

# EM-TF-EVK-AM5728

# Linux 软件开发指导

V1.0





# 版权声明

- ◆ EM-TF-EVK-AM5728评估板及其相关知识产权由深圳市英蓓特科技有限公司所有。
- 本文档由深圳市英蓓特科技有限公司版权所有,并保留一切权利,在未经英蓓特公司书面许可的情况下,不得以任何形式来修改、分发或复制本文档的任何部分。

# 免责声明

产品附带提供的程序源代码、软件、资料文档等,深圳市英蓓特有限公司不提供任何类型的担保;不 论是明确的,还是隐含的,包括但不限于合适特定用途的保证,全部的风险,由使用者来承担。



# 版本记录

版本	描述	作者	日期
V1.0	初稿	David/Yuding	20180827



長目

版本记录	1. 		
目录			4
第1章	环境排	答建	6
1.1	搭建开	干发环境	6
1.2	配置编	高译环境	6
1.3	其他]	L 具和服务	7
第2章	编译		8
2.1	U-Boo	t	8
	2.1.1	获取 U-Boot 源码	8
	2.1.2	编译 U-boot	8
2.2	Kernel	l	8
	2.2.1	获取内核源码	8
	2.2.2	编译镜像	8
2.3	外部驯	区动	9
	2.3.1	配置环境变量	9
	2.3.2	编译外部驱动	10
	2.3.3	安装外部驱动	10
第3章	制作镜	意像	13
3.1	制作每	竟像文件	13
	3.1.1	分区格式化	13
	3.1.2	复制 firmware	15
	3.1.3	复制根文件系统	16
	3.1.4	安装内核模块	
	3.1.5	安装外部驱动模块	
3.2	烧录利	D读取镜像	16
	3.2.1	烧录	16
	3.2.2	读取	17
第4章	TI SDK	开发	19
4.1	SDK 的	的安装和配置	19
4.2	顶层N	Makefile 使用	23
	4.2.1	编译整个 SDK	



#### EM-TF-EVK-AM5728-Linux\_DG-V1.0

		4.2.2	单独编译安装某个目标	24
第5	章	应用开	发	26
!	5.1	普通C	程序的交叉编译和执行	26
		5.1.1	普通 C 程序的交叉编译和执行	26
		5.1.2	交叉编译	26
		5.1.3	直接编译	26
		5.1.4	传递到开发板运行	26
!	5.2	QT 开发	È	27
		5.2.1	安装 Qt Creator	27
		5.2.2	配置 Qt Creator	34
		5.2.3	创建 demo	39
		5.2.4	在 AM5728 板上运行	45
	5.3	视屏采	集 demo	47
	5.4	双屏显	显示 demo	48
第6	章	基于 Y	octo 的根文件系统构建	50
	6.1	安装需	要的工具软件	50
	6.2	配置 b	ash	50
	6.3	安装编	译器	50
(	6.4	获取o	e-layertool-setup.sh	50
	6.5	bitbake	•构建	51
第7	章	附录		52
	7.1	硬件		52
第8	章	技术支	持和保修服务	53
		8.1.1	技术支持	53
		8.1.2	保修服务	53
第9	章	联系方	式	54



# 第1章 环境搭建

## 1.1 搭建开发环境

搭建开发环境需要:

- ◆ 硬件:至少 20GB 磁盘空间,2GB 运行内存
- ◆ 软件:Ubuntu 64 bit 操作系统,14.04 LTS,16.04LTS 或更高的 LTS 版本 (Ubuntu Desktop 或 Ubuntu

#### Server)

注:如需开发 QT,必须使用 Ubuntu Desktop 这类带 GUI 界面的 Ubuntu 版本。

也可以用虚拟机来运行 Ubuntu 64 bit 操作系统。

启动 Ubuntu 系统后,运行下列命令安装开发需要的软件。

sudo apt-get update
sudo apt-get install openssh-server
sudo apt-get install git
sudo apt-get install kpartx
sudo apt-get install lzop
sudo apt-get install lsb-core
sudo apt-get install libncurses5

### 1.2 配置编译环境

将发布文件夹的 gcc-linaro-6.2.1-2016.11-x86\_64\_arm-linux-gnueabihf.tar.xz 拷贝到 Linux 环境下的\$HOME 目录 下,解压:

\$tar -Jxvf gcc-linaro-6.2.1-2016.11-x86\_64\_arm-linux-gnueabihf.tar.xz

设置环境变量:

#### \$export

CROSS\_COMPILE=\$HOME/gcc-linaro-6.2.1-2016.11-x86\_64\_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-

#### \$export ARCH=arm

注意:每次编译 u-boot 和 kernel 前都要配置环境变量。为了方便起见,可以编辑一个脚本,然后 source 这 个脚本:

\$cd\$HOME \$ cat set\_am57\_env.sh #/bin/bash export CROSS\_COMPILE=\$HOME/gcc-linaro-6.2.1-2016.11-x86\_64\_arm-linux-gnueabihf/bin/arm-linux-gnueabihfexport ARCH=arm



#### \$ source ./set\_am57\_env.sh

注意:如未明确说明:

- ◆ 本文以\$开头的命令行操作均指在 ubuntu PC 上;
- ◆ 文中出现的\$HOME 没有强制要求,应以用户实际目录做相应的修改;
- ◆ 本文以#开头的命令行表示在开发板上执行。

# 1.3 其他工具和服务

开发过程中还可能会用到其他工具和服务,比如:

- ◆ ssh 登录 ubuntu/串口登录的 putty 软件
- ◆ 用于在 linux 和 windows 之间互传文件的 samba 服务
- NFS
- TFTP
- 🔷 🛛 Samba

关于这些常规的开发工具的配置与使用,本文从略。



# 第2章 编译

#### 2.1 U-Boot

#### 2.1.1 获取 U-Boot 源码

复制 u-boot\*.tar.gz 到\$HOME 并解压:

\$ cd \$HOME

\$ tar -xzvf u-boot\*.tar.gz

2.1.2 编译 U-boot

\$ cd \$HOME/u-boot

\$ make distclean

\$make som\_am572x\_defconfig

#### \$make

编译完成后在\$HOME/u-boot 目录下生成 u-boot.img 和 MLO。

#### 2.2 Kernel

#### 2.2.1 获取内核源码

复制 linux 内核源代码包 linux\*.tar.gz 到\$HOME 并解压:

\$ tar -zxvf linux\*.tar.gz

#### 2.2.2 编译镜像

\$ cd \$HOME/linux

\$ make distclean

\$ make embest\_ti\_am57xx\_defconfig

#### \$ make

编译完成生成 zImage 和 dtb:

- \$HOME/linux/arch/arm/boot/zImage
- \$HOME/linux/arch/arm/boot/dts/embest-SOM\_AM572x\_TM-mode0.dtb
- \$HOME/linux/arch/arm/boot/dts/embest-SOM\_AM572x\_TM-mode0-LCD.dtb

其中 embest-SOM\_AM572x\_TM-mode0.dtb 将 HDMI 作为主显示屏, embest-SOM\_AM572x\_TM-mode0-LCD.dtb 将 LCD 作为主显示屏。



### 2.3 外部驱动

由于 TI 的部分外设模块的驱动是单独发布的,因此这部分驱动程序需要额外编译。这些外设模块包括 2/3D 图像加速模块,硬件加/解密模块等。复制 extra.tar.gz 到\$HOME 并解压

#### \$ cd \$HOME/

\$ tar -xzf extra.tar.gz

#### 2.3.1 配置环境变量

编辑 Rules.make 文件,修改以下几个变量为相应的值:

◆ DESTDIR 根文件系统所在路径,如已经烧写镜像的 SD 卡在 ubuntu 下挂载路径,也可以是任意其他空白 目录。

◆ CROSS\_COMPILE 为交叉编译工具链所在路径

◆ LINUXKERNEL\_INSTALL\_DIR 为 linux 内核源代码路径,在编译外部驱动之前,先确保 linux 内核源代码被 正确地配置和编译,参见 2.2Kernel 的编译。

#### \$ cd extra

\$ ls

extra-drivers MakefileRules.make

\$cat Rules.make

#platform

#platform

PLATFORM=am57xx-evm

#root of the target file system for installing applications DESTDIR=\$(HOME)/extra/fakeroot

#Cross compiler prefix

#### export

CROSS\_COMPILE=\$(HOME)/gcc-linaro-6.2.1-2016.11-x86\_64\_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-

#The directory that points to the SDK kernel source tree

LINUXKERNEL\_INSTALL\_DIR=\$(HOME)/linux

CFLAGS= -march=armv7-a -marm -mfpu=neon -mfloat-abi=hard

#Strip modules when installing to conserve disk space INSTALL\_MOD\_STRIP=1

export TOOLCHAIN\_PREFIX=\$(CROSS\_COMPILE)



#### 2.3.2 编译外部驱动

编译之前,先创建 Rules.make 中 DESTDIR 变量指定的空白目录。此处我们创建 fakeroot 目录

\$ cd	extra

\$ mkdir fakeroot

\$ ls

extra-drivers fakeroot Makefile Rules.make

\$ make clean

#### \$ make

#### 2.3.3 安装外部驱动

### \$ make install

编译生成的 ko 文件以及相关的文件会被安装在 fakeroot 下,如果只需要更新 ko 文件,则只需要复制 lib/modules/4.9.28/extra 下的 ko 文件到目标板文件系统的相应目录下即可。

\$ tre	ee fakeroot/
fake	eroot/
	—— lib
	firmware
	jailhouse.bin
	L modules
	L 4.9.28
	extra
	bc_example.ko
	cmemk.ko
	cryptodev.ko
	debugss_kmodule.ko
	driver
	jailhouse.ko
	galcore.ko
	gdbserverproxy.ko
	pvrsrvkm.ko
	uio_module_drv.ko
	modules.alias
	modules.alias.bin
	modules.builtin.bin
	modules.dep
	modules.dep.bin
	modules.devname
	modules.softdep



modules.symbols
Lmodules.symbols.bin
usr
libexec
L jailhouse
jailhouse-cell-linux
jailhouse-cell-stats
jailhouse-config-create
jailhouse-hardware-check
Llinux-loader.bin
sbin
L jailhouse
share
bash-completion
L completions
L jailhouse
jailhouse
jailhouse-config-collect.tmpl
root-cell-config.c.tmpl
4 directories, 28 files

如果为了开发方便,可关闭内核中的 git 版本控制选项。以后编译出来的内核版本都是 linux-4.9.28,因此不 用反复的编译 extra 相关的驱动模块。

在 kernel 根目录输入 make menuconfig,打开配置界面,在 menuconfig 界面键入/,搜索 CONFIG\_LOCALVERSION\_AUTO 选项,根据提示找到相关选项,将其设置为[=n]:

Search Configuration Parameter	
Enter (sub)string or regexp to search for (with or withou	t "CONFIG_")
CONFIG_LOCALVERSION_AUTO	
< Ok > < Help >	



```
.config - Linux/arm 4.9.28 Kernel Configuration
> Search (CONFIG_LOCALVERSION_AUTO) ______ Search Results ______
Symbol: LOCALVERSION_AUTO [=n]
Type : boolean
Prompt: Automatically append version information to the version string
Location:
(1) -> General setup
Defined at init/Kconfig:91
Depends on: !COMPILE_TEST [=n]
```

在开发阶段,也可以使能 CONFIG\_MODULE\_FORCE\_LOAD。

	Enable loadable module support
[×]	Forced module loading
[*]	Module unloading
[*]	Forced module unloading
[*]	Module versioning support
[*]	<b>S</b> ource checksum for all modules
[]	Module signature verification
[]	<b>C</b> ompress modules on installation
[]	Trim unused exported kernel symbols



# 第3章 制作镜像

### 3.1 制作镜像文件

发布文件夹中有做好的镜像文件(如 EM-TF-EVK-AM5728-TI-ShipmentImage-SDcard-V1.0.3r04.img)可供用户用 dd 命令(linux 环境)和 Win32DiskImager.exe(Windows 环境)烧录到 SD 卡或者 EMMC 中,具体请参考用户手册。这一节我们将介绍如何将上一节中编译得到的产物做成一个可供烧录使用的镜像文件。

我们制作的这个镜像文件大小不能超过准备烧录的 SD 卡/EMMC 容量的大小。假设 SD 卡为 4GB,我们可以 创建一个 3800MB 的空白磁盘文件,文件名为 example.img,在 ubuntu 下:

#### \$ cd \$HOME

\$ sudo dd if=/dev/zero of=./example.img bs=1M count=3800

3800+0 records in

3800+0 records out

3984588800 bytes (4.0 GB, 3.7 GiB) copied, 9.85035 s, 405 MB/s

#### 3.1.1 分区格式化

将 example.img 分为两个区。第一个区为 FAT32,大小 64MB,用于存放 firmware;第二个分区为 ext4 格式 用于存放根文件系统。

# \$ cd \$HOME \$ sudo fdisk example.img Welcome to fdisk (util-linux 2.27.1). Changes will remain in memory only, until you decide to write them. Be careful before using the write command. Device does not contain a recognized partition table. Created a new DOS disklabel with disk identifier 0x928aa0d6. Command (m for help): o Created a new DOS disklabel with disk identifier 0x12076151. Command (m for help): n **Partition type** primary (0 primary, 0 extended, 4 free) р extended (container for logical partitions) е Select (default p): p



Partition number (1-4, default 1): 1

First sector (2048-7782399, default 2048):

Last sector, +sectors or +size{K,M,G,T,P} (2048-7782399, default 7782399): +64M

Created a new partition 1 of type 'Linux' and of size 64 MiB.

Command (m for help): t

Selected partition 1

Partition type (type L to list all types): c

Changed type of partition 'Linux' to 'W95 FAT32 (LBA)'.

Command (m for help): a

Selected partition 1

The bootable flag on partition 1 is enabled now.

Command (m for help): n Partition type

p primary (1 primary, 0 extended, 3 free)

e extended (container for logical partitions)

Select (default p): p

Partition number (2-4, default 2):

First sector (133120-7782399, default 133120):

Last sector, +sectors or +size{K,M,G,T,P} (133120-7782399, default 7782399):

Created a new partition 2 of type 'Linux' and of size 3.7 GiB.

Command (m for help): p Disk example.img: 3.7 GiB, 3984588800 bytes, 7782400 sectors Units: sectors of 1 \* 512 = 512 bytes Sector size (logical/physical): 512 bytes / 512 bytes I/O size (minimum/optimal): 512 bytes / 512 bytes Disklabel type: dos Disk identifier: 0x12076151 Device Boot Start End Sectors Size Id Type

example.img1 \* 2048 133119 131072 64M c W95 FAT32 (LBA) example.img2 133120 7782399 7649280 3.7G 83 Linux



Command (m for help): w

The partition table has been altered.

Syncing disks.

格式化 FAT32 分区并设置卷标为 boot,格式化 ext4 分区并设置卷标为 rootfs

\$ sudo losetup /dev/loop0 example.img

\$ sudo kpartx -av /dev/loop0

add map loop0p1 (253:0): 0 131072 linear 7:0 2048

add map loop0p2 (253:1): 0 7649280 linear 7:0 133120

\$ Is /dev/mapper/loop0p\*

/dev/mapper/loop0p1 /dev/mapper/loop0p2

\$ sudo mkfs.vfat -F 32 -n "boot" /dev/mapper/loop0p1

mkfs.fat 3.0.28 (2015-05-16)

mkfs.fat: warning - lowercase labels might not work properly with DOS or Windows

unable to get drive geometry, using default 255/63

\$ sudo mkfs.ext4 -L "rootfs" /dev/mapper/loop0p2

mke2fs 1.42.13 (17-May-2015)

Discarding device blocks: done

Creating filesystem with 956160 4k blocks and 239040 inodes

Filesystem UUID: 0820d179-521d-4f91-816f-df13309eee87

Superblock backups stored on blocks:

32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done

Writing inode tables: done

Creating journal (16384 blocks): done

Writing superblocks and filesystem accounting information: done

#### 3.1.2 复制 firmware

创建两个临时目录,分别用于挂载

#### \$ mkdir boot rootfs

\$ sudo mount /dev/mapper/loop0p1 boot/

\$ sudo mount /dev/mapper/loop0p2 rootfs/

#### 通过 lsblk 命令可以查看挂载是否成功

\$ lsblk				
NAME	MAJ:M	N RN	A SIZE	RO TYPE MOUNTPOINT
loop0	7:0	0	3.7G (	Гоор
├──loop0p2	253:1	0	3.7G	0 part /home/david/rootfs
└──loop0p1	253:0	0	64M	0 part /home/david/boot



\$ sudo cp \$HOME/u-boot/u-boot.img ./boot

\$ sudo cp \$HOME/u-boot/MLO./boot

\$ sudo cp \$HOME/linux/arch/arm/boot/zImage./boot

\$ sudo cp \$HOME/linux/arch/arm/boot/dts/embest-SOM\_AM572x\_TM-mode0.dtb ./boot

\$ sudo cp \$HOME/linux/arch/arm/boot/dts/embest-SOM\_AM572x\_TM-mode0-LCD.dtb ./boot

在./boot 中创建 uEnv.txt 文件,指定 dtb 文件,例如以 HDMI 作为主显示屏

\$ sudo touch ./boot/uEnv.txt

\$ sudo bash -c "echo fdtfile=embest-SOM\_AM572x\_TM-mode0.dtb > ./boot/uEnv.txt"

#### 3.1.3 复制根文件系统

解压 tisdk-rootfs-image-am57xx-evm.tar.xz 到\$HOME/rootfs-arago,复制 tisdk-rootfs-image-am57xx-evm.tar.xz 中全部内容到 rootfs 目录,注意此处需要使用 cp -ap 选项,以确保文件的属性保持不变。

\$ mkdir rootfs-arago

\$ tar -Jxvf tisdk-rootfs-image-am57xx-evm.tar.xz -C rootfs-arago/

\$ sudo cp -ap rootfs-arago/\* rootfs

#### 3.1.4 安装内核模块

安装内核模块需要 root 权限,因此也需要给 root 用户设置环境变量

\$ cd \$HOME/linux

\$ sudo make modules\_install INSTALL\_MOD\_PATH=\$HOME/rootfs ARCH=arm

CROSS\_COMPILE=\$HOME/gcc-linaro-6.2.1-2016.11-x86\_64\_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-

#### 3.1.5 安装外部驱动模块

参见 2.3.3 安装外部驱动模块,将 DESTDIR 变量设置为\$HOME/rootfs,然后安装即可。

如果用户没有升级内核版本的情况下,我们提供的 rootfs-arago.tar.gz 根文件系统中已经包含了内核模块以及 外部驱动模块,因此不需要重复安装。

拷贝完成后卸载两个分区,并同步文件系统

\$ sudo umount boot rootfs \$ sudo kpartx -d /dev/loop0 \$ sudo losetup -d /dev/loop0 \$ sync

### 3.2 烧录和读取镜像

3.2.1 烧录

参见用户手册。



#### 3.2.2 读取

开发过程中常需要将 SD 卡中镜像读取出来备份,可以使用如下命令得到 SD 卡的镜像文件

◆ 在 linux 环境下

将 SD 卡接入读卡器连接到电脑

/ti/linux\$ lsblk	
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT	
db 8:16 1 7.2G 0 disk	
sdb2 8:18 1 3.8G 0 part	
└──sdb1 8:17 1 64M 0 part	
6 dd if=/dev/sdb of=./sdcard.img	

♦ 在 windows 环境下

将 SD 卡连接到电脑,运行 Win32 Disk Imager

👒 Win32 Disk Imager	- 0 X
Image File	Device
	[H:\] ▼
Copy MD5 Hash: Progress	
Version: 0.9.5 Cancel Read Write	Exit

选择镜像文件的存放地址,如:D:/temp/SDCard.img

👒 Win32 Disk Imager	
Image File	Device
1	[H:\] 🔻
Copy MD5 Hash:	
Progress	Select Image File
Version: 0.9.5 Cancel	Read Write Exit
	tt.

点击 Read 将 SD 卡的内容读取到镜像文件中:



👒 Win32 Disk Imager	
-Image File	Device
D:/temp/SDCard.img	🔁 [G: \] 🔻
Copy MD5 Hash:	
- Progress	Read to the Image
Version: 0.9.5 Cancel Read	Write Exit
Read data from 'Device' to 'Image File'	H.

执行成功后,你会获得一个完整的 SD 镜像。



# 第4章 TI SDK 开发

# 4.1 SDK 的安装和配置

TI SDK 安装包可以在 ssh 命令行安装,也可以在 ubuntu 桌面环境安装,一下我们以桌面环境安装为例: 在 ubuntu 桌面按组合键 Ctrl+Alt+T 打开控制台

\$ cd \$HOME	
sudo chmod +x ti-processor-sd	k-linux-am57xx-evm-04.00.00.04-Linux-x86-Install.bin
sudo ./ti-processor-sdk-linux-a	m57xx-evm-04.00.00.04-Linux-x86-Install.bin
単出窗口 	
😣 🖨 Setup	
	Setup - ti-processor-sdk-linux-am57xx-evm-04.00.00.04
	Welcome to the ti-processor-sdk-linux-am57xx-evm-04.00.00.04 Setup Wizard.
	< Back Next > Cancel

根据指引信息,点击下一步,出现选择安装路径时,可以使用默认路径安装,也可以自定义安装路径



😣 🖨 Setup
Choose Destination Location
Setup will install ti-processor-sdk-linux-am57xx-evm in the following folder.
To install to this folder, click "Forward". To install to a different folder, click the browse icon and select another folder.
Destination Folder /opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04
InstallBuilder
< <u>B</u> ack <u>N</u> ext > Cancel
😣 🖨 Setup
Installing
Please wait while Setup installs ti-processor-sdk-linux-am57xx-evm-04.00.00.04 on your computer.
Installing
Unpacking /opt/ti-pro[]4/processor-sdk-linux-image-am57xx-evm.tar.xz
InstallBuilder - Cancel

大约几分钟后,安装完成,点击"Finish"





\$ cd /opt/ti	-processor-sdk-linux-am5	57xx-evm-04.00	).00.04/
\$ ls -l			
total 54222	0		
drwxr-xr-x	2smbusersmbuser	4096 Aug 10	16:18 bin
drwxr-xr-x	6smbusersmbuser	4096 Jun 29	2017 board-support
drwxr-xr-x	3smbusersmbuser	4096 Jun 29	2017 docs
drwxr-xr-x 1	19 smbusersmbuser	4096 Jun 29	2017 example-applications
drwxr-xr-x	2smbusersmbuser	4096 Jun 29	2017 filesystem
drwxr-xr-x	3 root root	4096 Aug 10	16:18 linux-devkit
-rwxr-xr-x	1 smbusersmbuser 5551	47047 Jun 29	2017 linux-devkit.sh
-rwxr-xr-x	1 smbusersmbuser	44597 Jun 29	2017 Makefile
-rwxr-xr-x	1 smbusersmbuser	1324 Aug 10	16:18 Rules.make
-rwxr-xr-x	1 smbusersmbuser	4188 Jun 29	2017 setup.sh



#### 其各个目录/文件用途如下:

bin	一些工具脚本,用于制作 SD 卡,设置 TFTP 等
board-support	1, u-boot 源代码;
	2, Kernel 源代码;
	3, 外部驱动源代码;
	4, 针对 TI 的评估板而预编译好的 firmware
docs	SDK 软件列表以及 license
example-applications	Demo 程序源代码
filesystem	基于 Arago 的预构建好的根文件系统
linux-devkit	编译整个 SDK 需要的根文件系统,其中包含了交叉编
	译工具链
linux-devkit.sh	linux-devkit 文件夹的压缩包,移动已经安装好的 SDK
	路径后,需要这个文件
Makefile	顶层 Makefile 文件,可以编译整个 SDK,包括:
	1, Linux Kernel
	2, U-boot
	3, 外部驱动
	4, Demo 程序
Rules.make	编译 SDK 需要用到的一些环境变量,如交叉编辑工具
	链,目标根文件系统路径
setup.sh	设置开发环境,实际是调用 bin 目录下的一些工具脚
	本



# 4.2 顶层 Makefile 使用

SDK 中的顶层 Makefile 中包含了很多目标,通过这个 Makefile 可以全部编译 all, clean, install,也可以单独 指定某个目标进行编译,清除,和安装。

<pre>     Open</pre>	😣 🗇 💷 Makefile (/opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04) - gedit
<pre>Makefile x -include Rules.make MAKE_JOBS ?= 1 all: linux matrix-gui arm-benchmarks am-sysinfo oprofile-example matrix-gui- browser refresh-screen cryptodev debugss-module-drv gdbserverproxy-module-drv openci-examples openmpacc-examples qt-opencv-openci-opengi-multithreaded linalg- examples u-boot-spl omapconf linux-dtbs ti-sgx-ddk-km cmem-mod pru-icss ti- gc320-driver barcode-roi dual-camera-demo image-gallery uto-module-drv ti-gc320- driver ti-ipc big-data-ipc-demo video-graphics-test jallhouse clean: linux_clean matrix-gui-clean arm-benchmarks_clean am-sysinfo_clean oprofile-example_clean matrix-gui-browser_clean refresh-screen_clean cryptodev_clean debugss-module-drv_clean gdbserverproxy-module-drv_clean opencl- examples_clean openmpacc-examples_clean qt-opencv-opencl-opengi- multithreaded_clean linalg-examples_clean gt-opencv-opencl-opengi- multithreaded_clean linalg-examples_clean pru-icss_clean ti-gc320- driver_clean barcode-roi_clean dual-camera-demo_clean image-gallery_clean uio- module-drv_clean ti-gs20-driver_clean ti-ipc_clean big-data-ipc-demo_clean video-graphics-test_clean jailhouse_clean install: linux_install matrix-gui_install arm-benchmarks_install am- sysinfo_install oprofile-example_install matrix-gui-browser_install refresh- screen_install oprofile-example_install inpenmpacc-examples_install gdbserverproxy- module-drv_install openci-examples_install u-boot-spl_install omapcon_install linux-dtbs_install ti-sgx.ddk-km_install cmem-mod_install pru- icss_install ti-ipc_install big-data-ipc-demo_install ti-gc320- driver_install ti-ipc_install big-data-ipc-demo_install video-graphics- test_install jailhouse_install</pre>	📄 📄 Open 👻 Save 🛛 🛃 🖌 Undo 🧀 🖌 🖷 🏥 🔍 🛠
<pre>-include Rules.make MAKE_JOBS ?= 1 all: linux matrix-gui arm-benchmarks am-sysinfo oprofile-example matrix-gui- browser refresh-screen cryptodev debugss-module-drv gdbserverproxy-module-drv opencl-examples openmpacc-examples qt-opencv-opencl-opengl-multithreaded linalg- examples u-boot-spl omapconf linux-dtbs ti-sgx-ddk-km cmem-mod pru-tcss ti- gc320-driver barcode-roi dual-camera-demo image-gallery uio-module-drv ti-gc320- driver ti-ipc big-data-ipc-demo video-graphics-test jailhouse clean: linux_clean matrix-gui_clean arm-benchmarks_clean am-sysinfo_clean oprofile-example_clean matrix-gui-browser_clean refresh-screen_clean cryptodev_clean debugss-module-drv_clean gdbserverproxy-module-drv_clean opencl- examples_clean openmpacc-examples_clean qt-opencv-opencl-opengl- multithreaded_clean linalg-examples_clean pru-icss_clean ti-gc320- driver_clean barcode-roi_clean dual-camera-demo_clean image-gallery_clean uio- module-drv_clean ti-gg20-driver_clean ti-ipc_clean big-data-ipc-demo_clean video-graphics-test_clean jailhouse_clean install: linux_install matrix-gui_install arm-benchmarks_install am- sysinfo_install oprofile-example_install matrix-gui-browser_install refresh- screen_install oprofile-example_install inpenmpacc-examples_install gdbserverproxy- opencl-opengl-multithreaded_install linalg-examples_install u-boot-spl_install omapconf_install linux-dtbs_install ti-sgx-ddk-km_install cmem-mod_install pru- icss_install ti-ipc_install big-data-ipc-demo_install ti-gc320- driver_install in-tipc_install big-data-ipc-demo_install video-graphics- test_install jailhouse_install</pre>	🗋 Makefile 🗙
<pre>MAKE_JOBS ?= 1 all: linux matrix-gui arm-benchmarks am-sysinfo oprofile-example matrix-gui- browser refresh-screen cryptodev debugss-module-drv gdbserverproxy-module-drv opencl-examples opennpacc-examples qt-opencv-opencl-opengl-multithreaded linalg- examples u-boot-spl omapconf linux-dtbs ti-sgx-ddk-km cmem-mod pru-icss ti- gc320-driver barcode-roi dual-camera-demo image-gallery uio-module-drv ti-gc320- driver ti-ipc big-data-ipc-demo video-graphics-test jailhouse clean: linux_clean matrix-gui_clean arm-benchmarks_clean am-sysinfo_clean oprofile-example_clean matrix-gui-clean gdbserverproxy-module-drv_clean opencl- examples_clean openmpacc-examples_clean u-opencv-opencl-opengl- multithreaded_clean linalg-examples_clean u-opencv-opencl-opengl- multithreaded_clean linalg-examples_clean u-boot-spl_clean inge-gallery_clean uio- module-drv_clean ti-gc320-driver_clean ti-ipc_clean big-data-ipc-demo_clean video-graphics-test_clean jailhouse_clean install: linux_install matrix-gui_install arm-benchmarks_install am- sysinfo_install opencl-examples_install openmpacc-examples_install qt-opencv- opencl-opengl-multithreaded_install ti-sgx-ddk-km_install ceme-mod_install qt-opencv- opencl-opengl-multithreaded_install ti-sgx-ddk-km_install came-mod_install pru- icss_install ti-gc320-driver_install big-data-ipc-demo_examples_install qual-camera- demo_install ti-gc320-driver_install big-data-ipc-demo_install fi-gc320- driver_install ti-ipc_install big-data-ipc-demo_install video-graphics- test_install ti-ipc_install big-data-ipc-demo_install video-graphics- test_install ti-ipc_install big-data-ipc-demo_install video-graphics- test_install invx-dtbs</pre>	- <b>include</b> Rules.make
<pre>all: linux matrix-gui arm-benchmarks am-sysinfo oprofile-example matrix-gui- browser refresh-screen cryptodev debugss-module-drv gdbserverproxy-module-drv opencl-examples openmpacc-examples qt-openct-opencl-opengl-multithreaded linalg- examples u-boot-spl omapconf linux-dtbs ti-sgx-ddk-km cmem-mod pru-icss ti- gc320-driver barcode-roi dual-camera-demo image-gallery uio-module-drv ti-gc320- driver ti-ipc big-data-ipc-demo video-graphics-test jailhouse clean: linux_clean matrix-gui_clean arm-benchmarks_clean am-sysinfo_clean oprofile-example_clean matrix-gui_browser_clean refresh-screen_clean cryptodev_clean debugss-module-drv_clean gdbserverproxy-module-drv_clean opencl- examples_clean openmpacc-examples_clean u-boot-spl_clean omapconf_clean linux- dtbs_clean ti-sgx-ddk-km_clean cmem-mod_clean pru-icss_clean ti-gc320- driver_clean barcode-roi_clean dual-camera-demo_clean image-gallery_clean uio- module-drv_clean ti-gc320-driver_clean ti-ipc_clean big-data-ipc-demo_clean video-graphics-test_clean jailhouse_clean install: linux_install matrix-gui_install amtrix-gui-browser_install refresh- screen_install oprofile-example_install matrix-gui-browser_install refresh- screen_install opencl-examples_install openmpacc-examples_install qt-opencv- opencl-opengl-multithreaded_install linalg-examples_install u-boot-spl_install omapconf_install ti-gc320-driver_install barcode-roi_install dual-camera- demo_install inaus-dtbs_install ti-sgx-ddk-km_install cmem-mod_install pru- icss_install ti-ipc_install big-data-ipc-demo_install ti-gc320- driver_install inaus-gui_install big-data-ipc-demo_install video-graphics- test_install jailhouse_install # Kernel build targets linux: linux-dtbs @echo ====================================</pre>	MAKE_JOBS ?= 1
	all: linux matrix-gui arm-benchmarks am-sysinfo oprofile-example matrix-gui- browser refresh-screen cryptodev debugss-module-drv gdbserverproxy-module-drv opencl-examples openmpacc-examples qt-openct-opencl-opengl-multithreaded linalg- examples u-boot-spl omapconf linux-dtbs ti-sgx-ddk-km cmem-mod pru-icss ti- gc320-driver barcode-roi dual-camera-demo image-gallery uio-module-drv ti-gc320- driver ti-ipc big-data-ipc-demo video-graphics-test jailhouse clean: linux_clean matrix-gui_clean arm-benchmarks_clean am-sysinfo_clean oprofile-example_clean matrix-gui_browser_clean refresh-screen_clean cryptodev_clean debugss-module-drv_clean gdbserverproxy-module-drv_clean opencl- examples_clean openmpacc-examples_clean u-boot-spl_clean omapconf_clean linux- dtbs_clean ti-sgx-ddk-km_clean cmem-mod_clean pru-icss_clean ti-gc320- driver_clean barcode-roi_clean dual-camera-demo_clean image-gallery_clean uio- module-drv_clean ti-gc320-driver_clean ti-ipc_clean big-data-ipc-demo_clean video-graphics-test_clean jailhouse_clean install: linux_install matrix-gui_install arm-benchmarks_install am- sysinfo_install oprofile-example_install matrix-gui-browser_install refresh- screen_install cryptodev_install debugss-module-drv_install gdbserverproxy- module-drv_install opencl-examples_install u-boot-spl_install omapconf_install linux-dtbs_install ti-sgx-ddk-km_install cmem-mod_install pru- icss_install ti-ipc_install big-data-ipc-demo_install dual-camera- demo_install linux-dtbs_install ti-sgx-ddk-km_install cme-mod_install pru- icss_install ti-ipc_install uio-module-drv_install video-graphics- test_install jailhouse_install uio-module-drv_install video-graphics- test_install jailhouse_install wide-ode-roi_install video-graphics- test_install jailhouse_install wide-drw_install video



#### 4.2.1 编译整个 SDK

顶层 Makefile 默认目标 all 可以编译整个 SDK,因此只需要一条命令即可:

#### \$ sudo make

编译过程从第一个目标 linux 开始解析,而 linux 依赖的第一个目标是 linux-dtbs,因此最先执行的编译目标 是 linux-debs。需要一定的时间,编译完成后如需安装编译产物到目标文件系统,则需要修改 Rule.make 中的 DESTDIR 为实际根文件系统所在目录。此处设置 DESTDIR=/opt/fakeroot,首先要先创建这个目录,然后 make install。

#### \$ cd /opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04

\$ sudo make install \$ cd ../fakeroot \$ ls -l total 12 drwxr-xr-x 2 root root 4096 Aug 10 17:39 boot drwxr-xr-x 4 root root 4096 Aug 10 17:39 lib drwxr-xr-x 4 root root 4096 Aug 10 17:39 usr

#### 4.2.2 单独编译安装某个目标

以编译 dual-camera 这个 demo 程序为例子:

\$ cd /opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04

\$ sudo make dual-camera-demo

Makefile:715: warning: overriding commands for target `ti-gc320-driver'

Makefile:595: warning: ignoring old commands for target `ti-gc320-driver'

Makefile:723: warning: overriding commands for target `ti-gc320-driver\_clean'

Makefile:603: warning: ignoring old commands for target `ti-gc320-driver\_clean'

Makefile:731: warning: overriding commands for target `ti-gc320-driver\_install'

Makefile:611: warning: ignoring old commands for target `ti-gc320-driver\_install'

\_\_\_\_\_

**Building Dual Camera Demo** 

\_\_\_\_\_

make[1]: Entering directory

`/opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/example-applications/dual-camera-demo-1.0'

echo "manisha" am57xx-evm

manisha am57xx-evm

make[2]: Entering directory

`/opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/example-applications/dual-camera-demo-1.0'

make[2]: Nothing to be done for `first'.

make[2]: Leaving directory



`/opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/example-applications/dual-camera-demo-1.0'

make[1]: Leaving directory

) opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/example-applications/dual-camera-demo-1.0

david@ubuntu:/opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04\$

安装

\$ sudo make dual-camera-demo\_install

Makefile:715: warning: overriding commands for target `ti-gc320-driver'

Makefile:595: warning: ignoring old commands for target `ti-gc320-driver'

Makefile:723: warning: overriding commands for target `ti-gc320-driver\_clean'

Makefile:603: warning: ignoring old commands for target `ti-gc320-driver\_clean'

Makefile:731: warning: overriding commands for target `ti-gc320-driver\_install'

Makefile:611: warning: ignoring old commands for target `ti-gc320-driver\_install'

Installing Dual Camera Demo - Release version

\_\_\_\_\_

make[1]: Entering directory

) opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/example-applications/dual-camera-demo-1.0

echo "manisha" am57xx-evm

manisha am57xx-evm

make[2]: Entering directory

`/opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/example-applications/dual-camera-demo-1.0'

make[2]: Nothing to be done for `first'.

make[2]: Leaving directory

`/opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/example-applications/dual-camera-demo-1.0' dual\_camera release version installed.

make[1]: Leaving directory

`/opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/example-applications/dual-camera-demo-1.0'



# 第5章 应用开发

### 5.1 普通 C 程序的交叉编译和执行

#### 5.1.1 普通 C 程序的交叉编译和执行

解压 application.tar.gz 到\$HOME,以编译 4G 通信测试程序为例:

#### 5.1.2 交叉编译

#### \$ cd \$HOME/application/4g\_test

\$ \$HOME/gcc-linaro-6.2.1-2016.11-x86\_64\_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-gcc 4G\_test.c -o

4G\_test

\$ ls

4G\_test 4G\_test.c readme.md

\$ file 4G\_test

4G\_test: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.24, BuildID[sha1]=69a5b54a2de0c56b075f871fff6710797250a72c, not stripped

#### 5.1.3 直接编译

我们发布的根文件系统中已经安装了编译工具链,因此可以复制 c 源代码文件到直接在板上使用 gcc 编译应 用程序

# gcc 4G\_test.c -o 4G\_test # ls

4G\_test 4G\_test.

#### 5.1.4 传递到开发板运行

◆ 将开发板与电脑接入统一局域网,使用 scp 命令传递文件

#### # scp \$HOME/application\_test\_programs/4g\_test/4g\_test ./

◆ 使用 U 盘等存储介质拷贝

略



# 5.2 QT 开发

### 5.2.1 安装 Qt Creator

Qt Creator 是一个图形化的设计器,因此这一部分的操作均在 ubuntu 桌面环境下进行。拷贝 qt-creator-opensource-linux-x86\_64-4.1.0.run 到\$HOME,增加执行权限。

\$ sudo chmod +x qt-creator-opensource-linux-x86\_64-4.1.0.run

执行安装程序

\$./qt-creator-opensource-linux-x86\_64-4.1.0.run

弹出图形窗口:

😣 💷 🛛 Qt Creator 4.1.0 Setup

**Ready to Install** 

Setup is now ready to begin installing Qt Creator 4.1.0 on your computer. Installation will use 258.04 MiB of disk space.

< <u>B</u> ack Install Cancel



😣 🗉 Qt Creator 4.1.0	Setup	
Qt Account - Your uni	fied 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 Once.		Confirm Password
Deploy Everywhere.		I accept the <u>service terms</u> .
Settings		< Back Skip Cancel



😣 🗊 Qt Creator 4.1.0	Setup
Setup - Qt Creator 4.1	0
Qt	Welcome to open source Qt Creator 4.1.0 setup.
Create Once. Deploy Everywhere.	
	< <u>B</u> ack <u>N</u> ext > Quit





/home/david/gtcrea	ator-4.1.0		Browse
			 -



😣 🗉 Qt Creator 4.1.0 Setup	
License Agreement Please read the following license agreement. You m in this agreement before continuing with the installa	ust accept the terms co ation.
This is the GNU General Public License version 3, annot Qt Company GPL Exception 1.0:	tated with The
The Qt Company GPL Exception 1.0	
As a special exception you may create a larger work we output of this application and distribute that work under choice, so long as the work is not otherwise derived from this application and so long as the work does not in its output that contains the output from this application in or modified form.	hich contains the er terms of your om or based on elf generate its original
Exception 2: As a special exception, you have permission to combine application with Plugins licensed under the terms of your choice, to I have read and agree to the terms contained in the agreements. I do not accept the terms and conditions of the above	e this produce an v license
agreements.	
< <u>B</u> ack <u>N</u> e	ext > Cancel



😣 🗉 Qt Creator 4.1.0 Setup
Ready to Install
Setup is now ready to begin installing Qt Creator 4.1.0 on your computer. Installation will use 258.04 MiB of disk space.
< <u>B</u> ack <u>I</u> nstall Cancel

根据指引信息,将 Qt creator 安装到\$HOME/qtcreator-4.1.0 安装完成后如下图:



😣 🗉 Qt Creator 4.1.0	Setup
Completing the Qt Cr	eator 4.1.0 Wizard
Qt	Click Finish to exit the Qt Creator 4.1.0 Wizard. ✓ Launch Qt Creator
Create Once. Deploy Everywhere.	
	Finish

最后一步先不要启动 Qt Creator,需要从 TI SDK 中启动,因为有一些必要的环境变量需要设置。

\$ source /opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/linux-devkit/environment-setup

然后切换到 Qt Creator 安装目录./qtcreator 启动

[linux-devkit]:~> cd ~/qtcreator-4.1.0/bin/

[linux-devkit]:~/qtcreator-4.1.0/bin> ./qtcreator



### 5.2.2 配置 Qt Creator

#### Qt Creator 启动后如下图所示:

	Qt Creator				_
<u>File</u> <u>E</u> dit	<u>Build Debug Analyze Tools M</u>	<u>/</u> indow <u>H</u> elp			
Welcome	Projects	+ New Project		🕞 Open Project	
Edit	Examples	Sessions		Recent Projects	
Design	Tutorials	▶ default			
🗰 Debug	New to Qt?				
<b>پر</b> Projects	Learn how to develop your own applications and explore Qt Creator.				
<b>?</b> Help	Get Started Now				
	Qt Account Online Community				
	Blogs				
	Oser Guide				
	■ P. Type to locate (Ctrl+K)	1 Issues 2 Search Results	3 Application Output	4 Compile Output 5 Debugger Co	onsole 🗢 🔺

在使用 Qt Creator 开发在 AM5728 上运行的程序前,需要配置交叉编译工具、QT 版本、Debuggers 版本、Kits。



#### ◆ 配置交叉编译工具链

点击菜单栏 Tool->Options->Build&Run->Compilers->Add->GCC, 点击 Browse 选择 Compiler path 为 /opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/linux-devkit/sysroots/x86\_64-arago-linux/usr/bin/arm-linuxgnueabihf-g++(TI SDK 的 gcc 路径), 然后点击 Open, 点击 Apply

Options		
Filter	Build & Run	
Environment	General         Kits         Qt Versions         Compilers         Debuggers         CMake	
Text Editor	Name Type	Add -
FakeVim	GCC (x86 64bit in /usr/bin) GCC	Clone
P Help	▼ Manual	Remove
{} c++		
Qt Quick		
🕕 Build & Run		
Debugger		
📡 Designer		
Analyzer	Name: GCC	
Version Control	Complier path: linux-devkit/sysroots/x86_64-arago-linux/usr/bin/arm-linux-gnueabim-g++ Browse	
🙀 Android	Platform linker flags:	
®anx QNX	ABI: arm-linux - linux - generic - elf - 32bit -	
Devices		
Code Pasting		
	Apply <u>C</u> ancel	<u>о</u> к



#### ◆ 配置 QT 版本

点击"Qt Versions->Add",选择 TI SDK 的 qmake 文件路径:

/opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/linux-devkit/sysroots/x86\_64-arago-linux/usr/bin/qt5/qmak e,然后点击 Apply

😣 🗉 Options	
Filter	Build & Run
Environment	General Kits Qt Versions Compilers Debuggers CMake
Text Editor	Name regraded Ty
FakeVim	▲ Manual Manual 05.5.7.1 in PATH (at5) /ont/ti-processor-ed/Jinux-am57xx-evm-04.00.0 /svsroots/x86.64-arago_linux/usr/hin/at5/amake
Help	
<b>{}</b> c++	
Qt Quick	
🕕 Build & Run	
🔎 Debugger	
💓 Designer	
Analyzer	
Version Control	
i Android	
≑anx QNX	
Devices	
Code Pasting	Version name: Qt %{Qt:Version} in PATH (qt5)
interview 🔁 🔁 🔁 🔁 🔁	qmake location: /opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/qt5/qmake
	Qt version 5.7.1 for Embedded Linux
	Apply <u>C</u> ancel <u>O</u> K



#### ♦ 配置 Debuggers 版本

点击 Debuggers 选项,点击 Add,点击 Browse,选择交叉编译器安装目录下的 GDB 编译器,如: /home/david/ti/gcc-linaro-6.2.1-2016.11-x86\_64\_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-gdb 更改 Name 选项,输入 AM5728-Debugger,点击 Apply,完成设置

😣 🗊 Options	
Filter	Build & Run
Environment	General         Kits         Qt Versions         Compilers         Debuggers         CMake
Text Editor	Name Location Add
FakeVim	System GDB at /usr/bin/gdb /usr/bin/gdb Clone
Help	AM5728-Debugger /home/david/ti/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf/bin/arm-linu Remove
<b>{}</b> C++	
Qt Quick	
🕕 Build & Run	
🔍 Debugger	
💓 Designer	
Analyzer	
Version Control	
🏟 Android	
∋anx QNX	
Devices	Name: AM5728-Debugger
Code Pasting	Path: id/ti/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-gdb Browse
> Obs	Type: GDB
	ABIs: arm-linux-generic-elf-32bit
	Version: 2016.11.0
	Apply <u>C</u> ancel <u>O</u> K



#### ◆ 配置 Kits

点击菜单栏"Tool->Options->Build & Run->Kits->Add",更改 Name 为 AM5728, Device type 为 Generic Linux Device,在 Qt mkspec 选项中输入 linux-oe-g++,配置完后点击 Apply,点击 OK

😣 🗉 Options				
Filter	Build & Run			
Environment	General Kits Qt	Versions Compilers	Debuggers	CMake
Text Editor	Name Auto-detected ▼ Manual 4 AM5728			
<ul> <li>Help</li> <li>C++</li> </ul>	\rm Desktop (de	fault)		
Qt Quick	Name: File system name:	AM5728		
Debugger	Device type:	Generic Linux Device		
🧏 Designer	Device:			
Analyzer	Sysroot: Compiler:	GCC (x86 64bit in /usr,	/bin)	
🧔 Android	Environment:	No changes to apply.		
∋anx QNX	Debugger:	AM5728-Debugger		
Devices	Qt version:	Qt 5.7.1 in PATH (qt5)		
Code Pasting	CMake Tool:	System CMake at /opt	/ti-processor-sdk	k-linux-am57xx-evm-04.00.00.04/linux-devkit/sysroots/x86_64-arago-lin
🥕 Qbs	CMake Generator: CMake Configuration	CodeLite - Ninja CMAKE_CXX_COMPILER	::STRING=%{Co	ompiler:Executable}; QT_QMAKE_EXECUTABLE:STRING=%{Qt:qmakeEx
				Apply <u>C</u> ancel <u>O</u> K



### 5.2.3 创建 demo

本章以一个简单的窗口程序 hello-world,介绍 Qt 界面的开发流程。

◆ 新建工程

同样,从 SDK 中启动 Qt Creator,源代码存放到/home/david/ti

800	Qt Creator				
<u>F</u> ile <u>E</u> dit	<u>Build Debug Analyze Tools Wir</u>	ndow <u>H</u> elp			
Welcome	Projects	+ New Project		🝃 Open Project	
Edit	Examples	Sessions		Recent Projects	
Design	Tutorials	default			
û Debug	New to Ot?				
<b>پ</b> Projects	Learn how to develop your own applications and explore Qt Creator.				
<b>?</b> Help	Get Started Now				
	Qt Account				
	Blogs				
	😮 User Guide				
<b>A</b> R					
~	■ P. Type to locate (Ctrl+K)	1 Issues 2 Search Results 3	Application Output 4	Compile Output 5 Debugger Co	nsole 💠 🔺



#### 点击左上角 File-New File or Project 菜单

😕 🔲 New File d	or Project		
Choose a template	:		All Templates 👻
Projects		🗖 Qt Widgets Application	Creates a Ot application for the deskton
Application Library Other Project Non-Qt Project Import Project Files and Classes C++ Modeling Qt GLSL General Java Python		<ul> <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 Canvas 3D Application</li> <li>Qt Labs Controls Application</li> </ul>	Includes a Qt Designer-based main window. Preselects a desktop Qt for building the application if available. Supported Platforms: Generic Linux Device
Qt Widget     Cocation     Kits     Details	s Application Introduction and This wizard generates a Qt an empty widget.	<b>Project Location</b> Widgets Application project. The application de	<u>Cancel</u> Choose erives by default from QApplication and includes
Summary	Name: hello-world		
	Create in: /home/david/	qt : location	Browse
			Next > Cancel



#### 选择 Kit: AM5728

80	Qt Widgets	Application				
	Location	Kit Select	on			
=>	Kits Details	Qt Creator can use the following kits for project <b>hello-world</b> :  Select all kits				
	Summary	✓ 🖵 AM5728				
		✓ Debug	/home/david/qt/build-hello-world-AM5728-Debug		Browse	
		✓ Release	/home/david/qt/build-hello-world-AM5728-Release		Browse	
		✓ Profile	/home/david/qt/build-hello-world-AM5728-Profile		Browse	
		🗌 🖵 Deskt	ор		Details 🔻	
				< <u>B</u> ack <u>N</u> ext >	Cancel	

#### 选择 Base Class: Qwidget

😣 🗉 🛛 Qt Wi	idgets Application	
Locatio Kits	Class Info Specify basic in	rmation formation about the classes for which you want to generate skeleton source code files.
Details Summa	ary <u>C</u> lass name: <u>B</u> ase class:	Widget QWidget
	<u>H</u> eader file: <u>S</u> ource file: <u>G</u> enerate form: Form file:	widget.h widget.cpp
	-	
		< <u>B</u> ack <u>Next</u> > Cancel



#### 点击 Finish 进入到新建工程

😣 🗈	Qt Widgets	Application								
	Location	Project Management								
	Kits	Add as a subproject to project:	<none></none>							
<b></b>	Details Summary	Add to version control:	<none> •</none>	Configure						
		Files to be added in								
		/home/david/qt/hello-worl	d:							
		hello-world.pro main.cpp widget.cpp widget.h widget.ui								
			< <u>B</u> ack <u>Finish</u>	Cancel						

◆ 设计 UI 界面

<u>File Edit Build Debug Analyze Tools</u>	<u>W</u> indow <u>H</u> elp
Projects 🗢 🗧 🕀	📼 < 🖒 🖬 🌛 widget.ui 🔶 🗧 🗧
▼ 👼 hello-world	This file can only be edited in <b>Design</b> mode.
Welcome  Headers	<pre>1 <?xml version="1.0" encoding="UTF-8"?> 2 </pre>
📰 🔍 🔽 Forms	<pre>3 <class>Widget</class></pre>
Edit 📝 widget.ui	4 ▼ <widget class="QWidget" name="Widget"> 5 ▼ <property name="geometry"></property></widget>
	6 ▼ <rect></rect>
	7 <x>0</x>
Design	8 <y>0</y>
	10 <height>300</height>
	11
Debug	12
يو ا	14 <pre>string&gt;Widget</pre>
Projects	15
	16
• • •	17 <layoutdefault margin="11" s<="" spacing="6" td=""></layoutdefault>
Help	19 <connections></connections>
	20
	21



#### 双击 widget.ui 进入 UI 设计器

<u>File</u> Edit	<u>B</u> uild	<u>D</u> ebug	<u>A</u> nalyze	Tools	<u>W</u> indow	<u>H</u> elp	D										
	n 📝	widget.u	i		⇒   ×		¥. 6	5 🕮	111	≡	ы	30		 5 🖻			
	Filter				•									 	 		
	-		Layouts														
Welcome	= v	ertical Lay	out				· · ·			· · ·			· · ·				
	000 н	orizontal L	ayout														1
Edit	G	rid Layout															
1	BB F	orm Layout	t														
Design	-		Spacers														
	[200] H	orizontal S	pacer		•												1
Debug	<u>x</u> v	ertical Spa	cer														
Debug	-		Buttons														
بر	ок р	ush Button															
Projects	<u></u>	ool Button															
2	● R	adio Buttor	n														
Help	S 🗹	heck Box					• • •				• • •	 -	• • •	 	 • •	• • •	 1
	🕤 C	ommand L	ink Button														
	<b>√x</b> D	ialog Butto	n Box														

在右下角的 Property 属性窗口中,修改 windowTitle 字段,这里输入 hello-world





#### 在左侧 Buttons 窗口拖动一个 Push Button 到 UI 上,双击修改名称为 close



选中 close 按钮,右键选择 Go to slot 进入到代码编辑,

QAbstractButton
QAbstractButton
QAbstractButton
QAbstractButton
QAbstractButton
QObject 💌

◆ 编辑代码

编辑 on\_pushButton\_clicked 函数的响应代码,这里输入 close(),功能是关闭窗口。

◆ 编译

点击左下角的锤子形状 的快捷按钮,编译工程,生产的可执行文件在

\$HOME/qt/build-hello-world-AM5728-Debug 目录



通过 file 命令可以看到这是一个运行在 ARM 平台的可执行程序

#### \$ file hello-world

hello-world: ELF 32-bit LSB executable, ARM, EABI5 version 1 (GNU/Linux), dynamically linked (uses shared libs), for GNU/Linux 2.6.32, BuildID[sha1]=cd9018247cd88be33eb2f59fb56fe7af7fee37ea, not stripped

😣 🕒 🗉 🛛 widget.cpp - hell	o-world - Qt Q	reator				
<u>File Edit Build Debug A</u>	nalyze <u>T</u> ools	<u>W</u> indow <u>H</u> el	р			
Projects	≑ ▼. ⇔ ⊟+	🖂 < 🖂 🖬	ု 📩 widget.cpp	<b>≑</b>  ×  ◆	Widget::on_pushButton_clicked(): voi	d \$ # »⊟+
Welcome Welcome Edit Pesign Projects	d.pro ui	1 # 2 # 3 4 W 5 6 7 7 8 9 } 10 7 8 9 } 10 7 4 13 14 } 12 { 13 14 } 15 7 8 10 7 7 8	<pre>include "widget.h" include "ui_widget.h" idget::Widget(QWidget     QWidget(parent),     ui(new Ui::Widget))     ui-&gt;setupUi(this); idget::~Widget()     delete ui; oid Widget::on_pushBut     close();</pre>	*parent) : ton_clicked()		
Debug	¢ ⊟+	Compile 0 processo usr/incl sysroots am57xx-e QtCore - x86_64-a arm-linu devkit/s opt/ti-p gnueabi/ 23:51:36 23:51:36	Dutput 4- r-sdk-linux-am57xx-evm ude/qt5/QtWidgets -I/o /armv7ahf-neon-linux-g ym-04.00.00.04/linux-d III/opt/ti-proce rago-linux/mkspecs/lin x-gnueabihf-g++sys ysroots/armv7ahf-neon- rocessor-sdk-linux-am5 usr/lib -lQt5Widgets - : The process "/usr/bi : Elapsed time: 00:02. 2 Search Results 3 AM	+ -04.00.00.04/linux opt/ti-processor-su- levkit/sysroots/arr essor-sdk-linux-am ux-oe-g++ - 0 widg proot=/opt/ti-proce linux-gnueabi - 0 i7xx-evm-04.00.00.1 Qt5Gui - l0t5Core n/make" exited nor optication Output 4	x-devkit/sysroots/armv7ahf-neon- dk-linux-am57xx-evm-04.00.00.04/ e/qt5/QtGui -I/opt/ti-processor- mv7ahf-neon-linux-gnueabi/usr/in 57xx-evm-04.00.00.04/linux-devki t.o/hello-world/widget.cpp essor-sdk-linux-am57xx-evm-04.00 hello-world main.o widget.o moc_ 04/linux-devkit/sysroots/armv7ah -IGLESv2 -lpthread rmally.	<pre>Invasion of the second se</pre>

#### 5.2.4 在 AM5728 板上运行

♦ 在 weston 环境运行

将生产的执行文件 hello-world 拷贝到 AM5728 板上,./hello-world 运行。运行后的显示如下:



#### EM-TF-EVK-AM5728-Linux\_DG-V1.0



鼠标点击 close 按钮(或者触摸屏点击)即可关闭窗口。

♦ 脱离 Weston 运行

QT 程序也可以脱离 weston 环境运行,先关闭 weston

- # /etc/init.d/weston stop
- # ./hello-world -platform eglfs

	2.12	and a second sec	", ', <u>``</u>	the second se	Sime U
	A	and the second			
close	1				-
					11
					and the second

或者

# ./hello-world -platform linuxfb







## 5.3 视屏采集 demo

视频采集功能上有预览,拍照,录像。运行 UI 界面中 Camera 应用(图标为 camera),可以测试视频的预览和拍照。

- ◆ Capture: 拍照快门, 用于拍摄一张图片
- ◆ Switch: 交换显示屏幕中的主显示和副显示窗口
- ◆ PIP: 关闭副显示窗口
- ◆ Exit: 退出应用



拍摄后的照片可以在 Gallery 中浏览 (图标为 Image Gallery), 其界面大致如下,







Delete、Quit、Previous、Next 按钮分别是删除当前图片、退出应用、上一张图片、下一张图片。

视频录制即录像功能,由 gstreamer 来实现,命令行参考:

# gst-launch-1.0 -e v4l2src device=/dev/video2 num-buffers=1000 io-mode=5 ! 'video/x-raw, \
format=(string)NV12, width=(int)1920, height=(int)1080, framerate=(fraction)30/1' ! ducatimpeg4enc
bitrate=12000 ! \

queue !mpeg4videoparse !qtmux !filesink location=out.mp4

播放上面录制的文件,用下面的命令(假设生成的 mp4 文件全路径是:/media/out.mp4):

# gst-launch-1.0playbinuri=file:///media/out.mp4 video-sink=kmssink

### 5.4 双屏显示 demo

双显示的一个 Demo,可以演示 OpenGL、双摄像头、双显示同时工作。此应用通过 OpenGL 构造一个旋转正 方体,通过 V4L2 视频采集接口获取双摄像头数据,然后分别显示在 LCD 和 HDMI 显示屏幕上。用来演示平 台强大的视频处理和显示能力。前提需要关闭 Weston

#### #/etc/init.d/weston stop

具体的测试命令如下:

◆ 双屏分别显示旋转立方体和一个摄像头

# kmscube-camera -a



◆ 双屏分别显示旋转立方体和 #0 摄像头

#### #kmscube-camera -a -i 0

◆ 双屏分别显示旋转立方体和 #1 摄像头

#kmscube-camera -a -i 1

◆ 单屏显示旋转立方体

#kmscube-camera -c 32

# kmscube-camera -c 36

◆ 双屏显示,一个屏幕显示旋转立方体,一个屏幕交替显示两个摄像头

#kmscube-camera -a -i 2



# 第6章 基于 Yocto 的根文件系统构建

Yocto 可以构建出完整的嵌入式系统镜像,本文侧重于构建根文件系统。更多关于 Yocto 的信息请访问官方 网站 <u>https://www.yoctoproject.org/docs/2.5/mega-manual/mega-manual.html</u>

搭建 Yocot 构建环境需要 PC 机有较好的硬件性能和足够的内存和硬件空间,并且需要充裕的网络带宽,推荐至少 200G 硬件空间和 8G 内存。

## 6.1 安装需要的工具软件

\$ sudo apt-get install git build-essential python diffstattexinfo gawk chrpath dos2unix wget unzip socatdoxygen libc6:i386 libncurses5:i386 libstdc++6:i386 libz1:i386

### 6.2 配置 bash

\$ sudo dpkg-reconfigure dash

Configuring dash
The system shell is the default command interpreter for shell scripts.
Using dash as the system shell will improve the system's overall performance. It does not alter the shell presented to interactive users.
Use dash as the default system shell (/bin/sh)?
<yes> ♪</yes>

选择"no"

### 6.3 安装编译器

如果此前已经安装好了交叉编译工具链,此处可以忽略

\$ wget

https://releases.linaro.org/components/toolchain/binaries/6.2-2016.11/arm-linux-gnueabihf/gcc-linaro-6.2.1-2016.11-x86\_64\_arm-linux-gnueabihf.tar.xz

2010.11-X80\_04\_arm-infux-grideabim.tar.xz

\$ tar -Jxvf gcc-linaro-6.2.1-2016.11-x86\_64\_arm-linux-gnueabihf.tar.xz -C \$HOME

### 6.4 获取 oe-layertool-setup.sh

\$ git clone git://arago-project.org/git/projects/oe-layersetup.git tisdk

\$ cd tisdk

\$ ./oe-layertool-setup.sh -f configs/processor-sdk/processor-sdk-04.00.00.04-config.txt



# 6.5 bitbake 构建

\$ cd ..

\$ cd build

\$ . conf/setenv

\$ export PATH=\$HOME/gcc-linaro-6.2.1-2016.11-x86\_64\_arm-linux-gnueabihf/bin:\$PATH

\$ MACHINE=am57xx-evm bitbake arago-core-tisdk-image

初次构建需要花费较长时间,大约几小时到几十小时不等,具体时间因 PC 性能和网络带宽而异。 构建完成后,生成的文件系统在

tisdk/build/arago-tmp-external-linaro-toolchain/deploy/images/am57xx-evm/tisdk-rootfs-image-am57xx-evm.tar. xz





# 7.1 硬件

详细硬件介绍请参考:

核心板:《EM-TF-SOM-AM5728 硬件用户手册》

底板: 《EM-TF-BB-AM5728 硬件用户手册》



# 第8章 技术支持和保修服务

#### 8.1.1 技术支持

英蓓特科技对所销售的产品提供一年的免费技术支持服务,技术支持服务范围:

- ◆ 提供英蓓特科技嵌入式平台产品的软硬件资源;
- 帮助用户正确地编译和运行我们提供的源代码;
- 用户在按照本公司提供的产品文档操作的情况下,如本公司的嵌入式软硬件产品出现异常问题,我们 将提供技术支持;
- ◆ 帮助用户判定是否存在产品故障。
- ◆ 以下情况不在我们的免费技术支持服务范围内,但我们将根据情况酌情处理:
  - 用户自行开发中遇到的软硬件问题;
  - 用户自行修改嵌入式操作系统遇到的问题;
  - 用户自己的应用程序遇到的问题;
  - 用户自行修改本公司提供的软件代码遇到的问题。

#### 8.1.2 保修服务

- ◆ 产品自出售之日起,在正常使用状况下为印刷电路板提供12个月的免费保修服务;
  - ▶ 以下情况不属于免费服务范围,英蓓特科技将酌情收取服务费用:
    - ◆ 无法提供产品有效购买凭证、产品识别标签撕毁或无法辨认,涂改标签或标签与实际产品不符;
    - 未按用户手册操作导致产品损坏的;
    - 因天灾 (水灾、火灾、地震、雷击、台风等) 或零件之自然耗损或遇不可抗拒力导致的产品外观及 功能损坏;
    - ◆ 因供电、磕碰、房屋漏水、动物、潮湿、杂 / 异物进入板内等原因导致的产品外观及功能损坏;
    - 用户擅自拆焊零件或修改而导致不良或授权非英蓓特科技认可的人员及机构进行产品的拆装、维修,变更产品出厂规格及配置或扩充非英蓓特科技公司销售或认可的配件及由此引致的产品外观及功能损坏;
    - 用户自行安装软件、系统或软件设定不当或由电脑病毒等造成的故障;
    - ◆ 非经授权渠道购得此产品者。
    - ◆ 非英蓓特科技对用户做出的超出保修服务范围的承诺(包括口头及书面等)由承诺方负责兑现, 英蓓特科技恕不承担任何责任;
- ◆ 若板卡需要维修,请联系技术支持服务部。

◆ 英蓓特科技公司对于未经本公司许可私自寄回的产品不承担任何责任。



# 第9章 联系方式

- ♦ 电话: +86-755-33190846/33190847/33190848
- ◆ 邮箱:
  - 技术支持: <u>support@embest-tech.com</u>
  - 销售: <u>chinasales@embest-tech.com</u>
- ◆ 传真: +86-755-25616057
- ◆ 网站: <u>http://www.embest-tech.cn</u>
- ◆ 地址:深圳市南山区留仙大道 4093 号南山云谷创新产业园山水楼 4 楼 B