МЕТОДИ І ПРИЛАДИ КОНТРОЛЮ ТЕХНОЛОГІЧНИХ ПАРАМЕТРІВ

УДК 004.2

DOI: 10.31471/1993-9981-2021-2(47)-46-54

СХЕМОТЕХНІКА ВУЗЛІВ ДЛЯ ФОРМУВАННЯ ПОЧАТКОВОГО СТАРТУ ПРОГРАМИ І МОНІТОРИНГУ РОБОТИ МІКРОКОНТРОЛЕРА У АВТОНОМНІЙ СИСТЕМІ ОПРАЦЮВАННЯ ДАНИХ ПРИ НЕРУЙНІВНОМУ КОНТРОЛІ

Ю. Й. Стрілецький

Інститут інформаційних технологій, Івано-Франківський національний технічний університет нафти і газу; 76019, м. Івано-Франківськ, вул. Карпатська, 15; e-mail: inst.energy@gmail.com

Застосування мікропроцесорів для опрацювання дискретних даних і здійснення регулювання в системах автоматичного управління, вимірювання і контролю дозволяє реалізувати складні і оптимальні алгоритми. Водночас необхідно проводити оптимізацію схемних рішень для побудови цих систем, стійких до збоїв. Однією із задач системи є безперебійна робота при незначному втручанні оператора бо і взагалі без стороннього втручання. Зважаючи на складність повного тестування систем і вплив на мікропроцесорні засоби множини сторонніх факторів виникають ризики неправильного функціонування системи, що призводить до збоїв у опрацюванні даних. Тому вкрай важливо виявляти некоректну роботу мікропроцесорної системи і приймати рішення для виправлення ситуації, а в подальшому не допускати схожих проблем.

Розглянуто варіант схеми вузла подачі живлення на мікроконтролер для забезпечення безпечного припинення роботи програми при раптовому вимкненні напруги живлення системи. Запропоноване рішення дозволяє визначити момент зникнення живлення і дає можливість провести підготовчі операції для закінчення поточних перетворень, підготувати мікроконтролер до зупинки і зупинити роботу програми. Наведене схемне рішення для визначення стану системи перед початком роботи програми із використанням елемента пам'яті на конденсаторі, що дало можливість встановити попередню причину початкового старту програми. Розроблено принципову схему вузла зберігання восьми біт дискретних даних за межами мікроконтролера. Використання вказаної схеми дозволяє зберігати інформацію про номер поточного стану роботи програми і при початковому старті програми збільшити інформативність про його причини. За допомогою вказаного методу зберігання можна визначити при якому стані роботи програми відбувся збій, що привів до початкового старту програми і в подальшому врахувати ці відомості для усунення таких збоїв. Також розроблено схему, що дозволяє сформувати апаратний сигнал початкового старту програми мікроконтролера при приєднанні роз'єму USB. В схемі передбачено можливість заборонити формування апаратного сигналу, що потрібно під час виконання критичних ділянок програми.

Ключові слова: мікроконтролер, початковий старт, сторожовий таймер, монітор живлення, стан роботи програми.

Применение микропроцессоров для обработки дискретных данных и регулирования в системах автоматического управления, измерения и контроля позволяет реализовать сложные и оптимальные алгоритмы. В то же время, необходимо проводить оптимизацию схемных решений для построения этих систем, устойчивых к сбоям. Одной из задач системы является бесперебойная работа при незначительном вмешательстве оператора или вообще без постороннего вмешательства. Учитывая сложность полного тестирования систем и влияние на микропроцессорные средства множества посторонних факторов, возникают риски неправильного функционирования системы, что приводит к сбоям в обработке данных. Поэтому очень важно выявлять некорректную работу микропроцессорной системы и принять решения для исправления ситуации, а в дальнейшем не допускать схожих проблем.

Рассмотрен вариант схемы узла подачи питания на микроконтроллер для обеспечения безопасного прекращения работы при внезапном выключении напряжения питания системы. Предложенное решение

позволяет определить момент исчезновения питания и позволяет провести подготовительные операции для окончания текущих преобразований, подготовить микроконтроллер к остановке и остановить работу программы. Представлено схемное решение для определения состояния системы перед началом работы программы с использованием элемента памяти на конденсаторе, что позволило установить предварительную причину исходного старта программы. Разработана принципиальная схема узла хранения восьми бит дискретных данных за пределами микроконтроллера. Использование указанной схемы позволяет сохранять информацию о номере текущего состояния работы программы и при начальном старте программы увеличить информативность о его причине. С помощью даного метода хранения можно определить при каком состоянии работы программы произошел сбой, что привел к начальному старту программы и в дальнейшем учесть эти сведения для устранения причин таких сбоев. Также разработана схема, позволяющая сформировать аппаратный сигнал начального старта программы микроконтроллера при подключении разъема USB. В схеме предусмотрена возможность запрета формирования аппаратного сигнала, который требуется при выполнении критических участков программы.

Ключевые слова: микроконтроллер, начальный старт, сторожевой таймер, монитор питания, состояние работы программы.

The use of microprocessors for processing discrete data and control in automatic control, measurement and control systems allows you to implement complex and optimal algorithms. At the same time, it is necessary to optimize circuit solutions to build these fault-tolerant systems. One of the tasks of the system is uninterrupted operation with little operator intervention or without any outside interference. Due to the complexity of complete testing of systems and the impact on microprocessors of many external factors, there are risks of system malfunction, which leads to failures in data processing. Therefore, it is extremely important to detect the incorrect operation of the microprocessor system and make decisions to correct the situation, and in the future to prevent similar problems.

A variant of the circuit of the power supply unit to the microcontroller is considered to ensure the safe termination of the program when the system supply voltage is suddenly turned off. The proposed solution allows you to determine the time of power failure and allows you to perform preparatory operations to complete the current transformations, prepare the microcontroller for stop and stop the program. Was presented the circuitry solution for determining the state of the system before starting the program using a memory element on the capacitor, which made it possible to establish the preliminary cause of the initial start of the program. The schematic diagram for store eight bits of discrete data outside the microcontroller is developed. Using of this circuit allow store information about the number of the current state of the program and at the initial start of the program will increase information about causes of reset.

With using the specified storage method, able to determine the state of the program that failed, and was causing the initial start of the program, and then take this information into account to eliminate such failures. Another circuit also developed to generate the hardware signal of the initial start of the microcontroller program when the USB connector is connected. The scheme provides the ability to prohibit the forming hardware signal. That require during the execution of critical parts of the program.

Key words: microcontroller, initial start, watchdog timer, power monitor, program operation status.

Актуальність проблеми За допомогою мікропроцесорів вирішується широке коло задач. Дуже важливо, щоб вони працювали стабільно, без збоїв. Однак існує багато факторів здатних зашкодити нормальній роботі програми. Неправильна послідовність включення джерел живлення, повільне наростання напруги живлення, погана її стабілізація, стрибки напруги живлення при приєднанні зовнішнього навантаження або коли напруга живлення у навантаженні виходить за межі допустимого діапазону. Все це може призвести до збоїв в роботі мікропроцесорної системи. До того ж, незважаючи на тривалий аналіз програмного забезпечення в коді програми містяться помилки. Наявність помилок і збоїв призводить до зупинки коректної роботи мікропроцесорної системи. У разі автономної роботи така зупинка призводить до втрати даних. Тому розробка нових підходів до виявлення і усунення некоректної роботи програми в автономних мікропроцесорних системах є актуальною задачею.

Постановка задачі Вдосконалення існуючих методів підвищення стійкості роботи автономно працюючої мікропроцесорної системи досягається розробкою принципової схеми блоків для: підвищення стійкості мікропроцесорної системи до появи збоїв внаслідок різкого вимкнення живлення,

підвищення інформативності про причини виникнення збоїв у роботі програми і формування сигналу апаратного початкового старту програми.

Аналіз літературних джерел Проектувальники систем роблять все, щоб мінімізувати вплив дестабілізуючих подій. Система повинна бути надійною навіть при впливі шкідливих факторів. При цьому основна мікропроцесорна система повинна кожен раз надійно стартувати, визначати наявність помилки системи, або можливість усувати чи мінімізувати її деструктивний вплив на хід програми. Крім цього, система повинна безпечно повертатися зі стану помилки з мінімальними втручаннями оператора, або й взагалі без нього. Особливого значення безперебійна робота мікропроцесорних систем має при віддаленому опрацюванні даних. Така потреба виникає при оцінці стану технологічних об’єктів[1,2].

Навіть добре опрацьовані системи мають помилки, що виникають не тільки при коливанні напруги живлення. Поганий код програми, неправильні сигнали синхронізації, або хибні відповіді зовнішніх пристроїв можуть примусити процесор працювати не за задуманим алгоритмом, або завести його в нескінченний цикл.

Мікроконтролери мають вмонтовані засоби моніторингу роботи програми, які контролюють періодичність виконання певної функції. Якщо періодичність порушується, то відбувається системне перезавантаження роботи програми. Вмонтовані системи можуть бути суб’єктами впливу всіх видів завад, викидів енергії, просідання напруги, закорочування виводів. Все це може призвести до неочікуваних збоїв, як в схемі так і в програмі.

Наслідки збоїв можуть бути від простого руйнування даних при передачі по шині до руйнування вмісту внутрішніх регістрів центрального процесора, можуть виключитися пристрої вводу/виводу, генератор швидкості може отримати помилкове значення швидкості і т.і.

Інтегральні схеми моніторів роботоздатності мікропроцесора забезпечують дешевий і ефективний метод контролю виконання описаних дій в системі. Скидання при включенні живлення реалізується вбудованими в мікроконтролери засобами. Найширше використовується захист мікропроцесорних систем від збоїв при включенні живлення. Більшість мікроконтролерів мають вбудовані засоби скидання при включенні живлення і при виявленні повільного зменшення напруги живлення нижче допустимої межі. Надійність цих схем цілком достатня. Існують також інтегральні генератори скидання. Найбільш відомим виробником таких мікросхем є фірма MAXIM. Цією фірмою випускаються генератори скидання в корпусах SOT23 MAX6332-MAX6334 і SC70 MAX809/ MAX810/ MAX803[3]. Об'єднуючи точний монітор напруги з точною схемою селектора часу, сучасні генератори скидання визначають, коли напруга живлення знаходиться в допустимих для роботи процесора межах і забезпечують гарантовану тривалість сигналу початкового старту.

Коли системі недостатньо часу, щоб перемістити великі кількості даних в енергонезалежну пам'ять можна скористатися батарейним живленням оперативно-запам’ятовуючого пристрою (ОЗП)[4]. Щоб поєднати подвійне живлення, служать мікросхеми MAX6361/MAX6363/MAX6364. Вони забезпечують один вхід для первинного Vcc і один для бортового акумулятора.

Коли рівень живлення достатній для збереження діяльності ОЗП, внутрішній перемикач з'єднує вихід напруги супервізора з Vcc. Коли Vcc падає нижче певного порогу, супервізор забороняє подальший запис в пам'ять, і починає скидання мікропроцесора. Якщо Vcc зменшується так сильно що вміст ОЗП може зруйнуватися, супервізор включає живлення ОЗП від резервного акумулятора. Запам'ятовуючі пристрої з малим струмом в неактивному режимі, можуть зберігати дані дуже довго, поки системне живлення Vcc не повернеться до встановленого рівня.

Випускається багато мікросхем супервізорів, які поєднують в собі ще й сторожовий таймер (промисловий стандарт MAX823 і більш нові MAX6316-MAX6318), які включають сторожовий таймер як дешевий засіб забезпечення періодичності виконання коду. Система сторожового таймера має періодично скидатися. Якщо скидання не відбулося за

певний час, то ця система ініціює початковий старт програми.

Багато мікропроцесорів тепер мають внутрішній сторожовий таймер, для контролю внутрішнього стану. Оскільки сторож - тільки елемент підтримки процесора, він звичайно програмований за часом і діапазоном періодів блокування. Процесор може також змінювати функцію сторожа за допомогою програмного забезпечення.

Основна частина При вимкненні напруги живлення, навіть при існуючих системах контролю мікроконтролер знаходиться в неконтрольованому режимі. Для зменшення впливу такого режиму у разі вимкнення живлення необхідно перевести мікроконтролер в режим очікування. На закінчення поточної дії програми і виконання процедури переходу в режим очікування необхідний певний час. Якщо живлення вимкнено, то певний заряд залишається в фільтруючих конденсаторах. Подовжити час роботи мікроконтролера на цьому заряді можна при відокремленні заряду декількох таких конденсаторів від загальної частини схеми. Найбільш просто це зробити послідовним включенням діода. Спрощена схема реалізації методу безпечного вимкнення живлення наведена на рис.1.

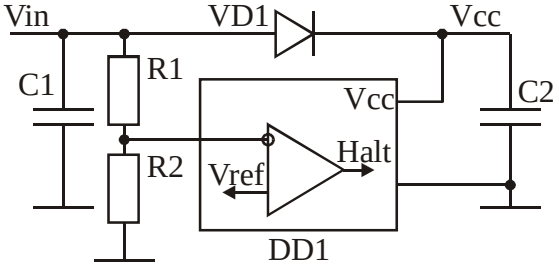

Рисунок 1- Схема затримки розрядження фільтруючих конденсаторів живлення мікроконтролера

Більшість мікроконтролерів мають вбудований аналоговий компаратор напруги із внутрішньою опорною напругою Vref. Вибравши рівень напруги Vlim при якій зниження живлення стає загрозливим для роботи мікроконтролера розраховують параметри подільника напруги R1,R2 так щоб при Vin=Vlim на виході подільника було напруга Vref.

Спрощена схема моделі роботи в режимі безпечного вимкнення представлена на рис.2.

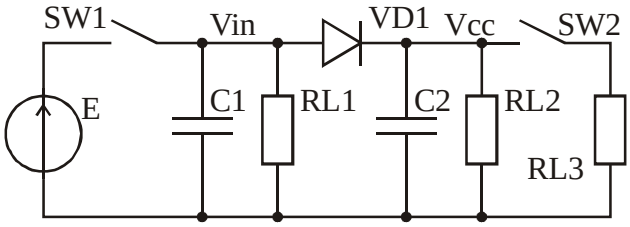

Рисунок 2- Спрощена схема моделі роботи системи при безпечному вимкненні живлення

Модель містить джерело живлення E, фільтруючі конденсатори системи C1 і фільтруючі конденсатори мікроконтролера C2, навантаження системи RL1 і навантаження пов'язане із роботою мікроконтролера в різних режимах RL2, RL3. Діод VD1 запобігає споживання струму, призначеного для роботи мікроконтролера. Ключ SW1 моделює раптове вимкнення напруги живлення, а ключ SW2 моделює різне навантаження, яке спричиняє мікроконтролер в різних режимах роботи.

Діаграма зміни напруги представлена на рис.3. При нормальній роботі системи (Normal) на вхід Vin поступає напруга живлення. Джерело живлення E в якийсь момент часу вимикається ключем SW1. Ключ SW2 при цьому залишається замкненим, імітуючи навантаження, яке створює мікроконтролер на джерело живлення в режимі нормальної роботи. За рахунок накопиченого в конденсаторі C1 заряду, напруга Vin не зникає вмить. Напруга Vin починає стрімко зменшуватися через велике споживання струму основною частиною системи через опір RL1. Однак напруга Vcc зменшується повільніше, оскільки мікроконтролер створює менше навантаження на конденсатор C2.

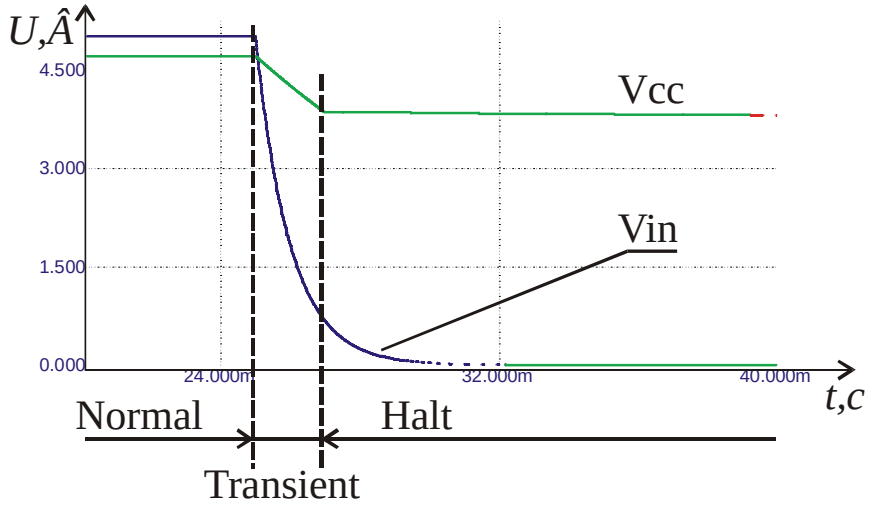

Рисунок 3 - Залежність напруги живлення мікроконтролера в часі при раптовому вимкнені живлення

Виявивши зменшення напруги Vin нижче допустимої межі, яка в даному випадку вибрана 4.5 В, відбувається перехід в режим збереження проміжних результатів роботи (Transient) і підготовка до переходу в режим зупинки роботи ядра мікроконтролера. По закінченню всіх підготовчих дій мікроконтролер переходить режим зупинки (Halt). В такому режимі мікроконтролер може безпечно вимкнутися. Споживання струму в цьому режимі зменшується. Перехід в режим супроводжується розмиканням ключа SW2. Навантаженням на C2 залишається тільки RL2. Повторний запуск відбудеться зі стану виключеної напруги живлення.

Не зважаючи на існуючі схеми і системи, що забезпечують формування початкового старту програми, необхідно проводити селекцію причин цієї події. Програма повинна оцінити причину початкового старту і внести корективи у свою подальшу роботу для зменшення ризиків втрати даних.

При цьому передбачається одна з двох причини виникнення скидання: включення системи і проблеми із виконанням програми, які призвели до спрацювання захисних механізмів системи і як наслідок до програмного скидання. Виявити причину скидання можна із використанням елементу пам'яті, який буде фіксувати наявність напруги живлення до моменту скидання.

Проблемою визначення причини початкового старту є те, що при старті стан внутрішніх елементів пам'яті мікроконтролера не контрольовані. Використання зовнішніх інтегральних запам'ятовуючих пристроїв із аварійним батарейним живленням вимагає більше затрат, а використання перезаписуваних запам'ятовуючих пристроїв має обмежений ресурс.

Найбільш простий елемент пам'яті в даному випадку є конденсатор, приєднаний до однієї із вхідних ліній мікроконтролера. Спрощена схема реалізації пам'яті стану живлення зображено на рис.4.

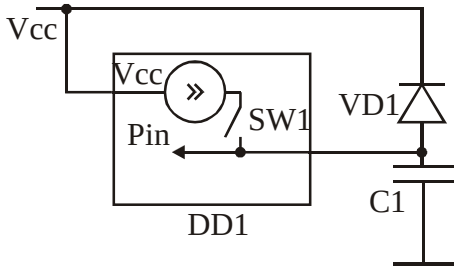

Рисунок 4 - Схема формування елемента пам'яті стану живлення на момент скидання.

Приєднаний до одного із входів вводу виводу загального призначення конденсатор C1 заряджається від внутрішнього джерела струму, який вмикається ключем SW1 після старту програми. Відповідно, коли система стартує вперше від початку включення живлення конденсатор є розряджений. Тому на вході Pin буде отримано рівень логічного нуля. Після вдалого старту ключ SW1 замикається і конденсатор за короткий час заряджається до напруги живлення. У випадку виникнення проблем в роботі програми і програмного скидання, ключ SW1 розмикається, всі змінні в

регістрах приймають початковий стан, однак заряд на конденсаторі залишається мало зміненим. Отримавши на вході Pin сигнал логічної одиниці, алгоритм опрацювання старту повинен врахувати наявність програмного скидання. Діод VD1 необхідний для швидкого розряджання конденсатора, щоб запобігти хибного висновку про причину початкового старту при проблемах з напругою живлення.

Більше інформації про початковий стан роботи системи може дати інформація записана в пам'ять самим мікроконтролером в процесі роботи. Використання для цього енергонезалежної пам’яті знову ж таки є недоцільним через недостатній ресурс перезапису. Самим простим рішенням цієї задачі є використання послідовного 8-ми розрядного регістра типу 74164. Схема включення такого регістра представлено на рис.5.

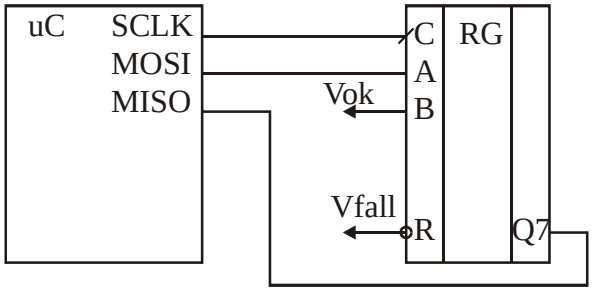

Рисунок 5 - Схема включення регістра 74164 для зберігання стану роботи програми

Таке включення регістра забезпечує зберігання 8-ми біт інформації із можливістю її зчитування впродовж 8-ми тактів на вході С. Для захисту від хибних записів в регістр використовується сигнал справності живлення Vok. При його встановленні в стан логічної 1 всі дані з входу А будуть записані в регістр і в подальшому зчитані. Також схема містить систему контролю наявності мінімальної напруги живлення. Якщо живлення знизиться нижче критичного значення вміст регістра може пошкодитися і тому варто його взагалі очистити низьким рівнем сигналу Vfall.

Таке включення регістра вимагає три лінії вводу виводу: SCLK, MISO, MOSI. Змінивши схему, як представлено на рис.6 можна скоротити цю кількість до двох.

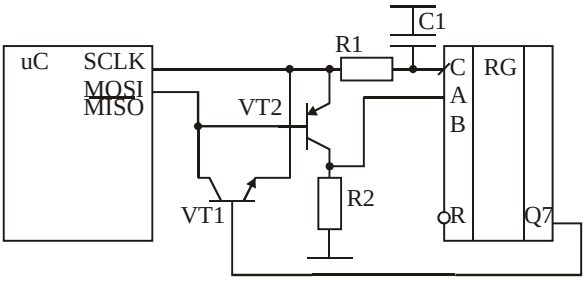

Рисунок 6 - Схема включення регістра 74164 для зберігання стану роботи програми зі зменшенням ліній інтерфейсу

Зважаючи на те, що запис в регістр проходить по передньому фронту тактового сигналу С в схемі (рис.6) реалізовано формування інформаційного сигналу А по високому рівню сигналу на виводі SCLK. Зчитування записаного в регістр через вісім тактів здійснюється з виходу Q7 при низьких рівнях сигналу SCLK. Така комутація одного сигналу на дві лінії досягається включенням транзисторів VT1,VT2. В залежності від рівня сигналу на виводі SCLK інвертором сигналу на базах в лінію колекторів стають то транзистор VT1, то транзистор VT2. Так при високому рівні SCLK, VT2 транслює інвертований сигнал MISO|MOSI на вхід А. При зміні полярності SCLK на низький рівень колектор розмикає сигнал і він встановлюється резистором R2. При цьому з виходу Q7 через VT1 сигнал поступає на вхід MISO|MOSI.

В даній схемі застосовано цифрові транзистори типу Q1- DTA113 Q2- PDTA143. На елементах R1,C1 реалізовано лінію затримки.

В програмі такий елемент пам’яті використовується для запису стану в якому знаходиться, чи в який намагається перейти програма. В разі успішного виходу з чергового стану, в пам'ять буде записано номер наступного стану, а в разі проблем, при повторному завантаженні, програма зможе оцінити, з якого стану їй не вдалося безпечно вийти.

Сучасні мікропроцесорні системи на мікроконтролерах є лише частиною системи керування чи вимірювання. З’єднання між двома системами здійснюється за допомогою інтерфейсу USB. При виникненні проблем із обміном даних із віддаленою системою

необхідно її перезавантажити. Інтерфейс USB забезпечує як передачу сигналів так і живлення на віддалений хост. Віддалені пристрої які користуються живленням лінії USB легко перезапускаються при фізичному від'єднанні і повторному приєднанні роз'єму USB.

Однак мікропроцесорна система, яка є елементом складної технологічної установки, і проводить регулювання процесом чи збір даних має власне джерело живлення. Тому просте переприєднання роз'єму може ініціалізувати тільки канал зв'язку, але не сам мікроконтролер. Особливо це стосується систем в яких обмін здійснюється через інтегральні мостові перетворювачі інтерфейсів типу CP2102.

Рішенням для початкового старту програми віддаленої мікропроцесорної системи, приєднаної до основного комп'ютера за допомогою інтерфейсу USB, є формування сигналу скидання, при появі живлення, внаслідок приєднання роз'єму USB.

На рис.7 наведена схема вузла, який дозволяє сформувати сигнал початкового старту програми при появі живлення на лінії USB.

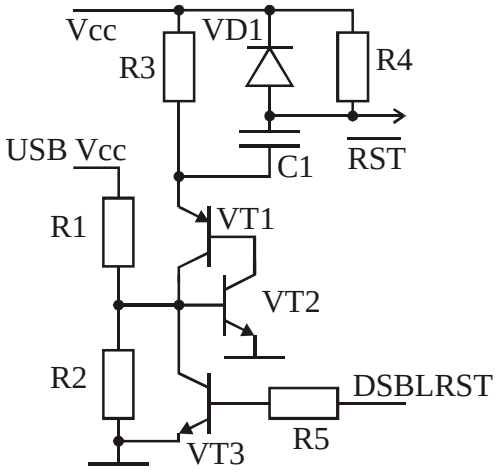

Рисунок 7- Схема формування сигналу скидання при підключенні роз'єму USB

Сигнал початкового старту $\overline{RST}$ формується при підключенні роз'єму USB, який подає і забезпечує сигнал USB Vcc. При цьому форсовано відкривається пара транзисторів VT1, VT2. Конденсатор C1, який розряджений через резистор R3 приєднується до спільної лінії і в цей момент на виході $\overline{RST}$ формується імпульс низької напруги, доки через R4 конденсатор C1 не зарядиться. Підбираючи ємність C1 можна встановити достатню для скидання мікроконтролера тривалість сигналу низького рівня. Резистори R1, R2 обмежують базовий струм транзистора VT2.

Зважаючи на особливості роботи мікроконтролера у системі управління, у схемі передбачено можливість заборони скидання при перепідключенні роз'єма USB. Заборона здійснюється подачею високого рівня на вхід сигналу DSBLRST. Сигнал заборони виставляється мікроконтролером при виконанні програми в критичні моменти, коли втрата можливості з'єднання менш важлива ніж втрата контролю над системою під час перезавантаження.

Дискусія Оскільки внутрішній сторож, працює від тієї ж напруги живлення що і тактовий генератор процесора, то він часто схильний до тих же короткочасних помилок що й процесор. Отже, найбільш стійкі системи змушені використовувати незалежного сторожа, який гарантує відповідні сигнали скидання на процесорі.

Окрім ефективних описаних рішень контролю часу виконання окремих процедур, можна контролювати перебір адрес при зверненні до зовнішньої пам'яті програм, або наявності певних специфічних сигналів на зовнішніх портах вводу/виводу. Проте цей спосіб вимагає строгої продуманості в кожному конкретному випадку і при модифікації програми з таким сторожем можуть виникнути проблеми.

Складність роботи сторожа в правильному написанні програми. Якщо код має помилку, яка призводить до виникнення безмежного циклу і щось в циклі скидає сторожа - система "повисає". Код буде ефективний, якщо програма написана з врахуванням цього небажаного явища.

Є два методи використання в програмі інтерфейсу зі схемою сторожа. Перший метод простіший. При виконанні програми прямо спілкуватися зі схемою. Другий метод передбачає використання абстрактного рівня між схемою і задачею. Саме цей рівень керує сторожем. Цей рівень зазвичай називають драйвером. Програма ставить завдання драйверу, а драйвер за всіма правилами роботи із конкретною схемою виконує це завдання.

Самим простим способом є очищення сторожа напряму. Однак при детальному розгляді він не кращий. Програма повинна вздовж всього коду вставляти очищення сторожа і при цьому може втратитися ідея самої програми. При зміні коду все повинно повторюватися спочатку. Програма може керувати механічними системами, збирати цифрові дані, проводити швидкі розрахунки. В цьому випадку найпростіше скидати сторожа, наприклад, кожних 20 ліній коду. При подальшому розвитку програми сторож практично пересичує програму.

Під дією шкідливих факторів в більшості випадків процесор виконує код, але в неправильній послідовності. Причиною збою може бути правильна послідовність дій, але із зіпсованими даними через пошкоджені входи/виходи.

Процесори з гарвардською архітектурою більш схильні виконувати неправильні інструкції[5]. При гарвардській архітектурі код програми записується здебільшого одним словом і тому практично в кожній комірці програми код правильний. Тому сторож очищатиметься як і раніше, хоч програма виконуватиметься неправильно.

Для керування сторожем за другим методом можна використати як мінімум два типи драйверів. Один з використанням базової системи переривань, другий із залученням викликаючого драйвера. Обидва способи перевіряють цілість оточення задачі системи перед скиданням сторожа. В принципі використання базової системи переривання не дуже вдала ідея, оскільки програма може збитися, а переривання через встановлений час буде викликатися і очищати сторожа.

Віртуальний сторож є простим програмним лічильником, який задача збільшує, а переривання зменшує. Якщо віртуальний сторож досягне 0, то переривання визначить що головна задача працює неправильно, забороняються переривання і вводиться визначений цикл протягом якого зовнішній сторож перезапустить процесор (тобто скидання зовнішнього сторожа припиняється і він спрацьовує).

Одиничний віртуальний сторож не ефективніший ніж просте розсипання очищення сторожа вздовж коду. Однак якщо використовується масив віртуальних сторожів можна перевіряти багато точок головної задачі. Це забезпечує більш-менш правильне спрацювання сторожа при помилковому перескакуванні лічильника програм на непередбачену послідовність. Наприклад обслуговується одна ділянка програми перед роботою якої вибирається власний номер віртуального сторожа. При перескакуванні на іншу ділянку модифікується інший віртуальний сторож а він системою переривань не обслуговується і проходить перезапуск процесора, або перехід на початок ділянки, яка була запрограмована для контролю. При багатозадачному ядрі повідомляється яка саме задача виконується неадекватно.

У випадку використання віртуального сторожа з масивом точок контролю загальна стійкість системи зростає.

На додаток до програмного сторожа драйвер можна розумно перевіряти адресу вершини стека. Частий симптом виходу процесора з під контролю є переповнення чи руйнування стеку. Хорошою практикою перед очищенням сторожа є перевірка стану стека. В залежності від специфіки конкретного процесу можна перевіряти й інші величини, наприклад статус критичних входів/виходів, контрольну суму запам’ятовуючих пристроїв.

Процесор може виконати скидання і в некерованому стані. Це може статися при послідовному наближенні програмного лічильника до місця де знаходиться обробка переривання, або при випадковому переході в результаті збою програмного лічильника на ділянку коду обробки переривання.

Для вирішення цих проблем перед процедурою обробки ставлять зациклений код. При попаданні в цей цикл програма зависає. Тоді, можливо, віртуальний сторож визначить зависання і перезапустить процесор. Для захисту від випадкового попадання всередину процедури обробки сторожа на виході з неї можна здійснити перевірку коректного початку. Перші інструкції заповнюють фіксовану ділянку пам’яті а перед очищенням “сторожа” ця ділянка перевіряється. Якщо встановлене значення не знайдене то сторож не очищається. Остання команда обробки очищає контрольну ділянку пам’яті.

Набагато простіше контролювати цілість одного куска коду, що використовується для скидання сторожа. Такий спосіб ефективніший. Однак використання віртуального сторожа ускладнює програму і тому треба ретельно стежити за розробкою такого типу захисту.

Висновки Розроблено схемне рішення вузла забезпечення безпечного завершення роботи програми при раптовому вимкненні живлення. Розроблено схему вузла запису стану програми, що дозволяє встановити стан при якому виникли причини початкового старту програми. Розроблено схему формування сигналу апаратного початкового старту програми із можливістю його апаратної заборони. Розроблені схеми в комплексі із відповідною алгоритмічною підтримкою дозволяють підвищити надійність роботи системи і забезпечити зберігання опрацьовуваних даних при роботі системи в автономних умовах.

Запропоновані схеми легко реалізуються і не вимагають додаткових коштів. При цьому вони доповнюють існуючі інтегральні рішення, призначені для формування початкового старту програми і збільшують інформативність при аналізі причин початкового старту програми.

Література

1. Striletskyi Y.Y. Using broadband signals for structural change detection in metal details / Striletskyi, Y.Y., Melnychuk, S.I., Gryga, V.M., Pashkevych, O.P. //Naukovyi Visnyk Natsionalnoho Hirnychoho Universytetu, 2020, 2020(3), pp. 19–26. DOI.ORG/10.33271/nvngu/2020-3/019

2. Striletskyy Yu.Yo. Method of determination of changes of plastic properties of a metal plate by means of frequencies of modes of the string stretched above it /Striletskyy, Yu.Yo., Rovinskyy, V.A.// Metallofizika i Noveishie Tekhnologii. 2017. 39(10). pp. 1377–1393. DOI:10.15407/mfint.39.10.1377

3. Keep the Product Working—Microprocessor Supervisors Offer Big Insurance in Small Packages / Maxim integrated// https://pdfserv.maximintegrated.com/en/an/AN720.pdf [електронний ресурс]

4. Microprocessor-based System and Supervisory Circuits / Maxim integrated//https://www.maximintegrated.com/en/design/technical-documents/app-notes/6/655.html [електронний ресурс]

5. Perrin Bob. Pedigree Protection – Watchdog Circuits, Considering the Details #1// Circuit Cellar Online. July 1. 1999.