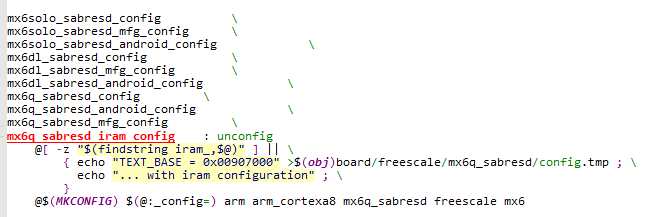
**u-boot 配置过程**

-- Makefile 命令： make mx6q\_sabresd\_config

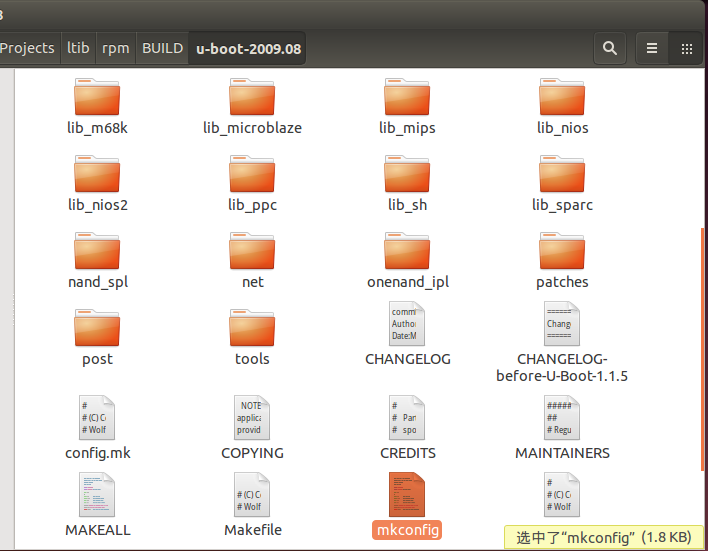


@$(MKCONFIG) $(@:\_config=) arm arm\_cortexa8 mx6q\_sabresd freescale mx6

mkconfig mx6q\_sabresd arm arm\_cortexa8 mx6q\_sabresd freescale mx6

MKCONFIG 定义

MKCONFIG := $(SRCTREE)/mkconfig //源文件下有一个mkconfig



-- mkconfig

#!/bin/sh -e

# Script to create header files and links to configure

# U-Boot for a specific board.

#

# Parameters: Target Architecture CPU Board [VENDOR] [SOC]

#

# (C) 2002-2006 DENX Software Engineering, Wolfgang Denk <wd@denx.de>

#

# mkconfig mx6q\_sabresd arm arm\_cortexa8 mx6q\_sabresd freescale mx6

# $0 1 2 3 4 5 6

APPEND=no # Default: Create new config file

BOARD\_NAME="" # Name to print in make output

# 分析传入参数

while [ $# -gt 0 ] ; do

case "$1" in

--) shift ; break ;;

-a) shift ; APPEND=yes ;;

-n) shift ; BOARD\_NAME="${1%%\_config}" ; shift ;;

\*) break ;;

esac

done

[ "${BOARD\_NAME}" ] || BOARD\_NAME="$1"

# BOARD\_NAME="mx6q\_sabresd"

# $# 代表传入参数的个数要求 >=4 或 <= 6

[ $# -lt 4 ] && exit 1

[ $# -gt 6 ] && exit 1

echo "Configuring for ${BOARD\_NAME} board..."

#

# Create link to architecture specific headers

#

# -- Makefile $SRCTREE == $OBJTREE

if [ "$SRCTREE" != "$OBJTREE" ] ; then

mkdir -p ${OBJTREE}/include

mkdir -p ${OBJTREE}/include2

cd ${OBJTREE}/include2

rm -f asm

ln -s ${SRCTREE}/include/asm-$2 asm

LNPREFIX="../../include2/asm/"

cd ../include

rm -rf asm-$2

rm -f asm

mkdir asm-$2

ln -s asm-$2 asm

else

cd ./include

rm -f asm

ln -s asm-$2 asm # 建立链接文件 ln -s asm-arm asm asm -> asm-arm

fi

rm -f asm-$2/arch # rm -f asm-arm/arch

# $6 mx6

if [ -z "$6" -o "$6" = "NULL" ] ; then

ln -s ${LNPREFIX}arch-$3 asm-$2/arch

else

ln -s ${LNPREFIX}arch-$6 asm-$2/arch # ln -s arch-mx6 asm-arm/arch

# asm-arm/arch -> arch-mx6

fi

if [ "$2" = "arm" ] ; then

rm -f asm-$2/proc

ln -s ${LNPREFIX}proc-armv asm-$2/proc # ln -s proc-armv asm-arm/proc

# asm-arm/proc -> proc-armv

fi

#

# Create include file for Make 生成配置文件

#

echo "ARCH = $2" > config.mk # > 新建一个文件， >> 内容追加

echo "CPU = $3" >> config.mk

echo "BOARD = $4" >> config.mk

# $5 freescale

[ "$5" ] && [ "$5" != "NULL" ] && echo "VENDOR = $5" >> config.mk

# $6 mx6

[ "$6" ] && [ "$6" != "NULL" ] && echo "SOC = $6" >> config.mk

# /u-boot-2009.08/include$ cat config.mk

# ARCH = arm

# CPU = arm\_cortexa8

# BOARD = mx6q\_sabresd

# VENDOR = freescale

# SOC = mx6

#

# Create board specific header file

#

if [ "$APPEND" = "yes" ] # Append to existing config file

then

echo >> config.h

else

> config.h # Create new config file # 创建 config.h

fi

echo "/\* Automatically generated - do not edit \*/" >>config.h

echo "#include <configs/$1.h>" >>config.h

echo "#include <asm/config.h>" >>config.h

exit 0

# /u-boot-2009.08/include$ cat config.h

# /\* Automatically generated - do not edit \*/

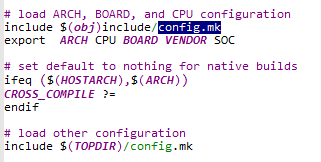
# #include <configs/mx6q\_sabresd.h>

# #include <asm/config.h>

因此 /u-boot-2009.08/include/configs/mx6q\_sabresd.h 为配置文件，如修改是否支持什么命令等

**u-boot 编译过程**

-- 继续分析Makefile



# /u-boot-2009.08/include$ cat config.mk

# ARCH = arm

# CPU = arm\_cortexa8

# BOARD = mx6q\_sabresd

# VENDOR = freescale

# SOC = mx6

#########################################################################

# U-Boot objects....order is important (i.e. start must be first)

OBJS = cpu/$(CPU)/start.o

LIBS = lib\_generic/libgeneric.a

LIBS += lib\_generic/lzma/liblzma.a

LIBS += lib\_generic/lzo/liblzo.a

LIBS += $(shell if [ -f board/$(VENDOR)/common/Makefile ]; then echo \

"board/$(VENDOR)/common/lib$(VENDOR).a"; fi)

LIBS += cpu/$(CPU)/lib$(CPU).a

# Always append ALL so that arch config.mk's can add custom ones

ALL += $(obj)u-boot.srec $(obj)u-boot.bin $(obj)System.map $(U\_BOOT\_NAND) $(U\_BOOT\_ONENAND)

all: $(ALL)

$(obj)u-boot.bin: $(obj)u-boot

$(OBJCOPY) ${OBJCFLAGS} -O binary $< $@

$(obj)u-boot: depend $(SUBDIRS) $(OBJS) $(LIBBOARD) $(LIBS) $(LDSCRIPT) $(obj)u-boot.lds

$(GEN\_UBOOT)

ifeq ($(CONFIG\_KALLSYMS),y)

smap=`$(call SYSTEM\_MAP,u-boot) | \

awk '$$2 ~ /[tTwW]/ {printf $$1 $$3 "\\\\000"}'` ; \

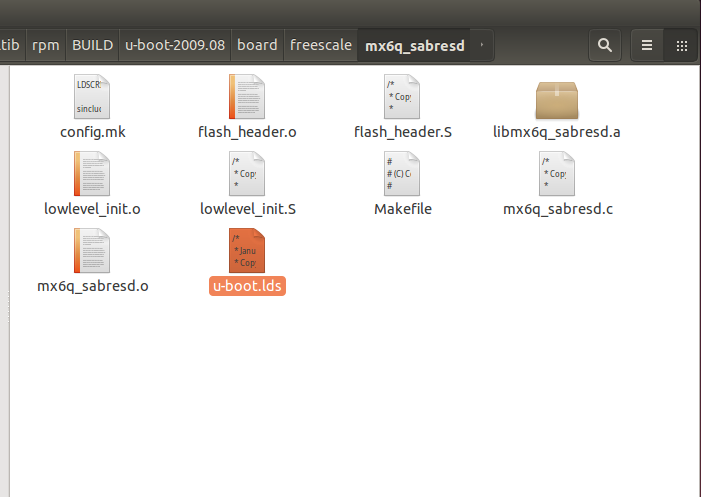
$(CC) $(CFLAGS) -DSYSTEM\_MAP="\"$${smap}\"" \

-c common/system\_map.c -o $(obj)common/system\_map.o

$(GEN\_UBOOT) $(obj)common/system\_map.o

endif

/u-boot-2009.08/board/freescale/mx6q\_sabresd/u-boot.lds（链接脚本）



SECTIONS

{

. = 0x00000000;

.text :

{

/\* WARNING - the following is hand-optimized to fit within \*/

/\* the sector layout of our flash chips! XXX FIXME XXX \*/

board/freescale/mx6q\_sabresd/flash\_header.o (.text.flasheader)

cpu/arm\_cortexa8/start.o

board/freescale/mx6q\_sabresd/libmx6q\_sabresd.a (.text)

. = .;

\_\_u\_boot\_cmd\_start = .;

.u\_boot\_cmd : { \*(.u\_boot\_cmd) }

\_\_u\_boot\_cmd\_end = .;

分析 Makefile 得出：

1) /u-boot-2009.08/cpu/arm\_cortexa8/start.S

2）链接地址:

1 /u-boot-2009.08/board/freescale/mx6q\_sabresd/u-boot.lds

2 . = 0x00000000; + 0x????????? 这里应该是 0x27800000 uboot 运行地址

/u-boot-2009.08/board/freescale/mx6q\_sabresd/config.mk TEXT\_BASE = 0x27800000

LDFLAG

/u-boot-2009.08$ cat config.mk

LDFLAGS += -Bstatic -T $(obj)u-boot.lds $(PLATFORM\_LDFLAGS)

ifneq ($(TEXT\_BASE),)

LDFLAGS += -Ttext $(TEXT\_BASE) uboot地址 0x27800000

Endif

u-boot 启动过程

第一阶段（功能类似） /u-boot-2009.08/cpu/arm\_cortexa8/start.S

硬件相关的初始化

a 设置cpu为管理模式（SVC）

b 关看门狗

c 屏蔽中断

d初始化 主要是SDRAM

e 设置栈

f 时钟 clock\_init

g 重定位 #ifndef CONFIG\_SKIP\_RELOCATE\_UBOOT 代码flash -> SDRAM

h 清 bss 段

i 调用c函数 \_start\_armboot: .word start\_armboot

第二阶段

Uboot的目标：引导/启动内核

1. 从 Flash 读出内核

能支持Flash的读写

Nor Flash 初始化 flash\_init ()

Nand Flash 初始化 nand\_init()

1. 启动内核

/\* main\_loop() can return to retry autoboot, if so just run it again. \*/

for (;;) {

main\_loop ();

}

这是主循环，等待输入命令

/u-boot-2009.08/common/main.c

void main\_loop (void){

s = getenv ("bootcmd"); 启动命令

run\_command (s, 0);

}

分析 bootcmd

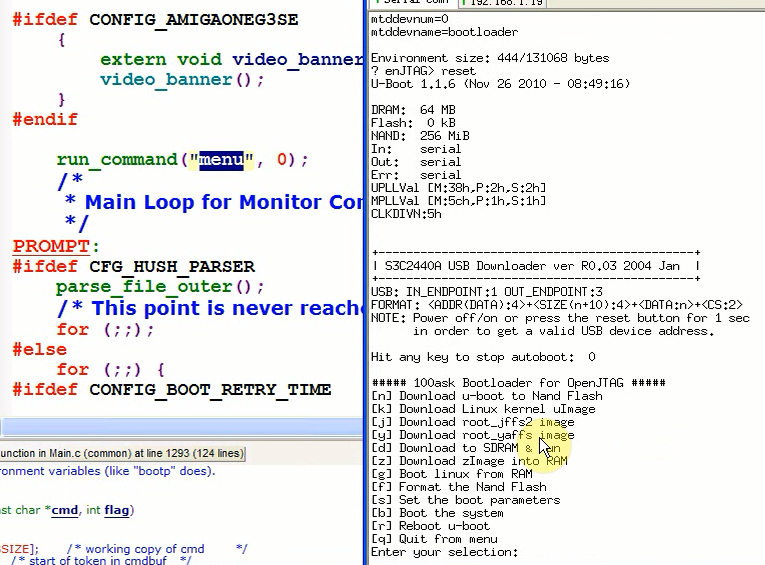
bootcmd = nand read.jffs2 0x30007FC0 kernel ; bootm 0x30007FC0

nand read.jffs2 0x30007FC0 kernel 是对应从Flash读出内核

bootm 0x30007FC0 是对应启动内核

u-boot 控制界面

s = getenv("menucmd");



len = readline (CONFIG\_SYS\_PROMPT); 读取串口的数据

strcpy (lastcommand, console\_buffer);

rc = run\_command (lastcommand, flag);