УДК 004.324

А. Павельчак[1], В. Самотий[1,2], Ю. Яцук[1]
[1]Національний університет "Львівська політехніка",
кафедра комп'ютеризованих систем автоматики;
[2]Politechnika Krakowska, Katedra Automatyki
i Technik Informacyjnych, Polska

ОРГАНІЗАЦІЯ МУЛЬТИМАЙСТЕРНОГО РЕЖИМУ ДЛЯ I²C-ІНТЕРФЕЙСУ МІКРОКОНТРОЛЕРІВ AVR

Запропоновано структуру даних та механізми їхнього оброблення для забезпечення функціонування мультимайстерного режиму I²C-інтерфейсу мікроконтролерів AVR.
Ключові слова: I²C, AVR, мікроконтролер.

In this paper, the authors have proposed the data structure and mechanisms for their processing that ensure multi-master mode of I²C-interface of AVR microcontroller.
Key words: I²C, AVR, microcontroller.

Вступ

Сучасні вбудовані системи мають широку периферійну інфраструктуру, необхідну для реалізації різноманітних складних завдань, які ставляться перед ними. На відміну від комп'ютерних систем, вбудовані системи орієнтують під виконання спеціалізованих завдань та оптимізують з метою зменшення собівартості кінцевого продукту. При цьому до таких систем ставляться жорсткіші вимоги щодо забезпечення їхньої надійності.

Для обміну із зовнішнім світом вбудовані системи підтримують велику кількість різних інтерфейсів: 1-Wire, I²C, SPI, CAN, LIN, RS-485, RS-232, USB, Firewire тощо. Кожен з перерахованих інтерфейсів має свої переваги і недоліки і визначається областю використання. Зокрема, перші 5 інтерфейсів переважно використовуються в мікроконтролерних системах, останні 4 – для зв'язку персонального комп'ютера з периферійним обладнанням. Великої популярності для мікроконтролерних систем набув I²C-інтерфейс завдяки простоті своєї реалізації, низькій собівартості та відносно непоганій швидкодії. За допомогою цього інтерфейсу підключаються до мікроконтролерів (МК) різноманітні давачі: АЦП, ЦАП, RAM, EEPROM та ін., здійснюється інформаційний обмін між МК. Усі сучасні МК містять вбудовані модулі для I²C-інтерфейсу, що дає можливість контролювати обмін інформацією на I²C-шині, а також організувати мультимайстерний режим обміну даними. Робота з I²C-модулем мікроконтролера може бути організована як програмно (постійно перевіряти відповідний прапорець на факт завершення поточної задачі), так і в режимі переривання від I²C-модуля. У режимі апаратного переривання ми можемо використати обчислювальні потужності МК для інших поточних задач і лише час від часу перериватися, щоб керувати роботою модуля на I²C-шині. Програмний підхід частіше описується в літературі, і тому у цій статті розглянуто організацію обміну даними на I²C-шині для AVR мікроконтролерів (на прикладі ATmega32A) із реалізацією мультимайстерного режиму обміну через апаратні переривання I²C-модуля МК AVR.

1. Аналіз питання

Детальну інформацію про I²C-інтерфейс можна знайти в [1], а про особливості функціонування I²C-модуля мікроконтролера ATmega32A – у [2]. Специфікація I²C-інтерфейсу передбачає різний набір швидкостей обміну від 100 кбіт/сек до 5 Мбіт/сек з 7- і 10-бітною адресацією

пристроїв на I²C-шині (розширення можливостей інтерфейсу здійснювалися у різні роки). Найпоширеніші пристрої з підтримкою інтерфейсу для 400 кбіт/сек та 7-бітною адресацією. Нижче розглянемо реалізацію саме для цього варіанта інтерфейсу.

Обмін даними на I²C-шині здійснюється за чіткою послідовністю. За керування обміном даними на шині відповідає головний пристрій (master), хоча на шині можуть бути присутні декілька головних пристроїв, але керувати передаванням у певний момент може лише один з них. Для цього він має «захопити» шину, виконати обмін даними з необхідними пристроями на шині та «відпустити» шину. Передача даних на лінії SDA відбувається послідовно біт за бітом, згідно із поданими імпульсами на лінії SCL. Біт на лінії SDA читається у момент, коли на синхролінії SCL присутній високий рівень «1». Передавання байта даних здійснюється, починаючи зі старшого біта. Синхронізацію на шині забезпечує головний пристрій – той що проявив ініціативу для обміну даних. Будь-яка I²C-посилка складається зі старту, адресного пакета, пакета даних (рестарту, адресного пакета, пакета даних і т.д.) та стопу [1]. Усі пакети завжди є 9-бітними. На рис. 1 наведено формати типових I²C-посилок.

Рис. 1. Формати посилок для I²C-протоколу

Мультимайстерний режим передбачає, що на I²C-шині можуть одночасно бути присутніми декілька головних пристроїв (МК), які виконуватимуть обмін даними з підлеглими пристроями. Кожен з МК може бути як головним (master), так і підлеглим (slave) пристроєм. Тобто, він може адресувати інші пристрої, а може і сам бути адресованим. У мультимайстерному режимі вести передачу на I²C-шині в певний момент часу може лише один пристрій, тобто, хто перший займе шину (виставить сигнал “старт”), той і зможе обмінюватися даними з підлеглими пристроями аж до кінця своєї посилки (сигналу “стоп”). Якщо ж сигнали “старт” виставлять одночасно декілька пристроїв, тоді між ними проводиться арбітраж.

Арбітраж виконується на лінії SDA при високому рівні сигналу на SCL. Якщо пристрій формує на лінії SDA високий рівень, а водночас інший пристрій формує низький рівень, то перший втрачає право бути головним і повинен перейти у режим підлеглого. Тобто арбітраж виграє той, що виставляє на лінії SDA «0» у той момент, коли інші виставляють «1». Спершу арбітраж проводиться для байта адреси. Якщо ж конкуруючі пристрої адресують цей самий підлеглий пристрій, тоді арбітраж проводиться для наступних за адресою байтів даних – аж до сигналу “топ” чи повторного старту. Може бути ситуація, коли головні пристрої виконують однотипний обмін даними з підлеглим пристроєм. Тоді вони не зможуть виявити конфлікт на шині.

Оскільки МК AVR можуть працювати з I^{2}C-шиною як в режимі master, так і в режимі slave, то може виникнути ситуація, що вони, окрім того, що можуть втратити арбітраж, ще можуть бути в цей момент адресованими іншими МК. I^{2}C-модуль МК AVR відпрацьовує усі ці ситуації: як функції арбітражу, так і роботу в slave-режимі.

Робота I^{2}C-модуля (в AVR має назву TWI) МК AVR забезпечується п'ятьма службовими регістрами, наприклад, для ATmega32A вони мають такі назви: TWBR – регістр швидкості передачі, TWAR – регістр адреси, TWDR – регістр даних, TWCR – регістр керування I^{2}C-модулем, TWSR – регістр статусу. Як тільки у регістр TWCR заноситься одна з команд керування, то I^{2}C-модуль починає одразу її реалізовувати, а по завершенню виконання виставляє прапорець TWINT, і програма переходить на підпрограму переривання I^{2}C-модуля. У підпрограмі переривання необхідно зчитати значення статусу у регістрі TWSR для виконаної команди та вжити подальших дій для обміну даними. Модуль формує коди статусів, які перекривають усі можливі ситуації, що трапляються на I^{2}C-шині: формування команд старту, повторного старту, передачі/прийому адреси та даних з підтвердженням (ACK) чи без підтвердження (NACK), втрати пріоритету тощо.

2. Постановка задачі

Розроблювана структура даних та механізми їхнього оброблення повинні забезпечувати:

– реалізацію функціональних можливостей I^{2}C-модуля мікроконтролерів AVR для роботи у мультимайстерному режимі;
– економію процесорного часу;
– використання апаратних переривань.

3. Організація структури даних для мультимайстерного режиму

Мультимайстерний режим передбачає оброблення великої кількості задач та ситуацій, що можуть виникати при передачі на I^{2}C-шині, тому для забезпечення необхідної функціональності введемо 4 буфери: буфери для прийому і передачі, інфобуфер та буфер даних для режиму Slave (рис. 2). Перші три з них є FIFO-буферами.

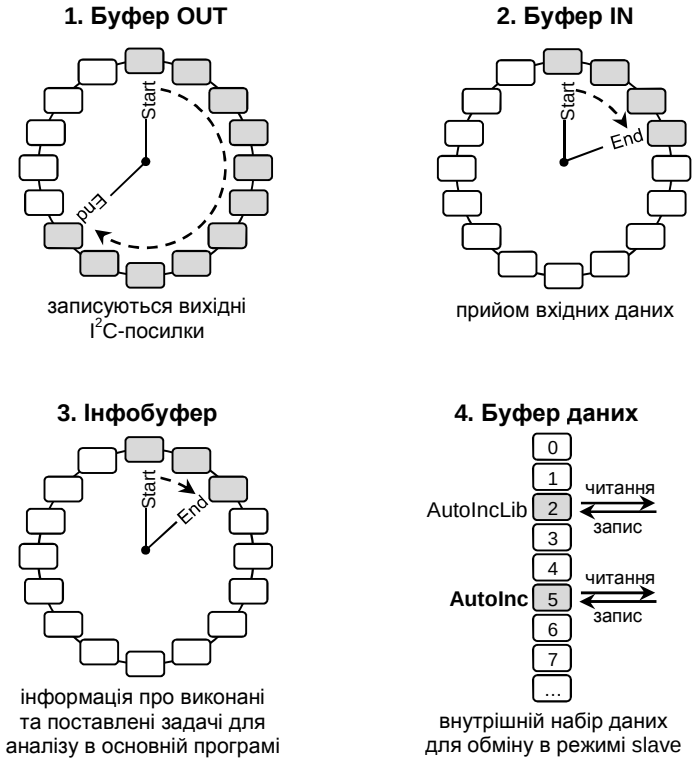

Рис. 2. Набір буферів для мультимайстерного режиму

У режимі master для забезпечення роботи з I^{2}C-шиною через апаратні переривання потрібно задіяти 2 кільцеві буфери: один на передавання, а другий – на приймання даних з I^{2}C-шини, та реалізувати для буфера передавання чітку послідовність кодів для запису у нього I^{2}C-посилки. Згідно з перериваннями I^{2}C-модуля коди з даними послідовно зчитуватимуться з буфера та відбуватиметься обмін даними з підлеглими пристроями на I^{2}C-шині.

Введемо такі псевдокоди для форматування I^{2}C-посилки у вихідному буфері OUT для I^{2}C-модуля:

0x04 – код для команди STOP;

0x10 – код для команди REPEAT START;

0x20 – код адреси (вказує, що у наступній комірці буфера буде адреса пристрою);

0x40 – код передавання байта даних (вказує, що у наступній комірці буфера буде байт даних, який потрібно відправити на I^{2}C-шину);

0x80 – код для читання байта даних з I^{2}C-шини з підтвердженням МК про прийнятий байт (ACK);

0x81 – код для читання останнього байта даних з I^{2}C-шини без підтвердження (NACK).

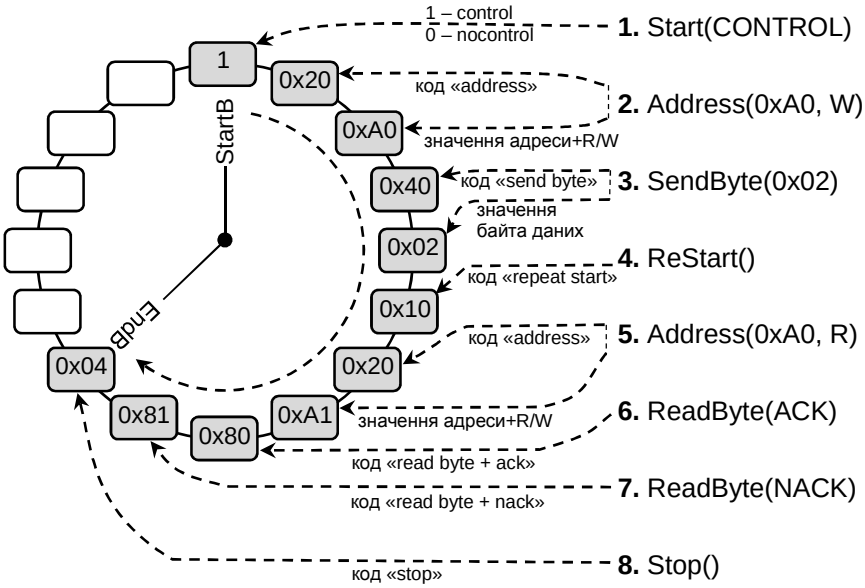

Рис. 3. Запис I^{2}C-посилки у буфер

На рис. 3 наведено приклад чіткого форматування I^{2}C-посилки та послідовність занесення псевдокодів із даними у вихідний буфер.

Окремі сегменти посилки заносяться у буфер через виклики набору функцій, які вже безпосередньо заносять псевдокоди з даними у буфер, а також забезпечують додаткову функціональність. Наприклад, при виклику функції Start(), яка формує команду початку сеансу на I^{2}C-шині, ми маємо передати у неї параметр CONTROL чи NOCONTROL, який вказує, чи потрібно після завершення передачі I^{2}C-посилки звітувати про виконання. Також функція Start() має здійснювати заборону на глобальні переривання, яка знімається аж у кінці форматування посилки функцією Stop(). Такий крок необхідний для того, щоб передача посилки не здійснювалася поки не буде повністю занесена у буфер.

Перед кожним байтом адреси чи даних завжди слідує відповідний псевдокод, за яким можна розпізнати, що міститься у наступній комірці. Формально передавання як адреси, так і даних є однотипним, але окремі коди дають можливість відслідковувати під час відлагодження, що ми передаємо.

Для читання даних введено псевдокод 0x80 з 0 чи 1 у молодшому розряді (якщо 1, тоді код буде 0x81). Якщо 0, то МК підтверджує прийняття байта і очікує надходження наступного. Якщо 1, тоді МК виставляє на шину NACK та завершує приймання даних.

У вихідному буфері OUT може бути одночасно записано декілька I^2C-посилок. Запуск першої посилки (сигнал START) виконує функція Stop(), а наступні у буфері за нею посилки виконуються вже згідно із підпрограмою переривання I^2C-модуля.

Якщо I^2C-модуль повинен прозвітувати про виконану посилку, тоді введену нами змінну I2Ctask буде збільшено на 1 (на початку вона дорівнює 0). Це означає, що у вхідному буфері IN є прийняті дані, які програма має зчитати. Після цього програма повинна буде зменшити змінну I2Ctask назад на 1.

Для контролю за процесом виконання I^2C-посилок необхідно кожній задачі присвоювати індивідуальні номери. Номер задачі має присвоюватися посилці під час її формування, а саме: при виклику функції Start(). Оскільки молодший біт значення, що передається цій функції, відповідає за контроль за виконанням посилки, то, відповідно, адреса може займати лише старші 7 бітів. Для цього найкраще передбачити ряд відповідних макровизначень, що спрощуватимуть нумерацію посилок.

Виконані задачі записуються в інфобуфер. У режимі master формат даних містить 2 байти: номер виконаної задачі та кількість прийнятих байтів (для контролю). У режимі slave записують лише по 1 байту (виконана задача прийнятого командного чи звичайного байта). Отримані байти вже безпосередньо зчитуються з буфера IN у порядку запису виконаних задач в інфобуфері. Якщо командний байт містить адресу в буфері даних, тоді інформація про такі задачі не заноситься в інфобуфер, а отримані байти даних записуються безпосередньо вже у буфер даних підпрограмою переривання I^2C-модуля. Аналізують виконані задачі в основній програмі за допомогою розгалуженого умовного блоку.

У процесі арбітражу через втрату пріоритету виконання поточної I^2C-посилки може бути перерване. Тому необхідно постійно тримати у тимчасовій змінній розміщення початку в буфері OUT поточної посилки і за необхідності відновлювати.

Від МК у режимі slave зовнішній пристрій може вимагати здійснення таких трьох дій:

– виконати певну задачу (ввімкнути світло, запустити процес оцифрування даних модуля АЦП і т.п.);

– прийняти байт даних;

– вислати байт даних з певною інформацією (наприклад, отримані результати від модуля АЦП).

Для цього на зразок периферійних пристроїв для режиму slave виділяємо окремий буфер даних (рис. 2), у якому за кожною коміркою буде закріплено певну величину. Наприклад, сюди можуть записуватися оцифровані значення з АЦП і за потреби зчитуватися зовнішнім пристроєм. Можуть міститися конфігураційні комірки для функціонування МК, значення яких змінюватимуться іншими пристроями тощо. Читання та запис даних здійснюється за вказаною адресою розміщення даних у буфері. Адреса при цьому автоматично інкрементується, щоб можна було виконувати операції з послідовними даними, наприклад, зчитати I^2C-посилкою одразу декілька байтів. Тобто, МК у режимі slave функціонує за правилами, що визначені I^2C-протоколом. Для поділу доступу до даних підпрограмою переривання I^2C-модуля та основною програмою ми передбачили дві змінні: I2C_AutoincBUFdata (осн. програма) та I2C_AutoincBUFdataLib (підпрограма переривання).

Для розробленої функціональної структури, що є обгорткою для I^2C-модуля МК AVR, було написано програмну бібліотеку мовою Сі. Усі необхідні файли проекту можна знайти на сайті автора http://pavelchak.at.ua.

Тестування та відлагодження розробленої бібліотеки під час роботи з I^2C-шиною у мультимайстерному режимі для функцій арбітражу виконували безпосередньо на фізичному макеті. Для цього до I^2C-шини було під'єднано два МК ATmega32A та мікросхему пам'яті EEPROM M24C08.

Код статусу 0x38 (втрата пріоритету в режимі master). Для цього тесту функцій арбітражу МК повинні одночасно виконувати звертання до пам'яті EEPROM. Одна кнопка є під'єднаною одразу до двох МК, а її натиск ініціалізує запуск процедури запису в пам'ять EEPROM. Кожен з МК звертається до своєї області пам'яті в EEPROM. Очікування натиску кнопки є зациклене у конструкції while{}. Як тільки кнопку буде натиснуто, мікроконтролери одночасно почнуть змагатися за доступ до пам'яті EEPROM. Хто перший видасть на лінію «0», той і отримає пріоритет для доступу, а інший буде очікувати звільнення шини. Зауважимо, що оскільки запис у пам'ять EEPROM триває деякий час (5–10 мсек.), то мікросхема відповідатиме на свою адресу лише після циклу запису, а до цього на лінії буде NACK.

0:	Test
1:	Test
2:	Test
3:	Switch
4:	External
5:	Second
6:	Minute
7:	Hour
…	

Рис.4. Структура буфера даних

Коди статусів 0x68, 0x78, 0xB0 (втрата пріоритету та перехід у режим роботи slave). Ці коди сигналізують, що МК при спробі вийти на I^2C-шину з передаванням даних був перерваний іншим МК для його адресації та обміну з ним даними. Для тестування цього випадку арбітражу до I^2C-шини було під'єднано ще мікросхему годинника реального часу PCF8583. Тепер обидва мікроконтролери матимуть можливість паралельно працювати з годинником та пам'яттю EEPROM, а також обмінюватися даними між собою. У цьому випадку ми реалізуємо 2 ситуації обміну даними МК між собою по I^2C-шині:

– відправлення першим МК пакета даних з 4-х байтів (власна адреса, секунди, хвилини, години) другому МК для їхнього запису в буфер даних;

– читання першим МК тестових значень та значення з перемикача «піаніно» з буфера даних другого МК.

Для обміну даними між МК ми розмітили простір буферів даних, як на рис. 4.

Програмний код основної програми було розроблено так, щоб його можна було використати для обох МК одночасно. Відмінність полягає лише у тому, що для одного з них макровизначення Master набуває значення 0, а для іншого 1. Так для них забезпечуються I^2C-адреси 0x78 та 0x79.

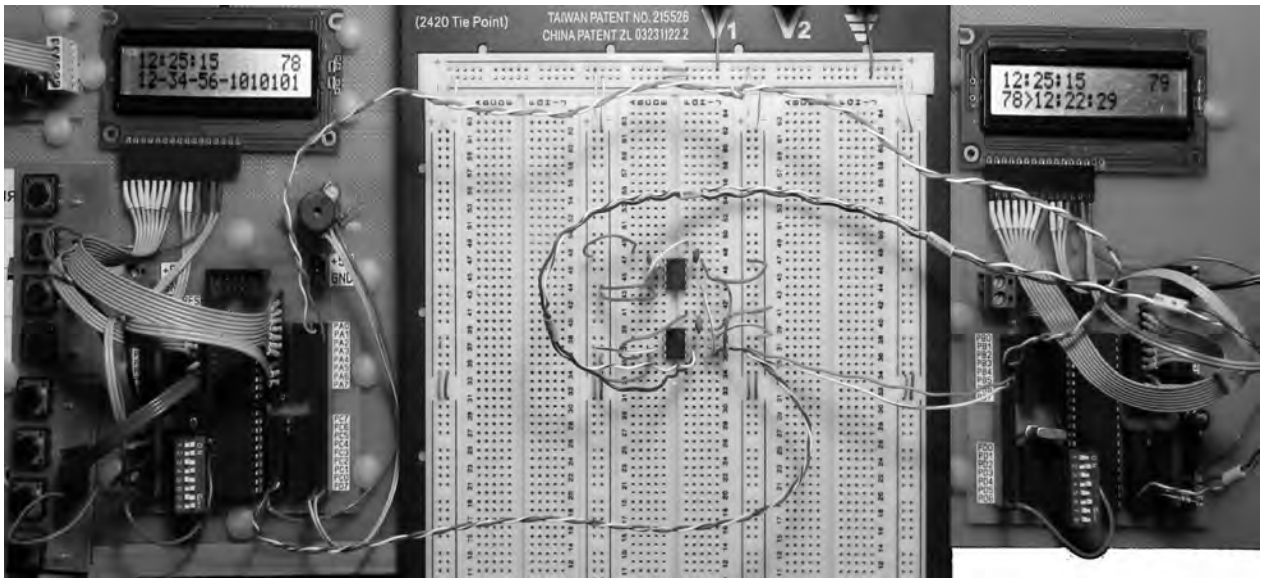

Рис. 5. Апробація дослідження роботи I^2C-інтерфейсу в мультимайстерному режимі

Для цього прикладу було проведено експериментальну апробацію роботи I^2C-інтерфейсу в мультимайстерному режимі на фізичному обладнанні (рис. 5). Також протестовано функції арбітражу, що відповідають кодам статусу I^2C-модуля 0x68, 0x78, 0xB0 (втрата пріоритету та перехід у режим роботи slave) за одночасної посилки даних МК один одному. Для цього було об'єднано однакові виводи PB6 та PB7. На макеті усі функції арбітражу були відпрацьовані коректно.

Висновок

Згідно із аналізом роботи I^2C-інтерфейсу в мультимайстерному режимі запропоновано структуру даних і механізми для їхнього оброблення, що забезпечують роботу з I^2C-шиною з використанням апаратних переривань. Розроблена структура дає можливість повною мірою реалізувати усі функції арбітражу для I^2C-шини. Усі отримані результати експериментально апробовано на фізичному обладнанні.

1. UM10204. I^2C-bus specification and user manual. – NXP Semiconductors. – 2012. – 64 p. 2. ATmega32A. Datasheets – Atmel Corporation. – 353 p. 3. PCF8583. Clock and calendar with 240 x 8-bit RAM. Datasheets – NXP Semiconductors. – 2010. – 37 p. 4. AVR315: Using the TWI module as I^2C master. Application Note – Atmel Corporation. – 14 p.