第1章　基于 Ubuntu 的 Qt Creator 远程调试开发板

章节导读

在嵌入式 Linux 平台进行 Qt 应用开发调试时，开发者常会面临诸多繁琐问题：传统模式需要先在主机完成交叉编译，再手动将程序文件传输到开发板，仅能通过日志打印粗略排查错误，无法实现断点调试、单步运行、变量追踪等核心调试功能；反复编译、拷贝、重启程序的流程耗时费力，严重影响开发效率。针对这一行业痛点，我们可以采用网口远程调试方案，通过网络实现主机与开发板的直连调试。

本章将基于 Ubuntu 系统下的 Qt Creator，实现通过网络远程连接开发板并进行程序调试。以往相关教程均未讲解 Qt 远程调试的相关配置与操作方法，导致多数使用者无法对编写的 Qt 程序进行远程调试，给嵌入式 Qt 开发带来诸多不便。本章将针对这一问题展开详细讲解，帮助读者快速掌握通过网口便捷调试开发板 Qt 程序的完整流程。

接下来正式开始实战配置教程。第一步，将虚拟机网络连接模式修改为桥接模式，并勾选「复制物理网络连接状态」选项。：

图 1-1 修改连接模式

修改完成后，如果主机的网口没有连接到网络则虚拟机会提示：“无法连接到网络”并且没有网络连接的图标，如图 1-2 所示：

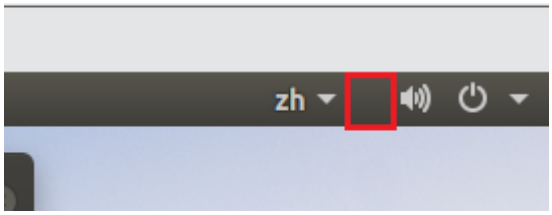

图 1-2 无网络连接图标

这时候再在虚拟机的终端输入 ifconfig 指令会发现没有分配 ip 地址：

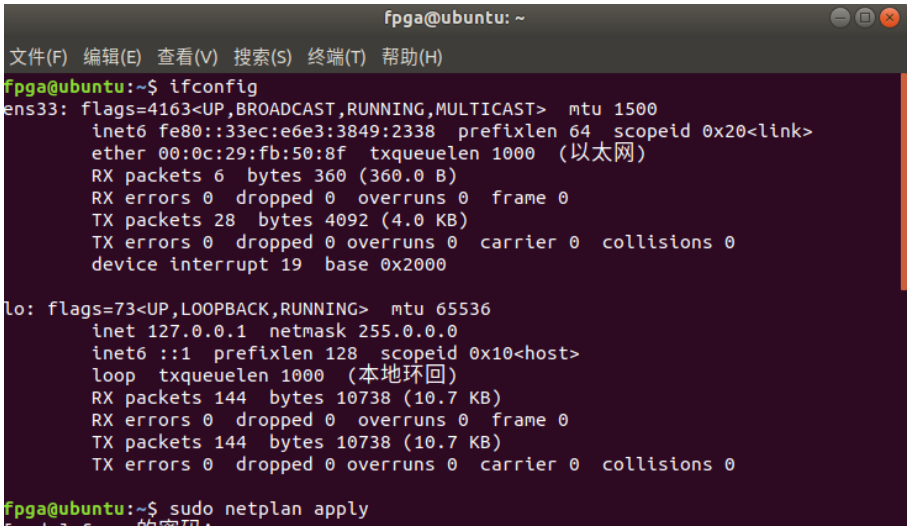

图 1-3 ip 地址未分配

所以我们需要修改文件，去手动强制分配一个未使用到的固定的 ip 地址。使用管理员权限打开虚拟机的/etc/netplan/01-network-manager-all.yaml 文件：

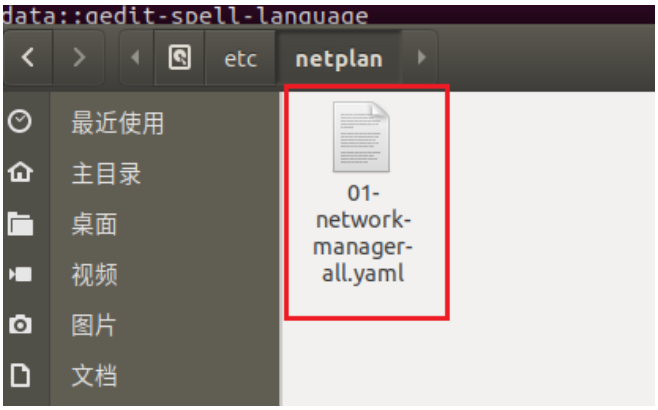

图 1-4 打开配置文件

配置文件内容如下：

图 1-5 修改文件内容

手动添加如下内容，注意缩进和空格：

```
ethernets:
```

```
ens33:
  dhcp4: no
  addresses:
   - 192.168.0.88/24     # 选一个未占用的 IP，比如 192.168.0.88
  nameservers:
   addresses: []         # 如果无需 DNS 可留空或填 192.168.0.1
```

图 1-6 添加内容

文件修改保存后，再在虚拟机终端中输入如下指令使配置生效并查看是否分配好 ip 地址：

```
sudo netplan apply
ifconfig ens33
```

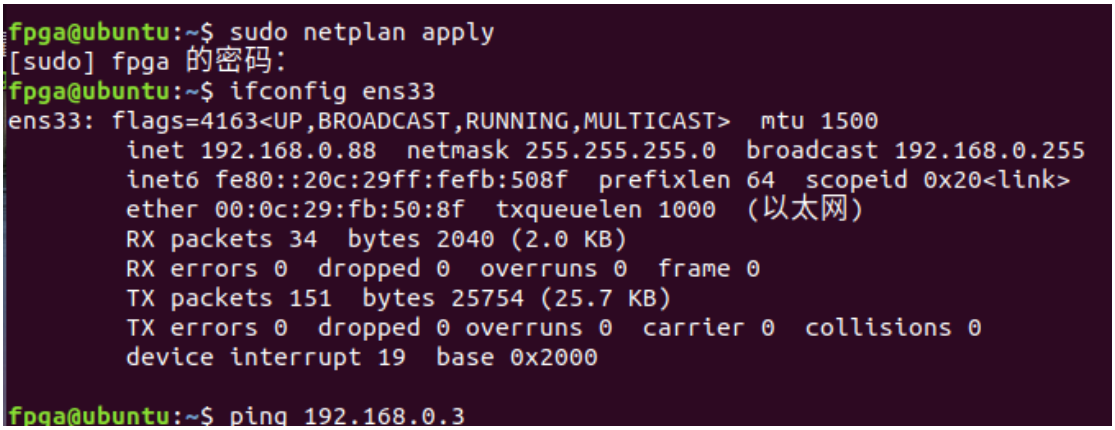

图 1-7 分配地址生效

由上述操作可知 ip 地址已经分配好了，为 192.168.0.88，这样就可以尝试 ping 主机和开发板：

```
fpga@ubuntu:~$ ping 192.168.0.3
PING 192.168.0.3 (192.168.0.3) 56(84) bytes of data.
64 bytes from 192.168.0.3: icmp_seq=1 ttl=64 time=0.747 ms
64 bytes from 192.168.0.3: icmp_seq=1 ttl=64 time=0.771 ms (DUP!)
64 bytes from 192.168.0.3: icmp_seq=2 ttl=64 time=0.222 ms
64 bytes from 192.168.0.3: icmp_seq=2 ttl=64 time=0.234 ms (DUP!)
64 bytes from 192.168.0.3: icmp_seq=3 ttl=64 time=0.326 ms
64 bytes from 192.168.0.3: icmp_seq=3 ttl=64 time=0.337 ms (DUP!)
64 bytes from 192.168.0.3: icmp_seq=4 ttl=64 time=0.247 ms
64 bytes from 192.168.0.3: icmp_seq=4 ttl=64 time=0.259 ms (DUP!)
64 bytes from 192.168.0.3: icmp_seq=5 ttl=64 time=0.381 ms
64 bytes from 192.168.0.3: icmp_seq=5 ttl=64 time=0.402 ms (DUP!)
^C
--- 192.168.0.3 ping statistics ---
5 packets transmitted, 5 received, +5 duplicates, 0% packet loss, time 4097ms
rtt min/avg/max/mdev = 0.222/0.392/0.771/0.193 ms
fpga@ubuntu:~$ ping 192.168.0.66
PING 192.168.0.66 (192.168.0.66) 56(84) bytes of data.
64 bytes from 192.168.0.66: icmp_seq=1 ttl=64 time=1.72 ms
64 bytes from 192.168.0.66: icmp_seq=2 ttl=64 time=0.724 ms
64 bytes from 192.168.0.66: icmp_seq=3 ttl=64 time=0.481 ms
64 bytes from 192.168.0.66: icmp_seq=4 ttl=64 time=0.449 ms

--- 192.168.0.66 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3050ms
rtt min/avg/max/mdev = 0.449/0.844/1.725/0.520 ms
fpga@ubuntu:~$
```

图 1-8 ping 通主机和开发板

确保主机与开发板网络连通且双向 ping 通，确认网络环境无异常后，即可开展 Qt 与开发板的连接配置。

首先将指定路径下的 gdbserver 工具（路径：/home/fpga/ZYNQ/Zynq_Linux_Base_Project/buildroot/gcc-linaro-11.3.1-2022.06-x86_64_arm-linux-gnueabihf/arm-linux-gnueabihf/usr/bin）复制到开发板的 /usr/bin 目录下，最后在开发板上启动 ssh 或轻量级 sshd 服务即可。

```
root@OV5640_linux_prj:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0A:35:00:1E:53
          inet addr:192.168.0.66  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::20a:35ff:fe00:1e53/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:924 (924.0 B)  TX bytes:540 (540.0 B)
          Interrupt:28 Base address:0xb000

root@OV5640_linux_prj:~# /usr/sbin/sshd
random: crng init done
root@OV5640_linux_prj:~#
root@OV5640_linux_prj:~#
```

图 1-9 确认 ip 地址分配

随后在虚拟机终端中输入 ssh root@192.168.0.66，测试能否成功远程连接开发板。若未设置 root 密码或遗忘密码，可直接在开发板终端运行 passwd root 命令，按照提示完成密码设置即可：

```
root@OV5640_linux_prj:~# passwd root
New password:
Retype new password:
passwd: password updated successfully
root@OV5640_linux_prj:~#
```

图 1-10 设置新密码

密码设置完成后，再到虚拟机中重复上次的连接操作，第一次连接需要输入 yes（完整输入），如果发现还是连接不上，那么就可能是 ssh 默认配置不允许密码连接：

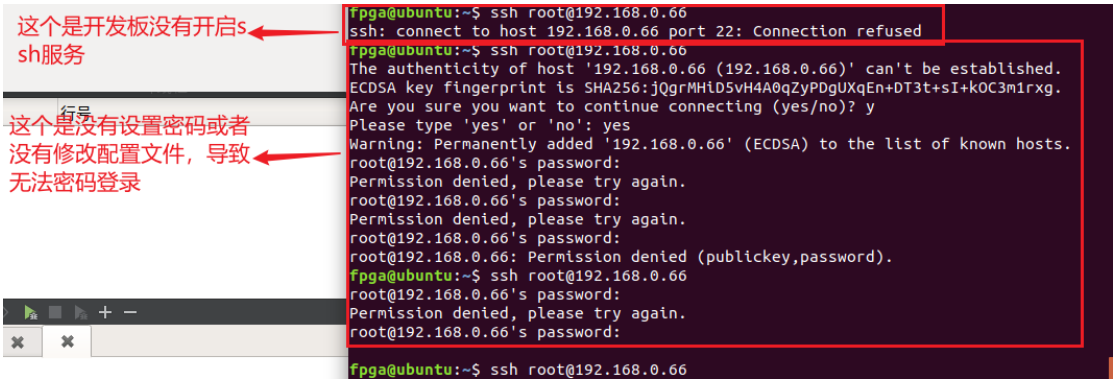

图 1-11 无法连接

这里需要修改一下 ssh 的配置文件：

```
root@OV5640_linux_prj:~# cat /etc/ssh/sshd_config | grep -i permitroot
#PermitRootLogin prohibit-password
# the setting of "PermitRootLogin without-password".
root@OV5640_linux_prj:~# sudo vim /etc/ssh/sshd_config
```

图 1-12 查找配置文件位置

将这这部分修改为： PermitRootLogin yes。删除行首的 #（取消注释）再把后面的 prohibit-password 改为 yes，或者直接在这一行后再添加一行 PermitRootLogin yes：

图 1-13 修改配置文件内容

修改完成并保存后重启开发板，再在开发板上运行/usr/sbin/sshd 和配置 ip 地址，最后在虚拟机里输入连接指令：

```
fpga@ubuntu:~$ ssh root@192.168.0.66
root@192.168.0.66's password:
root@OV5640_linux_prj:~#
root@OV5640_linux_prj:~#
root@OV5640_linux_prj:~#
root@OV5640_linux_prj:~# exit
logout
Connection to 192.168.0.66 closed.
fpga@ubuntu:~$
```

图 1-14 连接开发板

这里需要注意一下当你更换了根文件系统之后，想要再连接 ssh 的话，就需要输入如下指令将之前连接 ssh 的根文件系统的旧密钥给删掉，然后才能连接：

```
ssh-keygen -f "/home/fpga/.ssh/known_hosts" -R "192.168.0.66"
ssh root@192.168.0.66
```

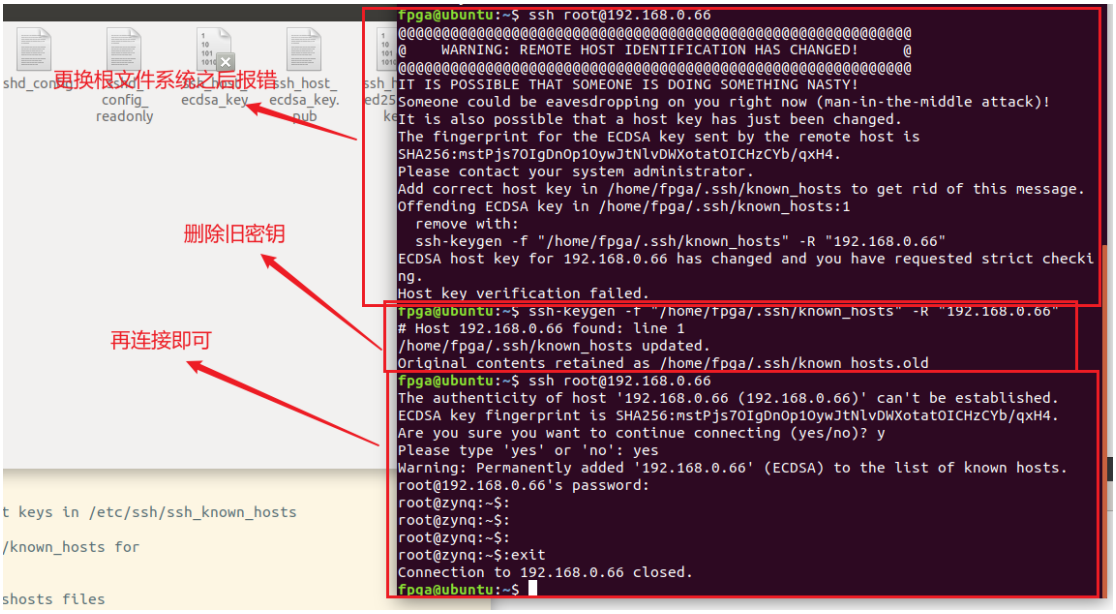

图 1-15 删除旧密钥再连接

有如上打印信息就证明开发板和虚拟机连通了，接着输入“exit”退出连接，再然后打开 Qt → 工具 → 选项 → 设备，按照如下样式填写配置：

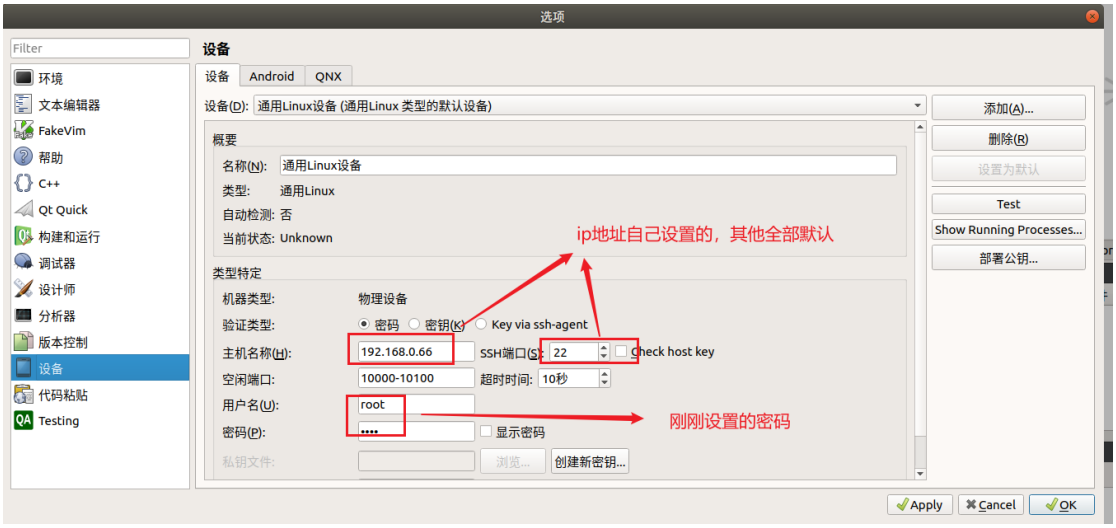

图 1-16 连接配置修改

随后点击 Test，进行连接验证，可以看到提示连接成功：

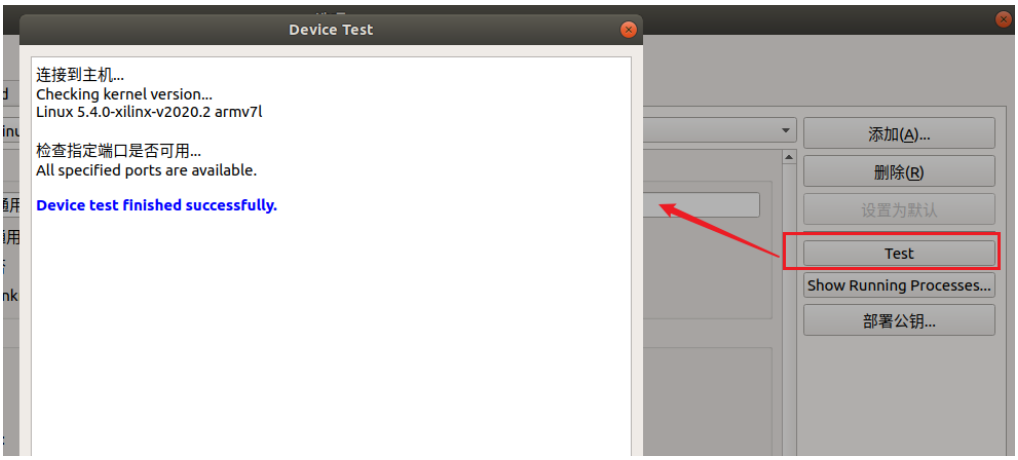

图 1-17 测试连接

这里我们点击 Apply 再点击 ok 后，即可进行 gdb 调试了，点击 Qt → 调试 → 开始调试 → Attach to …

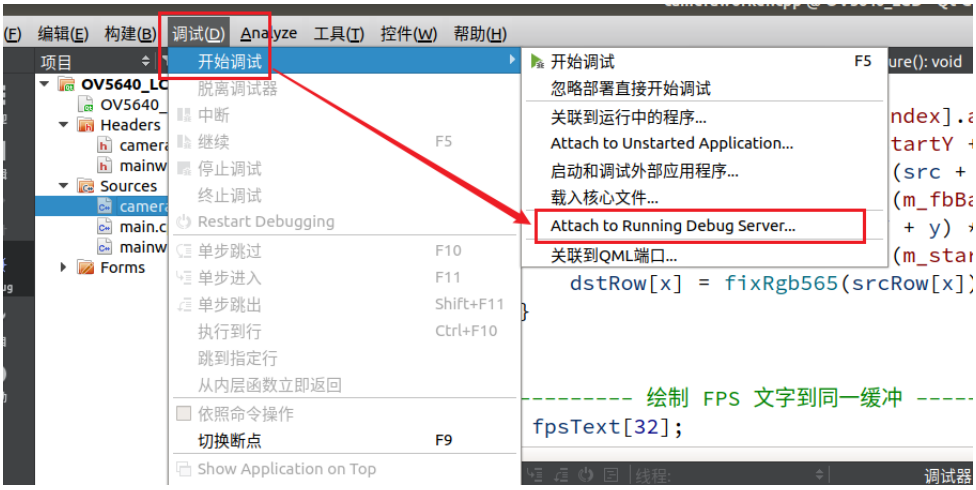

图 1-18 打开 debug 配置界面

根据如下图片填写配置内容：

图 1-19 配置调试器

到这里先别点击 ok，我们需要先在开发板上运行 gdbserver 后再进行 debug：

```
gdbserver 192.168.0.88:1234 ./OV5640_LCD
```

```
root@OV5640_linux_prj:~# gdbserver 192.168.0.88:1234 ./ov5640_test
Process ./ov5640_test created; pid = 222
Listening on port 1234
```

图 1-20 运行 gdbserver

输入上述指令后程序会停止运行等待 Qt 连接后再启动，这时候我们到虚拟机中点击 ok，开始运行程序：

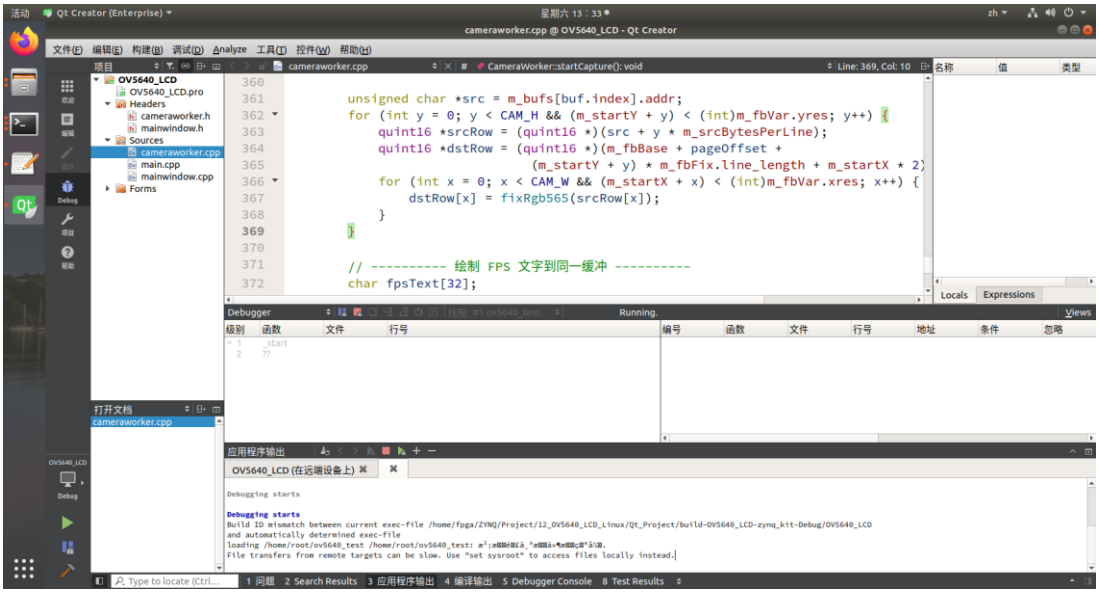

图 1-21 点击 ok 开始调试

可以观察到开发板开始运行程序：

```
root@OV5640_linux_prj:~# gdbserver 192.168.0.88:1234 ./ov5640_test
Process ./ov5640_test created; pid = 381
Listening on port 1234
Remote debugging from host 192.168.0.88, port 47172
xlnx-passthrough video2axi: get_fmt: pad=1, returning code=0x1008
xlnx-passthrough video2axi: get_fmt: pad=1, returning code=0x1008
xlnx-passthrough video2axi: s_stream: video-in-axi4s enable=1
xlnx-passthrough ov5640-data: s_stream: ov5640-data enable=1
ov5640 1-003c: ov5640_write_reg: write reg=0x3103, val=0x11
ov5640 1-003c: ov5640_write_reg: write reg=0x3008, val=0x82
ov5640 1-003c: ov5640_write_reg: write reg=0x3008, val=0x42
ov5640 1-003c: ov5640_write_reg: write reg=0x3103, val=0x03
ov5640 1-003c: ov5640_write_reg: write reg=0x3017, val=0xff
ov5640 1-003c: ov5640_write_reg: write reg=0x3018, val=0xff
ov5640 1-003c: ov5640_write_reg: write reg=0x3034, val=0x1a
ov5640 1-003c: ov5640_write_reg: write reg=0x3037, val=0x13
ov5640 1-003c: ov5640_write_reg: write reg=0x3108, val=0x01
ov5640 1-003c: ov5640_write_reg: write reg=0x3630, val=0x36
ov5640 1-003c: ov5640_write_reg: write reg=0x3631, val=0x0e
ov5640 1-003c: ov5640_write_reg: write reg=0x3632, val=0xe2
ov5640 1-003c: ov5640_write_reg: write reg=0x3633, val=0x12
ov5640 1-003c: ov5640_write_reg: write reg=0x3621, val=0xe0
ov5640 1-003c: ov5640_write_reg: write reg=0x3704, val=0xa0
ov5640 1-003c: ov5640_write_reg: write reg=0x3703, val=0x5a
ov5640 1-003c: ov5640_write_reg: write reg=0x3715, val=0x78
ov5640 1-003c: ov5640_write_reg: write reg=0x3717, val=0x01
ov5640 1-003c: ov5640_write_reg: write reg=0x370b, val=0x60
ov5640 1-003c: ov5640_write_reg: write reg=0x3705, val=0x1a
ov5640 1-003c: ov5640_write_reg: write reg=0x3905, val=0x02
ov5640 1-003c: ov5640_write_reg: write reg=0x3906, val=0x10
ov5640 1-003c: ov5640_write_reg: write reg=0x3901, val=0x0a
ov5640 1-003c: ov5640_write_reg: write reg=0x3731, val=0x12
```

图 1-22 程序运行

配置完成后，便可根据自身开发需求灵活开展程序调试。

第2章　基于 Ubuntu Qt 开发环境远程运行嵌入式开发板 Qt 程序的方案

有了上一章节的配置基础，本章的操作会更加简单易懂。只需点击 Qt Creator 界面左侧的项目按钮，再切换到 Run 运行选项，即可进入程序配置界面：

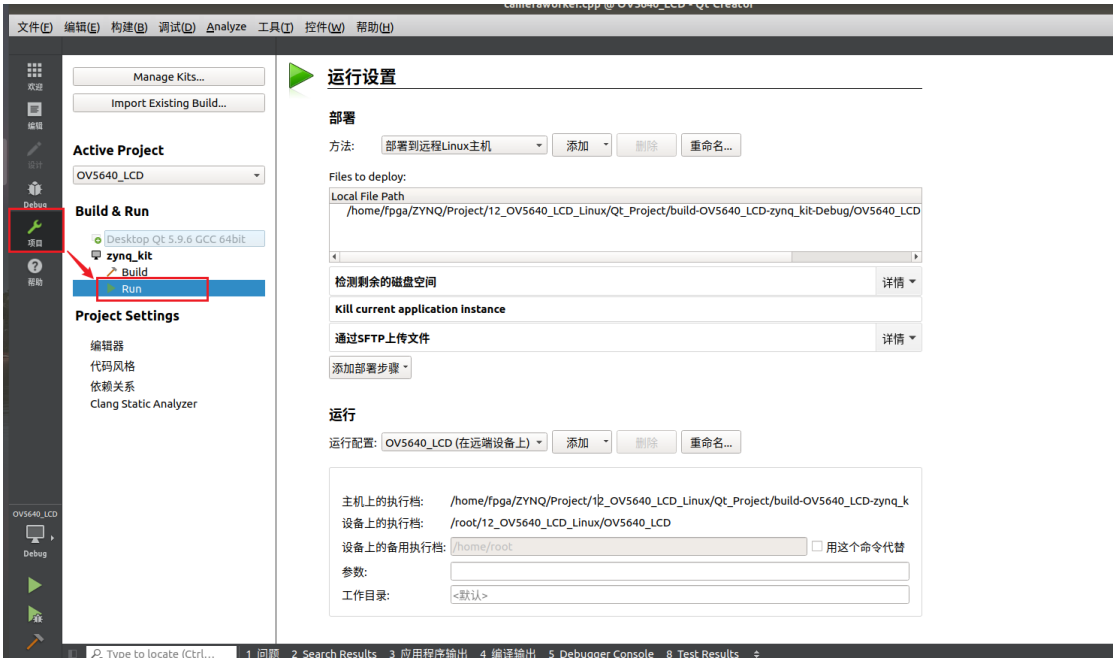

图 2-1 进入 run 配置界面

随后翻到页面最下方，找到 Run Environment 部分，点击批量编辑：

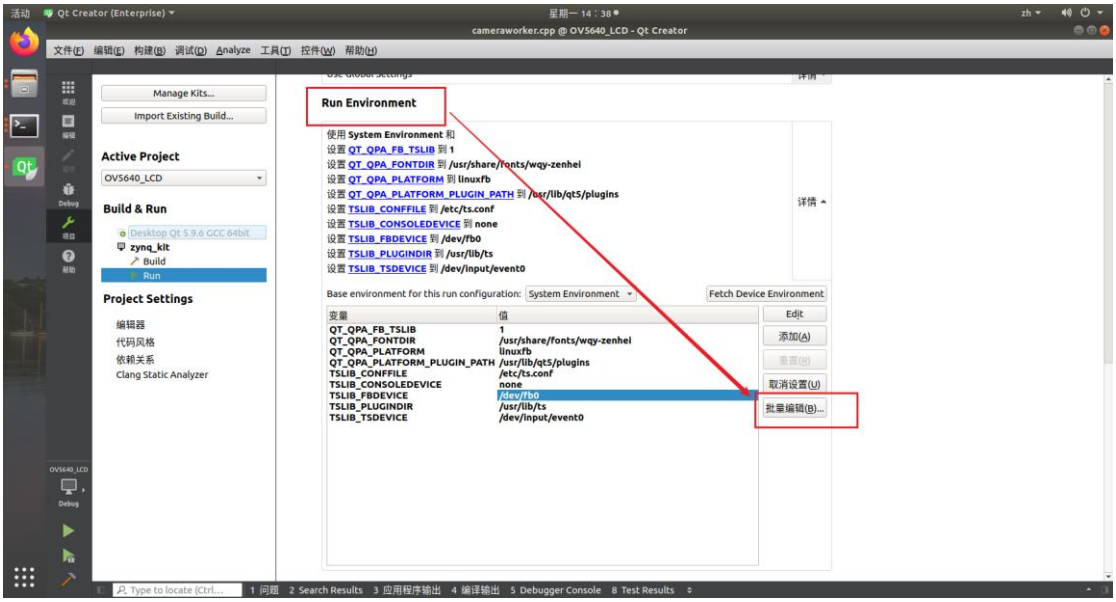

图 2-2 点击批量编辑

将如下环境变量内容复制到打开的弹窗中：

```
export TSLIB_CONSOLEDEVICE=none
export TSLIB_FBDEVICE=/dev/fb0
```

```
export TSLIB_TSDEVICE=/dev/input/event0
export TSLIB_CONFFILE=/etc/ts.conf
export TSLIB_PLUGINDIR=/usr/lib/ts
export QT_QPA_FB_TSLIB=1
export QT_QPA_PLATFORM=linuxfb
export QT_QPA_PLATFORM_PLUGIN_PATH=/usr/lib/qt5/plugins
export QT_QPA_FONTDIR=/usr/share/fonts/wqy-zenhei
```

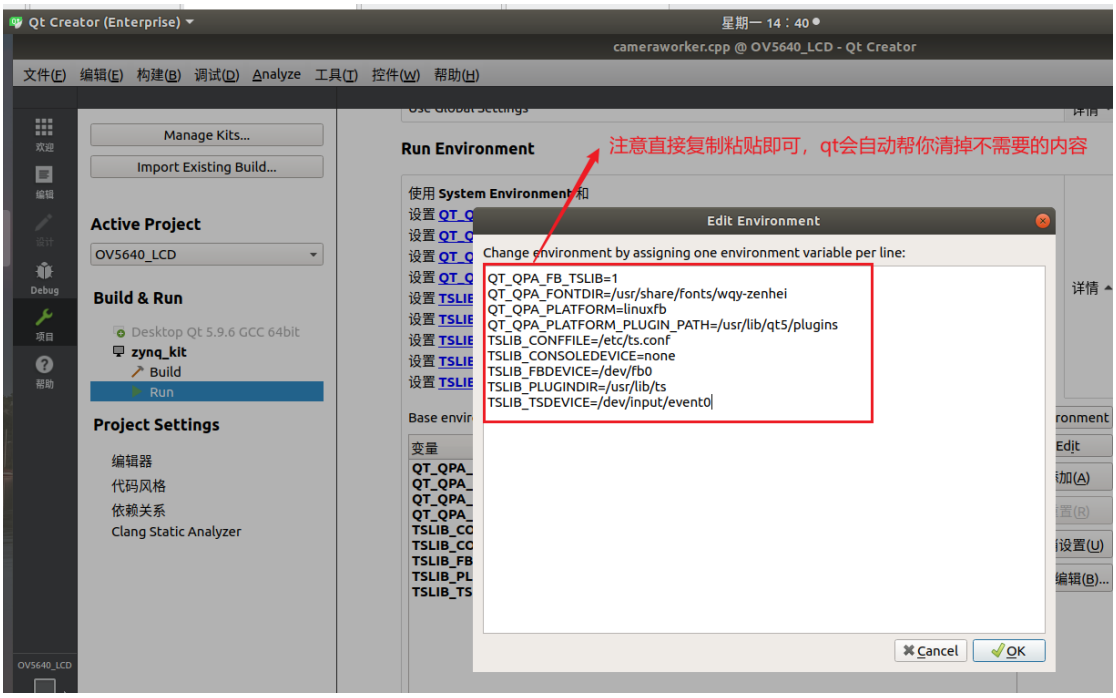

图 2-3 粘贴环境变量内容

点击 OK 按钮后，若出现以下界面，则代表运行环境变量设置完成：

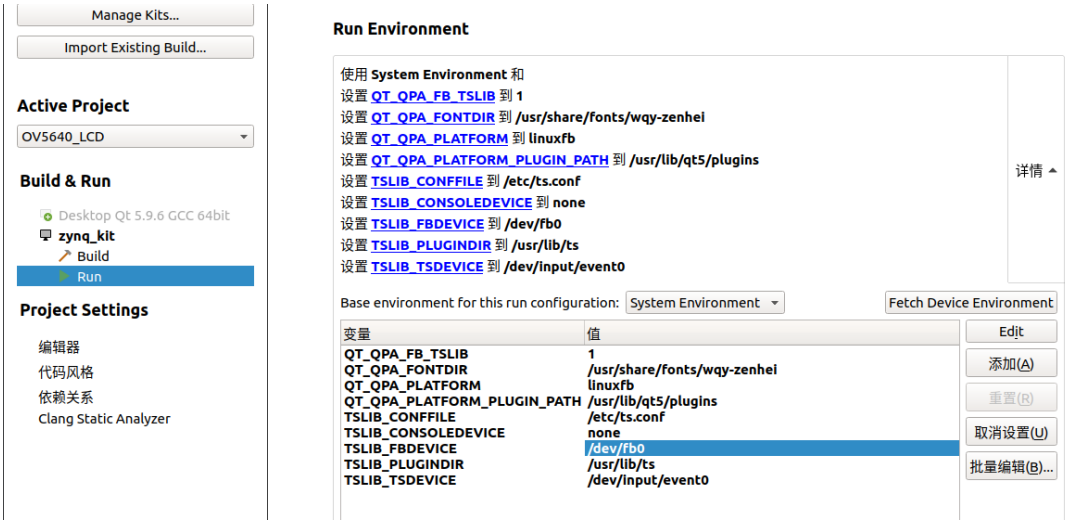

图 2-4 环境变量添加成功

接着需要添加部署远程目标设备的路径，打开 Qt 左边栏的编辑页面，进入 pro 文件，添加如下内容：

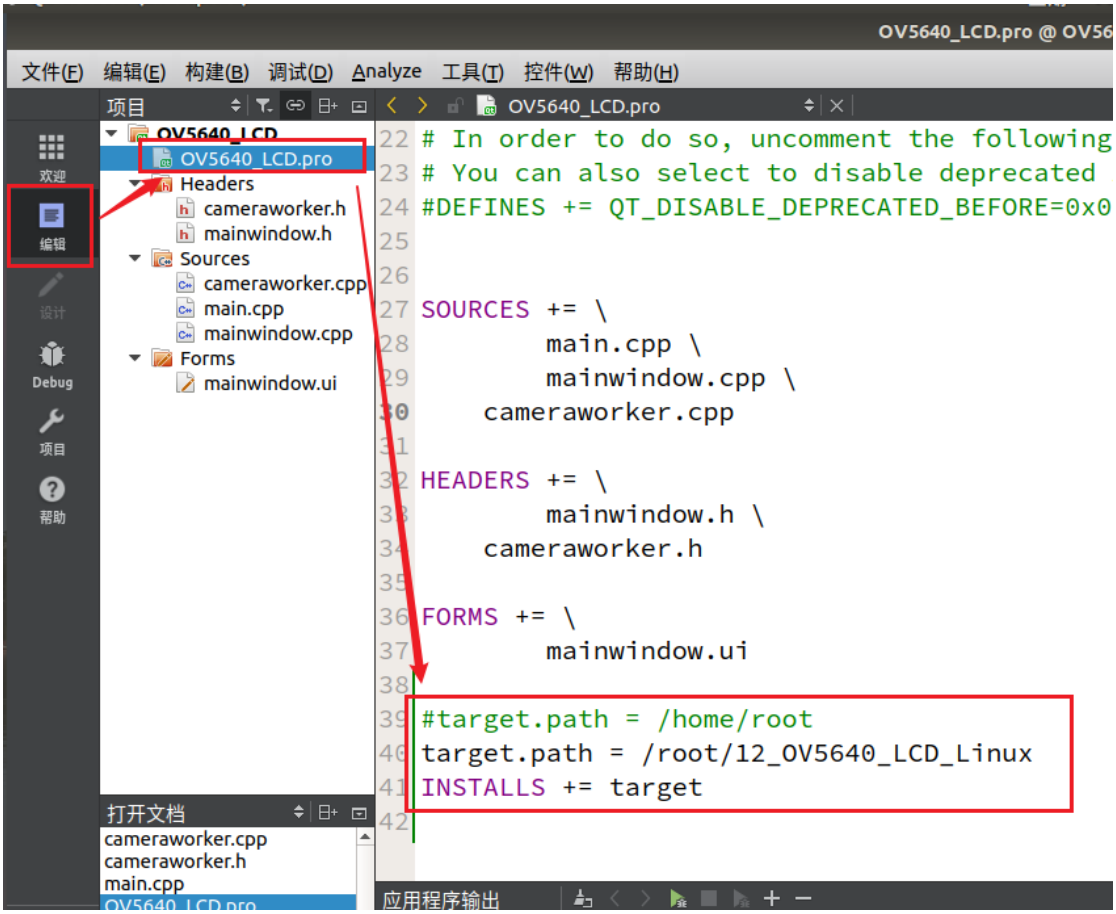

图 2-5 修改 pro 文件内容

target.path = /root/12_OV5640_LCD_Linux（修改为你的开发板上需要运行 Qt 程序的路径）
INSTALLS += target

按要求修改好后点击 run 按键即可在开发板上运行 Qt 程序了：

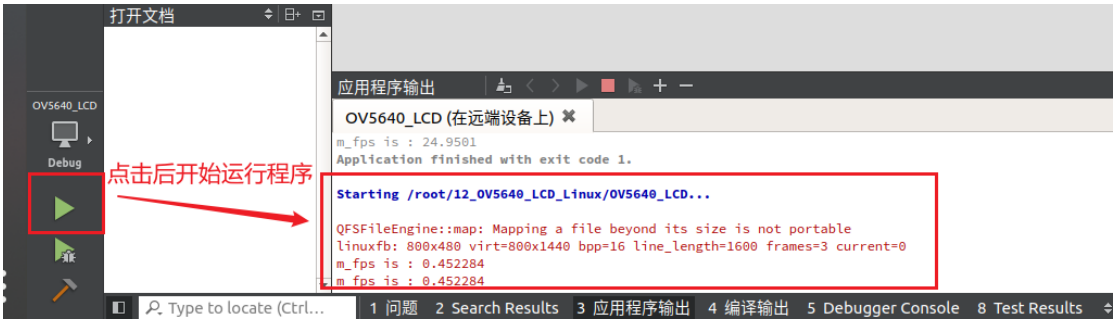

图 2-6 运行 Qt 程序

借助此调试方案，开发 Qt 程序时无需频繁插拔 SD 卡，仅需连通开发板与上位机网口，便可完成程序运行与在线调试，有效提升 Qt 项目开发与调试效率。