EFI Real Time Clock driver

S. Eranian <eranian@hpl.hp.com>
March 2000

I/ Introduction

This document describes the efirtc.c driver has provided for
the IA-64 platform.

The purpose of this driver is to supply an API for kernel and user applications
to get access to the Time Service offered by EFI version 0.92.

EFI provides 4 calls one can make once the OS is booted: GetTime(),
SetTime(), GetWakeupTime(), SetWakeupTime() which are all supported by this
driver. We describe those calls as well the design of the driver in the
following sections.

II/ Design Decisions

The original ideas was to provide a very simple driver to get access to,
at first, the time of day service. This is required in order to access, in a
portable way, the CMOS clock. A program like /sbin/hwclock uses such a clock
to initialize the system view of the time during boot.

Because we wanted to minimize the impact on existing user-level apps using
the CMOS clock, we decided to expose an API that was very similar to the one
used today with the legacy RTC driver (driver/char/rtc.c). However, because
EFI provides a simpler services, not all ioctl() are available. Also
new ioctl()s have been introduced for things that EFI provides but not the
legacy.

EFI uses a slightly different way of representing the time, noticeably
the reference date is different. Year is the using the full 4-digit format.
The Epoch is January 1st 1998. For backward compatibility reasons we don't
expose this new way of representing time. Instead we use something very
similar to the struct tm, i.e. struct rtc_time, as used by hwclock.
One of the reasons for doing it this way is to allow for EFI to still evolve
without necessarily impacting any of the user applications. The decoupling
enables flexibility and permits writing wrapper code is ncase things change.

The driver exposes two interfaces, one via the device file and a set of
ioctl()s. The other is read-only via the /proc filesystem.

As of today we don't offer a /proc/sys interface.

To allow for a uniform interface between the legacy RTC and EFI time service,
we have created the include/linux/rtc.h header file to contain only the
"public" API of the two drivers. The specifics of the legacy RTC are still
in include/linux/mc146818rtc.h.

III/ Time of day service

The part of the driver gives access to the time of day service of EFI.
Two ioctl()s, compatible with the legacy RTC calls:

Read the CMOS clock: ioctl(d, RTC_RD_TIME, &rtc);

Write the CMOS clock: ioctl(d, RTC_SET_TIME, &rtc);

The rtc is a pointer to a data structure defined in rtc.h which is close to a struct tm:

```
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;
};
```

The driver takes care of converting back an forth between the EFI time and this format.

Those two ioctl()s can be exercised with the hwclock command:

For reading:
```
# /sbin/hwclock --show
Mon Mar  6 15:32:32 2000  -0.910248 seconds
```

For setting:
```
# /sbin/hwclock --systohc
```

Root privileges are required to be able to set the time of day.

IV/ Wakeup Alarm service

EFI provides an API by which one can program when a machine should wakeup, i.e. reboot. This is very different from the alarm provided by the legacy RTC which is some kind of interval timer alarm. For this reason we don't use the same ioctl()s to get access to the service. Instead we have introduced 2 news ioctl()s to the interface of an RTC.

We have added 2 new ioctl()s that are specific to the EFI driver:

Read the current state of the alarm
ioctl(d, RTC_WKLAM_RD, &wkt)

Set the alarm or change its status
ioctl(d, RTC_WKALM_SET, &wkt)

The wkt structure encapsulates a struct rtc_time + 2 extra fields to get status information:

```
struct rtc_wkalrm {

        unsigned char enabled; /* =1 if alarm is enabled */
```

```
        unsigned char pending; /* =1 if alarm is pending  */

        struct rtc_time time;
}
```

As of today, none of the existing user-level apps supports this feature. However writing such a program should be hard by simply using those two ioctl().

Root privileges are required to be able to set the alarm.

V/ References.

Checkout the following Web site for more information on EFI:

http://developer.intel.com/technology/efi/