Preparations

- Odroid-C2/C4/HC4
- uSD-Card ⇒ over 8GB or eMMC Module
- PCF8563 RTC Module
- Requirement Linux kernel version
  - Odroid-C2
    - 3.14.65-66 or higher
  - Odroid-C4
    - 4.9.113 or higher

Configuration

Ubuntu

Insert the RTC Shield on your Odroid as the above pictures and turn on the system.

Then enable the RTC Shield driver in the Device Tree file and make sure the I2C driver loaded.

- Odroid-C2
- Odroid-C4/HC4

target

```
sudo apt install device-tree-compiler

sudo fdtput -t s /media/boot/meson64_odroidc2.dtb /i2c@c1108500/pcf8563@51 stat
us "okay"

echo "aml_i2c"|sudo tee -a /etc/modules

echo "rtc_pcf8563"|sudo tee -a /etc/modules

sudo reboot
```

Check the I2C & RTC module status after reboot, like the following contents.

- Odroid-C2
- Odroid-C4/HC4

target

```
odroid@odroid64:~$ cat /etc/modules

# /etc/modules: kernel modules to load at boot time.

#

# This file contains the names of kernel modules that should be loaded

# at boot time, one per line. Lines beginning with "#" are ignored.
```

```
aml_i2c
rtc_pcf8563
odroid@odroid64:~$ ls /dev/rtc*
/dev/rtc  /dev/rtc0
odroid@odroid64:~$ lsmod
.....
rtc_pcf8563             3996  0
aml_i2c                16898  0
.....
```

- If you want to choose a specific rtc file when using hwclock command, use **-f** option.
    - hwclock -s -f /dev/rtc1

Next, if you want to update system Time/Calendar from the Hardware RTC shield at boot time,

`Method one` is edit /etc/rc.local file.
Add the following line into the end of /etc/rc.local file.

target

```
hwclock -s
```

Then the contents will be like the following.

```
if [ -f /aafirstboot ]; then /aafirstboot start ; fi


hwclock -s


exit 0
```

Make sure to put that line into above the “exit 0” line. Then reboot your Odroid to take effect.

target

```
reboot
```

If everything worked correctly, the RTC shield on your Odroid should be initialized on boot and the stored current date and time will be loaded into Linux.
If it doesn't, check if the “rtc-pcf8563” line exists in the outputs of “lsmod” command if your kernel has built with a module.

`Method two` has introduced by @e-pirate in our forum Go see. Thank you for calling attention to us.

- Create a file “50-rtc.rules” which is located at /etc/udev/rules.d/ with below line.

```
KERNEL=="rtc0", TAG+="systemd"
```

- Create a file “hwrtc.service” which located at /lib/systemd/system/ with below lines.

```
[Unit]
Description=Synchronise System clock to hardware RTC
DefaultDependencies=no
Wants=dev-rtc0.device
After=dev-rtc0.device
Before=systemd-journald.service time-sync.target sysinit.target shutdown.target
Conflicts=shutdown.target

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/sbin/hwclock --hctosys --rtc=/dev/rtc0 -v --utc --noadjfile
RestrictRealtime=yes

[Install]
WantedBy=sysinit.target
```

- And then apply by this command “*# sudo systemctl enable hwrtc*” and reboot

target

```
root@odroid:~# systemctl enable hwrtc
root@odroid:~# reboot
.
.
root@odroid:~# systemctl status hwrtc
 ● hwrtc.service - Synchronise System clock to hardware RTC
  Loaded: loaded (/lib/systemd/system/hwrtc.service; enabled; vendor preset:
enabled)
  Active: active (exited) since Fri 2021-01-15 11:23:08 KST; 12min ago
```

```
     Main PID: 2116 (code=exited, status=0/SUCCESS)

     Tasks: 0 (limit: 3839)

     Memory: 0B

     CGroup: /system.slice/hwrtc.service

     Jan 15 11:23:08 odroid hwclock[2116]: ioctl(4, RTC_UIE_ON, 0): Invalid arg
ument

     Jan 15 11:23:08 odroid hwclock[2116]: Waiting in loop for time from /dev/r
tc0 to change

     Jan 15 11:23:08 odroid hwclock[2116]: ...got clock tick

     Jan 15 11:23:08 odroid hwclock[2116]: Time read from Hardware Clock: 2021/
01/15 02:23:08

     Jan 15 11:23:08 odroid hwclock[2116]: Hw clock time : 2021/01/15 02:23:08
= 1610677388 seconds since 1969

     Jan 15 11:23:08 odroid hwclock[2116]: Time since last adjustment is 161067
7388 seconds

     Jan 15 11:23:08 odroid hwclock[2116]: Calculated Hardware Clock drift is 0
.000000 seconds

     Jan 15 11:23:08 odroid hwclock[2116]: Calling settimeofday(NULL, 0) to loc
k the warp_clock function.

     Jan 15 11:23:08 odroid hwclock[2116]: Calling settimeofday(NULL, -540) to
set the kernel timezone.

     Jan 15 11:23:08 odroid hwclock[2116]: Calling settimeofday(1610677388.0000
00, NULL) to set the System time.
```

You can check the current time on the RTC shield by the following command.

target

```
sudo hwclock -r
```

If this is the first time you have run the RTC shield on your Odroid, it will show like “APR 20th 2016”, the default DateTime set into.

If your Odroid can connect to the internet, the date and the time should be set automatically using the internet. Otherwise, you can set them manually by the following command.

target

```
sudo date -s "20 APR 2016 18:00:00"
```

You can check the current OS DateTime with a command; “date”.

To save the DateTime from Linux OS to the RTC shield, use the following command.

target

```
sudo hwclock -w
```

Read the DateTime on the RTC shield by the following command to check if it is saved correctly.

target

```
sudo hwclock -r
```

- Kernel update can break up the RTC functionality.
  - To avoid this problem, you can use this workaround that will patch the dtb all the time the kernel updates.

- Odroid-C2
- Odroid-C4

target

```
sudo -s

echo '#!/bin/sh' > /etc/kernel/postinst.d/rtc

echo 'fdtput -t s /media/boot/meson64_odroidc2.dtb /i2c@c1108500/pcf8563@51 sta
tus "okay"' >> /etc/kernel/postinst.d/rtc

chmod +x /etc/kernel/postinst.d/rtc
```

Disable RTC on Android

- On Android, **RTC works by default**. Please see this guide if you want to use I2C for other purposes.
- This guide works on Android 5.1.1 V2.8 or higher(ODROID-C2)/ Android Pie 64bit (202007xx) or higher(ODROID-C4).

THIS GUIDE IS FOR DISABLING THE RTC FUNCTIONALITY.

- ODROID-C2
- ODROID-C4

1. Open File Manager app.
2. Edit /storage/internal/boot.ini like the below. Check near the end of the file. If you use Android v4.0 or later, edit /internal/boot.ini instead.
   a. Before edit.

```
I. movi read dtb 0 ${dtbaddr}
II. # load kernel from vfat or boot partition.
III. movi read boot 0 ${loadaddr}
IV. #fatload mmc 0:1 ${loadaddr} Image_android
V. booti ${loadaddr} - ${dtbaddr}
```

b. After edit.

```
I. movi read dtb 0 ${dtbaddr}
II. # load kernel from vfat or boot partition.
III. #movi read boot 0 ${loadaddr}
IV. fatload mmc 0:1 ${loadaddr} Image_android
V. booti ${loadaddr} - ${dtbaddr}
```

3. Load a kernel image from the vfat partition which not include rtc module.

If you see a message like "'fatload' command not found", remove /storage/internal/boot.ini (or /internal/boot.ini) file and reboot system then try again.

Don't use NTP service

- If your network firewall is blocking the NTP service or has no internet connection, you must stop the NTP service. Once the NTP failed, the RTC value could be reset.
- This guide was made by **phaseshifter** from our forum. Thanks.
  - Here is the original post.

There're two ways to stop NTP daemon.

target

```
sudo /etc/init.d/ntp stop


# or,


sudo service ntp stop
```

To prevent it from starting at boot time, enter the following command.

target

```
sudo update-rc.d -f ntp remove
```

Estimated life cycle of the battery

We did cleanly install the Android 5.1 V3.3 image and measured the power consumption with Odroid-C2.

- When C2 turns on, it consumes less than **1uA**.
- When C2 turns off, it consumes around **15uA**.

According to the CR2032 Panasonic datasheet, its nominal capacity is **225mAh**.

- http://www.cr2032.co/cms/prodimages/panasonic_cr2032_datasheet.pdf

So its backup time could be around 15000 hours (~ two years). If on/off ratio is 1:1, it can be 3~4 years.