源码位置frameworks/base/core/java/android/os/BatteryStats.java

**关于亮度级别和亮度时间处理的方法**

亮度的分级，目前是5个

**public static final int** NUM\_SCREEN\_BRIGHTNESS\_BINS = 5;

**static final String**[] SCREEN\_BRIGHTNESS\_NAMES = {

"dark", "dim", "medium", "light", "bright"

};

**static final String**[] SCREEN\_BRIGHTNESS\_SHORT\_NAMES = {

"0", "1", "2", "3", "4"

};

**public static final int** STATE\_BRIGHTNESS\_SHIFT = 0;

**public static final int** STATE\_BRIGHTNESS\_MASK = 0x7;//0b0000,0111

将亮度的等级封装在了一个BitDescription[]数组中，作为其中一个子项

**public static final** **BitDescription**[] HISTORY\_STATE\_DESCRIPTIONS = **new BitDescription**[] {

**new** BitDescription(**HistoryItem**.STATE\_CPU\_RUNNING\_FLAG, "running", "r"),

**new** BitDescription(**HistoryItem**.STATE\_WAKE\_LOCK\_FLAG, "wake\_lock", "w"),

...

**new** BitDescription(**HistoryItem**.STATE\_BRIGHTNESS\_MASK,

**HistoryItem**.STATE\_BRIGHTNESS\_SHIFT, "brightness", "Sb",

SCREEN\_BRIGHTNESS\_NAMES, SCREEN\_BRIGHTNESS\_SHORT\_NAMES),

...

};

**public static final class BitDescription** {

**public final int** mask;

**public final int** shift;

**public final String** name;

**public final String** shortName;

**public final String**[] values;

**public final String**[] shortValues;

**public** BitDescription(**int** mask, **String** name, **String** shortName) {

**this**.mask = mask;

**this**.shift = -1;

**this**.name = name;

**this**.shortName = shortName;

**this**.values = **null**;

**this**.shortValues = **null**;

}

**public** BitDescription(**int** mask, **int** shift, **String** name, **String** shortName,

**String**[] values, **String**[] shortValues) {

**this**.mask = mask;

**this**.shift = shift;

**this**.name = name;

**this**.shortName = shortName;

**this**.values = values;

**this**.shortValues = shortValues;

}

}

亮度的计算方法

frameworks/base/core/java/com/android/internal/os/BatteryStatsImpl.java

**public void** noteScreenBrightnessLocked(**int** brightness) {

// Bin the brightness.

**int** bin = brightness / (256/NUM\_SCREEN\_BRIGHTNESS\_BINS);//亮度级别bin = 当前亮度 / (256/5)，可能的结果是0~4

**if** (bin < 0) bin = 0;

**else if** (bin >= NUM\_SCREEN\_BRIGHTNESS\_BINS) bin = NUM\_SCREEN\_BRIGHTNESS\_BINS-1;

**if** (mScreenBrightnessBin != bin) {//上次记录的亮度级别和这次计算的bin不相等

//自开机后，经过的时间(毫秒数)，包括深度睡眠的时间，即实际消逝的时间

**final long** elapsedRealtime = mClocks.elapsedRealtime();

//自开机后，经过的时间(毫秒数)，不包括深度睡眠的时间，即处于唤醒状态时的时间

**final long** uptime = mClocks.uptimeMillis();

//先跟0xffff,fff8相与，然后跟bin<<0相或，bin的取值(0b0000, 0b0001, 0b0010, 0b0011, 0b0100)

mHistoryCur.states = (mHistoryCur.states&~**HistoryItem**.STATE\_BRIGHTNESS\_MASK)//~0b0000,0111

| (bin << **HistoryItem**.STATE\_BRIGHTNESS\_SHIFT);//<<0

**if** (DEBUG\_HISTORY) **Slog**.v(TAG, "Screen brightness " + bin + " to: "

+ **Integer**.toHexString(mHistoryCur.states));

//记录时间信息，详细请看窗口1-addHistoryRecordLocked分析

addHistoryRecordLocked(elapsedRealtime, uptime);

**if** (mScreenState == Display.STATE\_ON) {

**if** (mScreenBrightnessBin >= 0) {//如果屏幕是亮着的，停止上次记录对应该亮度级别的timer

mScreenBrightnessTimer[mScreenBrightnessBin].stopRunningLocked(elapsedRealtime);

}

mScreenBrightnessTimer[bin].startRunningLocked(elapsedRealtime);//开启bin对应亮度级别的timer

}

mScreenBrightnessBin = bin;//记录本次的亮度级别bin

}

}

**窗口1-addHistoryRecordLocked分析**

**void addHistoryRecordLocked**(**long** elapsedRealtimeMs, **long** uptimeMs) {

**if** (mTrackRunningHistoryElapsedRealtime != 0) {

//记录实际消逝的时间差，new-old

**final long** diffElapsed = elapsedRealtimeMs – mTrackRunningHistoryElapsedRealtime;

//记录处于唤醒状态时的时间差，new-old

**final long** diffUptime = uptimeMs – mTrackRunningHistoryUptime;

**if** (diffUptime < (diffElapsed-20)) {//我理解为唤醒状态时的时间间隔确实足够大了，唤醒消逝的总时间=实际消逝的时间-二者时间差中休眠占用的时间

**final long** wakeElapsedTime = elapsedRealtimeMs - (diffElapsed – diffUptime);

mHistoryAddTmp.setTo(mHistoryLastWritten);

mHistoryAddTmp.wakelockTag = **null**;

mHistoryAddTmp.wakeReasonTag = **null**;

mHistoryAddTmp.eventCode = HistoryItem.EVENT\_NONE;

mHistoryAddTmp.states &= ~HistoryItem.STATE\_CPU\_RUNNING\_FLAG;

addHistoryRecordInnerLocked(wakeElapsedTime, uptimeMs, mHistoryAddTmp);

}

}

mHistoryCur.states |= HistoryItem.STATE\_CPU\_RUNNING\_FLAG;

mTrackRunningHistoryElapsedRealtime = elapsedRealtimeMs;//重新记录最近实际消逝的时间

mTrackRunningHistoryUptime = uptimeMs;//重新记录处于唤醒状态时的时间

addHistoryRecordInnerLocked(elapsedRealtimeMs, uptimeMs, mHistoryCur);

}

如果改成6档位，代码支持，但是Battery Historian不支持，我们重新划分亮度等级和功耗值

|  |  |  |
| --- | --- | --- |
| 功耗 | 亮度 | 名称 |
| 75~150mA | 0-130 | dark |
| 150~200mA | 131-170 | dim |
| 200~240mA | 171-200 | medium |
| 240~300mA | 201-230 | light |
| 300~360mA | 231-255 | bright |