全志科技
Allwinner Technology

# A33

## 内存配置说明

# 文档履历

| 版本号 | 日期 | 制/修订人 | 制/修订记录 |
|---|---|---|---|
| V1.0 | 2014-07-30 | | 初始版本 |
| V2.0 | 2014-09-02 | | 增加 CMA 配置说明, 修改 ION 预留大小说明. |
| V3.0 | 2014-12-12 | | AndroidL 初始版本.<br>1. 删除 drop_cache, lowmemorykiller 说明<br>2. 修改 cma 预留方式<br>3. 增加配置文件 config_mem.ini 说明<br>4. 增加 hwui 配置说明 |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# 目　　录

# 1. 概述

## 1.1. 编写目的

介绍平板方案内存配置说明相关知识, 供方案定制和开发人员参考。

## 1.2. 适用范围

适用于 A33 平台；

## 1.3. 相关人员

本文档的参考人员为 A33 方案定制或开发人员.。

# 2. 术语、缩略语及概念

## 2.1. 预留内存

linux 标准函数不能分配超过 4M 的连续物理内存, 而硬件模块有时需要大于 4M 的连续物理内存. 预留内存就是为了解决这个问题.

## 2.2. vmalloc 区

指 linux 内核虚拟地址空间中, 0xFF000000 之前的一段区间, 大小不能超过 976M;

这段区间用于物理内存的动态映射, io 虚拟地址, vamlloc 函数等. 与它相对应的是低端内存区, 即线性映射区;

## 2.3. ION

android 引入的内存管理框架, 在 kernel 实现, 主要用于应用层访问连续物理内存.

## 2.4. zram

即压缩内存机制. 在系统内存紧张时, 将不活动内存进行压缩, 并回写到一块压缩内存区域, 以提高内存利用率.

## 2.5. CMA

连续内存分配器, Continuous Memory Allocator, 从 linux-3.5 引入.

CMA 实现了预留内存的充分利用. 通过 CMA, 预留内存的空闲部分可以被其他模块利用, 通过 alloc_page 申请, 从而避免了浪费.

# 3. kernel 相关配置

## 3.1. 通用内核配置

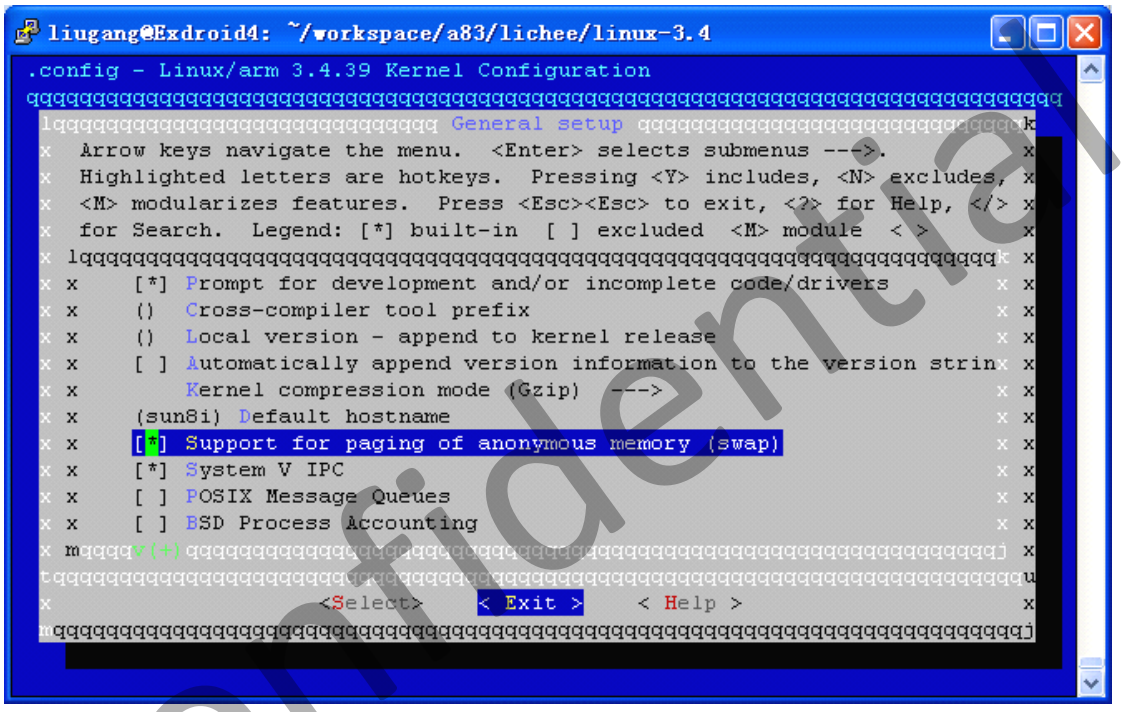
### 3.1.1. zram 配置

(1) CONFIG_SWAP:

1 在 linux3.4 目录下，输入 make ARCH=arm menuconfig

2 按照以下选项依次选择：

General setup ---> Support for paging of anonymous memory (swap)




(2) CONFIG_ZRAM

ZRAM 设置方法：

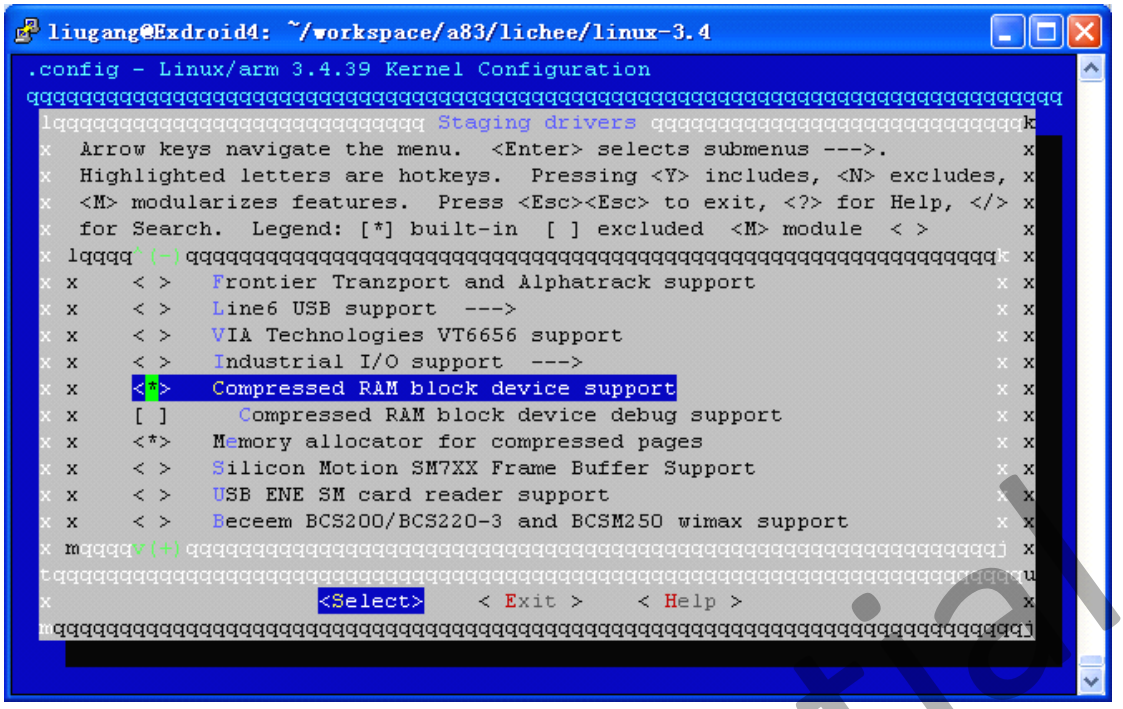
1 在 linux3.4 目录下，输入 make ARCH=arm menuconfig

2 按照以下选项依次选择：

Device Drivers ---> Staging drivers

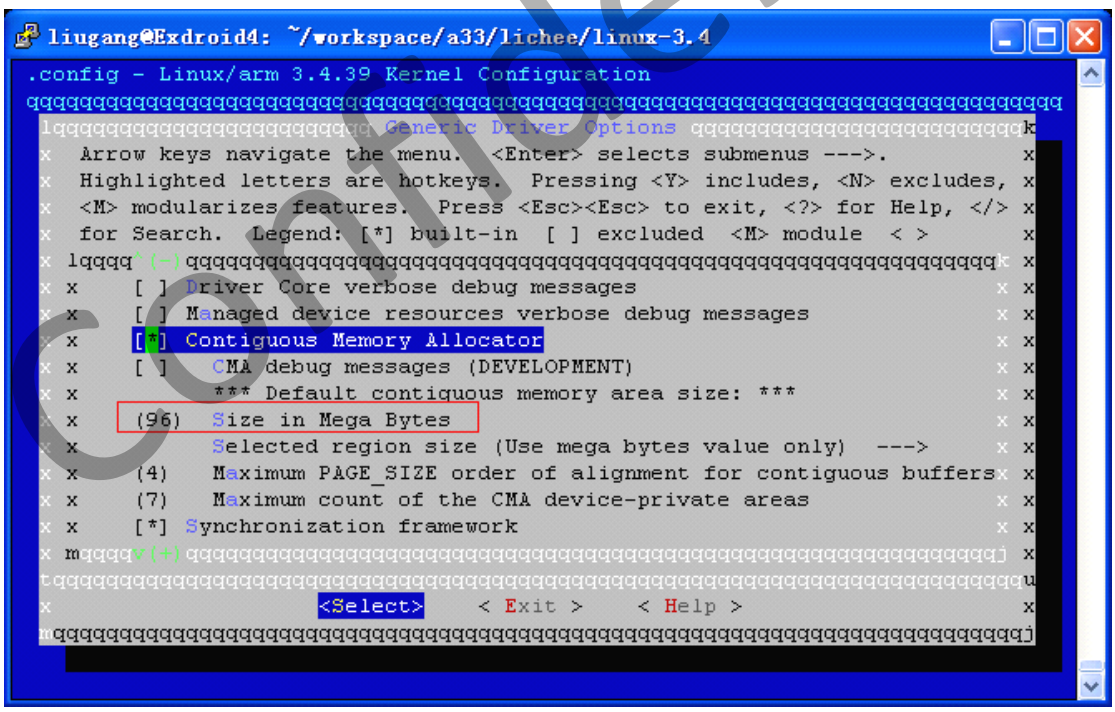
---> Compressed RAM block device support

---> Memory allocator for compressed pages




### 3.1.2. CMA 配置

(1) Device Drivers ---> Generic Driver Options ---> Contiguous Memory Allocator

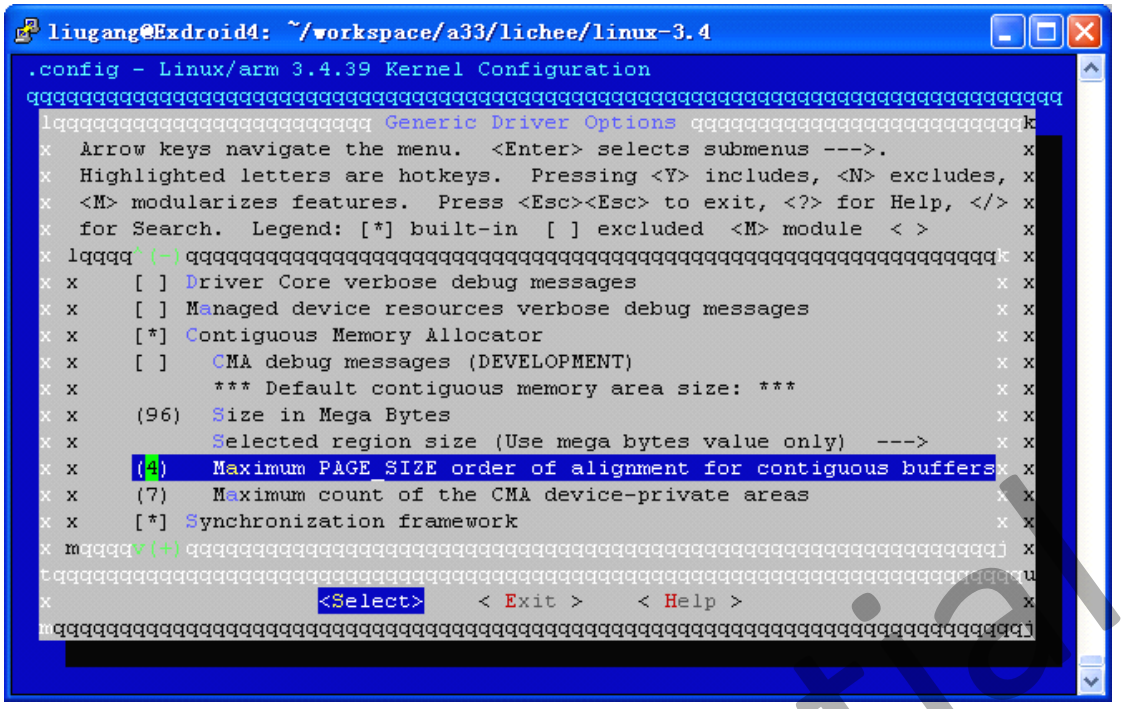



注: 上述"Size in Mega Bytes"的配置不起作用, 保持默认即可.

(2) "Maximum PAGE_SIZE order of alignment for contiguous buffers"

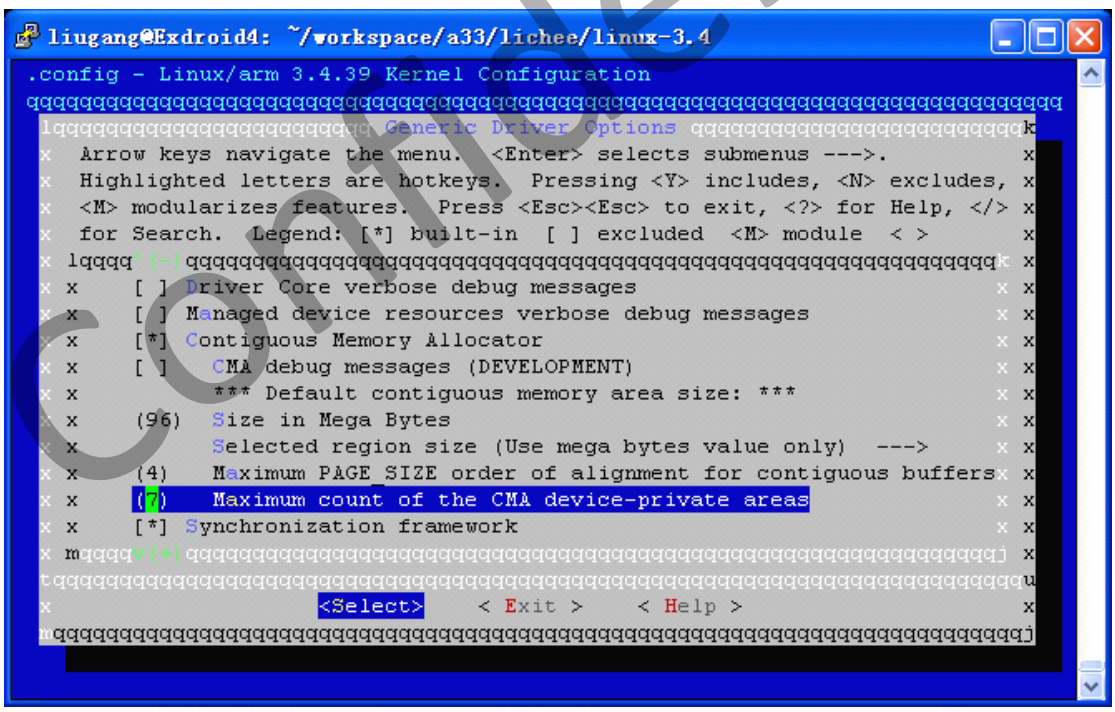
描述 CMA 分配内存时, 起始地址的对齐大小. 这里为 4, 表示每次分配时, 起始地址按 2^4 个 PAGE_SIZE (即 64K) 对齐.

保持默认即可, 无须修改.




(3) “Maximum count of the CMA device-private areas”

描述最多支持的设备私有 CMA 空间的个数. 目前我们用的是系统 CMA 区间, 没有用到设备私有区间, 因此该配置项无用.




## 3.2. ION 预留内存大小的设置

ION 预留多少合适? 需根据实际需要来定, 不同方案不一样, 估值依据如下:

1. 几个主要模块的内存消耗: GPU, VE, CAMERA, DISPLAY;
2. 规格场景的内存消耗: mirecast, 3D 游戏, CTS/GMS;

预留大小如何确定？一般分两步:

1. 根据规格，估算主要场景下，GPU/VE/CAMERA 等各模块消耗的内存，计算出总和 total;
2. 采用试凑法，将预留大小设为 total，测试主要场景下，是否有 ION 申请失败的打印.
   若有申请失败情况，则逐渐加大预留量，比如每次增加 16M，直到所有场景压力测试通过为止.
3. 内核配置了 CONFIG_CMA 时，预留内存会（比不使用 CMA）适当加大，以降低 ION 分配失败的概率.

ION 预留内存大小在 lichee/tools/pack/chips/sun8iw5p1/configs/default/env.cfg 中设置，比如:

```
ion_cma_list="120m,176m",ion_carveout_list="96m,150m"
```

(1) 若选择 CONFIG_CMA，则 ion_cma_list 才有效；否则 ion_carveout_list 才有效.

(2) 不同内存方案的 ION 预留内存大小，由几组逗号隔开的数值表示. 最多三组，分别对应 512M, 1G, 2G 方案.

比如 ion_cma_list="120m,176m"：表示内核选择 CONFIG_CMA 的前提下，512M, 1G 方案的预留内存大小分别为 120m, 176m；若有 2G 方案预留内存大小为 200m，则可以这样写: ion_cma_list="120m,176m,200m"

### 3.2.1. 512M 方案

以下面的配置为例:

```
ion_cma_list="120m,176m",ion_carveout_list="96m,150m"
```

当内核未配置 CONFIG_CMA 时, ION 预留大小为 96 MBytes.
当内核配置了 CONFIG_CMA 时, ION 预留大小为 120MBytes.

### 3.2.2. 1G 方案

以下面的配置为例:

```
ion_cma_list="120m,176m",ion_carveout_list="96m,150m"
```

当内核未配置 CONFIG_CMA 时, ION 预留大小为 150MBytes.
当内核配置了 CONFIG_CMA 时, ION 预留大小为 176MBytes.

## 3.3. vmalloc 区大小

### 3.3.1. 512M 方案

512M 方案下, vmalloc 区默认大小为 496M，从 0xD0000000 到 0xFF000000;

512M 方案使用默认配置即可，无须改动.

### 3.3.2. 1G 方案

1G 及以上方案中, vmalloc 区默认大小为 248M, 从 0xEF800000 到 0xFF000000;

在某些预留内存消耗大的场景下, 比如 miracast/3D 游戏/GMS/CTS, 默认 248M 可能不能满足需求, 因此建议将 vmalloc 区增大到 384M.

方法是在命令行增加"vmalloc=384m", 至少有以下两种方式, 推荐用方式一:

1. 在方案 env.cfg 文件中, 增加"vmalloc=384m"信息:

lichee\tools\pack\chips\sun8iw5p1\configs\default\env.cfg:

```
setargs_nand=setenv bootargs console=${console} root=${nand_root} vmalloc=384M
init=${init} loglevel=${loglevel} partitions=${partitions}
setargs_mmc=setenv bootargs console=${console} root=${mmc_root} vmalloc=384M
init=${init} loglevel=${loglevel} partitions=${partitions}
```

上述 sun8iw5p1 对应 A33.

2. 在方案中增加"vmalloc=384m":

android\device\softwinner\astar-y3\BoardConfig.mk:

```
BOARD_KERNEL_CMDLINE += vmalloc=384M
```

将上述 astar-y3 替换为实际方案目录.

# 4. android 相关配置

## 4.1. 内存配置文件

android 内存配置在文件: android\device\softwinner\astar-h7\configs\config_mem.ini 中.

这个文件格式类似于 windows 平台的 ini 文件. 包含 mainkey (主键), subkey (子键)
“#”开头的行, 为注释行, 不起作用;

```
#
# dalvik configurations
#
[dalvik_512m]
dalvik.vm.heapsize=128m
dalvik.vm.heapstartsize=5m
dalvik.vm.heapgrowthlimit=48m
dalvik.vm.heaptargetutilization=0.75
dalvik.vm.heapminfree=512k
dalvik.vm.heapmaxfree=2m
```

## 4.2. zram disksize

### 4.2.1. 512M 方案

目前默认大小为 320M.

文件: android\device\softwinner\astar-h7\fstab.sun8i
/dev/block/zram0 none swap defaults zramsize=335544320

### 4.2.2. 1G 方案

目前默认大小为 320M.

文件: android\device\softwinner\astar-h7\fstab.sun8i
/dev/block/zram0 none swap defaults zramsize=335544320

## 4.3. dalvik heap 参数

dalvik heap 参数会限制进程分配的内存大小, 在总内存很多时, 值越大则应用响应越快; 但在总内存较少时, 必须限制该值, 以免应用占用过多内存, 导致内核运行紧张.

### 4.3.1. 512M 方案

文件: android\device\softwinner\astar-h7\configs\config_mem.ini

```
[dalvik_512m]
dalvik.vm.heapsize=128m
dalvik.vm.heapstartsize=5m
dalvik.vm.heapgrowthlimit=48m
```

```
dalvik.vm.heaptargetutilization=0.75
dalvik.vm.heapminfree=512k
dalvik.vm.heapmaxfree=2m
```

### 4.3.2. 1G 方案

文件: android\device\softwinner\astar-h7\configs\config_mem.ini

```
[dalvik_1024m]
dalvik.vm.heapsize=128m
dalvik.vm.heapstartsize=5m
dalvik.vm.heapgrowthlimit=48m
dalvik.vm.heaptargetutilization=0.75
dalvik.vm.heapminfree=512k
dalvik.vm.heapmaxfree=2m
```

### 4.3.3. 2G 方案

文件: android\device\softwinner\astar-h7\configs\config_mem.ini

```
[dalvik_2048m]
# dalvik.vm.heapsize=384m   (#开头表示注释行，不起作用. 即使用系统默认配置)
# dalvik.vm.heapstartsize=8m
# dalvik.vm.heapgrowthlimit=64m
# dalvik.vm.heaptargetutilization=0.75
# dalvik.vm.heapminfree=512k
# dalvik.vm.heapmaxfree=8m
```

## 4.4. hwui 参数

[hwui_800], [hwui_1024], [hwui_1280], [hwui_1920], [hwui_2048], [hwui_2560]. 数字表示 lcd 分辨率的长边像素. 比如 1280*800, 1280*720 的屏, 其对应 hwui 参数是[hwui_1280].

```
[hwui_800]

[hwui_1024]

[hwui_1280]  (默认设置是针对 1280*800，所以大部分采用默认设置)
#ro.hwui.texture_cache_size=24  (texture cache，MB，至少 5 倍于 width * height * 32bit)
#ro.hwui.layer_cache_size=16  (layer cache，MB，至少 4 倍于 width * height * 32bit)
#ro.hwui.r_buffer_cache_size=2  (renderbuffer cache，MB，至少 2 倍于 width * height * 8bit)
#ro.hwui.path_cache_size=10  (path cache，MB，至少 1 倍于 width * height * 32bit)
#ro.hwui.drop_shadow_cache_size=2  (text drop shadow cache，MB，至少 2 倍于 width * height * 8bit)
#ro.hwui.text_small_cache_width=1024  (pixels)
#ro.hwui.text_small_cache_height=512  (pixels)
#ro.hwui.text_large_cache_width=2048  (pixels)
#ro.hwui.text_large_cache_height=512  (pixels)
#ro.hwui.gradient_cache_size=0.5  (gradient cache，MB)
#ro.hwui.vertex_cache_size=1  (tessellation cache，MB)
```

```
#ro.hwui.patch_cache_size=128    (KB)
#ro.hwui.texture_cache_flushrate=0.6    (flush 后保留的 texture cache 的比例）
#ro.hwui.disable_scissor_opt=false
[hwui_1920]

[hwui_2048]

[hwui_2560]
```

更详细的参数介绍可参考：https://source.android.com/devices/tech/debug/tuning.html