

D1-H Linux RTC 开发指南

版本号: 1.0

发布日期: 2021.04.26





版本历史

Tustusi

版本号	日期	制/修订人	内容描述	
1.0	2021.04.08	XAA0190	添加初版	1081
1.0	2021.04.15	XAA0190	修改格式	77.
1.0	2021.04.26	XAA0190	修改图片	

Transfer tra

NSU OST

版权所有 © 珠海全志科技股份有限公司。保留一切权利

Tugudes

15 P





录 目

1	概述	1
	1.1%编写目的 %	1
	16.2 适用范围。	₩1
	1.3 相关人员	1
2	模块介绍	2
	2.1 模块功能介绍	2
	2.2 相关术语介绍	2
	2.3 源码结构介绍	3
3	模块配置介绍	4
	3.1 kernel menuconfig 配置	4
	3.2 device tree 源码结构和路径	6
	3.2.1 源码结构图	7
	3.3 device tree 对 RTC 控制器的通用配置	7
	3.4 board.dts 板级配置	7
4	接口描述	Tranc
	4.1 打开/关闭 RTC 设备	8
	4.2 设置和获取 RTC 时间	8
5	模块使用范例	9
6	FAQ	11
	6.1 RTC 时间不准	11
	6.2 RTC 时间不走	11





1 根述 transt transt transt transt

1.1 编写目的

介绍 Linux 内核中 RTC 驱动的适配和 DEBUG 方法,为 RTC 设备的使用者和维护者提供参考。

1.2 适用范围

Tustosi	trangsi	th ^{angs¹} 表1-	1: 适用产品列表	itense itense
	产品名称	内核版本	驱动文件	
	D1-H	Linux-5.4	rtc-sunxi.c	

1.3 相关人员

RTC 驱动及应用层的开发/维护人员。

317951

tranger tranger tranger tranger tranger tranger tranger

Magar Mallagar

版权所有 © 珠海全志科技股份有限公司。保留一切权

H31051

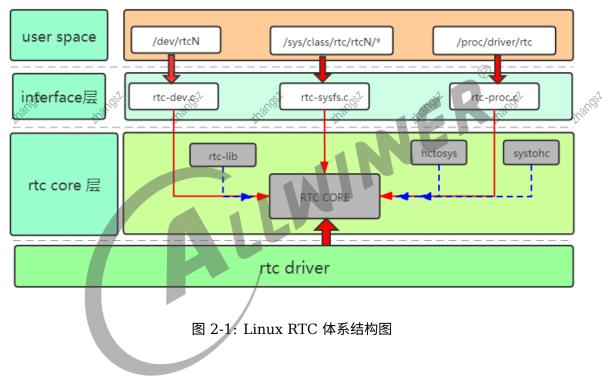
Narast

Nangs1



2.1 模块功能介绍

Linux 内核中,RTC 驱动的结构图如下所示,可以分为三个层次:



- 接口层,负责向用户空间提供操作的结点以及相关接口。
- RTC Core, 为 rtc 驱动提供了一套 API, 完成设备和驱动的注册等。
- RTC 驱动层,负责具体的 RTC 驱动实现,如设置时间、闹钟等设置寄存器的操作。

2.2 相关术语介绍

表 2-1: RTC 模块相关术语介绍

术语	解释说明
Sunxi	指 Allwinner 的一系列 SoC 硬件平台
RTC	Real Time Clock,实时时钟





2.3 源码结构介绍

linux-5.4 L-- drivers L-- rtc |-- class.c |-- hctosys.c |-- interface.c |-- dev.c |-- lib.c |-- proc.c |-- sysfs.c |-- systohc.c |-- rtc-core.h |-- rtc-sunxi.c L-- rtc-sunxi.h

Thangs!



3 模块配置介绍

3.1 kernel menuconfig 配置

在命令行中进入根目录,执行source build/envsetup.sh配置环境,执行lunch,按照提示配置平台、板型等信息(如果之前已经配置过,可跳过此步骤)。

然后在 linux-5.4 根目录执行make kernel_menuconfig,进入内核图形化配置界面,并按以下步骤操作:

```
选择Device Driver选项进入下一级配置,如下图所示:
               Hidden CRYPTO configs needed for GKI
                Hidden SND configs needed for GKI
               ] Hidden SND_SOC configs needed for GKI
               ] Hidden MMC configs needed for GKI
               ] Hidden GPIO configs needed for GKI
] Hidden QCOM configs needed for GKI
               ] Hidden Media configs needed for GKI
               ] Hidden Virtual configs needed for GKI
               ] Hidden wireless extension configs needed for GKI
                 Hiddel USB configurations needed for GKI
                ] Hidden SoC bus configuration needed for GKI
               ] Hidden RPMSG configuration needed for GKI
               ] Hidden GPU configuration needed for GKI
               ] Hidden IRQ configuration needed for GKI
               ] Hidden hypervisor configuration needed for GKI
                 GKI Dummy config options
                 Optional GKI features
                  xecutable file formats
                 Memory Management options
                 Networking support
                  File systems
                  Security options
                  Cryptographic API
                 Library routines --->
                 Kernel hacking
                                 < Exit >
                                             < Help >
                                                                      < Load >
                                                          < Save >
```

图 3-1: 内核根菜单

选择Real Time Clock进入下一级配置,如下图所示:

angst

版权所有 © 珠海全志科技股份有限公司。保留一切权利

TUSINOST

\$

- A145





```
[ ] Remote Controller support
<*> Multimedia support --->
     Graphics support --->
<*> Sound card support --->
HID support --->
[*] SB support -->
<*> MMC/SD/SDIO card support ->->
Sony MemoryStick card support ----
[ ] LED Support
[ ] Accessibility support ----
     InfiniBand support --
[*] DMA Engine support --->
     DMABUF options
[ ] Auxiliary Display support ----
<> Userspace I/O drivers ----
[ ] VFIO Non-Privileged userspace driver framework ----
    Virtualization drivers
    Virtio drivers --->
     Microsoft Hyper-V guest support ----
    Greybus support --->
[*] Staging drivers --->
[] Platform support for Goldfish virtual devices ----
 > Platform support for Goldrish Virtual devices ----
> Platform support for Chrome hardware (transitional)
] Platform support for Chrome hardware ----
] Platform support for Mellanox hardware ----
                        < Exit >
                                        < Help >
                                                       < Save >
                                                                       < Load >
```

图 3-2: 内核 rtc menuconfig 根菜单 如下图所示。

选择Allwinner sunxi RTC配置,如下图所示。

isues itsues

NSU OS.

版权所有 © 珠海全志科技股份有限公司。保留一切权利

Wallder

551 S

, sel



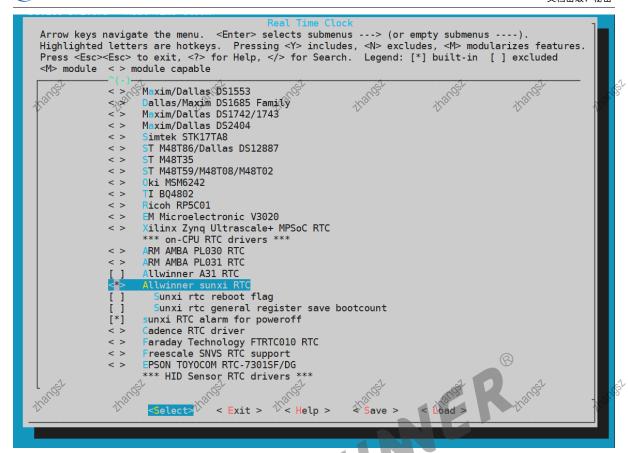


图 3-3: rtc menuconfig 菜单

由于在关机过程中,RTC 一般都是独立供电的,因此在 RTC 电源域中的寄存器不会掉电且 RTC 寄存器的值也不会恢复为默认值。利用此特性,Sunxi 平台支持 reboot 命令的一些扩展功能,但需要打开Sunxi rtc reboot flag和Sunxi rtc general register save bootcount选项,RTC 驱动才能支持这些扩展功能。

3.2 device tree 源码结构和路径

SoC 级设备树文件(sun*.dtsi)是针对该 SoC 所有方案的通用配置:

• SoC 级设备树的路径为: arch/riscv/boot/dts/sunxi/sun20iw1p1.dtsi

板级设备树文件(board.dts)是针对该板型的专用配置:

● 板级设备树路径: device/config/chips/dl-h/configs/nezha/board.dts

2051

版权所有 © 珠海全志科技股份有限公司。保留一切权利

1001693"



3.2.1 源码结构图

device tree 的源码结构关系如下:

```
board.dts
L-----sun20iwlp1.dtsi
```

3.3 device tree 对 RTC 控制器的通用配置

```
/ {
     rtc: rtc@7090000 {
       compatible = "allwinner,sun20iw1-rtc"; //用于probe驱动
3
       device_type = "rtc";
4
5
                                 //表示RTC是具备休眠唤醒能力的中断唤醒源
       wakeup-source;
       reg = <0x0 0x07090000 0x0 0x320>; //RTC寄存器基地址和映射范围
6
       interrupts-extended = <&plic0 160 IRQ_TYPE_LEVEL_HIGH>; //RTC硬件中断号_ 〇
       clocks = <&r ccu CLK R AHB BUS RTC>, <&rtc ccu CLK RTC SPI>, <&rtc ccu CLK RTC 1K>;
       RTC所用到的时钟
       clock-names = "r-ahb-rte", "rtc-spi", "rtc-1k";
10
       resets = <&r_ccu RST_R_AHB_BUS_RTC>;
11
       gpr_cur_pos = <6>;
                                      //当前被用作reboot
12
     };
13
```

在 Device Tree 中对每一个 RTC 控制器进行配置,一个 RTC 控制器对应一个 RTC 节点,节点属性的含义见注释。

3.4 board.dts 板级配置

board.dts用于保存每个板级平台的设备信息 (如 demo 板、demo2.0 板等等)。board.dts路径如下:

```
device/config/chips/dl-h/configs/nezha/board.dts
```

在board.dts中的配置信息如果在sun20iwlp1.dtsi中存在,则会存在以下覆盖规则:

- 1. 相同属性和结点,board.dts的配置信息会覆盖sun20iw1p1.dtsi中的配置信息
- 2. 新增加的属性和结点,会添加到编译生成的 dtb 文件中

inangs)

12051

NSUQ2,





4 接口描述

RTC 驱动会注册生成串口设备/dev/rtc0,应用层的使用只需遵循 Linux 系统中的标准 RTC 编程方法即可。

4.1 打开/关闭 RTC 设备

使用标准的文件打开函数:

```
int open(const char *pathname, int flags);
int close(int fd);
需要引用头文件:
```

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
```

4.2 设置和获取 RTC 时间

同样使用标准的 ioctl 函数:

```
int ioctl(int d, int request, ...);
```

需要引用头文件:

```
#include <sys/ioctl.h>
#include <linux/rtc.h>
```

Nang3"

版权所有 © 珠海全志科技股份有限公司。保留一切权利

BUOST

51



央使用范例

此 demo 程序是打开一个 RTC 设备,然后设置和获取 RTC 时间以及设置闹钟功能。

```
#include <stdio.h>
                                                                    /*标准输入输出定义*/
         #include <stdlib.h>
                                                                    /*标准函数库定义*/
  3
         #include <unistd.h>
                                                                     /*Unix 标准函数定义*/
         #include <sys/types.h>
  5
         #include <sys/stat.h>
         #include <fcntl.h>
                                                                    /*文件控制定义*/
          #include <linux/rtc.h>
                                                                   /*RTC支持的CMD*/
  8
         #include <errno.h>
                                                                    /*错误号定义*/
 9
         #include <string.h>
10
11
         #define RTC_DEVICE_NAME
                                                                         "/dev/rtc0"
                                                                                                                                                               I that the state of the state o
12
13
         int set_rtc_timer(int fd)
14
15
               struct rtc_time rtc_tm = {0};
16
              struct rtc_time rtc_tm_temp = {0};
17
               rtc tm.tm year = 2020 - 1900; /* 需要设置的年份,需要减1900
18
19
              rtc tm.tm mon = 11 - 1;
                                                                                             /* 需要设置的月份,需要确保在0-11范围 */
                                                                                     需要设置的日期 *
20
              rtc tm.tm mday = 21;
                                                                              /* 需要设置的时间
21
              rtc tm.tm hour = 10;
                                                                              /* 需要设置的分钟时间 */
22
              rtc_tm.tm_min = 12;
              rtc_tm.tm_sec = 30;
23
                                                                                      需要设置的秒数 */
24
25
               /* 设置RTC时间 */
              if (ioctl(fd, RTC_SET_TIME, &rtc_tm) < 0) {</pre>
26
27
                    printf("RTC_SET_TIME failed\n");
28
                    return -1;
29
30
31
               /* 获取RTC时间 */
              if (ioctl(fd, RTC_RD_TIME, &rtc_tm_temp) < 0) {</pre>
32
              printf("RTC_RD_TIME failed(n");
return -1;
33
34
35
36
               printf("RTC RD TIME return %04d-%02d-%02d %02d:%02d:%02d\n",
37
                                   rtc_tm_temp.tm_year + 1900, rtc_tm_temp.tm_mon + 1, rtc_tm_temp.tm_mday,
38
                                   rtc_tm_temp.tm_hour, rtc_tm_temp.tm_min, rtc_tm_temp.tm_sec);
39
               return 0;
40
         }
41
42
         int set_rtc_alarm(int fd)
43
44
              struct rtc_time rtc_tm = {0};
45
              struct rtc_time rtc_tm_temp = {0};
46
47
                                                                    /* 闹钟忽略年设置 */
               rtc_tm.tm_year = 0;
                                                                    /* 闹钟忽略月设置 */
48
               rtc_tm.tm_mon = 0;
49
               rtc_tm.tm_mday = 0;
                                                                    /* 闹钟忽略日期设置 */
```



```
rtc tm.tm hour = 10; /* 需要设置的时间 */
50
       rtc_tm.tm_min = 12;
                            /* 需要设置的分钟时间 */
51
                            /* 需要设置的秒数 */
52
       rtc_tm.tm_sec = 30;
53
54
       /* set alarm time */
                                                                                               Wal
55
       if (ioctl(fd, RTC_ALM_SET, &rtc_tm) < 0) {
56
         printf("RTC_ALM_SET failed\n");
57
         return -1;
58
59
60
       if (ioctl(fd, RTC AIE ON) < 0) {</pre>
         printf("RTC_AIE_ON failed!\n");
61
62
         return -1;
63
64
       if (ioctl(fd, RTC_ALM_READ, &rtc_tm_temp) < 0) {</pre>
65
66
         printf("RTC_ALM_READ failed\n");
67
         return -1;
68
       }
69
70
       printf("RTC_ALM_READ return %04d-%02d-%02d %02d:%02d:%02d\n",
71
                   rtc_tm_temp.tm_year + 1900, rtc_tm_temp.tm_mon + 1, rtc_tm_temp.tm_mday,
                                                 Thangs?
72
                   rtc_tm_temp.tm_hour, rtc_tm_temp.tm_min, rtc_tm_temp.tm_sec);
73
       return 0;
74
    }
75
76
    int main(int argc, char *argv[])
77
78
         int fd;
79
         int ret;
80
         /* open rtc device */
81
         fd = open(RTC_DEVICE_NAME, 0_RDWR);
82
         if (fd < 0) {
83
84
           printf("open rtc device %s failed\n", RTC_DEVICE_NAME);
85
           return -ENODEV;
86
         }
87
88
       /* 设置RTC时间 */
89
       ret = set_rtc_timer(fd);
90
       if (ret < 0) {
         printf("set rtc timer error\n");
91
92
         return -EINVAL;
93
       }
                                                                                               That
94
       ↑* 设置闹钟 * / 🌣
95
96
       ret = set_rtc_alarm(fd);
97
       if (ret < 0) {
         printf("set rtc alarm error\n");
98
         return -EINVAL;
99
100
101
       close(fd);
102
103
       return 0;
104
```

20052

版权所有 © 珠海全志科技股份有限公司。保留一切权利

NSU OZV

52





6.1 RTC 时间不准

1. 按照下图 RTC 时钟源的路径,确认一下 RTC 所使用的时钟源

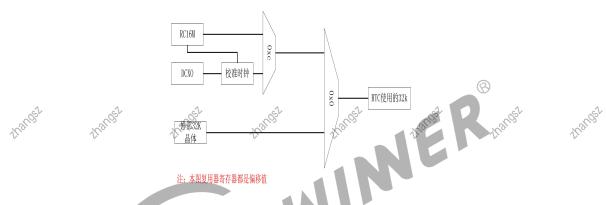


图 6-1: RTC 时钟源

- 2. 如果确认使用的时钟源为 RC16M,则确认一下有没有启用校准功能,因为 RC16M 有正负 50% 的偏差。
- 3. 如果使用外部晶体,则确认一下外部晶体的震荡频率是否正确。

6.2 RTC 时间不走

- 1. 请查看 RTC 时钟源图,确认一下使用的时钟源。
- 2. 当 RTC 时钟源为外部 32K 时,请确认一下外部 32k 晶体的起振情况。

🛄 说明

当使用示波器测量外部 32k 晶体起振情况时,有可能会导致 32k 晶体起振。

3. 当排查完时钟源,确认时钟源没有问题后,通过以下命令 dump rtc 相关寄存器,查看偏移 0x0 寄存器的状态位 bit7 和 bit8 是否异常置 1 了,如下所示:

20102V

版权所有 © 珠海全志科技股份有限公司。保留一切权利

1,13





/ # echo 0x07090000,0x07090200 > /sys/class/sunxi dump/dump; cat /sys/class/sunxi dump/dump 0x000000007000000: 0x00004010 0x00000004 0x0000000f 0x7a000000 0x000000007000010: 0x00000001 0x00000023 0x00000000 0x00000000 0×000000007000040 : 0×000000000 0×000000000 0×000000000 0×000000000 $0 \\ \times 0000000007000070: \ 0 \\ \times 00010003 \ 0 \\ \times 000000000 \ 0 \\ \times 000000000 \ 0 \\ \times 000000000$ 0x000000007000130: 0x00000000 0x000030ea 0x04001000 0x00006061 0x000000007000160: 0x083f10f7 0x00000043 0x00000000 0x00000000 $0 \times 0000000007000170$; 0×000000000 ; 0×00000000 ; 0×00000000 0x0000000070001a0: 0x000090ff 0x00000000 0x00000000 0x00000000 0×000000007000200: 0×10000000

rangs l

thates, thates, thates, thates, thates, thates, thates, thates,

21051

版权所有 © 珠海全志科技股份有限公司。保留一切权利

1



著作权声明

版权所有 © 2022 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护,其著作权由珠海全志科技股份有限公司("全志")拥有并保留一切权利。

本文档是全志的原创作品和版权财产,未经全志书面许可,任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部,且不得以任何形式传播。

商标声明



举)均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标,产品名称,和服务名称,均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司("全志")之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明,并严格遵循本文档的使用说明。您将自行承担任何不当使用行为(包括但不限于如超压,超频,超温使用)造成的不利后果,全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因,本文档内容有可能修改,如有变更,恕不另行通知。全志尽全力在本文档中提供准确的信息,但并不确保内容完全没有错误,因使用本文档而发生损害(包括但不限于间接的、偶然的、特殊的损失)或发生侵犯第三方权利事件,全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中,可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税(专利税)。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。

st therest therest therest therest therest therest therest