УДК 004.451.44:004.382:621.391 **doi: 10.32620/aktt.2024.4.09**

О. В. ЛЮБІМОВ, І. Б. ТУРКІН

Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна

ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РЕАКТИВНОСТІ ОПЕРАЦІЙНОЇ СИСТЕМИ FREERTOS В РЕЖИМАХ ЕНЕРГОЗБЕРЕЖЕННЯ МІКРОКОНТРОЛЕРА БОРТОВОГО ОБЧИСЛЮВАЧА

Бортовий обчислювач – це спеціалізована комп'ютерна система, що інтегрована в транспортний засіб, або в загальному випадку в інший складний технічний пристрій, який потребує автономного керування і високої надійності. ***Об'єкт дослідження*** *– реактивність операційної системи FreeRTOS в режимах енергозбереження мікроконтролера бортового обчислювача.* ***Предметом дослідження*** *є методи, інструменти та технологія експериментального дослідження реактивності операційної системи FreeRTOS.* ***Мета роботи*** *– розробити технологію проведення експериментальних досліджень показників реактивності та енергоспоживання обчислювальної платформи.* ***Завдання:*** *виконати аналіз систем реального часу та типових алгоритмів планування; зробити огляд існуючих методів оцінки продуктивності та енергетичної ефективності мікроконтролерів бортових обчислювачів; виконати планування експерименту для дослідження показників реактивності та енергоспоживання обчислювальної платформи; за результатами експерименту виконати аналіз авторської платформи за вказаними критеріями.* ***Висновки.*** *В роботі надано експериментальну оцінку реактивності та енергоспоживання обчислювальної платформи «Боривітер» розробленої авторами. Платформа реалізована на мікропроцесорі архітектури ARM Cortex-M7 - ATSAMV71 та працює під операційною системою реального часу з відкритим кодом – FreeRTOS. Проведені досліди підтверджують, що розроблена мікроконтролерна платформа є ефективною та надає можливість побудови керуючих систем з прогнозованою реактивністю та з прийнятними енергетичними витратами. Платформа може бути адаптована для завдань з швидко-змінюваним обчислювальним контекстом в умовах дії зовнішніх чинників. Недоліком роботи є відсутність перевірки розробленої технології експериментальних досліджень для режимів режими Sleep, Wait та Backup мікропроцесора ATSAMV71.*

Ключові слова: *бортовий обчислювач; програмне забезпечення; Боривітер/Falco ; обчислювальна ефективність; накладні витрати; енергоефективність; ATSAMV71; Cortex-M7; ARM; FreeRTOS.*

Вступ

Кількість та потреба вбудованих обчислювачів у різних сферах автоматизації та керування автоматичними та автоматизованими системами стрімко зростає з кінця 20 століття, коли обчислювальні ресурси стали меншими за вагою, розміром та збільшили ступень інтеграції. Будь-який сучасний вбудований обчислювач є архітектурно схожим з класичним персональним комп'ютером (ПК), оскільки він містить центральний процесор, мікропроцесор або мікроконтролер (залежно від складності завдань що обчислюються), постійний та оперативний запам'ятовуючі пристрої (ПЗП та ОЗП). Великою відмінністю вбудованих обчислювачів від архітектури сучасного ПК є інтеграція ЦП, ПЗП та ОЗП з периферійними пристроями на одній друкованій платі, та суттєва перевага у енергоспоживанні (яка частково обумовлена саме єдиною побудовою складових обчислювача).

У цій статті автори розглядають вбудовані обчислювачі, які притаманні керуючим системам реального часу, що мають обмеження в енергоживленні та можуть бути застосовані як складові авіоніки або наземних систем керування.

Сучасний рух до використання вторинних джерел живлення (акумуляторів, різноманітних мобільних перетворювачів енергії, як то водневі паливні комірки) навіть у наземних нерухомих системах керування, вимагає від розробників системного та чіткого розуміння: як і якими підсистемами споживається ця обмежена енергія.

Будь-які обчислення, від корисних, ті що виконують бізнес-завдання об'єкта керування (наприклад керування двигуном), до системних, як то діагностування стану обчислювача та підключених сенсорів та виконуючих механізмів, потребують програмного забезпечення (ПЗ) що виконується на центральному процесорі або мікроконтролері.

ISSN 1727-7337 (print)
ISSN 2663-2217 (online)

Розрахунок електричної потужності, яку споживає мікроконтролер, потребує математичних моделей, що можуть адекватно описати залежність енергоспоживання цільової обчислювальної платформи від режимних параметрів. Зазвичай це математична функція, яка співвідносить енергоспоживання з характеристиками, виміряними або оціненими на певному рівні абстракції та базової архітектури обладнання.

Модель споживаної потужності може використовуватися як вхідні дані під час розробки, так і для адаптивного керування режимами роботи обчислювача під час застосування. Залежно від конкретного цільового застосування модель потужності повинна задовольняти різні вимоги і, отже, відповідним чином проєктуватися. Аналіз потужності під час проектування використовує модель потужності в автономному режимі для обмеження простору проектних параметрів цільової обчислювальної платформи, дозволяючи оцінити їх показники енергії, потужності, продуктивності та інші показники якості на ранній стадії розробки. Навпаки, моніторинг споживаної електричної потужності під час роботи надає можливість адаптивного керування режимами роботи в реальному часі через порівняння фактичних оцінок енергоспоживання та попередньо-спроектованої моделі споживання енергії.

1. Системи реального часу та типові алгоритми планування

Системи реального часу повинні гарантувати виконання завдань в визначені терміни та часові обмеження та забезпечити при цьому енергетичну ефективність обчислень. Всі можливі завдання, що виконує мікроконтролер, класифікуємо наступним чином, який визначається характером виникнення подій:

– за причиною виконання: періодичні та спорадичні завдання;

– за характером обмеження: завдання з жорстким та м'яким крайнім терміном виконання.

Розклад виконання завдань системи реального часу є коректним, якщо витримані усі граничні часові обмеження. Розкладом для періодичних завдань буде таблиця, у якій вказано, у який момент часу яку задачу треба поставити на виконання. Для спорадичних завдань визначають мінімально можливий період виникнення подій, що дозволяє їх штучно віднести до класу періодичних. Цей розклад необхідно скласти на проміжок часу, що дорівнює гіперперіоду (найменшому загальному кратному періодів) усіх задач, щоби гарантувати можливість їх виконання без порушення часових обмежень.

Класична робота Лью та Лейланда [1] запропонувала дві головні концепції планування для систем реального часу на основі пріоритетів:

– статичне призначення пріоритетів в зворотному порядку від відомих періодів завдань - Rate-monotonic scheduling (RMS);

– динамічне пріоритетне планування, коли найвищий пріоритет призначається завданню з найближчим терміном виконання – Earliest Deadline First (EDF).

Планувальники на основі RMS працюють по перериваннях від таймеру, задачі при цьому – це просто підпрограми, що викликаються в потрібні моменти з обробника переривань. Переваги даного класу алгоритмів: виняткова простота, тому результати тестувань та перевірок вельми надійні. Недоліками є:

– негнучкість, оскільки планувальник фактично не реагує на події зовнішнього світу та працює виключно по перериваннях від таймера;

– складність планування спорадичних завдань в розрахунку на мінімально можливий період виникнення зовнішніх подій;

– дуже великий розмір таблиці з розкладом при відповідних співвідношеннях між періодами завдань.

Планувальники на основі EDF найбільш пріоритетним визначають завдання з найбільш раннім абсолютним терміном виконання. Однак у практичних системах реального часу відносний кінцевий термін виконання завдання не завжди дорівнює його періоду, тому наведене вище припущення сильно обмежує корисність точних тестів на основі використання EDF. Аналіз можливості точного планування для планування EDF з довільними відносними термінами потребують обчислення вимог процесора для завдання, встановленого в кожен абсолютний термін, щоб перевірити, чи є переповнення в заданому інтервалі часу. Цей інтервал обмежений певним значенням, яке гарантує, що ми можемо знайти точку відмови, якщо набір завдань не є планованим. Значні зусилля, необхідні для виконання перевірки можливості планування відповідно до EDF у реальних системах, обмежують можливості застосування EDF в системах реального часу. Як наслідок, алгоритм EDF не використовується так широко, як алгоритми з фіксованим пріоритетом у комерційних системах реального часу [2]

Концепцію послаблених систем жорсткого реального часу (Weakly Hard Real-Time Systems, або Firm Hard Real-Time System) було започатковано ще в 2001 році [3] для більш точної характеристики часових обмежень завдань реального часу. Концепція базується на диференціації завдань в припущенні, що не обов’язково всі часові обмеження кожного за-

вдання повинні бути задовільнені. В загальному випадку пропонується використовувати (m, k)-модель, яка заснована на припущенні, що з будь-яких k послідовних екземплярів (jobs) завдання (task) принаймні m екземплярів повинні задовольняти часовим обмеженням. Іншими словами, одноразове, або навіть багаторазове порушення часових обмежень однією задачею – не завжди відмова, якщо кількість цих порушень не перевищує заздалегідь визначеного числа. Це пояснюється існуючою надлишковістю систем та застосовується для завдань м'якого реального часу (Soft Real-Time) та завдань реального часу з жорсткими або твердими обмеженнями (Hard Real-Time або Firm Real-Time).

Концепція послаблених систем отримала подальший розвиток в багатьох роботах, наприклад в статтях [4 - 6]. В її основі наступна модель завдання у вигляді кортежу:

$$\tau_i = \langle C_i, D_i, T_i, m_i, K_i \rangle, \quad (1)$$

де C_i - максимальний час, потрібний завданню;

T_i - мінімальний інтервал часу прибуття завдання.

m_i – кількість разів виклику завдання коли дозволено порушувати кінцевий термін виконання – дедлайн.

K_i – Кількість завдань, яку треба спланувати.

Якщо це періодичне завдання, то T_i – це її період, якщо спорадична – то це мінімальний період виникнення переривання;

D_i – часове обмеження на виконання роботи завданням ($D_i \leq T_i$):

$$\langle m_i, K_i \rangle, \; m_i < K_i; \quad (2)$$

– послаблені обмеження завдань реального часу: на послідовності з K_i робіт цього завдання дозволено порушити дедлайни не більше m_i разів. Якщо це завдання відноситься до класу hard real-time, то:

$$m_i = 0, \; K_i = 1. \quad (3)$$

Алгоритм роботи планувальника в послабленій системі жорсткого реального часу містить наступні кроки.

1. Якщо всі заплановані часові обмеження всіх робіт для всіх завдань можуть бути дотримані за допомогою EDF, використати EDF та завершити роботу. Якщо ні – то на п. 2.

2. Відсортувати роботи всіх задач за критерієм кількості часових обмежень робіт, що ще дозволено порушити на інтервал часу планування. В клас «0» попадуть всі завдання, які не дозволяють жодного пропуску. В клас «1» – ті, що можуть 1 раз порушити часове обмеження. Приклад: якщо (m_i;K_i) для завдання дорівнює (2, 4), то дозволено порушити часове обмеження 2-х робіт з 4-х послідовних робіт, відповідно завдання може попасти в класи {0, 1, 2} в залежності від кількості вже пропущених робіт.

3. Використати EDF спочатку для класу 0, потім для класу 1 тощо.

Досить багато сучасних публікацій присвячено технологіям оцінки показників продуктивності мікроконтролерів та вимірюванні витрат часу на роботу типових алгоритмів як на широко використовуваних платформах масового виробництва, так і авторських розробках.

В роботі [7] виконано порівняння показників продуктивності платформ Raspberry Pi4, BeagleBone AI та TWR-K70F120M, а також часу виконання алгоритмів на цих платформах за умови багатопоточної реалізації деякої множини виконуваних завдань. Надано основні показники, що важливі для систем реального часу, а саме час виконання завдання, час виконання в найгіршому випадку, час очікування та затримки відповіді. Для досягнення мети розроблено багатопотокову тестову програму зі спеціальними обчислювальними інтенсивними операціями сортування, роботи з матрицями та з використанням криптографічної полегшеної крипто-бібліотеки wolfCrypt, яка написана на ANSI C і призначена для вбудованих систем, систем реального часу та середовищ з обмеженими ресурсами.

У статті [8] надано результати порівняння обчислювальної продуктивності відкритого STEM-подібного апаратного проекту Pi Pico від Raspberry на процесорі RP2040, та авторського рішення обчислювальної платформи «Falco SBC/CDHM» на основі мікроконтролеру ATSAMV71 (Microchip) з покращеною продуктивністю. Перевірку можливості використання мікросервісної архітектури та ефективності запропонованої нової платформи було проведено експериментально із вимірянням процесорного часу, необхідного для виконання трьох типових алгоритмів різної алгоритмічної складності та різної розмірності вихідних даних.

Спільним недоліком розглянутих робіт є відсутність оцінки енергетичної ефективності обчислювальних платформ, адже невід'ємною особливістю систем реального часу є надлишковість, в тому числі часового ресурсу, щоби гарантувати виконання часових обмежень за будь яких умов. В реальних проєктах така надлишковість існує як на апаратному, так і на програмно-алгоритмічному рівнях. Важливою є відповідь на питання, наскільки енергетично ефективно поводить себе платформа в той проміжок часу, коли всі поточні завдання вже виконано та немає потреби щось робити, аж доки не поступить нове переривання або зовнішнє від підключених пристроїв,

ISSN 1727-7337 (print)
ISSN 2663-2217 (online)

або внутрішнє від системного таймера. Ідеальною була б ситуація, коли платформа в цей час взагалі нічого не споживала, але це неможливо.

Зворотною стороною цієї проблеми є визначення показників погіршення реактивності обчислювальної системи, адже перед тим, як почати робити щось корисне після зовнішньої події, процесор та інші складові обчислювальної платформи повинні спочатку повністю відновити режим нормальної роботи. Подібна проблема для портативних комп'ютерів була розглянута в роботі [9], але в цьому випадку замість об'єктивного критерія реактивності системи було використано 2 критерії: середній сумарний штраф за порушення визначених строків на плановому інтервалі та штраф за скорочення тривалості роботи процесора за часом автономної роботи багатозадачної системи мобільного комп'ютерного пристрою.

2. Енергетична ефективність обчислень в системах реального часу

Енергоспоживання обчислювальної платформи можна структурувати, розділивши його на дві частини:

– статична частина включає витрати на підтримку контролера в робочому стані. Вона має слабку кореляцію з природою застосунків, що виконуються, а її контроль і управління практично неможливі;

– динамічна частина пов'язана з виконанням програм та суттєво залежить від характеристик завантаження, ступеня утилізації ресурсів та політики енергоспоживання, тому вона може сильно змінюватись. Саме тут виникає простір для оптимізації, оскільки саме ПЗ суттєво впливає на загальне енергоспоживання системи. Динамічна частина енергоспоживання, у свою чергу, складається зі споживання енергії процесором, пам'яттю та зовнішніми пристроями, а також витратами енергії на їх взаємодію.

2.1. Методи керування енергоспоживанням

В огляді [10] надано наступну класифікацію методів забезпечення енергетичної ефективності мікроконтролерів вбудованих систем:

– методи динамічного масштабування напруги та частоти (DVFS – dynamic voltage and frequency scaling) і планування з урахуванням потужності;

– використання режимів низького енергоспоживання, що називаються керуванням режимом живлення (Power Mode Management - PMM), або динамічним керуванням живленням (Dynamic Power Management - DPM);

– мікроархітектурні техніки для збереження енергії в окремих компонентах, таких як пам'ять де обчислювальний контекст зберігається в пам'яті під час повного або часткового відключення процесора;

– використання нетрадиційних обчислювачів, таких як DSP або GPU FPGA. Цей метод підходить для складних за обчислювальною потужністю завдань, де традиційні процесори загального призначення мають гіршу продуктивність (мВт/MIPS).

Стаття [11] досліджує можливість заощадження енергії в безпровідних сенсорних мережах за рахунок використання DVFS низькоенергетичних мікроконтролерів. Кількісною метрикою для оцінки енергоефективності є нормалізована потужність як відношення споживаної електричної потужності до продуктивності (мВт/ MIPS). Нормалізована потужність дозволяє точніше охарактеризувати енергоефективність мікроконтролера, оскільки враховує не тільки споживану електричну потужність, але й продуктивність, виражену в MIPS (Million Instructions Per Second - мільйон інструкцій в секунду). Чим нижче значення нормалізованої потужності, тим менше енергії буде споживати мікроконтролер.

Загалом, стаття дає хороший базис для подальших досліджень DVFS в безпровідних сенсорних мережах, але деякі аспекти потребують більш ретельного опрацювання. До недоліків статті відноситься складність узагальнити отримані результати на інші платформи та архітектури, відсутність оцінки накладних витрат на переходи між режимами DVFS, обмежена кількість вимірювань для різних комбінацій напруги/частоти.

Публікація [12] комплексно досліджує взаємозв'язки між трьома складовими:

– обмеженнями реального часу;

– обмеженнями на енергетичне споживання обчислювача;

– програмними методами забезпечення відмовостійкості.

Мова йде про взаємозалежності ймовірності постійних відмов від частоти та напруги живлення, ймовірності тимчасових відмов від часу виконання завдань, а також про залежність споживаної електричної потужності від обраних політик забезпечення відмовостійкості та режимних параметрів обчислювача. В статті запропоновано спільну модель для аналізу планування та відмов, щоб виділити формальні взаємодії між механізмами відмовостійкості та часовими властивостями. Стаття пропонує кілька ключових напрямів для майбутніх досліджень у галузі систем реального часу, стійких до збоїв:

– розробка алгоритмів планування, які враховують ймовірність збоїв з метою мінімізувати активний час завдань та їх сумарну ймовірність збоїв, зберігаючи при цьому планованість;

– аналіз впливу різних стратегій забезпечення відмовостійкості (повторне виконання, контрольні

точки, N-кратна надлишковість (NMR - "N-Modular Redundancy") на планування. Зокрема, інтеграція різних підходів та їх оптимізація для покращення планованості та відповідності вимогам щодо ймовірності відмов;

– застосування концепції змішаної критичності (Mixed-Criticality) з метою зробити системи сумісними з галузевими стандартами та кількісно оцінити ймовірність переходу в режим високої критичності;

– аналіз компромісів між управлінням енергоспоживанням (DVFS), тепловими ефектами, стійкістю до різних типів збоїв та вимогами реального часу;

– покращення надійності системного програмного забезпечення, такого як планувальник та механізми виявлення збоїв;

– використання ймовірнісної інформації про час виконання для обчислення більш точної оцінки ймовірності збоїв;

– розгляд інших моделей збоїв, таких як (k, n), наближені обчислення та зловмисні збої.

Стаття [13] аналізує надійність вбудованих супутникових систем реального часу, що працюють у жорсткому космічному середовищі. Розглянуто два типи помилок, які характерні для систем, що працюють в жорстких температурних та/або радіаційних умовах. Це:

– "Soft-error" або "м'яка помилка" - одноразовий збій (Single-Event Upsets, SEU), тимчасове спотворення бітового значення в пам'яті або регістрі процесора, спричинене зовнішніми факторами, що не призводить до постійних пошкоджень апаратури;

– "Hard-error" або "жорстка помилка" - постійне пошкодження апаратного компонента, спричинене зносом або деградацією матеріалів унаслідок тривалої експлуатації або, наприклад, радіаційного впливу у космічному використанні. Такі помилки класифікуються як одноразові «залипання» (Single Event LatchUps - SEL).

Обмеження систем реального часу враховуються шляхом використання моделі періодичних завдань для оцінки навантаження системи при надлишковому резервному виконанні завдань з метою виявлення м’яких помилок та їх подальшого усунення відповідно до вимог стандартів функціональної безпеки, таких як DO-178B, IEC-61508 та ISO-26262. Підвищення стійкості як до одноразових м'яких помилок, так і до постійних жорстких відмов досягається через розв’язок задачі оптимізації.

Стаття [14] присвячена аналізу та розробці методів для зменшення енергоспоживання в ультранизькопотужних вбудованих системах за допомогою динамічного масштабування напруги та частоти (DVFS). Автори виконали аналіз енергоспоживання під час виконання обчислювально-інтенсивних операцій, таких як швидке перетворення Фур'є (FFT), циклічний надлишковий контроль (CRC32), обчислення хеш-функцій MD5 та SHA256. За результатами експериментального тестування на мікроконтролері ARM Cortex-M0+ підтверджено, що застосуванням DVFS можна зберегти від 27,74 % до 47,74 % електричної енергії. До недоліків віднесемо відсутність порівняння з іншими існуючими методами енергозбереження та обмежений набір тестових операцій (FFT, CRC, хеш-функції), можливо не репрезентативний для інших типів навантажень. Нерозглянутим залишились проблеми перехідних процесів при динамічній зміні напруги та часових затримок, пов'язаних з цим.

Оцінка ефективності системного планування та енергозбереження у вбудованих системах реального часу з низькими обчислювальними ресурсами є проблемою, що розглядається в статті [15]. В операційних системах реального часу (ОСРЧ) характеристики реалізованої політики планування відіграють важливу роль як у плануванні, так і в споживанні енергії. В ідеалі політика планування повинна гарантувати дотримання розкладу завдань та низькі витрати енергії під час виконання, дозволяючи краще використовувати доступний вільний час у розкладі з метою енергозбереження. Запропонована в статті політика планування базується на плануванні з фіксованим пріоритетом (RMS), що забезпечує низькі накладні витрати і простоту реалізації. За цією схемою простого вектора пріоритетів достатньо, щоб вказати поточне завдання, готове до виконання. Однак результати планування зазвичай нижчі, ніж ті, що можна досягти динамічним плануванням пріоритетів, відповідно до якого пріоритети завдань призначаються під час виконання.

В мікроконтролері з операційною системою FreeRTOS управління обмеженим енергетичним бюджетом здійснюється за допомогою апаратних та програмних засобів [16]. Коли завдання очікує переривання або закінчення інтервалу часу, воно блокується. Якщо всі завдання переходять у заблокований стан, FreeRTOS виконуватиме завдання idle task з найнижчим пріоритетом. Тому, коли процесор простоює, завдання idle task може перевести його в режим енергозбереження. Цей механізм є корисним у деяких сценаріях, але якщо тактова частота занадто висока, процесор витрачатиме енергію та час на вхід у режим очікування та вихід із нього. Отже, економія електроенергії за допомогою цього механізму не принесе користі. Тому, щоб покращити відповідний механізм енергозбереження, була введена техніка холостого ходу без системного тику – Tickless idle [17]. Техніка використовує механізм відстеження часу, щоб вимкнути джерело періодичних тактів на певний

ISSN 1727-7337 (print)
ISSN 2663-2217 (online)

період часу для переводу процесора у режим глибокого сну, доки не відбудеться зовнішнє переривання або переривання ядра з вищим пріоритетом.

В роботі [18] запропоновано рішення для оптимізації енергетичної ефективності роботи планувальника операційної системи мікроконтролера на основі LM3S3748. Пропонується використовувати системний потік «idle», який після завершення власної роботи переводить мікроконтролер в режими Sleep, або Deep Sleep. В статті надано, як кількісні експериментальні результати вимірювань споживаної енергії мікроконтролером, так і оцінки зниження реактивності системи, через додаткові витрати часу на цикл призупинення-відновлення роботи мікроконтролера. Недоліком статті є відсутність узагальнення результатів у вигляді загальної технології оптимізації енергетичної ефективності роботи мікроконтролера.

Стаття [19] надає огляд основних алгоритмів енергозбереження DVFS (Dynamic Voltage and Frequency Scaling) - динамічна зміна напруги і частоти процесора та DPM (Dynamic Power Management) - динамічне управління енергоспоживанням, в основі якого – переведення процесора в режими низького енергоспоживання. В цьому огляді висвітлюються основні проблеми, пов'язані з реактивністю системи реального часу при застосуванні методів енергозбереження та підкреслюється складність балансування між енергоефективністю та підтримкою необхідного рівня реактивності в системах реального часу.

Наведені в статті часові показники реактивності поділимо на 2 класи.

До першого класу відносяться затримки, які визначаються переважно апаратною складовою:

– затримки пробудження (wake-up delays) характеризують витрати часу, потрібні для повного відновлення роботи процесора з режиму сну. Виміряти їх можна через визначення інтервалу часу від моменту запиту на переривання до першої корисної команди в обробнику переривання;

– час окупності (Break-even time) – інтегральна характеристика. Це мінімальний час, який процесор повинен провести в режимі низького енергоспоживання, щоб компенсувати енергетичні та часові витрати на перехід в цей режим і назад. Час окупності – це сума затримки пробудження (wake-up delays) та затримки переходу (transition delays), яка характеризує витрати часу, необхідні для переходу процесора з активного стану в режим сну.

До другого - затримки, які визначаються тільки алгоритмами програмного забезпечення:

– затримки прокрастинації (procrastination delays), коли деякі алгоритми навмисно відкладають виконання завдань, щоб збільшити тривалість періоду простою і ефективніше використовувати режими низького енергоспоживання. Ці затримки ретельно розраховуються, щоб не порушити часові обмеження завдань;

– затримки планування (scheduling delays) – час, необхідний для прийняття рішень про зміну режиму енергоспоживання та перепланування завдань.

– затримки, пов'язані з обчисленням оптимальних моментів переходу в режим сну та пробудження. Деякі алгоритми виконують складні обчислення для визначення цих моментів, що може вносити додаткові затримки.

– затримки, викликані раннім завершенням завдань (early completion delays), які створюють додатковий простір для енергозбереження, але потребують динамічного перепланування.

Врахування затримок першого класу є критичним для ефективного застосування режимів енергозбереження в системах реального часу, оскільки вони безпосередньо впливають на здатність системи дотримуватися часових обмежень при одночасному зниженні енергоспоживання, тому тільки ці затримки є предметом подальшого розгляду.

2.2. Огляд існуючих методів оцінки продуктивності та енергоефективності

Будь-яка задача оптимізації показників системи починається з завдання вимірювання параметрів системи та їх аналізу. Якщо для сучасних побутових ПК, існує велика кількість засобів вимірювання та оцінки як продуктивності, так і енергоефективності, то для вбудованих систем кількість таких засобів є обмеженою та мало-відомою для розробників.

Автори роботи [20] досліджують обчислювальну продуктивність та енергетичну ефективність різних мікропроцесорів, які використовуються як бортові комп'ютери в наносупутниках, в типових завданнях визначення орієнтації наносупутників та керування нею. Головною мотивацією авторів для розробки нового спеціалізованого бенчмарку (ПЗ для вимірювання та порівняння) замість використання відомих є наступні недоліки існуючих бенчмарків.

1. Бенчмарки EEMBC [21], ParMiBench [22], BEEBS[23] чи EmsBench [24] не містять достатньої кількості операцій з матрицями, кватерніонами та обчислень, типових для алгоритмів керування орієнтацією супутників.

2. Більшість існуючих бенчмарків орієнтовані на оцінку продуктивності, в той час як для супутникових систем критично важливим є енергетична ефективність обчислень через жорсткі обмеження щодо живлення.

3. Вимоги до великого обсягу пам'яті або використання зовнішніх файлів. Деякі бенчмарки, вимагають доступу до файлової системи, що може бути

проблематичним для вбудованих систем з обмеженими ресурсами пам'яті.

4. Відсутність відкритого коду або вимога платної підписки. Наприклад, бенчмарки EEMBC вимагають платної підписки для отримання доступу до тестових навантажень.

Автори протестували розроблений ними бенчмарк на трьох мікроконтролерах, які часто використовуються в наносупутниках: Arduino Uno, Texas Instruments MSP430 та STM32 Nucleo. В якості ключової метрики для порівняння різних платформ була використана споживана електрична потужність. В якості тестового набора завдань автори використали типові операції та алгоритми визначення орієнтації та керування наносупутниками, як то операції з матрицями довільної розмірності, обчислення кватерніонів.

В роботі [25] запропоновано корисні техніки вимірювання витрат часу на такі службові операції, як обробка переривань та затримки перемикання потоків для віртуальних машин, які в цій статті адаптовано для дослідження мікроконтролера.

3. Постановка задачі

Мотивація. Дослідження компромісу між реактивністю системи та використанням режимів енергозбереження обчислювальної платформи є важливим у багатьох ситуаціях, особливо в системах реального часу, що мають автономне джерело живлення. В загальному випадку це і вбудовані системи в промислових застосунках, і пристрої Інтернету речей (IoT), імплантовані медичні пристрої, системи оповіщення про надзвичайні ситуації тощо. В процесі роботи обчислювальної платформи цей компроміс досягається через програмно-реалізовані алгоритми, що дозволяють адаптивно керувати режимами роботи обчислювача в залежності від поточних умов та вимог до реактивності та енергоспоживання системи. Для вирішення питання про доцільність витрат на розробку нових алгоритмів та програмного забезпечення необхідне експериментальне дослідження з метою побудови математичної моделі, що комплексно характеризує реактивність та енергоспоживання обчислювача.

Ціль роботи. Розробити технологію експериментальних досліджень показників реактивності та енергоспоживання обчислювальної платформи.

4. Планування експерименту

Експериментальне дослідження провадиться на авторській обчислювальній платформі «Боривітер» [26] (рис. 1), що детально описана в роботі [8]. Платформа побудована на основі 32-розрядного мікроконтролера ATSAMV71 (рис. 2), який належить до лінійки мікроконтролерів ARM Cortex-M7 [27] та працює під керуванням операційної системи реального часу (ОСРЧ) з відкритим кодом - FreeRTOS.

Рис. 1. Авторський бортовий обчислювач подвійного призначення – «Боривітер» 0.1

Рис. 2. Мікроконтролер Microchip ATSAMV71

Пристрої на основі мікросхеми ATSAMV71 мають три режими низького енергозбереження, які можна вибрати програмно: режими Sleep, Wait and Backup [27]. У режимі Sleep процесор зупиняється, а всі інші функції можуть працювати. У режимі Wait всі годинники та функції зупиняються, але деякі периферійні пристрої можна налаштувати для пробудження системи на основі попередньо визначених умов. Ця функція, яка називається SleepWalking, виконує часткове асинхронне пробудження, таким чином дозволяючи процесору виходити з режиму сну лише за потреби. У режимі Backup працюють 32-bit low-power Real-time Timer (RTT), Real Time Clock (RTC) і логіка пробудження. Крім того, у цьому режимі пристрій може відповідати найсуворішим вимогам Key-Off, коли система або пристрій знаходяться у вимкненому стані, але мікропроцесор все ще продовжує працювати з певним рівнем активності. зберігаючи 1 Кбайт SRAM. Щоб оптимізувати енергоспоживання, тактова система була розроблена для підтримки різних тактових частот для вибраних

ISSN 1727-7337 (print)
ISSN 2663-2217 (online)

периферійних пристроїв. Крім того, тактову частоту процесора та шини можна змінювати, не впливаючи на роботу з USB, U(S)ART, AFE та лічильник таймера.

4.1. Обмеження та припущення

1. Напруга живлення є номінальною та дорівнює 3,3 В, оскільки від неї залежить надійність роботи обчислювача та його стійкість до одноразових збоїв. Залежність енергоспоживання від напруги не є предметом дослідження.

2. Пріоритети виконуваних завдань призначаються відповідно до класичної теорії Лью та Лейланда – RMS (Rate-monotonic scheduling). В цьому випадку кожне завдання τ_i є періодичним та характеризується двома числами:

$$\tau_i = \langle C_i, T_i \rangle, \tag{4}$$

де C_i – максимальний час, потрібний задачі;

T_i – її мінімальний період повторення.

3. Множина з N виконуваних завдань

$$T = \left\{ \cup_{i=1}^{N} \tau_i \right\} \tag{5}$$

завжди сформована таким чином, щоби вони відповідали достатній умові спланованості завдань систем жорсткого реального часу, сформульованої Лью та Лейландом [1]:

$$\sum_{i=1}^{N} \frac{T_i}{C_i} \leq N\left(2^{1/N} - 1\right). \tag{6}$$

4. Обмеження на величину системного тику, який визначає частоту переривань від системного таймера, отримано з документації FreeRTOS з урахуванням обмежень середовища розробки MPLAB X IDE:

$$T \in \{1\ \text{ms},\quad 4\ \text{ms}\}. \tag{7}$$

5. Обмеження на величину тактової частоти визначено на основі технічної документації на мікроконтролер ATSAMV71:

$$f \in \{30\ \text{MHz},\ 100\ \text{MHz},\ 300\ \text{MHz}\}. \tag{8}$$

4.2. Структурно-параметрична ідентифікація необхідна для побудови наступних моделей

1. **Залежність енергоспоживання** $N(f, m)$ бортового обчислювача в стаціонарному режимі від f – тактової частоти та m – режиму роботи. Множина можливих режимів поділяється на дві підмножини - режими активної роботи та режими енергозбереження:

$$m \in \text{ACTIVE} \cup \text{LPM}. \tag{9}$$

Режими активної роботи ACTIVE в свою чергу поділяються на обчислення з плаваючою точкою (Floating Point - FP) та цілочисельні (Integer - Int):

$$\text{ACTIVE} = \{\text{FP}, \text{Int}\}. \tag{10}$$

Режими збереження енергії LPM (Low Power Mode) відповідно до документації на мікроконтролер ATSAMV71 (Microchip) є такими:

$$\text{LPM} = \{\text{Sleep}, \text{Wait}, \text{Backup}\}. \tag{11}$$

2. **Витрати процесорного часу на обробку переривань** від системного таймера в залежності від тактової частоти обчислювальної платформи планується отримати у вигляді залежності: $\text{Lat}_{\text{Timer\_IRQ}}(f)$

3. **Затримка виконання першої інструкції процедури обробки переривань** – це час від моменту виникнення переривання до часу виконання першої команди обробника переривання:

$$\text{Lat}_{\text{IRQ}}(f, m). \tag{12}$$

Таким чином, незалежними факторами експерименту є тактова частота та режим роботи, а відгуками експерименту є енергоспоживання, витрати процесорного часу на обробку переривань від системного таймера та затримка виконання першої інструкції процедури обробки переривань. Заплановано повнофакторний експеримент, в якому фактори приймають наступні значення:

$$\begin{gathered} f \in \{30\ \text{MHz},\ 100\ \text{MHz},\ 300\ \text{MH}\}, \\ m \in \{\text{FP}, \text{Int}, \text{Sleep}, \text{Wait}, \text{Backup}\}. \end{gathered} \tag{13}$$

4.3. Техніка вимірювання

Вимірювальний стенд (рис. 3) для проведення досліджень зібрано за схемою (рис. 4).

1. **Енергоспоживання.** Для вимірювання споживаної електричної потужності використане високоточне стендове джерело живлення постійного струму Keithley 2281S, що гарантує точність вимірювань (похибка вимірювання часового інтервалу – не гірше 15 мс, похибка вимірювання електричної потужності не перевищує 0.0001 Вт).

2. **Порівняння витрат процесорного часу на обробку переривань в bare metal рішенні та під**

управлінням FreeRTOS. В обох випадках використана модель, коли одна задача в безкінечному циклі тільки навантажує процесор та запам'ятовує для наступного аналізу фактичну тривалість кожного циклу як різницю часу завершення роботи алгоритму та часу початку його роботи. Друга задача, виконання якої ініціюється перериванням, змінює стан зовнішнього порту вводу/виводу – Порт 1 на протилежне (від попереднього, тобто «0» на «1» або «1» на «0»). Зовнішнє переривання генерується генератором прямокутних імпульсів (меандр) та подається на порт вводу/виводу 3, який в свою чергу генерує системне процесорне переривання класу EXTI (EXTernal Interrupt) (див. рис. 4).

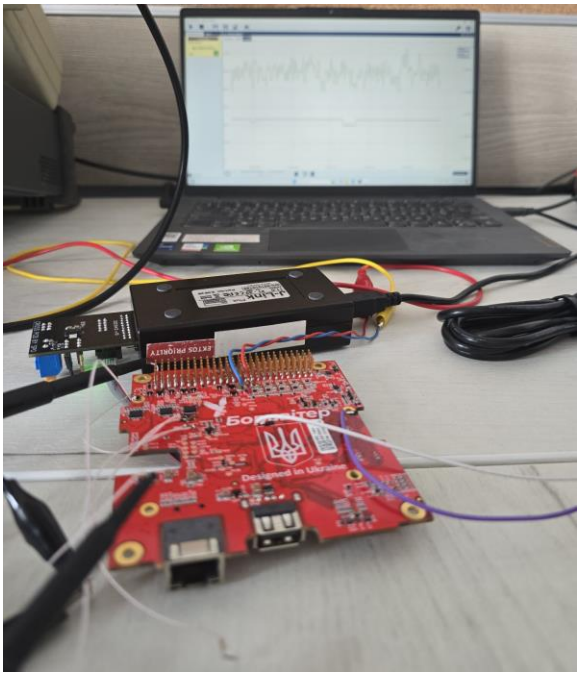
3. **Затримка виконання першої інструкції процедури обробки переривань.** Вимірювання проводиться через реєстрацію інтервалу часу за допомогою цифрового осцилографу LeCroy WavePro 7200A між двома подіями: генерованого зовні сигналу переривання (за допомогою генератора прямокутних сигналів), та першою командою обробника переривання, якою є зміна стану попередньо визначеного порту – Порт 1, на протилежний. Оскільки команда зміни порту є атомарною, тобто виконується за один обчислювальний такт в архітектурі RISC, то час зміни стану порту можна вважати незначним.

Витрати часу на перемикання контексту. Вимірювання проводиться з ПЗ під керуванням ОСРЧ FreeRTOS, яке має дві задачі з однаковим пріоритетом та обробник переривання. Перша – та сама що виконує обчислення в п.2. Друга – задача яка змінює стан порту вводу/виводу (Порт 2) після отримання системного сповіщення (xEvent) яке генерується в системній функції обробки переривання. Зовнішнє переривання синхронізує загальну циклограму вимірювання.

5. Результати експерименту

1. **Енергоспоживання** платформи лінійно залежить від частоти при фіксованій напрузі живлення (рис. 5). Використання режимів енергозбереження процесора дозволяє досягти навіть 60 % економії споживаної енергії. Ця перевага найбільше помітна при роботі процесора на максимальній частоті.

2. Порівняння витрат процесорного часу на обробку переривань. Завдяки специфіці обчислювальної архітектурі ARM Cortex-M, яка має два незалежних лічильника інструкцій: повний Program Counters (PC), та «емульований» - Link Register (LR) – обчислювач «Боривітер» демонструє достатньо високі показники реактивності. ОСРЧ FreeRTOS ефективно підтримує архітектуру ARM Cortex-M, оскільки не розрізняє функцій обробки переривань та звичайних функцій, тобто не підтримує класичних засобів обробки переривань як-то Linux на архітектурі x86 або amd64. Поєднуючи простоту реалізації обробки переривань ОСРЧ FreeRTOS та PC+LR засобів мікроконтролерної архітектури ARM Cortex M7 отримано наступні результати (рис. 6). Результати підтверджують, що використання ОСРЧ не погіршує реактивність в порівнянні з програмним рішенням,



а)

б)

Рис. 3. Тестовий стенд для дослідження (а – підключення БО «Боривітер» до щупів осцилографу, генератору та блоку живлення, б – вимірювальна техніка для проведення досліджень)

ISSN 1727-7337 (print)
ISSN 2663-2217 (online)

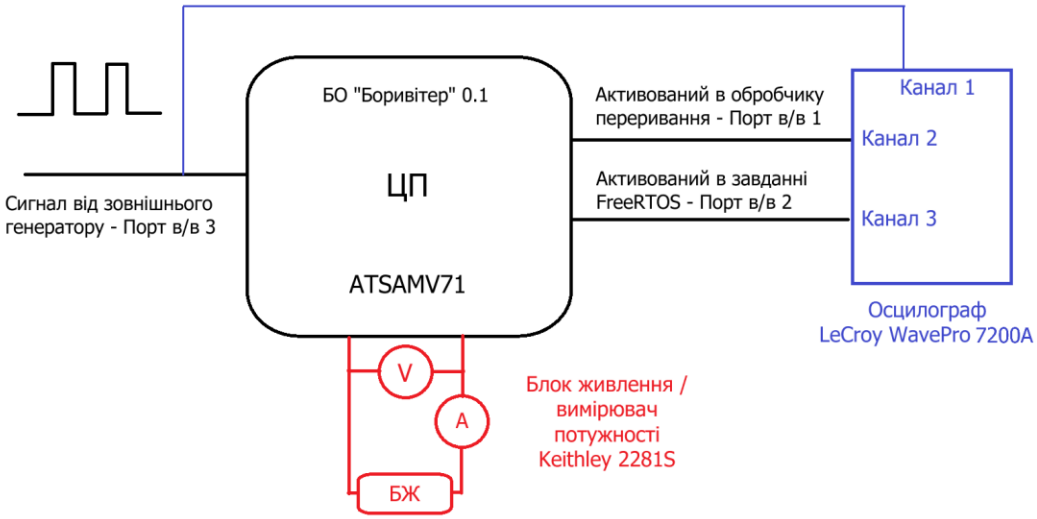


Рис. 4. Схема вимірювального стенду (БО «Боривітер», генератор прямокутних сигналів GW Instek GFG-8219A, Блок живлення/вимірювач потужності – Keithley 2281S-20-6, Багатоканальний запам'ятовуючий осцилограф – LeCroy WavePro 7200A)

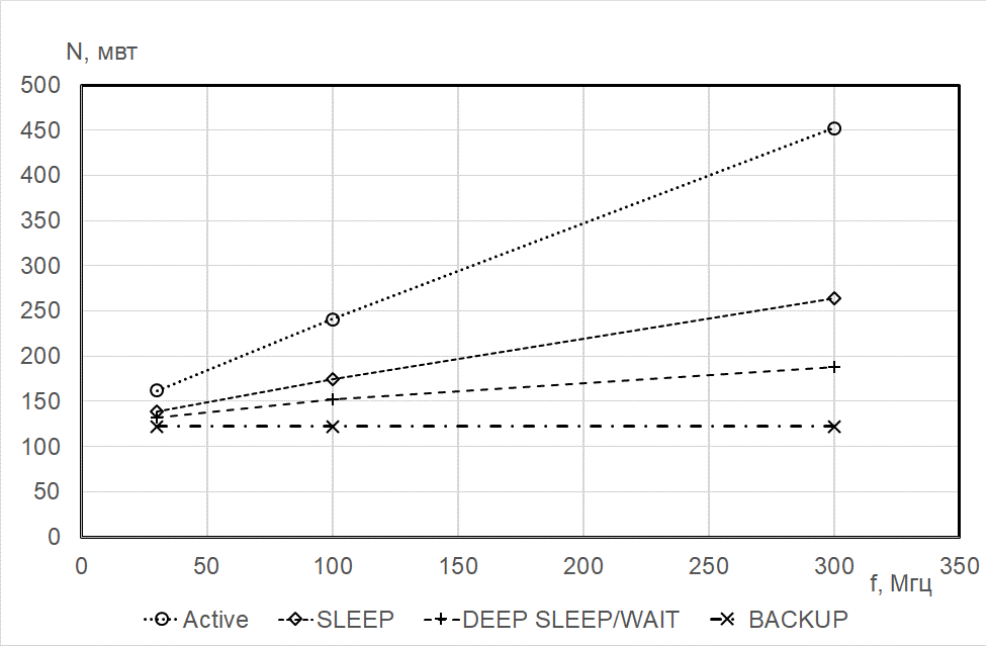
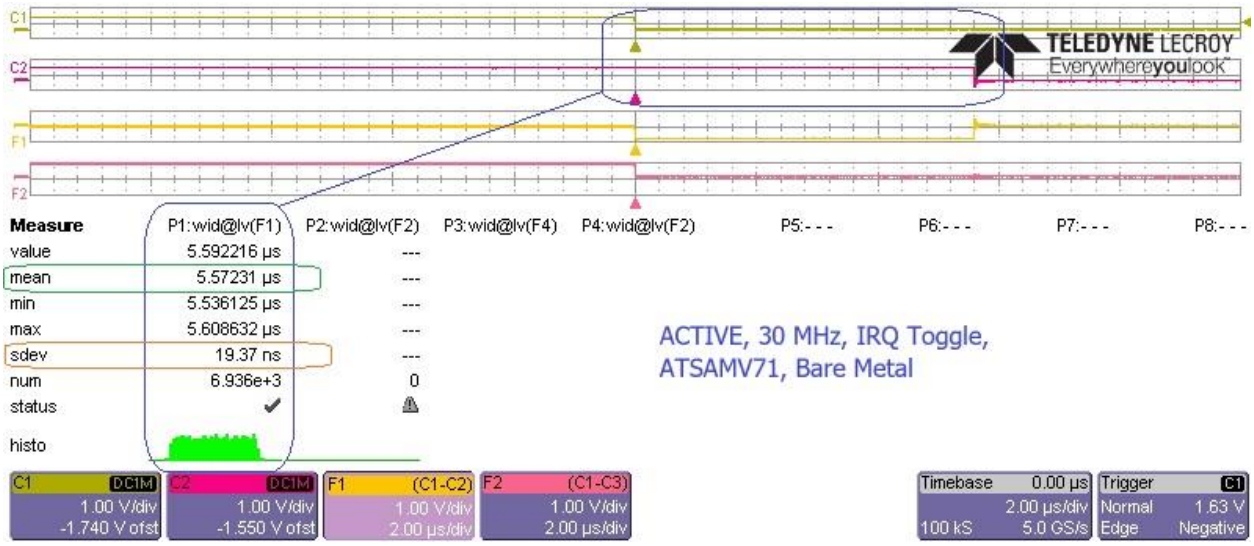


Рис. 5. Залежність енергоспоживання платформи від частоти при різних режимах роботи обчислювача та фіксованій напрузі живлення

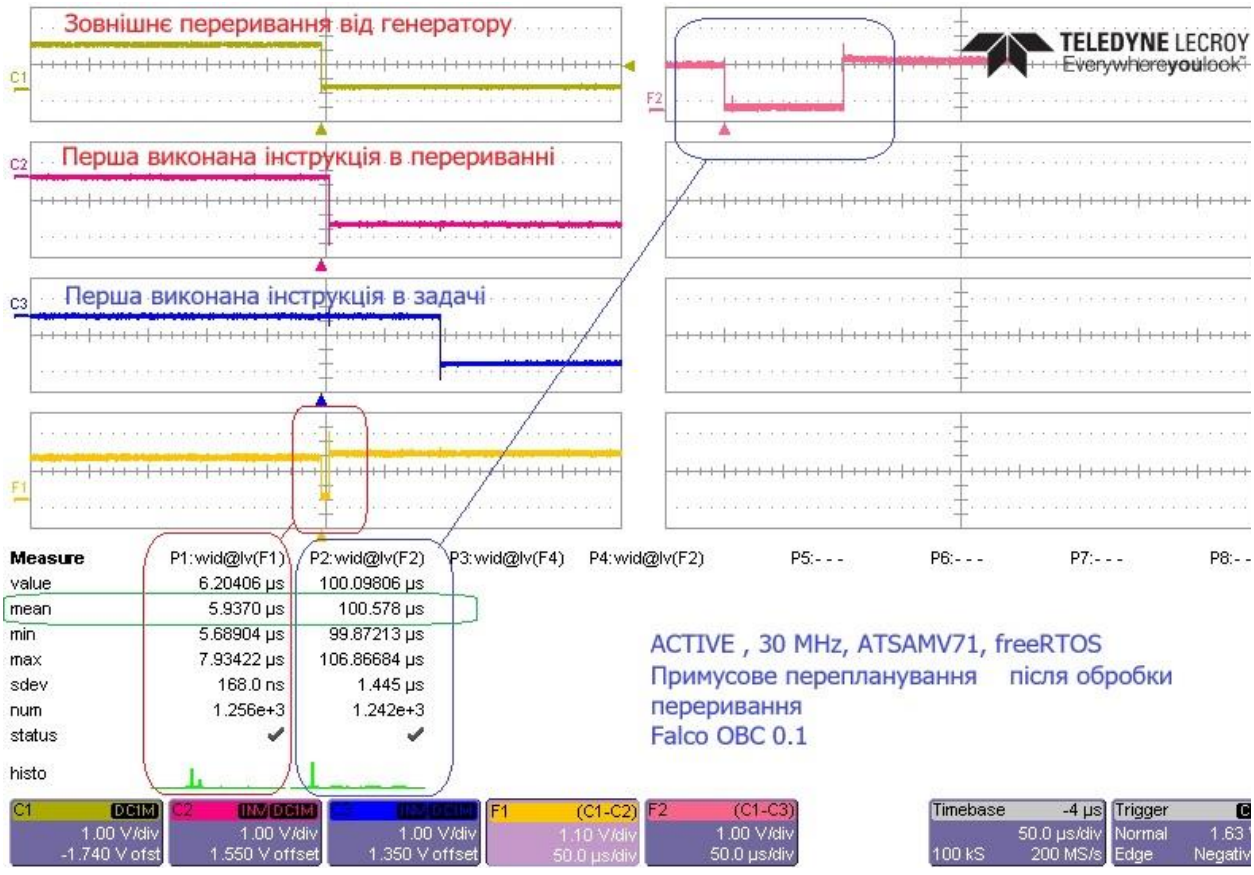
заснованим на Bare Metal. Витрати процесорного часу відповідно складають 5,57 мкс та 5,93 мкс при тактовій частоті 30 МГц.

3.Оцінку **затримки виконання першої інструкції процедури обробки переривань** наведено на рисунку 7. Якщо вторинна обробка переривання здійснюється в звичайному (непривілейованому) режимі, то ОСРЧ використовує механізм сповіщення через подію. Останнє означає, що гарантовано це відбудеться тільки після чергового системного тику, довжина якого задається в налаштуваннях ОСРЧ.

4. **Витрати часу на перемикання контексту** характеризує рисунок 8. Як видно з рисунку, система реагує на зовнішнє переривання приблизно в 15 разів швидке, ніж реалізує механізм кооперативної багатозадачності. В ОСРЧ FreeRTOS системна функція taskYIELD() негайно викликає перепланування, тобто змушує планувальник перевірити, чи існує інша задача, готова до виконання. Якщо така задача є і вона має вищий або рівний пріоритет у порівнянні з поточною, буде виконано переключення контексту на цю задачу.



а)



б)

Рис. 6. Результати вимірювання реактивності системи як інтервалу часу між надходженням переривання та виконананням першої команди в обробнику переривання:
а) – час на вхід в переривання (ПЗ без ОСРЧ);
б) – час на вхід в обробник переривання (ПЗ під керуванням ОСРЧ)

ISSN 1727-7337 (print)
ISSN 2663-2217 (online)

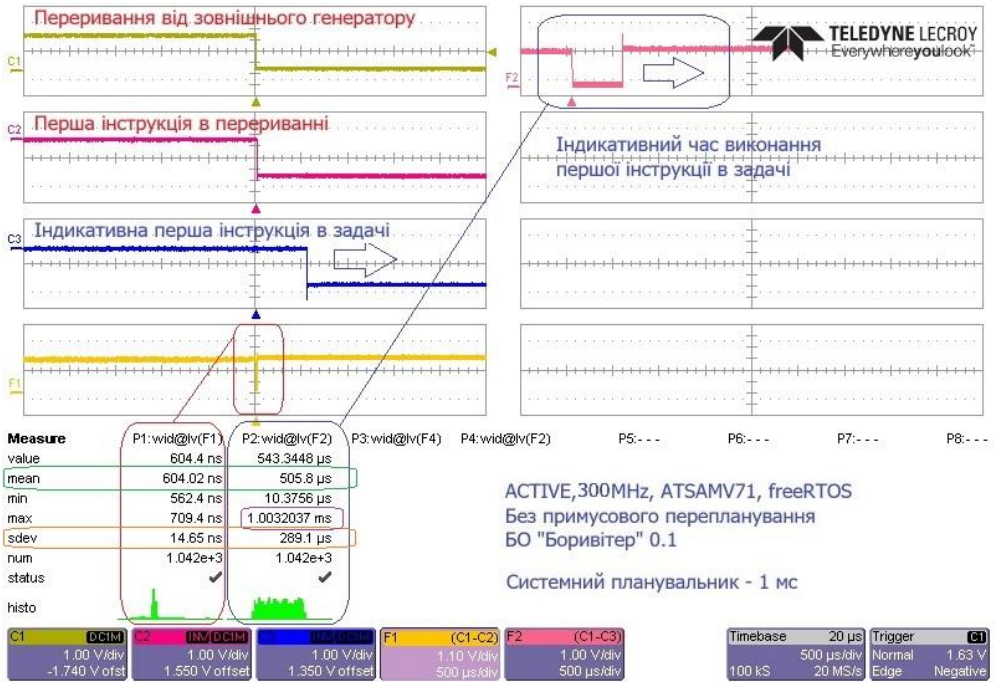


Рис. 7. Результати вимірювань часу входу в процедуру обробки переривання та часу реакції на переривання во вторинній задачі, що виконує основну обробку. Режим мікроконтролеру ACTIVE, тактова частота 300 МГц, системний планувальник – 1 мс. Без примусового перепланування

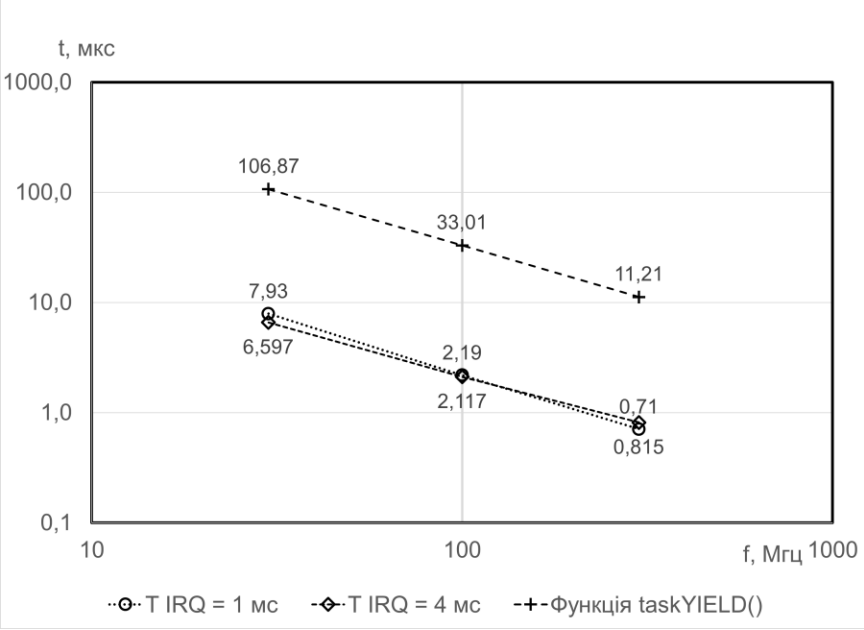


Рис. 8. Порівняння часу перемикання контексту внаслідок виконання функції taskYIELD() та часу входу в обробник переривання

Висновки

В роботі надано експериментальну оцінку реактивності та енергоспоживання обчислювальної платформи «Боривітер», яка реалізована на мікропроцесорі ATSAMV71 та працює під ОСРЧ з відкритим кодом – FreeRTOS. Проведені досліди з виміру часу перемикання контексту та можливості примусового керування перемикання контексту надають можливості розробникам БО «Боривітер» аналізу відповідності

системи вимогам реального часу. Гнучкість мікроконтролера з обрання тактової частоти, дозволяє витримати баланс між потрібною реактивністю та енергоспоживанням обчислювача.

Порівняння накладних витрат на обробку переривання з застосуванням ПЗ без ОСРЧ та ПЗ яке побудовано на ОСРЧ, надає підтвердження того, що обрана мікроконтролерна платформа є ефективною та те що накладні витрати не є суттєвими. Наявність засобів негайного перепланування завдань в ОСРЧ FreeRTOS надає можливість побудови керуючих систем з прогнозованою реактивністю та впевнено задовольняє вимоги до сучасних обчислювачів.

Оцінка реактивності платформи БО «Боривітер» під керуванням FreeRTOS підтверджує можливість використовувати платформу для завдань з швидкозмінюваним обчислювальним контекстом та зовнішніми чинниками.

Недоліком роботи є відсутність перевірки технології експериментальних досліджень показників реактивності обчислювальної платформи для режимів Sleep, Wait та Backup мікропроцесора ATSAMV71, що і є найближчими планами з подальшого розвитку цієї статті.

Внесок авторів: розробив апаратне та програмне забезпечення, виконав дослідження – **Олександр Любімов**; запропонував концепцію, перевірив отримані дані, відредагував – **Ігор Туркін**; виконав пошук ресурсів, зробив візуалізацію та провів верифікацію тексту – **Олександр Любімов.**

Конфлікт інтересів

Автори заявляють, що немає конфлікту інтересів щодо цього дослідження, фінансового, особистого, авторського чи іншого, який міг би вплинути на дослідження та його результати, представлені в цій статті.

Фінансування

Дослідження проведено в рамках проєкту 2023.04/0143 «*Експериментальне відпрацювання бортового обчислювача безпілотного літального апарата подвійного призначення*» за фінансової підтримки Національного фонду досліджень України.

Доступність даних

Рукопис не має пов'язаних даних.

Використання засобів штучного інтелекту

Автори підтверджують, що при створенні представленої роботи використовували технології штучного інтелекту тільки для вирішення технічних питань, як то визначення УДК, або формування переліку літератури відповідно до вимог оформлення.

Подяки

Автори висловлюють вдячність інжиніринговій компанії “Falco Engineering” LLC за надання до проведення експерименту апаратних платформ та допомогу у налаштуванні та налагодженні інструментів для портування. Для детальнішої інформації https://falco.engineering/.

Усі автори прочитали та погодилися з опублікованою версією цього рукопису.

Література

1. *Liu, C. L. Scheduling Algorithms for Multiprogramming in Hard-Real-Time Environment [Text] / C. L. Liu, & J. W. Layland // Journal of the ACM. – 1973. – Vol. 20, iss. 1. – P. 46-61. DOI: 10.1145/321738.321743.*

2. *Zhang, F. Schedulability Analysis for Real-Time Systems with EDF Scheduling [Text] / F. Zhang, & A. Burns // IEEE Transactions on Computers. – 2009. – Vol. 58, iss. 9. – P. 1250-1258. DOI: 10.1109/tc.2009.58.*

3. *Bernat, G. Weakly hard real-time systems [Text] / G. Bernat, A. Burns, & A. Liamosi // IEEE Transactions on Computers. – 2001. – Vol. 50, iss. 4. – P. 308-321. DOI: 10.1109/12.919277.*

4. *Kluge, F. Utility-Based Scheduling of (m, k)-Firm Real-Time Task Sets [Text] / F. Kluge, M. Neuerburg, & T. Ungerer // Architecture of Computing Systems – ARCS 2015. Lecture Notes in Computer Science. – 2015. – Vol. 9017. – P. 201-211. DOI: 10.1007/978-3-319-16086-3_16.*

5. *Choi, H. Job-Class-Level Fixed Priority Scheduling of Weakly-Hard Real-Time Systems [Text] / H. Choi, H. Kim, & Q. Zhu // IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS). – Montreal, QC, Canada, 2019. – P. 241-253, DOI: 10.1109/rtas.2019.00028.*

6. *Average Task Execution Time Minimization under (m, k) Soft Error Constraint [Text] / J. Shi, N. Ueter, J.-J. Chen, & K.-H. Chen // IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS). – San Antonio, TX, USA, 2023. – P. 1-13. DOI: 10.1109/RTAS58335. 2023.00008.*

7. *Adam, G. K. Timing and Performance Metrics for TWR-K70F120M Device [Text] / G. K. Adam // Computers. – 2023. – Vol. 12, iss. 8. – Article No. 163. DOI: 10.3390/computers12080163.*

8. *Agile Software Development Lifecycle and Containerization Technology for CubeSat Command and Data Handling Module Implementation [Text] / O. Liubimov, I. Turkin, V. Pavlikov, & L. Volobuyeva. // Computation. – 2023. – Vol. 11, iss. 9. – Article No. 182. DOI: 10.3390/computation11090182.*

9. *Turkin, I. Energy-Efficient Scheduling for Portable Computers as Bi-Criteria Optimization Problem*

ISSN 1727-7337 (print)
ISSN 2663-2217 (online)

[Text] / I. Turkin, & A. Vdovitchenko // Green IT Engineering: Concepts, Models, Complex Systems Architectures. Studies in Systems, Decision and Control. – 2017. – Vol. 74. – P. 87-100. DOI: 10.1007/978-3-319-44162-7_5.

10. Mittal, S. A survey of techniques for improving energy efficiency in embedded computing systems [Text] / S. Mittal // International Journal of Computer Aided Engineering and Technology. – 2014. – Vol. 6, iss. 4. – P. 440-459. DOI: 10.1504/IJCAET.2014.065419.

11. Accurate dynamic voltage and frequency scaling measurement for low-power microcontrollers in wireless sensor networks [Text] / R. Chéour, S. Khriji, M. Götz, M. Abid, & O. Kanoun // Microelectronics Journal. – 2020. – Vol. 105. – Article No. 104874. DOI: 10.1016/j.mejo.2020.104874.

12. Reghenzani, F. Software Fault Tolerance in Real-Time Systems: Identifying the Future Research Questions [Text] / F. Reghenzani, Z. Guo, & W. Fornaciari // ACM Computing Surveys. – 2023. – Vol. 55, iss. 14s. – Article No. 306. – P. 1-30. DOI: 10.1145/3589950.

13. Kim, D. Reliability Optimization of Real-Time Satellite Embedded System Under Temperature Variations [Text] / B. Kim, & H. Yang // IEEE Access. – 2020. – Vol. 8. – P. 224549-224564. DOI: 10.1109/ACCESS.2020.3044044.

14. Dynamic Voltage and Frequency Scaling as a Method for Reducing Energy Consumption in Ultra-Low-Power Embedded Systems [Text] / J. Zidar, T. Matic, I. Aleksi, & Z. Hocenski // Electronics. – 2024. – Vol. 13, iss. 5. – Article No. 826. DOI: 10.3390/electronics13050826.

15. Oliveira, G. Scheduling and energy savings for small scale embedded FreeRTOS-based real-time systems [Text] / G. Oliveira, G. Lima // Design Automation for Embedded Systems. – 2023. – Vol. 27. – P. 3-29. DOI: 10.1007/s10617-023-09267-7.

16. A Survey on Resource Management in IoT Operating Systems [Text] / A. Musaddiq, Y. B. Zikria, O. Hahm, H. Yu, A. K. Bashir, & S. W. Kim // IEEE Access. – 2018. – Vol. 6. – P. 8459-8482 DOI: 10.1109/ACCESS.2018.2808324.

17. Low Power Support. Tickless Idle Mode [Electronic Resource] / FreeRTOS. - Available at: http://www.freertos.org/low-power-tickless-rtos.html (accessed: 1.08.2024)

18. Simonovic, M. Power management implementation in FreeRTOS on LM3S3748 [Text] / M. Simonovic, L. Saranovac // Serbian Journal of Electrical Engineering. – 2013. – Vol. 10, iss. 1. – P. 199-208. DOI: 10.2298/SJEE1301199S.

19. Energy-Aware Scheduling for Real-Time Systems: A Survey [Text] / M. Bambagini, M. Marinoni, H. Aydin, & G. Buttazzo // ACM Transactions on Embedded Computing Systems. – 2016. – Vol. 15, iss. 1. – Article No. 7. – P. 1–34. DOI: 10.1145/2808231.

20. De Melo, A. C. C. P. Assessing Power Efficiency and Performance in Nanosatellite Onboard Computer for Control Applications [Text] / A. C. C. P. de Melo, D. C. Café, & R. A. Borges // IEEE Journal on Miniaturization for Air and Space Systems. – 2020. – Vol. 1, iss. 2. – P. 110-116. DOI: 10.1109/JMASS.2020.3009835.

21. A Benchmark Characterization of the EEMBC Benchmark Suite [Text] / J. Poovey, T. Conte, M. Levy, & S. Gal-On // IEEE Micro. – 2009. – Vol. 29, iss. 5. – P. 18-29. DOI: 10.1109/MM.2009.74.

22. Iqbal, S. ParMiBench - An Open-Source Benchmark for Embedded Multiprocessor Systems [Text] / S. Iqbal, Y. Liang, & H. Grahn // IEEE Computer Architecture Letters. – 2010. – Vol. 9, iss, 2. – P. 45-48. DOI: 10.1109/L-CA.2010.14.

23. Zoni, D. A Survey on Run-time Power Monitors at the Edge [Text] / D. Zoni, A. Galimberti, & W. Fornaciari // ACM Computing Surveys. – 2023. – Vol. 55, iss. 13s. – Article No. 325. – P. 1-33. DOI: 10.1145/3593044.

24. Kluge, F. EMSBench: Benchmark and Testbed for Reactive Real-Time Systems [Text] / F. Kluge, C. Rochange, & T. Ungerer // Leibniz Transactions in Embedded Systems (LITES). – 2017. – Vol. 4, Iss. 2. – P. 02:1-02:23. DOI: 10.4230/LITES-v004-i002-a002.

25. Fayyad-Kazan, H. Benchmarking the Performance of Microsoft Hyper-V server, VMware ESXi and Xen Hypervisors [Text] / H. Fayyad-Kazan, L. Perneel, & M. Timmerman // Journal of Emerging Trends in Computing and Information Sciences. – 2013. – Vol. 4, iss. 12. – P. 922-933. – Available at: https://www.academia.edu/48577184/Benchmarking_the_Performance_of_Microsoft_Hyper_V_server_VMware_ESXi_and_Xen_Hypervisors. (accessed 12.3.2024).

26. Сторінка-презентація ДКР авторів статті з розробки БО «Боривітер»/«Falco» [Електронний ресурс]. – Режим доступу: https://www.falco.engineering/ (дата звернення: 15.03.2024).

27. Atmel / SMART ARM-based Flash MCU [Electronic Resource]. – Available at: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-44003-32-bit-Cortex-M7-Microcontroller-SAM-V71Q-SAM-V71N-SAM-V71J_Datasheet.pdf. (accessed: 15.03. 024)

References

1. Liu C. L., & Layland, J. W. Scheduling Algorithms for Multiprogramming in Hard Real-Time Environment, *Journal of the ACM*, 1973, vol. 20, iss. 1, pp. 46-61. DOI: 10.1145/321738.321743.

2. Zhang, F., & Burns, A. Schedulability Analysis for Real-Time Systems with EDF Scheduling. *IEEE*

Transactions on Computers, 2009, vol. 58, iss. 9, pp. 1250-1258. DOI: 10.1109/tc.2009.58.

3. Bernat, G., Burns, A., & Liamosi, A. Weakly hard real-time systems. *IEEE Transactions on Computers*, vol. 50, iss. 4, pp. 308-321. DOI: 10.1109/12.919277.

4. Kluge, F., Neuerburg, M., & Ungerer, T. Utility-Based Scheduling of (m, k)-Firm Real-Time Task Sets. In: Pinho, L., Karl, W., Cohen, A., Brinkschulte, U. (eds) *Architecture of Computing Systems – ARCS 2015*. Lecture Notes in Computer Science, Springer, Cham, 2015, vol. 9017, pp. 201-211. https://doi.org/10.1007/978-3-319-16086-3_16

5. Choi, H., Kim, H., & Zhu, Q. Job-Class-Level Fixed Priority Scheduling of Weakly-Hard Real-Time Systems. *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, Montreal, QC, Canada, 2019, pp. 241-253, DOI: 10.1109/rtas.2019.00028.

6. Shi, J., Ueter,N., Chen, J.-J., & Chen, K.-H. Average Task Execution Time Minimization under (m, k) Soft Error Constraint. *IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, San Antonio, TX, USA, 2023, pp. 1-13, DOI: 10.1109/RTAS58335.2023.00008.

7. Adam, G. K. Timing and Performance Metrics for TWR-K70F120M Device. *Computers*, 2023, vol. 12, iss. 8, article no. 163. DOI: 10.3390/computers12080163.

8. Liubimov, O., Turkin, I., Pavlikov, V., & Volobuyeva, L. Agile Software Development Lifecycle and Containerization Technology for CubeSat Command and Data Handling Module Implementation. *Computation*, 2023, vol. 11, no. 9, article no. 182. DOI: 10.3390/computation11090182.

9. Turkin, I., & Vdovitchenko, A. Energy-Efficient Scheduling for Portable Computers as Bi-Criteria Optimization Problem. In: Kharchenko, V., Kondratenko, Y., Kacprzyk, J. (eds). *Green IT Engineering: Concepts, Models, Complex Systems Architectures. Studies in Systems, Decision and Control*, Springer, Cham, 2017, vol. 74, pp. 87-100. DOI: 10.1007/978-3-319-44162-7_5.

10. Mittal, S. A survey of techniques for improving energy efficiency in embedded computing systems. *International Journal of Computer Aided Engineering and Technology*, 2014, vol. 6, iss. 4, pp. 440-459. DOI: 10.1504/IJCAET.2014.065419.

11. Chéour, R., Khriji, S., Götz, M., Abid, M., & Kanoun, O. Accurate dynamic voltage and frequency scaling measurement for low-power microcontrollers in wireless sensor networks. *Microelectronics Journal*, 2020, vol. 105, article no. 104874, DOI: 10.1016/j.mejo.2020.104874.

12. Reghenzani, F., Guo, Z., & Fornaciari, W. Software Fault Tolerance in Real-Time Systems: Identifying the Future Research Questions. *ACM Computer Survey*, 2023, vol. 55, iss. 14s, article no. 306, pp. 1-30. DOI: 10.1145/3589950.

13. Kim, B., & Yang, H. Reliability Optimization of Real-Time Satellite Embedded System Under Temperature Variations. *IEEE Access*, 2020, vol. 8, pp. 224549-224564. DOI: 10.1109/ACCESS.2020. 3044044.

14. Zidar, J., Matic, T., Aleksi, I., & Hocenski, Z. Dynamic Voltage and Frequency Scaling as a Method for Reducing Energy Consumption in Ultra-Low-Power Embedded Systems. *Electronics*, 2024, vol. 13, iss. 3, article no. 826. DOI: 10.3390/ electronics13050826.

15. Oliveira, G., & Lima, G. Scheduling and energy savings for small scale embedded FreeRTOS-based real-time systems. *Design Automation for Embedded Systems*, 2023, vol. 27, pp. 3-29. DOI: 10.1007/s10617-023-09267-7.

16. Musaddiq, A., Zikria, Y. B., Hahm, O., Yu, H., Bashir, A. K., & Kim, S. W. A Survey on Resource Management in IoT Operating Systems. *IEEE Access*, 2018, vol. 6, pp. 8459-8482. DOI: 10.1109/ACCESS.2018.2808324.

17. *Low Power Support: Tickless Idle Mode*. FreeRTOS. Available at: http://www.freertos.org/low-power-tickless-rtos.html (Accessed 1 August 2024).

18. Simonovic, M., & Saranovac, L. Power management implementation in FreeRTOS on LM3S3748. *Serbian Journal of Electrical Engineering*, 2013, vol. 10, iss. 1, pp. 199-208. DOI: 10.2298/SJEE1301199S.

19. Bambagini, M., Marinoni, M., Aydin, H., & Buttazzo, G. Energy-Aware Scheduling for Real-Time Systems. *ACM Transactions on Embedded Computing Systems*, 2016, vol. 15, iss. 1, article no. 7, pp. 1–34. DOI: 10.1145/2808231.

20. De Melo, A. C. C. P., Café, D. C., & Borges, R. A. Assessing Power Efficiency and Performance in Nanosatellite Onboard Computer for Control Applications. *IEEE Journal on Miniaturization for Air and Space Systems*, 2020, vol. 1, iss. 2, pp. 110-116. DOI: 10.1109/JMASS.2020.3009835.

21. Poovey, J., Conte, T., Levy, M., & Gal-On, S. A Benchmark Characterization of the EEMBC Benchmark Suite. *IEEE Micro*, 2009, vol. 29, iss. 5, pp. 18-29. DOI: 10.1109/MM.2009.74.

22. Iqbal, S., Liang, Y., & Grahn, H. ParMiBench - An Open-Source Benchmark for Embedded Multiprocessor Systems. *IEEE Computer Architecture Letters*, 2010, vol. 9, iss. 2, pp. 45-48. DOI: 10.1109/L-CA. 2010.14.

23. Zoni, D., Galimberti, A., & Fornaciari, W. A Survey on Run-time Power Monitors at the Edge. *ACM Computing Surveys*, 2023, vol. 55, iss. 13s, article no. 325, pp. 1-33. DOI: 10. 1145/3593044.

24. Kluge, F., Rochange, C., & Ungerer, T. EMSBench: Benchmark and Testbed for Reactive Real-

ISSN 1727-7337 (print)
ISSN 2663-2217 (online)

Time Systems. *Leibniz Transcription on Embedded Systems*, 2017, vol. 4, iss. 2, pp. 02:1-02:23. DOI: 10.4230/LITES-v004-i002-a002.

25. Fayyad-Kazan, H., Perneel, L., & Timmerman, M. Benchmarking the Performance of Microsoft Hyper-V server, VMware ESXi and Xen Hypervisors. *Journal of Emerging Trends in Computing and Information Sciences*. 2013, vol. 4, iss. 12, pp: 922-933. Available at: https://www.academia.edu/48577184/Benchmarking_the_Performance_of_Microsoft_Hyper_V_server_VMware_ESXi_and_Xen_Hypervisors. (accessed 12 March 2024).

26. *Storinka-prezentatsiya DKR avtoriv statti z rozrobky BO «Boryviter»/«Falco»* [Page-presentation of the R&D work of the authors of the article on the development of the on-board computer "Boryviter"/"Falco"]. Available at: https://www.falco.engineering/. (Accessed 15 March 2024).

27. *Atmel | SMART ARM-based Flash MCU.* Available at: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-44003-32-bit-Cortex-M7-Microcontroller-SAM-V71Q-SAM-V71N-SAM-V71J_Datasheet.pdf (Accessed 15 March 2024).

Надійшла до редакції 20.07.2024, розглянута на редколегії 15.08.2024

EXPERIMENTAL STUDY OF FREERTOS OPERATING SYSTEM REACTIVITY IN POWER SAVING MODES OF THE ONBOARD COMPUTER MICROCONTROLLER

Oleksandr Liubimov, Ihor Turkin

An onboard computer is a specialized computer system integrated into a vehicle or, in general, another complex technical device that requires autonomous control and high reliability. **The objective of this research** is to investigate the reactivity of the FreeRTOS operating system in the energy-saving mode of an onboard computer microcontroller. **The subject of this research are** the methods, tools, and technologies used in the experimental study of the reactivity of the FreeRTOS operating system. **The aim of this work** is to develop a technology for conducting experimental studies on the reactivity and power consumption of computing platforms. **Tasks:** to analyze real-time systems and typical scheduling algorithms; to review existing methods for evaluating the performance and energy efficiency of microcontrollers of on-board computers; to plan an experiment to study the reactivity and energy consumption of the computing platform; based on the results of the experiment, to analyze the author's platform according to the specified criteria. **Conclusions**. This paper presents an experimental assessment of the reactivity and power consumption of the "Falco/Boryviter" computing platform developed by the authors. The proposed platform was implemented on the Cortex-M7 - ATSAMV71 microprocessor and runs on the open-source real-time operating system FreeRTOS. The experimental results confirmed that the developed microcontroller platform is effective and makes it possible to build control systems with predictable reactivity and acceptable energy costs. The proposed platform can be adapted for tasks with a rapidly changing computational context under the influence of external factors. The disadvantage of this work is the lack of verification of the developed technology through experimental studies for the Sleep, Wait, and Backup modes of the ATSAMV71 microprocessor.

Keywords: onboard computer; software; Boryviter/Falco; computational efficiency; overhead costs; energy efficiency; ATSAMV71; Cortex-M7; ARM; FreeRTOS.

Любімов Олександр Вікторович – асп. каф. інженерії програмного забезпечення, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Туркін Ігор Борисович – д-р техн. наук, проф., зав. каф. інженерії програмного забезпечення, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Oleksandr Liubimov – Ph.D. Student of the Department of Software Engineering, National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine,
e-mail: oleksandr.liubimov@gmail.com, ORCID: 0000-0003-3636-6939, Scopus Author ID: 58099287400.

Ihor Turkin – Doctor of Technical Science, Professor, Head of the Software Engineering Department, National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine,
e-mail: i.turkin@khai.edu, ORCID: 0000-0002-3986-4186, Scopus Author ID: 57203145725.