龙芯2K1000LA用户手册

目录

快速开始

本文是龙芯2K1000LA的用户手册，和该手册一起的包括：PMON源码、U-boot源码、Linux内核源码、文件系统、交叉工具链、EJTAG软件、龙芯2K1000LA处理器用户手册。本文主要说明上述软件及源码的使用、编译。

环境搭建

pmon编译环境搭建

将源码压缩包拷贝到工作目录，使用下述命令对源码进行解压。（注：如果使用的工作环境是虚拟机，请不要直接在共享文件夹下进行解压）

```
tar -zxvf pmon-ls2k1000la.tar.gz
```

将交叉工具链拷贝到常用目录，并解压：

```
tar -zxvf loongarch64-linux-gnu-2021-12-10-vector.tar.gz
```

安装编译依赖：

```
sudo apt install aptitude xutils-dev bison flex acpica-tools
```

进入pmon源码编译pmoncfg工具：

```
cd PMON源码/tools/pmoncfg
make pmoncfg
sudo cp pmoncfg /usr/bin
```

编译PMON：

```
 cd zloader.ls2k/

 make cfg all tgt=rom CROSS_COMPILE=/opt/loongarch64-linux-gnu-
2021-12-10-vector/bin/loongarch64-linux-gnu- DEBUG=-g
 make dtb CROSS_COMPILE=/opt/loongarch64-linux-gnu-2021-12-10-
vector/bin/loongarch64-linux-gnu-
```

> *编译选项解释：*
>
> *make cfg 对pmon进行配置；*
>
> *all为Makefile里的编译项；*
>
> *tgt=rom，指定tgt为rom，则会生成gzrom.bin文件；*
>
> *CROSS_COMPILE=loongarch64-linux-gnu-，指定编译工具前缀名；*
>
> *DEBUG=-g，设置编译的时候携带调试信息。*
>
> *make dtb , 编译设备树*

编译脚本如下，在PMON源码目录下执行：

```
vim cmd.sh
```

输入下面内容

```
 #!/bin/sh
 cd zloader.ls2k/

 make cfg all tgt=rom CROSS_COMPILE=/opt/loongarch64-linux-gnu-
2021-12-10-vector/bin/loongarch64-linux-gnu- DEBUG=-g
 make dtb CROSS_COMPILE=/opt/loongarch64-linux-gnu-2021-12-10-
vector/bin/loongarch64-linux-gnu-
 cp gzrom-dtb.bin ../
 cp gzrom.bin ../
 cp ls2k.dtb ../
```

保存退出后，执行

```
chmod 777 cmd.sh
./cmd.sh
```

在PMON源码目录生成bin文件和dtb文件，其中gzrom-dtb.bin带dtb，gzrom.bin不带dtb，ls2k.dtb为单独的dtb文件。

编译龙芯派PMON

```
 #!/bin/sh
 cd zloader.ls2k/
sed -i "2c TARGETEL=ls2k_lspi" ./Makefile.ls2k
 make cfg all tgt=rom CROSS_COMPILE=/opt/loongarch64-linux-gnu-2021-12-10-vector/bin/loongarch64-linux-gnu- DEBUG=-g
 make dtb CROSS_COMPILE=/opt/loongarch64-linux-gnu-2021-12-10-vector/bin/loongarch64-linux-gnu-
 cp gzrom-dtb.bin ../
 cp gzrom.bin ../
 cp ls2k.dtb ../
 sed -i "2c TARGETEL=ls2k" ./Makefile.ls2k
```

然后执行上面的PMON编译脚本，上面的命令将用`Targets/ls2k/conf/ls2k_lspi`作为配置文件，`Targets/ls2k/conf/ls2k_lspi.dts`作为dts文件

适配新板卡

手动适配

首先生成自己板卡的配置（可以参ls2k\ls2k_pai\ls2k.dts\ls2k_pai.dts），主要配置如下：

1. 内存配置

   AUTO_DDR_CONFIG :内存条需要选上该选项；没有spd(直接用内存颗粒)需要手动配置，不打开该选项

DDR_S1：关闭 AUTO_DDR_CONFIG 时需要根据具体的内存型号手动配置 s1 的值，具体配置参考《内存调试手册》

CONFIG_DDR_32BIT：32位内存配置

CONFIG_DDR_16BIT：16位内存配置

DDR_RESET_REVERT：内存复位外部无反向时选择

2. 外设配置

3. 服务配置

其他配置参考PMON手册中的配置文件说明

EJATG环境搭建

下载对应系统的ejatg软件包，对应关系如下：

```
x86             linux       ejtag-debug-$date.tar.gz
x86             windows     ejtag-debug-mingw-$date.tar.gz
mips            linux       ejtag-debug-mips64-$date.tar.gz
LoongArch       linux       ejtag-debug-la64-$date.tar.gz
```

解压软件包,并安装驱动：

```
#linux系统下执行命令，完成安装：
tar zxvf ejtag-debug-$date.tar.gz
#windows下直接右键解压，然后参考解压出来的文件夹中的doc/ejtag1.pdf安装驱动
```

拷贝配置文件，2022年6月之前的ejtag软件需要拷贝软件资料中的ejtag/config.ls2k到安装目录/config/config.ls2k;

运行程序：

```
#linux系统下执行命令，运行longarch程序（mips芯片仍用原来的方式启动）：
sudo ./la_dbg_tool_usb -t
#windows下直接双击la_dbg_tool_usb启动
```

配置ejtag：

```
#进入ejtag后执行:
source configs/config.ls2k
```

Linux编译环境搭建

安装编译依赖:

```
sudo apt install libncurses5-dev libssl-dev
```

指定交叉工具链:

```
export PATH=/opt/loongarch64-linux-gnu-2021-12-10-vector/bin:$PATH
```

采用2K1000的配置文件

```
cp arch/loongarch/configs/ls2k1000_defconfig .config
make menuconfig ARCH=loongarch
```

编译内核:

```
make vmlinuz ARCH=loongarch CROSS_COMPILE=loongarch64-linux-gnu- -j
4
```

编译完成后，会在当前目录下看到生成的vmlinux文件，与压缩后的内核文件vmlinuz。

编译脚本mymake如下:

```
#!/bin/sh
export LC_ALL=C LANG=C
export PATH=/opt/loongarch64-linux-gnu-2021-12-10-vector/bin:$PATH
make vmlinuz ARCH=loongarch CROSS_COMPILE=loongarch64-linux-gnu- -j
4 "$@"
```

编译时执行:

```
./mymake menuconfig
./mymake vmlinuz
```

buildroot文件系统制作

(1) 解压buildroot-la.tar.gz文件

tar -zxvf buildroot-la.tar.gz

(2) 设置编译配置文件

cp config-la.my .config

或者cp configs/loongisa_defconfig .config

(3) 指定交叉编译工具链

export PATH=/opt/loongarch_toolchain/bin:$PATH

(4) 打开图形化配置界面，并保存退出

make menuconfig ARCH=loongarch64

(5) 联网编译文件系统

make ARCH=loongarch64 CROSS_COMPILE=loongarch64-linux-gnu- -j4 "$@"

(6) 文件系统镜像

编译完成后，在buildroot源码的output/images/目录下会生成文件系统镜像文件。

Yocto文件系统制作

1. 解压yocto-la.tar.gz

   tar zxvf yocto-la.tar.gz

2. 解压交叉工具链

   sudo tar zxvf loongarch-toolchain.tar.gz

3. 进入yocto目录，执行下面的命令

   cd yocto

   . oe-init-build-env

4. 编译文件系统，执行下面的命令

   bitbake core-image-minimal

编译自己的应用程序

编译C/C++语言程序

```
#安装工具链
sudo ./loongarch64-toolchain.sh
#安装过程选择默认位置，第二个选择y即可
#设置环境变量
source /opt/poky/3.3+snapshot/environment-setup-loongarch64-poky-
linux
#编译
$CC -o hello hello.c
```

编译QT程序

资料包提供了编译应用程序的交叉工具链，主要针对QT应用

下载loongarch64-toolchain.sh,并执行,具体流程如下：

```
./loongarch64-toolchain.sh
source /opt/poky/3.3+snapshot/environment-setup-loongarch64-poky-
linux
#查看qmake版本
qmake -v
#进入需要编译的程序
qmake
make
```

编译其他程序，直接编译即可,如：

```
$CC -o hello hello.c
```

PMON烧写

pmon支持多种烧写方式，具体如下：

1. u盘更新

   插入u盘，上电，进入PMON，在pmon shell里输入如下命令

```
fload (usb0,0)/gzrom-dtb.bin    #gzrom-dtb.bin在u盘根目录，也可以加路径/gzrom-dtb.bin
```

2. 网络更新

需要有tftp服务器，进入PMON，在pmon shell里输入如下命令(假设用2K上的GMAC0口更新)

```
#自动获取ip，需要有dhcp服务器
lwdhcp
#手动设置ip
ifaddr syn0 ip
#更新pmon
fload tftp://server-ip/gztom-dtb.bin
```

3. 调试器更新

```
进入调试器目录，输入如下命令:
sudo ./la_dbg_tool_usb -t
source configs/config.ls2k
set     #先set后上电，set返回寄存器的值，其中pc需要时0x1c000000，如下图
program_cachelock ./gzrom-dtb.bin
```

启动Linux

手动启动

进入pmon命令行，依次输入如下命令:

```
load (wd0,0)/vmlinuz                        #加载内核
initrd (wd0,0)/rootfs.cpio.gz               #加载文件系统
g console=ttyS0,115200 rdinit=/sbin/init    #启动linux
```

自动启动

1. boot.cfg

PMON启动最后会去常用存储设备的boot目录找boot.cfg文件，并按照boot.cfg 的参数启动，例：

```
timeout 3
default 0
showmenu 1

title 'LoongOS power test'
        kernel (wd0,0)/boot/vmlinuz_test
        args  console=tty console=ttyS0,115200 root=/dev/sda1
mytest=power

title 'LoongOS '
        kernel (wd0,0)/boot/vmlinuz_2kla
        args  console=tty console=ttyS0,115200 root=/dev/sda1
loglevel=8

title 'LoongOS reboot test'
        kernel (wd0,0)/boot/vmlinuz_2kla
        args  console=tty console=ttyS0,115200 root=/dev/sda1
loglevel=8 mytest=reboot
```

2. 环境变量

如果没有boot.cfg,PMON会执行环境变量autocmd提供的命令，设置autocmd如下：

```
set autocmd "load (wd0,0)/vmlinuz;initrd
(wd0,0)/rootfs.cpio.gz;g console=ttyS0,115200
rdinit=/sbin/init"
```

如果没有设置autocmd或者设置的不是启动linux的命令，PMON会加载如下环境变量，并按照环境变量启动内核

```
#rd为文件系统路径；al1为内核路径；append为启动参数
set rd (wd0,0)/boot/rootfs.cpio.gz
set $al1 (wd0,0)/boot/vmlinuz
set append "g console=ttyS0,115200 rdinit=/sbin/init"
```

本手册导航

PMON源码的使用需要参考这里，其中包括PMON编译相关、PMON配置相关、DTS相关、PMON使用手册。

PMON手册

PMON编译

编译流程如下面的脚本

编译脚本

```
 #!/bin/sh
 cd zloader.ls2k/

 make cfg all tgt=rom CROSS_COMPILE=/opt/loongarch64-linux-gnu-
2021-12-10-vector/bin/loongarch64-linux-gnu- DEBUG=-g
 make dtb CROSS_COMPILE=/opt/loongarch64-linux-gnu-2021-12-10-
vector/bin/loongarch64-linux-gnu-
 cp gzrom.bin /home/tftpshare/
 cp gzrom.bin ../
 cp gzrom.bin /opt/ejtag-debug/
 cp gzrom-dtb.bin ../
```

修改配置文件：

指定采用龙芯派的配置文件和dts

```
sed -i "2c TARGETEL=ls2k_lspi" ./Makefile.ls2k  #指定配置文件，默认
ls2k,可以生成自己的配置文件
```

用自己的配置文件：

配置文件

PMON默认带有多款板卡的配置文件供用户选择,

DTS

目录结构

PMON命令

ifconfig命令

```
ifconfig syn0 readphy [phybase]    #读gmac phy寄存器
ifconfig syn0 readphy 0
ifconfig syn0 writephy offset:data,data  #写gmac phy 寄存器
ifconfig syn0 writephy 0x10:0x55,0xaa
```

date命令

```
date [yyyymmddhhmm.ss]   #读或写RTC
date
date 202205132200.00
```

bl命令

bl -d ide (wd0,0)/boot/boot.cfg

fload命令

fload 是更新BIOS的命令，命令格式为：`fload  path/gzrom-dtb.bin`。 例如网络更新：`fload tftp://192.168.1.11/gzrom-dtb.bin`,其中`92.168.1.11`为tftp服务器地址。

initrd命令

lwdhcp命令

lwdhcp是自动获取ip地址的命令，需要有dhcp服务器分配地址。命令格式：`lwdhcp syn0`,其中syn0是网卡的节点，可以通过**devls**命令查看所有网卡节点。

PMON源码

FileSystem包含（netfs）

read->netread->tftpread

dotik函数会打印|/-

mul_pin_def_cfg 引脚复用配置

mem_win_cfg 窗口配置

PMON常见用法

网络启动&PXE

pmon下设置环境变量

```
set bootdev tftp://10.50.122.143/pxeboot.cfg
```

启动后会从网络加载启动菜单。

设置MAC地址

1. 环境变量设置

   配置文件选上`option USE_ENVMAC`，pmon启动后在命令行设置环境变量：`set ethaddr 00:11:22:33:44:55`

2. EEPROM存储MAC地址

   eeprom存储mac地址, pmon启动自动读取eeprom内存储的mac地址，如果没有存或者存的值不符合要求，pmon会生成随机地址

pmon下设置mac地址的命令：`setmac syn1 "00:11:22:33:44:55"`

pcie bar

NAND使用

Yaffs2文件系统

用mkyaffs2image工具制作yaffs2文件系统镜像，源码如下：
http://sources.buildroot.net/yaffs2utils/0.2.9.tar.gz
下载完，解压0.2.9.tar.gz
进入目录下编译： make
编译出需要的工具 mkyaffs2、unyaffs2

```
sudo chmod 777 mkyaffs2
sudo chmod 777 unyaffs2
sudo cp  mkyaffs2  unyaffs2  /usr/bin/
sudo  mkyaffs2
mkyaffs2 0.2.9 - A utility to make the yaffs2 image

Usage: mkyaffs2 [-h|--help] [-e|--endian] [-v|--verbose]
                [-p|--pagesize pagesize] [-s|sparesize sparesize]
                [-o|--oobimg oobimage] [--all-root] [--yaffs-
ecclayout]
                dirname imgfile

Options:
  -h                 display this help message and exit.
  -e                 convert endian differed from local machine.
  -v                 verbose details instead of progress bar.
  -p pagesize        page size of target device.
                     (512|2048(default)|4096|(8192|16384) bytes)
  -s sparesize       spare size of target device.
                     (default: pagesize/32 bytes; max: pagesize)
  -o oobimage        load external oob image file.
  --all-root         all files in the target system are owned by
root.
  --yaffs-ecclayout  use yaffs oob scheme instead of the Linux MTD
default.
```

制作文件系统镜像，如下：

```
mkyaffs2 -p 4096 -s 128 --yaffs-ecclayout rootfs/ rootfs.img
```

PMON下烧写内核和文件系统镜像：

```
PMON> mtd_erase /dev/mtd0
PMON> mtd_erase /dev/mtd1
PMON> devcp tftp://ip/vmlinux /dev/mtd0
PMON> devcp tftp://ip/rootfs.img /dev/mtd1y
```

PMON下擦除所有坏块：

```
PMON> mtd_erase /dev/mtd1r
```

设置上电自启动：

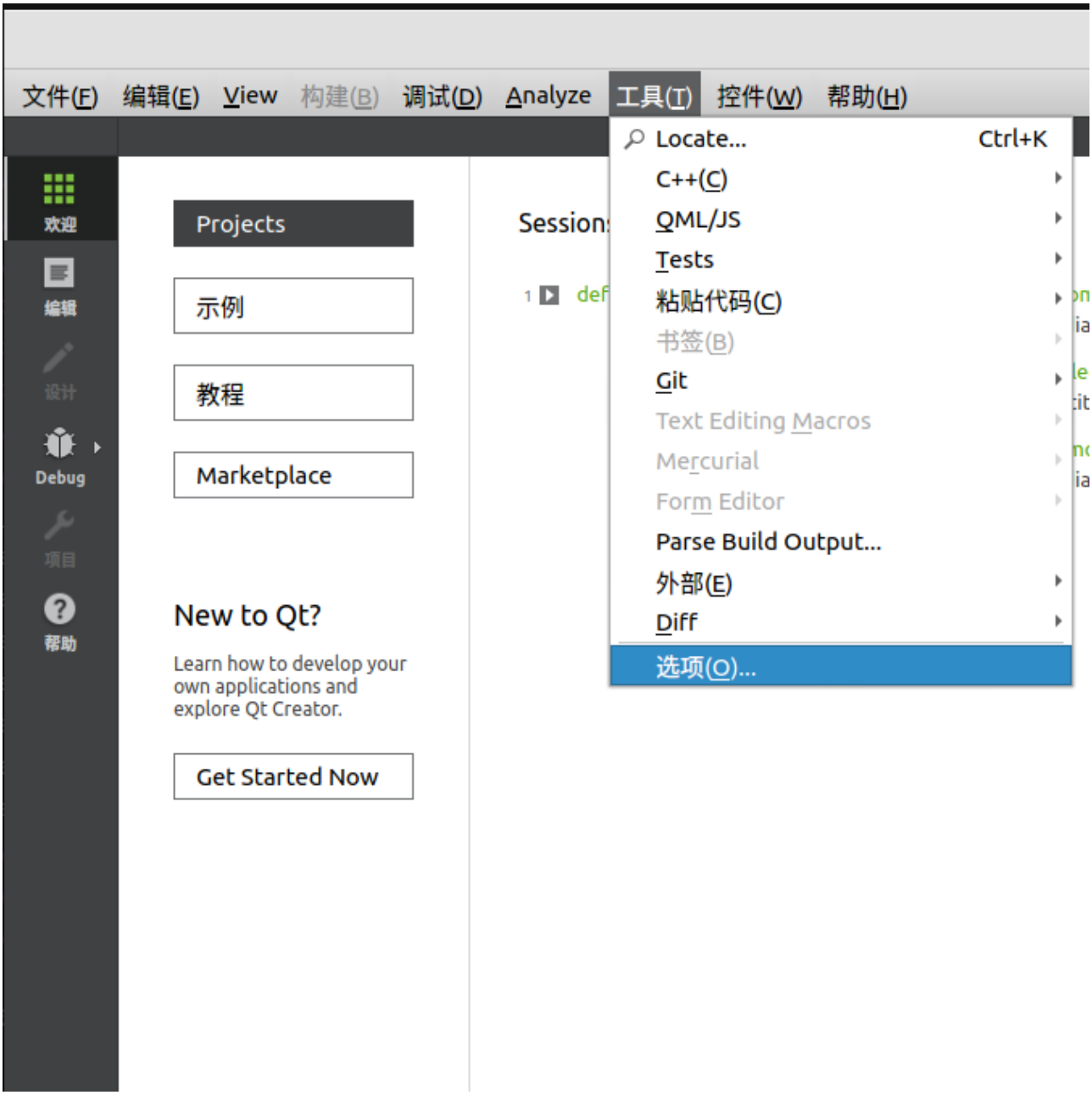
```
PMON> set al1 /dev/mtd0
PMON> set append "console=ttyS0,115200 init=/linuxrc rw
root=/dev/mtdblock1 rootfstype=yaffs2"
```

QT开发

QT Creator交叉开发环境搭建

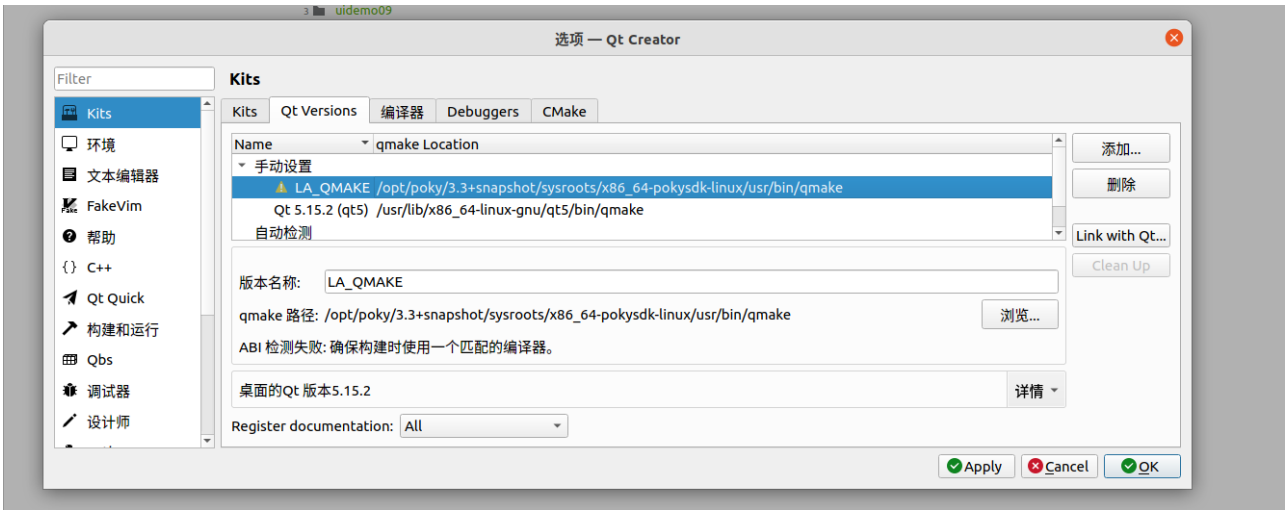
步骤如下：

1. 选择菜单栏的工具-选项

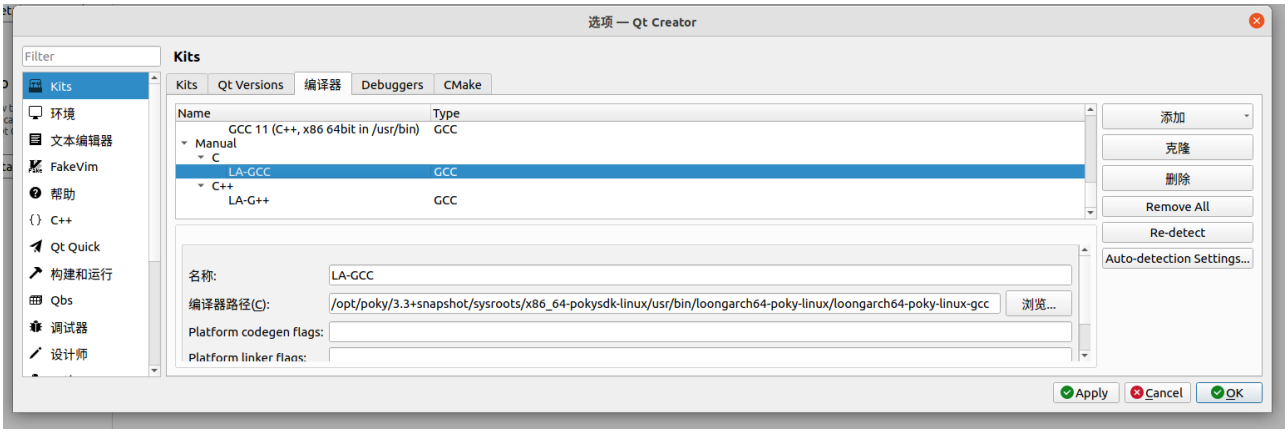


2. 设置qmake,如图Kits->QT versions 里添加qmke路径`/opt/poky/3.3+snapshot/sysroots/x86_64-pokysdk-`

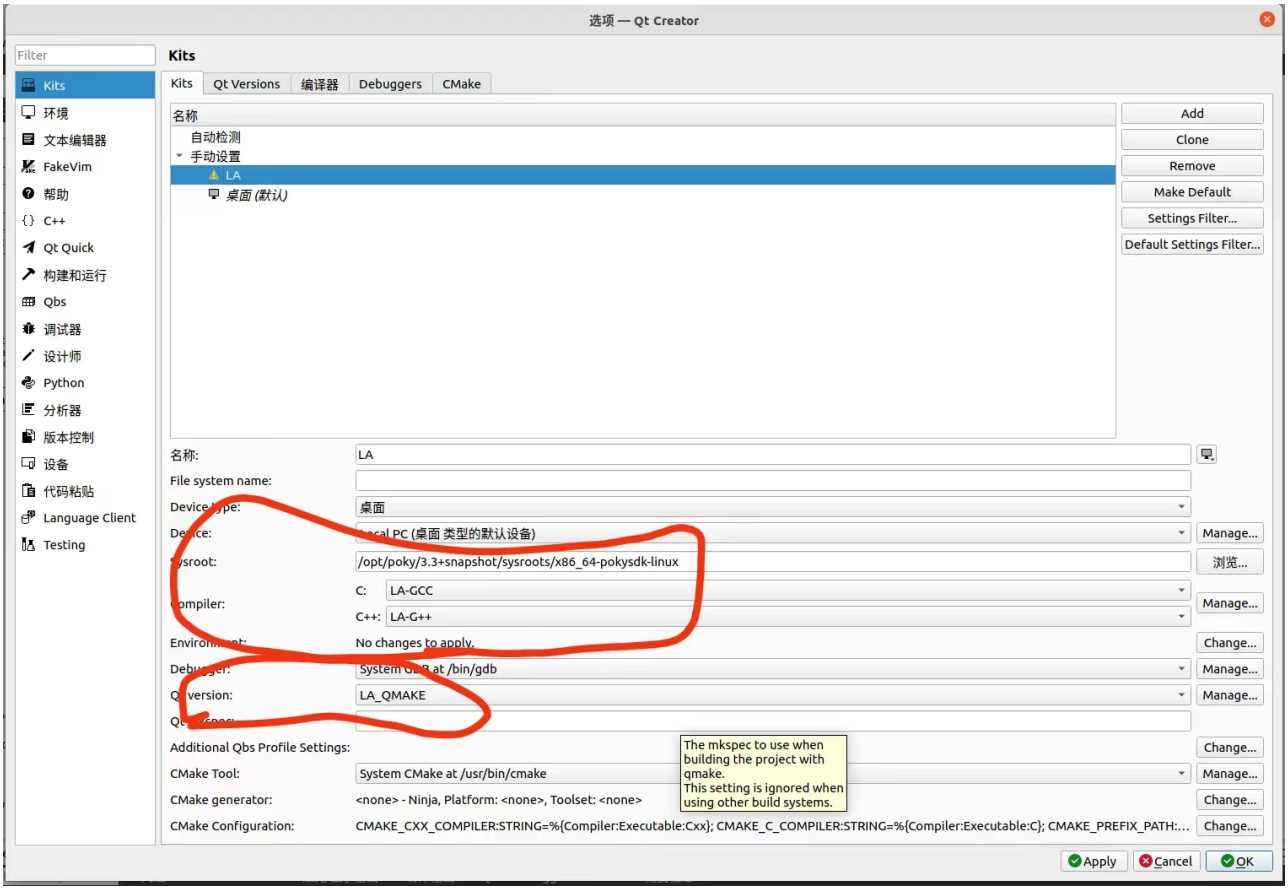
`linux/usr/bin/qmake`



3. 设置gcc&g++，如图Kits->编译器里添加gcc路径：`/opt/poky/3.3+snapshot/sysroots/x86_64-pokysdk-linux/usr/bin/loongarch64-poky-linux/loongarch64-poky-linux-gcc`，g++路径：`/opt/poky/3.3+snapshot/sysroots/x86_64-pokysdk-linux/usr/bin/loongarch64-poky-linux/loongarch64-poky-linux-g++`



4. 设置Kits，如图Kits->Kits 里添加sysroot路径及上面设置的QMAKE、gcc、g++,sysroot路径：`/opt/poky/3.3+snapshot/sysroots/x86_64-pokysdk-linux`



5. 设置完成后保存，编译工程时选择LA即可

地址空间详解

pcie空间

龙芯2K1000的硬件资源大多都在pcie总线上，这些外设的基址也需要通过访问pcie的配置空间得到
type0访问内部资源，访问基址0xba000000,读取对应设备配置空间基址偏移0x10的地址，基址如下

设备	配置空间访问	默认值
apb	0xba001000	0x1fe20000
gmac0	0xba001800	0x40040000
gmac1	0xba001900	0x40050000
usb-otg	0xba002000	0x40000000
usb-ehci	0xba002100	0x40060000
usb-ohci	0xba002200	0x40070000
gpu	0xba002800	0x40080000
dc	0xba003000	0x400c0000
hda	0xba003800	0x400d0000
sata	0xba004000	0x400e0000
pcie0-p0	0xba004800	0x40100000
pcie0-p1	0xba005000	0x50000000
pcie0-p2	0xba005800	0x54000000
pcie0-p3	0xba006000	0x58000000
pcie1-p0	0xba006800	0x60000000
pcie1-p0	0xba007000	0x70000000
dma	0xba007800	

IO地址空间

芯片配置寄存器

apb设备地址

```
uart\n      0x1fe20n00
can0        0x1fe20c00
can1        0x1fe20d00
i2c0        0x1fe21000
i2c1        0x1fe21800
pwm\n       0x1fe220n0
hpet        0x1fe24000
ac97        0x1fe25000
nand        0x1fe26000
acpi        0x1fe27000
rtc         0x1fe27800
des         0x1fe28000
aes         0x1fe29000
rsa         0x1fe2a000
rng         0x1fe2b000
sdio        0x1fe2c000
I2S         0x1fe2d000
E1          0x1fe2e000
```

type1访问外部pcie设备配置空间,基址0xbb000000,配置空间访问

```
31~24   23~16   15~11   10~8    7~0
0xbb    bus号   设备号  功能号  寄存器
```

spiflash

```
0~0xfb000  程序
0xfb000~0xff000 dtb
0xff000~0xff200 env
0xff200~0x100000 保留
```