CC26xx/CC13xx 私有模式如何和 Amiconn A7106/A7219 互联互通

Barbara Wu — *Central FAE*
Louis Lu — *EC EP FAE*
Victor Zheng — *Beijing EP FAE*

概述

CC26XX/CC13XX 私有模式可以分别实现 2.4GHz 或 Sub-1GHz 频段的私有协议通讯。在工业领域的应用中，除了提供整套私有模式通讯方案，还常常需要和客户现有私有协议应用方案兼容，而现有私有协议应用方案不一定是使用 TI 的私有模式芯片实现。因此实现和不同厂商的私有芯片互联互通，对 TI 方案的推进有重要的意义。本文介绍了如何实现 CC2650/CC1310 分别在 2.4GHz 和 Sub-1GHz 频段和 Amiconn A7106/A7219 的互联互通，包括基本通讯互通，CRC 校验互通，以及 Whitening 互通。

Document History

Version	Date	Author	Notes
0.1	Jan.24 2017	Barbara Wu, Louis Lu, Victor Zheng	First release

内容

插图

表

1 CC26xx/CC13xx 简介

CC26XX/CC13XX 系列芯片是德州仪器针对无线通讯应用而设计的搭载 2.4GHz 或 Sub-1GHz RF 收发器的 MCU。芯片在基于支持多种物理层和 RF 标准的平台上将灵活的超低功耗 RF 收发器和强大的 48MHz Cortex-M3 微控制器相结合，并集成一个 Cortex-M0 作为专用无线控制器来处理 ROM 或 RAM 中存储的底层 RF 控制协议命令，从而同时保持超低功耗和系统的灵活度，而不需要为了实现超低功耗而损失 RF 性能。

CC26XX 系列集成 2.4GHz RF 收发器。其系列中不同芯片可支持不同的通讯协议 (Zigbee, Zigbee RF4CE, 6LoWPAN, Bluetooth low energy)。同时，系列中的 CC2650 除了面向包括上述所有通讯协议之外，还支持 2.4GHz 私有协议。

CC13XX 系列集成 Sub-1GHz RF 收发器。全系列芯片都可支持 Sub-1GHz 6LoWPAN 和私有协议。另外系列中的 CC1350 还集成了对 2.4GHz 射频的支持，可以同时运行 Bluetooth low energy 和 Sub-1GHz 私有协议。

2 CC26xx/CC13xx 私有模式通讯协议基本数据格式

CC26XX/CC13XX 私有模式通讯协议的基本数据包格式有两种，一种标准格式可以兼容绝 大多数的 TI 以往的私有协议芯片的数据包格式，另一种高级格式可以提供更加灵活的组包格式，可以包含更多的数据参数。

2.1 标准格式

0–30 bytes	0–32 bytes	0 or 1 byte	0 or 1 byte	0–255 bytes	0 or 16 bits (0–32 bits)
Preamble	Sync word	Length field	Address	Payload	CRC

Figure 1 – 私有模式标准数据格式

如 Figure 1 所示一个标准的私有模式通讯协议数据格式由 Preamble, Sync Word, Length field, Address, Payload 和 CRC 组成。其中 Preamble, Sync word, Payload 是必须存在的部分，而 Length field， Address，CRC 是根据实际应用可以省略的部分。

Preamble: 是 1 和 0 间隔的一组长度为 0-30 bytes 的值。由于第一位的数值有 0 和 1 两种选择，因此 1 个 byte 的 preamble 有 0x55 和 0xAA 两种可能性，可以通过 CMD_PROP_RADIO_SETUP 或 CMD_PROP_RADIO_DIV_SETUP 配置。

Sync word: 接收端用于分辨当前包是否需要接收，如果相同则接收，否则不接收。可以是长度为 0-32 bytes 的任意数值。

Payload: 用户自定义数据部分。根据需求填入任意需要发送的长度为 0-255 bytes 数值。

Length field: 1 byte 长度字段，是数据包里的可选部分，表示长度字段以后 CRC 字段之前的数据长度。当使用不定长数据收发的时候，可以使用这个部分来判断数据包的长度，帮助接收器正确解析数据包。

Address: 1 byte 地址字段，是数据包里的可选部分。可以用来做地址过滤。

CRC: 2 bytes CRC 校验字段，是可选部分。私有模式中 CRC 为 16 bits, IEEE 802.15.4g 模式中，CRC 可以为 32 bits。

2.2 高级模式

0–30 bytes or repetitions	0–32 bytes	0–32 bits	0–8 bytes	Arbitrary	0 or 16 bits (0–32 bits)
Preamble	Sync word	Header	Address	Payload	CRC

Figure 2 – 私有模式高级数据格式

如 Figure 2 所示一个高级的私有模式通讯协议数据格式由 Preamble, Sync Word, Header, Address, Payload 和 CRC 组成。其中 Preamble, Sync word, Payload 是必须存在的部分，而 Header， Address，CRC 是根据实际应用可以省略的部分。

Preamble, Sync word, Address, Payload, CRC 的定义和标准格式的定义相同，Header 里包含长度字段和其他可自定义字段。

3 CC2650 和 A7106 基本通讯互通

在调试互联互通的时候, 都要先从调试基本通讯互通开始。基本通讯互通需要从几个方面设置: 基本射频参数配置和数据格式配置。只有发送端和接收端的基本射频参数和数据格式都匹配，数据收发才能正常通信。

在保证基本通讯互通之后，可以在此基础上再添加更复杂的配置，例如添加数据白化(Whitening)，CRC 校验等。

3.1 基本射频参数配置

基本射频参数包含几个关键参数: 通讯频点(frequency), 调制方式(modulation), 频偏(deviation), 通讯符号速率(symbol rate), 接收滤波器带宽(rx filter bandwidth)。这些基本参数接收端和发送端必须设置相同才可能正常通讯。如果接收端和发送端都采用同一颗射频收发芯片,则基本通讯调试最简单, 将参数设为完全一样即可。但在实际应用中, 常常发送端和接收端需要采用不同的射频收发芯片, 这时即使是采用同一个厂家的不同芯片, 都有可能需要微调基本参数配置，下文着重讨论 CC2650 和 A7106 的基本参数配置方法。

根据客户项目需求，为了兼容上一代产品 A7106 端射频配置不能修改，因此需要根据 A7106 的参数配置来设置 CC2650。A7106 的射频参数配置列表如下:

Table 1. A7106 射频参数配置

参数名称	参数数值
Modulation	GFSK
Frequency(MHz)	2401
Symbol rate(kbps)	100
Deviation(kHz)	186

通过查询 CC2650 的 Technical Reference Manual 文档可知，CC2650 的上述参数可以通过 CMD_PROP_RADIO_DIV_SETUP 和 CMD_FS 命令配置。CC2650 这一侧，首先可以容易的确定的配置如 Table 3:

Table 2. CC2650 射频参数配置一

参数名称	参数数值	相对应的 CC2650 的数值
Modulation	GFSK	modulation.modType = 1
Frequency(MHz)	2401	centerFreq = 2401 frequency =2401
Deviation(kHz)	186	modulation.deviation = 744

接下来看 Symbol rate 如何配置。从 TRM 文档 23.7.5.2 可以看到计算公式为:

$$f_{baud} = (R \times f_{clk}) / (p \times 2^{20})$$

where

- f_{baud} is the obtained baud rate
- f_{clk} is the system clock frequency of 24 MHz
- R is the rate word given by symbolRate.rateWord
- p is the prescaler value, given by symbolRate.preScale, which can be from 4 to 15

由此可以看到和 Symbol rate 相关的配置有 symbolRate.rateWord 和 symbolRate.preScale 两个，且 symbolRate.preScale 的配置范围在 4 到 15 之间。

通过计算可以确定 Symbol rate 的配置如 Table 3:

Table 3. CC2650 射频参数配置二

参数名称	参数数值	相对应的 CC2650 的数值
Symbol rate(kbps)	100	symbolRate.preScale = 15 symbolRate.rateWord = 65536

最后来看 Rx filter bandwidth 的配置。当 modulation 是 2GFSK 时，tx bandwidth 通常可以用 data rate 加两倍的 deviation 来估算。代入之前已经确定下来的值，100k + 2* 186kHz = 472kHz。再通过查询 TRM 文档 23.7.5.2 中的 Table 23-146，可以看到 CC26xx 中最接近且大于 472kHz 的配置应该是 9。

这时如果可以确认发射端和接收端使用同样的数据格式，那么就可以来调试和验证射频配置了。但在这个案例中，CC2650 还需要匹配 A7106 的数据格式。如何调整数据格式将在下一节介绍。

3.2 数据格式配置

根据第 2 章中介绍的 CC26xx 的数据格式，可以知道一个普通的数据包必须包含如下几个参数:前导码(preamble), 同步字(sync word), 数据主体(payload)。在调试的时候，为了避免复杂的数据包格式调试叠加在射频参数调试上，可以先将接收端和发射端的数据格式配置设为最简化的配置，即只有 preamble，sync word 和 payload 三个部分，且每个部分的数据长度都设为固定长度。这样当射频基本参数设置和 preamble，sync word 配置匹配上，就应该能够正常收发固定长度的 payload 数据了。在调试完成，能够正常收发 payload 数据后，再添加例如 CRC 等其他的数据格式。在本部分中，就主要介绍 CC26xx/CC13xx 中私有模式 preamble，sync word 和 payload 的配置调试方法，以及如何匹配 CC2650 和 A7106 的数据格式。CRC 和白化将在后续章节中专门介绍。

对 CC26xx/CC13xx 而言，可以通过 TRM 查找如何配置 preamble，sync word，payload。和 preamble 相关的设置可以在 CMD_PROP_RADIO_SETUP 或 CMD_PROP_RADIO_DIV_SETUP 命令里 preamConf 部分找到。通过 preamConf 可以配置 preamble 的总长度，以及首字位是以 1 或是 0 开始。和 sync word 相关的配置则可以在 CMD_PROP_TX 或 CMD_PROP_RX 命令的 syncWord 部分以及 CMD_PROP_RADIO_SETUP 或 CMD_PROP_RADIO_DIV_SETUP 命令中 formatConf.nSwBits 参数找到，它们分别配置了 sync word 的数值和 sync word 的长度。固定 payload 长度，发送端在 CMD_PROP_TX 的 pktLen 中配置长度，接收端有两个参数相关 CMD_PROP_RX 中的 pktConf.bVarLen 和 maxPktLen，分别对应了固定长度，和具体长度数值。

在调试 A7106 和 CC2650 的互联互通时, 为了首先确认射频参数的配置，我们请客户将 A7106 端的数据格式改为最简单的组成，即不包含 CRC 和白化，只有最基本的数据格式。客户最终提供 A7106 的数据格式如 Table 4 所示:

Table 4. A7106 数据格式设置

参数名称	参数数值
ID code	0x52, 0x56, 0x78, 0x54
Payload	16bytes 0x01

通过查询客户提供的 A7106 的数据手册, 在 16.1 章可以看到 A7106 的数据组成。比较 CC2650 的数据格式, 可以认为 A7106 的 ID code 与 CC2650 的 sync word 相对应。从数据手册可知 A7106 的 preamble 配置是根据 ID code 的首字位来确定的，如果 ID code 的首字位是 1 则 preamble 的首字位也为 1，如果 ID code 的首字位是 0 则 preamble 的首字位也为 0。因此在 CC2650 侧，确认如 Table 5 对应的配置:

Table 5. CC2650 数据格式配置

参数名称	参数数值	相对应的 CC2650 的数值
Preamble	If the first bit of ID code is 0, preamble shall be 0101…0101. If the first bit of ID code is 1 preamble shall be 1010…1010. Preamble length is recommended to set 4 bytes	preamConf.nPreamBytes = 4 preamConf.preamMode = 2
Sync word	0x52, 0x56, 0x78, 0x54	syncWord = 0x52567854 formatConf.nSwBits =32
Payload	16byte 0x01	Tx: pktLen = 16 Rx: pktConf.bVarLen = 0

		maxPktLen = 16

3.3 配置参数微调

由于当前 SmartRF Studio 工具还没有开放对 CC2650 私有模式的支持，所以参数配置都需要通过调试代码来验证。代码可以基于 rfPacketTx 和 rfPacketRx 来修改，调试思路按照以下步骤，首先使用两块 CC2650 开发板，一个做接收端，一个做发送端，均应用以上配置修改射频参数和数据格式，验证收发端是否能够正常通讯。如果在能正常通讯，然后再替换一端为 A7106 来调试射频参数和数据格式。

在使用两块 CC2650 开发板调试过程中发现，基于可以正常通讯的 rfPacketTx 和 rfPacketRx 只修改数据格式配置，仍然可以正常通讯，而只修改射频参数配置，则无法正常通讯，说明射频参数的修改存在问题。

通过逐一排除参数，和使用仪器检查发射端的发射信号，发现 Tx bandwidth 实际为 650kHz，比估算的 472kHz 大。因此再次查 TRM，把 Rx filter bandwidth 配置从对应值 9 增加到对应值 10。再次验证修改两块 CC2650 的射频参数配置，确认能够正常通讯的。然后将射频参数和数据格式修改同时应用于 CC2650 并与 A7106 联调，验证互联互通成功。具体代码修改在 6.1 中介绍。

4 CC2650/CC1310 和 A7106/A7219 CRC 校验互通

4.1 实现 CC2650 和 A7106 CRC 校验互通

4.1.1 A7106 的 CRC 计算方式

A7106 的 CRC 的计算范围只包括 Payload 部分，并不包含 Preamble 和 ID code。ID code 在 CC2640 被称为 Sync word。A7106 使用的多项式为 x^16 + x^12 + x^5 + 1，初始值为 0x1D0F。输入的数据和最后的结果都是 non-reflected。

这里有一个 CRC 的计算小工具可以帮助大家验证 CRC 的设置
http://www.sunshine2k.de/coding/javascript/crc/crc_js.html

4.1.2 CC2650 的默认 CRC 计算方式

CC2640 默认的 CRC 计算范围也不包含 Preamble 和 ID code。CC2640 使用的多项式为 x^16 + x^15 + x^2 + 1，初始值为 0xFFFF。输入的数据和最后的结果都是 non-reflected。下面的连接中介绍了一些常用的 CRC 的多项式及其 16 进制表示。

https://en.wikipedia.org/wiki/Polynomial_representations_of_cyclic_redundancy_checks

4.1.3 CC2650 的 CRC 的配置

CC2640 的 CRC 计算方式的配置是非常灵活的。CRC 计算的范围，CRC 的多项式和初始值都可以根据需要来修改。

在发送数据时我们可以使用 CMD_PROP_TX 或 CMD_PROP_TX_ADV 命令。如 TRM 中的 Table 24-157 所示，CMD_PROP_TX 中的只有一位 bUseCrc 控制是否使用 CRC。要想对 CRC 的生成做更灵活的配置，我们需要使用 CMD_PROP_TX_ADV 命令。如 TRM 中的 Table 24-158 所示，在 CMD_PROP_TX_ADV 命令中除了 bUseCrc 位之外还有 bCrcIncSw 位和 bCrcIncHdr 位。bCrcIncSw 控制是否包含 Sync Word 在 CRC 中。bCrcIncHdr 控制是否包含 Header 在 CRC 中。

在接收数据时我们可以使用 CMD_PROP_RX 或 CMD_PROP_RX_ADV 命令。CMD_PROP_RX 与 CMD_PROP_TX 一样仅有 bUseCrc 位。CMD_PROP_RX_ADV 命令中有 bUseCrc，bCrcIncSw 和 bCrcIncHdr 位分别控制是否使用 CRC，CRC 是否包含 Sync Word 和 CRC 是否包含 Header。

从之前的 4.1.1 和 4.1.2 可以看出，A7106 和 CC2640 的 CRC 计算都只包含 Payload 部分。不同在于 CRC 的多项式和初值。CC2640 的多项式和初始值的修改可以在 CMD_PROP_RADIO_SETUP 的 Overrides 数组中去修改。我们会在之后的 6.2 节介绍具体的代码修改方法。

4.2 实现 CC1310 和 A7219 CRC 校验互通

CC1310 和 CC2650 在 CRC 部分的架构是完全相同的，所以 CC1310 的 CRC 校验在代码中的修改调试和 CC2650 完全相同。这里主要介绍一种不使用代码调试而使用 SmartRF Studio 软件验证调试的方法。由于目前软件还不支持 CC2650 私有协议模式，因此只有在 CC1310 上可以使用这个方法。

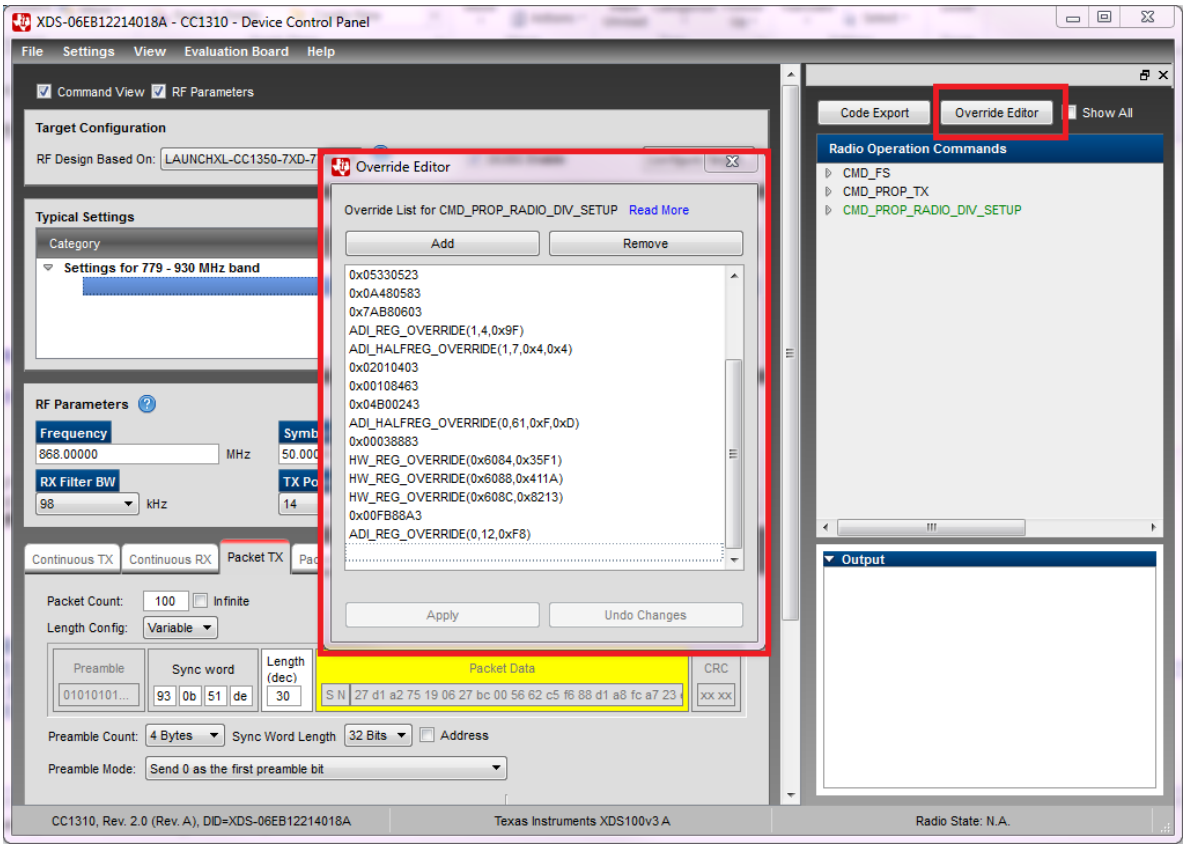
4.2.1 SmartRF Studio 简介

SmartRF™ Studio 是一个 Windows 应用程序，用于评估和配置德州仪器 (TI) 的低功耗射频 IC。这个应用程序可以将软件部分与硬件部分分开，帮助开发工程师调试射频系统或者在开发早起阶段评估射频芯片。它对生成配置寄存器值、实际测试射频系统和查找优化的外部组件值尤为有用。

4.2.2 使用 SmartRF Studio 修改 Overrides

在打开 SmartRF Studio 后，如果连接 TI CC1310 的开发板，在软件中是可以看到有 CC1310 的设备连接的。双击连接的设备可以打开对应的配置界面。

在配置界面里点击 Override Editor,会跳出如图所示窗口，在里面可以添加修改 overrides, 点击 Apply 后应用修改



5 CC1310 和 A7219 Whitening 互通

射频通信过程中，当发送数据是随机且无直流分量时候，发送的能量频谱分布会更加平滑且可提高射频链路的性能。实际通信过程中，数据包可能包含连续的 1 或者 0，进而影响通信的效果；这种情况下，对发送的数据进行白话处理，能够提高通信的性能；本节将简介 A7219 和 CC1310 的白化算法，以及在使能白化特性的情况下进行互通。

5.1 白化原理简介

白化是将要发送的数据与一伪随机码序列进行异或操作；当接收方收到数据后，将接收的数据与同序列的伪随机码进行异或操作，从而得到原始的用户数据; 以 PN9 白化举例，它可用多$X^9 + X^5 + X^0$表示。这个 PN9 算法可产生介于 1~511 内伪随机码序列。其电路原理图如图 1 所示。（详细的伪随机码序列及白化流程见参考文献一）

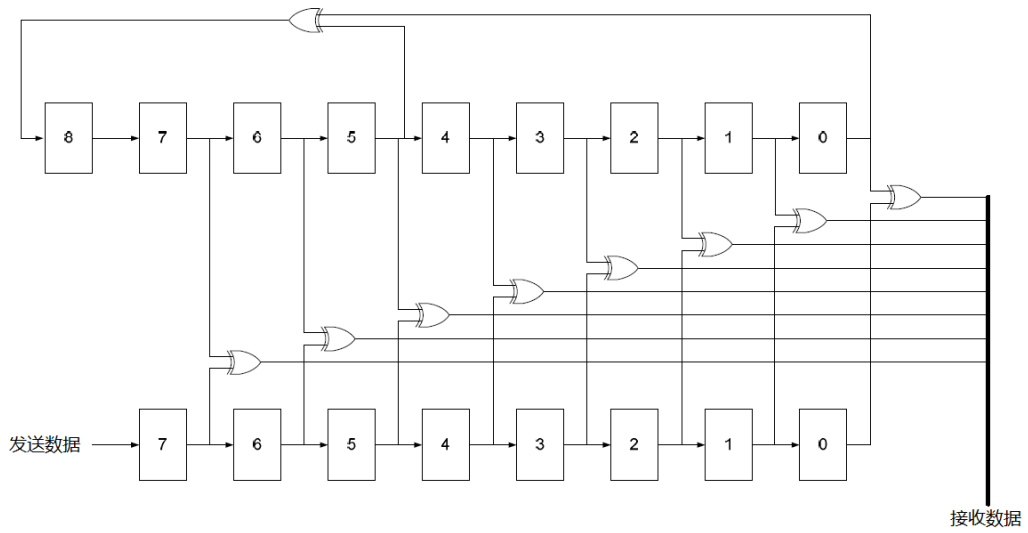


Figure 3 - 白化电路原理

5.2 A7219 及 CC1310 的白化算法

A7129 是 Amiccom 公司的一颗 SUB-1G 无线收发器, 其采用的白化算法是 PN7 算法, 多项式为$X^7 + X^4 + 1$，其伪随机码的序列种子由 WS[6:0]定义；

CC1310 是 TI 公司推出的一款集成低功耗 M3 内核和低功耗无线收发器的 SOC，默认采用 PN9 算法进行白化，多相式为$X^9 + X^5 + X^0$；

因此要使 CC1310 与 A7219 白化兼容，就需要修改 CC1310 的算法。这个需要在 CMD_PROP_RADIO_SETUP 的 Overrides 数组中去修改。我们会在之后的 6.3 节介绍具体的代码修改方法。

6 关键代码修改实现

本章主要介绍以上调试修改在代码中如何实现。

6.1 CC2650 和 A7106 基本互通

本章的代码实现是在 tirtos_cc13xx_cc26xx_2_21_00_06 的例程代码 rfPacketRx 和 rfPacketTx 上修改得来的。两个例程分别担当了发送方和接收方，可以直接在 TI demo 板上运行。这里以 rfPacketRx 为基础说明如何修改代码。

在例程中打开 smartrf_settings.c 文件，找到对应 CMD_PROP_RADIO_SETUP 的结构体, 其中.modulation.deviation 设置 deviation 为 186kHz，.symbolRate.rateWord 设置 symbol rate 为 100kbps，.rxBW 对应 rx bandwidth 650kHz 区间， .preamConf.nPreamBytes 设置 preamble 的长度为 4 bytes，.preamConf.preamMode 设置 preamble 的起始 bit 和 sync word 的起始 bit 相同。.formatConf.nSwBits 设置 sync word 的长度为 32 bits。

```
// CMD_PROP_RADIO_SETUP
rfc_CMD_PROP_RADIO_SETUP_t RF_cmdPropRadioDivSetup =
{
    .commandNo = 0x3806,
    .status = 0x0000,
    .pNextOp = 0, // INSERT APPLICABLE POINTER: (uint8_t*)&xxx
    .startTime = 0x00000000,
    .startTrigger.triggerType = 0x0,
    .startTrigger.bEnaCmd = 0x0,
    .startTrigger.triggerNo = 0x0,
    .startTrigger.pastTrig = 0x0,
    .condition.rule = 0x1,
    .condition.nSkip = 0x0,
    .modulation.modType = 0x1,
    //.modulation.deviation = 200,
    .modulation.deviation = 744,
    .symbolRate.preScale = 15,
    //.symbolRate.rateWord = 131072,     //200k
    .symbolRate.rateWord = 65536,     //100k
    //.rxBw = 7,
    .rxBw = 10,
    //.preamConf.nPreamBytes = 0x8,        //8 Bytes preamble
    .preamConf.nPreamBytes = 0x4,        //4 bytes preamble
    //.preamConf.preamMode = 0x0,
    .preamConf.preamMode = 0x2,         //same first bit
    //.formatConf.nSwBits = 16,
    .formatConf.nSwBits = 32,           //32 bits sync words
    .formatConf.bBitReversal = 0x0,
    .formatConf.bMsbFirst = 0x1,
    .formatConf.fecMode = 0x0,
    .formatConf.whitenMode = 0x0,
    .config.frontEndMode = 0x0,
    .config.biasMode = 0x0,
    .config.analogCfgMode = 0x0,
    .config.bNoFsPowerUp = 0x0,
    .txPower = 0x9324,
    .pRegOverride = pOverrides,
};
```

Figure 4 – CMD_PROP_RADIO_SETUP 相关参数

在 CMD_FS 对应的结构体中设置通讯频点。

```
// CMD_FS
rfc_CMD_FS_t RF_cmdFs =
{
    .commandNo = 0x0803,
    .status = 0x0000,
    .pNextOp = 0, // INSERT APPLICABLE POINTER: (uint8_t*)&xxx
    .startTime = 0x00000000,
    .startTrigger.triggerType = 0x0,
    .startTrigger.bEnaCmd = 0x0,
    .startTrigger.triggerNo = 0x0,
    .startTrigger.pastTrig = 0x0,
    .condition.rule = 0x1,
    .condition.nSkip = 0x0,
    // .frequency = 2440,
    .frequency = 2401,
    .fractFreq = 0x0000,
    .synthConf.bTxMode = 1,
    .synthConf.refFreq = 0x0,
};
```

Figure 5 – 通讯频点修改

在 CMD_PROP_RX 对应的结构体中修改 .pktConf.bUseCrc 为 0 从而关闭 CRC 校验，修改.pktConf.bVarLen 为 0 表示接收固定长度的数据包，通过.syncWord 设置同步字内容，修改.maxPktLen 来设置接收固定长度数据包的长度为 16 bytes。

```
// CMD_PROP_RX
rfc_CMD_PROP_RX_t RF_cmdPropRx =
{
    .commandNo = 0x3802,
    .status = 0x0000,
    .pNextOp = 0, // INSERT APPLICABLE POINTER: (uint8_t*)&xxx
    .startTime = 0x00000000,
    .startTrigger.triggerType = 0x0,
    .startTrigger.bEnaCmd = 0x0,
    .startTrigger.triggerNo = 0x0,
    .startTrigger.pastTrig = 0x0,
    .condition.rule = 0x1,
    .condition.nSkip = 0x0,
    .pktConf.bFsOff = 0x0,
    .pktConf.bRepeatOk = 0x0,
    .pktConf.bRepeatNok = 0x0,
    //.pktConf.bUseCrc = 0x1,
    .pktConf.bUseCrc = 0x0,
    //.pktConf.bVarLen = 0x1,
    .pktConf.bVarLen = 0x0,
    .pktConf.bChkAddress = 0x0,
    .pktConf.endType = 0x0,
    .pktConf.filterOp = 0x0,
    .rxConf.bAutoFlushIgnored = 0x0,
    .rxConf.bAutoFlushCrcErr = 0x0,
    .rxConf.bIncludeHdr = 0x1,
    .rxConf.bIncludeCrc = 0x0,
    .rxConf.bAppendRssi = 0x0,
    .rxConf.bAppendTimestamp = 0x0,
    .rxConf.bAppendStatus = 0x1,
    .syncWord = 0x52567854,
    //.maxPktLen = 0x7d, // MAKE SURE DATA ENTRY IS LARGE ENOUGH
    .maxPktLen = 0x10,
    .address0 = 0xaa,
    .address1 = 0xbb,
    .endTrigger.triggerType = 0x1,
```

Figure 6 – CMD_PROP_RX 相关参数

6.2 CC2650 和 A7106 CRC 校验互通

本章的代码实现是在 tirtos_cc13xx_cc26xx_2_21_00_06 的例程代码 rfPacketRx 和 rfPacketTx 上修改得来的。例程中使用了 CMD_PROP_TX 和 CMD_PROP_RX 来完成收发，CRC 计算时只包括 Payload。这里我们以接收为例说明代码如何修改代码。在例程 rfPacketRx 中打开 smartrf_settings.c，确认里面 CMD_PROP_RX 初值中的.pktConf.bUseCrc 为 1。使能 CRC。

```
// CMD_PROP_RX
rfc_CMD_PROP_RX_t RF_cmdPropRx =
{
    .commandNo = 0x3802,
    .status = 0x0000,
    .pNextOp = 0, // INSERT APPLICABLE POINTER: (uint8_t*)&xxx
    .startTime = 0x00000000,
    .startTrigger.triggerType = 0x0,
    .startTrigger.bEnaCmd = 0x0,
    .startTrigger.triggerNo = 0x0,
    .startTrigger.pastTrig = 0x0,
    .condition.rule = 0x1,
    .condition.nSkip = 0x0,
    .pktConf.bFsOff = 0x0,
    .pktConf.bRepeatOk = 0x0,
    .pktConf.bRepeatNok = 0x0,
    .pktConf.bUseCrc = 0x1,
    .pktConf.bVarLen = 0x0,
    .pktConf.bChkAddress = 0x0,
    .pktConf.endType = 0x0,
    .pktConf.filterOp = 0x0,
    .rxConf.bAutoFlushIgnored = 0x0,
```

Figure 7 – 使能 CRC

另外 smartrf_settings.c 中的 CMD_PROP_RADIO_SETUP 的 Overrides 的中，我们可以修改 CRC 的多项式值。如 Figure 4 的红色方框中的部分就是用来修改 CRC 的多项式和初值的。

```
// Overrides for CMD_PROP_RADIO_SETUP
uint32_t pOverrides[] = {
        MCE_RFE_OVERRIDE(1,0,0,1,0,0),
        HW_REG_OVERRIDE(0x4038,0x34),
        HW_REG_OVERRIDE(0x6088,0x3F1F),
        HW_REG_OVERRIDE(0x608C,0x8213),
        HW32_ARRAY_OVERRIDE(0x405C,1),
        (uint32_t) 0x1801F800,
        HW32_ARRAY_OVERRIDE(0x402C,1),
        (uint32_t) 0x00608402,
        (uint32_t) 0xc0040031,
        (uint32_t) &shape[0],
        (uint32_t) 0x00000343,
        (uint32_t) 0x001000a3,
        (uint32_t) 0x000484a3,
        (uint32_t) 0x1c8f0583,
        (uint32_t) 0x1c8f0543,
        (uint32_t) 0x65980603,
        (uint32_t) 0x00020623,
        (uint32_t) 0x659805c3,
        (uint32_t) 0x000205e3,
        (uint32_t) 0x02010403,
        HW32_ARRAY_OVERRIDE(0x4034,1),
        (uint32_t) 0x177F0408,
        (uint32_t) 0x00008463,
        (uint32_t) 0x00388473,
        (uint32_t) 0x00F388a3,
        HW32_ARRAY_OVERRIDE(0x2004, 1), // configure new CRC16 poly (=0x40012005 in pure hex)
        0x10210000, // new CRC16 poly: CRC-16-CCITT normal form, 0x1021 is x^16 + x^15 + x^5 + 1
        0xC0040051, // CRC initialization (address)
        0x1D0F0000,//0x00000000,//0x1D0F0000, // CRC initialization (value)
        (uint32_t)0xFFFFFFFF,
};
```

Figure 8 – 修改 CRC 多项式和初始值

第一行的“HW32_ARRAY_OVERRIDE(0x2004, 1)”是指明修改的寄存器的起始地址。0x2004 指向 CRC 相关的寄存器。随后的三行对应的是具体变量值。

第二行是 CRC 的多项式值，这里的我们写入 0x10210000 对应 A7106 的多项式对应多项式 x^16 + x^12 + x^5 + 1。

第三行是初始值的地址，是一个固定值 0xC0040051。

第四行是 CRC 的初值，我们按照 A7106 的要求写入 0x1D0F0000

第五行的 0xFFFFFFFF 表示整个 pOverrides 数组的结尾。一定要放在整个数组的最后一行。

另外，在调试的时候我们可以把接收的长度比实际的发送的数量多两个字节，这样发送过来的 CRC 值也可以被接收到。并且把 RF_cmdPropRx.rxConf.bAutoFlushCrcErr 设为 0，这样无法通过 CRC 校验的数据包会被保留。我们可以通过查看接收到的数据，用自己的多项式和初值计算 CRC，并与收到的 CRC 做比较来调试。

```
    /* Modify CMD_PROP_RX command for application needs */
    RF_cmdPropRx.pQueue = &dataQueue;           /* Set the Data Entity queue for received data */
    RF_cmdPropRx.rxConf.bAutoFlushIgnored = 0;  /* Discard ignored packets from Rx queue */
    RF_cmdPropRx.rxConf.bAutoFlushCrcErr = 0;   /* Discard packets with CRC error from Rx queue */
    RF_cmdPropRx.maxPktLen = 0x12;        /* Implement packet length filtering to avoid PROP_ERROR_RXBUF */
    RF_cmdPropRx.pktConf.bRepeatOk = 1;
    RF_cmdPropRx.pktConf.bRepeatNok = 1;
```

Figure 9 – 调整接收包长度和接收 CRC 错误包

6.3 CC1310 和 A7219 Whitening 互通

本章的代码实现是在 tirtos_cc13xx_cc26xx_2_21_00_06 的例程代码 rfPacketRx 和 rfPacketTx 上修改得来的。两个例程分别担当了发送方和接收方，可以直接在 TI demo 板上运行。这里以 rfPacketRx 为基础说明如何修改代码。

在例程 rfPacketRx 中打开 smartrf_settings.c，确认里面 cmdPropRadioDivSetup 初值中的.formatConf.whitenMode 为 0x02, 使能自动白化功能，如图 10。

```
rfc_CMD_PROP_RADIO_DIV_SETUP_t RF_cmdPropRadioDivSetup =
{
    .commandNo = 0x3807,
    .status = 0x0000,
    .pNextOp = 0, // INSERT APPLICABLE POINTER: (uint8_t*)&xxx
    .startTime = 0x00000000,
    .startTrigger.triggerType = 0x0,
    .startTrigger.bEnaCmd = 0x0,
    .startTrigger.triggerNo = 0x0,
    .startTrigger.pastTrig = 0x0,
    .condition.rule = 0x1,
    .condition.nSkip = 0x0,
    .modulation.modType = 0x1,
    .modulation.deviation = 0x64,
    .symbolRate.preScale = 0xF,
    .symbolRate.rateWord = 0x8000,
    .rxBw = 0x24,
    .preamConf.nPreamBytes = 0x4,
    .preamConf.preamMode = 0x0,
    .formatConf.nSwBits = 0x20,
    .formatConf.bBitReversal = 0x0,
    .formatConf.bMsbFirst = 0x1,
    .formatConf.fecMode = 0x0,
    .formatConf.whitenMode = 0x02,
    .config.frontEndMode = 0x0,
    .config.biasMode = 0x1,
```

Figure 10 – 使能自动白化功能

在 smartrf_settings.c 文件中，在 pOverrides 数组的(uint32_t)0xFFFFFFFF 前，添加下面添加对应的多项式设定命令，如图 11 所示；

```
// Overrides for CMD_PROP_RADIO_DIV_SETUP
static uint32_t pOverrides[] =
{
    MCE_RFE_OVERRIDE(0,4,0,1,0,0),
    HW_REG_OVERRIDE(0x4038,0x0037),
    (uint32_t)0x000684A3,
    HW_REG_OVERRIDE(0x4020,0x7F00),
    HW_REG_OVERRIDE(0x4064,0x0040),
    (uint32_t)0xB1070503,
    (uint32_t)0x05330523,
    (uint32_t)0x0A480583,
    (uint32_t)0x7AB80603,
    ADI_REG_OVERRIDE(1,4,0x9F),
    ADI_HALFREG_OVERRIDE(1,7,0x4,0x4),
    (uint32_t)0x02010403,
    (uint32_t)0x00108463,
    (uint32_t)0x04B00243,
    ADI_HALFREG_OVERRIDE(0,61,0xF,0xD),
    (uint32_t)0x00038883,
    HW_REG_OVERRIDE(0x6084,0x35F1),
    HW_REG_OVERRIDE(0x6088,0x411A),
    HW_REG_OVERRIDE(0x608C,0x8213),
    (uint32_t)0x00FB88A3,
    ADI_REG_OVERRIDE(0,12,0xF8),
    HW32_ARRAY_OVERRIDE (0x2000, 1),
    // set new whitener polynomial:
    0x22000000,
    // new polynomial: x^7 + x^4 + 1 (PN7 sequence), shifted to most significant bits
    0x000001E3 | (0x2A << 25),
    // set the initial value
    (uint32_t)0xFFFFFFFF,
};
```

Figure 11 – 修改白化多项式

第一条为设定多项式寄存器 PHAPOLY0 的地址 0x2000。

第二条为设置多项式$X^7 + X^4 + 1$的值；注意，X^7 隐式的存在最高的 Bit32(这里的最高位是 Bit31，所以 X^7 这里不显示)；X^4 对应第 Bit29,X^0 对应 Bit25，Bit24~Bit0 为保留，未使用。

第三条为设定白化的初始值，由 7 位 PN7 伪随机码初始值左移 25 位后的数值与 0x000001E3 异或得到。

完成上诉设置后，测试与 A7219 可白化互通，同时，此方法也适用于其他白化算法，只要知道对应白化算法的多项式以及伪序列的初始值。

参考资料

1. *CC13xx, CC26xx SimpleLink™ Wireless MCU Technical Reference Manual(SWCU117F)Battery Charging Speification Revision 1.2*
2. *CC1310 SimpleLink™ Ultra-Low-Power Sub-1 GHz Wireless MCU(SWRS181C)*

有关 TI 设计信息和资源的重要通知

德州仪器 (TI) 公司提供的技术、应用或其他设计建议、服务或信息，包括但不限于与评估模块有关的参考设计和材料（总称“TI 资源”），旨在帮助设计人员开发整合了 TI 产品的 应用； 如果您（个人，或如果是代表贵公司，则为贵公司）以任何方式下载、访问或使用了任何特定的 TI 资源，即表示贵方同意仅为该等目标，按照本通知的条款进行使用。

TI 所提供的 TI 资源，并未扩大或以其他方式修改 TI 对 TI 产品的公开适用的质保及质保免责声明；也未导致 TI 承担任何额外的义务或责任。TI 有权对其 TI 资源进行纠正、增强、改进和其他修改。

您理解并同意，在设计应用时应自行实施独立的分析、评价和 判断， 且应全权负责并确保 应用的安全性， 以及您的 应用 （包括应用中使用的所有 TI 产品）） 应符合所有适用的法律法规及其他相关要求。你就您的 应用声明，您具备制订和实施下列保障措施所需的一切必要专业知识，能够 (1) 预见故障的危险后果，(2) 监视故障及其后果，以及 (3) 降低可能导致危险的故障几率并采取适当措施。您同意，在使用或分发包含 TI 产品的任何 应用前， 您将彻底测试该等 应用 和该等应用所用 TI 产品的 功能。除特定 TI 资源的公开文档中明确列出的测试外，TI 未进行任何其他测试。

您只有在为开发包含该等 TI 资源所列 TI 产品的 应用时， 才被授权使用、复制和修改任何相关单项 TI 资源。但并未依据禁止反言原则或其他法理授予您任何TI知识产权的任何其他明示或默示的许可，也未授予您 TI 或第三方的任何技术或知识产权的许可，该等产权包括但不限于任何专利权、版权、屏蔽作品权或与使用TI产品或服务的任何整合、机器制作、流程相关的其他知识产权。涉及或参考了第三方产品或服务的信息不构成使用此类产品或服务的许可或与其相关的保证或认可。使用 TI 资源可能需要您向第三方获得对该等第三方专利或其他知识产权的许可。

TI 资源系“按原样”提供。TI 兹免除对 TI 资源及其使用作出所有其他明确或默认的保证或陈述，包括但不限于对准确性或完整性、产权保证、无屡发故障保证，以及适销性、适合特定用途和不侵犯任何第三方知识产权的任何默认保证。

TI 不负责任何申索，包括但不限于因组合产品所致或与之有关的申索，也不为您辩护或赔偿，即使该等产品组合已列于 TI 资源或其他地方。对因 TI 资源或其使用引起或与之有关的任何实际的、直接的、特殊的、附带的、间接的、惩罚性的、偶发的、从属或惩戒性损害赔偿，不管 TI 是否获悉可能会产生上述损害赔偿，TI 概不负责。

您同意向 TI 及其代表全额赔偿因您不遵守本通知条款和条件而引起的任何损害、费用、损失和/或责任。

本通知适用于 TI 资源。另有其他条款适用于某些类型的材料、TI 产品和服务的使用和采购。这些条款包括但不限于适用于 TI 的半导体产品 (http://www.ti.com/sc/docs/stdterms.htm)、评估模块和样品 (http://www.ti.com/sc/docs/sampterms.htm) 的标准条款。

邮寄地址：上海市浦东新区世纪大道 1568 号中建大厦 32 楼，邮政编码：200122