

RTC Device Driver User Guide

UG7407; Rev 1; 12/21

Abstract

This document provides an overview to the Linux real-time clock (RTC) Device Drivers and information on how to program and evaluate the MAX31343 RTC on the MAX31343 shield board (MAX31343SHLD) in a Linux environment.

Table of Contents

| 1. Linu | ux Device Driver | 5 |
|----------|--|-----|
| 1.1 W | hat is a Device Driver? | 5 |
| 1.2 W | ays to Implement a Driver as Kernel Module | 5 |
| 1.3 De | evice Driver Structure | 6 |
| 2. Devic | e Tree Parameters | 7 |
| 2.1 R | TC Device Trees | 7 |
| 2.2 De | evice Tree Compilation | 7 |
| 3. Linux | Generic RTC Device Drivers | 8 |
| 3.1 R | TC-Specific Default Features | 8 |
| 3.2 De | evice-Specific Custom Features | 9 |
| 4. MA | X31343 Device Driver Package | .11 |
| 4.1 Su | upported Features | .11 |
| 4.1. | 1 Basic Tests | .13 |
| 4.1. | 2 SysFs Entry Test of the Maxim Version Driver | .15 |
| 4.2 | Kernel Compilation Procedure | .21 |
| Referen | ces | .29 |
| Revisior | h History | .30 |

List of Figures

| Figure 1. Kernel image w/out modules. | 5 |
|---|----|
| Figure 2. Linux device management [1]. | 6 |
| Figure 3. Device tree workflow [2]. | 7 |
| Figure 4. System calls to device drivers [1]. | 8 |
| Figure 5. IOCTL parameters for MAX31343 | |
| Figure 6. Commands to prepare the Raspberry Pi for device driver compilation | |
| Figure 7. Command for compiling the device driver. | |
| Figure 8. Command for installing the device driver to the system | |
| Figure 9. Command for compiling and installing the device tree overlay | |
| Figure 10. Commands for disabling NTP. | |
| Figure 11. Command for changing the RTC time. | |
| Figure 12. Reading RTC time using systemctl. | |
| Figure 13. Reading RTC time using hwclock | |
| Figure 14. Sycnhronizing system clock with RTC | 15 |
| Figure 15. Reading power management mode | 15 |
| Figure 16. Reading the trickle charger settings | |
| Figure 17. Command for changing the user mode | |
| Figure 18. Reading RTC userram with hexdump | 16 |
| Figure 19. Writing data to RTC userram | |
| Figure 20. Reading the additional interrupt status. | |
| Figure 21. Command output when the interrupts are asserted | |
| Figure 22. Command output of RTC test script | |
| Figure 23. Installing dependencies | |
| Figure 24. Installing dependencies | |
| Figure 25. Command output of RTC Python test script. | 21 |
| Figure 26. Downloading the required tools for compiling the Raspberry Pi kernel | 21 |
| Figure 27. Downloading the Linux kernel for Raspberry Pi | 21 |
| Figure 28. Installing the system dependencies for kernel compiling | 22 |

| Figure 29. Device driver file located in kernel | 22 |
|--|----|
| Figure 30. Kconfig file after modifications. | 23 |
| Figure 31. Adding the device driver to Makefile of RTC drivers | 24 |
| Figure 32. Configuring the kernel before compiling | 24 |
| Figure 33. Selecting Device Drivers menu | 25 |
| Figure 34. Selecting Real Time Clock menu. | 26 |
| Figure 35. Enabling the Maxim MAX31343 device driver for compilation | 27 |
| Figure 36. Saving the configuration file. | 28 |
| Figure 37. Compiling the kernel | 29 |

List of Tables

| Table 1. Default RTC Features | 9 |
|--|----|
| | |
| Table 2. MAX31343 Kernel Version's Features. | 11 |
| | |
| Table 3. MAX31343 Pin Configuration. | 11 |

1. Linux Device Driver

This user guide explains Linux Device Driver concepts, especially the RTC drivers.

1.1 What is a Device Driver?

Device drivers are part of the Linux kernel. They make user space commands independent from system hardware designs so that there is no need for knowledge on hardware. There are generally two groups of users: board manufacturers and end customers.

1.2 Ways to Implement a Driver as Kernel Module

The Maxim RTC driver is implemented in two different ways:

- 1. Driver is released to kernel.org that is the main distribution point of source code for the Linux kernel.
 - a. The driver is included in the kernel that comes with the future Linux distribution. To compile the driver, KConfig (which is a Linux compilation feature extraction interface) must be configured to include the driver. See the *Kernel Compilation Procedure* section for more information.
- 2. Download driver from the Maxim website.
 - a. Add the driver into the downloaded kernel. Then, compile with the whole kernel. For more details, see the *Kernel Compilation Procedure* section.



Figure 1. Kernel image w/out modules.

b. Linux has a proper way to inject a driver into the running Linux distribution or Linux kernel module (LKM). The LKMs are not part of the main kernel. They are injected by users into the running kernel. The user can determine which LKM driver to include for kernel compilation. This helps to reduce the size of the kernel. For example, the ethernet LKM can be eliminated in a kernel compilation if an ethernet interface is not needed in a system. So, the bootloader and kernel module loader do not try to load the LKM into the RAM.

The LKM files are usually kept in the /lib/modules folder. They are loaded according to the distribution configuration such as device tree, scripts, etc.

1.3 Device Driver Structure



Kernel developers generate Linux's generic device driver. The device driver must support all the required features and functions. The driver code should report it in the device driver structures if optional features are not implemented.

2. Device Tree Parameters

Device drivers must learn hardware parameters due to the nature of the hardware environment. For example, the CPU's model, active core number, clock frequency, board's memory inputs, memory over bus drivers, which device is connected to which bus, almost everything is described in the device tree of the system. This feature supports Arm after Linux Kernel Version 3.7. The device tree is the hardware description for the kernel usually provided by the board manufacturer/provider.

2.1 RTC Device Trees

The RTC device trees usually contain information on the bus and signals connected to the RTC. Rest of the settings are device-specific parameters dependent on the RTC IC model and on-board design.

The bus number, interrupt lines, and device addresses are interpreted within the kernel. So, the driver only gets necessary bus structures from high-level application program interfaces (APIs). However, the device driver must have parser and default values for custom parameters like trickle charger settings or power management mode.



Figure 3. Device tree workflow [2].

2.2 Device Tree Compilation

The device tree has its own compiler to generate output. More information can be found here: <u>device-tree-compiler/manual.txt at master · vagrantc/device-tree-compiler · GitHub</u>.

3. Linux Generic RTC Device Drivers

The RTC device driver is a generic device driver for RTC ICs from manufacturers. System calls must be used in user space programs to access driver features from the user space. The IOCTL call is used to access the RTC features for the RTC driver.



Figure 4. System calls to device drivers [1].

3.1 RTC-Specific Default Features

M: Must have, O: Optional, N: Nice to have.

An invalid argument system code is returned to the user space if the must-have and optional features are not implemented. The user space programs can read the return code and respond to it if needed when the optional features are not implemented. Everything must be implemented for the must-have options.

Nice-to-have features may not be implemented in the kernel space.

| LINUX OPTIONS | DESCRIPTION |
|----------------------------|--|
| | |
| Set/Read Time (M) | RTC read/set time support |
| Set/Read Alarm (O) | Set/read alarm per user request |
| Periodic Interrupt (N) | Periodic interrupt support (2Hz to 8192Hz) |
| Alarm Interrupt (O) | Alarm interrupt generation based on user's alarm request |
| Update Interrupt (O) | On-demand update interrupt, usually 1 Hz |
| EPOCH (N) | RTC set/get time EPOCH format |
| NVMEM (N) | Battery-supported RAM or EEPROM-based memory within the RTC IC |
| Linux Power Management (N) | Registering a device as the wakeup of the system. Developer must set the CONFIG_PM_SLEEP option in the compilation parameters. |

Table 1. Default RTC Features.

An RTC driver is ready to release on kernel.org and work in a Linux system with the features mentioned in Table 1.

3.2 Device-Specific Custom Features

The RTC ICs may have more features than Linux requires. These features can be utilized through the IOCTL system calls and/or SysFs file interface.

Using the file APIs can change the device attribute or configuration using sysfs. A single sysfs file usually maps to a single attribute and is usually readable (and/or writable) using a simple text string. For example, the use of **cat** to read the state of the power management configuration and the **echo** shell command to change it.

Sysfs is a pseudo file system that can be used by the end/mid user. Those files are usually readonly files. But, some can take parameters through the command line. For example,

- \$ echo out >/sys/class/gpio/gpio24/direction
- \$ cat /sys/class/gpio/gpio24/direction

out

• \$ echo 1 >/sys/class/gpio/gpio24/value

The **cat** program reads data from a file in the Linux environment. Likewise, the **echo** program writes data to a file.

Every IOCTL option has its own function-specific code. The user and kernel spaces must have the same code numbers. So, the user space programs can use IOCTL functions through system calls.

Again, the user can use the command line or Linux's file API to access the sysfs as files.

The end/mid user must know the parameters to give and read in both the options. Figure 5 shows an example.

```
#define MXC_RTC_REG_READ ____IOWR('p', 0x20, int)
#define MXC_RTC_REG_WRITE ____IOW('p', 0x21, struct reg_data_s)
#define MXC_RTC_PWR_MGMT_READ ____IOR('p', 0x22, int)
#define MXC_RTC_PWR_MGMT_WRITE ____IOW('p', 0x23, int)
#define MXC_RTC_ALARM2_CONF_WRITE ____IOW('p', 0x24, struct alarm2_conf_s)
#define MXC_RTC_ALARM2_CONF_READ ____IOR('p', 0x25, struct alarm2_conf_s)
#define MXC_RTC_EXT_CLK_READ ____IOR('p', 0x26, int)
#define MXC_RTC_EXT_CLK_WRITE ____IOW('p', 0x27, int)
#define MXC_RTC_DATA_RET_READ ____IOR('p', 0x28, int)
#define MXC_RTC_DATA_RET_WRITE _____IOR('p', 0x29, int)
```

Figure 5. IOCTL parameters for MAX31343.

4. MAX31343 Device Driver Package

4.1 Supported Features

Table 2. MAX31343 Kernel Version's Features.

| STANDARD LINUX RTC CONFIGURATION OPTIONS | STATUS |
|---|---------------|
| Set/Read Time | Supported |
| Set/Read Alarm | Supported |
| Periodic Interrupt | Not Supported |
| Alarm Interrupt | Supported |
| Update Interrupt | Supported |
| EPOCH | N/A |
| Linux Power Management | Supported |
| NVMEM | Supported |

Here is the driver for Linux Kernel 4.x.x.



Table 3. MAX31343 Pin Configuration.

| MAX31343 | RPi 3 B+ |
|----------|----------------|
| Vcc | Pin1 - 3V3 |
| GND | Pin6 - Ground |
| SDA | Pin3 - GPIO2 |
| SCL | Pin5 - GPIO3 |
| INTA | Pin36 - GPIO16 |

- a. The driver must be compiled from the source code for portability. The Raspberry Pi kernel must be updated to the latest version before the compilation as shown in Figure 6.
- sudo apt-get update && sudo apt-get install --reinstall raspberrypi-bootloader raspberrypi-kernel

The RPi must then reboot with the **reboot** command.

| | | | | pi@raspb | errypi: ~ | | | | ~ | ^ | × |
|-------------------------|----------------|---------------------------|------------------------------------|---------------------|-----------|---------|---------|---------|----|-----|---------------|
| File | Edit | Tabs | Help | | | | | | | | |
| pi@ra | spber | r 🗙 | pi@raspberr | × | | | | | | | |
| pi@ras berryp | pbern i-boo | r ypi:∼ otloade | \$ sudo apt-get r raspberrypi-l | update && kernel | sudo a | apt-get | install | reinsta | 11 | ras | P 🔒 |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | $\overline{}$ |

Figure 6. Commands to prepare the Raspberry Pi for device driver compilation.

- b. Kernel headers must be installed to compile the MAX31343 LKM.
- sudo apt-get install raspberrypi-kernel-headers

The command installs the necessary kernel headers for development.

- c. RPi 3 B+ does not use i²c automatically. So, enable the i²c bus and RTC in /boot/config.txt.
- sudo nano /boot/config.txt

Use the command to open the file and add the following lines:

- o dtparam=i2c_arm=on
- o dtoverlay=i2c-rtc,MAX31343

- d. A Makefile is required to create the LKM from the driver file. A Makefile is provided to work with Raspberry Pi 3B+. The Makefile can be used to compile the driver, device tree, and install the LKM to run the Linux system.
- Go to the folder with the Makefile.
- The make clean command clears all outputs of the driver.
- The **make all** command compiles the driver and generates the LKM (.ko) file as shown in Figure 7.

| pi@raspberrypi:~/Desktop/max31343_linux_driver \$ make all |
|---|
| make -C /lib/modules/4.19.97-v7+/build M=/home/pi/Desktop/max31343_linux_driver modules |
| <pre>make[1]: Entering directory '/usr/src/linux-headers-4.19.97-v7+'</pre> |
| CC [M] /home/pi/Desktop/max31343_linux_driver/rtc-max31343.o |
| Building modules, stage 2. |
| MODPOST 1 modules |
| CC /home/pi/Desktop/max31343 linux driver/rtc-max31343.mod.o |
| LD [M] /home/pi/Desktop/max31343 linux driver/rtc-max31343.ko |
| make[1]: Leaving directory '/usr/src/linux-headers-4.19.97-v7+' |
| sudo cp rtc-max31343.ko /lib/modules/4.19.97-v7+/kernel/drivers/rtc |
| pi@raspberrypi:~/Desktop/max31343 linux driver \$ |
| Figure 7. Commond for commiling the device driver |

Figure 7. Command for compiling the device driver.

 The sudo make install command installs the driver into the LKM folder as shown in Figure 8. Most system folders are not accessible to users. Use the sudo command to allow make command to make changes to system directories. This step is required for installing the driver to the system.



Figure 8. Command for installing the device driver to the system.

• The **make dtbs** command compiles the device tree overlay (i²c-rtc-overlay.dts) as shown in Figure 9 and generates the i²c-rtc.dtbo output. It copies the output to RPi's /boot/overlays folder if the compilation is successful.

pi@raspberrypi:~/Desktop/max31343_linux_driver \$ make -q dtbs
pi@raspberrypi:~/Desktop/max31343 linux driver \$ _

Figure 9. Command for compiling and installing the device tree overlay.

Restart the Raspberry Pi after these steps. The driver is ready for testing.

4.1.1 Basic Tests

The **timedatectI** and **hwclock** commands can be used for basic tests such as setting/reading time. These commands are provided by the Raspberry Pi OS. So, they do not require any installation.

a. Manually disable the Network Time Protocol (NTP) before any test. The NTP updates the system time periodically if not disabled.

• sudo timedatectl set-ntp no

The command disables the NTP as shown in Figure 10.



Figure 10. Commands for disabling NTP.

- b. The hwclock command can be used to set the time in the RTC.
- Set RTC time: "sudo hwclock -- set --date "1/27/2020 14:50:00"

```
pi@raspberrypi:~/Desktop $ sudo hwclock --set --date "1/27/2020 14:50:00"
```

Figure 11. Command for changing the RTC time.

• Read RTC time: "timedatectl" or "sudo hwclock -r"

```
pi@raspberrypi:~/Desktop S timedatectl
Local time: Mon 2020-01-27 11:50:20 GMT
Universal time: Mon 2020-01-27 11:50:20 UTC
RTC time: Mon 2020-01-27 14:50:03
Time zone: Europe/London (GMT, +0000)
System clock synchronized: yes
NTP service: inactive
RTC in local TZ: no
```

Figure 12. Reading RTC time using systemctl.

pi@raspberrypi:~/Desktop \$ sudo hwclock -r 2020-01-27 11:47:05.120164+00:00

Figure 13. Reading RTC time using hwclock.

• Synchronizing system clock with RTC: "sudo hwclock --hctosys"

Figure 14. Sycnhronizing system clock with RTC.

4.1.2 SysFs Entry Test of the Maxim Version Driver

There is only one writable sysfs entry (userram) on this version of the driver. This interface is a nice-to-have feature from the Linux kernel, which is defined as the sysfs. Other sysfs entries are there to read some of the important configurations. Sysfs entries can be found in the "/sys/class/rtc/rtc<device_no>/device" folder.

a. The **cat power_mgmt** command reads the current power-management mode and backup battery threshold from the RTC as shown in Figure 15. Change these settings using the IOCTL or device tree.

| | | | pi@raspberrypi: ~/Desktop | |
|--|--|--|---|-------------|
| File | Edit | Tabs | Help | |
| pi@ra pi@ra pi@ra pi@ra Reg V Backu pi@ra pi@ra | spbern spbern spbern spbern alue = p Batt spbern spbern spbern | rypi:~ rypi:~ rypi:~ rypi:/s = 0x01: tery Th rypi:/s rypi:/s rypi:/s | \$ \$ \$ \$ cd /sys/class/rtc/rtc0/device ys/class/rtc/rtc0/device \$ cat power_m Power Management Auto and Trickle Cha reshold = 2.2V ys/class/rtc/rtc0/device \$ ys/class/rtc/rtc0/device \$ ys/class/rtc/rtc0/device \$ | gmt rger |

Figure 15. Reading power management mode.

b. The **cat trickle_charger** command reads the trickle charger settings as shown in Figure 16. Change the settings in the device tree.



Figure 16. Reading the trickle charger settings.

- c. A sysfs entry allows only a root user to write data.
- sudo -s

Enter the command mentioned above to change the user mode.

pi@raspberrypi:/sys/class/rtc/rtc0/device \$ sudo -s

Figure 17. Command for changing the user mode.

The userram (nvmem) is a binary interface unlike other sysfs entries. So, the **cat** command does not show meaningful data.

• hexdump -C userram

Use this command to read as shown in Figure 18.

```
root@raspberrypi:/sys/devices/platform/soc/3f804000.i2c/i2c-1/1-0069# hexdump -C userram
00000000 c7 cc f7 81 5b 39 49 ab 08 07 c8 fa 53 ad d6 74 |....[9I.....S..t]
00000010 0c 3a 65 ef 0d 13 01 06 3f 19 12 fb 7c 5c 86 44 |.:e....?...|\.D]
00000020 28 7d c5 83 b6 0e 2e bf 15 f7 b9 68 a4 90 dc b1 |(}.....h....|
00000030 ca 42 a0 d7 55 a1 dd 94 ba ef 90 36 4b 16 7a 73 |.B..U.....6K.zs|
00000040
```

Figure 18. Reading RTC userram with hexdump.

The root user access is a must to write data to userram.

echo "<data>" >> userram

The command works after getting access.

echo -n -e "<data in binary format>" >> userram

The command works to write data in binary.

```
pi@raspberrypi:/sys/devices/platform/soc/3f804000.i2c/i2c-1/1-00669 $ sudo su
root@raspberrypi:/sys/devices/platform/soc/3f804000.i2c/i2c-1/1-00669 # echo "Maxim Integrated" >> userram
00000000 4d 61 78 69 6d 20 49 6e 74 65 67 72 61 74 65 64 |Maxim Integrated|
00000010 6a 98 c6 dd 20 44 2f ba 8d 1f 89 c6 40 56 a8 10 |.... D/....@V...|
00000020 38 ec b4 16 55 59 17 61 08 08 38 7d 5f a1 6a 26 |8...UY.a.8}__i&]
00000030 13 b8 24 4d bc b8 ac 00 b8 8a ad ca 78 51 95 34 |..$M.....xQ.4|
00000040
root@raspberrypi:/sys/devices/platform/soc/3f804000.i2c/i2c-1/1-0069# echo -n -e '\x01\x02\x03\x04\x05\x06\x07' >> userram
root@raspberrypi:/sys/devices/platform/soc/3f804000.i2c/i2c-1/1-0069# hexdump -C userram
00000000 01 02 03 04 05 06 07 6e 74 65 67 72 61 74 65 64 |.....ntegrated|
00000010 0a 98 c0 dd 20 44 2f ba 8d 1f 89 c0 40 56 a8 10 |.... D/.....@V...|
00000020 38 ec b4 16 55 59 17 61 08 08 38 7d 5f a1 6a 26 |8...UV.a..8]_.j&|
00000003 13 b8 24 4d bc b8 ac 00 b8 8a ad ca 78 51 95 34 |..$M.....xQ.4|
00000004
root@raspberrypi:/sys/devices/platform/soc/3f804000.i2c/i2c-1/1-0069#
```

Figure 19. Writing data to RTC userram.

d. Use the **cat additional_interrupt** command to see the status of the remaining interrupts not used by the kernel. These are analog power fail, D1 input, loss of external clock, and oscillator failed interrupts.

If no interrupt is asserted:

```
pi@raspberrypi:/sys/class/rtc/rtc0/device $ cat additional_interrupt
This section shows EIF, ANA_IF, OSF and LOS interrupts.
NONE
```

Figure 20. Reading the additional interrupt status.

If interrupts are asserted:

```
This section shows EIF, ANA_IF, OSF and LOS interrupts.
Last Interrupts
LOS Int.: LOS of signal.
Analog Int.: Analog interrupt flag/Power fail flag.
External Int.: External interrupt flag for D1
```

Figure 21. Command output when the interrupts are asserted.

- e. A test program, rtc_ctest, is provided to test the IOCTL options including the userram, update interrupt, alarm interrupt, and sysfs read values. Call the program with security privilege.
- sudo chmod 777 rtc_ctest

Run the command mentioned above to ensure the file is readable, writable, and executable.

• sudo ./rtc_ctest

The command calls the program.

pi@raspberrypi:~ \$ sudo ./Desktop/latest_driver/rtc_test/rtc_ctest/bin/Debug/rtc_ctest Update IRQ Test Started Counting 5 update (1/sec) interrupts from reading /dev/rtc0: 1 2 3 4 5 Again, from using select(2) on /dev/rtc: 1 2 3 4 5 Current RTC date/time is 27-1-2020, 18:14:10. Alarm time now set to 18:14:15. Waiting 5 seconds for alarm... okay. Alarm rang.

 Periodic IRQ rate is 64HZ.

 Counting 20 interrupts at:

 2Hz:
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

 4Hz:
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

 8Hz:
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

 8Hz:
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

 16Hz:
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

 32Hz:
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

 32Hz:
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

 64Hz:
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

 User Ram Write Test Values: 0x68 0x62 0xAF 0xAE 0x89 0xF4 0x71 0xDC 0x04 0x5A 0x84 0x75 0xBD 0xEF 0x5F 0x35 0xA0 0x1C 0x77 0xC9 0x9F 0x60 0x9A 0x2F 0x9A 0x99 0x3D 0x 20 0xBC 0xC1 0x91 0x27 0x23 0x41 0xD5 0xAD 0x35 0x46 0x89 0x39 0xA0 0x0E 0xAE 0x5D 0xFD 0x0E 0x93 0x9E 0x2A 0x0A 0x67 0xC9 0x6A 0x02 0xF8 0x05 0x9B 0x35 0x25 0x57 0xF7 0xB7 0x7E 0x1A User Ram Read Test Values: 0x6B 0x62 0xAF 0xAE 0x89 0xF4 0x71 0xDC 0x04 0x5A 0x84 0x75 0xBD 0xEF 0x5F 0x35 0xA0 0x1C 0x77 0xC9 0x9F 0x60 0x9A 0x2F 0x9A 0x99 0x3D 0x 20 0xBC 0xC1 0x91 0x27 0x23 0x41 0xD5 0xAD 0x35 0x46 0x89 0x39 0xA0 0x0E 0xAE 0x5D 0xFD 0x0E 0x93 0x9E 0x2A 0x0A 0x67 0xC9 0x6A 0x02 0xF8 0x05 0x9B 0x35 0x25 0x57 0xF7 0xB7 0x7E 0x1A User Ram Test PASSED Trickle Charger Value From Device Tree 3k Ohm in series with a Schottky diode Reg Value = 0x01: Power Management Auto and Trickle Charger Backup Battery Threshold = 2.2V *** Test complet

Figure 22. Command output of RTC test script.

- f. Install the necessary packages first to use the test program in Python. Install them with the following commands:
- sudo pip3 install ioctl-opt

Figure 23. Installing dependencies.

• sudo pip3 install pytz



Figure 24. Installing dependencies.

After installing the necessary packages, run the program with the following command:

• sudo python3 rtc_python3Test.py "/dev/rtc0"

| RTC Driver Test Example. |
|---|
| *********IOCTL register read test********* Address: 0x55 Value = 0xAA |
| IOCTL register write test Address: 0x55 Value = 0xAA |
| IOCTL register write/read test SUCCESS!!! |
| **********IOCTL Data Retention Access Test******** |
| IOCTL Data Retention test SUCCESS!!!! |
| ***********IOCTL External Clock Access Test********* |
| IOCTL External Clock test SUCCESS!!!! |
| ***********IOCTL Power Management Access Test********* |
| IOCTL Power Management test SUCCESS!!!! |
| **********IOCTL Alarm2 Test******** |
| IOCTL Alarm2 set for one minute Min: 17 |
| IOCTL Waiting alarm2 |
| IOCTL Alarm2 test SUCCESS!!!! |
| Update IRQ Test Started |
| Counting 5 update (1/sec) interrupts from reading /dev/rtc0: 1 2 3 4 5 |
| Current RTC date/time is 27-02-2020, 12:17:05. |
| Alarm time now set to 12:17:10. |
| Waiting 5 seconds for alarm Okay. Alarm rang, |
| Periodic IRQ rate is 64Hz. |
| Counting 20 interrupts at: |
| 2Hz: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 4Hz: |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 8Hz: |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 16H7: |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 3047' |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 |
| User Ram Write Test Values: |
| 0x22 0x29 0x50 0x09 0x3F 0x85 0x38 0x19 0x6C 0x2A 0x39 0xD3 0x4C 0xC5 0xA3 0x1A 0xFD 0x90 0x75 0x5E 0xA1 0x95 0x34 0xA0 0xD2 0x93 0xE8 0x7D 0x93 0x37 AB 0x69 0x24 0x7B 0x13 0x19 0x98 0xA6 0x23 0x4E 0xD6 0x71 0x31 0x63 0x72 0xFF 0xB5 |
| User Ram Read Test Values: |
| 0x22 0x29 0x50 0x09 0x3F 0x85 0x38 0x19 0x6C 0x2A 0x39 0xD3 0x4C 0xC5 0xA3 0x1A 0xFD 0x90 0x75 0x5E 0xA1 0x95 0x34 0xA0 0xD2 0x93 0xE8 0x7D 0x93 0x37 AB 0x69 0x24 0x7B 0x13 0x19 0x98 0xA6 0x23 0x4E 0xD6 0x71 0x31 0x63 0x72 0xFF 0xB5 |
| User Ram Test PASSED |
| Trickle Charger Value From Device Tree |
| Sk Onm in Series with a Schottky didde |
| Power Management Mode & Threshold |
| neg netter most rener nanagement nato and rizerte ondiger |
| *** Test complete *** |
| pi@raspberrypi:~/Dēsktop \$ |
| |

Figure 25. Command output of RTC Python test script.

4.2 Kernel Compilation Procedure

Another way to compile and run the driver is to compile it with the whole kernel. Follow these instructions to build the kernel in the computer: (Ubuntu is used here to build the kernel. Use **this link** for another system.)

First, download and install toolchain with the following commands:

- git clone https://github.com/raspberrypi/tools ~/tools
- echo PATH=\\$PATH:~/tools/arm-bcm2708/arm-linux-gnueabihf/bin >> ~/.bashrc
- source ~/.bashrc

```
ens@ens-VirtualBox:~$ git clone https://github.com/raspberrypi/tools ~/tools
Cloning into '/home/ens/tools'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 25383 (delta 3), reused 8 (delta 3), pack-reused 25374
Receiving objects: 100% (25383/25383), 610.88 MiB | 2.44 MiB/s, done.
Resolving deltas: 100% (14884/14884), done.
Checking out files: 100% (19059/19059), done.
ens@ens-VirtualBox:~$ source ~/.bashrc
ens@ens-VirtualBox:~$ source ~/.bashrc
ens@ens-VirtualBox:~$
```

Figure 26. Downloading the required tools for compiling the Raspberry Pi kernel.

Install an additional set of libraries for a 32-bit operating system (for example, Raspberry Pi Desktop for the PC):

• sudo apt install zlib1g-dev:amd64

Download the kernel source after these steps:

• git clone --depth=1 https://github.com/raspberrypi/linux

ens@ens-VirtualBox:~\$ git clone --depth=1 https://github.com/raspberrypi/linux

Figure 27. Downloading the Linux kernel for Raspberry Pi.

Ensure the needed dependencies are there on the machine to build the sources for crosscompilation. Execute the following command:

• sudo apt install git bc bison flex libssl-dev make libc6-dev libncurses5-dev



Figure 28. Installing the system dependencies for kernel compiling.

| 〈〉 û Home | linux drivers rtc 🕶 | Q ःः ▼ ≡ - | • 😣 |
|-------------------|---------------------|-------------------------|--------------|
| () Recent | rtc | × rtc | |
| + Starred | Name | ✓ Size | Modified |
| ☆ Home | C rtc-m48t86.c | 7,8 kB | 14:37 |
| Desktop | C rtc-max6900.c | 6,3 kB | 14:37 |
| Documents | C rtc-max6902.c | 3,8 kB | 14:37 |
| Downloads | C rtc-max6916.c | 4,1 kB | 14:37 |
| Pictures | C rtc-max8907.c | 5,2 kB | 14:37 |
| ⊟ Videos | C rtc-max8925.c | 8,2 kB | 14:37 |
| 💼 Trash | C rtc-max8997.c | 12,5 kB | 14:37 |
| + Other Locations | C rtc-max8998.c | 7,8 kB | 14:37 |
| | C rto-max31343.c | 12,5 kB | 14:37 |
| | C rtc-max77686.c | 21,5 kB | 14:37 |
| | C stome13xxx c | "rtc-max31343.c" select | ed (12,5 kB) |

Copy rtc-MAX31343.c in ./linux/drivers/rtc after these steps.

Figure 29. Device driver file located in kernel.

Add the following lines to the drivers/rtc/Kconfig file:

config RTC_DRV_MAX31343

tristate "Maxim MAX31343"

help

If you say yes here you get support for Maxim

MAX31343 RTC chip.

This driver can also be built as a module. If so, the module is called rtc-max31343.

| 1897 witt be catted if tc-aspeed . 1898 comment "HID Sensor RTC drivers" 1990 1991 1991 config RTC_DRV_HID_SENSOR_TIME 1902 tristate "HID Sensor Time" 1903 depends on USB_HID 1904 depends on USB_NENSOR_HUB && IIO 1905 select HID_SENSOR_TIO_COMMON 1906 help 1907 Say yes here to build support for the HID Sensors of type Time. 1908 This drivers makes such sensors available as RTCs. 1909 If this driver is compiled as a module, it will be named 1911 rtc-hid-sensor-time. 1912 tristate "Goldfish Real Time Clock" 1914 tristate "Goldfish Real Time Clock" 1915 depends on OF && HAS_IOMEM 1916 depends on GOLDFISH COMPILE_TEST 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1920 Goldfish is a code name for the virtual platform developed by Google 1921 for Android enulation. 1922 Tor Android enulation. 1923 for you say yes here you get support for Maxim | Open | - n | | Kconfig | l eltre | Save | | | 8 |
|--|------------|---------------|------------------------------|----------------|---|-----------|--------|-----|---------|
| <pre>1998 1999 comment "HID Sensor RTC drivers" 1900 1901 config RTC_DRV_HID_SENSOR_TIME 1902 tristate "HID Sensor Time" 1903 depends on USB_HID 1904 depends on USB_HID 1905 select HID_SENSOR_IIO_COMMON 1906 help 1907 Say yes here to build support for the HID Sensors of type Time. 1908 This drivers makes such sensors available as RTCs. 1909 1910 If this driver is compiled as a module, it will be named 1911 rtc-hid-sensor-time. 1912 1913 config RTC_DRV_GOLDFISH 1914 tristate "Goldfish Real Time Clock" 1915 depends on G& A& HAS_IOMEM 1916 depends on GOLDFISH COMPILE_TEST 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1919 1920 Goldfish is a code name for the virtual platform developed by Google 1921 for Android enulation. 1922 1923 config RTC_DRV_MAX31343 1924 tristate "MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1931 1932 1933 endif # RTC_CLASS </pre> | 1897 | were be a | atteu rtc-aspe | eu . | 20165 | - | | | |
| <pre>1899 comment "HID Sensor RTC drivers" 1900 1901 config RTC_DRV_HID_SENSOR_TIME 1902 tristate "HID Sensor Time" 1903 depends on USB_HID 1904 depends on USB_HID 1905 select HID_SENSOR_IIO_COMMON 1906 help 1907 Say yes here to build support for the HID Sensors of type Time. 1908 This drivers makes such sensors available as RTCs. 1909 1910 If this driver is compiled as a module, it will be named 1911 rtc-hid-sensor-time. 1912 1913 config RTC_DRV_GOLDFISH 1914 tristate "Goldfish Real Time Clock" 1915 depends on G & HAS_IOMEM 1916 depends on GOLDFISH [] COMPILE_TEST 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1920 Goldfish is a code name for the virtual platform developed by Google 1921 for Android emulation. 1922 tristate "MaxII MAX3I343" 1924 tristate "MaxII MAX3I343" 1924 tristate "MaxII MAX3I343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX3I343 RTC chip. 1929 This driver can also be built as a module. If so, the module 1930 will be called rtc-max3I343. 1933 endif # RTC_CLASS</pre> | 1898 | | | | | | | | |
| <pre>1900 1901 config RTC_DRV_HID_SENSOR_TIME 1902 tristate "HID Sensor Time" 1903 depends on USB_HID 1904 depends on HID_SENSOR_HUB && IIO 1905 select HID_SENSOR_IIO_COMMON 1906 help 1907 Say yes here to build support for the HID Sensors of type Time. 1908 This drivers makes such sensors available as RTCs. 1909 If this driver is compiled as a module, it will be named 1911 rtc-hid-sensor-time. 1912 If config RTC_DRV_GOLDFISH 1914 tristate "Goldfish Real Time Clock" 1915 depends on GF && HAS_IOMEM 1916 depends on GF && HAS_IOMEM 1917 say yes to enable RTC driver for the Goldfish based virtual platform. 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1920 Goldfish is a code name for the virtual platform developed by Google 1921 for Android enulation. 1922 1923 config RTC_DRV_MAX31343" 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 This driver can also be built as a module. If so, the module 1931 1932 1933 endif # RTC_CLASS</pre> | 1899 comme | nt "HID Senso | r RTC drivers" | | | | | | |
| <pre>1901 config RTC_DRV_HID_SEMSOR_TIME 1902 tristate "HID Sensor Time" 1903 depends on USB_HID 1904 depends on HID_SENSOR_HUB && IIO 1905 select HID_SENSOR_IIO_COMMON 1906 help 1907 Say yes here to build support for the HID Sensors of type Time. 1908 This drivers makes such sensors available as RTCs. 1909 1910 If this driver is compiled as a module, it will be named 1911 rtc-hid-sensor-time. 1912 1913 config RTC_DRV_GOLDFISH 1914 tristate "Goldfish Real Time Clock" 1915 depends on GF && HAS_IONEM 1916 depends on GOLDFISH [] COMPILE_TEST 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1919 1920 Goldfish is a code name for the virtual platform developed by Google 1921 for Android emulation. 1922 1923 config RTC_DRV_MAX31343 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 1929 This driver can also be built as a module. If so, the module 1931 1932 1933 endif # RTC_CLASS</pre> | 1900 | | | | | | | | |
| 1902 tristate "HID Sensor Time" 1903 depends on USB_HID 1904 depends on USB_HID 1905 select HID_SENSOR_HUB && IIO 1906 help 1907 Say yes here to build support for the HID Sensors of type Time. 1908 This drivers makes such sensors available as RTCs. 1909 If this driver is compiled as a module, it will be named 1911 rtc-hid-sensor-time. 1912 1913 1914 tristate "Goldfish Real Time Clock" 1915 depends on GLDFISH 1916 depends on GLDFISH 1916 depends on GLDFISH 1916 depends on GLDFISH COMPILE_TEST 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1919 Goldfish is a code name for the virtual platform developed by Google 1921 for Android enulation. 1922 for Android enulation. 1923 for Jou say yes here you get support for Maxim 1924 If you say yes here you get support for Maxim 1925 help 1926 If you say yes here you get support for | 1901 confi | g RTC_DRV_HID | _SENSOR_TIME | | | | | | |
| 1993 depends on USB_HID 1994 depends on HID_SENSOR_HUB && IIO 1995 select HID_SENSOR_IIO_COMMON 1906 help 1907 Say yes here to build support for the HID Sensors of type Time. 1908 This drivers makes such sensors available as RTCs. 1909 If this driver is compiled as a module, it will be named 1911 rtc-htd-sensor-time. 1912 Tristate "Goldfish Real Time Clock" 1914 tristate "Goldfish Real Time Clock" 1915 depends on OF && HAS_IOMEM 1916 depends on GOLDFISH [] COMPILE_TEST 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1919 Goldfish is a code name for the virtual platform developed by Google 1921 for Android enulation. 1922 Goldfish MAX31343" 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. <t< td=""><td>1902</td><td>tristate "H</td><td>ID Sensor Time"</td><td>C</td><td></td><td></td><td></td><td></td><td></td></t<> | 1902 | tristate "H | ID Sensor Time" | C | | | | | |
| 1994 depends on HID_SENSOR_HUB && IIO 1995 select HID_SENSOR_IIO_COMMON 1996 help 1997 Say yes here to build support for the HID Sensors of type Time. 1998 This drivers makes such sensors available as RTCs. 1999 If this driver is compiled as a module, it will be named 1911 rtc-hid-sensor-time. 1912 1913 config RTC_DRV_GOLDFISH 1914 tristate "Goldfish Real Time Clock" 1915 depends on G G& HAS_IONEM 1916 depends on GOLDFISH COMPILE_TEST 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1920 Goldfish is a code name for the virtual platform developed by Google 1921 for Android emulation. 1922 1923 config RTC_DRV_MAX31343 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. 1932 1932 | 1903 | depends on | USB_HID | | | | | | |
| 1995 select HID_SENSOR_IIO_COMMON 1996 help 1997 Say yes here to build support for the HID Sensors of type Time. 1998 This drivers makes such sensors available as RTCs. 1999 If this driver is compiled as a module, it will be named 1911 rtc-hid-sensor-time. 1912 It chid sensor-time. 1913 Say (GLDFISH 1914 tristate "Coldfish Real Time Clock" 1915 depends on OF && HAS_IOMEM 1916 depends on GOLDFISH COMPILE_TEST 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1919 Goldfish is a code name for the virtual platform developed by Google 1921 for Android enulation. 1922 for Android enulation. 1923 for you say yes here you get support for Maxim 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. | 1904 | depends on | HID_SENSOR_HUB | && IIO | | | | | |
| <pre>1906 help 1907 Say yes here to build support for the HID Sensors of type Time. 1908 This drivers makes such sensors available as RTCs. 1909 1910 If this driver is compiled as a module, it will be named 1911 rtc-hid-sensor-time. 1912 1913 config RTC_DRV_GOLDFISH 1914 tristate "Goldfish Real Time Clock" 1915 depends on GOLDFISH 1916 depends on GOLDFISH [] COMPILE_TEST 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1919 1920 Goldfish is a code name for the virtual platform developed by Google 1921 for Android emulation. 1922 1923 config RTC_DRV_MAX31343 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 1929 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. 1931 1932 endif # RTC_CLASS</pre> | 1905 | select HID | SENSOR_IIO_COMM | ION | | | | | |
| 1997 Say yes here to bull support for the HID Sensors of type Time. 1998 This drivers makes such sensors available as RTCs. 1909 If this driver is compiled as a module, it will be named 1911 rtc-hid-sensor-time. 1912 1913 config RTC_DRV_GOLDFISH 1914 tristate "Goldfish Real Time Clock" 1915 depends on OF && HAS_IONEM 1916 depends on GOLDFISH [] COMPILE_TEST 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1919 1920 1921 for Android emulation. 1922 1923 config RTC_DRV_MAX31343 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. 1931 1932 1932 1933 endif # RTC_CLASS | 1906 | help | | | | | | | |
| 1908 This drivers makes such sensors available as RTCs. 1909 If this driver is compiled as a module, it will be named 1911 rtc-htd-sensor-time. 1912 1913 1914 tristate "Goldfish Real Time Clock" 1915 depends on OF && HAS_IOMEM 1916 depends on GOLDFISH 1917 depends on OF && HAS_IOMEM 1916 depends on GOLDFISH COMPILE_TEST 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1920 Goldfish is a code name for the virtual platform developed by Google 1921 for Android enulation. 1922 1923 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 1 1930 will be called rtc-max31343. 1931 1 1932 1 1933 endif # RTC_CLASS | 1907 | Say yes h | nere to build su | upport for the | e HID Sensors of t | ype Time. | | | |
| <pre>1999 1910 If this driver is compiled as a module, it will be named 1911 rtc-hid-sensor-time. 1912 1913 config RTC_DRV_GOLDFISH 1914 tristate "Goldfish Real Time Clock" 1915 depends on GF && HAS_IOMEM 1916 depends on GOLDFISH COMPILE_TEST 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1919 1920 Goldfish is a code name for the virtual platform developed by Google 1921 for Android emulation. 1922 config RTC_DRV_MAX31343 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 1929 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. 1931 1932 1933 endif # RTC_CLASS</pre> | 1908 | This driv | vers makes such | sensors avai | lable as RTCs. | | | | |
| 1910 If this driver is compiled as a module, it will be named 1911 rtc-hid-sensor-time. 1912 1913 config RTC_DRV_GOLDFISH 1914 tristate "Goldfish Real Time Clock" 1915 depends on OF && HAS_IONEM 1916 depends on GOLDFISH COMPILE_TEST 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1920 Goldfish is a code name for the virtual platform developed by Google 1921 for Android enulation. 1922 1923 config RTC_DRV_MAX31343 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 1929 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. 1931 1932 1933 endif # RTC_CLASS | 1909 | | | | | | | | |
| <pre>1911 rtc-hid-sensor-time. 1912 1912 1913 config RTC_DRV_GOLDFISH 1914 tristate "Goldfish Real Time Clock" 1915 depends on GO & AAS_IONEM 1916 depends on GOLDFISH COMPILE_TEST 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1919 1920 Goldfish is a code name for the virtual platform developed by Google 1921 for Android emulation. 1922 1923 config RTC_DRV_MAX31343 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 1929 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. 1931 1932 endif # RTC_CLASS</pre> | 1910 | If this o | friver is compil | ed as a modu | le, it will be nam | ed | | | |
| 1912 1913 config RTC_DRV_GOLDFISH 1914 tristate "Goldfish Real Time Clock" 1915 depends on GOLDFISH [] COMPILE_TEST 1916 depends on GOLDFISH [] COMPILE_TEST 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1919 0 1920 Goldfish is a code name for the virtual platform developed by Google 1921 for Android emulation. 1922 1923 config RTC_DRV_MAX31343 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 1 1929 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. 1931 1 1932 1 1933 endif # RTC_CLASS | 1911 | rtc-hid-s | sensor-time. | | | | | | |
| 1913 config RTC_DRV_GOLDFISH 1914 tristate "Goldfish Real Time Clock" 1915 depends on OF && HAS_IOMEM 1916 depends on GOLDFISH COMPILE_TEST 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1919 Goldfish is a code name for the virtual platform developed by Google 1921 for Android enulation. 1922 1923 config RTC_DRV_MAX31343 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. 1931 1932 1933 endif # RTC_CLASS | 1912 | | | | | | | | |
| 1914 tristate "Goldfish Real Time Clock" 1915 depends on OF && HAS_IONEM 1916 depends on GOLDFISH COMPILE_TEST 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1919 1920 1921 for Android emulation. 1922 1923 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 11 1929 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. 1931 1932 1933 endif # RTC_CLASS | 1913 confi | g RTC_DRV_GOL | DFISH | | | | | | |
| 1915 depends on OF && HAS_IONEM 1916 depends on GOLDFISH [] COMPILE_TEST 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1919 1920 1920 Goldfish is a code name for the virtual platform developed by Google 1921 for Android emulation. 1922 1923 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 1929 1930 will be called rtc-max31343. 1931 1932 1932 1933 endif # RTC_CLASS | 1914 | tristate "(| Goldfish Real Ti | me Clock" | | | | | |
| 1916 depends on GOLDFISH COMPILE_TEST 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1919 1920 1921 for Android emulation. 1922 1923 1923 config RTC_DRV_MAX31343 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 1929 1930 will be called rtc-max31343. 1931 1932 1933 endif # RTC_CLASS | 1915 | depends on | OF && HAS_IOMEM | 1 | | | | | |
| 1917 help 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1919 1920 1920 Goldfish is a code name for the virtual platform developed by Google 1921 for Android emulation. 1922 1923 config RTC_DRV_MAX31343 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 1 1929 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. 1931 1 1932 1 1933 endif # RTC_CLASS | 1916 | depends on | GOLDFISH COM | PILE_TEST | | | | | |
| 1918 Say yes to enable RTC driver for the Goldfish based virtual platform. 1919 Goldfish is a code name for the virtual platform developed by Google 1921 for Android enulation. 1922 1923 config RTC_DRV_MAX31343 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 1929 1930 will be called rtc-max31343. 1931 1932 1933 endif # RTC_CLASS | 1917 | help | | | | | | | |
| 1919 1920 Goldfish is a code name for the virtual platform developed by Google 1921 for Android emulation. 1922 1923 config RTC_DRV_MAX31343 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 1929 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. 1931 1932 1933 endif # RTC_CLASS | 1918 | Say yes t | to enable RTC dr | iver for the | Goldfish based vi | rtual pla | atform | - | |
| 1920 Goldfish is a code name for the virtual platform developed by Google 1921 for Android emulation. 1922 1923 config RTC_DRV_MAX31343 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 1929 1930 will be called rtc-max31343. 1931 1932 1933 endif # RTC_CLASS | 1919 | | | | | | | | |
| 1921 for Android enulation. 1922 1923 1923 config RTC_DRV_MAX31343 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 1929 1930 will be called rtc-max31343. 1931 1932 1933 endif # RTC_CLASS | 1920 | Goldfish | is a code name | for the virte | ual platform devel | oped by C | Google | | |
| 1922 1923 config RTC_DRV_MAX31343 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 1929 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. 1931 1932 1933 endif # RTC_CLASS | 1921 | for Andro | oid emulation. | | | | | | |
| 1923 config RIC_DRV_MAX31343 1924 tristate "Maxim MAX31343" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 1929 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. 1931 1932 1933 endif # RTC_CLASS | 1922 | | | | | | | | |
| 1924 tristate "Maxim Maxifas" 1925 help 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 1929 1929 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. 1931 1932 1933 endif # RTC_CLASS | 1923 CONTL | g RIC_DRV_MAX | 31343 | | | | | | |
| 1925 netp 1926 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 1929 1929 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. 1931 1932 1933 endif # RTC_CLASS | 1924 | tristate " | Maxim MAX31343" | | | | | | |
| 1920 If you say yes here you get support for Maxim 1927 MAX31343 RTC chip. 1928 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. 1931 1932 1933 endif # RTC_CLASS | 1925 | help | and the second second second | | er an | | | | |
| 1927 MAX31343 RTL Chip. 1928 1929 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. 1931 1932 1933 endif # RTC_CLASS | 1926 | IT you sa | ay yes here you | get support | TOR MAXIM | | | | |
| 1928 1929 This driver can also be built as a module. If so, the module 1930 will be called rtc-max31343. 1931 1932 1933 endif # RTC_CLASS | 1927 | MAX31343 | RIC Chip. | | | | | | |
| 1929 Ints driver can also be built as a module. It so, the module 1930 will be called rtc-max31343. 1931 1932 1933 endif # RTC_CLASS | 1928 | - | | | 1.3 | | | | |
| 1930 Will be called rtc-max31343. 1931 1932 1933 endif # RTC_CLASS | 1929 | Ints driv | ver can also be | DULLE as a M | bdule. IT so, the | module | | | |
| 1932 1932 1933 endif # RTC_CLASS | 1930 | will be o | called rtc-max31 | 1343. | | | | | |
| 1933 endif # RTC_CLASS | 1931 | | | | | | | | |
| 1955 ender # RIC_CLASS | 1932 and f | # DTC CLASS | | | | | | | |
| | 1933 end() | # RIC_CLASS | | | and the second second second second | | | 1.2 | 100.000 |

Figure 30. Kconfig file after modifications.

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

obj-\$(CONFIG_RTC_DRV_MAX31343) += rtc-max31343.o

| Open 🔻 利 | Makefile -/linux/drivers/rtc | | Save | | | | 8 |
|---|---------------------------------|---------------|-------|-----------|---|---|------|
| 154 obj-\$(CONFIG_RTC_DRV_S35390A) | += rtc-s35390a.o | | | | | | |
| 155 obj-\$(CONFIG_RTC_DRV_S3C) | += rtc-s3c.o | | | | | | |
| 156 obj-\$(CONFIG_RTC_DRV_S5M) | += rtc-s5m.o | | | | | | |
| 157 obj-\$(CONFIG_RTC_DRV_SA1100) | += rtc-sa1100.0 | | | | | | |
| 158 obj-\$(CONFIG_RTC_DRV_SC27XX) | += rtc-sc27xx.o | | | | | | |
| 159 obj-\$(CONFIG_RTC_DRV_SD3078) | += rtc-sd3078.0 | | | | | | |
| 160 obj-\$(CONFIG_RTC_DRV_SH) | += rtc-sh.o | | | | | | |
| 161 obj-\$(CONFIG_RTC_DRV_SIRFSOC) | += rtc-sirfsoc.o | | | | | | |
| 162 obj-\$(CONFIG_RTC_DRV_SNVS) | += rtc-snvs.o | | | | | | |
| <pre>163 obj-\$(CONFIG_RTC_DRV_SPEAR)</pre> | += rtc-spear.o | | | | | | |
| 164 obj-\$(CONFIG_RTC_DRV_STARFIRE) | += rtc-starfire.o | | | | | | |
| 165 obj-\$(CONFIG_RTC_DRV_STK17TA8) | += rtc-stk17ta8.0 | | | | | | |
| 166 obj-\$(CONFIG_RTC_DRV_ST_LPC) | += rtc-st-lpc.o | | | | | | |
| 167 obj-\$(CONFIG_RTC_DRV_STM32) | += rtc-stm32.0 | | | | | | |
| 168 obj-\$(CONFIG_RTC_DRV_STMP) | += rtc-stmp3xxx.o | | | | | | |
| 169 obj-\$(CONFIG_RTC_DRV_SUN4V) | += rtc-sun4v.o | | | | | | |
| 170 obj-\$(CONFIG_RTC_DRV_SUN6I) | += rtc-sun6i.o | | | | | | |
| 171 obj-\$(CONFIG_RTC_DRV_SUNXI) | += rtc-sunxi.o | | | | | | |
| 172 obj-\$(CONFIG_RTC_DRV_TEGRA) | += rtc-tegra.o | | | | | | |
| 173 obj-\$(CONFIG_RTC_DRV_TEST) | += rtc-test.o | | | | | | |
| 174 obj-\$(CONFIG_RTC_DRV_TPS6586X) | += rtc-tps6586x.o | | | | | | |
| 175 obj-\$(CONFIG RTC_DRV_TPS65910) | += rtc-tps65910.0 | | | | | | |
| 176 obj-S(CONFIG RTC DRV TPS80031) | += rtc-tps80031.0 | | | | | | |
| 177 obj-S(CONFIG RTC DRV TWL4030) | += rtc-twl.o | | | | | | |
| 178 obj-S(CONFIG RTC DRV TX4939) | += rtc-tx4939.0 | | | | | | |
| 179 obj-S(CONFIG RTC DRV V3020) | += rtc-v3020.0 | | | | | | |
| 180 obj-S(CONFIG RTC DRV VR41XX) | += rtc-vr41xx.o | | | | | | |
| 181 obj-S(CONFIG RTC DRV VRTC) | += rtc-mrst.o | | | | | | |
| 182 obj-S(CONFIG RTC DRV VT8500) | += rtc-vt8500.0 | | | | | | |
| 183 obj-S(CONFIG RTC DRV WILCO EC) | += rtc-wilco-ec.o | | | | | | - 14 |
| 184 obj-S(CONFIG RTC DRV WM831X) | += rtc-wm831x.o | | | | | | |
| 185 obj-S(CONFIG RTC DRV WM8350) | += rtc-wm8350.0 | | | | | | |
| 186 obj-S(CONFIG RTC DRV X1205) | += rtc-x1205.0 | | | | | | - 1 |
| 187 obj-S(CONFIG RTC DRV XGENE) | += rtc-xgene.o | | | | | | - 1 |
| 188 obj-S(CONFIG RTC DRV ZYNOMP) | += rtc-zynamp.o | | | | | | |
| 189 obj-S(CONFIG RTC DRV MAX31343) | += rtc-max31343.0 | | | | | | - 1 |
| | Makefile - Ti | ab Width: 8 🔻 | Ln 18 | 39, Col 5 | 0 | * | INS |

Figure 31. Adding the device driver to Makefile of RTC drivers.

Ensure the Maxim MAX31343 option is selected within the kernel configuration before building the kernel. Use menuconfig to configure the kernel. (Ubuntu is used here to build the kernel for Raspberry Pi 3B+. Use **this link** for another system.)

- KERNEL=kernel7
- make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bcm2709_defconfig
- make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig



Figure 32. Configuring the kernel before compiling.

Follow these figures:



Figure 33. Selecting Device Drivers menu.



Figure 34. Selecting Real Time Clock menu.

Figure 35. Enabling the Maxim MAX31343 device driver for compilation.

| | ens@ens-VirtualBox: ~/linux | |
|----------------|--|--|
| File Edit Viev | v Search Terminal Help | |
| .config - L | inux/arm 4.19.108 Kernel Configuration | |
| | Enter a filename to which this configuration should be saved as an alternate. Leave blank to abort. .config | |
| | | |

Figure 36. Saving the configuration file.

After the steps mentioned above, compile the kernel with the following commands:

- cd linux
- KERNEL=kernel7
- make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bcm2709_defconfig
- make ARCH

```
ens@ens-VirtualBox:~$ cd linux
ens@ens-VirtualBox:~/linux$ KERNEL=kernel7
ens@ens-VirtualBox:~/linux$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bcm2709_defconfig
#
# configuration written to .config
#
ens@ens-VirtualBox:~/linux$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage modules dtbs
Figure 37. Compiling the kernel
```

References

- https://sysplay.github.io/books/LinuxDrivers/book/
- http://robbie-cao.github.io/2016/09/device-tree
- https://www.jameco.com/Jameco/workshop/circuitnotes/raspberry-pi-circuit-note.html
- https://stackoverflow.com/questions/40529308/linux-driver-ioctl-or-sysfs
- https://www.raspberrypi.org/documentation/linux/kernel/building.md
- https://www.man7.org/linux/man-pages/man4/rtc.4.html
- https://www.kernel.org/doc/html/latest/admin-guide/rtc.html#new-portable-rtc-classdrivers-dev-rtcn

Revision History

| REVISION NUMBER | REVISION DATE | DESCRIPTION | PAGES CHANGED |
|--------------------|------------------|--|------------------|
| 0 | 06/21 | Initial release | _ |
| 1 | 12/21 | Updated the link for the rtc-max31343.c, removal of the i2c- rtc-overlay.dts link | 11,12 |
| | | | |
| | | | |
| | | | |

Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use, nor for any infringements of patents or other rights of third parties that may result from its use. Specifications subject to change without notice. No license is granted by implication or otherwise under any patent or patent rights of Analog Devices. Trademarks and registered trademarks are the property of their respective owners.