU 记为两部分 Extend Header、PDU Body：

Extend Header						PDU Body
PDU Header	Adv A	AD1	AD2 Length	AD2 Type	AD2 Manufacturer ID	AD2 Additional Data

则 BLE 广播包可以记为：

Preamble	Access Address	PDU		CRC
1 Byte	4 Bytes	Extend Header	PDU Body	3 Bytes
		15 Bytes	11 Bytes	

2、加扰/白化

BLE 会对 PDU 部分做扰码，即 Extend Header 和 PDU Body。BLE 和 XN297L 使用的是相同的扰码算法，仅仅在初始值的选取上有所不同。

该算法的特性使得 PDU Body 加扰后的结果与 Payload Header 的取值无关，仅受其长度影响。具体的算法实现见第 4 章。

3、射频发送

将上述数据转化成空中数据。和 XN297L 相同，PDU 的每个字节内部需要进行比特序的反转。

第4章 扰码和CRC算法的软件实现

1、白化/扰码 (Whitening)

BLE 和 XN297L 中使用的是相同的扰码算法。

这种算法由 7 位移位寄存器实现。移位寄存器生成一个伪随机序列，与 data in 进行异或，得到 data out。

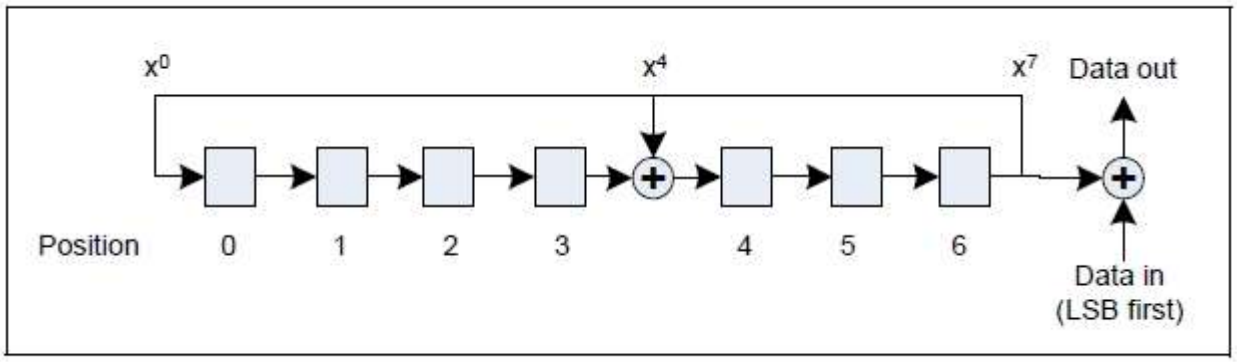


Figure 3.3: The LFSR circuit to generate data whitening

BLE 所使用的移位寄存器（或伪随机序列）的初始值与频点有关（channel index）：

(1) reg[0] 固定为 1

(2) reg[1] - reg[6]由 channel index 转为二进制得到。

(3) MSB 位于 reg[0], LSB 位于 reg[6]。

e.g., channel index = 23(0x17):

reg[0] = 1, reg[1] = 0, reg[2] = 1, reg[3] = 0, reg[4] = 1 reg[5] = 1, reg[6] = 1

BLE 使用到 3 个频点，2402MHz 对应 channel index = 37，2426MHz channel index = 38，2480Mhz channel index = 39。

在 XN297L 中，全部使用 channel index = 0x3F 进行初始化，即移位寄存器值全为 1。

该算法有两个特点：

（1）子序列的扰码不受其他子序列的取值影响，只和自身取值以及起始位置有关。

若对序列{X} = {$X_1, X_2, \ldots, X_n$}进行加扰，其中 X_i 为 X 的子序列，长度为 L_i。

X_1 加扰后的结果为 Whitening$^{(0)}$(X_1)，X2 加扰后的结果为 Whitening$^{(L1)}$(X_2)，X3 加扰后的结果为 Whitening$^{(L1+L2)}$(X_2)，以此类推。

上标（0）表示移位寄存器自循环 0 次时的输出值，即初始值。(L_1)表示自循环 L_1 次后的输出值。由于 data in 不会参与移位寄存器的自循环，因此子序列的加扰结果只和自身的取值以及子序列在整个被加扰序列中的起始位置有关。即：Whitening$^{(L_1+L_2+\cdots+L_{i-1})}(X_i)$

（2）对同一序列进行两次（初始值相同的）加扰，会得到原序列。

这是因为扰码算法的核心部分是异或运算，在扰码初始值相同时，做两次异或会返回原值。也即，解扰/逆白化操作等价于再做一次初始值相同的加扰/白化操作。

2、CRC (XN297L)

XN297L 中的 CRC 使用的是多项式为 0x1021 的 CRC16 算法，初始值为 0xFFFF，输出时的异或值为 0xFFFF。

CRC 的输入值为 XN297L 的 Address 和 Payload。

其中，Address 在输入时需要调整端序，即如果在 XN297L 的配置文件中，地址被配置为{0xC1, 0xC2, 0xC3}，则实际输入 CRC 的字节顺序为{0xC3, 0xC2,0xC1}。

Payload 在输入时需要反转比特序。最终的校验结果也需要先反转比特序，再异或 0xFFFF，最后输出。

第5章 版本信息

版本	日	内容
1.0	2018-01-26	新建
1.1	2018-04-02	修改匹配机制，将 XN297L 包嵌在 BLE 包内
1.2	2018-04-11	细化 BLE 广播包的介绍，暂时隐去了公式化的推导和转换过程