

# Guide for Manufacture Testing of RF4CE Products

#### Low-Power Wireless Products

#### ABSTRACT

This application report describes a sample implementation of a manufacturing line test program for RF4CE products. It is based on RemoTI-Linux, which is the Linux host driver for CC253x RemoTI Network Processor (RNP). It includes the ability to setup a fake pairing between the remote control and target side, how to enter test mode, how to change RF parameters, and so forth. It helps customers to understand RemoTI-Linux structure and utilize the API of TI RF4CE stack. Based on this application report, customers can easily implement their own test flow and meet varying factory environments.

The last section discusses the way to perform basic RF functions with test equipment. Besides implementing the test program, customers can perform RF tests, too.

#### Contents

1	RemoTI-Linux Introduction	. 2
2	Implement Test Program	5
	Setup for RF Test	
4	RemoTI Mass Production Test	21
5	Summary	22
6	References	23

#### List of Figures

1	RemoTI Network Processor Component Architecture	3
2	RemoTI-Linux Software Architecture	3
3	RemoTI-Linux Application Architecture	4
4	Network Enabling for NPI Client	4
5	The setup for the Test Program Implementation	5
6	Controller Could Send Packet to Target After Doing Fake Pairing	9
7	Setup for RF Test	12
8	Tx Test Mode0 Launch on Linux Laptop	13
9	Tx Test Mode0 Perform on Spectrum Analyzer	13
10	Tx test Mode1 Launch on Linux Laptop	14
11	Tx Test Mode1 Perform on Spectrum Analyzer	15
12	Rx Test Launch on Linux Laptop	16
13	Rx Test Perform on Spectrum Analyzer	16
14	PER Test Launch on Linux Laptop	17
15	Sniff Log While Performing PER Test	18
16	NPI Server and Client Run on Different Devices	18
17	Setup for PER Test With SmartRF Studio	19
18	Configuration on SmartRF Studio	21
19	Execute massProduction_Inx_x86_client	22
20	Result of RemoTI Mass Production Test	22

#### List of Tables

All trademarks are the property of their respective owners.



RemoTI-Linux Introduction

1	Acronyms	2
2	Configuration for Fake Pairing on Target and Controller	. 8
3	Pairing Table After Fake Pairing on Target and Controller	. 8
4	Configuration for Device Own Information on Target and Controller	. 9
5	Definition of nodeCapabilities	9
6	Test Modes Supported by RTI_TestModeReq	10
7	Server Configuration Difference on Target and Controller	17

### **1** RemoTI-Linux Introduction

This section introduces the architecture of RemoTI-Linux, which utilizes the RNP architecture for a Linux host. To simplify the development effort, this document provides some code snips that are described in the following sections. Then, you can implement your own manufacturing test program based on the RemoTI-Linux source code.

## 1.1 Acronyms Used in This Document

Acronym	Description
Device	A physical object consisting of an IEEE 802.15.4 radio
Node	A device containing RF4CE functionality
Target	A Node that implements the Target functionality as defined in the RF4CE specification (for example, a TV)
Controller	A Node that implements the Controller functionality as defined in the RF4CE specification (for example, a remote control)
ACK	Acknowledge
API	Application Programming Interface
ARC	Advance Remote Control
DUT	Device under test
EM	Evaluation module
IEEE	Institute of Electrical & Electronics Engineers, Inc.
LAN	Local Area Network
NPI	Network Processor Interface
RTIS	RemoTI application framework surrogate
RemoTI-Linux	A RF4CE application for a Linux host using a CC253x RNP
RF	Radio Frequency
RF4CE	Radio Frequency for Consumer Electronics
RNP	RemoTI Network Processor. An application configuration of the RemoTI stack that configures the CC253x device as a network processor
RTI	RemoTI Application Framework
STB	Set-Top Box
TE	Target Emulator
ТІ	Texas Instruments Incorporated
ZID	Zigbee Input Device, a name for a ZigBee RF4CE profile
ZRC	Zigbee Remote Control, a name for a Zigbee RF4CE profile

### Table 1. Acronyms



# 1.2 RemoTI Network Processor Architecture

Figure 1 shows the architecture of RemoTI Network Processor [1]. The right side is the network processor. An RTI surrogate module (RTIS) decodes the command frames sent over universal asynchronous receiver/transmitter (UART), serial peripheral interface, (SPI), inter-integrated circuit (I2C), or universal serial bus (USB) and calls appropriate C function of RemoTI application framework [2]. The left side is the application processor. The RTIS module abstracts the same C function interface on host processor, enabling application to call C functions remotely instead of building network processor interface directly.

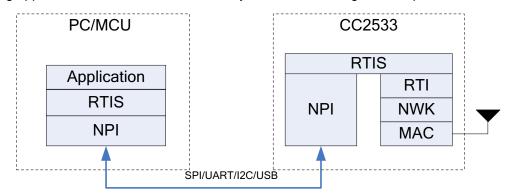


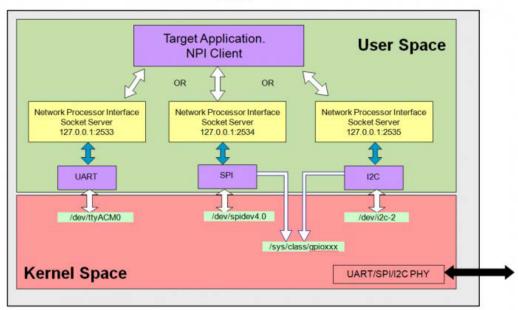
Figure 1. RemoTI Network Processor Component Architecture

# 1.3 Structure of RemoTI-Linux

RemoTI-Linux is a simple RF4CE target application on a linux host [3]. Figure 2 is an overview of the architecture. The linux software is split in two executables:

- An NPI server that creates an IP socket and connects it to the requested serial interface
- An NPI client that runs the RemoTI application

The NPI client sends all commands to the socket. The NPI server takes care of translating them to the RNP. The NPI client uses the RemoTI interface to drive the RNP.



Host

Figure 2. RemoTI-Linux Software Architecture



RemoTI-Linux Introduction

www.ti.com

Figure 3 illustrates the application architecture. The simple application creates an RF4CE network. By default, it allows pairing with any RF4CE controller node using ZRC or ZID profile. Once paired, all data sent by the controller node will be displayed on the terminal.

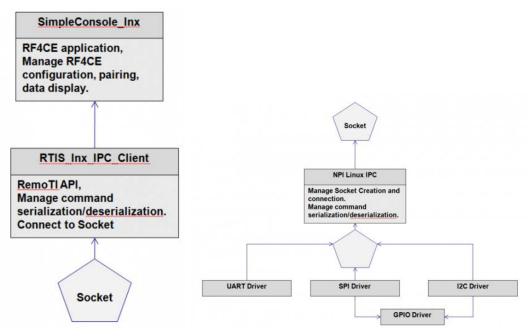


Figure 3. RemoTI-Linux Application Architecture

# 1.4 Network Enabling for NPI Client

Since the communication between NPI server and client adopts IP socket, NPI server and client can run on different devices. Figure 4 describes the scenario where NPI server is running inside the STB and NPI client is running inside a test PC. The STB and the test PC are under the same LAN via Ethernet on the production line. In this case, the NPI client including the test application can run on the test PC and test the CC253x on the actual STB. To enable this scenario, it just needs to set some configuration files. For more information, see Section 3.5.

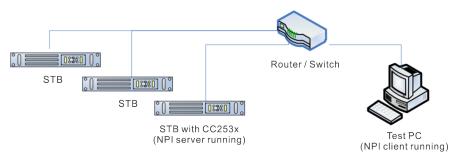


Figure 4. Network Enabling for NPI Client

# 1.5 Installation for RemoTI-Linux

The RemoTI-Linux is open source. It is available on Git HUB [4]. With the different configuration, RemoTI-Linux can perform as the RF4CE target or controller role. On the official release of RemoTI-Linux, there is a "SIMPLE\_Inx\_x86\_client" application project. You can base your project on this, then modify the application for your own manufacturing usage.



# 2 Implement Test Program

The environment of manufacturing line is different from the RND working space. The equipments are limited, much noise exists and products need to be verified on time. Besides, some customers make target only or controller only. The RemoTI-Linux can simulate target or controller function. Customers could utilize devices of the same hardware to test transmitter and receiver function. Figure 5 shows the setup for the implementation environment. Two CC2533 kits are connected to one laptop that runs Linux OS: EM and the RemoTI Target Board. One kit acts as controller, the other acts as target. Multiple instances of the client application can be run simultaneously, preferably in different sessions in separate terminals. For the setup with one controller and one target, two instances must run. The only difference is that the configuration file is used for controller and target.



Figure 5. The setup for the Test Program Implementation

Example 1 lists the default functions supported by RemoTI-Linux. The lastest RemoTI-Linux on the GitHub might be a little different. RemoTI provide many APIs [7]. This section utilizes some APIs to implement useful features for testing RF4CE products.



### Example 1. Default Init Menu list of RemoTI-Linux

### Init MENU:

- 1. Toggle Target/Controller
- 2. Set Node Capabilities
- 3. Supported Profiles
- 4. Supported Devices
- 5. Supported Target Types
- 6. Vendor ID
- 7. Vendor String
- 8. Toggle User String
- 9. User String
- i- Initialize without configuration. (Restore from NV)
- g- Get current configuration from RNP
- I- Show configuration. Note! Not Necessarily the One Written To RNP
- s- Apply Configuration and Move On To Application
- r- Back To Application, Do Not Apply Changes
- c- Start As Controller (default settings)
- t- Start As Target (default settings)



Example 2 provides an example of the modified Menu with code snips. Here, one fakePairing folder is created on "Projects\tools\LinuxHost\application\fakePairing". The NPI client is called FPA\_Inx\_x86\_client.

### Example 2. Updated Main Menu List

Init MENU:
q- exit
0- Config
1- Pairing
2- Unpairing
7- Send Data
8- Clear Pairing Table
9- Display Pairing Table
t- ToggleDEBUG_TIME_ACTIVE on Server
y- ToggleBIG_DEBUG on Server
c- Toggle simple data display
a- Check States
g- Get MAC Channel
r- Reset RNP
m- Show This Menu
f- Enter RF Test Mode
k- Configure Fake Pairing Table

## Example 3. RF Test Mode Menu List

- r- Return to Main Menu
- 1- Perform Tx Mode 0: unmodulated carrier
- 2- Perform Tx Mode 1: pseudo-random data
- 3- Perform Rx Mode
- 4- Perform PER Test
- 5- Perform PER2 by SmartRFStudio
- 6- STOP PER2 by SmartRFStudio
- c- Set Channel
- p- Set TxPower
- n- Set PER Counterl- List Current Channel, TxPower
- m- Show This Menu

### 2.1 Fake Pairing

Pairing is the first step to connect the RF4CE target and controller. But, it may take some time in a noisy environment. It would be very helpful if the pairing tables between DUTs (target and remote) could be configured by programming. Since the pairing process in this case is not negotiated through RF packets, it is called "Fake Pairing".

For RemoTI source code, the structure of "rcnNwkPairingEntry\_t" stores the pairing table of the peer. Table 2 lists an example to configure the pairing table on target and controller. Some elements (for example, panID, securityKey) need to be the same on both sides, while others must mirror each other (for example, NWK Address versus SRC NWK Address).

Element of Pairing Table, rcnNwkPairingEntry_t	on Target	on Controller
	[PAIRING_INFO]	[PAIRING_INFO]
pairingRef	PAIRINGREF=0X00	PAIRINGREF=0X00
devTypeList[]	DEVTYPE0=0x01	DEVTYPE0=0x02
	DEVTYPE1=0x00	DEVTYPE1=0x00
	DEVTYPE2=0x00	DEVTYPE2=0x00
ieeeAddress	IEEE=00 12 4B 00 12 34 01 00	IEEE=00 12 4B 00 12 34 00 01
nwkAddress	NWKADDR=0xAAAA	NWKADDR=0xBBBB
srcNwkAddress	SRCNWKADDR=0xBBBB	SRCNWKADDR=0xAAAA
panld	PANID=0x1234	PANID=0x1234
recCapabilities	RECCAP=0x0C	RECCAP=0x0F
recFrameCounter	RECFRAMECOUNTER=0x00000000	RECFRAMECOUNTER=0x00000000
vendorldentifier	VENDORID=0x0007	VENDORID=0x0007
securityKeyValid	KEY_VALID=1	KEY_VALID=1
securityKey[]	SECKEY=01 01 01 01 02 02 02 02 02 02 02 02 02 02 01 01 01 01 01	SECKEY=01 01 01 01 02 02 02 02 02 02 02 02 02 01 01 01 01
profileDiscs	PROFILE_DISC=0x02	PROFILE_DISC=0x02

### Table 2. Configuration for Fake Pairing on Target and Controller

Table 3 lists the printout on target and controller.

### Table 3. Pairing Table After Fake Pairing on Target and Controller

on Target		on Controller			
*****	**	******			
* Max number of pairing entries	s: 10	* Max number of pairing entries: 10			
*****	**	***********			
* Pairing Index:	0x00	* Pairing Index:	0x00		
* SRC NWK Address:	0xBBBB	* SRC NWK Address:	0xAAAA		
* Logical Channel:	0x0F	* Logical Channel:	0x19		
* IEEE Address:	00:12:4B:00:12:34:01:00	* IEEE Address:	00:12:4B:00:12:34:00:01		
* PAN Id:	0x1234	* PAN Id:	0x1234		
* NWK Address:	0xAAAA	* NWK Address:	0xBBBB		
* Rec Capabilities:	0x0C	* Rec Capabilities:	0x0F		
* Security Key Valid:	0x01	* Security Key Valid:	0x01		
* Vendor Identifier:	0x0007	* Vendor Identifier:	0x0007		
* Device Type List:	[0x01, 0x00, 0x00]	* Device Type List:	[0x02, 0x00, 0x00]		
* Received Frame Counter:	0x00000000 (0)	* Received Frame Counter:	0x0000000 (0)		
* Profiles Discovered:	0x0002	* Profiles Discovered:	0x0002		



Afterwards, target and controller are in the same RF4CE network and can send data packets to each other. Figure 6 shows the controller send data to the target. At the beginning, it sends data from channel 15. Because agility is enabled by default, it uses channel 20, 25. After that, it gets ACK on channel 25 and stays on that channel for the next packet.

File Tools D	Device	View	Window Help										
letwor 🖡 🗙	Traff	ic Vie	w Device Manag	ler									• >
•	i 🖉 📭		😐 🤪 🛷 🚳	, i 0   G		PAN	ID_c824	• ° 🖌 🔽 🖓 🖺					
-III >		Ln.	Timestamp	Time Delta	Ch.	Stack	Layer	Packet Information	PAN Dst.	MAC Src.	MAC Dst.	MAC Seq.	
	1	19	14:06:36.155745	209.68311;	15	RF4CE	RF4	Standard Data Frame: ZRC	0x1234	ØXAAAA	ØxBBBB	136	
Graphi 🗜 🗙	2	19	14:06:36.158081	0.002336	15	RF4CE	RF4	Standard Data Frame: ZRC	0x1234	ØXAAAA	ØxBBBB	136	
🛃 🖷 🖕	з	19	14:06:36.160417	0.002336	15	RF4CE	RF4	Standard Data Frame: ZRC	0x1234	ØXAAAA	ØxBBBB	136	
	4	19	14:06:36.162753	0.002336	15	RF4CE	RF4_	Standard Data Frame: ZRC	0x1234	ØXAAAA	ØxBBBB	136	
	5	19	14:06:36.154785	209.688264	20	RF4CE	RF4_	Standard Data Frame: ZRC	0x1234	ØXAAAA	ØXBBBB	137	
	6	19	14:06:36.182932	209.68841(	25	RF4CE	RF4_	Standard Data Frame: ZRC	0x1234	<b>ØXAAAA</b>	ØxBBBB	138	
	7	5	14:06:36.183924	0.000992	25	RF4CE	MAC	Acknowledgement				138	
$\bigcirc$	8	19	14:11:29.264668	293.080744	25	RF4CE	RF4	Standard Data Frame: ZRC	0x1234	ØXAAAA	ØxBBBB	139	
0xBBBB	9	5	14:11:29.265660	0.000992	25	RF4CE	MAC	Acknowledgement				139	

Figure 6. Controller Could Send Packet to Target After Doing Fake Pairing

## 2.2 Configured as Target or Controller

Customers might manufacture only target or controller devices. It would be helpful if their device could be configured to their peer types. For RemoTI source code, the structure of "appDevInfo\_t" stores the device own information. Table 4 provides an example to configure the device own information on target and controller.

Element of appDevInfo_t	on Target	on Controller
	[OWN_INFO]	[OWN_INFO]
	IEEE=00 12 4B 00 12 34 00 01	IEEE=00 12 4B 00 12 34 01 00
nodeCapabilities	NODECAP=0x0F	NODECAP=0x0C
tgtTypeList	TGTLIST=0x01 0x00 0x00 0x00 0x00 0x00	TGTLIST=0x02 0x00 0x00 0x00 0x00 0x00
appCapabilites	APPCAP=0x12	APPCAP=0x12
devTypeList	DEVTYPELIST=0x02 0x00 0x00	DEVTYPELIST=0x01 0x00 0x00
profileIdList	PROFILEIDLIST=0x01 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF	PROFILEIDLIST=0x01 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
vendorld	VENDORID=0x0007	VENDORID=0x0007
vendorString	VENDORSTRING="TI_LPRF"	VENDORSTRING="TI_LPRF"

#### Table 4. Configuration for Device Own Information on Target and Controller

"nodeCapabilities" is the most important element. Table 5 lists the bits definition [2]. For controller, bit0 and bit1 should be set as '0'.

### Table 5. Definition of nodeCapabilities

Bit	Description
0	0 – Controller type
	1 – Target type
1	0 – Battery powered
	1 - AC mains powered
2	0 – Security incapable
	1 – Security capable
3	0 – Channel normalization incapable
	1 – Channel normalization capable
4 - 7	Reserved

### 2.3 Enable RF Test Modes

To verify the RF performance on manufactured products, devices need to enter specific test mode. RemoTI provides one API (RTI\_TestModeReq) to place the radio in different test modes [2]. Table 6 lists the test modes supported by RTI\_TestModeReq.

#### Table 6. Test Modes Supported by RTI\_TestModeReq

Test Mode	Description
0x00	The device transmits unmodulated carrier at the specified frequency and with the specified transmit power.
0x01	The device will transmit pseudo-random data at the specified frequency and with the specified transmit power.
0x02	The device will have the radio placed in receive mode on the specified frequency.

Example code is shown in Example 4. The parameters of mode, txPower, and channel depend on test conditions. Disable agility when setting the channel on MainMenu to simplify the test.

#### Example 4. Example Code to Call RTI\_TestModeReq

```
uint8 mode = 0x00; //tx unmodulated carrier
int8 txPower = 0; //0 dBm
uint8 channel = 15; // channel 15
RTI_TestModeReq(mode, txPower, channel );
```

### 2.4 Change RF Features

}

To meet some RF test cases, customers might need to change some RF parameters, (for example, RF channel, agility, discovery threshold). The example code is shown in Example 5.

About setting discovery threshold, the threshold value is unsigned (from 0 to 255). It is compared to the incoming discovery request link quality indicator (LQI), which is also an unsigned value (0 to 255, 255 meaning that it is a very strong or good signal). If the discovery request LQI is lower than the threshold, the request is ignored. With TI RF4CE kit, ARC and a dongle, if the threshold value is set to 205, they cannot pair when the distance between ARC and dongle is farther than 50 cm.

### Example 5. Example Code to Change RF Parameters

```
// change device RF channel
{
   uint8 value=15; //depend on test case
   RTI_WriteItemEx(RTI_PROFILE_RTI, 0x61, 1, &value);
        //0x61: RTI_SA_ITEM_CURRENT_CHANNEL
}
// enable or disable agility
{
   uint8 value=1;
   RTI_WriteItemEx(RTI_PROFILE_RTI, 0x87, 1, &value);
       //0x87: RTI_SA_ITEM_AGILITY_ENABLE
}
// set discovery threshold
{
   uint8 value=205;
   RTI_WriteItemEx(RTI_PROFILE_RTI, 0x62, 1, &value);
       //0x62: RTI SA ITEM DISCOVERY LOI THRESHOLD
}
```

# 2.5 Implement PER Test

PER test verifies the packet error rate between controller and target. Example code is provided in Example 6. Controller call *RTI\_SendDataReq()* is required to send each packet with ACK. If the target receives a valid packet, it would reply ACK. Then, DUT checks the status on callback function *RTI\_SendDataCnf()*. After the number of successful transmission is received, the packet error rate could be calculated.

## Example 6. Example Code for PER Test

```
void appProcessPER()
{
    //send 1st pkt
   RTI_SendDataReq(0, RTI_PROFILE_ZRC, 0, /*RTI_TX_OPTION_ACKNOWLEDGED*/4, 2, data);
   app_RF_test_data_s.Sent_cntr++;
   appRfTestMode = APP_RF_TEST_STATE_PER_RECEIVING;//update on RTI_SendDataCnf
    //add timeout mechanism
                                   //clear timeout indication
   app_RF_test_data_s.TMOUT = 0;
   timeout = (app_RF_test_data_s.PER_CNTR / 10 ) * 5; //could be adjusted
    timer_start_timerEx(FPA_App_threadId, FAKEPAIRING_APP_EVT_RF_TMOUT, timeout);
   while((app_RF_test_data_s.Sent_cntr<app_RF_test_data_s.PER_CNTR) &&
    !(app_RF_test_data_s.TMOUT))
    //counter to be send
    {
        if(appRfTestMode == APP_RF_TEST_STATE_PER_SENDING)
        {
            RTI_SendDataReq(0, RTI_PROFILE_ZRC, 0, /*RTI_TX_OPTION_ACKNOWLEDGED*/4, 2, data)
                                          app_RF_test_data_s.Sent_cntr++;
                                          appRfTestMode = APP_RF_TEST_STATE_PER_RECEIVING;
                                         //update on RTI_SendDataCnf
        }
    }
   while((appRfTestMode == APP_RF_TEST_STATE_PER_RECEIVING) &&
    !(app_RF_test_data_s.TMOUT))
    {
        //wait the ack for the last packet
    }
    //calculate PER
   per =(1.0 - (app_RF_test_data_s.Rcv_cntr / app_RF_test_data_s.PER_CNTR))*100;
   printf("PER Test result = %2.1f %%\n", per);
}
void RTI_SendDataCnf(rStatus_t status)
{
   if (appState == AP_STATE_RF_TEST_MODE)
                                              //under RF test mode
    {
        if(appRfTestMode == APP_RF_TEST_STATE_PER_RECEIVING)
        ł
            if (status == RTI_SUCCESS)
                app_RF_test_data_s.Rcv_cntr++;
            printf("PER Test Rcv_cntr=%d\n",app_RF_test_data_s.Rcv_cntr);
            //appRfTestMode = APP_RF_TEST_STATE_PER_SENDING;
        }
        appRfTestMode = APP_RF_TEST_STATE_PER_SENDING; //toggle appRfTestMode
    }
```



#### 3 Setup for RF Test

Using Example 4 and Example 6, this section introduces how to setup the test environment and perform RF tests. Continuous Tx, Rx and PER test is also introduced. Except PER test, one DUT (RemoTI Target Board with CC2533EM) is used (see Figure 7). DUT connects to the spectrum analyzer with RF cable. In addition, it connects to one Linux laptop with a USB cable. The laptop executes the test program, then DUT emits an RF signal to the spectrum analyzer.

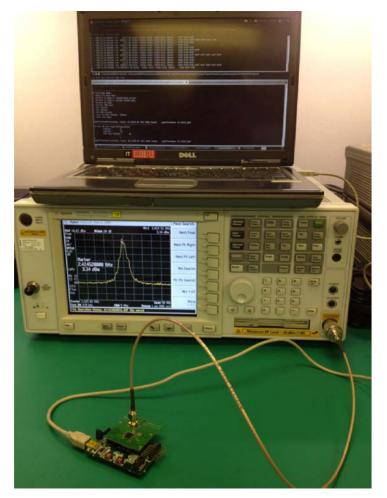


Figure 7. Setup for RF Test

### 3.1 Perform Tx Test Mode0

Tx test Mode0 verifies the RF quality while the DUT emits the unmodulated signal. To perform Tx test Mode0, open two terminals on a Linux laptop (see Figure 8) and run the following steps:

- 1. On one terminal (the upper one is shown in Figure 8), enter NPI server directory, "RemoTI-Linux/ Projects/tools/LinuxHost/out".
- 2. Launch NPI server by running "./NPI\_Inx\_x86\_server".
- 3. On another terminal, enter NPI client directory, "RemoTI-Linux/ Projects/tools/LinuxHost/application/fakePairing/out.
- 4. Launch NPI client by running "./FPA\_Inx\_x86\_client -c ../defCtl.cfg".
- 5. Type 'f' to select "Enter RF Test Mode".
- 6. Type '1' to select "Perform Tx Mode 0: unmodulated carrier".
- 7. Then, DUT emits an RF signal to the spectrum (see Figure 9).



🖸 Applications Places System 🕹 🖓 🔚			() (1) (1) (1) (1) (1) (1) (1) (1) (1) (
takePairing_app.c	workspace/RF4CE/Git/RF_test/FakeF		
1.010	vorkspace/RF4CE/Git/RF_test	/FakePairing/RemoTI-Linux/Projects/tools/Linu	
ile Edit View Terminal Help			
> 000:03:11.526683 (+0. 15492)] 07 bytes,	subsys 0x22, cmd10 0x21, DData: subsys 0x24, cmd10 0x21, DData:	8xF 6x2 8x86 6x86 8x2 8x86 8x88 8x88 8x88 8x88 8x86 8x2 8x81 6x86 8x2 9x7 6x4 8x33 8x86 8x12 9x86 8x13 8x88 8x86 8x84 8x86 8x86 8x84 8x87 8x86 8x84 8x86 8x86 8x86 8x86 8x86 8x86 8x86 8x86	
e Edit View Terminal Tabs Help	WF4CE/Git/RF_test/FakePairingRemoTi-Li	FakePairing/RemoTI-Linux/Projects/tools/Linux	Nost application/TakePairing/out
WITEX] Unlock appinit Mutex Test Mode MENN: Return to Main Menu Perform Tx Mode 8: unnodulated carrier Perform Tx Mode 9: unnodulated carrier Perform FXR Test Set Channel Set Channel Set TAR Counter Set PAR Counter Liss Corrent Channel, TxPower Show This Menu New Set		ж.	
pRfTestModeProcessKey, State: AP_STATE_RF_T		AP_STATE_INIT	
rrent RF test mode Configuration TAPower: 4 Channel: 15 PER Test Counter: 10		- AD CTATE TUT	
opRfTestModeProcessKey, State: AP_STATE_RF_T	est_Mode [exek] appkirestMode=	AP_STATE_INIT	

Figure 8. Tx Test Mode0 Launch on Linux Laptop

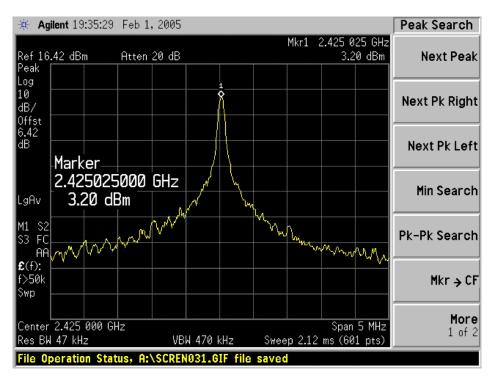


Figure 9. Tx Test Mode0 Perform on Spectrum Analyzer



### 3.2 Perform Tx Test Mode1

Tx test Mode1 verifies the RF quality while the DUT emits the modulated signal. To perform Tx test Mode1, open two terminals on a Linux laptop (see Figure 10) and run the following steps:

- 1. On one terminal, enter NPI server directory, "RemoTI-Linux/Projects/tools/LinuxHost/out".
- 2. Launch NPI server by running "./NPI\_Inx\_x86\_server".
- 3. On another terminal, enter NPI client directory, "RemoTI-Linux/ Projects/tools/LinuxHost/application/fakePairing/out".
- 4. Launch NPI client by running "./FPA\_Inx\_x86\_client -c ../defCtl.cfg".
- 5. Type 'f' to select "Enter RF Test Mode".
- 6. Type '2' to select "Perform Tx Mode 0: unmodulated carrier".
- 7. Then, DUT emits an RF signal (see Figure 11).

😋 Applications Places System 🌒 😨 🔚		🥡 💷 d 🚾 Tue Aug 6, 4:49 PM 🔞 root (
A A A A A A A A A A A A A A A A A A A	orkspace/RF4CE/Git/RF_test/Fakef	Pairing/RemoTI-Linux/Projects/tools/LinuxHost/application/fakePairing - Geany
0.000	workspace/RF4CE/Git/RF_test	t/FakePairing/RemoTi-Linux/Projects/tools/LinuxHost/out
File Edit View Terminal Help		
[> 000:07:51.996298 (+0. 15083)] 07 bytes,	subsy 6x2A, cmd10 8x21, pbata- subsy 6x2A, cmd10 4x21, pbata-	0xFF 9x22 0x06 6x00 0x20 0x00 0x00 0x00 0x00 0x00 0xFF 9x33 0x01 0xFF 0x43 0x0 0xFF 0x44 0x03 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x58 0x02 0x00 0x01 0x00 0xFF 0x58 0x02 0xFF 0x58 0x02 0xFF 0x50 0x00 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
File Edit View Terminal Tabs Help	vorkspace/RF4CE/Git/RF_test/	/FakePairing/RemoTi-Linux/Projects/tools/LinuxHost/application/fakePairing/out
workspace	/RF4CE/Git/RF_test/FakePairing/RemoTI-Li	inux/Pro ¥ workspace/RF4CE/Git/RF_test/FakePairing/RemoTI-Linux/Pro 1
[MUTEX] Unlock appInit Mutex f		
RF Test Mode MENN: r Return to Main Menu l- Perform Tx Mode 1: unmodulated carrier 2- Perform Rx Mode 1: pseudo-random data 3- Perform RX Mode 4- Perform PER Test c- Set Channel p- Set Tahower n- Set PER Counter l- List Current Channel, TxPower m- Show This Menu		
appRfTestModeProcessKey, State: AP_STATE_RF_	TEST_MODE [0x0A] appRfTestMode=	AP_STATE_INIT
Current RF test mode Configuration TXPower: 4 Channel: 15		
<ul> <li>PER Test Counter: 10</li> <li>2 appRfTestModeProcessKey, State: AP_STATE_RF_</li> </ul>	★ TEST_MODE (0x0A) appRfTestMode=	AP_STATE_INIT

Figure 10. Tx test Mode1 Launch on Linux Laptop

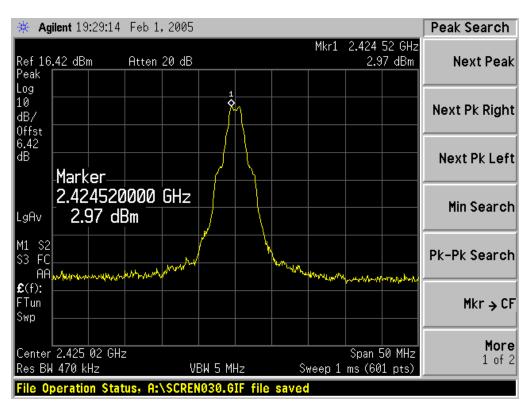


Figure 11. Tx Test Mode1 Perform on Spectrum Analyzer

## 3.3 Perform Rx Test

Rx test verifies whether DUT enters receiving mode and does not emit any RF signal on the specified frequency and its second harmonic. To perform Rx test, open two terminals on a Linux laptop (see Figure 12) and run the following steps:

- 1. On one terminal, enter NPI server directory, "RemoTI-Linux/ Projects/tools/LinuxHost/out.
- 2. Launch NPI server by running "./NPI\_Inx\_x86\_server".
- 3. On another terminal, enter NPI client directory, "RemoTI-Linux/ Projects/tools/LinuxHost/application/fakePairing/out".
- 4. Launch NPI client by running "./FPA\_Inx\_x86\_client -c ../defCtl.cfg".
- 5. Type 'f' to select "Enter RF Test Mode".
- 6. Type '3' to select "Perform Rx Mode".
- 7. Modify the spectrum with wider frequency range similar to Figure 13. Verify whether or not DUT emits an RF signal on the specified frequency and its second harmonic.



		Pairing /RemoTi-Linux/Projects/tools/LinuxHost); t/FakePairing/RemoTi-Linux/Projects/tools/Linux	
Edit View Terminal Help 000:02:57.644107 (+0. 364)] 03 bytes.	vorkspace/RF4CE/Git/RF_test	FakePairing/RemoTH inux/Projects/tools/Linux	
000:02:57.644107 (+0. 364)] 03 bytes.			Hostout
000:02:57.603865 (+0. 20019)         07 bytes.           000:02:57.673801 (+0. 10026)         08 bytes.           000:02:57.673801 (+0. 10026)         08 bytes.           000:02:57.673801 (+0. 20091)         04 bytes.           000:02:57.673801 (+0. 20091)         04 bytes.           000:02:57.703800 (+0. 9990)         08 bytes.           000:02:57.703800 (+0. 9990)         08 bytes.           000:02:57.703800 (+0. 9990)         08 bytes.           000:02:57.70380 (+0. 19940)         03 bytes.           000:02:57.70343 (+0. 19940)         05 bytes.           000:02:57.703404 (+0. 19940)         19 bytes.	subSys Bx2A, cmdId Bx21, pData: subSys Bx2A, cmdId Bx21, pData:	ውሎFF 0xA2 0x06 የአቶቂ 0xA2 0x06 0x00 0x00 0x00 የአቶቹ 0xA3 0x00 የአቶቂ 0x12 የአቶታ 0xA3 0x02 የአቶቂ 0x12 የአቶታ 0xA3 0x00 0x00 0x13 0x00 0x13 0x00 0x13 0x00 0x07 0x8F 0x8F 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x8F 0x6F 0x00 0x07 0x06 0x07 0x00 0x7F 0x00 0x07 0x06	
000:03:07.413880 (+9.682632)] 03 bytes.	, subSys 8x6A, cmdid 8x21, pData:	8x08 8x54 8x49 8x5F 8x4C 8x58 8x52 8x46 8x82 8x84 8x8F	
9.0 Edit View Terminal Tabs Help	/orkspace/RF4CE/Git/RF_test	/FakePairing/RemoTI-Linux/Projects/tools/Linux1	Host/application/fakePairing/out
a waxaa xaa waxaa waxaa waxaa ahaa ahaa	e/RF4CE/Git/RF_test/FakePairing/RemoTI-L		workspace/RF4CE/Git/RF_test/FakePairing/RemoTi-Linux/Pro.
est Made MENU: eturn to Main Menu erform Tx Mode 8: unnodulated carrier erform Tx Mode 1: pseudo-random data erform Rk Mode erform PER Test et Channel et TxAPower et PER Counter I PER Counter t PER Counter st Current Channel, TxPower how This Menu		×	
fTestModeProcessKey, State: AP_STATE_RF	_TEST_MODE [0x0A] appRfTestMode=	AP_STATE_INIT	
ent RF test mode Configuration TxPower: 4 Channel: 15 PER Test Counter: 10			
fTestModeProcessKey, State: AP_STATE_RF	_TEST_MODE [0x0A] appRfTestMode=	AP_STATE_INIT	

Figure 12. Rx Test Launch on Linux Laptop

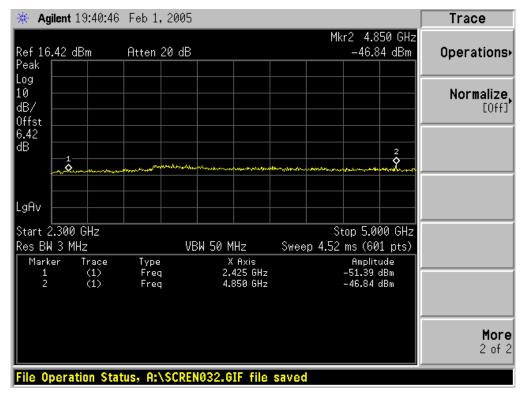


Figure 13. Rx Test Perform on Spectrum Analyzer



### 3.4 Perform PER Test

PER test verifies the packet error rate between the controller and target. The setup is similar to Figure 5. On a Linux laptop, two NPI server configuration files are needed for the controller and target; Table 7 lists the difference. Run two NPI server instances for controller and target. Open four terminals (see Figure 14) and run the following steps:

Setup for RF Test

- 1. On Terminal1 (the upper left one is on Figure 14), enter NPI server directory, "RemoTI-Linux/ Projects/tools/LinuxHost/out".
- 2. Launch NPI server for controller, "./NPI\_Inx\_x86\_server ./RemoTI\_RNP.cfg".
- On Terminal2 (lower left), enter NPI client directory, "RemoTI-Linux/ Projects/tools/LinuxHost/application/fakePairing/out".
- 4. Launch NPI client for controller, "./FPA\_Inx\_x86\_client -c ../defCtl.cfg".
- 5. On Terminal3 (upper right), enter NPI server directory, "RemoTI-Linux/ Projects/tools/LinuxHost/out".
- 6. Launch NPI server for target, "./NPI\_Inx\_x86\_server ./RemoTI\_RNP\_Tgt.cfg".
- 7. On Terminal4, enter NPI client directory, "RemoTI-Linux/ Projects/tools/LinuxHost/application/fakePairing/out".
- 8. Launch NPI client for target, "./FPA\_Inx\_x86\_client -c ../defTgt.cfg".
- 9. On Terminal2, type 'f' to select "Enter RF Test Mode".
- 10. Type '4' to select "Perform PER Test". After finishing test, it would print out the PER value. Figure 15 shows the sniff log. For good condition, after controller sends each packet, target would reply ACK.

#### Table 7. Server Configuration Difference on Target and Controller

on Target	on Controller
[PORT]	[PORT]
port=2533	port=2530
devPath="/dev/ttyUSB1";	devPath="/dev/ttyUSB0";
[LOG]	[LOG]
log="./NpiLnxLog_2.txt"	log="./NpiLnxLog.txt"

😧 Applications Places System 製 😰 📰					9 4-	🔜 Tue Aug	13, 10.40 A	M Street C
and a solution of the solution	kspace/RF4CE/Git - Geany		1 81828		worksp	ace/RF4CE/G	it/RF_test	FakéPairin
CERT C	workspace/RF4CE/G	it/FakePairing 2/Re	File Edit View Terminal Help					
File Edit View Terminal Help JevPath = '/dev/ttyUSB1'			> 000:00:34.784410 (+0. < 000:00:34.784473 (+0. 0x00 0x00 0x04 0x02 0x01	76)] 88 bytes,				8x88 8x88 0x81
logPath = './NpiLnxLog.txt' Port: 2533 Following IP addresses are available:			[> 000:00:34.804462 (+6. [<- 000:00:34.804516 (+6. 0x00 0x00 0x94 0x92 0x03	87)] 08 bytes.				0x00 0x00 0x01
IPv4: interface: lo IP Address 127.0.0.1 IPv6: interface: lo IP Address ::1 The socket will listen on the following IP add			[> 000:00:34.825742 (+0. [<- 000:00:34.825796 (+0. 0x00 0x00 0x04 0x02 0x01	70)] 08 bytes. 0x21	subSys 0x4A,	cndId 0x05,	pData:	0x00 0x00 0x01
1Pv4: 0.0.0.0 1Pv5: ::			[> 000:00:34.844430 (+0. < 000:00:34.844482 (+0. 0x00 0x00 0x04 0x02 0x03	66)] 08 bytes, 0x21	subSys 8x4A,	cmdId 0x05,	pData:	8×90 8×86 8×81
0.0.0.0 means it will listen to all available waiting for first connection on #3	IP address		[> 000:00:34.864424 (+0. [< 000:00:34.864485 (+0. 0x00 0x00 0x04 0x02 0x01	76)] 08 bytes, 0x21	subSys 0x4A,	cmdId 0x05,	pData:	8x88 8x88 8x81
Connected to #5.(127.0.0.1 / ::100:0:8a7f:0) BIG DEBUG ACTIVE set to FALSE DEBUG TIME ACTIVE set to FALSE			[> 000:00:34.885000 (+0. [< 000:00:34.885050 (+0. 0x00 0x00 0x04 0x02 0x03 [> 000:00:34.904419 (+0.	65)] 88 bytes, 0x21	subSys 8x4A,	cmdId 0x05,	pData:	0x00 0x00 0x0) 0x00
		NO	-					
	workspace/RF4CE/G			12	vorksp	ace/RF4CE/G	it/RF_test	/FakePairin
File Edit View Terminal Help t Toggle DEBUG TIME ACTIVE on Server y- ToggleBIG DEBUG on Server c- Toggle simple data display a - Check States g- Get MAC Channel r- Reset RMP w- Show This Menu j- Jerry test			File Edit View Terminal Heig PER Test Sent cntr = 2 PER Test Kev Cntr=2 PER Test Sent cntr = 3 PER Test Sent cntr = 3 PER Test Sent cntr = 4 PER Test Sent cntr=5 PER Test Sent cntr=5 PER Test Sent cntr=5 PER Test Sent cntr=6		vorksp	ace/RF4CE/G	it/RF_test	/FakePairin
File Edit View Terminal Help t Toggle DEBUG TIME ACTIVE on Server > Toggle Bispiel data display a check States 9 det MAC Channel - Reset RMP Bono This Menu 9 Jerry test MWTEX  Unlock applnit Mutex Display messages for simple testing	workspace/RF3CE/G	it/FakePairing_2/Ru	File Edit View Terminal Heip DER Test Sent cntr = 2 DER Test Sent cntr = 2 DER Test Sent cntr = 3 DER Test Sent cntr = 3 DER Test Sent cntr = 4 DER Test Sent cntr = 5 DER Test Sent cntr = 5 DER Test Sent cntr = 6 DER Test Sent cntr = 6 DER Test Sent cntr = 7 DER Test Sent cntr = 7		vorksp	ace/RF4CE/G	it/RF_test	/FakePairin
File Edit view Terminal Heip t- ToggleDEBUG TIME ACTIVE on Server ToggleBEG DEBUG on Server Toggle single data display a-check states Get MAC Channel Reset NNP Show This Menu Jerry Vest Show This Menu Jerry Vest Display messages for simple testing Source Ida: 0, profiled 1 (MTI PMOFILE ZRC), Source	workspace/RF4CE/G rxL01 220 (- 17 dBm) rxL01 228 (- 17 dBm) rxL01 228 (- 17 dBm) rxL01 228 (- 17 dBm)	Data: 1 33 Data: 1 33 Data: 3 33 Data: 1 33 Data: 3 33	File Edit View Terminal Heip DER Test Sent cntr = 2 DER Test Kevt cntr=2 DER Test Kevt cntr=3 DER Test Sent cntr = 3 DER Test Sent cntr = 4 DER Test Sent cntr = 5 DER Test Sent cntr = 5 DER Test Sent cntr = 6 DER Test Sent cntr = 7 DER Test Sent cntr = 8 DER Test Sent cntr = 8 DER Test Sent cntr = 8 DER Test Sent cntr = 9 DER Test Sent cntr = 9		vorksp	ace/RF4CE/G	it/RF_test	(FakePairin
File Edit View Terminal Help t- ToggleDEBUG TIME ACTIVE on Server y- ToggleBIG DEBUG on Server t- Toggle simple data display a check States y- Get MAC (chanel g- Get MAC (chanel J- Jerry test MUTER) Unlock applnit Mutex C Display mesages far simple testing Source 1dx: 0, profiled 1 (MTI PMOFILE ZRC), Source 1dx: 0, profiled 1 (MTI PMOFILE ZRC),	workspace/RF4CE/G rxL01 220 (- 17 dBs) rxL01 220 (- 17 dBs) rxL01 226 (- 17 dBs)	Duta: 1 33 Data: 1 33 Data: 1 33	File Edit View Terminal Heip PER Test Sent Cntr = 2 PER Test Kev Cntr=2 PER Test Kev Cntr=3 PER Test Kev Cntr=3 PER Test Sent Cntr=4 PER Test Sector Terms PER Test Sector Terms PER Test Sector Cntr=6 PER Test Sector Cntr=7 PER Test Sector Cntr=8 PER Test Sector Cntr=8	<b>k</b> DE (9x0A)	vorksp	ace/RF4CE/G	it/RF_test	(FakePairin

### Figure 14. PER Test Launch on Linux Laptop



ile Tools	Device		Analyzer Window Hel								- 0 ×
t 4 X		fic Viev		·					• X	Packet View	a i
ZigBee R						E DAN	ID_c824	• % % % %	• ^	RF4CE – Standard Data Frame: ZRC	
ap ¥ ×	1 2 3 4 5 6 7 7 8 9 9 7 7 8 9 9 10 11 12 13 14 15 16 16 16 16 17 18 20 20	Ln. 19 5 5 19 5 5 5 5 5 5 5 5 5 5 5 5 5	Treastamp 1713/158,23355 1713/58,234551 1713/58,245951 1713/58,269921 1713/58,269921 1713/58,269921 1713/58,269921 1713/58,269921 1713/58,269921 1713/58,269921 1713/58,289797 1713/58,289797 1713/58,289795 1713/58,299231 1713/58,299231 1713/58,299231 1713/58,292453 1713/58,29245 1713/58,29245 1713/58,29245 1713/58,29245 1713/58,2924 1713/58,2924 1715 1715 1715 1715 1715 1715 1715 171	Time Deta           42,703224           0.060992           0.015448           0.005784           0.00992           0.005784           0.00992           0.00952           0.00852           0.00992           0.00992           0.00992           0.00992           0.009584           0.009554           0.009554           0.009554           0.008304	Ch.	Stack Stack RF4CE RF4CE RF4CE RF4CE RF4CE RF4CE RF4CE RF4CE RF4CE RF4CE RF4CE RF4CE RF4CE RF4CE RF4CE RF4CE	Layer	Packet Information Standard bats Frame: ZBC Acknowledgement Standard bats Frame: ZBC Acknowledgement		<pre>▲ Frame Information: (19 bytes) Packet Number: 1 Protocol: Figher PACE Timestamp: 17:19:58.83559 Time UB: 42.709224 Channel: 5 Length: 492.70924 Channel: 5 Link Quality: -8 dem Source: UB2331 Layer: NF4CE Satura: Normal ▲ NGC Header: (9 bytes) ▷ Frame Control: ex861 Sequence Number: 174 Destination PAU [10: 60234 Destination PAU [10: 60234 Di New P</pre>	
										exceed 61 55 AE 34 12 56 56 AA AA 29 6x0000A 33 54 60 60 61 61 21 FF FF	84) 3T!
	٠								+ <del>*</del>	II 0/0 (	0/0 0/19

Figure 15. Sniff Log While Performing PER Test

# 3.5 Enable NPI Client on Network Test

As described in Section 1.4, the NPI server and client can run on different devices. An example is shown in Figure 16. Before performing the RF test, the NPI client needs to understand the physical IP address of the NPI server and fill to the IP\_ADDR of the NPI client's configuration file. If the NPI client runs as control role, defCtl.cfg is used. Otherwise, defTgt.cfg is used for target role.



Figure 16. NPI Server and Client Run on Different Devices



# Example 7. Modification on NPI Client

```
@defCtl.cfg
   [DEBUG_MODE]
   BIG=0 ; (1: ON, 0: OFF)
   TIMER=1 ; (1: ON, 0: OFF)
   APP=1 ; (1: ON, 0: OFF, 2: Verbose ON)
   [BASE_SETTINGS]
   IP_ADDR=192.168.0.66
@defTgt.cfg
   [DEBUG_MODE]
   BIG=0
   TIMER=0
   APP=1
   CLIENT=0
   [BASE_SETTINGS]
   IP_ADDR=192.168.0.66
```

# 3.6 Perform PER Test With SmartRF Studio

Besides Section 3.4, there is another option for PER test. The setup is shown in Figure 17. The SmartRF Studio runs PER Tx side, and the RNP runs PER Rx side.

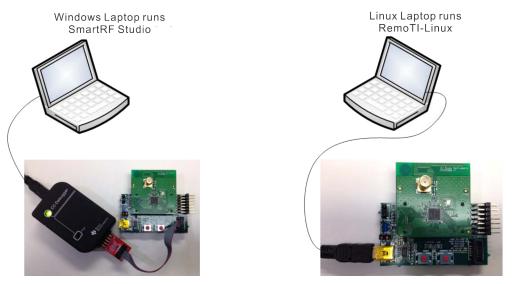


Figure 17. Setup for PER Test With SmartRF Studio

The RF4CE stack already embeds a test function,

RTI\_TestModeReq(RTI\_TEST\_MODE\_RX\_AT\_FREQ,..). After this API is called, the RF4CE stack starts to count the received MAC frames. If the sender finishes transmitting test packets, you can call RTI\_TestRxCounterGetReq(1) to get the counter of the received MAC frames. If you know in advance the number of packets sent and the time needed to send them, then you can have an estimation of the PER.

Use the SmartRF Studio to send test packets; it takes around 8s to send 100 packets.

Note that the network layer is still linked on the MAC/RF PHY, which means it can change RF parameters. To avoid this, disable the following:

- Standby mode : RTI\_StandbyReq(RTI\_STANDBY\_OFF);
- Frequency agility

Below are a list of the test steps:

- 1. On the RNP side, set test channel and launch test mode (see Example 8, call appProcessPER2Start ()).
- 2. On SmartRF Studio, configure the broadcast packet (see Figure 18). Then start to send packets.
- 3. After all packets are sent, on RNP side, stop the test mode and calculate the PER (see Example 8, call *appProcessPER2Stop()*). To simplify the test, disable agility when setting the channel on MainMenu.

Example 8. Example Code for PER Test With SmartRF Studio

```
//To reset the counter and start the MAC layer in reception (here one channel 15):
RTI_TestModeReq( RTI_TEST_MODE_RX_AT_FREQ, 0, 15);
void appProcessPER2Start()
{
     uint8 value=0;
    printf("Note, the number of expected packet is 100. To modify this, need to modify the
definitoin of PKT_exp inside the code\n");
     //the network layer is still linked to the MAC/PHY. Which means it can change RF parameters. To
avoid that disable the following:
     // - Standby mode
     // - Be sure the RF never turn off by application
     //RTI_StandbyReq(RTI_STANDBY_OFF); //disable standby mode
     //RTI_StandbyReq(RTI_STANDBY_ON);
     //To reset the counter (rtiRxConter) and start MAC Layer in reception
    RTI_TestModeReq(RTI_TEST_MODE_RX_AT_FREQ,0,app_RF_test_data_s.Channel);
}
#define PKT_exp 100 //set expected packet to 100
void appProcessPER2Stop()
{
     double per=0.0;
     uint8 res=0;
     double pkt_exp = 1.0 * PKT_exp;
     appRfTestMode = APP_RF_TEST_STATE_INIT; //set state
     res = RTI_TestRxCounterGetReq(1); //get counter, then reset it.
     printf("---- receive counter = %d ----\n", res);
     //calculate PER
    per = (1.0 - (res / pkt_exp))*100;
    printf("---- PER2 Test result = %10.1f %% ----\n", per);
}
```

e <u>S</u> ettings <u>V</u> iew <u>E</u> valuation Board <u>H</u>	elp		1		_
E Easy Mode 🔚 Expert Mode	💟 Register View	RF Parameters	9982 - 0	C2533 - Register V	/iew
Select configuration:				📳 Register e:	xport
IEEE 802.15.4 Beacon packet, chann IEEE 802.15.4 Command packet, chan			Registe		Value (Hex
IEEE 802.15.4 Data packet, channel	11			MFILTO	OD
SimpliciTI Ping packet, channel 15			P	MFILTO MFILT1	78
SimpliciTI Ping packet, channel 20				MFILI1 CMATCH	78 07
SimpliciTI Ping packet, channel 25				CMATCH CSHORTEN0	00
SimpliciTI Ping packet, channel 26				CSHORTEN1	00
				CSHORTEN2	00
				CEXTENO	00
				CEXTEN1	00
				CEXTEN2	00
cket TX Packet RX				MCTRLO	40
				MCTRL1	00
Data				ENABLE	00
				MASKSET	00
Preamble SFD Length Frame Control Seq. Number PAN	Dest. Addr Source Madress Payload			MASKCLR	00
Prear SFD Leng Seq. Numbest.	Dest. Addr Baddre FCS			IRQM0	00
4 1 1 2 1 2	2 2 2 2		► RF	IRQM1	00
A7 0xD 0x8801 0x00 0xFFFF	0xFFFF 0xBBBB FFFF		► RF	ERRM	00
			► FR	EQCTRL	0B
			► FR	EQTUNE	0F
Packet count: 100 Infinite			▶ тх	POWER	BC
			► TX	CTRL	69
Sent packets: 56		10	FS	MSTAT0	00
Channel: 0x0B		17	FS	MSTAT1	00
Output power: 0.3 dBm			► FIF	OPCTRL	40
	ТХ	RX	•		
Start Stop			Device r	eset Help	Refres

Figure 18. Configuration on SmartRF Studio

Since the process in Example 8 uses broadcast packets, check that another RF4CE device is not transmitting, otherwise, the PER will be erroneous. The other possibility is to use unicast packet. On SmartRF Studio, "Shource PAN" and "Source Address" need to match the values on the RNP target side (PANID and SHORT\_ADDRESS).

## 4 RemoTI Mass Production Test

To simplify the RF4CE RF test, the RemoTI Mass Production Test [8] [9] was provided, which utilizes the PC application to analyze the test result. The test flow is depicted in Figure 19. First, prepare the configuration on the PC side and launch the RemoTI-MassProduction-Test application. Then, execute the massProduction client on the STB side. The test packets automatically transmit. Finally, the PC application generates the test result (see Figure 20).

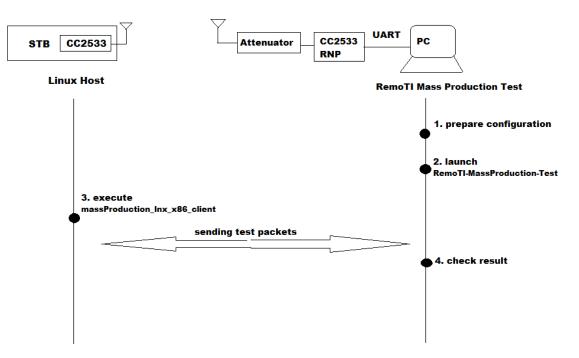


Figure 19. Execute massProduction\_Inx\_x86\_client

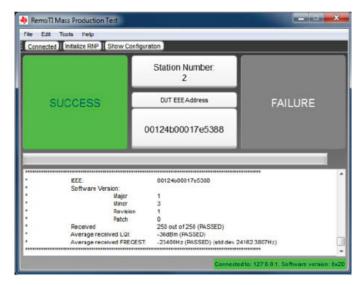


Figure 20. Result of RemoTI Mass Production Test

# 5 Summary

This application report illustrates the methods for "Fake Pairing" and provides some RF test cases, which can verify the manufactured devices. It also helps customers to manufacture RF4CE products easier.

### 6 References

- 1. RemoTI Host Processor Sample Application and Porting Guide (SWRA259)
- 2. RemoTI API (SWRA268)
- 3. RF4CE, simple linux target application wiki
- 4. TI-LPRF-Software/RemoTI-Linux
- 5. RemoTI Network Processor Developer's Guide (SWRU223)
- 6. RF4CE, simple linux target application wiki
- 7. RemoTI Network Processor Interface Specification API (SWRA271)
- 8. NPI Server (SWRA450)
- 9. RemoTI Mass Production Test (SWRA451)

#### **IMPORTANT NOTICE**

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information 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.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have *not* been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		Applications	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	TI E2E Community	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconne	ctivity	

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265 Copyright © 2015, Texas Instruments Incorporated