

Lab9 - Beagle Bone Black i2c communication and clock

Leonardo Fusser (1946995)

Objectives:

- Wiring the RTC to BBB.
- Use of i2c-dev module.
- Use i2c tools.
- Use hardware clock and system clock (`hwclock` command).

Hardware: BBB, ds1307 RTC board, miscellanea.

To hand in: Answer to the lab's questions. Screenshots and the code related to them in a word document.

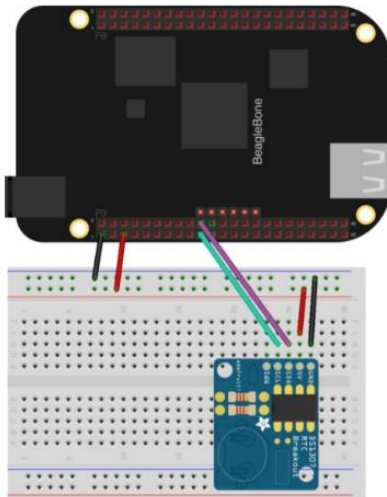
Preparation:

Read and understand the i2c and clock commands:

i2cget / i2cset / i2cdetect / i2cdump / hwclock

Part 1: Prototyping the RTC module to the BBB.

See pdf document or on internet (adafruit): [adding-a-real-time-clock-to-beaglebone-black.pdf](#) (p1 to p7).



Wiring:

- 1- VCC on the breakout board to the P9_5 (VCC 5V) or P9_7 (SYS 5V) pin on the BBB. NOTE: The P9_5 VCC 5V pin will only be powered if a 5V adapter is plugged in to the barrel jack. If powering over USB use the P9_7 (SYS 5V) pin instead!
- 2-Connect GND on the breakout board to the P9_1 (GND) pin on the BBB.
- 3-Connect SDA on the breakout board to the P9_20 pin of the BBB.
- 4-Connect SCL on the breakout board to the P9_19 pin of the BB.

The external pull-up resistor must be enabled by connecting the 3 yellow jumpers in enable position.

Part 2: i2c-tools utility.

The I2C bus allows multiple devices to be connected to your board, each with a unique address. It is very useful to be able to see which devices are connected to your board as a way of making sure everything is working. The i2c-tools utility provides 4 commands: `i2cdetect`, `i2cget`, `i2cset` and `i2cdump`. See class notes.

1. Type the following command to see the options:

```
root@beaglebone:~$ sudo i2cdetect
```

Note: If it does not work, you might have to install the tools:

```
root@beaglebone:~$ sudo apt-get install i2c-tools
```

Read about `i2cdetect` and options:

<https://www.systutorials.com/docs/linux/man/8-i2cdetect/>

2. Type the following command to see the list (-l) of available i2c bus adapters:

```
root@beaglebone:~$ sudo i2cdetect -l
```

```
COM6 - PuTTY
root@beaglebone:~# sudo i2cdetect -l
i2c-0    i2c          OMAP I2C adapter      I2C adapter
i2c-1    i2c          OMAP I2C adapter      I2C adapter
root@beaglebone:~#
```

i2c devices present on the BBB shown above.

3. Type the following command to see the functionality (-F) of bus adapter #1:

```
root@beaglebone:~$ sudo i2cdetect -F 1
```

```
root@beaglebone:~# i2cdetect -F 1
Functionalities implemented by /dev/i2c-1:
I2C                                     yes
SMBus Quick Command                   no
SMBus Send Byte                       yes
SMBus Receive Byte                   yes
SMBus Write Byte                     yes
SMBus Read Byte                      yes
SMBus Write Word                     yes
SMBus Read Word                      yes
SMBus Process Call                   yes
SMBus Block Write                    yes
SMBus Block Read                     no
SMBus Block Process Call             no
SMBus PEC                            yes
I2C Block Write                      yes
I2C Block Read                       yes
root@beaglebone:~#
```

Functionality available for bus adapter #1 on the BBB shown above.

4. Type the following command to probe (or read -r) the bus to see all the connected devices:

Option -y is to answer yes to questions.

```
root@beaglebone:~$ sudo i2cdetect -y -r 1
```

-- means that it was scanned.

UU means that it was skipped because the address is already allocated.

The hex number(s) that appear(s), if any, are the I2C addresses in use.

What are the addresses in use? **0x68**

```
COM6 - PuTTY
root@beaglebone:~# i2cdetect -y -r 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- --  UU  UU  UU  UU -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- --  68 -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
root@beaglebone:~#
```

All connected and available devices on BBB shown above.

To read the content of an i2c device connected to the bus, use the **i2cdump** command.

5. Dump the content of the module to the console. **Take a screenshot.**

```
COM6 - PuTTY
root@beaglebone:~# i2cdump -y 1 0x68 b
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f      0123456789abcdef
00: 28 04 15 03 09 12 20 13 00 00 00 00 00 00 00 00  (????? ?.....
10: 00 00 00 00 00 00 01 01 00 00 00 00 00 00 00 00  .....??.....
20: 00 00 00 00 00 00 00 00 00 00 00 00 02 00 00 00  .....?...
30: 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ?.....
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
root@beaglebone:~#
```

Content of i2c device 1 shown above.

6. In your report, circle individual values of the date: **year**, **month**, **day**, **hrs**, **min**, **sec**.

You can also read/write individual register using **i2cget** and **i2cset**.

7. Read the value of register of the “week day” using **i2cget**: **i2cget -y 1 0x68 0x03**

```
COM6 - PuTTY
root@beaglebone:~# i2cget -y 1 0x68 0x03
0x03
root@beaglebone:~#
```

*Reading the value of “week day” shown above using **i2cget**. Week day is “3”.*

8. Set the current time using `i2cset`.

```
COM6 - PuTTY
root@beaglebone:~# i2cset -y 1 0x68 0x00 0x00
root@beaglebone:~# i2cget -y 1 0x68 0x00
0x04
root@beaglebone:~# i2cset -y 1 0x68 0x01 0x18
root@beaglebone:~# i2cget -y 1 0x68 0x01
0x18
root@beaglebone:~# i2cset -y 1 0x68 0x02 0x03
root@beaglebone:~# i2cget -y 1 0x68 0x02
0x03
root@beaglebone:~#
```

Setting the current time using `i2cset` shown above. Time being set is 3:18:00 AM (was supposed to be 15:18:00 PM)

```
COM6 - PuTTY
root@beaglebone:~# i2cdump -y 1 0x68 b
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f      0123456789abcdef
00: 50 20 03 03 09 12 20 13 00 00 00 00 00 00 00 00  P  ??? ? .....
10: 00 00 00 00 00 00 01 01 00 00 00 00 00 00 00 00  .....?? .....
20: 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 00  .....?....
30: 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ?.....
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
root@beaglebone:~#
```

Time shown in 0, 1, 2 columns (secs, mins, hours) shown above using `i2cdump`.

Part 3: hardware clock and system clock.

In this last part, we are going to use the standard `hwclock` command to read and/or set the RTC.

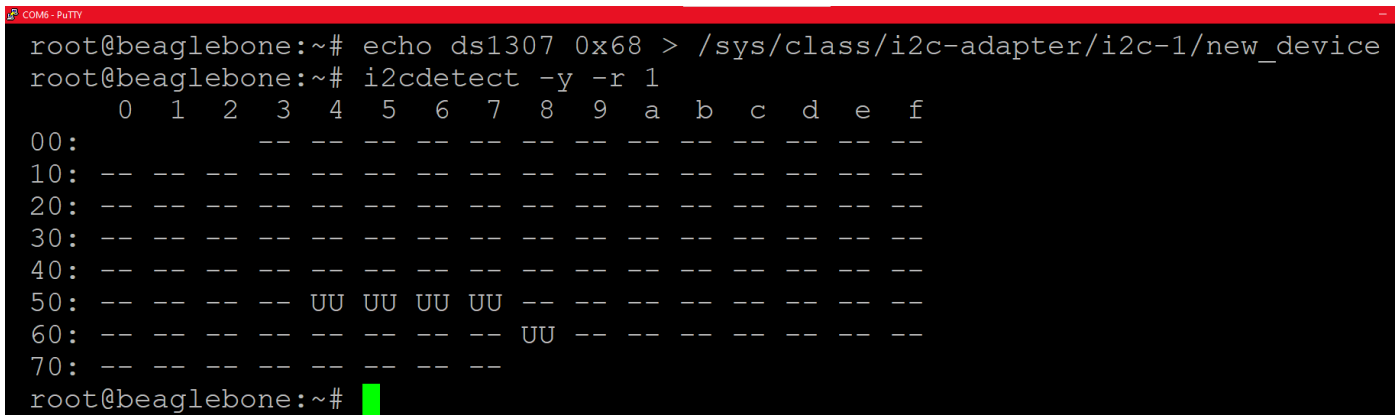
1. Linking the hwclock to the RTC board:

First, in super user mode type the following line to set up the module:

```
root@beaglebone:~# echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
```

Secondly, check the i2c devices again like from before:

```
root@beaglebone:~# i2cdetect -y -r 1
```



```
root@beaglebone:~# echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
root@beaglebone:~# i2cdetect -y -r 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  UU  UU  UU  UU  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  UU  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
root@beaglebone:~#
```

i2cdetect command result after linking hwclock to the RTC board shown above.

Explain the changes in the result: `0x68` has changed to `UU`. It is now part of the system.

2. If the board is not connected to the internet, then set the system clock manually.
Afterwards, set the hardware clock to the system clock.

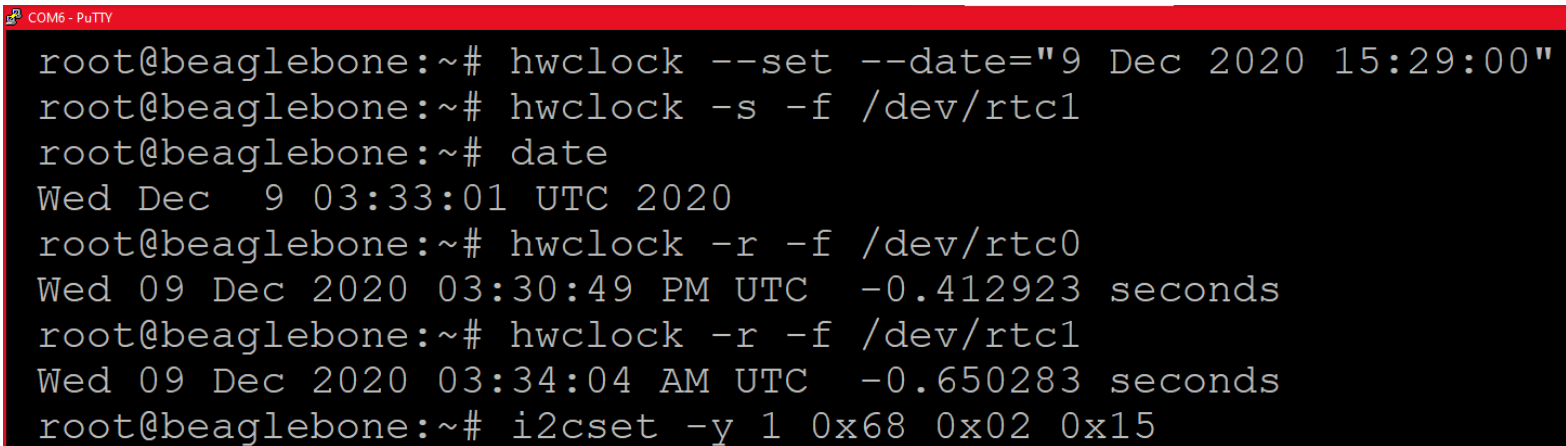
```
root@beaglebone:~# hwclock --set --date="29 Nov 2020 20:12:00"  
root@beaglebone:~# hwclock -s -f /dev/rtc1
```

3. Explain the results after entering the following commands:

```
root@beaglebone:~# date: shows system date and time.
```

```
root@beaglebone:~# hwclock -r -f /dev/rtc0: shows RTC0 date and time.
```

```
root@beaglebone:~# hwclock -r -f /dev/rtc1: shows RTC1 date and time.
```

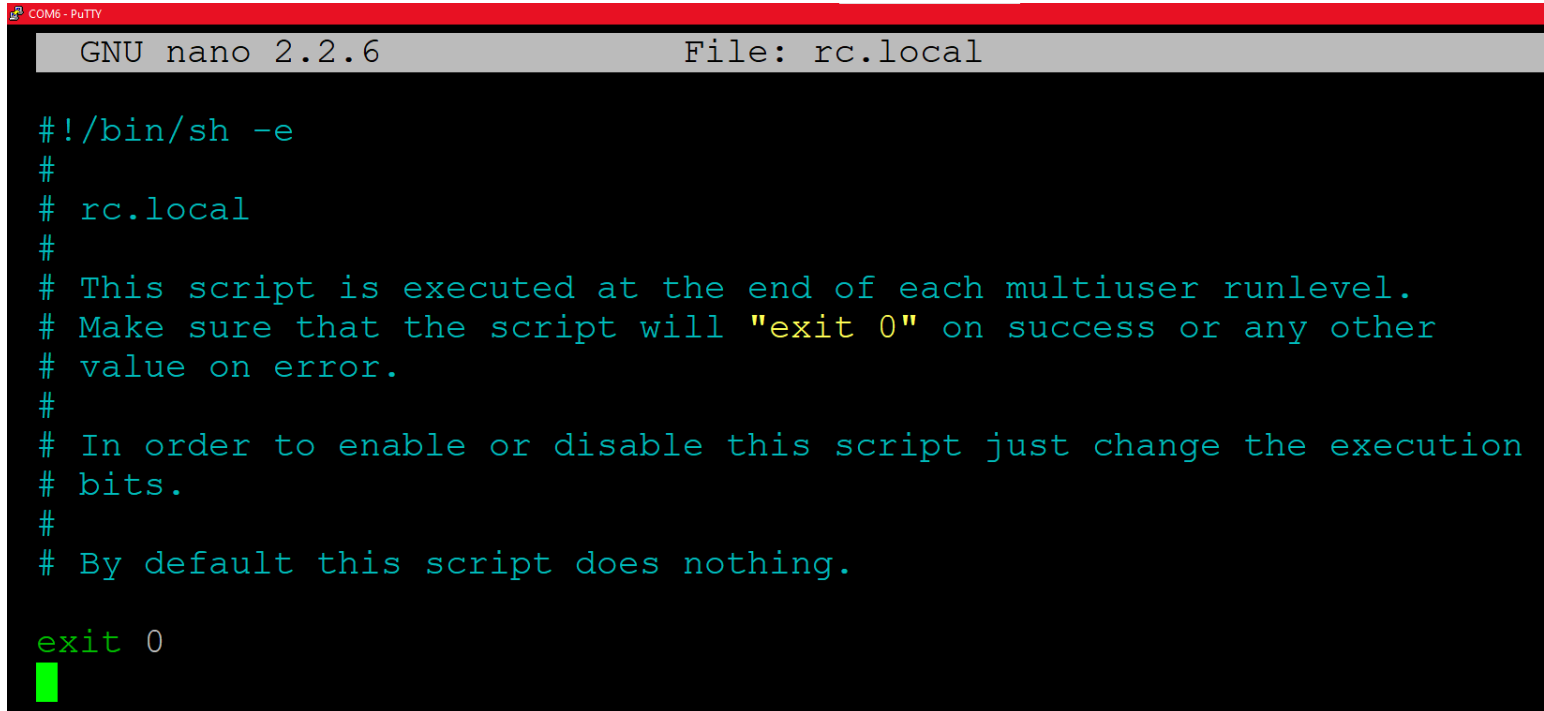


```
root@beaglebone:~# hwclock --set --date="9 Dec 2020 15:29:00"  
root@beaglebone:~# hwclock -s -f /dev/rtc1  
root@beaglebone:~# date  
Wed Dec 9 03:33:01 UTC 2020  
root@beaglebone:~# hwclock -r -f /dev/rtc0  
Wed 09 Dec 2020 03:30:49 PM UTC -0.412923 seconds  
root@beaglebone:~# hwclock -r -f /dev/rtc1  
Wed 09 Dec 2020 03:34:04 AM UTC -0.650283 seconds  
root@beaglebone:~# i2cset -y 1 0x68 0x02 0x15
```

Date and time for RTC0 and RTC1 shown above.

Part 4: hardware clock and system clock.

1. Read and explain `rc.local` script file:

A screenshot of a terminal window with a red title bar that says "COM6 - PuTTY". The terminal shows the GNU nano 2.2.6 editor editing the file rc.local. The script content is as follows:

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

exit 0
```

rc.local script file is used to start a custom service once all normal system services are started. Shown above is rc.local script file located in /etc/ on BBB.

2. Add these 3 lines to `rc.local` script file:

```
sleep 15
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
hwclock --hctosys -f /dev/rtc1
```

3. restart your system and check the date

```
root@beaglebone:~# date
```

```
root@beaglebone:~# hwclock -r -f /dev/rtc0
```

```
root@beaglebone:~# hwclock -r -f /dev/rtc1
```

```
COM6 - PuTTY
root@beaglebone:~# date
Wed Dec  9 16:56:52 UTC 2020
root@beaglebone:~# hwclock -r -f /dev/rtc0
Thu 12 Nov 2015 07:15:13 PM UTC -0.243062 seconds
root@beaglebone:~# hwclock -r -f /dev/rtc1
Wed 09 Dec 2020 04:57:04 PM UTC -0.623012 seconds
root@beaglebone:~#
```

Results before shutting down the BBB shown above.

```
COM6 - PuTTY
root@beaglebone:~# date
Wed Dec  9 16:59:26 UTC 2020
root@beaglebone:~# hwclock -r -f /dev/rtc0
Thu 12 Nov 2015 07:01:12 PM UTC -0.452590 seconds
root@beaglebone:~# hwclock -r -f /dev/rtc1
Wed 09 Dec 2020 04:59:34 PM UTC -0.315413 seconds
root@beaglebone:~#
```

Results after the BBB has started up again. RTC is working!

Note: you can set the clock with the following commands:

```
root@beaglebone:~# date --set="30 Nov 2020 18:00:00"
root@beaglebone:~# hwclock --systohc -f /dev/rtc1
```