

# Rockchip Baseparameter 分区格式定义和使用说明

---

文件标识: RK-YH-YF-194

发布版本: V1.0.0

日期: 2021-08-27

文件密级: ☐绝密 ☐秘密 ☐内部资料 ☒公开

## 免责声明

本文档按“现状”提供, 瑞芯微电子股份有限公司 (“本公司”, 下同) 不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因, 本文档将可能在未经任何通知的情况下, 不定期进行更新或修改。

## 商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标, 归本公司所有。

本文档可能提及的其他所有注册商标或商标, 由其各自拥有者所有。

版权所有 © 2021 瑞芯微电子股份有限公司

超越合理使用范畴, 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。

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址: 福建省福州市铜盘路软件园A区18号

网址: [www.rock-chips.com](http://www.rock-chips.com)

客户服务电话: +86-4007-700-590

客户服务传真: +86-591-83951833

客户服务邮箱: [fae@rock-chips.com](mailto:fae@rock-chips.com)

前言

概述

产品版本

芯片名称	内核版本
RK所有平台	Linux 4.19/Linux 5.10

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	黄家钊、林垚	2021-08-27	初始版本

# 目录

## Rockchip Baseparameter 分区格式定义和使用说明

1. 说明
2. Baseparameter 分区
  - 2.1 分区拓扑图
  - 2.2 Baseparameter 结构
    - 2.2.1 struct baseparameter\_info
    - 2.2.2 struct disp\_header
    - 2.2.3 struct disp\_info
    - 2.2.4 占用存储空间
  - 2.3 Baseparameter 基础结构说明
    - 2.3.1 struct screen\_info
    - 2.3.2 struct besh\_info
    - 2.3.3 struct csc\_info
    - 2.3.4 struct acm\_data
    - 2.3.5 struct overscan\_info
    - 2.3.6 struct gamma\_lut\_data
    - 2.3.7 struct cubic\_lut\_data
    - 2.3.8 struct framebuffer\_info
    - 2.3.9 struct reserved[244]
  - 2.4 Baseparameter 备份分区
3. Baseparameter 读写 API 说明
4. Baseparameter 解析工具使用说明
5. Android 软件配置流程和代码路径

# 1. 说明

Baseparameter 分区用于存储 Rockchip 平台显示分辨率、显示效果调节的配置等信息，确保关机重启后能维持和关机之前一样的效果，并确保整个开机过程显示效果保持一致。

## 2. Baseparameter 分区

### 2.1 分区拓扑图



### 2.2 Baseparameter 结构

## 2.2.1 struct baseparameter\_info

```
struct baseparameter_info base_info {
    char head_flag[4]; /* 头标识, "BASP" */
    u16 major_version; /* Baseparameter 大版本信息 */
    u16 minor_version; /* Baseparameter 小版本信息 */
    struct disp_header disp_header[8]; /* 通过head可以正确找到每一个显示设备的偏移,按现在每个disp_info的大小,最多支持8个disp */
    struct disp_info disp_info[8]; /* 显示设备信息 */
}
```

## 2.2.2 struct disp\_header

```
struct disp_header {
    u32 connector_type; /* 显示设备类型 */
    u32 connector_id; /* 显示设备id */
    u32 offset; /* disp_info 偏移 */
}
```

占用存储空间: 12 Byte。

## 2.2.3 struct disp\_info

```
struct disp_info {
    char disp_head_flag[6]; /* disp 头标识, "DISP_N", N 可以是0-7, size: 6 Byte*/
    struct screen_info screen_info[4]; /* 支持热插拔的设备接不同的设备, 如DP出来可能接DP->HDMI 或者 DP->VGA, size: 72 * 4 = 288 Byte */
    struct bcsh_info bcsh_info; /* 节亮度、对比度、饱和度、色度信息, size: 8 Byte */
    struct overscan_info overscan_info; /* 过扫描信息, size: 12 Byte */
    struct gamma_lut_data gamma_lut_data; /* gamma 信息, size: 6146 Byte */
    struct cubic_lut_data cubic_lut_data; /* 3D lut信息, size: 29480 Byte */
    struct framebuffer_info framebuffer_info; /* framebuffer信息, size: 12 Byte */
    /*
    u32 cacm_header; /*CACM头标识*/
    u32 reserved[243]; /* 预留, size: 972 Byte*/
    u32 crc; /* CRC 校验, size: 4 Byte */
    struct csc_info csc_info; /* 调节csc信息, size: 22 Byte */
    struct acm_data acm_data; /*调节acm信息, size: 2642 Byte*/
    u8 resv2[10*1024]; /* */
    u32 crc2;
    */
}
```

占用存储空间: 49886 Byte。

## 2.2.4 占用存储空间

struct disp\_info 占用存储空间: 49884 Byte;

struct disp\_header 占用存储空间: 12 Byte;

struct baseparameter\_info 合计占用存储空间：4 + 2 + 2 + 8 x 12 + 8 x 49884 = 399188 Byte。

Baseparameter 主分区和备份分区的大小分别是512KB，目前用到的有效数据是 399188 Byte，剩下未使用的空间预留；

## 2.3 Baseparameter 基础结构说明

### 2.3.1 struct screen\_info

struct screen\_info: 用于保持显示接口的类型、分辨率时序等信息，占用 72 Byte。

```
struct screen_info {
    u32 type; /* connector 类型, 4 bytes */
    u32 id; /* 4 byte, 用于区别相同 type 的不同设备 */
    struct drm_display_mode resolution; /* 52 bytes */
    enum output_format format; /* 4 bytes */
    enum output_depth depth; /* 4 bytes */
    u32 feature; /* 4 bytes */
};
```

- type: 屏的接口类型

```
#define DRM_MODE_CONNECTOR_Unknown 0
#define DRM_MODE_CONNECTOR_VGA 1
#define DRM_MODE_CONNECTOR_DVII 2
#define DRM_MODE_CONNECTOR_DVID 3
#define DRM_MODE_CONNECTOR_DVIA 4
#define DRM_MODE_CONNECTOR_Composite 5
#define DRM_MODE_CONNECTOR_SVIDEO 6
#define DRM_MODE_CONNECTOR_LVDS 7
#define DRM_MODE_CONNECTOR_Component 8
#define DRM_MODE_CONNECTOR_9PinDIN 9
#define DRM_MODE_CONNECTOR_DisplayPort 10
#define DRM_MODE_CONNECTOR_HDMIA 11
#define DRM_MODE_CONNECTOR_HDMIB 12
#define DRM_MODE_CONNECTOR_TV 13
#define DRM_MODE_CONNECTOR_eDP 14
#define DRM_MODE_CONNECTOR_VIRTUAL 15
#define DRM_MODE_CONNECTOR_DSI 16
#define DRM_MODE_CONNECTOR_DPI 17
#define DRM_MODE_CONNECTOR_WRITEBACK 18
```

- struct drm\_display\_mode: 扫描时序信息

```
struct drm_display_mode {
    /* Proposed mode values */
    int clock; /* in kHz */
    int hdisplay;
    int hsync_start;
    int hsync_end;
    int htotal;
    int vdisplay;
```

```

int vsync_start;
int vsync_end;
int vttotal;
int vrefresh;
int vscan;
unsigned int flags;
int picture_aspect_ratio;
};

```

- output\_format: 颜色格式，属性字符串为Auto时，选择output\_ycbcr\_high\_subsampling

```

enum output_format {
    output_rgb=0,
    output_ycbcr444=1,
    output_ycbcr422=2,
    output_ycbcr420=3,
    output_ycbcr_high_subsampling=4, /* (YCbCr444 > YCbCr422 > YCbCr420 > RGB)
*/
    output_ycbcr_low_subsampling=5, /* (RGB > YCbCr420 > YCbCr422 > YCbCr444) */
    invalid_output=6,
};

```

- depthc: 色深，属性字符串为Auto时，选择Automatic

```

enum output_depth{
    Automatic=0,
    depth_24bit=8,
    depth_30bit=10,
};

```

feature: feature 目前有如下flag，配置分辨率以及颜色的AUTO模式，是否开启hdcplx，是否过滤分辨率列表

```

#define RESOLUTION_AUTO          (1<<0)
#define COLOR_AUTO              (1<<1)
#define HDCP1X_EN               (1<<2)
#define RESOLUTION_WHITE_EN     (1<<3)

```

### 2.3.2 struct bcsb\_info

保存bcsb信息，其中bcsb取值范围0~100，默认值都是50，用于调节亮度、对比度、饱和度、色度

```

struct bcsb_info {
    unsigned short brightness;
    unsigned short contrast;
    unsigned short saturation;
    unsigned short hue;
};

```

### 2.3.3 struct csc\_info

保存csc信息，默认值都是256，用于调节亮度、对比度、饱和度、色度，颜色格式转换等，3.0之后版本才是用

```
struct csc_info {
    u16 hue;
    u16 saturation;
    u16 contrast;
    u16 brightness;
    u16 r_gain;
    u16 g_gain;
    u16 b_gain;
    u16 r_offset;
    u16 g_offset;
    u16 b_offset;
    u16 cscEnable;
};
```

### 2.3.4 struct acm\_data

保存颜色管理参数，3.0之后版本才是用

```
#define ACM_GAIN_LUT_HY_TOTAL_LENGTH    (9 * 17 * 3)
#define ACM_GAIN_LUT_HS_TOTAL_LENGTH    (13 * 17 * 3)
#define ACM_DELTA_LUT_H_TOTAL_LENGTH    (65 * 3)

struct acm_data {
    s16 delta_lut_h[ACM_DELTA_LUT_H_TOTAL_LENGTH];
    s16 gain_lut_hy[ACM_GAIN_LUT_HY_TOTAL_LENGTH];
    s16 gain_lut_hs[ACM_GAIN_LUT_HS_TOTAL_LENGTH];
    u16 y_gain;
    u16 h_gain;
    u16 s_gain;
    u16 acm_enable;
};
```

### 2.3.5 struct overscan\_info

过扫描缩放系数

```
struct overscan_info {
    unsigned int maxvalue;
    unsigned short leftscale;
    unsigned short rightscale;
    unsigned short topscale;
    unsigned short bottomscale;
};
```



### 2.3.6 struct gamma\_lut\_data

保存gamma lut data信息，size表示每个rgb lut表中有多少个数据，最大为1024

```
struct gamma_lut_data{
    u16 size;
    u16 lred[1024];
    u16 lgreen[1024];
    u16 lblue[1024];
};
```

### 2.3.7 struct cubic\_lut\_data

保存cubic lut data信息，size表示每个rgb lut表中有多少个数据，最大为 $17 \times 17 \times 17 = 4913$

```
struct cubic_lut_data{
    u16 size;
    u16 lred[4913];
    u16 lgreen[4913];
    u16 lblue[4913];
};
```

### 2.3.8 struct framebuffer\_info

预设 framebuffer 信息

```
struct framebuffer_info {
    u32 framebuffer_width;
    u32 framebuffer_height;
    u32 fps;
};
```

### 2.3.9 struct reserved[244]

预留配置信息

## 2.4 Baseparameter 备份分区

baseparameter 的后半部分用来保存初始数据，恢复出厂设置的时候将备份分区的数据拷贝到主分区。

## 3. Baseparameter 读写 API 说明

---

为方便应用层对 Baseparameter 分区的读写，将以库的形式提供统一的 API 供不同的应用调用。

库的源码位置在：hardware/rockchip/libbaseparameter

API说明：

```
bool have_baseparameter(); /* 判断是否有baseparameter分区 */
int dump_baseparameter(const char *file_path); /* 导出baseparameter分区数据到文件 */
int get_disp_info(unsigned int connector_type, unsigned int connector_id, struct
disp_info *info); /* 获取disp_info */
int set_disp_info(unsigned int connector_type, unsigned int connector_id, struct
disp_info *info); /* 设置disp_info */
int get_screen_info(unsigned int connector_type, unsigned int connector_id, int
index, struct screen_info *screen_info); /* 获取screen_info */
int set_screen_info(unsigned int connector_type, unsigned int connector_id, int
index, struct screen_info *screen_info); /* 设置screen_info */
unsigned short get_brightness(unsigned int connector_type, unsigned int
connector_id); /* 获取亮度 */
unsigned short get_contrast(unsigned int connector_type, unsigned int
connector_id); /* 获取对比度 */
unsigned short get_saturation(unsigned int connector_type, unsigned int
connector_id); /* 获取饱和度 */
unsigned short get_hue(unsigned int connector_type, unsigned int connector_id);
/* 获取色调 */
int set_brightness(unsigned int connector_type, unsigned int connector_id,
unsigned short value); /* 设置亮度 */
int set_contrast(unsigned int connector_type, unsigned int connector_id, unsigned
short value); /* 设置对比度 */
int set_saturation(unsigned int connector_type, unsigned int connector_id,
unsigned short value); /* 设置饱和度 */
int set_hue(unsigned int connector_type, unsigned int connector_id, unsigned
short value); /* 设置色调 */
int get_overscan_info(unsigned int connector_type, unsigned int connector_id,
struct overscan_info *overscan_info); /* 获取overscan_info */
int set_overscan_info(unsigned int connector_type, unsigned int connector_id,
struct overscan_info *overscan_info); /* 设置overscan_info */
int get_gamma_lut_data(unsigned int connector_type, unsigned int connector_id,
struct gamma_lut_data *data); /* 获取gamma数据 */
int set_gamma_lut_data(unsigned int connector_type, unsigned int connector_id,
struct gamma_lut_data *data); /* 设置gamma数据 */
int get_cubic_lut_data(unsigned int connector_type, unsigned int connector_id,
struct cubic_lut_data *data); /* 获取3dlut数据 */
int set_cubic_lut_data(unsigned int connector_type, unsigned int connector_id,
struct cubic_lut_data *data); /* 设置3dlut数据 */
int set_disp_header(unsigned int index, unsigned int connector_type, unsigned int
connector_id); /* 设置connector_id, connector_type和分区中disp_info的对应关系 */
bool validate(); /* 判断分区数据是否有效 */
int get_framebuffer_info(unsigned int connector_type, unsigned int connector_id,
framebuffer_info *info);
int set_framebuffer_info(unsigned int connector_type, unsigned int connector_id,
framebuffer_info *info);
int get_baseparameter_info(unsigned int index, baseparameter_info *info);
int set_baseparameter_info(unsigned int index, baseparameter_info *info);
```

## 4. Baseparameter 解析工具使用说明

saveBaseParameter 是 Android上 用来对 baseparameter 分区管理工具，实现了打印、修改分区数据等功能。

工具的源码位置：device/rockchip/common/baseparameter/saveBaseParameter

使用说明：

```
saveParameter: read and write baseparameter partition tool

Usage:
    -h          Help info
    -p          Print Baseparameter
    -t          output to target file (e: "/sdcard/baseparameter.img")
    -f          Framebuffer Resolution (e: 1920x1080@60)
    -c          Color (e: RGB-8bit or YCBCR444-10bit)
    -u          Is Enable Auto Resolution (auto resolution:"auto";set one fixed
resolution:hdisplay,vdisplay,vrefresh,hsync_start,hsync_end,htotal,vsync_start,vs
ync_end,vttotal,vscan,flags,clock e:
"1920,1080,60,2008,2052,2200,1084,1089,1125,0,5,148500")
    -o          Overscan (left,top,right,bottom e: overscan "100,100,100,100")
    -b          BCSH (brightness,contrast,saturation,hue e: "50,50,50,50")
    -R          Reset Baseparameter (1:only reset user setting baseparameter
partition; 2:reset baseparameter paratition include backup)
    -C          Choose Connector type and id to Setting (e: 11,0 or 16,0)

Example: saveBaseParameter -C "16,0" -f "1920x1080@60" -c Auto -u 2 -o
"100,100,100,100" -b "50,50,50,50"
```

## 5. Android 软件配置流程和代码路径

DisplayAjust源码位置：packages/apps/DisplayAdjust

流程图：

