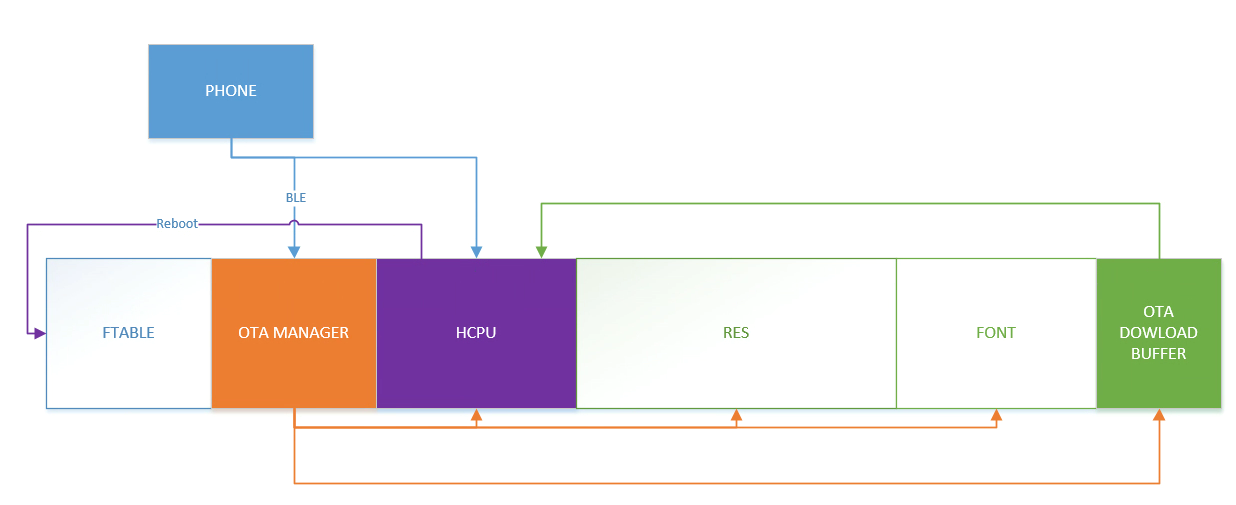
# NOR

## 概述



1. 单独使用ota manager bin负责升级业务。
2. Hcpu 收到升级请求后，重启，代码执行到ota manager后，会停在ota manager，然后进行对应的image下载，flash擦写，只有升级成功后才可以回到正常模式。
3. 如果有足够flash空间，也可以配置ota download buffer，hcpu可以下载压缩形式的代码，即使在hcpu下载中失败，也可以进入到正常系统，需要配置0.7倍HCPU size的空间
4. 56x nor默认都是覆盖方法，没有配置download buffer
5. 如果要升级OTA manager，要至少配置0.7倍OTA manager大小的空间

## 制作bin

示例：

app 0 0参数的含义

* app是bin的名字，比如当前是app.bin，就填app，如果是ER\_IROM1.bin，就填ER\_IROM1
* 第一个数字代表压缩类型，0是覆盖，16是压缩
* 第二个数字是Image ID，所有ID如下

DFU\_IMG\_ID\_HCPU = 0

DFU\_IMG\_ID\_LCPU = 1

DFU\_IMG\_ID\_LCPU\_ROM\_PATCH = 2

DFU\_IMG\_ID\_RES = 3

DFU\_IMG\_ID\_FONT = 4

DFU\_IMG\_ID\_EX = 5

DFU\_IMG\_ID\_OTA\_MANAGER = 6

DFU\_IMG\_ID\_TINY\_FONT = 7

* 只生成HCPU

gen\_dfu --img\_para app 0 0 --key=s01 --sigkey=sig --dfu\_id=1 --hw\_ver=51 --sdk\_ver=90006 --fw\_ver=1001001 --com\_type=0

* 生成hcpu，资源，字体

gen\_dfu --img\_para app 0 0 res 0 3 font 0 4 --key=s01 --sigkey=sig --dfu\_id=1 --hw\_ver=51 --sdk\_ver=90006 --fw\_ver=1001001 --com\_type=0

* ota manager只能单独升级，只支持压缩，且DFU\_ID = 2

run imgtool\_new.py gen\_dfu --img\_para ota 16 6 --key=s01 --sigkey=sig --dfu\_id=2 --hw\_ver=51 --sdk\_ver=7001 --fw\_ver=1001001 --com\_type=0

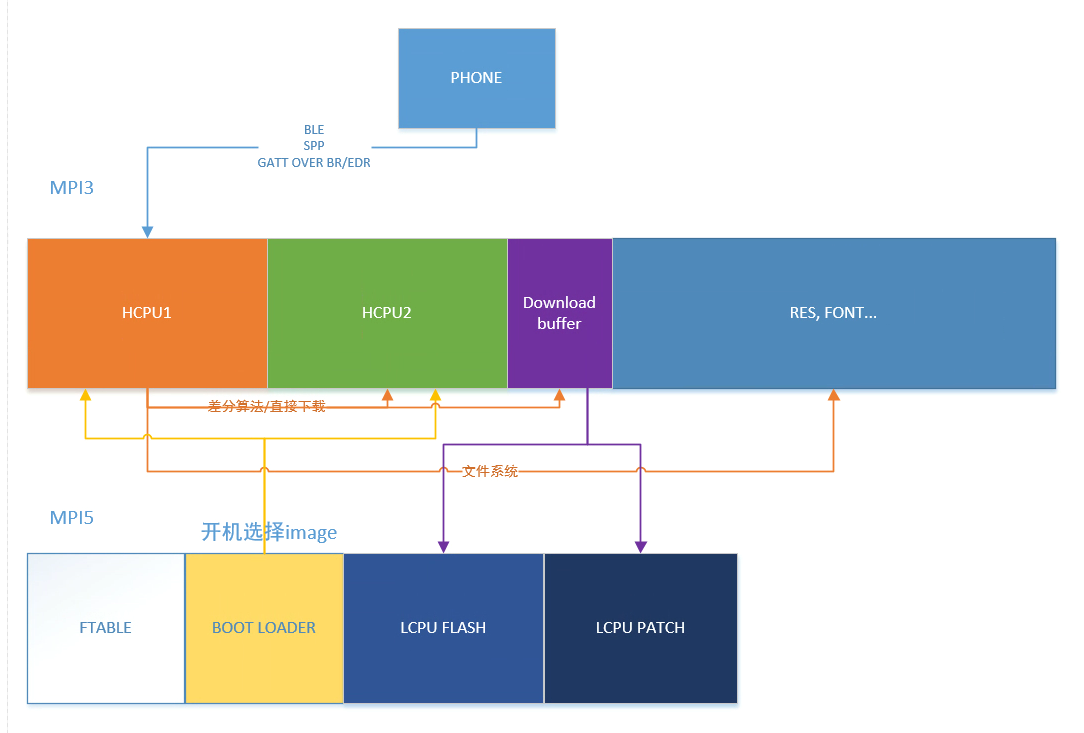
* 在nor上，LCPU ROM PATCH必须使用压缩的方式升级，参数16 2

例如，制作包含了HCPU，LCPU FLASH，LCPU ROM PATCH的升级包

gen\_dfu --img\_para app 0 0 lcpu 0 1 lcpu\_rom\_patch 16 2 --key=s01 --sigkey=sig --dfu\_id=1 --hw\_ver=51 --sdk\_ver=90006 --fw\_ver=1001001 --com\_type=0

# NAND

## 概述

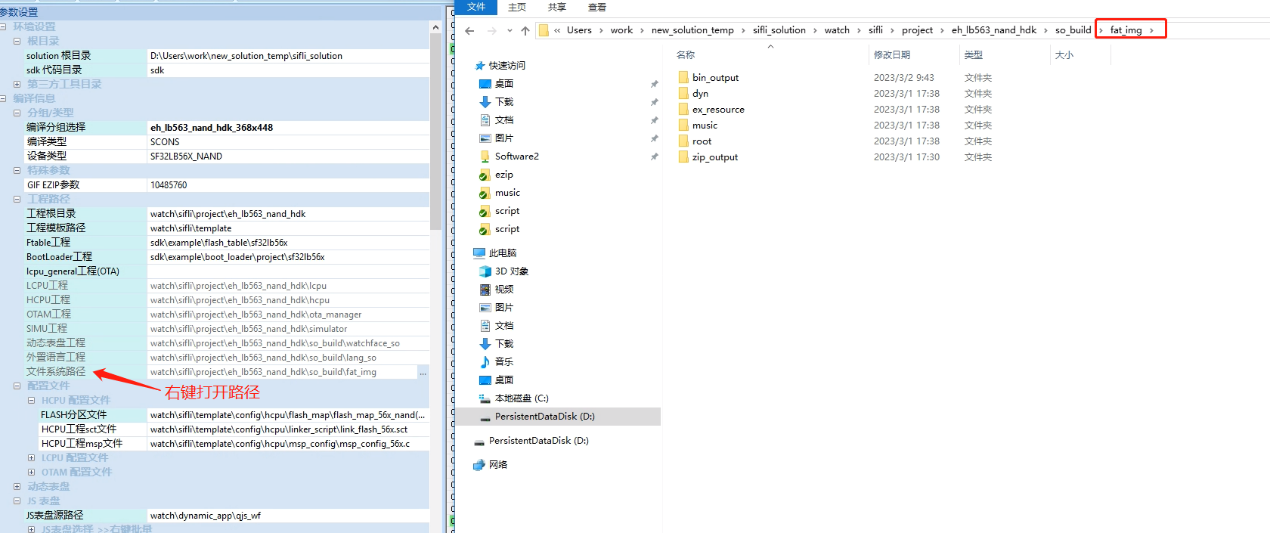


* 主要包含升级资源（文件部分），升级HCPU（image部分）。
* 升级HCPU部分和NOR升级基本一致
* 资源部分只会升级待升级的具体文件，通过当前HCPU下载，下载后使用文件系统进行相应的增，删，改操作。
* HCPU部分，通过pingpong的方式升级，当前工作的HCPU，会负责下载备用HCPU，
* 当前是直接下载的完整HCPU bin，将来会考虑下载差分patch，通过差分算法得到新的HCPU bin，可以减少传输内容。
* LCPU FLASH，LCPU ROM，如果需要升级，会下载到下载缓存区域，然后在OTA最后阶段，更新到MPI5。
* OTA之后会重启，如果HCPU的运行bin发生了切换，则会在boot loader中选择期望的HCPU进行跳转

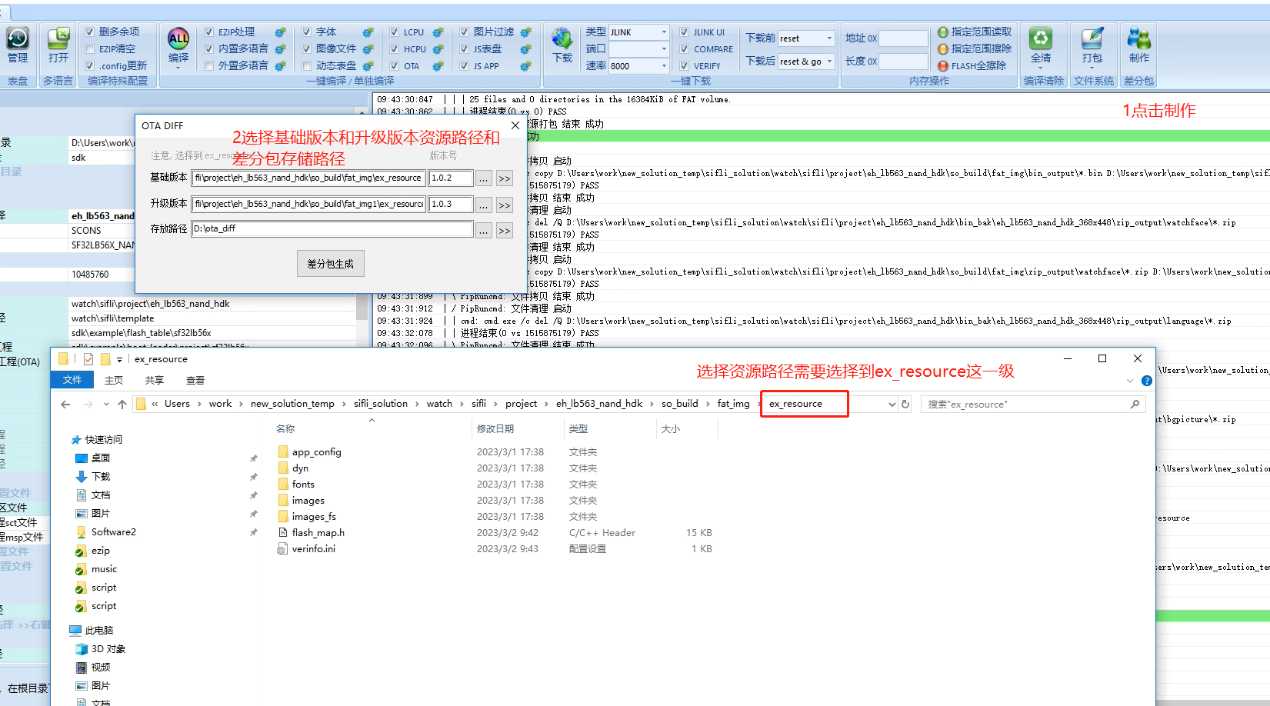
## 制作文件升级包

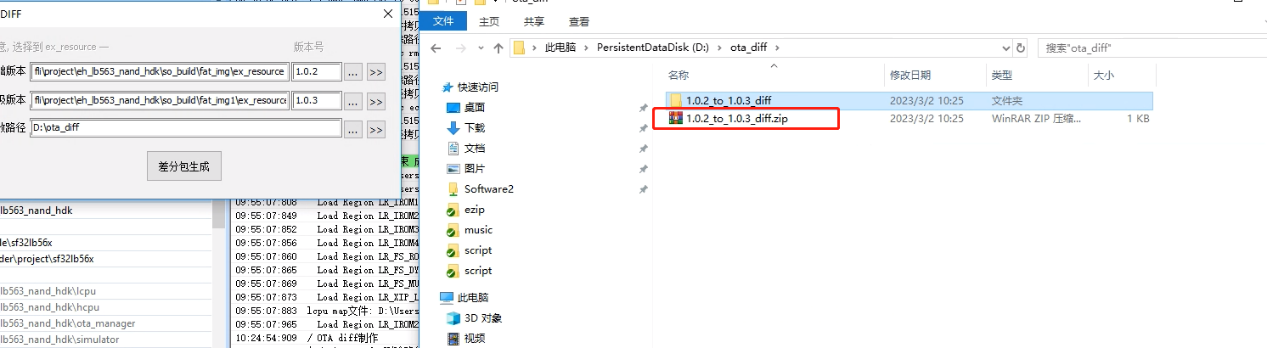
1：编译基础版本，保存fat\_img文件夹，如果升级工程和基础版本不是同一工程，需要copy到其他路径或者重命名，避免编译升级版本覆盖

2：编译升级版本，保存fat\_img文件夹



3:差分包制作，生成的zip包就是基础版本到升级版本的资源包





## 制作bin

示例：

app 0 0参数的含义

* app是bin的名字，比如当前是hcpu.bin，就填hcpu，如果是ER\_IROM1.bin，就填ER\_IROM1
* 第一个数字代表压缩类型，0是覆盖，16是压缩
* 第二个数字是Image ID，所有ID如下

IMAGE\_ID\_HCPU = 0

IMAGE\_ID\_LCPU = 1

IMAGE\_ID\_HCPU\_PATCH = 2

IMAGE\_ID\_FS\_ROOT = 3

IMAGE\_ID\_LCPU\_PATCH = 4

IMAGE\_ID\_FS\_DYN = 5

IMAGE\_ID\_FS\_MUSIC = 6

如果选择制作包含如下类型的升级文件

MAGE\_ID\_FS\_ROOT

IMAGE\_ID\_FS\_DYN

IMAGE\_ID\_FS\_MUSIC

是把文件系统的bin，做完整的擦除和升级，文件系统的bin比较大，全部擦写的话，升级慢，不推荐在用户手上完整升级此类文件。如果有升级需求，建议使用前面提到的差分制作文件升级包。

HCPU\_PATCH暂未使用

* 相比于nor的执行参数，增加了--bksize=2048

gen\_dfu --img\_para hcpu 0 0 --key=s01 --sigkey=sig --dfu\_id=1 --hw\_ver=51 --sdk\_ver=7001 --fw\_ver=1001001 --com\_type=0 --bksize=2048

gen\_dfu --img\_para ER\_IROM1 0 0 lcpu\_flash 0 1 lcpu\_rom\_patch 0 4 --key=s01 --sigkey=sig --dfu\_id=1 --hw\_ver=51 --sdk\_ver=7001 --fw\_ver=1001001 --com\_type=0 --bksize=2048

# Demo app

## Nor:

CTRL: 选择ctrl文件

HCPU：选择制作好的HCPU升级包，名字是outER\_IROM1.bin，如果本身的名字是ER\_IROM1.bin

RES：同上

Font：同上

Nor old是给55x和早期56x使用

Nor优化了部分OTA开始的命令

## Nand:

CTRL: 选择ctrl文件

HCPU:选择制作好的HCPU升级包，名字是outER\_IROM1.bin，如果本身的名字是ER\_IROM1.bin

LCPU和LCPU PATCH同理可选。

NAND RES：资源升级包zip

Start开始升级，resume尝试续传

