



Ingenic SDK 编译体系说明

文档历史:

版本	作者	注释
1.0	王秋伟	

目录

- 1 简介3
 - 1.1 目录结构.....3
 - 1.2 编译过程.....3
- 2 编译系统说明4
 - 2.1 sdk/根目录的编译系统.....4
 - 2.2 source 目录的编译系统5
 - 2.3 examples 目录的编译系统6

1 简介

本文档将介绍如何编译 sdk 以及讲解 sdk 的编译系统设计思路。

1.1 目录结构

进入 sdk 的根目录，执行 ls 命令查看目录结构，如下图：

```
qwwang@pc:~/work/ingenic/mpos/sdk$ ls
config.mk  configs  documents  examples  firmware  include  lib  Makefile  source
```

各个目录说明：

configs: 存放 sdk 的配置文件，例如 ms800 平台的配置文件是 ms800_sdk_defconfig

documents: 存放 sdk 的 API 说明文档和第三方库的说明文档

examples: 存放 sdk 各个模块 API 的使用 demo 程序

firmware: 存放 sdk 某些模块 API 依赖的固件，根据配置文件决定是否安装到文件系统

include: 存放 sdk 各个模块 API 的头文件，供应用程序开发使用

lib: 存放各个 sdk 配置文件编译出来的动态库，全局编译时会安装到文件系统

source: 存放 sdk 各个模块 API 的源码等，目前对客户不开放

config.mk 和 Makefile: 用于编译 sdk

1.2 编译过程

1) 配置，以 ms800 平台为例：make ms800_sdk_defconfig, 相当于把 configs/ms800_sdk_defconfig 复制到 sdk 的根目录 .config。如果 sdk 的根目录 .config 不存在直接编译，如下图：

```
qwwang@pc:~/work/ingenic/mpos/sdk$ make
config.mk:40: ***
config.mk:41: *** Configuration file "/home/qwwang/work/ingenic/mpos/sdk/.config" not found !!!
config.mk:42: *** Using defconfig file "/home/qwwang/work/ingenic/mpos/sdk/configs/ilock_sdk_defconfig"
config.mk:43: *** You can select a new configuration file by run "make xxxconfig"
config.mk:44: ***
make -sC /home/qwwang/work/ingenic/mpos/sdk/source all
```

当 sdk/.config 不存在时，打印出警告信息并且默认使用 ilock_sdk_defconfig 这个配置，所以在 sdk/.config 不存在或修改了 configs/下的配置文件时，都需要执行 make xxxconfig。

2) 编译：make -j4, 顺利完成将把 examples/测试 demo、lib/\$(TARGET_DEVICE)/的动态库和 firmware/下需要的固件，输出到 sdk/out/下的相应目录。

2 编译系统说明

sdk 的编译系统包含三部分，分别是 sdk/根目录、examples/目录以及 source/目录，这三个地方都包含各自的 config.mk 和 Makefile 文件，config.mk 都是在 Makefile 的开头包含进来。

2.1 sdk/根目录的编译系统

控制 sdk 编译的整体流程，包括：

1) 配置 sdk, 如下图:

```

42 #
43 # SDK config
44 #
45 %config:
46 > $(call sdk_config,$@)
47
48 defconfig:
49 > @cp $(TOPDIR)/configs/$(SDK_DEFCONFIG) $(TOPDIR)/.config
50 > @echo "#"
51 > @echo "# $(SDK_DEFCONFIG) is written to .config"
52 > @echo "#"
53
54 ilock_sdk_defconfig:
55 > $(call sdk_config,$@)
56
57 ms800_sdk_defconfig:
58 > $(call sdk_config,$@)
--

```

2) 判断 source 是否要进入 source/目录进行编译，创建 sdk 的输出目录，输出 firmware 和动态库存到输出目录在，如下图:

```

65 #
66 # Targets
67 #
68 libingenic:
69 ifeq ($(SOURCE_EXIST), exist)
70 > $(MAKE) -sC $(TOPDIR)/source all
71 endif
72 > @test -e $(SYSTEM_OUTDIR)/usr || mkdir -p $(SYSTEM_OUTDIR)/usr
73 > @test -e $(SYSTEM_OUTDIR)/usr/lib || mkdir -p $(SYSTEM_OUTDIR)/usr/lib
74 > @test -e $(SYSTEM_OUTDIR)/usr/firmware || mkdir -p $(SYSTEM_OUTDIR)/usr/firmware
75 > @rm -rf $(SYSTEM_OUTDIR)/usr/lib/*
76 > @rm -rf $(SYSTEM_OUTDIR)/usr/firmware/*
77 > @cp -af $(TOPDIR)/lib/$(CONFIG_TARGET_DEVICE)/*.so* $(SYSTEM_OUTDIR)/usr/lib
78 ifeq ($(CONFIG_EI_FACE_RECOGNIZE), y)
79 > @cp -af $(TOPDIR)/firmware/ei_face $(SYSTEM_OUTDIR)/usr/firmware/
80 endif
81 ifeq ($(CONFIG_FRMAEBUFFER_MANAGER), y)
82 > @cp -af $(TOPDIR)/firmware/freetype $(SYSTEM_OUTDIR)/usr/firmware/
83 endif
--

```

3) 进入 examples/目录编译测试 demo，如下图:

```

91 #
92 # Examples
93 #
94 examples: libingenic
95 > @rm -rf $(OUTDIR)/examples/*
96 > @$(MAKE) -sC examples all

```

4) 编译清理工作，注意 make distclean 才会删除 sdk/.config

2.2 source 目录的编译系统

这个目录下的 config.mk 和 Makefile 主要的工作是编译生成 libingenic.so 和输出 libingenic.so 及其依赖的动态库到 sdk/lib/\$(TARGET_DEVICE)/，流程如下：

1) 根据 sdk/.config 决定将哪些模块编译进 libingenic.so，目标的生成办法：

```

391 #
392 # Targets
393 #
394 $(TARGET): $(OBJS) $(LIBS)
395 > @rm -rf $(LIBS_TARGET_OUTDIR)/
396 > $(QUIET_LINK)$(LINK_OBJS) $(LDFLAGS) $(LDSHFLAGS) $(OBJS) $(LIBS) -o $(LIBS_TARGET_OUTDIR)/$@
397 > @$(STRIP) $(LIBS_TARGET_OUTDIR)/$@
398 > @echo -e '\n SDK: ${shell basename $(LIBS_TARGET_OUTDIR)}/$@ is ready\n'

```

2) 调用 scripts/libporter.sh 将 libingenic.so 依赖的其他动态库拷贝到 sdk/lib/\$(TARGET_DEVICE)/，如下图：

```

400 #
401 # Libporter
402 #
403 libporter: $(TARGET)
404 > @scripts/libporter.sh $(CONFIG_TARGET_DEVICE) $(TOPDIR) $(LDFLAGS)

```

说明：传给 scripts/libporter.sh 的参数有三个：

CONFIG_TARGET_DEVICE: 在 sdk 的配置文件中定义，如 ms800_sdk_defconfig，这参数的值就是“ms800”，跟在编译环境时执行 source build/envsetup.sh; lunch 得到的 TARGET_DEVICE 环境变量是一样的值。

TOPDIR: sdk 的根目录

LDFLAGS: 在 source/config.mk 定义

3) 更新固件到 sdk/firmware/，如下图：

```

408 # Firmware
409 firmware: $(TARGET)
410 ifeq ($(CONFIG_EI_FACE_RECOGNIZE), y)
411 > @rm -rf $(FIRMWARE_OUTDIR)/ei_face
412 > @cp -af face_recognize/ei/firmware $(FIRMWARE_OUTDIR)/ei_face
413 endif
414
415 #
416 #
417 # Freetype
418 ifeq ($(CONFIG_FRMAEBUFFER_MANAGER), y)
419 > rm -rf $(FIRMWARE_OUTDIR)/freetype
420 > cp -af lib/freetype/fontlib $(FIRMWARE_OUTDIR)/freetype
421 endif

```

2.3 examples 目录的编译系统

这个目录的 config.mk 和 Makefile 目的就是根据 sdk/.config 编译某些模块 API 的示例 demo，并输出到 sdk/out/examples/ 目录下，编译过程连接的动态库目录是 sdk/lib/\$(TARGET_DEVICE)/，如下图：

config.mk

```
55 LIBS_DIR := $(TOPDIR)/lib/$(CONFIG_TARGET_DEVICE)
56 $(if $(LIBS_DIR),,$(error librarys output directory "$(LIBS_DIR)" does not exist))
```

Makefile

```
21 LINK_LIBS := -L$(LIBS_DIR) -l$(TARGET_NAME)
```

编译某个模块 API 的示例 demo 的方法：


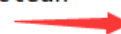




Makefile

```
23 define build_example
24 > $(QUIET_LINK)$(LINK_OBJS) $(LINK_LIBS) -o $(EXAMPLES_OUTDIR)/$(2) $(1) $(LD_FLAGS)
25 endef
```

config.mk

```
58 EXAMPLES_OUTDIR := $(OUTDIR)/examples
59 $(shell [ -d $(EXAMPLES_OUTDIR) ] || mkdir -p $(EXAMPLES_OUTDIR))
60 $(if $(EXAMPLES_OUTDIR),,$(error output directory "$(EXAMPLES_OUTDIR)" does not exist))
```

以编译 efuse 的示例 demo 讲解编译过程：

```
226 # EFUSE
227 #
228 ifeq ($(CONFIG_EFUSE_MANAGER), y)
229 EXAMPLE_EFUSE := test_efuse  目标名
230 EXAMPLE_EFUSE_CLEAN := test_efuse_clean
231 EXAMPLE_EFUSE_OBJ := efuse/main.o  目标的依赖源文件
232 $(EXAMPLE_EFUSE): $(EXAMPLE_EFUSE_OBJ)  编译方法，调用到 build_example
233 > $(call build_example,$^,$@)
234 $(EXAMPLE_EFUSE_CLEAN):  目标清除方法
235 > $(call clean_example,$(EXAMPLE_EFUSE_OBJ),$(EXAMPLE_EFUSE))
236
237 BUILD_EXAMPLES += $(EXAMPLE_EFUSE)  添加一个编译项
238 CLEAN_EXAMPLES += $(EXAMPLE_EFUSE_CLEAN)  添加到一个清除项
239 endif
```

```
813 # For build
814 #
815 all: $(BUILD_EXAMPLES)
```

当在 sdk/examples/ 执行 make all 时，各个编译项展开，并调用 build_example 方法生成示例 demo。