前言

关于海思的RTC，官方给出了一份文档

> RTC 应用指南.pdf

里面讲到需要编译对应的驱动和应用程序，代码地址位于

> Hi3531DV100_SDK_V1.0.5.0\drv\rtc

注意，如果你的arm-linux-gcc这个没有指向对应的工具链的话，就要先修改makefile，指定一下CC的路径

先编译好驱动和应用，并放到板子上加载，这部分的代码暂时没必要进行修改

使用

它的使用说明写的还是很清晰明了，先贴出在，后续会在它的基础上做一些工作

```
Usage: ./test [options] [parameter1] ...
Options:
        -s(set)             Set time/alarm,     e.g '-s time
2012/7/15/13/37/59'
        -g(get)             Get time/alarm,     e.g '-g alarm'
        -w(write)           Write RTC register, e.g '-w <reg> <val>'
        -r(ead)             Read RTC register,  e.g '-r <reg>'
        -a(alarm)           Alarm ON/OFF',      e.g '-a ON'
        -reset              RTC reset
        -b(battery monitor) battery ON/OFF,     e.g '-b ON'
        -f(requency)        frequency precise adjustment, e.g '-f <val>'
```

设置获取时间

通过如下命令可设置 RTC 时间：

```
#原型
./test -s time <year/month/day/hour/minute/second>
#举例
./test -s time 2012/7/15/13/37/59
```

通过如下命令可获取 RTC 时间：

```
./test -g time
```

打印输出如下

```
[RTC_RD_TIME]
Current time value:
year 2020
month 7
date 23
hour 13
minute 59
second 36
weekday 4
```

设置获取闹钟时间

通过如下命令可设置 RTC 闹钟时间：

```
./test -s alarm <year/month/day/hour/minute/second>
```

通过如下命令可获取 RTC 闹钟时间：

```
./test -g alarm
```

通过如下命令设置闹钟到期是否产生中断，驱动中断例程由用户根据需求自由补充。

```
./test -a ON/OFF
```

读取、设置 RTC 内部寄存器

通过如下命令可读取 RTC 内部寄存器，此功能多用于辅助调试。

```
./test -r <reg>
```

通过如下命令可设置 RTC 内部寄存器，此功能多用于辅助调试。

```
./test -w <reg> <value>
```

reg 取值，请参见各芯片的用户指南的实时时钟部分。

复位 RTC 模块

通过如下命令可复位 RTC 模块。

```
./test –reset
```

固定分频模式分频系数微调设置

通过如下命令可设置分频系数从而达到调整时钟的快慢效果。

```
./test -f <val>
```

`< val>`值为将要设置的分频系数的 10000 倍，例如要设置分频系数为 327.60，则val=3276000。通过直接敲`./test –f`命令可以查看当前分频系数，因为计算误差的问题，获取的分频系数可能和设置的分频系数有细小的偏差。分频系数可以配置范围为：`327.60~327.70`。

打开、关闭电池电量监测功能

通过如下命令可打开 RTC 电池电量监测功能。

./test –b ON

通过如下命令可关闭 RTC 电池电量监测功能。

./test –b OFF

解决linux系统时间问题

在linux系统中，应用层一般是通过标准API，如time去获取时间，在terminal中，也是通过date去查看和设置时间，现在海思自己整了一套rtc的东西出来，会导致date获取的时间，和海思api获取的时间不一致，而且经过我的测试，发现RTC的时间要慢于网络时间，误差还比较大，因此，针对rtc，需要做一些调整

首先，在加载脚本中，增加rtc的ko

```
vi /komod/load3531d
```

在合适的地方，添加ko加载代码

```
insmod hi_rtc.ko
```

其次，在启动脚本中，执行一个关于设置rtc的脚本，海思这个程序编出来叫test，我已经改名为hi_time了

```
#!/bin/sh

usage(){
    echo "just synchronizing system time : ./hi_rtc.sh"
    echo "set system time : ./hi_rtc.sh 2012/7/15/13/37/59"
}

get_time_form_hi_rtc(){
    year=`hi_time -g time   | grep year     | awk '{print $2}'`
    month=`hi_time -g time  | grep month    | awk '{print $2}'`
    date=`hi_time -g time   | grep date     | awk '{print $2}'`
```

```
    hour=`hi_time -g time   | grep hour     | awk '{print $2}'`
    minute=`hi_time -g time | grep minute   | awk '{print $2}'`
    second=`hi_time -g time | grep second   | awk '{print $2}'`
    echo "got time : ${year}.$month.$date-$hour:$minute:$second"
}

set_time_to_date(){
    if [ -n "$1" ];then
        hi_time -s time $1
    fi
    get_time_form_hi_rtc
    if [ -z "$year" ] || [ -z "$month" ] || [ -z "$date" ] \
        || [ -z "$hour" ] || [ -z "$year" ] || [ -z "$year" ];then
        echo "get time error, exit"
        exit 1
    fi
    time="${year}-${month}-${date} ${hour}:${minute}:${second}"
    date -s "$time"
}

wait_sys_ready() {
    for i in $(seq 1 $1)
    do
        rtc_dev=`ls /dev/hi_rtc`
        if [ -z "$rtc_dev" ];then
            echo "/dev/hi_rtc does not exist"
            sleep 1
        fi
    done

    rtc_dev=`ls /dev/hi_rtc`
    if [ -z "$rtc_dev" ];then
        echo "/dev/hi_rtc does not exist, time set failed"
        exit 1
    fi
}


main(){
    wait_sys_ready 5
    set_time_to_date $1
    hi_time -b ON
    exit 0
}

main $1
```

这个脚本，首先去获取 `/dev/hi_rtc` 设备是否存在，在系统启动后，加载完驱动的时候，这个设备节点其实还并不存在，得等一会才有，这个有点奇葩，暂时还没想明白是为什么，所以在测试过程中，出现了这种情况，调用海思的rtc程序去获取时间，发现获取不到，或者只获取到了一部分

```
got time : ..-::
get time error, exit
#或者
got time : ..15-13:41:47
get time error, exit
```

`wait_sys_ready`后面给的参数，是说尝试多少次，里面有sleep 1，如果多次尝试都读不到时间，就退出了

`set_time_to_date`这个函数带了一个参数，这个参数就是执行脚本时给的参数，可以用来设置海思的rtc时钟

如果有参数，就先设置rtc时钟，然后从rtc中读取时钟，并调用date接口，修改系统时间

这样在每次开机时，都会将系统时间与海思的rtc进行同步，于是就能通过标准API获取到时间了，注意一定要打开海思rtc的电池开关，不然时间不会保存

设置时区

默认情况下，时区不是中国的标准时间，所以开机后要设置一下时区

```
export TZ=CST-8
```

这句话建议加在启动的脚本，或者是什么地方，让系统自动执行

利用NTP获取网络时间

之前说过，rtc时钟的误差比较大，如果不知道怎么调硬件，或者寄存器，就最好去对齐网络时间

ntp需要去网上下源码进行编译

> http://doolittle.icarus.com/ntpclient/

这个源码比较小，只需要进去改一下makefile的CC，然后编译，然后将ntpclient程序拷贝到机器上即可

以下脚本实现了，定时去获取网络时间，一般情况下，这个脚本带一个 & ，让它后台执行，这个是死循环

如果检测到网络时间和rtc时间不一致，就自动对齐到网络时间

```
ntp_ip=120.25.108.11
#you can get ip from http://www.ntp.org.cn/pool.php
#ping that ip to check stability
check_time(){
    year=`hi_time -g time   | grep year     | awk '{print $2}'`
    month=`hi_time -g time  | grep month    | awk '{print $2}'`
    date=`hi_time -g time   | grep date     | awk '{print $2}'`
    hour=`hi_time -g time   | grep hour     | awk '{print $2}'`
    minute=`hi_time -g time | grep minute   | awk '{print $2}'`
    second=`hi_time -g time | grep second   | awk '{print $2}'`

    rtc_time="${year}-${month}-${date} ${hour}:${minute}:${second}"
    rtc_time=`date -d "$rtc_time" +%s`
    echo "rtc_time = $rtc_time"
    date_time=`date +%s`
    echo "date_time = $date_time"

    if [ "$rtc_time" != "$date_time" ];then
```

```
        date_time=`date "+%Y/%m/%d/%H/%M/%S"`
        hi_time -s time "$date_time"
        echo "synchronize network time successfully, time is $date_time"
    fi
}
get_time_form_ntp(){
    while [ 1 ]
    do
        ntp_state=`ntpclient -s -d -c 1 -i 5 -h $ntp_ip | grep "set time"`
        #echo $ntp_state
        if [ -n "$ntp_state" ];then
            check_time
        fi
        sleep 3
    done
}
main(){
    get_time_form_ntp
    exit 0
}
main
```

一开始是指定了从哪个Ip去获取网络时间，这个ip设的不合理，将导致获取可能失败，或者根本获取不到，建议去网站`http://www.ntp.org.cn/pool.php`搜一下，去ping一下那些ip，看哪个延迟小一点

`get_time_form_ntp`函数，是定时3S一次，死循环获取网络时间，如果获取到了，就执行`check_time`去检查时间是否一致

`check_time`这个函数里面用到的技巧，就是把时间全部转化为数字进行比较，具体是什么样子，也可以加打印看，此处的打印，后期最好是屏蔽掉，不然打印信息会有点多