Epson RX8010 Linux Driver - February 3, 2014 _ K3.8-v1.3
==========
Added second interrupt

Epson RX8010 Linux Driver - January 22, 2014 _ K3.8-v1.2
==========
Added register read/write and voltage low detection read/clear via ioctl.

Epson RX8010 Linux Driver - December 3, 2013
==========
Added implementation of IRQ.

Epson RX8010 Linux Driver - November 7, 2013
==========
The RX8010 Linux device driver provides the means for an application running in user space to access the Epson RX8010 RTC.

The driver can be easily modified according to the user's requirements and rebuilt. The driver was tested using Linux kernel 3.8.x and was developed using the ARMhf Ubuntu 12.04LTS distribution on a BeagleBone Black. It is expected that changes and additions will be required for driver implmentation on other platforms/interfaces based on the specific requirements of those platforms.

Installing the Driver
To install the driver source:

1. Copy the file rtc-rx8010.c into the directory ./linux-3.8.x/drivers/rtc. "linux-3.8.x" refers to the base of the linux kernel source tree.

2. Add the following lines into the drivers/rtc/Kconfig file:

```
config RTC_DRV_RX8010
  tristate "Epson RX8010SJ"
  help
    If you say yes here you get support for the Epson
    RX8010SJ RTC chip.

    This driver can also be built as a module. If so, the module
    will be called rtc-rx8010.
```

3. Add the following line to the drivers/rtc/Makefile:

```
obj-$(CONFIG_RTC_DRV_RX8010)  += rtc-rx8010.o
```

4. During the rebuild, make sure to include the 'Epson RX8010SJ' option in the Kernel Configuration window under 'Device Drivers' -> 'Real Time Clock'.

Hardware Considerations

The testing of the driver was done using an Epson RX8010 part mounted on a breadboard connected to the BeagleBone Black P9 header. The connections are as follows:

RX8010 pin	BeagleBone Black P9	Notes
#1 NC	--	
#2 NC	--	
#3 /IRQ2	P9_27	pull-up resistor recommended (see RX8010 spec)
#4 GND	P9_01	
#5 SDA	P9_20	pull-up resistor recommended (see RX8010 spec)
#6 SCL	P9_19	pull-up resistor recommended (see RX8010 spec)
#7 /IRQ1	P9_23	pull-up resistor recommended (see RX8010 spec)
#8 VDD	P9_03	

Testing Environment

The following assumes that the RX8010 driver has been modified as required, rebuilt, and included in the linux build as either a built-in driver or a module.

To test the driver, the "RTC breadboard" was setup as a cape on the BeagleBone Black header P9. This requires adding the RX8010 RTC into the device tree using a device tree overlay (.dtbo). A sample device tree source file (.dts) and the compiled .dtbo file are included in the driver package. These files must be copied to the /lib/firmware directory on the BeagleBone Black system.

Before loading the cape you have to enable GPIO modules. Please see note (at the end of this document) to create the bbb_enable_gpio.sh script to work around this problem.

Once the system has booted, change to the /sys/devices/bone_capemgr.# directory (where # is the number). Type the following (you may need root access depending on your setup/configuration):

```
./bbb_enable_gpio.sh
echo BBB-RX8010 > slots
cat slots
```

The RX8010 cape should now be listed in one of the slots and the driver should be loaded at /dev/rtc1 (again depending on your configuration). To confirm that the driver has loaded, type:

```
dmesg | grep rx8010
```

To test the read/write capabilities of the RX8010, we can now use the hwclock command. To get the current time, type:
hwclock -r -f /dev/rtc1

If the time has not been set yet, the rtc should return with Jan 1, 1970. Next, we can set the system date with the following example:
date -s "Thurs Nov 07 11:33:00 PDT 2013"

Write the new date to the clock:
hwclock -w -f /dev/rtc1

And read it back:
hwclock -r -f /dev/rtc1

The RX8010 is now set with the new time.

The driver package also includes small sample applications, rtctest.c and ioctl.c.
The rtctest demonstrates how to read and write the time from/to the RX8010 using the ioctls RTC_RD_TIME and RTC_SET_TIME.
The ioctl demonstrates how to read and write to the registers of RX8010.

For instance, to read register (01h):
./ioctl read 10
REG[10h]=>29h

to write a value (20h) to register (10h):
./ioctl write 10 20
REG[10h]<=20h

to read Voltage Low Flag bit status:
./ioctl vlr
0

to clear Voltage Low Flag bit status:
./ioctl vlc

Note:

1. Since the RX8010 is an I2C device, the I2C Tools package may be useful for debugging purposes. If your system does not already include the i2cdetect, i2cget, i2cset, etc., install the I2C tools package ("apt-get install i2c-tools" for Ubuntu distros).

2. To create bbb_enable_gpio.sh script copy and paste to your BeagleBone Black terminal the following lines and then: chmod +x bbb_enable_gpio.sh

```
echo '#!/bin/bash

#https://github.com/piranha32/IOoo/blob/master/examples/bbb_enable_gpio.sh
#workaround for a bug in bone-pinmux-helper to enable GPIO modules.
#based on the code from Luigi Rinaldi:
#https://groups.google.com/forum/!msg/beagleboard/OYFp4EXawiI/Mq6s3sg14HoJ
EXPORT=/sys/class/gpio/export
echo 5 > /sys/class/gpio/export
echo 65 > /sys/class/gpio/export
echo 105 > /sys/class/gpio/export' >> bbb_enable_gpio.sh
```