TW-AC6G-EVM 评估板

用户手册

版本：V1.0.0

修订历史

版本	日期	原因	修订者
V1.00	2020/04/03	创建文档	

目 录

1. 产品介绍

1.1 产品简介

TW-AC6G-EVM 为 Core-6GY 系列工业级核心板的评估板，以方便用户评估核心板及 CPU 的性能。

Core-6GY 系列工业级核心板基于 NXP(原 Freescale)公司的 i.MX6UL 系列高性能处理器设计，CPU 为 Cortex-A7 架构。评估板集成丰富的接口，支持摄像头接口、集成工业级 Wi-Fi、双路 CAN-bus 现场总线接口、双路以太网接口、8 路串口等，适用于快速开发一系列最具创新性的应用，如人机界面、工业 4.0、扫描仪、车载终端以及便携式医疗设备。

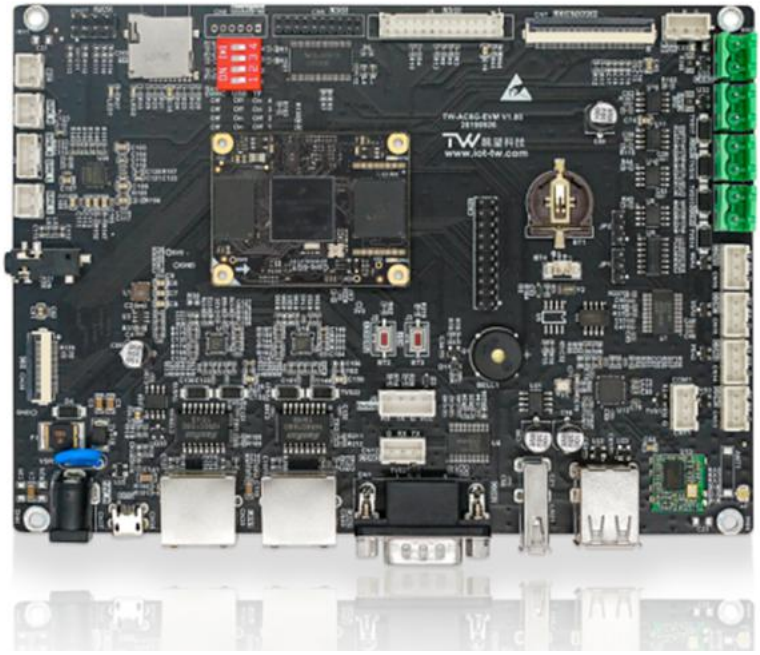


图 1.1 TW-AC6G-EVM 评估板外观

1.2 硬件资源介绍：

表 1.1 TW-AC6G-EVM 评估板参数表

产品名称	TW-AC6G-EVM 评估板
操作系统	Linux
处理器	i.MX6UL Cortex-A7
主频	528MHz
内存	256MB ~ 1G
电子硬盘	256MB ~ 4G
WiFi	支持，可选配
摄像头	1 路，CSI，可扩展模拟摄像头
LCD 最高分辨率	1366 * 768
VGA	可提供方案支持
LVDS	可提供方案支持
触摸屏	支持 4 线电阻式与电容触摸屏

音频接口	1 路输出，无需声卡 ，支持外扩声卡
USB	2 路 USB2.0（可提供方案扩展）
串口	最高 8 路（复用）
CAN-Bus	2 路
以太网	2 路
ADC	2 通道
SD 卡接口	最高 2 路（复用）
I2C	1 路
PWM	2 路（复用）
SPI	1 路
GPIO	30 路（复用）
电源供电	直流+12V 电源供电
机械尺寸	35mm * 45mm

注：受限于评估底板的尺寸与接口布局，核心板部分资源以插针方式引出。

1.3 软件资源介绍

TW-AC6G-EVM 评估板提供完善的 Linux BSP，包括 Linux 内核源码、和开发工具等，具体软件资源如下表所示：

表 1.2 TW-AC6G-EVM 评估板软件资源表

软件资源		说明
Linux 内核		4.1.15
文件系统		根文件系统采用 ext3，在根文件系统上可挂载多种文件系统，如：sysfs、yaffs2、ubifs 等
交叉编译器（内核）		arm-poky-linux-gnueabi-gcc 5.3.0
交叉编译器（应用程序）		arm-poky-linux-gnueabi-gcc 5.3.0
外设驱动	eMMC	驱动源码：/drivers/mmc/host/
	SD/MMC	驱动源码：/drivers/mmc/host/
	LCD	驱动源码：/drivers/video/fbdev/mxc/
	触摸屏	驱动源码：/drivers/input/touchscreen/
	摄像头	驱动源码：/drivers/media/platform/mxc/capture/
	I2C	驱动源码：/drivers/i2c/
	UART	驱动源码：/drivers/tty/serial
	USB	驱动源码：/drivers/usb
	以太网	驱动源码：/drivers/net/ethernet/freescale/
	CAN	驱动源码：/drivers/net/can
	WIFI	驱动源码：/drivers/net/wireless
	PWM	驱动源码：/drivers/pwm

	GPIO	驱动源码：/drivers/gpio
	RTC	驱动源码：/drivers/rtc
图形界面		使用 QT5.12
示例程序		提供串口、LED、网络、Web、数据库等开发例程
工具软件		如系统镜像烧写工具、串口调试工具、网络调试工具、tftp 服务器软件等

1.4 产品布局介绍

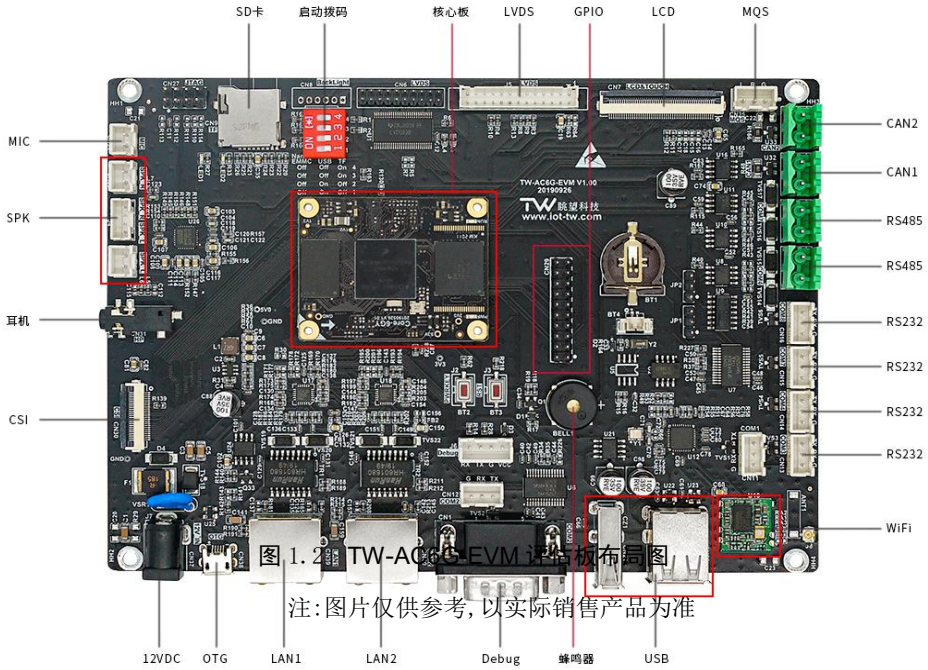


图 1.2 TW-AC6Q-EVM 评估板布局图

注:图片仅供参考,以实际销售产品为准

表 1.3 评估底板跳线与指示灯说明

序 号	评估底板标号	功能说明		描 述
		RS-485	RS-232	
1	JP1	短接 1、2 引脚	短接 2、3 引脚	COM4 端口
2	JP2	短接 1、2 引脚	短接 2、3 引脚	COM7 端口

表 1.4 评估底板跳线与指示灯说明

序号	评估底板标号	描述
1	LED1	运行指示灯，GPIO 控制
2	LED2	电源指示灯
3	BT3/RST	主板复位按键

2. 开发板运行和更新固件

2.1 安装 SecureCRT 软件

2.1.1 安装 SecureCRT

用户可以从“/眺望产品光盘资料/2、软件开发参考资料/5、软件工具” 下载 SecureCRT 安装程序。双击该文件开始安装，由于其安装过程比较简单，此处不再详细叙述。

2.1.2 使用 SecureCRT 登录

运行 SecureCRT 软件，弹出“连接”对话框，如下图 2-1 所示：

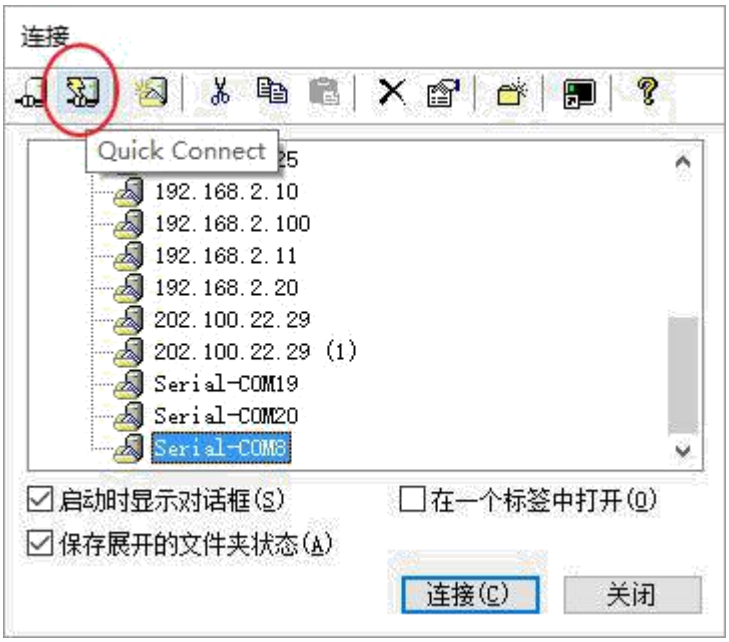


图 2.1 SecureCRT“连接”对话框

点击“Quick Connect”按钮，弹出“快速连接”对话框，如下图所示：

图 2.2 SecureCRT“快速连接”对话框

在该对话框中，选择“协议”为“Serial”，“端口”需根据主机实际使用的串口号进行选择（可从操作系统的“设备管理器”中获得该信息），并对串口参数进行相应的设置，值得

注意的是在设置串口参数时需关闭所有的“数据流控制”选项。在设置完成后，点击“连接”按钮，会出现一个用于登录的 SecureCRT 终端窗口。

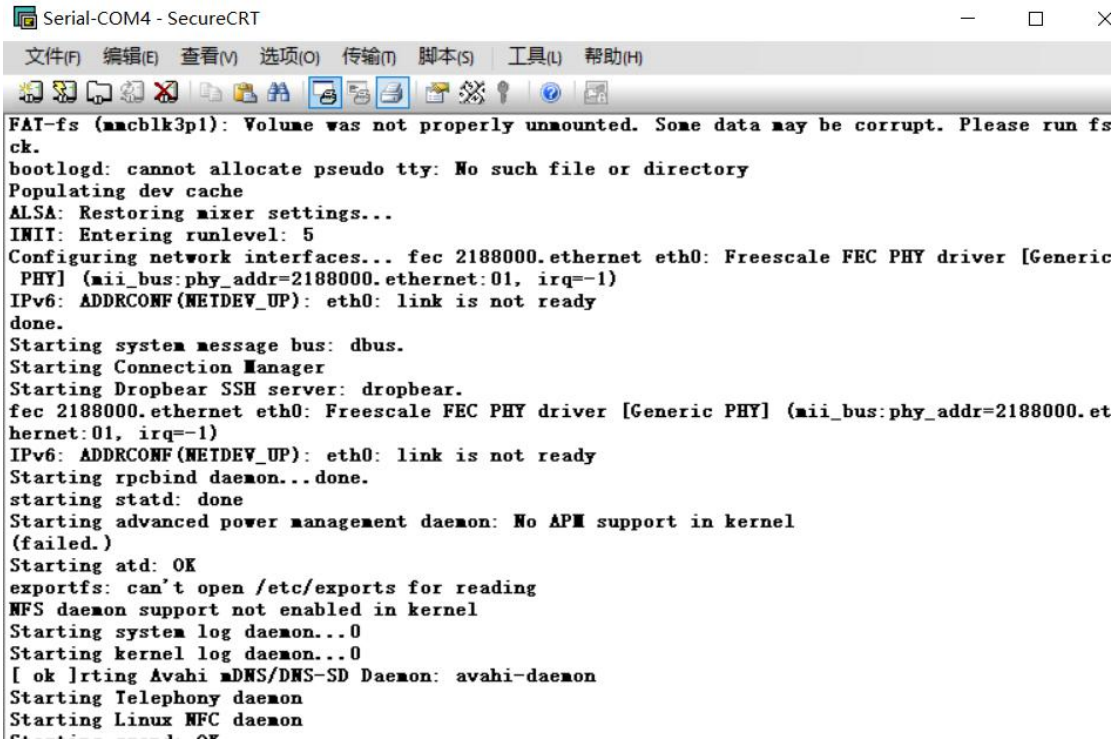


图 2.3 SecureCRT 终端登录窗口

2.2 mfgtools 工具烧写系统

可以使用 MfgTools 烧写工具（\TW-AC6G-EVM 光盘资料\4.固件更新与烧写方法\USB 烧写\mfgtools 工具）通过开发板的 USB 从口烧写系统映像。

2.2.1 mfgtool 工具的使用

1. Mfgtools 工具的下载目标固件的路径在（mfgtools 工具\Profiles\Linux\OS Firmware\files）中，目录内容如下图所示：

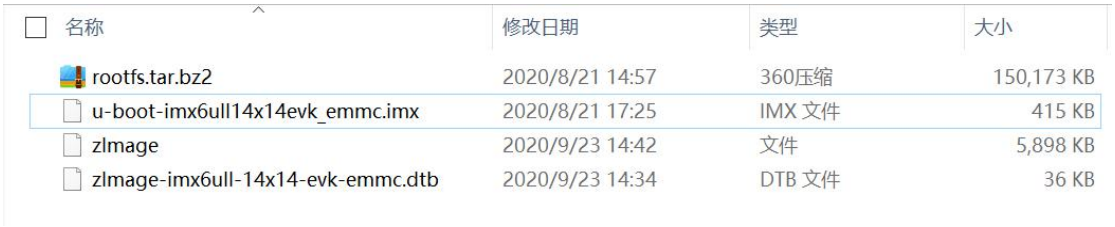
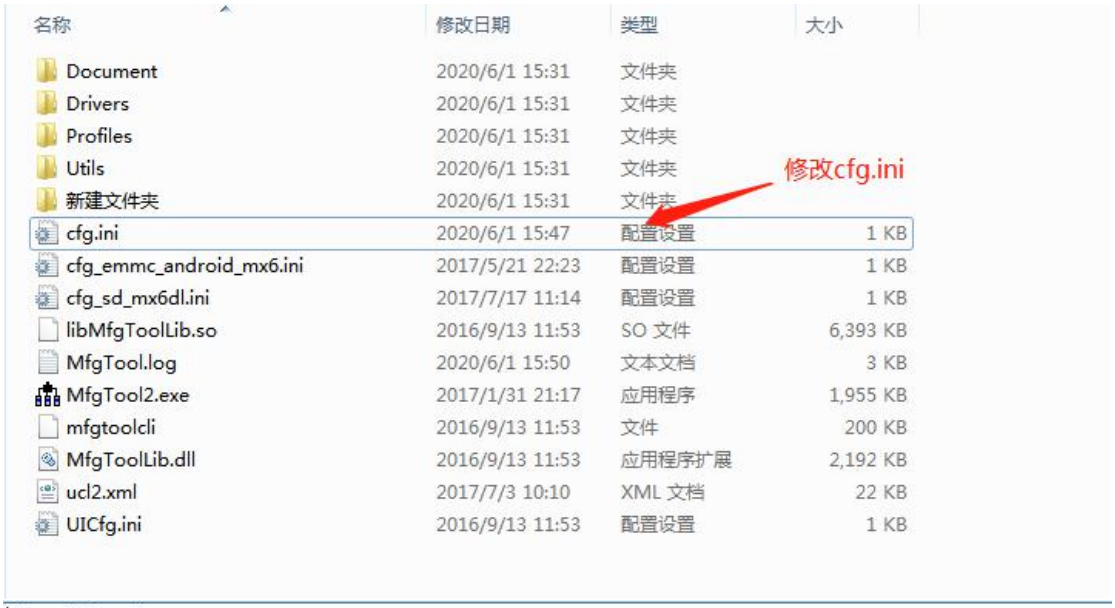


图 2.4 烧写固件 files 目录文件

Mfgtool 工具在下载时会将该目录内容下载到开发板上，用户在开发自己固件时，只需替换该目录下的对应文件即可，注意名字必须保持一致，否则无法烧录成功，另外 mfgtool 工具目录下的其余目录下的文件不要进行更改。

2. 如果用户只需要更新部分固件，可以修改 cfg.ini 文件




```
[profiles]
chip = Linux

[platform]
board = SabreSD

[LIST]
name = eMMC

[variable]
board = sabresd
mmc = 3
sxuboot=sabresd
sxdtb=sdb
7duboot=sabresd
7ddtb=sdb
6uluboot=14x14ddr3arm2
6uldtb=14x14-ddr3-arm2
6ulldtb=14x14-ddr3-arm2
ldo=
plus=
lite=1
initramfs=fsl-image-mfgtool-initramfs-imx_mfgtools.cpio.gz.u-boot
seek = 1
sxnor=qspi2
7dnor=qspi1
6ulnor=qspi1
nor_part=0
data_type=
folder=sabresd
```

可以根据ucl2.xml文件进行修改
如只更新内核：将eMMC改为eMMC Kernel
如只更新设备树:将eMMC改为eMMC DTB

2.2.2 使用 USB 方式烧写系统镜像

1.硬件连接如下：

（1）断开 TW-AC6G-EVM 评估板的供电电源;

（2）把 TW-AC6G-EVM 评估板设置为 USB 启动方式(U15 拨码开关拨到 off off on on);

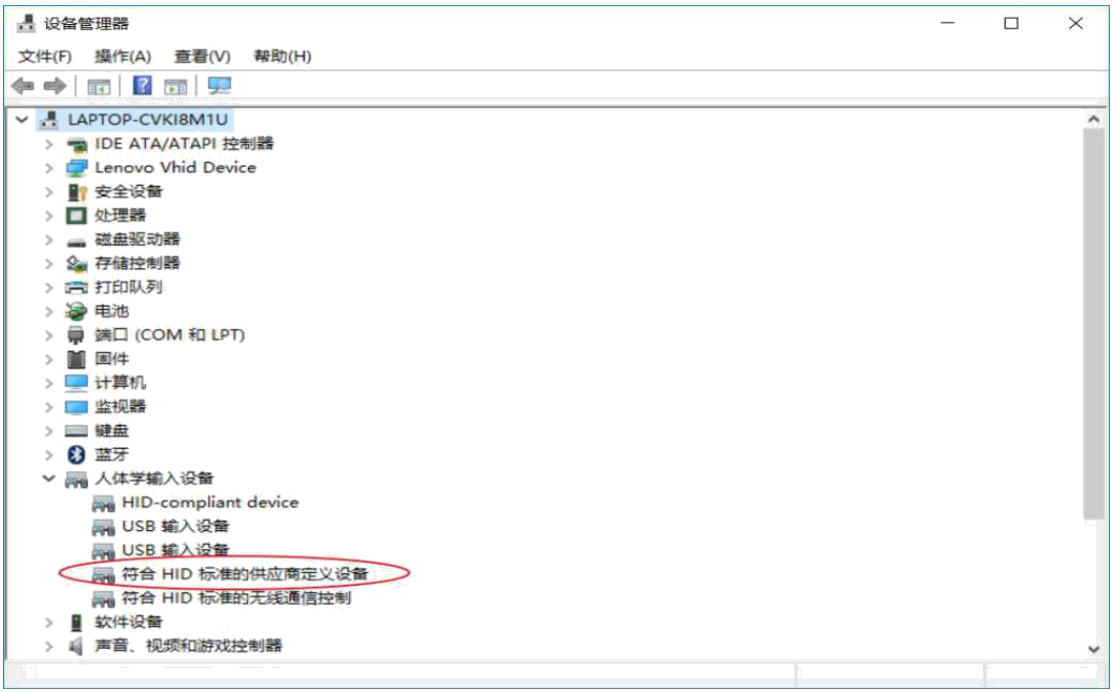
表 2.1 拨码开关启动选择说明

序号	U15 拨码顺序				描 述
	4	3	2	1	
1	off	off	off	off	NANDFlash /EMMC 启动
2	off	off	on	on	USB 启动
3	on	on	off	off	SD/TF 启动

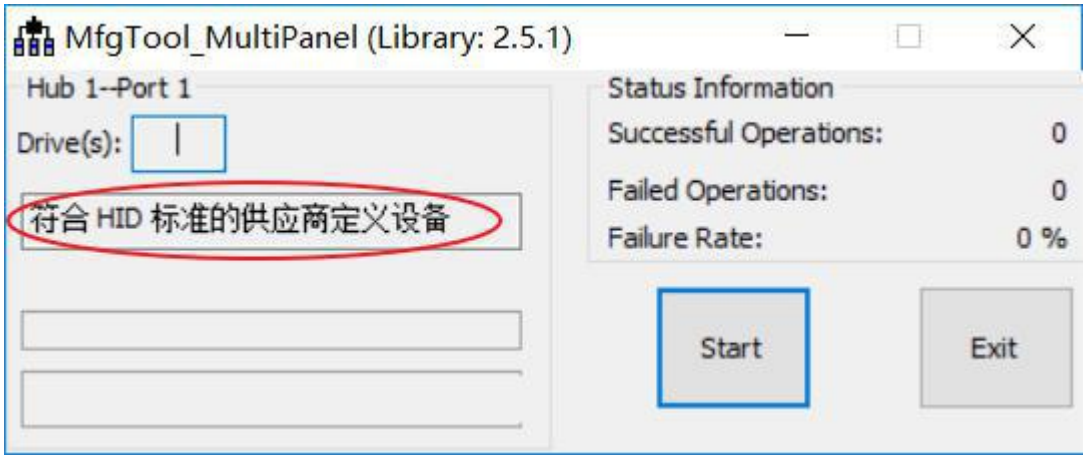
（3）使用 Miscro USB 线缆将 TW-AC6G-EVM 评估板的 USB 从口与计算机的 USB 端口相连;

（4）重新给 TW-AC6G-EVM 评估板上电。

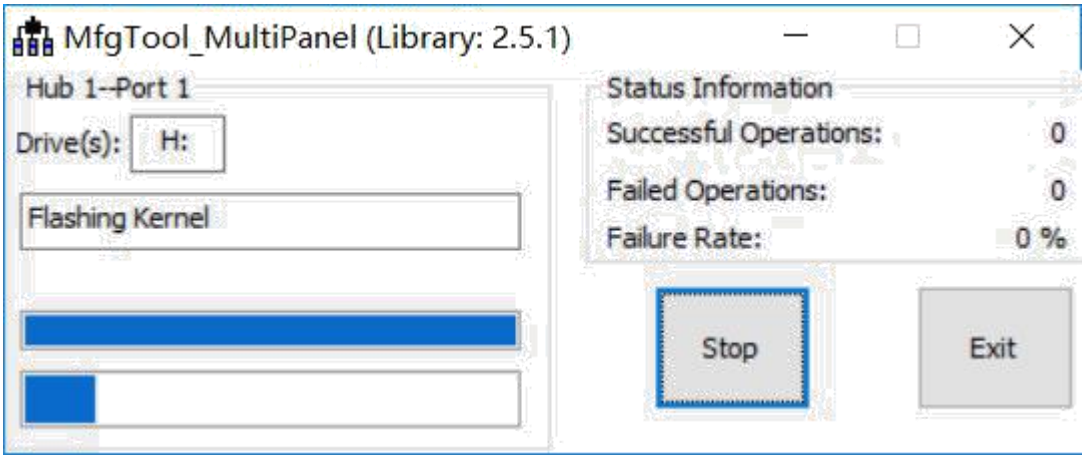
注：若上述 4 个步骤操作正确，则 TW-AC6G-EVM 重新上电之后，计算机应该能够检测到新的设备接入，并且可以在“设备管理器”中看到新的项目“人体学输入设备/符合 HID 标准的供应商定义设备‖，如图所示：



双击运行 mfgtools 文件夹中的 mfgtool2-yocto-mx-evk-emmc.vbs 脚本文件，打开 Mfgtool 软件的初始界面，界面显示正在监听“符合 HID 标准的供应商定义设备”，如图所示：

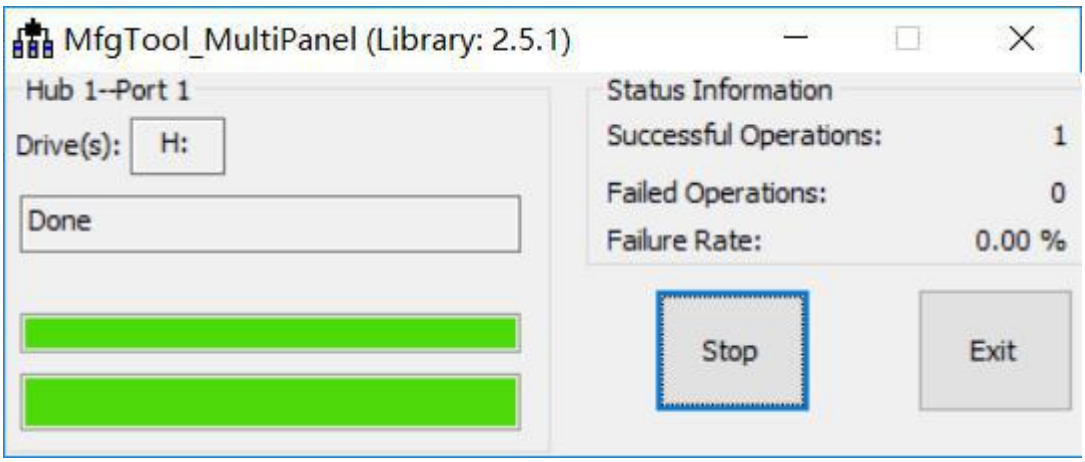


点击 Start 按钮开始进行固件的烧写工作，烧写过程如下图所示：



在烧写过程中，可以在调试串口上看到烧写过程。

待烧写完毕后，如下图所示：



点击－Stop‖按钮后，再点击－Exit‖按钮退出软件。接着断开 TW-AC6G-EVM 评估板的供电电源并将拨码开关拨到 off off off off 位置（即从 EMMC 启动），完成固件烧写工作。

2.2.3 SD 卡烧写系统镜像

1. SD 卡制作

可以参照（\TW-AC6G-EVM 光盘资料\4.固件更新与烧写方法\TF 卡烧写\TF 卡制作工具）路径中<<TF 启动卡制作.docx>>和<<tf 卡烧写系统说明.txt>>。

2. 硬件连接方法如下：

（1）断开 TW-AC6G-EVM 评估板的供电电源;

（2）把 TW-AC6G-EVM 开发板设置为 SD 启动方式(参照表 2.1);

（3）使用预先准备好的 SD 卡插入卡槽上;

（4）重新给 TW-AC6G-EVM 评估板上电。

2.3 单独更新设备树 DTB

本节介绍在操作系统命令行下，对于 emmc 存储版本单独烧写设备树文件 DTB 的步骤：

(1) 内核中设备树文件在/kernel-a7-tw-evm/arch/arm/boot/dts 路径下，找到相应的设备树文件，如 hd-imx6ull-core-emmc.dts 等。

(2) 修改后执行在源码/kernel-a7-tw-evm 目录下的 built-dtb.sh 文件，在/kernel-a7-tw-evm/arch/arm/boot/dts 目录下会生成 dtb 文件 hd-imx6ull-core-emmc.dtb。

(3) 将设备树文件拷贝到开发板/run/media/mmcblk1p1 目录下即可。对于 imx6ul 开发板，设备树文件名必须修改为 imx6ull-14x14-evk.dtb ，与 mfgtools 烧录文件名称一致，如下图所示：

```
root@twdz-IMX6ULL:/run/media# ls mmcblk1p1/
imx6ull-14x14-evk.dtb  zImage
```

(4) 输入命令 sync 同步

2.4 单独更新内核 zImage

本节介绍在操作系统命令行下，单独烧写操作系统映像文件（zImage）的步骤：

（1）执行在内核源码中顶层目录（kernel-a7-tw-evm 目录）下的 built-menuconfig.sh 文件。进入配置内核界面。

（2）配置好后执行中顶层目录（kernel-a7-tw-evm 目录）built-zImage.sh 文件，如果编译成功后，会在/kernel-a7-tw-evm/arch/arm/boot 路径下生成 zImage 文件。

（3）将内核映像文件（zImage）文件拷贝到开发板/run/media/mmcblk1p1 目录下即可，如下图所示：

```
root@twdz-IMX6ULL:/run/media# ls mmcblk1p1/
imx6ull-14x14-evk.dtb  zImage
```

(4) 输入命令 sync 同步

3. 功能测试

3.1 LED 测试

对开发板的 LED 进行测试，串口终端执行相应的指令去控制对应的 IO 来控制对应的器件，如可以改变当前的触发模式，并且通过相应的指令去控制 LED 灯的亮灭。

当执行以下指令后，开发板中 LED1 会亮起。

```
root@twdz-IMX6ULL:~# cd /sys/class/gpio/
root@twdz-IMX6ULL:/sys/class/gpio# echo 112 >export
root@twdz-IMX6ULL:/sys/class/gpio# cd gpio112
root@twdz-IMX6ULL:/sys/class/gpio/gpio112# echo out >direction
root@twdz-IMX6ULL:/sys/class/gpio/gpio112# echo 1 >value
```

```
root@twdz-IMX6ULL:/sys/class/gpio# echo 112 > export
root@twdz-IMX6ULL:/sys/class/gpio# cd gpio112
root@twdz-IMX6ULL:/sys/class/gpio/gpio112# echo out > direction
root@twdz-IMX6ULL:/sys/class/gpio/gpio112# echo 1 > value
```

当执行以下指令后，开发板中的 LED1 会灭掉。

```
root@twdz-IMX6ULL:/sys/class/gpio/gpio112# echo 0 > value
```

```
root@twdz-IMX6ULL:/sys/class/gpio/gpio112# echo 0 >value
```

3.2 蜂鸣器测试

通过指令对开发板的 Beep 进行测试，Beep 配置为普通 IO 口，通过控制 IO 口的高低电平可以驱动蜂鸣器，可在命令行下运行如下命令后，蜂鸣器会鸣叫：

```
root@twdz-IMX6ULL:/# echo 0 > /sys/class/pwm/pwmchip1/export
root@twdz-IMX6ULL:/# echo 1 > /sys/class/pwm/pwmchip1/pwm0/enable
root@twdz-IMX6ULL:/# echo 1000000 > /sys/class/pwm/pwmchip1/pwm0/period
root@twdz-IMX6ULL:/# echo 500000 > /sys/class/pwm/pwmchip1/pwm0/duty_cycle
```

执行以下命令后，蜂鸣器会关闭：

```
root@twdz-IMX6ULL:/# echo 0 > /sys/class/pwm/pwmchip1/pwm0/enable
```

3.3 串口测试

TW-AC6G-EVM 评估板提供了多个串口供用户使用:其中 COM2 为调试串口,COM1、COM3 、COM8、COM4 、COM7 为 RS232 通讯口,其中 COM4、COM7 为 RS232/RS485 复用通讯口,用户可以通过底板上的跳线进行切换。关于串口的详细说明可以参考《TWEvb-IMX6GY 数据手册》。在开发板 /home/demo/uart 目录下,运行 serialtest 程序可以对串口进行数据收发测试。

该程序在运行时,需要提供一个命令行参数,该参数既可以是需要打开的串口名,如:“COM2”、“COM3”、“COM4”等;也可以是需要打开的设备名,如/dev/ttymxc1、/det/ttymxc2、/dev/ttymxc3。例如需要通过 COM2 口进行数据收发,在命令行下执行如下命令:

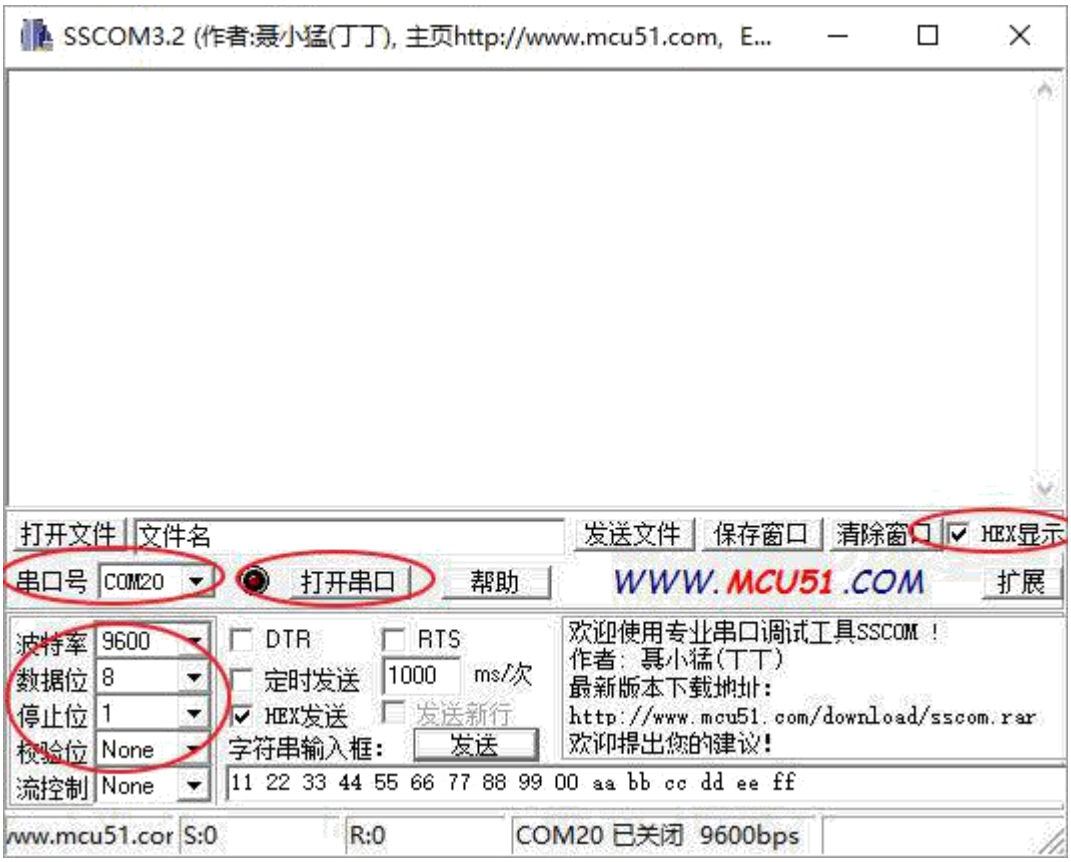
```
~ # cd /home/demo/uart
/home/demo/uart # ./serialtest COM2
```

该程序运行流程如下：

- 打开串口（串口通讯参数：波特率 115200，8 位数据位，1 位停止位，无数据校验位）;
- 通过串口发送一个 20 字节的数据;
- 从串口接收数据;
- 重复步骤 2~3，实现数据的循环发送和接收。

● 在主机上运行串口调试工具，该工具可接收开发板串口发送的数据，并可发送数据到开发板。Windows 下的串口调试软件比较多，此处以 sscom32.exe 为例介绍串口调试工具的使用，该工具可在“眺望产品光盘资料/2、软件开发参考资料/ 5、开发工具软件\SSCOM32”中直接获取。

运行 sscom32.exe 软件，弹出如下对话框，如下图所示：



在该对话框中设置主机侧的串口号，并设置串口通讯参数为：波特率 9600，8 位数据位，1 位停止位，无数据校验位。勾选“HEX 显示”选项。设置完成后，点击“打开串口”按钮。如果串口打开成功，sscom32 界面如下图所示：

如上图所示，红色标记部分表明串口打开正常。

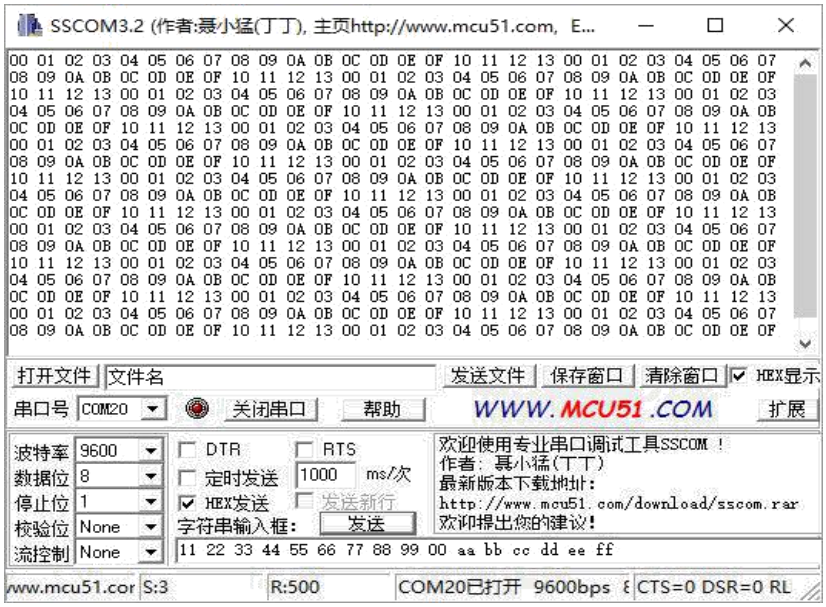
测试开发板发送数据是否正常。方法如下：

在开发板上运行 serialtest 测试程序。在命令行下执行如下命令：

```
~ # cd /home/demo/uart
```

```
/home/demo/uart # ./serialtest COM2
```

此处假定测试开发板 COM2 口。运行 serialtest 后，serialtest 会通过串口发送数据，此时可以在 sscom32 看到 serialtest 发送的数据，如下图所示：



测试开发板接收数据是否正常。方法如下：

在 sscom32 上，输入需要发送的数据，如果数据是二进制字节数据（即非 ASCII 字符）需勾选“HEX 发送”选项。设置完成后，点击“发送”按钮。如下图所示：

3.4 WIFI 测试

1、配置 WIFI 网络，首先进入下的 etc 目录

```
cd /etc
```

2、使能 wifi

输入命令 `rfkill unblock all`

```
root@twdz-IMX6ULL:/etc# rfkill unblock all
```

3、打开 wlan0

输入命令 ifconfig wlan0 up

```
root@twdz-IMX6ULL:/etc# ifconfig wlan0 up
==> rtl8188e_iol_efuse_patch
IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
```

先在开发板根文件系统的/etc 目录下创建一个名为“wpa_supplicant.conf”的配置文件，此文件用于配置要连接的 WIFI 热点以及 WIFI 密码，比如我要连接到“TWDZ”这个热点上，WIFI 密码为 twdz123456。

因此 wpa_supplicant.conf 文件内容如下所示：

```
ctrl_interface=/var/run/wpa_supplicant
ap_scan=1
network={
ssid="TWDZ"
psk="twdz123456"
}
```

4、连接 wifi

准备好以后就可以使用 wpa_supplicant 工具让 RTL8188 USB WIFI 连接到热点上，输入如下命令：wpa_supplicant -D wext -B -i wlan0 -c wpa_supplicant.conf

```
root@twdz-IMX6ULL:/home/wifi# wpa_supplicant -D wext -B -i wlan0 -c wpa_supplicant.conf
Successfully initialized wpa_suppRTL871X: set bssid:00:00:00:00:00:00
licant
ioctl[SIOCSIWAP]: Operation not RTL871X: set ssid [g▒isQ▒J▒).▒▒▒▒F|▒T▒▒vZ.c3▒>▒ↄc▒▒▒] fw_state=0x00000008
permitted
root@twdz-IMX6ULL:/home/wifi# RTL871X: indicate disassoc
RTL871X: set ssid [TWDZ] fw_state=0x00000008
RTL871X: set bssid:20:76:93:4e:b5:76
RTL871X: start auth
RTL871X: auth success, start assoc
RTL871X: assoc success
IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
RsvdPageNum: 8
RTL871X: recv eapol packet
RTL871X: send eapol packet
RTL871X: recv eapol packet
RTL871X: send eapol packet
RTL871X: set pairwise key camid:4, addr:20:76:93:4e:b5:76, kid:0, type:AES
RTL871X: set group key camid:5, addr:20:76:93:4e:b5:76, kid:2, type:TKIP
```

当 RTL8188 连接到 WIFI 热点上以后会输出“wlan0: CTRL-EVENT CONNECTED”字样。接下来就是最后一步了，设置 wlan0 的 IP 地址，这里使用 udhcpc 命令从路由器申请 IP 地址，输入如下命令：udhcpc -i wlan0

```
root@twdz-IMX6ULL:/etc# udhcpc -i wlan0
udhcp client (v0.9.8) started
Sending discover...
Sending select for 192.168.0.125...
Sending select for 192.168.0.125...
Lease of 192.168.0.125 obtained, lease time 86400
deleting routers
SIOCDELRT: No such process
adding dns 192.168.0.1
```

由上图可知，分配给 wlan0 的 IP 地址为 192.168.0.24。

5、wifi 测试

```
ifconfig eth0 down
ifconfig eth1 down
ping -I 192.168.0.24 www.baidu.com
```

上述命令中的 192.168.0.24 并非是固定的，可根据自身的分配到 IP 地址进行修改。如果只有 wlan0 工作时，可直接 ping www.baidu.com。

```
^Croot@twdz-IMX6ULL:/etc# ping baidu.com -c 3
PING baidu.com (220.181.38.148) 56(84) bytes of data.
64 bytes from 220.181.38.148: icmp_seq=1 ttl=53 time=191 ms
64 bytes from 220.181.38.148: icmp_seq=2 ttl=53 time=428 ms
64 bytes from 220.181.38.148: icmp_seq=3 ttl=53 time=860 ms

--- baidu.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 191.927/493.522/860.480/276.821 ms
```

3.5 4G 测试

方法一：首先进入\home 下的 4g 目录，直接执行./4g.sh

```
root@twdz-IMX6ULL:/# cd /home/4g/
root@twdz-IMX6ULL:/home/4g# ./4g.sh
```

```
root@twdz-IMX6ULL:/# cd home/4g/
root@twdz-IMX6ULL:/home/4g# ./4g.sh
mkdir: cannot create directory '/etc/ppp/': File exists
mkdir: cannot create directory '/etc/ppp/peers/': File exists
mkdir: cannot create directory '/var/lock': File exists
random: nonblocking pool is initialized
root@twdz-IMX6ULL:/home/4g# timeout set to 15 seconds
abort on (\nBUSY\r)
abort on (\nNO ANSWER\r)
abort on (\nRINGING\r\n\r\nRINGING\r)
timeout set to 40 seconds
send (AT^M)
expect (OK)
AT^M^M
OK
 -- got it

send (ATS0=0^M)
expect (OK)
^M
ATS0=0^M^M
OK
 -- got it

send (ATE0V1^M)
expect (OK)
^M
ATE0V1^M^M
OK
 -- got it

send (AT+CSQ^M)
expect (OK)
^M
^M
+CSQ: 26,99^M
^M
OK
 -- got it

send (AT+CGDCONT=1,"IP","CMNET"^M)
expect (OK)
^M
^M
OK
 -- got it
```

```
send (AT+CGDCONT=1,"IP","CMNET"^M)
expect (OK)
^M
^M
OK
 -- got it

send (ATDT*99***1#^M)
expect (CONNECT)
^M
^M
CONNECT
 -- got it

Script /usr/sbin/chat -s -v -f /etc/ppp/gprs-connect-chat finished (pid 674), status = 0x0
Serial connection established.
using channel 1
Using interface ppp0
Connect: ppp0 <--> /dev/ttyUSB2
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x7f219e8a> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x0 <asyncmap 0x0> <auth chap MD5> <magic 0x852ec875> <pcomp> <accomp>]
No auth is possible
sent [LCP ConfRej id=0x0 <auth chap MD5>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x7f219e8a> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x852ec875> <pcomp> <accomp>]
sent [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x852ec875> <pcomp> <accomp>]
sent [IPCP ConfReq id=0x1 <compress VJ 0f 01> <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
rcvd [LCP DiscReq id=0x2 magic=0x852ec875]
rcvd [IPCP ConfReq id=0x0]
sent [IPCP ConfNak id=0x0 <addr 0.0.0.0>]
rcvd [IPCP ConfRej id=0x1 <compress VJ 0f 01>]
sent [IPCP ConfReq id=0x2 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
rcvd [IPCP ConfReq id=0x1]
sent [IPCP ConfAck id=0x1]
rcvd [IPCP ConfNak id=0x2 <addr 10.170.20.11> <ms-dns1 221.179.38.7> <ms-dns2 120.196.165.7>]
sent [IPCP ConfReq id=0x3 <addr 10.170.20.11> <ms-dns1 221.179.38.7> <ms-dns2 120.196.165.7>]
rcvd [IPCP ConfAck id=0x3 <addr 10.170.20.11> <ms-dns1 221.179.38.7> <ms-dns2 120.196.165.7>]
Could not determine remote IP address: defaulting to 10.64.64.64
local  IP address 10.170.20.11
remote IP address 10.64.64.64
primary   DNS address 221.179.38.7
secondary DNS address 120.196.165.7
```

2.ping www.baidu.com

```
root@twdz-IMX6ULL:/home/4g# ping www.baidu.com
PING www.a.shifen.com (39.156.66.18) 56(84) bytes of data.
64 bytes from 39.156.66.18: icmp_seq=1 ttl=50 time=76.6 ms
64 bytes from 39.156.66.18: icmp_seq=2 ttl=50 time=59.0 ms
64 bytes from 39.156.66.18: icmp_seq=3 ttl=50 time=67.5 ms
64 bytes from 39.156.66.18: icmp_seq=4 ttl=50 time=109 ms

--- www.a.shifen.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 15453ms
rtt min/avg/max/mdev = 59.035/78.285/109.910/19.288 ms
```

方法二：

1.配置 4G 网络，首先进入\home 下的 4g 目录

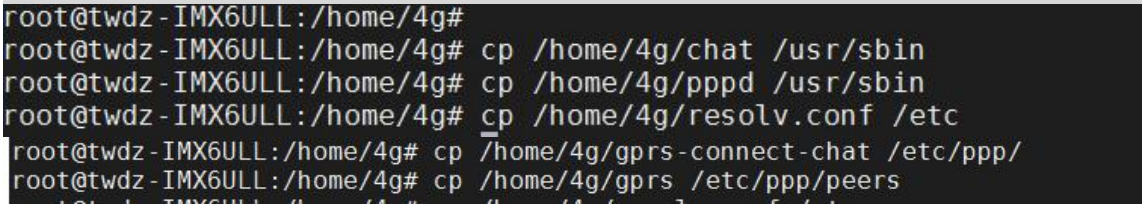
```
root@twdz-IMX6ULL:~# cd /home/4g/
root@twdz-IMX6ULL:/home/4g# ls
4g.sh  chat  gprs  gprs-connect-chat  picocom  pppd  resolv.conf
```

2. 在/etc/目录上，创建一下目录

```
root@twdz-IMX6ULL:/home/4g#mkdir /etc/ppp/
root@twdz-IMX6ULL:/home/4g# mkdir /etc/ppp/peers
root@twdz-IMX6ULL:/home/4g# mkdir /var/lock
root@twdz-IMX6ULL:/home/4g# mkdir /etc/ppp/
root@twdz-IMX6ULL:/home/4g# mkdir /etc/ppp/peers
root@twdz-IMX6ULL:/home/4g# mkdir /var/lock
```

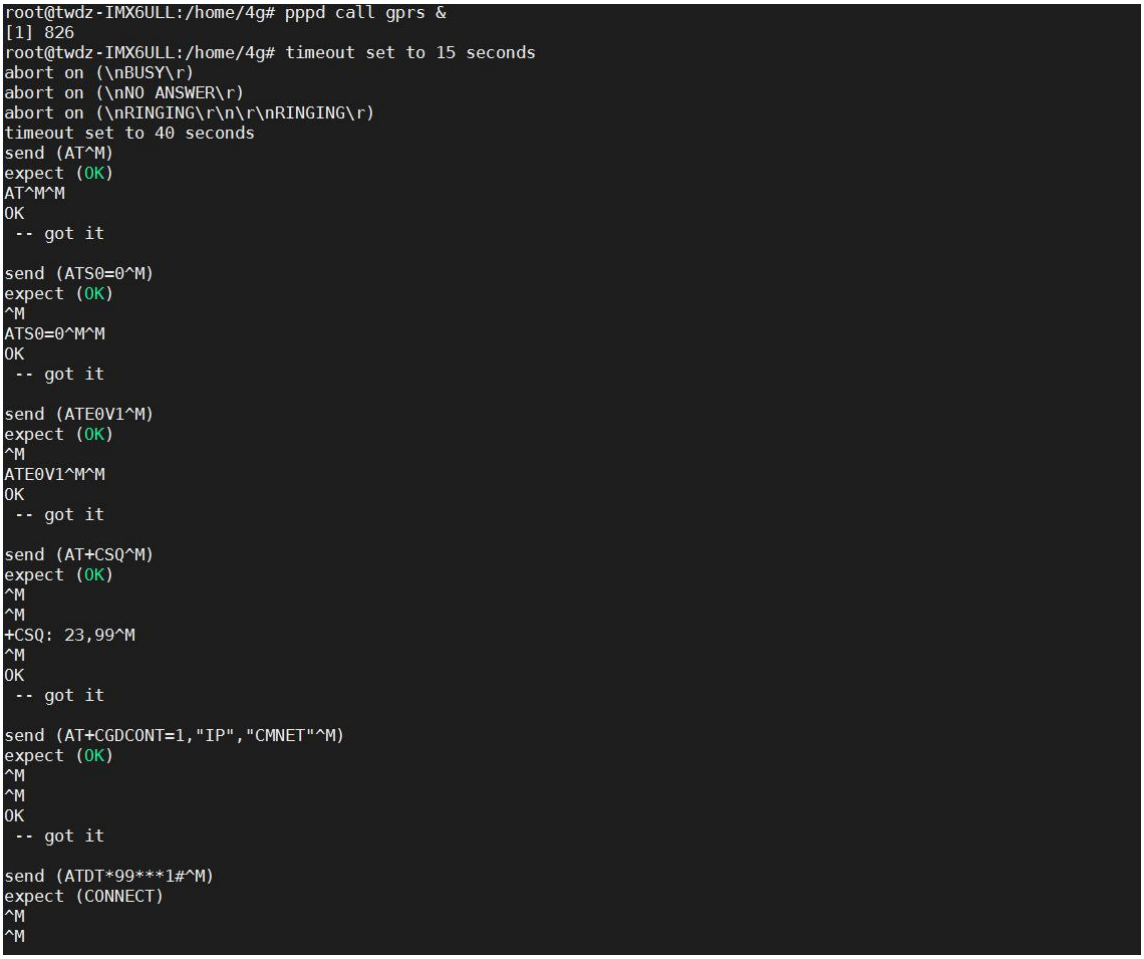
3. 将/home/4g 目录下的文件拷贝到其他目录，如下

```
root@twdz-IMX6ULL:/home/4g# cp /home/4g/chat /usr/sbin
root@twdz-IMX6ULL:/home/4g# cp /home/4g/pppd /usr/sbin
root@twdz-IMX6ULL:/home/4g# cp /home/4g/resolv.conf /etc
root@twdz-IMX6ULL:/home/4g# cp /home/4g/gprs-connect-chat /etc/ppp/
root@twdz-IMX6ULL:/home/4g# cp /home/4g/gprs /etc/ppp/peers
```



4.输入命令 pppd call gprs &，进行 4g 上网

```
root@twdz-IMX6ULL:/home/4g# pppd call gprs &
```



```
^M
^M
CONNECT
 -- got it

Script /usr/sbin/chat -s -v -f /etc/ppp/gprs-connect-chat finished (pid 827), status = 0x0
Serial connection established.
using channel 3
Using interface ppp0
Connect: ppp0 <--> /dev/ttyUSB2
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x580b2603> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x0 <asyncmap 0x0> <auth chap MD5> <magic 0x852484f8> <pcomp> <accomp>]
No auth is possible
sent [LCP ConfRej id=0x0 <auth chap MD5>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x580b2603> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x852484f8> <pcomp> <accomp>]
sent [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x852484f8> <pcomp> <accomp>]
sent [IPCP ConfReq id=0x1 <compress VJ 0f 01> <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
rcvd [LCP DiscReq id=0x2 magic=0x852484f8]
rcvd [IPCP ConfReq id=0x0]
sent [IPCP ConfNak id=0x0 <addr 0.0.0.0>]
rcvd [IPCP ConfRej id=0x1 <compress VJ 0f 01>]
sent [IPCP ConfReq id=0x2 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
rcvd [IPCP ConfReq id=0x1]
sent [IPCP ConfAck id=0x1]
rcvd [IPCP ConfNak id=0x2 <addr 10.90.213.175> <ms-dns1 221.179.38.7> <ms-dns2 120.196.165.7>]
sent [IPCP ConfReq id=0x3 <addr 10.90.213.175> <ms-dns1 221.179.38.7> <ms-dns2 120.196.165.7>]
rcvd [IPCP ConfAck id=0x3 <addr 10.90.213.175> <ms-dns1 221.179.38.7> <ms-dns2 120.196.165.7>]
Could not determine remote IP address: defaulting to 10.64.64.64
local  IP address 10.90.213.175
remote IP address 10.64.64.64
primary   DNS address 221.179.38.7
secondary DNS address 120.196.165.7
```

5.测试 4g

ping www.baidu.com

```
root@twdz-IMX6ULL:/home/4g# ping www.baidu.com
PING www.a.shifen.com (39.156.66.18) 56(84) bytes of data.
64 bytes from 39.156.66.18: icmp_seq=1 ttl=50 time=62.5 ms
64 bytes from 39.156.66.18: icmp_seq=2 ttl=50 time=57.8 ms
64 bytes from 39.156.66.18: icmp_seq=3 ttl=50 time=66.1 ms
64 bytes from 39.156.66.18: icmp_seq=4 ttl=50 time=56.6 ms
64 bytes from 39.156.66.18: icmp_seq=5 ttl=50 time=79.5 ms
64 bytes from 39.156.66.18: icmp_seq=6 ttl=50 time=68.7 ms
64 bytes from 39.156.66.18: icmp_seq=7 ttl=50 time=70.6 ms
64 bytes from 39.156.66.18: icmp_seq=8 ttl=50 time=79.1 ms

--- www.a.shifen.com ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 36270ms
rtt min/avg/max/mdev = 56.637/67.677/79.547/8.128 ms
```

3.6 时钟设置

Linux 将时钟分为系统时钟(System Clock)和硬件时钟(Real Time Clock，简称 RTC)两种。系统时钟是由 Linux 内核所维护的时钟，用户一般使用和看到的都是系统时钟。而硬件时钟则是由主板上的电池供电的主板硬件时钟。系统时钟在系统断电后即会消失，但 RTC 时钟在主板电池有电的情况下会长期运行。因此每次上电时，Linux 内核都会读取主板上的 RTC 时钟，并将它同步到系统时钟。下面列出一些与时钟相关的命令：

3.6.1 查看系统时钟：

使用 date 命令可以查看系统时钟：

```
root@twdz-IMX6ULL:/# date
Mon Dec  7 10:00:42 UTC 2020
```

3.6.2 查看 RTC 时钟：

使用 hwclock 命令可以查看 RTC 时钟：

```
root@twdz-IMX6ULL:/# hwclock
Mon Dec  7 10:01:30 2020  0.000000 seconds
```

3.6.3 设置 RTC 时钟：

使用 hwclock -w，可以将系统时钟写入 RTC 时钟：

```
root@twdz-IMX6ULL:/# hwclock -w
```

3.6.4 同步系统时钟：

使用 hwclock -s，可以将 RTC 时钟写入系统时钟：

```
root@twdz-IMX6ULL:/# hwclock -s
```

通过上面的叙述可以看出，如果想要改变当前的系统时间，且希望系统重启后改变依然生效，需要执行如下两步操作：

- 使用 date －s 命令修改当前的系统时钟;
- 使用 hwclock -w 命令将修改后的系统时钟写入 RTC 时钟。

例如需要将当前时钟设置为 2020-12-07 10：03：10，并希望该改变在系统重启后依然有效，应执行如下命令：

```
root@twdz-IMX6ULL:/# date -s "2020-12-07 10:03:10"
Mon Dec  7 10:03:10 UTC 2020
root@twdz-IMX6ULL:#  hwclock －w
```

重启计算机后，如果 RTC 正常的话，使用以下命令可以查看到刚刚设定的时间值：

```
root@twdz-IMX6ULL:/# hwclock
```

3.7 CAN 测试

1. 将 CAN1 和 CAN2 使用杜邦线或其他连接起来。
2. 执行/home/demo 目录下的 cantest 例子可验证是否正常通讯。
3. 命令如下： ./cantest can1 & 之后，再执行./cantest can2，正常如下图所示。

```
can write: length=4, packet: 10 11 12 13
can read ,length=4,frame id=1,packet: 10 11 12 13
can write: length=8, packet: 00 01 02 03 04 05 06 07
can write: length=8, packet: 08 09 0A 0B 0C 0D 0E 0F
can read ,length=8,frame id=1,packet: 00 01 02 03 04 05 06 07
can read ,length=8,frame id=1,packet: 00 01 02 03 04 05 06 07
can read ,length=8,frame id=1,packet: 00 01 02 03 04 05 06 07
can write: length=8, packet: 00 01 02 03 04 05 06 07
can read ,length=8,frame id=1,packet: 00 01 02 03 04 05 06 07
can read ,length=8,frame id=1,packet: 00 01 02 03 04 05 06 07
can read ,length=8,frame id=1,packet: 00 01 02 03 04 05 06 07
```

3.8 摄像头测试

注：在测试之前请接好相应的摄像头，注意引脚顺序，执行命令后就可以在液晶上看到相应的摄像头画面，TW-AC6G-EVM 开发板提供 CSI 摄像头接口：

在控制端输入 `v4l2-ctl --device=/dev/video0 --list-formats-ext` 查看摄像头支持格式、分辨率及帧率。

```
root@twdz-IMX6ULL:/# v4l2-ctl --device=/dev/video0 --list-formats-ext
ioctl: VIDIOC_ENUM_FMT
        Index       : 0
        Type        : Video Capture
        Pixel Format: 'YUYV'
        Name        : YUYV-16
                Size: Discrete 640x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x576
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1280x720
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1920x1080
                        Interval: Discrete 0.067s (15.000 fps)
                Size: Discrete 2592x1944
                        Interval: Discrete 0.067s (15.000 fps)
                Size: Discrete 176x144
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1024x768
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
```

为测试 CSI 接口, 在命令行下执行如下命令，采集图像显示于 LCD 屏：

```
root@twdz-IMX6ULL:/# /usr/unit_tests/V4L2/mx6s_v4l2_capture.out -ow 800 -oh 480 -m 2 -d /dev/video0 -t 50
```

3.9 网络测试

TW-AC6G-EVM 评估板有两路百兆以太网接口 CN39,CN40，使用标准的 RJ45 网口插座，插座内带状态指示灯。

可使用 ifconfig 指令来显示或者配置网络，查看网络信息。

```
root@twdz-IMX6ULL:~# ifconfig eth0 && ifconfig eth1
```

```
root@twdz-IMX6ULL:/#  ifconfig eth0 && ifconfig eth1fec 20b4000.ethernet eth0: Link is Down

eth0      Link encap:Ethernet  HWaddr 5a:0a:6d:92:4c:87
          inet6 addr: fe80::580a:6dff:fe92:4c87/64 Scope:Link
          inet6 addr: 240e:3b8:14f1:1740:580a:6dff:fe92:4c87/64 Scope:Global
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:162 errors:0 dropped:23 overruns:0 frame:0
          TX packets:87 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:17099 (16.6 KiB)  TX bytes:13831 (13.5 KiB)

eth1      Link encap:Ethernet  HWaddr 22:bb:25:49:0b:86IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready

          BROADCAST MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

插上网线到以太网接口处可以看到如下信息，系统自动获取了 ip，eth1 同理。

```
root@twdz-IMX6ULL:/# fec 20b4000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
```

```
root@twdz-IMX6ULL:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 2a:02:3c:22:87:8e
          inet addr:192.168.0.44  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::2802:3cff:fe22:878e/64 Scope:Link
          inet6 addr: 240e:3b8:14f1:1740:2802:3cff:fe22:878e/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:34 errors:0 dropped:3 overruns:0 frame:0
          TX packets:74 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5138 (5.0 KiB)  TX bytes:12402 (12.1 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:140 (140.0 B)  TX bytes:140 (140.0 B)
```

如果对应网卡没有自动获取到 IP，请使用下面的指令获取。“-i”是指定网卡名称，如不指定，会使用默认会使用 eth0。

```
root@twdz-IMX6ULL:~# udhcpc -i eth0
```

关闭与打开网口：

```
root@twdz-IMX6ULL:~# ifconfig eth0 down //关闭网口,网卡名字请根据实际情况修改，down 表示关闭
root@twdz-IMX6ULL:~# ifconfig eth0 up      //打开网口，网卡名字请根据实际情况修改，up 表示打开
```

测试网口是否能上网，以访问 www.baidu.com 为例，执行如下命令，“-I”代表指定网口， 不加“-I”则使用默认网卡（默认网卡指的是有网络接入的一端，如果两个网口都有网络接入，则使用 eth0 作为默认网卡）。按“Ctrl+c”终止 ping 指令。百度的实际地址根据网络运营商不同，访问的地址会不同。

```
root@twdz-IMX6ULL:~# ping www.baidu.com -I eth0
```

```
root@twdz-IMX6ULL:~#  ping www.baidu.com -I eth0
PING www.baidu.com (14.215.177.38) from 192.168.0.44 eth0: 56(84) bytes of data.
64 bytes from 14.215.177.38: icmp_seq=1 ttl=56 time=7.34 ms
64 bytes from 14.215.177.38: icmp_seq=2 ttl=56 time=7.54 ms
64 bytes from 14.215.177.38: icmp_seq=3 ttl=56 time=7.26 ms
64 bytes from 14.215.177.38: icmp_seq=4 ttl=56 time=7.18 ms
^C
--- www.baidu.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 7.186/7.333/7.540/0.144 ms
```

使用 route 命令查看网关后，并 ping 网关。

```
root@twdz-IMX6ULL:~# route
```

```
root@twdz-IMX6ULL:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.0.1     0.0.0.0         UG    10     0        0 eth0
192.168.0.0     *               255.255.255.0   U     0      0        0 eth0
```

由上可知网关为 gao.ke，根据路由器不同，网关可能不同。ping 网关可测试内网与开发板连接是否正常。下面指令不加“-I”参数，使用默认网卡。

```
root@twdz-IMX6ULL:~# ping gao.ke
```

```
root@twdz-IMX6ULL:~# ping gao.ke
PING g.17986.net (104.160.174.190) 56(84) bytes of data.
64 bytes from contact.MYSCORESHOME.COM (104.160.174.190): icmp_seq=1 ttl=53 time=162 ms
64 bytes from 190.174.160.104.in-addr.arpa (104.160.174.190): icmp_seq=2 ttl=53 time=170 ms
64 bytes from 190.174.160.104.in-addr.arpa (104.160.174.190): icmp_seq=3 ttl=53 time=164 ms
64 bytes from 190.174.160.104.in-addr.arpa (104.160.174.190): icmp_seq=4 ttl=53 time=170 ms
^C
--- g.17986.net ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 162.655/166.764/170.175/3.448 ms
```

3.10 音频功能

3.10.1 Headphone 测试&Speaker 测试

开发板系统音频输出功能默认是打开的，下面两条指令可不执行。

```
root@twdz-IMX6ULL:/# amixer sset 'Left Output Mixer PCM' on
root@twdz-IMX6ULL:/# amixer sset ' Right Output Mixer PCM' on
```

设置播放音量，执行如下命令，音量的单位是 dB，音量最小为 0，最大为 127。

```
root@twdz-IMX6ULL:/# amixer sset Headphone 110,110
root@twdz-IMX6ULL:/# amixer sset Speaker 110,110
```

接入耳机或扬声器，播放开发板文件系统自带的音频，执行下面指令：

```
root@twdz-IMX6ULL:/# aplay /usr/share/sounds/alsa/Front_Center.wav
root@twdz-IMX6ULL:/# aplay /usr/share/sounds/alsa/Front_Left.wav
root@twdz-IMX6ULL:/# aplay /usr/share/sounds/alsa/Front_Right.wav
```

执行播放音频的指令测试有声音出来即可。

3.10.2 MIC IN 录音测试

执行如下命令将录音前耳机和扬声器的音量设置为 0，防止干扰：

```
root@twdz-IMX6ULL:/# amixer sset 'Headphone Playback ZC' off
root@twdz-IMX6ULL:/# amixer sset Headphone 0,0
root@twdz-IMX6ULL:/# amixer sset 'Speaker Playback ZC' off
root@twdz-IMX6ULL:/# amixer sset Speaker 0,0
```

开始录音 10 秒钟：

```
root@twdz-IMX6ULL:/# arecord -r 44100 -f S16_LE -c 2 -d 10 record.wav
```

再次打开耳机和扬声器

```
root@twdz-IMX6ULL:/# amixer sset 'Headphone Playback ZC' on
root@twdz-IMX6ULL:/# amixer sset Headphone 125,125
root@twdz-IMX6ULL:/# amixer sset 'Speaker Playback ZC' on
root@twdz-IMX6ULL:/# amixer sset Speaker 125,125
```

播放录音。

```
root@twdz-IMX6ULL:/# aplay record.wav
```

3.11 TF 卡测试

将 TF 卡插入到开发板 TF 卡插槽中,Linux 操作系统会检测到 TF 卡,并在控制台终端上打印 TF 卡的相关信息,例如:

```
root@twdz-IMX6ULL:~# mmc0: host does not support reading read-only switch, assuming write-enable
mmc0: new high speed SDHC card at address b368
```

```
mmcblk0: mmc0:b368 SD08G 7.41 GiB
 mmcblk0: p1
FAT-fs (mmcblk0p1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
```

在此例中,从打印信息可以看出 Linux 操作系统检测到一个容量为 8GB 的 TF 卡,其对应的设备名为 mmcblk2,且 TF 卡有 1 个分区 p1。

假定 TF 卡被挂载到/run/media/mmcblk2p1 目录上,为了将 TF 卡根目录上的 test 文件拷贝到/home 目录下,可执行如下命令:

```
root@twdz-IMX6ULL:/#cp /run/media/mmcblk2p1/test /home
```

为了将/home 目录下的 test1 文件拷贝到 TF 卡根目录上,可执行如下命令:

```
root@twdz-IMX6ULL:/#cp /home/test1 /run/media/mmcblk2p1
```

TF 卡使用完毕后,在拔出 TF 卡前,需执行 umount 命令卸载 TF 卡所有的分区:

```
root@twdz-IMX6ULL:/#umount /run/media/mmcblk2p1
```

此处假定 TF 卡被挂载在/run/media/mmcblk2p1 目录下。umount 会确保所有缓存的数据都被正确的写入 TF 卡。在 umount 成功后,即可拔出 TF 卡。在调用 umount 前,必须确保 TF 卡上的文件没有被其他程序所占用且用户当前的工作目录不在 TF 卡的挂载目录上,否则调用 umount 会提示失败,如下所示:

```
umount: /run/media/mmcblk2p1/: target is busy (In some cases useful info about processes that use the device is found by lsof(8) or fuser(1).)
```

3.12 U 盘使用

将格式为 FAT32 的 U 盘插入到开发板 USB HOST 接口上,Linux 操作系统会检测到 U 盘,并在控制台终端上打印 U 盘的相关信息,例如:

```
root@twdz-IMX6ULL:~# usb 1-1.2: new high-speed USB device number 4 using ci_hdrc
usb-storage 1-1.2:1.0: USB Mass Storage device detected
scsi host0: usb-storage 1-1.2:1.0
scsi 0:0:0:0: Direct-Access     SMI      USB DISK         1100 PQ: 0 ANSI: 4
sd 0:0:0:0: [sda] 15769600 512-byte logical blocks: (8.07 GB/7.51 GiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] No Caching mode page found
sd 0:0:0:0: [sda] Assuming drive cache: write through
sda: sda1
sd 0:0:0:0: [sda] Attached SCSI removable disk
FAT-fs (sda1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
```

在此例中,从打印信息可以看出 Linux 操作系统检测到一个容量为 8GB 的 U 盘,其对应的设备名为 sda。

系统会在/mnt 目录下为 U 盘的每个分区都生成一个目录,目录的名字为 sdxn(x 用于区分不同的 U 盘,n 用于区分不同的分区,x=a、b、c…… n=1、2、3……)。U 盘的每个分区就挂载在这些目录下。U 盘挂载成功后,即可对 U 盘进行文件查看、文件拷贝等操作。以下以文件拷贝操作为例,介绍 U 盘的使用。

假定 U 盘被挂载到/run/media/sda1 目录上,为了将 U 盘根目录上的 test 文件拷贝到/home 目录下,可执行如下命令:

```
root@twdz-IMX6ULL:/# cp /run/media/sda1/test /home
```

为了将/home 目录下的 test1 文件拷贝到 U 盘根目录上,可执行如下命令:

```
root@twdz-IMX6ULL:/# cp /home/test1 /run/media/sda1
```

U 盘使用完毕后,在拔出 U 盘前,需执行 umount 命令卸载 U 盘所有的分区:

```
root@twdz-IMX6ULL:/# umount /run/media/sda1/
```

此处假定 U 盘被挂载在/run/media/sda1 目录下。umount 会确保所有缓存的数据都被正确的写入 U 盘。在 umount 成功后,即可拔出 U 盘。在调用 umount 前,必须确保 U 盘上的文件没有被其他程序所占用且用户当前的工作目录不在 U 盘的挂载目录上,否则调用 umount 会提示失败,如下所示:

```
umount: /run/media/sda1/: target is busy(In some cases useful info about processes that use the device is found by lsof(8) or fuser(1).)
```

3.13 USB 鼠标与 USB 键盘使用

将 USB 鼠标插入到开发板 USB HOST 接口上，Linux 操作系统会检测到 USB 鼠标，并在控制台终端上打印 USB 鼠标的相关信息，例如：

```
usb 1-1.2： new low-speed USB device number 4 using ci_hdrc
input : USB Optical Mouse as /devices/soc0/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1/1-1.2/1-1.2：1.0/0003：1BCF：0007.0001/input/ input2
hid-generic 0003：1BCF：0007.0001： input： USB HID v1.10 Mouse [USB Optical Mouse] on usb-ci_hdrc.1-1.2/input0
```

将 USB 键盘插入到开发板 USB HOST 接口上，Linux 操作系统会检测到 USB 键盘，并在控制台终端上打印 USB 键盘的相关信息，例如：

```
usb 1-1.2： new low-speed USB device number 6 using ci_hdrc
input： USB USBKeykoard as
/devices/soc0/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1/1-1.2/1-1.2：1.0/0003：1A2C：0002.0004/input/ input5
hid-generic 0003：1A2C：0002.0004： input： USB HID v1.10 Keyboard [USB USBKeykoard] on usb-ci_hdrc.1-1.2/input0
input： USB USBKeykoard as
/devices/soc0/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1/1-1.2/1-1.2：1.1/0003：1A2C：0002.0005/input/ input6
hid-generic 0003：1A2C：0002.0005： input： USB HID v1.10 Device [USB USBKeykoard] on usb-ci_hdrc.1-1.2/input1
```

3.14 LCD 背景亮度调节

LCD 屏幕的背光支持 8 级变化，亮度级数为 0～7。可以通过以下命令查看等级：

```
root@twdz-IMX6ULL:~# cd /sys/class/backlight/backlight
root@twdz-IMX6ULL:/sys/class/backlight/backlight# cat max_brightness
7
```

查看当前 LCD 屏幕背光亮度等级，执行命令如下：

```
root@twdz-IMX6ULL:/sys/class/backlight/backlight# cat actual_brightness
7
```

修改当前屏幕背光亮度等级，执行命令如下：

```
root@twdz-IMX6ULL:/sys/class/backlight/backlight# echo 2 > brightness
```

3.15 USB 接口测试

TW-AC6G-EVM 评估板有 2 路 USB2.0 HOST 接口 CN20，使用标准的双层 USB-A 插

座。1 路 USB2.0 HOST 接口 CZ1 和 CN21 二选一使用, CZ1 使用标准的 PH2.0-4A 插座，CN21 使用侧插 USB-AF 插座，方便用户扩展接口。

TW-AC6G-EVM 评估板有 1 路 Micro_USB 接口 CN38，使用标准的 Micro_USB 插座，该接口默认为 Device 接口。

TW-AC6G-EVM 评估板 USB/OTG 接口说明：

- USB_HOST1~USB_HOST3 为 HOST 模式，默认为 HOST 模式；
- USB1_HOST 支持 HOST 模式；USB_OTG1 支持切换为 HOST/DEVICE

以下测试根据使用的 U 盘的不同和实验环境不同，测试结果会有所差异。将 FAT32 格式 U 盘插到开发板 USB_HOST1~USB_HOST3/USB1_HOST 其中一个接口。

插入后会打印如下信息，可以从中看到 U 盘大小和挂载名，如下图所示：

```
root@twdz-IMX6ULL:/# usb 1-1.2: new high-speed USB device number 6 using ci_hdrc
usb-storage 1-1.2:1.0: USB Mass Storage device detected
scsi host1: usb-storage 1-1.2:1.0
scsi 1:0:0:0: Direct-Access     Kingston DataTraveler 3.0      PQ: 0 ANSI: 6
sd 1:0:0:0: [sda] 60437492 512-byte logical blocks: (30.9 GB/28.8 GiB)
sd 1:0:0:0: [sda] Write Protect is off
sd 1:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
 sda: sda1
sd 1:0:0:0: [sda] Attached SCSI removable disk
FAT-fs (sda1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
```

输入 df -h 查看 U 盘的挂载路径，可以看到下图 U 盘已经挂载在/run/media/下，挂载名为 sda1。

```
root@twdz-IMX6ULL:/# df -h
Filesystem                Size  Used Avail Use% Mounted on
/dev/root                 3.0G  421M  2.5G  15% /
devtmpfs                  485M  4.0K  485M   1% /dev
tmpfs                     503M  176K  502M   1% /run
tmpfs                     503M  200K  502M   1% /var/volatile
/dev/mmcblk1p1            500M  7.7M  493M   2% /run/media/mmcblk1p1
106.15.73.137:/tftpboot    40G   31G  6.6G  83% /mnt
/dev/sda1                  29G  1.9G   27G   7% /run/media/sda1
```

写速度测试：

输入 time dd if=/dev/zero of=/run/media/sda1/test bs=1024k count=100 conv=fdatasync

```
root@twdz-IMX6ULL:/# time dd if=/dev/zero of=/run/media/sda1/test bs=1024k count
=100 conv=fdatasync
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 11.9505 s, 8.8 MB/s

real    0m11.971s
user    0m0.000s
sys     0m1.820s
```

本次写 100MiB，速度为 8.8MB/s。

读速度测试：

执行 echo 3 > /proc/sys/vm/drop_caches 指令清除缓存。

```
root@twdz-IMX6ULL:/# echo 3 > /proc/sys/vm/drop_caches
sh (616): drop_caches: 3
```

执行 time dd if=/run/media/sda1/test of=/dev/null bs=1024k 读取前面 dd 指令写入的 test 文件。

```
root@twdz-IMX6ULL:/# time dd if=/run/media/sda1/test of=/dev/null bs=1024k
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 4.57857 s, 22.9 MB/s

real    0m4.622s
user    0m0.020s
sys     0m0.970s
```

这里一共读出了 100 MiB test 文件，速度为 22.9 MB/s。

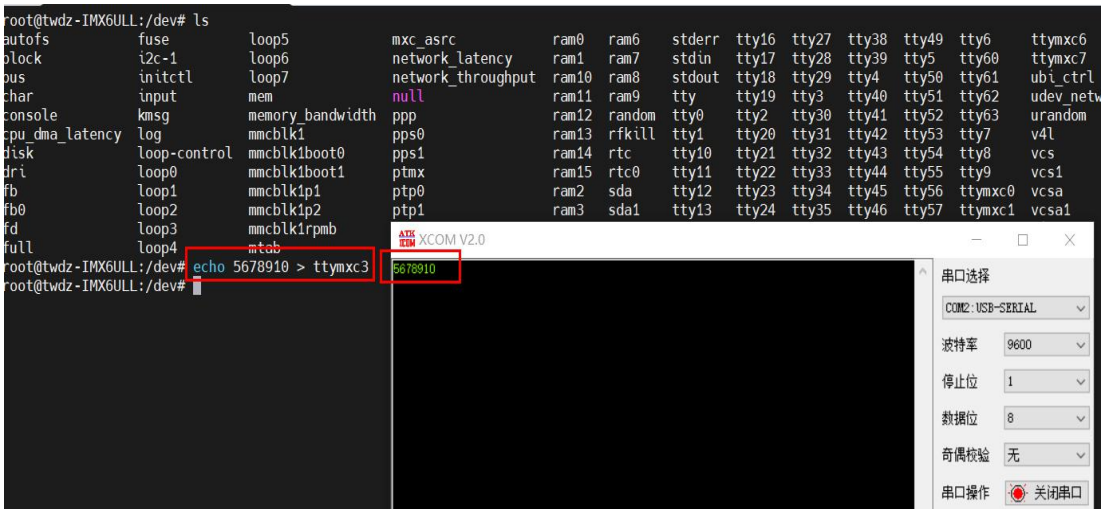
3.16 RS485 测试

本次测试中开发板使用 COM4 即 ttymxc3,要短接 JP1 的 1、2 引脚，详细可查看《TW-AC6G-EVM 评估板数据手册》4.8.3 小节。

设置电脑端的串口调试设置如下图所示（波特率 9600，停止位 1 位，数据位 8 位，无奇偶校验）

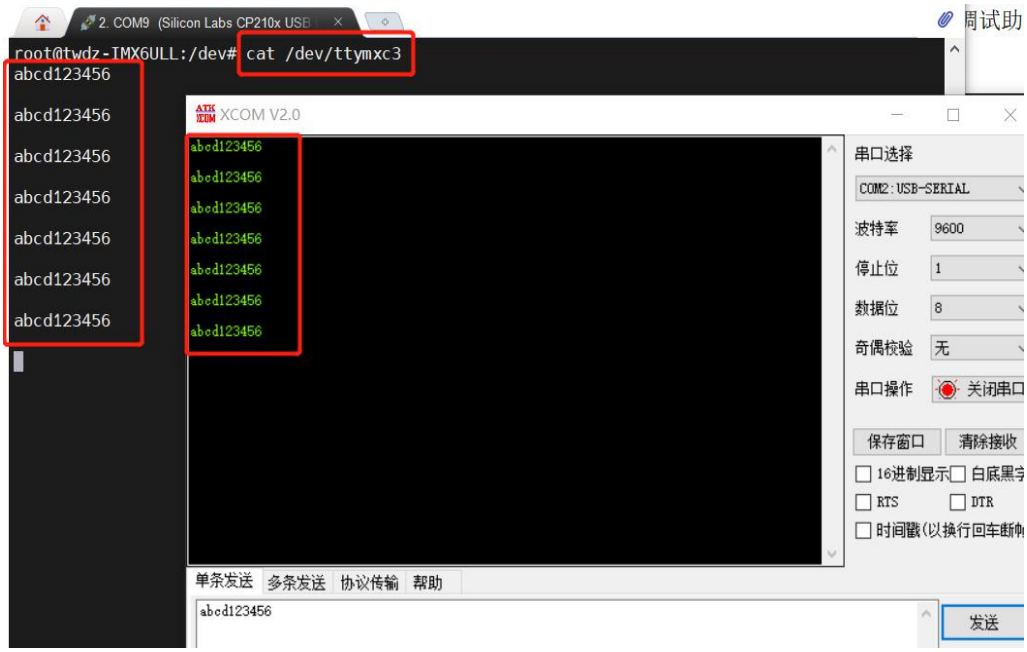
开发板发数据，电脑端接收数据：

在开发板中进入/dev 目录，输入 echo 5678910 > ttymxc3，在电脑串口调试助手可以接收到相应的数据，如下图所示。



电脑端发送数据，开发板接收并回传数据：

在开发板中进入/dev 目录，输入 cat /dev/ttymxc3 指令，在电脑串口调试助手发送数据后可以接收到相同的数据，如下图所示。



4. 例程案例

4.1 Serial 案例

该示例程序源码位于“眺望产品光盘资料/2、软件开发参考资料/2、示例 demo/serial_test”目录下。

1、原始状态下，串口之间的通信有某些参数是已经固定的，如波特率是 9600，时间超时，发送的打印信息等参数。

2、用户可以根据自己的情况去修改源代码的某些参数和语句。如在代码在将 argv[1]改成固定的串口号，可以方便我们在执行应用程序时，只需写./serial_test 等，修改说明可参考下图所示。

```
        if (argc != 2)
        {
              fprintf(stderr, "usage:serialtest com_port\n");
              return -1;
        }

        ret = serial_open(argv[1], 9600, SERIAL_PARITY_NO, 8,
SERIAL_STOPBIT_ONE, 1000);
        if (ret == 0)
        {
              fprintf(stderr, "open serial failed\n");
              return -1;
        }

        for(i=0; i<20; i++)
        {
              send_buffer[i] = i;
        }

        while(1)
        {
              serial_write(send_buffer, 20);
              //read
              while (serial_poll(1000)>0)
              {
                    serial_read(recv_buffer, sizeof(recv_buffer));
              }
        }

        serial_close();
```

argv[1]:在命令行输入的第二个参数(串口号)；波特率：9600；校验位：0；数据位：8；停止位：1，时间超时：1000（以上参数都可以自行去修改设定）

可自行修改发送的内容

将send_buffer存放的内容发送出去

接收其他串口发送过来的信息

4.2 LED 案例

该示例程序源码位于”眺望产品光盘资料/2、软件开发参考资料/2、示例 demo/led_test”目录下。

1、源码案例利用了地址映射去控制 LED 灯的亮灭，初始源码控制的 GPIO1_2 引脚口。

2、可以适当的修改对应的引脚口的地址，自主控制高低电平。

```
#define GPIO_START_ADDR 0x209c000

//LED toggle use GPIO1_2
#define GPIO1_DIR                   0x209c004
#define GPIO1_DOUT                  0x209c000

static void *map_base_gpio = NULL;

static int mem_fd = -1;

static void mem_write(unsigned int addr, unsigned int value)
{
      volatile unsigned int *virt_addr;
      if ( (addr & ADDR_MASK) == GPIO_START_ADDR)
      {
            virt_addr = map_base_gpio + (addr & MAP_MASK);
            *virt_addr =  value;
      }
}

static void set_led(int is_on)
{
      unsigned int value = mem_read(GPIO1_DOUT);
      if (is_on)
            mem_write(GPIO1_DOUT, value | 0x4);  //set GPIO1_2 data to 1
      else
            mem_write(GPIO1_DOUT, value & (~0x4));  //set GPIO1_2 data to 0

}
```

初始地址

GPIO口对应的地址

可以设置高低电平

4.3 Hello World

该示例程序源码位于“眺望产品光盘资料/2、软件开发参考资料/2、示例 demo /HelloWorld”目录下。用户可以直接运行开发板上的/home/demo/helloworld 程序来查看该示例程序的运行结果。

该例程功能十分简单,就是在控制台上打印“Hello World”字符串。关于该示例程序的具体说明可参考“Hello,World!”一节的内容。

4.4 蜂鸣器示例

该关于该示例程序的具体说明可参考“蜂鸣器测试”一节的内容。

4.5 串口编程示例

该示例程序源码位于“眺望产品光盘资料/2、软件开发参考资料/2、示例 demo/serial_test”目录下。用户可以直接运行开发板上的/home/demo/uart/serialtest 程序来查看该示例程序的运行结果。

该例程可以通过指定的串口收发数据。关于该示例程序的具体说明可参考“串口测试”一节的内容。

4.6 CAN 编程示例

该示例程序源码位于“眺望产品光盘资料/2、软件开发参考资料/2、示例 demo/can_test”目录下。用户可以直接运行开发板上的/home/demo/can/cantest 程序来查看该示例程序的运行结果。

TWEvb-IMX6GY 开发板有 2 个 CAN 口。运行 cantest 程序可以通过 CAN 口收发数据。

该程序在运行时,需要提供一个命令行参数,即需要打开的 CAN 口名,这个 CAN 名参数可以为“can1”、“can2”。例如需要通过 CAN1 口进行数据收发,在命令行下执行如下命令:

```
~ # cd /home/demo/can

/home/demo/can # ./cantest can1
```

该程序运行流程如下:

- 打开 CAN1 口,其中 CAN1 口的通讯速率为 125000;
- 通过 CAN1 口发送一个 20 字节的数据;
- 从 CAN1 口接收数据;
- 重复步骤 2~3,实现数据的循环发送和接收。

4.7 网络编程示例

该示例程序源码位于“眺望产品光盘资料/2、软件开发参考资料/2、示例 demo/ network”目录下。网络编程共包含 4 个例程,分别为 tcp 客户端、tcp 服务器、udp 客户端、 udp 服务器,具体说明如下:

示例程序	源码路径	开发板程序路径
tcp 客户端	tcp_client_test	/home/demo/network/ tcp_client_test
tcp 服务器	tcp_server_test	/home/demo/network/ tcp_server_test

udp 客户端	udp_client_test	/home/demo/network/ udp_client_test
udp 服务器	udp_server_test	/home/demo/network/ udp_server_test

表 4-1 网络编程示例

其中 tcp_client_test 和 tcp_server_test 为 tcp 通讯例程，可组合在一起进行测试；udp_client_test 和 udp_server_test 为 udp 通讯例程，可组合在一起进行测试。

1. tcp 通讯例程

tcp_client_test 在运行时，需要提供两个命令行参数，一个为服务器的 IP 地址，一个为服务器的端口号。例如需要和 ip 地址为 127.0.0.1，端口号为 9001 的本机服务器通讯，在命令行下执行如下命令：

```
~ # cd /home/demo/network
/home/demo/network # ./tcp_client_test 127.0.0.1 9001
```

该程序运行流程如下：

- 与服务器建立连接；
- 向服务器发送字符串数据；
- 从服务器接收字符串数据；
- 重复步骤 2~3，实现数据的循环发送和接收。

tcp_server_test 在运行时，需要提供一个命令行参数，即服务器侦听端口号。例如服务器需要在 9001 端口号上进行侦听，在命令行下执行如下命令：

```
/ # cd /home/demo/network
/home/demo/network # ./tcp_server_test 9001
```

该程序运行流程如下：

- 接受来自客户端的连接请求；
- 从客户端接收字符串数据；
- 向客户端发送字符串数据；
- 重复步骤 2~3，实现数据的循环接收和发送。

在实际测试时，可以在主机上运行两个终端，一个终端运行 tcp_client_test，命令如下：

```
/# cd /home/demo/network
/home/demo/network # ./tcp_client_test 127.0.0.1 9001
```

另一个终端运行 tcp_server_test，命令如下：

```
/home/demo/network # ./tcp_server_test 9001
```

当 tcp_client_test 和 tcp_server_test 都正常运行后，就可以在各自终端上看到报文收发的打印信息。

2. udp 通讯例程

udp_client_test 在运行时，需要提供两个命令行参数，一个为服务器的 IP 地址，一个为

服务器的端口号。例如需要和 ip 地址为 127.0.0.1, 端口号为 9001 的本机服务器通讯, 在命令行下执行如下命令:

```
~ # cd /home/demo/network

/home/demo/network # ./udp_client_test 127.0.0.1 9001
```

该程序运行流程如下:

- 与服务器建立连接;
- 向服务器发送字符串数据;
- 从服务器接收字符串数据;
- 重复步骤 2~3,实现数据的循环发送和接收。

udp_server_test 在运行时,需要提供一个命令行参数,即服务器侦听端口号。例如服务器需要在 9001 端口号上进行侦听,在命令行下执行如下命令:

```
~ # cd /home/demo/network
/home/demo/network # ./udp_server_test 9001
```

该程序运行流程如下:

- 从客户端接收字符串数据;
- 向客户端发送字符串数据;
- 重复步骤 1~2,实现数据的循环接收和发送。

在实际测试时,可以在主机上运行两个终端,一个终端运行 udp_client_test,命令如下:

```
~ # cd /home/demo/network
/home/demo/network # ./udp_client_test 127.0.0.1 9001
```

另一个终端运行 udp_server_test,命令如下:

```
/home/demo/network # ./udp_server_test 9001
```

当 udp_client_test 和 udp_server_test 都正常运行后,就可以在各自终端上看到报文收发的打印信息。

4.7.1 数据库编程示例

该例程使用 sqlite 数据库。sqlite 数据库的源码可以从“www.sqlite.org/download.html”下载或者从“眺望产品光盘资料/2、软件开发参考资料/3、软件源码/sqlite-autoconf-3140100.tar.gz”中直接获取。

该示例程序源码位于“眺望产品光盘资料/2、软件开发参考资料/ 2、示例 demo/sqlite_test”目录下。用户可以直接运行开发板上的/home/demo/sqlite_test 程序来查看该示例程序的运行结果。

sqlite_test 程序运行流程如下:

- 创建 test.db 数据库;
- 在数据库中创建名为 film 的数据库表;
- 向 film 数据表中插入 4 条记录;

- 查询 film 数据表中的全部记录;

- 关闭数据库

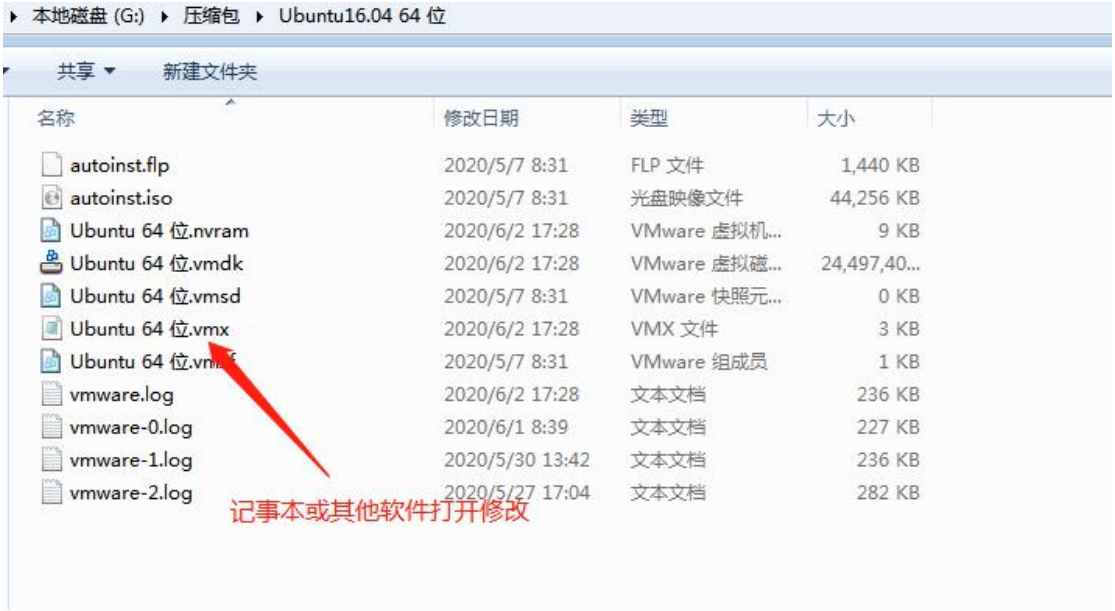
5. 环境搭建

5.1 安装虚拟机软件 VMware

安装 Ubuntu 的前提是什么？需要一个虚拟机 VMware 去启动它，虚拟机顾名思义就是虚拟出来的一个机器，然后在这个机器上安装任何你想要的系统，相当于在克隆出一个你的电脑，这样的话，主机上运行 Window 系统，当我们需要用到 Ubuntu 的话，打开安装有 Ubuntu 系统的虚拟机就可以了。

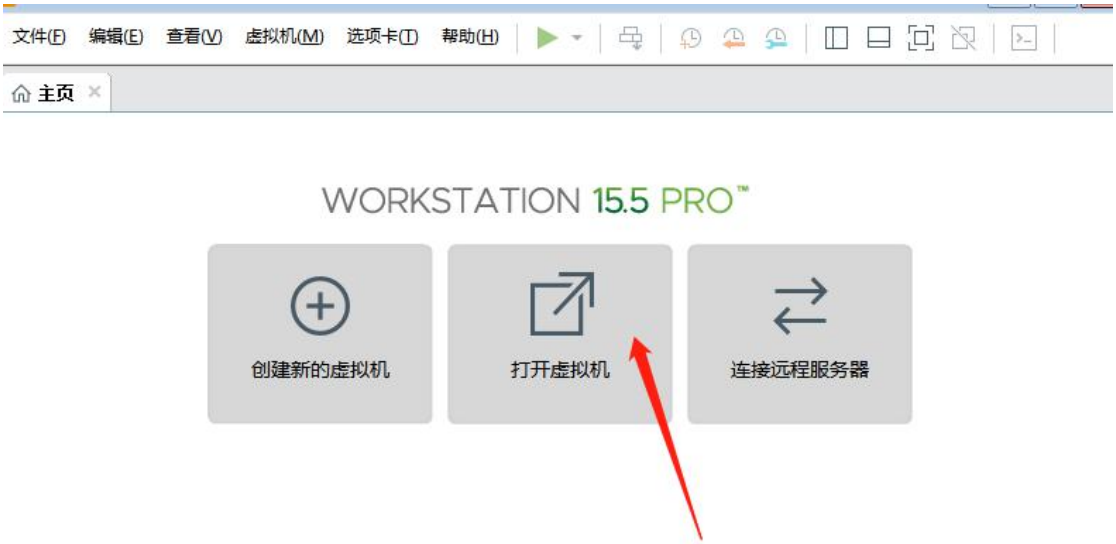
Vmware Workstation 软件可以在 Wmware 官网下载，下载地址：https：//www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html。下载最新版的 Vmware Workstation Pro 15。

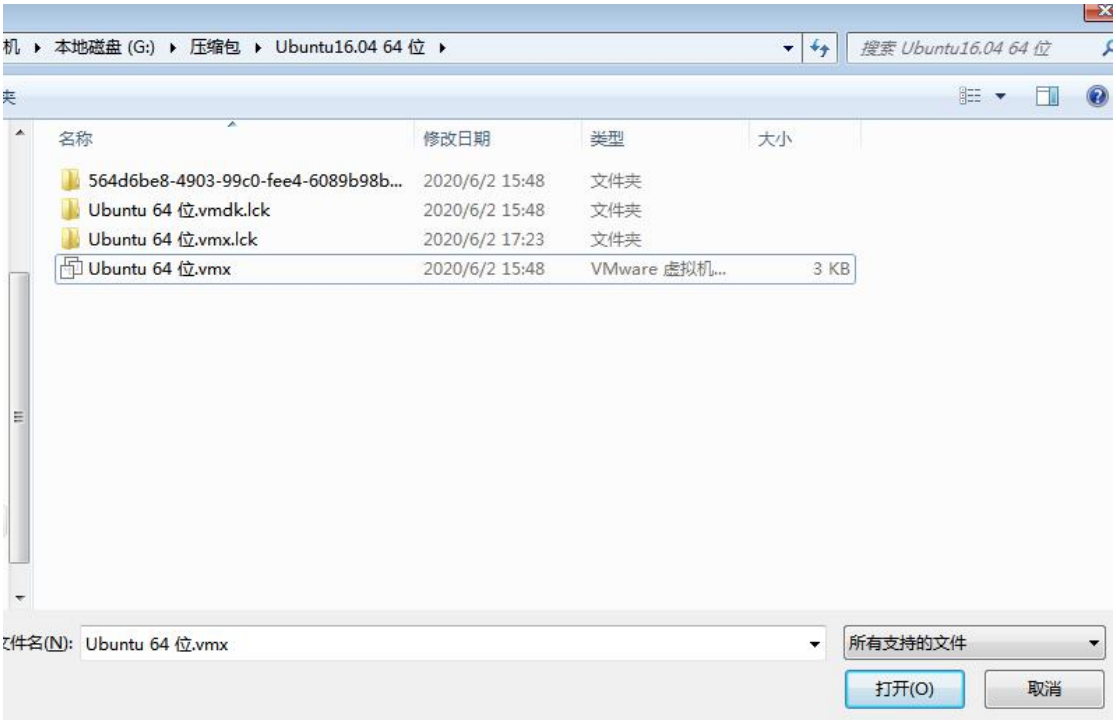
VMware 虚拟机的版本高低可能会导致 Ubuntu 无法正常开启。需要去修改光盘资料所提供 Ubuntu 的 Ubuntu 64 位.vmx 文件。



通过修改.vmx 文件里面的相应参数，如下图所示。

安装完毕后，打开 Vmware Workstation 软件，打开虚拟机，找到虚拟机解压的路径。





打开虚拟机，进入系统

5.2 交叉工具链

光盘资料中提供的 Ubuntu 已经把交叉工具链的环境搭建好了，用户可以直接进行使用。

Ubuntu 版本	16.04

交叉工具链版本	arm-poky-linux-gnueabi-gcc 5.3.0
交叉工具链的所在路径	/opt/fsl-imx-fb/4.1.15-2.0.1/sysroots/i686-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/

TW-AC6G-EVM 开发板使用 arm-poky-linux-gnueabi-gcc 5.3.0 版本交叉编译器，其安装包为 arm-poky-linux-gnueabi-gcc-5.3.0.tar.gz，可从“眺望产品光盘资料/2、软件开发参考资料/1、编译工具/arm-poky-linux-gnueabi-gcc-5.3.0.tar.gz”中直接获取。

可以简单的测试一下交叉编译器是否可以正常运行。方法如下：

在命令行下执行：

```
$source  /opt/fsl-imx-fb/4.1.15-2.0.1/environment-setup-cortexa9hf-neon-poky-linux-gnueabi
$arm-poky-linux-gnueabi-gcc -v
```

如果能显示编译器的版本信息，则表明编译器已经可以正常运行了，如下图所示：

```
twdz@ubuntu:~$ source /opt/fsl-imx-fb/4.1.15-2.0.1/environment-setup-cortexa9hf-neon-poky-linux-gnueabi
twdz@ubuntu:~$ arm-poky-linux-gnueabi-gcc -v
Using built-in specs.
COLLECT_GCC=arm-poky-linux-gnueabi-gcc
COLLECT_LTO_WRAPPER=/opt/fsl-imx-fb/4.1.15-2.0.1/sysroots/i686-pokysdk-linux/usr/libexec/arm-poky-linux-gnueabi/gcc/arm-poky-linux-gnueabi/5.3.0/lto-wrapper
Target: arm-poky-linux-gnueabi
Configured with: ../../../../../../../work-shared/gcc-5.3.0-r0/gcc-5.3.0/configure --build=i686-linux --host=i686-pokysdk-linux --target=arm-poky-linux-gnueabi --prefix=/opt/fsl-imx-fb/4.1.15-2.0.1/sysroots/i686-pokysdk-linux/usr --exec_prefix=/opt/fsl-imx-fb/4.1.15-2.0.1/sysroots/i686-pokysdk-linux/usr --bindir=/opt/fsl-imx-fb/4.1.15-2.0.1/sysroots/i686-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi --sbindir=/opt/fsl-imx-fb/4.1.15-2.0.1/sysroots/i686-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi --libexecdir=/opt/fsl-imx-fb/4.1.15-2.0.1/sysroots/i686-pokysdk-linux/usr/libexec/arm-poky-linux-gnueabi --datadir=/opt/fsl-imx-fb/4.1.15-2.0.1/sysroots/i686-pokysdk-linux/usr/share --sysconfdir=/opt/fsl-imx-fb/4.1.15-2.0.1/sysroots/i686-pokysdk-linux/etc --sharedstatedir=/opt/fsl-imx-fb/4.1.15-2.0.1/sysroots/i686-pokysdk-linux/com --localstatedir=/opt/fsl-imx-fb/4.1.15-2.0.1/sysroots/i686-pokysdk-linux/var --libdir=/opt/fsl-imx-fb/4.1.15-2.0.1/sysroots/i686-pokysdk-linux/usr/lib/arm-poky-linux-gnueabi --includedir=/opt/fsl-imx-fb/4.1.15-2.0.1/sysroots/i686-pokysdk-linux/usr/include --oldincludedir=/opt/fsl-imx-fb/4.1.15-2.0.1/sysroots/i686-pokysdk-linux/usr/include --infodir=/opt/fsl-imx-fb/4.1.15-2.0.1/sysroots/i686-pokysdk-linux/usr/share/info --mandir=/opt/fsl-imx-fb/4.1.15-2.0.1/sysroots/i686-pokysdk-linux/usr/share/man --disable-silent-rules --disable-dependency-tracking --with-libtool-sysroot=/home/sinc/fsl-release-bsp/imx6q-build-fb/tmp/sysroots/i686-nativesdk-pokysdk-linux --with-gnu-ld --enable-shared --enable-languages=c,c++ --enable-threads=posix --enable-multilib --enable-c99 --enable-long-long --enable-symvers=gnu --enable-libstdcxx-pch --program-prefix=arm-poky-linux-gnueabi- --without-local-prefix --enable-lto --enable-libssp --enable-libitm --disable-bootstrap --disable-libmudflap --with-system-zlib --with-linker-hash-style=gnu --enable-linker-build-id --with-ppl=no --with-cloog=no --enable-checking=release --enable-cheaders=c_global --without-isl --with-gxx-include-dir=/not/exist/usr/include/c++/5.3.0 --with-build-time-tools=/home/sinc/fsl-release-bsp/imx6q-build-fb/tmp/sysroots/i686-linux/usr/arm-poky-linux-gnueabi/bin --with-sysroot=/not/exist --with-build-sysroot=/home/sinc/fsl-release-bsp/imx6q-build-fb/tmp/sysroots/imx6qsabresd --enable-poison-system-directories --with-mpfr=/home/sinc/fsl-release-bsp/imx6q-build-fb/tmp/sysroots/i686-nativesdk-pokysdk-linux --with-mpc=/home/sinc/fsl-release-bsp/imx6q-build-fb/tmp/sysroots/i686-nativesdk-pokysdk-linux --enable-nls --with-arch=armv7-a
Thread model: posix
gcc version 5.3.0 (GCC)
twdz@ubuntu:~$
```

5.3 编译 Helloworld 源程序

在 Linux 宿主机任意目录下创建一个 HelloWorld 文件夹，用于存放 helloworld.c 源文件。此处假定在/home/tw/demo 下创建 HelloWorld 文件夹。执行的命令如下：

```
$mkdir /home/tw/demo
$mkdir /home/tw/demo/HelloWorld
$cd /home/tw/demo/HelloWorld
```

在该目录下新建一个 helloworld.c 的源文件，其内容如下：

```
#include<stdio.h>
int main()
{
    Printf(“helloworld!\n”);
    Return 0;
}
```

保存退出后，进入源码所在目录，在命令行下执行如下命令编译程序：

```
$CC  –o helloworld helloworld.c
```

将交叉编译出来的执行文件 helloworld 下载到开发板上运行。

```
~#chmod a+x helloworld
~# ./helloworld
helloworld
```

从执行结果可以看出，在终端上已经打印出了“Hello World”字符串。

5.4 编译概况

内核源码的顶层提供了各种编译执行脚本，用户可以根据不同的需求去执行相应的脚本，得到目标固件。

shell 脚本	说明	固件路径
built-all-tw.sh	清除以前的编译结果，全部重新编译	编译出来的目标文件如下表 dtb，modules，zImage 路径相同
built-dtb-tw.sh	编译设备树	源码顶层的 arch/arm/boot/dts/hd-imx6ull-core-emmc.dtb 源码顶层的 arch/arm/boot/dts/hd-imx6ull-core-256m.dtb
built-modules-tw.sh	编译内核模块	需根据在内核配置编译的模块的路径
built-zImage-tw.sh	编译内核映像	源码顶层的 arch/arm/boot/zImage
built-menuconfig.sh	使用 menuconfig 配置内核	

将目标固件如 zImage，.dtb，按照第二章节中 2.2 mfgtools 工具烧写系统，2.3 单独更新设备树 DTB，2.4 单独更新内核 zImage 去进行操作和替换。

5.4.1 内核源码简介

Linux 内核源码很复杂，包含多级目录，形成一个庞大的树状结构，通常称为 Linux 源码目录树。TW-AC6G-EVM 开发板使用 Linux 4.1.15 内核版本，本节以该版本为例介绍 Linux 源码的目录结构：

目录	说明
arch	包含各体系结构特定的代码，如 arm、x86、ia64、mips 等，在每个体系结构目录下通常都有： －boot 内核需要的特定平台代码 －kernel 体系结构特有的代码 －lib 通用函数在特定体系结构的实现 －math-emu 模拟 FPU 的代码，在 ARM 中，使用 mach-xxx 代替 －mm 特定体系结构的内存管理实现 －include 特定体系的头文件
block	存放块设备相关代码
crypto	存放加密、压缩、CRC 校验等算法相关代码
Documentation	存放相关说明文档，很多实用文档，包括驱动编写等
drivers	存放 Linux 内核设备驱动程序源码。驱动源码在 Linux 内核源码中站了很大比例，常见外设几乎都有可参考源码，对驱动开发而言，该目录非常重要。该目录包含众多驱动，目录按照设备类别进行分类，如 char、block、input、i2c、spi、pci、usb 等

firmware	存放处理器相关的一些特殊固件
fs	存放所有文件系统代码，如 fat、ext2、ext3、ext4、ubifs、nfs、sysfs 等
include	存放内核所需、与平台无关的头文件，与平台相关的头文件已经被移动到 arch 平台的 include 目录，如 ARM 的头文件目录<arch/arm/include/asm/>
init	包含内核初始化代码
ipc	存放进程间通信代码
kernel	包含 Linux 内核管理代码
lib	库文件代码实现
mm	存放内存管理代码
net	存放网络相关代码
samples	存放提供的一些内核编程范例
srcipts	存放一些脚本文件，如 menuconfig 脚本
security	存放系统安全性相关代码
sound	存放声音、声卡相关驱动
tools	编译过程中一些主机必要工具
usr	cpio 相关实现
virt	内核虚拟机 KVM

5.4.2 内核配置

1. 内核 Makefile 文件

源码目录树顶层 Makefile 是整个内核源码管理的入口，对整个内核的源码编译起着决定性作用。编译内核时，顶层 Makefile 会按规则递归遍历内核源码的所有子目录下的 Makefile 文件，完成各子目录下内核模块的编译。

在内核源码的子目录中，几乎每个子目录都有相应的 Makefile 文件，管理着对应目录下的代码，对该目录的文件或者子目录的编译进行控制。Makefile 中有两种表示方式来控制某个文件是否编译，一种是默认选择编译，用 obj-y 表示，如：

```
obj-y += usb-host.o #默认编译 usb-host.c 文件
obj-y += gpio/ #默认编译 gpio 目录
```

另一种表示则与内核配置选项相关联，编译与否以及编译方式取决于内核配置，例如：

```
obj-$(CONFIG_WDT) += wdt.o # wdt.c 编译控制
obj-$(CONFIG_PCI) += pci/ # pci 目录编译控制
```

是否编译 wdt.c 文件，或者以何种方式编译，取决于内核配置后的变量 CONFIG_WDT 值：如果在配置中设置为[*]，则静态编译到内核，如果配置为[M]，则编译为 wdt.ko 模块，否则不编译。

2. Kconfig 文件

内核源码树每个目录下都还包含一个 Kconfig 文件，用于描述所在目录源代码相关的内核配置菜单，各个目录的 Kconfig 文件构成了一个分布式的内核配置数据库。通过 make menuconfig（make xconfig 或者 make gconfig）命令配置内核的时候，从 Kconfig 文件读取菜单，配置完毕保存到文件名为.config 的内核配置文件中，供 Makefile 文件在编译内核时使用。

3. 内核配置工具

用户可以使用如下命令对内核进行配置：

1） make config：基于文本模式的交互式配置；

2） make menuconfig：基于文本模式的菜单型配置；

3） make oldconfig：使用已有的配置文件（.config），但是会询问新增的配置选项；

4） make xconfig：图形化的配置（需安装图形化系统）；

其中 make menuconfig 是最为常用的内核配置方式。下面将主要介绍 make menuconfig 的使用。

在运行 make menuconfig 前，主机必须先安装 ncurses 相关的库。在 Ubuntu 下执行如下命令完成 ncurses 的安装：

```
$sudo apt-get install libncurses5-dev
```

在 Linux 内核源码顶层目录下执行脚本文件：

./built-menuconfig.sh，可进入 Linux 内核配置主界面，如下图所示：

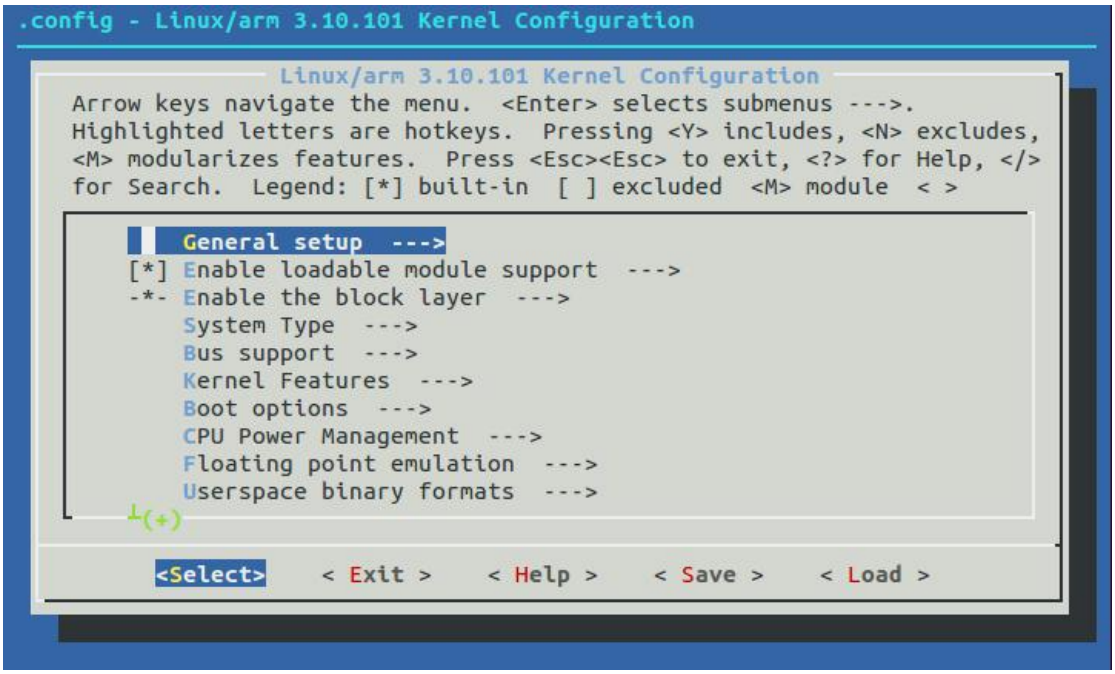


图 4.1 Linux 内核配置主界面

4. 内核配置主界面由三部分组成：

1） 最上面部分为基本操作的简要介绍；

2） 中间部分为内核配置的各个菜单项；

3） 最下面部分为功能菜单

5. 基于 Ncurses 的 Linux 内核配置界面不支持鼠标操作，必须用键盘操作。基本操作方法如下：

1） 使用上、下箭头可以选择内核配置菜单项；使用左、右箭头可以选择下排的功能菜单项；

2） 当功能菜单中的“Select”项被选中时，在某个具有子菜单的内核菜单项上按回车键可进入该菜单的子菜单；

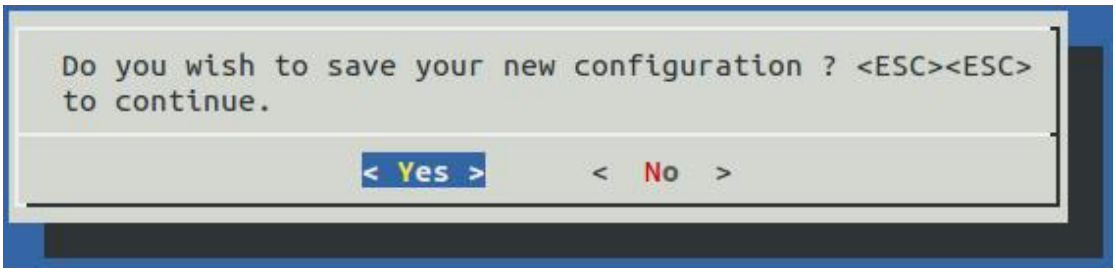
3） 对于某个菜单项，可以直接按“？”键查看该菜单项的帮助，或者用左右箭头键将功能菜单移到“Help”上，并按回车键查看帮助；

4） 根据菜单项的类型，其配置方式略有不同。可以分成三种情况，具体说明如下：

- 对于以[ ]开头的配置项，[*]表示选中，[ ]表示未选中。可以用空格键进行切换或者使用键盘快捷键（Y-选中，N-未选中）进行切换；
- 对于以<>开头的配置项，<*>表示静态编译，<M>表示编译为模块，<>表示未选中。可以使用空格键进行切换或者使用键盘快捷键（Y-静态编译，M-编译为模块，N-未选中）进行切换；
- 对于()开头的配置项，表明该配置是数值或者字符串，可以按回车键直接编辑。

5） 按斜线（/）可启用搜索功能，填入关键字后可搜索全部菜单内容；

6） 连续按两次 ESC 键或者用左右箭头键将功能菜单移到“Exit”上，并按回车键，将退回到上一级菜单。如果当前已经位于顶层菜单界面（即内核配置主界面），此时如果用户对内核配置进行了修改，则将弹出确认修改的提示画面，如下图所示：



选择“Yes”将保存对内核配置的修改并退出，选择“No”将不保存修改直接退出，连续按两次 ESC 键将重新返回内核配置画面，允许用户继续对内核配置进行修改。

内核配置完成后，其配置信息保存在内核源码顶层目录的.config 文件中。用户可以直接在命令行下将.config 文件备份为其他的文件名，以备日后使用，例如：

```
$cp .config .config_bak
```

为了使用以前备份的内核配置文件，可以直接将备份的内核配置文件覆盖内核源码顶层目录的.config 文件，例如：

```
$cp .cofnig_bak .config
```

或者执行 make menuconfig 进入内核配置主界面，并用左右箭头选中功能菜单上的“Load”项，并按回车键，弹出配置文件加载画面，如下图所示：

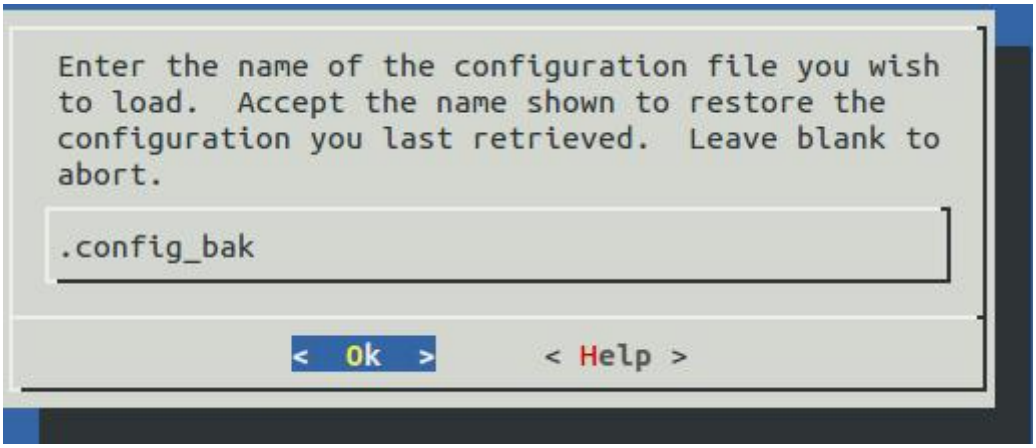


图 6.3 内核配置加载画面

在该画面中输入需要加载的备份文件的名字，选择“Ok”按钮后按回车键，此时会加载用户选择的备份文件，用户可对其进行修改并保存设置产生新的.config 文件。

5.4.3 内核编译

在“2.软件开发参考资料\3.软件源码\1.内核”目录中，包含了一个已经配置好的 Linux 内核源码文件 kernel-a7-tw-evm.tar.gz。编译 Linux 操作系统的步骤如下：

为了方便用户开发调试，提供了相对应的编译脚本文件，尽量使用脚本文件进行编译：

./built-zImage.sh	编译内核
./built-dtb.sh	编译设备树
./built-menuconfig.sh	对内核进行配置
./built-modules.sh	编译模块 ko
./built-all.sh	进行全编译

1. 将 Linux 内核源码文件 kernel-a7-tw-evm.tar.gz 拷贝到 Ubuntu 虚拟机中，此处假定将这该文件拷贝到/home/tw/imx6dl 中，并对其进行解压：

```
$cd /home/tw/imx6d/
$tar -zxvf kernel-a7-tw-evm.tar.gz
```

2. 设置编译环境。每次编译前都需要把编译环境加载到当前 shell 来，执行如下命令：

```
$source /opt/fsl-imx-fb/4.1.15-2.0.1/environment-setup-cortexa9hf-neon-poky-linux-gnueabi
```

3. 清除前一次的编译结果。执行如下命令：

```
$cd /home/tw/imx6d/kernel-a7-tw-evm
$ make ARCH=arm CROSS_COMPILE=arm-poky-linux-gnueabi- distclean
```

4. 使用开发板默认的配置文件生成编译内核时所需要的.config 文件。执行如下命令：

```
$ make ARCH=arm CROSS_COMPILE=arm-poky-linux-gnueabi- sc_imx6_defconfig
```

注：如果已经生成了.config 文件，或已通过 make menuconfig 对配置进行了修改，则可跳过该步骤。

5. 使用 make menuconfig 对内核进行配置。执行如下命令：

```
$ make ARCH=arm CROSS_COMPILE=arm-poky-linux-gnueabi- menuconfig
```

或者执行脚本 ./built-menuconfig.sh

注：如果不需要对内核配置进行了修改，则可跳过该步骤。

6. 编译内核。执行如下命令或执行脚本文件：./built-zImage.sh

```
$ make ARCH=arm CROSS_COMPILE=arm-poky-linux-gnueabi- all
```

编译完成后，会在内核源码顶层目录（即 kernel-a7-tw-evm 目录）的 arch/arm/boot 子目录下生成 zImage 文件。

5.5 编译 QT 的程序

Ubuntu 虚拟机提供了 5.12.8 版本的 QT creator，交叉工具 qmake 安装在路径 /home/twdz/ctools/fsl-imx-fb/4.1.15-2.0.1/sysroots/cortexa9hf-neon-poky-linux-gnueabi/usr/lib/qt5/bin/。

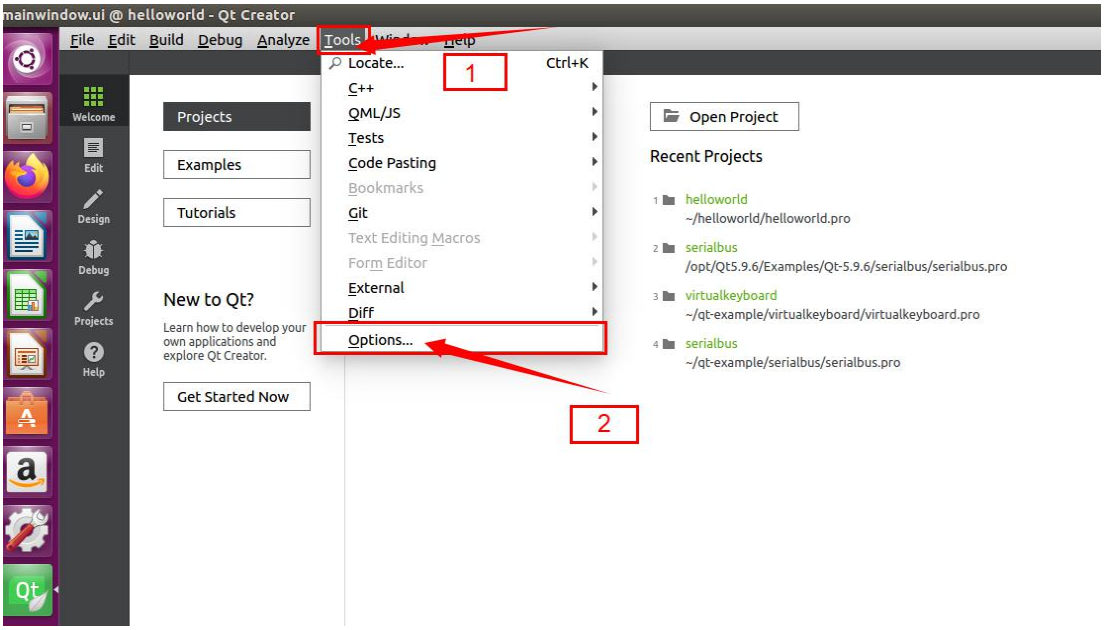
如果没有 qt5/bin 目录，请解压“2.软件开发参考资料\3.软件源码\4.Qt-5.12.8”目录下的 qt5.12.8.tar.gz 文件到/home/twdz/ctools/fsl-imx-fb/4.1.15-2.0.1/sysroots/cortexa9hf-neon-poky-linux-gnu

eabi/usr/lib 目录下(解压路径可自行修改，后续需指定解压出来的 qt5/bin/qmake)。qt5.12.8.tar.gz 是交叉编译 ARM 平台 Qt5.12.8 源码所得到的安装后的内容。

打开所提供的光盘资料中提供的 Ubuntu，其中已安装了 Qt Creator。一般来说已经配置好了 ARM 平台的 Qt Creator Kits --- TW-imx6DL。如果没有请看接下来的 Qt Creator Kits 配置过程。

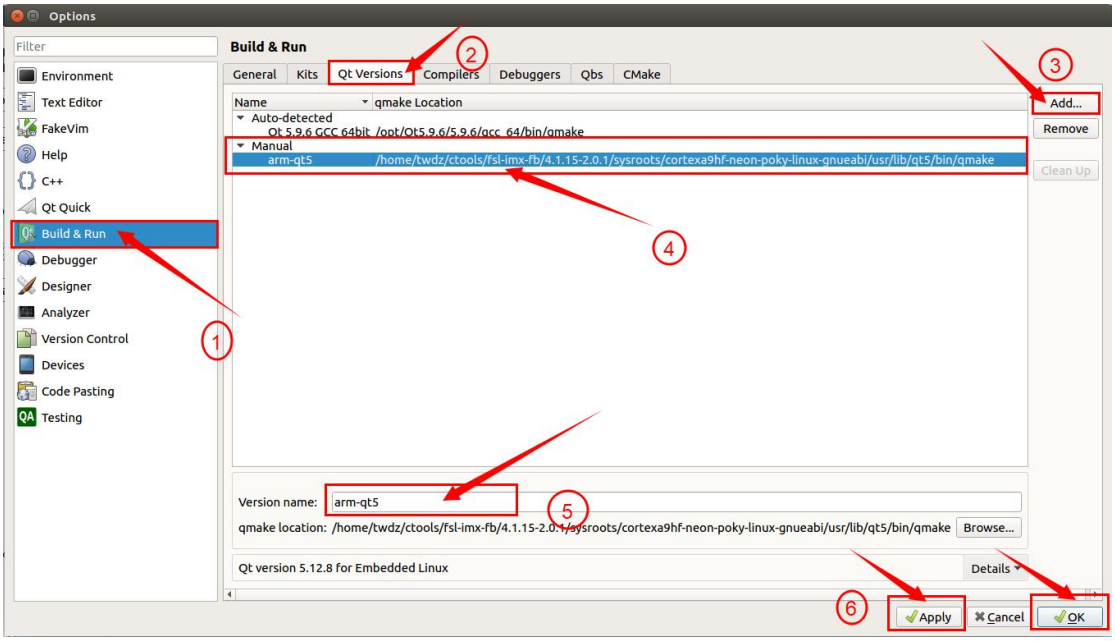
Qt Creator Kits 配置过程：

(1)点击“Tools”,选择 Options。



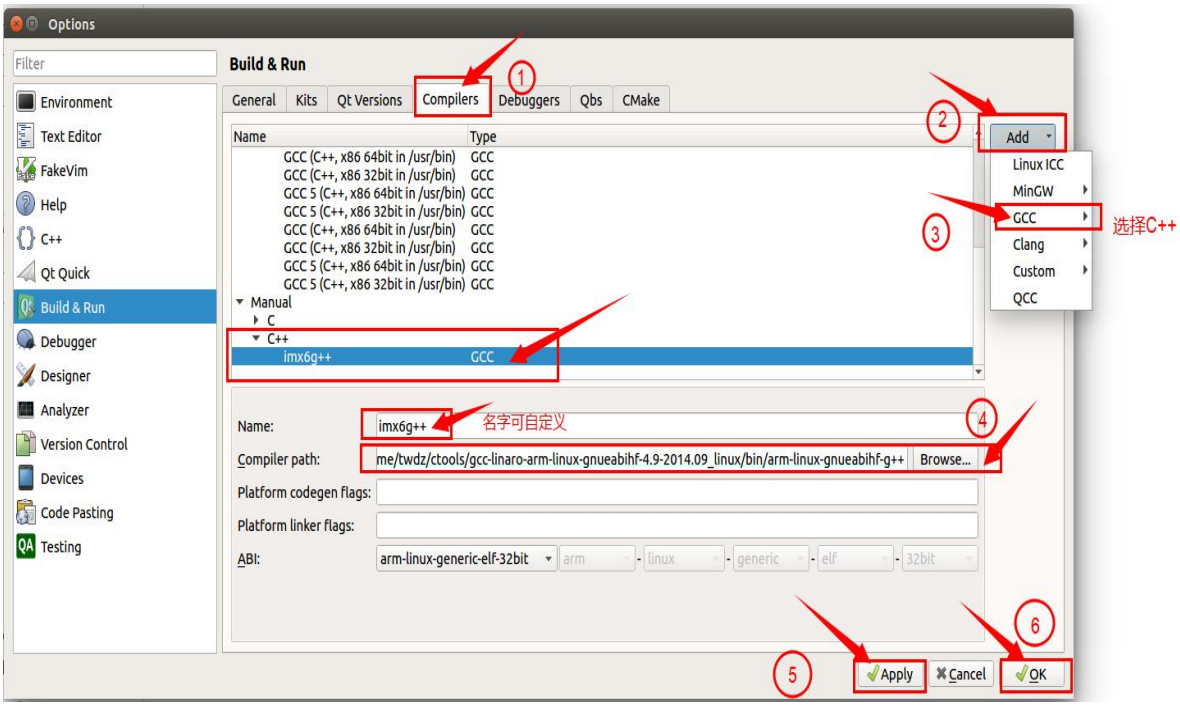
(2) 配置 qmake：如下图，添加.....(此处省略，请根据自身的目录修改)/qt5/bin/qmake。此处的 qt5 目录是解压 qt5.12.8.tar.gz 文件得到的。图示中完整路径：

/home/twdz/ctools/fsl-imx-fb/4.1.15-2.0.1/sysroots/cortexa9hf-neon-poky-linux-gnueabi/usr/lib/qt5/bin/qmake

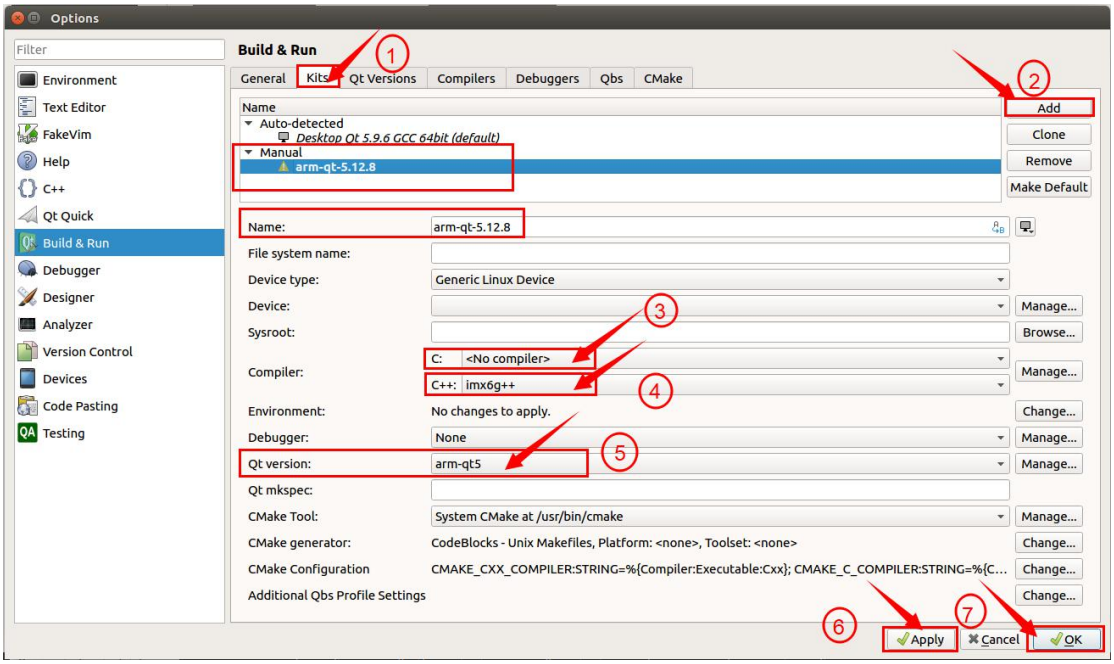


(3) 配置 C++编译器：如下图，点击 Compilers，点击 Add，选择 GCC 下的 C++,添加

交叉编译器路径/home/twdz/ctools/gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux/bin/arm-linux-gnueabihf-g++ (可根据自身交叉编译工具链目录路径自行修改)，然后先点击 Apply，后点击 OK。

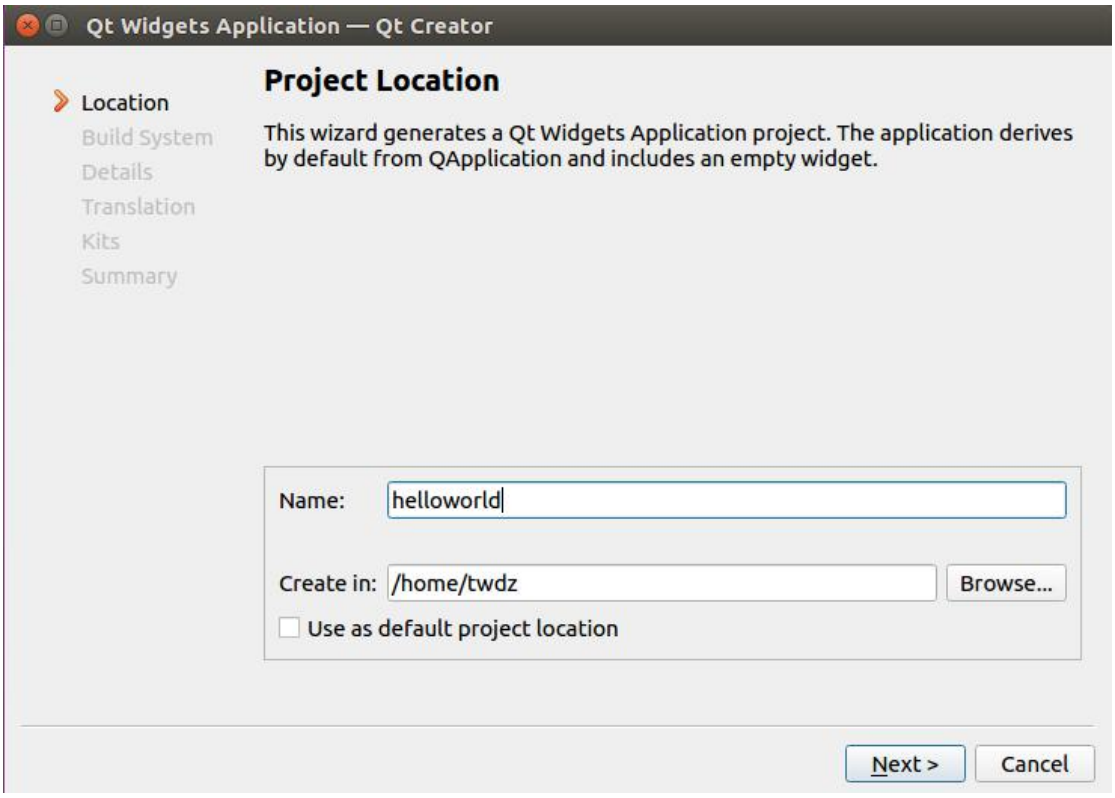


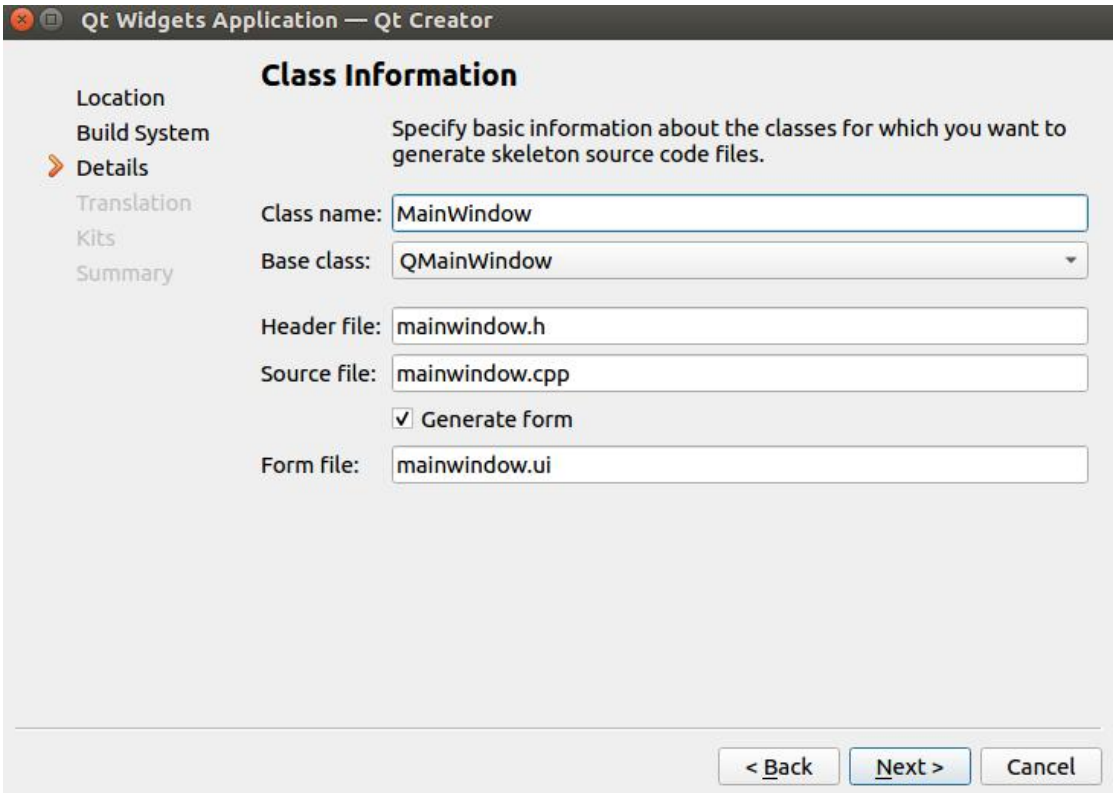
(4)配置 Kits：如图，要选择之前步骤中所创建的 C++编译器和 Qt verison，最后要先点击 Apply 再点击 OK。



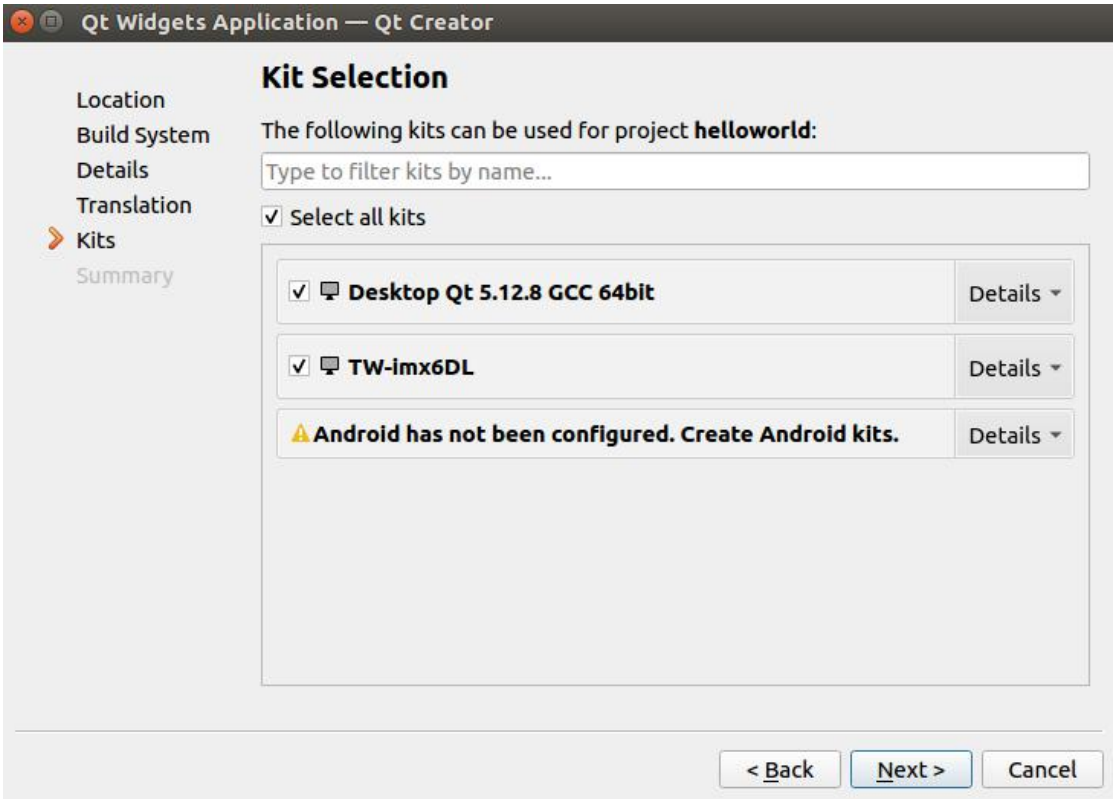
最终，arm-qt-5.12.8 就是我们所构建的 ARM 版本的交叉工具。

1. 打开 Qt Creator 应用软件，创建新的 Qt 程序，工程名为 helloworld，点击 next。

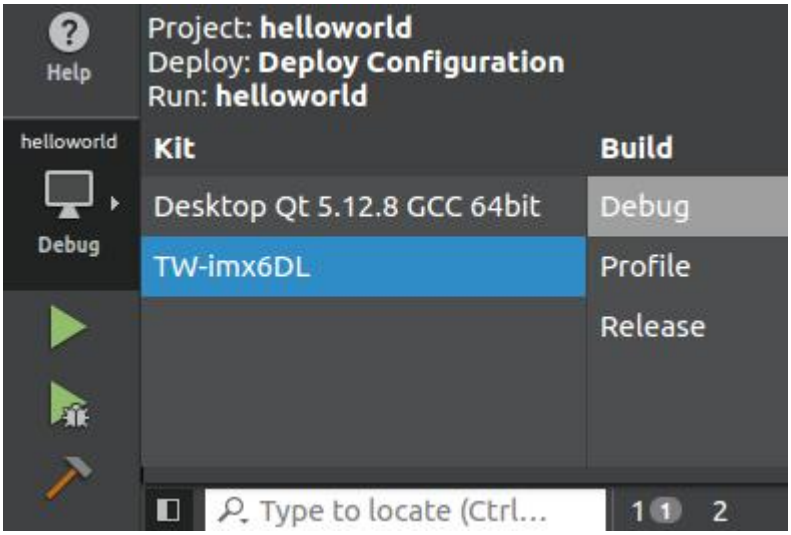




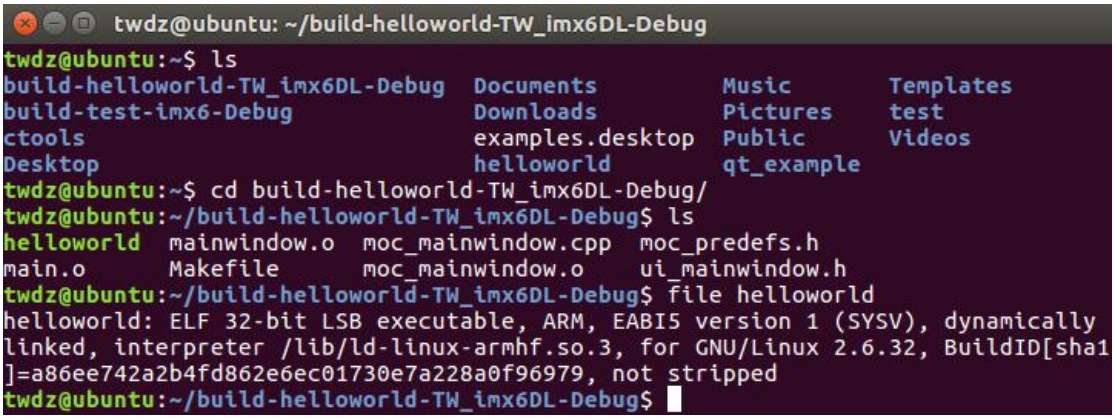
2. 选择编译工具，Desktop Qt 5.12.8 GCC 64bit 是 linux 下的编译器，TW-imx6DL 是 ARM 版本的交叉工具。



3. 选择 ARM 版本的编译方式。



4. 如果编译成功，可执行文件 helloworld 在 /home/twdz/build-helloworld-TW_imx6DL-Debug 路径下，通过命令：file helloworld 可以查看文件信息。


twdz@ubuntu: ~/build-helloworld-TW_imx6DL-Debug
twdz@ubuntu:~$ ls
build-helloworld-TW_imx6DL-Debug Documents Music Templates
build-test-imx6-Debug Downloads Pictures test
ctools examples.desktop Public Videos
Desktop helloworld qt_example
twdz@ubuntu:~$ cd build-helloworld-TW_imx6DL-Debug/
twdz@ubuntu:~/build-helloworld-TW_imx6DL-Debug$ ls
helloworld mainwindow.o moc_mainwindow.cpp moc_predefs.h
main.o Makefile moc_mainwindow.o ui_mainwindow.h
twdz@ubuntu:~/build-helloworld-TW_imx6DL-Debug$ file helloworld
helloworld: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-armhf.so.3, for GNU/Linux 2.6.32, BuildID[sha1]=a86ee742a2b4fd862e6ec01730e7a228a0f96979, not stripped
twdz@ubuntu:~/build-helloworld-TW_imx6DL-Debug$

6. 免责声明

本文档提供有关广州眺望电子科技有限公司产品的信息。本文档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止发言或其它方式授予任何知识产权许可。除广州眺望电子科技在其产品的销售条款和条件中声明的责任之外，概不承担任何其它责任。并且，产品的销售和／或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。广州眺望电子科技产品并非设计用于医疗、救生或维生等用途。广州眺望电子科技可能随时对产品规格及产品描述做出修改，恕不另行通知。

文档所属产品可能包含某些设计缺陷或错误，一经发现将收入勘误表，并因此可能导致产品与已出版的规格有所差异。如客户索取，可提供最新的勘误表。在订购产品之前，请您与我司销售处或分销商联系，以获取最新的规格说明。本文档中提及的含有订购号的文档以及其它文献可通过访问 http：//www.iot-tw.com/ 获得。