НАСТРОЙКА СИСТЕМНОЙ ДАТЫ И ВРЕМЕНИ ОДНОПЛАТНОГО КОМПЬЮТЕРА В АВТОНОМНОМ РЕЖИМЕ ПРИ ОТСУТСТВИИ ЧАСОВ РЕАЛЬНОГО ВРЕМЕНИ

SETTING THE SYSTEM DATE AND TIME OF A SINGLE-BOARD COMPUTER IN OFFLINE MODE WITHOUT REAL TIME CLOCK

УДК 681.518

Ван Евгений Витальевич, бакалавр, Национальный Исследовательский Университет «Высшая Школа Экономики», г. Москва

Van E.V. e.van@bk.ru

Аннотация

Статья содержит возможные варианты решения проблемы отсутствия встроенного модуля часов реального времени одноплатных компьютеров, а также способы работы с системными параметрами даты и времени. Предложенные решения реализованы в рамках работы над программно-аппаратным комплексом, вычислительным модулем которого является одноплатный компьютер. Предложенные методы имеют как преимущества, так и недостатки, которые описаны в статье.

Annotation

The article contains possible solutions to the problem of the lack of a built-in real-time clock module for single-board computers, as well as ways of working with the system date and time parameters. The proposed solutions are implemented within the framework of work on a software and hardware complex, the computing module of which is a single-board computer. The proposed methods have both advantages and disadvantages, which are described in the article.

Ключевые слова: одноплатный компьютер, часы реального времени, протокол Modbus RTU, язык C++.

Keywords: single board computer, real time clock, Modbus RTU protocol, C ++ language.

Введение

Использование одноплатных компьютеров является довольно популярным решением при разработке различных электронных устройств, например, для контроля каких-либо процессов, также в качестве вычислительного модуля измерительных устройств и др. Однако, не в каждой модели такого типа компьютеров предусмотрены часы реального времени, то есть такие, которые не зависят от подаваемого на компьютер питания. Обычно, настройка времени на таких устройствах производится автоматически при наличии подключения к сети Интернет. Но, в условиях включения одноплатного компьютера в состав автономной системы, не всегда можно обеспечить доступ к сети. При работе над программно-аппаратным комплексом была поставлена задача управления системным временем одноплатного компьютера из приложения. Это необходимо для корректного ведения лога событий, а также взаимодействия с системой более высокого уровня. Если работа системы требует настройки данного параметра, необходимо найти решения данной проблемы.

Решения

В качестве примера будет рассмотрен один из популярных одноплатных компьютеров - Raspberry Pi 4 с операционной системой Raspberry Pi OS (основана на Debian). Столкнувшись с проблемой отсутствия часов реального времени, было предложено два варианта решений. Первый - подключение внешнего модуля часов реального времени к пинам GPIO одноплатного компьютера. На рисунке 1 представлен модуль часов реального времени, который был использован для работы над программно-аппаратным комплексом.



Рис. 1. Модуль часов реального времени MOD-RTC2

Данный способ имеет как преимущества, так и недостатки. Различные виды микросхем часов реального времени знакомы операционной системе Raspberry Pi по их названию. Представленный на рисунке 1 модуль имеет микросхему DS3231.Обмен данными с часами реального времени обычно осуществляется при помощи протокола I2C, поэтому необходимо включить передачу данных по данной шине в настройках компьютера. Чтобы определить микросхему в системе, необходимо прописать её название в конфигурационном файле, после чего будут доступны команды терминала для работы с часами реального времени. Команда имеет вид hwclock. Дополнительные параметры команды позволяют установить системное время по часам реального времени и наоборот. Когда модуль подключен к Raspberry Pi, автоматически производится установка системных даты и времени по часам реального времени при включении одноплатного компьютера. Однако, предлагаемые на рынке микросхемы часто имеют разъем UEXT, распиновка которого не совпадает с распиновкой GPIO Raspberry Pi. В связи с этим, может потребоваться дополнительная пайка проводников для подключения, либо использование каких-либо коннекторов. Кроме того, если будет подключена микросхема, незнакомая одноплатному компьютеру, то такой вариант решения проблемы будет уже не таким удобным и простым. Может потребоваться разработка программного кода для работы с I2C. В рамках разработки программно-аппаратного комплекса первый способ не являлся основным, в связи с чем было решено не писать код для настройки работы шины.

Для объяснения второго решения требуется пояснить, что ранее упомянутый программно-аппаратный комплекс имеет, так называемую, плату контроля, которая является связующим звеном для связи Raspberry Pi с другими элементами системы. На данной плате установлен модуль часов реального времени. "Общение" компьютера с часами осуществляется по протоколу Modbus RTU. Конечно, требуется настройка работы с ним. Разработка программного обеспечения устройства велась на языке C++ с использованием фреймворка QT. Встроенные библиотеки для работы с Modbus RTU позволяют удобно и быстро реализовать работу протокола. Пример программного кода с созданием ведущего (Master) и ведомого (Slave)устройств с использованием средств фреймворка Qt представлен на рисунке 2.

```
if (mDevice != nullptr) {
    delete mDevice;
    mDevice = nullptr;
}
switch (mWorkMode) {
    case BUS_SLAVE: {
        QModbusRtuSerialSlave* mSlave = new SlaveRtuSerialModBusDevice(this);/
        mSlave->setServerAddress(mAddress);
        mDevice = mSlave;
        break;
    }
    case BUS_MASTER: {
        QModbusRtuSerialMaster* mMaster = new QModbusRtuSerialMaster(this);
        mMaster->setNumberOfRetries(4);
        mMaster->setTimeout(50);
        mMaster->setInterFrameDelay(5000);
        mDevice = mMaster;
        break;
    }
}
```

Рис. 2. Создание master и slave устройств Modbus RTU

Получая от часов данные в формате quint32 (32 битный беззнаковый целочисленный тип в Qt), можно осуществить перевод в структуру «tm», которая уже реализована в c++ и доступна при подключении заголовочного файла time.h. Пример кода, демонстрирующего преобразование данных даты и времени из quint32 в структуру tm представлен на рисунке 3.

```
mTime = static_cast<time_t> (boardTime);
tm* pTm = localtime(&mTime);
int year = pTm->tm_year + 1900;
int month = pTm->tm_mon + 1;
int day = pTm->tm_mday;
int hour = pTm->tm_hour;
int minute = pTm->tm_min;
int second = pTm->tm_sec;
```

Рис.3.Получение даты и времени в виде структуры

Далее, использовать данные структуры можно так, как требуется в рамках поставленных задач. К примеру, в программном обеспечении программно-аппаратного комплекса осуществляется установка системного времени и даты компьютера Raspberry Pi, с использованием команд терминала системы (date с необходимыми параметрами). На C++ разработаны методы, работающие как раз с такими командами. Так как программное обеспечение по техническому заданию должно быть кроссплатформенным, в методах установки системного времени используются макросы, определяющие тип операционной системы. Способы установки времени являются различными на Linux и Windows. Так, для установки системного времени в Linux-системах необходимо привести данные в строковый тип в определенном формате для даты и времени. Затем добавить эту строку к строке с описанием команды и выполнить её. Для Windows же существует структура, которую необходимо заполнить, а затем выполнить команду установки даты/времени с использованием структуры как аргумента. Пример реализации представлен на рисунках 4 и 5.

```
#ifdef OS_LINUX
    if(parameter == 0) {
        QString dateTime = third+ second + first;
        QString data = "sudo date +%Y%m%d -s ";
        data += "'";
        data += dateTime;
        data += "'";
        const char* date = qPrintable(data);
        system(date);
    } else {
        QString dateTime = first;
        dateTime += ":";
        dateTime += second;
        dateTime += ":";
        dateTime += third;
        QString data = "sudo date +%T -s ";
        data += "'";
        data += dateTime;
        data += "'";
        const char* time = qPrintable(data);
        system(time);
    }
```

Рис. 4. Установка системного времени в Linux

```
#ifdef OS_WINDOWS
    SYSTEMTIME st;
    GetSystemTime(&st);
    if(parameter == 0) {
       st.wYear = third.toInt();
       st.wMonth = second.toInt();
       st.wDay = first.toInt();
    } else {
        st.wHour = first.toInt();
        st.wMinute = second.toInt();
        st.wSecond = third.toInt();
    }
    SetLocalTime(&st);
#endif
```

Рис. 5. Установка системного времени в Windows

В рамках работы ПО, чтение данных часов и установка системного времени должны осуществляться при инициализации системы. Для настройки системного времени также разработано одно из окон приложения, в котором можно изменить системные параметры даты и времени. Отображение параметров производится при помощи средств Qt Quick, где имеется тип date и функция (рис. 6) для получения текущей системной даты и времени. Также можно установить формат отображения параметров (рис. 7). При установке системного времени в окне настройки параметров считываются строковые данные и далее передаются для выполнения ранее упомянутых команд (рис. 4, 5) установки системных даты и времени.

```
property date currentDate: new Date()
```

Рис. 6. Получение текущей даты средствами Qt Quick

```
date.text = "Дата: " + Qt.formatDateTime(new Date(), "dd.MM.yyyy")
time.text = "Время: " + Qt.formatDateTime(new Date(), "hh:mm:ss")
```

Рис. 7. Форматирование отображаемой даты и времени

При записи даты/времени в часы реального времени, сначала осуществляется получение текущих системных даты и времени. Строковое представление обрабатывается для получения параметров, а затем заполняется структура tm с последующим переводом в quint32, после производится запись даты/времени использованием протокола Modbus RTU. Приведение полученных даты и времени к типу quint32 представлен на рисунке 8.

```
QDate date = QDate::currentDate();
QString dateString = date.toString("dd.MM.yyyy");
QStringList dateList = dateString.split(QLatin1Char('.'));
QTime time = QTime::currentTime();
QString timeString = time.toString("hh:mm:ss");
QStringList timeList = timeString.split(QLatin1Char(':'));
tm dateTime;
dateTime.tm_mday = dateList[0].toInt();
dateTime.tm_mon = dateList[1].toInt() - 1;
dateTime.tm_year = dateList[2].toInt() - 1900;
dateTime.tm_hour = timeList[0].toInt();
dateTime.tm_min = timeList[1].toInt();
dateTime.tm_sec = timeList[2].toInt();
mTime = mktime(&dateTime);
boardNewTime = static_cast<quint32>(mTime);
```

Рис. 8. Подготовка данных для записи в часы реального времени

Выводы

В результате поиска решений проблемы настройки системной даты/времени при отсутствии часов реального времени у одноплатных компьютеров (конечно, не у всех) было предложено два варианта. Первый – наиболее простой, хотя и недостаточно универсальный. Второй – универсальный, но требует более сложной реализации с написанием программного кода. Но, так как протокол Modbus RTU является достаточно популярным вариантом «общения» в различных программно-аппаратных комплексах, такой способ будет более удобен. Выбор решений зависит от условий и требований к разработке. Конечно же, решение проблемы не ограничено лишь двумя способами, но именно эти варианты были использованы автором при работе над программно-аппаратным комплексом.

Литература

1. Гийом Лазар Р. П. Освоение Qt 5 [Электронный ресурс]. URL: https://books.google.ru/books?hl=ru&lr=&id=Ms3cDgAAQBAJ&oi=fnd&pg=PP1&dq=qt+framework&ots=6wPUoOXbVw&sig=CRVWxoLUerADjhQ9tzLG3mMSokQ&redir_esc=y#v=onepage&q=qtframework&f=false (дата обращения: 20.05.2021).
2. Пэн Д. Г. и другие. Разработка и реализация протокола Modbus на основе встроенной системы Linux // Труды Международной конференции по встроенному программному обеспечению и системам 2008 г., ICESS Симпозиум

3. Аптон Е., Халфакри Г. Руководство пользователя Raspberry Pi // Wiley, 2014. С. 93-233
4. Риналди Дж. Modbus // Createspace Independent Publishing Platform, 2015.
5. Уоррен Гей. Продвинутый Raspberry Pi // Apress, 2018. С. 259–282
6. DS3231 Документация //Maxim Integrated Products, Inc., 2015.

Literature

1. Guillaume Lazar R. P. Mastering Qt 5 [Electronic resource]. URL: https://books.google.ru/books?hl=ru&lr=&id=Ms3cDgAAQBAJ&oi=fnd&pg=PP1&dq=qt+framework&ots=6wPUoOXbVw&sig=CRVWxoLUerADjhQ9tzLG3mMSokQ&redir_esc=y#v=onepage&q=qtframework&f=false (date of the application: 20.05.2021).
2. Peng D. G. and others. Design and realization of modbus protocol based on embedded Linux system // Proceedings - The 2008 International Conference on Embedded Software and Systems Symposia, ICESS Symposia. , 2008. P. 275–280
3. Upton E., Halfacree G. Raspberry Pi User Guide // Wiley, 2014. P. 93-233
4. Rinaldi J.S. Modbus // Createspace Independent Publishing Platform, 2015.
5. Warren Gay. Advanced Raspberry Pi // Apress, 2018. P. 259–282
6. DS3231 Datasheet //Maxim Integrated Products, Inc., 2015.