УДК 004.46
DOI https://doi.org/10.32782/IT/2021-2-4

Володимир КУВАЄВ
доктор технічних наук, професор кафедри програмного забезпечення комп'ютерних систем, Національний технічний університет «Дніпровська політехніка», просп. Дмитра Яворницького, 19, м. Дніпро, Україна, 49005, kuvaiev.v.m@nmu.one
***ORCID:** 0000-0001-6329-071X*
***Scopus Author ID:** 6602411915*

Станіслав ПРОЦЕНКО
старший викладач кафедри кіберфізичних та інформаційно-вимірювальних систем, Національний технічний університет «Дніпровська політехніка», просп. Дмитра Яворницького, 19, м. Дніпро, Україна, 49005, protsenko.s.m@nmu.one
***ORCID:** 0000-0003-2624-0187*
***Scopus Author ID:** 57195672977*

Ольга ШЕВЦОВА
ассистент кафедри програмного забезпечення комп'ютерних систем, Національний технічний університет «Дніпровська політехніка», просп. Дмитра Яворницького, 19, м. Дніпро, Україна, 49005, shevtsova.o.s@nmu.one
***ORCID:** 0000-0002-0148-5877*
***Scopus Author ID:** 57220267804*

Бібліографічний опис статті: Куваєв, В., Проценко, С., Шевцова, О. (2021). Реалізація паралельної багатозадачності в системах реального часу на базі однокристального мікроконтролера. *Information Technology: Computer Science, Software Engineering and Cyber Security*, 2, 26–33, doi: https://doi.org/10.32782/IT/2021-2-4

РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНОЇ БАГАТОЗАДАЧНОСТІ В СИСТЕМІ РЕАЛЬНОГО ЧАСУ НА БАЗІ ОДНОКРИСТАЛЬНОГО МІКРОКОНТРОЛЕРА

Основним напрямом розвитку сучасного приладобудування є створення компактних інтелектуальних датчиків і приладів на базі однокристальних мікроконтролерів які реалізують багатозадачну функціональність. Для програмного забезпечення таких мікроконтролерів критичним є забезпечення режиму реального часу їх функціонування в умовах мінімальних обчислювальних ресурсів з урахуванням критичного часу виконання кожної з задач які відповідний датчик чи прилад реалізує. ***Метою роботи*** *є реалізація паралельної багатозадачності в системах реального часу на базі однокристального мікроконтролера. Реалізація поставленої мети передбачає вирішення завдання створення програмного механізму, що забезпечує паралельне скоординоване виконання декількох задач одночасно при мінімальних обчислювальних ресурсах, з урахуванням обмежень на час реакції системи, що побудована на однокристальному мікроконтролері, на зовнішні і внутрішні події.* ***Методологія*** *вирішення поставленого завдання полягає в виділенні основних типів задач в системах реального часу і створення механізму координації їх виконання, який потребує мінімум обчислювальних ресурсів мікроконтролера.* ***Наукова новизна****. У статті показано, що використання принципу примусової багатозадачності, який заснований на квантуванні часу і використанням алгоритму диспетчеризації з плануванням, що витісняє задачі за пріоритетом, дозволяє реалізувати режим реального часу і квазіпаралельної багатозадачності в системах на однокристальних мікроконтролерах з мінімальними обчислювальними ресурсами.* ***Висновки****. Використання запропонованого механізму реалізації паралельної багатозадачності для однокристальних мікроконтролерів дозволяє ефективно розділити обчислювальні ресурси між задачами, забезпечити функціонування системи на базі мікроконтролера в режимі реального часу і мінімізувати затрати часу на узгодження окремих програмних модулів при їх доробці.*

Ключові слова: *однокристальний мікроконтролер, багатозадачність, режим реального часу, мікро-ОС.*

Volodymyr KUVAIEV
Doctor of Engineering, Professor of Department of Software Engineering, Dnipro University of Technology, 19 Dmytra Yavornytskoho ave., Dnipro, Ukraine, 49005, kuvaiev.v.m@nmu.one
***ORCID:** 0000-0001-6329-071X*
***Scopus Author ID:** 6602411915*

Stanislav PROTSENKO
Senior Lecturer of Department of Cyberphysical and Information-Measuring Systems, Dnipro University of Technology, 19 Dmytra Yavornytskoho ave., Dnipro, Ukraine, 49005, protsenko.s.m@nmu.one
***ORCID:** 0000-0003-2624-0187*
***Scopus-Author ID:** 57195672977*

Olha SHEVTSOVA
Assistant Lecturer of Department of Software Engineering, Dnipro University of Technology, 19 Dmytra Yavornytskoho ave., Dnipro, Ukraine, 49005, shevtsova.o.s@nmu.one
***ORCID:** 0000-0002-0148-5877*
***Scopus-Author ID:** 57220267804*

To cite this article: Kuvaev, V., Protsenko, S, Shevtsova, O. (2021). Realizatsiia paralelnoi bahatozadachnosti v systemakh realnoho chasu na bazi odnokrystalnoho mikrokontrolera [Implementation of parallel multitasking in real-time systems based on a single-chip microcontroller]. *Information Technology: Computer Science, Software Engineering and Cyber Security*, 2, 26–33, doi: https://doi.org/10.32782/IT/2021-2-4

IMPLEMENTATION OF PARALLEL MULTI-TASKING IN A REAL-TIME SYSTEM BASED ON A SINGLE-CHIP MICROCONTROLLER

The future of modern instrument development is in the creation of compact intelligent sensors and devices based on single-chip microcontrollers. These have the capability of implementing multitasking functionality. The software of such microcontrollers is critical in providing real-time operation in terms of minimal computing resources, which take into account the critical execution time of each of the tasks that the respective sensor or device implements. ***The aim*** *of the work is to implement parallel multitasking in real-time systems based on a single-chip microcontroller. Realization of this set purpose decides the problem of creating a software mechanism which provides parallel coordinated performance of several tasks simultaneously at the minimum computing resources. It also takes into account restrictions on the reaction time of the system built on the single-chip microcontroller regarding external and internal events.* ***The methodology*** *for solving this problem is to identify the main types of problems in real-time systems and create a mechanism for coordinating their implementation, which requires a minimum of computing resources of the microcontroller.* ***Scientific novelty.*** *The article shows that the use of the principle of forced multitasking, which is based on time quantization and the use of algorithms with scheduling and displacing priority tasks, allows the implementation of real-time and quasi-parallel multitasking in systems on single-chip microcontrollers with minimal computing.* ***Conclusions.*** *The use of the proposed mechanism for implementing parallel multitasking for single-chip microcontrollers allows the user to effectively divide computing resources between tasks, ensure the operation of the system based on the microcontroller in real time and minimize the time spent on matching individual software modules.*

Key words: *single-chip microcontroller, multitasking, real-time mode, micro-OS.*

Актуальність проблеми. Розвиток приладобудування і первинних засобів перетворення інформації у системах автоматизації характеризується широким застосуванням в них однокристальних мікроконтролерів (ОМК) які дозволяють поєднувати в собі обчислювача, пристроїв введення-виведення для зв'язку з об'єктом та перетворювача, що забезпечує стандартний інтерфейс з комп'ютером. Особливістю таких ОМК є їх дуже обмежені обчислювальні ресурси. Це ускладнює створення прикладних програм, що забезпечують реалізацію багатозадачності при жорстких обмеженнях на час їх виконання – на реалізацію режиму реального часу функціонування пристрів на базі ОМК. Відтак проблема реалізації паралельної багатозадачності в системі реального часу на базі ОМК є актуальною.

Аналіз останніх досліджень і публікацій. Аналіз літератури свідчить, що при розробці програмного забезпечення систем на базі ОМК проблематиці програмування для забезпечення їх функціонування в режимі реального часу уваги практично не приділяється. Так Петин В. А. розглядаючи проекти з використанням мікроконтролера Arduino основну увагу

приділяє питанню застосування стандартних бібліотечних функцій для роботи з периферійними модулями різноманітного призначення (Петин В. А., 2015).

У інтернет-виданні В.Н. Гололобова основна увага приділяється настроюванню розподілених систем на базі ОМК (В.Н. Гололобова). За своєю природою така система є багатозадачною. Як один з варіантів програмного забезпечення, що застосовується в таких системах в роботі розглядається використання операційної системи Raspbian. В той же час дані, за якими можна оцінити спроможність цієї операційної системи забезпечити функціонування програмно-апаратного комплексу в режимі жорсткого реального часу, відсутні.

Поряд з цим існують сучасні операційні системи реального часу, що забезпечують реалізацію багатозадачності з прогнозованими параметрами на час реакції апаратно-програмного комплексу на зовнішні і внутрішні події. До таких операційних систем (ОС) відносяться, зокрема, ОС QNX [3] та ОС контролерів Siemens SIMATIC S7 [4].

Метою статті є дослідження результатів програмної реалізації паралельної багатозадачності в системах реального часу на базі ОМК з обмеженими обчислювальними ресурсами, що спирається на механізми повномасштабних ОС реального часу.

Виклад основного матеріалу. В одноплатних спеціалізованих електронних пристроях до яких належать інтелектуальні датчики і та компактні прилади широке застосування знайшли ОМК з архітектурою мікроконтролера MCS-51 фірми Intel. Вона використовується в мікроконтролерах фірм Siemens, Philips, Atmel та інш. [5]. Останньою повністю програмно сумісною розробкою в цій лінійці ОМК є мікроконтролер N76E003 південнокорейської фірми Novoton [6] (модель 2017 р). Вона повністю зберегла базові обчислювальні ресурси, але завдяки ядру, що виконано за технологією 1T 8051 (1 машинний цикл виконується за один такт), швидкодія мікроконтролера на порядок перевищує швидкодію мікроконтролера MCS-51. Тим не менш, завдяки збереженню базової архітектури і програмної сумісності в цій лінійці мікроконтролерів, механізми і методи реалізація паралельної багатозадачності в системі реального часу на базі цих ОМК ідентичні, хоча часові параметри реакцій на зовнішні події, час перемикання між задачами та інше. можуть відрізнятися.

Аналіз програм, що вирішуються в ОМК типу MCS-51 дозволив узагальнити їх функціональну структуру виділивши три основні типи задач:

– задача введення і первинної обробки інформації;

– задача формування і виведення інформації, що управляє (керуючих впливів);

– задача обміну інформацією з ПЕОМ верхнього рівня і діагностики технічних засобів приладу, інтелектуального датчика (далі – контролера) або керованого механізму. Тобто ця задача розширює верхній рівень обробки інформації.

Кожна з таких задач має свою інформаційну базу, що пов'язана з іншими задачами через загальну інформаційну базу. Перші дві задачі повинні працювати в режимі жорсткого реального часу, третя – функціонувати в фоновому режимі квазіреального часу (м’якого реального часу).

Для скорочення часу на розробку і налагодження програмного забезпечення одноплатних контролерів на базі ОМК була розроблена багатозадачна мікро-операційна система (мікро-ОС) з квазіпаралелізмом, яка орієнтована на типову структуру програмного забезпечення (ПЗ) ОМК, що дозволяє будувати на їх базі системи автоматизації, які працюють в реальному масштабі часу.

В мікро-ОС реалізований принцип примусової багатозадачності заснований на квантуванні часу. При перемиканні задач використовується алгоритм диспетчеризації, що забезпечує витісняюче планування, та засноване на пріоритетах. Взаємодія між задачами забезпечуються механізми, що засновані як на обміні повідомленнями через відповідні черги, так і на використанні загальних областей пам'яті. Реалізований механізм запитів до системи. Система переривань забезпечує буферизоване введення/виведення даних через універсальний асинхронний приймач-передавач (УАПП) ОМК, а також надає можливість перепризначення оброблювачів двох зовнішніх переривань INT0 і INT1. Мікро-ОС надає в розпорядження програміста ряд сервісних функцій які можуть бути використані для організації високорівневого доступу до ресурсів.

Мікро-ОС підтримує одночасне існування до трьох незалежних задач. Дане обмеження обумовлено малим обсягом внутрішньої пам'яті даних зі швидким доступом в ОМК. Пріоритети задач фіксовані, динамічна зміна пріоритетів неприпустима. Однак будь-яка задача може бути знята з виконання за власною вимогою, або з ініціативи системи, а потім відновлена системою при виникненні достатніх для цього умов. Чим вище пріоритет здачі, тим більше квантів часу відводиться їй в циклі черги задач.

Одна (будь-яка) з трьох задач може бути повторюваною, тобто отримувати управління через певні (постійні) проміжки часу.

Розподіл ресурсів здійснюється на етапі конфігурації мікро-ОС під конкретну задачу. Коректне використання ресурсів ґрунтується на дисциплінованості програміста, а поняття «недоступність», що зустрічається в тексті, слід трактувати як «заборона на використання».

Мікро-ОС функціонально включає в себе: область системних даних, процедуру початкової ініціалізації, супервізор, обробники переривань, функції системного сервісу. Вона надає наступні ресурси доступні із задач (див. рис. 1):

а) загальна область пам'яті (2К). Може використовуватися здачами без будь-яких обмежень. Відповідальність за розподіл пам'яті повністю лежить на програмістові;

б) черга системних повідомлень (16Б). Використовується задачами для направлення запитів до системи. Наприклад запитів на зняття. Доступна задачам тільки на запис;

в) буфер введення УАПП (64Б). Заповнюється оброблювачем переривання УАПП по надходженні даних з лінії зв'язку. Переглядається супервізором на предмет керуючих повідомлень. Доступний задачам тільки на читання;

г) буфер виведення УАПП (64Б). Дані знаходяться в буфері витягуються оброблювачем переривання УАПП для передачі по лінії зв'язку. Доступний здачам тільки на запис;

д) черги повідомлень задач (16Б). Використовується мікро-ОС для направлення повідомлень задачам, і задачами для зв'язку одна з одною. Своя черга повідомлень доступна задачі тільки на читання, черги інших задач – тільки на запис;

е) байт ознак стану черг і буферів. Відображає поточний стан ресурсів: 0-порожній, 1 – непорожній. Окремі біти ознак по кожному ресурсу встановлюються або скидаються як правило сервісними функціями, що відповідають за доступ до цих ресурсів. Переповнення / спустошення буферів і черг контролю-

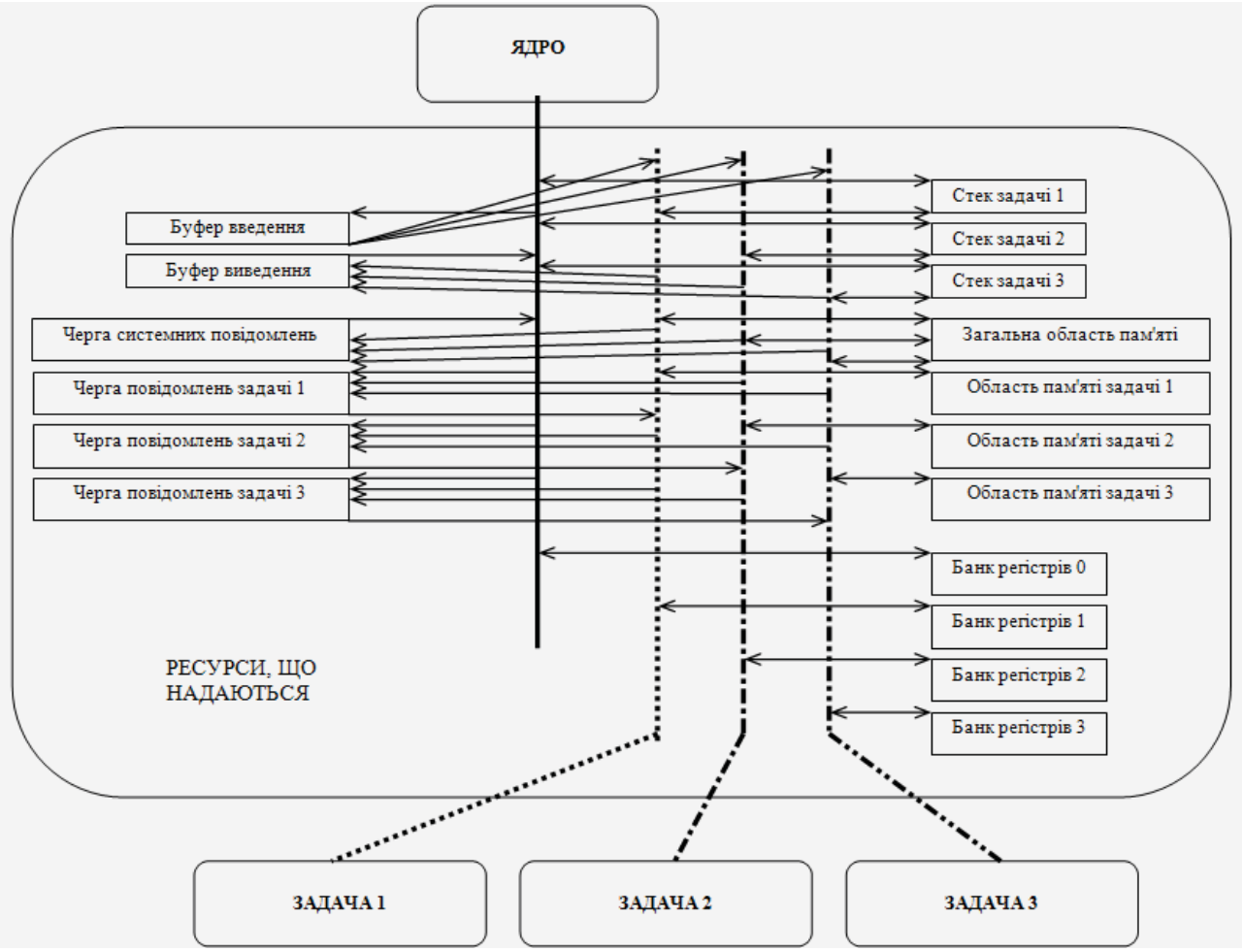

Рис. 1. Структура наданих ресурсів і права доступу до них (Стрілками вказані права доступу: читання/запис/запис-читання)

ються цими ж функціями. Задачам, в більшості випадків, достатньо лише контролювати ознаки;

ж) області пам'яті задач (2К). Кожній задачі надається область пам'яті, за замовчуванням 2К, недоступна іншим задачам і самій мікро-ОС. Може використовуватися задачами без будь-яких обмежень. Відповідальність за розподіл пам'яті повністю лежить на програмістові;

з) регістри задачі (8Б). Кожній задачі надається окремий банк регістрів. Перемикання банків здійснюється супервізором автоматично при перемиканні контексту задач;

і) стеки задач (16Б). Для кожної задачі виділено стек розміром 16 байт. Перемикання між стеками здійснюється супервізором при перемиканні контексту. Супервізор для своєї роботи використовує стек перерваної задачі.

Системні дані, в основному, не можуть бути доступні із задач, і використовуються мікро-ОС для налаштування на етапі ініціалізації, контролю та управління при перемиканні задач супервізором.

Задачам доступні на запис тільки біти умов відновлення при знятті у таблиці дескрипторів задач.

Область даних містить:

– константи ініціалізації;
– описи стандартних повідомлень і команд;
– таблицю дескрипторів задач;
– вказівники на голови і хвости черг і буферів, а також на області пам'яті, що відведені останнім;
– індикатор стану черг і буферів;
– лічильник квантів виконання поточної задачі;
– індикатор відновлення задачі, що повторюється;
– прапор наявності задачі, що повторюється;
– часові змінні;
– часові прапори.

Таблиця дескрипторів задач містить наступні поля для кожної задачі:

1 байт – номер (ім'я) задачі;

1 байт – вказівник стека задачі;

2 байта – точка початкового входу в задачу;

1 байт – пріоритет задачі (кількість квантів, що відводиться задачі);

1 байт – стан задач;

1 біт – тип задачі (0-циклічна / 1-повторювана);

1 біт – готовність задачі (0-немає готовності / 1- є готовність);

1 біт – ознака відновлення при наявності своїх повідомлень;

1 біт – ознака відновлення при наявності даних від УАПП;

3 біта – резерв.

При запуску системи здійснюється:

– настройка режимів таймерів/лічильників;
– налаштування УАПП;
– налаштування системи переривань;
– завантаження вказівника стека значенням бази системного стека. Системний стек необхідний тільки при першій активізації супервізора;
– обнуління оперативної пам'яті;
– ініціалізація таблиці дескрипторів задач відповідно до значень дескрипторів задач;
– формування стеків задач (вказівників стеків задач) для забезпечення коректного першого входження в кожну задачу;
– ініціалізація вказівників голів і хвостів черг і буферів введення/виведення;
– контроль наявності в системі задачі, що повторюється. (При відсутності такої скидається прапор наявності задачі, що повторюється. У разі присутності задачі, що повторюється, індикатор відновлення завантажується значенням з дескриптора відповідної задачі, і встановлюється прапор наявності задачі, що повторюється);
– настройка черги задач на першу задачу;
– дозвіл переривань;
– запуск таймера-лічильника ТЛ0.

Далі, до першого входження в супервізор, програма знаходиться в порожньому циклі.

Супервізор є по суті оброблювачем переривання від ТЛ0 і активізується кожні 10 мс виконуючи такі дії:

– при отриманні управління супервізор здійснює заборону переривань, перезавантажує коефіцієнт ділення ТЛ0, після чого зберігає контекст перерваної задачі зберігаючи в її стеку регістри A, B, PSW, DPH, DPL і перемикає банки регістрів, роблячи активним банк-0 (системний банк регістрів);

– після цього відпрацьовується процедура часу. Системний годинник реалізований у вигляді лічильників містять даний час з моменту запуску системи, в форматі: [десятки мс] [сотні мс] [секунди] [хвилини] [години]. По досягненню вмістом лічильника граничного значення він обнуляється, відбувається перенесення одиниці в наступний старший лічильник, і зводиться прапор переходу кордону відповідного інтервалу часу, в байті часових прапорів. Є також кільцевий лічильник квантів (по 10 мс);

– далі, контролюється наявність в системі задачі, що повторюється (за станом прапора наявності повторюваної задачі, що повторюється), і в разі її відсутності управління передається процедурі контролю пріоритетів задач;

– у разі, якщо прапор наявності задачі, що повторюється, встановлено (така присутня), – декрементується індикатор відновлення задачі, після чого контролюється його стан, і в разі нерівності нулю скидається прапор готовності задачі і управління передається процедурі контролю пріоритетів задач;

– у разі рівності нулю індикатора відновлення (прийшов час відновити повторювану задачу), – визначається номер задачі, що повторюється, і індикатор повторення перезавантажується значенням з дескриптора відповідної задачі, черга задач завантажується номером задачі, що повторюється, зводиться прапор готовності задачі. Після цього управління передається процедурі контролю запитів до системи минаючи контроль пріоритетів;

– потім здійснюється контроль поточного значення пріоритетного числа перерваної задачі, і, в разі якщо його значення не дорівнює нулю, воно декрементується, і здійснюється «негайне» повернення в перервану задачу. Цей процес завершується коли поточне значення пріоритетного числа дорівнює нулю;

– далі, здійснюється контроль стану черги системних повідомлень, і при наявності таких вони витягуються з черги шляхом виклику сервісної функції **take**, проводиться їх аналіз, і при виявленні запитів на зняття, біт готовності відповідної задачі скидається шляхом виклику сервісної функції **taskNotRedy**. Процес повторюється до повного спустошення черги системних повідомлень. Таким чином всі запити, що надійшли до чергового перемикання задач, будуть обслужені. Запити, що не є стандартними повідомленнями і командами, не обслуговуються і будуть втрачені;

– на наступному етапі контролюється наявність знятих (не готових) задач і, при виявленні останніх, аналізується можливість і умови їх відновлення. Якщо відновлення допустимо і необхідні умови виконуються, то біт готовності відповідної задачі зводиться шляхом звернення до сервісної функції **taskRedy;**

– нарешті починається процес безпосереднього перемикання задач.

Черга задач реалізована в регістрі R1 системного банку регістрів і містить номер наступної задачі. Цей номер аналізується, і якщо задача підлягає виконанню – є зараз не готовою (знятою), то відбувається просування черги задач (R1 інкрементується) і перехід до аналізу стану наступної задачі. Таким чином в разі неготовності всіх задач управління залишається всередині супервізора.

У тому випадку якщо наступна задача готова до виконання відбувається збереження стека перерваної задачі у відповідному полі таблиці дескрипторів. Якщо була перервана задача, що повторюється, – викликається сервісна функція **stackModel**, яка формує порожній стек і зберігає вказівник на нього у відповідному полі таблиці дескрипторів, для забезпечення можливості подальшого відновлення з точки початкового входу. Вказівник стека завантажується з таблиці дескрипторів значенням SP задачі, що активізується. Відбувається перемикання активного банку регістрів, «просування» черги задач, завантаження поточного пріоритетного числа значенням пріоритету задачі, що активується (з таблиці дескрипторів). Після чого здійснюється контроль коректності стеків задач за значеннями їх баз, відновлюються з стека регістри A, B, PSW, DPH, DPL задачі, що активізується, дозволяються переривання, і управління передається задачі через застосування команди **reti** в те місце, де ця задача була раніше перервана (включаючи вкладені функції і переривання), незалежно від того чи була вона до цього знятої чи ні.

При тактовій частоті 12 МГц MCS-51 час витрачається на перемикання контексту задач становить:

мінімум – близько 100 мкс (при порожній черзі системних повідомлень);

в середньому – близько 200 мкс (при наявності 3 повідомлень в черзі);

максимум – близько 1000 мкс (при повній черзі системних повідомлень).

Система переривань надає три апаратних переривання – від зовнішніх джерел INT0 і INT1, і від приймача УАПП. І одне комбіноване апаратно-програмне переривання передавача УАПП. Обидва переривання УАПП мають загальний оброблювач. Оброблювачі перериванъ INT0 і INT1 повинні перевизначитися для кожної конкретної конфігурації програмно-апаратного комплексу і за замовчуванням не виконують ніяких дій, відразу ж повертаючи управління. Оброблювач переривання УАПП виконує функції буферізованого введення/виведення по послідовному каналу зв'язку.

При виникненні переривання УАПП оброблювач визначає джерело переривання – приймач або передавач.

У разі переривання приймача, прийнятий байт зчитується з вхідного регістра УАПП у вхідний буфер УАПП (64Б), і, в разі якщо буфер був до цього порожній, встановлюється індикатор стану буфера. При переповненні буфера прийняті дані губляться і зводиться відповідна ознака.

Переривання передавача перший раз необхідно активізувати програмним шляхом, встановивши в 1 біт TI регістра SCON. ОМК з задачі вимагає виведення даних після підготовки інформації у вихідному буфері УАПП.

У разі переривання передавача скидається прапор запиту переривання передавача, черговий байт зчитується з вихідного буфера УАПП у вихідний регістр УАПП, і, в разі якщо буфер спустошений, індикатор стану буфера скидається. Відбувається повернення з оброблювача.

Після закінчення передачі байта, УАПП знову виставляє запит на переривання передавача. У разі якщо буфер вже порожній – здійснюється негайне повернення з оброблювача. Таким чином, якщо в буфері знаходиться *n* байт, то оброблювач активізується *n+1* раз (причому останній раз час обробки мінімальний).

Мікро-ОС надаються такі сервісні функції:

taskRedy – зводить ознаку готовності задачі;

taskNotRedy – скидає ознаку готовності задачі;

insert – помістити елемент в чергу або буфер. Розміщує байт в хвості, відповідним чином пересуваючи вказівник;

take – витягує елемент з черги або буфера. Витягує байт з голови, відповідним чином пересуваючи покажчик;

stackModel –формує порожній стек задачі, номер якої переданий в якості параметра, у вигляді: [точка початкового входу LO] [точка початкового входу HI] [0] [0] [PSW] [0] [0]; і зберігає вказівник на нього в полі дескрипторної таблиці, відповідному цієї задачі;

delay – реалізує програмну затримку виконання.

Кожній задачі передує, в своїй області пам'яті програм, дескриптор задачі розташований за строго фіксованою адресою. Дескриптор задачі має наступні поля:

1 байт – номер (ім'я) задачі;

1 байт – пріоритет задачі (кількість квантів для виконання);

1 байт – тип задачі (0-циклічна / 1-повторювана);

1 байт – адреса бази стека задачі;

1 байт – старший байт адреси точки початкового входу в задачу;

1 байт – молодший байт адреси точки початкового входу в задачу;

5 байт – часовий проміжок між поновленнями задачі.

Вміст дескриптора використовується системою на етапі початкової ініціалізації для заповнення таблиці дескрипторів. Далі, за дескрипторами, може слідувати безпосередньо код задачі.

Можливі два типи задач – циклічні і повторювані.

Циклічні задачі виконуються «постійно» і можуть бути зняті тільки за власною ініціативою або за ініціативою супервізора при виявлені помилки в роботі задачі. При знятті з власної ініціативи задача, перед викликом **taskNotRedy,** може встановити одну з ознак можливості відновлення за умовою. При виконанні зазначеної умови задача буде відновлена супервізором (наприклад відновлення за наявністю повідомлень в своїй черзі, і за наявністю даних у вхідному буфері УАПП).

Залежно від пріоритету задачі, їй в кожному циклі черги задач, надається (поспіль) кількість квантів число яких дорівнює її пріоритетному числу. Задача може бути знята тільки після відпрацювання всіх відведених їй в поточному циклі квантів, тобто тільки при перемиканні на іншу задачу.

Задача, що повторюється, виконується через певні, строго фіксовані, проміжки часу, і може мати пріоритет відмінний від нуля. Після закінчення задачі, що повторюється, незалежно від того яка задача була перервана, відбувається перемикання на наступну, по ходу черзі, задачу. Після закінчення задачі, що повторюється, не відбувається збереження поточного вказівника стека, таким чином вона завжди починається спочатку. З цього випливає, що для задачі, що повторюється, повинно бути відведено час (кількість квантів) який декілька перевищує час, що реально необхідний для повного завершення задачі. Задача має завершуватися циклом, можливо порожнім. Таким чином деякий непродуктивний простій практично неминучий.

Описана мікро-ОС була використана при модернізації програмного забезпечення приладу, який контролює магнітну фазу в прокаті на виході установки термомеханічного зміцнення прокату з прокатного нагріву [7].

Програмне забезпечення приладу складається з трьох модулів: задачі управління намагнічуванням прокату, задачі контролю кількості магнітної фази і задачі інформаційної підтримки. Всі три задачі працюють під управлінням розглянутої мікро-ОС.

Перші два задачі здійснюють управління датчиком магнітної фази і обробку первинної інформації в реальному масштабі часу. Третя задача забезпечує обмін інформацією з ПЕОМ, видачу інформації про режим термомеханічного зміцнення на панель приладу і введення з неї інформації від органів управління, а також здійснює функції діагностики приладу.

Використання розробленої мікро-ОС дозволило ефективно розділити обчислювальні ресурси КР 1816ВЕ51 між задачами, а також мінімізувати витрати часу на узгодження окремих програмних модулів при їх доопрацюванні.

Висновки. Структури обчислювальних ресурсів однокристальних мікроконтролерів накладають обмеження як на загальну кількість завдань, що функціонують квазіпаралельно, так і на можливості реалізації режиму реального часу кожної задачі. Доцільно в прикладному програмному забезпеченні однокристального мікроконтролера виділити три основні задачі: задачу введення і первинної обробки інформації, задачу формування і виведення інформації, що управляє, і задачу обміну інформацією з ПЕОМ верхнього рівня з реалізацією режиму реального часу тільки для перших двох типів задач. Запропоновані програмні механізми реалізації паралельної багатозадачності в системі реального часу на базі однокристального мікроконтролера забезпечують стійке функціонування приладів, спрощують розробку і модернізацію програмного забезпечення приладів та інтелектуальних датчиків.

ЛІТЕРАТУРА:

1. Петин В.А. Проекты с использованием микроконтроллера Arduino. 2-е изд. перераб. и доп. Санкт-Петербург : БХВ-Петербург, 2015. 464 с.

2. Гололобов В.Н. IoT как ИНТЕРАНЕТ вещей с Rasbtrry Pi и MajorDoMo. Москва, 2019. 217 с. URL: https://www.razym.org/tehnicheskaya/electronika/397319-gololobov-vn-iot-kak-intranet-veschey-s-raspberry-pi-i-majordomo.html.

3. Операционная система реального времени QNX Neutrino 6.3. Системная архитектура: Пер. с англ. Санкт-Петербург : БХВ-Петербург, 2005. 336 с.

4. Альтерман И.З. Программируемые контроллеры SIMATIC S7. 1-й уровень профессиональной подготовки S7_PROF1_PA. 2011. 68 с. URL: http://www.promautomatic.ru/catalog/S7_PROF1_PA.pdf.

5. Мікропроцесорна техніка [Текст]: навч. посібник / В.В. Ткачев, Г. Грулер, Н. Нойбергер та інш. Дніпро : Національний гірничий університет, 2012. 188 с.

6. Nuvoton 1T 8051-based Microcontroller. N76E003 Datasheet. 2017. 263 p. URL: https://www.nuvoton.com/export/resource-files/DS_N76E003_EN_Rev1.09.pdf.

7. Шеремет В.А., Бабенко М.А., Кекух А.В., Костюченко М.И., Кокшаров А.Н., Кузнецов Г.А., Куваев В.Н., Чигринский В.А., Иванов Д.А., Карпинский Ю.П. Электромагнитный контроль процесса термоупрочнения проката крупных сечений. *Металлургическая и горнорудная промышленность.* 2004. № 6. С. 102–105.

REFERENCES:

1. Petin V.A. Projects using the Arduino microcontroller. 2nd ed. revised and add. SPb : BHV-Petersburg, 2015. 464 p.

2. Gololobov V.N. IoT as INTERANET of things with Rasbtrry Pi and MajorDoMo. M., 2019. 217 p. Electronic resource for downloading: https://www.razym.org/tehnicheskaya/electronika/397319-gololobov-vn-iot-kak-intranet-veschey-s-raspberry-pi-i-majordomo.html.

3. The real-time operating system QNX Neutrino 6.3. System architecture: Per. from English. SPb : BHV-Petersburg, 2005. 336 p.

4. Alterman I.Z. SIMATIC S7 programmable controllers. 1st level of professional training S7_PROF1_PA. 2011. 68 p. Electronic resource for downloading: http://www.promautomatic.ru/catalog/S7_PROF1_PA.pdf.

5. Microprocessor technology “Text”: navch. posibnik / V.V. Tkachev, G. Gruler, N. Neuberger and insh. D : National State University, 2012. 188 p.

6. Nuvoton 1T 8051-based Microcontroller. N76E003 Datasheet. 2017. 263 p. Electronic resource for downloading https://www.nuvoton.com/export/resource-files/DS_N76E003_EN_Rev1.09.pdf

7. Electromagnetic control of the thermo-hardening process for large-section rolled steel. Sheremet, M.A. Babenko, A.V. Kekukh, M.I. Kostyuchenko, A.N. Koksharov, G.A. Kuznetsov, V.N. Kuvaev, V.A. Chygrynskiy, D.A. Ivanov, Yu.P. Karpinsky // Metallurgical and mining industry. 2004. No. 6. S.102-105.