МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ
імені адмірала Макарова

Г. В. Павлов, М. В. Покровський, І. Л. Вінниченко

МЕТОДИЧНІ ВКАЗІВКИ
до виконання курсового проекту
„Проектування систем управління на базі
мікроконтролерів серії *AVR*”
з дисципліни „Мікроконтролери в системах управління”

Рекомендовано Методичною радою НУК

Миколаїв ◊ НУК ◊ 2018

УДК 621.3.049.77(076)
ББК 32.844.150.2я73
П 12
Автори:
Г. В. Павлов, д-р техн. наук, професор;
М. В. Покровський, канд. техн. наук;
І. Л. Вінниченко
Рецензент О. К. Жук, канд. техн. наук, доцент

Павлов Г. В.
П 12 Методичні вказівки до виконання курсового проекту „Проектування систем управління на базі мікроконтролерів серії AVR” з дисципліни „Мікроконтролери в системах управління” / Г. В. Павлов, М. В. Покровський, І. Л. Вінниченко – Миколаїв : НУК, 2018. – 36 с.

Наведено відповіді на основні питання, пов’язані з організацією та методикою проведення курсової проекту із дисципліни „Мікроконтролери в системах управління”: вибір теми, визначення об’єкту та предмету дослідження, оформлення результатів.

Методичні вказівки призначені для студентів денної і заочної форми навчання спеціальності «Автоматизація та комп’ютерно-інтегровані технології». Можуть бути використані студентами інших спеціальностей, викладачами вищих навчальних закладів четвертого-третього рівнів акредитації, а також у системі перепідготовки та підвищення кваліфікації.

ВСТУП

Метою курсового проекту є оволодіння студентами методикою і навичками проектування систем управління на базі мікроконтролерів (МК) серії *AVR*. Курсовий проект виконується студентами поряд з вивченням ними теоретичних питань у курсі лекцій, проходженням лабораторного практикуму та опануванням мови програмування МК серії *AVR*. Курсовий проект є важливою формою професійної підготовки фахівця з напряму підготовки 6.050201 "Системна інженерія".

Курсовий проект має на меті виконання таких завдань: а) систематизація, закріплення та розширення теоретичних знань при розв’язанні конкретних технічних задач; б) розвиток навичок самостійної роботи з технічною літературою; в) оволодіння творчими навичками при самостійному вирішенні конкретних технічних задач; г) підготовка до дипломного проектування. За своєю сутністю курсовий проект є невеликою самостійною інженерною розробкою одного з актуальних питань теорії та практики конструювання систем управління на базі МК і характеризує опанування теоретичних знань, набуття практичних навичок у цій галузі.

Для виконання проекту необхідно розв’язати дві задачі: розробити апаратну частину і розробити програмне забезпечення. Під розробленням апаратної частини необхідно розуміти складання електричної принципової схеми пристрою, що проектується. Розроблення програмного забезпечення полягає у складанні алгоритмів роботи пристрою і написанні тексту програми на мові Асемблер. Кінцевим продуктом програмного забезпечення в курсовому проекті є лістинг програми. Програма повинна бути відтрансльована, а повідомлення транслятора повинно вказувати на відсутність помилок.

ЗАВДАННЯ НА КУРСОВЕ ПРОЕКТУВАННЯ

Завдання та конкретні параметри визначається викладачем. Студенту надається право вибору завдання. Студент може також запропонувати своє завдання з обґрунтуванням доцільності її розробки. Приклади завдань:

1. Двоканальний вимірювач частот та часових інтервалів;
2. Індикаторне табло на базі рідкокристалічного символьного індикатору;
3. Керування паливними та повітряними засувками;
4. Генератор сигналів;
5. Багатоканальний вимірювач напруги;
6. Стабілізатор змінної напруги ключового типу;
7. Інфрачервоний підігрів паяльного столу;
8. Електронний кодовий замок;
9. Ворота з дистанційним керуванням;
10. Світлодіодний символьний екран;
11. Регулятор швидкості обертання двигуна постійного струму;
12. Облік споживання електроенергії змінного струму;
13. Вимірювання та індикація рівня рідини;
14. Управління позіціонуванням за допомогою крокового двигуна;
15. Вимірювання та індикація температури об’єкту;
16. Управління сушильною піччю;
17. Збір інформації з цифрових датчиків температури;
18. Контроль параметрів електричної мережі змінного струму;
19. Ультразвуковий вимірювач відстані;
20. Управління пасажирським ліфтом.

В якості приклада далі буде розглядатися виконання завдання „Цифрова фільтрація сигналів за допомогою МК серії *AVR*”.

ВИМОГИ ДО ОФОРМЛЕННЯ КУРСОВОГО ПРОЕКТУ

Технічна документація курсового проекту розглядається як різновид звіту про виконану інженерно-конструкторську або науково-дослідну роботу. Матеріали курсового проекту повинні відповідати діючим стандартам у сфері оформлення звітів такого виду.

Технічна документація до курсового проекту включає розрахунково-пояснювальну записку (РПЗ) і графічний матеріал. РПЗ до курсового проекту оформлюється на одній стороні аркушів формату А4. Слід використовувати шрифт *Times New Roman* розміром 14 з міжрядковим інтервалом 1,15-1,5 в межах всього текстового документа. Таблиці та рисунки обов'язково повинні мати назву, номер і згадування в тексті РПЗ.

РПЗ повинна містити:

1. Титульний аркуш;
2. Завдання до курсового проекту;
3. Зміст;
4. Перелік умовних позначень символів, одиниць і термінів;
5. Вступ;
6. Опис основних функцій системи управління та вибір МК;
7. Побудову та опис принципової схеми;
8. Алгоритми роботи пристрою, що проектується;
9. Розроблене програмне забезпечення;
10. Список використаних джерел;
11. Додатки.

Графічний матеріал курсового проекту містить такі обов'язкові частини:

1. Схему електричну принципову розробленого пристрою;
2. Основні алгоритми роботи розробленого пристрою.

ОПИС ОСНОВНИХ ФУНКЦІЙ СИСТЕМИ УПРАВЛІННЯ ТА ВИБІР МІКРОКОНТРОЛЕРА

На етапі опису основних функцій системи управління студент виконує такі дії: визначає функціональний склад модулів пристрою, що проектується; обґрунтовує технічні вимоги до вказаних модулів і встановлює необхідні зв'язки між ними. Виходячи з отриманих вимог до пристрою проводиться вибір МК, який задовольняє цим вимогам.

Пристрої, що оброблюють сигнали від зовнішніх аналогових джерел, часто потребують цифрової фільтрації сигналів. Для максимально високої швидкодії фільтра зазвичай вибирають спеціалізовані цифрові сигнальні процесори, але в багатьох випадках вони виявляються занадто дорогими для використання [1]. В цьому випадку можливо використати 8- або 16-бітові МК загального призначення. МК серії *AVR* добре підходять для обробки сигналу, так як мають потужну архітектуру з великою кількістю регістрів загального призначення (РЗП), широкий набір інструкцій, більшість з яких виконується за 1 або 2 машинних циклу, а також вбудований багатоканальний 10-бітний аналого-цифровий перетворювач (АЦП) [1]. Сімейство *Mega* серії *AVR* також має апаратний помножувач, який значно збільшує швидкодію при цифрової обробці сигналів.

Всі лінійні цифрові, інваріантні за часом фільтри можуть бути описані диференціальним рівнянням [3]:

$$\sum_{i=0}^{M} a_i \cdot y[n-i] = \sum_{j=0}^{N} b_j \cdot x[n-j],$$

де x – вхідний сигнал, y – вихідний сигнал, n – номер вибірки сигналу.

Фільтр унікально визначається за його порядком (найбільше з M і N) і коефіцієнтам a_i і b_j. Порядок фільтра визначає максимальну затримку за часом між входом і виходом фільтра. Зазвичай коефіцієнти масштабуються

таким чином, що коефіцієнт a_0 дорівнює 1. Вихідне значення сигналу фільтра може бути обчислено за формулою:

$$y[n] = \sum_{j=0}^{N} b_j \cdot x[n-j] + \sum_{i=1}^{M} - a_i \cdot y[n-i].$$

Фільтри класифікуються за двома типами, залежно від значення M [3]:

1. Тип фільтра зі скінченною імпульсною характеристикою (СІХ) або *finite impulse response* (*FIR*), для нього $M=0$;
2. Тип фільтра з нескінченною імпульсною характеристикою (НІХ) або *infinite impulse response* (*IIR*), для нього $M \neq 0$.

Фільтри *IIR* обчислюються з рекурсією, тобто попередні значення ($y[n-1]$) беруть участь в обчисленні поточного вихідного значення ($y[n]$).

Часто цифрові фільтри описуються за допомогою Z-перетворення :

$$Y(z) \cdot \sum_{i=0}^{M} a_i \cdot z^{-i} = X(z) \cdot \sum_{j=0}^{N} b_j \cdot z^{-j} .$$

Для зручності частотного аналізу задається передавальна функція фільтра:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{j=0}^{N} b_j \cdot z^{-j}}{\sum_{i=0}^{M} a_i \cdot z^{-i}} = \frac{\sum_{j=0}^{N} b_j \cdot z^{-j}}{1 + \sum_{i=1}^{M} a_i \cdot z^{-i}} .$$

Чисельник передавальної функції описує частину фільтра без зворотного зв'язку, а знаменник – частину фільтра із зворотним зв'язком [3]. Для реалізації фільтра за заданою функції перетворення важливо знати, що змінна z представляє елемент затримки, а ступінь при змінній z задає довжину затримки в одиницях вибірок. На рис. 1 показана структурна схема цифрового фільтра.

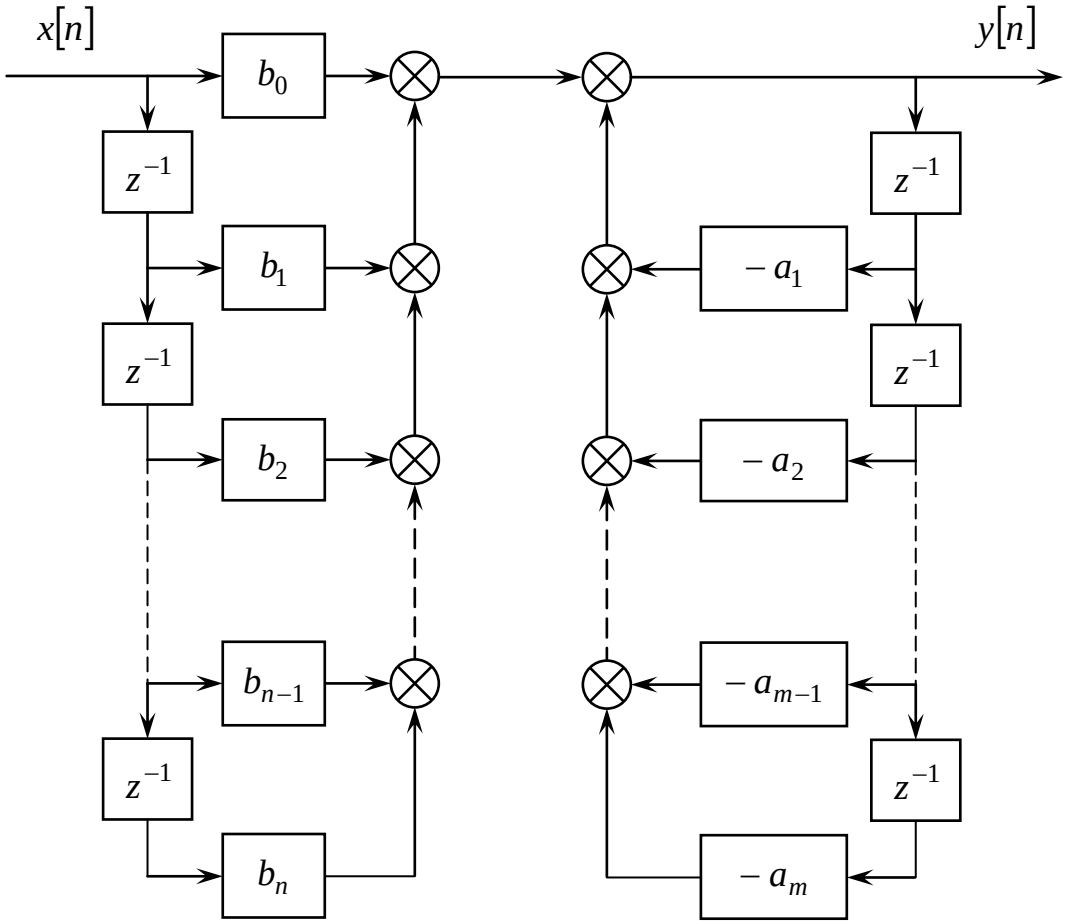

Рис. 1. Структурна схема цифрового фільтра

При проектуванні фільтра на вибраній архітектурі МК необхідно взяти до уваги кілька моментів [2]:

1. Точність представлення сигналу на вході і виході впливатиме на максимально допустиме посилення фільтра і його пропускну здатність;
2. Точність представлення коефіцієнтів фільтра впливає на амплітудно-частотну характеристику (АЧХ) і пропускну здатність фільтра;
3. Порядок фільтра впливає на пропускну здатність фільтра;
4. Дробові коефіцієнти фільтра вимагають деяких перетворень при використанні цілочисельного помножувача.

Алгоритм фільтрації складається з підсумовування результатів декількох кроків обчислень. Перший результат обчислюється простим перемножуванням між двома *N*-бітними числами, в результаті чого виходить 2·*N*-бітний результат. Перемноження виконується з одним байтом, який множиться на кожний з множників, і результат зберігається 2·*N*-бітному акумуляторі. Наступний результат обчислюється і додається в акумулятор, ніж виробляється так звана операція *multiply-and-accumulate* (*MAC*).

У серії *AVR* немає виділеного спеціального акумулятора, замість нього використовуються будь-які з 32 8-бітних РЗП в формі "віртуального акумулятора" [2]. Такий акумулятор більш гнучкий, ніж звичайні акумулятори, що використаються в інших архітектурах. Наприклад, якщо потрібний 24-бітний акумулятор, щоб перемножити дві 12-бітові величини, то комбінуються три 8-бітних РЗП в один 24-бітний акумулятор. Гнучкість акумулятора – важливий інструмент для обходу проблем переповнення в алгоритмах з фіксованою точкою. Переповнення може статися у двох місцях алгоритму фільтра – в проміжних результатах обчислень і на виході фільтра. Переповнення в проміжних результатах може відбутися:

1. При перемножуванні величин з розрядністю *N*1 і *N*2, в результаті вийде результат з *N*1+*N*2 біт;
2. Додавання величин може видати суму, яка має на 1 біт більше розмір, ніж найбільший за розрядності операнд додавання.

Наприклад, розглянемо фільтр *FIR* 4-го порядку. Вихідне значення фільтра *y*[*n*] дорівнює сумі 5 проміжних складових:

$$y[n] = \sum_{j=0}^{4} b_j \cdot x[n-j].$$

Якщо припустити, що вхідні вибірки $x[n]$ і коефіцієнти b_j мають розрядність 16 біт зі знаком, то алгоритм вимагатиме 34-бітного акумулятора:

$$N \geq 2 \cdot K + log_2(M) + 1 = 2 \cdot 15 + log_2(5) + 1 \approx 33.32 .$$

В цьому рівнянні N – необхідна розрядність акумулятора, K – розрядність операндів без урахування біта знака, M – кількість складових. Акумулятор для даного випадку повинен складатися з 5 8-бітних РЗП (40 біт), щоб зберегти максимальну величину, яка може зустрітися при обчисленні результату. Для фільтра *IIR* вихідні значення фільтра беруть участь в обчисленнях алгоритму фільтрації. Тому розмір акумулятора повинен бути підібраний з урахуванням розрядності виходу фільтра.

Зазвичай для більш точного уявлення числа воно повинно мати максимально можливе число біт. При застосуванні дробових коефіцієнтів фільтра в цілочисельному множенні проблема зводиться до масштабування всіх коефіцієнтів найбільшим загальним множником, який не викликає переповнень в їх поданні. Це масштабування також застосовується і до коефіцієнта a_0 із застосуванням масштабування виходу в зворотному напрямі для отримання коректного значення. Оскільки апаратне ділення недоступне, фактор масштабування повинен бути у формі 2^k, так як ділення і множення з основою 2 легко реалізується зсувами:

$$y[n] = \left(\sum_{j=0}^{N} \left(2^k \cdot b_j\right) \cdot x[n-j] + \sum_{i=0}^{M} \left(-2^k \cdot a_i\right) \cdot y[m-i] \right) \cdot 2^{-k} .$$

Біт знака (найстарший біт) повинен бути збережений при масштабуванні в зворотному напрямі. Простіше всього для цієї мети застосувати інструкцію асемблера *asr* (арифметичний зсув вправо).

Наприклад, завдано коефіцієнти фільтра b_j={0.9001, -0.6500,0.3000}.

Якщо використовується 16-бітове представлення чисел, то масштабовані коефіцієнти повинні бути в діапазоні [-2^{15} .. (2^{15}-1)]=[-32768 .. 32767]. Зазвичай коефіцієнт з найбільшим абсолютним значенням визначає обмеження максимального множника масштабування. В даному випадку максимальний множник без будь-якого переповнення дорівнює 2^{15}. Після округлення отримаємо значення коефіцієнтів {29494, -21299, 9830} і помилку округлення при масштабуванні в зворотному напрямі {$1.5\cdot10^{-5}$, $6.1\cdot10^{-6}$, $1.2\cdot10^{-5}$}.

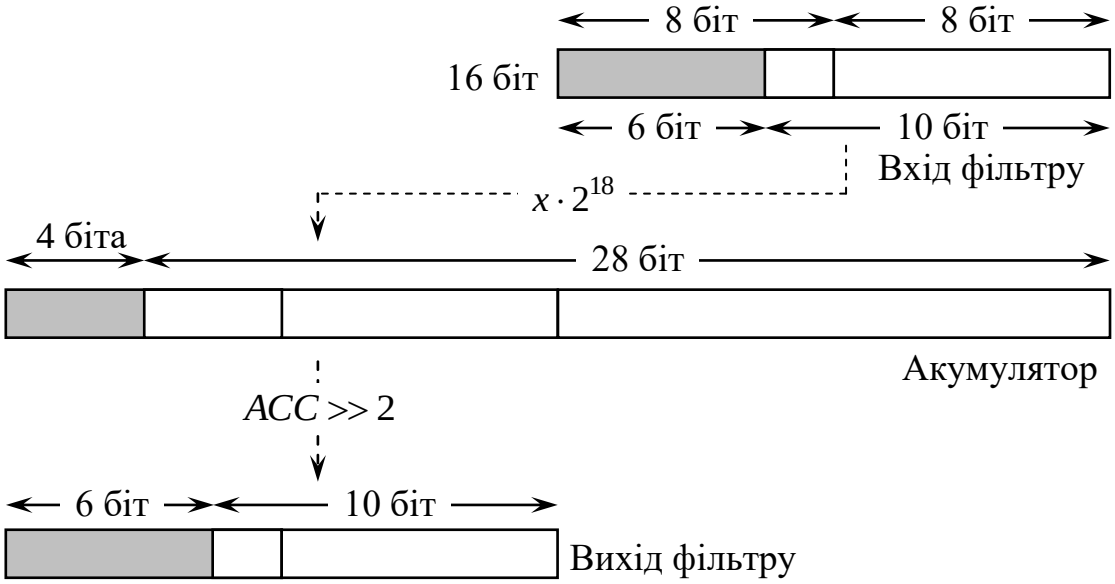

Рис. 2. Оптимізація масштабування в зворотному напрямі (сірим позначені біти, що не використаються при обчисленнях)

Оптимізація масштабування в зворотному напрямі можлива, якщо фактор *k* перевищує число 8, помножене на ціле число (1, 2, 3, ..). В цьому випадку програма при масштабуванні в зворотному напрямі може просто відкидати молодші байти результату (1, 2, 3, .. байта). Приклад такого масштабування показаний на рис. 2, де 32-бітний результат повинен бути масштабований в 2^{18} раз. Це виконується шляхом зсуву старших 16 біт тільки два рази вправо (при цьому молодші 16 біт просто відкидаються).

Масштабування в зворотному напрямі видаляє дробову частину

результату, залишаючи тільки цілу частину з необхідною кількістю біт, що означає зменшення точності. Це важливо для фільтрів *IIR*, так як у них є зворотний зв'язок. Якщо ефект від втрати точності створює проблему, то втрата точності може бути зменшена двома способами [2]:

1. Зробити посилення фільтра максимальним, при цьому вихід буде використовувати весь доступний діапазон значень;
2. Збільшити одночасно і точність виходу і посилення фільтра.

Якщо застосовується другий метод, то останній фактор може впливати на розмір коду і пропускну здатність, тому що може знадобитися акумулятор збільшеного розміру. Щоб прискорити роботу алгоритму фільтрації, може знадобитися зменшити точність коефіцієнтів або вхідних вибірок сигналу, що дозволить зменшити розмір акумулятора. Зменшений акумулятор означає, що алгоритм вимагатиме менше операцій на множення коефіцієнта фільтра і вибірки. Однак, перш ніж зменшувати точність, потрібно взяти до уваги два моменти:

1. Зменшення точності вхідних вибірок означає додавання шуму в систему, що зазвичай небажано;
2. Зменшення точності коефіцієнтів фільтра означає, що потрібну характеристику фільтра отримати буде складніше.

Базовим МК, який вивчається за дисципліною „Мікроконтролери в системах управління”, є МК *ATMega*16. Для розглянутого завдання можливості МК *ATMega*16 є завищеними, тому обрано МК *ATMega*8, який має 8 кб пам’яті програм, 1 кб оперативної пам’яті, апаратний помножувач, 23 лінії вводу-виводу, 3 таймера, АЦП, 3 послідовних інтерфейси (*UART, SPI, I2C*), тактову частоту до 16 МГц. Однак, цей МК не має у своєму складі цифро-аналогового перетворювача (ЦАП) для формування вихідного сигналу [4]. Тому ЦАП необхідно реалізовувати у вигляді зовнішньої по відношенню до МК схемі.

ПОБУДОВА ТА ОПИС ПРИНЦИПОВОЇ СХЕМИ

Розглянемо приклади реалізації двох фільтрів – фільтра *FIR* 2-го порядку і фільтра *IIR* 2-го порядку. В обох реалізаціях використається принципова схема, яка наведена на рис. 3.

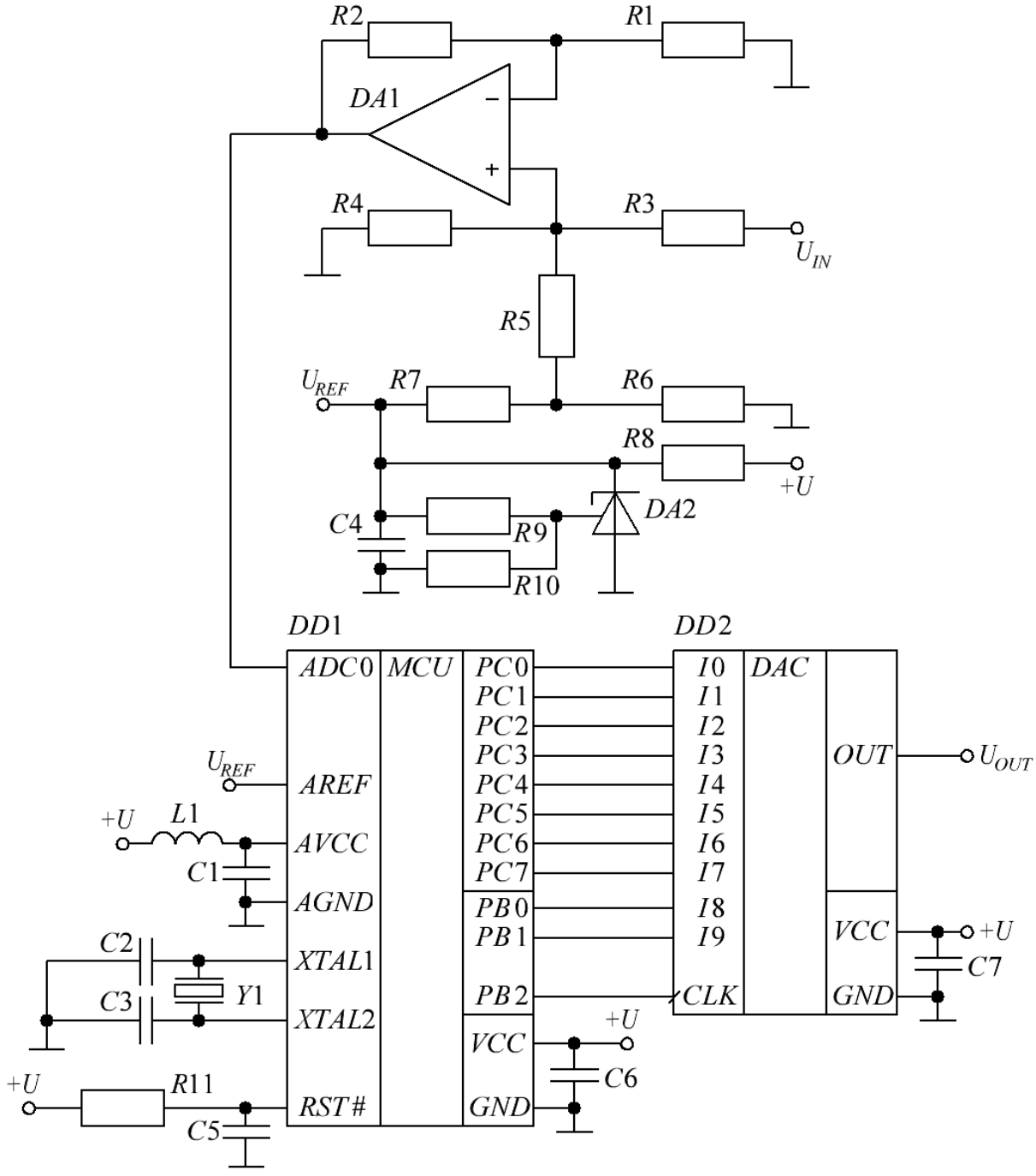

Рис. 3. Принципова схема цифрового фільтра

Основними частинами схеми є МК *DD*1, ЦАП *DD*2, вхідний підсилювач *DA*1 та джерело опорної напруги *DA*2. Вхідний підсилювач забезпечує масштабування вхідного сигналу U_{IN} та його зсув для вимірювання негативної напруги. Резистори *R*1-*R*5 встановлюють необхідні коефіцієнти посилення вхідного сигналу та напруги зсуву. Напруга зсуву формується за допомогою дільника на базі резисторів *R*6, *R*7. Ця напруга зазвичай встановлюється таким чином, щоб нульовому вхідному сигналу відповідала половина опорної напруга на вході АЦП. В цьому випадку при опорної напрузі 4,096 В, наприклад, 10-бітний АЦП може вимірювати сигнали у діапазоні від -2,048 В до 2,044 В. Джерело опорної напруги побудовано на базі регульованого стабілітрона *DA*2 та резистора *R*8. Резистори *R*9, *R*10 завдають величину опорної напруги. Конденсатор *C*4 служить для запобігання різким змінам опорної напруги. Опорна напруга подається на вивід *AREF* МК та на дільник *R*6, *R*7.

МК отримує вхідні дані по аналоговому входу *ADC*0, а формування вихідного сигналу відбувається за допомогою цифрових виходів *PC*7-*PC*0 та *PB*2-*PB*0. Живлення цифрової частини МК здійснюється за допомогою виводів *VCC* і *GND* (для зменшення перешкод по напрузі живлення використається конденсатор *C*6), а живлення аналогової частини – за допомогою виводів *AVCC* і *AGND* (для зменшення перешкод по напрузі живлення використаються дросель *L*1 та конденсатор *C*1). Для завдання тактової частоти МК використаються кварцовий резонатор *Y*1 та конденсатори *C*2, *C*3. Початкове скидання при подачі живлення забезпечується резистором *R*11 та конденсатором *C*5. ЦАП має 10-бітну паралельну шину *I*9-*I*0 та сигнал синхронізації *CLK*. Вихідний сигнал U_{OUT} знімається з виводу *OUT*. Живлення ЦАП здійснюється за допомогою виводів *VCC* і *GND* (для зменшення перешкод по напрузі живлення використається конденсатор *C7*).

ПОБУДОВА АЛГОРИТМІВ РОБОТИ СИСТЕМИ УПРАВЛІННЯ

Алгоритм є точно визначена процедура, відповідно до якої МК виконує певні дії щодо перетворення вихідних даних у перетворені вихідні дані. Тому розробка схеми вимагає точності і однозначності використовуваної атрибутики: символічних імен змінних, процедур, констант, портів вводу-виводу та інших елементів МК. За допомогою методу декомпозиції, при якому вся задача послідовно поділяється на менші функціональні модулі, кожен з модулів можна окремо від інших розробляти і налагоджувати.

В обох фільтрах використовуються 10-бітові відліки вхідного сигналу зі знаком. Фільтр *FIR* використовує 13-бітові коефіцієнти зі знаком, фільтр *IIR* використовує 12-бітові коефіцієнти зі знаком [4]. Це вимагає максимального розміру акумулятора 24 біта. Фільтри рекомендується реалізовувати на мові Асемблера з міркувань ефективності, але так, щоб функції фільтра можна було викликати з програми на мові *C*. Перед викликом функцій фільтра потрібно ініціалізувати пам'ять і елементи затримки – інакше початкові умови для роботи фільтра будуть невідомі. Всі параметри, які використаються для фільтрації, передаються під час виконання, тому функції можуть бути повторно використані для реалізації більш ніж одного фільтра без додаткових затрат пам’яті програм. Це може використовуватися для каскадного з'єднання фільтрів для отримання фільтра більш високого порядку. Однак, так як вихід кожного з фільтрів у каскаді масштабується в зворотному напрямі перед попаданням на вхід іншого фільтра, то є втрати точності між каскадами фільтрів.

На рис. 4 та рис. 5 наведені алгоритми обчислення для фільтрів *FIR* та *IIR* відповідно.

Рис. 4. Алгоритм обчислень для фільтра *FIR*

Алгоритм обчислень для фільтра *FIR* фактично є часткою алгоритму обчислень для фільтра *IIR*. Перший результат обчислюється простим

перемноженням між відповідними значеннями вхідного сигналу и коефіцієнту фільтра. Подальші результаті обчислюються з урахуванням попереднього значення акумулятору та із зсувом масиву вхідних даних.

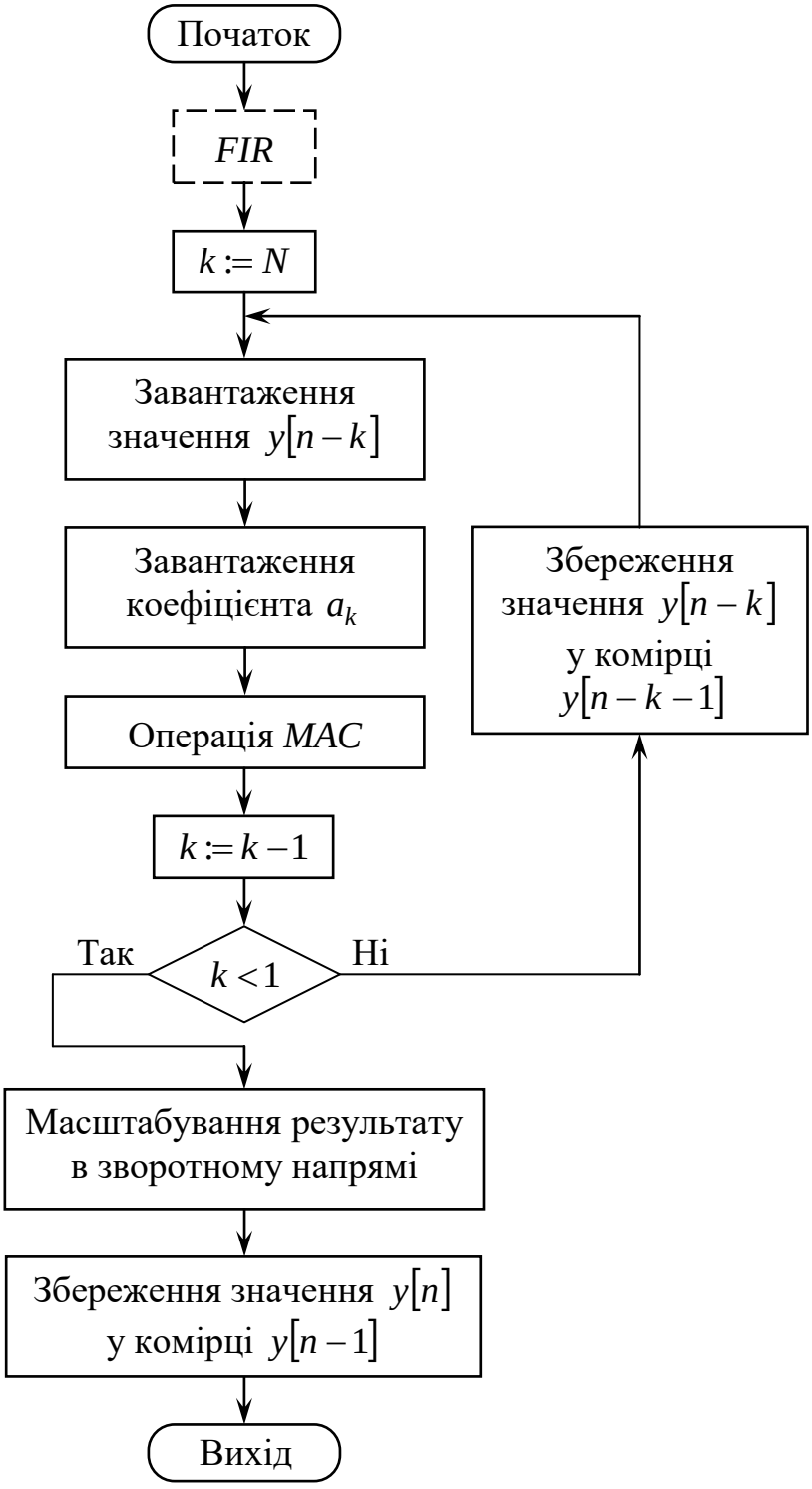

Рис. 5. Алгоритм обчислень для фільтра *IIR*

Цикл обчислень повторюється певну кількість раз (відповідно до порядку фільтра). Потім для фільтру *FIR* виконується масштабування вихідного значення в зворотному напрямі, а для фільтру *IIR* – продовження обчислень з використанням масиву вихідних даних.

Наведені алгоритми є загальними та не враховують деякі можливості для оптимізації програми з погляду швидкодії або об'єму пам'яті програм і даних. Для малих значеннях порядку фільтра алгоритми можуть бути перетворені до лінійних, що забезпечить збільшення швидкодії. Однак, це призводить до збільшення об'єму пам'яті програм. Зменшення об'єму пам'яті програм може бути досягнуто шляхом реалізації *MAC* як підпрограми замість макросу. Але це зменшить пропускну здатність фільтру за рахунок додаткового часу, який потрібен для роботи з підпрограмами. Зменшення об'єму пам'яті даних можливе у випадку розміщення значень коефіцієнтів у пам'яті програм.

Однак, для побудови фільтру необхідно розробити допоміжні алгоритми:

1. Алгоритм ініціалізації;
2. Алгоритм обробки переривання від таймера;
3. Алгоритм обробки переривання від АЦП.

Алгоритм ініціалізації виконується одноразово та забезпечує:

1. Ініціалізація стеку;
2. Завдання режиму роботи та напряму передачі інформації для виводів МК, яки використовуються у пристрої (*ADC*0 – аналоговий вхід, *PB*2-*PB*0 та *PC*7-*PC*0 – цифрові виходи);
3. Завдання режиму роботи таймера (наприклад, режим порівняння *A* таймера 1, попередній дільник відсутній) та дозвіл переривання від таймера (встановлення в 1 біта *OCIE*1*A* у регістрі *TIMSK*);

4. Завдання режиму роботи АЦП (частота модулю АЦП від 50 до 200 кГц, зовнішня опорна напруга), номер каналу – *ADC*0, дозвіл переривання по закінченню перетворення (встановлення в 1 біта *ADIE* у регістрі *ADCSR*) та запуск одноразового перетворення (встановлення в 1 біта *ADSC* у регістрі *ADCSR*);
5. Завдання початкових значень змінних;
6. Загальний дозвіл переривань;
7. Перехід до нескінченного циклу очікування переривань.

Алгоритм обробки переривання від таймера виконується багаторазово (з періодом дискретності, який визначаться ініціалізацією таймеру) та відповідає за наступні дії:

1. Запуск наступного одноразового перетворення (встановлення в 1 біта *ADSC* у регістрі *ADCSR*);
2. Визів процедури обчислень відповідно до типу та порядку фільтра;
3. Видача поточного рівня вихідного сигналу за допомогою ЦАП (передача поточного рівня у цифровому вигляді на виводи *PB*1-*PB*0 та *PC*7-*PC*0 та формування короткого позитивного імпульсу на виводі *PB*2).

Алгоритм обробки переривання від АЦП виконується багаторазово та відповідає за наступні дії:

1. Отримання значень після аналого-цифрового перетворення;
2. Збереження отриманих значень у комірці $x[n]$.

Оформлення допоміжних алгоритмів відбувається аналогічно оформленню основних алгоритмів. Якщо в курсовому проекті використовуються типові алгоритми (наприклад, алгоритм ділення), то ці алгоритми та відповідні процедури наводяться у додатках.

ПОБУДОВА ПРОГРАМИ СИСТЕМИ УПРАВЛІННЯ

Перетворення схеми алгоритму у текст програми для вибраного МК здійснюється з використанням інструментальних засобів розробки та налагодження програмного забезпечення. У даному проекті рекомендовано використовувати середовище *AVR Studio*, яке містить текстовий редактор, компілятор, модулі для симуляції роботи програми та засоби програмування МК. Система команд МК серії *AVR* наведена у Додатку А.

Далі наведено текст процедури, яка реалізує алгоритм обчислень для фільтра *IIR* 2-го порядку, з відповідними коментаріями. Текст процедури, яка реалізує алгоритм обчислень для фільтра *FIR* 2-го порядку, є складовою частиною процедури для фільтра *IIR* 2-го порядку. Реалізація процедури виконана так, що може використовуватися для різних фільтрів шляхом передачі адреси структури фільтра (*FILTER_POINTER*) та вхідної вибірки (*NDATAH*:*NDATAL*) в якості аргументів. Задіяні регістри: *R*0-*R*4, *R*16-*R*23, *Z*. Результат програми доступний у змінних *AC*2: *AC*1.

```
; Змінні
.def ZERO=R2              ; Для додавання прапора переносу
.def AC2=R17              ; Старший байт акумулятору
.def AC1=R16              ; Середній байт акумулятору
.def AC0=R3               ; Молодший байт акумулятору
.def FILTER_POINTER=R16   ; Адреса структури фільтра
.def NDATAL=R18      ; Молодший байт нової вибірки даних сигналу
.def NDATAH=R19           ; Старший байт нової вибірки даних сигналу
.def DATAL=R20            ; Молодший байт регістра для вибірок
.def DATAH=R21            ; Старший байт регістра для вибірок
.def COEFL=R22            ; Молодший байт регістра для коефіцієнтів
.def COEFH=R23            ; Старший байт регістра для коефіцієнтів
```

```
; Константи
; Зміщення для вхідних и вихідних значень у структурі фільтра
.equ   X1=0*2
.equ   X2=1*2
.equ   Y1=2*2
.equ   Y2=3*2
; Зміщення для коефіцієнтів у структурі фільтра
.equ   B0=4*2
.equ   B1=5*2
.equ   B2=6*2
.equ   A1=7*2
.equ   A2=8*2

; Макроси
.MACRO     LOAD_NODE          ; Завантаження значення
           ldd DATAL, Z+@0
           ldd DATAH, Z+@0+1
.ENDMACRO

.MACRO     UPDATE_NODE        ; Оновлення значення
           std Z+@0, DATAL
           std Z+@0+1, DATAH
.ENDMACRO

.MACRO     LOAD_COEF          ; Завантаження коефіцієнту
           ldd COEFL, Z+@0
           ldd COEFH, Z+@0+1
.ENDMACRO
```

```
.MACRO    MUL_MOV_24     ; Множення з 24-бітним результатом
          muls COEFH, DATAH
          mov AC2, R0
          mul COEFL, DATAL
          mov AC0, R0
          mov AC1, R1
          mulsu COEFH, DATAL
          add AC1, R0
          adc AC2, R1
          mulsu DATAH, COEFL
          add AC1, R0
          adc AC2, R1
.ENDMACRO

.MACRO    SMAC_24        ;Множення та додавання акумулятору
          muls COEFH, DATAH
          add AC2, R0
          mul COEFL, DATAL
          add AC0, R0
          adc AC1, R1
          adc AC2, ZERO
          mulsu COEFH, DATAL
          add AC1, R0
          adc AC2, R1
          mulsu DATAH, COEFL
          add AC1, R0
          adc AC2, R1
.ENDMACRO
```

```
; Функція IIR-фільтра
IIR2:
        clr ZERO                        ; Очищення регістра ZERO
        movw ZL, FILTER_POINTER         ; Копіювання адреси в покажчик
; обчислення B2*x[n-2]
        LOAD_COEF B2
        LOAD_NODE X2
        MUL_MOV_24
; обчислення B1*x[n-1]
        LOAD_COEF B1
        LOAD_NODE X1
        UPDATE_NODE X2
        SMAC_24
; обчислення B0*x[n]
        LOAD_COEF B0
        movw DATAL, NDATAL
        UPDATE_NODE X1
        SMAC_24
; обчислення A2*y[n-2]
        LOAD_COEF A2
        LOAD_NODE Y2
        SMAC_24
; обчислення A1*y[n-1]
        LOAD_COEF A1
        LOAD_NODE Y1
        UPDATE_NODE Y2
        SMAC_24
```

; Масштабування вихідного значення в зворотному напрямі на 2^{11}

*asr AC*2

*ror AC*1

*asr AC*2

*ror AC*1

*asr AC*2

*ror AC*1

; Оновлення значення за зміщенням *Y*1

std Z+*Y*1, *AC*1

std Z+*Y*1+1, *AC*2

ret

Після оформлення допоміжних алгоритмів та відповідних ділянок програми необхідно виконати моделювання програми у цілому. Для цього в середовищі розробки (наприклад, *AVR Studio*) завдаються параметри симуляції – модель і тактова частота МК. Також рекомендується встановити точки зупину для відстеження параметрів роботи МК на найбільш важливих ділянках програми. Надалі здійснюється запуск процесу симуляції в покроковому режимі або в режимі вільного виконання. Для імітації зовнішніх сигналів користувач може взаємодіяти з відповідними стимуляторами периферійних пристроїв МК або виконувати запис потрібних значень безпосередньо у РЗП, що використовуються у відповідних ділянках програми.

У разі наявності помилок виконання програми або отримання помилкових даних потрібно локалізувати ділянку програми, яка містить помилку, та виправити її. Після виправлення помилки процес симуляції повторюється.

ДОДАТКИ

Додаток А. Система команд мікроконтролерів серії *AVR* [4]

Таблиця А.1 Система команд

Мнемокод команди	Опис команди	Алгоритм команди	Зміна прапорів
1	**2**	**3**	**4**
КОМАНДИ АРИФМЕТИЧНИХ ОПЕРАЦІЙ			
ADD Rd,Rr	Складання двох РЗП	*Rd←Rd+Rr*	*Z,C,N, V,H*
ADC Rd,Rr	Складання двох РЗП з позикою	*Rd←Rd+Rr++C*	*Z,C,N, V,H*
ADIW Rdl,K	Складання слова з константою	*Rdh:Rdl← ←Rdh:Rdl+K*	*Z,C,N, V,S*
SUB Rd,Rr	Віднімання двох РЗП	*Rd←Rd-Rr*	*Z,C,N, V,H*
SUBI Rd,K	Віднімання константи з РЗП	*Rd←Rd-K*	*Z,C,N, V,H*
SBC Rd,Rr	Віднімання двох РЗП з позикою	*Rd←Rd-Rr-C*	*Z,C,N, V,H*
SBCI Rd,K	Віднімання константи з РЗП з позикою	*Rd←Rd-K-C*	*Z,C,N, V,H*
SBIW Rdl,K	Віднімання константи із слова	*Rdh:Rdl← ←Rdh:Rdl-K*	*Z,C,N, V,S*
INC Rd	Інкремент РЗП	*Rd←Rd*+1	*Z,N,V*
DEC Rd	Декремент РЗП	*Rd←Rd*-1	*Z,N,V*

Продовження табл. А.1

1	2	3	4
КОМАНДИ ЛОГІЧНИХ ОПЕРАЦІЙ			
MUL Rd,Rr	Множення беззнакових чисел	*R*1:*R*0← ←*Rd*×*Rr*	*Z,C*
MULS Rd,Rr	Множення знакових чисел	*R*1:*R*0← ←*Rd*×*Rr*	*Z,C*
MULSU Rd,Rr	Множення знакового та беззнакового чисел	*R*1:*R*0← ←*Rd*×*Rr*	*Z,C*
FMUL Rd,Rr	Множення дробових беззнакових чисел	*R*1:*R*0← ← *(Rd*×*Rr)*<<1	*Z,C*
FMULS Rd,Rr	Множення дробових знакових чисел	*R*1:*R*0← ←*(Rd*×*Rr)*<<1	*Z,C*
FMULSU Rd,Rr	Множення дробових знакового та беззнакового чисел	*R*1:*R*0← ← *(Rd*×*Rr)*<<1	*Z,C*
AND Rd,Rr	«Логічне І» двох РЗП	*Rd*←*Rd*∧*Rr*	*Z,N,V*
ANDI Rd,K	«Логічне І» РЗП та константи	*Rd*←*Rd*∧*K*	*Z,N,V*
OR Rd,Rr	«Логічне АБО» двох РЗП	*Rd*←*Rd*∨*Rr*	*Z,N,V*
ORI Rd,K	«Логічне АБО» РЗП та константи	*Rd*←*Rd*∨*K*	*Z,N,V*
EOR Rd,Rr	«Виключне АБО» двох РЗП	*Rd*←*Rd*⊕*Rr*	*Z,N,V*
COM Rd	Переведення в зворотний код	*Rd*←0*xFF-Rd*	*Z,C,N,V*
NEG Rd	Переведення в додатковий код	*Rd*←0*x*00-*Rd*	*Z,C,N, V,H*

Продовження табл. А.1

1	2	3	4
SBR Rd,K	Установлення бітів у РЗП	$Rd \leftarrow Rd \vee K$	*Z,N,V*
CBR Rd,K	Скидання бітів у РЗП	$Rd \leftarrow Rd \wedge \wedge (0xFF-K)$	*Z,N,V*
TST Rd	Перевірка РЗП на від'ємне або нульове значення	$Rd \leftarrow Rd \wedge Rd$	*Z,N,V*
CLR Rd	Скидання усіх бітів РЗП	$Rd \leftarrow Rd \oplus Rd$	*Z,N,V*
SER Rd	Установлення усіх бітів РЗП	$Rd \leftarrow 0xFF$	Не діє
КОМАНДИ ЗМІЩЕНЬ			
LSL Rd	Логічне зміщення ліворуч	$Rd(n+1) \leftarrow Rd(n)$, $Rd(0) \leftarrow 0$	*Z,C,N,V*
LSR Rd	Логічне зміщення праворуч	$Rd(n) \leftarrow Rd(n+1)$, $Rd(7) \leftarrow 0$	*Z,C,N,V*
ROL Rd	Зміщення ліворуч через перенос	$Rd(0) \leftarrow C$, $Rd(n+1) \leftarrow Rd(n)$, $C \leftarrow Rd(7)$	*Z,C,N,V*
ROR Rd	Зміщення праворуч через перенос	$Rd(7) \leftarrow C$, $Rd(n) \leftarrow Rd(n+1)$, $C \leftarrow Rd(0)$	*Z,C,N,V*
ASR Rd	Арифметичне зміщення праворуч	$Rd(n) \leftarrow Rd(n+1)$, $n \leftarrow 0..6$	*Z,C,N,V*
КОМАНДИ ПЕРЕДАЧІ КЕРУВАННЯ			
RJMP k	Відносний безумовний перехід	$PC \leftarrow PC+k+1$	Не діє
IJMP	Непрямий безумовний перехід	$PC \leftarrow Z$	Не діє

Продовження табл. А.1

1	2	3	4
JMP k	Абсолютний перехід	*PC←k*	Не діє
RCALL k	Відносний виклик підпрограми	*PC←PC+k*+1	Не діє
ICALL	Непрямий виклик підпрограми	*PC←Z*	Не діє
CALL k	Абсолютний виклик підпрограми	*PC←k*	Не діє
RET	Повернення із підпрограми	*PC←STACK*	Не діє
RETI	Повернення із підпрограми обробки переривання	*PC←STACK*	*I*
CPSE Rd,Rr	Порівняння та пропуск наступної команди, якщо числа рівні	Якщо *Rd=Rr* *PC←PC*+2/3	Не діє
CP Rd,Rr	Порівняння двох РЗП	*Rd-Rd*	*Z,N,V, C,H*
CPC Rd,Rr	Порівняння двох РЗП з урахуванням переносу	*Rd-Rr-C*	*Z,N,V, C,H*
CPI Rd,K	Порівняння РЗП з константою	*Rd-K*	*Z,N,V, C,H*
SBRC Rr,b	Пропуск наступної команди, якщо біт *b* РЗП скинутий	Якщо *Rr(b)*=0 *PC←PC*+2/3	Не діє
SBRS Rr,b	Пропуск наступної команди, якщо біт *b* РЗП встановлено	Якщо *Rr(b)*=1 *PC←PC*+2/3	Не діє
SBIC P,b	Пропуск наступної команди, якщо біт *b* РВВ скинутий	Якщо *P(b)*=0 *PC←PC*+2/3	Не діє

Продовження табл. А.1

1	2	3	4
SBIS P,b	Пропуск наступної команди, якщо біт *b* РВВ встановлено	Якщо *P(b)*=1 *PC*←*PC*+2/3	Не діє
BRBS s,k	Перехід, якщо біт *s* у регістрі статусу встановленв	Якщо *SREG(s)*=1 *PC*←*PC*+*k*+1	Не діє
BRBC s,k	Перехід, якщо біт *s* у регістрі статусу скинутий	Якщо *SREG(s)*=0 *PC*←*PC*+*k*+1	Не діє
BREQ k	Перехід, якщо числа рівні	Якщо *Rd=Rr* *(Z*=1*)* *PC*←*PC*+*k*+1	Не діє
BRNE k	Перехід, якщо числа не рівні	Якщо *Rd≠Rr* *(Z*=0*)* *PC*←*PC*+*k*+1	Не діє
BRCS k	Перехід, якщо прапор переносу установлений	Якщо *C*=1 *PC*←*PC*+*k*+1	Не діє
BRCC k	Перехід, якщо прапор переносу скинутий	Якщо *C*=0 *PC*←*PC*+*k*+1	Не діє
BRSH k	Перехід, якщо перше беззнакове число не менше за друге	Якщо *Rd≥Rr* *(C*=0*)* *PC*←*PC*+*k*+1	Не діє
BRLO k	Перехід, якщо перше беззнакове число менше за друге	Якщо *Rd<Rr* *(C*=1*)* *PC*←*PC*+*k*+1	Не діє

Продовження табл. А.1

1	2	3	4
BRMI k	Перехід, якщо прапор негативного значення установлений	Якщо $N=1$ $PC \leftarrow PC+k+1$	Не діє
BRPL k	Перехід, якщо прапор негативного значення скинутий	Якщо $N=0$ $PC \leftarrow PC+k+1$	Не діє
BRGE k	Перехід, якщо перше знакове число не менше за друге	Якщо $Rd \geq Rr$ $(N \oplus V=0)$ $PC \leftarrow PC+k+1$	Не діє
BRLT k	Перехід, якщо перше знакове число менше за друге	Якщо $Rd<Rr$ $(N \oplus V=1)$ $PC \leftarrow PC+k+1$	Не діє
BRHS k	Перехід, якщо прапор напівпереносу установлений	Якщо $H=1$ $PC \leftarrow PC+k+1$	Не діє
BRHC k	Перехід, якщо прапор напівпереносу скинутий	Якщо $H=0$ $PC \leftarrow PC+k+1$	Не діє
BRTS k	Перехід, якщо прапор пересилання установлений	Якщо $T=1$ $PC \leftarrow PC+k+1$	Не діє
BRTC k	Перехід, якщо прапор пересилання скинутий	Якщо $T=0$ $PC \leftarrow PC+k+1$	Не діє
BRVS k	Перехід, якщо прапор переповнення установлений	Якщо $V=1$ $PC \leftarrow PC+k+1$	Не діє
BRVC k	Перехід, якщо прапор переповнення скинутий	Якщо $V=0$ $PC \leftarrow PC+k+1$	Не діє

Продовження табл. А.1

1	2	3	4
BRIE k	Перехід, якщо глобальне переривання дозволено	Якщо *I*=1 *PC*←*PC*+*k*+1	Не діє
BRID	Перехід, якщо глобальне переривання заборонено	Якщо *I*=0 *PC*←*PC*+*k*+1	Не діє
КОМАНДИ ПЕРЕСИЛАННЯ ДАНИХ			
MOV Rd,Rr	Пересилання між РЗП	*Rd*←*Rr*	Не діє
MOVW Rd,Rr	Пересилання слів	*Rd*+1:*Rd*←*Rr*+1:*Rr*	Не діє
LDI Rd,K	Завантаження константи до РЗП	*Rd*←*K*	Не діє
LD Rd,X	Непряме завантаження	*Rd*←*[X]*	Не діє
LD Rd,X+	Непряме завантаження з постінкрементом	*Rd*←*[X]*, *X*←*X*+1	Не діє
LD Rd,-X	Непряме завантаження з преддекрементом	*X*←*X*-1, *Rd*←*[X]*	Не діє
LD Rd,Y	Непряме завантаження	*Rd*←*[Y]*	Не діє
LD Rd,Y+	Непряме завантаження з постінкрементом	*Rd*←*[Y]*, *Y*←*Y*+1	Не діє
LD Rd,-Y	Непряме завантаження з преддекрементом	*Y*←*Y*-1, *Rd*←*[Y]*	Не діє
LDD Rd,Y+q	Непряме відносне завантаження	*Rd*←*[Y+q]*	Не діє
LD Rd,Z	Непряме завантаження	*Rd*←*[Z]*	Не діє
LD Rd,Z+	Непряме завантаження з постінкрементом	*Rd*←*[Z]*, *Z*←*Z+1*	Не діє

Продовження табл. А.1

1	2	3	4
LD Rd,-Z	Непряме завантаження з преддекрементом	*Z←Z-1, Rd←[Z]*	Не діє
LDD Rd,Z+q	Непряме відносне завантаження	*Rd←[Z+q]*	Не діє
LDS Rd,k	Безпосереднє завантаження з ОЗП	*Rd←[k]*	Не діє
ST X,Rr	Непрямий запис	*[X]←Rr*	Не діє
ST X+,Rr	Непрямий запис з постінкрементом	*[X]←Rr, X←X*+1	Не діє
ST –X,Rr	Непрямий запис з преддекрементом	*X←X*-1, *[X]←Rr*	Не діє
ST Y,Rr	Непрямий запис	*[Y]←Rr*	Не діє
ST Y+,Rr	Непрямий запис з постінкрементом	*[Y]←Rr, Y←Y*+1	Не діє
ST –Y,Rr	Непрямий запис з преддекрементом	*Y←Y*-1, *[Y]←Rr*	Не діє
STD Y+q,Rr	Непрямий відносний запис	*[Y+q]←Rr*	Не діє
ST Z,Rr	Непрямий запис	*[Z]←Rr*	Не діє
ST Z+,Rr	Непрямий запис з постінкрементом	*[Z]←Rr, Z←Z*+1	Не діє
ST –Z,Rr	Непрямий запис з преддекрементом	*Z←Z*-1, *[Z]←Rr*	Не діє
STD Z+q,Rr	Непрямий відносний запис	*[Z+q]←Rr*	Не діє
STS k,Rr	Безпосередній запис в ОЗП	*[k]←Rr*	Не діє

Продовження табл. А.1

1	2	3	4
LPM	Завантаження даних із пам’яті програм	*R*0←*[Z]*	Не діє
LPM Rd,Z	Завантаження даних із пам’яті програм	*Rd*←*[Z]*	Не діє
LPM Rd,Z+	Завантаження даних із пам’яті програм з постінкрементом	*Rd*←*[Z], Z*←*Z*+1	Не діє
SPM	Запис в пам’ять програм	*[Z]*←*R*1*:R*0	Не діє
IN Rd,P	Пересилання із РВВ в РЗП	*Rd*←*P*	Не діє
OUT P,Rr	Пересилання із РЗП в РВВ	*P*←*Rr*	Не діє
PUSH Rr	Пересилання із РЗП в стек	*STACK*←*Rr*	Не діє
POP Rd	Пересилання із стеку в РЗП	*Rd*←*STACK*	Не діє
КОМАНДИ ОПЕРАЦІЙ З БІТАМИ			
SBI P,b	Установлення біта *b* РВВ	*P(b)*←1	Не діє
CBI P,b	Скидання біта *b* РВВ	*P(b)*←0	Не діє
SWAP Rd	Обмін місцями тетрад в РЗП	*Rd*(3..0)← ←*Rd*(7..4), *Rd*(7..4)← ←*Rd*(3..0)	Не діє
BSET s	Установлення біта *s* у регістрі статусу	*SREG(s)*←1	*SREG(s)*
BCLR s	Скидання біта *s* у регістрі статусу	*SREG(s)*←0	*SREG(s)*
BST Rr,b	Запис біта РЗП в прапор пересилання	*T*←*Rr(b)*	*T*

Продовження табл. А.1

1	2	3	4
BLD Rd,b	Завантаження біта РЗП із прапора пересилання	*Rd(b)*←*T*	Не діє
SEC	Установлення прапора *C*	*C*←1	*C*
CLC	Скидання прапора *C*	*C*←0	*C*
SEN	Установлення прапора *N*	*N*←1	*N*
CLN	Скидання прапора *N*	*N*←0	*N*
SEZ	Установлення прапора *Z*	*Z*←1	*Z*
CLZ	Скидання прапора *Z*	*Z*←0	*Z*
SEI	Загальний дозвіл переривань	*I*←1	*I*
CLI	Загальна заборона переривань	*I*←0	*I*
SES	Установлення прапора *S*	*S*←1	*S*
CLS	Скидання прапора *S*	*S*←0	*S*
SEV	Установлення прапора *V*	*V*←1	*V*
CLV	Скидання прапора *V*	*V*←0	*V*
SET	Установлення прапора *T*	*T*←1	*T*
CLT	Скидання прапора *T*	*T*←0	*T*
SEH	Установлення прапора *H*	*H*←1	*H*
CLH	Скидання прапора *H*	*H*←0	*H*
КОМАНДИ КЕРУВАННЯ СИСТЕМОЮ			
NOP	Нема операції	–	Не діє
SLEEP	Перехід в «сплячий» режим	–	Не діє
WDR	Скидання сторожового таймера	–	Не діє

ЗМІСТ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. **Белов А. В.** Конструирование устройств на микроконтроллерах / А. В. Белов. – С.-П.: Наука и техника, 2005. – 256 с.
2. **Голубцов М. С.** Мікроконтролери AVR: від простого до складного / М. С. Голубцов. – М.: Солон-Прес, 2003. – 288с.
3. **Евстифеев А. В.** Микроконтроллеры семейств Tiny и Mega фирмы Atmel / А. В. Евстифеев**.** – М.: “Додека-XXI”, 2004. – 366 с.
4. 8-bit AVR Microcontroller with 16K Bytes In-System Programmable Flash ATMega16. – Atmel Corporation. 2008. – 358 p.