Application Note

CC13xx IQ 样本

摘要

本应用报告介绍了 CC13xx SimpleLink™ Sub-1GHz 超低功耗无线微控制器 (MCU) 的 IQ 转储补丁。

本文中所讨论的工程配套资料和源代码可从以下链接下载：
http://www.ti.com/lit/zip/swra571。smartrf_settings.c 文件适用于 CC1310，但可以按照节 2.1.1 和节 2.1.2 中说明的步骤为其他 CC13x0 和 CC13x2 器件提供类似的文件。

内容

插图清单

表格清单

商标

1 引言

CC13xx SimpleLink™ Sub-1GHz 超低功耗无线微控制器 (MCU) 的核心是 ARM® Cortex®-M3 (CC13x0) 或 ARM Cortex-M4F (CC13x2) 系列处理器和自主射频内核，前者用于操控应用，后者用于操控无线传输数字位所需的所有低级无线电控制和处理。客户通常使用 CM3/CM4F 在物理层之上实现其应用/高级协议，但也可用于实现其他新颖或传统的物理层调制方案。为此，CM3/CM4F 需要在 RX 模式下获取原始 IQ 样本。默认使用 genfsk/prop PHY 时，需用射频内核获取 IQ 样本，并用专用补丁自动将 IQ 样本复制到部分读取 RX 条目。本应用手册介绍了如何使用 IQ 转储补丁获取这些 IQ 样本。

2 IQ 转储补丁

IQ 转储补丁 (rf_patch_mce_iqdump.h) 可在两种不同模式下运行；IQFifoBlind 和 IQFifoSync。IQFifoBlind 模式会立即开始复制 IQ 样本，而 IQFifoSync 模式会在检测到同步字后开始复制 IQ 样本。工作模式由 MCE_RFE 覆盖选择（参阅表 2-2）。对于这两种模式，IQ 样本通过射频内核的内部 FIFO 复制到系统 RAM 中的一个或多个部分读取 RX 条目。应用只是等待 RX_ENTRY_DONE 中断，表明部分读取条目已满。

IQ 采样率固定为 4 倍过采样，IQ 采样大小为 12 位。这意味着每个 IQ 对将占用 RAM 中的 3 个字节，格式如表 2-1 所示。格式为有符号型，表示 MSB 是符号位（二进制补码格式）。

表 2-1. 存储在 RAM 中的 IQ 样本格式

字节	位定义							
0	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0
1	Q_3	Q_2	Q_1	Q_0	I_{11}	I_{10}	I_9	I_8
2	Q_{11}	Q_{10}	Q_9	Q_8	Q_7	Q_6	Q_5	Q_4

该补丁具有内置测试模式，其中 IQ 样本被替换为两个计数器值。I-sample 替换为递增的计数器值，Q-sample 替换为递减的计数器值。测试模式通过以下寄存器覆盖启用：

```
HW_REG_OVERRIDE(0x52B4, 0x070D) // CC13x0
HW_REG_OVERRIDE(0x5328, 0x070D) // CC13x2
```

2.1 建议的运行限制

在 RX 中使用 IQ 转储补丁时，CC13x0 的数据速率限制为 12.5kbps，CC13x2 的数据速率限制为 25kbps。在 TX 中，补丁可以在与 genfsk/prop PHY 相同的运行限制内使用。应使用 SmartRF™ Studio [1] 中的 50kbps 设置作为起点。关联的 zip 文件包含一个 smartrf_settings.c 文件（对于 CC1310），该文件具有要与补丁一同使用的完整覆盖列表和 API 设置。

2.1.1 寄存器覆盖

运行 IQ 转储补丁时，需要修改 MCE_RFE 覆盖 [1]。表 2-2 显示了应如何执行此操作。此外，在运行补丁时还需要添加另外一个覆盖。

表 2-2. 覆盖和工作模式

覆盖	说明

MCE_RFE_OVERRIDE(1,0,2,1,0,0) // CC13x0
MCE_RFE_OVERRIDE(1,0,2,0,4,0) // CC13x2
	将工作模式设为 IQFifoBlind

MCE_RFE_OVERRIDE(1,0,3,1,0,0) // CC13x0
MCE_RFE_OVERRIDE(1,0,3,0,4,0) // CC13x2
	将工作模式设为 IQFifoSync

(uint32_t)0x001082C3	设置以避免内部 FIFO 溢出
HW_REG_OVERRIDE(0x52B4, 0x070D)（可选）// CC13x0 HW_REG_OVERRIDE(0x5328, 0x070D)（可选）// CC13x2	启用内置测试模式。只能在测试补丁时添加。相关详细信息，请参阅节 3（可选）。

[1] 覆盖列表中只能有一个 MCE_RFE。

此外，您需要添加补丁并更新 TI_RTOS RF 模式对象：

CC13x0：

```
#include DeviceFamily_constructPath(rf_patches/rf_patch_cpe_genfsk.h)
#include DeviceFamily_constructPath(rf_patches/rf_patch_mce_iqdump.h)
#include DeviceFamily_constructPath(rf_patches/rf_patch_rfe_genfsk.h)
#include "smartrf_settings.h"
// TI-RTOS RF Mode Object
RF_Mode RF_prop =
{
    .rfMode = RF_MODE_PROPRIETARY_SUB_1,
    .cpePatchFxn = &rf_patch_cpe_genfsk,
    .mcePatchFxn = &rf_patch_mce_iqdump,
    .rfePatchFxn = &rf_patch_rfe_genfsk,
};
```

CC13x2：

```
#include DeviceFamily_constructPath(rf_patches/rf_patch_cpe_prop.h)
#include DeviceFamily_constructPath(rf_patches/rf_patch_mce_iqdump.h)
#include "smartrf_settings.h"
// TI-RTOS RF Mode Object
RF_Mode RF_prop =
{
    .rfMode = RF_MODE_AUTO,
    .cpePatchFxn = &rf_patch_cpe_prop,
    .mcePatchFxn = &rf_patch_mce_iqdump,
    .rfePatchFxn = 0,
};
```

2.1.2 API 配置

使用补丁时，必须对从 SmartRF Studio 导出的 API 进行一些更改。CMD_PROP_RADIO_DIV_SETUP 中的 formatConf.bMsbFirst 必须设置为 0，从而先传输 LSB；CMD_PROP_RX 中的 maxPktLen 必须设置为 0，从而实现不受限制的数据包长度。RX 带宽应设置为 39/38.9kHz (CC13x0/CC13x2)，建议的起始偏差是将其设置为数据速率的一半（参阅表 2-3）。

表 2-3. API 修改

API 字段	值	注释
RF_cmdPropRadioDivSetup.formatConf.bMsbFirst	0	首先传输最低有效位
RF_cmdPropRx.maxPktLen	0	长度不受限制
RF_cmdPropRadioDivSetup.modulation.deviation	0x19	6.25kHz
RF_cmdPropRadioDivSetup.rxBw	0x20 0x4D	39kHz (CC13x0) 38.9kHz (CC13x2)
RF_cmdPropRadioDivSetup.symbolRate.rateWord	0x2000	12.5kbps

3 构建软件示例

若要测试补丁的射频性能，请参阅 *rfPacketRX* 示例（下载 [2] 或 [3] 时可提供该示例）。

smartrf_settings.c 文件必须替换为可从以下链接下载的 zip 文件：http://www.ti.com/lit/zip/swra571。必须对 rfPacketRX.c 进行以下修改才能测试补丁：

1. 确定您需要多少个 IQ 样本对。

```
#define NUMBER_OF_SAMPLE_PAIRS 300
```

将 NUMBER_OF_SAMPLE_PAIRS 设为 300 意味着使用的每个数据条目必须有 300 x 3 字节的空间。

2. 为接收的数据配置两个部分读取缓冲区。确保这些缓冲区是 4 字节对齐的。

```
#define PARTIAL_RX_ENTRY_HEADER_SIZE 12

#if defined(__TI_COMPILER_VERSION__)
#pragma DATA_ALIGN (rxDataEntryBuf1, 4);
static uint8_t rxDataEntryBuf1[PARTIAL_RX_ENTRY_HEADER_SIZE +
                               (NUMBER_OF_SAMPLE_PAIRS * 3)];
#pragma DATA_ALIGN (rxDataEntryBuf2, 4);
static uint8_t rxDataEntryBuf2[PARTIAL_RX_ENTRY_HEADER_SIZE +
                               (NUMBER_OF_SAMPLE_PAIRS * 3)];
#endif
rfc_dataEntryPartial_t* partialReadEntry1 = (rfc_dataEntryPartial_t*)&rxDataEntryBuf1;
rfc_dataEntryPartial_t* partialReadEntry2 = (rfc_dataEntryPartial_t*)&rxDataEntryBuf2;
rfc_dataEntryPartial_t* currentReadEntry = (rfc_dataEntryPartial_t*)&rxDataEntryBuf1;

void *mainThread(void *arg0)
{
    RF_Params rfParams;
    RF_Params_init(&rfParams);
    partialReadEntry1->length = (NUMBER_OF_SAMPLE_PAIRS * 3) + 4;
    partialReadEntry1->config.type = DATA_ENTRY_TYPE_PARTIAL;
    partialReadEntry1->status = DATA_ENTRY_PENDING;

    partialReadEntry2->length = (NUMBER_OF_SAMPLE_PAIRS * 3) + 4;
    partialReadEntry2->config.type = DATA_ENTRY_TYPE_PARTIAL;
    partialReadEntry2->status = DATA_ENTRY_PENDING;

    partialReadEntry1->pNextEntry = (uint8_t*)partialReadEntry2;
    partialReadEntry2->pNextEntry = (uint8_t*)partialReadEntry1;

    dataQueue.pCurrEntry = (uint8_t*)partialReadEntry1;
    dataQueue.pLastEntry = NULL;
```

3. 删除 *RFQueue_defineQueue* 和 *RF_cmdPropRX* 的修改，但 *RF_cmdPropRx.pQueue* 除外。

```
// if( RFQueue_defineQueue(&dataQueue,
//                         rxDataEntryBuffer,
//                         sizeof(rxDataEntryBuffer),
//                         NUM_DATA_ENTRIES,
//                         MAX_LENGTH + NUM_APPENDED_BYTES))
//{
//    /* Failed to allocate space for all data entries */
//    while(1);
//}

RF_cmdPropRx.pQueue = &dataQueue;
/* Discard ignored packets from Rx queue */
    // RF_cmdPropRx.rxConf.bAutoFlushIgnored = 1;
    /* Discard packets with CRC error from Rx queue */
    // RF_cmdPropRx.rxConf.bAutoFlushCrcErr = 1;
    /* Implement packet length filtering to avoid PROP_ERROR_RXBUF */
    // RF_cmdPropRx.maxPktLen = MAX_LENGTH;
    // RF_cmdPropRx.pktConf.bRepeatOk = 1;
    // RF_cmdPropRx.pktConf.bRepeatNok = 1;
```

4. 在回调中实现 IQ 样本的处理。在回调中，只需从数据条目中读取样本，以使数据条目可用于新样本。IQ 样本的处理应在回调之外完成，本应用报告不对此赘述。下面的代码简单地展示了如何获取样本和处理队列。

```
void callback(RF_Handle h, RF_CmdHandle ch, RF_EventMask e)
{
    if (e & RF_EventRxEntryDone)
    {
        // Toggle pin to indicate RX
        PIN_setOutputValue(pinHandle,
                           Board_PIN_LED2,!PIN_getOutputValue(Board_PIN_LED2));
        // Get a pointer to the first IQ sample byte
        packetDataPointer = &currentReadEntry->rxData;
        //------------------------------------------------------------------------
        // Implement code for handling the IQ data
        //    .
        //    .
        //    .
        //    .
        //------------------------------------------------------------------------
        currentReadEntry->status = DATA_ENTRY_PENDING;
        currentReadEntry = (rfc_dataEntryPartial_t*)currentReadEntry->pNextEntry;
    }
}
```

4 使用内置测试模式测试补丁

若要测试数据条目是否设置正确以及补丁是否正常工作，您可以启用内置测试模式（参阅表 2-2），并声明可用来保存“接收到的”I 和 Q 样本的两个数组（iSamples 和 qSamples）。

```
#define NUMBER_OF_BUFFERS 5
static uint16_t iSamples[NUMBER_OF_SAMPLE_PAIRS*NUMBER_OF_BUFFERS];
static uint16_t qSamples[NUMBER_OF_SAMPLE_PAIRS*NUMBER_OF_BUFFERS];
```

出于测试目的，将 *NUMBER_OF_SAMPLE_PAIRS* 设置为一个较低的数字 [2]，从而能够更轻松地遍历数组以看看一切是否正常。

```
#define NUMBER_OF_SAMPLE_PAIRS 8
```

在回调（应该实现了用于处理这些样本的代码）中，添加了以下代码：

```
static uint16_t index = 0;
void callback(RF_Handle h, RF_CmdHandle ch, RF_EventMask e)
{
    if (e & RF_EventRxEntryDone)
    {
        // Toggle pin to indicate RX
        PIN_setOutputValue(ledPinHandle,
                           Board_PIN_LED2,!PIN_getOutputValue(Board_PIN_LED2));
        // Get a pointer to the first IQ sample byte
        packetDataPointer = &currentReadEntry->rxData;
        //------------------------------------------------------------------------
        // Implement code for handling the IQ data
        // In this example, I and Q data are simply copied into two separate array
        {
            uint16_t i;
            // IQ Sample Handling
            for (i = index; i < (NUMBER_OF_SAMPLE_PAIRS + index); i++)
            {
                iSamples[i] = (((*(packetDataPointer + 1)) << 8) |
                               (*packetDataPointer)) & 0x0FFF;
                qSamples[i] = (((*(packetDataPointer + 2)) << 8) |
                               (*(packetDataPointer + 1))) >> 4;
                packetDataPointer += 3;
            }
        }
        index += NUMBER_OF_SAMPLE_PAIRS;
        if (index == (NUMBER_OF_SAMPLE_PAIRS*NUMBER_OF_BUFFERS))
        {
            index = 0;
        }
        //------------------------------------------------------------------------
        currentReadEntry->status = DATA_ENTRY_PENDING;
        currentReadEntry = (rfc_dataEntryPartial_t*)currentReadEntry->pNextEntry;
    }
}
```

[2] 在此示例中，*NUMBER_OF_SAMPLE_PAIRS* 不能设为低于 8，因为这会使数据条目溢出 (*RF_cmdPropRx.status* = *PROP_ERROR_RXOVF*)

图 4-1 显示了五个缓冲区，每个缓冲区有 8 个 IQ 样本对，存储在 iSamples 和 qSamples 数组中，每个数组保存 40 个样本 (*NUMBER_OF_BUFFERS* · *NUMBER_OF_SAMPLE_PAIRS*)。

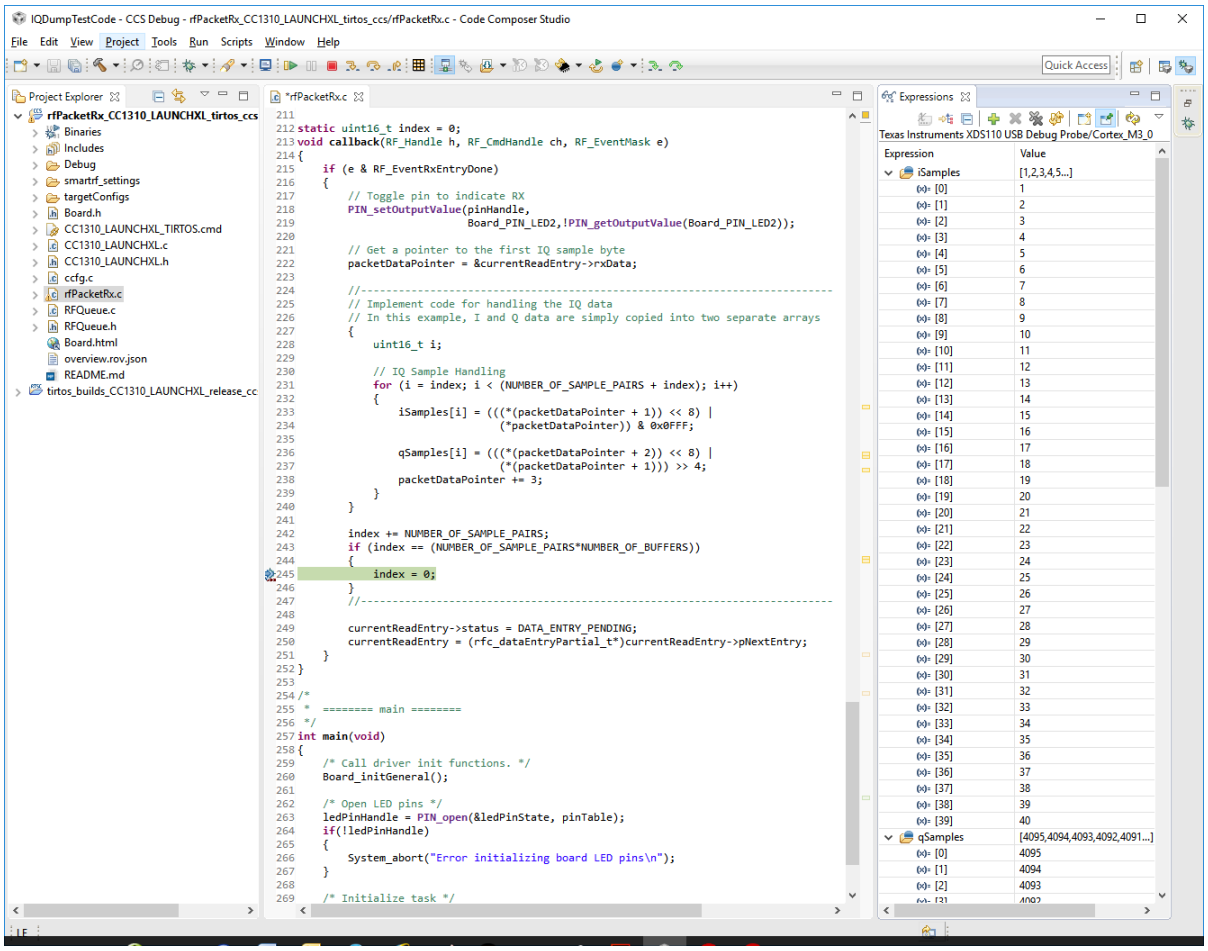

图 4-1. 存储为 I 和 Q 样本的内置测试模式

5 参考文献

1. *SmartRF Studio 7*
2. *SimpleLink™ Sub-1GHz CC13x0 软件开发套件*
3. *SimpleLink™ CC13x2 和 CC26x2 软件开发套件*

6 修订历史记录

注：以前版本的页码可能与当前版本的页码不同

Changes from Revision A (April 2019) to Revision B (August 2020)	Page
• 更新了整个文档中的表格、图和交叉参考的编号格式。	1
• 使用适用于 CC13x2 的 MCE_RFE_OVERRIDE 更新了表 2-2 *覆盖和工作模式*	2

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265