Цифровая фильтрация на микроконтроллере AVR
Часть 2

Вадим Баранов (Украина, Харьков)

В первой части было рассказано о теории рекурсивного цифрового фильтра, описаны способы моделирования его работы, а также приведена принципиальная электрическая схема цифрового фильтра на базе микроконтроллера AVR. Вторая часть статьи описывает разработку и отладку программы цифровой фильтрации, написанной на ассемблере.

Цифровой фильтр на микроконтроллере: программа на ассемблере

Для проверки описываемого далее программного цифрового фильтра была использована плата 12-разрядного АЦП, описание которого можно найти на сайте [1]. Упрощённая структурная схема устройства приведена на рисунке 7.

Режим работы устройства определяется командами, передаваемыми компьютером по USB. Для проверки программного цифрового фильтра использовался режим запуска преобразования АЦП по прерыванию Таймера 0 микроконтроллера.

После соответствующей команды компьютера микроконтроллер устанавливает на линии Clk уровень логической 1. При обнаружении фронта сигнала на линии Clk АЦП начинает преобразование входного аналогового сигнала. Микроконтроллер удерживает уровень логической 1 на линии Clk в течение времени, достаточного для выполнения преобразования АЦП. Затем микроконтроллер подаёт на линию Clk уровень логического 0 и считывает полученные в результате преобразования данные по 12-разрядной шине данных ШД1.

В микроконтроллере выполняется программная цифровая фильтрация принятых данных, а полученный результат передаётся побайтно по 8-разрядной шине данных ШД2 во внешнюю память. На 19-разрядной шине адреса ША формируется адрес ячейки памяти, в которую будет записан младший или старший байт результата при появлении уровня логического 0 на линии Wr.

Такая процедура «преобразования – фильтрации – записи в память» повторяется до заполнения памяти, после чего микроконтроллер передаёт в компьютер по USB сообщение о завершении работы. По запросам компьютера сохранённый в памяти массив отфильтрованных данных передаётся небольшими блоками в микроконтроллер и далее по USB – в компьютер. Извлечение данных из памяти выполняется побайтно. Для этого на шине адреса ША формируется адрес ячейки памяти, из которой будет считан очередной байт результата.

Компьютерное приложение отображает полученные данные в виде графика.

Рис. 7. Структурная схема цифрового фильтра на AVR: ШД1, ШД2 – шина данных 1, шина данных 2; ША – адресная шина; Clk – тактовый сигнал для АЦП; Rd – линия управления считыванием внешней памяти; Wr – линия управления записью во внешнюю оперативную память

На вход такого устройства с программным фильтром можно подавать сигналы с одинаковой амплитудой, но разной частоты, и записывать отфильтрованные сигналы. Записанные сигналы разных частот можно просмотреть на графике и зафиксировать амплитуду отфильтрованного сигнала для каждой из частот.

По полученным амплитудам записанных выходных сигналов можно построить АЧХ программного цифрового фильтра.

В листинге 3 (исходный текст программы содержится в архиве, который доступен для загрузки на сайте журнала www.soel.ru) приведена сокращённая программа устройства, позволяющая выполнить отладку подпрограммы цифровой фильтрации. Сама подпрограмма находится в файле, подключаемом директивой `.include ClcFlt.asm` в последней строке листинга 3, и будет рассмотрена позднее.

Выполнение программы начинается с метки `RESET`. После инициализации сторожевого таймера, портов ввода-вывода и стека вызывается подпрограмма записи коэффициентов фильтра `StScales`. Подпрограмма, помещающая коэффициенты цифрового фильтра в оперативную память микроконтроллера, находится в подключаемом файле ClcFlt.asm.

Установка разряда SRE в регистре `MCUCR` разрешает работу микроконтроллера с внешней оперативной памятью, а запись в регистры `XMCRA` и `XMCRB` определяет режим работы микроконтроллера при обращении к внешней памяти. Выбран режим без дополнительного времени ожидания с обращением ко всему блоку памяти в 64 К.

После разрешения прерываний командой `sei` вызывается подпрограмма `ADCTimer`, запускающая преобразование входного сигнала в код, цифровую фильтрацию кода и запись полученных значений во внешнюю оперативную память до её заполнения с переходом в бесконечный цикл (мет-

ка `cycle`) по завершении выполнения подпрограммы.

В полной программе подпрограмма `ADCTimer` вызывается по команде, полученной от компьютера, а по её завершении записанные данные передаются в компьютер.

Подпрограмма ADCTimer

Так как по линии CLK микроконтроллер передаёт в АЦП тактовые импульсы, соответствующая линия микроконтроллера переводится в режим передачи, на ней устанавливается уровень логического 0.

В переменную `Ust` помещается значение, соответствующее режиму записи stWr. В пару регистров `YH:YL` помещается адрес, начиная с которого во внешнем ОЗУ будут размещаться отфильтрованные данные.

В регистр `CLKclr` помещается состояние, в котором должен находиться PORTE при передаче микроконтроллером уровня логического 0 по линии CLK, в регистр `CLKset` – состояние для PORTE при передаче уровня логической 1 по линии CLK.

В регистр `T0rpt` помещается значение, определяющее счёт Таймера 0 до следующего переполнения.

В регистры `TCNT0`, `TCCR0` и `TIMSK` передаются значения, инициализирующие работу Таймера 0 и его переполнение, повторяющееся с частотой 1024 Гц (для этого в обработчике прерывания Таймера 0 нужно снова записывать в регистр `TCNT0` значение, хранящееся в регистре `T0rpt`).

Микроконтроллер переходит в режим ожидания окончания преобразования и записи 30 000 выборок (60 000 байт) во внешнее ОЗУ, о чём будет свидетельствовать обнуление переменной `Ust`. Пока же в этой переменной состояние, отличное от нуля, выполняется возврат к метке `Wr1`, а Таймер 0 продолжает работу.

По окончании преобразования, фильтрации и записи данных Таймер 0 останавливается, а в рабочей программе дополнительно вызывается подпрограмма, передающая в компьютер сообщение об окончании записи (в упомянутом листинге 3 она отсутствует).

Обработчик прерывания Таймера 0

В начинающемся меткой `OVF0` обработчике прерывания переполнения Таймера 0 командой `wdr` выполняется сброс сторожевого таймера.

Поскольку промежуточные результаты при вычислениях, связанных с цифровой фильтрацией, будут храниться во внутреннем ОЗУ микроконтроллера, очищается регистр `MCUCR`, в частности определяющий, с каким ОЗУ (внутренним или внешним) работает микроконтроллер.

В счётчик Таймера 0 помещается содержимое регистра `T0rpt`, что обеспечит следующее прерывание переполнения Таймера 0 через 1/1024 с.

В PORTE выводится содержимое регистра `CLKset`, обеспечивающее уровень логической 1 на линии CLK. При таком состоянии линии данные могут быть считаны из микросхемы АЦП.

Двенадцать линий данных АЦП соединены с линиями портов E и F микроконтроллера. В регистр `r1` младший байт результата преобразования АЦП считывается через порт E, в регистр `r2` считывается старший полубайт через порт F.

После считывания в PORTE передаётся содержимое регистра `CLKclr`, что вызывает появление уровня логического 0 на линии CLK и начало нового преобразования АЦП.

Полученное значение выборки входного сигнала размещается во внутренней памяти микроконтроллера в паре ячеек с адресами `ax4` и `ax4+1`.

Подпрограмма clcFlt выполняет фильтрацию входной выборки, вычисляя значение выходной выборки. Выходная выборка, сохранённая подпрограммой в паре ячеек внутреннего ОЗУ микроконтроллера, передаётся в переменные `c2` и `c1`.

В регистр `MCUCR` помещается значение `1<<SRE`, переводящее микроконтроллер в режим работы с внешним ОЗУ, а значения выходной выборки, скопированные в переменные `c2:c1`, сохраняются в ячейках внешнего ОЗУ по адресам, определяемым содержимым пары регистров `YH:YL`.

Командой `adiw YL,1` к содержимому пары регистров `YH:YL` добавляется единица. Когда последняя ячейка 64-килобайтной страницы внешнего ОЗУ будет заполнена, регистры `YH:YL` будут содержать значение `$ffff` – адрес последней ячейки внешнего ОЗУ. Добавление единицы приведёт к обнулению регистров `YH:YL`, что вызовет очистку переменной Ust перед выходом из обработчика прерывания, чтобы сигнализировать о заполнении памяти данными. В противном случае произойдёт переход к метке `ovf2` и возврат из обработчика прерывания в подпрограмму `ADCTimer`.

Подпрограмма цифровой фильтрации

Листинг 4 (исходный текст программы содержится в архиве, который доступен для загрузки на сайте журнала www.soel.ru) включает в себя подпрограмму `StScales`, размещающую коэффициенты фильтра во внутреннем ОЗУ микроконтроллера, и подпрограмму цифровой фильтрации `clcFlt`.

В начале листинга в виде комментария приведены формулы для вычисления значения выходной выборки с некоторыми упрощениями, рассмотренными при разборе приложения, написанного в Delphi.

Здесь же определяется необходимое число байт для размещения результата суммирования входных выборок x0...x4, исходя из того, что максимальное значение входной выборки при использовании 12-разрядного АЦП составляет `$0fff` или `4095`. Сумма же `x0 + x4 + 4 * x1 + 4 * x3 + 6 * x2` не превысит величины `$0fff + $0fff + + 4 * $0fff + 4 * $0fff + 6 * $0fff`, что составит `$fff0`. То есть, для хранения суммы входных выборок с коэффициентами достаточно двух байт.

Округлённые коэффициенты `Ky0`…`Ky3` во время вычислений удобно хранить не во Flash-памяти, а в оперативной памяти микроконтроллера, это сократит время, необходимое для их извлечения.

Модуль коэффициента `Ky0`, равный `$00CD61`, помещается подпрограммой StScales в три ячейки внутреннего ОЗУ микроконтроллера, начиная с ячейки по адресу aKy0. Аналогично размещаются коэффициенты `Ky1`…`Ky3`.

Подпрограмма также размещает начальные значения пяти входных и пяти выходных выборок в ОЗУ микроконтроллера. Начальные значения входных выборок соответствуют половине шкалы АЦП (`$07ff` или `2047`), для выходных выборок – это половина шкалы АЦП, умноженная на коэффициент Gain.

При вычислениях, необходимых для выполнения цифровой фильтрации, вместо деления на коэффициент Gain, равный 40929, будем выполнять деление на 65536, что эквивалентно сдвигу результата вправо 16 раз или отбрасыванию двух младших байт. И хотя такая замена в 1,6 раза уменьшит величину выходного сигнала, она позволит значительно уменьшить время вычисления за счёт отказа от громоздкой операции деления. Уменьше-

ние выходного сигнала может быть скомпенсировано, например, увеличением коэффициента усиления в аналоговых цепях, если за микроконтроллером следуют ЦАП и усилитель, или введением дополнительного коэффициента при построении графика, как в нашем случае.

Итак, начальные значения выходных выборок составят `$7ff * $10000` или `$07ff0000`.

Подпрограмма ClcFlt

Подпрограмма `ClcFlt` находится в конце листинга 4 и вызывается обработчиком прерывания Таймера 0 при поступлении очередной входной выборки.

Подпрограмма `ClcFlt` выполняет те же действия, что и рассмотренная С-программа.

Последовательный сдвиг входных и выходных выборок выполняется подпрограммой `ShftXY`.

Суммирование входных выборок с умножением их на необходимые коэффициенты выполняется при вызове подпрограммы `sumX`. Результат суммирования оказывается в паре ячеек ОЗУ, адрес которых – `asumX`.

Суммирование выходных выборок с умножением на соответствующие коэффициенты выполняется в подпрограмме `sumY`. Результат остаётся в регистрах `b5:b4:b3:b2`.

К полученному четырёхбайтному результату добавляется сумма входных выборок, сохранённая ранее в ОЗУ.

Общая сумма входных и выходных выборок, умноженных на индивидуальные коэффициенты, записывается в четыре ячейки памяти, начиная с адреса `ay4`.

Подпрограмма сдвига входных и выходных выборок shftXY

Подпрограмма выполняет замещение старых выборок новыми. В С-программе эта операция записывалась так:

```
x0 = x1; x1 = x2; x2 = x3;
x3 = x4;
y0 = y1; y1 = y2; y2 = y3;
y3 = y4;.
```

В программе на ассемблере выполняется побайтное извлечение значения более поздней (например, третьей) выборки в регистр временного хранения `tmp`, а затем содержимое этого регистра сохраняется в ячейку для хранения более ранней (второй) выборки.

Для входных выборок извлекаются и сохраняются по два байта, для выходных – по четыре байта, в соответствии с длиной входных и выходных выборок.

Подпрограмма суммирования входных выборок sumX

Как уже было показано, максимально возможное значение результата суммирования входных 12-разрядных выборок при коэффициентах 1, 4, 6, 4, 1 не превышает величины `$fff0`. То есть, как весь результат, так и промежуточные результаты потребуют не более двух байт памяти.

При программировании вычислений на ассемблере ищется компромисс между минимизацией времени вычисления и минимизацией количества используемых регистров.

В рассматриваемой задаче предпочтительным оказалось суммирование выборок с одинаковыми коэффициентами: сначала суммируются выборки `x0` и `x4` с единичными коэффициентами, затем – выборки `x1` и `x3`. После суммирования результат сдвигается влево на два разряда, что эквивалентно умножению выборок на коэффициент 4.

Полученный результат добавляется к накапливаемой сумме, хранящей `x0 + x4`.

Затем в пару переменных `b3:b2` помещается значение выборки `x4`, его однократный сдвиг влево соответствует умножению на 2. Результат добавляется к накапливаемой сумме, затем выполняется ещё один сдвиг содержимого регистров `b3:b2`. Теперь регистры хранят учетверённое значение выборки `x4`, которое также добавляется к накапливаемой сумме. То есть, мы добавили `2 * x4 + 4 * x4` или `6 * x4`.

Полученный двухбайтный результат суммирования входных выборок помещается в ОЗУ микроконтроллера по адресу `asumX`, так как регистры, использовавшиеся для суммирования входных выборок, понадобятся для операций с выходными выборками.

Подпрограмма суммирования выходных выборок sumY

Выражение для суммы, определяющей значение выходной выборки `y4`, имеет вид:

```
y4 = (x0 + x4) + 4 * (x1 + x3)+
+ 6 * x2 + Ky0 * y0 + Ky1 * y1 +
+ Ky2 * y2 + Ky3 * y3
```

или

```
y4 = sumX + sumY.
```

Сумма входных выборок вычисляется подпрограммой sumX.

Подпрограмма sumY должна вычислить сумму выходных выборок:

```
sumY = Ky0 * y0 + Ky1 * y1 +
+ Ky2 * y2 + Ky3 * y3.
```

Перепишем выражение для sumY, указав знак при коэффициентах Ky:

```
sumY = -Ky0 * y0 + Ky1 * y1 -
- Ky2 * y2 + Ky3 * y3.
```

После группировки с учётом знака коэффициента:

```
sumY = - (Ky0 * y0 + Ky2 * y2)+
+ (Ky1 * y1 + Ky3 * y3).
```

Именно такая последовательность вычислений – определение суммы выходных выборок с отрицательными коэффициентами, затем вычисление суммы выборок с положительными коэффициентами, а затем вычисление общей суммы – применяется в подпрограмме `sumY`.

Суммирование выборок с отрицательными коэффициентами начинается извлечением из ОЗУ микроконтроллера выборки `y0`, размещённой в четырёх ячейках, начиная с адреса `ay0`, и помещением её в четыре регистра `c3:c0`.

Из ОЗУ микроконтроллера извлекается коэффициент `Ky0`, размещённый в трёх ячейках, начиная с адреса `aKy0`, и помещается в регистры `b2:b0`. Напомним, что в ОЗУ были размещены модули отрицательных коэффициентов. Подпрограмма `mul4x3` выполняет умножение четырёхбайтного числа, хранимого в регистрах `c3:c0`, на трёхбайтное число, хранимое в регистрах `b2:b0`. Результат умножения оказывается в семи регистрах b6:b0, он копируется в семь ячеек ОЗУ микроконтроллера, начиная с адреса `asumYM`.

В такой же последовательности выполняется операция умножения выборки `y2` на коэффициент `Ky2`. Результат умножения остаётся в регистрах `b5:b0`, а в переменные `c6:c0` из ОЗУ помещается полученный перед этим результат умножения `y0 * Ky0`, хранящийся по адресу `asumYM`.

Оба результата складываются, а сумма вновь помещается в ОЗУ по адресу `asumYM`.

Аналогично выполняется умножение и суммирование выходных выборок с положительными коэффициентами `y1 * Ky1 + y3 * Ky3`, но лишь промежуточный результат умножения `y1 * Ky1` размещается в ОЗУ по адресу `asumYP`.

По окончании суммирования выходных выборок с положительными коэффициентами результат остаётся в регистрах `b6:b0`.

В регистры `c6:c0` из ОЗУ микроконтроллера передаётся сохранённая по адресу `asumYM` сумма выходных выборок с отрицательными коэффициентами. Она вычитается из суммы выходных выборок с положительными коэффициентами. Результат остаётся в шести регистрах `b6:b0`.

Подпрограмма умножения четырёхбайтного числа на трёхбайтное mul4x3

Подпрограмма умножения традиционна: три байта множителя помещаются в младшие байты `b2:b0` семибайтного результата `b6:b0`.

Содержимое байтов `b6:b0` сдвигается вправо. Если перед сдвигом в младшем разряде была единица, после сдвига установится флаг C регистра флагов микроконтроллера, тогда к старшим четырём байтам результат `b6:b3` будет прибавляться множимое `c3:c0`. В противном случае добавление к результату не выполняется. Процесс сдвига и суммирования повторяется 24 раза, по числу разрядов трёхбайтного множителя, пока весь множитель, за счёт сдвига, не окажется замещённым результатом.

Отладка программы на ассемблере

Отладка выполняется в AVR Studio. Интерес при отладке представляет определение времени выполнения подпрограммы `ClcFlt`. При отладке подпрограммы `mul4x3` следует учитывать, что время её выполнения максимально, когда множитель в двоичном представлении содержит максимальное число единиц. Время минимально, если множитель равен нулю, тогда не выполняется ни одной операции добавления множимого.

Ориентировочно время выполнения подпрограммы `ClcFlt` составляет 113 мкс при тактовой частоте микроконтроллера 14,7456 МГц, что позволяет увеличить частоту переполнения Таймера 0, а значит и частоту выборки АЦП, с 1024 до 8000 Гц.

Для С-программы, откомпилированной в IAR-EWB без оптимизации кода, время вычислений, связанных с цифровой фильтрацией, составило 311 мкс при той же тактовой частоте микроконтроллера, то есть, почти втрое больше.

Заключение

При реализации математических вычислений в программе микроконтроллера стоит уделить внимание упрощению и сокращению вычислений – попытаться перейти от вычислений с плавающей запятой к целочисленным вычислениям, возможно, сгруппировать отдельные части формулы с одинаковыми коэффициентами, где допустимо, перейти от умножения и деления к сдвигу. Подобные мероприятия могут ускорить вычисления и удешевить конечное устройство.

Литература

1. АЦП. www.adc-usb.narod.ru