君正®
T41 SDK 安装及使用指南

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 customerdevelopment. 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.com.cn

前言

概述

本文为 Ingenic T41 SDK 安装及使用指南，方便使用者能快速在 T41 DEMB 板上搭建好 SDK 运行环境。

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 SDK 安装及使用指南	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	**2 安装升级 T41DEMO 开发环境** 2.2 uboot 启动涉及新增 2.3 系统烧录涉及修改
2023-07	2.0	**1 首次安装 SDK** 1.6 如何安装 toolchain

目录

1 首次安装 SDK

如果您已安装过 SDK，可以直接参看 2 安装、升级 T41 DEMO 板开发环境。

1.1 T41 SDK 包位置

在发布的百度云链接或 ftp 服务器目录下，您可以看到一个 ISVP-T41-x.x.x-202xxxxx.7z 的文件，该文件就是 T41 的软件开发包。

1.2 解压缩 SDK 包

在 linux 服务器上（或者一台装有 linux 的 PC 上，主流的 linux 发行版本均可以，建议是 64bit），使用命令：$ 7z x ISVP-T41-x.x.x-202xxxxx.7z，解压缩该文件,可以得到一个 ISVP-T41-x.x.x-202xxxxx 目录。

1.3 为什么需要搭建开发环境

由于嵌入式单板的资源有限，不能在单板上运行开发和调试工具，通常需要交叉编译调试的方式进行开发和调试，即“宿主机＋目标机”的形式。宿主机和目标机一般采用串口连接显示交互信息，网口连接传输文件。

但宿主机和目标机的处理器一般不相同。宿主机需要建立适合于目标机的交叉编译环境。程序在宿主机上经过“编译－链接－定位”得到可执行文件。通过一定的方法将可执行文件烧写到目标机中，然后在目标机上运行。

1.4 安装 Linux 服务器

建议选择常用的 Linux 发行版，便于寻找各类技术资源。例如：

RedHat 较新的发行版如 RedHat Fedora Core 系列和 RedHat Enterprise Linux、RedHat 3.4.4-2、RedHat 较老的发行版如 RedHat 9.0 等。

推荐使用较新版本 64 位，以方便获取各类资源，如 Fedora Core 系列、Ubuntu12

版本以上。

1.5 交叉工具链

Toolchain 即交叉编译工具链，是 Linux Host 机上用来编译和调试嵌入式设备程序的一系列工具的集合。ISVP 中的 Toolchain 版本信息如下：

1) gcc 版本：7.2.0
2) libc 版本：
   - glibc 版本：2.29
   - uclibc 版本：0.9.33.2

1.6 如何安装 Toolchain

第一步：

根据 Host 机 CPU 位宽选择 mips-gcc720-glibc229-r5.1.9.tar.bz2 进行解压。例如：

```
$ tar -jxvf mips-gcc720-glibc229-r5.1.9.tar.bz2
```

第二步：

通过 export PATH=xxxx:$PATH 命令，将 toolchain 下的 bin 目录添加到 PATH 环境变量中或者在~/.bashrc 中加上下面一句永久改变。

```
$ vim ~/.bashrc
$ export PATH=/opt/mips-gcc720-glibc229-r5.1.9/bin:$PATH
```

第三步：

测试 toolchain 可执行：

```
$ mips-linux-gnu-gcc --version
mips-linux-gnu-gcc (Ingenic Linux-Release5.1.9-Default_xburst2_glibc2.29
Fix: uclibc popen and pclose 2023.08-15 09:56:16) 7.2.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

若出现如上信息则可确认 toolchain 安装正确。

1.6.1 如何进行 glibc 和 uclibc 编译

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。

1) glibc 程序编译方法：

默认 link 的 libc 即为 glibc

2) uclibc 程序编译方法：

C_FLAGS+=-muclibc CXX_FLAGS+=-muclibc,LD_FLAGS+=-muclibc

1.7 交叉工具包

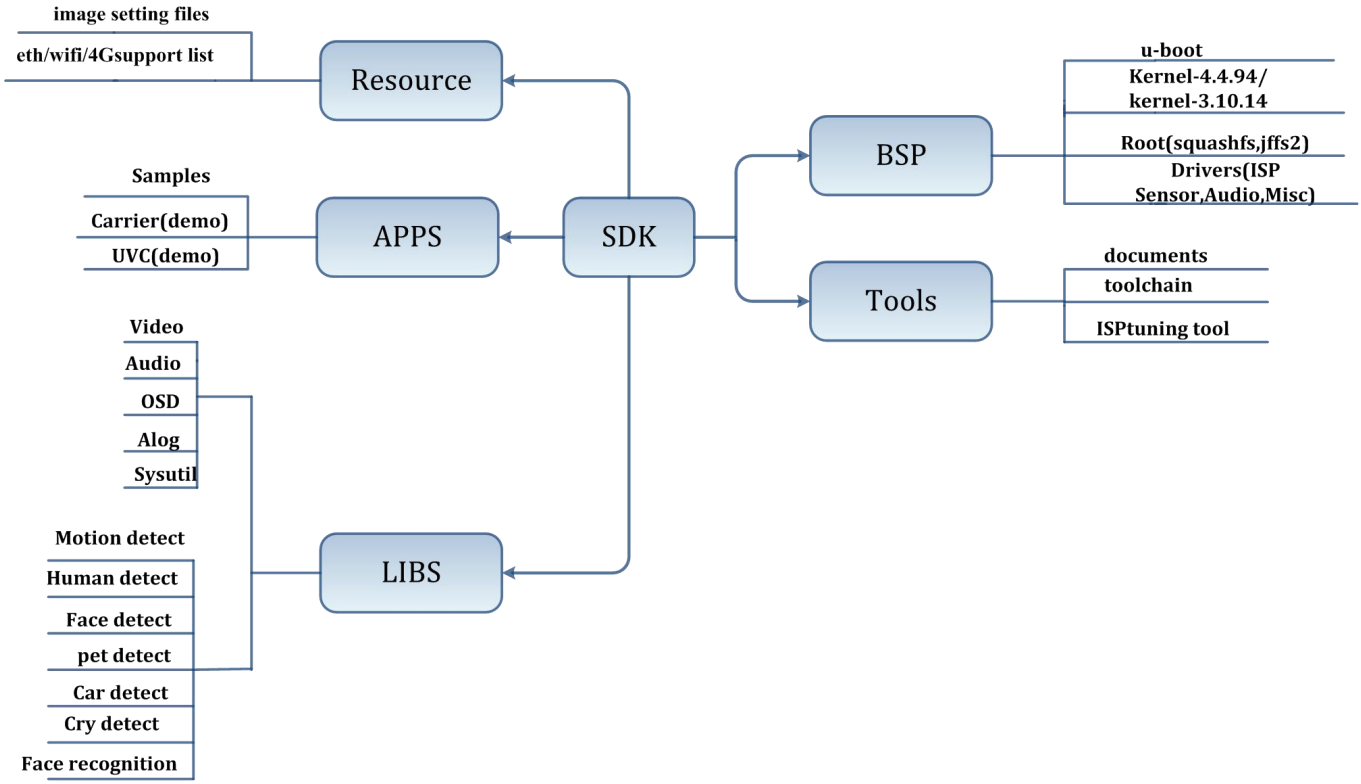
交叉工具链是一个软件安装包，是由很多工具的集合。有我们常用的 gcc 编译工具、objdump 反汇编工具、as 汇编器、ld 链接器、size 查看二进制文件大小等。

1.8 SDK 目录介绍

1.8.1 SDK 介绍

ISVP SDK，即软件开发工具包，包括 API 库、开源源码、文档、Samples 等。开发者可以通过 SDK 快速的开展产品功能开发。以下是 ISVP SDK 的内容概览图：

图 1-1　SDK 内容概览图



1.8.2 SDK 目录介绍

ISVP-T41-1.x.x-202xxxxx 目录结构如下：

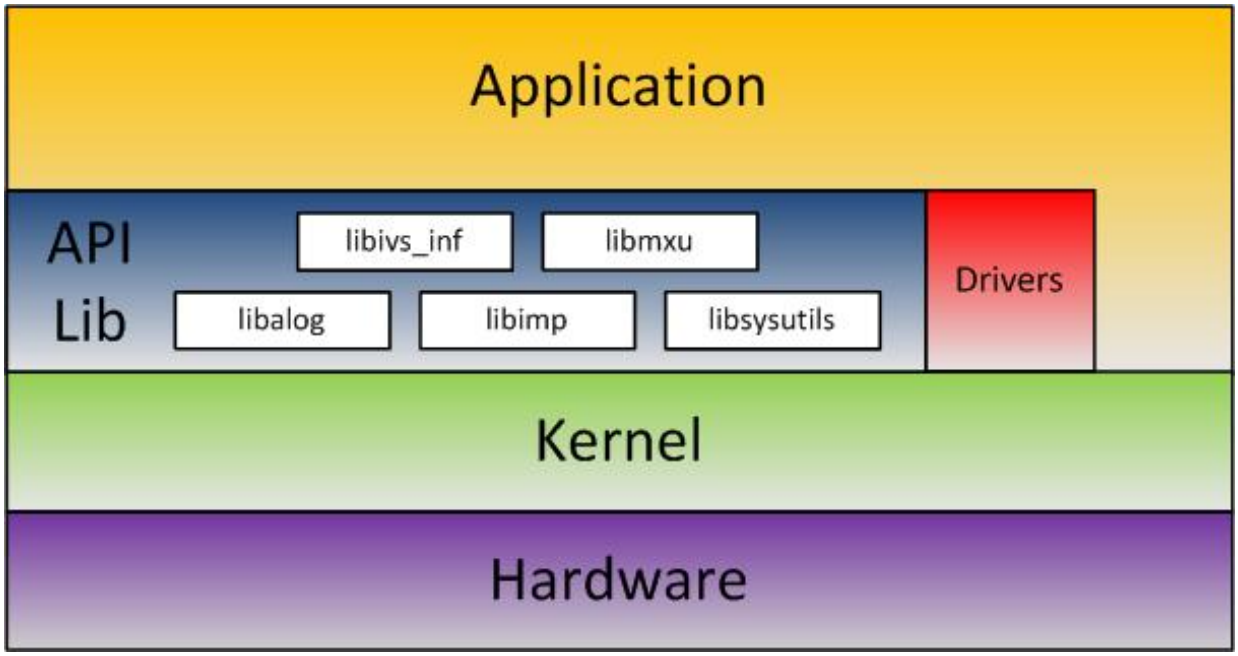
```
├── hardware
│   └── T41 HDK202xxxxx.rar      #硬件参考资料
└── software
    ├── ChangeLog.md
    ├── doc                      #T41 开发文档
    │   ├── BSP
    │   │   ├── T41 SFC 开发指南.pdf
    │   │   ├── IMP T41 BSP 开发指南 V1.0.pdf
    │   │   └── T 系列芯片 WiFi 移植指南.pdf
```

```
│   ├── IMP
│   │   └── T41 码率控制说明.pdf
│   ├── Ingenic T41 SDK 安装及使用指南.pdf
│   ├── Ingenic T41 开发资源编译.pdf
│   ├── Ingenic T41 文件系统制作指南.pdf
│   ├── Ingenictool
│   │   └── USBCloner 烧录指南.pdf
│   └── 调试手册.pdf
└── sdk                          #T41 软件开发包
    ├── Ingenic-SDK-T41-1.x.x-2022xxxx
    │   ├── ChangeLog.md
    │   ├── opensource
    │   │   ├── busybox
    │   │   ├── drivers
    │   │   │   ├── audio
    │   │   │   ├── avpu
    │   │   │   ├── i2c
    │   │   │   ├── isp-t41
    │   │   │   ├── misc      #exfat，电机，定时器，pwm 等驱动
    │   │   │   ├── sensors-t41
    │   │   │   ├── soc-nna
    │   │   │   └── wifi
    │   │   ├── kernel-4.4.94
    │   │   └── uboot
    │   │   └── resource
    │   │   ├── image_t41          # 固件
    │   │   ├── rootfs_720         # 文件系统
    │   │   ├── sensor_settings_t41       # 效果文件
    │   │   ├── toolchain                  # 工具链
```

```
|   |   └── tools_t41          # 调试工具和软件
|   |       ├── bin
|   |       ├── config         #音频配置文件
|   |       ├── factory_test_plan
|   |       ├── opensource_library    #开源库
|   |       ├── pc             #串口驱动，USB 烧录工具，Carrier
|   |       └── script         #rootfs 制作和卸载 ko 的相关脚本
|   ├── sdk
|   |   ├── include            #  SDK 头文件
|   |   ├── lib                #  SDK 库文件
|   |   └── samples            #  sample
|   └── Versions
├── t41_env_setup.sh           #工具链路径和环境变量设置
└── t41_image_mk.sh            #自动解压 SDK，并制作固件
```

1.8.3 SDK 层次结构

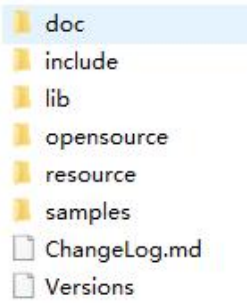
图 1-2 SDK 层次机构



- Hardware: 硬件层，完成 I/O 等具体的硬件功能。
- Linux Kernel: 内核层，完成基础的系统功能，定义硬件资源。
- Drivers: ko 模块驱动，可通过 driver 进行硬件操作。
- API Lib: 接口库，实现硬件功能的抽象，方便于应用层的开发。API 库主要有五部分：
  - libimp: 多媒体功能库，如 H265/H264 编码，JPEG 编码，IVS 和 Audio 等。
  - libsysutils: 系统功能库，如重启，设置系统时间和电池功能等。
  - libalog: ISVP-SDK 的 log 实现库。
  - libivs_inf: IVS 算法库，包括越线检测，周界防范等。
  - libmxu: 512 位 mxu 加速指令算子库。
- Application: 应用层，实现功能逻辑等。

Application 推荐使用 SDK 库提供的 API 并配合 drivers 进行开发。对于一些特殊的功能需求，也可以直接调用内核接口进行开发。

1.8.4 SDK LIB



doc：指导文档

include：相关头文件

lib：相关库文件

opensource：kernel、uboot、drivers、busybox 等

resource：文件镜像、效果文件等

samples：相关应用代码

2 安装升级 T41 DEMO 开发环境

一个全新的板子里面什么都没有(准确的说芯片里面有固化的 Bootrom)，只是一个冷冰冰的机器，怎么让这个机器变得有灵魂呢？看下面介绍。

2.1 烧写 uboot、kernel、fs

2.1.1 准备工作

首先阅读“ IMP T41 BSP 开发指南 V2.1 ”，了解 T41 SDK 的功能、结构、接口等信息。

1. 如果您拿到的单板没有 uboot，就需要使用 SD 卡启动开发板，SD 卡启动卡制作流程参考 2.2.1 SD 卡启动。
2. 如果您拿到的单板已经有 uboot，可以参考 2.3 系统烧录使用网口烧写 uboot、kernel 以及 rootfs 到 Flash 中。

本章所有的烧写操作都是在串口上进行，烧写到 SPI NOR Flash 上。

2.2 uboot 启动

2.2.1 SD 卡启动

SD 卡启动是一个全新开发板最方便的启动方式。SD 卡插上读卡器就可以在 PC 上操作，使用分区工具把 SD 卡分出一个新分区，把编译好的 uboot 拷贝到这个分区上的固定偏移位置上，然后指导 Bootrom 指向 SD 卡的这个位置，启动 uboot。

2.2.1.1 SD 卡启动卡制作

格式化 sd 卡，并将卡启动 uboot 烧录到 SD 卡。

此步骤只需执行一次，若不更新卡起 uboot 则不需要再次执行此步骤。经过此步骤处理的 SD 卡可当作正常卡使用。

将 SD 卡插入电脑


注意

确定插入你的电脑上 TF 卡对应的设备节点文件是不是 sdb(有可能是 sdc 或者 sdd，可以通过插拔 SD 卡确认)；如果对应错了可能会破坏电脑的硬盘文件系统。

步骤一：卸载 SD 卡

```
# umount /media/user/*
```

步骤二：将 SD 卡重新分区

```
# sudo fdisk /dev/sdb
Command (m for help): o                    --创建一个新的分区表
Command (m for help): n                    --添加一个分区 n
--此时会提示：
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p):
--回车默认是 p，创建主分区
Partition number (1-4, default 1):
--回车默认创建 1 号分区
First sector (2048-15261695, default 2048):
--回车使用默认值 2048
--2048 号扇区在 1MB 的位置，给 uboot 空出了空间
Last sector, +sectors or +size{K,M,G,T,P} (2048-15261695, default
15261695):
--回车使用默认值 15261695
Command (m for help): w              --最后一步，向 SD 卡中写入分区表
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
```

到这里分区表已创建完成

步骤三：执行 sync，拔卡，重新插卡，PC 将重新识别分区。

步骤四：格式化 SD 卡为通用的 VFat 文件系统：

```
# mkfs.vfat /dev/sdb1        //这里重新分区的是 sdb1
```

步骤五：编译卡起 u-boot，选择对应的 uboot 类型，将生成的 u-boot-with-spl.bin 文件烧录进 sd 卡的 17KB 偏移处(注意是 dev/sdb 分区，不是 dev/sdb1)：

```
# dd if=u-boot-with-spl.bin of=/dev/sdb bs=1024 seek=17
```

到此，可以用来烧录的 SD 卡制作完成。下面可以使用这个启动卡启动板子。

2.2.2 Nor/Nand 启动

nor/nand 是板子上的存储介质，本质上和上面 SD 卡里的存储介质类似，但这些存储器是焊接到板子上的，不能直接拿下来操作，所以第一次需要用 SD 卡把 uboot 烧录到 flash 的固定偏移位置。这些闪存是 EEPROM 掉电也不丢失数据，通过把 uboot 拷贝到 nor/nand 闪存中后，再次启动开发板时 Bootrom 会直接读取 flash 中的 uboot 分区，则不需要每次插卡启动。

2.2.3 UART 启动

为了解决板卡贴片空的 flash 或者 EMMC，没有预留卡槽以及 USB 接口，不方便烧录的问题，T41 新增了 UART 启动。T41 UART 通过 YMODEM 协议进行数据传输，支持 UART0/1/2 三个接口通信，通信的波特率为常用的 115200 和 9600。

2.2.3.1 UART 启动操作方法

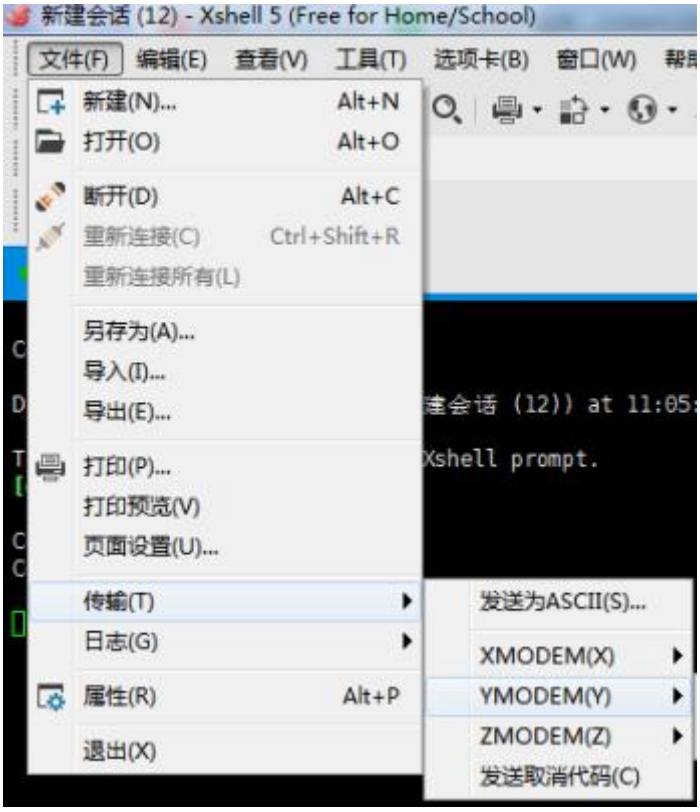
BOOT_SEL1 配置为 1，BOOT_SEL0 配置为 0 进入 UART 启动。

isvp_t41n_uart_msc 配置可以支持启动后烧录 SD/EMMC 等 MSC 接口设备。

isvp_t41n_uart_sfc 配置可以支持启动后烧录 flash 等 SFC 接口设备。

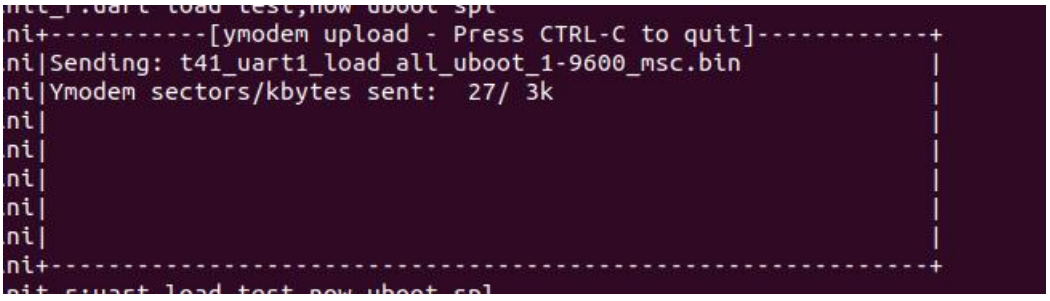
1) Xshell 操作：右键---->传输--->选择 YMODEM-->用 YMODEM 发送--->选择传输文件

图 2-1 Xshell 操作方式



2) Minicom 操作：

图 2-2 Minicom 操作方式



2.3 系统烧录

2.3.1 TFTP 传输与烧录

(1) 目标

将文件通过 tftp 方式，从 PC 端，下载到 Uboot 的内存中，然后写到 flash 中。

(2) 前提

硬件：开发板上网卡；板子通过网线连接到路由器或者交换机上，PC 也连到该路由器或者交换机上，并且处于同一网段上；

软件：PC 端安装并设置好 TFTP 服务，把相应的 u-boot 等文件放到 TFTP 根目录下；uboot 中支持网卡驱动并且支持相应的 TFTP 操作。

(3) 操作

在 uboot 中，执行 $ tftpboot mem_addr file_name ；可以将文件 file_name 传送到 uboot 的内存地址 mem_addr 中。

(4) 传输实例

通过 tftp 命令 load 文件到内存；

```
$ tftpboot 0x80600000 u-boot-with-spl.bin
$ tftpboot 0x80640000 uImage
$ tftpboot 0x808c0000 root-uclibc-toolchain720.squashfs
```

分别把 u-boot-with-spl.bin、uImage、root-uclibc-toolchain720.squashfs 下载到内存的 0x80600000、0x80640000、0x808c0000 处。


注意

根据实际文件大小来修改偏移位置。

(5) 烧录到 Nor

把下载到内存上的文件，写到开发板上的 nor flash 上，重启后可以从 nor 上直接读取文件，而不必每次手动 load 文件到内存上。烧录命令如下：

```
$ sf probe;sf erase 0x0 0x1000000;sf write 0x80600000 0x0 0x1000000
```

- sf probe 在使用 sf read、sf write 之前，定要调用 sf probe；
- sf erase 擦除指定位置，指定长度的 flash 内容，擦除后内容全 1；
- sf write 写内存数据到 flash 上。

2.3.2 SD 卡传输与烧录

(1) 目标

将某个文件写到内存中，然后写到 flash 上。

(2) 前提

硬件：开发板上有 SD 卡槽；有读卡器把 SD 卡插到 PC 上拷贝数据。

软件：uboot 中支持 SD 卡驱动。

(3) 操作

在 uboot 中，执行 $ fatls mmc 0 查看 SD 卡中的文件，

$ fatload mmc 0 mem_addr file_name 可以将文件 file_name 传送到 Uboot 的内存地址 mem_addr 中了。

(4) 传输实例

通过 fatload 命令 load 文件到内存；

```
$ fatload mmc 0 0x80600000 u-boot-with-spl.bin
$ fatload mmc 0 0x80640000 uImage
$ fatload mmc 0 0x808c0000 root-uclibc-toolchain720.squashfs
```

分别把 u-boot-with-spl.bin、uImage、root-uclibc-toolchain720.squashfs 下载到内存的 0x80600000、0x80640000、0x808c0000 处。


注意

根据实际文件大小来修改偏移位置。

(5) 烧录到 Nor

把下载到内存上的文件，写到开发板上的 nor flash 上，重启后可以从 nor 上直接读取文件，而不必每次手动 load 文件到内存。烧录命令如下：

```
$ sf probe;sf erase 0x0 0x1000000;sf write 0x80600000 0x0 0x1000000
```

- sf probe 在使用 sf read、sf write 之前，一定要调用 sf probe；
- sf erase 擦除指定位置，指定长度的 flash 内容，擦除后内容全 1；
- sf write 写内存数据到 flash 上。

2.3.3 USB 烧录

1) 目标

将某个文件写到 flash 上。

2) 前提

硬件：开发板上有 USB 烧录接口，数据线。

软件： USB 烧录 PC 工具。

3) 操作

参考“ USBCloner 烧录指南.pdf ”。

2.4 串口连接

开发板串口负责交互信息传输，内核打印会通过串口发送到显示屏上，开发板通过串口接收键盘的输入信息。常用的串口工具有 minicom (linux)、putty、Xshell 等软件。

PC 端串口与开发板串口的连接，需要配置对应的参数，如串口号，波特率，奇偶校验，数据位，停止位等。比如配置串口号为 com6，波特率为 115200，8 位数据位，一位停止位，无奇偶校验位。

3 开发前环境准备

3.1 管脚复用

无。

3.2 连接串口

通过 DEMO 板的串口连接到 CPU。

3.3 NFS 环境

通过 DEMO 板的网口连接 NFS。

4 使用 SDK 和 DEMO 板进行开发

4.1 开启 Linux 下的网络

步骤一：设置网络。

ifconfig eth0 xxx.xxx.xxx.xxx

route add default gw xxx.xxx.xxx.1

步骤二：ping 其他机器，如无意外，网络将能正常工作。

4.2 使用 NFS 文件系统进行开发

步骤一：在开发阶段，推荐使用 NFS 作为开发环境，可以省去重新制作和烧写根文件系统的工作。

步骤二：挂载 NFS 文件系统的操作命令：

```
mount -t nfs -o nolock 193.169.4.2:/home_b/nfsroot/user  /mnt/
```

步骤三：然后就可以在/mnt 目录下访问服务器上的文件，并进行开发工作。

4.3 运行 APP 业务

步骤一：在串口上，进入 system/lib/modules 目录，加载驱动, 例：

```
cd system/lib/modules
insmod tx-isp-t41.ko clk_name="mpll" isp_clk=200000000
insmod sensor_gc2053_t41.ko
insmod avpu.ko clk_name="mpll" avpu_clk=650000000
```

步骤二：进入各 sample 目录下执行相应样例程序(sample 需要先在服务器上成功编译过)

```
cd system/bin;./carrier-server --st gc2053
```