Epson RA4000CE and RA8000CE SPI/I2C Linux Driver - May 2022_K5.15.32-v1.0

The RA4000CE SPI and RA8000CE I2C Linux device driver provides the means for an application running in user space to access these Epson RTC devices from a single Linux device driver. Since the driver supports both the RA4000CE and RA8000CE RTC devices, the driver is named rtc-ra48000ce.

The driver can be easily modified according to the user's requirements and rebuilt. The driver was tested using Linux kernel 5.15.32 and was developed natively on a Raspberry PI 4 using the **Raspberry PI OS Lite (64-bit)** distribution (released 2022-04-04). 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.

To create an SDCARD image with the Raspberry PI OS image, download and run the Raspberry PI Imager Utility.
Windows 10: https://downloads.raspberrypi.org/imager/imager_latest.exe
Ubuntu PC: https://downloads.raspberrypi.org/imager/imager_latest_amd64.deb

Select the 64-bit flavor of the Raspberry PI OS (lite version). Currently, it has a release date of 2022-04-04. If it is newer, then the kernel release may be different than 5.15.32.

Choose OS	-> **Raspberry PI OS (other)** -> **Raspberry PI OS Lite (64-bit)**
Click **Gear Icon**	-> **Set hostname** -> use something like ***rtcdev*** or any other valid host name
	-> **Enable SSH** -> **Use password authentication**
	-> **Set username and password** -> username = **pi**, password = *some_password*
	-> **Set locale settings** -> Time zone = ***America/Los_Angeles*** (or your specific time zone)
	-> Keyboard layout = ***us*** (or your specific keyboard layout)
Choose Storage	-> Select inserted **32GB** (or larger) SDCARD which will be ERASED!
Click **Write**	wait for the image to download and be written to the SDCARD

Insert the newly imaged SDCARD into the RPI (metal fingers up) and boot. The first boot takes a while to prepare the new image. The RPI can be connected to an HDMI display and a keyboard can be used to communicate with the RPI. If the RPI is connected to a network, the RPI can also be used via an SSH connection from another computer. The IP address of the RPI must be known, and this can be obtained by typing **hostname -I** from the keyboard, the results will be shown on the HDMI display (for example *192.168.1.42*).

Use the HDMI display and keyboard, or from another computer, ssh into the RPI:
ssh pi@*192.168.1.42* (use the actual IP address shown by **hostname -I** above)

Note of command input.

Don’t omit "sudo" of all command head.

If nothing "sudo" in command, don't add "sudo",

1. Perform the following preparation steps.

a) Update the Raspberry PI OS image to the latest packages:
sudo apt update
sudo apt full-upgrade

b) Set your desired locale. **en_US.UTF-8** is an example (see **/etc/locale.gen** for a list of all valid locales):
sudo raspi-config nonint do_change_locale *en_US.UTF-8*
Note that Raspberry PI OS defaults to: en_GB.UTF-8

c) Enabled the SPI and I2C interfaces:
sudo raspi-config nonint do_i2c 0
sudo raspi-config nonint do_spi 0

d) Optionally, to disable auto-login on the HDMI display, enter this command:
sudo rm -f /etc/systemd/system/getty@tty1.service.d/autologin.conf

e) Reboot the Raspberry PI for the above changes to take effect:
sudo reboot

f) All extracted files from Zip-file, copy to working directory.:

Example of create working directory.

```
>cd
>mkdir EPSON-RTC
>cd EPSON-RTC
>mkdir RX48901
>mkdir RA48000
>sudo chmod 777 RX48901
>sudo chmod 777 RA48000
>cd
>sudo chmod 777 EPSON-RTC
>cd EPSON-RTC
>cp rx48901-linux.tar.gz RX48901
>cp ra48000-linux.tar.gz RA48000
```

Note; Permission change by Chmod 777 is important.

2. Using the RTC driver as an out-of-tree kernel module.

Building an out-of-tree kernel module allows the kernel module to be used without having to rebuild the entire Linux kernel. It is useful for rapid development and testing.

a) Get Linux kernel headers (only needed to install once):
sudo apt install raspberrypi-kernel-headers
Will be installed to: /usr/src/linux-headers-`*uname -r*`

b) Build the RA4000CE and the RA8000CE device trees and install to /boot/overlays:
cd ra48000-linux
sudo dtc -@ -I dts -O dtb ra4000ce-overlay.dts -o /boot/overlays/ra4000ce.dtbo
sudo dtc -@ -I dts -O dtb ra8000ce-overlay.dts -o /boot/overlays/ra8000ce.dtbo

c) Enable either the RA4000CE or RA8000CE device tree (not both at the same time):
sudo nano /boot/config.txt
Append only ONE of these lines depending on which RTC part is being used:
dtoverlay=ra4000ce
or
dtoverlay=ra8000ce
Press **CTRL-S** to save changes, press **CTRL-X** to exit the nano editor.

d) Reboot the Raspberry PI for the selected RTC device tree to take effect:
sudo reboot

e) After rebooting, build and load the RTC kernel module:
cd ra48000-linux
make
sudo make load

f) Show kernel messages to check if RTC kernel module loaded OK (example shown for RA4000CE):
dmesg

```
[41.603642] rtc_ra48000ce: loading out-of-tree module taints kernel.
[41.604063] SPI driver ra4000ce has no spi_device_id for epson,ra8000ce
[41.604238] ra4000ce spi0.0: SPI settings: bits_per_word=8, max_speed_hz=2000000, mode=7h
[41.604775] ra4000ce spi0.0: ra48000_probe: IRQ=66
[41.604818] ra4000ce spi0.0: Voltage was low, data is invalid!
[41.605320] ra4000ce spi0.0: registered as rtc0
[41.605366] ra4000ce spi0.0: hctosys: unable to read the hardware clock
```

The message "SPI driver ra4000ce has no spi_device_id for epson,ra8000ce" can be safely ignored.

g) If first time RTC power-up or RTC has lost power, RTC data will be invalid. Copy current system time to RTC:
sudo hwclock -w

h) Now test if time can be read from RTC:
sudo hwclock -r
This should display something like: *2022-05-15 10:24:12.609572-07:00*

i) The RTC kernel module can be unloaded with:
sudo make unload

Loading and unloading the RTC kernel module is transient for this session only. To permanently install the RTC module into the kernel tree, use:
sudo make install
This will create the file /lib/modules/`*uname -r*`/updates/rtc-ra48000ce.ko and add it to the kernel's dependencies.

If the RTC has a backup power supply, after a complete power cycle, the RTC kernel module will auto-load on reboot and set the system time to the RTC time.

To confirm the RTC kernel modules is installed and working, shutdown the Raspberry PI:
sudo poweroff
Wait until RPI green LED goes out, then remove power for over 1 minute (or more).
After rebooting from a clean power cycle, confirm the RTC kernel module was automatically loaded by kernel:
dmesg | grep 000ce

```
[7.395444] rtc_ra8000ce: loading out-of-tree module taints kernel.
[7.453069] SPI driver ra4000ce has no spi_device_id for epson,ra8000ce
[7.472907] ra4000ce spi0.0: SPI settings: bits_per_word=8, max_speed_hz=2000000, mode=7h
[7.476739] ra4000ce spi0.0: ra8000_probe: IRQ=66
[7.477069] ra4000ce spi0.0: ra8000_get_time: sec=35 min=37 hour=17 mday=15 mon=5 year=2022 wday=0
[7.477355] ra4000ce spi0.0: ra8000_read_alarm: alarm read 00 00 00
[7.477451] ra4000ce spi0.0: ra8000_read_alarm: sec=0 min=0 hour=0 mday=0 mon=0 wday=-1 enabled=0 pending=0
[7.478376] ra4000ce spi0.0: ra8000_get_time: sec=35 min=37 hour=17 mday=15 mon=5 year=2022 wday=0
[7.479124] ra4000ce spi0.0: ra8000_get_time: sec=35 min=37 hour=17 mday=15 mon=5 year=2022 wday=0
[7.571155] ra4000ce spi0.0: registered as rtc0
[7.595778] ra4000ce spi0.0: ra8000_get_time: sec=35 min=37 hour=17 mday=15 mon=5 year=2022 wday=0
[7.597366] ra4000ce spi0.0: setting system clock to 2022-05-15T17:37:35 UTC (1652636255)
```

Now test if time can be read from RTC:
sudo hwclock -r
Should display something like: *2022-05-15 10:38:21.526091-07:00*

To remove the RTC kernel module from the kernel tree (will no longer auto-load RTC kernel module on reboot):
sudo make uninstall

3. Using the RTC driver compiled into kernel tree.

Building a kernel driver into the kernel tree allows the driver to be incorporated into the compiled kernel as either a kernel module or embedded directly into the kernel binary image.

a) Get the Raspberry PI Linux kernel sources. The current kernel 5.15.32 source can be downloaded from:
https://github.com/raspberrypi/linux/tree/1.20220331
as a **.zip** file. Alternatively, the current kernel 5.15.32 source can be downloaded using git:
sudo apt install git
git clone https://github.com/raspberrypi/linux.git --branch 1.20220331 --depth 1
The location of the kernel source directory will be referred to as: *path_to_kernel*

b) Switch to the RTC driver package working directory:
cd ra48000-linux

c) Copy the device tree overlay files into the kernel source tree:
cp ra4000ce-overlay.dts *path_to_kernel***/arch/arm/boot/dts/overlays/**
cp ra8000ce-overlay.dts *path_to_kernel***/arch/arm/boot/dts/overlays/**

d) Edit the file "*path_to_kernel***/arch/arm/boot/dts/overlays/Makefile**" to add the new RTC overlays.
Insert the following two lines after this line: rra-digidac1-wm8741-audio.dtbo \
**ra4000ce.dtbo **
**ra8000ce.dtbo **

e) Copy the new RTC driver source code into the kernel source tree:
cp rtc-ra48000ce.c *path_to_kernel***/drivers/rtc/**

f) Edit the file "*path_to_kernel***/drivers/rtc/Makefile**" to add the new RTC driver.
Insert the following line after this line: obj-$(CONFIG_RTC_DRV_RX4581) += rtc-rx4581.o
obj-$(CONFIG_RTC_DRV_RA48000CE) += rtc-ra48000ce.o

g) Edit the file "*path_to_kernel***/drivers/rtc/Kconfig**" to add the new RTC driver configuration.
Copy and paste the contents of the file: **rtc-ra48000ce-Kconfig.txt**
in the section immediately after the line: comment "SPI and I2C RTC drivers"

h) Configure, build, and install the new updated kernel.
From the detailed instructions (see link below), make sure to issue these commands:
KERNEL=kernel8
make bcm2711_defconfig
before configuring the kernel with:
make menuconfig
Note that "menuconfig" requires the dependency **libncurses5-dev** to be installed.
Configure the kernel to enable **RTC_DRV_RA48000CE** as either a kernel module or an embedded kernel driver.
For further detailed instructions on how to configure, compile, and install a Raspberry PI kernel, please see the following: https://www.raspberrypi.com/documentation/computers/linux_kernel.html
Follow specific instructions for building the 64-bit kernel, not the 32-bit kernel.

i) Enable either the RA4000CE or RA8000CE device tree (not both at the same time):
sudo nano /boot/config.txt
Append only ONE of these lines depending on which RTC part is being used:
dtoverlay=ra4000ce
or

dtoverlay=ra8000ce
Press **CTRL-S** to save changes, press **CTRL-X** to exit the nano editor.

j) Reboot the Raspberry PI for the selected RTC device tree to take effect in the newly built kernel:
sudo reboot

k) Show kernel messages to check if RTC kernel driver loaded OK (example shown for RA4000CE):
dmesg | grep 000ce

```
[7.635757] SPI driver ra4000ce has no spi_device_id for epson,ra8000ce
[7.636222] ra4000ce spi0.0: SPI settings: bits_per_word=8, max_speed_hz=2000000, mode=7h
[7.640330] ra4000ce spi0.0: ra48000_probe: IRQ=66
[7.640508] ra4000ce spi0.0: Voltage was low, data is invalid!
[7.656566] ra4000ce spi0.0: registered as rtc0
[7.657174] ra4000ce spi0.0: hctosys: unable to read the hardware clock
```

The message "SPI driver ra4000ce has no spi_device_id for epson,ra8000ce" can be safely ignored.

l) If first time RTC power-up or RTC has lost power, RTC data will be invalid. Copy current system time to RTC:
sudo hwclock -w

m) Now test if time can be read from RTC:
sudo hwclock -r
This should display something like: *2022-05-15 10:44:17.357893-07:00*

If the RTC has a backup power supply, after a complete power cycle, the RTC kernel driver will auto-load on reboot and set the system time to the RTC time.

4. Additional RTC Setup Information.

The RTC interrupt pin defaults to GPIO 4. It can be changed in /boot/config.txt like this (17 is an example):
dtoverlay=ra4000ce,interrupt=17
or
dtoverlay=ra8000ce,interrupt=17

The RA4000CE SPI chip select pin defaults to CE0. It can be changed to CE1 in /boot/config.txt like this:
dtoverlay=ra4000ce,ce1

5. RTC Module Connection to Raspberry PI.

The following connections must be made between the RTC module and the Raspberry PI J8 pin header. Only one RTC module (RA4000CE or RA8000CE) shall be connected to the Raspberry PI at the same time.

RA4000CE pin	**Raspberry PI J8 pin**	**Notes**
1 DI	19 MOSI	SPI master-out slave-in signal
2 V_{DD}	1 or 17 3.3V	Connect 0.1uF capacitor to GND
3 CE	24 CE0	SPI chip select signal
4 /INT	7 GPIO 4	RTC interrupt signal (requires a 10Kohm pull-up resistor to 3.3V)
5 /RST		*n/c*
6 EVIN2		*n/c*
7 DO	21 MISO	SPI master-in slave-out signal
8 CLK	23 SCLK	SPI clock signal
9 GND	25 GND	Connected to any Raspberry PI J8 GND pin (6,9,14,20,25,30,34,39)
10 SOUT		*n/c*

RA8000CE pin	**Raspberry PI J8 pin**	**Notes**
1 EVIN1		*n/c*
2 V_{DD}	1 or 17 3.3V	Connect 0.1uF capacitor to GND
3 /INT	7 GPIO 4	RTC interrupt signal (requires a 10Kohm pull-up resistor to 3.3V)
4 FOUT		*n/c*
5 N.C.		*n/c*
6 EVIN2		*n/c*
7 SDA	3 SDA	I2C data signal
8 SCL	5 SCL	I2C clock signal
9 GND	6 GND	Connected to any Raspberry PI J8 GND pin (6,9,14,20,25,30,34,39)
10 SOUT		*n/c*

The V_{DD} pin can be optionally connected to a 3V backup power supply (for example a CR2032 battery) to provide continuous RTC power. It is recommended that a ~50ohm resistor is placed in series between the V_{DD} pin and the backup power supply's positive output. If the V_{DD} pin is connected to a backup power supply, then it shall not be connected to pin 1 or 17 of the Raspberry PI J8 connector.

Note:

The connect of INT and GPIO4 is very important.

When INT and GPIO4 is disconnect, driver returns Error and stopped.

6. RTC SRAM Interface.

The RA4000CE and RA8000CE RTC modules have 32-bytes of SRAM that will remain valid as long as the RTC has an active power supply.

This can be viewed (as a binary hex dump) with:
hexdump -Cv /sys/bus/nvmem/devices/ra48000ce_nvmem0/nvmem

It can be "zeroed" with the command:
sudo dd if=/dev/zero bs=32 count=1 of=/sys/bus/nvmem/devices/ra48000ce_nvmem0/nvmem

Example of writing 32-bytes of random data:
sudo dd if=/dev/urandom bs=32 count=1 of=/sys/bus/nvmem/devices/ra48000ce_nvmem0/nvmem

Example of writing the contents of a file (32-bytes or less):
echo "This is some text in RTC SRAM" > test.txt
sudo dd if=test.txt of=/sys/bus/nvmem/devices/ra48000ce_nvmem0/nvmem

6. Using only the RA4000CE or RA8000CE RTC to set the system time.

If the RTC has a backup power supply, and if the RTC kernel module is installed in the kernel tree, then the RTC can set the Raspberry PI's system time without requiring an internet connection or using the "fake" hardware clock service.

To disable these time-keeping services, use:
sudo systemctl disable fake-hwclock.service
sudo systemctl disable systemd-timesyncd.service