Повышение разрешающей способности АЦП микроконтроллера EFM8LB12

Часть 2

Алексей Кузьминов (compmicrosys@mail.ru)

Во второй части статьи описаны программные средства сопряжения микроконтроллера EFM8LB12F64 с компьютером посредством интерфейса USB с помощью изолированных преобразователей USB-SPI на базе микроконтроллера EFM8UB10 и цифровых изоляторов ADUM3160 и SI8662, а также приведены результаты измерений, подтверждающие повышение разрешающей способности АЦП.

Программные средства

Программная часть проекта включает в себя программу для компьютера, обеспечивающую приём/передачу данных по интерфейсу USB; программу микроконтроллера UB10 для работы в преобразователе интерфейсов USB-SPI; программу микроконтроллера LB12 для работы с его АЦП и передачи данных по интерфейсу SPI.

Далее будут описаны основные принципы и алгоритмы работы программ (их исходные тексты и сами оттранслированные программы приведены в дополнительных материалах к статье на сайте www.soel.ru).

Сначала несколько слов о том, что представляет собой сумма 2048 чисел и как из этой суммы получить усреднённое значение. Шестнадцатеричное представление числа 2048_{10} есть число 0x800 (0x7FF+1), а максимальное значение суммы из 2048 двухбайтных чисел равно 0x7FFFFF. Для получения 19-разрядного кода, максимальное значение которого равно 0x7FFFF, число 0x7FFFFF необходимо сдвинуть вправо ровно на 1 байт, что соответствует осреднению по 256 значениям.

При этом число 0x7FFFF нельзя передать по однобайтному интерфейсу, каковым является SPI, поэтому в программе для микроконтроллера используется совмещение (union) одного длинного беззнакового целого числа типа unsigned long ADC0.UL длиной 4 байта с массивом из 4 однобайтных беззнаковых целых чисел типа unsigned char ADC0.BT[0]...ADC0.BT[3], где последний элемент содержит младший байт числа. Это совмещение распределяет одно и то же место в памяти микроконтроллера для хранения числа ADC0.UL и массива ADC0.BT[3] и не требует преобразования одного длинного целого в 4 однобайтных числа и обратно. С другой стороны, такой подход позволяет использовать при расчёте суммы 2048 чисел длинное целое число, а для передачи по интерфейсу SPI – 4 однобайтных числа. Таким образом, если ADC0.UL=0x7FFFFFF, то ADC0.BT[0]=0x7F, ADC0.BT[1]=0xFF, ADC0.BT[2]=0xFF, ADC0.BT[3]=0xFF.

Вместо того чтобы сдвигать всё число ADC0.UL вправо на 1 байт, достаточно передавать по интерфейсу SPI только ADC0.BT[0], ADC0.BT[1] и ADC0.BT[2], отбрасывая ADC0.BT[3], т.к. этот элемент при сдвиге обнуляется (теряется).

Число ADC0.UL=0x7ffff=524 287_{10}, а с учётом нулевого значения общее количество отсчётов будет на 1 больше, т.е. 524 288. Поэтому, для того чтобы получить напряжение в вольтах, необходимо умножить число ADC0.UL на V_{ref} и разделить на 524 288: U=ADC0.UL×V_{ref}/ /524 288=ADC0.UL×2,4/524 288 при V_{ref}= =2,4 В.

Все предыдущие рассуждения касались однократного суммирования, или, другими словами, однократного осреднения всех 2048 чисел при N=1, где N – количество осреднений. Если же произвести двукратное осреднение (N=2), т.е. получить два числа ADC0.UL, сложить их между собой и разделить на 2, то их сумма будет равна Summa=(ADC0.UL[1]+ADC0.UL[2])/2. Вместо деления на 2, число Summa можно сдвинуть вправо на 1 бит, при N=4 число нужно сдвинуть вправо на 2 бита, а при N=8 – на 3 бита. Именно на таком принципе построена программа для микроконтроллера F067, имеющего 16-разрядный АЦП [1]. В данном случае АЦП LB12 14-разрядный. Максимальное значение 14-разрядного числа равно 0x3FFF, что в 4 раза меньше максимального значения 16-разрядного кода (0xFFFF), поэтому для получения 19-битного результата (0x7FFFF) необходимо произвести 4-кратное осреднение (N=4). При 8-кратном осреднении результат необходимо сдвинуть вправо на 1 бит, при 16-кратном – на 2 бита, а при 32-кратном – на 3 бита. Однако при 2-кратном осреднении 19-разрядный результат может быть получен, если сумму сдвинуть на 1 бит влево, а при однократном – если сумму сдвинуть влево уже на 2 бита. На таком принципе и построена программа для LB12. Передача 3 байт ADC0.BT[0]...ADC0.BT[2] по интерфейсу SPI организована аналогично F067, как описано ранее.

Оценим время, требующееся микроконтроллеру LB12 для однократного осреднения. Как будет показано далее, время одного аналого-цифрового преобразования и автоматической записи в память XRAM его результата для LB12 при тактовой частоте 72 МГц составляет 1,11 мкс (частота 900 кГц). Если производится 2048 преобразований, то такая процедура потребует времени 1,11×2048≈2,27 мс.

Суммирование двух четырёхбайтных чисел, как было показано в [1], занимает не менее 30 тактов, а суммирование 2048 чисел потребует 30×2048= =61 440 тактов. При тактовой частоте 72 МГц время суммирования составит 61 440/72 000 000=0,85 мс. Таким образом, общее время составит 2,27+ +0,85=3,12 мс. Если предполагается проводить измерения по 8 каналам, то это время будет в 8 раз больше и составит 0,025 с. В реальных условиях это время чуть меньше и составляет около 0,022 с (см. далее) при N=1. Таким образом, интервал времени измерения Δt можно задавать со следующей дискретностью: 0,025; 0,05; 0,1; 0,2; 0,5 и 1 с. Другими словами, задав интервалы времени с такой дискретностью, можно быть уверенным, что LB12 успеет произвести аналого-цифровые преобразования и, соответственно, 1, 2, 4, 8, 16 и 32 осреднения по 8 каналам. В программе для F067 данные интервалы составляли 0,1; 0,2; 0,5 и 1 с [1].

В программах для F067 указанные интервалы времени Δt задавались с помощью таймеров (T0 и таймера массива счётчиков PCA). На монитор компьютера выводились результаты АЦП через каждый интервал времени Δt по 8 каналам, т.е. строка, состоящая из 8 шестизначных чисел с запятой после первой цифры (например, 1,23456 В). В программах для LB12 использовалась более простая процедура без использования таймеров, позволяющая оценить время, соответствующее 1, 2, 4, 8, 16 или 32 осреднениям. Вместо однократного получения и вывода 8 шестизначных чисел производился десятикратный цикл, т.е. на монитор выводились 10 строк с результатами по 8 шестизначных чисел в строке. Перед выводом на монитор в программе на компьютере включался программный таймер. После полного вывода 10 строк таймер выключался. Для нахождения Δt (в однократном цикле) время по таймеру (T) просто делилось на 10. Если, например, время T составляло 0,22 с, то Δt = 0,022 с. Использование 10-кратного цикла даёт двойное преимущество по сравнению с однократным. Во-первых, выборка, состоящая из 10 строк по 8 чисел является более репрезентативной, чем выборка из 8 чисел. Во-вторых, 10-кратный цикл увеличивает время T в 10 раз, что позволяет более точно определить интервал Δt, подсчитанный как среднее (T/10).

Перейдём непосредственно к описанию программ (программа для компьютера приведена в дополнительных материалах к статье на сайте www.soel.ru: исходный текст – In8U.clw, исполняемый файл – In8U.exe, файл проекта – In8U.prj). При запуске программы на экран монитора выводится окно, в верхней левой части которого расположено поле ввода количества осреднений N, а внизу – две кнопки: «Выход» и «Запуск/Продолжить». Пользователь вводит число N, нажимает кнопку «Запуск/Продолжить», и через некоторое время на экран выводятся результаты. Эта программа передаёт по интерфейсу USB массив из 32 однобайтных элементов (A[32]), первый из которых содержит число N (A[1]=N), а остальные зарезервированы для пользовательских целей. Следует обратить внимание, что индексация массива на языке Clarion начинается с 1. Далее в 10-кратном цикле программа принимает массив из 32 однобайтных элементов, первые 24 из которых содержат информацию о напряжении, измеренном АЦП LB12 по 8 каналам (т.е. по 3 байта на канал), а остальные байты зарезервированы. Программа использует функции API обращения к интерфейсу USB из пакета USBXpress, бесплатно поставляемого компанией Silicon Laboratories. Применение этих функции подробно описано в [2, 3] в программе TestUSB. Расчёт напряжений и вывод их на экран монитора производится точно так же, как и в программе In8U, подробно описанной в [1].

Программа для микроконтроллера UB10 приводится в виде файла с исходным текстом USBXpress_Echo_main.c и оттранслированного файла USBXpress_Echo_12.hex. Данная программа принимает массив из 32 однобайтных элементов по интерфейсу USB и передаёт этот массив по интерфейсу SPI. Затем в 10-кратном цикле принимает массив из 32 однобайтных элементов по интерфейсу SPI и передаёт этот массив по интерфейсу USB. Для программирования обращений к интерфейсу USB со стороны микроконтроллера UB10 программа также использует функции API из библиотеки USBXpress. Использование этих функций подробно освещено в описании программы USBXpress_Echo [2, 3].

Программа для микроконтроллера LB12 приводится в виде файла с исходным текстом EFM8LB1_ADC_Lib_Autoscan_Large_Buffer.c и оттранслированного файла EFM8LB1_ADC_Lib_Autoscan_Large_Buffer_4.hex. Данная программа принимает по интерфейсу SPI массив из 32 однобайтных элементов A[32]. В первом элементе этого массива содержится число осреднений A[0]=N (индексация массивов в языке C начинается с 0). Затем производится 2048-кратное аналого-цифровое преобразование напряжения, поданного на вход АЦП, и заполнение памяти XRAM двухбайтными словами 2048 раз, т.е. всего 4096 байт. Далее из памяти извлекаются и суммируются все двухбайтные слова, и результат помещается в переменную AD0.UL типа unsigned long. После этого, в зависимости от числа осреднений N, производится N-кратное повторение измерений, суммируются все AD0.UL по каждому измерению, и результат помещается в переменную ADCSR типа unsigned long. Затем для получения 19-разрядного кода ADCSR сдвигается вправо или влево в зависимости от N (см. ранее). Измерения производятся в 8-кратном цикле (как для 8 каналов). После этого формируется массив из 32 однобайтных элементов, первые 24 элемента (по 3 байта на 8 каналов) которых несут полезную информацию, а остальные зарезервированы. Этот массив передаётся по интерфейсу SPI, работающему в 4-проводном ведущем режиме. Измерения и вывод массива повторяются 10 раз.

Программа написана на основе демонстрационной программы EFM8LB1_ADC_Lib_Autoscan_Large_Buffer.c от Silicon Laboratories. Она также заполняет буфер из 4096 байт в памяти результатами измерений АЦП, производит осреднение результатов измерения, вычисление напряжения, но результат передаёт по интерфейсу RS-232. Кроме того, она использует выводы микроконтроллера для зажигания светодиодов. В настоящей программе коммуникация по RS-232 заменена на коммуникацию по SPI, удалены обращения к вычислениям напряжения и зажиганиям светодиодов. Кроме того, существенно изменены настройка работы АЦП, настройка системной тактовой частоты работы микроконтроллера, его процессора и др.

Далее будет описана процедура настройки микроконтроллера с помощью конфигуратора в среде Simplicity Studio V3, бесплатно поставляемой компанией Silicon Laboratories. В результате данной процедуры в автоматическом режиме будет сформирован файл инициализации устройства InitDevice.c. Затем будет показано, какие дополнительные команды необходимо вставить в подпрограммы для коммуникации по интерфейсу SPI, чтобы обмен данными по SPI работал корректно.

В настоящее время компанией Silicon Laboratories разработана новая версия Simplicity Studio V4, которая имеет много новшеств по сравнению с Simplicity Studio V3, однако в ОС Windows XP некоторые функции Simplicity Studio V4 работают некорректно. Дальнейшее изложение будет вестись для Simplicity Studio V3.

Настройка микроконтроллера

После запуска среды Simplicity Studio необходимо в проекте EFM8LB1_ADC_Lib_Autoscan_Large_Buffer_4 выбрать конфигурационный файл EFM8LB1_ADC_Lib_Autoscan_Large_Buffer.hwconf. После этого на экран будет выведено окно DefaultMode Peripherals, отражающее все периферийные устройства

Рис. 6. Периферийные устройства LB12

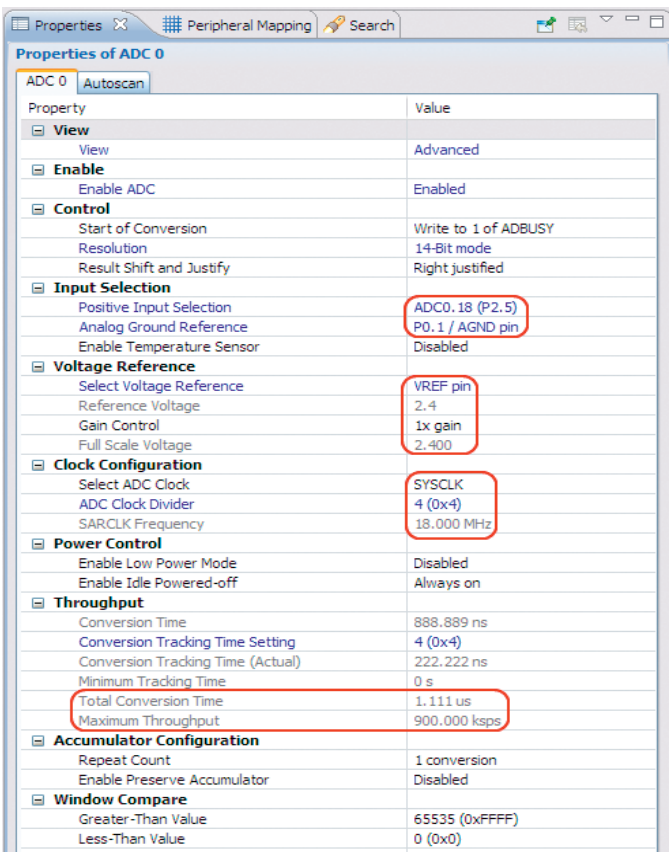


Рис. 7. Параметры АЦП

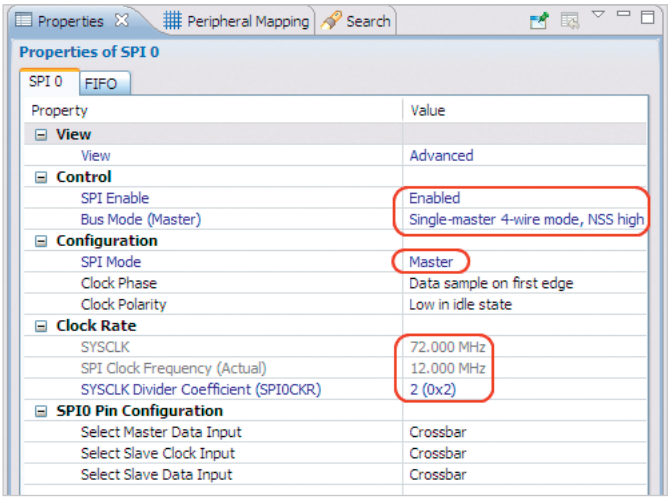


Рис. 8. Свойства SPI

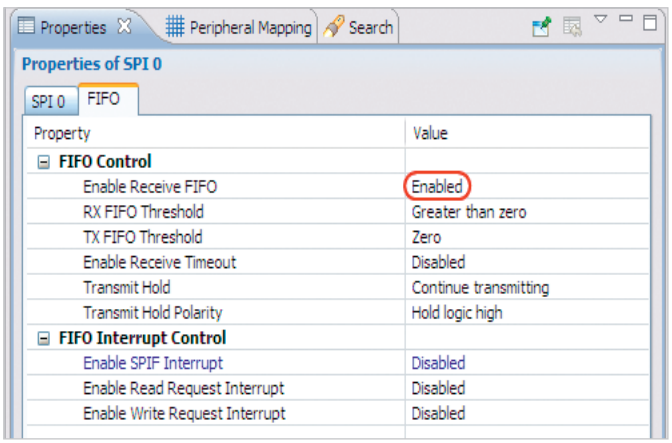


Рис. 9. Свойства FIFO SPI

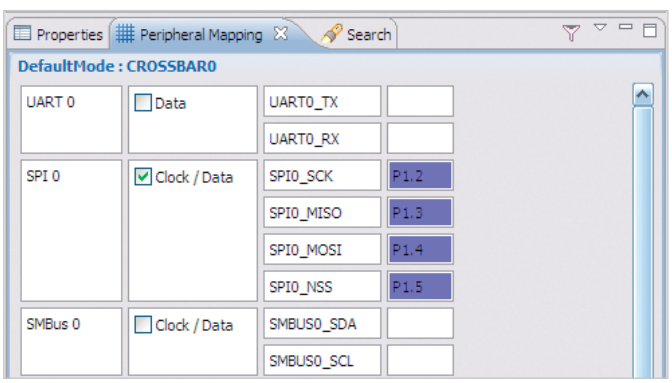


Рис. 10. Разрешение SPI в матрице соединений

LB12 (см. рис. 6). Основные устройства, работу которых необходимо настроить, показаны красными галочками.

Сначала необходимо настроить параметры внешнего генератора, нажав кнопку EXTOSC0. На экран будет выведено окно, в котором необходимо выбрать режим CMOS и установить частоту 72 МГц. После этого следует нажать кнопку Clock Control (см. рис. 6) и в открывшемся справа окне выбрать External Oscillator и SYSCLK/1, в результате чего системная частота установится равной 72 МГц.

Затем необходимо выбрать параметры работы ядра процессора, нажав кнопку Core (см. рис. 6), и в открывшемся справа окне выбрать параметр SYSCLK is below 75 MHz. Вместо внешнего генератора можно выбрать внутренний, тоже с частотой 72 МГц. Для этого необходимо нажать кнопку Clock Control (см. рис. 6) и в открывшемся окне выбрать Internal High Frequency Oscillator 1 и SYSCLK/1, в результате чего системная частота также установится равной 72 МГц. В обоих случаях программа будет работать одинаково корректно. Внешний генератор следует выбирать, если требуется измерять или задавать точные интервалы времени с помощью таймеров, поскольку точности внутреннего генератора для этих целей недостаточно.

Для настройки параметров работы АЦП и его опорного напряжения необходимо нажать кнопку Voltage Reference (см. рис. 6) и в открывшемся окне установить Internal 2.4V (output to VREF pin). Далее следует нажать кнопку ADC0 (см. рис. 6) и в открывшемся окне выбрать параметры АЦП в соот-

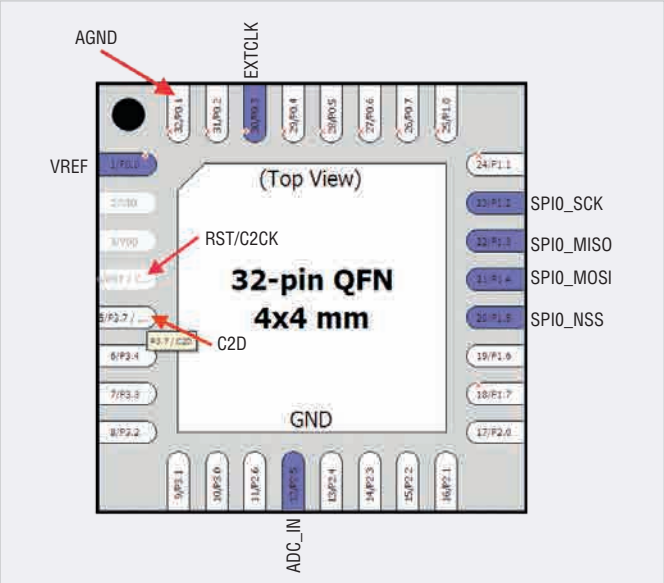


Рис. 11. Порты LB12

Листинг 1

```
//---------------------------------
void outspi(unsigned char byte) {
//---------------------------------
       SPI0DAT = byte; // Вывод байта по SPI
       while (!SPI0CN0_SPIF); // Ожидание окончания
вывода байта
       SPI0CN0_SPIF = 0; // Сброс флага окончания
передачи
       DEL20NS(); // Задержка для гарантированного
сброса флага SPIF
}
//---------------------------------
```

Листинг 2

```
//---------------------------------
unsigned char inspi() {
//---------------------------------
       unsigned char byte;
       SPI0DAT = 0x0f; // Ввод байта в микрокон-
троллер
       while (!SPI0CN0_SPIF); // Ожидание окончания
ввода байта
       SPI0CN0_SPIF = 0; // Сброс флага окончания
приёма
       byte = SPI0DAT;
       DEL20NS(); // Задержка для гарантированного
сброса флага SPIF.
       return (byte);
}
//---------------------------------
```

ветствии с рисунком 7, уделив особое внимание параметрам, выделенным красным.

Для настройки работы интерфейса SPI необходимо нажать кнопку SPI0 (см. рис. 6) и в открывшемся справа окне выбрать параметры SPI, уделив особое внимание параметрам, выделенным красным (см. рис. 8). Затем в этом окне следует выбрать вкладку FIFO и в открывшемся справа окне установить параметры в соответствии с рисунком 9. Далее следует разрешить работу SPI в матрице соединений, для чего в правом верхнем углу меню нужно нажать на «решётку» (CROSSBAR0) и в открывшемся справа окне (см. рис. 10) выбрать пункт Clock/Data. В этом случае порты (SPI) P1.2...P1.5 будут отмечены синим цветом.

Далее необходимо настроить работу портов LB12. Для этого в нижней части меню следует переключить режим, нажав кнопку Default Mode Port I/O. На экране будет отображён рисунок микросхемы LB12 с её портами (см. рис. 11). С помощью команды Skip необходимо «передвинуть» сигналы SPI (SPI0_SCK, SPI0_MISO, SPI0_MOSI и SPI0_NSS) в правую часть микросхемы. Далее следует установить для портов P1.2 (SPI0_SCK), P1.4 (SPI0_MOSI) и P1.5 (SPI_NSS) режим Digital Push-Pull Output, для порта P1.3 (SPI0_MISO) – Digital OpenDrain I/O, для порта P0.3 (EXTCLK) – Digital OpenDrain I/O, для портов P0.0 (VREF), P0.1 (AGND) и P2.5 (ADC_IN) – Analog I/O. После выполнения всех перечисленных операций сконфигурированный режим работы устройства следует записать на диск, в результате чего будет сгенерирован файл InitDevice.c.

Рассмотрим подробнее подпрограммы приёма/передачи по интерфейсу SPI, т.к. при их написании и использовании был обнаружен ряд проблем. В описании микроконтроллера EFM8LB12 указано, что в SPI добавлен новый режим FIFO глубиной до 4 байт. При применении режима FIFO, вместо стандартных процедур ввода/вывода по SPI и, в частности, использования флага SPIF (SPI0CN0_SPIF) для контроля ввода/вывода байта, должны применяться специальные процедуры. Вместе с тем в описании сказано, что если не использовать FIFO, то должны применяться стандартные процедуры ввода/вывода по SPI и, в частности, должен использоваться флаг SPIF для контроля приёма/передачи. При этом отмечается, что после команды сброса флага SPIF (SPI0CN0_SPIF=0;) для её гарантированного выполнения должен пройти как минимум один машинный цикл. Один машинный цикл может быть задан, например, командой _nop_ () (пустая операция). Тогда, в соответствии с описанием, если не использовать FIFO, его необходимо запретить, т.е. в строке Enable Recive FIFO установить опцию Disabled вместо Enabled (см. рис. 9). Кроме того, в стандартных подпрограммах вывода байта (см. листинг 1) и ввода байта (см. листинг 2) после команды сброса флага SPIF необходимо ввести оператор _nop_(). Однако в такой конфигурации обмен по SPI работать не будет. Если же разрешить FIFO, то обмен данными заработает, но первые несколько байт будут прочитаны неправильно. Не помогает и многократное включение команды _nop_(). Проблема может быть решена, если после команды сброса флага SPIF обратиться к подпрограмме задержки DEL20NS() (см. листинг 3), представляющей собой два пустых цикла (`for (i = 0; i < 2; i++) {}`) при использовании генератора на 72 МГц. При использовании генератора на 50 МГц количество пустых циклов можно уменьшить до одного (`for (i = 0; i < 1; i++) {}`).

Листинг 3

```
//---------------------------------
void DEL20NS() {
//      _nop_();
       unsigned char i;
       for (i = 0; i < 2; i++) {}
}
//---------------------------------
```

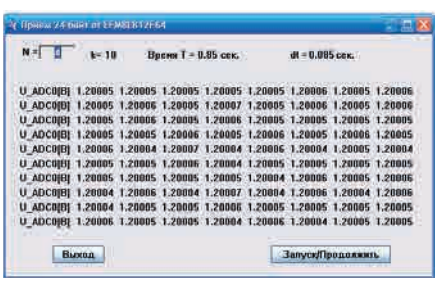


Рис. 12. Окно программы с результатами при *N*=4

Результаты

Результаты работы программы измерений напряжений при *N*=1, 2, 4, 8, 16 и 32 приведены в дополнительных материалах к статье на сайте www.soel.ru. На рисунке 12 представлены результаты измерений при *N*=4. Как видно из рисунка, 5-й знак после запятой изменяется от 4 до 7.

Результаты работы АЦП LB12 при различных значениях *N*

Режим измерений	Номер канала							
	1	2	3	4	5	6	7	8
	U, В							
N=1, Δ*t*=0,022 с	1,20007	1,20003	1,20006	1,20007	1,20004	1,20003	1,20002	1,20005
N=2, Δ*t*=0,042 с	1,20004	1,20007	1,20007	1,20004	1,20005	1,20008	1,20005	1,20002
N=4, Δ*t*=0,085 с	1,20006	1,20004	1,20007	1,20004	1,20006	1,20004	1,20005	1,20004
N=8, Δ*t*=0,167 с	1,20005	1,20005	1,20005	1,20005	1,20005	1,20004	1,20005	1,20005
N=16, Δ*t*=0,334 с	1,20005	1,20005	1,20005	1,20005	1,20005	1,20005	1,20005	1,20005
N=32, Δ*t*=0,667 с	1,20005	1,20005	1,20005	1,20005	1,20005	1,20005	1,20005	1,20005

В таблице приведены результаты, взятые из 5-й строки каждой из 6 серий измерений. С увеличением *N* прослеживается явная тенденция к уменьшению разброса значений напряжений. Если при *N*=8 в 4-й строке присутствует только одно значение (1,20004 В), отличающееся от остальных (1,20005 В), то в следующих двух строках 5-й знак после запятой уже не меняется. Это позволяет утверждать, что максимальная точность результата достигается уже при *N*=16 и осреднения при *N*=32 проводить не имеет смысла. Поскольку при *N*=16 Δ*t*=0,334 с, то, выбрав интервал времени, через который необходимо производить измерения Δ*t*=0,4 с, можно быть уверенным, что АЦП LB12 успеет произвести преобразования с максимальной точностью. Другими словами, максимальная точность результата может быть достигнута при Δ*t* не более 0,4 с.

Представляет интерес сравнение результатов АЦП LB12 с результатами АЦП F067 [1]. 5-й знак после запятой для F067 не меняется при Δ*t*=0,5 с, а для LB12 этот интервал времени Δ*t* не превышает 0,4 с. Кроме того, минимальный интервал времени Δ*t* при измерениях F067 составляет 0,1 с, в то время как минимальный интервал времени для LB12 составляет 0,022 с при *N*=1. Другими словами, LB12 позволяет использовать интервалы времени Δ*t*=0,025 с и Δ*t*=0,05 с, а F067 такой возможностью не обладает. Из сравнения LB12 и F067 следует явное преимущество первого перед вторым по скорости измерений, тогда как по точности измерений они друг от друга не отличаются.

Здесь необходимо отметить, что точность до 5-го знака после запятой в большинстве случаев является избыточной. Даже цифровые вольтметры и мультиметры, измеряющие напряжение с точностью до 4-го знака после запятой, сто́ят довольно дорого, а цена приборов с разрешением до 5-го знака после запятой может достигать нескольких тысяч долларов. Стоимость же LB12 составляет не более $2.

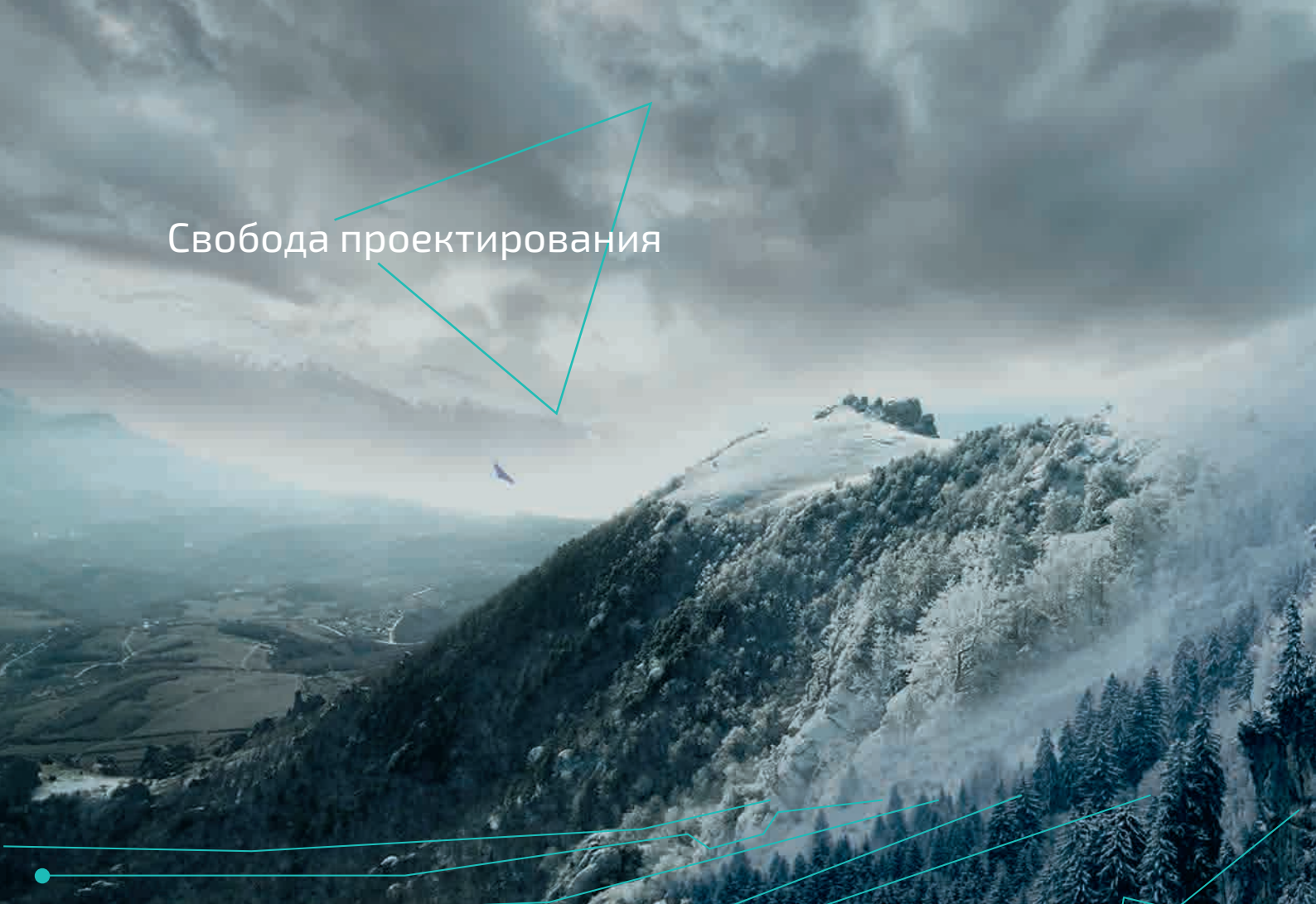
Как известно, SAR-АЦП имеют несколько строго детерминированных типов погрешностей: интегральная нелинейность, погрешность нуля и полной шкалы, погрешность каждого канала. Кроме того, для предотвращения перенапряжений на входах АЦП перед ними в обязательном порядке необходимо устанавливать микросхемы защиты, например MAX4507, одновременно предохраняющую от перенапряжений 8 каналов. Сопротивления каждого канала таких микросхем также несколько отличаются между собой. Детерминированность всех этих погрешностей легко компенсируется вводом специальных калибровочных коэффициентов для каждого канала, и впоследствии в программе для компьютера каждое из измеренных напряжений по конкретному каналу компенсируется путём умножения на соответствующий калибровочный коэффициент. Умножение в компьютере занимает ничтожно малое время и не оказывает никакого влияния на скорость вывода результатов. Опыт показывает, что однажды введённые коэффициенты не изменяются в течение достаточно долгого времени, исчисляемого месяцами и годами. В ΔΣ-АЦП, например в ADS1210, калибровку нуля и полной шкалы приходится проводить перед каждым измерением. В таких АЦП перед измерением также требуется установка цифрового фильтра, на что тратится дополнительное время. Помимо прочего, достаточно высокая стоимость ΔΣ-АЦП не выдерживает никакой конкуренции с SAR-АЦП.

Заключение

Результаты, приведённые в статье, свидетельствуют о том, что метод передискретизации и осреднения, применённый для АЦП микроконтроллера EFM8LB12F64, позволяет использовать этот микроконтроллер вместо C8051F067 в самых различных средствах измерения напряжений, в том числе в компьютерно-микроконтроллерных системах сбора и обработки измерительной информации. Применение микроконтроллера EFM8LB12F64 в подобных системах позволит существенно улучшить их скоростные характеристики и надёжность, значительно снизить стоимость и тем самым повысить их конкурентоспособность.

Литература

1. *Кузьминов А.* Как заставить встроенный в микроконтроллер АЦП поразрядного уравновешивания работать с разрешением дельта-сигма-АЦП. Современная электроника. 2012. № 3.
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.* Изготовление устройств на печатных платах с высоким разрешением в домашних условиях. Технологии в электронной промышленности. 2010. № 8–10. 2011. № 1, 2.
5. *Кузьминов А.* Технология изготовления печатных плат с высоким разрешением в любительских условиях. Радио. 2017. № 10.
6. Improving ADC Resolution by Oversampling and Averaging. AN118: www.silabs.com.



DeltaDesign

САПР электроники

В состав Delta Design, обеспечивающей сквозной цикл проектирования печатных плат, входят модули:

- Менеджер библиотек
- Схемотехнический редактор
- Схемотехническое моделирование
- HDL-симулятор
- Редактор правил
- Редактор печатных плат
- Топологический редактор плат TopoR
- Коллективная работа для предприятий

EREMEX

www.eremex.ru

+7 (495) 232-1864 / sales@eremex.ru