Application Report
SLOA210–November 2014

Build Instructions for neard (Linux NFC) for AM335x + TRF7970A

Josh Wyatt, Erick Macias, and Tim Simerly — *Texas Instruments*
Mark Greer — *Animal Creek Technologies*

ABSTRACT

The purpose of this document is to describe the steps / process required to correctly build and test Linux NFC (neard) on AM335x (Cortex-A8) based BeagleBone + BeagleBone RF Cape + TRF7970ATB connected platform, using the Texas Instruments Sitara Software Development Kit (SDK) v07.00 and associated TI SDK file system in conjunction with the neard NFC stack.

Contents

List of Figures

Sitara is a trademark of Texas Instruments.
ARM is a registered trademark of ARM Ltd.
All other trademarks are the property of their respective owners.

1 Scope

This document describes how to build the necessary components for NFC using the Sitara Software Development Kit (SDK) 07.00. It also covers the adding of a Linux-based NFC driver to TI's Software Development Kit (SDK) 07.00, the building of the neard stack, installing or placing onto a target file system, and installing the associated tools (Python) that are needed to test NFC functions on that target file system and associated hardware.

URL for accessing the AM335x collateral:

http://software-dl.ti.com/sitara_linux/esd/AM335xSDK/latest/index_FDS.html

SDK Product Download URL:

http://software-dl.ti.com/sitara_linux/esd/AM335xSDK/latest/exports/ti-sdk-am335x-evm-07.00.00.00-Linux-x86-Install.bin

Code Composer Studio for Sitara ARM (choose Linux):

http://processors.wiki.ti.com/index.php/Download_CCS#Code_Composer_Studio_Version_6_Downloads

The building of the kernel, the associated driver modules, neard, and Python tools is done on a host machine (Linux PC) and the resulting output is copied to the respective directories on the target file system (SD Card).

NOTE: This build was performed and tested on a Linux PC with Ubuntu 12.04. Problems were encountered when building the neard stack with Ubuntu 10.04, because certain v10.04 libraries are out of date. Future releases of AM335x SDKs from Texas Instruments will require Ubuntu 12.04.

2 Hardware

Figure 1 through Figure 3 show the hardware that is used with the provided software.

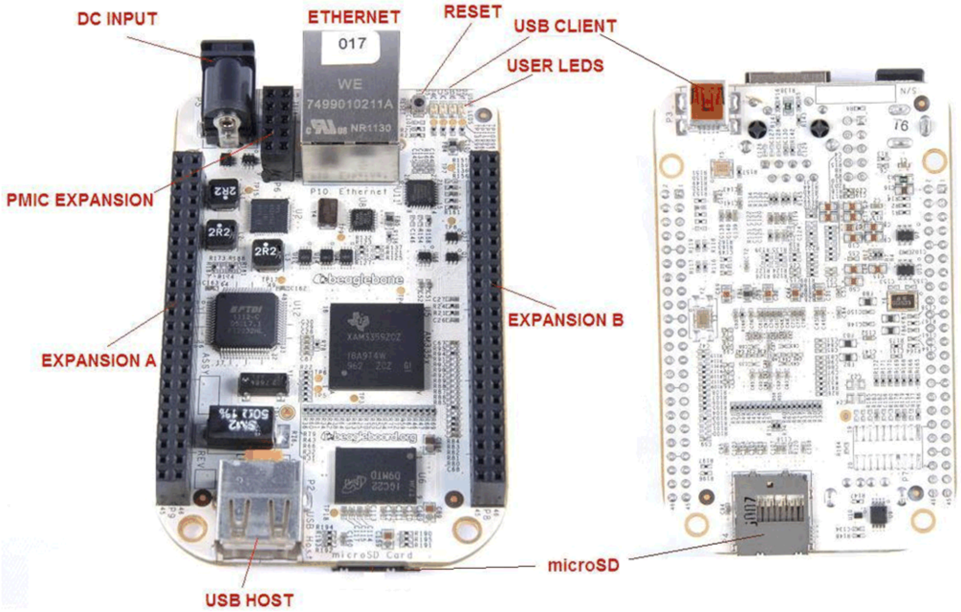


Figure 1. BeagleBone White Development Board

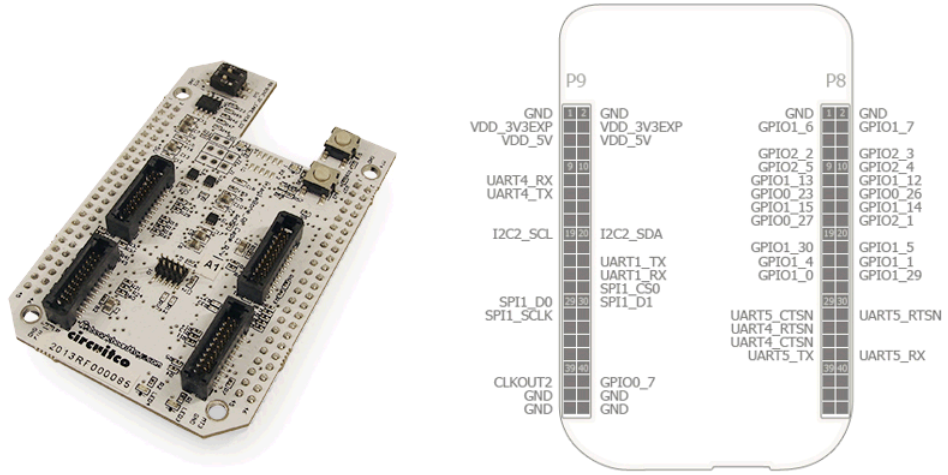


Figure 2. BeagleBone RF Cape Board and Pinout

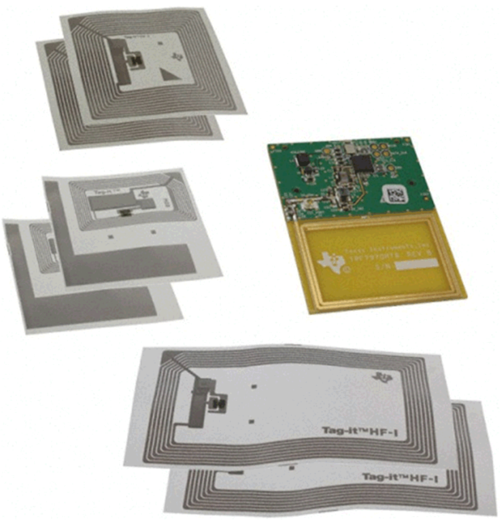


Figure 3. TRF7970ATB NFC/RFID Transceiver Module (With NFC/RFID Inlays)

Figure 4 shows the boards stacked together to create the complete hardware platform.

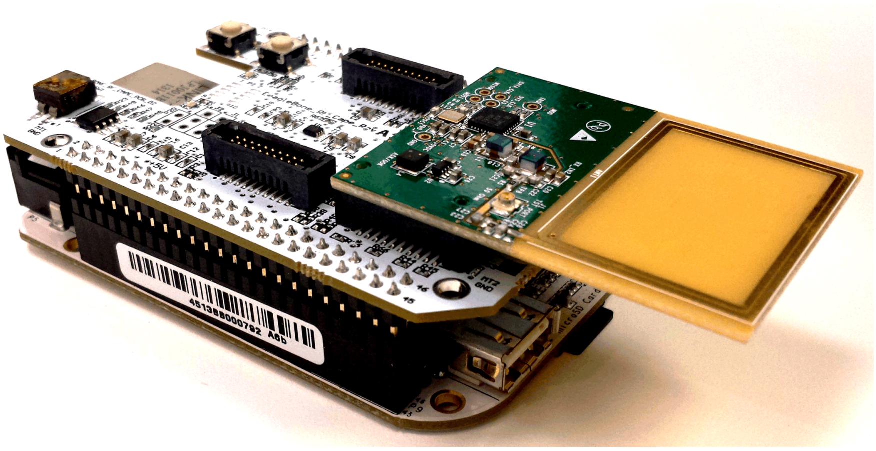


Figure 4. Connected Hardware Stack-Up

3 Adding NFC to the Linux Kernel

Create a working directory and download the latest NFC code base from this link:

http://software-dl.ti.com/dsps/forms/self_cert_export.html?prod_no=trf7970a_support.tar.bz2&ref_url=http://software-dl.ti.com/msp430/msp430_public_sw/mcu/msp430/Linux-NFC-TRF7970A

Perform the following steps (assuming you have git installed on your Ubuntu v12.04 PC):

1. $ cd **<SDK Root Directory>**/board-support
2. $ tar –xjf trf7970a_support.tar.bz2 (these are the kernel patches)
3. $ cd linux-3.12.10-ti2013.12.01
4. $ git checkout –b ti/master (makes new branch with name ti/master)
5. $ git add . (this stages any files that may have not been previously committed)
6. $ git commit (this commits the files that have been staged)
7. $ git checkout –b nfc-patches (makes new branch and checks it out, this is for adding the patches from the tarball)
8. $ git am .. /trf7970a_support/* (this applies the patches from the tarball)

4 Building the Modified Linux Kernel

Perform the following steps to create a new kernel image with supporting modules:

1. export PATH=**<SDK Root Directory>**/linux-devkit/sysroots/i686-arago-linux/usr/bin:$PATH
2. make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- distclean
3. make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- tisdk_am335x-evm_defconfig
4. make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig

   [*]Networking Support --->

   <*>NFC subsystem support --->

   <*> NFC Digital Protocol stack support

   Near Field Communication (NFC) devices --->

   <M> Texas Instruments TRF7970a NFC driver
5. make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage
6. make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- modules
7. make ARCH=arm INSTALL_MOD_PATH=<**Target File System**> modules_install (where **Target File System** equals the SD card path)
8. make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- am335x-bone.dtb
9. cp **<SDK Root Directory>**/arch/arm/boot/zImage <**Target File System**>/boot
10. cp **<SDK Root Directory>**/arch/arm/boot/dts/am335x-bone.dtb <**Target File System**>/boot

NOTE: Step 7, 9, and 10 may require root access to complete.

5 Building the neard Stack

To build the neard stack, follow the steps below:

1. Clone the latest neard code base with the following command:

   git clone locations:

   git://git.kernel.org/pub/scm/network/nfc/neard.git

   https://git.kernel.org/pub/scm/network/nfc/neard.git

   https://kernel.googlesource.com/pub/scm/network/nfc/neard.git
2. $ cd neard

The latest version of neard has known issues; therefore a known working version should be used at this time. The command below checks out this known working version.

3. $ git checkout -b working 647F2705
4. Edit the file "bootstrap-configure" and replace:

   --prefix=/usr \

   --sysconfdir=/etc \

   with the following:

   $CONFIGURE_FLAGS \
5. Run the environment setup script:

   $ source <**SDK Root Directory**>/linux-devkit/environment-setup

NOTE: Make sure that libtool is installed (that is, sudo apt-get install libtool).

6. $ ./bootstrap-configure (this sets up the make files)
7. $ make CFLAGS+=-Wno-cast-align (this builds neard)
8. Copy the following files to the target file system:

   (a) "src/org.neard.conf" to the target file system "/etc/dbus-1/system.d/"

   (b) "src/neard" to the target file system "<**create your own NFC directory**>/neard"

   (c) "tools/nfctool/nfctool" to the target file system "<**create your own NFC directory**>/nfctool"

   (d) All the files in the test directory "test/*" to the target file system "<**create your own NFC directory**>/test/*"

6 Building Python

From the host PC, ensure you have not performed source <**SDK Root Directory**>/linux-devkit/environment-setup in the shell window to be used for the instructions below:

1. Download "Python-2.7.3.tgz" tarball from the following location:

   http://www.python.org/download/releases/2.7.3/
2. Download the cross compile patch for 2.7.3 "Python-2.7.3-xcompile.patch" from this location:

   http://randomsplat.com/id5-cross-compiling-python-for-embedded-linux.html
3. Untar the tarball image with the following command:

   $ tar –zxvf Python-2.7.3.tgz
4. Then go to the new created directory with the following command:

   $ cd Python-2.7.3
5. Build with the following set of commands:

   (a) $ ./configure

   (b) $ make python Parser/pgen

   (c) $ mv python hostpython

   (d) $ mv Parser/pgen Parser/hostpgen

   (e) $ make distclean
6. Now apply the patch with the following command:

   $ patch -p1 < ../Python-2.7.3-xcompile.patch
7. Now change the environment to the target build:

   (a) $ source < **SDK Root Directory** >/linux-devkit/environment-setup

   (b) $ ./configure --host=arm-linux --build=i686-linux-gnu --prefix=`pwd`

   (c) $ make HOSTPYTHON=./hostpython HOSTPGEN=./Parser/hostpgen BLDSHARED="arm-linux-gnueabihf-gcc -shared" CROSS_COMPILE=arm-linux-gnueabihf- CROSS_COMPILE_TARGET=yes HOSTARCH=arm-linux BUILDARCH=i686-linux-gnu

NOTE: You may encounter a message about modules not found at the end of the build. In this case, these can be ignored as they are not needed.

(d) $ make install HOSTPYTHON=./hostpython BLDSHARED="arm-linux-gnueabihf-gcc -shared" CROSS_COMPILE=arm-linux-gnueabihf CROSS_COMPILE_TARGET=yes prefix=<**full path of temp staging directory**>

8. Copy the files that are stored in <**temp staging directory>** to the target file system.

   For example:

   $ sudo cp –R <**temp staging directory>** /* <**target file system**>/usr

NOTE: Make sure that the files just copied to the target file system are owned by root.

7 Building Python Dbus Libraries

From the host PC, ensure you have not performed source <**SDK Root Directory**>/linux-devkit/environment-setup in the shell window to be used for the instructions below:

1. Download the Dbus tarball from the following location: http://dbus.freedesktop.org/releases/dbus-python/dbus-python-1.1.1.tar.gz
2. Untar the Dbus image with:

   $ tar –zxvf dbus-python-1.1.1.tar.gz
3. Export the Python executable:

   $ export PYTHON=<**directory where Python was built**>/hostpython
4. Copy three files: "**python-config**, **python2-config**, and **python2.7-config**" from <**temp staging directory>** to the directory where Python-2.7.3 was built (make sure that you are the owner of these files).
5. While in Python-2.7.3, copy python-config to hostpython-config
6. Change the first line of hostpython-config to match the full path of the Python-2.7.3 directory

   For example:

   #!/home/user/ti-sdk-am335x-evm_7_00/example-applications/Python-2.7.3/hostpython
7. From the dbus-python-1.1.1 directory, perform the following steps:

   (a) $ source < **SDK Root Directory** >/linux-devkit/environment-setup

   (b) $ ./configure $CONFIGURE_FLAGS

NOTE: You can ignore any warning here about unrecognized options

(c) $ make

(d) $ make install prefix=<**full path of temp staging directory>**

(e) Copy the files that are in <**temp staging directory>** to the **<target file system>**/usr

For example: sudo cp –R <python directory>/dbus_target/* **<target file system>**/usr

NOTE: Make sure the files just copied to the target file system are owned by root.

8 Testing NFC/RFID Tag Functionality Using neard

To interact (read or write) with NFC tag platforms from the target file system, place or present an NFC tag platform to the antenna on the TRF7970ATB board.

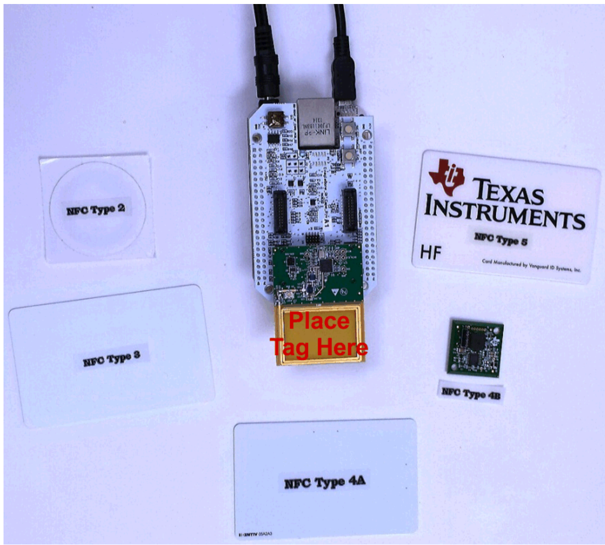
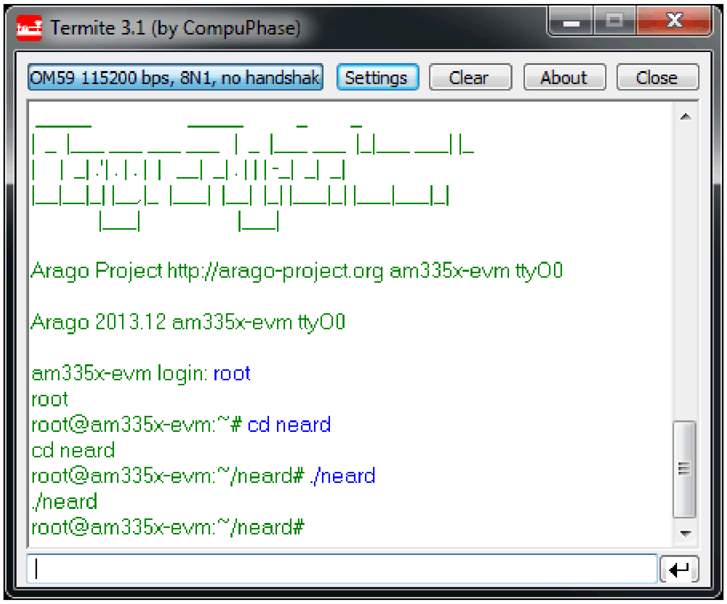


Figure 5. Supported NFC Forum Tag Platforms with Target Hardware

Go to the location on the target file system where the neard files reside and start neard.


Termite 3.1 (by CompuPhase)
OM59 115200 bps, 8N1, no handshak
Settings
Clear
About
Close

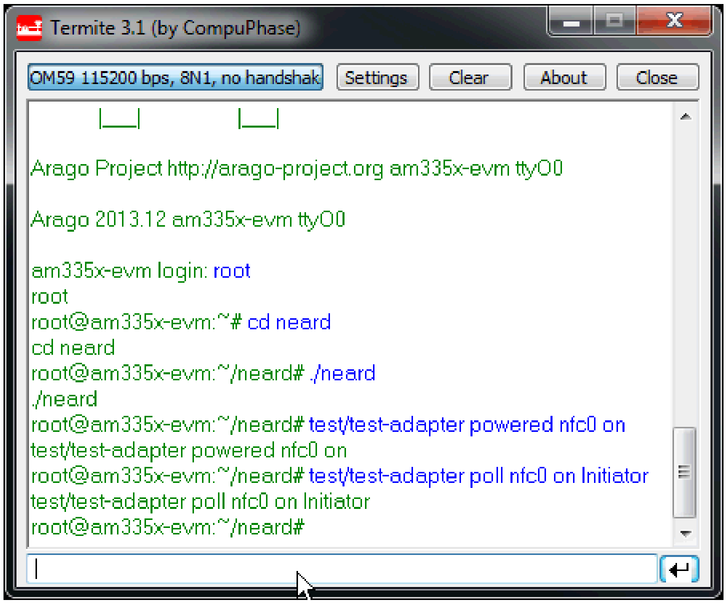
```
 _____                    _____           _         _
|  _  |___ ___ ___ ___   |  _  |___ ___  |_|___ ___| |_
|     |  _| .'| . | . |  |   __|  _| . | | | -_|  _|  _|
|__|__|_| |__,|_  |___|  |__|  |_| |___|_| |___|___|_|
              |___|                    |___|

Arago Project http://arago-project.org am335x-evm ttyO0

Arago 2013.12 am335x-evm ttyO0

am335x-evm login: root
root
root@am335x-evm:~# cd neard
cd neard
root@am335x-evm:~/neard# ./neard
./neard
root@am335x-evm:~/neard#
```

Figure 6. Logging in, Navigating to, and Starting neard

In the neard directory on the target file system, the following commands are to be done in the order listed. Each time a tag is presented, step 2 must precede the list or dump commands. Otherwise, the transmitter will not be on to power the passive tag.

1. test/test-adapter powered nfc0 on
2. test/test-adapter poll nfc0 on Initiator
3. test/test-tag list
4. test/test-tag dump

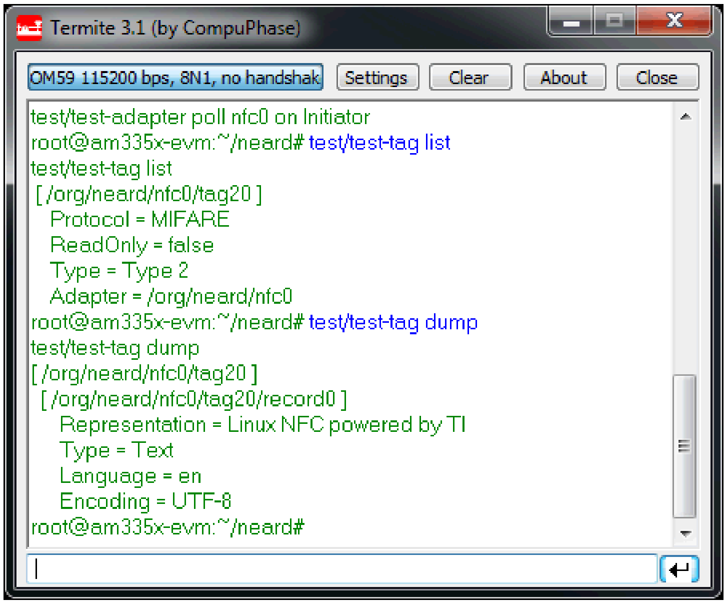
NOTE: Make sure that you leave the NFC card over the NFC antenna on the TRF7970ATB during these steps. Otherwise you will only get a command prompt back after typing each of the "test/test-tag" attempts.


Termite 3.1 (by CompuPhase)
OM59 115200 bps, 8N1, no handshak
Settings
Clear
About
Close

```
        |___|             |___|

Arago Project http://arago-project.org am335x-evm ttyO0

Arago 2013.12 am335x-evm ttyO0

am335x-evm login: root
root
root@am335x-evm:~# cd neard
cd neard
root@am335x-evm:~/neard# ./neard
./neard
root@am335x-evm:~/neard# test/test-adapter powered nfc0 on
test/test-adapter powered nfc0 on
root@am335x-evm:~/neard# test/test-adapter poll nfc0 on Initiator
test/test-adapter poll nfc0 on Initiator
root@am335x-evm:~/neard#
```

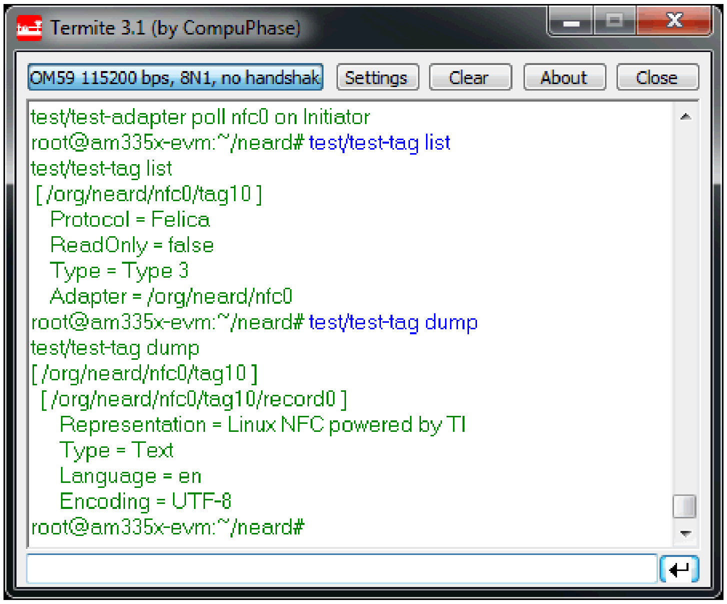
Figure 7. Configuring TRF7970A and Entering Polling Loop for NFC Tags

8.1 Testing NFC/RFID Tag Reading Using neard


Termite 3.1 (by CompuPhase)
OM59 115200 bps, 8N1, no handshak
Settings
Clear
About
Close

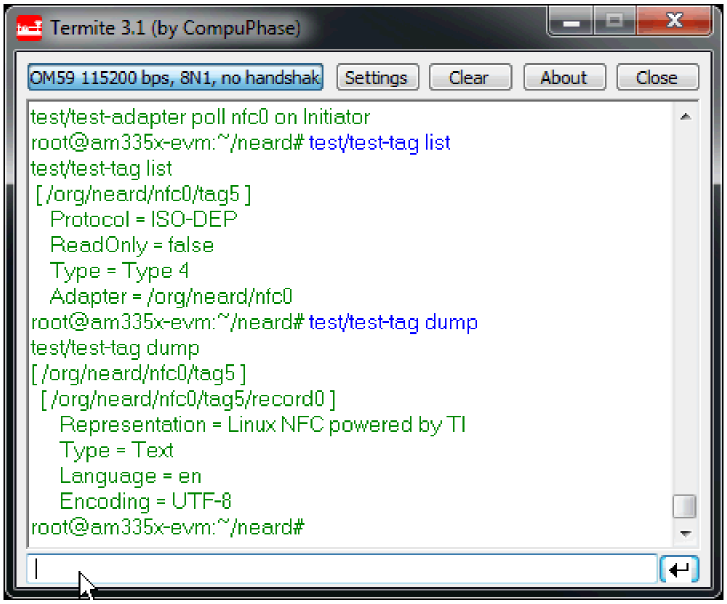
```
test/test-adapter poll nfc0 on Initiator
root@am335x-evm:~/neard# test/test-tag list
test/test-tag list
 [ /org/neard/nfc0/tag20 ]
    Protocol = MIFARE
    ReadOnly = false
    Type = Type 2
    Adapter = /org/neard/nfc0
root@am335x-evm:~/neard# test/test-tag dump
test/test-tag dump
[ /org/neard/nfc0/tag20 ]
  [ /org/neard/nfc0/tag20/record0 ]
     Representation = Linux NFC powered by TI
     Type = Text
     Language = en
     Encoding = UTF-8
root@am335x-evm:~/neard#
```

Figure 8. Reading NFC Type 2 Tag Platform (List and Dump)


Termite 3.1 (by CompuPhase)
OM59 115200 bps, 8N1, no handshak
Settings
Clear
About
Close

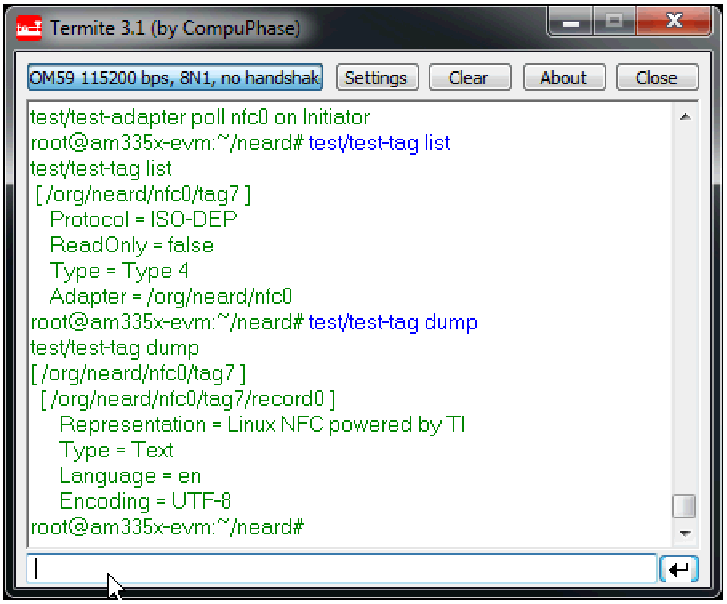
```
test/test-adapter poll nfc0 on Initiator
root@am335x-evm:~/neard# test/test-tag list
test/test-tag list
 [ /org/neard/nfc0/tag10 ]
   Protocol = Felica
   ReadOnly = false
   Type = Type 3
   Adapter = /org/neard/nfc0
root@am335x-evm:~/neard# test/test-tag dump
test/test-tag dump
[ /org/neard/nfc0/tag10 ]
  [ /org/neard/nfc0/tag10/record0 ]
     Representation = Linux NFC powered by TI
     Type = Text
     Language = en
     Encoding = UTF-8
root@am335x-evm:~/neard#
```

Figure 9. Reading NFC Type 3 Tag Platform (List and Dump)


Termite 3.1 (by CompuPhase)
OM59 115200 bps, 8N1, no handshak
Settings
Clear
About
Close

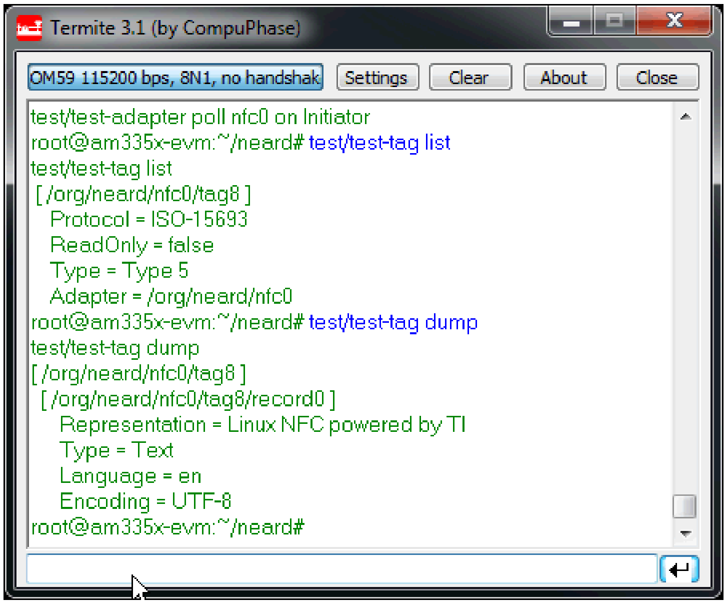
```
test/test-adapter poll nfc0 on Initiator
root@am335x-evm:~/neard# test/test-tag list
test/test-tag list
 [ /org/neard/nfc0/tag5 ]
   Protocol = ISO-DEP
   ReadOnly = false
   Type = Type 4
   Adapter = /org/neard/nfc0
root@am335x-evm:~/neard# test/test-tag dump
test/test-tag dump
[ /org/neard/nfc0/tag5 ]
  [ /org/neard/nfc0/tag5/record0 ]
     Representation = Linux NFC powered by TI
     Type = Text
     Language = en
     Encoding = UTF-8
root@am335x-evm:~/neard#
```

Figure 10. Reading NFC Type 4A Tag Platform (List and Dump)


Termite 3.1 (by CompuPhase)
OM59 115200 bps, 8N1, no handshak
Settings
Clear
About
Close

```
test/test-adapter poll nfc0 on Initiator
root@am335x-evm:~/neard# test/test-tag list
test/test-tag list
 [ /org/neard/nfc0/tag7 ]
    Protocol = ISO-DEP
    ReadOnly = false
    Type = Type 4
    Adapter = /org/neard/nfc0
root@am335x-evm:~/neard# test/test-tag dump
test/test-tag dump
[ /org/neard/nfc0/tag7 ]
  [ /org/neard/nfc0/tag7/record0 ]
      Representation = Linux NFC powered by TI
      Type = Text
      Language = en
      Encoding = UTF-8
root@am335x-evm:~/neard#
```

Figure 11. Reading NFC Type 4B Tag Platform (List and Dump)

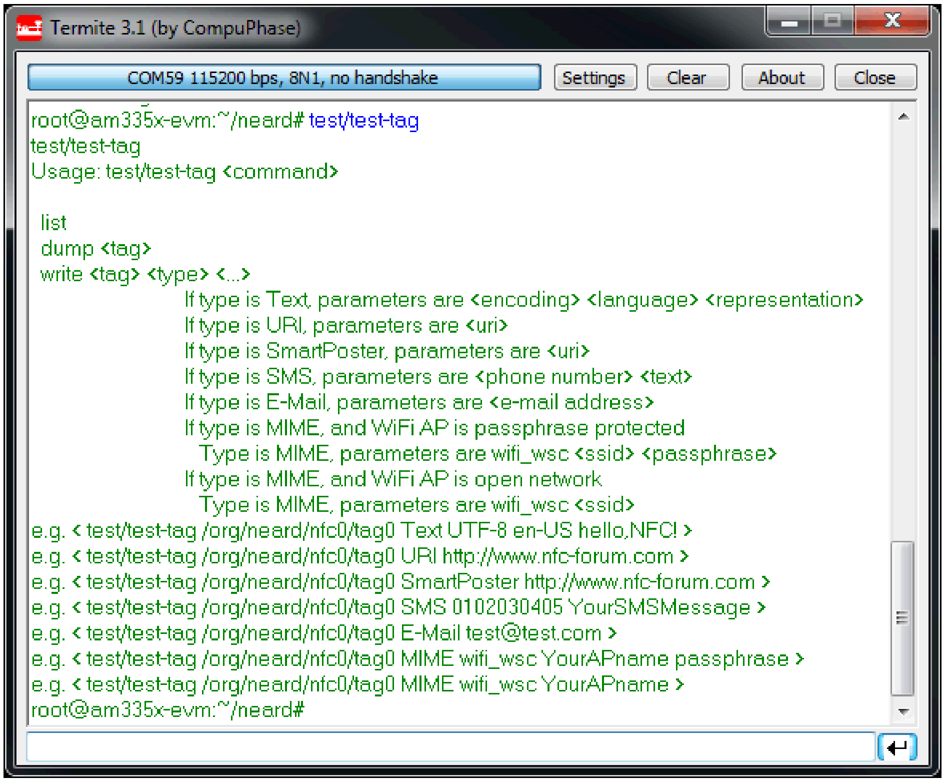

Termite 3.1 (by CompuPhase)
OM59 115200 bps, 8N1, no handshak
Settings
Clear
About
Close

```
test/test-adapter poll nfc0 on Initiator
root@am335x-evm:~/neard# test/test-tag list
test/test-tag list
 [ /org/neard/nfc0/tag8 ]
    Protocol = ISO-15693
    ReadOnly = false
    Type = Type 5
    Adapter = /org/neard/nfc0
root@am335x-evm:~/neard# test/test-tag dump
test/test-tag dump
[ /org/neard/nfc0/tag8 ]
  [ /org/neard/nfc0/tag8/record0 ]
      Representation = Linux NFC powered by TI
      Type = Text
      Language = en
      Encoding = UTF-8
root@am335x-evm:~/neard#
```

Figure 12. Reading NFC Type 5 Tag Platform (List and Dump)

8.2 Testing NFC/RFID Tag Writing Using neard

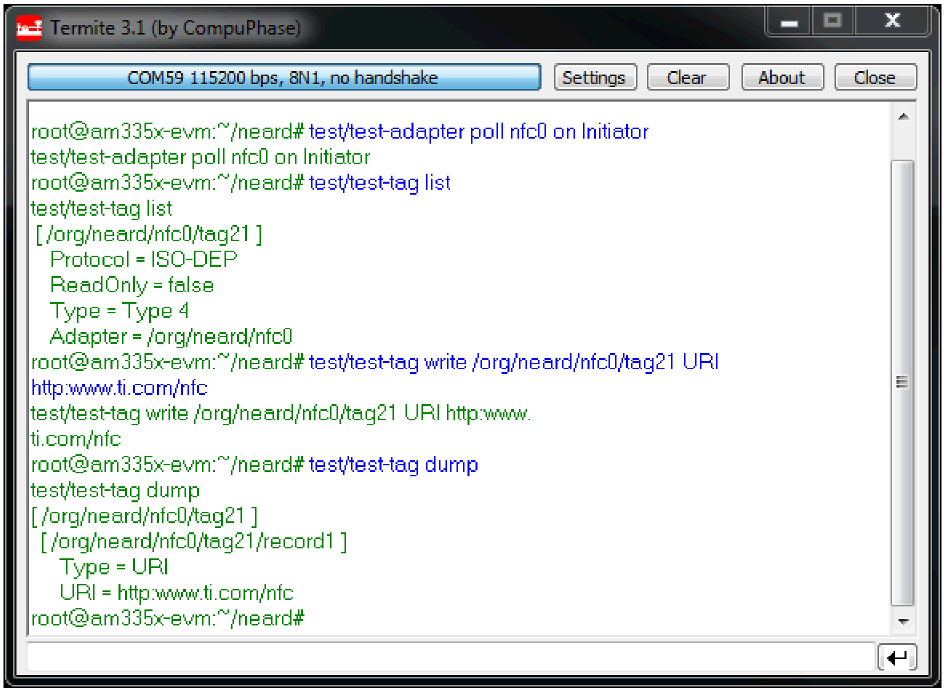
When writing tags, the format shown in Figure 13 must be followed. This menu is always available for reference by typing "test/test-tag"


Termite 3.1 (by CompuPhase)
COM59 115200 bps, 8N1, no handshake
Settings
Clear
About
Close

```
root@am335x-evm:~/neard# test/test-tag
test/test-tag
Usage: test/test-tag <command>

 list
 dump <tag>
 write <tag> <type> <...>
                    If type is Text, parameters are <encoding> <language> <representation>
                    If type is URI, parameters are <uri>
                    If type is SmartPoster, parameters are <uri>
                    If type is SMS, parameters are <phone number> <text>
                    If type is E-Mail, parameters are <e-mail address>
                    If type is MIME, and WiFi AP is passphrase protected
                      Type is MIME, parameters are wifi_wsc <ssid> <passphrase>
                    If type is MIME, and WiFi AP is open network
                      Type is MIME, parameters are wifi_wsc <ssid>
e.g. < test/test-tag /org/neard/nfc0/tag0 Text UTF-8 en-US hello,NFC! >
e.g. < test/test-tag /org/neard/nfc0/tag0 URI http://www.nfc-forum.com >
e.g. < test/test-tag /org/neard/nfc0/tag0 SmartPoster http://www.nfc-forum.com >
e.g. < test/test-tag /org/neard/nfc0/tag0 SMS 0102030405 YourSMSMessage >
e.g. < test/test-tag /org/neard/nfc0/tag0 E-Mail test@test.com >
e.g. < test/test-tag /org/neard/nfc0/tag0 MIME wifi_wsc YourAPname passphrase >
e.g. < test/test-tag /org/neard/nfc0/tag0 MIME wifi_wsc YourAPname >
root@am335x-evm:~/neard#
```

Figure 13. test/test-tag Help Screen

Figure 14 shows process of writing a URI RTD. Note that polling loop was started and a tag dump was done to get the tag number (in this case tag21) which is needed by the tag write command.


Termite 3.1 (by CompuPhase)
COM59 115200 bps, 8N1, no handshake
Settings
Clear
About
Close

```
root@am335x-evm:~/neard# test/test-adapter poll nfc0 on Initiator
test/test-adapter poll nfc0 on Initiator
root@am335x-evm:~/neard# test/test-tag list
test/test-tag list
 [ /org/neard/nfc0/tag21 ]
    Protocol = ISO-DEP
    ReadOnly = false
    Type = Type 4
    Adapter = /org/neard/nfc0
root@am335x-evm:~/neard# test/test-tag write /org/neard/nfc0/tag21 URI
http:www.ti.com/nfc
test/test-tag write /org/neard/nfc0/tag21 URI http:www.
ti.com/nfc
root@am335x-evm:~/neard# test/test-tag dump
test/test-tag dump
[ /org/neard/nfc0/tag21 ]
  [ /org/neard/nfc0/tag21/record1 ]
     Type = URI
     URI = http:www.ti.com/nfc
root@am335x-evm:~/neard#
```

Figure 14. Writing a URI RTD to Tag and Reading Back

9 Peer-to-Peer Modes

In addition to reading and writing tags, NFC Peer-to-Peer Initiator and Target modes are supported. To put the system into target mode and wait for an initiator, pass "Target" instead of "Initiator" to the test/test-adapter script.

When operating as a peer, the format below must be followed. This menu is always available for reference by typing "test/test-device"

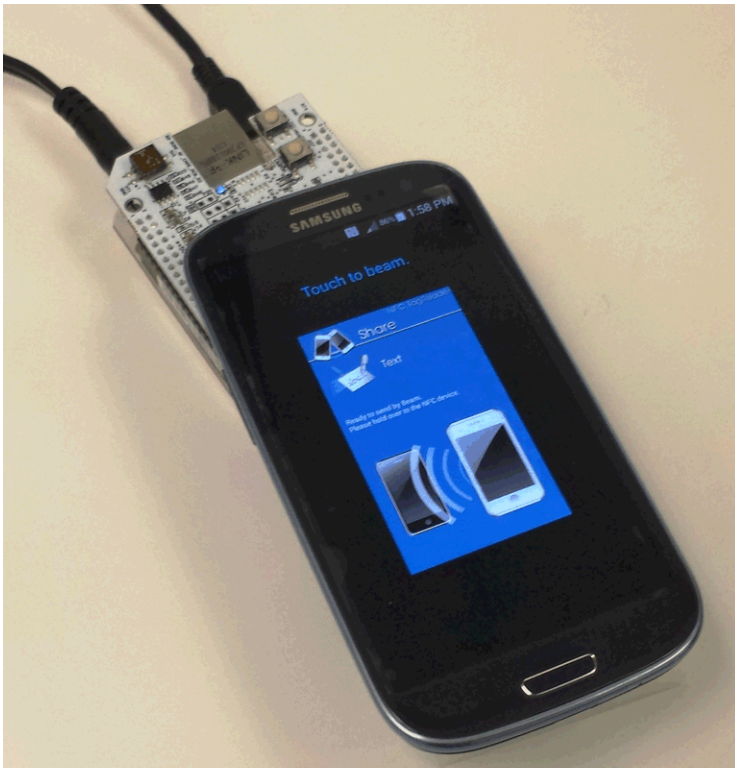
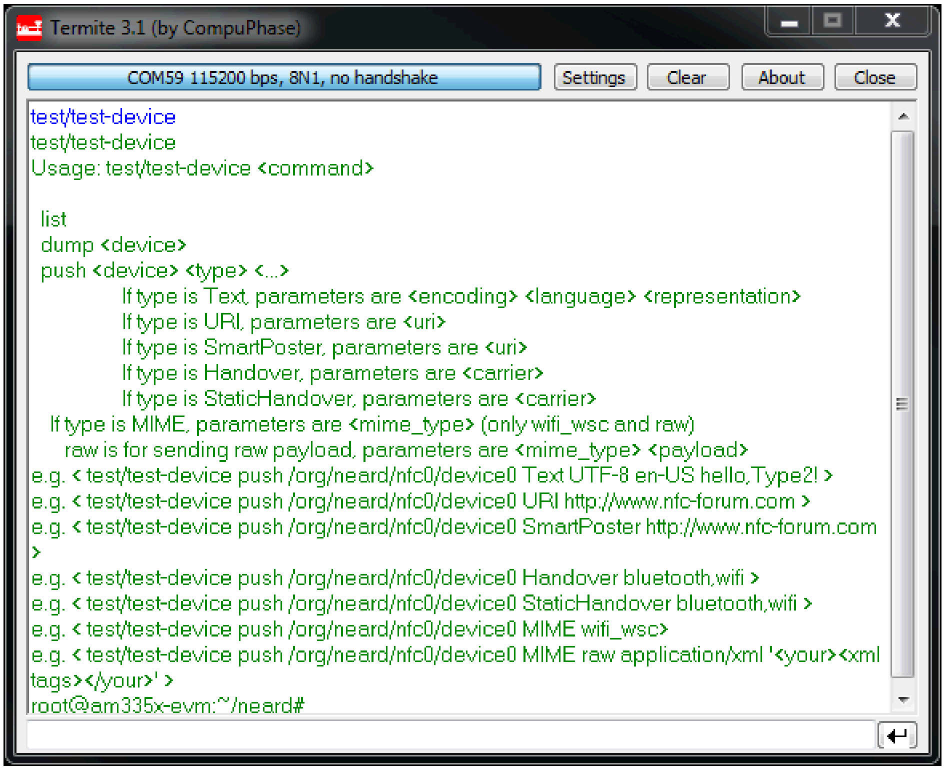


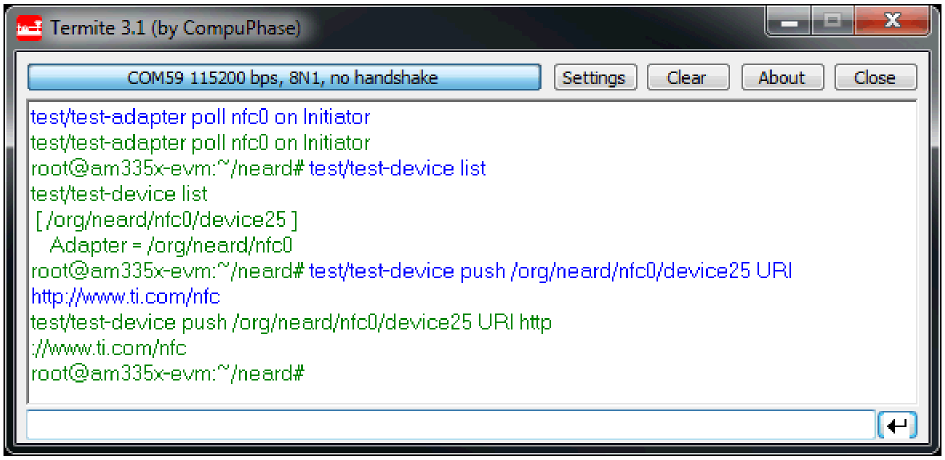
Figure 15. Presenting a Handset to Hardware for P2P Operations


Termite 3.1 (by CompuPhase)
COM59 115200 bps, 8N1, no handshake
Settings
Clear
About
Close

```
test/test-device
test/test-device
Usage: test/test-device <command>

 list
 dump <device>
 push <device> <type> <...>
            If type is Text, parameters are <encoding> <language> <representation>
            If type is URI, parameters are <uri>
            If type is SmartPoster, parameters are <uri>
            If type is Handover, parameters are <carrier>
            If type is StaticHandover, parameters are <carrier>
  If type is MIME, parameters are <mime_type> (only wifi_wsc and raw)
    raw is for sending raw payload, parameters are <mime_type> <payload>
e.g. < test/test-device push /org/neard/nfc0/device0 Text UTF-8 en-US hello,Type2! >
e.g. < test/test-device push /org/neard/nfc0/device0 URI http://www.nfc-forum.com >
e.g. < test/test-device push /org/neard/nfc0/device0 SmartPoster http://www.nfc-forum.com
>
e.g. < test/test-device push /org/neard/nfc0/device0 Handover bluetooth,wifi >
e.g. < test/test-device push /org/neard/nfc0/device0 StaticHandover bluetooth,wifi >
e.g. < test/test-device push /org/neard/nfc0/device0 MIME wifi_wsc>
e.g. < test/test-device push /org/neard/nfc0/device0 MIME raw application/xml '<your><xml
tags></your>' >
root@am335x-evm:~/neard#
```

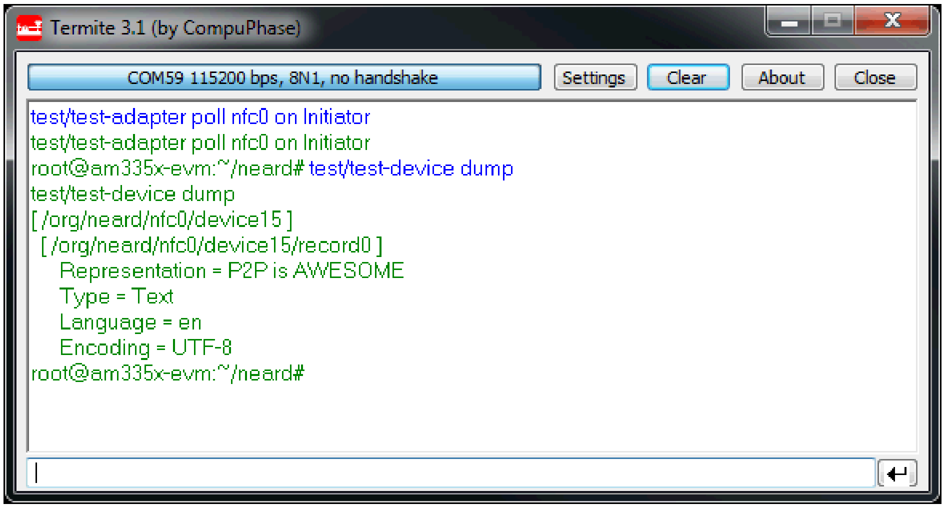
Figure 16. P2P Help Screen (Using test/test-device Command)

NOTE: Due to Initial RF collision handling, the user may need to issue the poll command multiple times when in Initiator mode.


Termite 3.1 (by CompuPhase)
COM59 115200 bps, 8N1, no handshake
Settings
Clear
About
Close

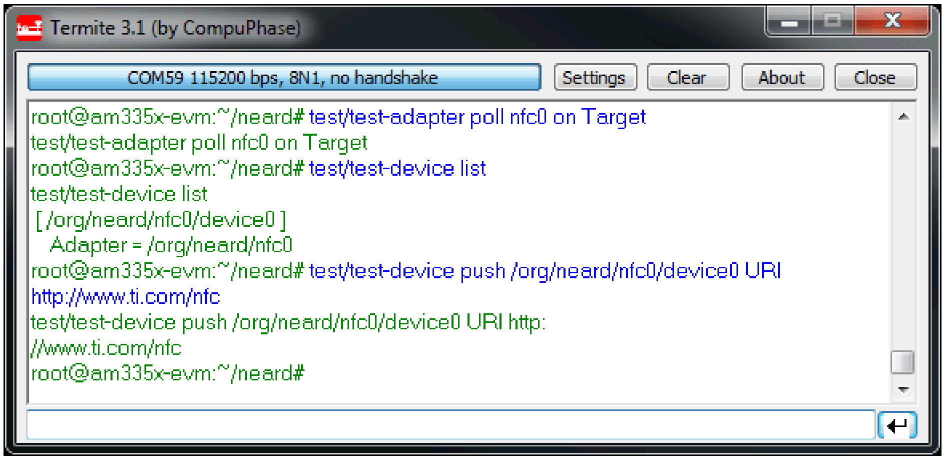
```
test/test-adapter poll nfc0 on Initiator
test/test-adapter poll nfc0 on Initiator
root@am335x-evm:~/neard# test/test-device list
test/test-device list
 [ /org/neard/nfc0/device25 ]
   Adapter = /org/neard/nfc0
root@am335x-evm:~/neard# test/test-device push /org/neard/nfc0/device25 URI
http://www.ti.com/nfc
test/test-device push /org/neard/nfc0/device25 URI http
://www.ti.com/nfc
root@am335x-evm:~/neard#
```

Figure 17. P2P Initiator Mode Example (Sending)


Termite 3.1 (by CompuPhase)
COM59 115200 bps, 8N1, no handshake
Settings
Clear
About
Close

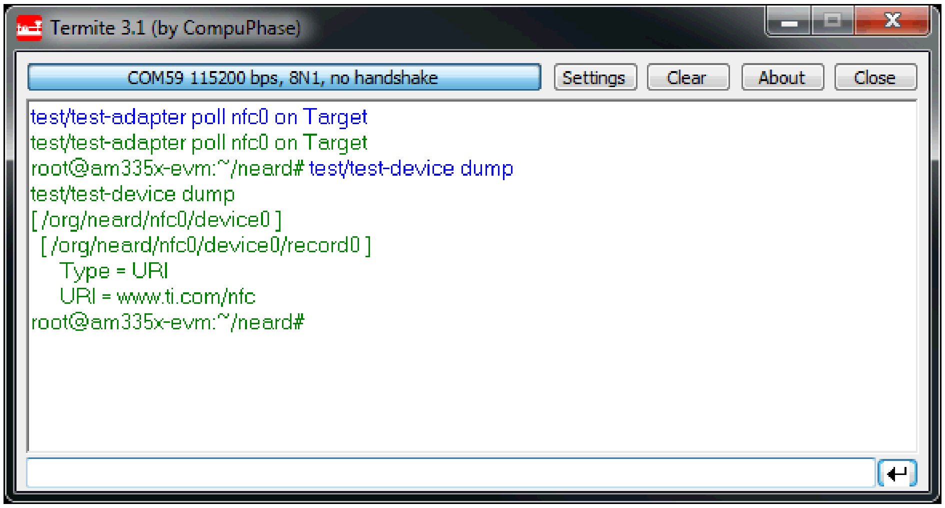
```
test/test-adapter poll nfc0 on Initiator
test/test-adapter poll nfc0 on Initiator
root@am335x-evm:~/neard# test/test-device dump
test/test-device dump
[ /org/neard/nfc0/device15 ]
  [ /org/neard/nfc0/device15/record0 ]
     Representation = P2P is AWESOME
     Type = Text
     Language = en
     Encoding = UTF-8
root@am335x-evm:~/neard#
```

Figure 18. P2P Initiator Mode Example (Receiving)


Termite 3.1 (by CompuPhase)
COM59 115200 bps, 8N1, no handshake
Settings
Clear
About
Close

```
root@am335x-evm:~/neard# test/test-adapter poll nfc0 on Target
test/test-adapter poll nfc0 on Target
root@am335x-evm:~/neard# test/test-device list
test/test-device list
 [ /org/neard/nfc0/device0 ]
   Adapter = /org/neard/nfc0
root@am335x-evm:~/neard# test/test-device push /org/neard/nfc0/device0 URI
http://www.ti.com/nfc
test/test-device push /org/neard/nfc0/device0 URI http:
//www.ti.com/nfc
root@am335x-evm:~/neard#
```

Figure 19. P2P Target Mode Example (Sending)


Termite 3.1 (by CompuPhase)
COM59 115200 bps, 8N1, no handshake
Settings
Clear
About
Close

```
test/test-adapter poll nfc0 on Target
test/test-adapter poll nfc0 on Target
root@am335x-evm:~/neard# test/test-device dump
test/test-device dump
[ /org/neard/nfc0/device0 ]
  [ /org/neard/nfc0/device0/record0 ]
     Type = URI
     URI = www.ti.com/nfc
root@am335x-evm:~/neard#
```

Figure 20. P2P Target Mode Example (Receiving)

10 Porting Notes

The TRF7970A driver has been tested on an AM335x based BeagleBone White board + an RF Cape + a TRF7970ATB. Here are some notes to help you get the TRF7970A working on a different set of hardware.

The processor side communicates with the TRF7970A using SPI with Slave Select, so to be able to communicate with the device; your hardware must have an SPI interface and a working SPI driver. Be sure that the SPI and TRF7970A CONFIG_ options are enabled before building the kernel.

The TRF7970A driver uses Device Tree (DT) properties to get the information it needs to function. There is some documentation in Documentation/devicetree/bindings/net/nfc/trf7970a.txt (in the kernel source) to help you set it up.

This snippet example is used for the BeagleBone platform:

```
&spi1 {
        status = "okay";
        trf7970a@0 {
              compatible = "ti,trf7970a";
              reg = <0>;
              pinctrl-names = "default";
              pinctrl-0 = <&trf7970a_default>;
              spi-max-frequency = <2000000>;
              interrupt-parent = <&gpio0>;
              interrupts = <31 0>;
              ti,enable-gpios = <&gpio2 3 GPIO_ACTIVE_LOW>,
              <&gpio2 4 GPIO_ACTIVE_LOW>;
              vin-supply = <&ldo3_reg>;
              vin-voltage-override = <5000000>;
              irq-status-read-quirk;
              en2-rf-quirk;
              status = "okay";
      };
};
```

The vin-voltage-override property allows you to override the voltage determined by the regulator subsystem. This could be used when the voltage to the TRF7970A has been stepped up (for example, on the TRF7970ATB board). Until there is a new revision of the TRF7970A, you should specify both irq-status-read-quirk and en2-rf-quirk. The TRF7970A's driver expects your platform's interrupt and regulator information to be set up correctly. If your platform has a pin multiplexer (pinmux), then it must be set up correctly, too.

11 References

1. *AM3358 Sitara™ ARM® Cortex-A8 Microprocessor* (AM3358)
2. *Linux EZ Software Development (EZSDK) for Sitara™ ARM® Processors* (AM335x SDK)
3. *BeagleBone Evaluation Module* (BeagleBone EVM)
4. *BeagleBone RF Cape Module* (BeagleBone RF Cape)
5. *TRF7970A NFC/RFID Transceiver Data Sheet* (SLOS743)
6. *TRF7970ATB Evaluation Module* (TRF7970ATB)
7. *Linux NFC Project* (neard)
8. *NFC Forum* (http://nfc-forum.org/)

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/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265