MSR Software Radio Academic Kit Readme File

Copyright (C) 2009-2010 by Microsoft Corporation

This file contains some information about the MSR software Radio Academic Kit.

For more information, check our website <http://research.microsoft.com/en-us/projects/sora/>**.**

Start from latest V2.0, SORA platform supports MIMO. See [MIMO support in SORA](#_MIMO_support_in) for details.

# System requirements

## Target Operation System

* Windows XP Professional Edition (**32 bit, service pack 3**)
* Windows 7/8 (64 bit)
  + SORA supports Windows 7/8 64 bit edition. 64 bit driver for Sora RCB is pre-built and included in the package.  
    There’s also a 64bit HWTest driver. These drivers are signed with a test certificate.
  + 32 bit user mode application development is still fully supported. 32 bit user mode application can seamlessly work with 64bit sora driver.

## Target Hardware

|  |  |
| --- | --- |
| CPU/Freq | quad-core/2.66GHz (or above) |
| Memory | 1GB or above |
| PCIe-x8/x16 slot | 1 |
| Hard Disk | 100M of free space |
| Radio hardware | Microsoft Research Software Radio Control Board (RCB) |
|  | Compatible RF front-end boards  (currently, WARP RF daughter board  or USRP XCVR2450 board with respective RF Adaptor Board) |

# Post-setup actions

## Install Sora Drivers

The setup package copies all necessary files to your local disk.

It doesn't install the Windows driver for radio control board (RCB) and the hardware test driver.

The x86 driver binaries are built for Windows XP and Windows 2003.

The amd64 driver binaries are built for Windows 7/8 64bit edition.

If you want to install the drivers, please make sure the RCB is correctly plugged into your mainboard, and then follow the guidance in "**Sora Device Drivers Installation.pdf**".

## Build Sora sample code

The setup package creates two shortcuts in [Start\Programs\Microsoft Research Asia\ Software Radio Academic Kit 2.0]:

**x86 Checked Build**: points to the command line window for winxp\_x86 checked build.

**x86 Free Build**: points to the command line window for winxp\_x86 free build.

**x64 Checked Build**: points to the command line window for win7\_amd64 checked build.

**x64 Free Build**: points to the command line window for win7\_amd64 free build.

Please note that the amd64 build environment is only for the sample Miniport driver.

The sample Miniport driver can be built as either 64 bit NDIS6 driver or 32 bit NDIS 5 driver.

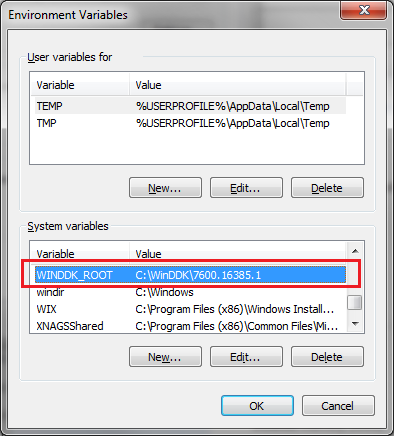
NDIS5 is for Windows XP 32 bit edition, while NDIS6 is for Windows 7/8 64 bit edition.

Before you start building the driver, run **%sora\_root%\src\set\_dirs\_x86.cmd** or **%sora\_root%\src\set\_dirs\_x64.cmd** to ensure the dirs files point to the correct folder.

To build the SORA sample code, Windows DDK is required.

Please specify a windows environment variable WINDDK\_ROOT which points to your Windows DDK installation path.

Here’s the screenshot as an example:



When the environment variable WINDDK\_ROOT is specified, open the command line window by clicking the shortcut to sora build window.

In the command line window, type “**bcz**” to build the whole source tree.

All target files (exe, dll, lib, sys, etc) are copied to **%SORA\_ROOT%\target**

To be noted, the 802.11a sample code is designed to support both 44MHz and 40MHz sample rate radio front end.

But you need to specify which sample rate hardware are supported at build time.

In **%SORA\_ROOT%\src\bb\dot11a\dot11\kernel\sources**, and **%SORA\_ROOT%\src\bb\dot11a\dot11\user\sources**, **\_\_HW\_40MHZ\_SAMPLE\_RATE** is for 40MHz sample rate front end, **\_\_HW\_44MHZ\_SAMPLE\_RATE** for 44MHz sample rate.

If the base band and radio front end are mismatched, the software PHY would not work.

Additionally, SDRMiniport device supports both 802.11a and 802.11b. You can choose to use ‘a’ or ‘b’ base band by modifying driver installation file sdr.inf, ModMode registry section, from “802.11a” to “802.11b”.

## Use Sora tools

Refer to “**Chapter 10. Tools and Utilities”** in “**The Sora Manual-v1\_8.pdf”** for detail.

# Files in package

The directory structure shown here assumes the Sora SDK is installed at d:\SORASDK2.0

|  |  |
| --- | --- |
| **D:\SORASDK2.0** | |
| │  AcademicKit-LA.pdf | Agreement to purchase the academic kit |
| │  MSR-LA.pdf | MSR License agreement |
| │  Sample Code-LA.pdf | MSR License agreement for the sample source code |
| │  Readme.htm | The ReadMe file |
| ├─bin |  |
| │  │  dut.exe | Hardware diagnosis tool. Run dut without any command line parameter for help. |
| │  │  HwVeri.exe | A helpful tool to test and configure Sora hardware components. Refer to Chapter 10.4 for detail. |
| │  │  dot11config.exe | SDR miniport driver configuration tool. See Chapter 9.1 for command line reference. Source code provided in %SORA\_ROOT%\src\driver\SDRMiniport\exe |
| │  │  demod11.exe | Command line tool to demodulates 802.11a(b) dump files and displays statistics about data frames. Source code provided in %SORA\_ROOT%\src\bb\exe |
| │  │  UMXDot11.exe | User mode 802.11 decoder based on UMX. It has a full featured 802.11a/b/g decoder. It is also able to modulate a frame and send it through UMX. Refer to Chapter 7.4 for detail. Source code provided in %SORA\_ROOT%\src\bb\UMXDot11 |
| │  │  sdscope-11a.exe | User mode utility which demodulates 802.11a frames from dump file and displays intermediate results in GUI. Refer to Chapter 10.2 for detail. |
| │  │  sdscope-11b.exe | User mode utility which demodulates 802.11b framesfrom dump file and displays intermediate results in GUI. Refer to Chapter 10.2 for detail. |
| │  │  SrView.exe | A simple Sora dump file viewer. See Chapter 10.3 for detail. |
| │  │  IntFiltr.reg | Interrupt-Affinity Filter registry setting |
| │  │  IntFiltr.sys | Interrupt-Affinity Filter driver |
| │  │  IntFiltrCmd.exe | Interrupt-Affinity Filter utility |
| │  ├Config | Configuration file used by sdscope-11b |
| │  ├─ProtocolRunInfo | Configuration file used by sdscope-11b |
| │  ├─HWTest | Test driver used by the diagnosis tool |
| │  └─PCIE | Radio Control Board driver |
| ├─build |  |
| ├─doc | Sora manual and hardware/driver installation guide |
| ├─inc | Software radio framework header files |
| ├─lib | Software radio framework library files |
| └─src | Sora sample code |
| ├─bb | Baseband library sample |
| │  ├─dot11a | 802.11a source code |
| │  ├─dot11b | 802.11b source code |
| │  ├─brick11 | 802.11a/b implementation based on BRICK model |
| │  ├─UMXDot11 | UMX extension, a full featured user mode 802.11 a/b/g decoder. |
| │  └─demod11 | Sample tools to modulate/demodulate 802.11a/b frames |
| ├─umxsdrbrick | User mode sdr modem which supports 802.11 a/b/n, it’s implemented with BRICK model. |
| ├─umxsdra | Old implementation of user mode sdr modem which supports 802.11a |
| ├─kmsdr | Miniport driver sample |
| │  ├─ll | Link layer |
| │  ├─mac | Mac layer |
| │  ├─phy | Physical layer |
| │  └─SDRMiniport |  |
| │      ├─dot11config | Miniport driver configuration tool |
| │     ├─NDIS6 | NDIS6 miniport driver |
| │      └─NDIS5 | NDIS5 miniport driver |
| ├─dbgplot-sample | Sample app which demonstrates the usage of DebugPlot library |
| ├─Samples | Sample app which demonstrates the usage of user mode extension and Sora DSP library (SDL) |
| ├─inc | Header files used by the 802.11 a/b sample driver |
| └─util | Common utilities used by the 802.11 a/b sample driver |

# MIMO support in SORA

In previous versions, SORA platform supports only 1 radio with radio index 0. **HWTest** and **SDRMiniport** driver bind this radio during the load process. Once the driver is successfully loaded, the radio 0 is also designated. This scenario leads **dut** to ignore the parameter of radio index. User is able to just specify **dut start** for starting the radio 0 because it is implicit in this command.

In MIMO version, SORA supports up to 8 radios. **dut** command must specify the radio index explicitly. For instance,

**dut start --radio 0**

*start the radio 0*

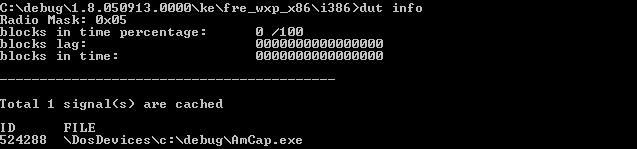
**dut rxpa --radio 0 --value 0x1000**

*set the rxpa value of radio 0 to 0x1000*

**dut tx --radio 0 --sid 524288**

*transmit the signal with id 524288 via radio 0*

The multiple radios support somehow affects the design of **HWTest** and **SDRMiniport** driver. For **HWTest** driver, no radio is bound at first without user’s specification. And **dut start** with a radio index binds the designated radio to **HWTest** driver dynamically. Multiple radios are allowed to bind to **HWTest** driver. **dut info** tells the current binding status. The API ***SoraURadioStart*** does the same as **dut start** while user is programming with **HWTest** driver.



**Figure 1 dut info, the Radio Mask tells radio 0 and radio 2 are bound to HWTest driver**

A new command **dut mimotx** is included. For example,

**dut mimotx --radio 0 --radio 1 --sid 524288 --sid 1048576**

*radio 0 transmits signal with id 524288 and radio 1 transmits signal with id 1048576 concurrently*

A new API ***SoraURadioMimoTx*** is added for MIMO TX.

***HRESULT SoraURadioMimoTx(IN ULONG\* RadioIndex, IN ULONG\* TxID, ULONG Count);***

***ULONG RadioIndex[] = { 0, 1 };***

***ULONG TxID[] = { 524288, 1048576 };***

***SoraURadioMimoTx(RadioIndex, TxID, 2);***

*radio 0 transmits signal with id 524288 and radio 1 transmits signal with id 1048576 concurrently*

Another difference which is worthy to mention is that transferring signal to RCB memory is now independent of radio. The user mode extension API ***SoraURadioTransfer*** and ***SoraURadioTransferEx*** are not changed because of backward compatibilities. User just neglects the useless first parameter. Based on this, several kernel mode transfer related APIs are modified by using a transfer object instead of radio object. New APIs ***SoraKAllocateTransferObj*** and ***SoraKFreeTranferObj*** are added to obtain and free a transfer object.

# Contact us

Project main site: <http://research.microsoft.com/en-us/projects/sora/>

Discussion forum: <http://social.microsoft.com/Forums/en-US/sora>