PET_RK3562_P01 安卓主板
开发板编译烧写手册

一、搭建编译环境

1、使用配置好的虚拟机编译

在开发工具目录下有虚拟机磁盘镜像文件（VMware 15.5.6 及以上版本），虚拟机内存设置最少需要16G，磁盘镜像文件所在的 windows 磁盘分区剩余容量大于 300G，虚拟机磁盘文件放到固态盘分区内。

虚拟机磁盘镜像文件已经安装好相关软件，不用再运行下面的安装命令，可以直接复制源码后解压编译。

虚拟机默认用户名和密码为 gzpeite

2、新创建编译主机

安装 Ubuntu 22.04 64 位桌面操作系统。

安装依赖软件，关闭不用的系统服务，live-build_20230502_all.deb 文件位于开发工具目录内。

```
sudo apt update
sudo apt -y upgrade
sudo apt -y dist-upgrade
sudo apt -y install gcc make openssh-server net-tools samba git vim bzr ecj cvs python2
sudo apt -y install unzip xsltproc gawk flex quilt mercurial texinfo bison liblz4-tool cmake
sudo apt -y install build-essential libncurses5-dev zlib1g-dev libssl-dev libxml-parser-perl
sudo apt -y install lib32z1 lib32z1-dev libc6:i386 libstdc++6:i386 libgmp-dev libmpc-dev
sudo apt -y install autoconf gettext lzop gcc-multilib g++-multilib libncurses5 curl libxml2-utils pip
sudo apt -y install dos2unix device-tree-compiler u-boot-tools live-build expect qemu-user-static
sudo apt -y install openjdk-19-jdk
sudo ln -sf /usr/bin/python2 /usr/bin/python
sudo dpkg -i live-build_20230502_all.deb

sudo systemctl stop systemd-oomd.service
sudo systemctl disable systemd-oomd.service
sudo apt -y autoremove --purge systemd-oomd
```

二、编译 Android 安卓系统

1、解压源代码

将源代码压缩文件全部复制到 Ubuntu 系统下，保证所在磁盘剩余空间要大于 300G，使用以下命令解压源代码（注意参数中是大写 J）：

```
tar xvJf PET_RK3562_P01_Android13_Source.tar.xz
```

2、编译 Android 源码

编译安卓 13 最少需要 16GB 内存，如果内存容量太小，可能会引起编译错误。

如果使用虚拟机，需要将虚拟机磁盘文件放到固态盘分区内，否则可以出现编译卡死不动的情况。

如果编译过程中出现错误提示:“Exception in thread "main" java.lang.OutOfMemoryError: Java heap space”可以尝试命令行设置环境变量：export _JAVA_OPTIONS="-Xmx8G"，同时减少虚拟机 cpu 核心的个数。

```
cd PET_RK3562_P01_Android13
编译 RK3562：  ./gzpeite_build_android.sh
编译 RK3562J：./gzpeite_build_android.sh -j
编译完成后正确提示如下：
```

```
boot,Add file: ./Image/boot.img done,offset=0x481000,size=0x229e800,userspace=0x453d,flash_addres
s=0x0000c800
Add file: ./Image/recovery.img
recovery,Add file: ./Image/recovery.img done,offset=0x271f800,size=0x2def800,userspace=0x5bdf,fla
sh_address=0x00020800
Add file: ./Image/baseparameter.img
baseparameter,Add file: ./Image/baseparameter.img done,offset=0x550f000,size=0x100000,userspace=0
x200,flash_address=0x0020cc00
Add file: ./Image/gzpeite.img
gzpeite,Add file: ./Image/gzpeite.img done,offset=0x560f000,size=0xc00000,userspace=0x1800,flash_
address=0x0020d400
Add file: ./Image/logo_rk.img
logo_rk,Add file: ./Image/logo_rk.img done,offset=0x620f000,size=0x8000000,userspace=0x10000,flas
h_address=0x00215400
Add file: ./Image/super.img
super,Add file: ./Image/super.img done,offset=0xe20f000,size=0x660f7510,userspace=0xcc1ef,flash_a
ddress=0x00257400
Add CRC...
Make firmware OK!
------ OK ------
********rkImageMaker ver 2.23********
Generating new image, please wait...
Writing head info...
Writing boot file...
Writing firmware...
Generating MD5 data...
MD5 data generated successfully!
New image generated successfully!
Making update.img OK.
Make update image ok!
/root/Work/PET_RK3562_Android_13
Total compile time is 63 minutes
root@gzpeite:~/Work/PET_RK3562_Android_13#
```

编译完成后在 PET_RK3562_P01_Android13/rockdev/Image-rk3562_t 目录下生成 update.img 烧写镜像文件。首次编译会很耗时，后续修改 uboot、kernel、android 的某个源码后再次编译会快很多。

3、清理 Android 源码

```
cd PET_RK3562_P01_Android13
./gzpeite_build_android.sh -c          会自动清除所有编译过程产生的文件。
```

三、编译 Linux 系统

1、解压源代码

将源代码压缩文件全部复制到 Ubuntu 系统下，使用以下命令解压源代码（注意参数中是大写 J）：

```
tar xvJf PET_RK3562_P01_Linux_Source.tar.xz
```

2、编译 buildroot

```
cd PET_RK3562_P01_Linux
编译 RK3562：  ./gzpeite_build_linux.sh
编译 RK3562J：./gzpeie_build_linux_j.sh
```

```
        find . arch/arm64 -maxdepth 1 -name Makefile\*
        find include -type f -o -type l
        find arch/arm64 -name module.lds -o -name Kbuild.platforms -o -name Platform
        find $(find arch/arm64 -name include -o -name scripts -type d) -type f
        find arch/arm64/include Module.symvers -type f
        echo .config
} | tar --no-recursion --ignore-failed-read -T -        -cf "/root/Work/PET_RK3562_Linux/output/l
inux-headers/linux-headers-aarch64.tar"

        # Pack kbuild
        tar -uf "/root/Work/PET_RK3562_Linux/output/linux-headers/linux-headers-aarch64.tar" -C "
/root/Work/PET_RK3562_Linux/output/linux-headers/linux-kbuild-aarch64" scripts/ tools/
+ cd /root/Work/PET_RK3562_Linux
Unpacking /root/Work/PET_RK3562_Linux/output/linux-headers/linux-headers-aarch64.tar ...
Debian control file:
Package: linux-headers-5.10-arm64
Source: linux-rockchip (5.10)
Version: 5.10-rockchip
Architecture: aarch64
Section: kernel
Priority: optional
Multi-Arch: foreign
Maintainer: Tao Huang <huangtao@rock-chips.com>
Homepage: https://www.kernel.org/
Description: Kbuild and headers for Rockchip Linux 5.10 arm64 configuration
Packing linux-headers-5.10-arm64_aarch64.deb...
Running mk-kernel.sh - linux-headers-aarch64 succeeded.
Running mk-kernel.sh - linux-headers succeeded.
Running 99-all.sh - build_all succeeded.
/root/Work/PET_RK3562_Linux
Build buildroot ok!
Total compile time is 134 minutes
root@gzpeite:~/Work/PET_RK3562_Linux#
```

编译完成后会在 PET_RK3562_P01_Linux/rockdev 目录下生成 update_buildroot.img 烧写文件

3、编译 debian11

编译 debian 之前，需首先完成过 buildroot 的编译，编译主机需要能连接互联网，如果编译后的文件大小与 SDK 里面的差异较大需要检查联网情况。

cd PET_RK3562_P01_Linux

编译 RK3562： ./gzpeite_build_linux.sh debian

编译 RK3562J：./gzpeie_build_linux_j.sh debian

```
        find include -type f -o -type l
        find arch/arm64 -name module.lds -o -name Kbuild.platforms -o -name Platform
        find $(find arch/arm64 -name include -o -name scripts -type d) -type f
        find arch/arm64/include Module.symvers -type f
        echo .config
} | tar --no-recursion --ignore-failed-read -T -        -cf "/root/work/PET_RK3562_Linux/output/l
inux-headers/linux-headers-aarch64.tar"

        # Pack kbuild
        tar -uf "/root/work/PET_RK3562_Linux/output/linux-headers/linux-headers-aarch64.tar" -C "
/root/work/PET_RK3562_Linux/output/linux-headers/linux-kbuild-aarch64" scripts/ tools/
+ cd /root/work/PET_RK3562_Linux
Unpacking /root/work/PET_RK3562_Linux/output/linux-headers/linux-headers-aarch64.tar ...
Debian control file:
Package: linux-headers-5.10-arm64
Source: linux-rockchip (5.10)
Version: 5.10-rockchip
Architecture: aarch64
Section: kernel
Priority: optional
Multi-Arch: foreign
Maintainer: Tao Huang <huangtao@rock-chips.com>
Homepage: https://www.kernel.org/
Description: Kbuild and headers for Rockchip Linux 5.10 arm64 configuration
Packing linux-headers-5.10-arm64_aarch64.deb...
Running mk-kernel.sh - linux-headers-aarch64 succeeded.
Running mk-kernel.sh - linux-headers succeeded.
Running 99-all.sh - build_all succeeded.
/root/work/PET_RK3562_Linux
/root/work/PET_RK3562_Linux
Build debian ok!
Total compile time is 18 minutes
root@gzpeite:~/work/PET_RK3562_Linux#
```

编译完成后会在 rockdev 目录下生成 update_debian.img 烧写文件

4、编译 ubuntu 22.04

编译 ubuntu 之前，需首先完成过 buildroot 的编译，编译主机需要能连接互联网，如果编译后的文件大小与 SDK 里面的差异较大需要检查联网情况。

cd PET_RK3562_P01_Linux

编译 RK3562： ./gzpeite_build_linux.sh ubuntu

编译 RK3562J：./gzpeie_build_linux_j.sh ubuntu

```
        find arch/arm64 -name module.lds -o -name Kbuild.platforms -o -name Platform
        find $(find arch/arm64 -name include -o -name scripts -type d) -type f
        find arch/arm64/include Module.symvers -type f
        echo .config
} | tar --no-recursion --ignore-failed-read -T -        -cf "/root/work/PET_RK3562_Linux/output/l
inux-headers/linux-headers-aarch64.tar"

        # Pack kbuild
        tar -uf "/root/work/PET_RK3562_Linux/output/linux-headers/linux-headers-aarch64.tar" -C "
/root/work/PET_RK3562_Linux/output/linux-headers/linux-kbuild-aarch64" scripts/ tools/
+ cd /root/work/PET_RK3562_Linux
Unpacking /root/work/PET_RK3562_Linux/output/linux-headers/linux-headers-aarch64.tar ...
Debian control file:
Package: linux-headers-5.10-arm64
Source: linux-rockchip (5.10)
Version: 5.10-rockchip
Architecture: aarch64
Section: kernel
Priority: optional
Multi-Arch: foreign
Maintainer: Tao Huang <huangtao@rock-chips.com>
Homepage: https://www.kernel.org/
Description: Kbuild and headers for Rockchip Linux 5.10 arm64 configuration
Packing linux-headers-5.10-arm64_aarch64.deb...
Running mk-kernel.sh - linux-headers-aarch64 succeeded.
Running mk-kernel.sh - linux-headers succeeded.
Running 99-all.sh - build_all succeeded.
/root/work/PET_RK3562_Linux
/root/work/PET_RK3562_Linux
/root/work/PET_RK3562_Linux
Build ubuntu ok!
Total compile time is 6 minutes
root@gzpeite:~/work/PET_RK3562_Linux#
```

编译完成后会在 rockdev 目录下生成 update_ubuntu.img 烧写文件

5、清理 Linux 源码

```
cd PET_RK3562_P01_ Linux
./gzpeite_build_linux.sh cleanall
```

会自动清除所有编译过程产生的文件。

四、镜像文件烧写

1、安装驱动并连接硬件

解压开发工具目录下的 DriverAssitant_USB 驱动程序.7z，右击以管理员权限运行 DriverInstall.exe，安装驱动程序。

注意 win10 或 win11 系统需要关闭操作系统的驱动签名验证功能才能正常安装驱动，win11 不能永久关闭这个功能，建议安装一个 win7 的虚拟机，在虚拟机内进行驱动安装和固件烧写。

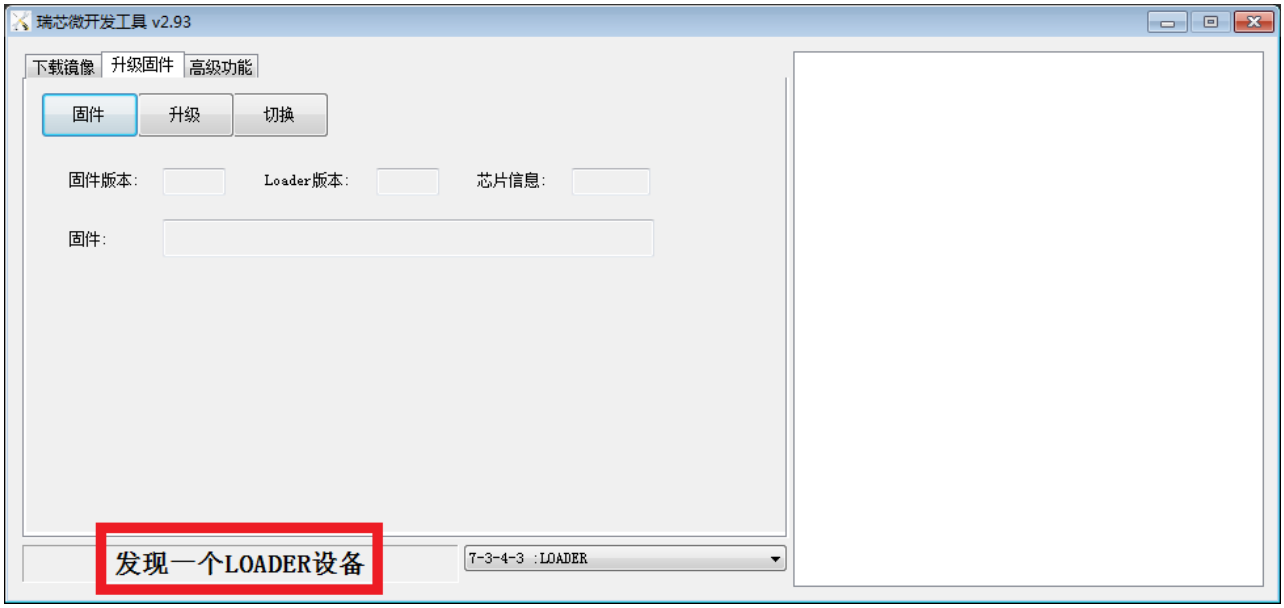
将主板与 PC 机用 TYPE-C 线连接好，如果出现无法识别的情况可以通过重新连接、更换 PC 机 USB 接口、更换 USB 线、更换 PC 机等方式重试。

解压开发工具目录下的 RKDevTool.7z，右击以管理员权限运行 RKDevTool.exe

2、进入烧写模式

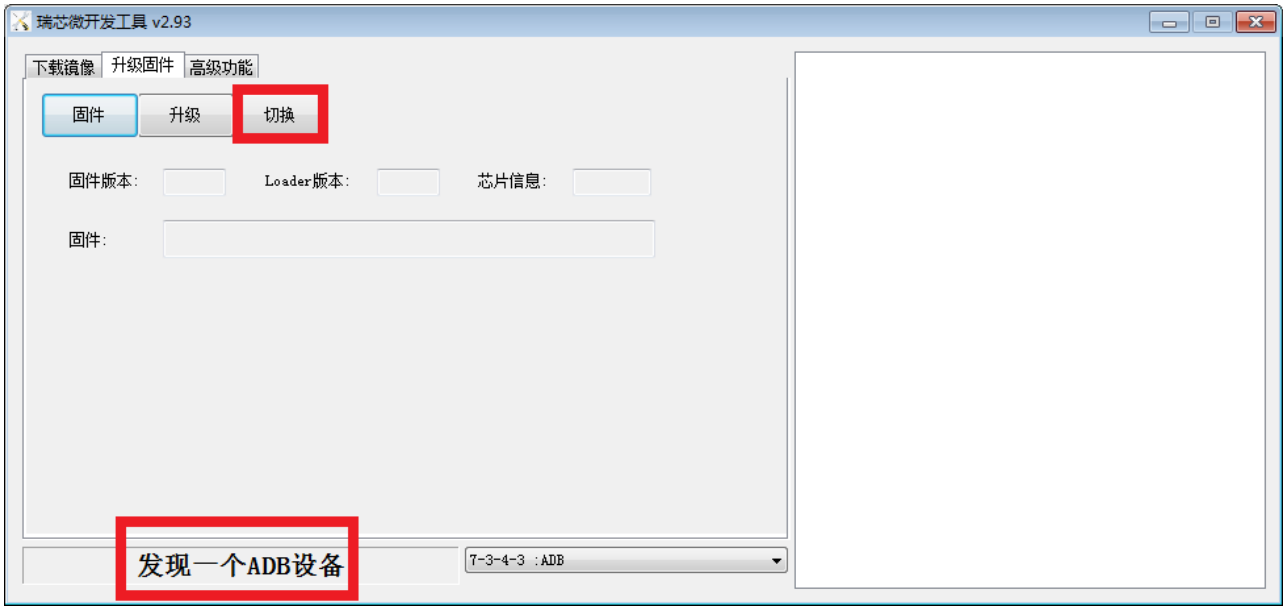
主板处于 Loader 或 Maskrom 模式时可以对系统进行格式化和烧写系统镜像文件操作。

进入 Loader 模式



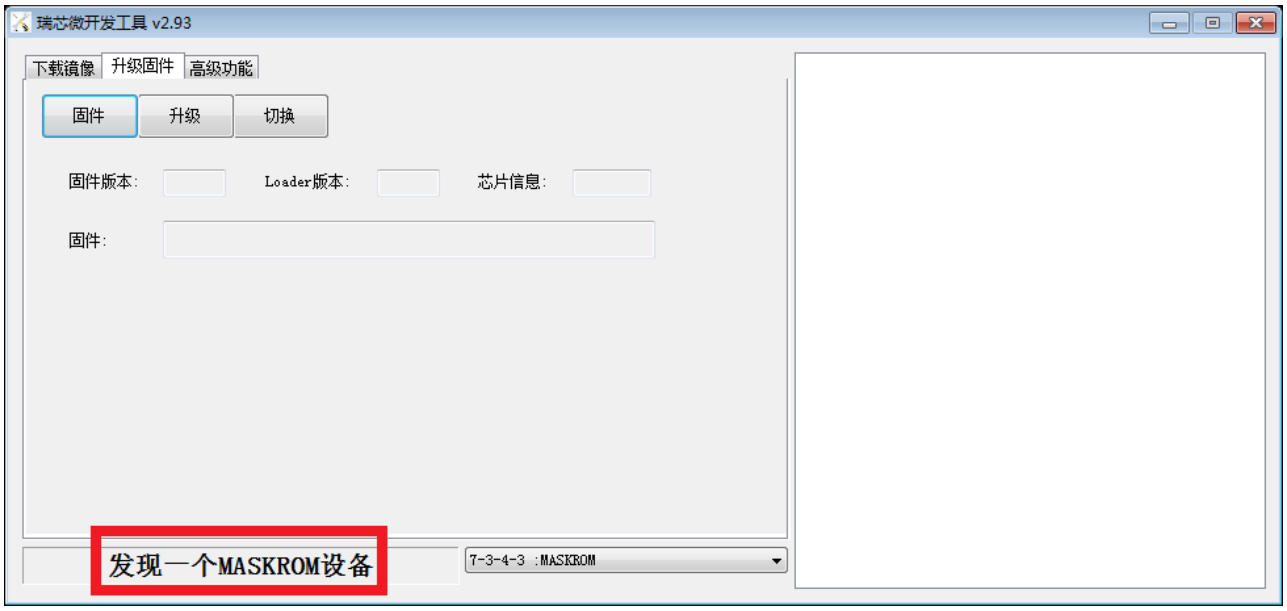
方式一、将主板断电，首先用 TYPE-C 线将主板与 PC 机连接好，按下主板上 SW1 烧写键，并保持按下状态，然后再上电开机，主板会进入 Loader 操作模式，注意这种方式适用于主板上的 BootLoader 可正常工作的情况。

方式二、将主板接通电源，进入系统，用 TYPE-C 线将主板与 PC 机连接好如下图，RKDevTool 软件会识别到一个 ADB 设备，单击切换按钮，主板会重启进入 Loader 模式



方式三、在调试串口控制台或其他控制终端输入 reboot loader，主板会重启进入 Loader 模式

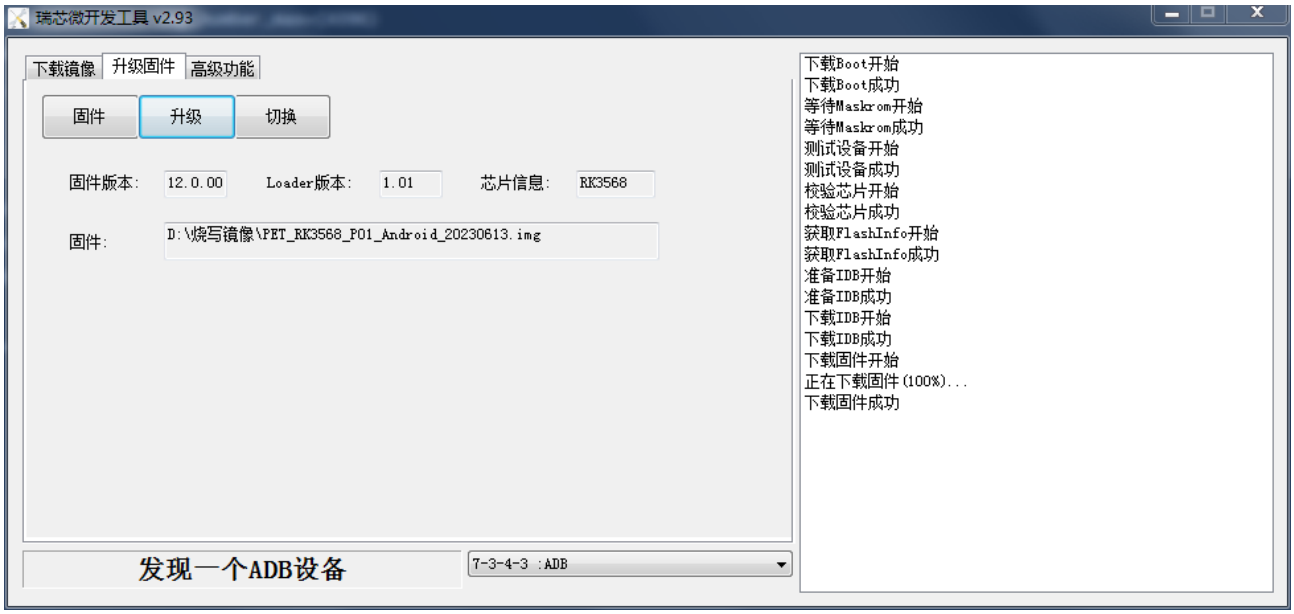
3、进入 Maskrom 模式



一般仅在 uboot 损坏，无法进入 Loader 模式时使用。将主板断电，首先用 TYPE-C 线将主板与 PC 机连接好，按下主板上的 SW2 ROM 键，并保持按下状态，然后再上电开机，主板会进入 MaskRom 操作模式，然后松开按键烧写系统即可。

4、系统烧写流程

首先将主板进入 Loader 或 Maskrom 模式，打开 RKDevTool 软件，点击固件按钮选择需要烧写的镜像文件，然后点击升级按钮，右侧窗口会显示烧写进度，烧写完成后，主板会自动重启开机。



五、联系方式

总公司　　：广州佩特电子科技有限公司
总公司地址：广州市天河区大观中路新塘大街鑫盛工业园 A1 栋 201
总公司网站：http://www.gzpeite.net
SMT 子公司：广州佩特精密电子科技有限公司（全资子公司）
子公司地址：广州市白云区人和镇大巷村顺景路 11 号
SMT 网站　：http://www.gzptjm.com
官方淘宝店：https://shop149045251.taobao.com

微信扫描二维码联系支持人员：

广州佩特电子科技有限公司

2024 年 12 月