ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА КОМП'ЮТЕРНА ТЕХНІКА

УДК 004.438

Ю. В. Шевчук[1]
Д. П. Проценко[1]
А. І. Міхєєв[2]

ДОСЛІДЖЕННЯ ПРОДУКТИВНОСТІ РОБОТИ STM32F407VGT6 З .NET MF

[1]Вінницький національний технічний університет
[2]ВП «Южно-Українська АЕС»

Проведено дослідження втрат швидкодії процесора STM32f407VGT6 програмованої платформи STMF4DISCOVERY з використанням .Net Micro Framework для реалізації інженерних та наукових алгоритмічних задач. Здійснено оцінку впливу кількості вихідних даних на швидкість виконання алгоритмів.

Ключові слова: мікроконтролер, embedded, hardware, .Net MF, bench-mark тест, TinyCLR, швидкодія.

Вступ

Нові наукові результати в багатьох галузях науки, техніки та технологій на сьогодні досягнуто завдяки розвитку та широкому використанню мікропроцесорної техніки, інтенсивну еволюцію якої спостерігає уже декілька поколінь людей. Тому для сучасних науковців важливо постійно відкривати для себе нові можливості засобів мікропроцесорної техніки, які дають змогу по-новому розглянути ті чи інші наукові проблеми чи задачі та в перспективі отримати нові, покращені результати.

Розробка прототипів вбудованих систем керування, контролю чи діагностики з нуля і програмування їх на C++, C, або на асемблері може виявитися посильним завданням не для всіх науковців, які хочуть перевірити експериментально роботу запропонованих методів, моделей, алгоритмів, тощо. Розробники embedded платформ пишуть код, який є дуже близьким до hardware і безпосередньо взаємодіє з пам'яттю. Тому програма, написана для одного контролера на іншій платформі не працюватиме, навіть якщо ядра процесорів такі ж самі. Засоби для відлагодження роботи пристроїв є складними у використанні, що також обмежує коло спеціалістів. Інша справа — використання технології .NET Micro Framework (.Net MF), яка описує апаратні компоненти, як об'єкти, що дозволяє звертатись до них, використовуючи базові класи. Тобто у програмуванні використовується об'єктно-орієнтована модель, замість того, щоб вникати в деталі архітектури мікроконтролера. А для налаштування периферії, потрібно встановити властивості того чи іншого об'єкта. Цей підхід робить запропоновану реалізацію незалежною від конкретної платформи [1, 2]. Програмування embedded платформ з використанням .Net MF можна здійснювати за допомогою C# і Visual Studio. Якщо наявний елементарний досвід програмування на вказаній мові будь-яких програм для персональних комп'ютерів, серверів чи мобільних пристроїв, то процес вивчення .Net MF та програмування мікроконтролерів із застосуванням приведеної технології не викличе ніяких труднощів. Для функціонування .Net MF на програмовану платформу не потрібно встановлювати операційну систему. Для використання технології в мікроконтролер потрібно записати врізану версію Common Language Runtime (TinyCLR) [3]. Згадане середовище займає невеликий обсяг пам'яті, обмежений кількома сотнями кілобайтів оперативної пам'яті, також не є необхідною наявність блока керування пам'яттю (Memory Management Unit). Таким чином, .Net MF оболонка може працювати на недорогих 32-розрядних мікроконтролерах, не споживаючи при цьому енергії більше, ніж під час виконання інструкцій, написаних на native-коді. Крім того зазначена технологія в розробці мікропроцесорних пристроїв дозволяє користуватися ще однією важливою перевагою, такою як managed code.

Але використання TinyCLR та .Net MF оболонки в роботі мікроконтролера зменшує його швидкодію та продуктивність функціонування. Це та ціна, яку потрібно платити за зручність та простоту використання вищезгаданої технології. В проаналізованих наукових джерелах та технічній

літературі не було знайдено інформації про оцінку швидкодії роботи мікроконтролерів із TinyCLR та .Net MF технологією на борту.

Отже, *метою роботи* є дослідження ефективності роботи мікропроцесора STM32F407VGT6 із .Net MF програмною оболонкою. Для досягнення поставленої мети потрібно реалізувати ряд тестових алгоритмів на native code та managed code. Традиційно для дослідження ефективності роботи мікроконтролерів використовують bench-mark програми з подальшим вимірюванням обсягу коду та часу виконання програм. Серед програмних інструкцій в програмах для мікроконтролерів в середньому найбільшу частку складають оператори присвоювання (49 %), виклик процедур — 18 %, цикли — 17 %, умовні оператори — 16 % [4]. Оператори присвоювання, в свою чергу, містять 55 % логічних операцій, 45 % арифметичних операцій. Тому bench-mark тести будемо формувати з урахуванням вищенаведеного аналізу.

Результати дослідження

Проведемо дослідження втрат продуктивності у разі використання .Net MF на прикладі програмованої відлагоджувальної плати stmf4discovery, яка містить STM32F407VGT6 – ARM Cortex-M4 32-bit мікроконтролер із 1MB Flash/192+4 KB RAM, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs [5]. Наявний на платі мікроконтролер може працювати на максимальній частоті тактування 168 МГц, та як показано в переліку характеристик має насичений сет периферійних пристроїв, що створює передумови активного застосування зазначеної платформи для вирішення сучасних наукових задач.

Для здійснення оцінки швидкодії реалізуємо ряд алгоритмів, які в своїх програмних реалізаціях на C++ та C# будуть містити тестові оператори [4]. Bench-mark тест міститиме в собі оператор циклу, в якому буде здійснюватись та чи інша арифметична операція до тієї пори, поки не виконається певна умова. В результаті виконання умови на одному із виводів порта введення/виведення буде встановлюватись високий логічний рівень на певний час. Після цього все має повторюватись, а тривалість паузи, яку можна буде спостерігати на осцилографі, буде характеризувати час затрачений процесором на виконання тестового циклу арифметичних операцій.

На рис. 1 показано тестовий алгоритм, в якому виконуються арифметичні операції додавання та присвоєння. Запропонований алгоритм реалізований на native-інструкціях та з використанням .Net MF. Реалізація наведеного тестового алгоритму (див. рис. 1) за допомогою C++ показана на лістингу 1 та — за допомогою C# на лістингу 2.

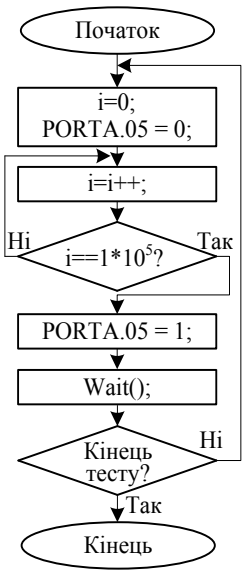

Рис. 1. Тестовий алгоритм додавання та присвоєння

Лістинг 1 — Реалізація першого алгоритму на мові програмування C++

```
i=0;
while(1)
{
    i = i+1;
    if(i > 100000)
    {
      GPIO_SetBits(GPIOA, GPIO_Pin_0);
      Delay(500000);//delay ~100mS

      GPIO_ResetBits(GPIOA, GPIO_Pin_0);
      i = 0;
    }
}
```

Лістинг 2 — Реалізація першого алгоритму на мові програмування C#

```
public static void Main()
{
```

Продовження лістингу 2

```
        OutputPort led_1 = new OutputPort(Cpu.Pin.GPIO_Pin5, false);
        int i = 0;
        while (true)
        {       i++;
                if (i == 100000)
                {
                        led_1.Write(true);
                        Thread.Sleep(100);
                        led_1.Write(false);
                        i = 0;
                }
        }
}
```

Результат роботи програмованої платформи бачимо на рис. 2 та на рис. 3, відповідно, у разі реалізації тестового алгоритму на C++ та на C#. Час виконання тестової частини t_{test} складає $t_{period} - t_{wait}$. В першому випадку $t_{test\_c++} = 414$ мс, в другому — $t_{test\_c\#} = 975$ мс.

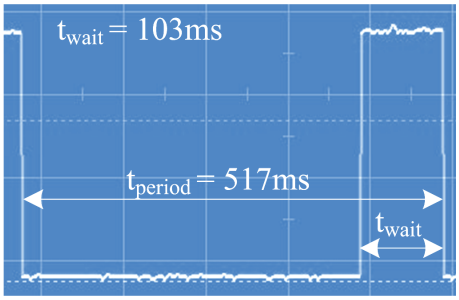

Рис. 2. Осцилограма часових результатів виконання операцій присвоєння та додавання при реалізації на C++

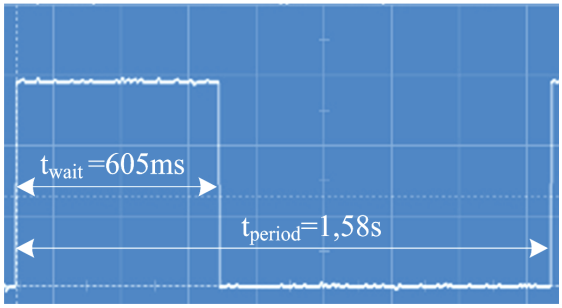

Рис. 3. Осцилограма часових результатів виконання операцій присвоєння та додавання при реалізації на C#

На рис. 4 показаний тестовий алгоритм, в якому передбачено виконання операцій множення та присвоєння. Результат роботи процесора бачимо на рис. 5 та на рис. 6, відповідно. Час виконання тестової частини t_{test} складає $t_{period} - t_{wait}$. В першому випадку $t_{test\_c++} = 3$ мс, в другому — $t_{test\_c\#} = 76$ мс.

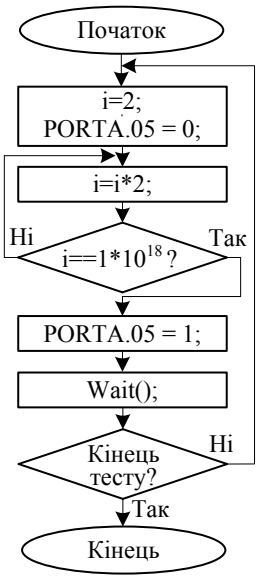

Рис. 4. Тестовий алгоритм операцій присвоєння та множення

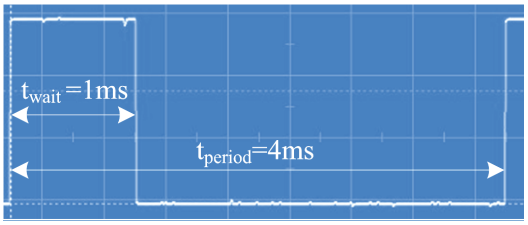

Рис. 5. Осцилограма часових результатів виконання операцій присвоєння та множення у разі реалізації на C++

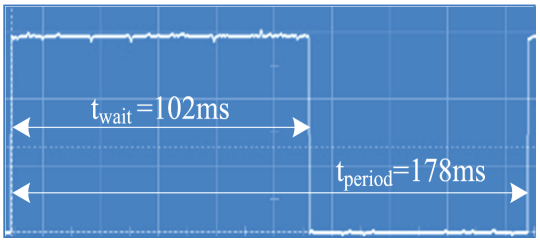

Рис. 6. Осцилограма часових результатів виконання операцій присвоєння та множення у разі реалізації на C#

На рис. 7 показаний тестовий алгоритм, який описує послідовність виконання арифметичних операцій множення та присвоєння. Результат роботи програмованої платформи бачимо на рис. 8 та на рис. 9, відповідно. У випадку C++ — $t_{test\_c++} = 38$ мс, C# — $t_{test\_c\#} = 903$ мс.

Тестовий алгоритм для дослідження швидкодії виконання арифметичних операцій ділення та операцій присвоєння показаний на рис. 10. На рис. 11 та 12 приведений результат роботи програмованої платформи, відповідно, для першого та другого випадку. В першому випадку $t_{test\_c++} = 3{,}5$ мс, в другому — $t_{test\_c\#} = 85$ с.

Таким чином отримані результати для наочності можна звести в таблицю.

Порівняння результатів виконання тестових алгоритмів

№ тесту	Час виконання на C++ (*t*, мс)	Час виконання на C# (*t*, мс)	Втрата швидкодії (*n*, рази)
1	414	975	2,36
2	3	76	25,33
3	38	903	23,76
4	3,5	85	24,29

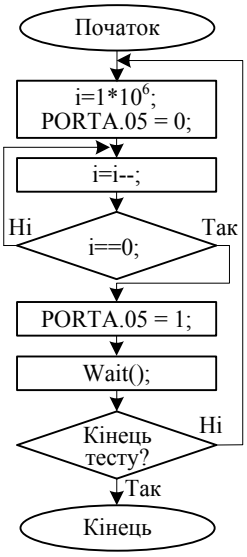

Рис. 7. Тестовий алгоритм операцій присвоєння та віднімання

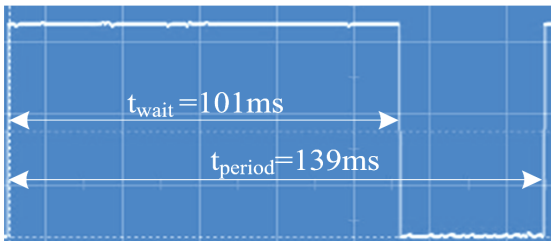

Рис. 8. Осцилограма часових результатів виконання операцій присвоєння та віднімання у разі реалізації на C++

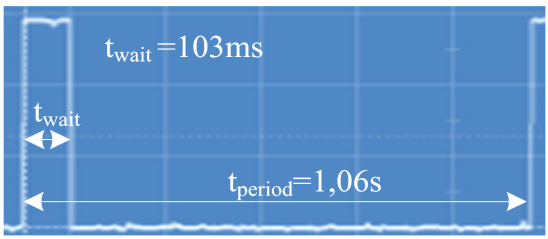

Рис. 9. Осцилограма часових результатів виконання операцій присвоєння та віднімання у разі реалізації на C#

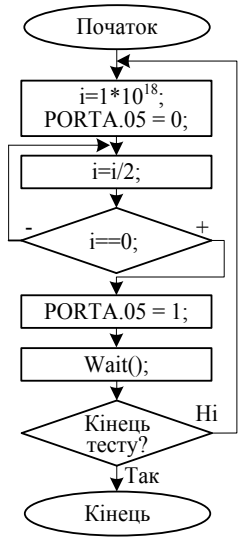

Рис. 10. Тестовий алгоритм операцій присвоєння та ділення

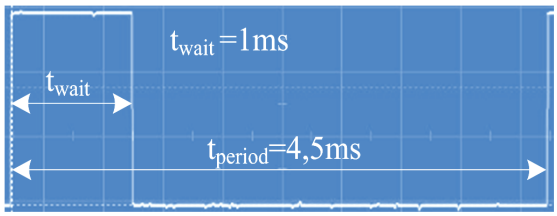

Рис. 11. Осцилограма часових результатів виконання операцій присвоєння та ділення у разі реалізації на C++

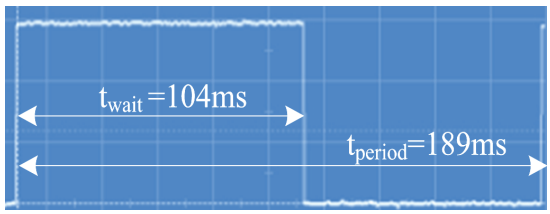

Рис. 12. Осцилограма часових результатів виконаннія операцій присвоєння та ділення у разі реалізації на C#

Із даних таблиці видно, що з використанням .Net MF спостерігається значне зниження швидкодії у порівнянні із реалізаціями на C++ у випадках всіх наведених тестових алгоритмів. Отже, програмування на C# може мати місце у випадках реалізації взаємодії периферійних пристроїв мікро-контролера та простих обчислень або в задачах керування об'єктами з тривалою зміною контрольованих параметрів.

Для адекватнішої оцінки продуктивності досліджуваного мікроконтролера також здійснимо реалізацію алгоритму зі складністю $O(n^2)$ на вищевказаних мовах програмування, причому для різної кількості вхідних даних. Таким алгоритмом для нашого дослідження може слугувати алгоритм бульбашкового сортування не впорядкованого масиву випадкових даних. Аналогічно до попередніх тестів, сортування також реалізуємо на двох мовах програмування.

На рис. 13 показані графіки залежності часу виконання тестових програм залежно від кількості вихідних даних масиву, побудовані в логарифмічному масштабі. Крива 1 описує реалізації з використанням .Net MF, 2 — з використанням native інструкцій.

Із графіків (див. рис. 13) випливає, що кількість оброблюваних даних майже не впливає на зміну швидкодії мікроконтролера у порівнянні двох зазначених реалізацій. Тому, розробляючи програмну частину тієї чи іншої мікропроцесорної системи, можна орієнтуватись на стале значення втрати швидкодії.

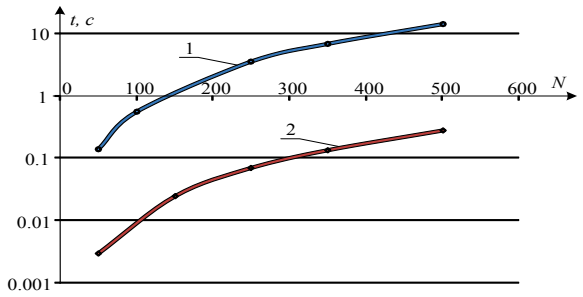

Рис. 13. Графіки залежності швидкодії виконання алгоритму бульбашкового сортування та ділення

Висновки

Проведені дослідження показали, що використання .Net MF дає значне зниження швидкодії у порівнянні із реалізаціями на C++ у тестових алгоритмах, які базуються на виконанні заданої кількості арифметичних операцій та операторів присвоєння. Також можна стверджувати, що кількість оброблюваних даних майже не впливає на зміну швидкодії мікроконтролера у разі порівняння двох зазначених реалізацій. Тому, розробляючи програмну частину тієї чи іншої мікропроцесорної системи, можна орієнтуватись на стале значення втрати швидкодії. Отже програмування на C# може використовуватись у випадках реалізації взаємодії периферійних пристроїв мікроконтролера та простих обчислень або в задачах керування об'єктами з тривалою зміною контрольованих параметрів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Jens Kühner. Expert .NET Micro Framework. Second Edition**;** [Lead Editor: Jonathan Hassell]. — USA. Apress, 2009. — 482 p — ISBN-13 (pbk): 978-1-4302-2387-0. — ISBN-13 (electronic): 978-1-4302-2388-7.

2. Reduce time to market and development costs using .net micro framework on blackfin processors. Drastically reduce development time and cost. [Електронний ресурс] // Analog devices. — Режим доступу : http://www.analog.com/static/imported-files/white_papers/DotNet_Micro_Frmwrk_onBlackfin_WP.PDF.

3. Beginners guide to c# and micro framework. running netmf applications on hardwar. [Електронний ресурс]. — Режим доступу : http://www.ghielectronics.com/downloads/FEZ/Beginne rs%20guide%20t o%20NETMF.pdf

4. Моршнев В. В. Оценка и анализ эффективности архитектуры микроконтроллеров. Методика оценки эффективности мк. [Электронный ресурс] / В. В. Моршнев. — Режим доступа: http://www.moko.ru/ mc/RatingMC.pdf.

5. STM32F405xx - STM32F407xx. Datasheet — production data. [Електронний ресурс]. — Режим доступу: http://www.st.com/web/en/resource/technical/document/ datasheet/DM00037051.pdf

Рекомендована кафедрою електромеханічних систем автоматизації в промисловості і на транспорті

Стаття надійшла до редакції 9.04.2014

Шевчук Юрій Володимирович — канд. техн. наук, старший викладач кафедри електромеханічних систем автоматизації в промисловості і на транспорті, e-mail: yuriy.shevchuck@gmail.com;

Проценко Дмитро Петрович — канд. техн. наук., доцент кафедри електромеханічних систем автоматизації в промисловості і на транспорті.

Вінницький національний технічний університет, Вінниця;

Міхєєв Андрій Іванович — інженер з релейного захисту та автоматики.

ВП «Южно-Українська АЕС», Южно-Українськ

Yu. V. Shevchuk[1]
D. P. Protsenko[1]
A. I. Mikhieev[2]

Research of the work productivity of STM32F407VGT6 with .Net MF

[1]Vinnytsia National Technical University
[2]South Ukraine Nuclear Power Plant, Yuzhno-Ukrainsk

The losses of programmable platforms using STMF4DISCOVERY CPU — STM32F407VGT6 with using of .Net Micro Framework for the implementation of engineering and scientific algorithmic problems are studied in the paper. The impact baseline data number on the algorithms performance is estimated.

Keywords: microcontroller, embedded, hardware, NET MF, bench-mark test, TinyCLR, performance.

Shevchuk Yurii V. — Cand. Sc. (Eng.) Senior Lecturer of the Chair of Electromechanics Systems of Automation in Industry and on Transport, e-mail: yuriy.shevchuck@gmail.com;

Protsenko Dmytro P. — Cand. Sc. (Eng.), Assistant Professor of the Chair of Electromechanics Systems of Automation in Industry and on Transport;

Mikhieev Andrii I. — Engineer of relay protection and automation

Ю. В. Шевчук[1]
Д. П. Проценко[1]
А. И. Михеев[2]

Исследование продуктивности работы STM32F407VGT6 с .Net MF

[1]Винницкий национальный технический университет
[2]ОП «Южно-Украинская АЭС»

Проведено исследование потерь быстродействия процессора STM32F407VGT6 программной платформы STMF4DISCOVERY с использованием .Net Micro Framework для реализации инженерных и научных алгоритмических задач. Оценено влияние количества исходных данных на скорость выполнения алгоритмов.

Ключевые слова: микроконтроллер, embedded, hardware, .Net MF, bench-mark тест, TinyCLR, быстродействие.

Шевчук Юрий Владимирович — канд. тех. наук, старший преподаватель. кафедры электромеханических систем автоматизации в промышленности и на транспорте, e-mail: uriy.shevchuck@gmail.com;

Проценко Дмитрий Петрович — канд. техн. наук, доцент кафедры электромеханических систем автоматизации в промышленности и на транспорте;

Михеев Андрей Иванович — инженер релейной защиты и автоматики