Барометр-гигрометр-термометр с батарейным питанием на базе MEMS-датчика BME280, микроконтроллера EFM8SB10F8 и ЖКИ-модуля H1313. Часть 1

Алексей Кузьминов (г. Москва)

В статье приведены принципиальная схема, разводка плат и конструкция барометра-гигрометра-термометра на базе MEMS-датчика BME280 (компании Bosch Sensortec), нового микропотребляющего 51-совместимого микроконтроллера (МК) EFM8SB10F8 (Silicon Laboratories) и ЖКИ-модуля H1313 на базе контроллера HT1616 (Holtek). Для питания устройства могут быть использованы либо две мизинчиковые литиевые батарейки FR03 по 1,5 В, либо литиевая батарейка CR2477 (таблетка) с напряжением 3 В. Ёмкости батареек (около 1 А·ч) достаточно для работы прибора как минимум 10 лет при обновлении показаний атмосферного давления, влажности и температуры раз в 5 минут.

Введение

Барометры-гигрометры-термометры на базе BME280 широко распространены. Однако все они либо требуют отдельного сетевого источника питания, либо достаточно ёмких аккумуляторов, нуждающихся в частой перезарядке. Изредка можно найти в Интернете подобные устройства с питанием от батареек, однако срок их работы недолог, и батарейки часто требуется менять. Такое положение вещей объясняется тем, что все эти устройства для своей работы потребляют значительную энергию, понизить которую не позволяют три причины.

Во-первых, в большинстве случаев в таких устройствах используют широко распространённые 32-разрядные МК (в платах типа Arduino, Raspberry Pi и т.п.), STM32 и им подобные, потребляющие значительный ток (десятки мА). Изредка можно встретить 8-разрядные МК с небольшим энергопотреблением, использующиеся для подобных целей, однако и они даже в режиме сна потребляют ток до нескольких мкА.

Во-вторых, хотя BME280 позволяет производить обмен данными с МК по двум интерфейсам SPI и I^2C, в 99% случаев связь МК с BME280 производится по низкоскоростному двухпроводному интерфейсу I^2C, требующему наличия достаточно низкоомных нагрузочных резисторов, подключённых к питанию, которые потребляют значительный ток (до единиц мА). Низкая скорость I^2C определяется, физическими свойствами этого интерфейса, – достаточно сложным протоколом обмена данными BME280 с МК, который тратит на такой обмен много времени (и программной памяти). I^2C целесообразно использовать, когда к МК подключены несколько устройств на одну и ту же 2-проводную шину, но если подключено всего одно устройство (например, BME280), то использование этого интерфейса становится бессмысленным (если не сказать безграмотным). В отличие от I^2C интерфейс SPI не требует никаких резисторов для своего функционирования: связь МК и BME280 осуществляется напрямую. Кроме того, SPI имеет на порядок бóльшую скорость обмена (до 10 Мбод у МК EFM8SB10F8 и BME280) благодаря своей физической реализации и примитивному протоколу обмена, из-за чего обмен информацией МК и BME280 занимает очень короткое время. А чем меньше это время, тем меньше работают МК и BME280 в активном режиме, потребляя ток до единиц мА, в отличие от режима сна (sleep-режим), в котором МК и BME280 потребляют десятые доли мкА (EFM8SB10F8 – 0,5 мкА, BMP280 – 0,1 мкА). Поэтому если такой обмен информацией идёт, например, раз в 5 минут (за такое время давление, температура и влажность вряд ли существенно изменятся), а всё остальное время чип находится в состоянии сна, то общее потребление энергии подобного устройства существенно снижается.

В-третьих, устройства, связанные с высоким потреблением тока, – это средства отображения измерительной информации. В подавляющем большинстве устройств, использующих BME280, в качестве таких средств применяются светодиодные индикаторы, потребляющие до десятков, а то и сотен мА. Реже используются матричные OLED-дисплеи (20–50 мА). Иногда можно найти устройства, где используются матричные ЖК-дисплеи, потребляющие ток до нескольких мА. В то же время имеются семисегментные ЖКИ, потребление тока которых составляет всего несколько мкА (максимум до 20 мкА), но устройства на базе МК и BME280 с такими ЖКИ встречаются крайне редко, и они, как правило, используют МК с достаточно высоким потреблением тока.

Здесь необходимо добавить, что в последнее время в широкой продаже появились дисплеи, которые получили название «электронная бумага» – E-paper или «электронные чернила» – E-inc. Это матричные дисплеи с разрешением 158×158 или 200×200 пикселов (наиболее дешёвые и востребованные варианты). Потребление тока у них существенно выше (до нескольких мА), чем у ЖКИ, но только во время обмена информацией с МК. Зато когда обмена нет, эти дисплеи вообще ничего не потребляют, т.е. они сохраняют информацию на экране даже при выключении питания и причем достаточно долгое время (до нескольких месяцев и более). Кроме того, некоторые из таких дисплеев имеют возможность обновлять информацию не всего экрана, а только определённой его области, что существенно снижает их энергопотребление. Применительно к BME280 это означает, что можно, например, вывести в определённую область экрана слова «Давление», «Влажность», «Температура» один раз, а обновлять только цифровые значения этих физических величин в другой области экрана. Матричное строение таких дисплеев, конечно, требует большего объёма

как программной, так и оперативной памяти по сравнению с примитивным обменом с ЖКИ, и «потянет» ли МК EFM8SB10F8 с программной памятью всего 8 кБ и оперативной 128 байт (data) плюс 512 байт «внешней» (xdata) подобный обмен, ещё неизвестно (по подсчётам автора, скорее всего, да). Если нет, можно использовать похожий МК EFM8SB20F16 (16 кБ программной памяти и 4 кБ xdata), который по стоимости не отличается от EFM8SB10F8 (и даже немного дешевле).

Таким образом, резюмируя вышесказанное, можно отметить, что имеются три проблемы, не позволяющие использовать BME280 в устройствах на базе МК с батарейным питанием и достаточно долгим сроком эксплуатации. В настоящей статье приведено устройство, где все эти три проблемы сняты.

Дальнейшее изложение построено следующим образом. Вначале приводится принципиальная схема устройства, далее кратко описаны его программные средства, затем показана разводка и внешний вид его платы, после этого рассказано о конструкции устройства (фотографии в открытом корпусе и общий вид работающего устройства) и о результатах его работы.

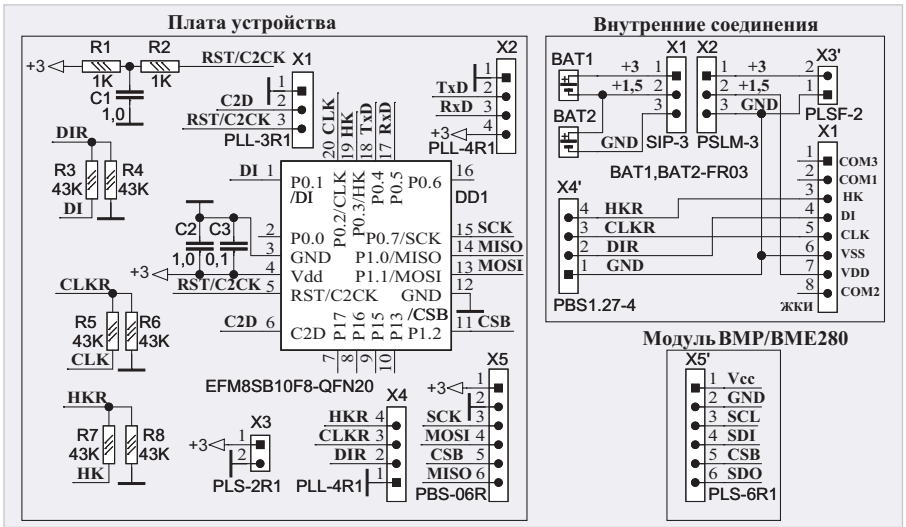


Рис. 1. Принципиальная схема устройства

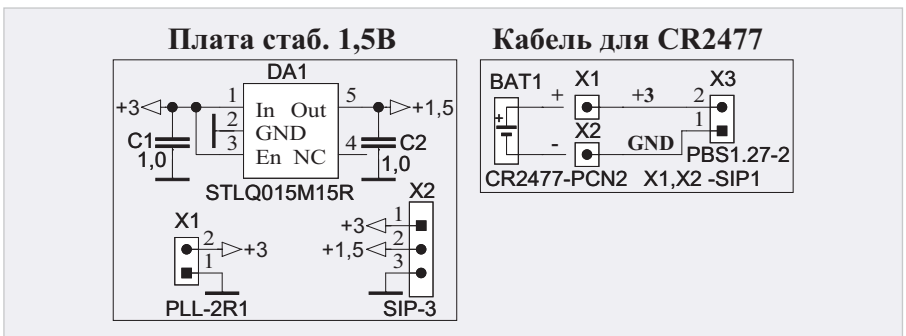


Рис. 2. Схема подключения батарейки CR2477

Принципиальная схема устройства

Как видно из рис. 1, схема устройства достаточно проста. В качестве МК используется DD1 (EFM8SB10F8G-A-QFN20) в корпусе QFN20 размером 3×3 мм. Потребление тока МК в sleep-режиме составляет 0,5 мкА (SB в названии МК – сокращение от Sleepy Bee). В качестве датчика давления, влажности и температуры используется готовый модуль, в состав которого входит собственно микросхема BME280, несколько резисторов и конденсаторов. Стоимость такого модуля, как ни странно, существенно ниже стоимости самой микросхемы BME280, поэтому он и использован в приборе. Кроме того, вместо модуля на BME280 можно использовать модуль на основе BMP280, который измеряет только давление и температуру (он почти на порядок дешевле модуля с BME280). Модуль с BME280 сопрягается с МК по 4-проводному (для BME280) интерфейсу SPI сигналами MISO, MOSI, SCK и CSB. Последний сигнал используется в качестве выбора кристалла (Chip Select – CS). Для МК это 3-проводный SPI (без использования сигнала NSS, выставляемого автоматически в 4-проводном режиме после передачи/приёма каждого байта). Сигнал CSB устанавливается в программе «вручную» тогда, когда это требуется (в начале и в конце обмена при передаче или приёме байта или массива байт).

Здесь необходимо добавить, что, для того чтобы снизить ток потребления модулей BME280/BMP280 до уровня потребления тока самих микросхем BME280/BMP280, рекомендуется выпаять из этих модулей все 4 резистора номиналом 10 кОм, подключённых к линиям интерфейса (входным сигналам SCL, SDA, CSB и выходному сигналу SDO). Дело в том, что входные сигналы для BME280 одновременно являются выходными для МК (SCK, MOSI и CSB соответственно), которые настроены как цифровые (пушпульные – push-pull) выходы и в этих резисторах не нуждаются, а выходной сигнал BME280 (SDO) является входным для МК (MISO), настроенным как цифровой вход со слаботоковой подтяжкой к питанию (с эквивалентным сопротивлением около 500 кОм), в связи с чем также не нуждается в дополнительном резисторе.

ЖКИ-модуль H1313 на базе контроллера HT1611/HT1613 (далее ЖКИ) потребляет ток всего 3 мкА (типовое значение), поскольку его напряжение питания 1,5 В, что в 2 раза меньше напряжений питания МК и BME280 (3 В). В связи с этим имеются две проблемы для использования этого ЖКИ в приборе. Во-первых, источник питания должен обеспечить два напряжения питания (1,5 В и 3 В), во-вторых, поскольку цифровые сигналы МК имеют амплитуду около 3 В, а сигналы управления ЖКИ – в 2 раза меньше, требуется их согласование по амплитуде. Первая проблема может быть решена двумя способами: либо использовать 2 батарейки с напряжением по 1,5 В каждая, включённые последовательно (BAT1 и BAT2 – литиевые FR03 – в пунктирном прямоугольнике в верхней правой части рис. 1), либо применить 3-вольтовую батарейку (например, литиевую CR2477) и стабилизатор с выходным напряжением 1,5 В STLQ015M15R (DA1, рис. 2), потребляющий ток не более 1 мкА. Применение литиевых батареек объясняется тем, что, во-первых, такие батарейки могут служить до 10 лет и разряжаются не более чем на 1% в год, во-вторых, их кривая разряда (зависимость выходного напряжения от времени и тока раз-

ряда) практически горизонтальная, в отличие от алкалиновых, напряжение которых падает в зависимости от времени (и тока разряда), и тем более солевых, у которых кривая разряда падает ещё существенней. Кроме того, срок службы алкалиновых батареек – всего 2-3 года, а солевых и того меньше. Для согласования амплитуд сигналов МК и ЖКИ используются делители напряжения (в 2 раза), построенные на резисторах R3–R8. Для передачи данных в ЖКИ используется сигнал DI, который стробируется импульсами CLK. ЖКИ может работать в двух режимах: режим отображения времени (часы, минуты и секунды) и режим отображения информации, переданной МК с помощью сигналов DI и CLK. Переключение режимов работы ЖКИ осуществляется сигналом НК. При низком уровне НК (лог. 0) ЖКИ отображает информацию, переданную сигналами DI и CLK (этот режим используется в приборе), при высоком (лог. 1) – часы (этот режим при необходимости может быть также использован: схема это позволяет).

Напряжение (3 В) на плату поступает с 2-штырькового разъёма PLS-2R1 (X3), к которому подключается 2-контактное ответное гнездо PLSF-2 (X3' – в правом верхнем пунктирном прямоугольнике рис. 1). ЖКИ подключается к 4-штырьковому разъёму PLL-4R1 (X4) ответным гнездом PBS1.27-4 (X4' в правом верхнем пунктирном прямоугольнике). Модуль BME280 подключается к 6-контактному гнезду PBS-06R (X5) с помощью 6-контактного штыревого разъёма PLS-6R1, установленного на этом модуле (X5' – в правом нижнем пунктирном прямоугольнике рис. 1).

Программироваться МК может двумя способами:

- 1-й вариант – с помощью USB DEBUG адаптера, который сопрягается с компьютером по интерфейсу USB, а с МК – по 2-проводному интерфейсу C2. Для этого предназначен 3-контактный штыревой разъём PLL-3R1 (X1), на который выведены 2 сигнала: RST/C2CK, C2D и «земля». Для сопряжения используется кабель, который одним концом (ответное 3-контактное гнездо) подключается к разъёму X1, а второй его конец подключается к выходному разъёму USB DEBUG адаптера. Схему такого кабеля можно найти в [1]. Цепочка R1R2C1 используется для штатной работы интерфейса C2 и штатной работы МК при включении питания (Power On Reset – POR);
- 2-й вариант – по интерфейсу RS-232 с помощью COM-порта компьютера (COM1). Для сопряжения используется 4-контактный штыревой разъём PLL4-R1 (X2), на который выведены 2 сигнала (TxD, RxD), питание (+3 В) и «земля». К этому разъёму подключается преобразователь уровней интерфейса RS-232-TTL, а к нему – кабель сопряжения с COM-портом компьютера. Все схемы и подробное описание этого режима программирования можно найти в [2]. Для перевода МК в этот режим программирования необходимо замкнуть перемычкой (джампером) контакты 1–2 разъёма X1.

Конденсаторы C2 и C3 – блокировочные; они предназначены для штатной работы МК. Все резисторы и конденсаторы (керамические, рассчитанные на пробивное напряжение не менее 10 В) – для поверхностного монтажа размером 0603. Как видно из вышеизложенного, схема не отличается особой сложностью, а потому плата устройства легко разводится и имеет габариты всего 16×18 мм.

Программные средства

Программа для BME280 в уже готовом загрузочном *.hex-формате – EFM8SB10F8G-A-QFN24.hex (EFM8SB10F8G-A-QFN24_2.hex – для BMP280) приведена в дополнительных материалах к статье на сайте журнала. Её можно запрограммировать в МК с помощью одного из двух способов программирования, о которых говорилось выше. В принципе, на этом можно было бы и закончить раздел статьи о программных средствах, однако автор счёл своим долгом поделиться некоторыми ключевыми моментами программы и «багами», обнаруженными автором, с теми, кто имеет желание, возможность и навыки программирования и захочет самостоятельно написать свою программу. Дальнейшее изложение предполагает, что информация о МК EFM8SB10F8G_A_QFN20 (справочный листок – datasheet и руководство пользователя – Reference Manual) и о BMP280 (справочный листок – datasheet) уже известны (если нет, их можно легко найти в Интернете и ознакомиться с ними). Кроме того, предполагается, что информация о среде программирования Simplisity Studio v.4 (Silicon Laboratories) также известна.

Но вначале о сути самой программы. Она может быть условно разделена на две части. Это основная программа и программа инициализации устройств (InitDevice.c). Об инициализации устройств речь пойдет далее, а здесь кратко остановимся на основной программе. Но прежде несколько слов о BME280.

Эта микросхема разработана по MEMS-технологии, включает в свой состав 20-разрядный АЦП, имеющий возможность производить передискретизацию и осреднение результатов. Для снижения шума АЦП BME280 оборудована специальным фильтром, который можно включать и выключать, если в нём нет необходимости. У неё 3 режима работы: режим сна (sleep-mode), в котором измерения не производятся и потребление тока минимально, нормальный (normal-mode), когда производятся измерения через определённый и задаваемый интервал времени (от 0,5 до 1000 мс), причем, когда измерения не производятся, BME280 автоматически переходит в sleep-режим, и, наконец, режим принудительного (форсированного) измерения (force-mode); в этом режиме измерения производятся тогда, когда это требуется, и при этом после проведения измерений микросхема также автоматически переходит в sleep-режим. Этот режим (force-mode) и использован в приборе. Для повышения точности измерений в BME280 имеются калибровочные коэффициенты (3 – для температуры, 9 – для давления и 6 – для влажности), которые настраиваются на заводе-изготовителе и записываются в постоянную память. Эти коэффициенты доступны только для чтения. Кроме того, у микросхемы имеется идентификационный номер (60h – у BME280 и 58h – у BMP280), который также доступен для чтения. После окончания измерений расчёт давления, температуры и влажности производится по определённым формулам, в которые входят измеренные АЦП значения этих физических величин, с учётом калибровочных коэффициентов. Эти формулы приведены в справочном листке (datasheet). Для обмена информацией с МК, как уже упоминалось, микросхема оснащена двумя интерфейсами: I^2C и SPI. Максимальная скорость обмена по SPI составляет 10 Мбод (точнее, максимальная частота сигнала, стробирующего данные (SCK), составляет 10 МГц).

Теперь более подробно рассмотрим работу программы. Вначале производится чтение идентификационного номера (ID) BME280 и вывод его на экран ЖКИ, на котором ID отражается в течение около 2 секунд. Если он равен «60»,

то это означает, что всё в порядке, что микросхема именно BME280, и что её связь с МК по SPI работает верно. Далее читаются и запоминаются в оперативной памяти МК (data) калибровочные коэффициенты. После этого происходит инициализация BME280, т.е. в неё загружается требуемый режим работы (в начале – normal-mode), количество осреднений (sampling) по каждому из измерений (давления – P, температуры – T и влажности – H), фильтр выключается, а время работы в нормальном режиме (standby_time) устанавливается на максимум (1 секунду). Далее включается режим force-mode, и на этом инициализация заканчивается. В этом месте программы устанавливается метка “start”, на которую программа возвращается после всех измерений, индикации показаний на ЖКИ и окончания режима сна МК (sleep-mode), т.е. примерно через каждые 5 минут. Значения АЦП, отражающие значения измеренных величин, считываются подряд, по 3 байта для P, T и H (т.е. всего 9 байт, при этом старший байт H не используется), начиная со старшего байта (Most Significant Byte – MSB) P, т.е. с адреса 0xf7 (BME280_REG_PRESS_MSB). После этого производится расчёт T, P и H именно в такой последовательности, поскольку значение T используется для расчёта P и H. Далее рассчитанные значения P, H и T выводятся на ЖКИ, и МК переводится в режим сна (sleep-режим), выход из которого осуществляется по тревожному сигналу (alarm) от специального таймера (Real Time Clock – RTC). Это событие, как уже указывалось, происходит каждые 5 минут. После этого программа возвращается на метку “start”, о которой говорилось выше, т.е. всё повторяется в бесконечном цикле. Вот и вся суть программы.

Здесь необходимо добавить, что после входа в sleep-режим все устройства МК, включая его процессор, отключены, кроме специального устройства управления потреблением энергии (Power Monitor Unit – PMU) и специального таймера (Real Time Clock – RTC) со встроенным в него микропотребляющим НЧ генератором (LFOSC0), работающим на частоте 16,4 кГц, поэтому МК в sleep-режиме и потребляет ток 0,5 мкА.

Один из ключевых моментов программы – вход МК в режим сна (sleep-режим) и выход из него. Ниже этот момент программы рассмотрен более подробно.

Для переключения МК в sleep-режим в PMU предусмотрен специальный бит, который можно изменять программно. Но поскольку процессор в sleep-режиме выключен, он не может выполнять никакие команды (инструкции) программы. Поэтому выход из sleep-режима организуется аппаратно по нескольким причинам, а точнее, событиям.

Одной из причин (или событий) выхода из sleep-режима является событие достижения счётчика в RTC своего максимального значения (в программе используется именно оно). Это значение (эквивалентное времени около 5 минут) записывается в RTC ещё на этапе инициализации устройств МК (см. далее). RTC, досчитав до этого максимума, автоматически сбрасывается (авторесет) и начинает счёт сначала. При этом RTC генерирует специальный «тревожный» сигнал (alarm). Это событие (alarm от RTC – источник выхода из sleep-режима) кодируется в регистре PMU0CF (PMU) специальным битом. Кроме того, при наступлении этого события в регистре PMU0CF автоматически устанавливается специальный флаг, для которого предусмотрен свой бит. При выходе из sleep-режима этот флаг необходимо сбросить программно. В активном режиме системная тактовая частота работы процессора МК (SYSCLK) устанавливается равной частоте специального малопотребляющего ВЧ генератора LPOSC0 (20 МГц). Это происходит ещё на этапе инициализации устройств (см. далее).

Теперь, после этих предварительных замечаний, несложно уже понять рекомендуемую производителем МК EFM8SB10 последовательность входа/выхода в/из sleep-режима. Как следует из справочного руководства (Reference Manual – RM), она состоит в следующем.

1. Вначале необходимо отключить все аналоговые периферийные устройства (АЦП, компараторы и т.п.). Но поскольку в данном случае они не используются, их можно отключить ещё на этапе инициализации устройств (см. далее), поэтому этот пункт выполнять не требуется. А вот интерфейс SPI, скорость работы которого определяется системной тактовой частотой (SYSCLK), а она, в свою очередь, определяется частотой работы маломощного генератора LPOSC0 (20 МГц), следует отключить, поскольку следующим пунктом будет отключение генератора LPOSC0 и переключение SYSCLK на частоту работы RTC, а она, в свою очередь, определяется частотой работы микромощного низкочастотного генератора LFOSC0 (16,4 кГц). Поэтому, во избежание неадекватной работы SPI при смене частот SYSCLK, и требуется отключение SPI.

2. Переключить SYSCLK на работу от RTC (16,4 кГц).

3. Войти в sleep-режим, установив в регистре PMU0CF SLEEP-бит и бит выхода из sleep-режима по alarm’у от RTC. Здесь следует добавить, что никакие логические операции (логического умножения “&” или логического сложения “|”) с регистром PMU0CF не допускаются, или, другими словами, в PMU0CF должно быть записано строго определённое число.

4. Выйти из sleep-режима, предварительно выполнив 4 команды NOP (No Operation – нет операции, т.е. пустая команда), чтобы обеспечить повторную синхронизацию НЧ-генератора LFOSC0 с процессором. После этого переключить SYSCLK на работу от маломощного ВЧ-генератора LPOSC0 (20 МГц) и дождаться установки бита адекватной работы SYSCLK.

Здесь следует добавить, что после того как произойдёт событие, по которому осуществляется выход из sleep-режима, в данном случае – по alarm’у от RTC, в PMU автоматически установится бит (флаг) этого события. Поэтому для повторного входа в sleep-режим (через время, определяемое RTC, в данном случае – 5 минут) этот флаг необходимо сбросить программно. Для этого в регистр PMU (PMU0CF) необходимо также записать определённое число, обнуляющее этот флаг и бит переключения в sleep-режим, а также сохраняющее бит выхода из sleep-режима по alarm’у от RTC. В конце выхода из sleep-режима необходимо включить SPI.

Несмотря на такое «пространное» объяснение, по фрагменту основной программы (на C51), связанному с входом и выходом в/из sleep-режима, приведённому ниже, можно убедиться, что это достаточно простая процедура. Здесь необходимо разъяснить, что после входа в sleep-режим (т.е. после выполнения команды PMU0CF=0x84;) сразу следуют 4 пустые команды (nop_();), и на первый взгляд кажется, что выполнение этих команд начинается сразу же после выполнения предыдущей команды (по крайней мере, так написано в программе). Однако, поскольку в sleep-режиме процессор МК остановлен, а эти команды (инструкции) выполняются именно им, то они не будут выполняться до тех пор, пока не произойдёт событие выхода из sleep-режима, а этот выход, в свою очередь, наступит только тогда, когда поступит тревожный сигнал (alarm) от RTC, счёт-

чик которого досчитает до максимального значения и обнулится по авторесету, т.е. через 5 минут. Сами же эти 4 NOP'а требуются для синхронизации «проснувшегося» процессора с тактовой частотой НЧ-генератора LFOSC0 (16,4 кГц), встроенного в RTC.

Фрагмент основной программы с некоторыми комментариями, связанный с входом/выходом в/из sleep-режима, приведён ниже.

```
//-----------------------
 // Вход в sleep-режим
 //-----------------------
SPI0CN0 &= 0xfe;//Запрет SPI.
CLKSEL=0x83; //SYSCLK = частота
работы RTC (LFOSC0 - 16 кГц).
_nop_ ();
_nop_ ();
_nop_ ();
_nop_ ();
      PMU0CF=0x84;//Установка
бита sleep-режима и бита
                  //выхода из
него по alarm'у от RTC(5 минут).
 //------------------------
 // Выход из sleep-режима
 //------------------------
_nop_ ();
_nop_ ();
_nop_ ();
_nop_ ();
CLKSEL=0x04; //SYSCLK = LPOSC (20 Мгц).
DEL10MS();   //Задержка 10 мс.
while ((CLKSEL & 0x80)==0);//Ожи-
дание установки SYSCLK = LPOSC.
PMU0CF=0x24;//Сброс бита sleep-
режима, флага от предыдущего входа
  // в него и сохранение бита
разрешения выхода
  // из sleep-режима по alarm'у
от RTC.
SPI0CN0 |= 0x01;//Разрешение SPI.
//-----------------------
```

При программировании чтения по SPI из BME280 автор обнаружил один неприятный «баг». Он состоит в том, что при чтении идентификационного номера (ID) и калибровочных коэффициентов используется стандартная процедура чтения по SPI, а при чтении показаний АЦП эта процедура даёт неадекватные значения. Для объяснения этого эффекта и снятия этого бага сделаем некоторое отступление по поводу стандартных протоколов чтения и записи по SPI.

Для того чтобы вывести байт по SPI, требуется записать его в регистр SPI0DAT, дождаться установки бита (флага) окончания передачи (SPI0CN0_SPIF) и сбросить этот бит. Эта стандартная процедура (подпрограмма записи по SPI) приведена ниже.

```
void outspi(uint8_t byte) {
SPI0DAT = byte;// Вывод байта по SPI
while (!SPI0CN0_SPIF); // Ожидание
окончания вывода байта
SPI0CN0_SPIF = 0;// Сброс флага
окончания передачи.
}
```

При чтении байта по SPI требуется записать в SPI0DAT ничего не значащий фиктивный (подставной – dummy) байт (например, 0xff), дождаться установки флага SPI0CN0_SPIF и сбросить его. Результат чтения будет в регистре SPI0DAT. Подпрограмма чтения по SPI приведена ниже.

```
uint8_t inspi() {
   uint8_t byte;
   SPI0DAT = 0xff;       // Вывод
фиктивного байта.
while (!SPI0CN0_SPIF);// Ожидание
окончания ввода байта.
SPI0CN0_SPIF = 0;     // Сброс
флага окончания приема.
byte = SPI0DAT;       // Ввод
байта в микроконтроллер
return (byte);
}
```

Возвращаясь к прерванной последовательности изложения, следует пояснить, что сама процедура чтения из области памяти BME280 состоит в том, что вначале требуется записать байт адреса, откуда необходимо получить информацию (вышеприведённой подпрограммой записи), а затем – прочитать эту информацию (вышеприведённой подпрограммой чтения). Причем, если требуется прочитать несколько (2 и более) подряд расположенных байт, или, другими словами, массив байт (многократное чтение – multiple byte read), нет необходимости перед чтением каждого байта указывать его адрес. Достаточно записать адрес первого элемента массива и далее просто читать подряд столько раз, сколько элементов в массиве. Другими словами, перед чтением каждого следующего байта происходит автоматическое инкрементирование адреса.

Например, для чтения первого двухбайтного калибровочного коэффициента температуры (в справочном листке он обозначен как dig_T1), расположенного по адресам 0x88 (старший байт) и 0x89 (младший байт), необходимо вначале записать адрес старшего байта (0x88), а затем произвести двукратное чтение. В этом случае такая процедура чтения работает правильно, и к ней никаких претензий нет.

Иное дело, если требуется прочитать показания АЦП. Эти показания для давления (ADC_P), температуры (ADC_T) и влажности (ADC_H) расположены в памяти BME280 также подряд, начиная с адреса 0xF7 (старший байт показания давления – press_msb). Причем каждое показание АЦП расположено в 3 байтах. Например, показание ADC_P расположено по адресам 0xF7 (старший байт), 0xF8 (средний байт – press_lsb) и 0xF9 (младший байт – press_xlsb). Далее идёт показание ADC_T, а за ним – ADC_H. Таким образом, все три показания занимают 3×3 = 9 байт. Если требуется прочитать подряд все 9 байт, то, применив вышеприведённую процедуру многократного чтения, вначале следует записать в BME280 адрес первого элемента этого массива, т.е. 0xF7, а затем осуществить 9-кратное чтение. Однако в этом случае получится полная белиберда. В чём же здесь дело? Как выяснил автор, при записи адреса 0xF7 вышеприведённой подпрограммой записи байта и последующего чтения байта вышеприведённой подпрограммой чтения будет произведена операция чтения не с адреса 0xF7, а со следующего (0xF8). Другими словами, после записи адреса 0xF7 (см. подпрограмму записи) в регистр SPI0DAT сразу же запишется содержимое 0xF7, а при последующем чтении после записи фиктивного байта (см. подпрограмму чтения) в SPI0DAT запишется содержимое уже следующего байта (0xF8), т.е. после записи адреса происходит его автоматическое инкрементирование, и, таким образом, содержимое регистра с адресом 0xF7 теряется.

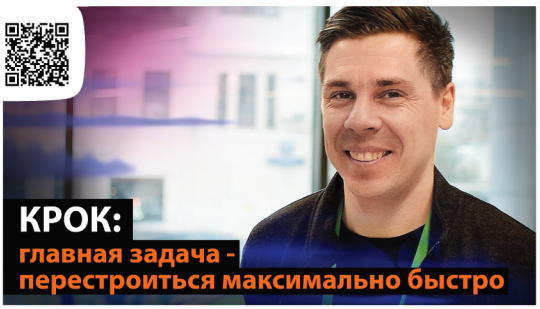
Решение этой проблемы, а также разводка печатных плат, корпус устройства и результаты тестирования итогового изделия будут рассмотрены в следующей части статьи.

Литература

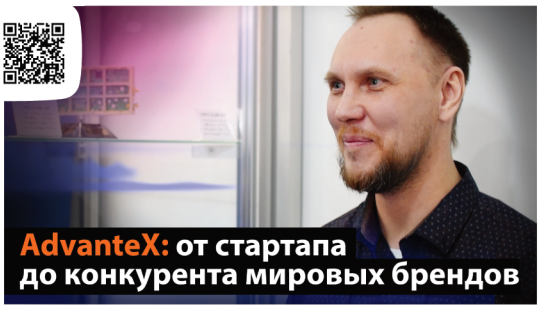
1. *Кузьминов А.Ю.* Связь между компьютером и микроконтроллером. Современные аппаратные и программные средства. М.: Перо, 2018.
2. *Кузьминов А.* Программирование микроконтроллеров EFM8 с помощью встроенного загрузчика программ // Радио. 2018. № 12.

YouTube

Смотрите на канале
СОВРЕМЕННАЯ ЭЛЕКТРОНИКА

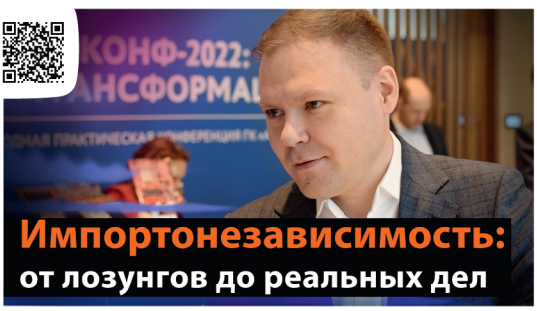


КРОК: главная задача – перестроиться максимально быстро
Валентин Губарев. КРОК

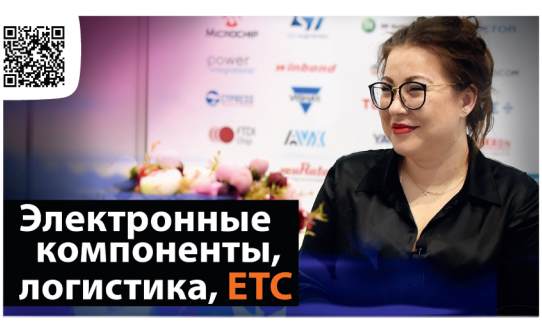


AdvanteX: от стартапа до конкурента мировых брендов
Андрей Поляков, AdvanteX

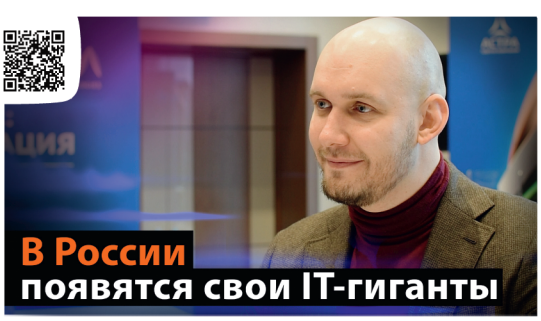
IT-инфраструктура под контролем: новые возможности Астра Линукс
Наталья Царёва и Александр Гришин, ISPSystem



Импортонезависимость: от лозунгов до реальных дел
Ильдар Вагизов, АйСиЭл Техно



Электронные компоненты, логистика
Катерина Кулаковская, ETC



В России появятся свои IT-гиганты
Станислав Иодковский, IVA Technologies