CV180X & CV181X RTC 使用手册

Version: 2.0.0

Release date: 2023-02-08

©2022 北京晶视智能科技有限公司
本文件所含信息归北京晶视智能科技有限公司所有。
未经授权，严禁全部或部分复制或披露该等信息。

目录

1 声明 **2**

2 RTC 操作指南 **3**
2.1 模块介绍 3
2.2 计数时钟频率 3
2.3 操作准备 3
2.4 使用方式 3
2.4.1 Ioctl 控制 RTC 3
2.4.2 ioctl 使用范例 4
2.4.3 结构体 5

3 Linux 系统中 RTC 命令 **6**
3.1 date 和 hwclock 6

修订记录

Revision	Date	Description
1.0.1	2022/06/08	初稿
1.1	2022/08/02	Change version
2.0.0	2022/02/08	cv180x/cv181x 文档兼容

1 声明

法律声明

本数据手册包含北京晶视智能科技有限公司（下称“晶视智能”）的保密信息。未经授权，禁止使用或披露本数据手册中包含的信息。如您未经授权披露全部或部分保密信息，导致晶视智能遭受任何损失或损害，您应对因之产生的损失/损害承担责任。

本文件内信息如有更改，恕不另行通知。晶视智能不对使用或依赖本文件所含信息承担任何责任。

本数据手册和本文件所含的所有信息均按“原样”提供，无任何明示、暗示、法定或其他形式的保证。晶视智能特别声明未做任何适销性、非侵权性和特定用途适用性的默示保证，亦对本数据手册所使用、包含或提供的任何第三方的软件不提供任何保证；用户同意仅向该第三方寻求与此相关的任何保证索赔。此外，晶视智能亦不对任何其根据用户规格或符合特定标准或公开讨论而制作的可交付成果承担责任。

联系我们

地址 北京市海淀区丰豪东路 9 号院中关村集成电路设计园（ICPARK）1 号楼

深圳市宝安区福海街道展城社区会展湾云岸广场 T10 栋

电话 +86-10-57590723 +86-10-57590724

邮编 100094（北京）518100（深圳）

官方网站 https://www.sophgo.com/

技术论坛 https://developer.sophgo.com/forum/index.html

2 RTC 操作指南

2.1 模块介绍

RTC(real time clock) 硬件时钟，为系统提供与记录时间。若 RTC 由电池供电，当处理器处于下电关机或休眠状态时，RTC 仍会继续计数并维护时间信息不丢失。

Linux 内核将 RTC 作为时间与日期维护器，当 Linux 系统启动时，内核读取 RTC 时间以初始化系统（软件）时钟达成时间同步。内核在需要时，亦可将时间与日期回写到 RTC 中。

2.2 计数时钟频率

RTC 的计数时钟采用 32.768KHz 时钟，运行基于一个 32-bit 加法计数器提供秒计数，计数最大时间为:

2^32 秒 = 49710 天 = 136 年

2.3 操作准备

RTC 的操作准备如下:

- 使用 SDK 发布的 kernel
- 插入模块: insmod cv180x_rtc.ko/ cv181x_rtc.ko

2.4 使用方式

2.4.1 Ioctl 控制 RTC

应用层可以通过 ioctl 控制 RTC，设备节点为/dev/rtc0

使用方式如下:

```
int ioctl(int fd, ind cmd);
```

ioctl 指令功能描述

指令	描述
RTC_ALM_READ	读取闹钟时间
RTC_ALM_SET	设置闹钟时间
RTC_RD_TIME	读取时间与日期
RTC_SET_TIME	读取时间与日期
RTC_PIE_ON	开 RTC 全局中断
RTC_PIE_OFF	关 RTC 全局中断
RTC_AIE_ON	使能 RTC 闹钟中断
RTC_AIE_OFF	禁止 RTC 闹钟中断
RTC_UIE_ON	使能 RTC 更新中断
RTC_UIE_OFF	禁止 RTC 更新中断
RTC_IRQP_SET	设置中断频率

2.4.2 ioctl 使用范例

```
static const char default_rtc[] = "/dev/rtc0";
struct rtc_time rtc_tm;
int fd;

fd = open(rtc, O_RDONLY);
if (fd == -1) {
  perror(rtc);
  exit(errno);
}
```

通过如下命令可获取 RTC 时间：

```
/* Read the RTC time/date */
retval = ioctl(fd, RTC_RD_TIME, &rtc_tm);
if (retval == -1) {
  perror("RTC_RD_TIME ioctl");
  exit(errno);
}
fprintf(stderr, "\n\nCurrent RTC date/time is %d-%d-%d, %02d:%02d:%02d.\n",
    rtc_tm.tm_mday, rtc_tm.tm_mon + 1, rtc_tm.tm_year + 1900,
    rtc_tm.tm_hour, rtc_tm.tm_min, rtc_tm.tm_sec);
```

通过如下命令可设置 RTC 时间：

```
retval = ioctl(fd, RTC_SET_TIME, &rtc_tm);
if (retval == -1) {
  perror("RTC_RD_TIME ioctl");
  exit(errno);
}
```

2.4.3 结构体

- rtc_time

```
struct rtc_time {
    int tm_sec;
    int tm_min;
    int tm_hour;
    int tm_mday;
    int tm_mon;
    int tm_year;
    int tm_wday;
    int tm_yday;
    int tm_isdst;
    };
```

tm_mday：一个月中的日期，取值区间为 [1,31]

tm_wday：一个星期的第几天，星期日为 0, 星期一为 1, 以此类推

tm_yday：一年中的第几天，取值区间为 [0,365]，其中 0 代表 1 月 1 日，1 代表 1 月 2 日，以此类推

tm_isdst：判断是否为夏令时, 1 是夏令时; 0 非夏令时

3 Linux 系统中 RTC 命令

3.1 date 和 hwclock

- date 命令可以查询/更改目前 Linux 上的系统 (软件) 时钟

例: 写入时间

```
# date "2020-10-17 9:48:30"
Sat Oct 17 09:48:30 CST 2020
```

例: 读取时间

```
# date
Sat Oct 17 09:48:34 CST 2020
```

- Hwclock 则用于查询/更改硬件时钟 (RTC) 的时间

例: 查询硬件时间 (RTC 时间)

```
# hwclock
Sat Oct 17 09:56:03 2020  0.000000 seconds
```

例: 将系统时间写到 RTC 时间

```
# hwclock -w
```

例: 将 RTC 时间写入系统时间

```
# hwclock -s
```

(可以在/etc/inittab 中添加 /bin/hwclock -s 系统开机时自动读取 RTC 时间, 同步到系统时钟)