



# PET-A64-P01 安卓主板/开发板 开发手册



## 一、编译环境搭建指南

- 安装 Ubuntu 16.04 64 位。  $\triangleright$
- ≻

安装依赖软件
sudo apt clean
sudo apt update
sudo apt -y upgrade
sudo apt -y dist-upgrade
sudo apt -y install openssh-server
sudo apt -y install git flex bison gperf build-essential libncurses5-dev:i386
sudo apt -y install libx11-dev:i386 libreadline6-dev:i386 libgl1-mesa-dev g++-multilib
sudo apt -y install tofrodos python-markdown libxml2-utils xsltproc zlib1g-dev:i386
sudo apt -y install dpkg-dev libsdl1.2-dev libesd0-dev
sudo apt -y install git-core gnupg flex bison gperf build-essential
sudo apt -y install zip curl zlib1g-dev gcc-multilib g++-multilib
sudo apt -y install libc6-dev-i386
sudo apt -y install lib32ncurses5-dev x11proto-core-dev libx11-dev
sudo apt -y install lib32z-dev ccache
sudo apt -y install libgl1-mesa-dev libxml2-utils xsltproc unzip m4
sudo apt -y install gawk fakeroot g++-multilib gcc-multilib
sudo apt -y install u-boot-tools make texinfo clang cmake dos2unix unix2dos
sudo apt -y install libssl-dev
安装 openjdk-8-jdk。
sudo apt -y install openjdk-8-jdk
输入命令 java -version 检查 java 的主版本号是否为 1.8。
在开发工具目录下有安装好的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 200G。
编译安卓如果遇到内存不足错误可以加大内存容量或减少编译线程数量。
如果使用虚拟机内存容量不足,编译安卓系统,默认参数可能会因内存不足引起错误,可以尝试
将虚拟机内存设置为 8G,进行以下修改后再重启虚拟机进行编译。
文件 prebuilts\sdk\tools\jack-admin
JACK_SERVER_COMMAND="java -XX:MaxJavaStackTraceDepth=-1 -Djava.io.tmpdir=\$TMPDIR
\$JACK_SERVER_VM_ARGUMENTS -cp \$LAUNCHER_JAR \$LAUNCHER_NAME"
修改为
JACK_SERVER_COMMAND="java -Xmx8G -XX:MaxJavaStackTraceDepth=-1 -Djava.io.tmpdir=\$TMPDIR

\$JACK\_SERVER\_VM\_ARGUMENTS -cp \$LAUNCHER\_JAR \$LAUNCHER\_NAME"

重启虚拟机,再运行编译命令。

## 二、解压源代码

≻

≻  $\triangleright$ 

 $\triangleright$  $\triangleright$ 

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 100G,使用以下命令解 压源代码:



tar xvJf PET\_A64\_P01\_Source.tar.xz

## 三、编译安卓 Android

首次编译请严格按照步骤进行内核、uboot、android 的编译,否则编译可能会出现错误。

#### 1、编译内核

cd lichee
./build.sh -p sun50iw1p1_android -k linux-3.10
编译完成后正确提示如下:
<pre>make: Leaving directory `/root/Work/A64_7.1.1_SKD_BASE/lichee/linux-3.10/modules/gpu' [GPU]: mali400 device driver has been built. regenerate rootfs cpio 15756 blocks 15756 blocks build_ramfs Copy boot.img to output directory Copy modules to target</pre>
sun50iw1p1 compile Kernel successful
INFO: build kernel OK. INFO: build rootfs INFO: skip make rootfs for android INFO: build rootfs OK. INFO:INFO: build lichee OK. INFO:
root@SeKeDe:~/Work/A64_7.1.1_SKD_BASE/lichee#

#### 2、编译 uboot

首次编译或修改 uboot 代码后需要执行这一步骤。

cd lichee/brandy ./build.sh -p sun50iw1p1 编译完成后正确提示如下

/root/Work/A64\_7.1.1\_SKD\_BASE/lichee/brandy/u-boot-2014.07/../gcc-linaro, opy -0 binary /root/Work/A64\_7.1.1\_SKD\_BASE/lichee/brandy/u-boot-2014.07 14\_sunxi\_spl/sunxi\_spl/sbrom/sboot.axf /root/Work/A64\_7.1.1\_SKD\_BASE/lich ./../bootloader/uboot\_2014\_sunxi\_spl/sunxi\_spl/sbrom/sboot.bin /root/Work/A64\_7.1.1\_SKD\_BASE/lichee/brandy/u-boot-2014.07/../../bootloade nxi\_spl/../../tools/add\_hash.sh -f /root/Work/A64\_7.1.1\_SKD\_BASE/lichee/k /bootloader/uboot\_2014\_sunxi\_spl/sunxi\_spl/sbrom/sboot.bin -m sboot build\_sboot 'sboot\_sun50iw1p1.bin' -> '/root/Work/A64\_7.1.1\_SKD\_BASE/lichee/k ack/chips/sun50iw1p1/bin/sboot\_sun50iw1p1.bin' root@SeKeDe:~/Work/A64\_7.1.1\_SKD\_BASE/lichee/brandy#

3、编译 android

cd android
source build/envsetup.sh
lunch tulip_p3-eng
extract-bsp



make -j4

#### pack

编译完成后正确提示如下

config.fex Len: 0xc400
split_xxxx.fex Len: 0x200
sys_partition.fex Len: 0x12c8
sunxi.fex Len: 0x14000
boot0 nand.fex Len: 0x8000
boot0_sdcard.fex_Len: 0x8000
u-boot fex Len: 0xd4000
tocl.fex Len: 0x8
toc0.fex Len: 0x8
fest fex Len: 0x4100
boot package fex Len: 0x130000
ushtool.fex Len: 0x23600
aultools fex Len: 0x2847b
aultis32 fex Len: 0x24d23
cardtool fex Len: 0x41600
cardscript fex Len: 0x6de
sunxi mbr.fex.len: 0x10000
dlinfo fex Len: 0x4000
arisc fex Len: 0x6
hot-resource fex len: 0x4ef400
Vhot-resource fex Len: 0x4
env fex Len: 0x2000
Very fex Len: 0x4
boot fex Len: 0xf30800
Vhoot fex Len: 0x4
system fex Len: 0x2e768144
Vsvstem fex Len: 0x4
recovery fex Len: 0x110c800
Vrecovery fex Len: 0x4
diskfs.fex Len: 0x200
Vdiskfs fex Len: 0x4
BuildIng 0
Dragon execute image cfg SUCCESS 1
indge ib de
<pre>/root/Work/A64_7.1.1_SKD_BASE/lichee/tools/pack/sun50iw1p1_android_p3_uart0.img</pre>
pack finish

编译完成后会在 lichee/tools/pack 目录下生成 sun50iw1p1\_android\_p3\_uart0.img 系统烧写镜像文件。

## 四、GPIO 编程参考

通过 sysfs 方式控制 GPIO, GPIO 的操作接口包括 direction 和 value 等, direction 控制 GPIO 输入和输入模式, 而 value 可控制 GPIO 输出或获得 GPIO 输入。

例如控制调试灯 GPIO 操作如	下(串口终端命令行方式):
调试灯 GPIO 设置为输出	echo out > /sys/class/gpio/gpio235/direction
调试灯 GPIO 输出高电平	echo 1 > /sys/class/gpio/gpio235/value
调试灯 GPIO 输出高低平	echo 0 > /sys/class/gpio/gpio235/value
调试灯 GPIO 设置为输入	echo in > /sys/class/gpio/gpio235/direction
读取调试灯 GPIO 输出输入电	平 cat /sys/class/gpio/gpio235/value

当 GPIO 处于输出和输入模式时都可以读取,当设置为输入模式时读取的是 GPIO 实际电平,当设置为输出 模式时读取的是设置的值(如果设置为高电平输出,外部将引脚电平拉低后,读取的值依然是 1)。

应用程序控制请参考源码下的 demo 程序源码

	GPIO 对	应控制目录列表	
丝印	接口	脚位	目录



LED6	调试状态灯		/sys/class/gpio/gpio235
	DMIC-DIN <u>LED R138</u> 2.2K LED6 RED DGND		
J7	J7 IDC 20-4*2	2 脚	/sys/class/gpio/gpio38
	DCDC1 $\frac{1}{20}$ $\frac{1}{1}$ $\frac{2}{20}$ $\frac{2}{1}$ $\frac{PB_EINT6}{20}$ $AP-PCM0-DOUT$	4 脚	/sys/class/gpio/gpio37
	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	5 脚	/sys/class/gpio/gpio36
	TWI1-SDA ((	6 脚	/sys/class/gpio/gpio39
J34	J34 0R/NC R127	1脚	/sys/class/gpio/gpio233
		3 脚	/sys/class/gpio/gpio142
	CSI-STBY-R TWI2-SCK 30 1 4 6 TWI1-SCK TWI1-SCK CSI-STBY-R TWI2-SDA 50 5 6 6	5 脚	/sys/class/gpio/gpio143
	BB-WAKE	7 脚	/sys/class/gpio/gpio360
	DGND		
J23	J23	5 脚	/sys/class/gpio/gpio139
	HEADER_2.0.2°11 $10 1 2 0^2$ ODLD03	7 脚	/sys/class/gpio/gpio137
	DGND CSI-D7 50 3 4 0 CSI-D7 50 5 6 6 0 CSI-D7 50 5 6 8 CSI-D6 0 CSI-D6 0 CSI-D7 50 5 6 0 CSI-D7 50 5 6 0 CSI-D7 50	8 脚	/sys/class/gpio/gpio138
	CSI-D3 CSI-D3 9 10 CSI-D4 CSI-D4 CSI-D3 CSI-D1 11 9 10 11 12 CSI-D2 CSI-D4	9 脚	/sys/class/gpio/gpio135
	CSI-VSYNC CSI-DO CSI-MCLK 150 13 14 014 CSI-D0 CSI-MCLK 25-CSI-D0 CSI-MCLK 150 15 16 016 CSI-HSYNC CSI-HSYNC CSI-SCK 170 15 16 018 CSI-PCLK CSI-HSYNC	10 脚	/sys/class/gpio/gpio136
	CSI-SDF CSI-STBY-F 19 19 20 CSI-SDA CSI-SDA CSI-SDA CSI-SDA CSI-SDA	11 脚	/sys/class/gpio/gpio133
		12 脚	/sys/class/gpio/gpio134
		13 脚	/sys/class/gpio/gpio131
		14 脚	/sys/class/gpio/gpio132
		15 脚	/sys/class/gpio/gpio129
		16 脚	/sys/class/gpio/gpio130
		17 脚	/sys/class/gpio/gpio140
		18 脚	/sys/class/gpio/gpio128
		19 脚	/sys/class/gpio/gpio145
		20 脚	/sys/class/gpio/gpio141
		21 脚	/sys/class/gpio/gpio144
		22 脚	/sys/class/gpio/gpio202

## 五、WatchDog 看门狗编程参考

进入内核后默认会启动看门狗,内核崩溃等情况出现,会在 60 秒内自动复位主板。

上层应用程序打开看门狗后,内核将看门狗控制权交由上层应用程序控制,上层应用程序的喂狗间隔建议 不少于 10 秒。

看门狗的使用流程为 打开看门狗→循环喂狗→停止喂狗→关闭看门狗

喂狗之前必须先打开看门狗,关闭看门狗之前需停止喂狗操作。

打开看门狗后如果 60 秒内没有喂狗或关闭看门狗,系统会自动复位。

命令行测试:



## 六、串口 UART 编程参考

J8	串口/dev/ttyS1	PH2.0 4Pin	标配	默认为 RS232 串口,与蓝牙功能不可同时使
				用
19	串口/dev/ttyS3	PH2.0 4Pin	标配	默认为 RS232 串口,可配置为 TTL 串口
J10	串口/dev/ttyS5	PH2.0 4Pin	标配	默认为 TTL 串口(只能接收数据), 与 GPS 功
				能不可同时使用
J13	串口/dev/ttyS5	PH2.0 5Pin	非标配	与 J10 复用且位置重叠,不能同时使用
J11	串口/dev/ttyS0	XH2.54 4Pin	标配	默认为 TTL 串口,调试串口
J49	串口/dev/ttyS2	PH2.0 4Pin	标配	默认为 TTL 串口,与 485 功能不可同时使用
J14	485 通讯 /dev/ttyS2	PH2.0 4Pin	标配	默认为 TTL 串口

调试串口修改为普通串口使用请参考安卓主板系统说明书进行源码修改与编译系统镜像文件。 安卓系统串口编程请参考以下链接:

https://github.com/Geek8ug/Android-SerialPort

## 七、动态隐藏/显示系统状态栏和导航栏

注意: 仅在未将系统设置为强制全屏时有效。

隐藏状态栏和导航栏在应用 app 里面向系统发送广播

gzpeite.intent.systemui.hidenavigation 和 gzpeite.intent.systemui.hidestatusbar

显示状态栏和导航栏在应用 app 里面向系统发送广播

gzpeite.intent.systemui.shownavigation 和 gzpeite.intent.systemui.showstatusbar

测试命令如下:

am broadcast -a "gzpeite.intent.systemui.hidenavigation"

am broadcast -a "gzpeite.intent.systemui.hidestatusbar"

am broadcast -a "gzpeite.intent.systemui.shownavigation" am broadcast -a "gzpeite.intent.systemui.showstatusbar"

## 八、静默安装/卸载应用

安装 APK 时,向系统发送 gzpeite.intent.action.install\_apk 广播

卸载 APK 时,向系统发送 gzpeite.intent.action.uninstall\_apk 广播 测试命令如下:

am broadcast -a "gzpeite.intent.action.install\_apk" --es apk\_path "/mnt/media\_rw/0000-4823/GPSTest.apk" am broadcast -a "gzpeite.intent.action.uninstall\_apk" --es pkg\_name "com.android.gpstest"



## 九、编译 Linux + QT5.9

注意 Linux+QT 环境下不支持 opengl,不支持视频硬件编解码,如果需要进行视频播放或复杂的图形显示等应用,建议用安卓系统或 QT for Android <u>https://doc.qt.io/qt-5/android.html</u> 请首先新开一个控制台进行编译操作。

首次编译请严格按照步骤进行内核、uboot、Rootfs 的编译,否则编译可能会出现错误。

#### 1、编译内核

cd lichee ./build.sh -p sun50iw1p1\_dragonboard -k linux-3.10 编译完成后正确提示如下 Copy boot.img to output directory ... ./scripts/build.sh sun50iw1p1 compile Kernel successful INFO: build rooffs ... Regenerating dragonboard Rootfs... extract dragonboard rootfs.tar.xz INFO: Prepare dragonboard toolchain ... cp: cannot stat 'extra/\*': No such file or directory generating rootfs... blocks: 695M -> 1024M Creating filesystem with parameters: size: 1073741824 Block size: 4096 Blocks jeer group: 32768 Inodes per group: 3192 Inode size: 256 Journal blocks: 4096 Label: Blocks: 262144 Block groups: 8 Reserved block group size: 63 Created filesystem with 14921/65536 inodes and 186636/262144 blocks e2fsck 1.44.1 (24-Mar-2018) fsck.ext4: Bad magic number in super-block while trying to open rootfs.ext4 success in generating rootfs Build at: sun Apr 7 20:29:02 CST 2019 INFO: build ricke ok. INFO:

#### 2、编译 uboot

首次编译或修改 uboot 代码后需要执行这一步骤。

首先切换到 uboot 目录 cd lichee/brandy ./build.sh -p sun50iw1p1 编译完成后正确提示如下



/root/work/A64\_7.1.1\_skD\_BASE/lichee/brandy/u-boot-2014.07/../gcc-linaro, opy -0 binary /root/work/A64\_7.1.1\_SKD\_BASE/lichee/brandy/u-boot-2014.07 14\_sunxi\_spl/sunxi\_spl/sbrom/sboot.axf /root/work/A64\_7.1.1\_SKD\_BASE/lich ./../bootloader/uboot\_2014\_sunxi\_spl/sunxi\_spl/sbrom/sboot.bin /root/work/A64\_7.1.1\_SKD\_BASE/lichee/brandy/u-boot-2014.07/../../bootloader/uboot\_2014\_sunxi\_spl/sunxi\_spl/sbrom/sboot.bin /root/work/A64\_7.1.1\_SKD\_BASE/lichee/brandy/u-boot-2014.07/../../bootloader/uboot\_2014\_sunxi\_spl/sunxi\_spl/sbrom/sboot.bin /root/work/A64\_7.1.1\_SKD\_BASE/lichee/brandy/u-boot-2014.07/../../bootloader/uboot\_2014\_sunxi\_spl/sunxi\_spl/sbrom/sboot.bin -m sboot /bootloader/uboot\_2014\_sunxi\_spl/sunxi\_spl/sbrom/sboot.bin -m sboot build\_sboot 'sboot\_sun50iw1p1.bin' -> '/root/work/A64\_7.1.1\_SKD\_BASE/lichee/brandy/uack/chips/sun50iw1p1/bin/sboot\_sun50iw1p1.bin' root@SeKeDe:~/work/A64\_7.1.1\_SKD\_BASE/lichee/brandy#

#### 3、编译 QT Rootfs

首先切换到 pack 目录
cd lichee/tools/pack/
./pack -c sun50iw1p1 -p dragonboard -b p3
编译完成后正确提示如下
<pre>% 倖元成后止佣提示如下 config.fex Len: 0xc400 sys_partition.fex Len: 0xaa0 sunxi.fex Len: 0x14000 boot0_nand.fex Len: 0x8000 u-boot.fex Len: 0x8000 u-boot.fex Len: 0x8000 toc0.fex Len: 0x8 fes1.fex Len: 0x8 fes1.fex Len: 0x4100 boot_package.fex Len: 0x130000 usbtool.fex Len: 0x23600 aultools.fex Len: 0x247b aultols.fex Len: 0x24423 cardtool.fex Len: 0x41600 cardscript.fex Len: 0x4000 arisc.fex Len: 0x4000 dlinfo.fex Len: 0x4000 arisc.fex Len: 0x44 boot-resource.fex Len: 0x4 env.fex Len: 0x4 boot-fex Len: 0x4 boot-f</pre>
<pre>/root/Work/A64_7.1.1_SKD_BASE/lichee/tools/pack/sun50iw1p1_dragonboard_p3_uart0.img</pre>
pack finish

编译完成后会在 lichee/tools/pack 目录下生成 sun50iw1p1\_dragonboard\_p3\_uart0.img 系统烧写镜像文件。

#### 4、修改 Rootfs

完成首次编译后,rootfs的所有文件位于 lichee\buildroot\target\dragonboard\rootfs 目录下。 如果需要修改或添加文件,需要将文件复制到 lichee\buildroot\target\dragonboard\extra 相同目录下, 然后再修改,重新编译内核、uboot、rootfs 即可。



例如需要修改 rootfs/etc/init.d/S00peite 这个系统初始化设置脚本文件 cd lichee/buildroot/target/dragonboard mkdir -p extra/etc/init.d cp -rf rootfs/etc/init.d/S00peite extra/etc/init.d/S00peite 修改 extra/etc/init.d/S00peite 后重新编译即可生成新的烧写镜像文件

#### 5、更换 Rootfs 为 Linux 或 Linux + QT

首先删除 lichee\buildroot\target\dragonboard\rootfs 目录。 将开发资料《源代码》目录下 rootfs 文件更名为 rootfs.tar.xz 复制 rootfs.tar.xz 到 lichee\buildroot\target\dragonboard 覆盖同名文件 重新编译即可 Rootfs 类型有: Linux\_Full --- Linux 全功能版,不包含 QT Linux\_Lite --- Linux 部分功能版,不包含 QT QT\_Full --- Linux+QT 全功能版 QT\_Lite --- Linux+QT 部分功能版

#### 6、交叉编译其他应用

系统使用的交叉编译器位于开发资料《开发工具/交叉编译器》目录下:

gcc-linaro-7.2.1-2017.11-x86\_64\_aarch64-linux-gnu.tar.xz

编译所需的其他库文件是 sysroot\_peite.tar.xz(交叉编译器目录内)可根据需要进行解压使用,客户可自行编译其他未包含的支持库、应用程序等。

## 十、修改 Linux 内核编译选项

首先切换到 linux 内核目录 cd lichee/linux-3.10/ 加载默认配置 make sun50iw1p1smp\_android\_defconfig

启动内核配置 make menuconfig



Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><to exit, <?> for Help, </> for Search. Legend: [\*] built-in [] excluded <M> module <> module capable General setup ---> Enable loadable module support Enable the block layer ---> Platform selection ---> us support ---ernel Features Event Features ---> Boot options ---> Userspace binary formats Power management options CPU Power Management ----[\*] Networking support ---> Device Drivers ---> irmware Drivers **Г** ] <select> < E < Н < 5 < L 修改内核选项时不要选择编译成模组文件,可以选择直接编译进内核。 完成配置后保存退出, 将内核根目录下的 .config 文件复制保存为 arch/arm64/configs/sun50iw1p1smp\_android\_defconfig 此步骤非常重要,如果不执行的话会自动恢复为默认配置。 cp.config arch/arm64/configs/sun50iw1p1smp android defconfig

完成内核配置修改后,从新编译 android 或 linux 即可。 注意 Android 和 Linux 共用同一个默认配置,修改内核选项对 android 和 Linux+QT 同时生效。

## 十一、镜像文件烧写

开发过程中,一般使用 PhoenixSuit 进行镜像文件的烧写,具体操作方式请参考开发工具目录下的 《PhoenixSuit 使用说明文档.pdf》,除了 Android 系统我司的 Linux+QT 系统也支持这种烧写方式。

将开发板的 MicroUSb 接口连接到系统主机后, Linux+QT 系统检测到的设备信息如下:





烧写操作需要首先通过 Micro USB 数据线连接主机的开发板,在进行烧写时如果出现主机识别到新的 设备没有正常安装驱动的情况时,需要手动安装设备驱动程序,驱动程序位于开发工具文件夹内。

注意,在点击烧写镜像后,设备会重启黑屏,如果没有开始烧写进程,此时需要在 PC 端的设备管理 区中对黄色感叹号设备手动安装镜像烧写设备驱动 (AW\_Driver)。

## 十二、建立 QT 应用程序编译环境

所需工具位于开发资料的《开发工具/QT》目录下:

- 1、解压交叉编译器 gcc-linaro-7.2.1-2017.11-x86\_64\_aarch64-linux-gnu.tar.xz sudo tar -xJf gcc-linaro-7.2.1-2017.11-x86\_64\_aarch64-linux-gnu.tar.xz -C /usr/local
- 2、解压库文件 sysroot\_peite\_qt.tar.xz sudo tar -xJf sysroot\_peite\_qt.tar.xz -C /usr/local
- 3、解压安装 qt-creator-opensource-linux-x86\_64-4.4.0.tar.xz

chmod +x qt-creator-opensource-linux-x86\_64-4.4.1.run ./qt-creator-opensource-linux-x86\_64-4.4.1.run



Qt Creator 4.2.1 Setup

#### Qt Account - Your unified login to everything Qt

		Please log in to Qt Account
	Login	Email
		Password
		Forgot password?
		Need a Qt Account?
	Sign-up	Valid email address
		Password
Create Orace		Confirm Password
Create Once.		I accept the <u>service terms</u> .
Deploy Everywhere.		
Settings		< <u>B</u> ack Skip Cancel

注意在上面这一步选择 Skip,其他直接选择 Next 即可

- 4、 启动 qt creator 设置交叉编译器和 QT 库文件路径。
- 5、 /opt/qtcreator-4.4.1/bin/qtcreator



			Qt Creato	r	
<u>F</u> ile <u>E</u> dit	<u>B</u> uild <u>D</u> ebug <u>A</u> nalyze <u>T</u> ools	<u>W</u> indow <u>H</u> elp			
Welcome	Projects	+ New Project		🔄 Open Project	
Edit	Examples	Sessions		Recent Projects	
Design	Tutorials	1 🕨 default			
<b>W</b> Debug	New to Qt?				
م √ Projects	Learn how to develop your own applications and explore Qt Creator.				
? Help	Get Started Now				
	1 Qt Account				
	Online Community				
	Blogs				
	🕜 User Guide				
Sik					
	Ω. Type to locate <i>(C</i> trl	1 Issues - 2 Search Pesults	3 Application Output	1 Compile Output 5	Dabuagar Cancola 🔺

选择菜单 Tools->Options->Build & Run->Compilers, 点击 Add ->Custom->C++ 按钮, 添加 C++编译器, Complier path:

/usr/local/arm/gcc-linaro-7.2.1-2017.11-x86\_64\_aarch64-linux-gnu/bin/ aarch64-linux-gnu-g++



٥r	Build & Run					
Environment	General Kits Qt	Versions Compiler	s Debuggers	Qbs CMake		
1	Name		Туре			Add
Text Editor	<ul> <li>Auto-detected</li> </ul>					Clana
FakeVim	GCC (C++, GCC (C++,	x86 64bit in /usr/bin) x86 32bit in /usr/bin)	GCC GCC			Remove
Help	Clang (C++ Clang (C++ VC	, x86 64bit in /usr/bin , x86 32bit in /usr/bin	) Clang ) Clang			
¢ C++	GCC (C, x86 GCC (C, x86 Class (C, x86	64bit in /usr/bin) 32bit in /usr/bin) 26.64bit in /usr/bin)	GCC GCC			
Qt Quick	Clang (C, x8 Clang (C, x8	36 32bit in /usr/bin)	Clang			
Build & Run	▼ C++ PEITE-G++ ▼ C		Custom			
Debugger	PEITE-GCC		Custom			
Designer					<b>A</b>	
	Name:	PEITE-G++				
Analyzer	<u>C</u> ompiler path:	7.11-x86_64_aarch64-	linux-gnu/bin/aaro	h64-linux-gnu-g++	Browse	
Version Control	<u>M</u> ake path:	/usr/bin/make			Browse	
Devices	<u>A</u> BI:	arm 🔻 - linux	🔻 - generie	: • elf •	- 64bit 💌	
Code Pasting	Predefined macros:	Empty			Details 🔻	
Test Settings	<u>H</u> eader paths:	Empty			Details 🔻	
-	C++11 <u>f</u> lags:					

Add ->Custom->C 添加 C 编译器 Complier path:

/usr/local/arm/gcc-linaro-7.2.1-2017.11-x86\_64\_aarch64-linux-gnu/bin/ aarch64-linux-gnu-gcc



neral Kits Q	t Versions	Compilers	Debuggers	Qbs	CMake			
ame		Т	уре				Add	
Auto-detected							Clone	
GCC (C++,	. x86 64bit in	n/usr/bin) (	GCC				Cione	-
GCC (C++	GCC (C++, x86 32bit in /usr/bin)						Remov	/e
Clang (C+-	+, x86 64bit	in /usr/bin) (	Clang				·	
Clang (C+-	+, x86 32bit	in /usr/bin) (	Clang					
GCC (C. x8	6 64bit in /us	sr/bin) (	GCC					
GCC (C, x8	6 32bit in /us	sr/bin) (	GCC					
Clang (C, x	86 64bit in /	usr/bin) (	Clang					
Clang (C, x	«86 32bit in /	'usr/bin) (	Clang					
Manual								
PEITE-G++		0	Custom					
🕶 C								
PEITE-GCC			Custom					
							<b>^</b>	
Name:	PEITE-GCC							
Name: <u>C</u> ompiler path:	PEITE-GCC	54_aarch64-li	nux-gnu/bin/aa	rch64-lir	nux-gnu-gcc	Browse		
Name: <u>C</u> ompiler path: Make path:	PEITE-GCC	54_aarch64-li ake	nux-gnu/bin/aa	rch64-lir	nux-gnu-gcc	Browse		
Name: <u>C</u> ompiler path: <u>M</u> ake path:	PEITE-GCC 17.11-x86_6 /usr/bin/ma	54_aarch64-li ake	inux-gnu/bin/aa	rch64-lir	nux-gnu-gcc	Browse Browse		
Name: Compiler path: Make path: ABI:	PEITE-GCC 17.11-x86_6 /usr/bin/ma arm	54_aarch64-li ake • - linux	nux-gnu/bin/aa	rch64-lir c 💌 -	nux-gnu-gcc elf	Browse Browse		
Name: Compiler path: Make path: ABI: Predefined macros:	PEITE-GCC 17.11-x86_6 /usr/bin/ma arm Empty	54_aarch64-li ake • - Iinux	nux-gnu/bin/aa	rch64-lir c v-	elf v	Browse Browse - 64bit Details		
Name: Compiler path: Make path: ABI: Predefined macros: Header paths:	PEITE-GCC 17.11-x86_f /usr/bin/ma arm Empty Empty	54_aarch64-li ake 👻 - [linux	nux-gnu/bin/aa	rch64-lin c v	elf v	Browse Browse - 64bit Details		

选择菜单 Tools->Options->Build & Run->Qt Versions,点击 Add 按钮,配置如下:

	3		Debuggers	Qbs	CMake		
ame	+ qmake	Location					Add
Manual							Remove
Qt 5.9.6 (0	Qt-5.9.6) /usr/lo	ical/arm/sysro	ot_peite_qt/us	r/local/Q	t-5.9.6/bin/	qmake	
							Clean Up
		(0. 5.0.5					
/ersion name:	Qt %{Qt:Vers	ion} (Qt-5.9.6	)				
/ersion name: 1make location	Qt %{Qt:Vers	ion} (Qt-5.9.6 /sysroot_peite	) e_qt/usr/local/C	)t-5.9.6/	pin/qmake	Browse	

选择菜单 Tools->Options->Devices,点击 Add 按钮,配置如下:



Devic	e Configuration Wizard Selection			
Available device t	ypes: vice			
首先需要将开发板与 Connection	主机在同一局域网内连接好,主机可以了 New Generic Linux Device Conf	正常 ping 通开发板 iguration Setup		
Connection Summary	The name to identify this configuration: The device's host name or IP address: The username to log into the device: The authentication type: The user's password: The file containing the user's private key:	Generic Linux Device 192.168.1.85 root Password O Key /root/.ssh/id_rsa	Browse	
			Next >	Cancel

测试通过后的配置如下:



Devices Android	QNX		
Device: Generic Lin	ux Device (default for Generic Linux)	•	<u>A</u> dd
<u>N</u> ame:	Generic Linux Device		<u>R</u> emove
Туре: О	Generic Linux		Set As Default
Auto-detected: Current state:	lo Jnknown		Test
Type Specific			Show Running Processes
Machine type:	Physical Device		Deploy Public Key
Authentication ty	pe: Password O Key		
<u>H</u> ost name:	2.168.1.85 SSH port: 22		
Free ports:	10000-10100 Timeout: 10s 🗘		
Username:	root		
Password:	Show password		
Private key file:	Browse Create New	Ŧ	
		Apply	y Cancel OK

选择菜单 Tools->Options->Build & Run->Kits, 点击 Add 按钮, 配置如下:



Build & Run							
General Kits	Qt Versions	Compilers	5 Debuggers	Qbs	CMake		
Name							Add
Auto-detecte ▼ Manual							Clone
Desk PEITE	top (default)						Remove
							Make Default
Name:		PEITE					₽.
File system nar	me:						
Device type:		Generic Li	nux Device			•	
Device:		Generic Li	nux Device (defa	ault for O	eneric Linux)	•	Manage
Sysroot:		/usr/local/a	arm/sysroot_pei	te_qt/usr	/local/Qt-5.9.6		Browse
Compiler		C: PEITE-GCC -				•	Managa
compiler.		C++: PEITE-G++					Manage
Environment:		No changes to apply.					Change
Debugger:		System GDB at /usr/bin/gdb				Manage	
Qt version:		Qt 5.9.6 (Qt-5.9.6)				Manage	
Qt mkspec:							
CMake Tool:		System C	/lake at /usr/bin/	cmake		•	Manage
CMake generat	or:	CodeBlocks	- Unix Makefile:	s, Platfor	m: <none>, Too</none>	lset: <none></none>	Change
4					Jan	No. 1	
				L	¢₽ОК	X Cancel	✓ Apply



# 十三、创建并编译 QT 程序

		New Project	
Choose a template	:		All Templates 👻
Projects Application Library Other Project Non-Qt Project Import Project Files and Classes		<ul> <li>Qt Widgets Application</li> <li>Qt Console Application</li> <li>Qt Quick Application</li> <li>Qt Quick Controls 2 Application</li> <li>Qt Quick Controls Application</li> <li>Qt Quick Controls Application</li> <li>Qt Canvas 3D Application</li> </ul>	Creates a Qt application for the desktop. Includes a Qt Designer- based main window. Preselects a desktop Qt for building the application if available. Supported Platforms: Generic Linux Device
			<u>C</u> ancel Choose
		Qt Widgets Application	
Nocation     Kits     Details     Summary	Introduction and This wizard generates and includes an empty	<b>Project Location</b> a Qt Widgets Application project. The application widget.	tion derives by default from QApplication
	Name: peite_test Create in: /root Use as default pro	ject location	Browse
			<u>N</u> ext > Cancel



		Qt Widgets Application					
Location	Kit Selecti	on					
Kits	Qt Creator can use the following kits for project peite_test:						
Details	✓ Select all ki	ts					
Summary	V 🖵 EPITE			Details 🔺			
	✓ Debug	/root/build-peite_test-EPITE-Debug		Browse			
	✓ Release	/root/build-peite_test-EPITE-Release		Browse			
	✓ Profile		Browse				
			< <u>B</u> ack Next	:> Cance			
		Qt Widgets Application					
Location	Class Infor	mation					
Kite	Specify basic in	formation about the classes for which you way	nt to generate skeleton source co	de files.			

🔊 Details		
Summary	<u>C</u> lass name:	MainWindow
	<u>B</u> ase class:	QMainWindow -
	<u>H</u> eader file:	mainwindow.h
	<u>S</u> ource file:	mainwindow.cpp
	<u>G</u> enerate form:	
	Form file:	mainwindow.ui
		< <u>B</u> ack <u>N</u> ext > Cancel



		Qt Widgets Application
Location	Project Management	
Kits	Add as a subproject to project	: <none></none>
Details ∞∑Summarv	Add to <u>v</u> ersion control:	<none>  Configure</none>
,	Files to be added in	
	/root/peite_test:	
	main.cpp mainwindow.cpp mainwindow.h mainwindow.ui peite_test.pro	
		< Back Finish Cancel
元成切建向,需 target.pat INSTALLS	局安修仪工作日求下的 pe th = /root += target	te_test.pro 文件, 在文件取应称加下面网行代码 peite_test.pro - peite
<u>File</u> Edit B	uild <u>D</u> ebug <u>A</u> nalyze ]	Cools Window Help
Pro	o ≑  T. ⊕ H+ ⊑ <	> 🖬 🔂 peite_test.pro 🗧 🗢 🗧
	peite_test.pro	TARGET = peite_test TEMPLATE = app
vveicome	Headers	The following define makes your compiler emit warning
Edit	Forms 15 #	any feature of Qt which as been marked as deprecated depend on your compiler). Please consult the document
<u>p</u>	17 # 18 0	<pre># deprecated API in order to know how to port your code DEFINES += QT_DEPRECATED_WARNINGS</pre>
Design	19 20 <b>#</b>	You can also make your code fail to compile if you us
<b>₩</b>	21 # 22 #	f In order to do so, uncomment the following line. f You can also select to disable deprecated APIs only u
Debug	23 <b>#</b> 24	DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000 #
J.	25 26 S	SOURCES += main.cpp\
Projects	27 28	mainwindow.cpp
Help	29 H 30	HEADERS += mainwindow.h
	31 F 32	ORMS += mainwindow.ui
	33 t 34 J	arget.path = /root INSTALLS += target
	35	

然后在 QT Creator 中重新打开工程,编译、运行后,可以在开发板上查看运行效果。



## 十四、定制编译 QT 源码

客户可以自行编译 QT 的源码,可以对 QT 源码进行修改及定制,以下执行步骤需要 ROOT 权限。

- 1、复制 QT\_Source 目录及所有文件到编译主机。
- 2、进入 QT\_Source 目录,运行 config.sh 进行编译环境准备及选项配置。
- 3、运行 build.sh 编译
- 4、编译完成后的 QT 安装目录为 /usr/local/arm/sysroot\_peite\_qt/usr/local/Qt-5.9.6

## 十五、联系方式

- 地址 : 广州市天河区大观中路新塘大街鑫盛工业园 A1 栋 201
- 电话 : 020-85625526
- 传真 : 020-85625526-606
- 主页 : <u>http://www.gzpeite.net</u>
- 淘宝店 : <u>https://shop149045251.taobao.com</u>
- 核心板 : 王先生
- 移动电话: 18926288206
- 电子信箱: 18926288206@gzpeite.net
- 业务 QQ: 594190286
- 定制研发:杨先生
- 微信: 18902281981
- 电子信箱: 18902281981@gzpeite.net
- 业务 QQ: 151988801

广州佩特电子科技有限公司

#### 2020年8月