

**FIBOCOM\_MINIDOWNLOADTOOL**

**程序设计方案**

文档版本：V1.0

更新日期：2017.09.06

适用型号

|  |  |  |
| --- | --- | --- |
| 序号 | 产品型号 | 说明 |
| 1 | NA | NA |

版权声明

版权所有©2017 深圳市广和通无线股份有限公司。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

注意

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

版本记录

|  |  |  |
| --- | --- | --- |
| 文档版本 | 更新日期 | 说明 |
| V1.0 | 2017-09-06 | 初始版本 |

**目录**

[1 前言 5](#_Toc25058)

[2 升级流程概述 5](#_Toc19479)

[2.1 查找VIDPID流程概述 6](#_Toc19832)

[2.2 获取LIBUSB句柄流程概述 7](#_Toc5061)

[2.3 升级文件解析流程概述 8](#_Toc573)

[2.4 同步流程概述 12](#_Toc20649)

[2.5 BOOT下载流程概述 13](#_Toc15898)

[2.6 DOWNLOAD下载流程概述 14](#_Toc14618)

[3 错误码 15](#_Toc21970)

[3.1 LIBUSB错误码 15](#_Toc3952)

[3.2 m\_error\_txt 15](#_Toc18555)

# 前言

MiniDownloadTool为Intel平台升级工具，此工具基本C代码、仅依赖libusb库文件、使用FLS3格式进行升级。此工具仅满足升级功能，占用空间小，基本C语言实现，易于移植。

# 升级流程概述

开始

查找VIDPID

获取LIBUSB库句柄

解析升级文件

下发同步指令(ATAT)

Boot PSI EBL

Boot 结束

下载PSI EBL CODE

下载结束，重启模块

升级成功

升级失败

程序结束

## 查找VIDPID流程概述

开始

找到升级ID

结束

遍历已支持VIDPID表

获取USB设备VIDPID

获取USB设备描述符列表

获取USB设备列表

获取USB设备列表

启动60秒定时器

是

否

定时器工作

60秒定时时间到

## 获取LIBUSB句柄流程概述

参数解析：m\_usb\_handle，即libusb打开的文件句柄，可理解为此句柄即为升级通道。

参数解析：m\_ep\_out(通道序号及方向)，out为HOST->DEVICE，如AT发送

参数解析：m\_ep\_in(通道序号及方向)，in为DEVICE->HOST，如AT收接

参数解析：m\_vid升级设备厂商ID，m\_pid升级设备产品ID

开始

libusb\_init

libusb\_get\_device\_list

libusb\_open\_device\_with\_vid\_pid

libusb\_claim\_interface

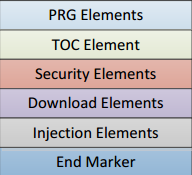
获取全局m\_usb\_handle

获取全局m\_ep\_out m\_ep\_in m\_vid m\_pid

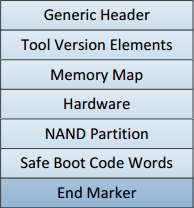
结束

## 升级文件解析流程概述

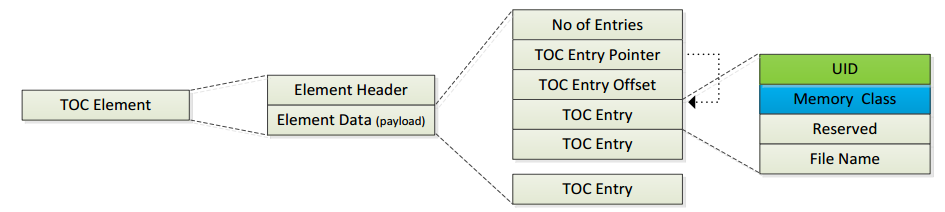
FLS文件包含PRG、TOC、Security、Download、Injection元素，如下图所示：



PRG元素，定义所有需要的控制信息，如：目标硬件相关设置及目标内存用法。下图是PRG文件格式：

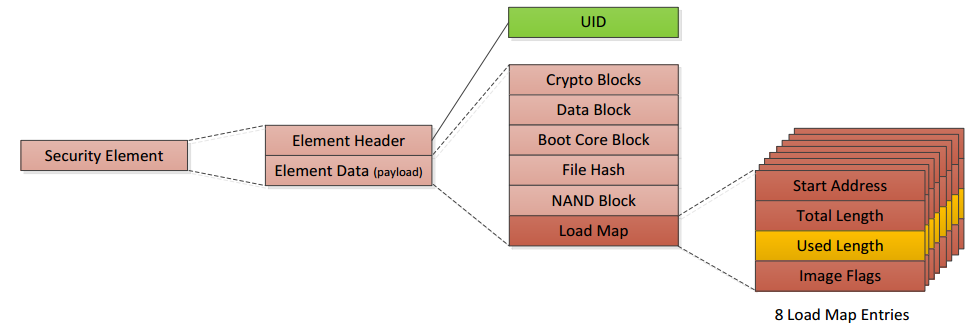


TOC元素，是The Table of Content的缩写，TOC元素主要包含着许多TOC的入口，每个TOC的入口对应于内部FLS2图像文件。每个TCO元素分别包含着一个头部及数据，数据包含内容有： TOC入口指针，偏移量，大小，文件名等。



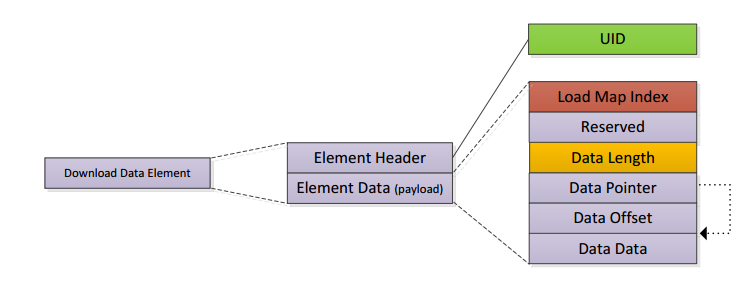
Security元素常被称为”sec pack”, 它是一个数据块，拥有一些系统配置及系统的安全数据，对每个可下载的镜像来讲sec pack都是唯一的，并且包含一些元素像分区表及LoadMap（对一个专门的区域数据量）。

Security元素是”\_SecureBlock.bin”后缀的数据文件。



Download元素包括要下载到目标闪存中的特定位置的数据块，一个可下载的图像由若干下载数据元素组成（最多8个）。下载数据元素是后缀为“\_LoadMapX.bin”的数据文件，其中X是下载的索引号

LoadMap中的数据元素。



Injection元素，未有详细解析。

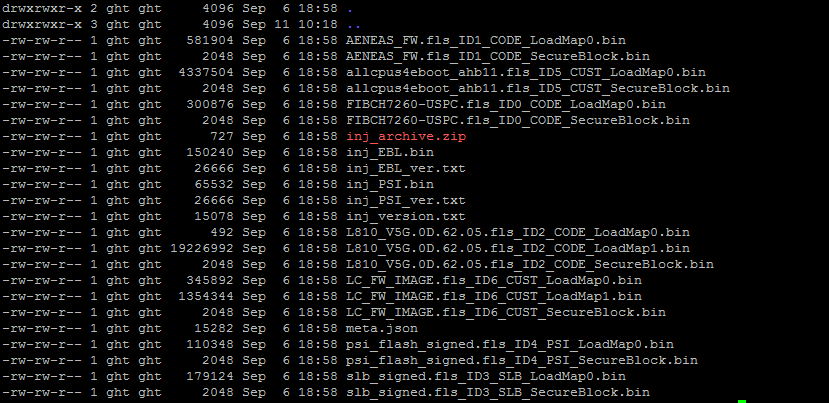
FLS3文件：

在FLS3文件中，所有下载需要的的元数据及烧录不同的二进制文件的程序都存储在名为“meta.json”的文件中，该文件跟随JSON开放标准格式。

meta.json文件包含信息元素，如内存映射，分区信息，下载文件信息等，

FLS3文件基本上是一个包含多个文件对象的TAR文件格式的压缩包，文件扩展名是“.fls3”。

一个Fls3文件包含的内容如下图所述：



FLS3文件解析流程：

打开FLS3文件，按规则填充全局结构体m\_toc，其中TOC元素成员：

typedef struct

{

char filename[100]; ---- TOC文件名

uint32\_t rec\_offset; ---- 当前指针偏移量

uint32\_t img\_offset; ---- 镜像指针偏移量

uint32\_t img\_length; ---- 镜像长度

} fls3\_file\_toc\_rec;

根据填充后的TOC元素，查找meta.json文件，并打开meta.json文件。然后再一行一行读取文件。按照文件格式分别解析并填充, m\_bootfiles 及m\_downloadfiles

typedef struct

{

char filename[STR\_LEN]; // Name of the boot file

char type[STR\_LEN]; // Type name of the boot file

uint32\_t size; // Size, in bytes, of the boot file

} boot\_file\_t;

typedef struct

{

char org\_filename[STR\_LEN]; // The original name of the FLS file packed into the FLS3 file.

char sec\_filename[STR\_LEN]; // Name of the file holding the secure package (SecPack) to be used during download

char type[STR\_LEN]; // Type name of the download file.

region\_t region[MAX\_DOWNLOAD\_REGIONS];

} download\_file\_t;

m\_bootfiles及m\_downloadfiles为需要下载的文件。

## **同步流程概述**

当模块切换至升级模式时，需要升级工具下发同步指令ATAT，让模块进入升级状态。

其中VidPid为8087/07F5时，需要下发ATAT，而非此VID时，需要先下发一个AT，然后再下发一个AT。

同步成功时，模块先返回2字节回应，其中第一个字节表示LER版本号，第二个字节表示剩余需要读取的回复。其中VidPid为8087/0716时，剩余读取长度需要减2字节长度。

开始

判断VIDPID

结束

8087/07F5

Write AT

Write ATAT

同步失败

同步成功

Write AT

Read Success

## **BOOT下载流程概述**

以BOOT PSI下载为例，先打开BOOT PSI文件，获取文件大小。

校验和：PSI文件内容异或相加即 psi\_xor ^= psi\_code[i]

PSI包长度及长度命令: psi\_cmd\_length = (psi\_info.size << 8) + 0x30。

BOOT\_PSI下发流程：先下发psi\_cmd\_length, 再下发psi\_code,再下发psi\_xor。

开始

获取BOOT PSI文件名，类型，大小

结束

下发包长度命令

下发检验和

计算文件检验及包长度命令

下发包长度数据

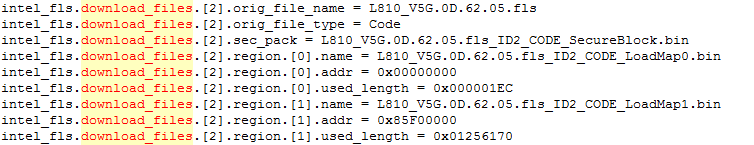
判断1字节psi\_rom\_ack

判断1字节psi\_rom\_stat

判断2字节psi\_alive\_rsp

## **DOWNLOAD下载流程概述**

以L810\_V5G.0D.62.05.fls文件为例，首先需要下载SecureBlock.bin文件，再下载LoadMap0.bin及LoadMap1.bin. 其中所有.fls后缀文件的SecureBloack.bin文件大小均为2048大小。



以SecureBlock.bin文件下载为例，讲述下载流程。下载SecureBlock时需要在其数据前添加EBL头部，其中EBL2数据头格式：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bytes 0 | Bytes 1 | Bytes 2 | Bytes 3 | Bytes 4 | Bytes 5 | Bytes 6 | Bytes 7 |
| 校验和(checksum) | | 包类型(pack\_id) | | 包长度(payload\_length) | | | |

校验和：校验初始值+文件数据逐步字节相加累加和。

其中检验初始值为高16位与低16位的纯数字相加(checksum += ((payload\_length >> 16) + (payload\_length & 0x0000FFFF));)

文件数据逐步字节相加累加和:

checksum += pack[EBL2\_16\_HEAD\_SIZE+i];

开始

获取SecureBlock.bin文件大小(2048)

结束

在原数据长度前添加EBL2头

发送相应数据

检查返回情况

# 错误码

## LIBUSB错误码

|  |  |
| --- | --- |
| 错误码 | 注释 |
| LIBUSB\_SUCCESS =0 | Success (no error) |
| LIBUSB\_ERROR\_IO =-1 | Input/output error |
| LIBUSB\_ERROR\_INVALID\_PARAM =-2 | Invalid parameter |
| LIBUSB\_ERROR\_ACCESS = -3 | Access denied (insufficient permissions) |
| LIBUSB\_ERROR\_NO\_DEVICE = -4 | No such device (it may have been disconnected) |
| LIBUSB\_ERROR\_NOT\_FOUND = -5 | Entity not found |
| LIBUSB\_ERROR\_BUSY = -6 | Resource busy |
| LIBUSB\_ERROR\_TIMEOUT = -7 | Operation timed out |
| LIBUSB\_ERROR\_OVERFLOW = -8 | Overflow |
| LIBUSB\_ERROR\_PIPE = -9 | Pipe error |
| LIBUSB\_ERROR\_INTERRUPTED = -10 | System call interrupted (perhaps due to signal) |
| LIBUSB\_ERROR\_NO\_MEM = -11 | Insufficient memory |
| LIBUSB\_ERROR\_NOT\_SUPPORTED = -12 | Operation not supported or unimplemented on this platform |
| LIBUSB\_ERROR\_OTHER = -99 | Other error |

## m\_error\_txt

|  |  |
| --- | --- |
| 错误字符串 | 注释 |
| libusb init failed <%s> |  |
| libusb get device list failed |  |
| enumerataion timeout |  |
| libusb claim interface 0 failed |  |
| Write failed <%s> |  |
| requested length was not fully written |  |
| Write failed <%s> |  |
| requested length was not fully written |  |
| Read failed <%s> |  |
| read timeout |  |
| Cannot open file %s\n", filename |  |
| Current select file is not closed trying to open image %s |  |
| FLS3 file is not open trying to open image %s |  |
| fls3\_read length exceeds image end |  |
| fls3\_read fread failed |  |
| could not open log file %s |  |
| Could not find boot file of type <%s> |  |
| proc\_set\_traits VID/PID vas not available from comm module |  |
| proc\_open failed. VID/PID 0x%04X/0x%04X unknown |  |
| Malloc failed for psi code\n"); goto ERROR |  |
| Boot-ROM ACK signal failure 0x%02X |  |
| Unexpected boot-ROM ACK 0x%02X |  |
| Boot-ROM STAT signal failure (No valid next state) 0x%02X |  |
| Boot-ROM STAT signal failure (No valid next state) 0x%02X |  |
| Unexpected boot-ROM STAT 0x%02X |  |
| Unexpected PSI ALIVE response 0x%04X |  |
| Unexpected Load EBL PSI command response 0x%04X |  |
| Malloc failed for psi code |  |
| Unexpected EBL length response 0x%04X |  |
| EBL load failure 0x%04X |  |
| Unexpected EBL load response 0x%04X |  |
| EBL2\_16 protocol payload exceeds mac size |  |
| EBL2 error: File: %s Line: %d %s |  |
| EBL2\_16 protocol return TXT class error package |  |
| EBL2\_16 protocol return unknown error package |  |
| ProtConf response unexpected 0x%04X |  |
| BaudChange response unexpected 0x%08X |  |
| ReqFlashId response unexpected 0x%04X |  |
| ReqCfiInfo\_2 response unexpected 0x%04X |  |
| Malloc failed for FLS buffer |  |
| ReqSecStart response unexpected 0x%04X |  |
| ReqFlashEraseStart response unexpected 0x%04X |  |
| ReqFlashEraseCheck response unexpected 0x%04X |  |
| ReqFlashSetAddress response unexpected 0x%04X |  |
| ReqSecEnd response unexpected 0x%04X |  |