旷明 XOS SDK 快速
上手指南

部门	
文档编号	
版 本 号	V0.0.4
作　　者	David G

更新记录

日期	更新人	版本	备注
2024/12/21	David G	V0.0.1	初稿
2025/05/23	David G	V0.0.2	添加模块编译命令和备注说明
2025/06/23	David G	V0.0.3	添加编译产物说明
2025/09/06	David G	V0.0.4	Review更新

目录

1、 引言

1.1 编写目的

本文用于指导用户如何快速上手 XOS SDK 开发环境，包括 SDK 代码获取、环境搭建、工程编译、配置和烧录等关键步骤，实现能基于此 SDK 快速实现定制化开发。

1.2 预期读者和阅读建议

本文档可提供给客户、研发人员、技术支持工程师和测试工程师使用。

1.3 缩略术语

词语	解释
SDK	Software Development Kit
XOS	旷明统一操作系统

1.4 参考资料

《旷明 XOS 开发环境搭建指南》
《旷明 XOS 烧录升级指南》

2、 获取 SDK

请从旷明官方合作渠道获取最新版本的 XOS SDK 开发包。

XOS SDK 一般以*.tag.gz 压缩包形式提供，类似：X-AIOS-abc-Vx.y.z.tar.gz，其中：

- abc 是区分不同产品类别，比如 ESL-价签；CAM-相机；
- x,y,z 区别不同的版本号，比如 x,yz 分别为 1,2,1 时，表示当前 SDK 版本号为 V1.2.1

具体 SDK 版本以实际获取的 SDK 版本为准。

3、 构建开发环境

SDK 开发环境基于 Ubuntu 主机开发，需要安装必要的开发工具及依赖库，再根据目标平台提供的编译工具链，就可以编译 XOS SDK。

根据如下检查并构建 SDK 的开发环境：

1、参考文档《旷明 XOS 开发环境搭建指南》搭建编译环境；

2、SDK 编译工具链：都是在 XOS SDK 中提供，工具链在 SDK 的目录：tools/toolchain 下，根据芯片不通过，工具链可能不同。

4、简介及配置

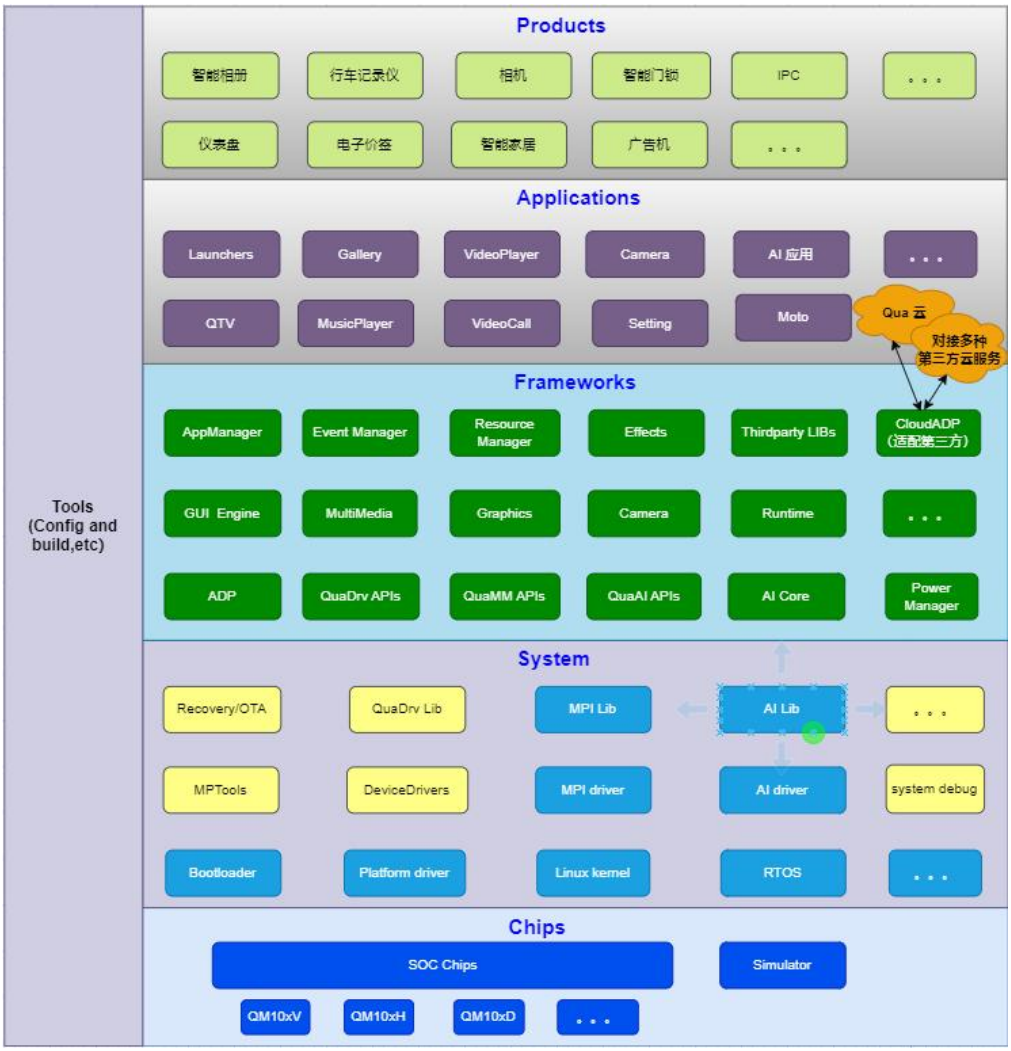
4.1 代码架构

4.1.1 目录结构

~/xos/$ tree -L 1

├── *base*	------	BSP，包括 uboot,kernel,rootfs....
├── *build*	------	XOS 编译架构目录
├── *core*	------	XOS APP,MultiMedia,FMk 等
├── *docs*	------	文档目录
├── *inc*	------	SDK .h 文件目录
├── *Kconfig -> build/kconf/Kconfig*	------	menuconfig 配置入口文件
├── *Makefile -> build/make/Makefile*	------	make 命令入口文件
├── *out*	------	编译生成的 image 目录
├── *product*	------	项目配置目录
└── *tools*	------	工具目录

4.1.2 系统框架



XOS 系统架构

4.1.3 系统存储

XOS 产品系统存储介质，一般是 SPI-Norflash、SPI-Nandflash、eMMC 的某一种，不同的介质推荐的文件系统不同。

下图是存储介质与推荐的文件系统对应关系

操作系统	存储介质	推荐文件系统
Linux	NOR	JFFS2
	NAND	UBIFS
	eMMC	EXT4

XOS 文件系统存储配置

4.2 代码配置

SDK 支持两种配置方法：

一、defconfig 文件配置

二、menuconfig 菜单配置

4.2.1 defconfig 修改

直接修改 product/xxx 目录中 product_xxx_defconfig 文件，完成配置修改。

下面以 sphoto 项目的配置为例，来说明修改配置。不同的 SDK，以实际项目的 project_xxx_defconfig 为准。

1. 配置命令：

$ make project_sphoto_defconfig ------- 指定 *sphoto* 项目配置文件

这句命令，会在 SDK 根目录生成.config 文件，后续 xos 编译会依赖.config。

2. 配置内容：

```
guzhihuan@build01:~/xos/xos-2/product$ cat sphoto/sphoto_defconfig
CONFIG_PRODUCT_sphoto=y
CONFIG_CHIPSET_mc331x=y
CONFIG_CHIPSET_mc3312=y
CONFIG_TOOLCHAIN_arm-linux-molchip=y
CONFIG_XOS_BUILD_SDK="linux_128"
CONFIG_XOS_USE_APP_ALARM=y
CONFIG_XOS_USE_APP_CALCULATOR=y
CONFIG_XOS_USE_APP_CALENDAR=y
CONFIG_XOS_USE_APP_GALLERY=y
CONFIG_XOS_USE_APP_SETTING=y
CONFIG_XOS_USE_APP_SPHOTO=y
CONFIG_XOS_FWK_GUIENG_LVGL9=y
CONFIG_XOS_USE_TINY_TTF=y
CONFIG_XOS_USE_GIF=y
CONFIG_XOS_USE_PNG=y
CONFIG_XOS_USE_TJPG=y
CONFIG_XOS_FWK_EFFECTS=y
CONFIG_XOS_FWK_PLAYER=y
CONFIG_XOS_EXT_FFMPEG=y
CONFIG_XOS_EXT_GIFLIB=y
CONFIG_XOS_EXT_LIBPNG=y
CONFIG_XOS_EXT_ZLIB=y
CONFIG_XOS_HW_QUAMMAPI=y
```

3. 根据实际项目需要，修改 project_sphoto_defconfig 中的宏选项，保存。然后，执行 Make project_sphoto_defconfig 让新配置生成新的.config.

4.2.2 menuconfig 配置

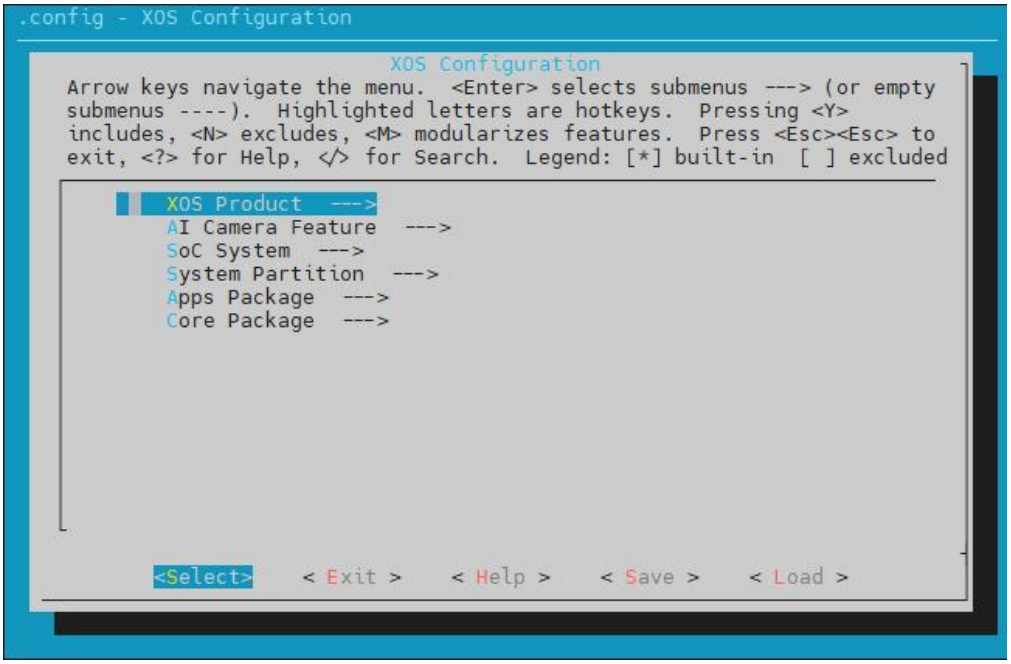
SDK 支持界面友好的 menuconfig 菜单配置方法，完成配置修改。

1. 配置命令：

$ make menuconfig -------- 进入 menuconfig 配置界面

2. 配置界面：

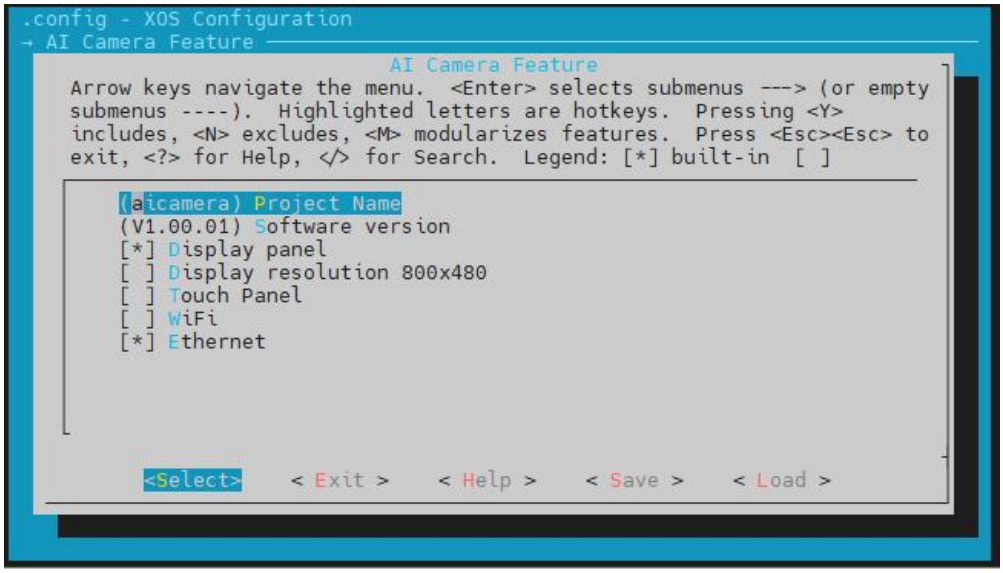
首界面


.config - XOS Configuration
XOS Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded
XOS Product --->
AI Camera Feature --->
SoC System --->
System Partition --->
Apps Package --->
Core Package --->
<Select> < Exit > < Help > < Save > < Load >

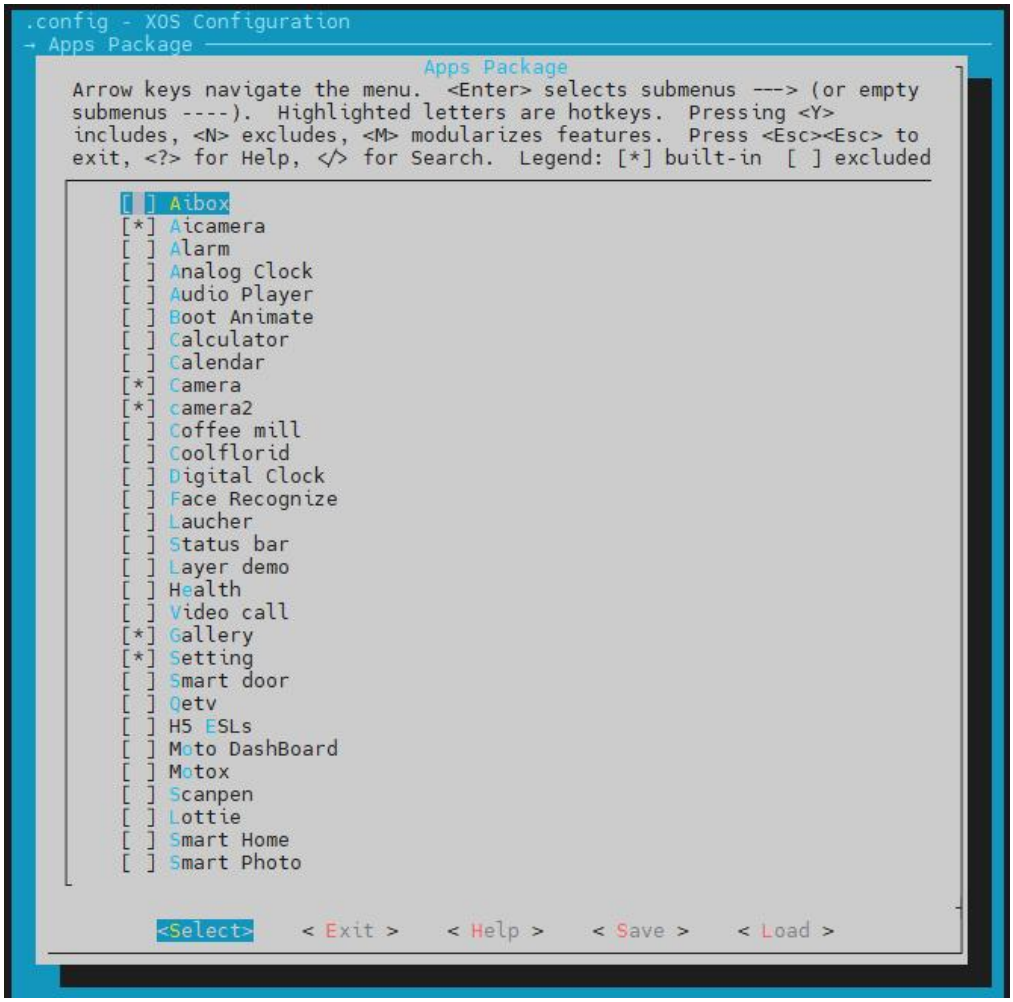
产品选择界面


.config - XOS Configuration
→ XOS Product
XOS Product
Use the arrow keys to navigate this window or press the hotkey of the item you wish to select followed by the <SPACE BAR>. Press <?> for additional information about this
(X) aicamera
( ) sphoto
( ) motox
( ) ipc
( ) gallery
( ) smarthome
↓(+)
<Select> < Help >

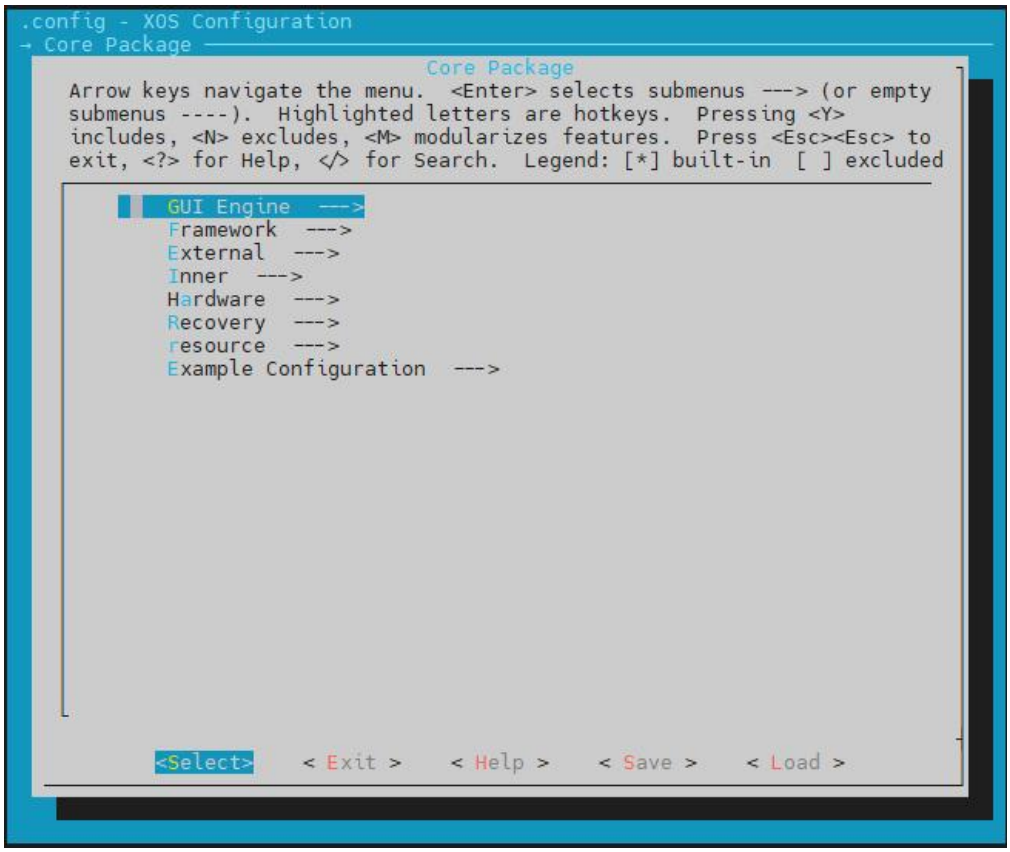
产品基本功能配置界面



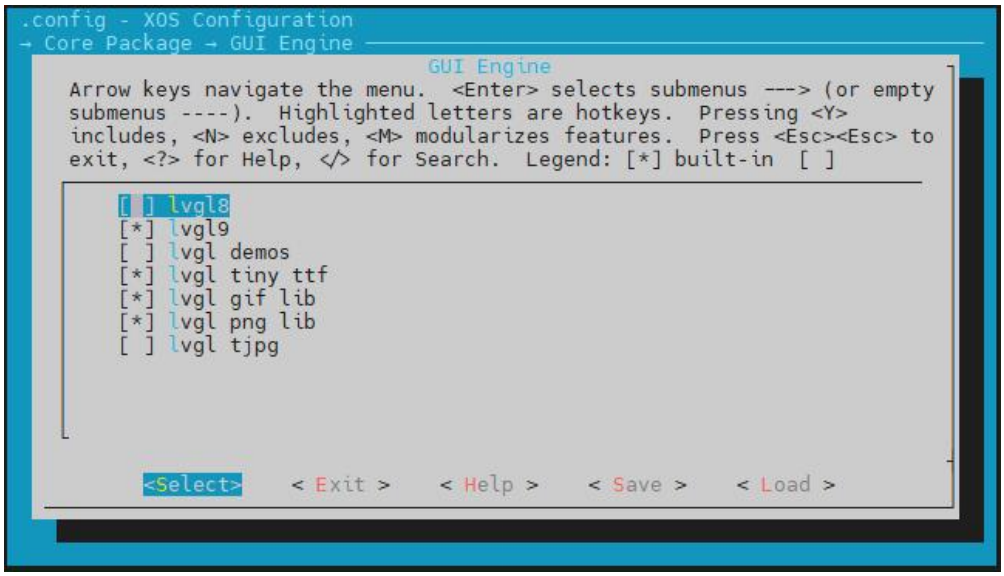
Apps Package ---> 应用选择



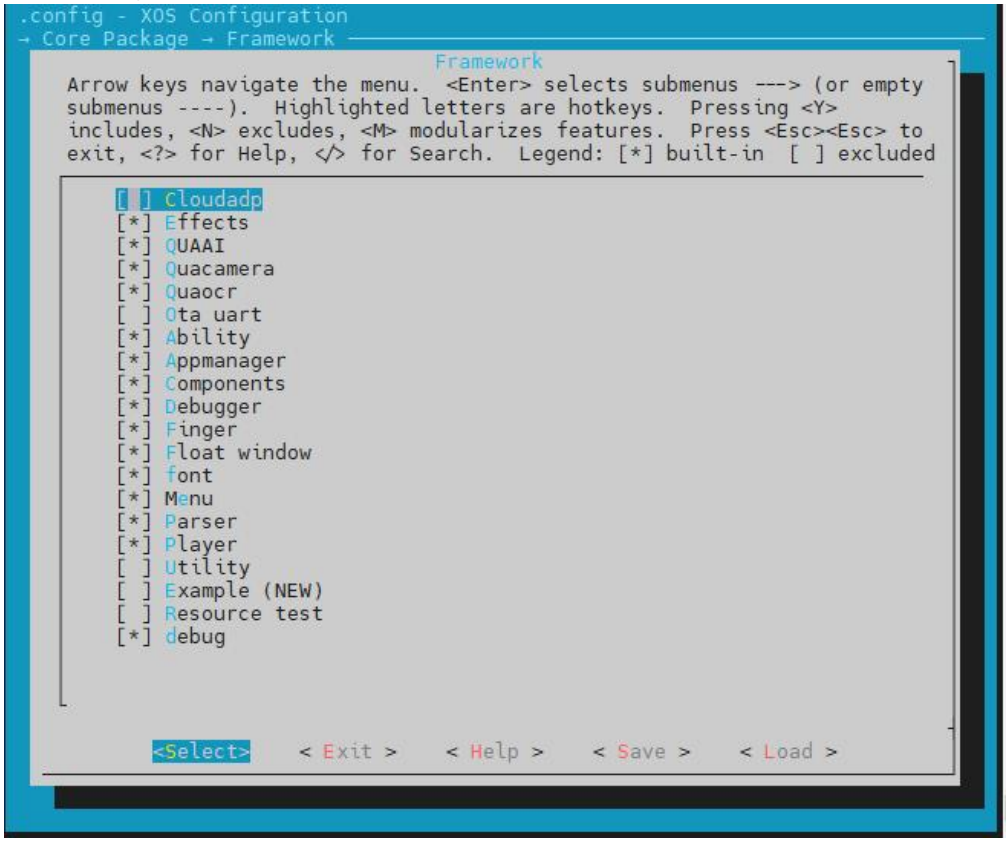
Core package ---> 核心软件配置



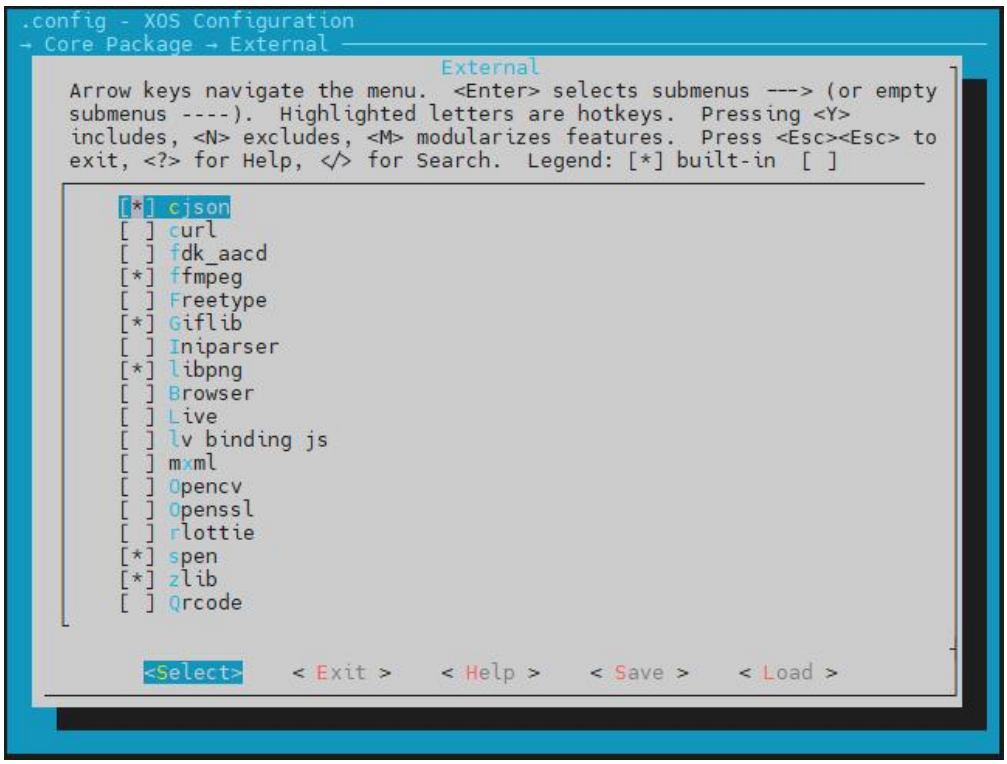
Core Package --> GUI Engine 配置



Core Package --> Framework 软件配置



Core Package --> External 软件配置



保存配置：

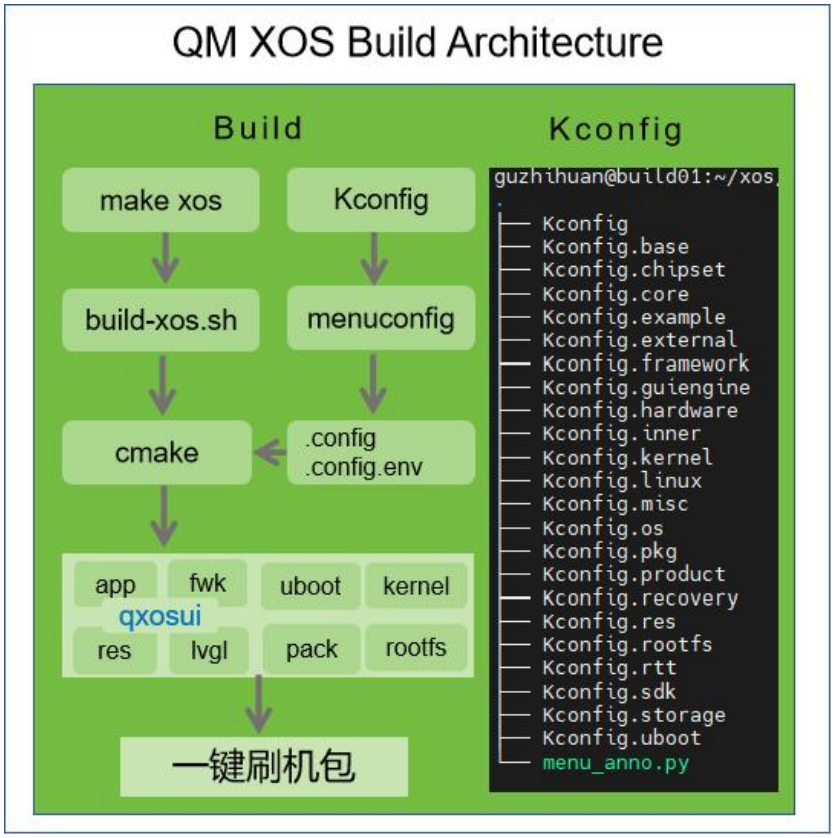
把所有想要的配置项配完后，save 保存到.config.

把.config 覆盖到原本的 project_xxx_defconfig，就可更新了 xxx 项目的配置。

5、编译

5.1 编译架构

SDK 采用旷明自研的 XOS 编译系统，架构图如下：



5.2 编译命令

5.2.1 完整编译

$ make distclean	-------	清除编译
$ make project_xxx_defconfig	-------	选择项目配置文件
$ make xos	-------	开始全编译

编译成功后，编译结果在：out/xxxxxxx_linux/qmimages 目录下，xxxxxx 是该 project 选用的芯片名字，比如 qm10xv，qm10xd，qm10xh 等

*qmimages 目录下，一般包括可烧录的 boot、刷机包、编程器烧录固件、以及用于调试 backup***.tar.gz 文件备份包。*

关于 boot 烧录、刷机、量产时编程器烧录的使用方法，参考《旷明 XOS 烧录升级指南》等专项文档。

5.2.2 模块编译

以 sphoto 项目为例：

(1). 编译 uboot

编译命令：*$ make xos-uboot*

编译产物：*spl, uboot image*

(2). 编译 kernel

编译命令：*$ make xos-kernel*
编译产物：out/qm10xd_linux/image/zImage-dtb
(3). 编译 APP
编译命令：*$ make xos-qxosui*
编译产物：output/qxosui.elf
(4). 打包文件系统和升级包
编译命令：*$ make xos-package*
编译产物：out/qm10xd_linux/qmimages 刷机脚本，文件系统，OTA 升级包等

5.2.3 编译 APP 应用模拟器版本

$ make distclean	-------	清除编译
$ make project_xxx_defconfig	-------	选择项目配置文件
$ make xos-sim	-------	开始编译模拟器版本

模拟器编译结果在：out/simulator/xos/bin/qxosui
可在 X86 的 ubuntu 下直接运行，或用 gdb 加载运行，可用于快速开发 APP 应用。

Gdb 加载运行示例：

```
gdb ./out/simulator/xos/bin/qxosui   #加载
r   #运行
q #退出
```

补充说明：1、当 3.1 编译板子固件、和 3.2 编译模拟器交替执行时，先删除或重命名备份根目录下的 out 目录，再执行编译命令

5.2.4 其他命令

$ make menuconfig	-------	menuconfig 配置
$ make savedefconfig	-------	.config 配置生成 defconfig
$ make distclean	-------	清除编译
$ make xos -j <jobs>	-------	多线程编译,例如:make xos -j 16

备注：全编译完成后，才可以进行模块编译。

5.2.5 编译选项

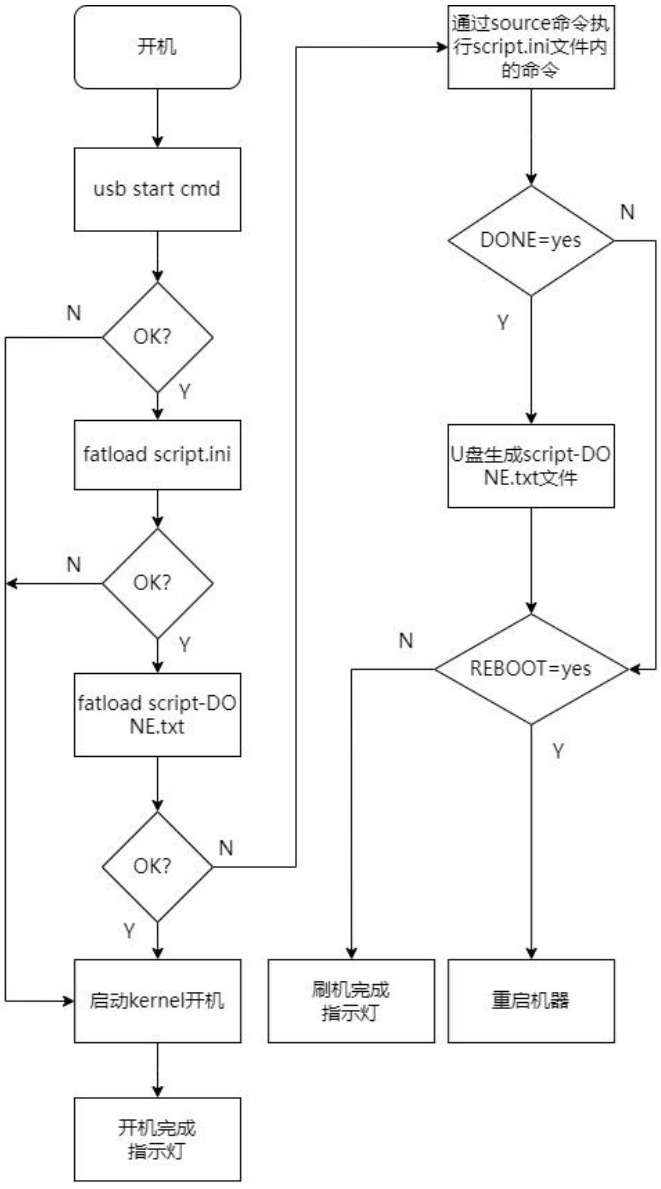
常用的编译选项（优化级别、调试选项等） ---- 待补充

6、烧录

6.1 开发烧录方法

编译好的固件，烧录 boot 及刷机方法参考文档《旷明 XOS 烧录升级指南》。

下图为 XOS U 盘（SD 卡）刷机实现流程图。



6.2 量产烧录方法

两种量产方式：

1、工具烧录 boot，再加刷机模式，需夹具； （适用于 spi nor、spi nand 和 emmc）

2、编程器烧录完整固件到 flash，然后再贴片； （适用于 spi nor 和 spi nand）

7、调试

待补充

7.1 调试方法

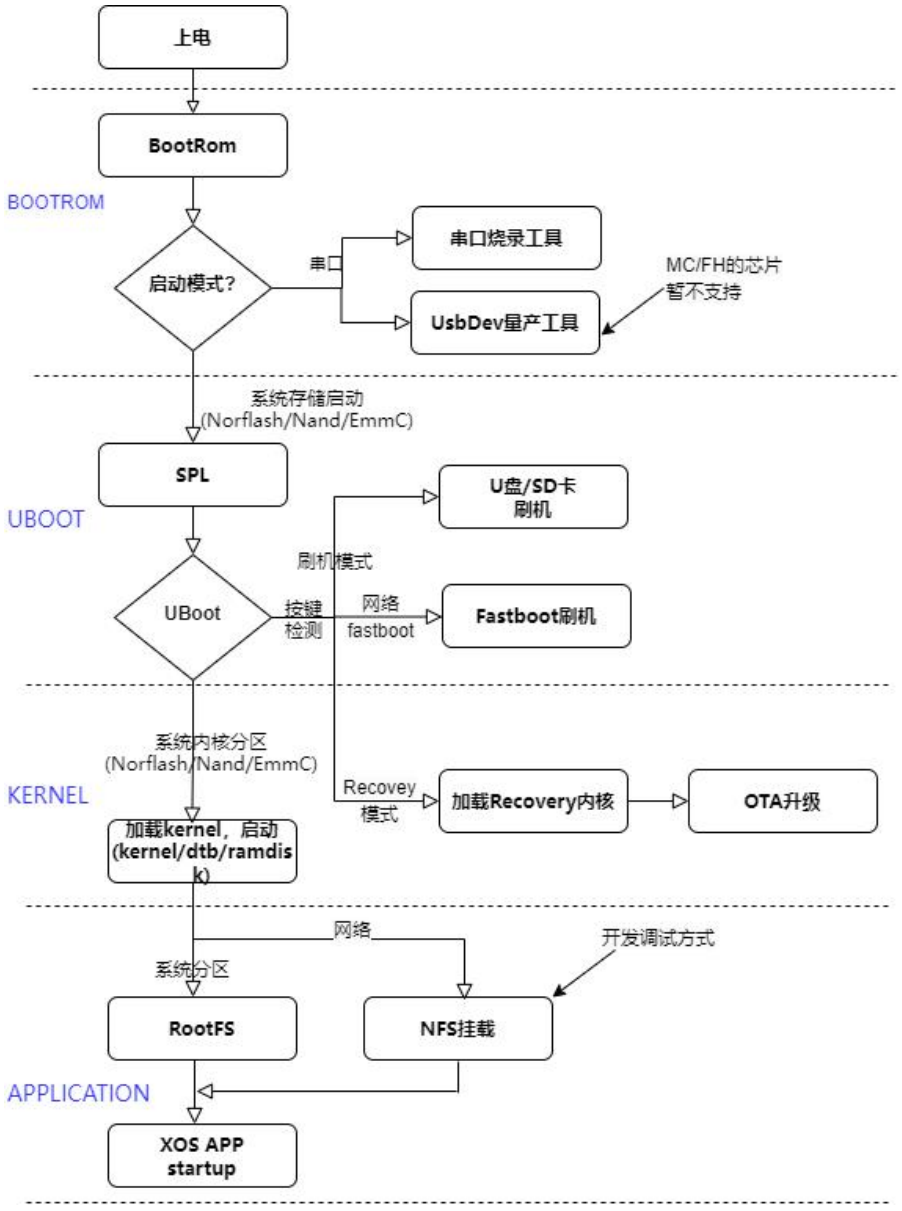
7.2 常见调试场景

8、开机流程

待补充

8.1 开机流程

XOS 开机流程图如下：



8.2 日志示例

下面给出不同阶段的开机 log 示例，仅供参考，不同产品 SDK 以实际运行为准。

1. Boot Rom 阶段 log:

```
ROM:       Use nand flash.
ROM:       Flash may not support, use default para.
ROM:       Download start, enter mode
ROM:       Xmodem start
ROM:       Please upload firmware using Xmodem
ROM:       Waiting...
ERR:       203
ERR:       203
ERR:       204
ERR:       204
ERR:       203
ROM:       Ok
RAM :      extra info not ready
....
```

2. Uboot 阶段 log

```
U-Boot 2016.11 (Dec 26 2024 - 20:28:19 +0800)
SPI:       ready
DRAM:    64 MiB
[ALGO]: 'Crypto' register name is aes-ecb
[ALGO]: 'Crypto' register name is aes-cbc
[ALGO]: 'Crypto' register name is aes-ctr
[ALGO]: 'Crypto' register name is aes-ofb
[ALGO]: 'Crypto' register name is aes-cfb
[ALGO]: 'Crypto' register name is des-ecb
[ALGO]: 'Crypto' register name is des-cbc
[ALGO]:NAND:    SPINAND: read id ! 0x0b, 0x11 0x00, 0x11!
SPINAND: XT26G10C is found.
MMC:
EMMC/MMC/SD controller initialization.
MMC FLASH INIT: No card on slot0!
No SD0 device found !
fhmci: 0
In:        serial
Out:       serial
Err:       serial
Net:       fh internal phy reset go...
  [cp_val_0] :: train val 2
[cp_val_1] :: train val 2
[val_100M_0] :: train val 11
[val_100M_1] :: train val 11
```

[val_10M_0] :: train val 15

[val_10M_1] :: train val 15

find phy driver [Generic PHY 0] : internal phy : inf mii

phy interface support 0

FH_EMAC_0

Warning: FH_EMAC_0 (eth0) using random MAC address - 76:0c:8f:25:9e:60

start boot_logo!

.....

3. Kernel 阶段日志

Starting kernel ...

[0.000000] Booting Linux on physical CPU 0x0

[0.000000] Linux version 4.9.129 (wuhaibin@build01) (gcc version 6.5.0 (arm_fhv512-uclibc-ng_20240109)) #1 Wed Dec 18 14:45:24 CST 2024

[0.000000] CPU: ARMv7 Processor [410fc075] revision 5 (ARMv7), cr=30c5387d

[0.000000] CPU: div instructions available: patching division code

[0.000000] CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache

[0.000000] Machine: QM102V

[0.000000] Memory policy: Data cache writeback

[0.000000] CPU: All CPU(s) started in SVC mode.

[0.000000] Built 1 zonelists in Zone order, mobility grouping on. Total pages: 11176

[0.000000] Kernel command line: console=ttyS0,115200 root=/dev/ram0 mem=44M

[0.000000] PID hash table entries: 256 (order: -2, 1024 bytes)

[0.000000] Dentry cache hash table entries: 8192 (order: 3, 32768 bytes)

[0.000000] Inode-cache hash table entries: 4096 (order: 2, 16384 bytes)

[0.000000] Memory: 35448K/45056K available (3527K kernel code, 254K rwdata, 1124K rodata, 3904K init, 157K bss, 9608K reserved, 0K cma-reserved)

[0.000000] Virtual kernel memory layout:

[0.000000] vector : 0xffff0000 - 0xffff1000 (4 kB)

[0.000000] fixmap : 0xffc00000 - 0xfff00000 (3072 kB)

[0.000000] vmalloc : 0xc3000000 - 0xff800000 (968 MB)

[0.000000] lowmem : 0xc0000000 - 0xc2c00000 (44 MB)

[0.000000] modules : 0xbf000000 - 0xc0000000 (16 MB)

[0.000000] .text : 0xc0008000 - 0xc0379f20 (3528 kB)

[0.000000] .init : 0xc04b4000 - 0xc0884000 (3904 kB)

[0.000000] .data : 0xc0884000 - 0xc08c3be0 (255 kB)

[0.000000] .bss : 0xc08c3be0 - 0xc08eb2f8 (158 kB)

.....

4. 启动 init.d

starting pid 105, tty '': '/etc/init.d/rcS'

[RCS]: /etc/init.d/S10udev

Starting udev: [OK]

[RCS]: /etc/init.d/S12bootanimation

[18.069629] fh_osal_init-22

[RCS]: /etc/init.d/S20init_rootfs

main, get img_width=240

main, get img_height=240

main, get intf_sync=51

main, get screen size [320,240]

get frame_rate=12

main, CHIP_NAME=qm10xv OS_NAME=linux

version: commit 6a7a43076a3b211a362f6094e4414820abeba099 build: 2024-12-12 19:09:59 running on qm10xv_linux

system init success

init_vb, vb_config=6,308224;2,3111936

[dsp] version: V1.0.1 0.0.0(build 000)(g8c7dd7c),build: 2024-10-10

[qua_thread_start 210] sched policy 0

[qua_thread_proc 115] bootanimation default priority 0 sched priority 0

bootanimation_thread_entry, open /usr/media/boot.h264 failed

....

[RCS]: /etc/init.d/S22config-para

check_cp_config done

[RCS]: /etc/init.d/S40network

[RCS]: /etc/init.d/S99autorun

******autorun script******

[21.002701]

[21.002701] [VENC Drv Version]:[VXC00_TAG_2024110709 47_1.0.1-7-g4091a30] [Build Time]:[Nov 11 2024, 23:11:54]

starting pid 401, tty '': '/sbin/inetd -f -e /etc/inetd.conf'

starting pid 402, tty '': '-/bin/sh'

BusyBox v1.26.2 (2024-11-14 14:11:55 CST) built-in shell (ash)

Enter 'help' for a list of built-in commands.

qm-linux# param_init