УДК 621.62

ДОСЛІДЖЕННЯ РЕЖИМІВ ЕНЕРГОЗБЕРЕЖЕННЯ МІКРОПРОЦЕСОРНИХ СИСТЕМ

Мосур І. В., Голубєв Л. П.

Київський національний університет технологій та дизайну

Мета. *Виконати аналіз режимів енергозбереження мікропроцесорних систем. Проаналізувати роботу мікропроцесора з використанням різних елементів живлення. Розробити нове технічне рішення для збільшення часу автономної роботи мікропроцесорної системи.*

Методика. *У роботі використана методика технології переривань та режимів енергозбереження мікропроцесорної системи.*

Результати. *Проведено аналіз та порівняння джерел живлення. Розроблено технологію енергозбереження для збільшення часу автономної роботи мікропроцесорної системи.*

Наукова новизна. *Розроблено новий комбінований механізм збереження енергії мікропроцесорної системи на основі використання режимів енергозбереження мікропроцесора та технології переривань.*

Практична значимість. *Розроблений механізм енергозбереження є універсальним і може бути застосований для збільшення часу автономної роботи в різних мікропроцесорних системах.*

Ключові слова: *мікропроцесорна система, мікропроцесор, режим енергозбереження, переривання, елемент живлення*

Питання про альтернативні режими енергозбереження, що здатні значно подовжити час автономної роботи, не збільшуючи при цьому розмір приладу та джерел живлення, стоїть дуже гостро. У світлі швидкого розвинення технологій технічне рішення де працівнику тієї чи іншої інстанції постійно потрібно слідкувати за роботою та електроживленням приладу, замість того щоб приділити увагу роботі в цілому – є недоречним. Тим паче що більше споживання енергії елементів живлення веде за собою їх швидке зношення, що в свою чергу потреба у швидкій утилізації. А підприємств, які належним чином можуть утилізувати елементи живлення, в Україні немає.

Постановка завдання

На сьогоднішній день проблема автономності систем є однією з найважливіших в різних сферах де використовуються мікропроцесорні системи. Проблема полягає в тому що, хоч система і не потребує постійної роботи, а тільки коли спрацьовує запит, в режимі очікування цього запиту система продовжує споживати енергію. Як наслідок більше споживання енергії, що в свою чергу веде до зменшення часу автономної роботи і потреба в постійній заміні елементів живлення.

Тому виникає необхідність створити механізм енергозбереження який здатний значно подовжити час автономної роботи системи.

Результати досліджень

Збільшення енергозбереження приладу можна досягти завдяки двом чинникам:

- правильно підібраного елементу живлення.
- енергоефективного режиму роботи самої системи.

В першу чергу треба розглянути та дослідити тип елементу живлення, який має дуже важливе значення для автономності пристрою.

Елементи живлення (акумулятори і батареї) відносяться до хімічних джерел струму (ХДС) – пристроїв, в якому хімічна енергія безпосередньо перетворюється в електричну в процесі протікання хімічної реакції між речовиною окислювачем і речовиною відновником [1, 2, 5].

В даний час ХДС використовують у всіх областях техніки і народного господарства. Кількість елементів і акумуляторів, що виготовляються щорічно у всьому світі, обчислюється мільярдами. При одночасному їх включенні можна було б отримати електричну потужність, порівнянну з потужністю всіх електростанцій світу.

За принципами роботи ХДС поділяють на три групи: первинні, вторинні і паливні елементи.

На відміну від акумуляторів, всі батареї не перезаряджаються і підлягають утилізації після того, як вони виробили свій ресурс. Завдяки використанню акумуляторів можна зменшити забруднення хімічними джерелами живлення навколишнього середовища України.

Акумулятори (вторинні ХДС). Акумулятори – джерела струму, в яких хімічні процеси є оборотними. Акумулятори, на відміну від батарейок, можна перезаряджати за допомогою зарядних пристроїв. Для акумуляторів число циклів заряд-розряд в середньому становить 1000 циклів. Ємність акумуляторів зазвичай вимірюють в ампер-годинах.

Тривалість роботи акумулятора можна підрахувати за формулою:

$$T_ч = \frac{C_{мА*ч}}{I_{мА}} \quad (1)$$

де С – ємність акумулятора (мА * ч); I – струм (мА).

Різні типи акумуляторів мають відмінності за основними параметрами: кількістю циклів перезарядки, максимального терміну зберігання, ємності, розмірами, температурному діапазону роботи, можливостям прискореної зарядки і характеризуються різною вартістю. Характеристики акумулятора залежать від матеріалу електродів і складу електроліту.

В даний час існує більше двох десятків типів акумуляторів, найбільш поширені акумулятори наступні, нікель-кадмієві (NiCd), нікель-метал-гидридні (NiMH), літій-іонні (Li-ion), характеристики яких наведені в таблиці.

Таблиця

Переваги та недоліки елементів живлення

Тип акумулятора	Переваги	Недоліки
Нікель-кадмієві акумулятори (NiCd)	1. Низька вартість 2. Довговічність	1. Містять в своєму складі токсичний кадмій 2. Негативно переживають перерозряд.
Нікель-метал-гидридні акумулятори (NiMH)	1. Не містять кадмію 2. Велика питома ємність	1. Падіння ємності акумулятора після 200-300 циклів 2. Більший струм саморозряду
Літій-іонні акумулятори (Li-ion)	1. Досить висока ємність акумулятора 2. Відносно не висока вартість	1. Схильні до старіння (стають непридатними якщо ними не користуються)

Отже треба зробити висновок що для досягнення більшої автономності тип елементу живлення має дуже важливе значення. Головна перевага літій-іонних акумуляторів полягає у високій питомій ємності (щонайменше в два рази більшою, ніж у нікель-кадмієвих акумуляторів). Літій-іонні акумулятори мають відносно низький саморозряд і завдяки своїм перевагам підходять для живлення найбільше. Але щоб збільшити час автономної роботи пристрою лише використовування літій-іонних акумуляторів буде замало. Для того щоб значно подовжити роботу системи треба застосувати правильний енергозберігаючий режим для самої системи. Правильно підібраний елемент живлення і налагоджена робота мікропроцесорної системи – запорука тривалої та стабільної роботи системи.

Визначившись за елементом живлення, треба дослідити яким чином мікропроцесорна система може зберігати енергію, за допомогою яких режимів, як

використовувати технологію переривань щоб подовжити час автономної роботи системи [3, 6, 8].

В першу чергу треба розглянути чому взагалі мікроконтролер взагалі споживає енергію. В мікроконтролері є багато транзисторів, що працюють в так званому ключовому режимі (по типу звичайної кнопки). Основне споживання потужності йде в момент перемикання ключа, тобто чим менше перемикань – тим менше енергоспоживання. У звичайному режимі роботи сам чіп на 8 МГц споживає приблизно 10,2 мА що абсолютно не підходить для автономних пристроїв, таких як дистанційні вимикачі, пристрої охорони, годинник і метеостанції, і інші.

Є кілька способів знизити споживання мікросхеми:

- зменшити кількість перемикачів, відключивши ядро або додаткові генератори і таймери;
- відключити аналогову периферію, яка вимагає купу енергії;
- перевести виводи «ніжки» мікросхеми в Z-стан;

Частина цих способів автоматично виконується при включенні різних режимів енергозбереження. В роботі [9] детально розглянуті режими енергозбереження мікропроцесора. Існують наступні режими енергозбереження.

Idle – в цьому режимі відключається CPU – процесор, наш обробник команд і flash-пам'ять. При цьому залишаються включеними послідовний інтерфейс SPI, USART, аналоговий компаратор, аналого-цифровий перетворювач (ADC), інтерфейс TWI – він же I2C, все таймери, в тому числі і сторожовий, ну і система переривань. Якщо нам не потрібно використовувати компаратор, АЦП або сторожовий таймер, то до засипання можна відключити це в ручну, і заощадимо ще більше енергії. Цей режим буквально створений для роботи з периферією, коли потрібно швидко прокинутися по команді від зовнішніх інтерфейсів і не гальмувати. Але це нажаль не досить енергозберігаючий режим, адже виходить навіть не «сон», а якась «гібернація».

Power-Down – в цьому режимі зупиняється практично все, крім обробки зовнішніх переривань, інтерфейсу TWI і сторожового таймера. Викликати в такому стані переривання можуть тільки: зовнішні скидання (reset), скидання по сторожовому таймеру, при провалі напруги. Також може перерватися при роботі TWI і зовнішніми перериваннями. Більшість лічильників і таймерів тут теж зупиняються – таким чином, в такому режимі з мікросхемою можуть працювати тільки асинхронні інтерфейси. Загалом, в такому режимі чіп засинає «намертво», прокинутися самому йому досить

проблематично, а вже якщо розбудили – прокидатися він буде доволі довго, пропускаючи мимо виводів всю інформацію від переривань.

Power-save – цей режим схожий на Power-down, але якщо таймер встановлений в асинхронному режимі – біт ACCP.AS2 = 1 – то він буде працювати; якщо таймер так не встановлений, то DataSheet радить використовувати Power-down – адже при пробудженні, регістр цього таймера буде не визначений. В такому режимі мікросхема занурена в глибокий сон, але десь глибоко в підсвідомості чіп про себе зазначає, скільки часу він знаходився в стані «сон». Цей режим хороший для пристроїв, що вимагають знання про частоту виходу із сплячого режиму, наприклад, для годинників. Але, на жаль, він не підходить для точного вимірювання часових відрізків між подіями.

Standby – він теж схожий на Power-down, але використовується при роботі від зовнішнього джерела тактування. Цей режим використовується, якщо потрібно зловити будь-яку подію – наприклад, в клавіатурі чіпу треба прокинутися до того, як користувач відпустить клавішу. Ще один гарний приклад – автоспуск плівкового фотоапарата на блискавку.

Extended Standby – схожий на Power-Save, але залишає включеним джерело тактування. Ось цей режим якраз для пристроїв, яким потрібно точно знати, скільки часу минуло після входу в режим сну – наприклад, для вимірювача швидкості обертання колеса велосипеда.

ADC Noise Reduction – режим який буде найдоречнішим. Цей режим зупиняє процесор, але залишає працювати АЦП, систему зовнішніх переривань, інтерфейс TWI, сторожовий таймер і таймер або лічильник, відповідно зупиняються тільки годинник, CPU і FLASH. Все це призначено для того, щоб зменшити внутрішні перешкоди (що виникають при перемиканні транзисторів самої мікросхеми) при роботі АЦП – так виходять вимірювання більш високої точності.

Важливим є те, що вся робота, як і у випадку програмного режиму, здійснюється самим мікропроцесором, зовнішня подія просто тимчасово відволікає його від виконання програми. Реакція на зовнішню подію по перериванню, в загальному випадку, повільніша, ніж при програмному режимі. Як і у випадку програмного обміну, тут усі сигнали на магістралі виставляються процесором, тобто він цілком контролює магістраль. Для обслуговування переривань у систему іноді вводиться спеціальний модуль контролера переривань, але він в обміні інформацією участі не бере. Його завдання полягає в тому, щоб спростити роботу процесора з зовнішніми запитами

переривань. Цей контролер, зазвичай, програмно керується процесором та системною магістраллю. Природно, ніякого прискорення роботи системи переривання не дає. Його застосування дозволяє тільки відмовитися від постійного опитування прапора, а значить дозволяє такій системі перейти в режим «сон», до моменту коли мікропроцесорна система буде знов задіяна.

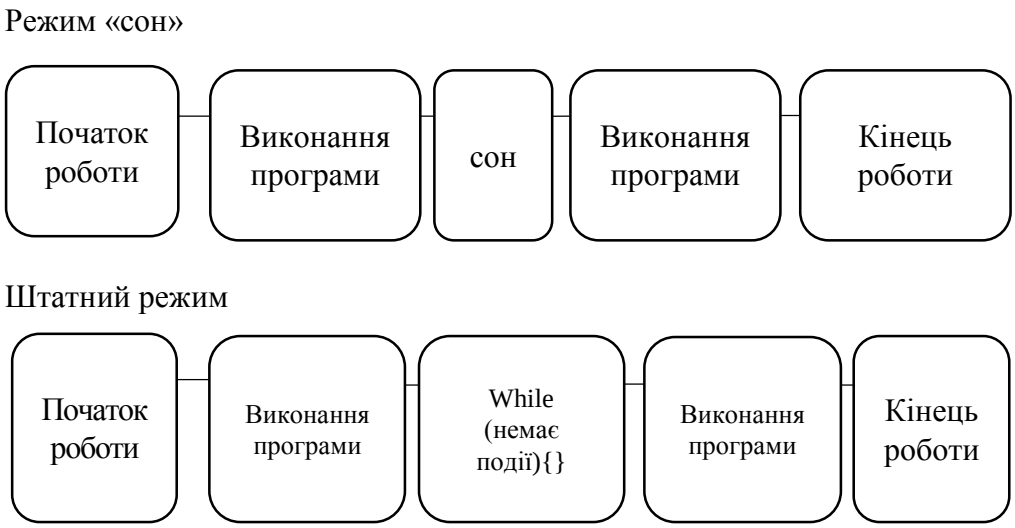

Рис. 1. **Принцип роботи програми у сплячому і в штатному режимах**

Принцип роботи мікропроцесорної системи у штатному режимі та в «сплячому» режимі показаний на рис. 1. Виходить, що в більшості режимів мікросхема взагалі не розуміє, що була в енергозберігаючому режимі, і продовжувала споживати енергію. За роботу зі сплячими режимами відповідає регістр MCUCR [9]. Регістри MCUCR в ATtiny 2313 і ATmega 8/32 показані на рис. 2.

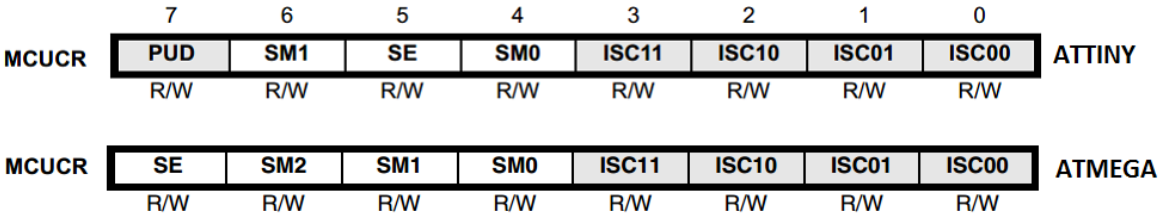

Рис. 2. **Регістр MCUCR в ATtiny 2313 і ATmega 8/32**

Нам будуть потрібні біти SM0, SM1 І SM2 – щоб задати тип сплячого режиму. Наприклад, в ATmega8 сплячі режими задаються так як показано на рис. 3.

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC Noise Reduction
0	1	0	Power-down
0	1	1	Power-save
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby

Рис. 3. **Сплячі режими ATmega8**

Після визначення типу режиму залишається тільки встановити біт SE в 1 та перевести в потрібний момент мікросхему в сплячий режим командою асемблера – asm.sleep, потрібно буде тільки продумати як виводити систему з цього режиму [4, 7]. Принцип роботи програми з обробкою переривань показаний на рис. 4.

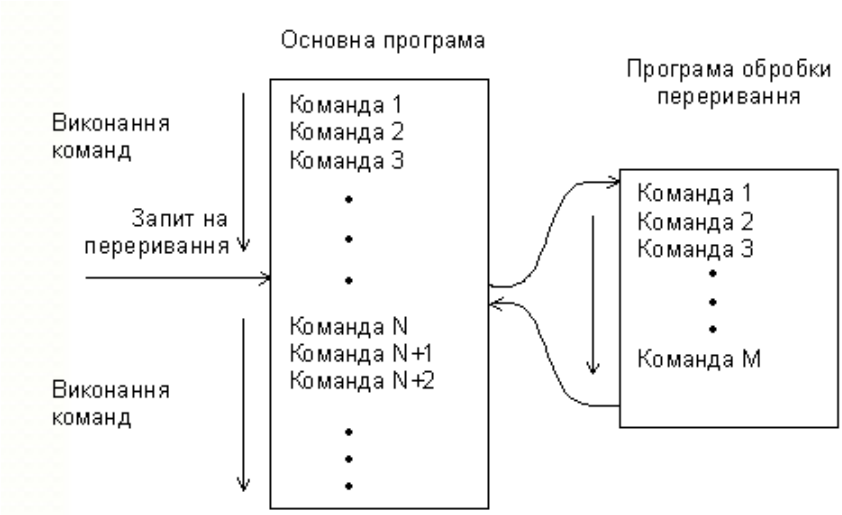

Рис. 4. **Схема роботи програми з обробкою переривань**

Нижче наведено фрагмент програмного коду для мікропроцесорної системи в якому за допомогою використання комбінованого методу (режимів енергозбереження та технології переривань) збільшується час автономної роботи системи в 2 рази. Для тестування було взято найпростішу мікропроцесорну систему, у якій на пін (INT0) встановлена кнопка (PD2), а на PB0 встановлений світлодіод. Принцип роботи наступний: натискаємо на кнопку – світлодіод деякий час горить і мікросхема переходить в сплячий режим до наступного натискання.

```
typedef unsigned char byte;
sbit ddrButton at ddD2_bit;
sbit pinButton at pinD2_bit;
sbit portButton at portD2_bit;
sbit ddrLight at ddB0_bit;  // пін для світлодіода, включається 0 по кнопці
sbit portLight at portB0_bit;
byte flagButton = 0; // прапор натискання кнопки; натиснута – 1
void INT0_interrupt() iv IVT_ADDR_INT0
/*потрібно написати порожню функцію –
  тому що компілятор сам не створює функцію. Інакше працювати не буде*/
{
  flagButton = 1;
}
//обробка натискання кнопки – з урахуванням брязкоту
void buttonLight()
{
  if(flagButton) // якщо натиснута кнопка
  {
    portLight = 0;  // вмикаємо світлодіод
    delay_ms(500);
    portLight = 1;  // вимикаємо світлодіод
    flagButton = 0;
  }
}
void main()
{
  // ініціалізація всіх використовуваних портів
  ddrB =  0;
  portB = 0;
  ddrD =  0;
```

```
    portD = 0;
    // ініціалізація кнопки – на зовнішній виклик
    portButton = 1;
    ddrButton = 0;
    //ініціалізація світлодіода, який включається по 0 і натисканню кнопки
    portLight = 1;
    ddrLight = 1;
    // налаштовуємо систему на зовнішні переривання
    GICR.INT0 = 1; // дозволяємо зовнішнє переривання INT0
    MCUCR.ISC00 = 0;  // переривання генерується по логічному 0 на INT0
    MCUCR.ISC01 = 0;
    asm sei;//SREG.B7 = 1; // дозволяємо переривання
    /*визначаємо сплячий режим: SM2–SM1–SM0 = 000 – режим Idle; 010 –
    PowerDown*/
    MCUCR.SM2 = 0;
    MCUCR.SM1 = 1;
    MCUCR.SM0 = 0;
    MCUCR.SE = 1;  // дозволяємо системі перехід в сплячий режим
    while(1)
    {
      buttonLight();
      asm sleep;  // переводимо систему в сплячий режим
    }
  }
```

Застосування такого технічного рішення значно подовжило час автономної роботи пристрою. Вдалося досягти того що в режимі Idle можна знизити споживання до 5 мА, в режимі Power-down – до 1 мкА. Розраховували за формулою (1).

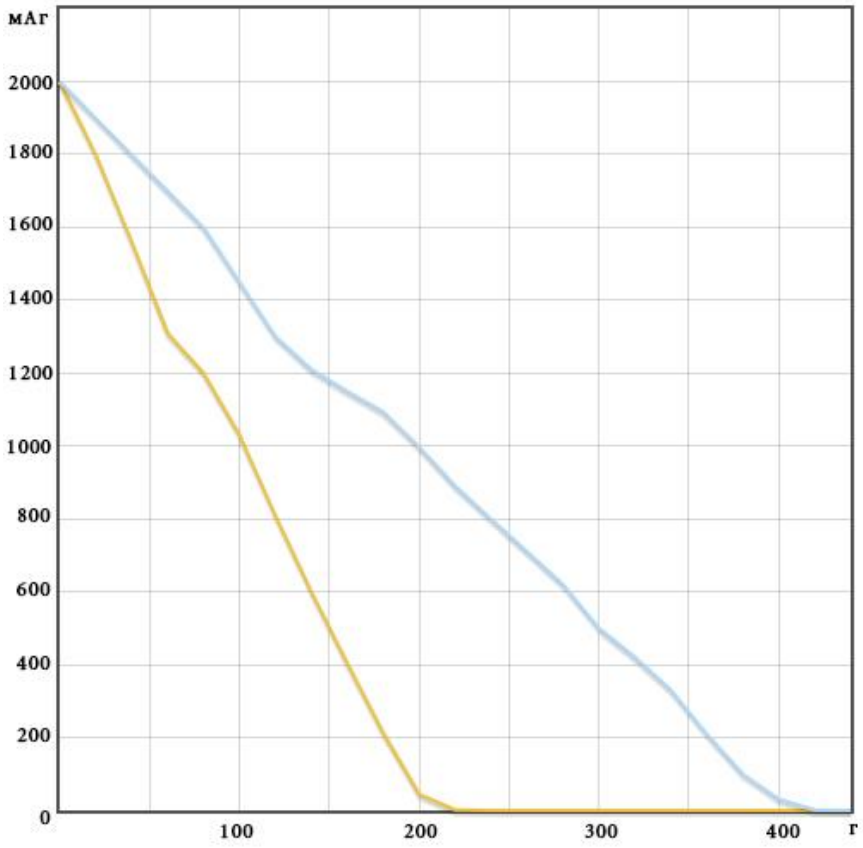

Рис. 5. **Графік роботи в штатному режимі та в режимі енергозбереження**

На графіку показано на скільки вдалося подовжити час автономної роботи пристрою завдяки розробленому технічному рішенню. Жовтим кольором позначений час роботи в стандартному режимі, синім кольором позначений час роботи з застосуванням комбінованого методу (режиму енергозбереження і технології переривань). Отже як бачимо на графіку завдяки застосуванню режиму енергозбереження та технології переривань вдалося подовжити час автономної роботи трохи більше ніж в 2 рази.

Проаналізовано перспективи переходу на більш енергоефективні режими роботи мікропроцесорних систем. Зроблено порівняння режимів роботи мікропроцесора з використанням різних елементів живлення. Визначено переваги та недоліки режимів роботи та джерел живлення. Розроблено новий спосіб збільшення часу автономної роботи мікропроцесорної системи.

Висновок

В роботі проаналізовано перспективи переходу на більш енергоефективні режими роботи мікропроцесорних систем з використанням технології переривань. Зроблено

порівняння режимів роботи мікропроцесора з використанням різних елементів живлення. Визначено переваги та недоліки режимів роботи та джерел живлення.

В результаті проведеного порівняння джерел живлення показано, що літій-іонні акумулятори більш енергоефективні за інші джерела живлення.

Розроблено новий механізм енергозбереження мікропроцесорної системи. Нове технічне рішення збільшує час автономної роботи в 2 рази. Енергозбереження досягається завдяки використовуванню технології переривань та режими сну мікропроцесора.

Розроблений механізм енергозбереження є універсальним і може бути застосований в будь-яких автоматизованих мікропроцесорних системах – в виробничих системах, системах контролю управління доступом, системах сигналізації, системах метеостанцій і т. д.

Список використаних джерел

1. Лаврус В. С. Батарейки и аккумуляторы. НиТ. / В. С. Лаврус // Публикация НиТ, 1995. – 47 с.
2. Чупин Д. П. Исследования методов диагностики аккумуляторов. / Д. П. Чупин // Омский государственный технический университет, 2012. – 5 с.
3. Вуд А. Микропроцессор в вопросах и ответах/ А. Вуд // пер. высшая школа, 1985. – 190 с.
4. Рвин Р. Язык ассемблера для процессоров Intel / Р. Рвин // Издательский дом Уильямс. – 2004. – №4. М.: Вильямс, 2005. – 905 с.
5. Кривецкий А. П. Заряд кислотных аккумуляторов / А. П. Кривецкий // Компоненты и технологии. – 2004. – №4. – С. 15-18.
6. Петин В. Arduino и Raspberry Pi в проектах Internet of Things. / В. Петин – СПб. : БХВ-Петербург, 2016. – 320 с.
7. Соммер У. Программирование микроконтроллерных плат Arduino/Freeduino / У. Соммер – СПб. : БХВ-Петербург, 2015.

References

1. Lavrus, V. (1995). *Batareyki i akkumulyatory* [Batteries and accumulators] NiT Publication of Nit [in Russian].
2. Chupin, D.P. (2012). *Issledovaniya metodov diagnostiki akkumulyatorov.* [Research methods for diagnosing batteries] Omsk State Technical University [in Russian].
3. Wood, A. (1985). *Mikroprotsessor v voprosakh i otvetakh* [Microprocessor in questions and answers] Per. high school [in Russian].
4. Rvin, R. (2004). *Yazyk assemblera dlya protsessorov Intel, chetvertoye izdaniye* [Assembler language for Intel processors, fourth edition] Williams Publishing House [in Russian].
5. Krivetsky, A. (2004). *Zaryad kislotnykh akkumulyatorov* [Acid Battery Charge] Components and technologies №4 [in Russian].
6. Petin, V. (2016). *Arduino i Raspberry PI v proektah Internet of Things* [Arduino and Raspberry PI in Internet of Things projects] St. Petersburg: BHV-Peterburg [in Russian].
7. Sommer, U. (2015). *Programmirovanie mikrokontrollernyh plat Arduino / Freeduino* [Programming of microcontroller cards

8. Офіційна документація проекту Arduino [Електронний ресурс]. Режим доступу: http://www.arduino.ru
9. Мортон Д. Микроконтроллеры AVR. Вводный курс / Д. Мортон // Журнал Мировая Электроника 2015. – 272 с.

Arduino / Freeduino]. St. Petersburg: BHV-Peterburg [in Russian].
8. *Oficijna dokumentaciya proektu Arduino* [Arduino Project Official Documentation]. Retrieved from: http://www.arduino.ru [in Russian].
9. Morton, D. (2015). *Mikrokontrollery AVR. Vvodnyy kurs* [AVR: An Introductory Course] World Electronics Magazine [in Russian].

Mosur Ivan
nothappynow8@gmail.com
Kyiv National University of Technologies and Design

Golubev Leontiy
ORCID: *https://orcid.org/0000-0002-2980-8017*
golubevl@ukr.net
Kyiv National University of Technologies and Design

Исследование режимов энергосбережения микропроцессорных систем

Мосур И. В., Голубев Л. П.
Киевский национальный университет технологий и дизайна

Цель. *Выполнить анализ режимов энергосбережения микропроцессорных систем. Проанализировать работу процессора с использованием различных элементов питания. Разработка нового технического решения для увеличения времени автономной работы микропроцессорной системы.*

Методика. *В работе использована методика технологии прерываний и режимов энергосбережения микропроцессорной системы.*

Результаты. *Проведен анализ и сравнение источников питания. Разработана технология энергосбережения для увеличения времени автономной работы микропроцессорной системы.*

Научная новизна. *Разработан новый комбинированный механизм сохранения энергии микропроцессорной системы на основе использования режимов энергосбережения микропроцессора и технологии прерываний.*

Практическая значимость. *Разработанный механизм энергосбережения является универсальным и может быть применен для увеличения времени автономной работы в различных микропроцессорных системах.*

Ключевые слова: *микропроцессорная система, процессор, режим энергосбережения, прерывание, элемент питания*

Research of energy saving modes of microprocessors systems

Mosur I. V, Golubev L. P.
Kiev National University of Technology and Design

Purpose. *Perform analysis of power saving modes of microprocessor systems. Analyze the operation of the processor using different batteries. Development of a new technical solution to increase the battery life of the microprocessor system.*

Methodology. *We used the technique of technology of interrupts and power saving modes on the microprocessor system.*

Findings. *The analysis and comparison of power sources. Developed energy-saving technology to increase time of Autonomous operation of the microprocessor system.*

Originality. *Developed a new combined mechanism of energy conservation microprocessor system based on the use of microprocessor power saving modes and interrupt technology.*

Practical value. *The mechanism of energy conservation is universal and can be applied to increase the time of Autonomous work in various microprocessor-based systems.*

Keywords: *microprocessor system, the processor, the power saving mode, interruption, battery*

