Rockchip Linux CMA开发文档

文件标识：RK-KF-YF-117

发布版本：V1.1.0

日期：2022-06-02

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。

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

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

前言

概述

本文提供一个标准模板供套用。后续模板以此份文档为基础改动。

产品版本

芯片名称	内核版本
RK3588, RK3568, RK3399, RK3288, RK3368, RV1126,PX30	Linux-4.4, 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

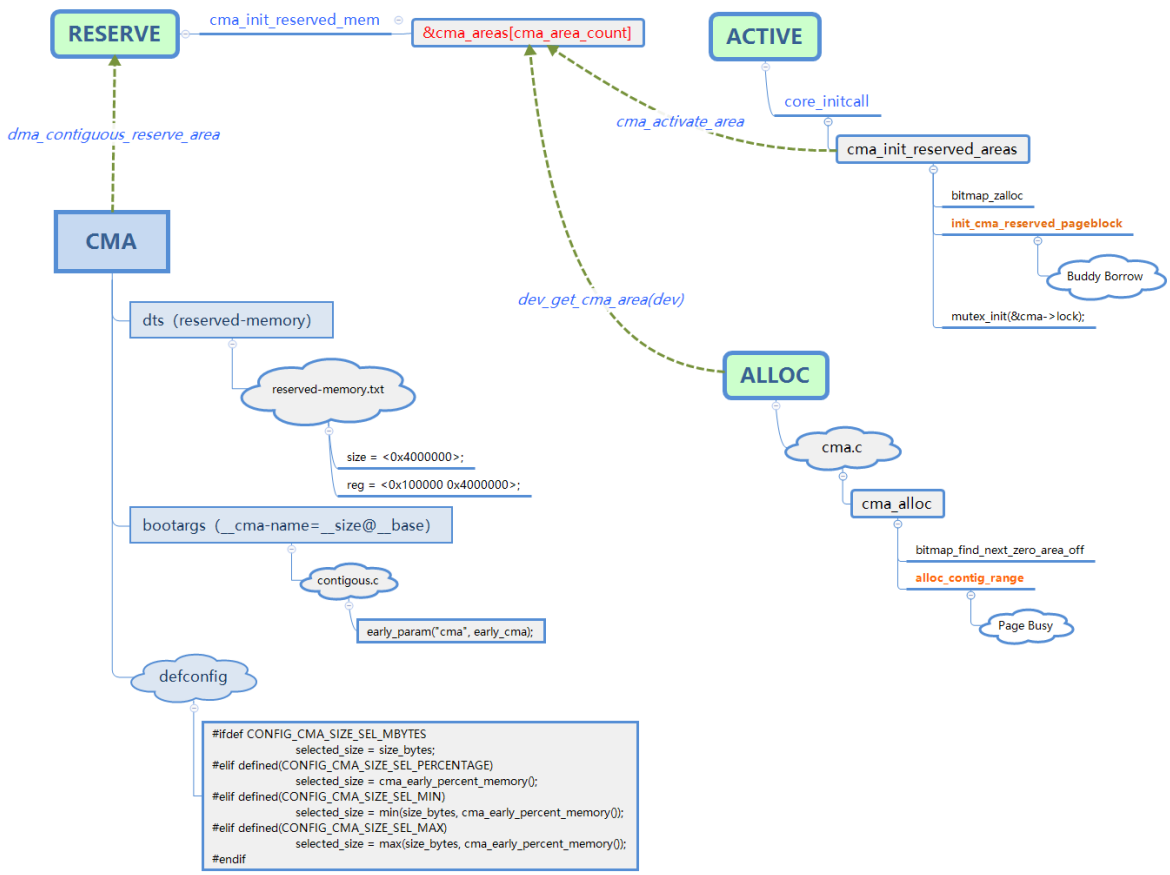
软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	许剑群	2020-10-18	初始版本
V1.1.0	许剑群	2022-06-02	增加CMA框架图

目录

1. CMA驱动框架



CMA主要有3部分驱动:

1. size@address

   有三种方法可以传参，优先顺序是DTS > CMDLINE > DEFCONFIG，即如果dts有合法的节点，会以dts节点配置为最终参数，否则看cmdline是否有cma=size@base，最后才看defconfig是否有如上4个宏定义。

2. reserved cma_areas[]

   通过dts可以配置多个cma节点，通过cmdline和defconfig只能定义default节点。所有reserved的内存会在全局的cma_areas[]存储。

3. active cma_area

   解析参数、预留内存都在内核setup_arch阶段完成，active是core_initcall调用，保证在所有driver之前完成；active主要完成cma area的bitmap创建、cma pages加入buddy system。

2. 内存预留的分配方法

2.1 分配方法一: 通过menuconfig预留16MByte

CONFIG_DMA_CMA=y
CONFIG_CMA_SIZE_MBYTES=16
CONFIG_CMA_SIZE_SEL_MBYTES=y

2.2 分配方法二: 通过dts的reserved-memory节点

参考`Documentation/devicetree/bindings/reserved-memory/reserved-memory.txt`技术文档

```
reserved-memory {
    #address-cells = <1>;
    #size-cells = <1>;
    ranges;

    /* global autoconfigured region for contiguous allocations */
    linux,cma {
        compatible = "shared-dma-pool";
        reusable;
        size = <0x1000000>;
        alignment = <0x2000>;
        linux,cma-default;
    };
};
```

3. 内存独占的分配方法

上述方法分配到的是系统共享内存块，在多设备多次进行申请释放后，会有碎片化问题。有些外设需要较大的CMA内存块，碎片化会引起外设工作异常，为此需要给设备单独分配内存块

3.1 分配方法一: 分配的内存块独立存在, 会被系统借用

```
reserved-memory {
    #address-cells = <1>;
    #size-cells = <1>;
    ranges;

    cma_region: region@88000000 {
        compatible = "shared-dma-pool";
        reusable;
        reg = <0x88000000 0x1800000>;
    };
};
```

该方法分配的内存块，可以通过`of_reserved_mem_device_init`给指定设备用

```
#include <linux/of_reserved_mem.h>
static inline int of_reserved_mem_device_init(struct device *dev)
{
    return of_reserved_mem_device_init_by_idx(dev, dev->of_node, 0);
}
```

3.2 分配方法二: 分配的内存块memblock

```
reserved-memory {
    #address-cells = <1>;
    #size-cells = <1>;
    ranges;

    display_reserved: framebuffer@78000000 {
        reg = <0x78000000 0x800000>;
    };
};
```

可以通过`/sys/kernel/debug/memblock/reserved`节点查看分配结果

节点展示的memblock范围是经过合并整理的

该方法分配的内存块是没有内核页管理，使用者根据自身需要，去创建page scale list

```
console:/sys/kernel/debug/cma # cat /sys/kernel/debug/memblock/reserved
    0: 0x60004000..0x60007fff
    1: 0x60100000..0x61404a07
    2: 0x62e00000..0x62eeffff
    3: 0x68300000..0x6830d07f
...
```

4. 内存预留调试方法

4.1 CMA调试宏开关

```
CONFIG_CMA_DEBUGFS=y #打开调试文件节点
CONFIG_CMA_DEBUG=y #打印CMA日志信息
```

4.2 CMA调试节点

```
console:/sys/kernel/debug/cma # ls
cma-region@8800 cma-reserved
console:/sys/kernel/debug/cma/cma-reserved # ls
alloc    bitmap free     order_per_bit      used
base_pfn count  maxchunk
```

- alloc：用于从当前CMA块分配内存，单位：页
- bitmap：标记当前CMA块各个页是否被分配，1表示已被申请，0表示未被申请
- free：用于当前CMA块释放内存，单位：页
- order_per_bit：表示一个BIT所代表的页的阶数
- used：当前被申请的页的个数，单位：页
- base_pfn：当前CMA块首个页的页帧数
- maxchunk：当前CMA块最大连续空闲页的数量，单位：页

5. 开机过程内存预留信息

```
[    0.000000] Booting Linux on physical CPU 0xf00
[    0.000000] CPU: ARMv7 Processor [410fc075] revision 5 (ARMv7), cr=10c5387d
[    0.000000] CPU: div instructions available: patching division code
[    0.000000] CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction
cache
[    0.000000] OF: fdt: Machine model: Rockchip RK3126 bnd-d708 board
[    0.000000] earlycon: uart8250 at MMIO32 0x20068000 (options '')
[    0.000000] bootconsole [uart8250] enabled
[    0.000000] Memory policy: Data cache writealloc
[    0.000000] Reserved memory: created CMA memory pool at 0x88000000, size 24
MiB
[    0.000000] OF: reserved mem: initialized node region@88000000, compatible id
shared-dma-pool
[    0.000000] cma: Reserved 16 MiB at 0x9f000000
[    0.000000] On node 0 totalpages: 258560
[    0.000000] Normal zone: 1728 pages used for memmap
[    0.000000] Normal zone: 0 pages reserved
[    0.000000] Normal zone: 193024 pages, LIFO batch:63
[    0.000000] HighMem zone: 65536 pages, LIFO batch:15
[    0.000000] psci: probing for conduit method from DT.
[    0.000000] psci: PSCIv65535.65535 detected in firmware.
[    0.000000] psci: Using standard PSCI v0.2 function IDs
[    0.000000] psci: MIGRATE_INFO_TYPE not supported.
[    0.000000] psci: SMC Calling Convention v1.0
[    0.000000] percpu: Embedded 17 pages/cpu s38924 r8192 d22516 u69632
[    0.000000] pcpu-alloc: s38924 r8192 d22516 u69632 alloc=17*4096
[    0.000000] pcpu-alloc: [0] 0 [0] 1 [0] 2 [0] 3
[    0.000000] Built 1 zonelists, mobility grouping on.  Total pages: 256832
[    0.000000] Dentry cache hash table entries: 131072 (order: 7, 524288 bytes)
[    0.000000] Inode-cache hash table entries: 65536 (order: 6, 262144 bytes)
[    0.000000] mem auto-init: stack:off, heap alloc:off, heap free:off
[    0.000000] Memory: 960940K/1034240K available (11264K kernel code, 985K
rwdata, 3180K rodata, 1024K init, 2104K bss, 32340K reserved, 40960K cma-
reserved, 245024K highmem)
```

```
...
```

第一块CMA分配成功信息

```
[    0.000000] Reserved memory: created CMA memory pool at 0x88000000, size 24 MiB
[    0.000000] OF: reserved mem: initialized node region@88000000, compatible id shared-dma-pool
```

第二块CMA分配成功信息（系统默认）

```
[    0.000000] cma: Reserved 16 MiB at 0x9f000000
```