Application Report
SWRA571–August 2017

CC13x0 IQ Samples

Siri Johnsrud and Trond Rognerud

ABSTRACT

This application report describes an IQ dump patch for the CC13xx SimpleLink™ Sub-1 GHz ultra-low power wireless microcontroller (MCU).

Project collateral and source code mentioned in this document can be downloaded from the following link: http://www.ti.com/lit/zip/swra571.

Contents

List of Figures

List of Tables

Trademarks

SimpleLink, SmartRF are trademarks of Texas Instruments.
ARM, Cortex are registered trademarks of ARM Limited.
All other trademarks are the property of their respective owners.

1 Introduction

CC13xx SimpleLink™ Sub-1 GHz ultralow power wireless microcontroller (MCU) is centered around an ARM® Cortex®-M series processor (CM3) that handles the application and an autonomous RF Core that handles all the low-level radio control and processing needed to transfer digital bits over the air. Normally the customers use the CM3 to implement their application/high level protocols on top of the physical layer, but it is also possible to use the CM3 to implement other novel or legacy physical layer modulation schemes. In order to do so, the CM3 requires access to the raw IQ samples in RX mode. With the default genfsk PHY, IQ samples are not available outside RF Core and a dedicated patch is needed to automatically copy IQ samples to a partial read RX entry. This application note describes how to get access to these IQ samples using the IQ dump patch.

Copyright © 2017, Texas Instruments Incorporated

2 IQ Dump Patch

The IQ Dump patch (rf_patch_mce_iqdump.h) can run in two different modes; IQFifoBlind and IQFifoSync. IQFifoBlind mode starts copying IQ samples immediately while IQFifoSync mode starts copying IQ samples after a sync word has been detected. The mode of operation is selected by the MCE_RFE override (see Table 1). For both modes, IQ samples are copied through the RF Core's internal FIFO to one or more partial read RX entries in the system RAM. The application simply waits for an RX_ENTRY_DONE interrupt saying that a partial read entry is full.

The IQ sample rate is fixed to 4 times oversampling and the IQ sample size is 12 bits. This means that each IQ pair will occupy 3 bytes in RAM in the format shown in Table 1. The format is signed meaning that MSB is the sign bit (two's complement format).

Table 1. Format of IQ Samples Stored in RAM

Byte	Bit Definition							
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

The patch has a built-in test pattern where the IQ samples are replaced with two counter values. The I-sample is replaced with an increasing counter value and the Q-sample is replaced with a decreasing counter value. The test pattern is enabled by the following register override:

```
HW_REG_OVERRIDE(0x52B4, 0x070D)
```

2.1 Recommended Operating Limits

When using the IQ Dump patch in RX the data rate is limited upwards to 12.5 kbps. In TX, the patch can be used within the same operating limits as the genfsk PHY (10 - 500 kbps 2-GFSK settings from SmartRF Studio). The 50-kbps settings from SmartRF™ Studio [1] should be used as a starting point. The associated zip file contains a smartrf_settings.c file that has the complete override list and API settings to be used with the patch.

2.1.1 Register Overrides

The MCE_RFE override (1) needs to be modified when running the IQ Dump patch. Table 2 shows how this should be done. In addition, there is one other override necessary to add when running the patch.

(1) Only one MCE_RFE can be present in the override list.

Table 2. Overrides and Mode of Operation

Override	Description/Comment
MCE_RFE_OVERRIDE(1,0,2,1,0,0)	Setting the mode of operation to IQFifoBlind
MCE_RFE_OVERRIDE(1,0,3,1,0,0)	Setting the mode of operation to IQFifoSync
(uint32_t)0x001082C3	Set to avoid internal FIFO overflow
HW_REG_OVERRIDE(0x52B4, 0x070D) (optional)	Enable built-in test pattern. Should only be included when testing the patch. For more details, see Section 3 (optional).

In addition you need to include the patch and update the TI_RTOS RF Mode Object:

```
#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,
```


Copyright © 2017, Texas Instruments Incorporated

```
    .mcePatchFxn = &rf_patch_mce_iqdump,
    .rfePatchFxn = &rf_patch_rfe_genfsk,
};
```

2.1.2 API Configuration

When using the patch some changes have to be done to the API exported from SmartRF Studio. formatConf.bMsbFirst in CMD_PROP_RADIO_DIV_SETUP must be set to 0 to allow for LSB to be transmitted first and maxPktLen in CMD_PROP_RX must be set to 0 for unlimited packet length. The RX Bandwidth should be set to 39 kHz, and a good starting point for the deviation is to set it to half the data rate (see Table 3).

Table 3. API Modifications

API Field	Value	Comment
RF_cmdPropRadioDivSetup.formatConf.bMsbFirst	0	Least significant bit transmitted first
RF_cmdPropRx.maxPktLen	0	Unlimited length
RF_cmdPropRadioDivSetup.modulation.deviation	0x19	6.25 kHz
RF_cmdPropRadioDivSetup.rxBw	0x20	39 kHz
RF_cmdPropRadioDivSetup.symbolRate.preScale	0xF	12.5 kbps

3 Building a Software Example

To test the RF performance of the patch, see the *rfPacketRX* example available when downloading [2]. For more information, see the *Proprietary RF Quick Start Guide* from this download. A local link to this guide can be found in the documentation_overview_simplelink_cc13x0_sdk.html found here: C:\ti\simplelink_cc13x0_sdk_x_xx_xx_xx\docs (assuming that installation has been to the default location).

The smartrf_settings.c file must be replaced with the one from the zip file that can be downloaded from the following link: http://www.ti.com/lit/zip/swra571. The following modifications must be done to rfPacketRX.c to be able to test the patch:

1. Define how many IQ sample pairs you want.

   ```
   #define NUMBER_OF_SAMPLE_PAIRS 300
   ```

   Setting `NUMBER_OF_SAMPLE_PAIRS` to 300 means that each data entry used must have room for 300 x 3 bytes.

2. Configure two partial read buffers for the received data. Make sure that the buffers are 4 byte aligned.

   ```
   #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;

   static void rxTaskFunction(UArg arg0, UArg arg1)
   {
       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;
   ```


Copyright © 2017, Texas Instruments Incorporated

```
    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. Remove *RFQueue_defineQueue* and the modifications of *RF_cmdPropRX*, except for the *RF_cmdPropRx.pQueue*.

```
// if( RFQueue_defineQueue(&amp;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;
// RF_cmdPropRx.rxConf.bAutoFlushIgnored = 1;
// RF_cmdPropRx.rxConf.bAutoFlushCrcErr = 1;
// RF_cmdPropRx.maxPktLen = MAX_LENGTH;
// RF_cmdPropRx.pktConf.bRepeatOk = 1;
// RF_cmdPropRx.pktConf.bRepeatNok = 1;
```

4. Implement handling of the IQ samples in the callback. In the callback the samples should simply be read from the data entries to make the data entries available for new samples. The processing of the IQ samples should be done outside the callback. It is not the scope of this application report to show how this can be done. The code below simply shows how to get access to the samples and how to handle the queue.

```
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 Testing the Patch Using the Built-In Test Pattern

To test that the data entries are set up correctly and that the patch is working you can enable the built-in test pattern (see Table 2) and declare two arrays (iSamples and qSamples) that can hold the “received” I and Q samples.

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


Copyright © 2017, Texas Instruments Incorporated

```
static uint16_t qSamples[NUMBER_OF_SAMPLE_PAIRS*NUMBER_OF_BUFFERS];
```

For test purposes, set *NUMBER_OF_SAMPLE_PAIRS* to a low number [1] to easier be able to go through the array to see that everything is OK.

```
#define NUMBER_OF_SAMPLE_PAIRS 8
```

In the callback, where code for handling the samples should be implemented, the following code was added:

```
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(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
        // 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;
    }
}
```

(1) In the example above, *NUMBER_OF_SAMPLE_PAIRS* cannot be set lower than 8, as this will make the data entry overflow (*RF_cmdPropRx.status* = *PROP_ERROR_RXOVF*)

Copyright © 2017, Texas Instruments Incorporated

Figure 1 shows the five buffers with eight IQ sample pairs in each stored in an iSamples and qSamples array, each holding 40 samples (*NUMBER_OF_BUFFERS · NUMBER_OF_SAMPLE_PAIRS*).

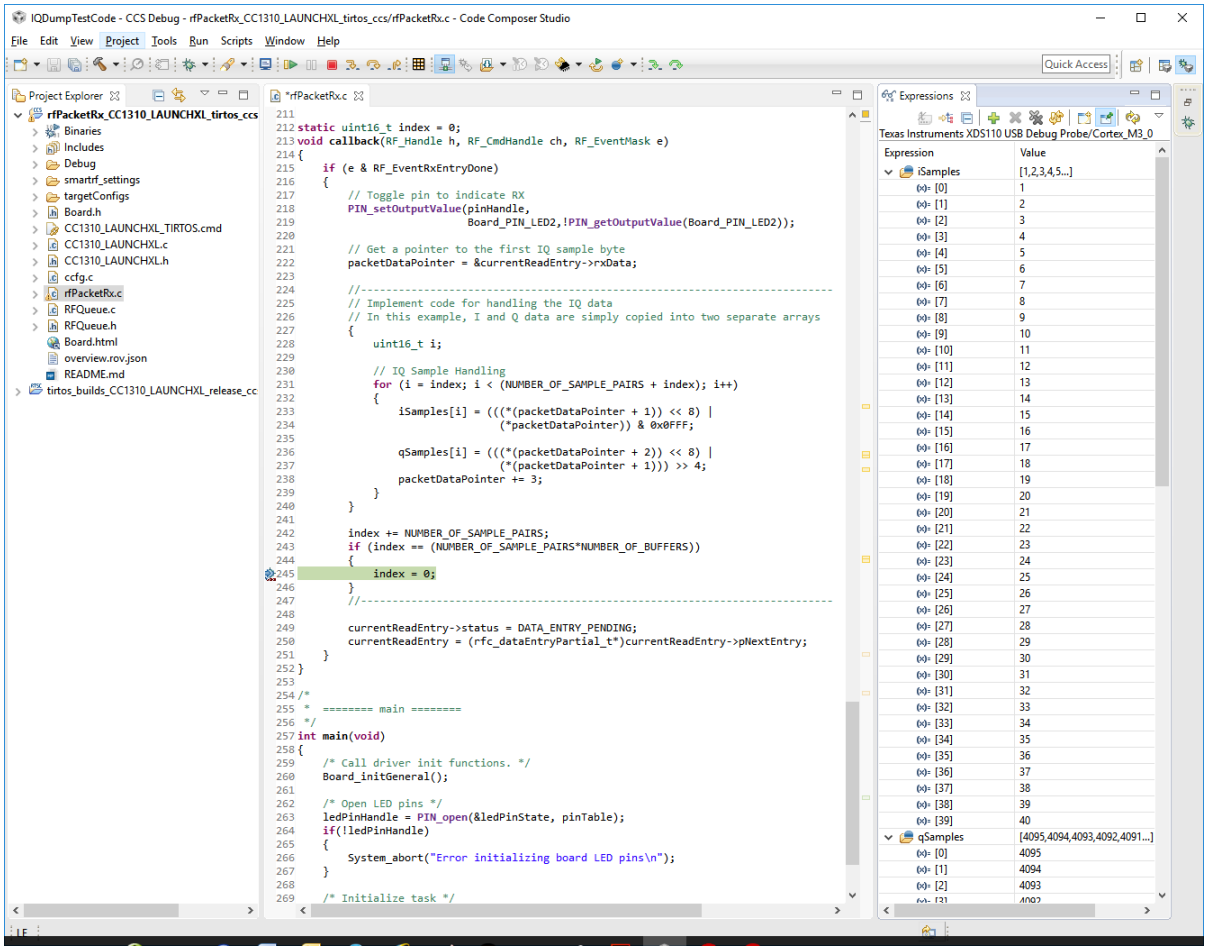

Figure 1. Built-In Test Pattern Stored as I and Q Samples

5 References

1. *SmartRF Studio 7*
2. *SimpleLink™ Sub-1 GHz CC13x0 Software Development Kit*

Copyright © 2017, Texas Instruments Incorporated

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2017, Texas Instruments Incorporated