GigaDevice Semiconductor Inc.

GD32W51x
Arm® Cortex®-M33 32-bit MCU

安全启动用户指南

1.0 版本

(2021 年 11 月)

目录

缩略语

缩略语	英文含义	中文含义
AROT	Application Root of Trust	应用根信任
EFUSE	One Time Program memory	只能烧写一次的存储空间
IBL	Immutable Boot Loader	不可变引导
MBL	Mutable Boot Loader	可变引导
MP	Mass Production	量产，这里指模组产测
NSPE	Non-Secure Processing Environment	非安全区执行环境
OTA	Over the Air upgrade	空口升级
PROT	PSA Root of Trust	PSA 根信任
PSA	Platform Security Architecture	平台安全架构
ROTPK	Root of Trust Public Key	根信任公钥
ROM	Read-Only Memory	只读存储
SPC	Security Protection	安全保护

图索引

表索引

1. 安全启动

1.1. 简介

安全启动是为了确保从系统运行的第一条指令到主应用程序之间所有代码的合法性和完整性。这里有两个关键点：一是可信的第一条指令；二是发生程序段跳转时，确保下一个可执行程序段的合法性和完整性。

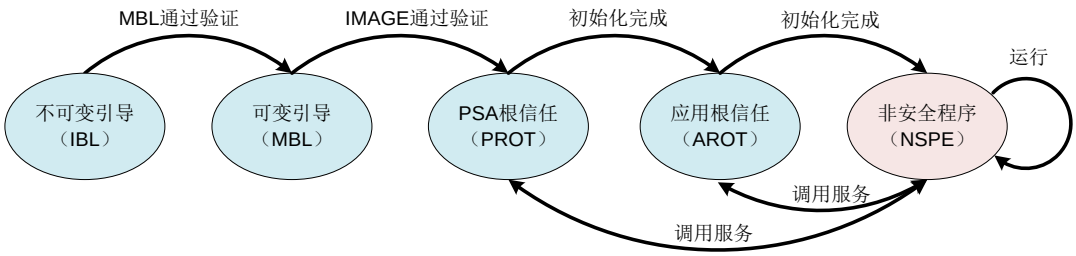
为了保证第一条指令始终是安全的，我们将第一个可执行程序段固化在 ROM 中，称之为不可变引导（IBL）。当启动方式被锁定为安全启动后，不论是上电还是重启，系统都会跳转到 IBL 来运行第一条指令，任何方式不能篡改。因此第一条指令就拥有了根信任。

第二个可执行程序段，放在 FLASH 开头，称之为可变引导（MBL）。IBL 会负责验证 MBL 的合法性和完整性，验证通过后，才能跳转到 MBL。

后面的可执行程序段，可以根据用户选择放在 FLASH 或者 SRAM。如果没有特殊情况，会建议将代码放在 FLASH 运行。根据客户选择，后面加载的可执行程序段包括 PSA 信任根程序（PROT），应用信任根程序（AROT）和非安全程序（NSPE）。

引导过程示意图见*图 1.1. 引导过程*，其中 IMAGE 是指 PROT、AROT 和 NSPE 三者合体。AROT 根据客户情况定制，发布的 SDK 中为空。

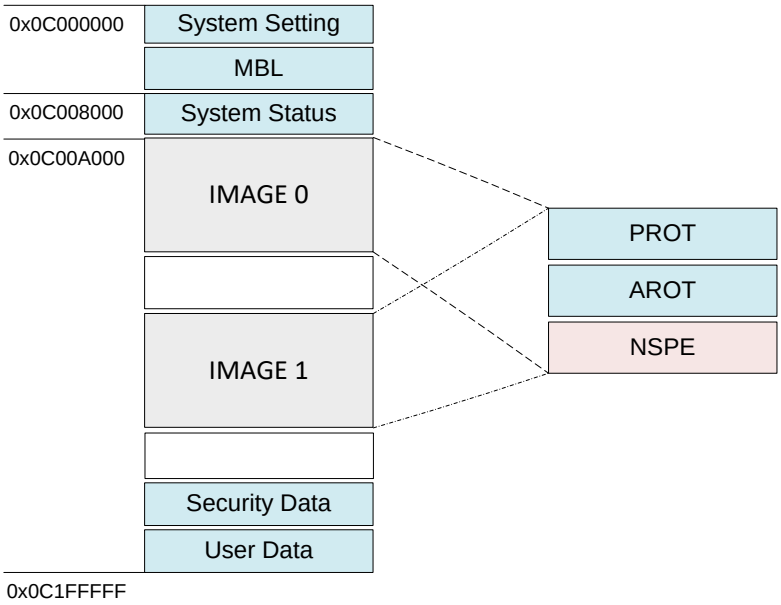
图 1.1. 引导过程



1.2. FLASH 规划

FLASH 规划见*图 1.2. FLASH 规划*。FLASH 逻辑基地址分为安全基地址（0x0c000000）和非安全基地址（0x08000000），当存储空间设为安全属性时，使用安全地址访问，反之使用非安全地址访问。为了方便了解整体规划，*图 1.2. FLASH 规划*以安全基地址为例来说明各部分的分配情况。

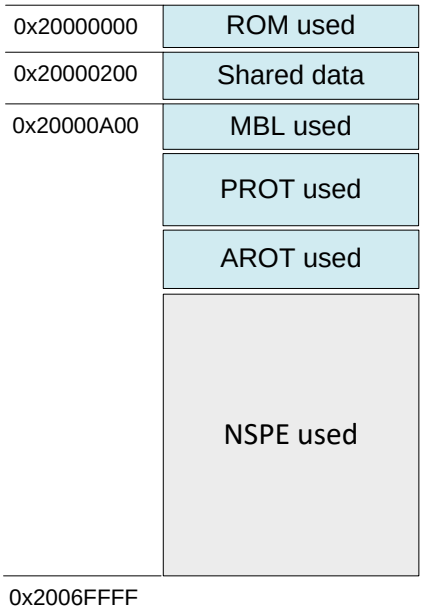
图 1.2. FLASH 规划



1.3. SRAM 规划

SRAM 规划见*图 1.3. SRAM 规划*。SRAM 逻辑基地址也分安全基地址（0x30000000）和非安全基地址（0x20000000），当存储空间设为安全属性时，使用安全地址访问，反之使用非安全地址访问。图中以非安全基地址为例来说明各部分的分配情况。

图 1.3. SRAM 规划



2. 硬件配置

如果要启用安全启动功能，需要设置 EFUSE 以及 Flash 选项字节。如果是 SIP Flash，需要设置 EFUSE 和 Flash 选项字节。如果是 QSPI Flash，仅需要设置 EFUSE。

请烧写 image-mp.bin，打开串口工具，根据以下章节依次输入命令，烧写必要的参数。

2.1. SIP Flash

- 启动方式（EFUSE）
  - 将安全启动的引导选项打开。
  - *# efuse set ctl 1*
- 设置验证选项（EFUSE）
  - 验证选项决定 ROM 是否会验证 MBL 固件和证书的合法性，如果未设置，就不会验证也就没有真正使能安全启动。
  - *# efuse set tzctl 0xc0*
- 设置 ROTPK（EFUSE）
  - ROTPK 用来验证 MBL 固件的签名和证书，它的产生方法请参考*密钥和证书生成*小节，以下命令中的 ROTPK 可以验证发布的 SDK 生成的证书和固件。设置 tzctl 的 BIT2 之后，ROTPK 被锁住，不会再被破坏。下述命令中 ROTPK 数据仅为范例。
  - *# efuse set rotpk f0cc01c8a384bdc8694b254f4dd3f8b86a25ae6ca2082c838e780e42c2b157a2*
  - *# efuse set tzctl 4*
- 使能 Trust Zone（FLASH OP）
  - 使能 Trust Zone 并将 SPC 级别设置为 0。0xAA 代表读保护级别为 0，0x55 代表读保护级别为 0.5，0x11 代表读保护级别为 1。
  - *# fmcob set obr 0x80AA*
- 设置 SECMARK（FLASH OP）
  - 寄存器 SECMARK[0-3]可以用来配置 FLASH 安全区域范围，命令格式为：
  - fmcob set secmark <index> <start page> <end page>
  - 其中，index 为 0-3，[start page, end page]范围内为安全空间，page 大小为 4K。这里需要根据用户自己的 FLASH 规划，将 PROT 固件尾部的 Page Index 填到 end page，并且预留出 PROT 将来升级膨胀的空间。SECMARK 的详细定义请参考 GD32W51x User Manual。
  - *# fmcob set secmark 0 0 0x39*
- 设置 AESK（EFUSE）
  - 可选项。如果选择对固件加密，还需要烧写固件加密密钥。需要注意的是，一旦烧写后就只能运行 AESK 加密的固件，无法再退回。以下 AESK 需要跟固件包生成时使用的密钥相同。设置 usctl 的 BIT5 之后，AESK 被锁住，不会再被破坏。下述命令中 AESK 数据仅为范例。
  - *# efuse set aesk 11223344556677889aabbccddeeff00*
  - *# efuse set usctl 0x20*

2.2. QSPI Flash

- 启动方式（EFUSE）
  - 将安全启动选项打开。
  - *# efuse set ctl 1*
- 设置验证选项（EFUSE）
  - 验证选项决定 ROM 是否会验证 MBL 固件和证书的合法性，如果未设置，就不会验证，也就没有真正使能安全启动。
  - *# efuse set tzctl 0xc0*
- 设置 ROTPK（EFUSE）
  - ROTPK 用来验证 MBL 固件的签名和证书，它的产生方法请参考*密钥和证书生成*小节，以下命令中的 ROTPK 可以验证发布的 SDK 生成的证书和固件。设置 tzctl 的 BIT2 之后，ROTPK 被锁住，不会再被破坏。下述命令中 ROTPK 数据仅为范例。
  - *# efuse set rotpk f0cc01c8a384bdc8694b254f4dd3f8b86a25ae6ca2082c838e780e42c2b157a2*
  - *# efuse set tzctl 4*
- 使能 Trust Zone（EFUSE）
  - *# efuse set tzctl 1*
- 设置 AESK（EFUSE）
  - 可选项。如果选择对固件加密，还需要烧写固件加密密钥。需要注意的是，一旦烧写后就只能运行 AESK 加密的固件，无法再退回。以下 AESK 需要跟固件包生成时使用的密钥相同。设置 usctl 的 BIT5 之后，AESK 被锁住，不会再被破坏。下述命令中 AESK 数据仅为范例。
  - *# efuse set aesk 11223344556677889aabbccddeeff00*
  - *# efuse set usctl 0x20*

3. 固件包制作

3.1. 环境准备

3.1.1. Keil uVision

为了支持 ARMCM33，建议安装 uVision 5.25 及以上的版本。

3.1.2. Python

建议安装 Python3，安装之后，运行%SDK%\setup.bat，会自动安装固件封装脚本需要用到的库。安装成功后将 Python 路径添加系统环境变量 PATH 中。

3.1.3. OpenSSL

如果是 Windows，建议安装 OpenSSL1.1.1。安装成功之后将 OpenSSL 路径添加系统环境变量 PATH 中。

3.2. 固件包格式

通常固件包有两种，一种是出厂包，一种是升级包。

3.2.1. 出厂包

出厂包是指在模组量产的时候，烧录到 FLASH 的固件包。出厂包包含的内容参见*图 1.2. FLASH 规划*，System Status 和 User Data 区域使用 0xFF 补空，IMAGE1 存放产测固件(image-mp.bin)，IMAGE0 存放用户固件(image-all.bin)，Security Data 存放身份数据，例如云端服务器签发的设备证书等等。

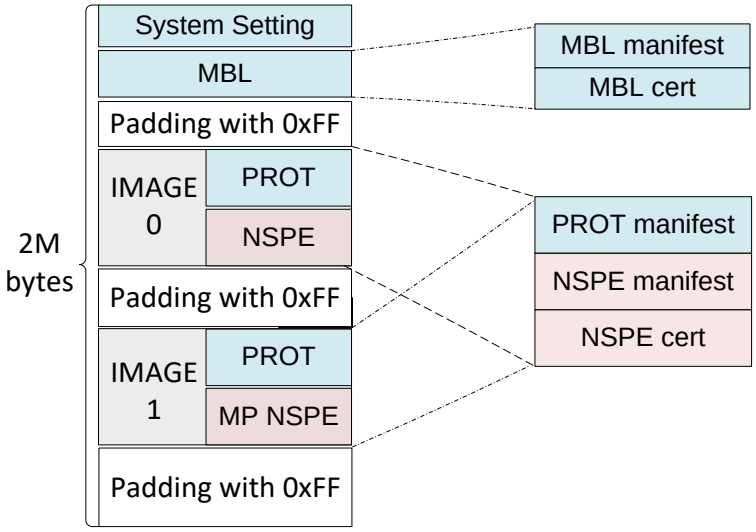
烧录之后，默认跳转到产测固件。当产测完成后，再输入串口命令切换到用户固件，简单测试后等待出厂。

本文示例的出厂包以发布的 SDK 为基础，具有以下特点：

- 没有 AROT；
- 没有 Security Data；
- 有两份证书，MBL 证书和 NSPE 证书；
- 每个可执行程序段都拥有自己的固件清单，MBL 固件清单，PROT 固件清单和 NSPE 固件清单。

本文示例的出厂包格式如下*图 3.1. 出厂固件包示意图*所示，其中固件清单的格式见*封装格式*小节。

图 3.1. 出厂固件包示意图



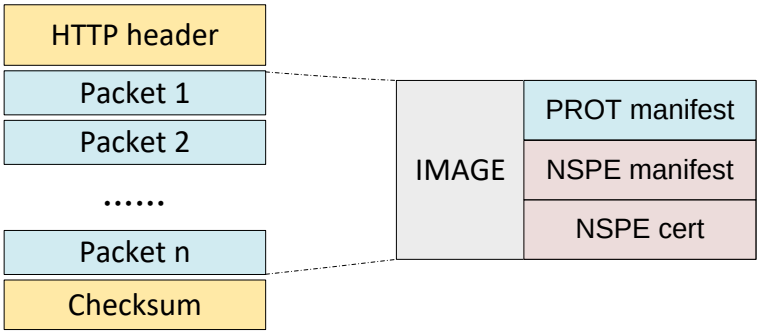
3.2.2. 升级包

升级包是指 IMAGE0 或者 IMAGE1，每次固件更新以 IMAGE 为单位，PROT、AROT 和 NSPE 三者整体更新。由于 NSPE 需要调用 PROT、AROT 提供的服务，因此需要链接二者提供的 NSC 库，也就意味着这两者如果有改动，NSPE 必须同步更新。再考虑到 PROT、AROT 应该尽可能小，升级过程尽量简单，因此将三者合在一起统一更新。本文示例的升级包中没有 AROT。

IMAGE 的版本号由 NSPE 的版本号来代表。

通常使用 HTTPS 协议来从远端服务区下载固件升级包。升级包示意见下*图 3.2. 升级固件包示意图*。

图 3.2. 升级固件包示意图



3.3. 固件封装

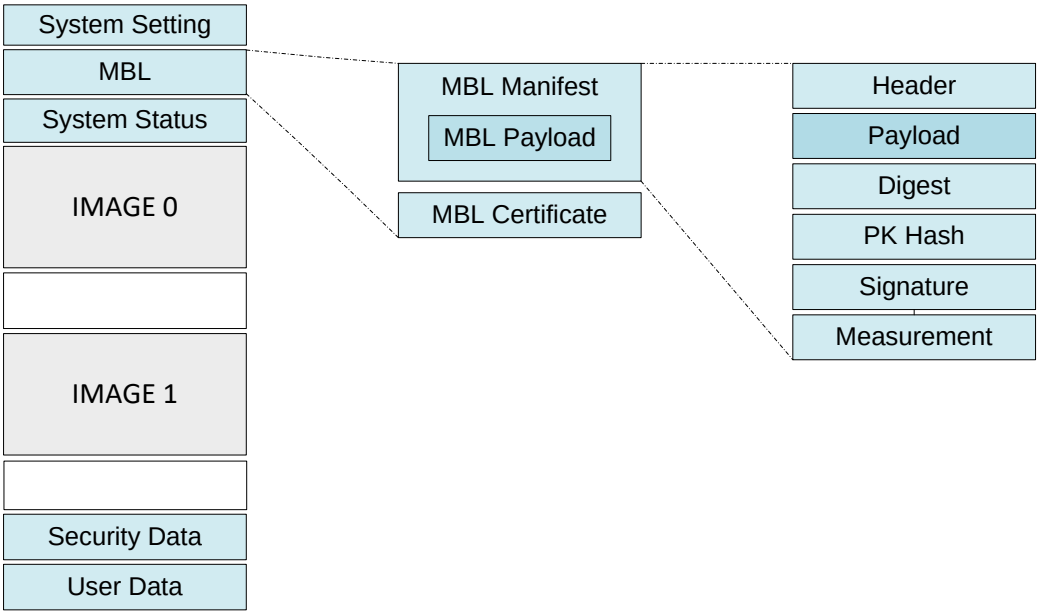
介绍完固件包，来看一下固件包中每个固件，也即每个可执行程序是如何封装的。

3.3.1. 封装格式

以 MBL 为例，见下*图 3.3. 固件封装格式*，MBL 可执行程序 Payload 被包裹在固件清单中，尾

部再跟证书。

图 3.3. 固件封装格式



3.3.2. 密钥和证书生成

在讨论如何封装之前，先看下如何生成密钥对、证书以及需要生成哪些密钥和证书。

一共需要生成三个密钥对，分别是 ROT 密钥对、MBL 密钥对和 NSPE 密钥对。需要生成三份证书，分别是 ROT 证书、MBL 证书和 NSPE 证书。

打开 Window CMD 窗口，切换到用来保存密钥和证书的目录%SDK%\scripts\images\，接下来依次输入下文的命令生成需要用到的密钥对和证书。

ROT 密钥对和证书

生成 ROT 密钥对：penssl req -key rot-key.pem -new -out rot-req.csr

图 3.4. 生成 ROT 密钥对

```
D:\work\test>openssl req -newkey ED25519 -new -out rot-req.csr
Generating a ED25519 private key
writing new private key to 'privkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:cn
State or Province Name (full name) [Some-State]:js
Locality Name (eg, city) []:sz
Organization Name (eg, company) [Internet Widgits Pty Ltd]:gd
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:gigadevice.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

需要设置 PEM 密码，本文默认使用“12345678”。然后依次填写证书请求相关信息，成功后会得到 privkey.pem 和 rot-req.csr。其中 privkey.pem 就是 ROT 私钥，为了区分，我们把它更名为 rot-key.pem。由于公钥可以由私钥推算出来，所以只需要保存私钥。

> move privkey.pem rot-key.pem

生成 ROT 证书，由它给 MBL 证书签名

> openssl x509 -req -in rot-req.csr -signkey rot-key.pem -out rot-cert.pem -days 3650

图 3.5. 生成 ROT 证书

```
D:\work\test>openssl x509 -req -in rot-req.csr -signkey rot-key.pem -out rot-cer
t.pem -days 3650
Signature ok
subject=C = cn, ST = js, L = sz, O = gd, CN = gigadevice.com
Getting Private key
Enter pass phrase for rot-key.pem:
```

rot-req.csr 是证书请求，我们使用 rot-key.pem 自签名生成 rot-cert.pem。

这步需要输入上面设置的 PEM 密码“12345678”，之后会生成 rot-cert.pem。

获得 ROTPK：> openssl pkey -in rot-key.pem -pubout -out rot-key.txt -text

使用上面的命令获得 rot-key.txt，内容如下*图 3.6. ROT 密钥内容*，将其中的“pub”部分保存下来，就是将来需要烧写到 EFUSE 的 ROTPK。

图 3.6. ROT 密钥内容

```
-----BEGIN PUBLIC KEY-----
MCowBQYDK2VwAyEAXxFfWdjpDFtvVrk9+vPgsQ9U9jk+FcPQ6bIiCsUZMPQ=
-----END PUBLIC KEY-----
ED25519 Private-Key:
priv:
    35:ba:2c:ee:49:5a:c4:8c:67:30:b3:1d:ca:ae:e4:
    0f:fb:56:12:62:aa:2c:3f:9e:9c:86:84:a2:08:50:
    26:29
pub:
    5f:11:5f:59:d8:e9:0c:5b:6f:56:b9:3d:fa:f3:e0:
    b1:0f:54:f6:39:3e:15:c3:d0:e9:b2:22:0a:c5:19:
    30:f4
```

MBL 密钥对和证书

生成 MBL 密钥对：> openssl req -newkey ED25519 -new -out mbl-req.csr

图 3.7. 生成 MBL 密钥对

```
D:\work\test>openssl req -newkey ED25519 -new -out mbl-req.csr
Generating a ED25519 private key
writing new private key to 'privkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:cn
State or Province Name (full name) [Some-State]:js
Locality Name (eg, city) []:sz
Organization Name (eg, company) [Internet Widgits Pty Ltd]:gd
Organizational Unit Name (eg, section) []:gigadevice.com
Common Name (e.g. server FQDN or YOUR name) []:gigadevice.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

运行成功会得到privkey.pem和mbl-req.csr。为了区分，我们把privkey.pem 更名为mbl-key.pem。

> move privkey.pem mbl-key.pem

生成 MBL 证书：> openssl x509 -req -in mbl-req.csr -out mbl-cert.pem -signkey mbl-key.pem -CA rot-cert.pem -CAkey rot-key.pem -CAcreateserial -days 3650

图 3.8. 生成 MBL 证书

```
D:\work\test>openssl x509 -req -in mbl-req.csr -out mbl-cert.pem -signkey mbl-ke
y.pem -CA rot-cert.pem -CAkey rot-key.pem -CAcreateserial -days 3650
Signature ok
subject=C = cn, ST = js, L = sz, O = gd, OU = gigadevice.com, CN = gigadevice.co
m
Getting Private key
Enter pass phrase for mbl-key.pem:
Getting CA Private Key
Enter pass phrase for rot-key.pem:
```

运行成功后生成 mbl-cert.pem。

NSPE 密钥对和证书

生成 NSPE 密钥对：> openssl req -newkey ED25519 -new -out nspe-req.csr

图 3.9. 生成 NSPE 密钥对

```
D:\work\test>openssl req -newkey ED25519 -new -out nspe-req.csr
Generating a ED25519 private key
writing new private key to 'privkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:cn
State or Province Name (full name) [Some-State]:js
Locality Name (eg, city) []:sz
Organization Name (eg, company) [Internet Widgits Pty Ltd]:gd
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:gigadevice.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

运行成功会得到 privkey.pem 和 nspe-req.csr。为了区分，我们把 privkey.pem 更名为 nspe-key.pem。

> move privkey.pem nspel-key.pem

生成 NSPE 证书：> openssl x509 -req -in nspe-req.csr -out nspe-cert.pem -signkey nspe-key.pem -CA mbl -cert.pem -CAkey mbl-key.pem -CAcreateserial -days 3650

图 3.10. 生成 NSPE 证书

```
D:\work\test>openssl x509 -req -in nspe-req.csr -out nspe-cert.pem -signkey nspe
-key.pem -CA mbl-cert.pem -CAkey mbl-key.pem -CAcreateserial -days 3650
Signature ok
subject=C = cn, ST = js, L = sz, O = gd, CN = gigadevice.com
Getting Private key
Enter pass phrase for nspe-key.pem:
Getting CA Private Key
Enter pass phrase for mbl-key.pem:
```

运行成功后生成 nspe-cert.pem。

3.3.3. 规划配置文件

规划配置文件即%SDK%\config\config_gdm32.h，见下*图 3.11. 规划配置文件*。本文件分为四段，第一段是基地址，第二段是 SRAM 规划，第三段是 FLASH 规划，第四段是固件版本。用户可以根据项目的实际需要来进行配置。标注为“Keep unchanged!”的行不能修改，这部分是跟芯片硬件绑定的。

在各个可执行程序进行编译、链接、封装的过程中，都会用到这个文件。也就是说，SRAM 和 FLASH 规划的改动只需要修改这一个文件。

图 3.11. 规划配置文件

```
1   /* REGION DEFINE */
2   #define RE_FLASH_BASE_S       0x0C000000    /* !Keep unchanged! */
3   #define RE_FLASH_BASE_NS      0x08000000    /* !Keep unchanged! */
4   #define RE_SRAM_BASE_S        0x30000000    /* !Keep unchanged! */
5   #define RE_SRAM_BASE_NS       0x20000000    /* !Keep unchanged! */
6
7   /* SRAM LAYOUT */
8   #define RE_SHARED_DATA_START  0x0200        /* !Keep unchanged! */
9   #define RE_MBL_DATA_START     0x0A00        /* !Keep unchanged! */
10  #define RE_PROT_DATA_START    0x0A20
11  #define RE_AROT_DATA_START    0xE000
12  #define RE_NSPE_DATA_START    0xE000  // 0x4B00
13
14  /* FLASH LAYEROUT */
15  #define RE_VTOR_ALIGNMENT     0x200         /* !Keep unchanged! */
16  #define RE_SYS_SET_OFFSET     0x0           /* !Keep unchanged! */
17  #define RE_MBL_OFFSET         0x1000        /* !Keep unchanged! */
18  #define RE_SYS_STATUS_OFFSET  0x8000
19  #define RE_IMG_0_PROT_OFFSET  0xA000
20  #define RE_IMG_0_AROT_OFFSET  0x3A000
21  #define RE_IMG_0_NSPE_OFFSET  0x3A000  // 0xA000
22  #define RE_IMG_1_PROT_OFFSET  0xF0000
23  #define RE_IMG_1_AROT_OFFSET  0x120000
24  #define RE_IMG_1_NSPE_OFFSET  0x120000  // 0x100000
25  #define RE_IMG_1_END_OFFSET   0x1D6000
26  #define RE_SST_OFFSET         0x1F5000
27  #define RE_AUDIT_OFFSET       0x1FF000
28
29  /* FW_VERSION */
30  #define RE_MBL_VERSION        0x01000000
31  #define RE_PROT_VERSION       0x01000000
32  #define RE_AROT_VERSION       0x00000001
33  #define RE_NSPE_VERSION       0x01000000
34
```

3.3.4. 可执行程序生成

打开%SDK%\ MultiProject.uvmpw，会看到三个 Project，依次是 MBL、PROT 和 NSPE。依次成功编译之后，可以在各自输出目录得到相应的 AXF 文件，即 mbl.axf，prot.axf 和 nspe.axf。这三个文件在下面一个章节会用来转换出相应的二进制文件。

3.3.5. 可执行程序封装

每个可执行程序封装的目标是从 AXF 文件生成可以下载到 FLASH 运行的二进制文件或 HEX 文件，这个最终文件中包含了固件清单和固件证书，如果需要，也包括固件加密。

如果不想深入理解封装过程的细节，可以直接跳过*封装过程*小节。封装过程已经在 afterbuild 脚本中实现，在编译链接后会自动运行，生成封装后的固件。

封装脚本

%SDK%\scripts\imgtool\目录中包含了一些由 Python 写成的小工具。

- imgtool.py：为可执行程序的二进制文件添加固件清单，固件证书。
- hextool.py：将二进制文件转换为 Intel HEX，方便直接使用 JLINK 下载。
- aestool.py：对二进制文件进行 AES-CTR 加密。
- sysset.py：生成 System Setting 的二进制文件 sysset.bin。

封装过程

以 NSPE 为例，封装过程如下：

1) 首先从 AXF 文件中转换出二进制 BIN 文件

   - *> C:\Keil_v5\ARM\ARMCC\bin\fromelf.exe --bin --8x1 --bincombined --output=.\freertos \nspe.bin .\freertos\output\nspe.axf*

2) 为 nspe.bin 添加固件清单和证书

   - *> python %IMGTOOL% sign --config %CONFIG_FILE% ^*

     -k %CERT_PATH%\nspe-key.pem ^

     -P %KEY_PASSPHRASE% ^

     -t "NSPE" ^

     --cert %CERT_PATH%\nspe-cert.pem ^

     .\freertos\nspe.bin ^

     %OUTPUT_IMAGE_PATH%\nspe-sign.bin

   - 其中%CONFIG_FILE%就是指%SDK%\config\config_gdm32.h，在组装固件头部时需要填上固件版本号。

- nspe-key.pem 就是 ***NSPE 密钥对和证书***小节中生成的 NSPE 私钥，用来给 NSPE 固件签名。
- “12345678”是前面定义的 MBL PEM 打开密码。
- “NSPE”是指当前固件的类型，其他还有“MBL”，“PROT”和“AROT”。
- nspe-cert.pem 就是 ***NSPE 密钥对和证书***小节中生成的 NSPE 证书。

3) 加密固件（可选）

- *> python %AESTOOL% --c %CONFIG_FILE% ^*

  -t "IMG_%INDEX%_NSPE" ^

  -i %OUTPUT_IMAGE_PATH%\nspe-sign.bin ^

  -o %OUTPUT_IMAGE_PATH%\nspe-sign%AES_SUFFIX%.bin ^

  -k %AESK%
- 由于加密选用地址作为计数器，这里根据"IMG_%INDEX%_NSPE"类型从%CONFIG_FILE%找到相应的可执行程序起始偏移地址，作为计数器。
- %AESK%是加密密钥，需要跟烧录到 EFUSE 的 AESK 保持一致。

4) 转换为 HEX 文件（可选，如果需要通过 JLINK 单独下载）

- *> python %HEXTOOL% --c %CONFIG_FILE% ^*

  -t "IMG_%INDEX%_NSPE" ^

  -e %SREC_CAT% ^

  %OUTPUT_IMAGE_PATH%\nspe-sign%AES_SUFFIX%.bin ^

  %OUTPUT_IMAGE_PATH%\nspe-sign%AES_SUFFIX%.hex
- 同上，通过"IMG_%INDEX%_NSPE"类型从%CONFIG_FILE%中找出相应的可执行程序起始偏移地址和基地址，得到该固件存放的绝对地址。
- 将绝对地址和 nspe-sign-aes.bin 传给%SREC_CAT%（srec_cat.exe），生成 Intel HEX 格式文件 nspe-sign-aes.hex。

与 NSPE 的封装过程相比，MBL 的封装还需要增加一步，就是先生成 sysset.bin，然后跟 mbl.bin 合在一起成为 mbl-sys.bin 之后再进行以上操作。

AfterBuild 脚本

上节中的过程，已经在每个 Project 目录下的 xxxx_afterbuild.bat 中写好了，例如 NSPE，可以找相应的 nspe_afterbuild.bat 来查看以上过程。

当使用 KEIL 编译这些 Project 时，会自动在编译后执行 xxxx_afterbuild.bat 文件，自动完成固件封装，得到相应的二进制文件。

3.4. 固件打包

3.4.1. 打包工具

gentool.py：将封装过的固件打包在一起，生成出厂包或者升级包。

3.4.2. 打包过程

- 出厂包
  - *python gentool.py --config ..\..\config\config_gdm32.h --sys_set ..\images\mbl-sys.bin --nspe_0 ..\images\mp-sign.bin --prot_1 ..\images\prot-sign.bin --nspe_1 ..\images\nspe-sign.bin -o ..\images\image-all-release.bin*
  - 出厂包目前没有自动生成，需要将 Window Command 命令窗口中执行以上命令。在执行命令之前，需要先编译好 SDK 所有可执行程序段，并将原厂发布的产测固件签名后得到 mp-sign.bin 拷贝到%SDK%\scripts\images\目录下，然后将 Command 窗口的目录切换到%SDK%\scripts\imgtool\，执行以上命令即可。
- 升级包
  - *> python %GENTOOL% --config %CONFIG_FILE% ^*
    --prot_0 %OUTPUT_IMAGE_PATH%\prot-sign.bin ^
    --nspe_0 %OUTPUT_IMAGE_PATH%\nspe-sign.bin ^
    -o %OUTPUT_IMAGE_PATH%\image-%VERSION%.bin

升级包在 nspe_afterbuild.bat 脚本中会自动生成。

4. 固件升级

在发布的 SDK 里面提供了 ota_demo.c，用户在开发自己的 OTA 代码时，可以参考。这个示例代码假定已经搭建了一个 HTTP 服务器，并将新固件放在了服务器上。

用户可以通过修改文件开头的宏 DOWNLOAD_URL 来更改新固件所在目录的链接地址，新固件文件名透过调用 ota_demo(bin_name)传递进来。

简单描述设备升级的操作如下：

1) 设备首先确定当前运行的 IMAGE 是 0 还是 1，如果当前是 0，则目标烧写位置为 1，反之为 0；
2) 设备跟 HTTP 服务器建立 TCP 连接；
3) 设备向服务器发送 HTTP Request；
4) 服务器响应 HTTP response 200 OK，将 IMAGE 固件分片发送出来；
5) 设备收到正确响应后擦除目标烧写位置的数据，并依次将分片内容烧写进 FLASH；
6) 服务器在固件包数据发送结束后，发送固件包的校验和；
7) 设备收到校验和之后，读取目标烧写位置的 FLASH 内容计算校验和进行比较；
8) 校验通过则标注目标烧写位置的 Image Status 为 NEW，当前运行位置的 Image Status 为 Old；
9) 重启；
10) MBL 验证新固件的证书和签名，如果验证通过，则跳转到新固件运行。

5. 版本历史

表 5-1. 版本历史

版本号	说明	日期
1.0	初稿发布	2021 年 11 月 23 日

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the “Product”), is owned by the Company under the intellectual property laws and treaties of the People’s Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as it’s suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as it’s suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.