

TCC805x Android 10 SDK

Getting Started

Rev. 1.41 [A]
2023-04-06

※ The information in this document is subject to change without notice and should not be construed as a commitment by Telechips, Inc.

Kindly visit www.telechips.com for more information.

© 2023 Telechips, Inc. All rights reserved.

TABLE OF CONTENTS

Contents

TABLE OF CONTENTS	2
1 Introduction	5
2 Board Description	5
2.1 EVB Version	5
2.1.1 CPU Board.....	6
2.1.2 Main Board	9
2.1.3 12.3" 1920x720 LCD.....	11
2.1.4 Jog Navigation Sub-board	12
2.2 Boot Mode	13
2.2.1 USB Boot Mode.....	13
2.2.2 eMMC Boot Mode	14
2.2.3 SNOR/eMMC Boot Mode	14
2.2.4 SNOR/UFS Boot Mode	15
2.2.5 UFS Boot Mode	15
3 Set Up Environment	16
3.1 Set Up Linux Host Machine	16
3.1.1 Hardware Requirements	16
3.1.2 Software Requirements.....	16
3.1.2.1 Install Required Packages	16
3.1.2.2 Java Development Kit (JDK).....	18
3.1.2.3 Key Packages	18
3.1.2.4 Toolchain Path	18
3.1.3 Test Environment of Telechips	19
4 Build Guide.....	20
4.1 TCC805x EVB (64-bit SDK)	20
4.1.1 Download SDK.....	20
4.1.2 Compile and Build Android Framework	21
4.1.2.1 Set Up Compiling Environment	21
4.1.2.2 Compile Bootloader	23
4.1.2.2.1 Set Configuration of Toolchain for U-Boot	23
4.1.2.2.2 Build U-Boot and Make Rom-file	24
4.1.2.3 Compile Linux Kernel.....	25
4.1.2.3.1 Set Kernel as Default.....	25
4.1.2.3.2 Kernel Configuration – Device Tree	25
4.1.2.3.3 Kernel Configuration	26
4.1.2.3.4 Compile Kernel.....	26
4.1.2.3.5 Output	27
4.1.2.4 Compile Framework	28
4.1.3 Boot-firmware.....	29
4.1.4 Build Sub-core by Autolinux	30
4.1.4.1 Autolinux Usage.....	31
4.1.4.2 Build Sub-core with Configurations	32
4.1.4.3 Prebuilt Sub-core images without Building sub-core	33
4.1.5 Make SD Data.....	34
4.1.5.1 SD Data Partition Information for TCC805x EVB (64-bit SDK)	34
4.1.5.2 Partition List File	35
4.1.5.3 Use mktcimg	36
4.1.5.4 Make SNOR ROM Image	37
4.1.5.4.1 "mkimage.cfg" File	37
5 Firmware Downloader (FWDN) Guide.....	39
5.1 System Requirement.....	39
5.1.1 Software Requirement.....	39
5.1.2 Hardware Requirement.....	39
5.2 Install VTC Driver	40
5.3 Firmware Download Sequence	41
6 Partition Update.....	42
6.1 Fastboot Flash Memory Update	42
6.1.1 What is Fastboot	42
6.1.2 Fastboot Mode	42
6.1.2.1 Fastboot in Bootloader (fastboot)	42
6.1.2.2 Fastboot in Recovery (fastbootd)	43
6.1.3 Install USB Driver.....	44

6.1.4	Fastboot Flash Memory Update.....	44
7	Fast Cold Boot	45
7.1	Optimize Kernel.....	45
7.2	Optimize Android Framework.....	45
8	Android Application Guide.....	46
8.1	Launcher	46
8.2	Local Media Player	47
8.3	Heating, Ventilation, and Air Conditioning (HVAC).....	49
9	Telechips Application Guide.....	50
9.1	Overview	50
9.2	Application List	50
9.3	Enable/Disable Option	50
9.4	TCCarLauncher.....	51
9.5	TMPlayer	52
9.5.1	Overview.....	52
9.5.2	Feature	52
9.5.3	GUI Guide	52
9.5.4	Audio Play	53
9.5.5	Video Play	54
10	Debugging Guide	55
10.1	Android Debug Bridge (ADB) Guide	55
10.1.1	Set Up ADB	55
10.1.2	Enable ADB Debugging Mode	55
10.1.3	Useful ADB Commands	56
10.2	Logcat Command-line Tool	57
10.3	Dmesg.....	57
11	References	58
12	Revision History.....	59
	Rev. 1.41: 2023-04-06	59
	Rev. 1.40: 2023-01-20	59
	Rev. 1.30: 2022-11-23	59
	Rev. 1.22: 2022-10-14	59
	Rev. 1.21: 2022-04-28	59
	Rev. 1.20: 2021-12-23	60
	Rev. 1.12: 2021-03-29	60
	Rev. 1.11: 2020-11-05	60
	Rev. 1.10: 2020-09-25	60
	Rev. 1.00: 2020-09-21	60
	Rev. 0.01: 2020-07-14	61

Figures

Figure 2.1	TCC8050_53_LPD4_4X322_V1.0.0.....	6
Figure 2.2	TCC8059_LPD4_4X321_V1.0.0	7
Figure 2.3	TCC803XP_LPD4321_V1.0.0.....	8
Figure 2.4	TCC805X_6X_MAIN_V1.0.0.....	9
Figure 2.5	TCC80XX_BOE_WLCD_12.3_SV1.0.0.....	11
Figure 2.6	TCC80XX_KEY_SV0.1 (Top View).....	12
Figure 2.7	TCC80XX_KEY_SV0.1 (Bottom View).....	12
Figure 5.1	Installation Screen of VTC Driver	40
Figure 6.1	Run Fastboot	42
Figure 6.2	Connected Fastboot.....	43
Figure 8.1	Bootup Screen	46
Figure 8.2	List of All Applications.....	46
Figure 8.3	List of Music Application (1)	47
Figure 8.4	List of Music Application (2)	47
Figure 8.5	Local Media Player	48
Figure 8.6	Music Player Screen.....	48
Figure 8.7	HVAC Application	49
Figure 9.1	Bootup Screen	51
Figure 9.2	List of All Applications.....	51
Figure 9.3	Navigation.....	51
Figure 9.4	Audio Play Screen	53
Figure 9.5	Audio Information Pop-up.....	53
Figure 9.6	Audio File List Screen	54
Figure 9.7	Video Play Screen.....	54
Figure 10.1	USB Debugging Option	55

Figure 10.2 ADB Device List.....	55
----------------------------------	----

Tables

Table 2.1 EVB Version	5
Table 2.2 Codec Selection.....	10
Table 2.3 Set to USB 2.0 Boot Mode	13
Table 2.4 Set to USB 3.0 Boot Mode	13
Table 2.5 Set to eMMC Boot Mode.....	14
Table 2.6 Set to SNOR/eMMC Boot Mode	14
Table 2.7 Set to SNOR/UFS Boot Mode	15
Table 2.8 Set to UFS Boot Mode.....	15
Table 4.1 Partition Information of SD Data for TCC805x EVB.....	34
Table 5.1 Software Requirement	39
Table 5.2 Hardware Requirement	39
Table 10.1 ADB Command	56

1 INTRODUCTION

This document describes Automotive Android SDK (AAS) usage as follows:

- Board Description
- Build Guide
- FWDN Guide
- Partition Update
- Fast Cold Boot
- Android Application Guide
- Telechips Application Guide
- Debugging Guide
- Set up environment (Toolchain)

2 BOARD DESCRIPTION

2.1 EVB Version

Table 2.1 EVB Version

Board	Board Name/Version
CPU Board	TCC8050_53_LPD4_4X322_V1.0.0
	TCC8059_LPD4_4X321_V1.0.0
	TCC803XP_LPD4321_V1.0.0
Main Board	TCC805X_6X_MAIN_V1.0.0
	TCC805X_6X_MAIN_V1.2.2
	TCC805X_6X_MAIN_V1.3.3
12.3" 1920x720 LCD	TCC80XX_BOE_WLCD_12.3_SV1.0.0
Jog Navigation Sub-board	TCC80XX_KEY_SV0.1

2.1.1 CPU Board

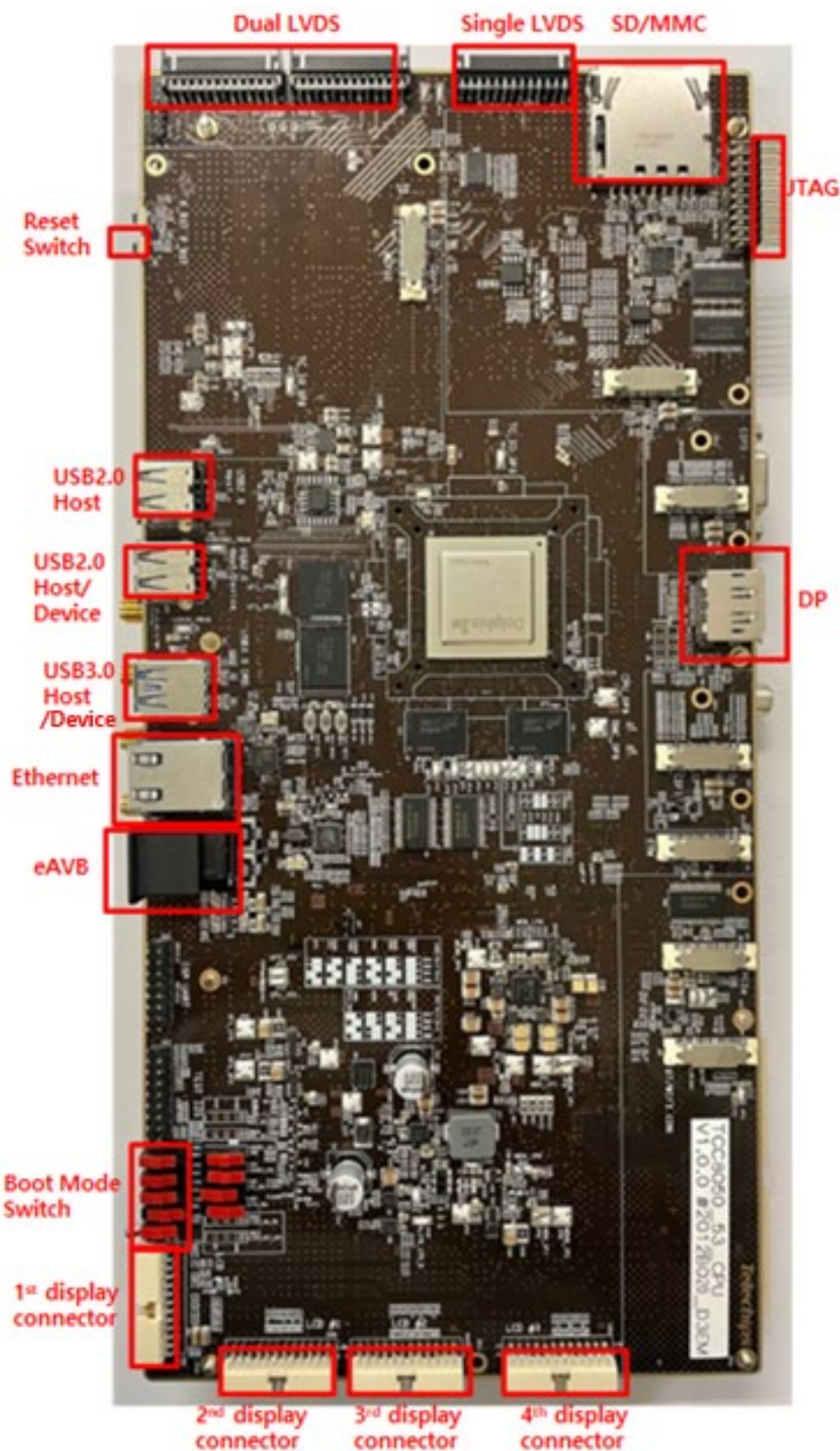
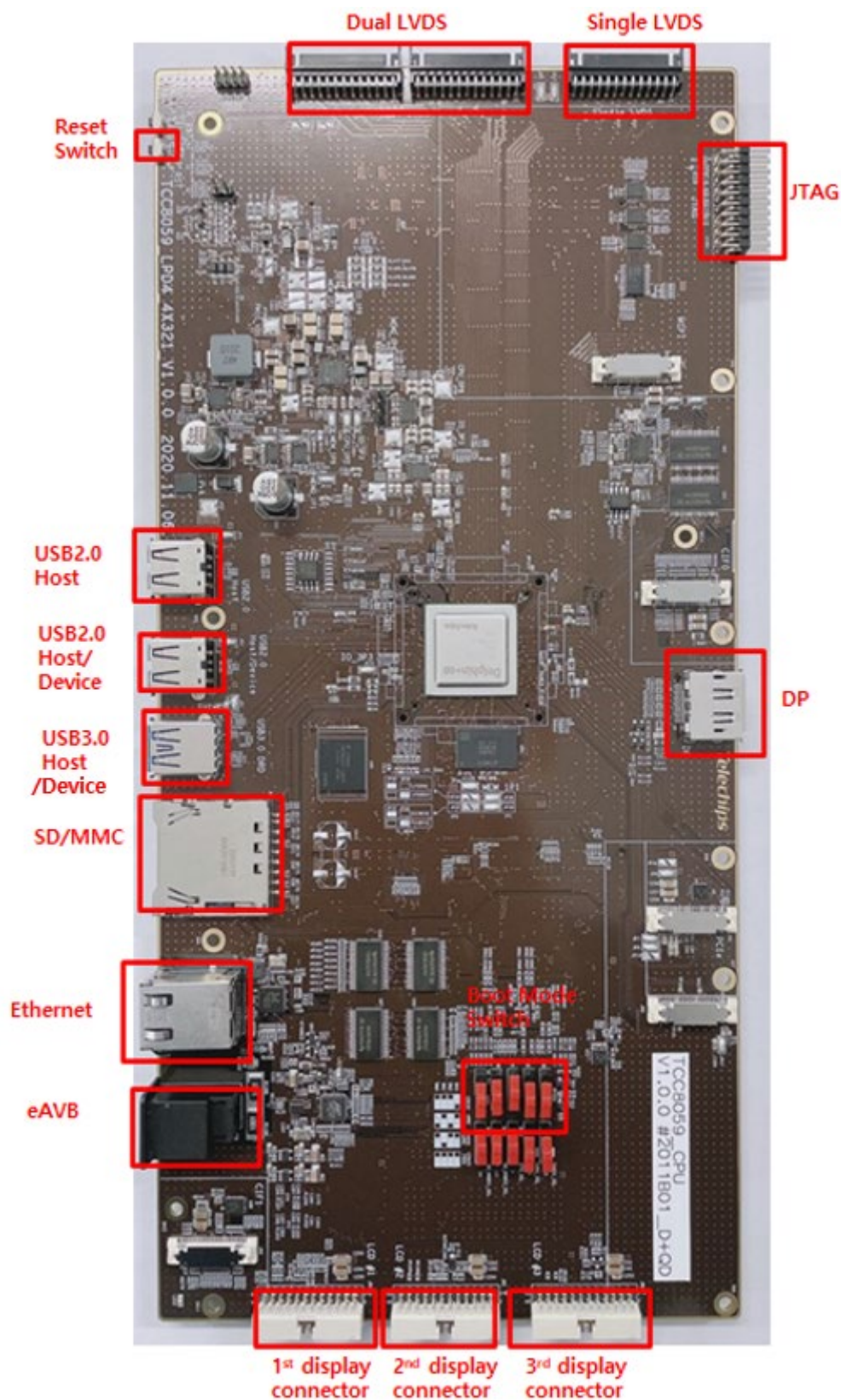


Figure 2.1 TCC8050_53_LP4_4X32_V1.0.0

**Figure 2.2 TCC8059_LPD4_4X321_V1.0.0**

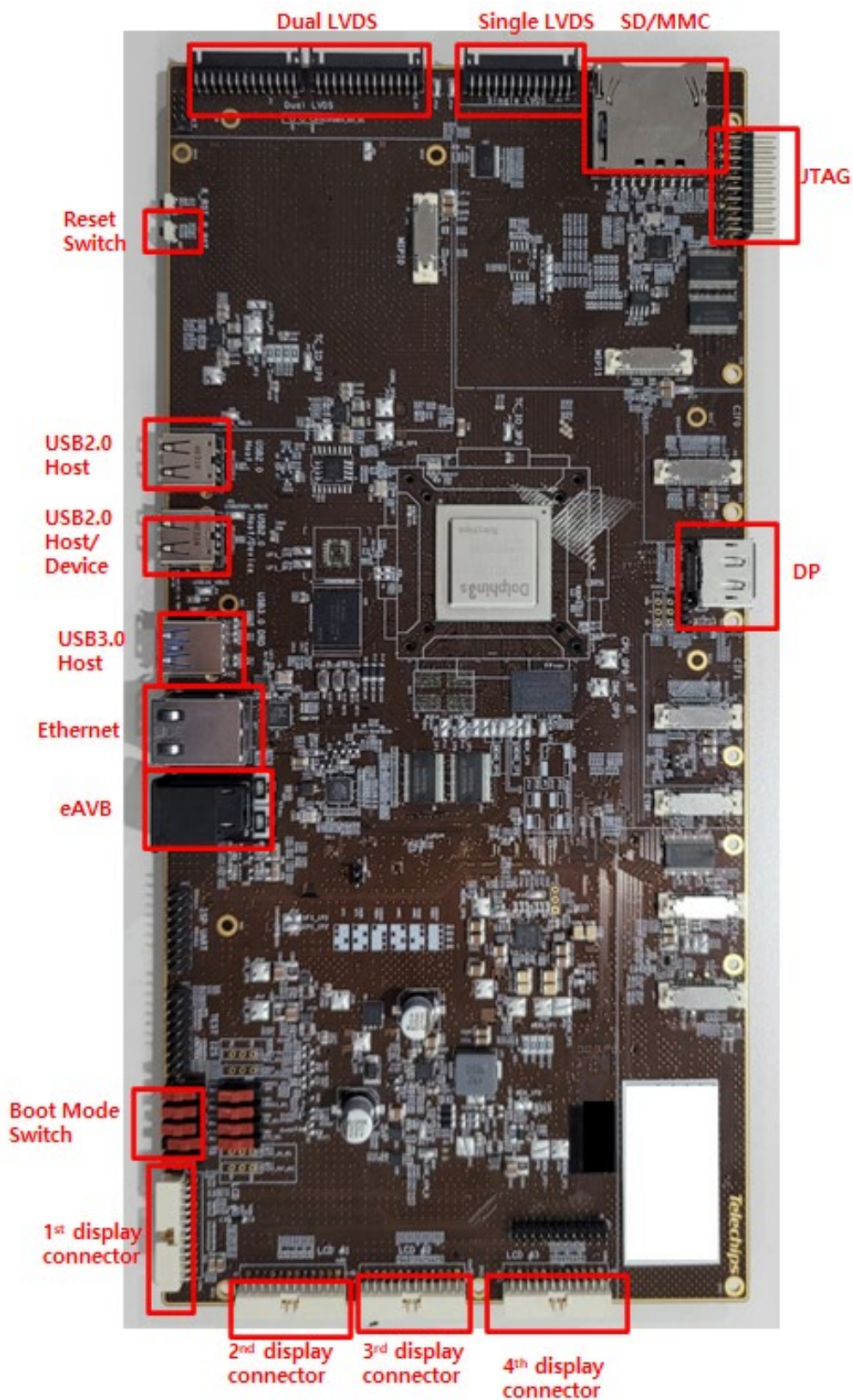


Figure 2.3 TCC803XP_LPD4321_V1.0.0

2.1.2 Main Board

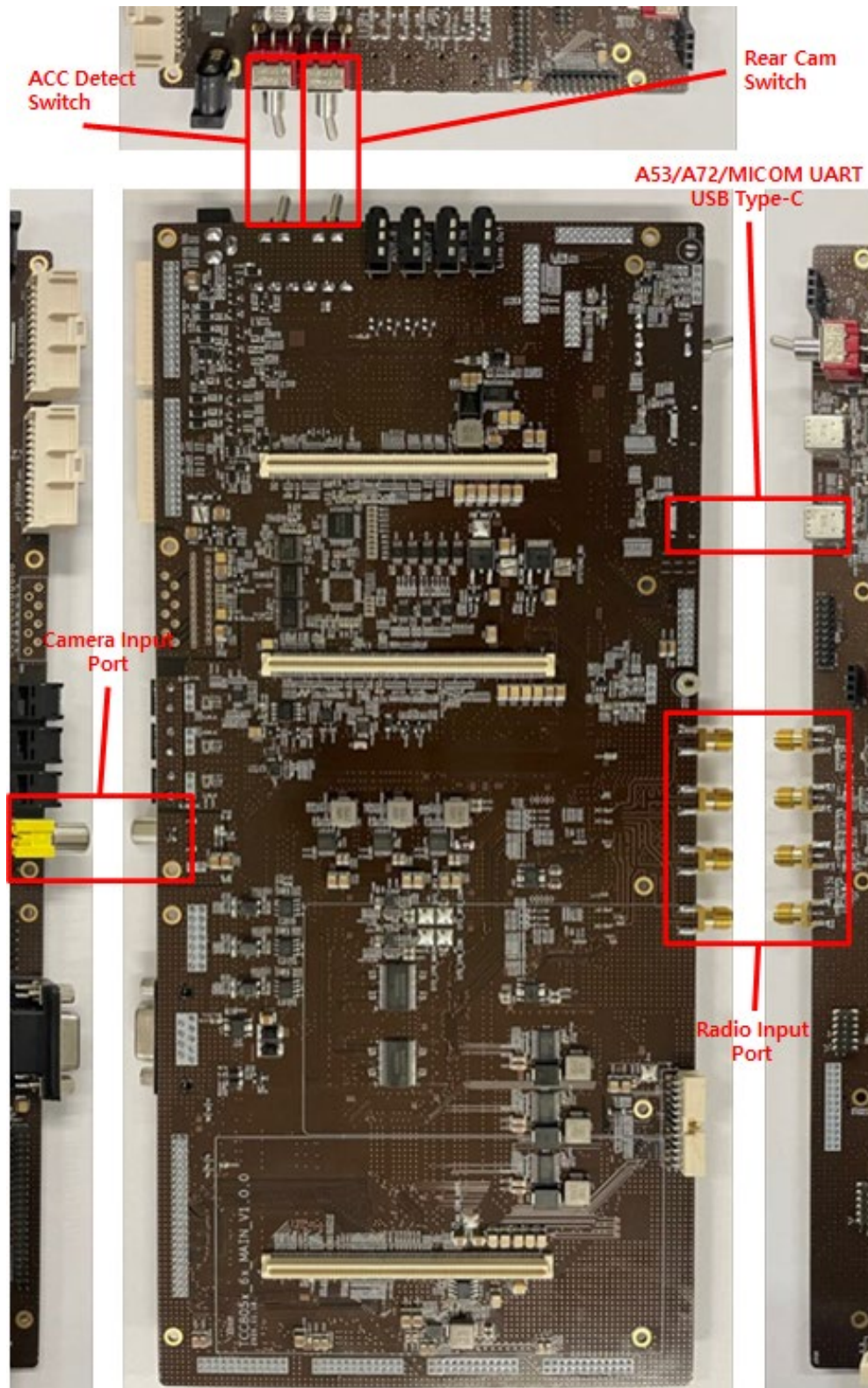




Figure 2.4 TCC805X_6X_MAIN_V1.0.0

Note: The figures of main board for V1.2.2 and V1.3.3 are the same as V1.0.0

Note: In the case of TCC805x_Android10_IVI_3.0.0, AK4601 codec is supported by default. Refer to Table 2.2.

Table 2.2 Codec Selection

Category	Function Switch	Description
CODEC_SEL		AK4601
		TCC7604

2.1.3 12.3" 1920x720 LCD

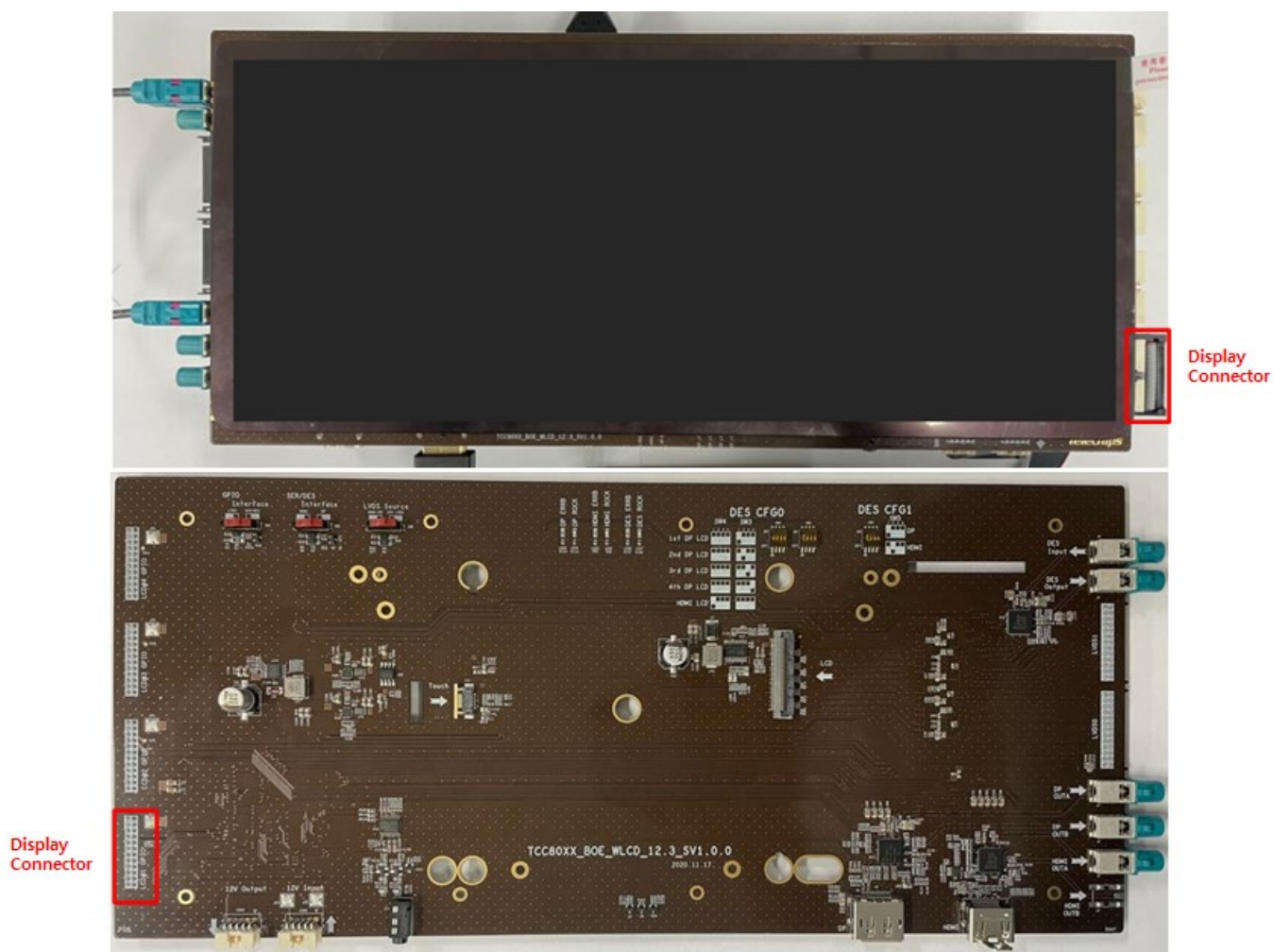


Figure 2.5 TCC80XX_BOE_WLCD_12.3_SV1.0.0

2.1.4 Jog Navigation Sub-board

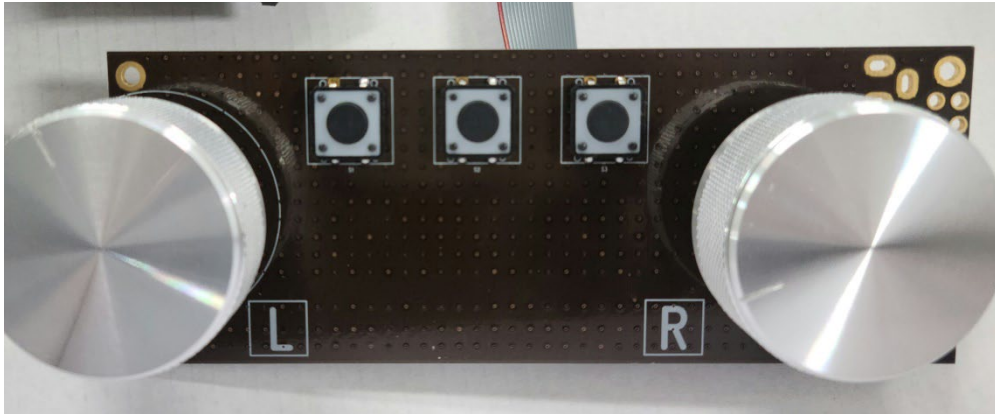


Figure 2.6 TCC80XX_KEY_SV0.1 (Top View)

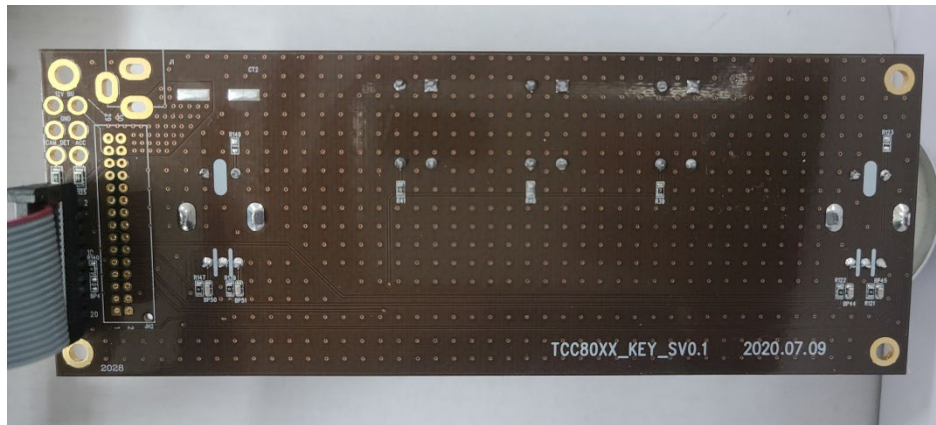


Figure 2.7 TCC80XX_KEY_SV0.1 (Bottom View)

2.2 Boot Mode

2.2.1 USB Boot Mode

Table 2.3 Set to USB 2.0 Boot Mode

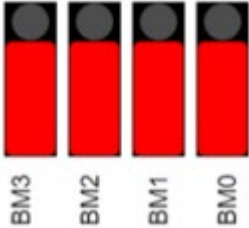
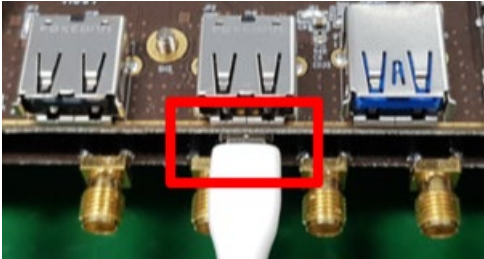
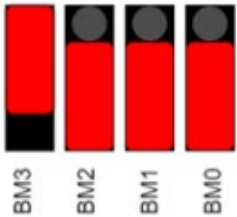
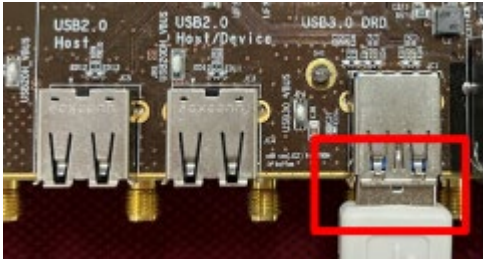
Category	USB Boot Mode Switch	FWDN Port
TCC805x/TCC803xPE		

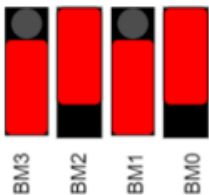
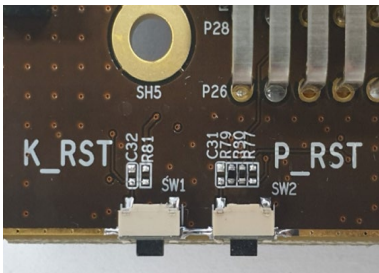
Table 2.4 Set to USB 3.0 Boot Mode

Category	USB Boot Mode Switch	FWDN Port
TCC805x		

Note: If you download the firmware by using USB3.0 Boot mode, download speed will be improved. Use a USB3.0 cable.

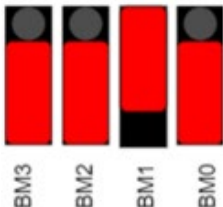
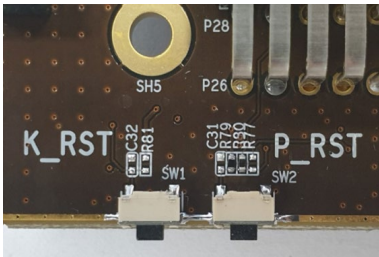
2.2.2 eMMC Boot Mode

Table 2.5 Set to eMMC Boot Mode

Category	eMMC Boot Mode Switch	Reset Switch	Note
TCC805x/TCC803xPE	 <p>BM3 BM2 BM1 BM0</p>		<p>Reset button on the bottom left corner of the board.</p> <p>K_RST resets power management I/O block. P_RST reboots a system.</p>

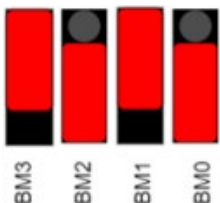
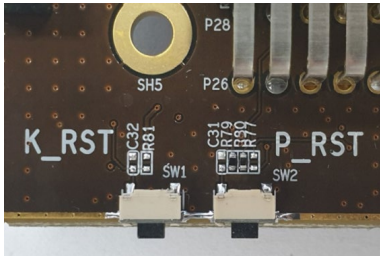
2.2.3 SNOR/eMMC Boot Mode

Table 2.6 Set to SNOR/eMMC Boot Mode

Category	SNOR/eMMC Boot Mode Switch	Reset Switch	Note
TCC805x/TCC803xPE	 <p>BM3 BM2 BM1 BM0</p>		<p>Reset button on the bottom left corner of the board.</p>

2.2.4 SNOR/UFS Boot Mode

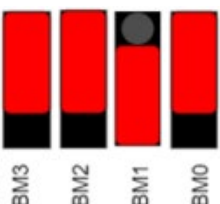
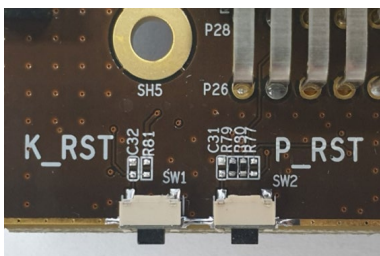
Table 2.7 Set to SNOR/UFS Boot Mode

Category	SNOR/UFS Boot Mode Switch	Reset Switch	Note
TCC8050/TCC8053			Reset button on the bottom left corner of the board.

Note: TCC8059 does not support UFS.

2.2.5 UFS Boot Mode

Table 2.8 Set to UFS Boot Mode

Category	UFS Boot Mode Switch	Reset Switch	Note
TCC8050/TCC8053			Reset button on the bottom left corner of the board.

3 SET UP ENVIRONMENT

This chapter describes how to set up the environment to build the Android source files.

Important: You must use Linux or Mac OS because building on Windows is not currently supported.

3.1 Set Up Linux Host Machine

3.1.1 Hardware Requirements

Your development host machine should meet the following hardware requirements:

- 64-bit environment is required.
- At least 250 GB of free disk space to check out the code and an extra 150 GB to build the code are required. If you conduct multiple builds, you will need even more space.
- If you run Linux in a virtual machine, you need at least 16 GB of RAM/Swap.

3.1.2 Software Requirements

3.1.2.1 Install Required Packages

To set up the build environment recommended by google, refer to <https://source.android.com/docs/setup/start/initializing>.

If you want to set up a build environment tested by Telechips, see the followings:

- **On Fedora 20**
 - Required packages

```
yum -y install glibc*
yum -y install glibc-devel*
yum -y install libstdc++*
yum -y install libstdc++-devel*
yum -y install python-devel
yum -y install perl-XML-Xerces.x86_64*
yum -y install aalib-libs*
yum -y install antlr-javadoc*
yum -y install baekmuk-bdf-fonts*
yum -y install baekmuk-ttf-fonts-batang*
yum -y install baekmuk-ttf-fonts-dotum*
yum -y install baekmuk-ttf-fonts-gulim*
yum -y install baekmuk-ttf-fonts-hline*
yum -y install createrepo*
yum -y install deltarpm*
yum -y install docbook-utils-pdf*
yum -y install fonts-IS08859*
yum -y install gitk*
yum -y install global*
yum -y install graphviz*
yum -y install gtksourceview2*
yum -y install gupnp-devel*
yum -y install gypsy-devel*
yum -y install jakarta-commons-beanutils-javadoc*
yum -y install libdiscid*
yum -y install libogg-devel*
yum -y install libspiro*
yum -y install libssh2-devel*
yum -y install libuser-devel*
yum -y install libX11*
yum -y install libXrandr*
yum -y install man-pages-ko*
yum -y install meld*
yum -y install monodoc*
yum -y install ncurses-devel*
```



```

yum -y install perl-devel*
yum -y install perl-ExtUtils-MakeMaker*
yum -y install rapidsvn*
yum -y install SDL-devel*
yum -y install sos*
yum -y install system-config-date*
yum -y install system-config-httpd*
yum -y install vim-X11*
yum -y install w3m*
yum -y install wxBase*
yum -y install wxGTK*
yum -y install dx*
yum -y install xfsprogs*
yum -y install xorg-x11-fonts-IS08859*
yum -y install xorg-x11-fonts-misc*
yum -y install xz-lzma-compat*
yum -y install readline*
yum -y install libX11-devel*
yum -y install glibc-devel*
yum -y install readline-devel*
yum -y install acpi*
yum -y install libx86*
yum -y install libssl*
yum -y install hdparm*
yum -y install gucharmap*
yum -y install libX11*
yum -y install libX11-devel*
yum -y install x11-ssh-askpass*
yum -y install libX11-devel*
yum -y install libX11.x86_64*
yum -y install python
yum -y install bleachbit.noarch*
yum -y install libstdc++.i686*
yum -y install zlib.i686*
yum -y install zlib-devel.i686*
yum -y install python-devel.i686*
yum -y install xerces-c-devel.i686*
yum -y install readline.i686*
yum -y install libX11-devel.i686*
yum -y install glibc-devel.i686*
yum -y install readline-devel.i686*
yum -y install libx86.i686*
yum -y install libX11.i686*
yum -y install libX11-devel.i686*
yum -y install libX11.i686*
yum -y install libX11-devel.i686*
yum -y install libXrender.i686*
yum -y install vim-X11.x86_64*
yum -y install gperf
yum -y install perl-Switch.noarch
yum -y install libmp*
yum -y install tmux
yum -y install java-1.8.0-openjdk*
yum -y install vim
yum -y install sysstat
yum -y install zlib-devel
yum -y install openldap-clients nss-pam-ldapd
yum -y install java-1.8.0-openjdk*

```

■ GCC upgrade (v5.3.1)

```

yum install gcc-c++ --nogpgcheck --releasever=23
yum install libgcc-5.3.1-6.fc23.i686 --nogpgcheck --releasever=23

```

- **On Ubuntu 20.04**
 - Required packages

```
apt install -y net-tools make build-essential cmake gcc g++ perl openjdk-11-jdk ncftp chrony diffstat
ctags

apt install -y git gnupg flex bison zip curl zlib1g-dev gcc-multilib g++-multilib libc6-dev libc6-dev-
i386 lib32ncurses-dev x11proto-core-dev libx11-dev lib32z1-dev libgl1-mesa-dev libxml2-utils xsltproc
unzip fontconfig sqlite libffi-dev libbz2-dev libreadline-dev libsqlite3-dev libssl-dev lzop dos2unix
python2 python-dev python3-dev libncurses5 xz-utils tk-dev manpages-dev bzip2 device-tree-compiler
libspiro-dev p7zip
apt install -y gawk wget texinfo chrpath socat cpio python3 python3-pip python3-pexpect debianutils
iputils-ping python3-git python3-jinja2 libegl1-mesa libstdl1.2-dev pylint3 xterm python3-subunit mesa-
common-dev
apt install -y libncurses* subversion automake
apt install -y zstd
apt install -y git
```

Note 1: For the details on the required packages for Linux Host Machine to build sub-core, refer to Chapter 3.4.3 List of Packages Required for Host Development System to Use Yocto Project in "*TCC805x Linux SDK-Getting Started*". [6]

Note 2: Required packages to be installed may be different depending on the version of Ubuntu and Fedora. If the build does not work, the required package may not be installed.

3.1.2.2 Java Development Kit (JDK)

The AOSP master branch comes with a prebuilt version of OpenJDK, so additional installation is not required.

3.1.2.3 Key Packages

The AOSP master branch comes with a prebuilt version of GNU Make, so additional installation is not required.

- GNU Make 3.81 to 3.82 for previous versions
- Python 2.7.5
- Git 1.7 or higher

For more information, refer to <https://source.android.com/setup/build/requirements>

3.1.2.4 Toolchain Path

Bootloader

The AAS uses U-Boot as the bootloader. Refer to Chapter 4.1.2.2.1 Set Configuration of Toolchain for U-Boot

Kernel

Refer to Chapter 4.1.2.3.3 Kernel Configuration for Toolchain of Kernel.

3.1.3 Test Environment of Telechips

- Tested on Fedora
 - Fedora-20 (Linux-3.19.8-100.fc20.x86_64-x86_64++debug)
 - GNU Make 3.82 or higher
 - Python 2.7.5
 - Git 2.25.1 or higher
 - OpenJDK 9 or higher
- Tested on Ubuntu
 - Ubuntu20.04.5-LTS
 - GNU Make 3.82 or higher
 - Python 3.8.10
 - Git 2.25.1 or higher
 - OpenJDK 9 or higher

4 BUILD GUIDE

4.1 TCC805x EVB (64-bit SDK)

If you build the default SDK without modification, the followings are supported:

- Default display: DisplayPort (DP)
- Default storage: eMMC

4.1.1 Download SDK

Download SDK from **TCC805x_Android10_IVI_3.0.0** in the following Telechips Android Git server.

```
$ mkdir mydroid
$ cd mydroid

$ repo init -u ssh://android.telechips.com/android_avn/android/platform/manifest.git -
b TCC805x_Android10_IVI_3.0.0

$ repo sync
```

Note: It is recommended to use the **c** option for repo sync to reduce the time for downloading SDK.

The **c** option means that only the source of the revision branch specified in the manifest file is synchronized. (Usage: **repo sync -c -j8**)

To reduce the downloading time and SDK size, **clone-depth = "1"** for a specific AOSP git in the manifest file is specified as follows. The manifest file, "default.xml", is in the **.repo/manifests** path.

```
<project clone-depth="1" groups="pdk-fs" name="platform/prebuilts/abi-dumps/ndk" path="prebuilts/abi-dumps/ndk" />
<project clone-depth="1" groups="pdk-fs" name="platform/prebuilts/abi-dumps/vndk" path="prebuilts/abi-dumps/vndk" />
<project clone-depth="1" groups="pdk-fs" name="platform/prebuilts/android-emulator" path="prebuilts/android-emulator" />
<project clone-depth="1" groups="pdk" name="platform/prebuilts/build-tools" path="prebuilts/build-tools" />
<project clone-depth="1" groups="pdk" name="platform/prebuilts/checkcolor" path="prebuilts/checkcolor" />
<project clone-depth="1" groups="pdk" name="platform/prebuilts/checkstyle" path="prebuilts/checkstyle" />
<project clone-depth="1" groups="pdk" name="platform/prebuilts/clang-tools" path="prebuilts/clang-tools" />
```

The **clone-depth** option means that a shallow clone is created with a history truncated to the number of commits specified by the **depth** option.

However, the shallow clone cannot be pushed to the server. If you need to push all the sources to the server or push the commit, you should use the following command. You can get all commit historys for the shallow clone by using the following command.

Note: You can check the remote-name of the specific git by using the **git remote -v** command.

```
$ git fetch --unshallow [remote-name]
```

To apply the **git fetch** command above to all gits with **clone-depth** in the SDK, you should use the following command.

```
$ repo forall -c "git fetch --unshallow [remote-name]"
```


4.1.2 Compile and Build Android Framework

4.1.2.1 Set Up Compiling Environment

TARGET_PRODUCT must be set correctly for board configuration before compiling. Execute the commands below.

Note 1: The content in this chapter can be automatically executed by "build-all-script.sh" in AOSP root folder. Consider the following examples as the described steps for build. You can follow the steps below for manual build. However, U-Boot build toolchain must be set in PATH. Refer to Chapter 4.1.2.2.1.

Note 2: There is a space between dot (.) and **build/envsetup.sh** to execute shell scripiter.

Note 3: The "build-all-script.sh" is for only Telechips' EVB and Telechips does not support customer request for the script file. You should modify the script file depending on your development environment.

```
$ cd ~/mydroid/android
$ . build/envsetup.sh

$ lunch

You're building on Linux

Lunch menu... pick a combo:
 1. aosp_arm-eng
 2. aosp_arm64-eng
 3. aosp_blueline-userdebug
 4. aosp_bonito-userdebug
 5. aosp_car_arm-userdebug
 6. aosp_car_arm64-userdebug
 7. aosp_car_x86-userdebug
 8. aosp_car_x86_64-userdebug
 9. aosp_crosshatch-userdebug
10. aosp_marlin-userdebug
11. aosp_sailfish-userdebug
12. aosp_sargo-userdebug
13. aosp_taimen-userdebug
14. aosp_walleye-userdebug
15. aosp_walleye_test-userdebug
16. aosp_x86-eng
17. aosp_x86_64-eng
18. beagle_x15-userdebug
19. car_tcc803x-eng
20. car_tcc803x_arm64-eng
21. car_tcc803xp_arm64-eng
22. car_tcc8050_arm64-eng
23. car_tcc8059_arm64-eng
24. fuchsia_arm64-eng
25. fuchsia_x86_64-eng
26. full_a80-userdebug
27. full_elm-userdebug
28. full_geekbox-userdebug
29. full_pc-userdebug
30. full_pc_x86_64-userdebug
31. hikey-userdebug
32. hikey64_only-userdebug
33. hikey960-userdebug
34. hikey960_tv-userdebug
35. hikey_tv-userdebug
36. m_e_arm-userdebug
37. mini_a80-userdebug
38. mini_elm-userdebug
39. mini_emulator_arm64-userdebug
40. mini_emulator_x86-userdebug
41. mini_emulator_x86_64-userdebug
42. mini_geekbox-userdebug
43. mini_pc-userdebug
44. mini_pc_x86_64-userdebug
45. poplar-eng
46. poplar-user
```

```

47. poplar-userdebug
48. qemu_trusty_arm64-userdebug
49. tcc8990-eng
50. tcc8995-eng
51. tcc9010-eng
52. uml-userdebug

```

Which would you like? [aosp_arm-eng] 22

```

=====
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=10
TARGET_PRODUCT=car_tcc8050_arm64
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
TARGET_ARCH=arm64
TARGET_ARCH_VARIANT=armv8-a
TARGET_CPU_VARIANT=cortex-a72
TARGET_2ND_ARCH=arm
TARGET_2ND_ARCH_VARIANT=armv8-a
TARGET_2ND_CPU_VARIANT=cortex-a53
HOST_ARCH=x86_64
HOST_2ND_ARCH=x86
HOST_OS=linux
HOST_OS_EXTRA=Linux-3.19.8-100.fc20.x86_64-x86_64-Fedora-20-(Heisenbug)
HOST_CROSS_OS=windows
HOST_CROSS_ARCH=x86
HOST_CROSS_2ND_ARCH=x86_64
HOST_BUILD_TYPE=release
BUILD_ID=QTR1.210629.001
OUT_DIR=out
=====

```

For example, if you want to use TCC805x EVB or TCC803xPE EVB of 64-bit SDK, select **car_tcc803xp_arm64-eng** with input **21**, **car_tcc8050_arm64-eng** with input **22**, or **car_tcc8059_arm64-eng** with input **23**.

Note: If you want to set the mode to **user** mode, use **choosevariant** command. This selection should be done before compiling kernel because it affects the kernel configuration of architecture.

4.1.2.2 Compile Bootloader

You must compile bootloader and kernel first and then framework respectively.

- Bootloader path: bootable/bootloader/u-boot

4.1.2.2.1 Set Configuration of Toolchain for U-Boot

Before building a bootloader, you should download toolchain from the following website.

- <https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-a/downloads/9-2-2019-12>

The information of toolchain to compile U-Boot is `gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu.tar.xz` and `gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabi.tar.xz`.

You can build after all prerequisites are set. If you select a configuration according to CPU board, it is set accordingly. Set toolchain path and configuration of U-Boot as below

```
$ cd gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabi/bin
$ export PATH=$PWD:$PATH
$ cd gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu/bin
$ export PATH=$PWD:$PATH
$ echo $PATH // you can check toolchain path is applied in $PATH
$ cd bootable/bootloader/u-boot
$ source scripts/envsetup.sh
[1] Telechips Android Platform
[2] Telechips Android Platform (64bit)
[3] Telechips Linux Platform
[4] Telechips Linux Platform (64bit)
Select platform (default 4): 2

[1] TCC805x
[2] TCC803x
Select SoC family (default 1): 1

[1] TCC8050 EVB 1.0
[2] TCC8053 EVB 1.0
[3] TCC8050 EVB 0.1
[4] TCC8053 EVB 0.1
[5] TCC8059 EVB 0.1
Select board (default 1): 1

[1] Maincore
[2] Subcore
Select core (default 1): 1

### U-Boot Environment Setup Complete ###

- ARCH=arm64
- CROSS_COMPILE=aarch64-none-linux-gnu-
- DEVICE_TREE=tcc8050-evb_sv1.0
```

4.1.2.2.2 Build U-Boot and Make Rom-file

To build the bootloader (U-Boot) for EVB of 64-bit SDK, use the following default configuration files and execute the **make** command depending on the chipset.

- TCC805x : tcc805x_android_10_defconfig
- TCC803xPE : tcc803x_android_10_defconfig

```
$ make tcc805x_android_10_defconfig
```

Note 1: For **UFS** boot mode, use **tcc805x_android_10_ufs_defconfig**.

Note 2: To configure the boot mode to single mode only for TCC803xPE, remove the build option as follows:

```
$ make menuconfig
  ARM architecture ->
    Telechips architecture specifics --->
      [ ] Boot subcore kernel by maincore
```

Do compile. You can get "ca72_bl3.rom" in bootable/bootloader/u-boot.

```
$ make -j8
```


4.1.2.3 Compile Linux Kernel

After compiling kernel, you should compile "boot.img" and "system.img".
If you modify the kernel code, the "boot.img" must be compiled again.

4.1.2.3.1 Set Kernel as Default

Lunch menu must be selected before setting Linux kernel configuration because the lunch menu affects the kernel configuration of architecture in Android 10 SDK.

4.1.2.3.2 Kernel Configuration – Device Tree

The board configuration files are all in **kernel/arch/arm64/boot/dts** path. The ".dts" file includes a specific configuration depending on LCD panel.

Also, detailed configuration files are listed in **kernel/arch/arm64/boot/dts/tcc** path.

For TCC8050_53, "tcc8050-android-lpd4x322_sv1.0.dts" and "tcc8050-android-ivi.dtsi" are used for kernel device configuration. After compiling kernel, these dt-files produce binaries as "tcc8050-android-lpd4x322_sv1.0.dtb" in two paths

- kernel/arch/arm64/boot/dts/tcc
- out/target/product/car_tcc8050_arm64

For TCC8059, "tcc8059-android-lpd4x321_sv0.1.dts" and "tcc8059-android-ivi.dtsi" are used for kernel device configuration. After compiling kernel, these dt-files produce binaries as "tcc8059-android-lpd4x321_sv0.1.dtb" in two paths

- kernel/arch/arm64/boot/dts/tcc
- out/target/product/car_tcc8059_arm64

For TCC803xPE, "tcc803xp-android-lpd4321_sv1.0.dts" and "tcc803xp-android-ivi.dtsi" are used for kernel device configuration. After compiling kernel, these dt-files produce binaries as "tcc803xp-android-lpd4321_sv1.0.dtb" in two paths.

- kernel/arch/arm64/boot/dts/tcc
- out/target/product/car_tcc803xp_arm64

4.1.2.3.3 Kernel Configuration

To set the kernel configuration, use the following default configuration files in **kernel/arch/arm64/configs** depending on the chipset.

- TCC805x : tcc805x_android_10_ivi_defconfig
- TCC803xPE : tcc803x_android_10_ivi_defconfig

Note: For **UFS** boot mode, use **tcc805x_android_10_ivi_ufs_defconfig**.

Move to Linux kernel directory and execute the command below to set the appropriate default kernel configuration according to TCC805x or TCC803xPE.

The **".config"** file is created in the kernel directory.

You must unset predefined environment variable before kernel configuration.

```
$ cd ~/mydroid/android/kernel
$ unset ARCH
$ unset CROSS_COMPILE
$ make ARCH=arm64 tcc805x_android_10_ivi_defconfig
```

To configure the boot mode to single boot mode, remove build options as follows (for both TCC805x and TCC803xPE):

```
$ make ARCH=arm64 menuconfig
Device Drivers --->
  SOC (System On Chip) specific Drivers --->
    *- Telechips SOC drivers support --->
      [ ] Reserve PMAP memory for subcore kernel
```

Note: To enable the **GPU Virtualization**, refer to the following (for TCC805x only):

- To use the GPU in the main core and the sub-core, you should set the **GPU Virtualization config** of main core as follows:

```
$ make ARCH=arm64 menuconfig
Device Drivers ->

-> Graphics support
<*> PowerVR GPU
  [*] PowerVR Virtualisation
```

- To set the **GPU Virtualization config** of sub-core, refer to configurations for building sub-core in Chapter 4.1.4 Build Sub-core by Autolinux.

4.1.2.3.4 Compile Kernel

Execute **make** in kernel directory.

```
$ make ARCH=arm64 CLANG_TRIPLE=aarch64-linux-gnu- CROSS_COMPILE=aarch64-linux-androidkernel-
CC=../prebuilts/clang/host/linux-x86/clang-r353983c/bin/clang -j8
```

4.1.2.3.5 Output

After compiling kernel is completed, check that the zImage, Image, and "xxx.dtb" are generated.

Each device has a different Device Tree Blob (DTB) file to boot in **arch/arm64/boot/dts/tcc** path.

- For TCC8050, you must use "tcc8050-android-lpd4x322_sv1.0.dtb".
- For TCC8059, you must use "tcc8059-android-lpd4x321_sv0.1.dtb".
- For TCC803xPE, you must use "tcc803xp-android-lpd4321_sv1.0.dtb"

4.1.2.4 Compile Framework

Note 1: For **UFS** boot mode, modify `TCC_UFS_SUPPORT := false` to `true` in "device.mk" in device/telechips/car_tcc8050_arm64.

Note 2: To enable the **GPU Virtualization**, refer to the following:

- To use the GPU in the main-core and sub-core, you should modify `TCC_GPU_VZ ?= false` to `true` in "device.mk" in device/telechips/car_tcc8050_arm64.

Execute **make** command, then you can build Android framework. It takes time depending on user environment.

```
$ cd ~/mydroid/maincore
$ make -j8
```

You should check the detailed log below. This log can be seen when you select **car_tcc805x**.

```
=====
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=10
TARGET_PRODUCT=car_tcc8050_arm64
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
TARGET_ARCH=arm64
TARGET_ARCH_VARIANT=armv8-a
TARGET_CPU_VARIANT=cortex-a72
TARGET_2ND_ARCH=arm
TARGET_2ND_ARCH_VARIANT=armv8-a
TARGET_2ND_CPU_VARIANT=cortex-a53
HOST_ARCH=x86_64
HOST_2ND_ARCH=x86
HOST_OS=linux
HOST_OS_EXTRA=Linux-3.19.8-100.fc20.x86_64-x86_64-Fedora-20-(Heisenbug)
HOST_CROSS_OS=windows
HOST_CROSS_ARCH=x86
HOST_CROSS_2ND_ARCH=x86_64
HOST_BUILD_TYPE=release
BUILD_ID= QTR1.210629.001
OUT_DIR=out
=====
```

4.1.3 Boot-firmware

Firmware essential for the booting process is called boot-firmware.

The boot-firmware repository contains several prebuilt firmware images such as HSM, Trusted Firmware-A, Storage Core Firmware, and binary files in /prebuilt directory.

The boot-firmware configurations of TCC805x and TCC803xPE are different.

■ TCC805x prebuilt images

```
$ cd ~/mydroid/maincore/bootable/bootloader/u-boot/boot-firmware/prebuilt
$ ls
bconf.bin      ca72_bl2.rom    hsm.cs.bin      r5_fw_TCC8050.rom  snor_legacy      tcc8059_snor.rom
ca53_bl1.rom   dram_params.bin mcert.bin       r5_fw_TCC8059.rom  tcc8050_snor.cs.rom  ufs
ca53_bl2.rom   fwdn.rom        optee.rom       r5_sub_fw.rom      tcc8050_snor.rom
ca72_bl1.rom   hsm.bin         r5_bl1-snor.rom scfw.rom           tcc8059_snor.cs.rom
```

■ TCC803xPE prebuilt images

```
$ cd ~/mydroid/maincore/bootable/bootloader/u-boot/boot-firmware-tcc803x/prebuilt
$ ls
ap.rom      dram_params_803x.bin  hsm.bin      micom.cs.bin  r5_sub_fw.bin
bconf.bin   dram_params.bin       keypackages.bin micom.xpe.bin  tcc803xpe_snor_boot.rom
bl2-2.rom   fwdn.rom              micom-bl1.rom optee.rom     tcc803x_snor_boot.rom
```


4.1.4 Build Sub-core by Autolinux

Note 1: The packages required on Linux Host Machine to build the sub-core, refer to Chapter 4.4.3 List of Packages Required for Host Development System to Use Yocto Project in "*TCC805x Linux SDK-Getting Started*". [6]

Note 2: To build the sub-core by using Autolinux, you should install python3 in Linux Host Machine.

Note 3: For details on building the sub-core, refer to Chapter 4.6.2.3 Autolinux Usage in "*TCC805x Linux SDK-Getting Started*". [6]

For Autolinux usage, the python3 should be installed in /bin. If python3 does not exist in /bin, create a symbolic link as follows:

```
sudo ln -s [your python3 path] /bin/python3
```

4.1.4.1 Autolinux Usage

```
~/mydroid/subcore$ ./autolinux --help
```

```
usage: autolinux [-h] -c COMMAND [COMMAND ...] [-V] [-s SDK] [-m MACHINE]
               [-x MANIFEST] [-f FEATURES [FEATURES ...]]
               [-sf SUBFEATURES [SUBFEATURES ...]]
```

Linux SDK Configurations

optional arguments:

-h, --help show this help message and exit

General Command:

```
-c COMMAND [COMMAND ...], --command COMMAND [COMMAND ...]
    General Command to use autolinux(required option)
    -commands-
    update: sync to recipes matched manifest.xml, Be careful local changes will be lost
    configure [option] [name]: setup the newer build environment
        conf_opt args : additionally set conf opts
                        refers to 'Configuration Options'
                        Ex) autolinux -c configure -m tcc8050-main
        save [name] : save autolinux.config file to .config/
        load [name] : load configure file from .config/
    clean [option]: move current built files to recycle directory. recycle dir:build/delete
        old : delete current built files and recycle directory.
        all : delete total build directory
    build [option]: build SDK Image, choose the name of the image to build
        'args'      : extra command strings
                        Ex) autolinux -c build 'linux-telechips -C compile'
        sub 'args' : building on sub-core, available only the with-subcore option is enabled
                        Ex) autolinux -c build sub 'linux-telechips -C compile'
    devtool 'args' : devtool package to develop easily
                        Ex) autolinux -c devtool 'modify linux-telechips'
    make_fai : make SDdata.fai for fwdn
    info : show information of current configuration
    modify option: modify specific configure at current set-up
        feature      : modify features
        sub-feature  : modify subcore features
        image        : modify image
```

-V, --version print version and exit

Configuration Options:

```
-s SDK, --sdk SDK enter the sdk to download
                    sdk lists
                    subcore :['tcc805x_android_ivi'] -m MACHINE, --machine MACHINE
                    enter the machine to build
-x MANIFEST, --manifest MANIFEST
                    enter the manifest to build
-f FEATURES [FEATURES ...], --features FEATURES [FEATURES ...]
                    enter the feature list
                    Ex) -f with-subcore gpu-vz meta-update
-sf SUBFEATURES [SUBFEATURES ...], --sub-features SUBFEATURES [SUBFEATURES ...]
                    enter the sub-feature list
                    Ex) -sf rvc gpu-vz meta-update
```

4.1.4.2 Build Sub-core with Configurations

Note 1:

- To enable the **GPU Virtualization**, refer to the following:
 - You must choose the `gpu-vz` feature to use GPU in the sub-core.

Note 2:

- To enable the **cluster application**, refer to the followings:
 - If you choose the `gpu-vz` feature in the sub-core, the `cluster` feature is enabled in the sub-core's feature select step.
 - To run the cluster application in the sub-core, you must choose the `gpu-vz` feature and `cluster` feature.
- Configuration for building sub-core

```
~/mydroid/subcore$ ./autolinux -c configure

The command is configure or Add configuration options(sdk,machine,manifest)
Configure Start

Choose a Machine to build
1.tcc8031p-sub
2.tcc8034p-sub
3.tcc8050-sub
4.tcc8053-sub
5.tcc8059-sub
Choose Machine(1-5): 3
Choose the Features at tcc8050-sub
1. ufs                Changed Boot Storage to UFS(supported : tcc8050, tcc8053)
* 2. meta-micom       Enable Micom
* 3. meta-update       Enable Update
* 4. rvc               Enable Rear Camera
5. gpu-vz             GPU Virtualization
* 6. early-camera      Enable early camera at tcc803x
* 7. kernel-4.14       Set Kerenl version to 4.14
8. meta-str           Enable Suspend To Ram(supported:tcc8050, tcc8053, tcc8059)
0.Exit

Choose Features enable/disable (1-8): 0
Read configuration from autolinux.config

=====
SDK=tcc805x_android_ivi
MANIFEST=tcc805x_android_subcore.xml
DATE=up-to-date
MACHINE=tcc8050-sub
VERSION=release
FEATURES=meta-micom,meta-update,rvc,early-camera,kernel-4.14
=====

~/mydroid/subcore$ ./autolinux -c build
```

4.1.4.3 Prebuilt Sub-core images without Building sub-core

You can use prebuilt sub-core images without building sub-core. Sub-core images are in the following paths:

- TCC8050
 - For using a prebuilt sub-core image with configuring the `gpu-vz` feature and `cluster` feature
 - `device/telechips/car_tcc8050_arm64/subcore_cluster`
 - For using a prebuilt sub-core image without configuring the `gpu-vz` feature and `cluster` feature
 - `device/telechips/car_tcc8050_arm64/subcore`
- TCC8059
 - For using a prebuilt sub-core image with configuring the `gpu-vz` feature and `cluster` feature
 - `device/telechips/car_tcc8059_arm64/subcore_cluster`
 - For using a prebuilt sub-core image without configuring the `gpu-vz` feature and `cluster` feature
 - `device/telechips/car_tcc8059_arm64/subcore`
- TCC803xPE : `device/telechips/car_tcc803xp/subcore`
 - TCC803xPE does not support a `gpu-vz` feature and `cluster` feature in sub-core.

Note: Sub-core image is required to be built by using *autolinux* in UFS boot mode, and prebuilt image was built in eMMC boot mode. Therefore, above images are not required when you build the sub-core images in UFS boot mode.

4.1.5 Make SD Data

4.1.5.1 SD Data Partition Information for TCC805x EVB (64-bit SDK)

Partition information of SD Data is shown in the table below.

Table 4.1 Partition Information of SD Data for TCC805x EVB

FWDN Partition	Size (KB)	Label Name	Block Device	Description	File Name
Partition 1	2048	bl3_ca72_a	mmcblk0p1	Main Core Boot Loader A	ca72_bl3.rom
Partition 2	2048	bl3_ca72_b	mmcblk0p2	Main Core Boot Loader B	ca72_bl3.rom
Partition 3	2048	bl3_ca53_a	mmcblk0p3	Sub Core Boot Loader A	ca53_bl3.rom
Partition 4	2048	bl3_ca53_b	mmcblk0p4	Sub Core Boot Loader B	ca53_bl3.rom
Partition 5	36864	boot_a	mmcblk0p5	Kernel image A	boot.img
Partition 6	36864	boot_b	mmcblk0p6	Kernel image B	boot.img
Partition 7	4124672	super	mmcblk0p7	Resizable image (system.img + vendor.img)	super.img
Partition 8	8192	dtbo_a	mmcblk0p8	Device tree Overlay A	dtbo.img
Partition 9	8192	dtbo_b	mmcblk0p9	Device tree Overlay B	dtbo.img
Partition 10	153600	cache	mmcblk0p10	Cache partition	cache.img
Partition 11	1024	env	mmcblk0p11	Environment	-
Partition 12	5120	splash	mmcblk0p12	splash partition	-
Partition 13	1024	misc	mmcblk0p13	misc partition	-
Partition 14	1024	subcore_misc	mmcblk0p14	Sub-core misc partition	-
Partition 15	1024	tcc	mmcblk0p15	tcc partition	-
Partition 16	8192	sest	mmcblk0p16	sest partition	-
Partition 17	40960	subcore_boot_a	mmcblk0p17	Sub-core Kernel Image	tc-boot-tcc8050-sub.img
Partition 18	40960	subcore_boot_b	mmcblk0p18	Sub-core Kernel Image	tc-boot-tcc8050-sub.img
Partition 19	409600	subcore_root_a	mmcblk0p19	Sub-core Device tree	telechips-subcore-image-tcc8050-sub.ext4
Partition 20	409600	subcore_root_b	mmcblk0p20	Sub-core Device tree	telechips-subcore-image-tcc8050-sub.ext4
Partition 21	1024	subcore_dtb_a	mmcblk0p21	Sub-core root file system	tcc8050-linux-subcore_sv1.0-tcc8050-sub.dtb
Partition 22	1024	subcore_dtb_b	mmcblk0p22	Sub-core root file system	tcc8050-linux-subcore_sv1.0-tcc8050-sub.dtb
Partition 23	1024	vbmeta_a	mmcblk0p23	Hashtree used for Android Verified Boot (AVB)	vbmeta.img
Partition 24	1024	vbmeta_b	mmcblk0p24	Hashtree used for Android Verified Boot (AVB)	vbmeta.img
Partition 25	8192	tamper_evidence	mmcblk0p25	tamper_evidence partition	-
Partition 26	16384	metadata	mmcblk0p26	metadata partition	-
Partition 27	1000	subcore_env	mmcblk0p27	Sub-core Environment	-
Partition 28	40000	subcore_splash	mmcblk0p28	Sub-core splash partition	splash_1920x720x32.img
Partition 29	0	userdata	mmcblk0p29	User storage	-

Note: Partitions (vbmeta, tamper_evidence, and metadata) are used for Android Verified Boot (AVB).

- For more details, refer to <https://android.googlesource.com/platform/external/avb/+master/README.md>.

4.1.5.2 Partition List File

Partition List File is a file that contains the partition information of the ".fai" file to be created.

When using GPT, the sum of GPT partition size must be less than or equal to the value that subtracts 34 KB from storage size because 34 KB is used as a space to construct GPT Header and GPT Entry.

However, if the last partition size is set to 0, the remaining size is automatically applied.

GPT partition information is as follows:

- TCC8050/TCC8053 : device/telechips/car_tcc8050_arm64/gpt_partition_for_arm64*.list.
- TCC8059 : device/telechips/car_tcc8059_arm64/gpt_partition_for_arm64*.list.
- TCC803xPE : device/telechips/car_tcc803xp_arm64/gpt_partition_for_arm64*.list.

The gpt_partition_for_arm64_dp.list is copied to bootable/bootloader/u-boot/boot-firmware/tools/mkting by using **make** command.

Because the SDK supports dynamic partition by default, the gpt_partition_for_arm64_dp.list must be used.

For the details, refer to "TCCxxx Android SDK-User Guide for Dynamic Partitions." [7]

- Format required for creating GPT Partition List File is as follows:

<Partition Name><Separator":"><Partition Size><Separator "@"><File Name>
--

■ Example

```
b13_ca72_a:2048k@../../../../ca72_b13.rom
b13_ca72_b:2048k@../../../../ca72_b13.rom
b13_ca53_a:2048k@../../../../ca53_b13.rom
b13_ca53_b:2048k@../../../../ca53_b13.rom
boot_a:36864k@../../../../out/target/product/car_tcc8050_arm64/boot.img
boot_b:36864k@../../../../out/target/product/car_tcc8050_arm64/boot.img
super:4124672k@../../../../out/target/product/car_tcc8050_arm64/super.img
dtbo_a:8192k@../../../../out/target/product/car_tcc8050_arm64/dtbo.img
dtbo_b:8192k@../../../../out/target/product/car_tcc8050_arm64/dtbo.img
cache:153600k@../../../../out/target/product/car_tcc8050_arm64/cache.img
env:1024k@
splash:5120k@
misc:1024k@
subcore_misc:1024k@
tcc:1024k@
sest:8192k@
subcore_boot_a:40960k@../../../../out/target/product/car_tcc8050_arm64/tc-boot-tcc8050-sub.img
subcore_boot_b:40960k@../../../../out/target/product/car_tcc8050_arm64/tc-boot-tcc8050-sub.img
subcore_root_a:409600k@../../../../out/target/product/car_tcc8050_arm64/telechips-subcore-image-tcc8050-sub.ext4
subcore_root_b:409600k@../../../../out/target/product/car_tcc8050_arm64/telechips-subcore-image-tcc8050-sub.ext4
subcore_dtb_a:1024k@../../../../out/target/product/car_tcc8050_arm64/tcc8050-linux-subcore_sv1.0-tcc8050-sub.dtb
subcore_dtb_b:1024k@../../../../out/target/product/car_tcc8050_arm64/tcc8050-linux-subcore_sv1.0-tcc8050-sub.dtb
vbmeta_a:1024k@../../../../out/target/product/car_tcc8050_arm64/vbmeta.img
vbmeta_b:1024k@../../../../out/target/product/car_tcc8050_arm64/vbmeta.img
tamper_evidence:8192k@
metadata:16384k@
subcore_env:1M@
subcore_splash:40M@../../../../drivers/camera/splash/splash_1920x720x32.img
userdata:0k@
```

4.1.5.3 Use mktcimg

Before making SD Data, you should build an image in sub-core.
For building the sub-core, refer to Chapter 4.1.4 Build Sub-core by Autolinux.

After building the the sub-core image, copy the built sub-core image to the path defined in `gpt_partition_for_arm64.list` as follows:

```
$ cd ~/mydroid/subcore/build/tcc8050-sub/tmp/deploy/images/tcc8050-sub
$ cp ca53_b13.rom ../../../../../../maincore/bootable/bootloader/u-boot
$ cp tc-boot-tcc8050-sub.img ../../../../../../maincore/out/target/product/car_tcc8050_arm64
$ cp telechips-subcore-image-tcc8050-sub.ext4 ../../../../../../maincore/out/target/product/car_tcc8050_arm64
$ cp tcc8050-linux-subcore_sv1.0-tcc8050-sub.dtb ../../../../../../maincore/out/target/product/car_tcc8050_arm64
```

Note 1: If you use prebuilt sub-core image, copy the prebuilt sub-core image to the path defined in `gpt_partition_for_arm64.list` as follows:

```
$ cd ~/mydroid/maincore/device/telechips/car_tcc8050_arm64/subcore
$ cp ca53_b13.rom ~/mydroid/maincore/bootable/bootloader/u-boot
$ cp tc-boot-tcc8050-sub.img ~/mydroid/maincore/out/target/product/car_tcc8050_arm64
$ cp telechips-subcore-image-tcc8050-sub.ext4 ~/mydroid/maincore/out/target/product/car_tcc8050_arm64
$ cp tcc8050-linux-subcore_sv1.0-tcc8050-sub.dtb ~/mydroid/maincore/out/target/product/car_tcc8050_arm64
```

Note 2: If you build an image in single boot mode, which is boots only in the main core, delete the definitions of sub-core image in `gpt_partition_for_arm64.list`.

The definitions of the sub-core image are as follows:

```
bl3_ca53_a:2048k@../../../../ca53_b13.rom
bl3_ca53_b:2048k@../../../../ca53_b13.rom
subcore_misc:1024k@
subcore_boot_a:40960k@../../../../out/target/product/car_tcc8050_arm64/tc-boot-tcc8050-sub.img
subcore_boot_b:40960k@../../../../out/target/product/car_tcc8050_arm64/tc-boot-tcc8050-sub.img
subcore_root_a:409600k@../../../../out/target/product/car_tcc8050_arm64/telechips-subcore-image-tcc8050-sub.ext4
subcore_root_b:409600k@../../../../out/target/product/car_tcc8050_arm64/telechips-subcore-image-tcc8050-sub.ext4
subcore_dtb_a:1024k@../../../../out/target/product/car_tcc8050_arm64/tcc8050-linux-subcore_sv1.0-tcc8050-sub.dtb
subcore_dtb_b:1024k@../../../../out/target/product/car_tcc8050_arm64/tcc8050-linux-subcore_env:1M@
subcore_splash:40M@../../../../drivers/camera/splash/splash_1920x720x32.img
```

To make SD Data, use **mktcimg** in boot-firmware as follows:

```
In case of EMMC,
$ cd ~/mydroid/maincore/bootable/bootloader/u-boot/boot-firmware/tools/mktcimg
$ ./mktcimg --parttype gpt --storage_size 7818182656 --fplist gpt_partition_for_arm64.list --outfile SD_Data.fai --area_name "SD Data" --gptfile SD_Data.gpt

In case of UFS,
$ cd ~/mydroid/maincore/bootable/bootloader/u-boot/boot-firmware/tools/mktcimg
$ ./mktcimg --parttype gpt --storage_size 31998345216 --fplist gpt_partition_for_arm64.list --outfile SD_Data.fai --area_name "SD Data" --gptfile SD_Data.gpt --sector_size 4096
```

4.1.5.4 Make SNOR ROM Image

There are pre-built images as follows:

- TCC805x : "tcc8050_snor.cs.rom" and "tcc8059_snor.cs.rom" in boot-firmware/prebuilt directory
- TCC803xPE : "tcc803xp_snor_boot.rom" in boot-firmware-tcc803x/prebuilt directory

If you need to change the binary stored in the SNOR ROM, you must make new SNOR ROM image.

To make SNOR ROM image of TCC805x, use the *tcc805x_snor_mkimage* in boot-firmware as follows:

```
$ cd ~/mydroid/maincore/bootable/bootloader/u-boot/boot-firmware/tools/tcc805x_snor_mkimage/
$ ./tcc805x-snor-mkimage -i tcc8050.cs.cfg -o ../../tcc805x_snor.cs.rom
```

Note: To make a SNOR ROM image of TCC8059, use the *tcc8059.cs.cfg* instead of the *tcc8050.cs.cfg* in the above command.

To make SNOR ROM image of TCC803xPE, use the *tcc803x_snor_mkimage* in boot-firmware as follows:

```
$ cd ~/mydroid/maincore/bootable/bootloader/u-boot/boot-firmware-tcc803x/tools/d3s_snor_mkimage/
$ ./tcc803x-snor-mkimage-tools -c tcc803xpe.cfg -o ./tcc803x_snor_boot.rom
```

4.1.5.4.1 "mkimage.cfg" File

tcc8050.cs.cfg file lists parameters input to the *tcc805x-snor-mkimage*.

- **Example:** tcc8050.cs.cfg

```
## NOTE:
## When inserting annotation using character '#', do not insert word spaces or tabs in the first place
## of each line.

# Chip Revision (ES: 0, CS: 1)
REVISION=1

# SNOR Flash memory size(MByte)
SNOR_SIZE=4

# R5 UART Port number for debug
DEBUG_ENABLE=0
DEBUG_PORT=14

# Input files
MCERT_BIN=../../prebuilt/mcert.bin
HSM_BIN=../../prebuilt/hsm.cs.bin
R5BL1_BIN=../../prebuilt/r5_bl1-snor.rom
UPDATE_BIN=../../prebuilt/r5_sub_fw.rom
#MICOM_BIN=../../prebuilt/micom.bin
MICOM_BIN=../../prebuilt/r5_fw_TCC8050.rom
```

tcc803xpe.cfg file lists parameters input to the *tcc803x-snor-mkimage*.

■ **Example:** tcc803xpe.cfg

```
## NOTE:
## When inserting annotation using character '#', do not insert word spaces or tabs in the first place
of each line.

# REVISION 1:CS, 2:ES
REVISION=1

# UART Port configuration
UART_DEBUG_CFG=14

# SNOR Flash memory size(MByte)
SNOR_SIZE=4

# Input files
BL1_BIN=../../prebuilt/micom-bl1.rom
MICOM_BIN ../../prebuilt/micom.xpe.bin
UPDATER_BIN=../../prebuilt/r5_sub_fw.bin
HSM_ROM=../../prebuilt/hsm.bin
KEYPACKAGES_BIN=../../prebuilt/keypackages.bin
```

5 FIRMWARE DOWNLOADER (FWDN) GUIDE

This chapter describes how to download the AAS to board.

For more information on how to use the tools for Firmware Downloader, refer to the following documents. [5]

- **FWDN V8** : "TCCxxx Common-User Guide for Firmware Downloader V8"
- **mktcimg** : "TC Common-User Guide for mktcimg"

5.1 System Requirement

5.1.1 Software Requirement

Table 5.1 Software Requirement

Category	Description	Path	Note
VTC Driver	It is a driver for Windows PC used to download AAS firmware. VTC driver corresponding to the Windows version must be installed. It is installed only once on your PC.	vendor/telechips/tools/FWDN/out	You must use version 5.0.0.14 or higher. Refer to VTC Driver information file path: <ul style="list-style-type: none"> ■ vendor/telechips/tools/FWDN/out
FWDN	Tool for downloading AAS firmware	vendor/telechips/tools/FWDN/out	You must use version FWDN V8 or higher.

5.1.2 Hardware Requirement

Table 5.2 Hardware Requirement

Category	Description	Note
Demo Board	Supported chipsets are TCC805x and TCC803xPE	-
Power Adapter	12V power adapter is used to supply power to the board and LCD.	Recommended to use the power adapter (12V) provided by Telechips.
USB Cable	It is used to connect a demo board to PC.	Refer to Chapter 2.2.1 for connection of USB cable for each board.
UART	Serial port for debugging	Refer to Chapter 2.1.2 to check the port location of UART USB C-type for CA53/CA72/MCU.

To use the USB C-type above, check the driver first. If the driver is not installed, install or re-install UART driver. You can download the latest versions at the official homepage:

- <https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers>

The serial ports assigned to each core are as follows:



Note 1: After downloading the firmware and booting the board, you can check the debug log in the UART console.

Note 2: The partition layout uses the GUID Partition Table (GPT) format.

5.2 Install VTC Driver

Install USB driver (VTC Driver V5.0.0.14) for **FWDN V8** in the Windows PC.

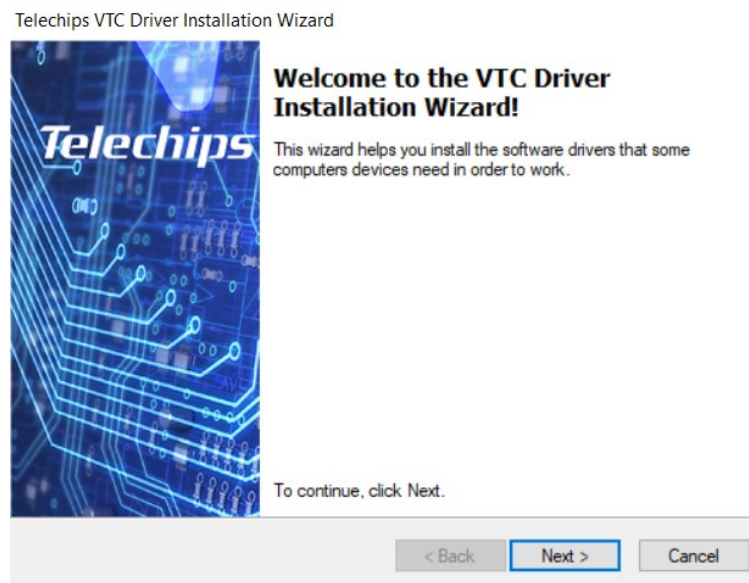


Figure 5.1 Installation Screen of VTC Driver

5.3 Firmware Download Sequence

Note 1: Bootloaders such as BL1 and firmware for each core are in the following paths:

- TCC805x : bootable/bootloader/u-boot/boot-firmware/prebuilts/
 - You can find download configuration of bootcode in bootable/bootloader/u-boot/boot-firmware/tcc805x_boot.json.
 - You can ignore tcc805x_bl3.json which is no longer used.
- TCC803xPE : bootable/bootloader/u-boot/boot-firmware-tcc803x/prebuilts/
 - You can find download configuration of bootcode in bootable/bootloader/u-boot/boot-firmware-tcc803x/tcc803xpe_boot.json.

Note 2: Refer to a separate document for more information on *mktcimg* and *FWDN V8*. [5]

1. Set the boot mode switch to USB boot mode.
2. Connect 12V power adapter to board.
3. Download the binary image in bootable/bootloader/u-boot/boot-firmware/tools/mktcimg/SD_Data.fai
4. Connect USB cable to board.
5. Run *FWDN V8*.

■ TCC805x eMMC

```
fwdn --fwdn Z:\projects\auto\8050_onDev\maincore\bootable\bootloader\u-boot\boot-
firmware\tcc805x_fwdn.cs.json
fwdn --low-format --storage emmc
fwdn --low-format --storage snor
fwdn -w Z:\projects\auto\8050_onDev\maincore\bootable\bootloader\u-boot\boot-firmware\tcc805x_boot.cs.json
fwdn -w Z:\projects\auto\8050_onDev\maincore\bootable\bootloader\u-boot\boot-
firmware\tools\mktcimg\SD_Data.fai --storage emmc --area user
fwdn --write Z:\projects\auto\8050_onDev\maincore\bootable\bootloader\u-boot\boot-
firmware\tools\tcc805x_snor_mkimage\tcc805x_snor.cs.rom --area die1 --storage snor
```

■ TCC8050/TCC8053 UFS

```
fwdn --fwdn Z:\projects\auto\8050_onDev\maincore\bootable\bootloader\u-boot\boot-
firmware\tcc805x_fwdn.cs.json
fwdn --low-format --storage ufs
fwdn --low-format --storage snor
fwdn -w Z:\projects\auto\8050_onDev\maincore\bootable\bootloader\u-boot\boot-
firmware\tcc805x_ufs_boot.json
fwdn -w Z:\projects\auto\8050_onDev\maincore\bootable\bootloader\u-boot\boot-
firmware\tools\mktcimg\SD_Data.fai --storage ufs --area user --sector-size 4096
fwdn --write Z:\projects\auto\8050_onDev\maincore\bootable\bootloader\u-boot\boot-
firmware\tools\tcc805x_snor_mkimage\tcc805x_snor.cs.rom --area die1 --storage snor
```

■ TCC803xPE eMMC

```
fwdn --fwdn Z:\projects\auto\803xpe_onDev\maincore\bootable\bootloader\u-boot\boot-firmware-
tcc803x\tcc803xpe_fwdn.json
fwdn --low-format --storage emmc
fwdn --low-format --storage snor
fwdn -w Z:\projects\auto\803xpe_onDev\maincore\bootable\bootloader\u-boot\boot-firmware-
tcc803x\tcc803xpe_boot.json
fwdn -w Z:\projects\auto\803xpe_onDev\maincore\bootable\bootloader\u-boot\boot-firmware-
tcc803x\tools\mktcimg\SD_Data.fai --storage emmc --area user
fwdn --write Z:\projects\auto\8050_onDev\maincore\bootable\bootloader\u-boot\boot-firmware-
tcc803x\tools\d3s_snor_mkimage\tcc803x_snor_boot.rom --area die1 --storage snor
```

6 PARTITION UPDATE

6.1 Fastboot Flash Memory Update

6.1.1 What is Fastboot

Fastboot is a function of Android Debug Bridge (ADB) used to update partition contents of flash memory on board in bootloader. Functions such as Flash, Erase, and Reboot can be performed according to each partition region of flash memory. The ADB USB driver must be installed to use Fastboot.

All partitions excluding logical partitions (such as system and vendor partitions) can be partially updated by using Fastboot in bootloader.

This SDK supports dynamic partitions, so logical partitions (such as system and vendor partitions) can be partially updated by using fastbootd.

To flash logical partitions by using fastbootd, you should unlock the bootloader. However, you cannot unlock the bootloader without using OPTEE and modifying the relevant U-Boot configuration. If you want to use OPTEE and modify the relevant U-Boot configuration, you must obtain a commercial license for OPTEE. If you want to obtain the license, contact Telechips. [1]

Note: Use Fastboot when partial update is required after downloading an image by using *FWDN V8*

6.1.2 Fastboot Mode

6.1.2.1 Fastboot in Bootloader (fastboot)

To enter Fastboot mode, bootloader must be written by using *FWDN V8*. If bootloader is already written, follow the procedure below.

1. Set the boot mode switch to boot mode (Refer to Chapter 2.2.2, Chapter 2.2.3, Chapter 2.2.4, or Chapter 2.2.5).
2. Connect the board to PC by UART serial cable and open UART console.
3. Supply power and select **FASTBOOT** of U-Boot boot menu in console as Figure 6.1.
4. (Alternatively, enter **reboot bootloader** in console when booting is completed.)
5. Fastboot mode is shown as follows:

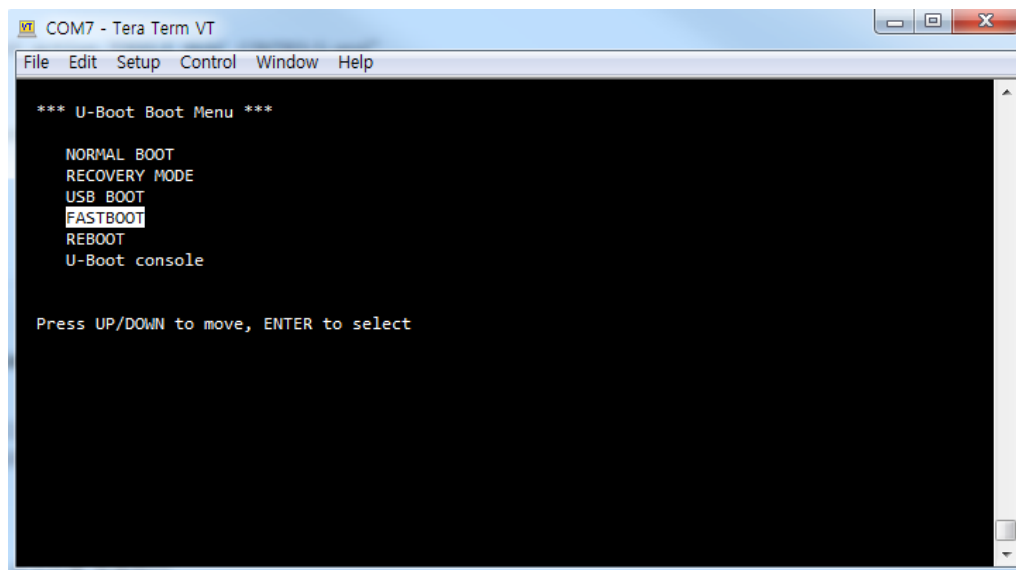
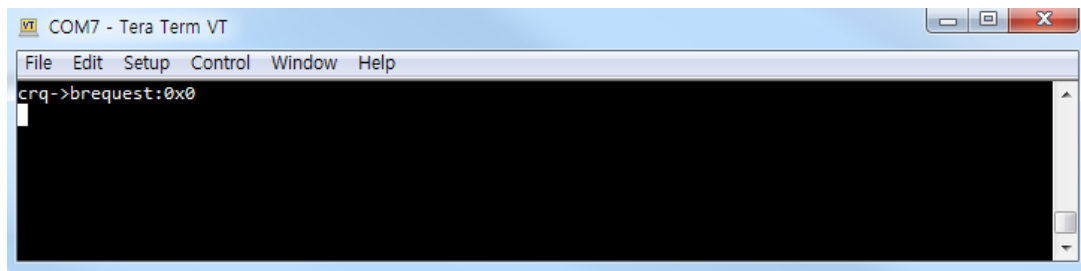


Figure 6.1 Run Fastboot

6. Connect USB cable to USB2.0_DRD (USB OTG) port or USB3.0_Host port.
The device number of USB2.0_DRD port is 1 and the device number of USB3.0_Host port is 0.
7. If Android USB driver is not installed on PC, install the driver first.
8. Once Fastboot is processed, processing commands appear as Figure 6.2.

**Figure 6.2 Connected Fastboot**

6.1.2.2 Fastboot in Recovery (fastbootd)

To enter fastbootd mode, bootloader must be written by using *FWDN V8*. If bootloader is already written, follow the procedure below.

1. Set the boot mode switch to boot mode (Refer to Chapter 2.2.2, Chapter 2.2.3, Chapter 2.2.4, or Chapter 2.2.5).
2. Connect the board to PC by UART serial cable and open UART console.
3. Supply power and input **reboot fastboot** in console.
4. Connect USB cable to USB2.0_DRD (USB OTG) port or USB3.0_Host port.
5. If Android USB driver is not installed on PC, install the driver first.

6.1.3 Install USB Driver

- Downloading USB Driver: <https://developer.android.com/studio/run/win-usb#download>
- Downloading Android SDK Platform-Tools: <https://developer.android.com/studio/releases/platform-tools>
- Installing USB Driver: <http://developer.android.com/tools/extras/oem-usb.html#InstallingDriver>

6.1.4 Fastboot Flash Memory Update

For the location of GPT partition information, refer to Chapter 4.1.5.2

Use **fastboot** command in Windows PC or in Linux PC to write, erase, and reboot partition.

Function	Command
Write	fastboot flash <partition name> <image file name>
Erase	fastboot erase <partition name>
Reboot	fastboot reboot

- Example of Writing Partition (Fastboot in Bootloader)

```
reboot bootloader
fastboot flash boot_a Z:\projects\auto\8050_onDev\maincore\out\target\product\car_tccxxx\boot.img
fastboot flash vbmeta_a Z:\projects\auto\8050_onDev\maincore\out\target\product\car_tccxxx\vbmeta.img
```

- Example of Writing Partition (Fastboot daemon in Recovery)

```
reboot bootloader
fastboot flashing unlock
fastboot reboot
reboot fastboot
fastboot flash system_a Z:\projects\auto\8050_onDev\maincore\out\target\product\car_tccxxx\system.img
fastboot flash vendor_a Z:\projects\auto\8050_onDev\maincore\out\target\product\car_tccxxx\vendor.img
```


7 FAST COLD BOOT

This SDK is optimized to improve the booting speed of EVB, and this improvement is called Fast Cold Boot. The optimized parts are described as follows:

7.1 Optimize Kernel

- Applied fast cold boot to SDK for modularization of Kernel driver.
 - Modularized drivers: SDMMC, Sound, and USB
 - After building the kernel, for the detailed driver name, refer to "module.order" file in kernel directory.
 - Modified files for modularization of kernel driver are as follows:
 - For installing driver modules in vendor partition
 - device/telechips/car_tccxxx/device.mk
 - For installing driver modules during booting
 - device/telechips/car_tccxxx/init.tccxxx.rc
- Disable console output.
 - Apply **quiet** option.
 - Add **quiet** option to BOARD_KERNEL_CMDLINE in device/telechips/<device>/BoardConfig.mk as follows:

```
BOARD_KERNEL_CMDLINE := \  
    console=ttyAMA0,115200n8 androidboot.console=ttyAMA0 \  
    androidboot.hardware=$(TARGET_BOARD_PLATFORM) \  
    androidboot.selinux=permissive printk.devkmsg=on quiet
```

- When the **quiet** option is applied, check the kernel log by using **dmesg** command.

7.2 Optimize Android Framework

- Optimize boot animation
- Optimize Zygote
- Optimize system application
 - Remove the unnecessary system applications.
 - Calendar, Contacts, DeskClock, Email, and ExactCalculator
 - Camera2, MusicFX, NfcNci, and EmergencyInfo

Note:

- All optimizations of this SDK are based on Telechips EVB with Android Open Source Project (AOSP) car product.
- You should optimize and add the system applications and services.
- Optimization points of this SDK may be changed to improve performance.

8 ANDROID APPLICATION GUIDE

8.1 Launcher

When the device booting is completed, the following screen is shown on LCD.



Figure 8.1 Bootup Screen

When you touch the red box area in Figure 8.1, the screen is shown as Figure 8.2.

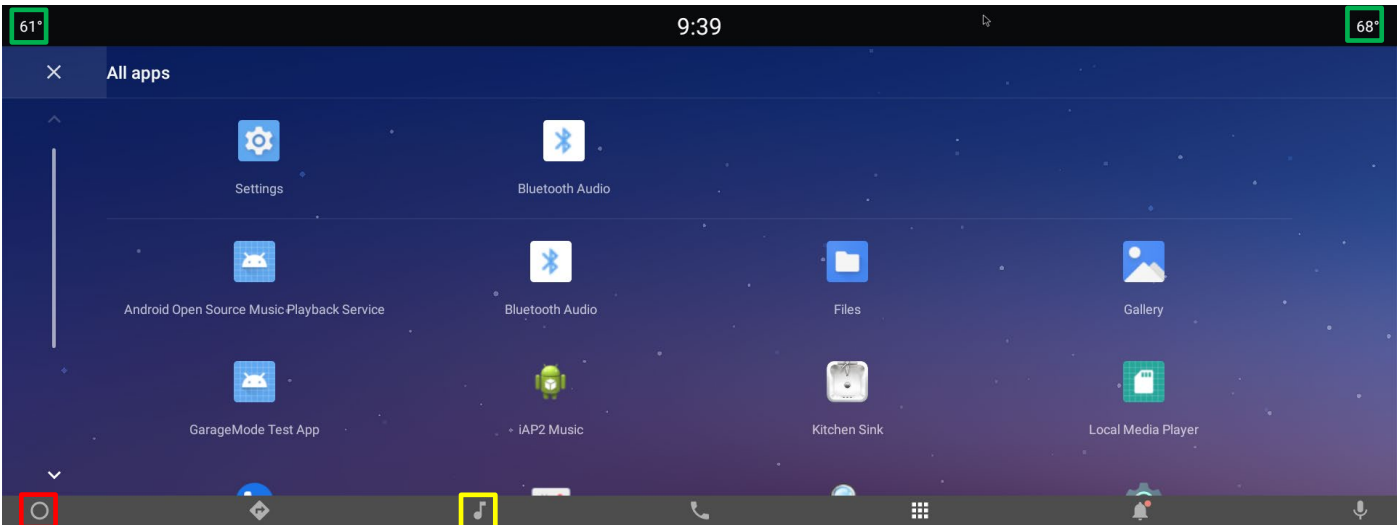


Figure 8.2 List of All Applications

AOSP Google applications and the installed applications are shown in Figure 8.2.
The red box in Figure 8.2 is the home button to move to the home screen when touched.

8.2 Local Media Player

When you touch the **yellow box** area in Figure 8.2, the screen is shown as Figure 8.3.

When you touch the **blue box** area in Figure 8.3, applications related to music are shown as Figure 8.4.

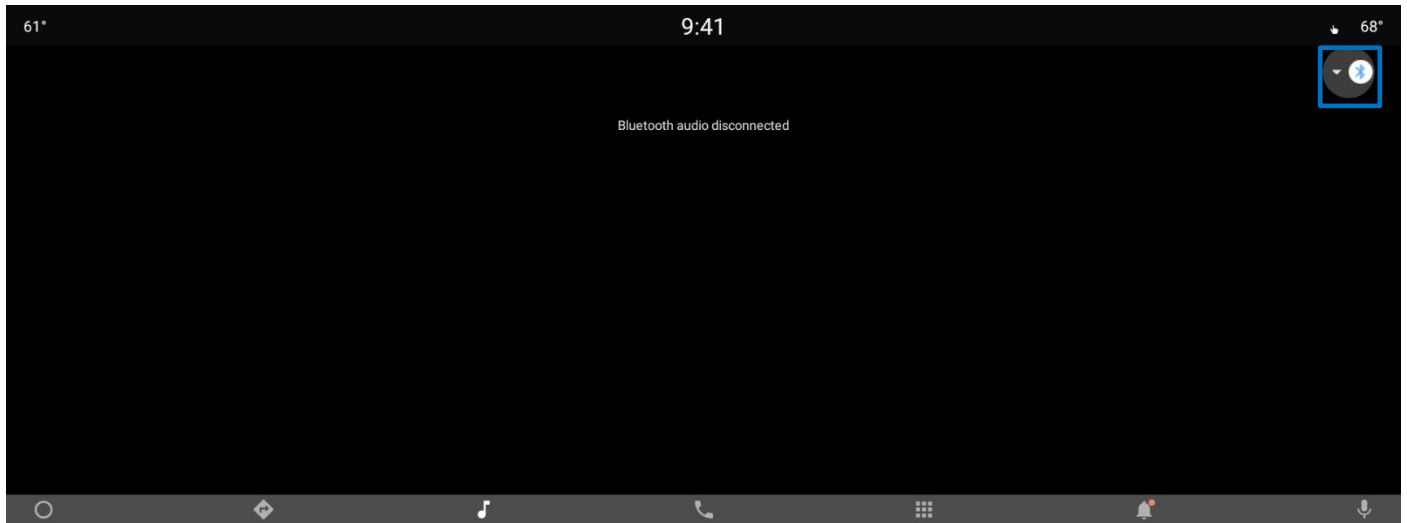


Figure 8.3 List of Music Application (1)

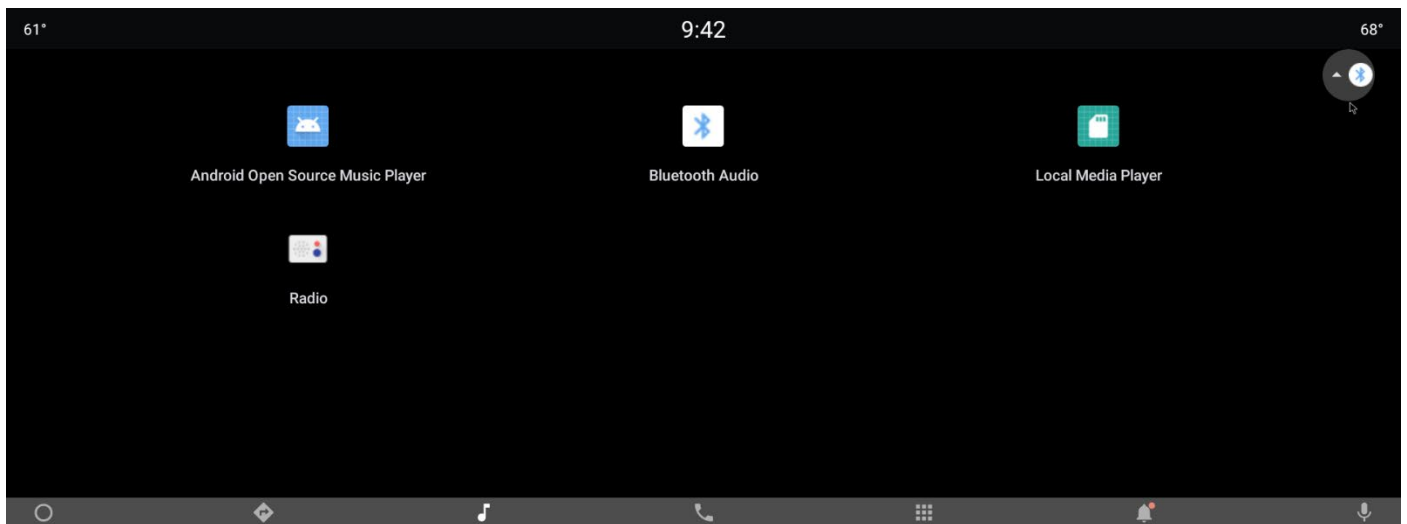


Figure 8.4 List of Music Application (2)

When you select **Local Media Player** icon, the music player screen in Figure 8.5 and Figure 8.6 is shown. The local media player classifies audio files into four categories: **Folders**, **Albums**, **Artists**, and **Genres**.

If you select **Folders** in Figure 8.5, local system audio and music in storage are shown as Figure 8.5. If you select the **red box** in Figure 8.5, you can see the playlist in storage.

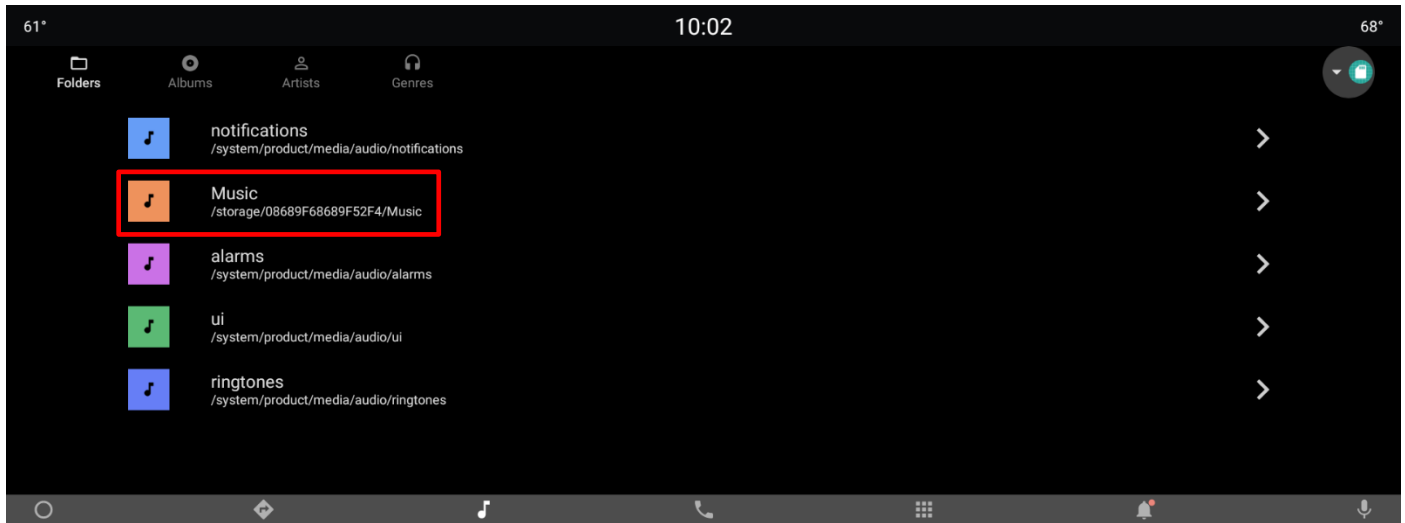


Figure 8.5 Local Media Player

When you select one file in the playlist, music player screen is shown and music is played as Figure 8.6.

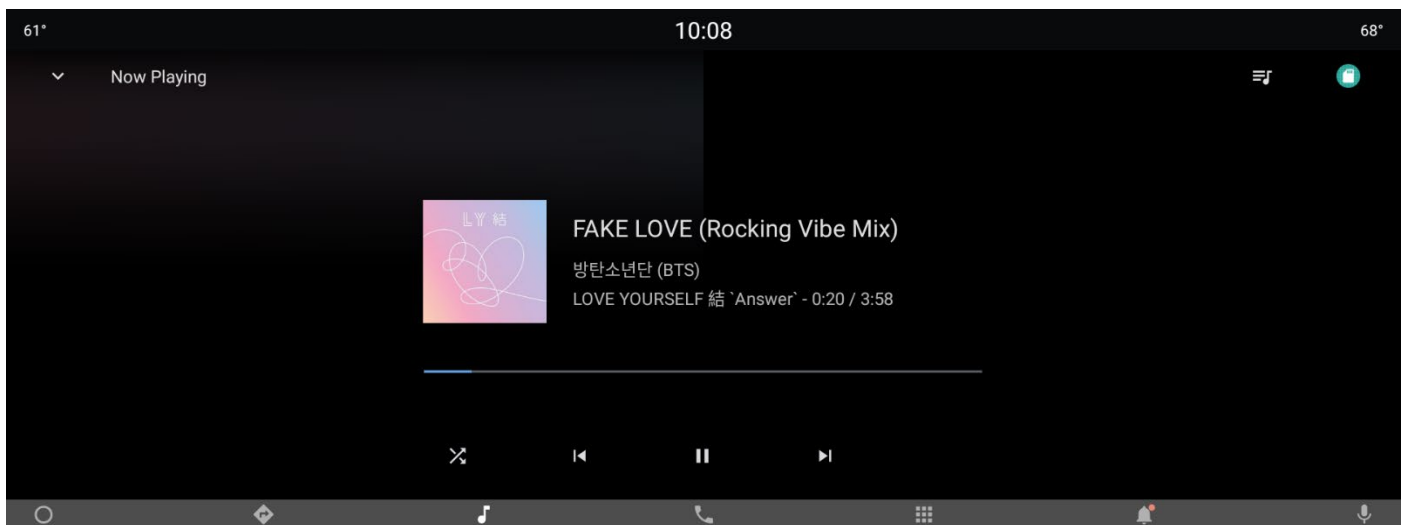


Figure 8.6 Music Player Screen

8.3 Heating, Ventilation, and Air Conditioning (HVAC)

When you touch one of two green boxes in Figure 8.2, HVAC application is shown as Figure 8.7.

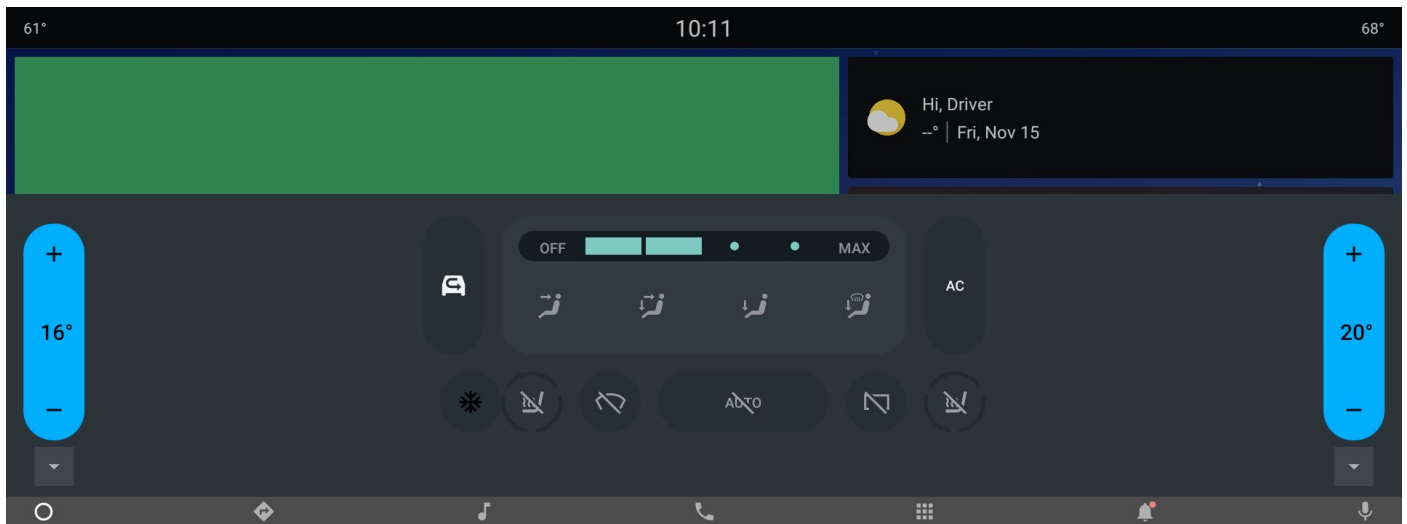


Figure 8.7 HVAC Application

HVAC application is an AOSP Google application and provides HVAC UI which can set wind mode, air in/out, air condition, temperature, and so on.

The communication between the HVAC application and Android automotive framework is provided by default in AOSP. A reference code for Vehicle Hardware Abstraction Layer (VHAL), which communicates with MICOM, is provided by Telechips when the reference code is generated.

9 TELECHIPS APPLICATION GUIDE

9.1 Overview

Telechips application is a reference for complex motion scenario, so only Android Application Package (APK) files are provided without source code.

9.2 Application List

Application	Description
TCCarLauncher	Navigation and media information display
Navit	Open source navigation
TMPlayer	Audio/Video player
TCKeyInput	Hardware key management
TCGoogleTTS	Google text-to-speech application
TMPlayerService	Audio playback service
TCModeService	Mode resource management service
TCKeyInputService	Hardware key resource management service
TCAuxService	Auxiliary playback service

9.3 Enable/Disable Option

Telechips application is installed by default.

If you want to use AOSP application without Telechips application, you need to set **MODE_MANAGER_ENABLED** option to false. The **MODE_MANAGER_ENABLED** option can be changed in "integration-ui.mk" in device/telechips/common.

```
MODE_MANAGER_ENABLED := true
```


9.4 TCCarLauncher

When the device booting is completed, the following screen is shown on LCD.

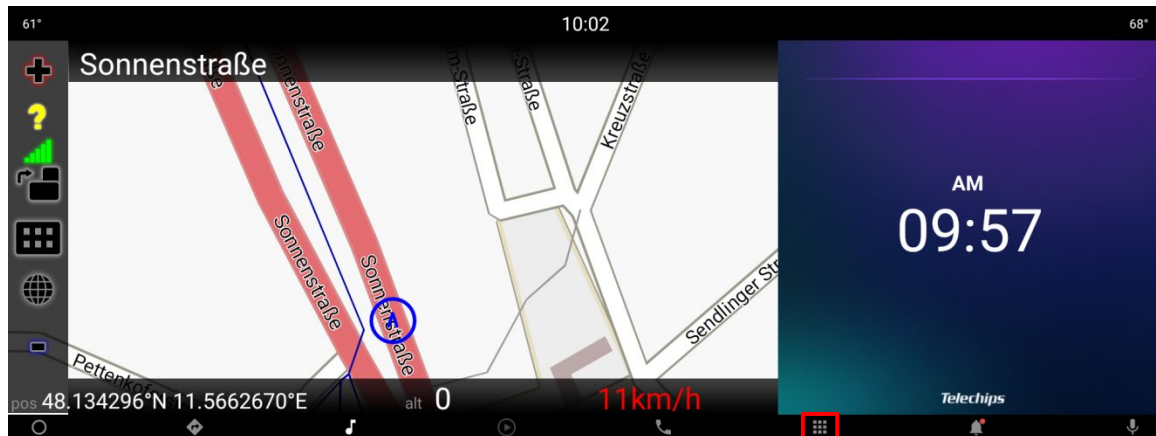


Figure 9.1 Bootup Screen

When you touch the **red box** area in Figure 9.1, the screen is shown as Figure 9.2.



Figure 9.2 List of All Applications

AOSP Google applications and the installed applications are shown in Figure 9.2 above. The **red box** in Figure 9.2 is the home button to move to the home screen when touched.

When you touch the **blue box** area in Figure 9.2, the screen is shown as Figure 9.3.

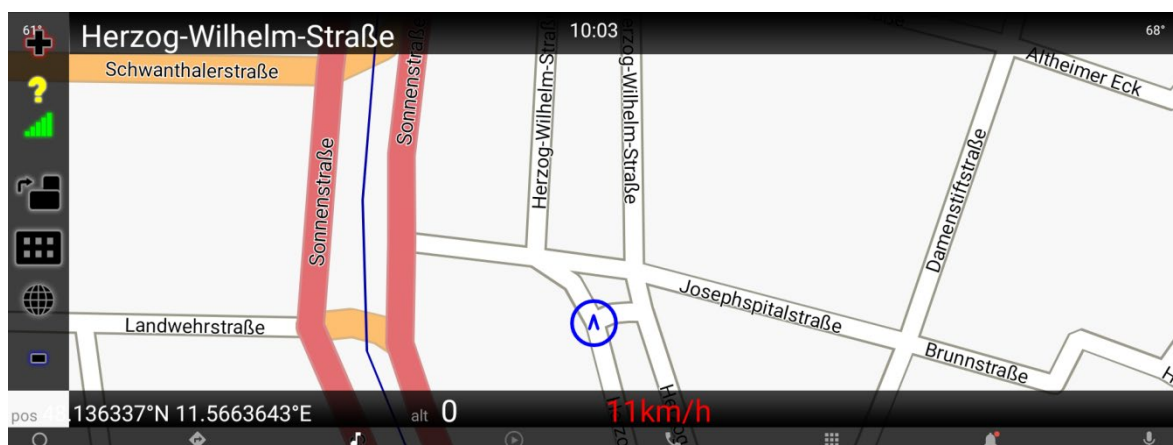


Figure 9.3 Navigation

9.5 TMPlayer

When the storage device is recognized, the icon in the yellow box in Figure 9.2 is activated.

9.5.1 Overview

The **TMPlayer** is a multimedia player to play the audio and video files in an automotive environment, and it supports AUX and iAP2 play.

The **TMPlayer** generates metadata for media files by recognizing a storage device and includes playing media files and Graphic User Interface (GUI).

9.5.2 Feature

- Plays audio content
- Plays video content
- Supports file browsing
- Supports file seek and pause/resume
- Supports shuffle/repeat mode
- Supported device: USB and SDMMC
- Supports multi-device and device change
 - Three USBs and one SDMMC can be simultaneously mounted, and a device can be selected by changing the mode.
- For supported codec list, refer to "TCCxxx Automotive Android SDK-Reference Manual for Supported Media Format-Android 10". [7]

9.5.3 GUI Guide

GUI of **TMPlayer** consists of audio and video screens.

If there is only an audio file in a storage device, it is not available to switch GUI from audio UI to video UI.

If there is only a video file in a storage device, it is not available to switch GUI from video UI to audio UI.

9.5.4 Audio Play

Figure 9.4 shows audio GUI.

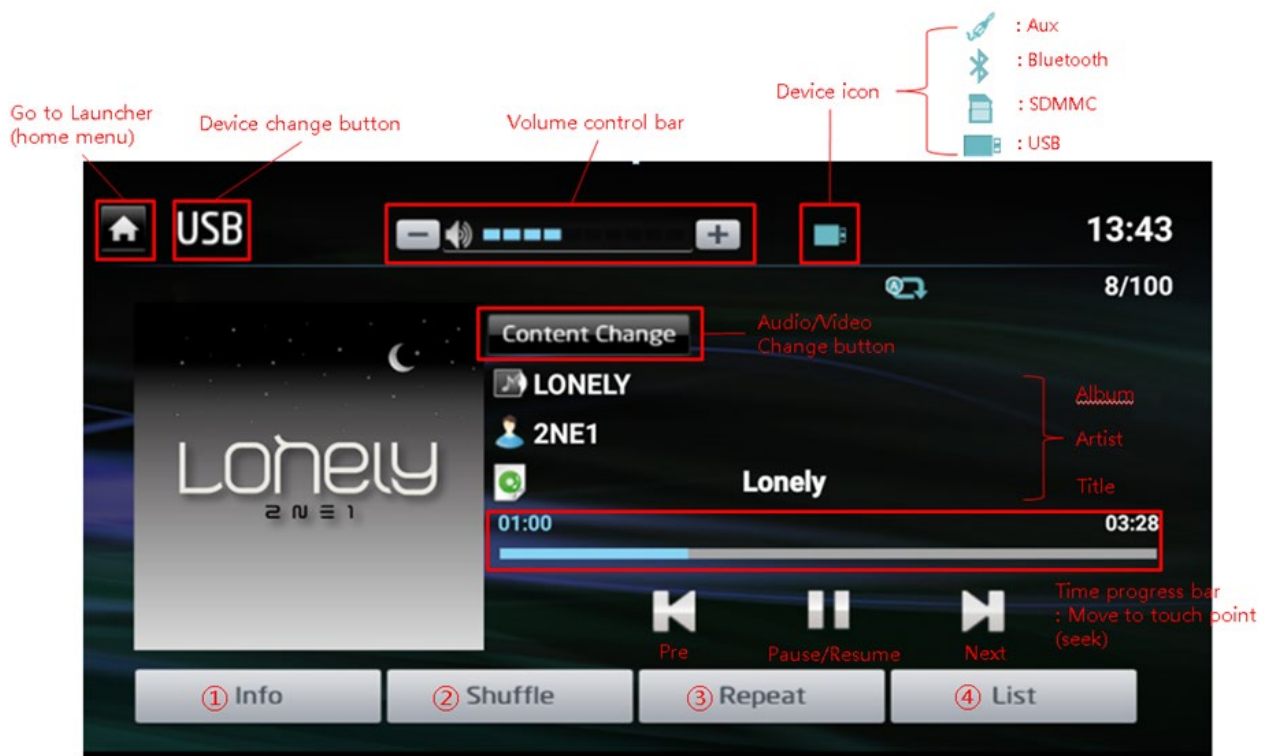


Figure 9.4 Audio Play Screen

■ Info

- If button ① is selected, meta information of a currently played file appears in the pop-up window.

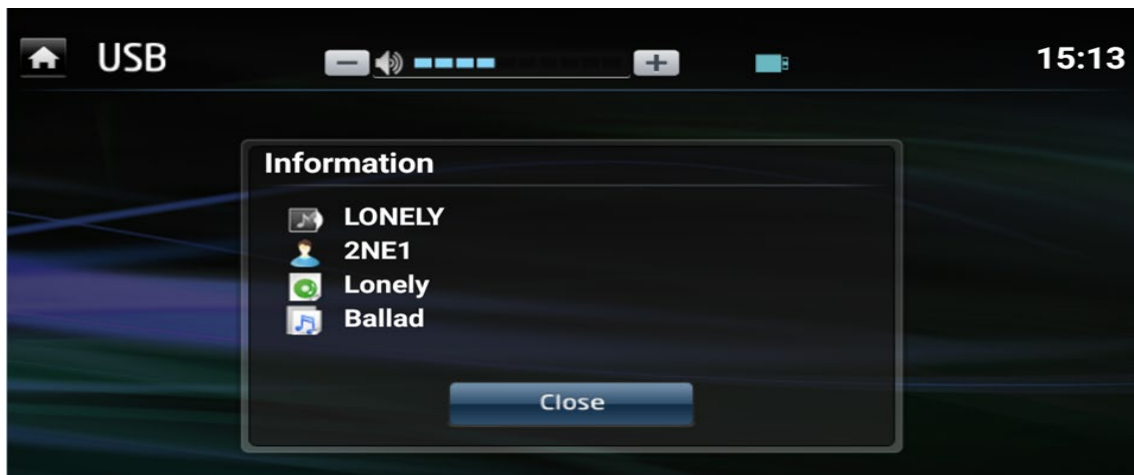


Figure 9.5 Audio Information Pop-up

■ Shuffle

- Button ② shuffles the playlist of the current file.
- Non-shuffle (normal mode) and All shuffle are supported, and an icon is displayed at the top of the screen mode.
 - All shuffle



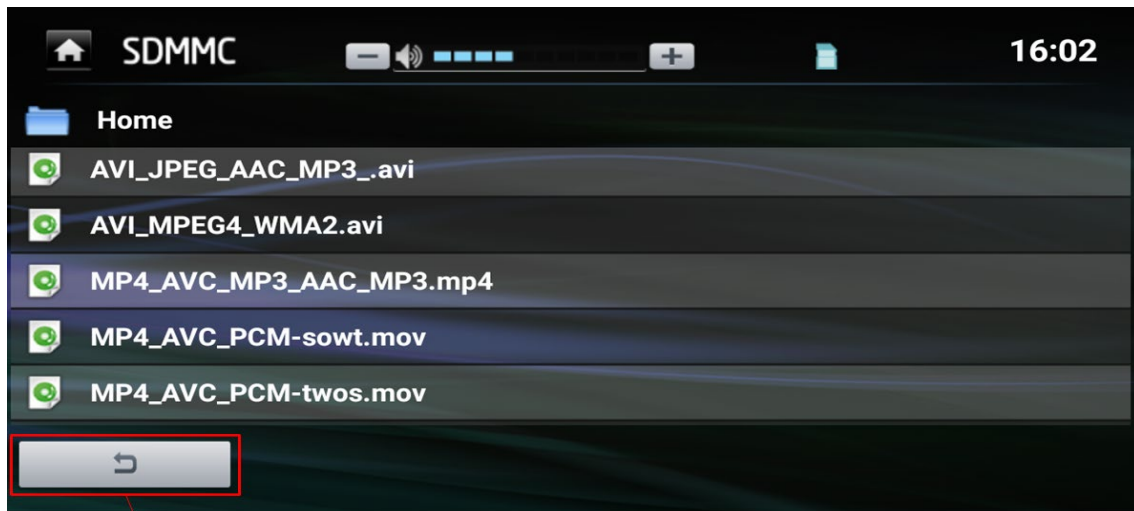
■ Repeat

- Button ③ repeats the playlist of the current file.
 - One song repeat



■ List

- Button ④ shows the playlist.
- The playlist provides a list of files.
- The playlist is sorted by file name, and it can be changed by shuffle or repeat.



Move to play screen

Figure 9.6 Audio File List Screen

9.5.5 Video Play

Figure 9.7 shows the video play screen of GUI.

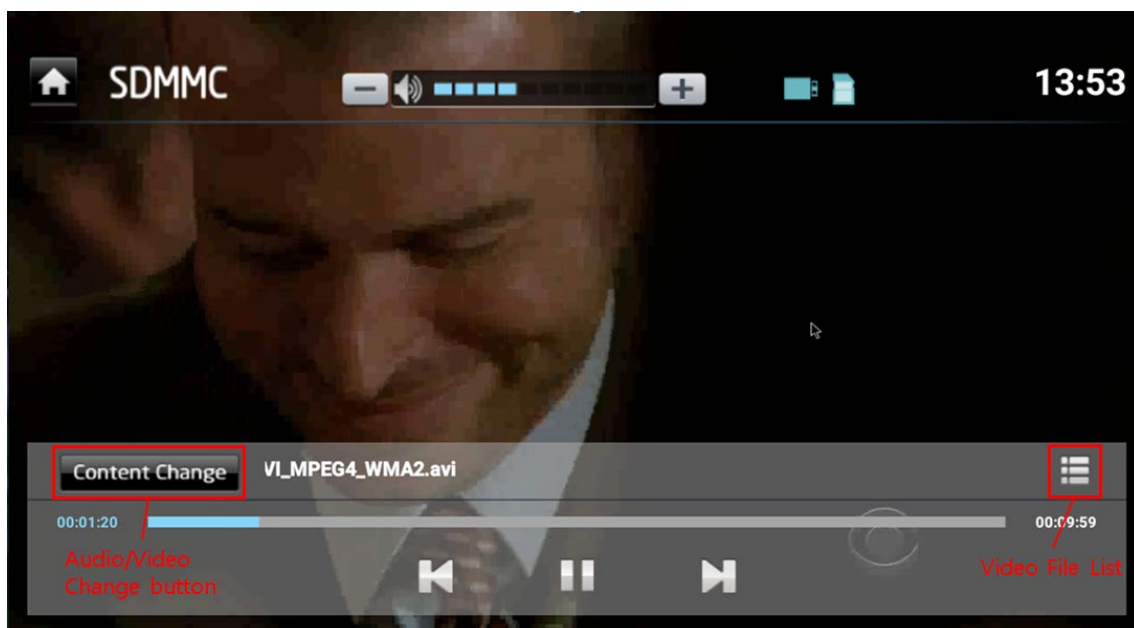


Figure 9.7 Video Play Screen

10 DEBUGGING GUIDE

10.1 Android Debug Bridge (ADB) Guide

This chapter describes how to use the ADB in Android 10. ADB is a versatile command-line utility that communicates with devices.

ADB can performs followings:

- Controlling your device on your computer via USB
- Copying files back and forth
- Installing and uninstalling applications
- Running `shell` commands

10.1.1 Set Up ADB

Download and install the platform-tools package by referring to the following site. This contains ADB and other utilities.

- <https://developer.android.com/studio/releases/platform-tools>

10.1.2 Enable ADB Debugging Mode

To use ADB in a device connected by USB, you must enable **USB debugging** in the **red box** in the device system settings under **Developer options**.

- Setting → System → Advanced → Developer Options → Debugging → USB debugging



Figure 10.1 USB Debugging Option

Also, you can enable or disable ADB with the following command as root access.

- Enable ADB.
 - `# su`
 - `# setprop sys.usb.config adb`
- Disable ADB.
 - `# su`
 - `# setprop sys.usb.config host`

To test whether ADB is working properly, connect the EVB to your computer by using a USB cable and run the following command. The device should be listed. If the device is connected but nothing appears on the list, you should install the appropriate driver.

```
C:\W>adb devices
List of devices attached
0000190227195531000A    device
```

Figure 10.2 ADB Device List

10.1.3 Useful ADB Commands

The followings are useful ADB commands.

Table 10.1 ADB Command

Command	Description	Example
adb device	Show the connected ADB devices.	adb device
adb install	Install the package in C:\ package.apk to EVB.	adb install C:\ package.apk
adb push	Push files from your computer to EVB.	adb push C:\log.txt /data
adb pull	Import the file from EVB to your computer.	adb pull /data/test.txt C:\
adb logcat	View log on Android device.	adb logcat
adb shell	Provide an interactive Linux command-line shell of the device.	adb shell

For more information on ADB, refer to <https://developer.android.com/studio/command-line/adb>.

10.2 Logcat Command-line Tool

Logcat is a command-line tool that dumps a log of system messages.

The **logcat** command can be run at console as follows:

- # logcat

Also, you can run **logcat** as the following **ADB** command.

- \$ adb logcat

You can create and execute a shell connection to a device with the following commands:

- \$ adb shell
- # logcat

The followings are some of character values, which can determine the property and they are arranged from the lowest priority to the highest priority.

- V Verbose (lowest priority)
- D Debug
- I Info
- W Warning
- E Error
- F Fatal
- S Silent (highest priority, on which nothing is ever printed)

The following filter expression shows all log messages with priority level Error and higher on all tags:

- #adb logcat *:E

The log class allows you to create log entries in your code displayed in the **logcat**. You can designate the log priority.

The followings are examples:

- Added code:
 - Log.i("MyActivity", "MyClass.getView() - get item number \$position")
- **Logcat** output:
 - I/MyActivity(1557): MyClass.getView() - get item number 1

For more information, refer to <https://developer.android.com/studio/command-line/logcat>.

10.3 Dmesg

dmesg is a command that dumps a log of system messages. **dmesg** can be useful when performing trouble shooting or getting system information.

The **dmesg** command can be run in console as follows:

- # dmesg

The followings are the options.

- -C Clear ring buffer without printing.
- -c Clear ring buffer after printing.
- -n Set kernel logging LEVEL (1 - 9).
- -r Raw output (with <level markers>)
- -S Use syslog (2) rather than /dev/kmsg.
- -s Show the last SIZE in bytes.
- -T Show human-readable timestamps.
- -t Do not print timestamps.
- -w Keep waiting for more output.

11 REFERENCES

- [1] Contact Telechips for more details: sales@telechips.com
- [2] For more information about ADB: <https://developer.android.com/studio/releases/platform-tools>
- [3] For ADB Command: <https://developer.android.com/studio/command-line/adb>
- [4] For more information about Toolchain: <https://developer.android.com/ndk/guides/build>
- [5] TC Common-User Guide for mktcing,
TCCxxxx Common-User Guide for Firmware Downloader V8
- [6] TCC805x Linux SDK-Getting Started
- [7] TCxxxxx Android SDK-User Guide for Dynamic Partitions

12 REVISION HISTORY

Rev. 1.41: 2023-04-06

- Updated
 - Chapter 3.1.3: Description
 - Chapter 4.1.4.1 – Chapter 4.1.4.3: Some descriptions are relocated as sub-chapters of Chapter 4.1.4, description
 - Chapter 4.1.5.1:
 - Table 4.1: Description
 - Chapter 4.1.5.2, Chapter 4.1.5.3: Description
 - Chapter 6.1.4: Description
 - Chapter 7.1: Description

Rev. 1.40: 2023-01-20

- Updated
 - Chapter 2.2
 - Figures of Boot switch
 - Added Table 2.4
 - Chapter3: Chapter number changed from Chapter 10
 - Chapter 4.1: Description
 - Chapter 4.1.2.2: Description
 - Chapter 4.1.2.3: Description
 - Chapter 4.1.2.4: Description
 - Chapter 4.1.4: Chapter number changed from Chapter 3.1.2.7
 - Chapter4.1.5: Chapter number changed from Chapter 3.1.2.5
 - Added **Note**
 - Chapter5.1.2 and Chapter 5.3: Description
 - Chapter 6.1.4: Description
 - Chapter 11: Description
- Added
 - Chapter 4.1.3 Boot-firmware
 - Chapter 4.1.5.3 Use mktcimg

Rev. 1.30: 2022-11-23

- Updated
 - Chapter ~~3.1.2.5~~: Description
 - Chapter 6.1.1: Description
 - Chapter 6.1.2.1: Description
- Added
 - Chapter ~~3.1.2.5.1 SD Data Partition Information (64-bit SDK)~~
 - Chapter ~~3.1.2.5.2 Partition List File~~
 - Chapter 6.1.2.2 Fastboot in Recovery (fastbootd)

Rev. 1.22: 2022-10-14

- Updated
 - Chapter 2.1: Table 2.1: added main board (V1.2.2 and V1.3.3)
 - Chapter 2.1.2: added **Note**

Rev. 1.21: 2022-04-28

- Updated
 - Chapter 4.1.2.3.3: Kernel Configuration
 - Chapter 4.1.5.4: Description
 - Chapter~~3.1.2.7~~: Autolinux usage

Rev. 1.20: 2021-12-23

- Updated
 - Chapter 2.1: Table 2.1
 - Chapter 2.1.2, Chapter 4.1.1, Chapter 4.1.2.1: Description
 - Chapter 4.1.2.2.1, Chapter 4.1.2.2.2: Description
 - Chapter 4.1.2.3.2, Chapter 4.1.2.3.3, Chapter 4.1.2.3.4: Description
 - Chapter 4.1.2.4, Chapter 3.1.2.5, Chapter 4.1.5.4, Chapter 4.1.5.4.1: Description
 - Chapter 5.1.1: Table 5.1
 - Chapter 5.3, Chapter 6.1.4, Chapter 3.1.2.4: Description
- Added
 - Chapter 3.1.2.7 ~~Build Sub-core by Autolinux~~

Rev. 1.12: 2021-03-29

- Updated
 - Chapter 2.1: Chapter title and Table 2.1: EVB Version
 - Chapter 2.1.1: Figure 2.1 and Figure 2.2
 - Chapter 2.1.2: Figure 2.4 and Note
 - Chapter 2.1.3: Figure 2.5
 - Chapter 2.2: Table 2.3
 - Chapter 2.2.2: Table 2.5
 - Chapter 2.2.3: Chapter title and Table 2.6
 - Chapter 0: Table 2.7
 - Chapter 2.2.5: ~~Table 2.7~~
 - Chapter 4.1.1, Chapter 4.1.2.1, Chapter 4.1.2.2.1, Chapter 4.1.2.3.2, Chapter 4.1.2.3.5: Description
 - Chapter 4.1.5.4, Chapter 4.1.5.4.1: Description
 - Chapter 5.1.2: Table 5.2
 - Chapter 5.3, Chapter 6.1.2: Description
 - Chapter 9.5.2: Description
- Added
 - Chapter 2.1.1: Figure 2.3 TCC803XP_LPD4321_V1.0.0

Rev. 1.11: 2020-11-05

- Updated
 - Chapter 0: Table 2.7: Boot mode switch
 - Chapter 4.1.2.2.1: Script
 - Chapter 4.1.2.3.3: Description
 - Chapter 11: [5]

Rev. 1.10: 2020-09-25

- Updated
 - Chapter ~~3.1.2.5~~: Chapter title and description
- Added
 - Chapter 4.1.5.4 Make SNOR ROM Image

Rev. 1.00: 2020-09-21

- Updated
 - Chapter 1: Description
 - Chapter 2.2: Table 2.3
 - Chapter 2.2.2: Table 2.5
 - Chapter 2.2.3: Table 2.6
 - Chapter 4.1.1: Git server
 - Chapter 4.1.2.1, Chapter 4.1.2.2.1, Chapter 4.1.2.3.2: Description
 - Chapter 5.3, Chapter 6.1.4: Description

- Added
 - Chapter 2.1.1: Figure 2.2
 - Chapter 2.2.4 and 2.2.5
 - Chapter 9 Telechips Application Guide
- Changed
 - Document title from “TCC805x Automotive Android SDK-Getting Started for Android 10”

Rev. 0.01: 2020-07-14

- Preliminary version release

DISCLAIMER

All information and data contained in this material are without any commitment, are not to be considered as an offer for conclusion of a contract, nor shall they be construed as to create any liability. Any new issue of this material invalidates previous issues. Product availability and delivery are exclusively subject to our respective order confirmation form; the same applies to orders based on development samples delivered. By this publication, Telechips, Inc. does not assume responsibility for patent infringements or other rights of third parties that may result from its use.

Further, Telechips, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of Telechips, Inc.

This product is designed for general purpose, and accordingly customer be responsible for all or any of intellectual property licenses required for actual application. Telechips, Inc. does not provide any indemnification for any intellectual properties owned by third party.

Telechips, Inc. cannot ensure that this application is the proper and sufficient one for any other purposes but the one explicitly expressed herein. Telechips, Inc. is not responsible for any special, indirect, incidental, or consequential damage or loss whatsoever resulting from the use of this application for other purposes.

COPYRIGHT STATEMENT

Copyright in the material provided by Telechips, Inc. is owned by Telechips unless otherwise noted.

For reproduction or use of Telechips' copyright material, permission should be sought from Telechips. That permission, if given, will be subject to conditions that Telechips' name should be included and interest in the material should be acknowledged when the material is reproduced or quoted, either in whole or in part. You must not copy, adapt, publish, distribute, or commercialize any contents contained in the material in any manner without the written permission of Telechips. Trademarks used in Telechips' copyright material are the property of Telechips.

Important Notice

This material may include technology owned by the 3rd party licensor and the technology may be subject to its associated licenses. It is solely customer's responsibility to identify and comply with such licenses. No other licenses are granted or implied by Telechips with making available this material.

For customers who use licensed Codec ICs and/or licensed codec firmware of mp3:

"Supply of this product does not convey a license nor imply any right to distribute content created with this product in revenue-generating broadcast systems (terrestrial. Satellite, cable and/or other distribution channels), streaming applications (via internet, intranets and/or other networks), other content distribution systems (pay-audio or audio-on-demand applications and the like) or on physical media (compact discs, digital versatile discs, semiconductor chips, hard drives, memory cards and the like). An independent license for such use is required. For details, please visit <http://mp3licensing.com>".

For customers who use other firmware of mp3:

"Supply of this product does not convey a license under the relevant intellectual property of Thomson and/or Fraunhofer Gesellschaft nor imply any right to use this product in any finished end user or ready-to-use final product. An independent license for such use is required. For details, please visit <http://mp3licensing.com>".

For customers who use Digital Wave DRA solution:

"Supply of this implementation of DRA technology does not convey a license nor imply any right to this implementation in any finished end-user or ready-to-use terminal product. An independent license for such use is required."

For customers who use DTS technology:

"This product made under license to certain U.S. patents and/or foreign counterparts."
"© 1996 – 2011 DTS, Inc. All rights reserved."

For customers who use Dolby technology:

"Supply of this Implementation of Dolby technology does not convey a license nor imply a right under any patent, or any other industrial or intellectual property right of Dolby Laboratories, 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 Dolby Laboratories."

For customers who use Google technology:

"Copyright © 2013 Google Inc. All rights reserved."

└

For customers who use MS technology:

"This product is subject to certain intellectual property rights of Microsoft and cannot be used or distributed further without the appropriate license(s) from Microsoft."