


Linux 平台 zdownloader 功能使用指南

V1.0

法律声明

若接收深圳市中兴微电子有限公司（以下称为“中兴微电子”）的此份文档，即表示您已同意以下条款。若不同意以下条款，请停止使用本文档。

本文档版权所有人深圳市中兴微电子有限公司，保留任何未在本文档中明示授予的权利。文档中涉及中兴微电子的专有信息。未经中兴微电子事先书面许可，任何单位和个人不得以任何形式或者手段复制、传递、分发、租赁、销售、使用和泄漏该文档的任何组成部分以及该文档包含的任何图片、表格、数据及其他信息。

 SANECHIPS™ 是中兴微电子的注册商标。中兴微电子产品的名称和标志是中兴微电子的商标或注册商标。在未经中兴微电子或本文档中可能涉及的第三方权利人事先书面同意的情况下，阅读本文档并不表示阅读者被以默示、不可反言或其他方式授予任何使用本文档中出现的任何中兴微电子或者其他机构的标记的权利。

本产品符合有关环境保护和人身安全方面的设计要求，产品的存放、使用和弃置应遵照产品手册、相关合同或相关国法律、法规的要求进行。

本文档按“现状”和“仅此状态”提供。本文档中的信息随着中兴微电子产品和技术进步将不断更新，中兴微电子不再通知此类信息的更新。

前言

手册说明

该文档提供 Linux 平台 zdownloader 工具使用指导。

本书约定

本书采用的约定及标志如下。

1. 符号约定

带尖括号“< >”表示键名、按钮名以及操作员从终端输入的信息；带方括号“[]”表示人机界面、菜单条、数据表和字段名等，多级菜单用“→”隔开。如 [文件→新建→文件夹] 多级菜单表示 [文件] 菜单下的 [新建] 子菜单下的 [文件夹] 菜单项。


2. 键盘操作约定


格式	意义
加尖括号的字符	表示键名、按钮名。如<Enter>、<Tab>、<Backspace>、<a>等分别表示回车、制表、退格、小写字母 a
<键 1+键 2>	表示在键盘上同时按下几个键。如<Ctrl+Alt+A>表示同时按下“Ctrl”、“Alt”、“A”这三个键
<键 1, 键 2>	表示先按第一键，释放，再按第二键。如<Alt, F>表示先按<Alt>键，释放后，紧接着再按<F>键

3. 鼠标操作约定

格式	意义
单击	快速按下并释放鼠标的左键
双击	连续两次快速按下并释放鼠标的左键
右击	快速按下并释放鼠标的右键
拖动	按住鼠标的左键不放，移动鼠标

4. 标志

小心、注意、警告、危险前使用符号“”。

说明、提示、小窍门前使用符号“”。

版本更新说明

资料版本	资料编号	资料更新说明
V1.0		手册第一次发行

目录

1 简介	6
2 ZDOWNLOADER 编译与使用	6
2.1 编译说明	6
2.2 参数说明	6
3 客户需要准备的内容	7
4 代码目录结构	7
5 调试流程	8
6 调试中可能遇到的异常问题	9
附录 A 术语和缩略语表	10
A.1 术语表	10
A.2 缩略语表	10
附录 B LINUX 平台驱动修改	11
B.1 串口驱动代码修改	11
B.2 串口驱动的编译	13
B.3 串口驱动加载	13
B.4 串口设备识别	14

1 简介

本文档主要介绍 Linux 平台 **zdownloader** 参考应用。该应用可以实现模块的版本下载功能。

2 zdownloader 编译与使用

2.1 编译说明

本参考应用提供 **Makefile** 文件，用户使用时需根据实际环境修改 **Makefile** 中的编译链配置，然后执行下面语句，即可编译生成 **zdownloader** 可执行文件：

```
make clean all
```

2.2 参数说明

zdownloader 应用程序提供以下参数。注意，由于涉及到文件夹的读写和 **usb** 访问，该应用程序需要 **root** 权限。

参数	说明
-h	参数简要帮助说明；
-p	指定烧写的 bin 文件及所在路径；
-c	传入要校验的版本号，若不设置，则不校验版本号；
-s	指定需要保存的 log 存储路径，若不指定，则选择默认路径。默认路径在代码 LogInfo.h 中通过宏 DEFAULT_LOG_PATH 定义。

3 客户需要准备的内容

- a)客户需在 linux 的设备驱动中增加下载口和 AT 口的 pid 配置（下载口的 pid（19D2: 0256）、AT 端口的 pid（19D2: 0582）），具体方法可参考附录 B;
- b)客户需提供重启指令，能够通过此指令让模块掉电重启。具体就是修改 main.c 中的 HardWareReset 函数，在此函数中实现模块掉电重启。

```
static void HardWareReset(int type)
{
    char cversion[1024] = {0};
    char dev_tty[MAX_PORT_PATH_LENGTH] = {0};
    char AT_Reset[MAX_CMD_LENGTH] = "\r\nat+zzsoftreset\r\n";
    char cmdDevReboot[MAX_CMD_LENGTH] = "echo reboot > /dev/ttyUSB";

    snprintf(dev_tty, MAX_PORT_PATH_LENGTH, "/dev/ttyUSB%d", g_at_ttyId);
    snprintf(cmdDevReboot, MAX_CMD_LENGTH, "echo reboot > /dev/ttyUSB%d", g_dl_ttyId);

    if (type == 0)
    {
        SendATString(dev_tty, AT_Reset, cversion);
    }
    else
    {
        printf("HardWareReset(1):: %s \r\n", cmdDevReboot);
        system(cmdDevReboot);
    }
}
```

- c)客户确保在下载过程中，不会有其它进程因与模块交互而占用端口。

4 代码目录结构

```
C CSerial.c
C CSerial.h
C define.h
C devUsb.c
C devUsb.h
C download.c
C download.h
C LogInfo.c
C LogInfo.h
C main.c
M Makefile
```

- a) `define.h` 为公用头文件。
- b) `CSerial.h` 和 `CSerial.c` 为串口读写接口, 其中包括端口设置, 打开关闭端口, 端口数据读写接口。
- c) `LogInfo.h` 和 `LogInfo.c` 为 log 打印接口。
- d) `Download.h` 和 `download.c` 为下载流程代码, 里面包括了所有的下载协议。详细的下载协议见下面小节的描述。
- e) `devUsb.h` 和 `devUsb.c` 为查找端口对应 `ttyUSB` 设备号。
- f) `main.c` 是下载应用 `main` 接口代码, 里面包括了下载流程的调用, 也包括了一些校验流程。

5 调试流程

- a) 编译代码。使用 `cd` 命令切换到代码目录, 使用 `make clean; make` 命令编译代码。对编译出来的下载应用获取执行权限, 并拷贝到设备的下载应用存储目录上。
- b) 将版本文件拷贝到设备。由于文件比较大, 所以需要确保有足够空间存放。
- c) 禁止设备跟模块进行数据交换。
- d) 执行下载应用, 比如:

版本文件路径为 `/tmp/7520V3SCV2.01.01.02P42U05_MDL.bin`,

版本号为 `7520V3SCSDKV2.01.01.02P42U05`,

则使用命令:

```
// 不检查版本号
./zdownloader -p /tmp/7520V3SCV2.01.01.02P42U05_MDL.bin
// 检查版本号
./zdownloader -p /tmp/7520V3SCV2.01.01.02P42U05_MDL.bin -c
7520V3SCSDKV2.01.01.02P42U05
```

若还要修改 `log` 保存路径, 则使用命令:

```
// 若 log 保存在/var 下
./zdownloader -p /tmp/7520V3SCV2.01.01.02P42U05_MDL.bin -c
7520V3SCSDKV2.01.01.02P42U05 -s /var
```


e) 下载应用执行完成自动退出应用。

```
ExchangeImagefunc:: 10 OK...
DownloadImagefunc:: case nand...
DownloadOneImagefunc:: Open file OK
DownloadOneImagefunc:: STAT_SET_PACKET_SIZE
DownloadOneImagefunc:: CommandBuffer = compat_write uboot2 00000000 26730
ReadDataExtraFuncB:: readCount = 14, dwReadCount = 14
DownloadOneImagefunc:: STAT_SET_PACKET_SIZE OK
DownloadOneImagefunc:: not CRCCheck
ReadDataExtraFuncB:: readCount = 5, dwReadCount = 5
DownloadOneImagefunc:: Image data send success!
ExchangeImagefunc:: 11 OK...
DownloadImagefunc:: case nand...
DownloadOneImagefunc:: Open file OK
DownloadOneImagefunc:: STAT_SET_PACKET_SIZE
DownloadOneImagefunc:: CommandBuffer = compat_write zloader 00000000 2000
ReadDataExtraFuncB:: readCount = 14, dwReadCount = 14
DownloadOneImagefunc:: STAT_SET_PACKET_SIZE OK
DownloadOneImagefunc:: not CRCCheck
ReadDataExtraFuncB:: readCount = 5, dwReadCount = 5
DownloadOneImagefunc:: Image data send success!
ExchangeImagefunc:: 12 OK...
OpenDownloadPortAndDownloadfunc:: PrimaryDoDownload OK
OpenDownloadPortAndDownloadfunc:: HardWare Reset 2, please wait
HardWareReset(1):: echo reboot > /dev/ttyUSB0
main:: download success.....
```

f) 如果下载失败可查看 log。默认的 log 路径可在 LogInfo.h 中查看宏定义 DEFAULT_LOG_PATH，也可自行修改：

6 调试中可能遇到的异常问题

- ①没有 root 权限，需要以管理员权限登陆。
- ②输入参数错误，查看调用应用时是否参数是否正确。
- ③版本文件解析错误，查看版本文件名称和绝对路径是否正确，或者版本文件是否正式发布的模块版本。代码会对版本文件前 16 个字节进行校验。
- ④下载 image 文件时中断，需要 check 是否未关闭设备与 modem 交互的流进程。
- ⑤AT 口或 DL 口无法正常传输数据，检查 AT 口或 DL 口是否可以正常找到。

附录 A 术语和缩略语表

A.1 术语表

术语定义	英文对应词	含义

A.2 缩略语表

缩略语	全称	中文含义
AP	Application Processor	应用处理器

附录 B Linux 平台驱动修改

对于 usb 虚拟串口和 NDIS 网卡模式，Linux 平台的桌面操作系统在某些情况下需要自己编译驱动文件，比如 pid 的新增、修改，接口号顺序的变化。

B.1 串口驱动代码修改

(1) 在使用的系统上，用 `uname -a` 命令查看系统的内核版本号，并从 kernel.org 上面下载对应的内核版本。查看版本号例子如下：

```
$ uname -a
Linux zte-QiTianM4500-N000 4.15.0-112-generic #113~16.04.1-Ubuntu SMP Fri Jul 10 04:37:08 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
```

所查看到的系统内核版本号为 4.15.0-112-generic，对于内核版本号，只需要关注到 4.15.0 即可，后面的-112-generic 不用关注，查看 Linux 版本遵照此规则。从 <https://www.kernel.org/> 网站找到对应的内核源码包，然后下载。

(2) 从下载的内核版本中提取虚拟串口驱动文件 `option.c` 和必须的头文件 `usb-wwan.h`，网卡驱动文件 `qmi_wwan.c`。文件的路径如下：

```
linux_src\drivers\usb\serial\option.c
linux_src\drivers\usb\serial\usb-wwan.h
linux_src\drivers\net\usb\qmi_wwan.c
```

(3) 在 `linux_src\drivers\usb\serial\option.c` 文件中添加驱动 `idProduct`(即 `pid`)，包括 DL 端口的 `pid` (19D2: 0256)。

在静态结构体数组 `static const struct usb_device_id option_ids[]` 中增加需要配置的 `pid` 信息。在这个数组中找到包含“ZTE_VENDOR_ID”的最后一行，在其后添加如下代码，其中斜体加黑的是需要增加的内容：

```
{ USB_VENDOR_AND_INTERFACE_INFO(ZTE_VENDOR_ID, 0xff, 0x02, 0x01) },
{ USB_VENDOR_AND_INTERFACE_INFO(ZTE_VENDOR_ID, 0xff, 0x02, 0x05) },
{ USB_VENDOR_AND_INTERFACE_INFO(ZTE_VENDOR_ID, 0xff, 0x86, 0x10) },
{ USB_DEVICE_AND_INTERFACE_INFO(ZTE_VENDOR_ID, 0x0256, 0xff, 0xff, 0xff) },
{ USB_DEVICE_AND_INTERFACE_INFO(ZTE_VENDOR_ID, 0x0532, 0xff, 0xff, 0xff) },
{ USB_DEVICE_AND_INTERFACE_INFO(ZTE_VENDOR_ID, 0x0536, 0xff, 0xff, 0xff) },
{ USB_DEVICE_AND_INTERFACE_INFO(ZTE_VENDOR_ID, 0x0542, 0xff, 0xff, 0xff) },
{ USB_DEVICE_AND_INTERFACE_INFO(ZTE_VENDOR_ID, 0x0573, 0xff, 0xff, 0xff) },
{ USB_DEVICE_AND_INTERFACE_INFO(ZTE_VENDOR_ID, 0x0582, 0xff, 0xff, 0xff) },
{ USB_DEVICE_AND_INTERFACE_INFO(ZTE_VENDOR_ID, 0x0543, 0xff, 0xff, 0xff),
.driver_info = (kernel_ulong_t)&net_intf1_blacklist},
```

```
{ USB_DEVICE_AND_INTERFACE_INFO(ZTE_VENDOR_ID, 0533, 0xff, 0xff, 0xff),
.driver_info = (kernel_ulong_t)&net_intf14_blacklist },
```

(4) adb 端口的过滤

模块类产品含有 adb 调试端口，需要剔除不适配串口驱动，否则会被当做串口。在 option.c 的 option_probe 函数中增加如下代码：

其中黑色斜体表示如果识别到 PID 是 0x0582 的设备的接口 3 时，就返回不适配这个接口。

```
if (serial->dev->descriptor.idVendor == ZTE_VENDOR_ID &&
    serial->dev->descriptor.idProduct == 0x0582 &&
    serial->interface->cur_altsetting->desc.bInterfaceClass == 0x3)
    return -ENODEV;
```

注：以上代码在不同内核版本上不一样，根据自己的内核版本修改即可。

串口驱动编译脚本

由于仅是对系统驱动模块进行增加修改，所以不用重新编译内核，仅把对应模块编译成.ko 文件即可。所以编译脚本不使用内核源码进行编译，而是直接在所用的系统上通过 kernel-devel 进行编译。kernel-devel 包含了用于内核开发环境所必须的内核头文件和 Makefile。编译使用的 Makefile 如下：

这个 Makefile 文件仅需要修改要编译的模块名称即可，也就是下面斜体代码中的倒数第二句 your_module.o，将 your_module 修改为编译的模块名字。编译完成后直接生成.ko 文件。

注：如果编译的代码是取自内核文件，那么必须保持其模块名与内核版本命名一致，否则会导致加载匹配错误。

```
# To build modules outside of the kernel tree, we run "make"
# in the kernel source tree; the Makefile these then includes this
# Makefile once again.
# This conditional selects whether we are being included from the# kernel Makefile or
not.
ifeq ($(KERNELRELEASE),)
# Assume the source tree is where the running kernel was built# You should set
KERNELDIR in the environment if it's elsewhere
KERNELDIR ?= /lib/modules/$(shell uname -r)/build
#KERNELDIR ?= /lib/modules/2.6.28-11-generic/build
# The current directory is passed to sub-makes as argument
PWD := $(shell pwd)modules:
$(MAKE) -C $(KERNELDIR) M=$(PWD) modules
modules_install:
```

```
$(MAKE) -C $(KERNELDIR) M=$(PWD) modules_install
clean:
rm -rf *.o *~ core .depend *.cmd *.ko *.mod.c .tmp_versions
.PHONY: modules modules_install clean
else
# called from kernel build system: just declare what our modules are
obj-m := option.o
endif
```

B.2 串口驱动的编译

option 仅需要三个文件 Makefile, option.c, usb-wwan.h。

(E:) ▶ drivers_develop ▶ linux_drivers ▶ option_qmi_wwan ▶ fedora21_3.17.4 ▶ option			
帮助(H)			
新建文件夹			
名称	修改日期	类型	大小
Makefile	2015/12/19 星期...	文件	1 KB
option.c	2015/12/19 星期...	C Source	109 KB
usb-wwan.h	2015/12/19 星期...	C/C++ Header	2 KB

编译完成后，生成 option.ko。

B.3 串口驱动加载

编译完成的.ko 文件通过 insmod 命令进行加载。在终端进入到编译目录所在，然后输入如下命令：

```
insmod ./option.ko
```

./是指 option.ko 所在的当前目录。

直接加载一般都会遇到文件依赖问题，提示如下：

insmod: cannot insert 'option.ko': Unknown symbol in module

如果不是驱动修改引起的问题，一般就是所依赖的驱动没有加载导致的。需要查看所编译的驱动依赖哪些驱动文件。用如下命令：

```
modinfo ./option.ko
```

会给出一大段内容，最主要的下给出的内容的结尾，是驱动依赖：

depends: usb_wwan,usbserial

然后执行:

```
modprobe usb_wwan
modprobe usbserial
```

再执行:

```
insmod ./option.ko
```

用 lsmod 命令查看驱动加载情况:

```
lsmod | grep usb
```

```
$ lsmod | grep usb
usb_wwan                20480    1 option
usbserial               49152    5 cp210x,usb_wwan,option
usbnet                  45056    3 rndis_wlan,rndis_host,cdc_ether
mii                     16384    2 r8169,usbnet
```

如果驱动已经加载,那么返回内容中会包含这两个驱动名字。如果没有返回内容或者返回内容中没有这两个,说明驱动未加载。

B.4 串口设备识别

insmod option.ko 后看一下设备口是否识别,如下图所示 option driver 已启动成功:

```
$ lsusb -t
/: Bus 04.Port 1: Dev 1, Class=root hub, Driver=xhci_hcd/2p, 5000M
/: Bus 03.Port 1: Dev 1, Class=root hub, Driver=xhci_hcd/10p, 480M
|   Port 5: Dev 58, If 6, Class=CDC Data, Driver=cdc_acm, 480M
|   Port 5: Dev 58, If 4, Class=Vendor Specific Class, Driver=option, 480M
|   Port 5: Dev 58, If 2, Class=Vendor Specific Class, Driver=option, 480M
|   Port 5: Dev 58, If 0, Class=Communications, Driver=rndis_host, 480M
|   Port 5: Dev 58, If 5, Class=Communications, Driver=cdc_acm, 480M
|   Port 5: Dev 58, If 3, Class=Vendor Specific Class, Driver=, 480M
|   Port 5: Dev 58, If 1, Class=CDC Data, Driver=rndis_host, 480M
|   Port 6: Dev 17, If 0, Class=Vendor Specific Class, Driver=cp210x, 12M
/: Bus 02.Port 1: Dev 1, Class=root hub, Driver=ehci-pci/2p, 480M
|   Port 1: Dev 2, If 0, Class=Hub, Driver=hub/6p, 480M
/: Bus 01.Port 1: Dev 1, Class=root hub, Driver=ehci-pci/2p, 480M
|   Port 1: Dev 2, If 0, Class=Hub, Driver=hub/4p, 480M

$ lsusb
Bus 002 Device 002: ID 8087:8000 Intel Corp.
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 002: ID 8087:8008 Intel Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 017: ID 10c4:ea60 Cygnal Integrated Products, Inc. CP210x UART Bridge / myAVR mySmartUSB light
Bus 003 Device 058: ID 19d2:0582 ZTE WCDMA Technologies MSM
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

上图中所圈位置 IF=4 为 AT 口。