Цифровой вольтметр с высоким разрешением
Часть 2. Программные средства

Алексей Кузьминов (compmicrosys@mail.ru)

Статья посвящена цифровому вольтметру с разрешением 6 десятичных разрядов, на базе микроконтроллера EFM8LB12, оснащённого 14-разрядным SAR АЦП. Высокое разрешение прибора получено в результате использования известного метода передискретизации и осреднения, позволяющего существенно поднять разрешающую способность АЦП. В первой части были представлены принципиальные схемы устройств. Во второй части речь пойдёт о программных средствах.

Программа для вольтметра использует метод передискретизации и осреднения, подробно описанный в [1]. В результате получается 4-байтное целое беззнаковое (ulong) число, которое содержит результат аналого-цифрового преобразования. Помимо отличий, перечисленных ниже, настоящая программа имеет два дополнения, отсутствующих в программе [1]. Первое дополнение – подпрограмма вывода информации на ЖКИ. Об этом уже было подробно написано в первой части. Второе – процедура записи во флеш-память микроконтроллера и чтения из неё значения коэффициентов калибровки нуля и полной шкалы.

Отличия настоящей программы от программы, приведённой в [1], обусловлены следующими факторами:

1. ИОН выбран внешний с напряжением Vref = 3 В (взамен внутреннего с напряжением 2,4 В);
2. выбран внутренний тактовый генератор с частотой 72 МГц (взамен внешнего той же частоты);
3. режим работы интерфейса SPI выбран 3-проводный (взамен 4-проводного), частота импульсов SCK выбрана около 400 кГц (взамен 12 МГц);
4. корпус микроконтроллера QFN24 (взамен QFN32).

Эти изменения касаются подпрограммы инициализации устройства (InitDevice.c). Поэтому вначале в среде Simplicity Studio следует выбрать опцию конфигуратора устройства, в котором выбрать микроконтроллер EFM8LB12F64E-A-QFN24. В общем меню конфигуратора (см. рис. 8) потребуется настройка следующих опций: Voltage Reference, ADC0, Clock Control, HFOSC 0/1 и SPI0. Эти опции обведены красными овалами. Все остальные настройки такие же, что и в программе [1], их также следует настроить, взяв за основу конфигурационный файл EFM8LB1_ADC_Lib_Autoscan_Large_Buffer.hwconf.

Сначала следует выбрать опцию Voltage Reference и в свойствах выбрать внешний ИОН напряжением 3 В (см. рис. 9а). Затем следует выбрать опцию ADC0 и в свойствах указать, что используется вход VREF, аналоговая «земля» подключена к порту P0.1, а входной аналоговый сигнал следует подавать на порт P0.2 (см. рис. 9б).

Далее следует выбрать опцию HFOSC 0/1 и в свойствах выбрать внутренний генератор 72 МГц (см. рис. 10а). Затем выбрать опцию Clock Control и в свойствах выбрать источник тактирования как внутренний высокочастотный генератор и делитель SYSCLK/1 (см. рис. 10б).

Далее выбрать опцию SPI 0 и в свойствах выбрать режим Master, Clock Phase и Clock Polarity и установить делитель частоты равным 90, который соответствует частоте 395,604 кГЦ (см. рис. 11).

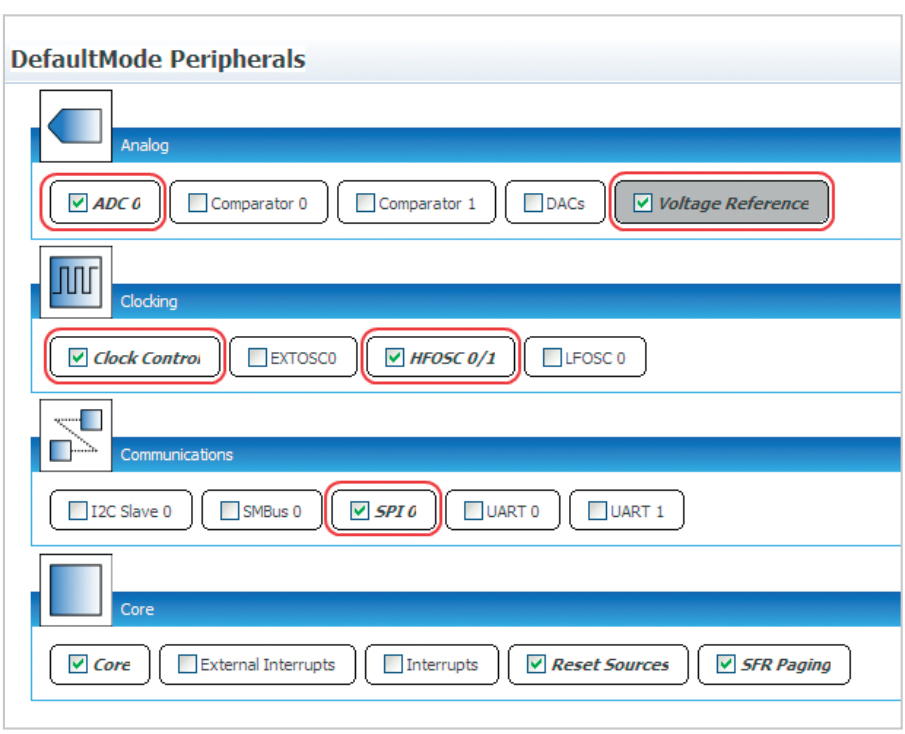

Рис. 8. Общее меню настроек конфигуратора

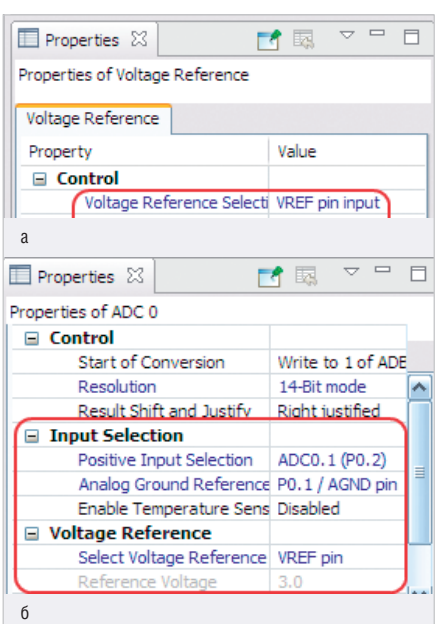

Рис. 9. Настройка опорного напряжения Vref: выбор внешнего Vref (а), установка порта входа АЦП (P0.2), аналоговой «земли» (P0.1) и внешнего ИОН Vref (б)

Далее, открыв файл EFM8LB12F64E-A-QFN24.hwconf, увидим картинку с портами микроконтроллера EFM8LB12F64E-A-QFN24. С помощью команд Skip необходимо придать портам микроконтроллера вид, представленный на рисунке 12.

Порты P1.2 и P1.3 необходимо настроить как цифровые выходы (см. рис. 13а), а порты P0.3, P1.5 и P1.6 – как цифровые входы со слаботочными подтяжками (см. рис. 13б).

Далее следует заменить исходный конфигурационный файл созданным файлом EFM8LB12F64E-A-QFN24.hwconf, который и будет использован при автоматическом создании файла инициализации устройства InitDevice.c. Для этого следует скопировать созданную конфигурацию EFM8LB12F64E-A-QFN24.hwconf, вставить её рядом с исходной EFM8LB1_ADC_Lib_Autoscan_Large_Buffer.hwconf и удалить исходную. К счастью, среда Simplicity Studio допускает такие операции над «деревом» программы.

После этого необходимо сменить модель памяти со Small на Large. Для этого в проекте выбрать опцию Properties, в меню выбрать опцию Setting, как показано на рисунке 14а, затем выбрать модель Large (см. рис. 14б). После этого нажать кнопки Apply и OK.

Рассмотрим процедуру записи и чтения коэффициентов, полученных в результате калибровки нуля и полной шкалы (см. далее), во флеш-память. Для этого была использована готовая программа, приведённая в Simplicity Studio в качестве примера и предназначенная для микроконтроллера EFM8LB1 (EFM8LB1_Flash). В этом примере используется основной файл EFM8LB1_Flash.c и 4 дополнительных: EFM8LB1_FlashPrimitives.h, EFM8LB1_FlashUtils.h, EFM8LB1_FlashPrimitives.c, EFM8LB1_FlashUtils.c. В файле EFM8LB1_FlashUtils.c имеется несколько подпрограмм, из которых было оставлено только две: запись в память (FLASH_Write) и чтение (FLASH_Read). Подпрограмма EFM8LB1_FlashPrimitives.c

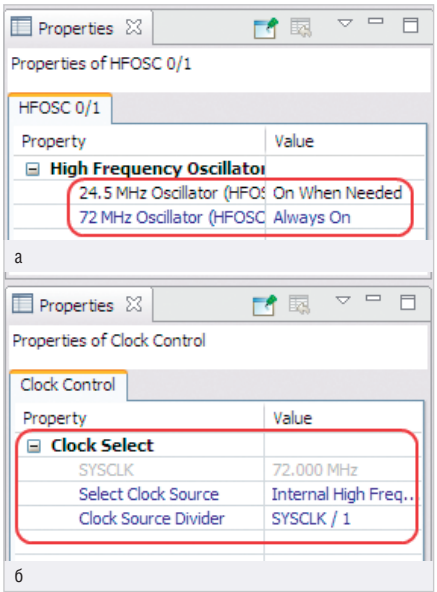

Рис. 10. Установка внутреннего генератора 72 МГц для тактирования микроконтроллера: включение внутреннего генератора в работу (а), установка источника тактирования от внутреннего генератора (б)

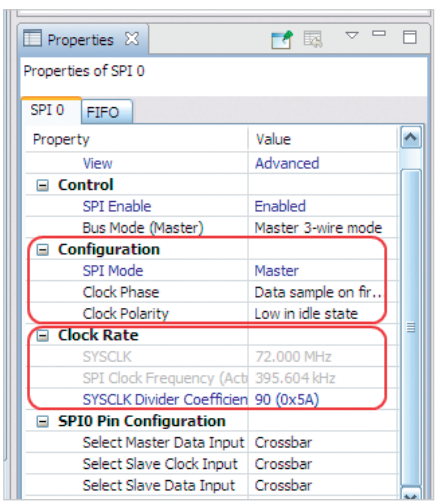

Рис. 11. Настройка интерфейса SPI в 3-проводном режиме master и выбор коэффициента делителя (90) для установки частоты импульсов SCK 395,604 кГц

использована полностью (в ней как раз приведена подпрограмма стирания страницы памяти объемом 512 байт – FLASH_PageErase).

Подпрограммы FLASH_Write и FLASH_Read предназначены для записи во

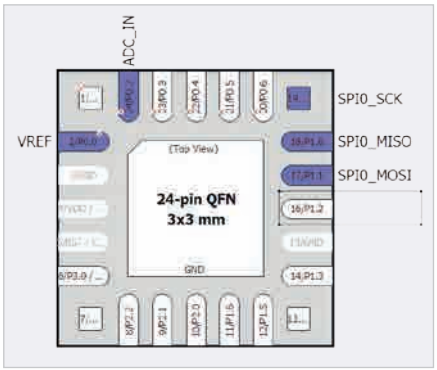

Рис. 12. Общий вид портов микроконтроллера в корпусе QFN24

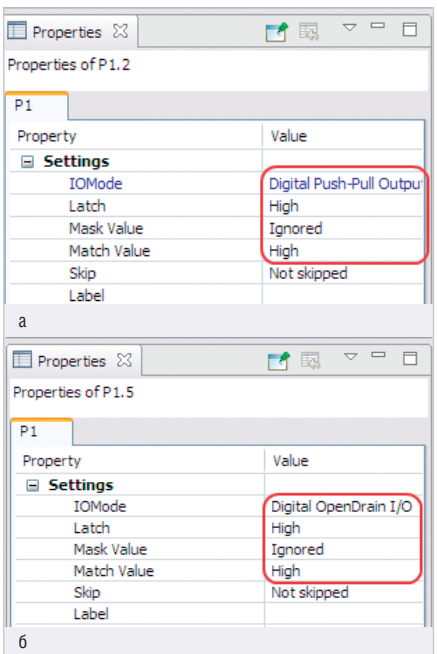

Рис. 13. Настройка портов вывода сигналов CSML и CSST (а) и ввода сигналов CAL0, CAL1 и DIAP (б)

флеш-память и чтения из неё массива однобайтовых чисел. Но коэффициенты, полученные в результате калибровки нуля и полной шкалы, являются числами с плавающей запятой. Поэтому записать напрямую в память эти коэффициенты, пользуясь подпрограммами FLASH_Write и FLASH_Read, невозможно, т.к. число с плавающей запятой занимает 4 байта. Однако если использовать совмещение (в нотации C++ «union» – объединение) массива размерностью

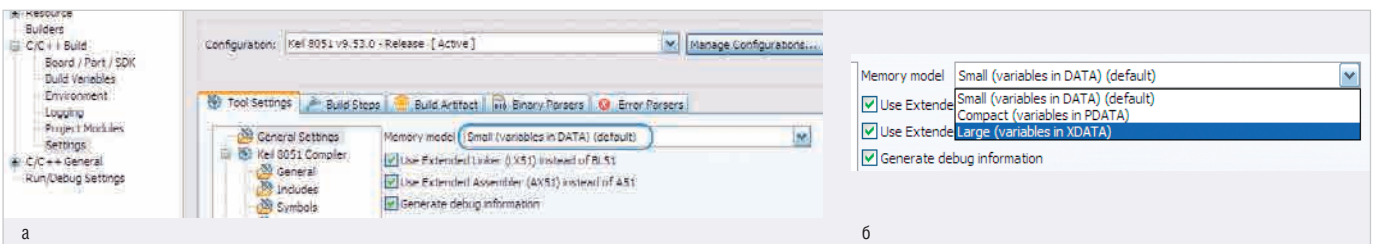

Рис. 14. Замена модели памяти со Small (а) на Large (б) для получения доступа к функции sprintf(), работающей с числами с плавающей запятой

Листинг 1

```
//-------------------------------------------------------------------
--------
union { // KWR.F - 4-х байтное float число - коэффициент K для записи.
 unsigned char B[4]; // KWR.B[0]- Самый ст.б.
 float F;            // KWR.B[1]- Ст.б.
 }KWR;               // KWR.B[2]- Ср.б.
                     // KWR.B[3]- Мл.б.
//-------------------------------------------------------------------
--------
 union { // KRD.F - 4-х байтное float число - коэффициент K для чтения.
 unsigned char B[4]; // KRD.B[0]- Самый ст.б.
 float F;            // KRD.B[1]- Ст.б.
 }KRD;               // KRD.B[2]- Ср.б.
                     // KRD.B[3]- Мл.б.
 //-------------------------------------------------------------------
--------
```

Листинг 2

```
//-------------------------------------------------------------------
----------
// Global Variables
//-------------------------------------------------------------------
----------
SI_LOCATED_VARIABLE_NO_INIT(flash_write_array[512], uint8_t, const
SI_SEG_CODE, START_ADDRESS_B);
SI_LOCATED_VARIABLE_NO_INIT(flash_write_array1[512], uint8_t, const
SI_SEG_CODE, START_ADDRESS_K);
SI_LOCATED_VARIABLE_NO_INIT(flash_write_array2[512], uint8_t, const
SI_SEG_CODE, START_ADDRESS_P);
//-------------------------------------------------------------------
----------
```

4 (например, B[4]) с числом с плавающей запятой (например, F), как показано в листинге 1, т.е. для записи использовать 4-байтный массив KWR.B, а для чтения – массив KRD.B, то эта проблема решается очень просто (см. листинг 1)

Здесь необходимо напомнить, что совмещение (union) двух разнотипных элементов (в данном случае – 4-байтного числа с плавающей запятой и 4-х байтного массива) означает, что эти элементы занимают один и тот же участок в памяти. Причём в этом случае число с плавающей запятой может быть полностью восстановлено по значениям массива и наоборот.

Пользуясь вышеописанными подпрограммами, перед записью 4-х байтного массива в память требуется очистить для них место (т.е. стереть как минимум 4 байта, начиная с того адреса, с которого будет производиться запись). Однако, как это ни странно, стереть 4 байта нельзя, а вот стереть целую страницу памяти размером 512 байт можно. Для этого используется подпрограмма стирания страницы памяти (FLASH_PageErase). Но при калибровке нуля и полной шкалы используются два разных коэффициента, причём при калибровке полной шкалы используется коэффициент, полученный при калибровке нуля. Поэтому адреса, по которым записываются эти коэффициенты, должны отличаться между собой по крайней мере на 512 байт, иначе, например при калибровке полной шкалы, можно стереть записанный в память коэффициент калибровки нуля (при использовании стирания страницы памяти FLASH_PageErase).

Необходимо отметить, что программы под общим названием EFM8LB1_Flash работают при частоте внутреннего генератора 24,5 МГц (HFOSC0). Кроме того, в справочном листке на микроконтроллер EFM8LB12 приведено время записи и стирания страницы памяти при условии работы микроконтроллера также при частоте 24,5 МГц. Но настоящая программа вольтметра использует внутренний генератор частотой 72 МГц (HFOSC1, см. рис. 10). Будет ли работать запись во флеш-память и чтение из неё на частоте 72 МГц? Для этого можно попробовать установить настройку времени чтения флеш-памяти (Flash Read Timing) в состояние «SYSCLK is below 75 MHz», т.е. чтобы системная тактовая частота была меньше 75 МГц. Этот параметр задаётся в основном меню, в разделе Core (см. рис. 8). В качестве проверки работоспособности такого подхода можно взять число с плавающей запятой, например 1,23456, записать его в память, затем прочитать и вывести на дисплей. Такой эксперимент был проведён, и он полностью подтвердил, что запись во флеш-память и чтение из неё на частоте 72 МГц работают безупречно.

В связи с этим адрес массива для записи/чтения коэффициента калибровки нуля выбран 1A00h (6656_{10}), адрес массива для записи/чтения коэффициента калибровки полной шкалы выбран (как в примере EFM8LB1_Flash) 1600h (5632_{10}), разница между ними составляет 6656 - 5632 = 1024, что заведомо больше 512 байт. Адрес числа с плавающей запятой для тестирования правильности записи/чтения выбран 1200h (4608_{10}), он на 1024 байт меньше адреса 1600h:

```
#define START_ADDRESS_B 0x1A00 //
для кал. 0
#define START_ADDRESS_K 0x1600 //
для кал. 1
#define START_ADDRESS_P 0x1200 //
для проверки
```

Для работы подпрограмм требуется завести три глобальные переменные, использующие вышеуказанные адреса (START_ADDRESS_B, START_ADDRESS_K и START_ADDRESS_P) (см. листинг 2).

После этого для записи во флеш-память, например, коэффициента калибровки полной шкалы необходимо использовать следующие два оператора:

```
FLASH_PageErase(START_
ADDRESS_K);//Стирание страницы
флеш-памяти 512 байт.
FLASH_Write(START_ADDRESS_K,
KWR.B, sizeof(KWR.B));//Запись
KWR.F во флеш-память.
```

Для чтения – оператор:

```
FLASH_Read(KRD.B, START_
ADDRESS_K, sizeof(KRD.B));
//Чтение KRD.F из флеш-памяти.
```

Аналогично в программе используется запись и чтение коэффициента калибровки нуля (это числа соответственно BWR.B и BRD.B), а также запись и чтение проверочного числа (PWR.F и PRD.F).

Текст основной программы для вольтметра и все дополнительные файлы, требующиеся для её работы, а также оттранслированная и готовая к загрузке программа в hex-формате приведены в дополнительных материалах к статье.

Запрограммировать микроконтроллер EFM8LB12, как указывалось выше, можно двумя способами.

По интерфейсу C2 с помощью USB-DEBUG адаптера. Для этого потребуется изготовить кабель, который одним концом подключается к выходному разъёму USB-DEBUG адаптера, а на втором его конце распаян 3-контактный разъём, на который выведены сигналы интерфейса C2: RST/C2CK, C2D и «земля». Схему такого

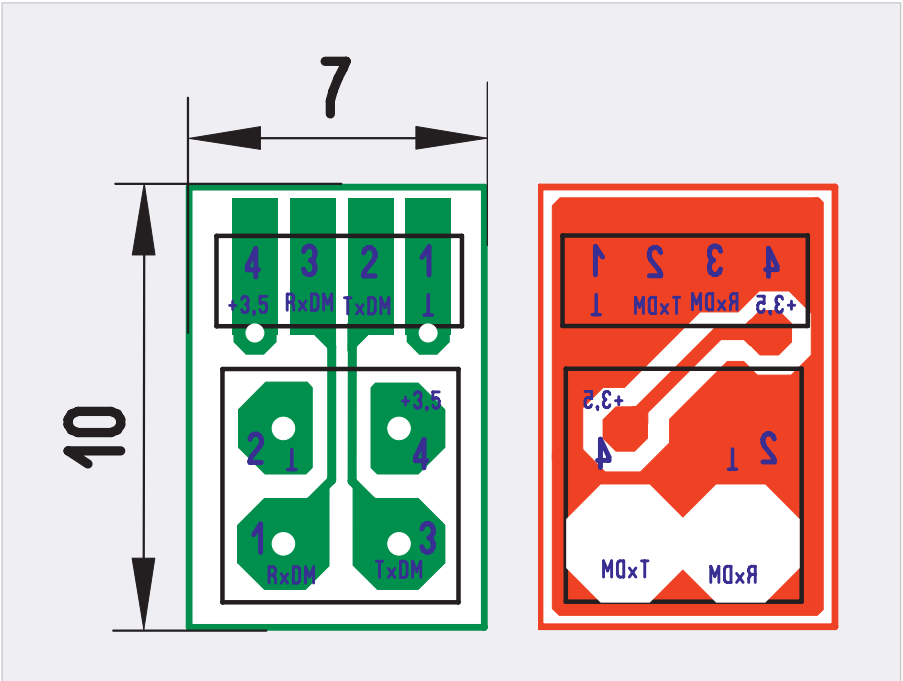

Рис. 15. Разводка платы-переходника для программирования микроконтроллера по интерфейсу RS-232

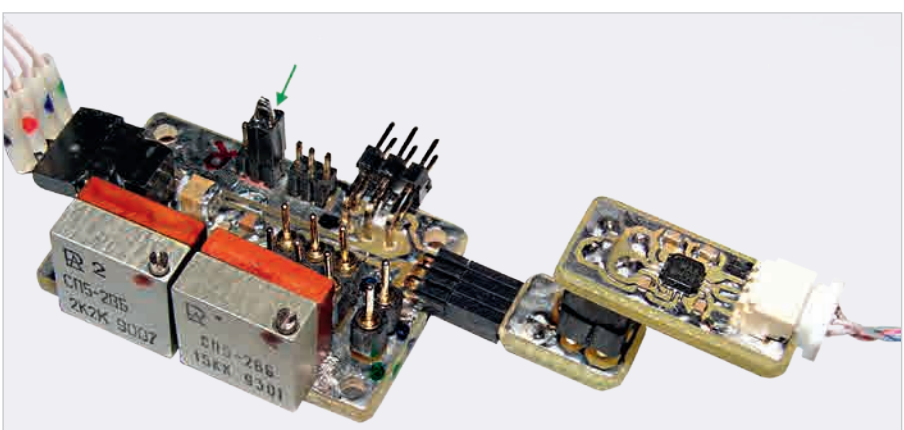

Рис. 16. Подключение микроконтроллера EFM8LB12 для программирования по интерфейсу RS-232 на плате вольтметра

кабеля можно найти в [1, 3]. На схемах на рисунках 1 и 4 для программирования по интерфейсу C2 предусмотрен разъём XB.

Второй способ заключается в программировании по интерфейсу RS-232 через COM-порт компьютера. Для этого потребуется изготовить одноканальный преобразователь уровней интерфейса RS-232 в TTL-уровни. Схему и разводку платы такого преобразователя можно найти в [3, 4]. Кроме того, потребуется изготовить плату-переходник (см. рис. 15), на одном конце которого расположен 4-контактный разъём PBS1.27-04 (4 гнезда с шагом 1,27 мм), а на втором – 4-контактный разъём PBDM-2*2 (4 цанговых гнезда с шагом 2,54 мм). Эта плата одним концом (разъёмом PBS1.27-04) подключается к разъёму XRS платы вольтметра (см. рис. 1, 4), а ко второму его концу (к разъёму PBDM-2*2) подключается плата одноканального преобразователя уровней RS-232 в уровни TTL (его можно заметить в правом нижнем углу на рисунке 16). При программировании в этом режиме на контакты 1 и 2 разъёма XB (см. рис. 1, 4) необходимо надеть перемычку, т.е. заземлить сигнал C2D (на рисунке 16 эта перемычка показана зелёной стрелкой).

Программные средства для программирования по интерфейсу RS-232 всех микроконтроллеров EFM8, а также подробное описание работы с ними можно найти в [4].

В следующей части статьи будут представлены разводка, изготовление печатных плат и фотографии готового вольтметра.

Литература

1. *Кузьминов А.* Повышение разрешающей способности АЦП микроконтроллера EFM8LB12. Современная электроника. 2018. № 8, 9.
2. *Кузьминов А.* Преобразователь интерфейсов USB-SPI на базе нового 51-совместимого микроконтроллера EFM8UB1. Современная электроника. 2017. № 1 – 3.
3. *Кузьминов А. Ю.* Связь между компьютером и микроконтроллером. Современные аппаратные и программные средства. – М.: «Перо». 2018.
4. *Кузьминов А.* Программирование микроконтроллеров EFM8 с помощью встроенного загрузчика программ. Радио. 2018. № 12.