Цифровая фильтрация на микроконтроллере AVR
Часть 1

Вадим Баранов (Украина, Харьков)

Использование цифровых сигнальных процессоров для организации цифровых фильтров не всегда оправдано. Если разрабатываемое устройство предназначено для обработки низкочастотных сигналов, то для построения цифрового фильтра вполне достаточно использовать ресурсы 8-разрядного микроконтроллера. В статье рассмотрен пример проектирования рекурсивного цифрового НЧ-фильтра четвёртого порядка.

Введение

Для определения значения выборки Y выходного сигнала рекурсивный цифровой фильтр четвёртого порядка требует сохранения в памяти значений четырёх предшествующих выборок входного сигнала и четырёх выборок выходного сигнала ($X_0...X_3$ и $Y_0...Y_3$ соответственно), как показано на рисунке 1.

В состав устройства должен входить аналого-цифровой преобразователь (АЦП), выполняющий выборки входного сигнала – преобразования уровня входного сигнала в цифровой код через равные временные интервалы T.

Значение очередной выборки входного сигнала X, поступающее от АЦП в момент времени t, а также хранящиеся в памяти значения предшествующих входных и выходных выборок фильтра $X_0...X_3$ и $Y_0...Y_3$, умножаются на индивидуальные весовые коэффициенты и суммируются для получения значения очередной выходной выборки Y.

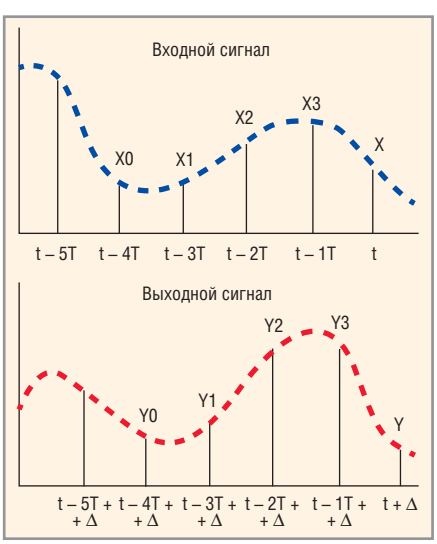

Рис. 1. Пример графика входного и выходного сигнала цифрового фильтра

При необходимости получения выходного аналогового сигнала, представленного на рисунке 1 штриховой линией, устройство следует дополнить цифроаналоговым преобразователем (ЦАП).

Для вычисления очередной выходной выборки, передачи выходного цифрового кода в ЦАП, а также выполнения вспомогательных действий устройству требуется дополнительное время Δ, которое не должно превышать интервал между выборками T. Этим условием определяется максимально допустимая частота выборки, величина которой составляет 1/T.

Если нет желания погружаться в теорию цифровой фильтрации, можно воспользоваться ресурсами интернета для расчёта коэффициентов, необходимых для построения фильтра.

Например, на сайте [1] можно ввести исходные параметры цифрового фильтра:

- выбрать тип фильтра – низкочастотный, высокочастотный, полосовой, а также полином, аппроксимирующий характеристику фильтра – Чебышева, Бесселя, Баттерворта;
- порядок фильтра;
- частоту выборки;
- одну или две частоты среза, в зависимости от типа фильтра;
- неравномерность частотной характеристики в полосе пропускания для полинома Чебышева.

По введённым параметрам будут рассчитаны и предоставлены необходимые коэффициенты и формулы для построения рекурсивного фильтра. При необходимости можно запросить графики амплитудно-частотной (АЧХ) и фазочастотной (ФЧХ) характеристик, а также импульсной характеристики фильтра.

При выборе аппроксимирующего полинома нужно помнить, что использование полинома Баттерворта обеспечит максимальную равномерность характеристики фильтра в полосе пропускания. Фильтр Чебышева гарантирует наиболее крутой спад от полосы пропускания к полосе подавления за счёт некоторой неравномерности в полосе пропускания. Фильтр Бесселя обладает хорошей фазовой характеристикой, запаздывание в полосе пропускания будет практически постоянным, и фильтр Бесселя обеспечит хороший переходной процесс [2]. Следует также учитывать, что для фильтров Бесселя и Баттерворта на частоте среза спад частотной характеристики составляет –3 дБ, для фильтра Чебышева он не превышает заданной величины неравномерности АЧХ в полосе пропускания.

По ссылке [1] можно также разыскать и загрузить исходные файлы проекта, предназначенного для расчёта коэффициентов и графиков. Предложенный проект выполнен на C, а интерфейс программы предусматривает ввод исходных параметров фильтра в командной строке.

Вариант приложения с более удобным интерфейсом можно скачать в Интернете [3], окно приложения показано на рисунке 2. В окне справа представлены результаты расчёта коэффициентов для низкочастотного фильтра Чебышева четвёртого порядка при частоте выборки 1024 Гц, частоте среза 30 Гц и неравномерности АЧХ в полосе пропускания –0,5 дБ. Для фильтра Чебышева приложение рассчитывает также коэффициент передачи, соответствующий выбранной неравномерности (параметр ~K = 0,94406 соответствует максимальному отклонению АЧХ в –0,5 дБ в полосе пропускания).

Для получения значения выборки на выходе фильтра при поступлении значения очередной выборки X от АЦП выполняется ряд операций:

1) выполняется сдвиг выборки в памяти устройства; так, в ячейки, хранившие значение выборки X_0, помещается значение выборки X_1, обозначим эту операцию как $X_0 < -X_1$. Операция производится со всеми хранящимися в памяти значениями выборок в следующей последовательности:

$$X_0 < -X_1 \quad X_1 < -X_2 \quad X_2 < -X_3 \quad X_3 < -X_4$$
$$X_4 < -X \text{ (новая выборка)}$$
$$Y_0 < -Y_1 \quad Y_1 < -Y_2 \quad Y_2 < -Y_3 \quad Y_3 < -Y_4;$$

2) рассчитывается очередное выходное значение Y:

$$Y = K_{x0}X_0 + K_{x1}X_1 + K_{x2}X_2 + K_{x3}X_3 + K_{x4}X_4 + K_{y0}Y_0 + K_{y1}Y_1 + K_{y2}Y_2 + K_{y3}Y_3 + K_{y4}Y_4;$$

3) полученное значение записывается в память: Y4 < –Y;

4) рассчитывается значение выходного кода, предназначенного, например, для передачи в ЦАП: $Y_{out} = Y_4 / Gain$.

Заметим, что значение Gain для выбранного фильтра составляет 40929,19, то есть, значения выходных выборок Y0...Y4, которые сохраняются в памяти для последующих вычислений, более чем в 40 000 раз превышают значение выходного сигнала, если полагать, что коэффициент передачи фильтра в полосе пропускания должен быть равен единице.

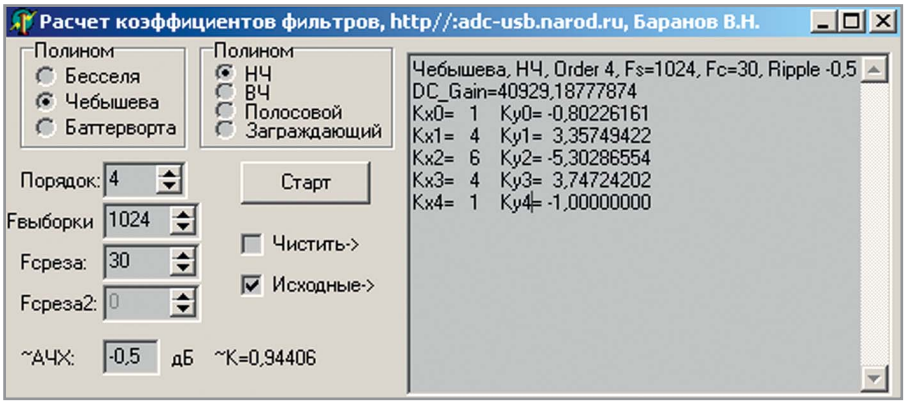

Рис. 2. Интерфейс программы для расчёта коэффициентов фильтров

Схема устройства, на котором проверялась работа программы, представлена на рисунке 3 (значение ёмкости C7 должно выбираться в зависимости от частоты выборки).

Контакты 1...6 предназначены для подключения последовательного программатора к микроконтроллеру DD1. Для преобразования аналогового сигнала в цифровой код используется 10-разрядный АЦП микроконтроллера, сигнал подаётся на вход нулевого канала АЦП (Аналоговый ВХОД 3, контакт 9 схемы) относительно Аналогового ОБЩЕГО (контакт 10 схемы).

Для хранения входных и выходных выборок сигнала используется оперативная память микроконтроллера, а вычисленный микроконтроллером выходной код передаётся по SPI (Serial Peripheral Interface) в 12-разрядный ЦАП DA1, преобразующий выходной код в аналоговый сигнал.

Низкочастотный фильтр R6–C7 должен подавлять частоты, превышающие частоту выборки, поэтому значение ёмкости C7 должно выбираться в зависимости от частоты выборки. Пока же частота выборки 1024 Гц выбрана условно, так как до окончания отладки программы в симуляторе нет возможности проверить, будет ли микроконтроллер успевать выполнить все необходимые вычисления при указанной частоте.

Источник опорного напряжения на микросхеме DA2 стабилизирует напряжение 4 В, определяющее диапазон входных и выходных сигналов (0...4 В).

Если в качестве источника использовать низкочастотный генератор,

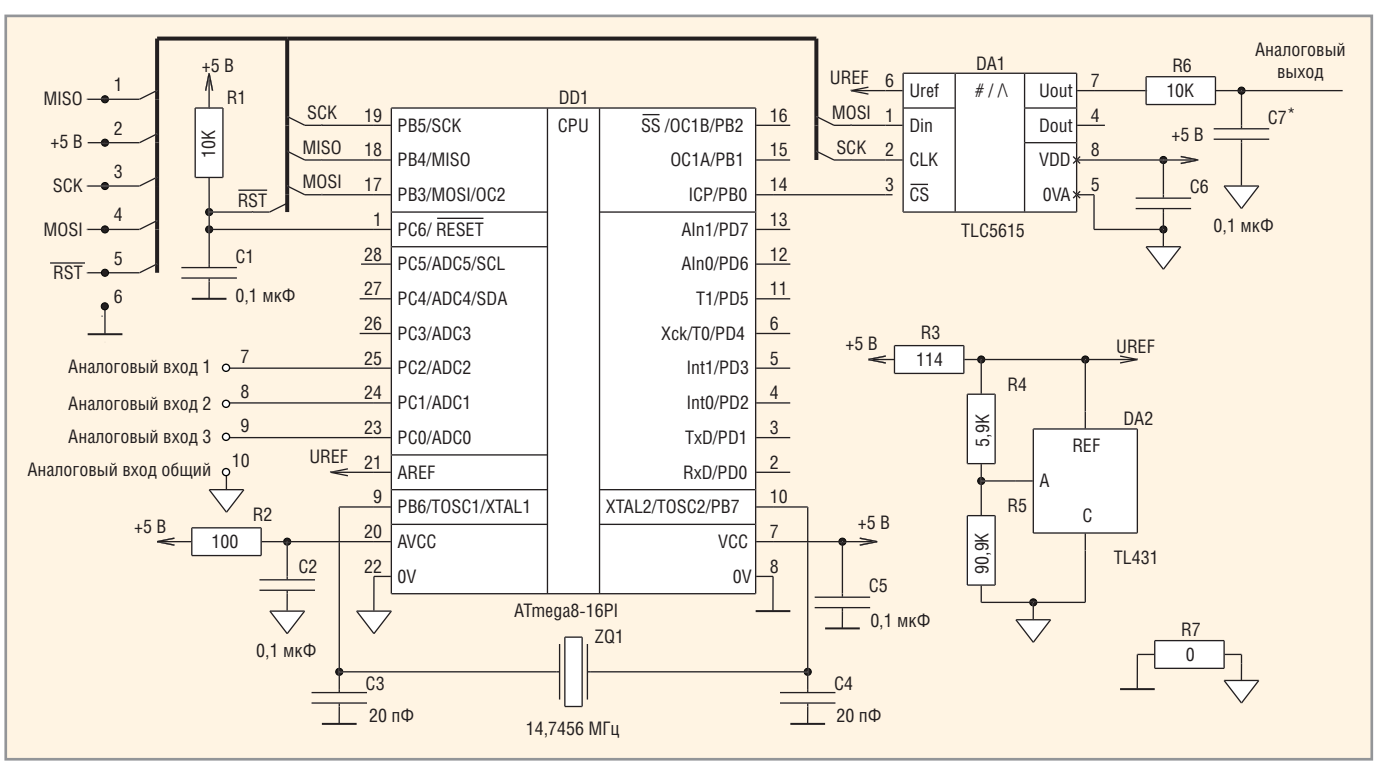

Рис. 3. Принципиальная электрическая схема фильтра на микроконтроллере AVR

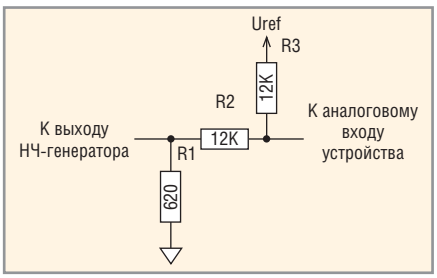

Рис. 4. Схема согласования уровней сигнала

выходной сигнал которого не имеет постоянной составляющей, понадобится устройство согласования уровней. Обычно для этого используется схема с использованием операционного усилителя, но можно обойтись более простой схемой, представленной на рисунке 4.

Входное сопротивление схемы около 600 Ом, поэтому она может подключаться напрямую к генераторам, требующим такую нагрузку. Резисторами R2, R3 сигнал, поступающий от генератора, делится пополам, к нему добавляется смещение, составляющее половину напряжения Uref, значит на входе АЦП будет присутствовать сигнал Uгенератора / 2 + Uref / 2.

Программно реализовать рассматриваемый фильтр проще на языке высокого уровня, поддерживающем операции с действительными числами, например, на C.

Проект цифрового фильтра в IAR Embedded Workbench for AVR

Для работы использовалась версия IAR Embedded Workbench for AVR 4.12 (далее IAR). Разработанная программа приведена в листинге 1.

Константа ADCNum определяет рабочий вход АЦП микроконтроллера ATmega8. Присвоив нулевое значение константе, мы выбираем вход ADC0, и сигнал следует подавать на контакт 9 схемы на рисунке 3 (Аналоговый ВХОД 3).

Константы Pre0 и Tick0 определяют частоту переполнения Таймера 0 в 1024 Гц. Константа Pre0 определяет коэффициент предварительного делителя частоты, поступающей на Таймер 0. Выбранное значение 3 соответствует делению тактовой частоты микроконтроллера 14 745 600 Гц на 64. Каждый раз при переполнении Таймера 0 в его счётчик будет загружаться константа Tick0, и таймер снова будет отсчитывать 225 импульсов до нового переполнения, наступающего при

Листинг 1

```
#include <iom8.h>
#include <ina90.h>
#include <stdio.h>
#include <math.h>

#define ADCNum (0x00)

#define Pre0 3
#define Tick0 256 - 225

/*1024Hz 30Hz cutting, Gain = 40928/4 = 10232*/
#define Gain 40929>>4
#define ky0 - 0.8022616140
#define ky1 3.3574942168
#define ky2 - 5.3028655427
#define ky3 3.7472420208
/*512Hz 30Hz cutting, Gain = 2832/4 = 708*/
unsigned char *il,*ih,bt;
void *ptri;
unsigned short i;
signed long j;
signed short k;
unsigned short x0=2048,x1=2048,x2=2048,x3=2048,x4=2048,x;
float z, y0=83822592,y1=83822592,y2=83822592,y3=83822592,y4=83822592;

#pragma vector=SPI_STC_vect
__interrupt void SPI_interrupt(void)
{bt=0;
}

#pragma vector=ADC_vect
__interrupt void ADC_interrupt(void)
{ADCSR = 0;}

#pragma vector=TIMER0_OVF_vect
__interrupt void Tim0_interrupt(void)
{TCNT0 = Tick0;
 TCCR0 = Pre0;
 _WDR();
 _SEI();
 ADCSR = (1<<ADEN)|(1<<ADSC)|(1<<ADIE)|(1<<ADPS2)|(1<<ADPS1)|(1<<AD
PS0);
 MCUCR = SE;
 _SLEEP();
while (ADCSR); //Only for debug

 x0 = x1;
 x1 = x2;
 x2 = x3;
 x3 = x4;
 x4 = ADC;
 y0 = y1;
 y1 = y2;
 y2 = y3;
 y3 = y4;
 x = x0 + x4 + 4*(x1 + x3) + 6*x2;
 y4 = (ky0*y0 + ky1*y1 + ky2*y2 + ky3*y3 + x);
 j = y4;
 j = j/Gain;
```

Окончание листинга 1

```
 i = j;
 PORTB = 0;
 SPDR = *ih;
 _SLEEP();
 bt=1; while(bt); //Only for debug
 SPDR = *il;
 _SLEEP();
 bt=1; while(bt); //Only for debug
 PORTB = 1;
}

void main(void)
{ptri = &i;
 ih = ptri;
 il = ih++;
 WDTCR =  (1<<WDE)|(1<<WDCE);
 WDTCR = (1<<WDE)|(1<<WDP2)|(1<<WDP1)|(1<<WDP0);
 DDRB = 0x2D; /*Init PortB for SPI work SCK + MOSI
                      + SS + PB0 = 00101101 = 0x2D - outputs*/
 PORTB = 1;
 SPCR = (1<<SPIE)|(1<<SPE)|(1<<MSTR);
 ADMUX = ADCNum;
 TIMSK = 1<<TOIE0;
 TCNT0 = Tick0;
 TCCR0 = Pre0;
 _SEI();

while (1);
}
```

изменении состояния его счётчика с 255 на 0.

Значения констант, определяющих коэффициенты фильтра, соответствуют расчётным, за исключением константы Gain. Её значение уменьшено в четыре раза (сдвинуто вправо на два разряда). Так как для оцифровки сигнала используется 10-разрядный АЦП микроконтроллера, коды отфильтрованного сигнала при расчётном значении параметра Gain окажутся также 10-разрядными. Коды передаются на 12-разрядный ЦАП, для полного использования его диапазона 10-разрядные коды фильтра следует преобразовать в 12-разрядные (увеличить в четыре раза). Поскольку константа Gain используется в качестве делителя выходных кодов фильтра, достаточно уменьшить её в четыре раза для получения необходимого результата. Такое решение к тому же сократит время вычислений, поскольку исчезает необходимость в каждом цикле увеличивать результат в четыре раза перед передачей в ЦАП.

Начальные значения выборок входного сигнала x0...x3 выбраны равными половине шкалы АЦП. Для выборок выходного сигнала y0...y3 начальные значения в Gain-раз больше.

В процедуре main указателю ptri присваивается адрес двухбайтной переменной i, в которую будет помещаться округлённое значение выходной выборки. Адрес старшего байта этой переменной совпадает с адресом самой переменной и копируется в указатель на байт ih. Адрес младшего байта переменной i на единицу больше, он помещается в переменную il. После помещения очередного результата в переменную i можно последовательно передать в ЦАП сначала старший байт, адрес которого хранится в указателе ih, затем младший байт, адрес которого помещён в указатель il.

Сторожевой таймер инициализируется на выполнение сброса микроконтроллера через две секунды, если не выполнена очередная команда сброса сторожевого таймера _WDR().

Запись в регистры порта B и регистр SPCR инициализирует работу SPI-интерфейса микроконтроллера, обеспечивающего передачу данных в ЦАП. В регистр ADMUX заносится номер рабочего канала АЦП. Запись значения 1<<TOIE0 в регистр TIMSK разрешает прерывания переполнения Таймера 0, а занесение оговорённых значений Tick0 и Pre0 в регистры TCNT0 и TCCR0, соответственно, определяет частоту работы таймера.

Команда _SEI() разрешает аппаратные прерывания.

Завершается процедура main бесконечным циклом while, в котором ожидается переполнение счётчика Таймера 0, а при его наступлении выполняется переход на вектор прерывания TIMER0_OVF_vect и обработка соответствующего прерывания Tim0_interrupt.

В обработчике прерывания Tim0_interrupt в счётчик Таймера 0 снова заносится значение Tick0, выполнение команды _WDR() вызывает очередную отсрочку срабатывания сторожевого таймера (пока микроконтроллер работает нормально, срабатывания сторожевого таймера не произойдёт).

После этого снова выполняется команда _SEI(), разрешающая аппаратные прерывания до окончания обработки прерывания Таймера 0. Здесь она необходима, так как переход к обработке любого прерывания автоматически запрещает все аппаратные прерывания, но до окончания обработки прерывания таймера нужно запустить АЦП и дождаться прерывания АЦП, затем передать два байта данных по SPI в ЦАП, каждый раз ожидая прерывания SPI.

Запись значения (1<<ADEN)|(1<<ADSC)|(1<<ADIE)|(1<<ADPS2)|(1<<ADPS1) в регистр ADCSR вызывает однократное преобразование АЦП (ADEN – разрешить работу АЦП, ADSC – старт преобразования АЦП, ADIE – разрешить прерывание АЦП, ADPS2, ADPS1 и ADPS0 – определяют тактовую частоту АЦП как Fck/128 или 115 200 Гц).

Микроконтроллер переводится в спящий режим для того, чтобы дождаться прерывания окончания преобразования АЦП. Выбор типа спящего режима определяется значением, записываемым в регистр MCUCR (выбран режим Idle), перевод в спящий режим – командой _SLEEP().

Окончание преобразования АЦП выводит микроконтроллер из спящего режима и вызывает переход на вектор прерывания ADC_vect, в результате чего выполняется обработчик прерывания ADC_interrupt, в котором в регистр ADCSR записывается нулевое значение, вызывая останов АЦП.

К сожалению, использовавшиеся версии IAR и AVR Studio не позволи-

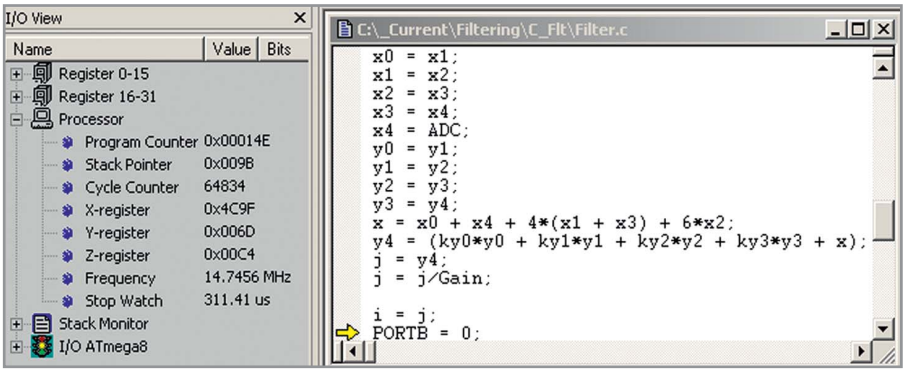

Рис. 5. Определение времени вычисления выборки сигнала

ли симулировать команду _SLEEP() в AVR Studio, хотя при отладке программ, написанных на ассемблере в AVR Studio, симулирование аналогичной команды ассемблера sleep выполняется. Поэтому в целях отладки после команды _SLEEP() была добавлена команда while (ADCSR). Симулятор корректно просчитывает время, требуемое для выполнения преобразования АЦП, поэтому очистка регистра ADCSR, а значит и выполнение цикла while (ADCSR), завершатся по окончании преобразования АЦП.

После возврата из обработчика прерывания АЦП значения входных и выходных выборок сдвигаются, в переменную x4 записывается результат последнего преобразования АЦП. Вычисляется значение выходной выборки y4, результат округляется передачей значения действительной переменной y4 в целую переменную j.

После деления результата, хранящегося в переменной j, на коэффициент Gain в переменной j оказывается значение выходной выборки, имеющее не более 12 значащих двоичных разрядов. Это значение копируется в двухбайтную переменную i для передачи в ЦАП.

Разрешение передачи данных в микросхему ЦАП выполняется установкой нуля в младшем разряде порта B. Старший байт результата, на который ссылается указатель ih, передаётся в регистр SPDR. Микроконтроллер переводится в спящий режим.

По окончании передачи байта данных по SPI микроконтроллер пробуждается, переходя к вектору прерывания SPI_STC_vect. В результате выполняется обработчик прерывания SPI_interrupt, не содержащий ни одной команды, так как задача данного прерывания – лишь пробуждение микроконтроллера и переход к команде, следующей за командой _SLEEP().

После команд _SLEEP(), отработка которой не выполняется симулятором, введён цикл while(bt). Цикл while(bt) повторяется, пока переменная bt не будет обнулена в обработчике прерывания SPI_interrupt – прерывания окончания передачи байта по SPI.

Аналогично в ЦАП передаётся младший байт результата, на который ссылается указатель il. После передачи в младшем разряде порта B устанавливается логическая 1, препятствующая записи в микросхему ЦАП.

Программа возвращается к бесконечному циклу while (1) процедуры main в ожидании следующего прерывания Таймера 0.

Отладка программы

Для того чтобы фильтр работал нормально, время, затрачиваемое микроконтроллером на обработку прерывания Таймера 0, должно быть меньше интервала времени между соседними прерываниями Таймера 0. Для настройки времени можно воспользоваться отладчиком AVR Studio.

Перед компилированием программы в IAR EWB for AVR следует открыть меню Project > Options, в открывшемся окне опций для следующих категорий (окошко Category) выбрать указанные опции:

- категория General Options: на закладке Target в окошке Processor Configuration выбрать –cpu=m8, ATmega8; в окошке Memory model выбрать Small; на закладке System установить флажок Enable bit definitions in I/O-Include files;
- категория C/C++ Compiler: закладка Optimizations – выбрать Speed, None (Best debug support);
- категория Linker: закладка Output, панель Format, выбрать Other, в окошке Output format выбрать ubrof 8 (forced), в результате в окошке на панели Output file окажется имя файла с расширением dbg, генерируемого при компилировании (файл Filter.dbg);
- категория Debugger: на закладке Setup в окошке Driver выбрать Simulator, в окошке Run to установить флажок для перехода на процедуру main.

Выполнить компилирование проекта.

Открыть AVR Studio (в работе использовалась среда разработки AVR Studio 4.12 Service Pack 2). В AVR Studio открыть файл Filter.dbg, который находится в папке <путь к проекту в IAR>\Debug\Exe.

AVR Studio предложит создать и сохранить проект Filter_dbg.aps. После сохранения появится окно Select debug platform and device. В качестве Debug platform предлагается AVR Simulator. В качестве устройства выберите ATmega8 из списка Device.

Перед отладкой выберите меню Debug > AVR Simulator Options и в окошке Frequency установите частоту 14,7456 МГц.

При отладке контролируйте на панели I/O View параметр Processor > Stop Watch.

Установите курсор в начале обработчика прерывания Таймера 0 (строка листинга __interrupt void Tim0_interrupt(void)), выполните отладку до курсора (Run to Cursor), нажав Ctrl + F10.

Сбросьте параметр Stop Watch, снова установите курсор в начало обработчика прерывания Таймера 0, нажмите Ctrl + F10. Параметр Stop Watch приобрёл значение 976,56 мкс – это интервал между двумя прерываниями Таймера 0. Величина, обратная этому интервалу, – 1024 Гц – частота выборки АЦП.

Пользуясь описанной методикой, можно определить время преобразования АЦП (время от команды ADCSR = ... до команды x0 = x1). Это время составляет 114 мкс, что незначительно превышает время преобразования АЦП, занимающее, согласно документации на микроконтроллер [4], 13 циклов выбранной тактовой частоты АЦП 115 200 Гц.

Таким же образом определим время, необходимое для вычисления выходной выборки (это время от команды x0 = x1 до команды i = j). Время, необходимое для вычислений для программы, написанной на C, составляет 311 мкс (см. рис. 5).

Общее время выполнения обработчика прерывания Таймера 0 составляет

435 мкс. Это позволяет увеличить частоту выборки вдвое, до 2048 Гц. Период этой частоты составит 488,3 мкс, что превышает полученное время обработки прерывания Таймера 0.

При необходимости использования внешнего АЦП, например, для увеличения разрешения до 12 разрядов, время выполнения обработчика прерывания сведётся ко времени вычисления и составит чуть более 311 мкс, из них 235 мкс требуется для вычисления y4 и 46 мкс – для операции деления на коэффициент Gain.

Целочисленное вычисление значения выходной выборки

Сократить время подсчёта выходной выборки позволит переход от вычислений с плавающей запятой к работе с целыми числами, а также программирование на ассемблере.

В программе Filter.c (см. листинг 1) уже было использовано округление коэффициента Gain (40929,18777874) до целого значения, это сократило время вычисления на 14 мкс.

Очевидно, что округлять все используемые в программе числа с плавающей запятой нельзя, в этом случае изменение коэффициентов ky0...ky3 оказалось бы недопустимо большим.

Прежде чем писать программу на ассемблере, нужно оценить, до какого числа двоичных разрядов допустимо округлять тот или иной параметр. Также надо учесть разрядность используемого АЦП.

Смоделировать АЧХ-фильтр при округлении параметров поможет небольшая программа, написанная в Delphi (см. листинг 2).

Далее приведены пояснения для строк, содержащих комментарий вида //1.

//1. Форма приложения содержит контейнер графика Chart1.

//2. А также один график Series1.

//3. График рассчитывается и строится сразу после создания формы автоматически, вычисления и построение графика выполняются в процедуре FormCreate.

//4. Амплитуда входного сигнала фильтра 2047 единиц (предполагается, что оцифровку входного сигнала выполняет 12-разрядный АЦП, его диапазон 0...(2^{12} – 1) или 0...4095; 2047 – половина диапазона).

//5. Параметр округления NK – это коэффициент, на который умножаются параметры фильтра ky0...ky3. Так как позже на этот же коэффициент придёт-

Листинг 2

```
unit AFCintu;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  StdCtrls, ExtCtrls, TeeProcs, TeEngine, Chart, Series, Spin;
type
  TForm1 = class(TForm)
    Chart1: TChart;                                  //1
    Series1: TFastLineSeries;                        //2
    procedure FormCreate(Sender: TObject);                  //3
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.FormCreate(Sender: TObject);
const Ampl=2047;                                     //4

var
    i,j,Fmax:integer;
    max,min:array[0..1000]of real;
    NK,ky0,ky1,ky2,ky3,f:longword;
    X,x0,x1,x2,x3,x4:word;
    y0,y1,y2,y3,y4,y, Gain:integer;

begin
//===Параметры фильтра: ===
Gain:=40929;
f:=1024;              //Частота выборки
Fmax:=60;             //Максимальная частота на графике
NK:=65536;         //параметр округления                   //5
ky0:=round(0.8022616140*NK);
ky1:=round(3.3574942168*NK);
ky2:=round(5.3028655427*NK);
ky3:=round(3.7472420208*NK);
x0:=Ampl;x1:=Ampl;x2:=Ampl;x3:=Ampl;x4:=Ampl;
y0:=round(Ampl*Gain); y1:=y0; y2:=y0; y3:=y0; y4:=y0;
Series1.Clear;
for j:=1 to Fmax do begin                                //6
  max[j]:=-10000000; min[j]:=10000000;                   //7
  for i:=0 to (10*f div j)-1 do begin
//8
    x4:=Ampl+round(Ampl*sin(2*Pi*j*i/f));                //9
    x0:=x1; x1:=x2; x2:=x3; x3:=x4;                      //10
    y0:=y1; y1:=y2; y2:=y3; y3:=round(y4);
    y4:=(x0+x4)+4*(x1+x3)+6*x2 + ((ky1*y1+ky3*y3-(ky2*y2+ky0*y0))div
NK);   //11
    y:=y4 div Gain;                                      //12
    if i>5*f div j then begin                            //13

      if max[j]<y then max[j]:=y;                        //14
      if min[j]>y then min[j]:=y;                        //15
    end;
  end;
  Series1.AddXY(j,20*ln((max[j]-min[j])/2/Ampl)/ln(10));   //16
end;
end;

end.
```

Рис. 6. График АЧХ фильтра

ся делить результат вычисления, удобно выбрать этот параметр равным 2^N, где N – целое положительное число. Тогда деление результата может быть сведено к его сдвигу вправо на N разрядов, что удобно при программировании на ассемблере. Выбранный коэффициент 65536 или 2^{16} потребует сдвига результата на 16 двоичных разрядов вправо или попросту отбрасывания двух младших байтов результата.

Параметр NK подбирался экспериментально: по построенным графикам определялась неравномерность АЧХ в полосе пропускания. Если моделирование фильтра с использованием чисел с плавающей запятой даёт расчётную неравномерность АЧХ (0,5 дБ), то, например, фильтр с параметром округления 16384 (или 2^{14}) имеет неравномерность АЧХ около 0,8 дБ. Фильтр с параметром округления 65536 оказался приемлемым (неравномерность АЧХ составила 0,6 дБ). При этом учитывалось, что увеличение этого параметра приводит к замедлению вычислений.

//6. АЧХ рассчитывается для диапазона частот 1 Гц...Fmax, шаг построения графика составляет 1 Гц.

//7. Для сигнала каждой частоты j из диапазона 1 Гц...Fmax будут определены максимальное и минимальное значения и занесены в массивы max[j] и min[j]. Предварительно элементу массива max[j] присваивается значение, меньшее минимально возможного значения сигнала, а в элемент min[j] – большее максимально возможного значения сигнала.

//8. В цикле выполняется расчёт (10 * f div j) выборок. При частоте выборки f указанное число выборок составит десять периодов сигнала частотой j. Например, в десяти периодах сигнала частотой j = 100 Гц окажется (10 * 1024 div 100) или 102 полных периода частоты выборки 1024 Гц.

//9. Вычисляется значение i-той выборки входного сигнала x4 частотой j при частоте выборки f. Как в реальном АЦП, сигнал имеет постоянное смещение (значение Ampl выбрано равным половине диапазона 12-разрядного АЦП, т.е., 4095 div 2 или 2047). Амплитуда входного синусоидального сигнала также выбрана равной Ampl. Значит, сигнал изменяется в диапазоне Ampl – Ampl...Ampl + Ampl или 0...2Ampl.

//10. До строки с комментарием //12 выполняются уже знакомые по С-программе сдвиги значений входных и выходных выборок, а также вычисление значения выходной выборки.

//11. При вычислении значения y4 сумма, в подсчёте которой участвуют ky0...ky3, делится на параметр округления NK, что компенсирует предварительное умножение этих коэффициентов на параметр округления (см. комментарий к строке //5).

Все приведённые в листинге вычисления проведены с округлением, их несложно выполнить при написании программы на ассемблере.

//12. Для получения выходной выборки значение y4 следует разделить на коэффициент Gain.

//13. Первые пять периодов сигнала частотой j пропускаются (это составляет 5 * f div j выборок). Так как после подачи входного сигнала фиксированной частоты происходит установление выходного сигнала, первые пять периодов исключаются при определении максимума и минимума сигнала.

//14, //15. Определяется максимальное и минимальное значения сигнала частотой j и заносятся в массивы max и min соответственно.

//16. К графику Series1 (АЧХ) добавляется нормированный коэффициент передач фильтра, рассчитанный для частоты j. Значение коэффициента вычисляется в логарифмическом масштабе как 20 * lg((max[j] – min[j]) / (2 * Ampl)).

Полученный график изображён на рисунке 6.

Можно изменить параметры фильтра, параметр округления, посмотреть, как это скажется на графике. Также возможно увеличение выбранной области на графике (Zoom), это позволяет оценить неравномерность АЧХ.

Для вывода АЧХ в виде таблицы необходимо добавить на форму экземпляр Memo1 компонента TMemo. А за строкой //16 достаточно добавить строку:

```
Memo1.Lines.Add(Format('%3d Гц
%6.2f dB', [j, 20 * ln((max[j] -
min[j]) / 2 / Ampl) / ln(10)])).
```

Если задача позволяет снизить требования к крутизне спада на частотах вне полосы пропускания, то можно задать меньшую неравномерность частотной характеристики фильтра, например, 0,4 дБ вместо требуемых 0,5 дБ. В результате использования округления неравномерность станет немного выше, например, требуемые 0,5 дБ. При этом несколько снизится крутизна спада вне полосы пропускания фильтра.

Во второй части статьи будет рассказано о разработке программы цифрового фильтра для микроконтроллера на ассемблере и о процедуре отладки созданной программы.

Литература

1. *Fisher Tony.* Butterworth / Bessel / Chebyshev Filters. www-users.cs.york.ac.uk/~fisher/mkfilter/trad.html.
2. *Хоровиц П., Хилл У.* Искусство схемотехники. М. МИР. 1993.
3. www.adc-usb.narod.ru/fltcalc.rar.
4. 8-bit Atmel with 8Kbytes In-System Programmable Flash ATmega8, ATmega8L. Техническое описание микроконтроллера ATmega8. www.atmel.com/Images/Atmel-2486-8-bit-AVR-microcontroller-ATmega8_L_datasheet.pdf.

РОССИЙСКАЯ НЕДЕЛЯ ВЫСОКИХ ТЕХНОЛОГИЙ

НАВИТЕХ • NAVITECH

ufi Approved Event

XI МЕЖДУНАРОДНЫЙ НАВИГАЦИОННЫЙ ФОРУМ

www.glonass-forum.ru

9-я международная выставка НАВИТЕХ

www.navitech-expo.ru

25–28 апреля 2017

ЦВК «ЭКСПОЦЕНТР»
МОСКВА

Реклама 12+

При поддержке: МИНПРОМТОРГ РОССИИ

Под патронатом: ТОРГОВО-ПРОМЫШЛЕННАЯ ПАЛАТА РОССИЙСКОЙ ФЕДЕРАЦИИ

Организатор форума: ГЛОНАСС/ГНСС Форум

Оператор форума: ПроКонф

Стратегический партнер форума: НП "ГЛОНАСС" Федеральный сетевой оператор

Организатор выставки: ЭКСПОЦЕНТР МОСКВА