君正®

T41 软件资源编译指南

Date: 2023-07

北京君正集成电路股份有限公司
Ingenic Semiconductor Co., Ltd.

Trademarks and Permissions

Disclaimer

All the deliverables and data in this folder serve only as a reference for customer development. Please read through this disclaimer carefully before you use the deliverables and data in this folder. You may use the deliverables in this folder or not. However, by using the deliverables and data in this folder, you agree to accept all the content in this disclaimer unconditional and irrevocable. If you do not find the content in this disclaimer reasonable, you shall not use the deliverables and data in this folder.

The deliverables and data in this folder are provided "AS IS" without representations, guarantees or warranties of any kind (either express or implied). To the maximum extent permitted by law, Ingenic Semiconductor Co., Ltd (Ingenic) provides the deliverables and data in this folder without implied representations, guarantees or warranties, including but not limited to implied representations, guarantees and warranties of merchantability, non-infringement, or fitness for a particular purpose. Deviation of the data provided in this folder may exist under different test environments.

Ingenic takes no liability or legal responsibility for any design and development error, incident, negligence, infringement, and loss (including but not limited to any direct, indirect, consequential, or incidental loss) caused by the use of data in this folder. Users shall be responsible for all risks and consequences caused by the use of data in this folder.

北京君正集成电路股份有限公司
地址：北京市海淀区西北旺东路 10 号院东区 14 号楼君正大厦
电话:**(86-10)56345000**
传真:**(86-10)56345001**
Http: //www.ingenic.cn

前言

概述

本文为 Ingenic-T41 软件资源编译指南，方便使用者能快速在 T41 DEMB 板上搭建好开发资源环境。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
T41	V2.0

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

日期	版本	修订章节
2022-07	1.0	第一次正式版本发布
2022-08	1.1	**1 uboot 编译** uboot 编译涉及修改 **5 ISVP-SDK 编译** 5.1 SDK sample 涉及修改
2023-07	2.0	**1 uboot 编译** uboot 编译涉及修改

目录

1 Uboot 编译

1.1 Uboot 配置

1.1.1 Uboot 命令学习

- reset：重新启动嵌入式系统。
- printenv：打印当前系统环境变量。
- setenv：设置环境变量，格式：setenv name value，表示将 name 变量设置成 value 值。
- saveenv：保存环境变量到 flash 中。
- sleep：延迟执行，格式：sleep N，可以延迟 N 秒钟执行。
- run：执行环境变量中的命令，格式：run var，可以跟几个环境变量名。
- cp：在内存中复制数据块，格式：cp source target count，第一个参数是源地址，第二个参数是目的地址，第三个参数是复制数目。
- tftpboot：通过 tftp 协议下载文件到内存，格式 tftpboot target source。
- fatls：显示 SD 卡的文件，格式：fatls mmc 0 。
- bootm：可以引导启动存储在内存中的程序映像。格式：bootm addr1 addr2，第一个参数是程序映像的地址，第二个参数一般是 ramdisk 地址。

详细介绍可以在 uboot 命令行输入 help 查看。

1.1.2 Uboot 环境变量

板子上电启动，进入 uboot 命令行（读秒时按下 Enter 键）；通过 uboot 命令 printenv 可以查看 uboot 的环境变量：

```
   isvp_t41# printenv
   bootargs=console=ttyS1,115200n8 64M@0x0 rmem=64M@0x4000000 init=/linuxrc
rootfstype=squashfs root=/dev/mtdblock2 rw
mtdparts=jz_sfc:256k(boot),2560k(kernel),2048k(root),-(appfs)
   bootcmd=sf0 probe;sf0 read 0x80600000 0x40000 0x280000; bootm 0x80600000
```

```
bootdelay=1
ipaddr=193.169.4.151
serverip=193.169.4.2
```

参数介绍：

- bootargs：内核通过 bootargs 找到文件系统。
- console=ttyS1,115200n8：设置串口号为 ttyS1，波特率为 115200。
- mem=64M@0x0 rmem=64M@0x4000000 分配内存。
- init=/linuxrc：系统首先运行/linuxrc 文件。
- rootfstype=squashfs：使用压缩只读文件系统 squashfs。
- root=/dev/mtdblock2 rw：内核根文件从 mtdblock2 挂载。
- mtdparts=jz_sfc:256k(boot),2560k(kernel),2048k(root),-(appfs)：mtd 分区情况。
- bootcmd：启动命令。
- sf probe;sf read 0x80600000 0x40000 0x280000; bootm 0x80600000：从 nor 启。
- ipaddr 193.169.4.151：开发板 IP 地址。
- serverip=193.169.4.2：服务器 IP 地址。

1.1.3 更改 mtd 分区大小

MTD 分区大小，在 uboot 中以命令行参数的形式传入 kernel，配置文件在 include/configs/isvp_t41.h 中，对于 8M Flash，找到 CONFIG_SFC_NOR，参考平台定义其中：

```
mtdparts=jz_sfc:256k(boot),2560k(kernel),2048k(rootfs),-(appfs)
```

MTD 的分区情况，共 4 块分区，uboot 256k，kernel 2.5M，rootfs(squashfs)2M, -(appfs)具有自适应功能，当 Flash 为 8M 时，appfs 分区会自动分配成 3.25M；当 Flash 为 16M 时会自动分配成 11.25M。如果想修改，请保持总大小不变的情况下，按如上格式修改。一般 uboot、kernel、rootfs 大小固定不变，建议用户不要修改，可以调整 appfs 分区大小。

1.1.4 更改 rmem 内存大小

rmem 大小同样是在 include/configs/isvp_t41.h 中修改，参考平台定义如下：

```
#define BOOTARGS_COMMON "console=ttyS1,115200n8 mem=32M@0x0
rmem=32M@0x2000000
```

其中：

- mem=32M@0x0 rmem=32M@0x2000000 对 mem 大小的相关设置。
- mem=32M@0x0 从 0 地址开始给系统分配 32M 内存。
- rmem=32M@0x2000000 从 0x2000000（32M） 地址开始给 SDK 分配 32M。

T41 内嵌一块 64M/128M/256M 的内存，如要修改 rmem，请在总大小不变情况下按格式进行修改。

如：以 64M 内存为基准，假设 rmem 减小 4M，mem 应增大 4M，如果想减小 rmem，那么 mem 应该增大。所以 mem 大小为 36M，从 0 地址开始，rmem 应为 28M，地址应该从 36M 开始。

```
#define BOOTARGS_COMMON "console=ttyS1,115200n8 mem=36M@0x0
rmem=28M@0x2400000"
```

1.2 Uboot 编译

1. Uboot 编译流程：

u-boot 可单独编译，不依赖其他代码。T41 u-boot 的板机配置文件位于 include/configs/isvp_t41.h。

第一步： **$ make distclean** 清除旧配置。

第二步: **$ make isvp_t41x_xxx** 根据对应芯片类型编译 uboot，生成对应的 u-boot-with-spl.bin。

表 1-1 T41 芯片编译 SD 卡启动的 uboot

型号	编译命令	说明	主频	DDR
T41L	make isvp_t41l_msc0	编译 SD 卡启动的 uboot，针对 T41L 芯片(msc0 接口)	1104	600
T41N	make isvp_t41n_msc0	编译 SD 卡启动的 uboot，针对 T41N 芯片(msc0 接口)	1104	750
T41LQ	make isvp_t41lq_msc0	编译 SD 卡启动的 uboot，针对 T41LQ 芯片(msc0 接口)	1104	600
T41NQ	make isvp_t41nq_msc0	编译 SD 卡启动的 uboot，针对 T41NQ 芯片(msc0 接口)	1104	700
T41ZX	make isvp_t41zx_msc0	编译 SD 卡启动的 uboot，针对 T41ZX 芯片(msc0 接口)	1104	700
T41XQ	make isvp_t41xq_msc0	编译 SD 卡启动的 uboot，针对 T41XQ 芯片(msc0 接口)	1104	700
T41L	make isvp_t41l_msc0_lp	编译低性能/低功耗（备选方案）的 SD 卡启动的 uboot，针对 T41L 芯片(msc0 接口)，支持 5MP/30fps	1008	550
T41N	make isvp_t41n_msc0_lp	编译低性能/低功耗（备选方案）的 SD 卡启动的 uboot，针对 T41N 芯片(msc0 接口)，支持 5MP/30fps	1008	550
T41LQ	make isvp_t41lq_msc0_lp	编译低性能/低功耗（备选方案）的 SD 卡启动的 uboot，针对 T41LQ 芯片(msc0 接口)，支持 5MP/30fps	1008	550

T41NQ	make isvp_t41nq_msc0_lp	编译低性能/低功耗（备选方案）的 SD 卡启动的 uboot，针对 T41NQ 芯片(msc0 接口)，支持 5MP/30fps	1008	600
T41ZX	make isvp_t41zx_msc0_lp	编译低性能/低功耗（备选方案）的 SD 卡启动的 uboot，针对 T41ZX 芯片(msc0 接口)，支持 5MP/30fps	1008	550
T41XQ	make isvp_t41xq_msc0_lp	编译低性能/低功耗（备选方案）的 SD 卡启动的 uboot，针对 T41XQ 芯片(msc0 接口)，支持 5MP/30fps	1008	550
T41L	make isvp_t41l_msc1	编译 SD 卡启动的 uboot，针对 T41L 芯片(msc1 接口)	1104	600
T41N	make isvp_t41n_msc1	编译 SD 卡启动的 uboot，针对 T41N 芯片(msc1 接口)	1104	700
T41LQ	make isvp_t41lq_msc1	编译 SD 卡启动的 uboot，针对 T41LQ 芯片(msc1 接口)	1104	600
T41NQ	make isvp_t41nq_msc1	编译 SD 卡启动的 uboot，针对 T41NQ 芯片(msc1 接口)	1104	700
T41ZX	make isvp_t41zx_msc1	编译 SD 卡启动的 uboot，针对 T41ZX 芯片(msc1 接口)	1104	700
T41XQ	make isvp_t41xq_msc1	编译 SD 卡启动的 uboot，针对 T41XQ 芯片(msc1 接口)	1104	700
T41L	make isvp_t41l_msc1_lp	编译低性能/低功耗（备选方案）的 SD 卡启动的 uboot，针对 T41L 芯片(msc1 接口)，支持 5MP/30fps	1008	550
T41N	make isvp_t41n_msc1_lp	编译低性能/低功耗（备选方案）的 SD 卡启动的 uboot，针对 T41N 芯片(msc1 接口)，支持 5MP/30fps	1008	550
T41LQ	make isvp_t41lq_msc1_lp	编译低性能/低功耗（备选方案）的 SD 卡启动的 uboot，针对 T41LQ 芯片(msc1 接口)，支持 5MP/30fps	1008	550
T41NQ	make isvp_t41nq_msc1_lp	编译低性能/低功耗（备选方案）的 SD 卡启动的 uboot，针对 T41NQ 芯片(msc1 接口)，支持 5MP/30fps	1008	600
T41ZX	make isvp_t41zx_msc1_lp	编译低性能/低功耗（备选方案）的 SD 卡启动的 uboot，针对 T41ZX 芯片(msc1 接口)，支持 5MP/30fps	1008	550
T41XQ	make isvp_t41xq_msc1_lp	编译低性能/低功耗（备选方案）的 SD 卡启动的 uboot，针对 T41XQ 芯片(msc1 接口)，支持 5MP/30fps	1008	550

表 1-2　T41 芯片编译 nand flash 启动的 uboot

型号	编译命令	说明	主频	DDR
T41L	make	编译 nand flash 启动的 uboot，针对	1104	600

	isvp_t41l_sfc0_nand	T41L 芯片(sfc0 接口)		
T41N	make isvp_t41n_sfc0_nand	编译 nand flash 启动的 uboot，针对 T41N 芯片(sfc0 接口)	1104	750
T41LQ	make isvp_t41lq_sfc0_nand	编译 nand flash 启动的 uboot，针对 T41LQ 芯片(sfc0 接口)	1104	600
T41NQ	make isvp_t41nq_sfc0_nand	编译 nand flash 启动的 uboot，针对 T41NQ 芯片(sfc0 接口)	1104	700
T41ZX	make isvp_t41zx_sfc0_nand	编译 nand flash 启动的 uboot，针对 T41ZX 芯片(sfc0 接口)	1104	700
T41XQ	make isvp_t41xq_sfc0_nand	编译 nand flash 启动的 uboot，针对 T41XQ 芯片(sfc0 接口)	1104	700
T41L	make isvp_t41l_sfc0_nand_lp	编译低性能/低功耗（备选方案）的 nand flash 启动的 uboot，针对 T41L 芯片(sfc0 接口)，支持 5MP/30fps	1008	550
T41N	make isvp_t41n_sfc0_nand_lp	编译低性能/低功耗（备选方案）的 nand flash 启动的 uboot，针对 T41N 芯片(sfc0 接口)，支持 5MP/30fps	1008	550
T41LQ	make isvp_t41lq_sfc0_nand_lp	编译低性能/低功耗（备选方案）的 nand flash 启动的 uboot，针对 T41LQ 芯片(sfc0 接口)，支持 5MP/30fps	1008	550
T41NQ	make isvp_t41nq_sfc0_nand_lp	编译低性能/低功耗（备选方案）的 nand flash 启动的 uboot，针对 T41NQ 芯片(sfc0 接口)，支持 5MP/30fps	1008	550
T41ZX	make isvp_t41zx_sfc0_nand_lp	编译低性能/低功耗（备选方案）的 nand flash 启动的 uboot，针对 T41ZX 芯片(sfc0 接口)，支持 5MP/30fps	1008	550
T41XQ	make isvp_t41xq_sfc0_nand_lp	编译低性能/低功耗（备选方案）的 nand flash 启动的 uboot，针对 T41XQ 芯片(sfc0 接口)，支持 5MP/30fps	1008	550
T41L	make isvp_t41l_sfc1_nand	编译 nand flash 启动的 uboot，针对 T41L 芯片(sfc1 接口)	1104	600
T41N	make isvp_t41n_sfc1_nand	编译 nand flash 启动的 uboot，针对 T41N 芯片(sfc1 接口)	1104	750
T41LQ	make isvp_t41lq_sfc1_nand	编译 nand flash 启动的 uboot，针对 T41LQ 芯片(sfc1 接口)	1104	600
T41NQ	make isvp_t41nq_sfc1_nand	编译 nand flash 启动的 uboot，针对 T41NQ 芯片(sfc1 接口)	1104	700
T41ZX	make isvp_t41zx_sfc1_nand	编译 nand flash 启动的 uboot，针对 T41ZX 芯片(sfc1 接口)	1104	750
T41XQ	make	编译 nand flash 启动的 uboot，针对	1104	700

	isvp_t41xq_sfc1_nand	T41XQ 芯片(sfc1 接口)		
T41L	make isvp_t41l_sfc1_nand_lp	编译低性能/低功耗（备选方案）的 nand flash 启动的 uboot，针对 T41L 芯片(sfc1 接口)，支持 5MP/30fps	1008	550
T41N	make isvp_t41n_sfc1_nand_lp	编译低性能/低功耗（备选方案）的 nand flash 启动的 uboot，针对 T41N 芯片(sfc1 接口)，支持 5MP/30fps	1008	550
T41LQ	make isvp_t41lq_sfc1_nand_lp	编译低性能/低功耗（备选方案）的 nand flash 启动的 uboot，针对 T41LQ 芯片(sfc1 接口)，支持 5MP/30fps	1008	550
T41NQ	make isvp_t41nq_sfc1_nand_lp	编译低性能/低功耗（备选方案）的 nand flash 启动的 uboot，针对 T41NQ 芯片(sfc1 接口)，支持 5MP/30fps	1008	550
T41ZX	make isvp_t41zx_sfc1_nand_lp	编译低性能/低功耗（备选方案）的 nand flash 启动的 uboot，针对 T41ZX 芯片(sfc1 接口)，支持 5MP/30fps	1008	550
T41XQ	make isvp_t41xq_sfc1_nand_lp	编译低性能/低功耗（备选方案）的 nand flash 启动的 uboot，针对 T41XQ 芯片(sfc1 接口)，支持 5MP/30fps	1008	550

表 1-3　T41 芯片编译 nor flash 启动的 uboot

型号	编译命令	说明	主频	DDR
T41L	make isvp_t41l_sfc_nor	编译 nor flash 启动的 uboot，针对 T41L 芯片	1104	600
T41N	make isvp_t41n_sfc_nor	编译 nor flash 启动的 uboot，针对 T41N 芯片	1104	750
T41LQ	make isvp_t41lq_sfc_nor	编译 nor flash 启动的 uboot，针对 T41LQ 芯片	1104	600
T41NQ	make isvp_t41nq_sfc_nor	编译 nor flash 启动的 uboot，针对 T41NQ 芯片	1104	700
T41ZX	make isvp_t41zx_sfc_nor	编译 nor flash 启动的 uboot，针对 T41ZX 芯片	1104	700
T41XQ	make isvp_t41xq_sfc_nor	编译 nor flash 启动的 uboot，针对 T41XQ 芯片	1104	700
T41L	make isvp_t41l_sfc_nor_lp	编译低性能/低功耗（备选方案）的 nor flash 启动的 uboot，针对 T41L 芯片，支持 5MP/30fps	1008	550
T41N	make isvp_t41n_sfc_nor_lp	编译低性能/低功耗（备选方案）的 nor flash 启动的 uboot，针对 T41N	1008	550

		芯片，支持 5MP/30fps		
T41LQ	make isvp_t41lq_sfc_nor_lp	编译低性能/低功耗（备选方案）的 nor flash 启动的 uboot，针对 T41LQ 芯片，支持 5MP/30fps	1008	550
T41NQ	make isvp_t41nq_sfc_nor_lp	编译低性能/低功耗（备选方案）的 nor flash 启动的 uboot，针对 T41NQ 芯片，支持 5MP/30fps	1008	550
T41ZX	make isvp_t41zx_sfc_nor_lp	编译低性能/低功耗（备选方案）的 nor flash 启动的 uboot，针对 T41ZX 芯片，支持 5MP/30fps	1008	550
T41XQ	make isvp_t41xq_sfc_nor_lp	编译低性能/低功耗（备选方案）的 nor flash 启动的 uboot，针对 T41XQ 芯片，支持 5MP/30fps	1008	550

表 1-4　T41 芯片编译 uart 启动的 uboot

型号	编译命令	说明	主频	DDR
T41L	make isvp_t41l_uart_msc	编译 uart 启动 uboot，且 uboot 命令行支持 msc 相关命令，针对 T41L 芯片	1104	600
T41N	make isvp_t41n_uart_msc	编译 uart 启动 uboot，且 uboot 命令行支持 msc 相关命令，针对 T41N 芯片	1104	750
T41LQ	make isvp_t41lq_uart_msc	编译 uart 启动 uboot，且 uboot 命令行支持 msc 相关命令，针对 T41LQ 芯片	1104	600
T41NQ	make isvp_t41nq_uart_msc	编译 uart 启动 uboot，且 uboot 命令行支持 msc 相关命令，针对 T41NQ 芯片	1104	700
T41ZX	make isvp_t41zx_uart_msc	编译 uart 启动 uboot，且 uboot 命令行支持 msc 相关命令，针对 T41ZX 芯片	1104	700
T41XQ	make isvp_t41xq_uart_msc	编译 uart 启动 uboot，且 uboot 命令行支持 msc 相关命令，针对 T41XQ 芯片	1104	700
T41L	make isvp_t41l_uart_msc_lp	编译低性能/低功耗（备选方案）的 uart 启动 uboot，且 uboot 命令行支持 msc 相关命令，针对 T41L 芯片，支持 5MP/30fps	1008	550
T41N	make isvp_t41n_uart_msc_lp	编译低性能/低功耗（备选方案）的 uart 启动 uboot，且 uboot 命令行支持 msc 相关命令，针对 T41N 芯片，支持 5MP/30fps	1008	550
T41LQ	make	编译低性能/低功耗（备选方案）的	1008	550

	isvp_t41lq_uart_msc_lp	uart 启动 uboot，且 uboot 命令行支持 msc 相关命令，针对 T41LQ 芯片，支持 5MP/30fps		
T41NQ	make isvp_t41nq_uart_msc_lp	编译低性能/低功耗（备选方案）的 uart 启动 uboot，且 uboot 命令行支持 msc 相关命令，针对 T41NQ 芯片，支持 5MP/30fps	1008	550
T41ZX	make isvp_t41zx_uart_msc_lp	编译低性能/低功耗（备选方案）的 uart 启动 uboot，且 uboot 命令行支持 msc 相关命令，针对 T41ZX 芯片，支持 5MP/30fps	1008	550
T41XQ	make isvp_t41xq_uart_msc_lp	编译低性能/低功耗（备选方案）的 uart 启动 uboot，且 uboot 命令行支持 msc 相关命令，针对 T41XQ 芯片，支持 5MP/30fps	1008	550
T41L	make isvp_t41l_uart_sfc	编译 uart 启动 uboot，且 uboot 命令行支持 sfc 相关命令，针对 T41L 芯片	1104	600
T41N	make isvp_t41n_uart_sfc	编译 uart 启动 uboot，且 uboot 命令行支持 sfc 相关命令，针对 T41N 芯片	1104	700
T41LQ	make isvp_t41lq_uart_sfc	编译 uart 启动 uboot，且 uboot 命令行支持 sfc 相关命令，针对 T41LQ 芯片	1104	600
T41NQ	make isvp_t41nq_uart_sfc	编译 uart 启动 uboot，且 uboot 命令行支持 sfc 相关命令，针对 T41NQ 芯片	1104	700
T41ZX	make isvp_t41zx_uart_sfc	编译 uart 启动 uboot，且 uboot 命令行支持 sfc 相关命令，针对 T41ZX 芯片	1104	700
T41XQ	make isvp_t41xq_uart_sfc	编译 uart 启动 uboot，且 uboot 命令行支持 sfc 相关命令，针对 T41XQ 芯片	1104	700
T41L	make isvp_t41l_uart_sfc_lp	编译低性能/低功耗（备选方案）的 uart 启动 uboot，且 uboot 命令行支持 sfc 相关命令，针对 T41L 芯片，支持 5MP/30fps	1008	550
T41N	make isvp_t41n_uart_sfc_lp	编译低性能/低功耗（备选方案）的 uart 启动 uboot，且 uboot 命令行支持 sfc 相关命令，针对 T41N 芯片，支持 5MP/30fps	1008	550
T41LQ	make isvp_t41lq_uart_sfc_lp	编译低性能/低功耗（备选方案）的 uart 启动 uboot，且 uboot 命令行支持 sfc 相关命令，针对 T41LQ 芯片，	1008	550

		支持 5MP/30fps		
T41NQ	make isvp_t41nq_uart_sfc_lp	编译低性能/低功耗（备选方案）的 uart 启动 uboot，且 uboot 命令行支持 sfc 相关命令，针对 T41NQ 芯片，支持 5MP/30fps	1008	550
T41ZX	make isvp_t41zx_uart_sfc_lp	编译低性能/低功耗（备选方案）的 uart 启动 uboot，且 uboot 命令行支持 sfc 相关命令，针对 T41ZX 芯片，支持 5MP/30fps	1008	550
T41XQ	make isvp_t41xq_uart_sfc_lp	编译低性能/低功耗（备选方案）的 uart 启动 uboot，且 uboot 命令行支持 sfc 相关命令，针对 T41XQ 芯片，支持 5MP/30fps	1008	550

2. Uboot 配置文件中常见修改点

1) CONFIG_BOOTARGS

主要修改点是内核启动以后的内存配置，分区大小配置。


注意

mem 表示内核启动以后保留内存，rmem 表示预留给 SDK 的内存（包括 ISP 模块的内存），两者相加为芯片真实内存大小，具体大小可参考代码。

2) CONFIG_BOOTCOMMAND

配置 uboot 启动执行的命令。例如：norflash 启动模式下使用 sd 卡启动的命令，"sf probe;sf read 0x80600000 0x40000 0x280000; bootm 0x80600000" 改为 "mmc read 0x80600000 0x1800 0x3000; bootm 0x80600000"。

3) CONFIG_BOOTDELAY

配置 uboot 的等待时间。

4) uboot 中添加密码功能

修改配置文件，修改 isvp_t41.h 中添加如下内容：

#define CONFIG_AUTOBOOT_KEYED // 必配。

#define CONFIG_AUTOBOOT_STOP_STR “123456” //必配，uboot 设置的密码。

#define CONFIG_AUTOBOOT_PROMPT “Press xxx in %d second” // bootdelay, 选配，uboot 提示信息。

#define CONFIG_AUTOBOOT_DELAY_STR “linux” //选配，uboot 提示信息代码的具体实现在 common/main.c 中 abortboot_keyed(int bootdelay)；可以根据自己的需要具体改动。

5) SD 卡升级问题

在 isvp_t41.h 中添加 #define CONFIG_AUTO_UPDATE 定义。具体代码实现在 common/cmd_sdupdate.c 中。

注意

- LOAD_ADDR 表示把 SD 卡上的相应内存加载到内存的位置。程序中默认设置为 0x82000000，由于这个地址位于 uboot 的堆上，常见 uboot 的堆大小的设置在 isvp_t41.h 中 CONFIG_SYS_MALLOC_LEN 宏的配置；所以这个地址能够被使用的大小将受到堆大小的限制，还有 uboot 代码中 malloc 空间的限制。
- 需要读取较大文件的时候，可以适当增大 CONFIG_SYS_MALLOC_LEN）

6) 编译出的 uboot 大于限制的大小处理方法

uboot 代码中默认限制 spl 部分为 26Kbytes，uboot 部分为 214Kbytes；一共 240Kbytes 大小的限制。如果 uboot 编译生成的 u-boot-with-spl.bin 文件大于 240Kbytes，则无法启动。

解决方案一：

增加uboot 的限制;修改 isvp_t41.h 中CONFIG_SYS_MONITOR_LEN 宏的定义；同时修改 CONFIG_BOOTARGS 变量中 boot 分区的大小及以后分区的偏移地址。

解决方案二：

如果生成的 u-boot-with-spl.bin 超出 240Kbytes 较少可以采取压缩 uboot 的方式。在 isvp_t41.h 中 修改 #undef CONFIG_SPL_LZOP 为 #define CONFIG_SPL_LZOP；然后重新编译烧录的文件名 u-boot-lzo-with-spl.bin。

7) uboot 网络问题

默认的 isvp 配置是包含以太网部分的代码，如果产品在 uboot 阶段不需要 TFTP 下载或者 NFS 挂载，可以把以太网部分代码裁剪掉，以便减小 uboot。

具体操作：

打开 isvp_t41.h 配置文件，把#define CONFIG_CMD_NET 宏注掉即可。

2 Kernel 编译

2.1 编译流程

T41 支持两个版本的内核，分别为 kernel-3.10.14 版本和 kernel-4.4.94 版本。支持两个版本的原因是相同的配置文件两个版本的内核编译出来的内核文件大小不一样。

kernel 可单独编译，不依赖其他代码。以 isvp 板级编译为例，进入 kernel 源码目录。在 arch/mips/configs/ 文件夹下存放了内核的板级配置文件。T41 芯片板级根据 demo 板名称分为：

- Marmot 全功能开发板： isvp_marmot_defconfig
- Tomcat 38 板： isvp_tomcat_defconfig。

下列操作以 38 板为介绍：

第一步：**$make isvp_tomcat_defconfig** 使用相对配置好的板级文件

第二步：**$ make menuconfig** 根据需求选择性编译

第三步：**$ make uImage** 编译内核文件

如果报错，执行 **$ make distclean** ，然后从第一步重新开始。


注意

对于 kernel 之外的 ko 编译，依赖 kernel，且 kernel 必须先编译。每当 kernel 更改之后，可能会出现 ko 无法 insmod 的情况，或者可以 insmod 但会出现未知错误。这是因为 kernel 重新编译后，函数 Symbol 表有变化，需要重新编译 ko driver 以正确 link 函数。

内核默认的 config 留有一定余量，而实际产品往往需要进行内核裁减以释放更多的空间。

2.2 Kernel 裁剪选项及说明

1) CONFIG_NETWORK_FILESYSTEMS

网络文件系统，一般用来方便开发，但会占用较多空间，也可用 tftp 进行替代。若不支持 NFS，可以 Disable。

2) CONFIG_KALLSYMS &CONFIG_KALLSYMS_ALL

内核函数符号表，会占用较多空间，在 panic 时的函数栈可以显示出函数名。当内核稳定后，可以考虑 Disable 此功能，但建议完全稳定之前使能此功能。

3) 其他文件系统

一般文件系统会占用较多空间，开发者可根据需求对文件系统的选项进行裁减，比如 ext 文件系统等。

4) 和产品定义无关的模块

比如 USB，以太网，TF 卡等等。

注意

内核的深度裁减有一定的技术难度，开发者尽量在深入了解配置的情况下再进行深度裁减。

3 Driver 编译

3.1 T41 需要加载驱动

驱动名称	驱动介绍
tx-isp-t41.ko	ISP 驱动
sensor_xxxx_t41.ko	Sensor 驱动
avpu.ko	视频编码驱动
sinfo.ko	Sinfo 探测驱动
audio.ko	音频驱动

3.2 T41 驱动编译

3.2.1 T41 驱动编译注意事项

1. 编译驱动前应先编译 kernel。
2. 编译好 kernel 后优先编译 isp 驱动，在编译之前，进入 Makefile 指定好内核路径。
3. 编译完成 isp 驱动后，再编译 sensor 驱动，编译 sensor 驱动需要进入 Makefile 指定好 kernel 路径和 isp 驱动路径。
4. 其他驱动编译前只需要指定好 kernel 路径即可

4 应用程序编译

4.1 应用程序编译

4.1.1 编译注意

应用程序编译注意有以下几点：

1. ISVP 的 toolchain 包含了 glibc 和 uclibc，因此基于 glibc 或者 uclibc 的程序均可使用此 toolchan 进行编译。
   - glibc 程序编译方法：默认 link 的 libc 即为 glibc
   - uclibc 程序编译方法：C_FLAGS+=-muclibc CXX_FLAGS+=-muclibc，LD_FLAGS+=-muclibc
2. 关于 API 库的链接顺序：[IVS 库] [mxu 库] [libimp/libsysutils] [libalog]
3. 由于 libimp 中依赖 C++库，因此需要使用 mips-linux-gnu-g++进行链接，若使用 gcc 链接，需要手动添加 LD_FLAGS+=stdc++。
4. 如何优化 elf 文件的大小：
   - 编译等级选择 O2：C_FLAGS/CXX_FLAGS += -O2
   - DFLAG += -Wl,-gc-sections，不链接不必要的段。
   - elf 文件执行 mips-linux-gnu-strip，链接后删除不必要的段。
5. 若系统中有多个 elf 文件需要链接库文件，可使用动态链接的方式，若只有一个文件链接库文件，请使用静态链接的方式（注，libimp 相关功能在系统中只能存在一份实例）。在调试时选择动态链接的方式可以方便的进行 debug 及问题反馈。

4.1.2 运行应用程序

要运行编译好的应用程序，首先需要将其添加到目标机中，然后完成以下工作：

1. 将应用程序和需要的库文件（如果有）等添加到目标机的根文件系统相应的目录中。通常将应用程序放到 /bin 目录里，库文件放到 /lib 目录里，配置文件则放

到 /etc 目录里。

2. 制作包含新应用程序的根文件系统。

注意

如果执行应用程序，需要读写文件系统。请选择 squashfs、jffs2 文件系统。如果新添加的应用程序需要系统启动后自动运行，请在编写文件系统时，编辑 /etc/init.d/rcS 文件，添加需要启动的应用程序路径。

5 ISVP-SDK 编译

5.1 SDK sample 编译

进入 ISVP-SDK 路径下的：

ISVP-T41-x.x.x-202xxxxx/software/sdk/Ingenic-SDK-T41-x.x.x-202xxxxx/sdk/samples/libimp-samples

你会看到一个 Makefile 文件，执行 make 命令即可编译 SDK sample 文件，生成的 sample 位于当前路径下，可拷贝至开发板中直接使用。

编译之前需要先指定当前 sensor，打开 sample-common.h 文件，修改对应 sensor 信息后再进行 sample 编译。

5.2 SDK sample 介绍

Sample 程序位于/samples/libimp-samples，目录下是依赖 SDK 库的应用程序，包括录音放音、回应消除等应用程序。用户可以参考提供的 sample 程序，编写自己相应工程代码。

完成编译后，拷贝可执行程序到开发板上测试即可。详细介绍请参考下表：

应用文件	功能	执行命令	执行结果
sample-Framesource.c	保存对应格式的图片	./sample-Framesource	会在 /tmp 下生成对应格式的图片
sample-Encoder-h265-ivpu-jpeg.c	保存 H265 码流和抓取图片	./sample-Encoder-h265-ivpu-jpeg	会在/tmp 下生成 h265 码流和 Jpeg 图片
sample-Encoder-jpeg.c	把 NV12 图片编码成 jpeg	./sample-Encoder-jpeg	把 NV12 图片编码成 jpeg 图片，并保存在/tmp 下
sample-Encoder-video.c	获取一段对应格式的视频流	./sample-Encoder-video	会在 /tmp 下产生对应编码格式的视频
sample-Encoder-video-direct.c	主码流开启直通功能，并获取直通后的 h265 码	./sample-Encoder-video-direct	会在 /tmp 下产生 h265 的视频

	流		
sample-Encoder-jpeg-ivpu.c	抓取 helix jpeg 图片	./sample-Encoder-jpeg-ivpu	会在/tmp 下生成 jpeg 图片
sample-Encoder-video-IVS-move.c	移动检测	./sample-Encoder-video-IVS-move	会在 /tmp 对应格式的视屏带移动检测
sample-Snap-YUV.c	抓取 yuv 图片	./sample-Snap-YUV	会在 /tmp 下生成 yuv 文件
sample-Snap-Raw.c	抓取 raw 图片	./sample-Snap-Raw	会在 /tmp 下生成 raw 文件
sample-OSD.c	在视频上叠加图片	./sample-OSD	会在 /tmp 下产生抓取的一张带时间戳的图片
sample-ISP-flip.c	图像的镜像翻转	./sample-ISP-flip	会在 /tmp 下生成镜像翻转后的视频
sample-Setfps.c	设置帧率	./sample-Setfps	设置帧率
sample-GetFrameEx.c	多进程获取 YUV 数据,直通下不可使用	./sample-GetFrameEx	会在/tmp 下生成 nv12 图片
sample-AutoZoom.c	AF 功能种实现某一部分图片放大	./sample-AutoZoom	会在/tmp 下生成分辨率变化的使用
sample-Change-Resolution.c	改变分辨率流程	./sample-Change-Resolution.	会在/tmp 生成多种分辨率 h265 视频