

# PMON 运行并 load 内核启动的方法

(v1.00)

本文档面向对运行 PMON 方法和 load 内核不清楚的读者，以帮助他们快速的通过我们提供的 soc\_up\_33M.bit 在开发板上运行 PMON，并在 PMON 里 load 内核并运行，以测试开发板上网口、DDR3 颗粒、串口、SPI flash、nand flash 等芯片的正确性。

为此，需要依次完成以下步骤：

- (1) 烧写 PMON 文件（gzrom.bin）到可插拔 SPI flash 上。
- (2) 下载 bit 流文件（soc\_up\_33M.bit）。
- (3) 运行 PMON。
- (4) 搭建 tftp 服务器 Load 内核(vmlinux)。
- (5) 启动内核。

## 1 烧写 PMON

烧写 PMON 到可插拔 flash 有两种方法。

方法一：使用 EZP2010 编程器：

需要自行购买该编程器，具体可在淘宝上搜索，价格约 100 元左右。

使用该编程器，需手动安装驱动。

方法二：使用基于本 FPGA 实验箱的串口编程 flash 的 bit 流文件

该方法是使用龙芯开源的 gs132 核搭建了一个小的 soc，

该 soc 外设有串口、flash 芯片和指令数据 ram。

该 soc 在 FPGA 上生成的 bit 流文件可实现通过串口在线编程 flash 芯片。

编程过程中，不需要拔下 flash 芯片，且速率达到 6KB/sec。

推荐大家使用方法二，因为这样不需要插拔 flash 芯片，防止 flash 芯片引脚损坏。

烧写完成后，将芯片插入开发板上，注意方向（开发板方正，flash 芯片有缺口部分朝左）！

## 2 下载 bit 流文件

将开发板与主机间的下载线连接好，开发板上电，使用 Vavidao 工具里的 Open Hardware Manager 下载 soc\_up\_33M.bit 到开发板上。

具体下载方法参见发布包 doc/Vivado 使用说明.pdf 里的第 1.5 节。

## 3 运行 PMON

第 1 节烧写的 PMON 运行在第 2 节下载的 soc\_up 上，需要使用串口展示运行信息。

将开发板与主机间的串口线连接好，打开串口软件，波特率设置为 57600，具体见 3.1 节。

### 3.1 串口软件

#### 3.1.1 Linux 下

##### (1) 配置

在终端下运行：

```
[abc@www ~]$ minicom -s
```

选择 Serial port setup:

```
ne sample memory test
+-----[configuration]-----+
| Filenames and paths          |
| File transfer protocols      |
| Serial port setup            |
| Modem and dialing            |
| Screen and keyboard          |
| Save setup as dfl             |
| Save setup as..              |
| Exit                          |
+-----+-----+-----+-----+
```

进入配置界面：

```
+-----+-----+-----+-----+
| A - Serial Device           : /dev/ttyUSB0 |
| B - Lockfile Location       : /var/lock    |
| C - Callin Program          :              |
| D - Callout Program         :              |
| E - Bps/Par/Bits            : 57600 5N1    |
| F - Hardware Flow Control   : No          |
| G - Software Flow Control   : No          |
|                               |            |
| Change which setting?      |            |
+-----+-----+-----+-----+
```

其中 E 行依据开发板上串口控制器的初始化代码中设置的波特率进行选择，F 和 G 行选择 NO。

配置完成后按 Enter 返回，选择 Save setup as dfl 保存为默认设置。

##### (2) 运行

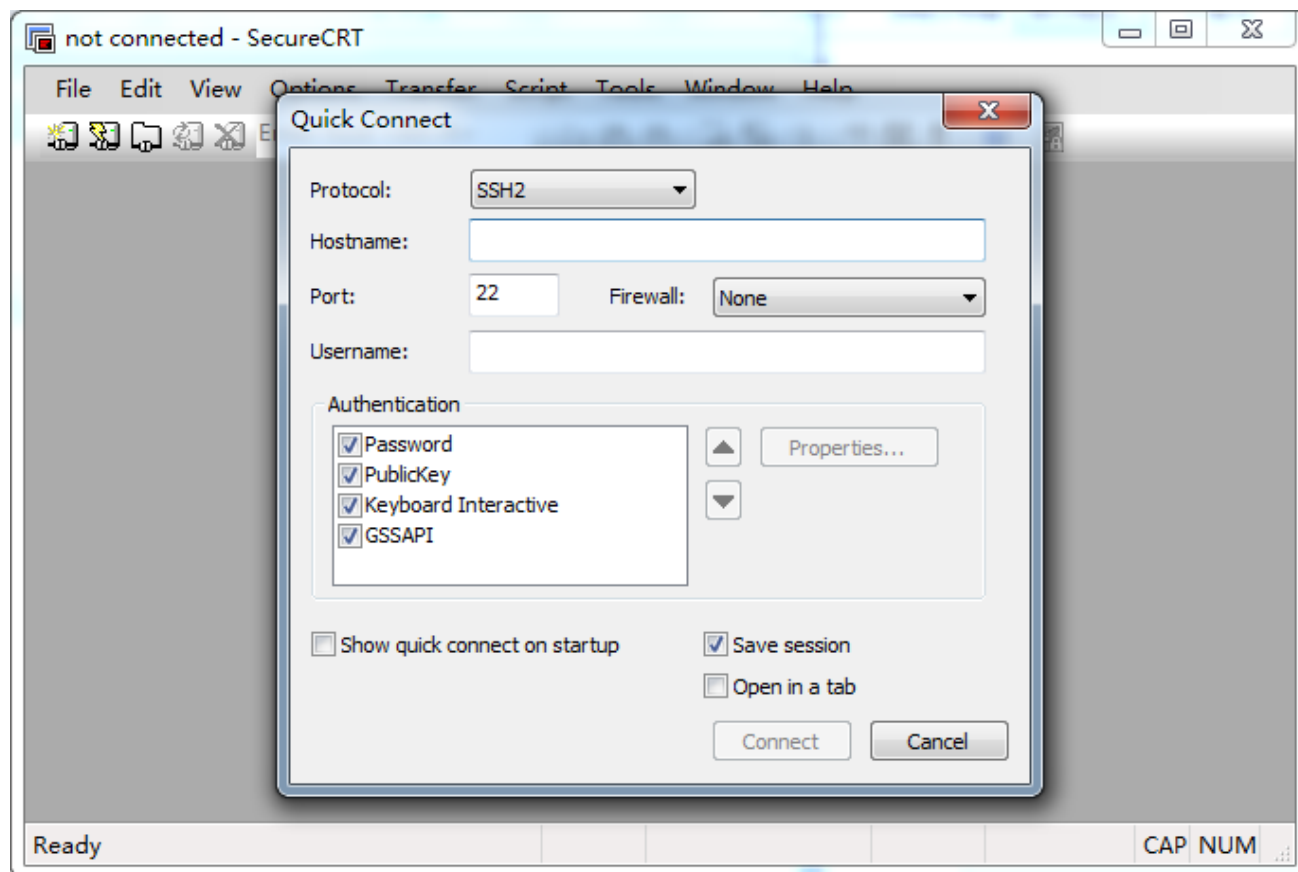
将 USB 转串口一端连到电脑上，一端连到串口线上，串口线另一端连接到开发板上串口接口上。在 Linux 终端运行如下命令，开启电脑上的串口界面，开发板即可与电脑进行交换了。

```
[abc@www ~]$ sudo minicom
```

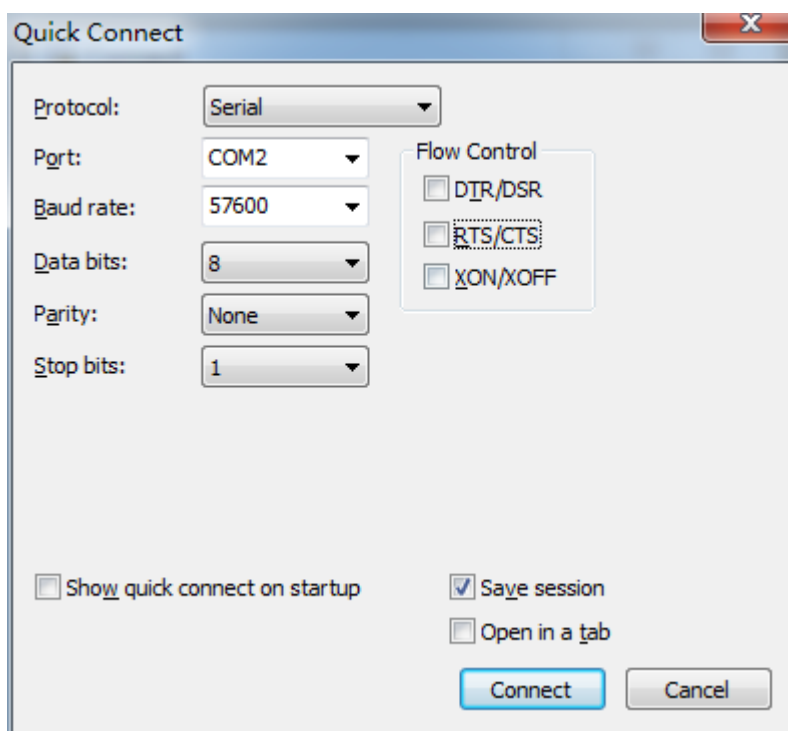
### 3.1.2 Windows 下

Windows 下可以使用免安装的 SecureCRTPortable 串口软件。先使用 USB 转串口和串口连接线将电脑和开发板相连。

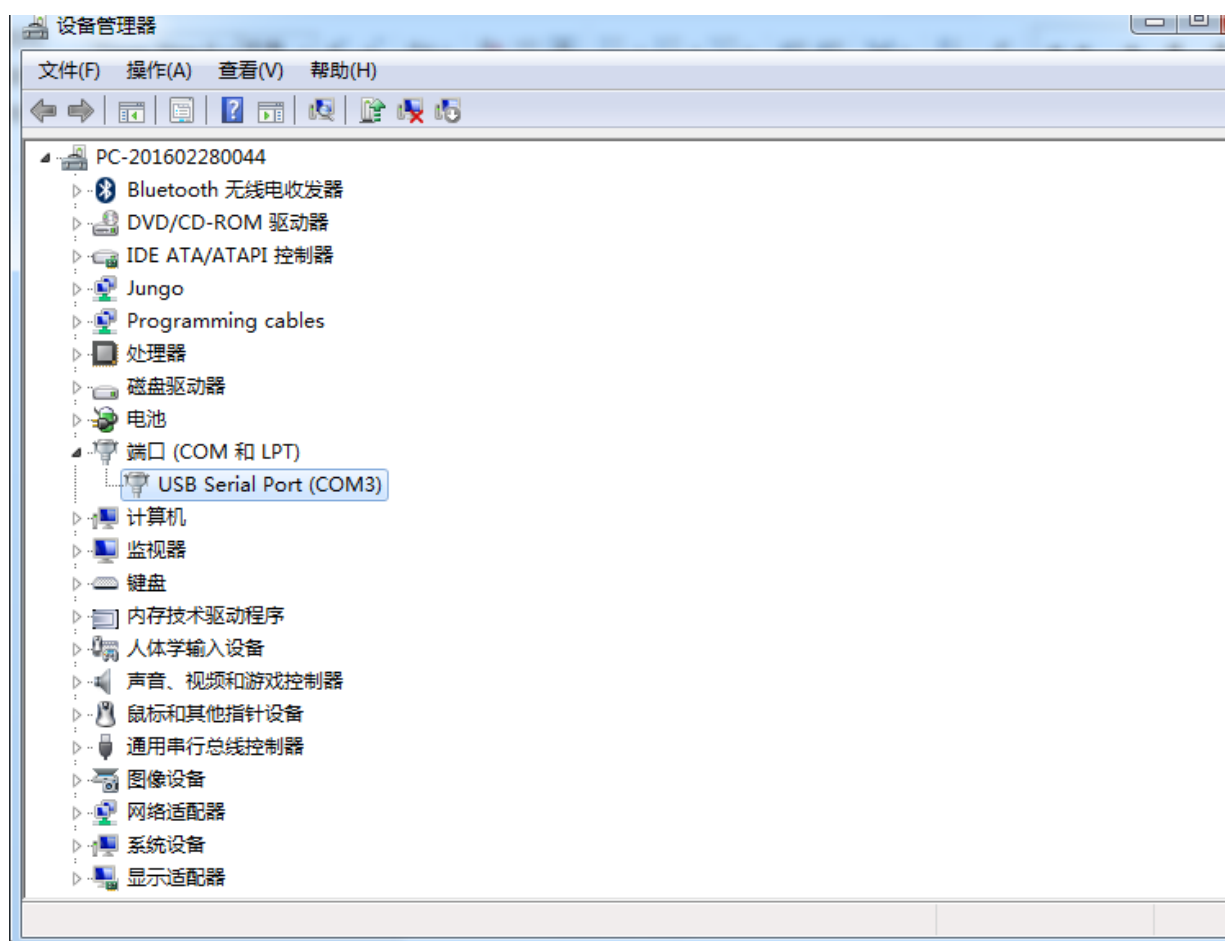
双击程序打开，第一次启动界面如下：



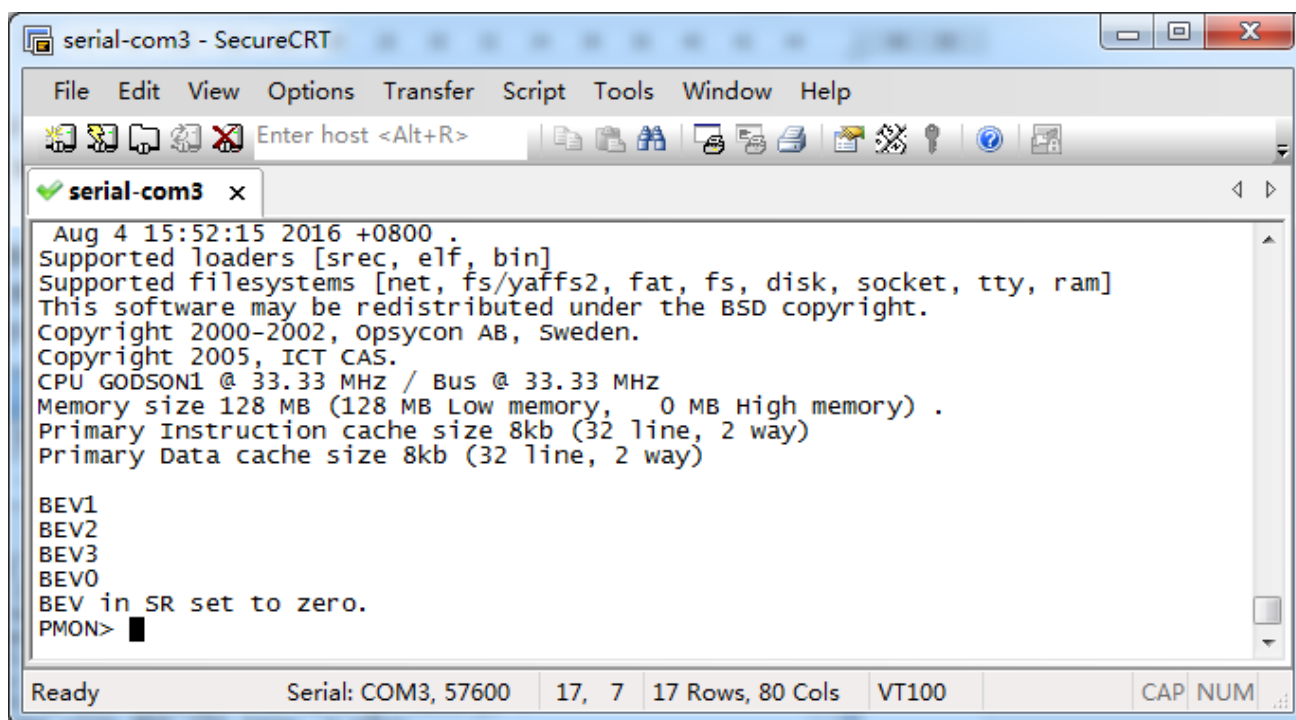
第一行 **Protocol** 下拉选择 **Serial**，如下：



其中 Baud rate 为选择波特率，需根据开发板上串口控制器的初始化代码中设置的波特率进行选择(对于本次校验设备，波特率需选择 57600)。右侧 **Flow Control** 全不选。Port 的选择需根据 Windows 电脑上的端口进行选择，可以右键电脑选择设备管理器进入**设备管理器**查看：

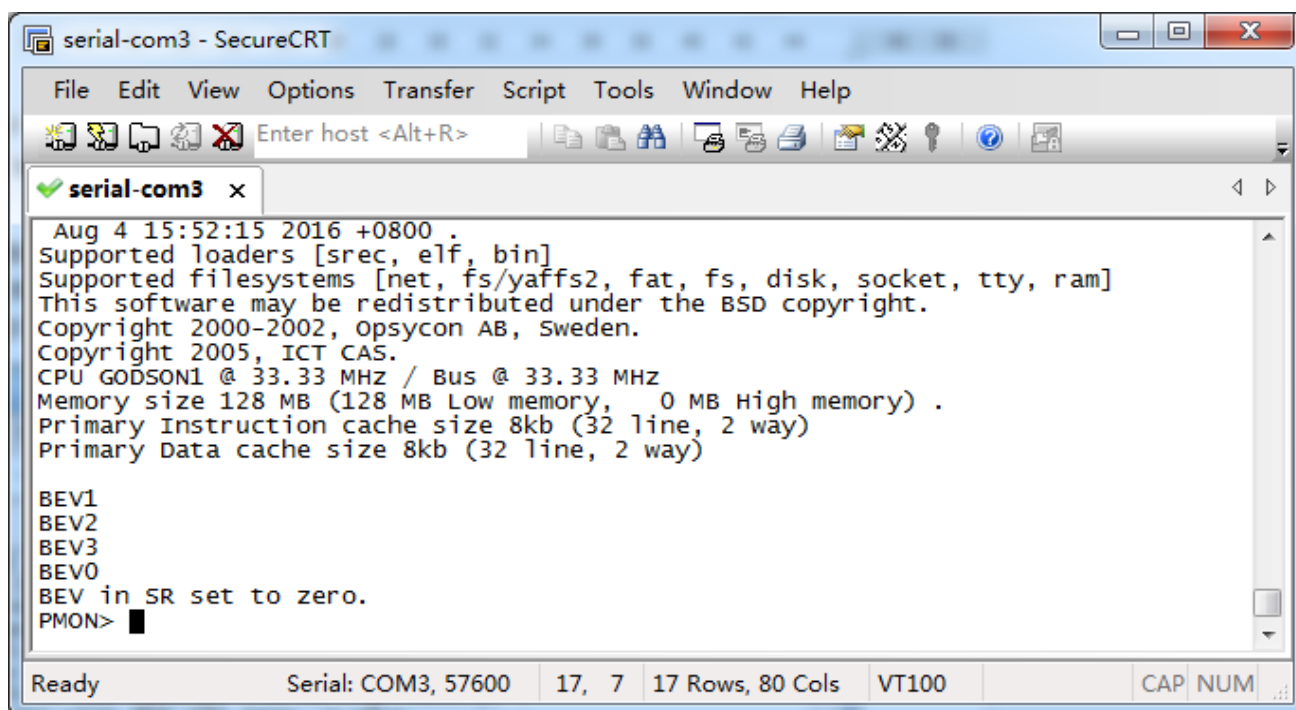


配置好串口后，点击 **connect**，即可进入串口界面，在波特率设置正确的情况下，可以通过串口与开发板进行交互，如下：



## 3.2 PMON 命令

连接上串口，打开串口软件，设置好波特率，则可以在串口窗口中看到 PMON 运行信息，运行成功后则会进入 PMON 提示符，此时可以输入 PMON 命令。



比如， SoC\_up 中具有 MAC 控制器，PMON 中也有 MAC 驱动，则我们输入命令“ifconfig dmfe0

**10.90.50.44**”则可以给开发板上的网卡配置 IP 为 **10.90.50.44**（具体需配置的 IP 请查阅同网段的电脑 IP），假设同网段的电脑 IP 为 **10.90.50.43**,则可以继续输入命令“**ping 10.90.50.43**”用于查看网络是否成功接入。Linux 在 ping 网络是会一直发 ping 包，可以 **Ctrl+C** 取消 ping。运行结果如下：

```
PMON> ifconfig dmfe0 10.90.50.44
rx ring 70acee0
tx ring 70acf60
DE4X5_BMR= fe000000
DE4X5_TPD= 0
DE4X5_RRBA= 70acee0
DE4X5_TRBA= 70acf60
DE4X5_STS= f0660004
DE4X5_OMR= 32002242
TX error status2 = 0x00000000
After setup
DE4X5_BMR= fe000000
DE4X5_TPD= 0
DE4X5_RRBA= 70acee0
DE4X5_TRBA= 70acf60
DE4X5_STS= f0660004
DE4X5_OMR= 32002242
PMON> ping 10.90.50.43
PING 10.90.50.43 (10.90.50.43): 56 data bytes
64 bytes from 10.90.50.43: icmp_seq=0 ttl=64 time=3.708 ms
64 bytes from 10.90.50.43: icmp_seq=1 ttl=64 time=2.331 ms
64 bytes from 10.90.50.43: icmp_seq=2 ttl=64 time=2.235 ms

--- 10.90.50.43 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 2.235/2.750/3.708 ms
PMON>
```

## 4 运行 Linux

由于运行 linux 时，最初的内核需要使用网口 load 进入内存执行，因而需要先搭建 Tftp 服务器。

具体方法参见发布包 soc\_run\_os/lab\_environment/tftp。

目前运行 Linux 的方法是，先运行 PMON，随后通过网口 load Linux 内核进入 FPGA 上的 DDR3 内存上，load 命令为“load tftp://10.90.50.43/vmlinux”。其中 10.90.50.43 为搭建的 tftp 服务器的 IP。

目前 Linux 内核在教学 SoC 上运行时，串口的波特率为 115200，与 PMON 的不同。

具体过程如下：

启动 PMON 后，输入命令“ifconfig dmfe0 10.90.50.44”给开发板上的网卡配置 IP 为 10.90.50.44（具体需配置的 IP 请查阅同网段的电脑 IP），假设同网段的电脑 IP 为 10.90.50.43,则可以继续输入命令“ping 10.90.50.43”用于查看网络是否成功接入。

网络配置好了，需要通过网络下载 Linux 内核，需要先搭建好 tftp 服务器，假设搭建的 tftp 服务器 IP 为 10.90.50.43，将要下载 Linux 内核放到 tftp 服务器的根目录下，输入命令“load tftp://10.90.50.43/vmlinux”即可 load 内核进入 FPGA 上的内存。

```
PMON> load tftp://10.90.50.43/vmlinux
Loading file: tftp://10.90.50.43/vmlinux (elf)
0x80200000/5350380 + 0x8071a3ec/169220(z) + 6926 syms\
Entry address is 802041f0
PMON>g console=ttyS0,115200 rdinit=sbin/init
```

上表中显示 load 成功了，输入命令“g console=ttyS0,baudrate rdinit=sbin/init”即可运行该内核，命令中 baudrate 需为数字，即为串口控制器设置的波特率，设置不对时，串口显示字符为乱码。Linux 内核运行时波特率为 115200。

```
.....
mount: mounting n on /proc/bus/usb failed: No such file or directory
mdev: /sys/class: No such file or directory
login
Godson2@[/]>ls
bin      etc      lib      mnt      root     sys      usr
dev      hello.c  linuxrc  proc     sbin     tmp
Godson2@[/]>cd root/
Godson2@[~]>vi 1.txt
Godson2@[~]>ls
1.txt
Godson2@[~]>cat 1.txt
hello,world!
```

```
Godson2@[~]>
```

当运行 Linux 内核成功后，会出现“Godson2@[~]>”提示符，可以使用常用的 Linux 命令，如上表。

## 5 加载内存到 NandFlash

当前 SoC\_up 支持 128MB 的 NandFlash 作为电脑中的硬盘功能。因而可以将 Linux 内核加载到 NandFlash 上。

如果将内核加载至 NandFlash 中，且 PMON 中设置好参数。则复位实验箱后，会先自动运行 PMON 对设备进行初始化，随后 PMON 会自动加载 NandFlash 中的 Linux 内核进行启动，这就是通常电脑启动的过程。

Linux 内核加载至 NandFlash 中并配置 PMON 的方法如下：

- (1) 实验箱运行至 PMON;
- (2) 擦除 NandFlash, PMON 命令: `mtd_erase /dev/mtd0r`  
`mtd_erase /dev/mtd1r`
- (3) 设置网口 IP, PMON 命令: `ifconfig dmfe0 x.x.x.x`, 其中 x.x.x.x 为配置的 IP;
- (4) 拷贝内核文件, PMON 命令: `devcp tftp://x.x.x.x /vmlinux /dev/mtd0`, 其中 x.x.x.x 为搭建的 tftp 服务器的 IP;

如果传输过程中卡顿了，请按 Ctrl+C 取消本次传输后，在输入上述命令开始传输。如果多次取消后，依然传输失败，请复位开发板后重新来过。

如果 devcp 命令报错了 Exception，请复位开发板后重新来过。

- (5) 设置分区空间大小, PMON 命令: `set mtdparts nand-flash:50M@0(kernel)ro,-(rootfs);`

如果出现 warning，不用管。

- (6) 设置启动分区及参数, PMON 命令:

```
set al /dev/mtd0;
```

```
set append "console=ttyS0,115200 rdinit=/sbin/init initcall_debug=1 loglevel=20"
```

- (7) 重启 FPGA 实验箱，会自动完成本章开头描述的启动过程，自动运行到 Linux 内核状态：

```
initcall xfrm4_transport_init+0x0/0x10 returned 0 after 24 usecs
calling xfrm4_mode_tunnel_init+0x0/0x10 @ 1
initcall xfrm4_mode_tunnel_init+0x0/0x10 returned 0 after 20 usecs
calling cubictcp_register+0x0/0x98 @ 1
TCP cubic registered
initcall cubictcp_register+0x0/0x98 returned 0 after 4199 usecs
calling packet_init+0x0/0x5c @ 1
NET: Registered protocol family 17
initcall packet_init+0x0/0x5c returned 0 after 6830 usecs
calling init_oops_id+0x0/0x5c @ 1
initcall init_oops_id+0x0/0x5c returned 0 after 348 usecs
calling disable_boot_consoles+0x0/0x74 @ 1
initcall disable_boot_consoles+0x0/0x74 returned 0 after 11 usecs
calling pm_qos_power_init+0x0/0xfc @ 1
initcall pm_qos_power_init+0x0/0xfc returned 0 after 24167 usecs
calling random32_rseed+0x0/0x6c @ 1
initcall random32_rseed+0x0/0x6c returned 0 after 561 usecs
calling seqgen_init+0x0/0x20 @ 1
initcall seqgen_init+0x0/0x20 returned 0 after 1178 usecs
calling scsi_complete_async_scans+0x0/0x114 @ 1
initcall scsi_complete_async_scans+0x0/0x114 returned 0 after 12 usecs
calling rtc_hctosys+0x0/0x160 @ 1
drivers/rtc/hctosys.c: unable to open rtc device (rtc0)
initcall rtc_hctosys+0x0/0x160 returned -19 after 9997 usecs
calling tcp_congestion_default+0x0/0xc @ 1
initcall tcp_congestion_default+0x0/0xc returned 0 after 20 usecs
calling ip_auto_config+0x0/0xe0 @ 1
initcall ip_auto_config+0x0/0xe0 returned 0 after 252 usecs
calling initialize_hashrnd+0x0/0x28 @ 1
initcall initialize_hashrnd+0x0/0x28 returned 0 after 330 usecs
async_waiting @ 1
async_continuing @ 1 after 18 usec
Freeing unused kernel memory: 1376k freed
Algorithmics/MIPS FPU Emulator v1.5
mount: mounting n on /proc/bus/usb failed: No such file or directory
mdev: /sys/class: No such file or directory
login
Godson2@ />
Godson2@ />
Godson2@ />
Godson2@ />ls
bin      etc      lib      mnt      root     sys      usr
dev      hello.c linuxrc  proc     sbin     tmp
```

Ready Serial: COM3, 57600 43, 14 43 Rows, 166 Cols VT100 CAP NUM