

IAP升级核心结构：IAP-BIN

介绍：

- 为确保不是任意BIN文件都可被升级到某一个项目的设备中，对于IAP的设备，需要添加管控策略；
- 每个BIN文件如果想通过IAP被升级到设备中，需要在该BIN文件的头部添加一个128字节的header信息；
- 128字节的header信息有项目管控信息、公司区分信息、芯片区分信息、硬件版本、软件版本、升级管控信息、BIN文件本身加密信息等，尽量保证IAP升级的安全性；
- 为使管控信息可配，并可合入到原始BIN文件，并生成一个新的BIN文件，需要制作一个IAP-BIN文件生成工具；

升级header格式说明

序号	数据项	长度	偏移地址	数据类型	说明	配置方法	设备端存放位置
1	头文件标识	8	0	字符	固定为“INGCHIPS”	仅ini文件	【BOOT】
2	芯片代号	16	8	字符	各字节含义： 1：芯片代号有效长度len 2~16：芯片代号【最大15字节，不足部分补0xFF】 例如：[1]=12, [2~16] = "ING91683C_TB"	UI+ini	【APP】
3	项目代号	24	24	字符	各字节含义： 1：客户代号有效长度len 2~23：芯片代号【最大23字节，不足部分补0xFF】 例如：[1]=5, [2~23] = "HS_KB"	UI+ini	【APP】
4	硬件版本	6	48	字符	各字节含义： 1：固定为V，标识版本的开始 2：大版本号x（字符0~9） 3：英文点号	UI+ini	【APP】

					4：中版本号y（字符0~9） 5：英文点号 6：小版本号z（字符0~9） 版本号为：Vx.y.z 例如：V2.1.3（不支持V2.1.13这样的版本号）		
5	软件版本	6	54	数字	各字节含义： 1：固定为V，标识版本的开始 2：大版本号x（字符0~9） 3：英文点号 4：中版本号y（字符0~9） 5：英文点号 6：小版本号z（字符0~9） 版本号为：Vx.y.z 例如：V2.1.3（不支持V2.1.13这样的版本号）	UI+ini	【APP】
6	总校验信息	6	60	HEX	各字节含义： 1：校验类型：0代表CRC校验，1代表 和校验SUM ，其它值保留； 2：校验数据长度len 3~6：校验数据，有效长度为len的值，最大4字节 其中， 和校验 只2字节有效，小端模式。	校验类型： UI+ini 其它见附加说明。	【BOOT】
7	每块大小	2	66	HEX	各字节含义【小端模式】： 1: 低字节 2: 高字节 范围：12~8192字节之间	UI+ini	【BOOT】
8	升级总块数	2	68	HEX	各字节含义【小端模式】： 1: 低字节 2: 高字节	自动计算 不可配置	【BOOT】

9	升级类型	1	70	HEX	各字节含义： 1：升级类型标识： 0x00 = 代表仅升级APP应用程序； 0x01 = 代表升级platform+APP程序； 【键盘软件暂不支持】 0x02 = 代表升级platform+二级boot； 【键盘软件暂不支持】 0x03 = 代表升级platform+二级boot+APP程序； 【键盘软件暂不支持】 其它值保留；	UI+ini	【BOOT】
10	BIN加密信息	35	71	HEX	各字节含义： 1：是否启用加密：0不启用，1启用内置密钥加密 2：加密方式：0异或加密XOR，1 AES128加密； 3：密钥长度len； 4~19：密钥信息，长度为len，最大16字节； 19~35：iv信息，AES128专用，异或加密不用，最大16字节； 详情请见附加说明。	UI+ini	【BOOT】
11	load_address	4	106	HEX	新版程序的加载地址，对齐到sector，升级platform+app时固定为0x02003000		
12	bin_size	4	110	HEX	原始bin文件的大小，升级platform+app时为merge后的bin文件大小		
13	填充字段	12	114	HEX	预留字段，将header填充，默认值0xFF	\	\
14	CRC	2	126	HEX	对header的校验，校验范围：[0, 126)，校验算法一致(crc16_modbus)		

附加说明

IAP文件输出命名：

- INGIAP_HS_KB_HW2_1_3_SW1_0_2_CRC_A_N_20230818_1340.bin
- 各字段含义：
 - **INGIAP**: 文件标识;
 - **HS_KB**: 取项目代号前5个字符（如果大于等于5，否则有多少取多少）；
 - **HW2_1_3**: 硬件版本，见硬件版本；
 - **SW1_0_2**: 软件版本，见软件版本；
 - **A**: 代表升级类型，**A**: APP，**PA**: platform+APP，**PB**: platform+BOOT，**PBA**: platform+BOOT+APP; **X**: 其它。
 - **N**: 代表加密类型，**N**: 不加密，**Y0**: BCC加密，**Y1**: AES128加密; **X**: 其它。
 - **20230818**: 日期2023年8月18日；
 - **1340**: 时间，24小时制，13:40；

BIN文件填充

- 如果加载的BIN文件的大小不是**16字节**的整数倍，则填充**0xFF**到**16字节**整数倍，作为新的BIN文件。
- 后续的**校验**和**加密**等等，均是对**新的原始BIN文件**进行操作的。其中**校验**是对**明文**进行的。

校验数据相关

- 校验类型在UI界面可选，其它项数据均为自动生成，最终校验值可在UI显示；
- 校验对象为bin加载路径的bin文件，bin文件永远只有一个，如果用户要升级platform+二级boot+APP程序，请自行使用downloader的merge功能将它们合并为一个bin文件；
- **CRC校验**: 多项式= 1021 CRC-16/X25，初值0xFFFF，**CRC-16(Modbus)**;

例如：

如下十六进制数据计算的CRC值为**0x786C**：

11 22 33 44 44 44 44 45 44 44 45 63 46 34 56 34 3D DD DE DD EC F4 00 22 22 22 22 22 22 22 22 22

可通过如下网址的工具验证：

<https://www.23bei.com/tool/59.html>

16进制(CRC16)(MODBUS RTU通讯)校验码在线计算器

输入1-2个标题中的文字

搜索

直达

字节数(10进制)	32
字节数(16进制)	20
CRC-16(MSB-LSB)	6C78
CRC-16(Modbus)	786C

11 22 33 44 44 44 45 44 44 44 45 63 46 34 56 34 3D DD DE DD EC F4 00 22 22 22 22 22 22 22 22 22 22

计算

清除

- 和校验：采用16bit，小端模式，即将新BIN文件每个字节累加值，最后只保留最低有效位的16bit作为和校验；

BIN加密信息

- BIN加密**指的是利用密钥对原始BIN文件加密后与HEADER部分组成新的BIN文件，这样IAP-BIN就被密钥保护起来了，注意，**HEADER信息不会被加密**；
- 当用户选择**不启用加密**时，加密方式，密钥长度和密钥各项均无效；
- 当用户选择启用**内置密钥**加密时，设备解密的密钥来自于**预先设置值**，开发者需要确保程序内部密钥和IAP-BIN生成工具上填写的密钥两者是一致的，否则无法升级。当选择内置密钥时，则用户需要在UI界面选择一种加密方式，并输入16字节密钥信息，原始BIN文件会被密钥加密；因为HEADER信息不会被加密，所以HEADER中的密钥值处填入的不是原始密钥，而是用原始密钥对自身进行加密后的密文，程序收到该密文值后会解密，并与自身解密密钥比对，二者相同时才能升级；
- XOR加密**：取16字节XOR密钥，然后分别对原始BIN数据的每16字节异或，替换原来的16字节值，直到将所有数据进行异或加密；接收方用同样的方式进行异或解密；这种方式更严格的说法叫混淆，并不是加密，因为密钥是公开的。
- 需要特别注意的是，HEADER中的总校验信息是针对解密后的BIN文件进行校验的，**而不是针对密文进行校验**；

配置方法

- ini**：即生成一个ini配置文件，用来保存配置信息，用户可打开该配置文件修改信息，下一次软件打开时会加载新的配置；且此配置会被加载到header中；
- UI**：指的是PC软件的用户配置页面，表明UI的选项需要在界面设置配置框，用户可在配置框上进行配置，且配置会被保存到ini配置文件中；同时，也会被修改到header中；

大小端问题

- 没有特殊说明，则字节序均根据flash特性选用**小端模式**，即低字节在前，高字节在后；这样可以简化C程序；
- 举例说明：
 - 假设我们生成**CRC16 = 0x1234**
 - 那么传输时，先传输**0x34**，再传输**0x12**

PC软件偏移问题

- 如果PC端在做某些字段时需要凑4字节整数倍，可利用最后预留字节进行拼凑，此时需要更改上面升级header表格的某些字段的位置，可灵活调整；请与设备端开发者沟通。

设备端CRC16计算

代码块

```
1 //使用ing916 rom程序中的crc16，它的计算速度会快一些
2 /**
3
4     *****
5     *****
6     * @brief Calculate a 16bit CRC code (Polynomial  $x^{16}+x^{15}+x^5+1$ )
7     *
8     * @param[in] buffer      input bytes
9     * @param[in] len         data length
10    * @return                CRC result
11
12    *****
13    *****
14    */
15 typedef uint16_t (* f_crc_t)(uint8_t *buffer, uint16_t len);
16 #define crc      ((f_crc_t)(0x00001d21))
17
18 // 演示用法
19 static uint16_t IAP_Get_CRC(uint8_t *buffer, uint16_t len){
20     return crc(buffer, len);
21 }
22 // 该方法对应如下网址: https://www.23bei.com/tool/59.html 详见上方校验数据相关 说明
```

