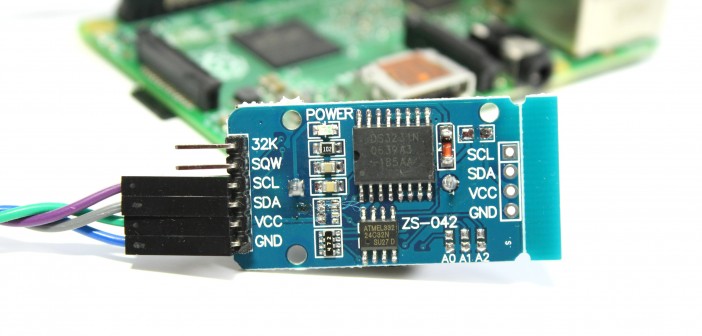
**I. Часы реального времени (Real Time Clock)**

**1. Подключение и установка зависимостей**

**1.1. Аппаратное подключение**

Подключаем устройство в соответствии с таблицей 1.

**Таблица 1.** Подключение RTC-модуля

и RaspBerry Pi\*

|  |  |
| --- | --- |
| **RTC-модуль DS3231** | Pi GPIO |
| GND | P1-06 (GND) |
| VCC | P1-01 (3.3V) |
| SDA | P1-03 (I2C SDA) |
| SCL | P1-05 (I2C SCL) |

**Рисунок 1. RTC-модуль DS3231**

***Примечание*:** на момент написания курсовой все, вышедшие миникомпьютеры (RPi 1,2,3 и Zero), имели одинаковую распиновку (pinout). То есть описанное подключение RTC-модуля подходит для любой из плат.

**1.2. Установка зависимостей и настройка системы**

Сначала необходимо **проверить/выполнить обновление** **системы**:

sudo apt-get update

sudo apt-get -y upgrade

***Примечание***: флаг **“-y”** отвечает на все вопрос об установке “yes” (если у модуля не проверен сертификат и т.п.).

**Устанавливаем средства** для работы с I2C шиной:

sudo apt-get install i2c-tools

**Выполняем команду:**

sudo nano /etc/modules

И **добавляем модули,** приведенные ниже, в конец файла:

snd-bcm2835

i2c-bcm2835

i2c-dev

rtc-ds1307

**В результате** файл «/etc/modules» должен содержать приблизительно следующее:

**Листинг 1.** Файл “/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.

**snd-bcm2835**

**i2c-bcm2835**

**i2c-dev**

**rtc-ds1307**

***Примечание***: драйвер **rtc-ds1307** полностью совместим с устройством DS3231.

Для активация программного доступа к I2C-шине в самой системе (используется Raspbian) используем команду:

sudo raspi-config

В открывшемся меню проходим по следующей цепочке:

“Intefacing Option” -> “I2C” -> “Enable? Yes”.

Затем выбираем «**Finish**» и перезагружаемся (sudo reboot)

**1.3. Проверка корректности настроек и подключения**

**1.3.1. Проверка подключения**

После завершения загрузки системы проверяем подключение устройств на I2C-шине. Для этого выполняем команду:

sudo i2cdetect -y 1

***Примечание***: флаг -y указывает I2C шину. Обычно это 0, 1 или 2 шина.

В данном случае RPi предоставляет первую I2C-шину.

**Результат исполнения:**

pi@raspberrypi:~ $ **sudo i2cdetect -y 1**

0 1 2 3 4 5 6 7 8 9 a b c d e f

00: -- -- -- -- -- -- -- -- -- -- -- -- --

10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

50: -- -- -- -- -- -- -- **57** -- -- -- -- -- -- -- --

60: -- -- -- -- -- -- -- -- **68** -- -- -- -- -- -- --

70: -- -- -- -- -- -- -- --

0x68 – это статический шестнадцатеричный адрес RTC таймера на i2c-шине. На используемом таймере ZS-042 возможно изменение i2c-адреса посредствам физического соединения (пайки) перемычек A0, A1 и/или A2. В таком случае адресе варьируются 3 младших бита.

0x57 – это статический шестнадцатеричный i2c-адрес AT24C32-блока энергонезависимой памяти (EEPROM). Обязательная часть RTC-модуля, которая используется модулем DS3132.

***Примечание***: в данной работе рассматривается взаимодействие только с DS3132.

**1.3.2. Проверка зависимостей и корректности настроек**

На данный момент отображается адрес 0x68 – это не только указывает адрес устройства, но и говорит о том, что устройство не обслуживается никаким из драйверов, хотя необходимые драйверы уже подгружены в ядро. **Теперь необходимо указать системе, что новое устройство по адресу 0х68 – это RTC-часы, и для его обслуживания можно использовать модуль ядра «rtc-ds1307».**

Для этого выполним команду:

sudo /bin/sh -c "/bin/echo **ds1307** **0x68** > /sys/class/i2c-adapter/i2c-1/**new\_device**"

Теперь проверим как отображается устройство на шине:

pi@raspberrypi:~ $ **sudo i2cdetect -y 1**

0 1 2 3 4 5 6 7 8 9 a b c d e f

00: -- -- -- -- -- -- -- -- -- -- -- -- --

10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

50: -- -- -- -- -- -- -- **57** -- -- -- -- -- -- -- --

60: -- -- -- -- -- -- -- -- **UU** -- -- -- -- -- -- --

70: -- -- -- -- -- -- -- --

Теперь адрес устройство по адресу 0x68 (см.пересечении строки 60 и столбца 8) имеет значение UU – это означает, что RTC таймер DS3231 контролируется подгруженным модулем ядра (и это хорошо). Именной такой результат должен быть.

Теперь для окончательной проверки успешности всех подготовительных настроек, выполним команду получения системного времени и времени RTC-модуля

pi@raspberrypi:~ $ **sudo date; sudo hwclock -r**

Fri 26 May 14:55:58 UTC 2017

Fri 26 May 2017 14:55:58 UTC -0.027346 seconds

***Примечание***: время не обязательно должно совпадать, т.к. это только проверка связи с модулем (в данном случае на модуле уже установлено точное время).

**1.4. RTC демон**

**1.4.1. Описание демона**

Алгоритм работы демона следующий: когда система запускается, при старте демона сначала выполняется подготовительная команда регистрации RTC-модуля в системе, и сразу же после этого происходит синхронизация часов внутреннего времени RaspBerry Pi с часами RTC-модуля. После этого демон завершает свою работу, однако из-за используемого флага RemainAfterExit=yes видится в системе как активный.

Когда система выключается, демон выполняет обратную операцию: синхронизирует часы RTC-модуля с часами RPi (т.к. в ходе работы может произойти синхронизация времени через NTP, или же пользователь самостоятельно изменит системное время или часовой пояс). После останова, демон выполняет команду отключения RTC-модуля из системы, и модуль вновь становится не зарегестрированным.

Файл конфигурации **«rtc.service»** необходимо поместить по адресу «/etc/systemd/system/».

**rtc.service**

**[Unit]**

Description=Daemon for resyncing of system clock by RTC Timer DS3231

Before=getty.target

**[Install]**

WantedBy=multi-user.target

**[Service]**

Type=oneshot

RemainAfterExit=yes

**ExecStartPre**=/bin/sh -c "/bin/echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new\_device"

**ExecStart**=/sbin/**hwclock -s**

**ExecStop**=/sbin/**hwclock -w**

**ExecStopPost**=/bin/sh -c "/bin/echo 0x68 > /sys/class/i2c-adapter/i2c-1/delete\_device"

KillMode=process

KillSignal=SIGTERM

SendSIGKILL=yes

TimeoutStartSec=5

TimeoutStopSec=5

**Описание конфигурации:**

**Before**  (модуль должен загружаться ДО запуска всех TTY-терминалов);

**WantedBy** (загрузка должен производиться на 3 уровне, в слое

пользовательских программ);

**Type** (тип демона – одиночный скрипт, который быстро завершится);

**RemainAfterExit** (по завершению процесса, считать демон запущенным);

**ExecStartPre** (перед запуском демона, выполнить указанную команду

регистрации устройства в системе);

**ExecStart** (синхронизировать время RPi, со временем RTC-модуля);

**ExecStop** (синхронизировать время RTC-модуля, со временем RPi);

**ExecStopPost** (после выключения демона, сделать устройство

незарегистрированным);

**KillMode** (отключать конкретный процесс, не контрольную группу);

**KillSignal** (посылать указанный сигнал по команде stop);

**SendSIGKILL** (посылать SIGKILL спустя определенное время (TimeoutStopSec),

чтобы отключить сервис, который по какой-либо причине смог не

выключиться);

**TimeoutStartSec** (время на запуск демона);

**TimeoutStopSec** (время на остановку демона).

**1.4.2. Установка и запуск**

Для регистрации и активации демона выполнить:

sudo systemctl enable rtc # **добавить** **в автозапуск**

sudo systemctl start rtc # **запустить** созданную службу

**1.4.2. Тестирование работы**

Перед началом тестирования:

* регистрируем/запускаем демон;
* отключаем NTP-демон командой **«sudo systemctl disable ntp».**

После завершения подготовительного этапа устанавливаем заведомо неправильное время на RTC-модуле, с помощью следующей команды:

sudo hwclock --set --date="**Sat May 6 15:59:01 2020**"

Еще раз сравниваем текущее системное время и время модуля:

pi@raspberrypi:~ $ **sudo date; sudo hwclock -r**

Sat May 6 **19:36:23** UTC **2017**

Sat May 6 **15:59:01** **2020** -0.399754 seconds

Команда показывает, что системное время и время RTC-таймера отличается. Перезагружаем устройство (RTC модуль продолжает работать):

sudo reboot

После загрузки проверяем статус демона:

pi@raspberrypi:~/daemonRTC $ **sudo systemctl status rtc**

● rtc.service - Daemon for resyncing of system clock by RTC Timer DS3231

Loaded: loaded (/etc/systemd/system/rtc.service; enabled)

Active: active (exited) since Fri 2017-05-26 15:35:14 UTC; 4s ago

Process: 1353 **ExecStart**=/sbin/hwclock -s (code=exited, status=0/**SUCCESS**)

Process: 1347 **ExecStartPre**=/bin/sh -c /bin/echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new\_device (code=exited, status=0/**SUCCESS**)

Main PID: 1353 (code=exited, status=0/SUCCESS)

May 26 15:35:14 raspberrypi systemd[1]: **Started Daemon for resyncing of system clock by RTC Timer DS3231.**

Вновь проверяем время в системе и на модуле:

pi@raspberrypi:~ $ **sudo date; sudo hwclock -r**

Sat May 6 **19:36:23** UTC **2017**

Sat May 6 **19:36:24 2017**  -0.399754 seconds

***Примечание***: если установить системное время отличное от RTC-часов, и остановить или перезагрузить демон (можно перезагрузить устройство), то время на RTC-таймере будет изменено на время системных часов.

**1.5 Подведение итогов**

Время показывается правильно, и теперь система при загрузке сразу получает точное время от таймера, а при выключении – таймер получает обновленное время от RPI и хранит его до включения основного устройства.