

**U-boot:** 一种开源 **bootloader**, 作用是用来引导操作, 以及给开发人员提供测试调试工具。

## 1 help : 输出 u-boot 所支持的命令及简要帮助信息

TINY4412 # **help**

? - alias for 'help'  
base - print or set address offset  
bdfinfo - print Board Info structure  
boot - boot default, i.e., run 'bootcmd'  
bootd - boot default, i.e., run 'bootcmd'  
bootelf - Boot from an ELF image in memory  
bootm - boot application image from memory  
bootp - boot image via network using BOOTP/TFTP protocol  
bootvx - Boot vxWorks from an ELF image  
chpart - change active partition  
cmp - memory compare  
coninfo - print console devices and information  
cp - memory copy  
crc32 - checksum calculation  
dcache - enable or disable data cache  
dnw - dnw - initialize USB device and ready to receive for Windows server (specific)  
  
echo - echo args to console  
editenv - edit environment variable  
emmc - Open/Close eMMC boot Partition  
env - environment handling commands  
exit - exit script  
ext2format- ext2format - disk format by ext2  
  
ext2load- load binary file from a Ext2 filesystem  
ext2ls - list files in a directory (default /)  
ext3format- ext3format - disk format by ext3  
  
false - do nothing, unsuccessfully  
fastboot- fastboot- use USB Fastboot protocol  
  
fatformat- fatformat - disk format by FAT32  
  
fatinfo - fatinfo - print information about filesystem  
fatload - fatload - load binary file from a dos filesystem  
  
fatls - list files in a directory (default /)  
fdisk - fdisk for sd/mmc.

**go** - start application at address 'addr'  
**help** - print command description/usage  
**icache** - enable or disable instruction cache  
**iminfo** - print header information for application image  
**imxtract**- extract a part of a multi-image  
**itest** - return true/false on integer compare  
**loadb** - load binary file over serial line (kermit mode)  
**loads** - load S-Record file over serial line  
**loady** - load binary file over serial line (ymodem mode)  
**loop** - infinite loop on address range  
**md** - memory display  
**mm** - memory modify (auto-incrementing address)  
**mmc** - MMC sub system  
**mmcinfo** - mmcinfo <dev num>-- display MMC info  
**movi** - movi - sd/mmc r/w sub system for SMDK board  
**mtdparts**- define flash/nand partitions  
**mtest** - simple RAM read/write test  
**mw** - memory write (fill)  
**nfs** - boot image via network using NFS protocol  
**nm** - memory modify (constant address)  
**ping** - send ICMP ECHO\_REQUEST to network host  
**printenv**- print environment variables  
**reginfo** - print register information  
**reset** - Perform RESET of the CPU  
**run** - run commands in an environment variable  
**saveenv** - save environment variables to persistent storage  
**setenv** - set environment variables  
**showvar** - print local hushshell variables  
**sleep** - delay execution for some time  
**source** - run script from memory  
**test** - minimal test like /bin/sh  
**tftpboot**- boot image via network using TFTP protocol  
**true** - do nothing, successfully  
**usb** - USB sub-system  
**version** - print monitor version  
TINY4412 #

## 2 ? : help 的别名

TINY4412 # ?  
? - alias for 'help'  
base - print or set address offset  
.....  
2 / 35

### 3 bdfinfo : 查看配置信息

TINY4412 # **bdfinfo**

**arch\_number** = 0x00001200 ->开发板的机器码，用来引导操作系统的内核

**boot\_params** = 0x40000100 ->启动参数存储的内存位置

**DRAM bank** = 0x00000000 -> DRAM 编号，从开始，这里是第 0 个 DRAM

-> **start** = 0x40000000 -->DRAM 的起始地址

-> **size** = 0x10000000 -->DRAM 的大小 256\*1024\*1024 字节 (256M)

**DRAM bank** = 0x00000001 -> DRAM 编号，从开始，这里是第 1 个 DRAM

-> **start** = 0x50000000 -->DRAM 的起始地址

-> **size** = 0x10000000 -->DRAM 的大小 (256M)

**DRAM bank** = 0x00000002

-> **start** = 0x60000000

-> **size** = 0x10000000 (256M)

**DRAM bank** = 0x00000003

-> **start** = 0x70000000

-> **size** = 0x0FF00000 (256M)

**ethaddr** = 00:40:5c:26:0a:5b ->网卡 MAC 地址

**ip\_addr** = 192.168.0.20 ->开发板的 IP

**baudrate** = 0 bps ->波特率，这里是代码有问题，应该 115200

**TLB addr** = 0x3FFF0000 ->MMU 映射表存储位置

**relocaddr** = 0xC3E00000 ->代码重新定位的地址

**reloc off** = 0x00000000

**irq\_sp** = 0xC3CFBF58

**sp start** = 0xC3CFBF50

**FB base** = 0x00000000

TINY4412 #

关于 **arch\_number** = 0x00001200 ->开发板的机器码，用来引导操作系统的内核

```
TINY4412 # bdfinfo
arch_number = 0x00001200
boot_params = 0x40000100
DRAM bank   = 0x00000000
-> start    = 0x40000000
-> size     = 0x10000000
DRAM bank   = 0x00000001
-> start    = 0x50000000
-> size     = 0x10000000
DRAM bank   = 0x00000002
-> start    = 0x60000000
-> size     = 0x10000000
DRAM bank   = 0x00000003
-> start    = 0x70000000
-> size     = 0x0FF00000
ethaddr     = 00:40:5c:26:0a:5b
ip_addr     = 192.168.0.20
baudrate    = 0 bps
TLB addr    = 0x3FFF0000
```

1201	visstrim_mv10	MACH_VISSTRIM_MV10	VISSTRIM_MV10	4159
1202	mx28_wilma	MACH_MX28_WILMA	MX28_WILMA	4164
1203	msm8625_ffa	MACH_MSM8625_FFA	MSM8625_FFA	4166
1204	vpul01	MACH_VPU101	VPU101	4167
1205	baileys	MACH_BAILEYS	BAILEYS	4169
1206	familybox	MACH_FAMILYBOX	FAMILYBOX	4170
1207	ensemble_mx35	MACH_ENSEMBLE_MX35	ENSEMBLE_MX35	4171
1208	sc_sps_1	MACH_SC_SPS_1	SC_SPS_1	4172
1209	tiny4412	MACH_TINY4412	TINY4412	4608
/whz/linux-3.5/linux-3.5/arch/arm/tools/mach-types ASCII=77,HEX=4D,1209,11-17 1209-100%				

路径

```
351 /*
352  * machine type
353  */
354
355 #define MACH_TYPE_TINY4412 4608 /* Tiny4412 machine ID */
356
357 #define CONFIG_ENV_OFFSET 0x0007C000
358
359 /*-----
360  * Boot configuration
361  */
362
363 #include <u-boot_tiny4412/include/configs/tiny4412.h>
```

路径

## 4 环境变量

### 4. 1 printenv : 打印环境变量 (pri 或者 print)

```
TINY4412 # printenv
baudrate=115200
bootcmd=movi read kernel 0 40008000;movi read rootfs 0 41000000 100000;bootm 40008000 41000000
bootdelay=3
ethaddr=00:40:5c:26:0a:5b
gatewayip=192.168.0.1
ipaddr=192.168.0.20
netmask=255.255.255.0
serverip=192.168.0.10
```

Environment size: 252/16380 bytes

说明:

baudrate: 当前的波特率。一般不修改。

bootcmd: 启动命令。

bootdelay: 启动命令 bootcmd 延时执行的时间。

ethaddr: 网卡 MAC 地址。

gatewayip: 网关 IP 地址。

ipaddr: 开发板 IP 地址。

netmask: 子网掩码。

serverip: 服务器 IP(一般是 PC 的 IP,给开发板提供各种网络服务的主机的 IP)

**bootargs:u-boot** 传递给操作系统内核的启动参数。(很重要, 后边会讲)

## 4.2 打印特定的环境变量:

```
TINY4412 # printenv bootdelay
```

```
bootdelay=3
```

```
TINY4412 #
```

```
TINY4412 # setenv tekkaman echo "I am Tekkaman Ninja!"
```

```
TINY4412 # echo I Love Linux ;${tekkaman}
```

```
I Love Linux
```

```
I am Tekkaman Ninja!
```

```
TINY4412 #
```

## 4.3 setenv: 设置, 删除, 修改环境变量

```
TINY4412 # help setenv
```

→查看 setenv 的帮助

```
setenv - set environment variables
```

→命令的作用

Usage:

→命令使用

```
setenv name value ...
```

→修改或新增环境变量命令格式,

- set environment variable 'name' to 'value ...' →命令说明

```
setenv name
```

→删除环境变量命令格式,

- delete environment variable 'name'

→命令说明

修改环境变量:

```
TINY4412 # setenv bootdelay 7
```

→设置 bootdelay 等于 7

```
TINY4412 # saveenv
```

```
Saving Environment to SMDK bootable device...
```

```
done
```

```
TINY4412 #
```

重新启动开发板: 倒计时就是 7 秒了。

```
Checking Boot Mode ... SDMMC
REVISION: 1.1
MMC Device 0: 3724 MB
MMC Device 1: 3728 MB
MMC Device 2: N/A
Net:      No ethernet found.
Hit any key to stop autoboot:  7
```

## 4.4 增加环境变量

当前环境变量中还没有 bootargs 变量，下面新增这个环境变量：

```
TINY4412 #setenv bootargs 'root=/dev/mmcblk0p2 rootfstype=ext3 console=ttySAC0,115200
init=/linuxrc uhost0=y ctp=2 skipcali=y lcd=S70'
```

```
TINY4412 # saveenv
```

Saving Environment to SMDK bootable device...

done

```
TINY4412 #
```

打印环境变量：

```
TINY4412 # printenv
```

baudrate=115200

```
bootargs=root=/dev/mmcblk0p2 rootfstype=ext3 console=ttySAC0,115200 init=/linuxrc
uhost0=y ctp=2 skipcali=y lcd=S70
```

```
bootcmd=movi read kernel 0 40008000;movi read rootfs 0 41000000 100000;bootm 40008000
41000000
```

```
bootdelay=7
```

```
ethaddr=00:40:5c:26:0a:5b
```

```
gatewayip=192.168.0.1
```

```
ipaddr=192.168.0.20
```

```
netmask=255.255.255.0
```

```
serverip=192.168.0.10
```

Environment size: 370/16380 bytes

```
TINY4412 #
```

## 4.5 删除环境变量

1) 先新加一个无用的环境变量用来测试。

```
TINY4412 # setenv myvar abc=123 bcd=456 root=/dev/nfs
```

```
TINY4412 # saveenv
```

Saving Environment to SMDK bootable device...

done

```
TINY4412 # printenv
```

baudrate=115200

```
bootargs=root=/dev/mmcblk0p2 rootfstype=ext3 console=ttySAC0,115200 init=/linuxrc
uhost0=y ctp=2 skipcali=y lcd=S70
```

```
bootcmd=movi read kernel 0 40008000;movi read rootfs 0 41000000 100000;bootm
40008000 41000000
```

```
bootdelay=7
```

```
ethaddr=00:40:5c:26:0a:5b
```

```
gatewayip=192.168.0.1
```

```
ipaddr=192.168.0.20
```

```
myvar=abc=123 bcd=456 root=/dev/nfs
```

```
netmask=255.255.255.0
```

```
serverip=192.168.0.10
```

```
Environment size: 408/16380 bytes
```

```
TINY4412 #
```

## 2) 删除新增加的环境变量。

```
TINY4412 # setenv myvar
```

```
TINY4412 # printenv
```

```
baudrate=115200
```

```
bootargs=root=/dev/mmcblk0p2 rootfstype=ext3 console=ttySAC0,115200 init=/linuxrc
```

```
uhost0=y ctp=2 skipcali=y lcd=S70
```

```
bootcmd=movi read kernel 0 40008000;movi read rootfs 0 41000000 100000;bootm  
40008000 41000000
```

```
bootdelay=7
```

```
ethaddr=00:40:5c:26:0a:5b
```

```
gatewayip=192.168.0.1
```

```
ipaddr=192.168.0.20
```

```
netmask=255.255.255.0
```

```
serverip=192.168.0.10
```

```
Environment size: 370/16380 bytes
```

```
TINY4412 # saveenv
```

```
Saving Environment to SMDK bootable device...
```

```
done
```

```
TINY4412 #
```

## 4.6 引用环境变量

### 引用环境变量

```
TINY4412 # setenv time 5
```

```
TINY4412 # setenv bootdelay ${time}
```

```
TINY4412 # printenv
```

```
baudrate=115200
```

```
bootargs=root=/dev/mmcblk0p2 rootfstype=ext3 console=ttySAC0,115200 init=/linuxrc
```

```
uhost0=y ctp=2 skipcali=y lcd=S70
```

```
bootcmd=movi read kernel 0 40008000;movi read rootfs 0 41000000 100000;bootm  
40008000 41000000
```

```
bootdelay=5
```

```
ethaddr=00:40:5c:26:0a:5b
```

```
gatewayip=192.168.0.1
```

```
ipaddr=192.168.0.20
```

```
netmask=255.255.255.0
```

```
serverip=192.168.0.10
```

```
time=5
```

Environment size: 379/16380 bytes

TINY4412 #

```
TINY4412 # setenv time 5
TINY4412 # setenv bootdelay ${time}
TINY4412 # printenv
baudrate=115200
bootargs=root=/dev/mmcblk0p2 rootfstype=ext3 console=ttySAC0,115200 init=/linuxrc uhost0=y ctp=2 skipcali=y
lcd=S70
bootcmd=movi read kernel 0 40008000;movi read rootfs 0 41000000 100000;bootm 40008000 41000000
bootdelay=5
ethaddr=00:40:5c:26:0a:5b
gatewayip=192.168.0.1
ipaddr=192.168.0.20
netmask=255.255.255.0
serverip=192.168.0.10
time=5

Environment size: 379/16380 bytes
TINY4412 #
```

**ping** : 检测主机网络是否连通

说明: **ping ip** 地址就检测你的开发板和目标 IP 主机是否连通, 可以通过 PC 下使用 **ping** 命令来理解, 下面是拼百度的 IP.

```
C:\Users\zhifachen>ping 119.75.218.77

正在 Ping 119.75.218.77 具有 32 字节的数据:
来自 119.75.218.77 的回复: 字节=32 时间=45ms TTL=53
来自 119.75.218.77 的回复: 字节=32 时间=47ms TTL=53
来自 119.75.218.77 的回复: 字节=32 时间=47ms TTL=53
来自 119.75.218.77 的回复: 字节=32 时间=61ms TTL=53

119.75.218.77 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 45ms, 最长 = 61ms, 平均 = 50ms
```

在开发板使用 **ping** 的方法是一样的。

TINY4412 # **ping 192.168.1.30**

Tiny4412 u-boot 不支持 USB 网卡, 这个不可用。

## 5 reset : 复位 CPU

作用是和按下复位按键一样。

TINY4412 # **reset**

resetting ...

reset...

OK

U-Boot 2010.12 (Nov 19 2014 - 15:59:58) for TINY4412



CPU: S5PC220 [Samsung SOC on SMP Platform Base on ARM CortexA9]  
APLL = 1400MHz, MPLL = 800MHz

Board: TINY4412  
DRAM: 1023 MiB  
..... 省略

## 6 mmcinfo : 查看 sd/eMMC 设备的信息

查看命令使用方法:

TINY4412 # **help mmcinfo**

mmcinfo - mmcinfo <dev num>-- display MMC info

Usage:

mmcinfo ->输出 **mmc0** 的信息

mmcinfo **<dev num>** ->输出指定编号 mmc 的信息, **<dev num>**是要指定的编号

*编号说明: mmc 的编号是会变化的, Tiny4412 板上有 EMMC, 有 SD 卡, 这两个都归类为 MMC。编号是 0, 1。但是谁是 0, 谁是 1, 是不确定的, 和启动方式有关。在哪个启动, 哪个就是编号 0。*

当前是 SD 卡启动:

TINY4412 # **mmcinfo**

Device: S3C\_HSMMC2

Manufacturer ID: 2

OEM: 544d

Name: SA04G

Tran Speed: 0

Rd Block Len: 512

SD version 2.0

High Capacity: Yes

**Size: 3724MB (block: 7626752)**

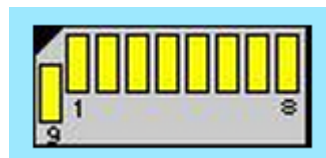
**Bus Width: 4-bit** SD 接口是 4 线的。

Boot Partition Size: 0 KB

补充: SD 接口定义:

表 1 SD 卡引脚功能

引脚号	名 称	功能(SD 模式)	功能(SPI 模式)
1	DAT3/CS	数据线 3	片选/从选(SS)
2	CMD/DI	命令线	主出从入(MOSI)
3	VSS1	电源地	电源地
4	VDD	电源	电源
5	CLK	时钟	时钟(SCK)
6	VSS2	电源地	电源地
7	DAT0/DO	数据线 0	主入从出(MISO)
8	DAT1/IRQ	数据线 1	保留
9	DAT2/NC	数据线 2	保留



查看 mmc0 信息，由于当前是 SD 启动，所以得到的信息是 SD 卡的。

**TINY4412 # mmcinfo 0**

Device: S3C\_HSMMC2

**Manufacturer ID: 2**

OEM: 544d

**Name: SA04G**

Tran Speed: 0

Rd Block Len: 512

**SD version 2.0**

**High Capacity: Yes**

**Size: 3724MB (block: 7626752)**

**Bus Width: 4-bit**

Boot Partition Size: 0 KB

---SD 卡的协议版本号

---是否是高速卡

---SD 卡的容量

---SD 卡的总线线宽

查看 MMC1 信息：当前是 SD 启动，所以返回的信息是 eMMC 的信息。

**TINY4412 # mmcinfo 1**

Device: S5P\_MSHC4

**Manufacturer ID: 15**

OEM: 100

**Name: M4G1Y**

Tran Speed: 0

Rd Block Len: 512

**MMC version 4.0**

**High Capacity: Yes**

**Size: 3728MB (block: 7634944)**

**Bus Width: 8-bit**

Boot Partition Size: 2048 KB

TINY4412 #

---MMC 卡协议版本

---MMC 卡是否高速卡

---MMC 卡的容量（每个块 512 字节）

--MMC 卡的总线线宽

## 7 mmc : mmc 子系统相关命令

这里不是一个命令，是一个子系统，有多个命令。

查看 mmc 帮助：

```
TINY4412 # help mmc
```

```
mmc - MMC sub system
```

Usage:

```
mmc read <device num> addr blk# cnt --从 mmc 指定扇区读取数据到 ddr 中
mmc write <device num> addr blk# cnt --写 ddr 中的数据到指定 mmc 扇区中
mmc rescan <device num> --重新扫描指定设备，相当于重新初始化。
mmc erase <boot | user> <device num> <start block> <block count> --擦除指定扇区
mmc list - lists available devices --列出有效的 mmc 设备
```

命令说明：read：读；write：写；rescan：扫描；erase：擦除。list：mmc 列表

参数说明：

<device num>: mmc 编号，编号原则同前面说的。

addr: DDR3 内存地址；

blk#: 要读/写的 mmc 扇区地址起始地址；

cnt: 要读/写的 mmc 扇区数量；

boot: 引用分区，一般是操作 bl1,bl2,u-boot 的 mmc 扇区范围。(boot 是有一个地址范围)。

user:用户分区，一般是操作内核，文件系统的 mmc 扇区范围。(user 是有一个地址范围)

<start block>:要擦除的 mmc 扇区起始地址；

<block count>: 要擦除的 mmc 扇区数量；

**mmc 命令中的参数都是 16 进制表示，不是 10 进制表示**

### 7.1 mmc list: 列出有效的 mmc 设备

```
TINY4412 # mmc list
```

```
S3C_HSMMC2: 0 --- 0 编号的 mmc 设备，这里接 SD 卡
```

```
S5P_MSHC4: 1 --- 1 编号的 mmc 设备，这里接 eMMC 卡
```

### 7.2 mmc rescan 重新扫描指定设备

**相当于重新初始化。**

```
TINY4412 # mmc rescan 0
```

### 7.3 mmc read : 从 mmc 指定扇区读取数据到 ddr 中

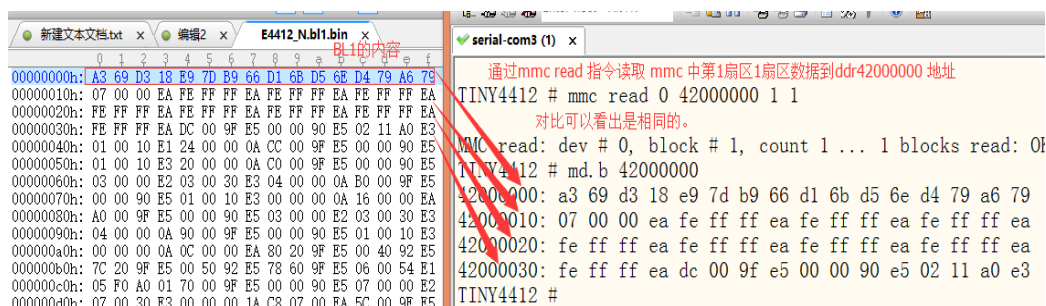
TINY4412 # **mmc read 0 42000000 1 1**

MMC read: dev # 0, block # 1, count 1 ... 1 blocks read: OK

**md.b** :以字节方式显示指定地址的数据

TINY4412 # **md.b 42000000**

```
42000000: a3 69 d3 18 e9 7d b9 66 d1 6b d5 6e d4 79 a6 79      .i...}.f.k.n.y.y
42000010: 07 00 00 ea fe ff ff ea fe ff ff ea fe ff ff ea      .....
42000020: fe ff ff ea fe ff ff ea fe ff ff ea fe ff ff ea      .....
42000030: fe ff ff ea dc 00 9f e5 00 00 90 e5 02 11 a0 e3      .....
TINY4412 #
```



### 7.4 mmc write --写 ddr 中的数据到指定 mmc 扇区中

先读取要测试目标扇区数据内容:

TINY4412 # **mmc read 0 42000000 3e8 1**

MMC read: dev # 0, block # 1000, count 1 ... 1 blocks read: OK

显示目标扇区当前的数据:

TINY4412 # **md.b 42000000**

```
42000000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      .....
42000010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      .....
42000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      .....
42000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      .....
TINY4412 # mmc read 0 42000000 1 1
```

读取 SD 卡中的 BL1 数据的前面 512 字节:

TINY4412 # **mmc read 0 42000000 1 1**

MMC read: dev # 0, block # 1, count 1 ... 1 blocks read: OK

为了和后面操作做对比, 先把它显示出来:

TINY4412 # **md.b 42000000**

```

42000000: a3 69 d3 18 e9 7d b9 66 d1 6b d5 6e d4 79 a6 79    .i...}.f.k.n.y.y
42000010: 07 00 00 ea fe ff ff ea fe ff ff ea fe ff ff ea    .....
42000020: fe ff ff ea fe ff ff ea fe ff ff ea fe ff ff ea    .....
42000030: fe ff ff ea dc 00 9f e5 00 00 90 e5 02 11 a0 e3    .....
TINY4412 #

```

把读取到内存中的数据写入:

```
TINY4412 # mmc write 0 42000000 3e8 1
```

MMC write: dev # 0, block # 1000, count 1 ... 1 blocks written: OK

读取目标扇区数据:

```
TINY4412 # mmc read 0 42000010 3e8 1
```

MMC read: dev # 0, block # 1000, count 1 ... 1 blocks read: OK]

显示读取到的数据:

为了和前面的 42000000 数据区分, 这里把目标地址修改为 42000010

```
TINY4412 # md.b 42000010
```

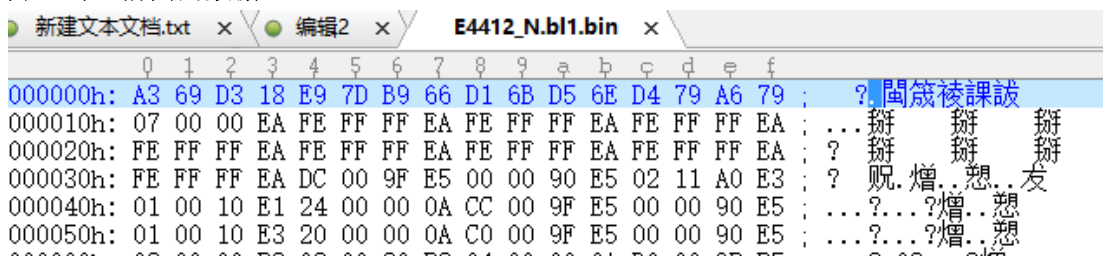
```

42000010: a3 69 d3 18 e9 7d b9 66 d1 6b d5 6e d4 79 a6 79    .i...}.f.k.n.y.y
42000020: 07 00 00 ea fe ff ff ea fe ff ff ea fe ff ff ea    .....
42000030: fe ff ff ea fe ff ff ea fe ff ff ea fe ff ff ea    .....
42000040: fe ff ff ea dc 00 9f e5 00 00 90 e5 02 11 a0 e3    .....

```

TINY4412 #

和面 42000000 的数据从对比, 可以发现是相同的。也可以直接把 BL1 文件使用 UE 打开, 对比前面的数据。



```

新建文本文档.txt x 编辑2 x E4412_N.bl1.bin x
0 1 2 3 4 5 6 7 8 9 a b c d e f
000000h: A3 69 D3 18 E9 7D B9 66 D1 6B D5 6E D4 79 A6 79 ; ? 閩箴裱課諒
000010h: 07 00 00 EA FE FF FF EA FE FF FF EA FE FF FF EA ; ... 癡 癡 癡
000020h: FE FF FF EA FE FF FF EA FE FF FF EA FE FF FF EA ; ? 癡 癡 癡
000030h: FE FF FF EA DC 00 9F E5 00 00 90 E5 02 11 A0 E3 ; ? 貺. 癡.. 癡.. 友
000040h: 01 00 10 E1 24 00 00 0A CC 00 9F E5 00 00 90 E5 ; ... ?...? 癡.. 癡
000050h: 01 00 10 E3 20 00 00 0A C0 00 9F E5 00 00 90 E5 ; ... ?...? 癡.. 癡

```

## 7.5 cp 内存拷贝

```
TINY4412 # help cp
```

cp - memory copy

Usage:

```
cp [.b, .w, .l] source target count
```

```
TINY4412 # cp.b 40000000 42000000 10
```

```
TINY4412 #
```

## 7.6 mmc erase 擦除指定扇区

`mmc erase <boot | user> <device num> <start block> <block count>` --擦除指定扇区

- 1) 先读取 5 扇区的内容，显示出来：

```
TINY4412 # mmc read 0 42000010 5 1
```

```
mmc erase boot 0 5 1
```

```
MMC read: dev # 0, block # 1000, count 1 ... 1 blocks read: OK
```

```
TINY4412 # md.b 42000010
```

```
42000010: a3 69 d3 18 e9 7d b9 66 d1 6b d5 6e d4 79 a6 79 .i...}.f.k.n.y.y
```

```
42000020: 07 00 00 ea fe ff ff ea fe ff ff ea fe ff ff ea .....y.y
```

```
42000030: fe ff ff ea fe ff ff ea fe ff ff ea fe ff ff ea .....y.y
```

```
42000040: fe ff ff ea dc 00 9f e5 00 00 90 e5 02 11 a0 e3 .....y.y
```

```
TINY4412 #
```

```
TINY4412 # mmc read 0 42000010 3e8 1
```

```
MMC read: dev # 0, block # 1000, count 1 ... 1 blocks read: OK
```

```
TINY4412 # md.b 42000010
```

```
42000010: a3 69 d3 18 e9 7d b9 66 d1 6b d5 6e d4 79 a6 79 .y.y
```

```
42000020: 07 00 00 ea fe ff ff ea fe ff ff ea fe ff ff ea ...
```

```
42000030: fe ff ff ea fe ff ff ea fe ff ff ea fe ff ff ea ...
```

```
42000040: fe ff ff ea dc 00 9f e5 00 00 90 e5 02 11 a0 e3 ...
```

```
TINY4412 #
```

- 2) 擦除 5 扇区的内容：

```
mmc erase boot 0 5 1
```

- 3) 再次读取 5 扇区的内容，显示出来：

- 4) 对比两操作结果

## 8 fatinfo 显示接口的文件系统信息

```
TINY4412 # help fatinfo
```

```
fatinfo - fatinfo - print information about filesystem
```

Usage:

```
fatinfo <interface> <dev[:part]>
```

```
- print information about filesystem from 'dev' on 'interface'
```

参数说明：

<interface>: mmc 或 usb;

dev: 设备编号;  
part: 设备分区号;

测试 SD 卡启动时候 mmc 文件系统信息:

**TINY4412 # fatinfo mmc 0**

-----Partition 1-----

Partition1: Start Address(0x4b800), Size(0x6fa800)

-----

-----Partition 2-----

Partition1: Start Address(0x4b800), Size(0x6fa800)

-----

-----Partition 3-----

Partition1: Start Address(0x4b800), Size(0x6fa800)

-----

-----Partition 4-----

Partition1: Start Address(0x4b800), Size(0x6fa800)

-----

Interface: SD/MMC

Device 0: Vendor: Man 02544d Snr 417b1ea3 Rev: 1.0 Prod: SA04G

Type: Removable Hard Disk

Capacity: 7.2 MB = 0.0 GB (14896 x 512)

**Partition 1: Filesystem: FAT32 "NO NAME"**

TINY4412 #

## 9 fatls -列出指定目录下的文件

TINY4412 # help fatls

**fatls - list files in a directory (default /)**

Usage:

fatls <interface> <dev[:part]> [directory]

- list files from 'dev' on 'interface' in a 'directory'

参数说明:

**<interface>:** mmc 或 usb;

**dev:** 设备编号;

**part:** 设备分区号;

**[directory]:** 目录, 是可选, 可以不写, 不写默认 / 目录

查看 SD 启动, SD 卡中的文件列表:

查看 / 文件列表:

TINY4412 # **fatls mmc 0 /**

Partition1: Start Address(0x4b800), Size(0x6fa800)

```

        system volume information/
0    1.txt
    dir1/
0    mp
0    icrosoft word ocx

```

3 file(s), 2 dir(s)

查看子目录不能使用/开头，否则会出错：

**TINY4412 # fatls mmc 0 /dir1**

**\*\* Invalid boot device, use `dev[:part]` \*\***

查看 / 路径下的 **dir1** 文件夹列表：

**TINY4412 # fatls mmc 0 dir1**

Partition1: Start Address(0x4b800), Size(0x6fa800)

```

    ./
    ../
    dir2/

```

0 file(s), 3 dir(s)

查看 / 路径下的 **dir1/dir2** 文件夹列表：

**TINY4412 # fatls mmc 0 dir1/dir2**

Partition1: Start Address(0x4b800), Size(0x6fa800)

```

    ./
    ../
0    3.txt

```

1 file(s), 2 dir(s)

TINY4412 #

## 10 fatload 从一个 DOS 文件系统中加载一个二进制文件

先查看 **fatload** 使用方法：

**TINY4412 # help fatload**

fatload - fatload - load binary file from a dos filesystem

Usage:

fatload <interface> <dev[:part]> <addr> <filename> [bytes]

- load binary file 'filename' from 'dev' on 'interface'

to address 'addr' from dos filesystem

示例：**fatload mmc 0 42000000 E4412\_N.bl1.bin**



**参数说明:****<interface>:** mmc 或 usb;**dev:** 设备编号;**part:** 设备分区号;**[directory]:** 目录, 是可选, 可以不写, 不写默认 / 目录**<addr>:** DDR 内存地址**<filename>:** 要加载二进制文件 (包含完整路径)**[bytes]:** 要加载数据大小, 字节为单位。可选的, 可以不写, 不写时候默认等于文件大小。

把卡取出插入 PC, 复制一些文件到卡里, 然后再重新插入开发板中 (不要断电, 目的想测试一下 **mmc rescan** 命令作用)

先不要重新扫描 **mmc** 设备, 直接输入 **fatls**, 就会出错:

**TINY4412 # fatls mmc 0**

count: 1

# Tx: Inverter delay / Rx: Inverter delay

count: 2

## Tx: Basic delay / Rx: Inverter delay

count: 3

## Tx: Inverter delay / Rx: Basic delay

count: 4

### Tx: Basic delay / Rx: Basic delay

count: 5

# Tx: Disable / Rx: Basic delay

count: 6

## Tx: Disable / Rx: Inverter delay

count: 7

### Tx: Basic delay / Rx: Disable

count: 8

### Tx: Inverter delay / Rx: Disable

mmc read failed ERROR: -19

data.dest: 0xc3cfbbe4

data.blocks: 1

data.blocksize: 512

MMC\_DATA\_READ

\*\* Can't read from device 0 \*\*

**\*\* Unable to use mmc 0:1 for fatls \*\***

接下来重新扫描 mmc0 :

TINY4412 # **mmc rescan 0**

再次读取文件列表:

TINY4412 # **fatls mmc 0**

Partition1: Start Address(0x4b800), Size(0x6fa800)

system volume information/

0 1.txt

392 led.bin

14336 bl2.bin

8192 e4412\_n.bl1.bin

bl1/

bl2/

led/

4 file(s), 4 dir(s)

开始测试 fatload 命令:

TINY4412 # **fatload mmc 0 42000000 E4412\_N.bl1.bin**

Partition1: Start Address(0x4b800), Size(0x6fa800)

reading E4412\_N.bl1.bin

8192 bytes read

TINY4412 # **md.b 42000000**

42000000: a3 69 d3 18 e9 7d b9 66 d1 6b d5 6e d4 79 a6 79 .i...}.f.k.n.y.y

42000010: 07 00 00 ea fe ff ff ea fe ff ff ea fe ff ff ea .....

42000020: fe ff ff ea fe ff ff ea fe ff ff ea fe ff ff ea .....

42000030: fe ff ff ea dc 00 9f e5 00 00 90 e5 02 11 a0 e3 .....

TINY4412 #

```
TINY4412 # md.b 42000000
42000000: a3 69 d3 18 e9 7d b9 66 d1 6b d5 6e d4 79 a6 79 .i...}.f.k.n.y.y
42000010: 07 00 00 ea fe ff ff ea fe ff ff ea fe ff ff ea .....
42000020: fe ff ff ea fe ff ff ea fe ff ff ea fe ff ff ea .....
42000030: fe ff ff ea dc 00 9f e5 00 00 90 e5 02 11 a0 e3 .....
TINY4412 #
```

新建文本文档.txt																编辑2																E4412_N.bl1.bin																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
		0		1		2		3		4		5		6		7		8		9		a		b		c		d		e		f																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				</	

对比以两个数据，完全相同。

## 11 cmp 比较内存数据是否相同

1) 先直接比较两块内存数据

```
TINY4412 # cmp.b 41000000 42000000 10
```

```
byte at 0x41000000 (0x00) != byte at 0x42000000 (0xa3)
```

```
Total of 0 bytes were the same
```

结果显示，第 1 个字节不同了。

2) 在要比较的两块内存中读取相同的数据

```
TINY4412 # mmc read 0 41000000 1 1
```

```
MMC read: dev # 0, block # 1, count 1 ... 1 blocks read: OK
```

```
TINY4412 # mmc read 0 42000000 1 1
```

```
MMC read: dev # 0, block # 1, count 1 ... 1 blocks read: OK
```

3) 再次比较两块内存的数据

```
TINY4412 # cmp.b 41000000 42000000 10
```

```
Total of 16 bytes were the same
```

```
TINY4412 #
```

提示比较数据完全相同。

## 12 mm 地址以自动增加的方式修改内存数据

1) 先查看 mm 使用方法

```
TINY4412 # help mm
```

```
mm - memory modify (auto-incrementing address)
```

```
Usage:
```

```
mm [.b, .w, .l] address
```

```
TINY4412 #
```

2) 测试 mm 使用效果

先把修改前内容打印出来:

```
TINY4412 # md.b 41000000
```

```
41000000: a3 69 d3 18 e9 7d b9 66 d1 6b d5 6e d4 79 a6 79      .i...}.f.k.n.y.y
```

```
41000010: 07 00 00 ea fe ff ff ea fe ff ff ea fe ff ff ea      .....
```

```
41000020: fe ff ff ea fe ff ff ea fe ff ff ea fe ff ff ea      .....
```

```
41000030: fe ff ff ea dc 00 9f e5 00 00 90 e5 02 11 a0 e3      .....
```

修改内容:

**TINY4412 # mm.b 41000000**

```
41000000: a3 ? 1      →把 a3 修改为 1
41000001: 69 ? 2      →把 69 修改为 2
41000002: d3 ?        →直接回车, 不修改
41000003: 18 ? 3      →把 18 修改为 3
41000004: e9 ? TINY4412 # →修改完成了, 按键 Ctrl+c 结束修改
```

查看修改的结果:

**TINY4412 # md.b 41000000**

```
41000000: 01 02 d3 03 e9 7d b9 66 d1 6b d5 6e d4 79 a6 79 .....}.f.k.n.y.y
41000010: 07 00 00 ea fe ff ff ea fe ff ff ea fe ff ff ea .....
41000020: fe ff ff ea fe ff ff ea fe ff ff ea fe ff ff ea .....
41000030: fe ff ff ea dc 00 9f e5 00 00 90 e5 02 11 a0 e3 .....
TINY4412 #
```

和前面修改前对比:

修改的内容:

```
TINY4412 # mm.b 41000000
41000000: a3 ? 1
41000001: 69 ? 2
41000002: d3 ?
41000003: 18 ? 3
41000004: e9 ? TINY4412 #
```

修改前:

```
TINY4412 # md.b 41000000
41000000: a3 69 d3 18 e9 7d b9 66 d1 6b d5 6e d4 79 a6 79
41000010: 07 00 00 ea fe ff ff ea fe ff ff ea fe ff ff ea
41000020: fe ff ff ea fe ff ff ea fe ff ff ea fe ff ff ea
41000030: fe ff ff ea dc 00 9f e5 00 00 90 e5 02 11 a0 e3
```

修改后:

```
TINY4412 # md.b 41000000
41000000: 01 02 d3 03 e9 7d b9 66 d1 6b d5 6e d4 79 a6 79
41000010: 07 00 00 ea fe ff ff ea fe ff ff ea fe ff ff ea
41000020: fe ff ff ea fe ff ff ea fe ff ff ea fe ff ff ea
41000030: fe ff ff ea dc 00 9f e5 00 00 90 e5 02 11 a0 e3
TINY4412 #
```

其他类似命令:

**mm.w:** 一次修改 2 字节

**mm.l:** 一次修改 4 字节

**md** 也有其他类似命令:

**md.w:** 以 2 字节为单位显示

**md.l:** 以 4 字节为单位显示

TINY4412 # md.b 41000000

```
41000000: 01 02 d3 03 e9 7d b9 66 d1 6b d5 6e d4 79 a6 79
41000010: 07 00 00 ea fe ff ff ea fe ff ff ea fe ff ff ea
41000020: fe ff ff ea fe ff ff ea fe ff ff ea fe ff ff ea
41000030: fe ff ff ea dc 00 9f e5 00 00 90 e5 02 11 a0 e3
```

TINY4412 # md.w 41000000

```
41000000: 0201 03d3 7de9 66b9 6bd1 6ed5 79d4 79a6
41000010: 0007 ea00 fffe eaff fffe eaff fffe eaff
41000020: fffe eaff fffe eaff fffe eaff fffe eaff
41000030: fffe eaff 00dc e59f 0000 e590 1102 e3a0
41000040: 0001 e110 0024 0a00 00cc e59f 0000 e590
41000050: 0001 e310 0020 0a00 00c0 e59f 0000 e590
41000060: 0003 e200 0003 e330 0004 0a00 00b0 e59f
41000070: 0000 e590 0001 e310 0000 0a00 0016 ea00
```

TINY4412 # md.l 41000000

```
41000000: 03d30201 66b97de9 6ed56bd1 79a679d4
41000010: ea000007 eaffffff eaffffff eaffffff
41000020: eaffffff eaffffff eaffffff eaffffff
41000030: eaffffff e59f00dc e5900000 e3a01102
41000040: e1100001 0a000024 e59f00cc e5900000
41000050: e3100001 0a000020 e59f00c0 e5900000
41000060: e2000003 e3300003 0a000004 e59f00b0
```

## 13 cp 内存复制

### 1) 查看使用方法

TINY4412 # **help cp**

cp - memory copy

Usage:

cp [.b, .w, .l] source target count

### 2) 先查看两个内存块数据

TINY4412 # md.b 41000000

```
41000000: 01 02 d3 03 e9 7d b9 66 d1 6b d5 6e d4 79 a6 79 .....}.f.k.n.y.y
41000010: 07 00 00 ea fe ff ff ea fe ff ff ea fe ff ff ea .....
41000020: fe ff ff ea fe ff ff ea fe ff ff ea fe ff ff ea .....
41000030: fe ff ff ea dc 00 9f e5 00 00 90 e5 02 11 a0 e3 .....
```

TINY4412 # md.b 42000000

```
42000000: a3 69 d3 18 e9 7d b9 66 d1 6b d5 6e d4 79 a6 79 .i...}.f.k.n.y.y
```

```

42000010: 07 00 00 ea fe ff ff ea fe ff ff ea fe ff ff ea .....
42000020: fe ff ff ea fe ff ff ea fe ff ff ea fe ff ff ea .....
42000030: fe ff ff ea dc 00 9f e5 00 00 90 e5 02 11 a0 e3 .....
两不内存块数据不同。

```

3) 把第 1 块数据前 512 字节复制到第 2 块数据的前 512 字节内存中。

**TINY4412 # cp.b 41000000 42000000 512**

4) 再次查看第 2 块内存数据，并且与第一块内存数据比较

**TINY4412 # md.b 42000000**

```

42000000: 01 02 d3 03 e9 7d b9 66 d1 6b d5 6e d4 79 a6 79 .....}.f.k.n.y.y
42000010: 07 00 00 ea fe ff ff ea fe ff ff ea fe ff ff ea .....
42000020: fe ff ff ea fe ff ff ea fe ff ff ea fe ff ff ea .....
42000030: fe ff ff ea dc 00 9f e5 00 00 90 e5 02 11 a0 e3 .....
TINY4412 #

```

```

TINY4412 # md.b 41000000
41000000: 01 02 d3 03 e9 7d b9 66 d1 6b d5 6e d4 79 a6 79
41000010: 07 00 00 ea fe ff ff ea fe ff ff ea fe ff ff ea
41000020: fe ff ff ea fe ff ff ea fe ff ff ea fe ff ff ea
41000030: fe ff ff ea dc 00 9f e5 00 00 90 e5 02 11 a0 e3
TINY4412 # md.b 42000000
42000000: a3 69 d3 18 e9 7d b9 66 d1 6b d5 6e d4 79 a6 79
42000010: 07 00 00 ea fe ff ff ea fe ff ff ea fe ff ff ea
42000020: fe ff ff ea fe ff ff ea fe ff ff ea fe ff ff ea
42000030: fe ff ff ea dc 00 9f e5 00 00 90 e5 02 11 a0 e3
TINY4412 # cp.b 41000000 42000000 512
TINY4412 # md.b 42000000
42000000: 01 02 d3 03 e9 7d b9 66 d1 6b d5 6e d4 79 a6 79
42000010: 07 00 00 ea fe ff ff ea fe ff ff ea fe ff ff ea
42000020: fe ff ff ea fe ff ff ea fe ff ff ea fe ff ff ea
42000030: fe ff ff ea dc 00 9f e5 00 00 90 e5 02 11 a0 e3
TINY4412 #

```

比较结果是相同的。

## 14 loady 使用串口下载二进制数据到内存中

1) 查看使用方法

**TINY4412 # help loady**

loady - load binary file over serial line (ymodem mode)

Usage:

loady [ off ] [ baud ]

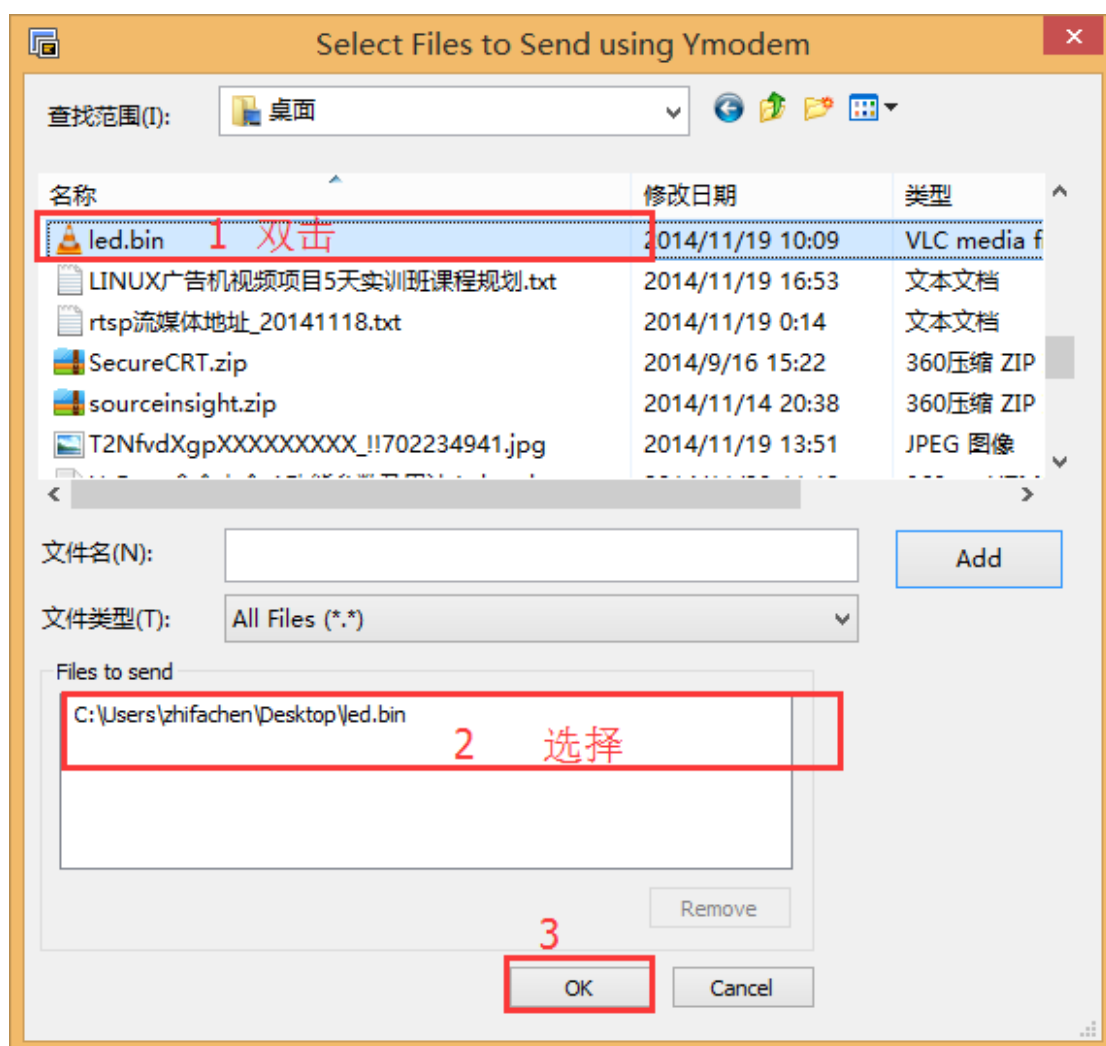
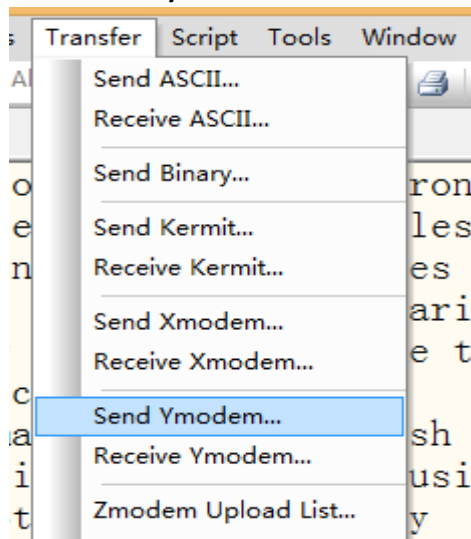
- load binary file over serial line with offset 'off' and baudrate 'baud'

参数说明:

[ off ]: DDR 内存地址, 可选。

[ baud ]: 使用多快的波特率下载, 可选。

## 2) 测试 loady



```

Starting ymodem transfer. Press Ctrl+C to cancel.
Transferring E4412_N.bl1.bin...
  100%      8 KB      1 KB/sec    00:00:06      0 Errors

## Total Size      = 0x00002000 = 8192 Bytes
TINY4412 # █

TINY4412 # loady
## Ready for binary (ymodem) download to 0xC3E00000 at 0 bps...
CCCC
Starting ymodem transfer. Press Ctrl+C to cancel.
Transferring led.bin...
  100%     392 bytes   98 bytes/sec 00:00:04      1 Errors

## Total Size      = 0x00000188 = 392 Bytes

```

## 15 go – CPU 跳转到指定的地址执行代码

一旦 **go** 指令执行后，**u-boot** 就不存在了，它的使命就完成了。  
 直接在上一步基础输入 以下指令 **CPU** 就会去执行指定地址处的代码。

```

TINY4412 # go 0xC3E00000
## Starting application at 0xC3E00000 ...

```

以上下载的是 **led.bin**，按下按键可以点亮 **LED**。

其他的 **loadx/z** 指令自己测试。

**xmodem**:简单通用，传输信息单位是“包=128B”,传输速度慢，适合电话线路质量差的情况下用

**ymodem**:由 **XMODEM** 演变来，效率可靠性高，包=128\*8B；一次传输可发送或接受几个文件

**zmodem**:于上两种不同，已连续的数据流发送数据，效率更高

以下两命令：**fdisk**，**movi** 很重要，暂时不讲，以后产品发布时候讲。

```

TINY4412 # fdisk
Usage:
fdisk <-p> <device_num>
fdisk <-c> <device_num> [<sys. part size(MB)> <user data part size> <cache part size>]

TINY4412 # movi
Usage:
movi      - sd/mmc r/w sub system for SMDK board

TINY4412 #

```



```

TINY4412 # pri
baudrate=115200 从0设备读内核到0x40008000内存, 从0设备读文件系统到0x41000000内存, 读100000大小
bootcmd=movi read kernel 0 40008000;movi read rootfs 0 41000000 100000;bootm 40008000 41000000
bootdelay=5 从0x40008000启动内核, 从0x41000000启动文件系统
ethaddr=00:40:5c:26:0a:5b
gatewayip=192.168.1.1
ipaddr=192.168.1.99
netmask=255.255.255.0
serverip=192.168.1.30

```

movi 就是从 sd/mmc 里读数据到内存里。

比如 bootcmd=movi read kernel 30008000; 就是从 sd/mmc 读偏移量为 kernel 的地方读数据到内存 30008000 中去。

movi read rootfs 30B00000 300000;同样是从 sd/mmc 读偏移量为 rootfs 的地方读数据到内存 30B00000 中去。后面 300000 是读数据的多少;

## 16 bootargs (了解)

U-boot 的环境变量值得注意的有两个: bootcmd 和 bootargs。

u-bootcmd

前面有说过 bootcmd 是自动启动时默认执行的一些命令, 因此你可以在当前环境中定义各种不同配置, 不同环境的参数设置, 然后设置 bootcmd 为你经常使用的那种参数。

u-bootargs

bootargs 是环境变量中的重中之重, 甚至可以说整个环境变量都是围绕着 bootargs 来设置的。bootargs 的种类非常非常的多, 我们平常只是使用了几种而已。bootargs 非常的灵活, 内核和文件系统的不同搭配就会有不同的设置方法, 甚至你也可以不设置 bootargs, 而直接将其写到内核中去 (在配置内核的选项中可以进行这样的设置, 了解), 正是这些原因导致了 bootargs 使用上的困难。

下面介绍一下 bootargs 常用参数, bootargs 的种类非常的多, 而且随着 kernel 的发展会出现一些新的参数, 使得设置会更加灵活多样。

例子 1:

```

setenv bootargs root=/dev/nfs nfsroot=192.168.1.127:/rootfs
ip=192.168.1.99:192.168.1.127:192.168.1.1:255.255.255.0:www.arm.com:eth0:off
console=ttySAC0

```

/\* \*\*\*\*\*解释\*\*\*\*\* \*/

bootargs 变量为后面所有的字符串; root 就是 “/dev/nfs” 字符串变量, 表示用网络文件系统; nfsroot 就是 “192.168.1.127:/rootfs” 字符串, 192.168.1.127 是网络文件系统服务器 (就是虚拟机) 的 ip; ip 就是 “= ” 后面的所有字符串, 192.168.1.99 是板子 linux 系统的 ip, 192.168.1.127 是 nfs 服务器的 ip, 192.168.1.1 是网关, 255.255.255.0 是子网掩码, www.arm.com 字符串可任意写, eth0 是 nfs 服务器的网卡, off 结束; console 是虚拟机的串口终端。

\*\*\*\*\* \*/

**例子 2:**

```
bootargs=root=/dev/mmcblk0p2 rootfstype=ext3 console=ttySAC0,115200 init=/linuxrc
uhost0=y ctp=2 skipcali=y lcd=S70
```

**A. root** :用来指定 rootfs 的位置， 常见的情况有:

```
root=/dev/ram rw
root=/dev/ram0 rw
root=/dev/mtdx rw
root=/dev/mtdblockx rw
root=/dev/mtdblock/x rw
root=31:0x
```

上面的这几个在一定情况下是通用的，当然这要看你当前的系统是否支持，不过 mtd 是字符设备，而 mtdblock 是块设备，有时候你的挨个的试到底当前的系统支持上面那种情况下，不过 root=/dev/mtdblockx rw 比较通用。此外，如果直接指定设备名可以的话，那么使用此设备的设备号也是可以的。

```
root=/dev/nfs
```

在文件系统为基于 nfs 的文件系统的时候使用。当然指定 root=/dev/nfs 之后，还需要指定 nfsroot=serverip:nfs\_dir，即指明文件系统存在那个主机的那个目录下。

**B. rootfstype** :这个选项需要跟 root 一起配合使用，一般如果根文件系统是 ext2 的话，有没有这个选项是无所谓的，但是如果是 jffs2,squashfs 等文件系统的话，就需要 rootfstype 指明文件系统的类型，不然会无法挂载根分区。

**C. console** : console=tty 使用虚拟串口终端设备。

console=ttyS[,options] 使用特定的串口，options 可以是这样的形式 bbbbpnx，这里 bbbb 是指串口的波特率，p 是奇偶位（从来没有看过使用过），n 是指的 bits。

console=ttySAC[,options] 同上面。

看你当前的环境，有时用 ttyS，有时用 ttySAC，网上有人说，这是跟内核的版本有关，2.4 用 ttyS，2.6 用 ttySAC，但实际情况是官方文档中也是使用 ttyS，所以应该是跟内核版本没有关联的。可以查看 Documentation/serial-console.txt 找到相关描述。

**D. mem**: mem=xxM 指定内存的大小，不是必须的

**E. ramdisk\_size**

```
ramdisk=xxxxx      不推荐
ramdisk_size=xxxxx  推荐
```

上面这两个都可以告诉 ramdisk 驱动，创建的 ramdisk 的 size，默认情况下是 4m(s390 默认 8M)，你可以查看 Documentation/ramdisk.txt 找到相关的描述，不过 ramdisk=xxxxx 在新版的内核都已经没有提了，不推荐使用。

**F. initrd, noinitrd**: 当你没有使用 ramdisk 启动系统的时候，你需要使用 noinitrd 这个参数，但是如果使用了的话，就需要指定 initrd=r\_addr,size, r\_addr 表示 initrd 在内存中的位置，size 表示 initrd 的大小。

**G. init**: init 指定的是内核启起来后，进入系统中运行的第一个脚本，一般 init=/linuxrc，或者 init=/etc/preinit，preinit 的内容一般是创建 console,null 设备节点，运行 init 程序，挂载一些文件系统等等操作。请注意，很多初学者以为 init=/linuxrc 是固定写法，其实不然，/linuxrc 指的是/目录下面的 linuxrc 脚本，一般是一个连接罢了。

**H. mtdparts**

```
mtdparts=fc000000.nor_flash:1920k(linux),128k(fdt),20M(ramdisk),4M(jffs2),38272k(user),256k(
```

env),384k(uboot)

要想这个参数起作用，内核中的 mtd 驱动必须要支持，即内核配置时需要选上 Device Drivers  
---> Memory Technology Device (MTD) support ---> Command line partition table parsing  
mtdparts 的格式如下：

mtdparts=[:

:= [:]

:= [@offset][ro]

:= unique id used in mapping driver/device

:= standard linux memsize OR "-" to denote all remaining space

:= (NAME)

因此你在使用的时候需要按照下面的格式来设置：

mtdparts=mtd-id:@(),@()

这里面有几个必须要注意的：

- mtd-id 必须要跟你当前平台的 flash 的 mtd-id 一致，不然整个 mtdparts 会失效
- size 在设置的时候可以为实际的 size(xxM,xxk,xx)，也可以为 '-' 这表示剩余的所有空间。

举例：

假设 flash 的 mtd-id 是 sa1100，那么你可以使用下面的方式来设置：

mtdparts=sa1100:- → 只有一个分区

mtdparts=sa1100:256k(ARMboot)ro,-(root) → 有两个分区

可以查看 drivers/mtd/cmdlinepart.c 中的注释找到相关描述。

## 1. ip

指定系统启动之后网卡的 ip 地址，如果你使用基于 nfs 的文件系统，那么必须要有这个参数，其他的情况下就看你自己的喜好了。设置 ip 有两种方法：

ip = ip addr

ip=ip addr:server ip addr:gateway:netmask::which netcard:off

这两种方法可以用，不过很明显第二种要详细很多，请注意第二种中 which netcard 是指开发板上的网卡，而不是主机上的网卡。

说完常见的几种 bootargs，那么我们来讨论平常我经常使用的几种组合：

- 1). 假设文件系统是 ramdisk（虚拟内存盘），且直接就在内存中，bootargs 的设置应该如下：

```
setenv bootargs 'initrd=0x32000000,0xa00000 root=/dev/ram0 console=ttySAC0 mem=64M
init=/linuxrc'
```

- 2). 假设文件系统是 ramdisk，且在 flash 中，bootargs 的设置应该如下：

```
setenv bootargs 'mem=32M console=ttyS0,115200 root=/dev/ram rw init=/linuxrc'
```

注意这种情况下你应该要在 bootm 命令中指定 ramdisk 在 flash 中的地址，如 bootm  
kernel\_addr ramdisk\_addr (fdt\_addr)

- 3). 假设文件系统是 jffs2（闪存日志型文件系统第 2 版）类型的，且在 flash 中，bootargs 的设置应该如下

```
setenv bootargs 'mem=32M console=ttyS0,115200 noinitrd root=/dev/mtdblock2 rw
rootfstype=jffs2 init=/linuxrc'
```

4). 假设文件系统是基于 nfs 的, bootargs 的设置应该如下

```
setenv bootargs 'noinitrd mem=64M console=ttySAC0 root=/dev/nfs nfsroot=192.168.0.3:/nfs
ip=192.168.0.5:192.168.0.3:192.168.0.3:255.255.255.0::eth0:off'
```

或者

```
setenv bootargs 'noinitrd mem=64M console=ttySAC0 root=/dev/nfs nfsroot=192.168.0.3:/nfs
ip=192.168.0.5'
```

上面就是我们经常使用的几种 bootargs 的组合, 老实说, bootargs 非常非常的灵活, 所以设置的方法有很多中形式, 具体的还应该根据你的平台具体的情况来设置。

## 17 综合: 设置开机自动加载 sd 卡中运行 led 程序

```
TINY4412 # setenv bootcmd 'fatload mmc 0 0x02020000 led.bin;go 0x02020000'
```

```
TINY4412 # pri
```

```
baudrate=115200
```

```
bootcmd=fatload mmc 0 0x02020000 led.bin;go 0x02020000
```

```
bootdelay=3
```

```
ethaddr=00:40:5c:26:0a:5b
```

```
gatewayip=192.168.0.1
```

```
ipaddr=192.168.0.20
```

```
netmask=255.255.255.0
```

```
serverip=192.168.0.10
```

```
Environment size: 212/16380 bytes
```

```
TINY4412 # save
```

```
Saving Environment to SMDK bootable device...
```

```
done
```

```
TINY4412 # reset
```

## 17 把 sd 卡中的 bl1,bl2,u-boot 烧写到 emm 中

```
TINY4412 # movi r f 0 40000000;emmc open 1;movi w z f 1 40000000;emmc close 1;
```

```
reading FWBL1 ..device 0 Start 1, Count 16
```

```
MMC read: dev # 0, block # 1, count 16 ... 16 blocks read: OK
```

```
completed
```

```
eMMC OPEN Success.!!
```

!!!Notice!!!

!You must close eMMC boot Partition after all image writing!

!eMMC boot partition has continuity at image writing time.!

!So, Do not close boot partition, Before, all images is written.!

**writing FWBL1** ..device 1 Start 0, Count 16

MMC write: dev # 1, block # 0, count 16 ... 16 blocks written: OK

completed

eMMC CLOSE Success.!!

**TINY4412 # movi r b 0 40000000;emmc open 1;movi w z b 1 40000000;emmc close 1;**

**reading BL2** ..device 0 Start 17, Count 32

MMC read: dev # 0, block # 17, count 32 ... 32 blocks read: OK

completed

eMMC OPEN Success.!!

!!!Notice!!!

!You must close eMMC boot Partition after all image writing!

!eMMC boot partition has continuity at image writing time.!

!So, Do not close boot partition, Before, all images is written.!

**writing BL2** ..device 1 Start 16, Count 32

MMC write: dev # 1, block # 16, count 32 ... 32 blocks written: OK

completed

eMMC CLOSE Success.!!

**TINY4412 # movi r u 0 40000000;emmc open 1;movi w z u 1 40000000;emmc close 1;**

**reading bootloader**..device 0 Start 49, Count 656

MMC read: dev # 0, block # 49, count 656 ... 656 blocks read: OK

completed

eMMC OPEN Success.!!

!!!Notice!!!

!You must close eMMC boot Partition after all image writing!

!eMMC boot partition has continuity at image writing time.!

!So, Do not close boot partition, Before, all images is written.!

**writing bootloader**..device 1 Start 48, Count 656

MMC write: dev # 1, block # 48, count 656 ... 656 blocks written: OK

completed

eMMC CLOSE Success.!!

**TINY4412 # movi r t 0 40000000;emmc open 1;movi w z t 1 40000000;emmc close 1;**

**reading 0 TrustZone** S/W.. Start 705, Count 320

MMC read: dev # 0, block # 705, count 320 ... 320 blocks read: OK

completed

eMMC OPEN Success.!!

!!!Notice!!!

!You must close eMMC boot Partition after all image writing!

!eMMC boot partition has continuity at image writing time.!

!So, Do not close boot partition, Before, all images is written.!

**writing 1 TrustZone** S/W.. Start 704, Count 320

MMC write: dev # 1, block # 704, count 320 ... 320 blocks written: OK

completed

eMMC CLOSE Success.!!

TINY4412 # OK

**从 SD 启动 uboot，进入 uboot 命令：**

把 sd 卡中 u-boot 的第一阶段的 bl0 数据复制到内存，然后再写入 emmc 对应位置

```
movi r f 0 40000000;emmc open 1;movi w z f 1 40000000;emmc close 1;
```

把 sd 卡中 u-boot 数据复制到内存，然后再写入 emmc 对应位置

```
movi r b 0 40000000;emmc open 1;movi w z b 1 40000000;emmc close 1;
```

把 sd 卡中 u-boot 数据复制到内存，然后再写入 emmc 对应位置

```
movi r u 0 40000000;emmc open 1;movi w z u 1 40000000;emmc close 1;
```

把 sd 卡中 u-boot 安全加密数据复制到内存，然后再写入 emmc 对应位置

```
movi r t 0 40000000;emmc open 1;movi w z t 1 40000000;emmc close 1;
```

把 sd 卡中内核数据复制到内存，然后再写入 emmc 对应位置

```
movi r k 0 40000000;movi w k 1 40000000;
```

**USB 操作指令**

指令	功能
usb start	启动 USB 功能
usb reset	初始化 USB 控制器
usb stop [f]	关闭 USB 控制器
usb tree	已连接的 USB 设备树
usb info [dev]	显示 USB 设备[dev]的信息
usb storage	显示已连接的 USB 存储设备
usb dev [dev]	显示和设置当前 USB 存储设备
usb part [dev]	显示 USB 存储设备[dev]的分区信息
usb read addr blk# cnt	读取 USB 存储设备数据

在所有的命令使用前，必须先插入 **USB** 设备，然后使用：**usb reset**，以初始化 **USB** 控制器，获取设备信息。

我将一个 4G 的 kingstonU 盘（可引导盘）插入开发板，然后读取他的头 512 字节（MBR）：

```
[u-boot@MINI2440]# usb reset
```

```
(Re)start USB...
```

```
USB: scanning bus for devices... 2 USB Device(s) found
```

```
scanning bus for storage devices... 1 Storage Device(s) found
```

```
[u-boot@MINI2440]# usb tree
```

```
Device Tree:
```

```
1 Hub (12 Mb/s, 0mA)
```

```
| OHCI Root Hub
```

```
|
```

```
+--2 Mass Storage (12 Mb/s, 100mA)
```

```
Kingston DT 101 II 0019E02CB6EB5B8B1B120051
```

```
[u-boot@MINI2440]# usb info
1: Hub, USB Revision 1.10
- OHCI Root Hub
- Class: Hub
- PacketSize: 8 Configurations: 1
- Vendor: 0x0000 Product 0x0000 Version 0.0
  Configuration: 1
    - Interfaces: 1 Self Powered 0mA
      Interface: 0
        - Alternate Setting 0, Endpoints: 1
        - Class Hub
        - Endpoint 1 In Interrupt MaxPacket 2 Interval 255ms

2: Mass Storage, USB Revision 2.0
- Kingston DT 101 II 0019E02CB6EB5B8B1B120051
- Class: (from Interface) Mass Storage
- PacketSize: 64 Configurations: 1
- Vendor: 0x0951 Product 0x1613 Version 1.0
  Configuration: 1
    - Interfaces: 1 Bus Powered 100mA
      Interface: 0
        - Alternate Setting 0, Endpoints: 2
        - Class Mass Storage, Transp. SCSI, Bulk only
        - Endpoint 1 In Bulk MaxPacket 64
        - Endpoint 2 Out Bulk MaxPacket 64

[u-boot@MINI2440]# usb storage
Device 0: Vendor: Kingston Rev: PMAP Prod: DT 101 II
          Type: Removable Hard Disk
          Capacity: 3875.0 MB = 3.7 GB (7936000 x 512)

[u-boot@MINI2440]# usb dev 0

USB device 0:
Device 0: Vendor: Kingston Rev: PMAP Prod: DT 101 II
          Type: Removable Hard Disk
          Capacity: 3875.0 MB = 3.7 GB (7936000 x 512)
... is now current device
[u-boot@MINI2440]# usb part 0
print_part of 0

Partition Map for USB device 0 -- Partition Type: DOS

Partition Start Sector Num Sectors Type
4 63 7935937 c
```

```
[u-boot@MINI2440]# usb read 0x30008000 0 200

USB read: device 0 block # 0, count 512 ... .....
512 blocks read: OK
[u-boot@MINI2440]# md.b 0x30008000 200

30008000: fa 31 c0 8e d8 8e c0 8e d0 bc 00 7c fb fc 89 e6
.1.....|....
30008010: bf 00 06 b9 00 01 f3 a5 ea dc 06 00 00 10 00 01
.....
30008020: 00 00 7c 00 00 00 00 00 00 00 00 00 80 3f 00
..|.....?.
30008030: ff 00 ed 01 1e 0e 1f 3a 16 10 00 74 06 1f ea 36
.....:t...6
30008040: e7 00 f0 3d fb 54 75 05 8c d8 fb eb 1d 80 fc 08
...=.Tu.....
30008050: 75 1b e8 81 00 8a 36 13 00 fe ce 8b 0e 15 00 86
u....6.....
30008060: cd c0 e1 06 0a 0e 11 00 31 c0 f8 eb 65 80 fc 02
.....1...e...
30008070: 72 cb 80 fc 04 77 c6 60 80 cc 40 50 be 00 00 c7
r...w.`..@P....
30008080: 04 10 00 30 e4 89 44 02 89 5c 04 8c 44 06 66 31
...0..D..\..D.f1
30008090: c0 66 89 44 0c 88 f0 f6 26 11 00 88 cf 88 eb c0
.f.D....&.....
300080a0: ef 06 81 e1 3f 00 01 c8 48 89 c7 a1 13 00 f7 26
....?...H.....&
300080b0: 11 00 f7 e3 01 f8 81 d2 00 00 89 44 08 89 54 0a
.....D..T.
300080c0: 58 30 c0 8a 16 10 00 e8 0c 00 88 26 03 00 61 a1
X0.....&..a.
300080d0: 02 00 1f ca 02 00 9c ff 1e 22 00 c3 80 fa 8f 7f
.....".....
300080e0: 04 88 16 2d 06 be 87 07 e8 8d 00 be be 07 31 c0
...-.....1.
300080f0: b9 04 00 f6 04 80 74 03 40 89 f5 81 c6 10 00 e2
.....t.@.....
30008100: f2 48 74 02 cd 18 bf 05 00 be 1d 06 c7 44 02 01
.Ht.....D..
30008110: 00 66 8b 46 08 66 89 44 08 b8 00 42 8a 16 2d 06
.f.F.f.D...B..-.
30008120: cd 13 73 0d 4f 74 49 30 e4 8a 16 2d 06 cd 13 eb
..s.Otl0...-....
```



```

30008130: d8 a1 fe 7d 3d 55 aa 75 37 fa 66 a1 4c 00 66 a3
...}=U.u7.f.L.f.
30008140: 3f 06 be 13 04 8b 04 48 89 04 c1 e0 06 8e c0 31
?.....H.....1
30008150: ff be 1d 06 b9 60 00 fc f3 a5 c7 06 4c 00 17 00
.....`.....L....
30008160: a3 4e 00 fb 8a 16 2d 06 89 ee fa ea 00 7c 00 00
.N....-.....|..
30008170: be aa 07 e8 02 00 eb fe ac 20 c0 74 09 b4 0e bb
..... .t....
30008180: 07 00 cd 10 eb f2 c3 53 74 61 72 74 20 62 6f 6f
.....Start boo
30008190: 74 69 6e 67 20 66 72 6f 6d 20 55 53 42 20 64 65
ting from USB de
300081a0: 76 69 63 65 2e 2e 2e 0d 0a 00 42 6f 6f 74 20 66
vice.....Boot f
300081b0: 61 69 6c 65 64 00 00 00 ea eb d4 ca 00 00 00 00
ailed.....
300081c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.....
300081d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.....
300081e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 80 01
.....
300081f0: 01 00 0c fe 7f ec 3f 00 00 00 c1 17 79 00 55 aa
.....?.....y.U.

```

## (10) FAT 文件系统指令

fatinfo: 显示文件系统的相关信息

格式: fatinfo <interface> <dev[:part]>

Interface: 代表接口, 如 usb、mmc;

dev: 代表设备编号, 如 0、1.....;

part: 代表存储设备中的分区, 如 1、2.....。

fatload: 从 FAT32 文件系统中读取二进制文件到 SDRAM。

格式: fatload <interface> <dev[:part]> <addr> <filename> [bytes]

Interface、dev 和 part 同上;

addr: 代表写入 SDRAM 的地址;

filename: 代表存储设备中的文件名;

bytes: 代表从存储设备中读取的文件大小, 可不填; 如果填的数据比文件小, 就只读取 bytes 字节, 如果填的数据比文件大, 也只读取文件的大小。

fatls: 列出 FAT32 文件系统中目录里的文件。

格式: fatls <interface> <dev[:part]> [directory]

Interface、dev 和 part 同上;

directoryr: 代表所要查看的目录, 可不填, 默认为/。

这些指令基本上要和 U 盘或者 SD 卡同时使用, 主要用于读取这些移动存储器的 FAT32 分区。

使用范例:

```
[u-boot@MINI2440]# usb part 0
```

print\_part of 0

Partition Map for USB device 0 -- Partition Type: DOS

Partition Start Sector Num Sectors Type

4 63 7935937 c

```
[u-boot@MINI2440]# fatinfo usb 0:4
```

Interface: USB

Device 0: Vendor: Kingston Rev: PMAP Prod: DT 101 II

Type: Removable Hard Disk

Capacity: 3875.0 MB = 3.7 GB (7936000 x 512)

Partition 4: Filesystem: FAT32 "7600\_16385\_"

```
[u-boot@MINI2440]# fatls usb 0:4
```

boot/

efi/

sources/

support/

upgrade/

43 autorun.inf

383562 bootmgr

111880 setup.exe

256220 u-boot.bin

4 file(s), 5 dir(s)

```
[u-boot@MINI2440]# fatls usb 0:4 /boot/
```

./

../

fonts/

zh-cn/

262144 bcd

3170304 boot.sdi

1024 bootfix.bin

97280 bootsect.exe

4096 etfsboot.com

485440 memtest.exe

6 file(s), 4 dir(s)

```
[u-boot@MINI2440]# fatload usb 0:4 0x30008000 u-boot.bin
```

reading u-boot.bin

.....

```
256220 bytes read
```

```
[u-boot@MINI2440]# fatload usb 0:4 0x30008000 u-boot.bin 200  
reading u-boot.bin
```

```
512 bytes read
```

脚本运行指令

`run var [...]`

**var** : ENV 中的脚本名

使用范例:

```
[u-boot@MINI2440]# setenv a_run_test echo $bootfile \; version
```

```
[u-boot@MINI2440]# run a_run_test
```

```
zImage.img
```

```
U-Boot 2009.11 ( 4&aelig;#339;#710; 04 2010 - 12:09:25)
```