Epson RX8900 I2C Linux Driver - February 3, 2014 - K3.8-v1.0
==========
No functional change

Epson RX8900 I2C Linux Driver - January 20, 2014 - K3.8-v1.0_Beta
==========
The RX8900 I2C Linux device driver provides the means for an application running in user space to access the Epson RX8900 RTC.

The I2C 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 implementation 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-rx8900.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_RX8900
        tristate "Epson RX8900SA/LC"
        help
          If you say yes here you get support for the Epson
          RX8900SA/LC RTC chips.

          This driver can also be built as a module. If so, the module
          will be called rtc-rx8900.
```

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

```
obj-$(CONFIG_RTC_DRV_RX8900)+= rtc-rx8900.o
```

4. During the rebuild, make sure to include the 'Epson RX8900SA/LC' option in the Kernel Configuration window under 'Device Drivers' -> 'Real Time Clock'.

Hardware Considerations

The testing of the driver was done using an Epson RX8900 part mounted on a breadboard connected to the BeagleBone Black P9 header. The connections are as follows:

| RX8900 pin | BeagleBone Black P9 | Notes |
|---|---|---|
| GND | P9_01 | |
| VDD | P9_03 | |
| SCL | P9_19 | /* i2c2 scl: gpio0_13 */ |
| SDA | P9_20 | /* i2c2 sda: gpio0_12 */ |
| /INT | P9_23 | /* gpio1_17: used for IRQ */ |

See Note and pull-up resistor recommended (see RX8900 spec)

Testing Environment

The following assumes that the RX8900 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 RX8900 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.

Once the system has booted type the following (you may need root access depending on your setup/configuration):
echo BBB-RX8900 > /sys/devices/bone_capemgr.*/slots
cat /sys/devices/bone_capemgr.*/slots

The RX8900 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 rx8900

To test the read/write capabilities of the RX8900, 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 RX8900 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 RX8900 using the ioctls RTC_RD_TIME and RTC_SET_TIME.
The ioctl demonstrates how to read and write to the registers of RX8900.

For instance, to read register (0h):
./ioctl read 0
REG[00h]=>29h

to write a value (20h) to register (0h):
./ioctl write 0 20
REG[00h]<=20h

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

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