GoDroid

Developer Guide

v1.0

Copyright Statement

Copyright in this document is owned by GoWarrior Community. Any person is hereby authorized to view, copy, print and distribute this document subject to the following conditions:

- The document may be used for informational purposes only
- The document may be used for non-commercial purposes only
- Any copy of this document or portion thereof must include this copyright notice

This document is provided "as is" without any warranty of any kind, either express or implied, statutory or otherwise; without limiting the foregoing, the warranties of satisfactory quality, fitness for a particular purpose or non-infringement are expressly excluded and under no circumstances will GoWarrior Community be liable for direct or indirect loss or damage of any kind, including loss of profit, revenue, goodwill or anticipated savings.

This document is intended only to assist the reader in the use of the product. GoWarrior Community makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, which is used at your own risk and should not be relied upon. The information could include technical inaccuracies or typographical errors. No license, whether express, implied, arising by estoppel or otherwise, to any intellectual property right is granted by this document.

The product described in this document is subject to continuous development and improvements. GoWarrior Community also reserves the right to make changes to the specifications and product description at any time without notice.

Third-party brands and names mentioned in this publication are for identification purpose only and may be the property of their respective owners.

Android™ is a registered trademark of Google Inc. Linux® is a registered trademark of Linus Torvalds. Microsoft® and Windows® are registered trademarks of Microsoft Corporation. Supply of this product does not convey a license nor imply a right under any patent, or any other industrial or intellectual property right of Google Inc., Linus Torvalds, and Microsoft Corporation to use this implementation in any finished end-user or ready-to-use final product. It is hereby notified that a license for such use is required from Google Inc., Linus Torvalds and Microsoft Corporation.

For the latest version of this document refer to:

www.gowarriorosh.com

Copyright © 2016 GoWarrior Community All Rights Reserved.

Table of Contents

List of Tables

List of Figures

Preface

Overview

This manual aims to provide detailed guidance to developers on how to perform software development by using GoDroid v1.0 package. It describes how to configure and compile each component of GoDroid, how to develop and debug applications, and so on. It is comprised of the following chapters:

- **Chapter 1: Installation**

This chapter mainly introduces how to access the source codes and toolchains, the software and hardware requirements, and how to install the toolchains.

- **Chapter 2: Compilation**

This chapter introduces how to compile each components of the GoDroid to generate the image files for burn purpose.

- **Chapter 3: Starting up GoDroid**

This chapter provides information on starting up GoDroid and how to get serial console via serial tools for control over the TIGER Board.

- **Chapter 4: Application Development and Debugging**

This chapter introduces commonly used functions of Android Debug Bridge (abbreviated to ADB).

- **Chapter 5: FAQ**

This chapter lists frequently asked questions and answers.

Audience

This manual is primarily written to provide complete guidance for those who wants to exploit the GoWarrior platform, such as makers, tinkers, innovators, students, etc.

Applicable Products

This manual is applicable for the GoWarrior TIGER Board.

Reference Documents

- GoWarrior_GoDroid_Programming Guide_Ext-Recovery

- GoWarrior_GoDroid_ Programming Guide _Ext-Uboot

- GoWarrior_GoDroid_Application Notes_How to Build Customized Projects

- GoWarrior_GoDroid_Application Notes_Customization of Boot Logo and Animation

- GoWarrior_GoDroid_Application Notes_Recovery Upgrade Package Making

- GoWarrior_GoDroid_Application Notes_Compiling Server Installation and Configuration Guide

- GoWarrior_GoDroid_Release Notes

Conventions

Typographical Conventions

Item	Format
codes, keyboard input commands, file names, equations, and math	Courier New, Size 10.5
Variables, code variables, and code comments	*Courier New, Size, Italic*
Menu item, buttons, tool names	Ebrima, Size 10.5, Bold e.g. Select USB Debugging
Screens, windows, dialog boxes, and tabs	**Ebrima, Size 10.5, Bold** **Enclosed in double quotation marks** **e.g. Open the "Debug Configuration" dialog box**

Table 1. Typographical Conventions

Symbol Conventions

Item	Description
⚠️ *Caution*	Indicates a potential hazard or unsafe practice that, if not avoided, could result in data loss, device performance degradation, or other unpredictable results.
Note	Indicates additional and supplemental information for the main contents.
💡 *Tip*	Indicates a suggestion that may help you solve a problem or save your time.

Table 2. Symbol Conventions

How to Contact Us

Submit all your comments and error reports to the author at:

info@gowarriorosh.com

Tel: +886-2-8752-2000

Fax: +886-2-8751-1001

For questions regarding GoWarrior, contact our support team at the email listed below:

support@gowarriorosh.com

1 Introduction

This chapter focuses on requirements of the host PC and the TIGER Board, for development based on GoDroid v1.0, also how to obtain GoDroid package, and how to install toolchains.

1.1 Introduction

GoDroid is an Android development kit of the GoWarrior platform, which is based on AOSP (Android Open Source Project). It is mainly applicable for Android and Linux developers. GoWarrior is a compact, open-source, community-supported embedded Android/Linux computing platform geared toward maker/hacker/entrepreneur/dreamer/artist/student/ inventor/hobbyist/tinker. It brings together a rich feature set of innovative building blocks with cloud-driven back-end service deployment. It can be used to build complex applications that interact high-level software and low-level electronic circuits, helping you from idea through prototype to commercial mass production delivery.

GoDroid Development Kit includes some important components, such as U-Boot, Embedded Media Engine, Androidized Linux Kernel and GoWarrior-customized AOSP source codes.

1.2 Preparation for Installation

GoDroid can only run on TIGER Board with ALi M3733 chipset.

Essential software needs to be installed on the host PC for operations such as development, compilation, and debugging based on GoDroid.

1.2.1 Hardware Requirements

The table below lists the device where GoDroid can run.

Device	Chipset	Components
TIGER Board	M3733-AFAAA	5V/2A stabilized power adapter, Micro USB cable, HDMI cable or 3.5mm to RCA composite video and stereo audio cable, USB keyboard, USB mouse, Remote controller, Monitor, Ethernet cable if need wired network, USB to TTL Serial cable

Table 3. Hardware Requirements

1.2.2 Host (PC) Requirement

For development and compilation environment of GoDroid, please refer to "*GoWarrior_GoDroid_Application Notes_Compiling Server Installation and Configuration Guide*".

The host development environment of GoDroid is based on 64-bit Ubuntu, for instance Ubuntu 12.04 (*http://www.ubuntu.com/desktop/get-ubuntu/download*) and Oracle JDK 6 tool.

1.3 Getting Source Code

1.3.1 Getting GoDroid Source Code

1. Download GoDroid patch and Android source code from GitHub by

executing the following commands:

```
$ git clone https://github.com/GoWarrior/GoDroid_V1.0_patch

$ repo init -u https://github.com/GoWarriorGoDroid/manifest -b
android-4.4.4_r2

$ repo sync
```

2.   Unzip the patch by executing the following commands:

```
$ unrar e GoDroid_V1.0_patch.tar.part01.rar

$ xz -d GoDroid_V1.0_patch.tar.xz

$ tar -xvf GoDroid_V1.0_patch.tar

$ cd GoDroid_V1.0_patch/

$ ls

GoDroid_V1.0.patch  patch  runpatch.sh
```

3.   Copy *GoDroid_V1.0.patch*, *runpatch.sh* and the *patch* folder to the `android-4.4.4_r2 folder:`

```
$ cp -rf GoDroid_V1.0_patch/*  ../android-4.4.4_r2

$ cd  ../android-4.4.4_r2

$ ls

abi  bionic   build  dalvik  development  docs  frameworks hardware
libnativehelper ndk      patch  prebuilts    sdk     tools

art  bootable  cts   developers  device  external
GoDroid_V1.0.patch libcore  Makefile        packages pdk
runpatch.sh  system
```

4.   Run the patch by executing the following command:

```
$ sh  runpatch.sh
```

1.3.2 Getting PDK Source Code

1.   Download the GoDroid PDK source code by executing the following command:

```
git clone https://github.com/GoWarrior/PDK_GoDroid.git
```

2. After downloading, switch to the root directory of PDK, and then change its branch to *godroid-1.0* by executing the following command:

```
git checkout godroid-1.0
```

1.4 Installing Toolchains

1.4.1 Installing Kernel Toolchains

You may install kernel toolchain using any user account and any directory to which you have write access. For example, $HOME/toolchain, the toolchain on Linux host is arm-linux-gnueabi-gcc.

Procedure:

1. Get ARM cross compiling toolchain package.

Get the zipped package from github: source code package directory:

```
git clone git@github.com:GoWarrior/tools_arm-linux-gnueabi.git

/tools_arm-linux-gnueabi/
gcc-linaro-arm-linux-gnueabi-2012.01-20120125_linux.tar.gz
```

2. Create installation directory for compiling toolchain on GNU/Linux host: Mkdir arm_toolchain, then decompress the installation package to the newly created arm_toolchain directory.

3. Configure the environment variable PATH.

```
PATH=$HOME/arm_toolchain/bin:$PATH

Export PATH
```

1.4.2 Installing AOSP Toolchains

Please refer to "*GoWarrior_GoDroid_Compiling Server Installation and Configuration Guide*".

2   Compilation

The sections below describe how to compile each component.

2.1   Compiling Bootloader (U-Boot)

Procedure：

1.   Access the U-Boot directory.

```
$ cd uboot
```

2.   Configure U-Boot version

The U-Boot version number needs to be updated if there is any update in U-Boot. In this case, `include/configs/ali_3921_cbu.h` needs to be modified.

```
$ vim include/configs/ali_3921.h

#define BOOTLOADER_VERSION "ALiBoot1.2.0"
```

Note

The naming rule of U-Boot version is customizable according to actual requirements.

3.   Compile

```
$ make rebuild_3921
```

4.   Output

Upon completion of compilation, the following four files will be generated in the directory `/uboot`.

u-boot.bin,

u-boot.dis,

u-boot.map,

u-boot.out

5.   View output files

All the u-boot file versions will be generated in the directory：
uboot/uboot_merger/uboot_output.

```
$cd uboot/uboot_merger/uboot_output

$ls

uboot_unify_1GB_training.abs…
```

2.2 Compiling Audio Engine

Procedure：

1.   Switch to the audio engine directory by executing the following command:

```
$ cd /tds/ae_sdk/src/ae/m36f_linux
```

2.   Compile the audio engine module by executing the following commands:

```
$ make path

$ make config_3921_release

$ make generate_ae
```

2.3 Compiling SEE

Procedure：

1.   Switch to the SEE directory by executing the following command:

```
$ cd /tds/see_sdk/src/see/m36f_linux
```

2.   Compile the SEE module by executing the following commands:

```
$ make path
```

```
$ make config_3921_release

$ make generate_see
```

2.4 Compiling Android Linux Kernel

Procedure：

1. Kernel configuration

For information on configuring Android Linux kernel, please refer to
"*GoWarrior_GoDroid_Application Notes_How to Build Customized Projects*".

This step is performed only when necessary. As Android has made some
expansion and customization of Linux kernel, refer to
*http://source.android.com/devices/tech/kernel.html for kernel configuration
items recommended by Android.*

If you want to add or delete a configuration item, please modify the file
`linux/kernel/boards/TIGER_BOARD/defkernelconfig_linux-l`
`inaro-3.4-rc3.`

Or execute the following command:

```
$ cd /linux

$ make BOARD=TIGER_BOARD linux-config
```

2. Pre-copy

Run `cp_bootabs.sh` in Linux compilation root directory to copy the file
(`uboot_*.abs,ae_bin.abs,see_bin.abs`) required by Kernel
compilation.

```
$ cd /linux

$./ cp_bootabs_release.sh TIGER_BOARD
```

3. Compile

As Kernel source codes are contained in GoDroid package, execute the
commands below and you can compile Android kernel.

```
$ cd /linux

$ make BOARD= TIGER_BOARD
```

Upon completion of compilation, the output files will be generated in the directory: `/linux/install/bin`. The files required by generating Flash burning files are listed below. They need to be copied to different directories according to the Flash type.

4.   Copy and Rename.

`ae_bin.abs`

`main_bin.abs`

`see_bin.abs`

`uboot_unify_1GB_training.abs`

Copy and Rename

Access `/linux/install/bin`, and copy `see_bin.abs`, `ae_bin.abs` to `device/gowarrior/tigerboard/image/bin/`. Copy `main_bin.abs`，`uboot_unify_*_training.abs` to `device/gowarrior/tigerboard/image/fs_ubi/`.

```
$

cp  /linux/install/bin/see_bin.abs  AOSP/device/gowarrior/tigerboard
/image/bin/

cp  /linux/install/bin/ae_bin.abs  AOSP/device/gowarrior/tigerboard
/image/bin/

cp /linux/install/bin/main_bin.abs  AOSP/device/gowarrior/tigerboard
/image/fs_ubi/main_bin.abs

cp /linux/install/bin/ uboot_unify_1GB_training.abs

AOSP/device/gowarrior/tigerboard/image/fs_ubi/
```

2.5   Compiling Android File System

2.5.1 GoDroid Configuration

For GoDroid configuration, please refer to "*GoWarrior_GoDroid_Application Notes_How to Build Customized Projects".* The configuration files are located in the directory `device/gowarrior/tigerboard`, and include several main configuration files listed in the table below.

File	Description
`shell/tigerboard_init.sh`	The GoDroid custom configuration script after Android is started.

`input/tigerboard_ir.conf,`
`input/tigerboard_panel.kl,`
`input/tigerboard_remote.idc,`
`input/tigerboard_remote.kl`
	They are the key configuration files of remote controller and front panel. For more information, visit the official website at:
	
	http://source.android.com/devices/tech/input/index.html
	
	tigerboard_ir.conf is the mapping table between the physical key values and key codes of the remote controller and front panel. It can be modified according to the TIGER Board and remote controller.

`BoardConfig.mk`	It is one of the main configuration files. It defines the arguments related to Board during compilation, such as IC type, instruction sets supported, and some Board definitions.
`device.mk`	It is very important, because the variables in it define lots of information related to Board, and almost all the project settings are included in this file. For variables in this file, please refer to `build/core/product.mk`.
`fstab.tigerboard.ubifs`	The mount argument list of file system. It includes some partitions' mount arguments of NAND Flash UBIFS file system type and mount arguments of USB external USB flash disk or mobile hard disk. For more information , please refer to the introduction in the official website: *http://source.android.com/devices/tech/storage/index.html*
`fstab.tigerboard.recovery`	The mount argument list used by Recovery.
`init.tigerboard.rc`	It is a very important Android start script.

File	Description
`init.tigerboard.recovery.rc`	It is the start script of Recovery system.
`init. tigerboard.rtl8723.rc`	This is the script to enable Wi-Fi.
`system.prop`	System attributes configurations, such as time zone.
`version.mk`	It is the version configuration file used to set the version number for bootloader, recovery, and system.

Table 4. GoDroid Configuration File

2.5.2 Compiling GoDroid

GoDroid compilation mode is very similar with the native Android compilation mode. The only difference is that GoDroid compilation mode requires selection of `aosp_tigerboard-eng` during lunch, as shown in Compilation Step 2.

Preparation:

If you need to modify SDK configurations after compiling, you need to make clean first.

```
$ make clean
```

Compiling Procedure:

1. Execute `$ . build/envsetup.sh` in the directory where AOSP is located.

```
$ . build/envsetup.sh


including device/generic/armv7-a-neon/vendorsetup.sh

including device/gowarrior/tigerboard/vendorsetup.sh

including sdk/bash_completion/adb.bash
```

2. **Execute** `$ lunch`

```
$ lunch


You're building on Linux


Lunch menu... pick a combo:

    1. aosp_arm-eng

    2. aosp_x86-eng

    3. aosp_mips-eng

    4. vbox_x86-eng

    5.  mini_armv7a_neon-userdebug

    6.  aosp_tigerboard-eng

    7.  aosp_tigerboard-userdebug

    8. aosp_tigerboard-user

Which would you like? [aosp_arm-eng]
```

3. **Select** `aosp_tigerboard-eng` **as shown the figure above.**

```
Which would you like? [aosp_arm-eng] 6


==========================================

PLATFORM_VERSION_CODENAME=REL

PLATFORM_VERSION= 4.4.4

TARGET_PRODUCT= aosp_tigerboard

TARGET_BUILD_VARIANT=eng

TARGET_BUILD_TYPE=release

TARGET_BUILD_APPS=

TARGET_ARCH=arm

TARGET_ARCH_VARIANT=armv7-a-neon

TARGET_CPU_VARIANT=cortex-a9
```

```
HOST_ARCH=x86

HOST_OS=linux

HOST_OS_EXTRA=Linux-3.2.0-29-generic-x86_64-with-Ubuntu-12.04-precis
e

HOST_BUILD_TYPE=release

BUILD_ID= KTU84Q

OUT_DIR=out

==========================================

Source
/shsa022/usrhome/bobby.zhou/work/SDK6.0ha.1.3.4_AAND/device/gowarrio
r/tigerboard/shell/build_slot.sh...

Run
/shsa022/usrhome/bobby.zhou/work/SDK6.0ha.1.3.4_AAND/device/gowarrio
r/tigerboard/shell/build_pre.sh...

python
/shsa022/usrhome/bobby.zhou/work/SDK6.0ha.1.3.4_AAND/build/tools/hos
ttools/python/genpartsizemk.py
/shsa022/usrhome/bobby.zhou/work/SDK6.0ha.1.3.4_AAND/image/Ali_nand_
desc.xml
/shsa022/usrhome/bobby.zhou/work/SDK6.0ha.1.3.4_AAND/device/gowarroi
r/tigerboard/image/fs_ubi/partsize.mk
```

4. Compile make

```
$ make –j8
```

-j8 in the make parameter indicates that eight threads are used to compile AOSP. The make –j I usually gotten by `cat /proc/cpuinfo |grep -c 'processor'`. AOSP compilation may takes some time. It usually takes about 40 to 50 minutes in case of quad-core and eight threads, and takes about ten minutes in case of 48 cores and 48 threads.

5. Output

The output of AOSP compilation is located in the directory

`/out/target/product/tigerboard/` which mainly includes the following three directories.

```
data/
```

```
root/

system/
```

2.5.3 Compiling GoDroid Recovery

You can compile recovery after compiling GoDroid.

Compiling command:

```
$ make recoveryimage
```

Compiled Recovery is outputted in this directory:
`out/target/product/tigerboard/`. It mainly includes:

```
recovery/

recovery_system/
```

2.6 Generating Image Files

After completing the compilation steps above, you need to generate other files like `cache.img` and `ALI.ini`.A complete burn package can be generated by just running the build image command from the root directory of Android compilation. The burn package directory is located in the image directory of the root directory.

```
$ build image

$cd image/burn

$ls

$

cache.img

ALI.ini

deviceinfo.abs

recovery.ubosdram_C3921_BGA462_1GB_1600Mbps.abs

backup.abs

baseparams.abs
```

```
kernel.ubo

system.img

bootargs.abs

uboot_unify_1GB_training.abs

bootmedia.ubo

NandList_v2.ran

userdata.img
```

2.7 Burning GoDroid Image Files

After compiling all the components of GoDroid and generating corresponding image files, you can burn all the image files to the TIGER Board. For more information, please refer to "*GoWarrior_ GoDroid_GoDroid Installation* ".

3 Starting up GoDroid

3.1 Starting up GoDroid

You can start up GoDroid after connect TIGER Board and monitor by the following steps.

Procedure:

1. Connect a HDMI cable to the HDMI interface of the TIGER Board and to HDMI input port on Monitor; or connect the CVBS video cable to CVBS output port and video input port on Monitor. Besides, ensure that the correct Monitor video input port is selected.

   Turn on the Monitor and ensure that correct Monitor input port (HDMI or video) is selected.

2. Connect the Ethernet cable (optional).

3. Connect one end of the serial debugging daughter board to TTL serial port of the TIGER Board and the other end to the host, the connection is shown in Figure 1. Run a serial console on the host and set information like serial port and baud rate (115200) as shown in the figure below.
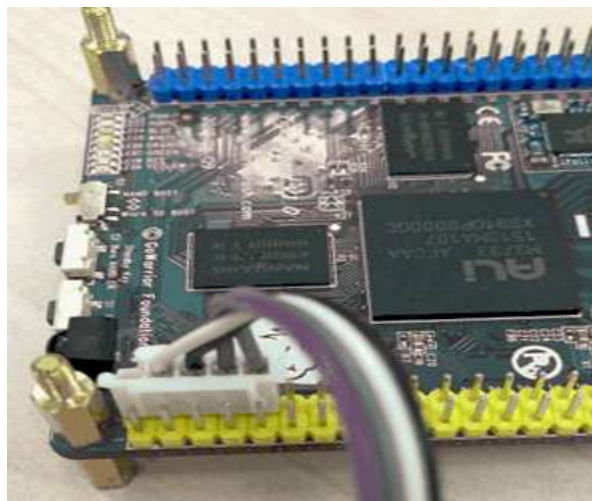

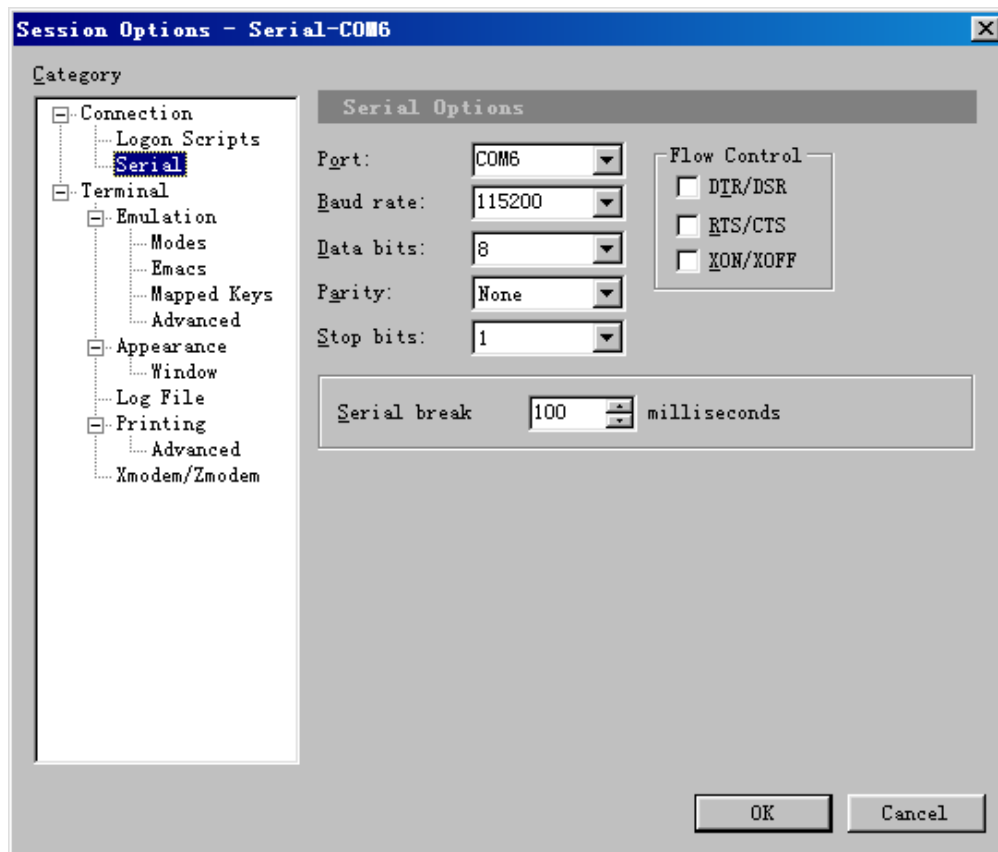
Figure 1. Connecting Serial Port

Figure 2. Serial Port Setup

Note:

 "Parity" needs to be set according to the actual situation, like the status of board and serial port daughter board.

4. Insert a USB flash disk or portable HDD to USB Port1 or Port2 (optional).

5. Connect the USB keyboard or USB mouse to USB Port1 or Port2 (optional).

6. Connect the USB cable to USB port0.
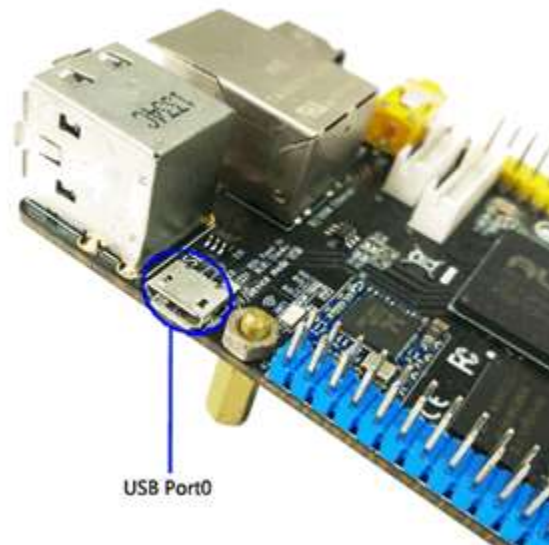
Figure 3. USB Port0

Figure 4. USB Port1/Port2

3.2 Serial Console

After the TIGER Board is powered on and GoDroid starts up, the serial console will print the GoDroid boot processes. The printed information is as shown below：

```
U-Boot 2012.04.01 (Dec 12 2014 - 10:20:39)



CPU  : ALi M3921 BGA package

Board: S3921 SB/DB

Top of RAM usable for U-Boot at: 8bff0000

Reserving 780k for U-Boot at: 8bf2c000

Reserving 12288k for malloc() at: 8b32c000

Reserving 36 Bytes for Board Info at: 8b32bfdc

Reserving 120 Bytes for Global Data at: 8b32bf64

New Stack Pointer is: 8b32bf58

DRAM:  192 MiB

Now running in RAM - U-Boot at: 8bf2c000
```

Until the GoDroid startup is completed, after TV displays the home screen, you can enter control commands on serial console to display system information or to control system operation. For example, you can use some commonly-used commands such as cd, ls, and logcat.

" `ls`" command is running as below:

```
~ # ls
acct
cache
config
d
data
default.prop
dev
etc
file_contexts
fstab.tigerboard
init
init.environ.rc
init.rc
init.tigerboard.rc
init.tigerboard.rtlwifi.rc
init.tigerboard.usb.rc
init.trace.rc
```

```
init.usb.rc
initlogo.rle.bak                23
mnt
proc
property_contexts
root
sbin
sdcard
seapp_contexts
sepolicy
storage
sys
system
tmp
ueventd.rc
ueventd.tigerboard.rc
usbdrive
vendor

~ #
```

4 Application Development and Debugging

4.1 Using ADB Debugger

ADB is a versatile tool that lets you manage the state of the Android-powered device. For more information about what is possible with ADB, see Android Debug Bridge page at
http://developer.android.com/tools/help/adb.html.

The ADB tool can be used to

- Download an application from a host machine, install and run it on the target board.

- Start a remote shell in the target instance.

- Debug applications running on the device using the debugging tool DDMS (Dalvik Debug Monitor Server) which runs on top of ADB connection.

- Copy files from and to the board.

4.1.1 ADB over Ethernet

Procedure:

1. Select "**USB debugging**" from "**Developer options**" in the system settings menu.

2. Type the "`start adbd`" command in the serial output of the target table.

3. Make sure that the host PC and the TIGER Board is properly connected via a network cable and can be pinged mutually.

4. Type the "`netcfg`" command in the serial output of the target table,

the IP address: 192.168.1.100 is the IP address of the TIGER Board.

```
~ # netcfg

lo       UP                          127.0.0.1/8    0x00000049
00:00:00:00:00:00

p2p0     DOWN                        0.0.0.0/0      0x00001002
5a:63:56:10:d6:e2

eth0     UP                      192.168.1.100/24   0x00001043
00:41:4c:69:a0:27

wlan0    DOWN                        0.0.0.0/0      0x00001002
58:63:56:10:d6:e2
```

5. On the host PC, use the following commands to establish ADB connection.

```
$ adb connect <target_ip_address>
```

6. Verify device connectivity by executing the command "$ adb devices". If connected, you can find the device name list.

7. If the device isn't displayed, please restart ADB server to reconnect the TIGER Board.

```
$ adb kill-server

$ adb start-server

$ adb connect <target_ip_address>
```

4.1.2 ADB over USB

For Detail Information of ADB over USB, please refer to "*GoWarrior_GoDroid_Application Notes_USB ADB Debugging*".

4.2 Operations over ADB

Once after ADB on host PC has been connected to the TIGER Board, ADB tool can be used to do some operations as follows.

4.2.1 Installing Application (.apk files)

You can use ADB tool for package installation from the host PC.

```
$ adb install <package.apk>
```

4.2.2 Uninstalling Application (.apk) Using ADB

To uninstall an application (.apk), firstly execute the following command on the host machine.

```
$ adb shell pm list packages
```

Then view what packages (application) there are, and execute the following command to uninstall the package (application) what you want.

```
$ adb uninstall <package name>
```

4.2.3 Copying Files From and To the Board over ADB

Use the ADB commands "push" and "pull" to copy files to and from the board.

To copy a file or directory (recursively) from the board, use

```
$ adb pull <remote> <local>
```

To copy a file or directory (recursively) to the board, use

```
$ adb push <local> <remote>
```

In the above commands, for `<local>` and `<remote>`, please refer to the paths of the file or directory on your development host (local) and on the target instance (remote).

Here's an example:

```
$ adb push Antutu_4.1.5.apk /data/app/Antutu_4.1.5.apk
```

It means that Antutu_4.1.5.apk under the local directory will be copied to the `/data/app/`directory on the target board.

4.3   Bootloader Development

GoDroid bootloader uses open source project of U-Boot. For more information on modification in U-Boot, please refer to "*GoWarrior_GoDroid_Programming Guide_Ext-Uboot*".

4.4   Recovery Development

GoDroid provides a complete recovery (system upgrade) solution that supports OTA/IP/USB. This recovery module possesses enhanced functionalities based on Android native Recovery module. For detailed information, please refer to "*GoWarrior_GoDroid_Programming Guide_Exit-Recovery*".

For information on making a recovery upgrade package, please refer to "*GoWarrior_GoDroid_Application Notes_Recovery Package Making*".

4.5   Customization of Android Boot Logo and Animation

For details, please refer to "*GoWarrior_GoDroid_Application Notes_ Customization of Boot Logo and Animation*".

4.6   Using GoDroid SDK's Android Java Library in Android Studio

Android Studio can use different versions of SDK that can be downloaded from Google website by SDK Manager. APP developers can use the downloaded SDK to develop APP to run on GoDroid. But if you want to use the extra features which are provided by GoDroid, you need to use the GoDroid SDK's Android Java library instead of the standard one from Google.

To use the GoDroid SDK's Android Java library, it's only requires to replace the Android Java library file in Android Studio SDK with the GoDroid. The location of Android SDK in Android Studio is displayed in the dialog of SDK manager.
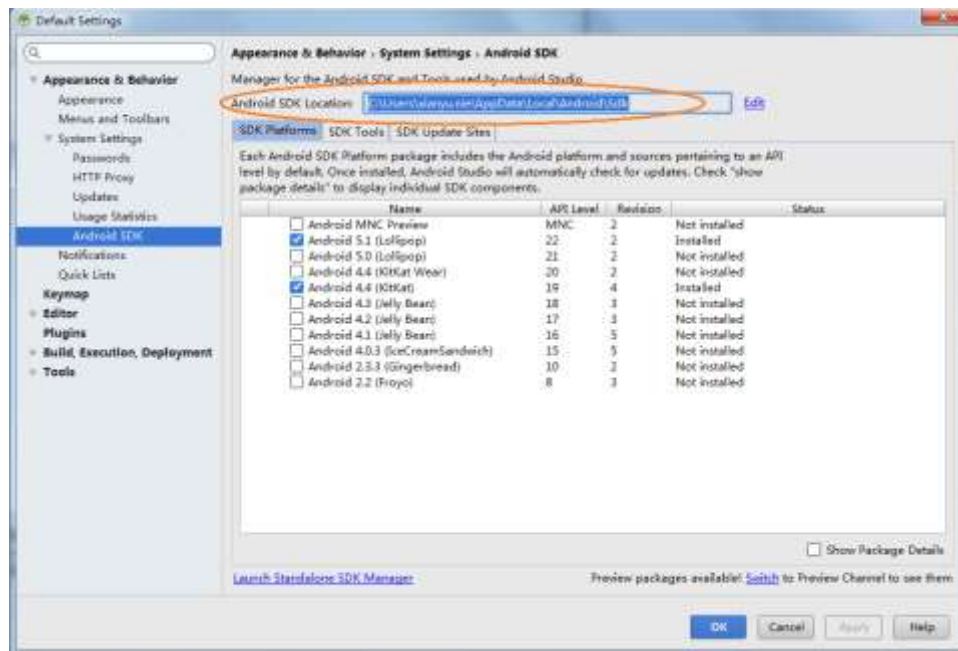
Figure 5. SDK Manager

As GoDroid 1.0 is based on Android 4.4 (KitKat), so you only need to replace the `android.jar` for API level 19. Which is to replace the Android SDK's following path `platforms\android-19\android.jar` with the `android.jar` from GoDroid SDK.

In the example of Figure 5, the file needs to be replaced with `C:\Users\xianyu.nie\AppData\Local\Android\Sdk\platforms\android-19\android.jar`.

5　FAQ

No.	Problem Description	Solution
1	Cannot compile `$ make BOARD=TIGER_BOARD` when building Kernel	Check if the compiler path of `/linux/kernel/boards/TIGER_BOARD variables.mak` file is the compiler path of your compiling server. If it is not the following default path, please change it to the compiler path of your compiling server. export DEV_CROSS_COMPILE_PATH = /opt/arm-linux-gnueabi
2	No printout for TTL serial of board	Please confirm your serial board and check out whether your serial line is a parallel or crossover cable. Besides, please make sure that the serial console configuration items like Baud rate and Parity are set correctly.

3	No response for ADB commands	First check if the ADB tool has been connected to the TIGER Board using the Ebrima`$ adb devices"` command. If not connected, please use the "`$ ADB connect"` command, or re-plug the USB cable to confirm whether connection has been established.
		
		 ADB may be unstable sometimes. Use the following commands to restart ADB.
		
		 `$ adb kill-server`
		
		 `$ adb start-server`

Table 5. FAQ List

Appendix: Glossary

Abbr.	Full Name
ADB	Android Debug Bridge
AOSP	Android Open Source Project
GoDroid	GoWarrior Android Development Kit

Table 6. List of Abbreviations

Revision History

Document Change History

Revision	Changes	Date
v1.01	Updated GoDroid and PDK source code download commands. Updated the Kernel compile steps.	March 11, 2016
v1.0	Initial Release	September 07, 2015

Table 7. Document Change History

Software Changes

Revision	Changes	Date
v1.0	Initial Release	September 07, 2015

Table 8. Software Change History