Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ
КАФЕДРА «ПРОМИСЛОВОЇ ЕЛЕКТРОНІКИ»

РОЗПОДІЛЕНІ МІКРОПРОЦЕСОРНІ СИСТЕМИ

КОНСПЕКТ ЛЕКЦІЙ

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для аспірантів,
які навчаються за спеціальністю 171 «Електроніка»,
спеціалізацією «Електронні компоненти і системи»

Київ 2018

Розподілені мікропроцесорні системи: конспект лекцій [Електронний ресурс]: для підготовки докторів філософії в галузі знань 17 Електроніка та телекомунікація за спеціальністю 171 Електроніка за спеціалізацією «Електронні системи» / КПІ ім. Ігоря Сікорського ; уклад.: Т. О. Терещенко – Електронні текстові данні (1 файл:5544 кбайт). – Київ : КПІ ім. Ігоря Сікорського, 2018. – 192 с.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 9 від 24.05.2018 р) за поданням Вченої ради факультету електроніки (протокол № 04/2018 від 23.04.2018 р.)

Електронне мережне навчальне видання

РОЗПОДІЛЕНІ МІКРОПРОЦЕСОРНІ СИСТЕМИ

Конспект лекцій для аспірантів напрямку підготовки 171

Укладачі: *Терещенко Тетяна Олександрівна,* докт. техн. наук.

Відповідальний редактор *Ямненко Ю. С.,* завідувач кафедри промислової електроніки, д-р техн. наук, проф.

Рецензенти: *Михайлов С.Р.,* доцент кафедри електронних приладів та пристроїв, канд. техн. наук, доц.

Метою посібника «Розподілені мікропроцесорні системи: конспект лекцій» є висвітлення основних принципів побудови розподілених мікропроцесорних систем та сприяння набуттю практичних навичок побудови апаратурної частини та програмного забезпечення таких систем

Київ 2018

ЗМІСТ

ВСТУП

Курс "Розподілені мікропроцесорні систем" є компонентом циклу загальної підготовки професійної підготовки фахівців рівня «Доктор філософії» в галузі знань 17 «Електроніка та телекомунікація» за спеціальністю 171 «Електроніка» за спеціалізацією «Електронні системи» за денною формою навчання і відноситься до навчальних дисциплін для здобуття глибинних знань зі спеціальності.

Курс базується на дисциплінах підготовки магістрів: «Мікропроцесорні системи» та «Сучасні напрямки комп'ютерної і мікропроцесорної техніки».

Мета курсу – здатність використовувати основні архітектурно-структурні ідеї проектування мультимікропроцесорних систем на базі сучасних інтерфейсів та отримання практичних навичок побудови таких систем.

Розділ 1 . РОЗПОДІЛЕНІ МУЛЬТИМІКРОКОНТРОЛЕРНІ СИСТЕМИ

ЛЕКЦІЯ 1.

Області застосування та принципи побудови розподілених мікроконтролерних систем. Мультимікропроцесорна система з інтерфейсом UART

На сьогоднішній день розподілені мікроконтролерні системи керування займають провідне положення в сфері автоматизації процесів, починаючи від великих виробництв і закінчуючи автоматизацією будівель.

Інтелектуальні вузли розподіленої мікроконтролерної мережі (fieldbus-системи або промислової мережі) вирішують задачі обробки даних з датчиків і видачу керуючих впливів на керуючі пристрої. Розподілена система дозволяє замінити один високої вартості високопродуктивний процесор на кілька менш продуктивних і дешевих процесорів.

Основні переваги при використанні розподілених мікроконтролерних систем у порівнянні із централізованими системами полягають у меншій вартості установки розподіленої системи, її супроводженні й модернізації, більшої надійності, гнучкості й здатності до інтеграції. Однак, наявність послідовної шини передачі даних, як правило, накладає обмеження на топологію системи і її швидкодію.

Протягом останніх п'яти років в розвитку архітектури мікросхем мікроконтролерів провідних світових виробників спостерігається стійка тенденція зростання кількості вбудовуваних інтерфейсів. На відміну від мікросхем мікроконтролерів попередніх поколінь, з мінімумом інтерфейсів і цифрової периферії (як правило, інтерфейс послідовного порту UART, декілька таймерів і підсистема переривань), сучасні мікроконтролери оснащуються чималим набором додаткових цифрових і аналогових підсистем. До складу цифрових підсистем входять: чимала Flash пам'ять

програм (даних), що має розмір від восьми до декількох сотень кілобайт; оперативна пам'ять з об'ємом до декількох десятків кілобайт; розвинена система переривань, оброблювальна до тридцяти джерел переривань; спеціалізовані («сторожеві») таймери WDT (Watchdog Times); різні типи генераторів і велика кількість інтерфейсів.

Аналіз оснащеності сучасних мікроконтролерів провідних світових фірм Silicon Laboratories Corp., Atmel Corp., MAXIM-Dallas Corp.дозволяє стверджувати, що серед сучасних мікроконтроллерів 100% оснащені одним і приблизно 25% - двома послідовними портами UART. Приблизно 85% мікроконтролерів мають в своєму складі послідовний периферійний інтерфейс SPI (Serial Peripheral Interface). Приблизно 40% оснащуються інтерфейсом системної шини SMBus, сумісним з I2C. Менше 20% мікроконтролерів оснащуються всіма іншими типами інтерфейсів, такими як: універсальна послідовна шина - USB (Universal Serial Bus); інтерфейс мережі мікроконтролерів - CAN (Controller Area Network), однопровідний двонаправлений інтерфейс MICROLAN і ряд інших.

При цьому слід зазначити, що якщо ще кілька років тому практично кожен з мікроконтролерів оснащувався інтерфейсом UART, то в даний час цей інтерфейс упевнено втрачає свої лідируючі (у минулому) позиції. Для зв'язку з персональним комп'ютером все частіше використовується універсальна послідовна шина – USB, оскільки має вищі швидкості передачі і забезпечує цілий ряд інших додаткових функціональних можливостей. Для організації мікроконтролерних мереж використовуються інтерфейси: CAN, SPI, SMBus і MICROLAN. Для внутрісистемних (внутрішньоплатних) зв'язків в сучасних контроллерах найчастіше використовуються інтерфейси SPI і SMBus.

Розрізняють кілька рівнів моделей розподіленних систем.

На першому рівні моделі застосовуються інтерфейси:

- повнодуплексний послідовний інтерфейс RS-232, RS-485;

- послідовний інтерфейс USB (Universal Serial Bus);
- універсальний асинхронний приймач-передавач UART (Universal Asynchronic Receiver Transmitter);
- синхронний послідовний інтерфейс SPI (Serial Peripheral Interface);
- інтерфейс мережі мікроконтролерів інтерфейс системної керуючої шини SMBus, сумісним з I2C;
- CAN (Controller Area Network);
- однопровідний двоспрямований інтерфейс MicroLAN;
- двопровідний інтерфейс TWI (Two wire Interface);
- однопровідний інтерфейс OWI (One wire Interface);
- безпровідний інтерфейс ZigBee.

На другому рівні моделі застосовують протоколи обміну, що відрізняються призначенням і потужністю. Всі протоколи fieldbus-систем можна згрупувати за видом мережного арбітражу, а саме:

- принцип опитування;
- принцип;
- принцип випадковому доступу.

Принцип опитування використовуються в тих випадках, коли в мережі присутній один головний пристрій і кілька підлеглих, що не мають "права голосу". Прикладом такої системи є мережа Modbus

У мережах з передачею маркера (мережі P-NET, Profibus) правом заняття шини володіє пристрій з "маркером", що передається від одного пристрою до іншого.

При випадковому принципу доступу (мережі LonWorks, EIB, Ethernet) пристрої прослуховують мережу й контролюють свою передачу на предмет виявлення спотворення інформації з метою контролю колізій. У випадку колізії, вузол мережі відповідно до певного алгоритму (існує кілька варіантів) вибирає випадковим способом час повторного відправлення повідомлення.

Відмінною рисою перших двох методів арбітражу є можливість гарантування часу відповіді, що дозволяє їх віднести до систем реального часу. Третій метод дозволяє гарантувати час відповіді лише з певною ймовірністю, що залежить від кількості пристроїв в одному сегменті мережі.

Мультимікропроцесорна система з інтерфейсом UART
Универсальный асинхронный приемопередатчик (UART)

Універсальний асинхронний приймач-передавач UART (Universal Asynchronic Receiver Transmitter) призначений для забезпечення послідовного обміну даними. Інтерфейс UART являє собою повнодуплексний інтерфейс, тобто приймач і передавач можуть працювати одночасно, незалежно друг від друга.

До складу UART входять:

- тактовий генератор зв'язку (бодрейт-генератор),
- керуючі регістри,
- статусні регістри,
- буфери
- регістри зсуву приймача й передавача.

Бодрейт-генератор задає тактову частоту прийомопередавача для даної швидкості зв'язку.

Керуючі регістри задають режим роботи послідовного порту і його переривань.

У статусному регістрі встановлюються прапори за різними подіями.

У буфер приймача потрапляє прийнятий символ, у буфер передавача поміщають переданий.

Регістр зсуву передавача видає біти переданого символу (кадру). Регістр зсуву приймача накопичує прийняті з порту біти. По різних подіях встановлюються прапори й генеруються переривання (завершення прийому/відправлення кадру, звільнення буфера, різні помилки).

Лінію порту приймача позначають RX, передавача - TX. Послідовною установкою рівнів на цих лініях щодо загального проводу ("землі") і передається інформація. За замовчуванням передавач встановлює на лінії одиничний рівень. Передача починається посилкою біта з нульовим рівнем (старт-біта), потім йдуть біти даних молодшим бітом уперед (низький рівень - "0", високий рівень - "1"), завершується посилка передачею одного або двох бітів з одиничним рівнем (стоп-бітів).

Електричний сигнал кадру посилки виглядає так:

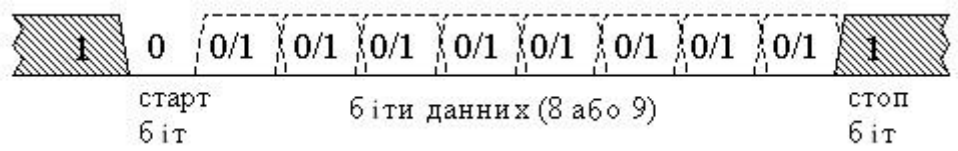


Рис. 1.1. Кадр посилки UART

Перед початком зв'язку між двома пристроями необхідно настроїти їх прийомопередавачи на однакову швидкість зв'язку й формат кадру.

Швидкість зв'язку (baudrate) виміряється в бодах як число переданих бітів у секунду (включаючи старт і стоп-биті). Задається ця швидкість у бодрейт-генераторі діленням системної частоти на заданий коефіцієнт. Типовий діапазон швидкостей: 2400 ... 115200 бод.

Формат кадру визначає число стоп-бітів (1 або 2), число бітів даних (8 або 9), а також призначення дев'ятого біта даних. Все це залежить від типу контролера.

Приймач і передавач тактується, як правило, з 16-кратною частотою відносно бодрейта. Це потрібно для семплірування сигналу. Приймач, визнавши наявність падаючого фронту старт-біта, відраховує кілька тактів і наступні три такти зчитує (семплірує) порт RX. Це саме середина старт-бита. Якщо більшість значень семплів - "0", старт-біт вважається прийнятим, інакше приймач приймає його за шум і чекає наступного падаючого фронту.

Після вдалого визначення старт-біта, приймач так само семплірує середини бітів даних і по більшості семплів визначає біт "0" або "1", записуючи їх у регістр зсуву. Стоп-біти теж семпліруються, і якщо рівень стоп-біту не "1" - UART визначає помилку кадру й установлює відповідний прапор у керуючому регістрі.

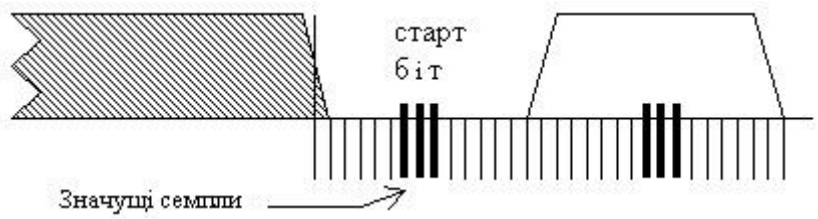


Рис. 1.2. Визначення бітів

Оскільки бодрейт установлюється діленням системної частоти, при перенесенні програми на пристрій з іншим кварцовим резонатором, необхідно змінити відповідні настроювання UART.

В якості приклада розглянемо послідовний інтерфейс мікроконтролера MCS-51, структурна схема якого наведена на рис. 1.3. Схема містить зсувний регістр прийому; зсувний регістр передачі; буферний регістр прийому-передачі SBUF; схеми управління та обробки зовнішніх сигналів управління.

Рис. 1.3 Структура послідовного порту

Запис байту у буфер SBUF програмним шляхом, наприклад MOV SBUF, A. призводить до автоматичного переписування байта в зсувний регістр передачі та ініціює початок передачі байта. Наявність буферного регістра прийому дозволяє поєднувати операцію пересилання раніше прийнятого байта у внутрішню пам'ять з прийомом наступного байта. Якщо до моменту закінчення прийому байта попередня інформація не була зчитана з SBUF, вона втрачається.

При закінченні передачі байту формується сигнал переривання TI основної програми мікроконтролера. При заповненні регістра приймача формується сигнал переривання RI. Ці прапори апаратно не скидаються відповідною програмою переривань, необхідно використовувати програмні засоби.

Управління роботою UART здійснюється за допомогою регістра SCON (Serial port CONtrol), який має побітову адресацію. Призначення бітів регістра SCON наведено у табл. 1.1.

Таблиця 1.1 – Призначення бітів регістра SCON

Біти	Позначення	Призначення
1	2	3
7	**SM0**	Визначають один з 4-х режимів роботи послідовного порту: SM0 SM1 0 0 – режим “0”; 0 1 – режим “1”; 1 0 – режим “2”; 1 1 – режим “3”.
6	**SM1**	
5	**SM2**	Дозвіл мультипроцесорної роботи: у режимі “0” SM2 повинен бути скинутий в нуль; у режимі “1” при SM2 = 1 прапорець RI установлюється в одиницю при надходженні стоп-біта, що дорівнює одиниці; у режимах “2” і “3” при SM2 = 1 прапорець RI дорівнює нулю, якщо дев’ятий біт прийнятих даних дорівнює нулю.
5	**SM2**	Дозвіл мультипроцесорної роботи: у режимі “0” SM2 повинен бути скинутий в нуль; у режимі “1” при SM2 = 1 прапорець RI установлюється в одиницю при надходженні стоп-біта, що дорівнює одиниці; у режимах “2” і “3” при SM2 = 1 прапорець RI дорівнює нулю, якщо дев’ятий біт прийнятих даних дорівнює нулю.

4	**REN**	Дозвіл прийняття послідовних даних. Установлюється і скидається програмно відповідно для дозволу та заборони прийняття даних
3	**TB8**	Дев'ятий біт переданих даних у режимах "2" і "3". Встановлюється і скидається програмно
2	**RB8**	У режимі "0" RB8 не використовується; у режимі "1" при SM2 = 0 – прийнятий стоп-біт; у режимах "2" і "3" – дев'ятий біт прийнятих даних
1	**TI**	Прапорець переривання передавача. У режимі "0" – установлюється апаратно при видачі восьмого біта; у інших режимах – установлюється апаратно при формуванні стоп-біта. Скидається програмно в усіх режимах
0	**RI**	Прапорець переривання приймача. При SM2 = 0: установлюється апаратно при прийнятті восьмого біта в режимі "0" або через половину інтервалу стоп-біта в режимах "1", "2", "3" При SM2 = 1: в режимі "1" RI не встановлюється, якщо не прийнятий стоп-біт; дорівнює одиниці у режимах "2" та "3"; RI не встановлюється, якщо дев'ятий прийнятий біт даних дорівнює одиниці. Скидається програмно в усіх режимах

Після вибору режиму робота інтерфейса наступна:

Режим 0. Інформація передається і приймається через вхід RXD приймача (вивід P3.0). Через вихід передавача TXD (вивід P3.1) видаються імпульси синхронізації, що стробують кожен передаваний або прийманий біт інформації. Формат посилки – 8 біт. Частота прийому і передачі дорівнює тактовій частоті мікроконтролера.

Режим 1. Інформація передається через вихід передавача TXD, а приймається через вхід приймача RXD. Формат посилки – 10 біт: старт-біт (нуль), вісім біт даних, програмований дев'ятий біт і стоп-біт (одиниця). Частота прийому і передачі задається таймером/лічильником 1.

Режим 2. Інформація передається через вихід передавача TXD, а приймається через вхід приймача RXD. Формат посилки – 11 біт: старт-біт (нуль), вісім біт даних, програмований дев'ятий біт і 2 стоп-біта (одиниці). Передаваний дев'ятий біт даних набуває значення біта TB8 з регістра спеціальних функцій SCON. Біт TB8 в регістрі SCON може бути програмно

встановлений в «0» або в «1», або в нього, наприклад, можна помістити значення біта P з регістра PSW для підвищення достовірності інформації, що приймається (контроль за паритетом). При прийомі дев'ятий біт даних прийнятої посилки поступає в біт RB8 регістра SCON. Частота прийому і передачі в режимі 2 задається програмно і може дорівнювати тактовій частоті мікроконтроллера діленою на 32 або на 64.

Режим 3. Режим 3 повністю ідентичний режиму 2 за винятком частоти прийому і передачі, яка в режимі 3 задається таймером/лічильником 1.

Швидкість приймання-передавання даних у послідовному порту залежить від режиму роботи. У режимі “0” частота передавання залежить лише від резонансної частоти кварцового резонатора $f_0 = f_Q/12$. За один машинний цикл послідовний порт передає 1 біт інформації.

У режимах 1, 2 та 3 швидкість приймання-передавання залежить від значення керуючого біта *SMOD* у регістрі *SCON*. У режимі “2” частота передавання визначається формулою $f_2 = (2^{SMOD}/64)\cdot f_Q$. Інакше кажучи, при SMOD=0 частота передавання дорівнює $(1/64)f_Q$, а при $SMOD = 1$ дорівнює $(1/32)f_Q$.

У режимах “1” та “3” у формуванні частоти передавання, крім біта *SMOD*, приймає участь *таймер-лічильник 1*. При цьому частота передавання залежить від частоти переповнення (*OVT*1) і визначається таким чином: $f_{1,3} = (2^{SMOD}/32)\cdot f_{out}$. Переривання від таймера-лічильника 1 в цьому випадку повинно бути заблокованим. Сам таймер-лічильник 1 може працювати і як таймер, і як лічильник подій у будь-якому з трьох режимів. Однак найбільш зручно використовувати режим таймера з автоперевантаженням (старша тетрада $TMOD = 0010B$). При цьому частота передавання визначається так:

$$f_{1,3} = (2^{SMOD}/32)\cdot(f_Q/12)/(256-(TH1)).$$

Приклад системи з інтерфейсами RS232 і UART

Система освітлення студій телеканалу “Інтер” призначення для регулювання рівня освітлення в двох студіях телеканалу – новин та передач. Система (рис 1.4) містить некерований трифазний випрямляч В , 64 каналів регуляторів електроживлення галогенових ламп та мультипроцесорну систему кеування. Кожний канал містить ШІП (ШІП1-ШІП64), що працює з частотою 5 кГц, і мікроконтролерну систему керування МК1-МК64.

Використання постійної напруги для живлення з низьким рівнем пульсацій дозволяє підвищити якість освітлення при телезйомках.

Загальне керування каналами здійснює мультимікропроцесорна система з трьома рівнями ієрархії. На першому рівні знаходиться персональні електронно-обчислювальні машини двох телестудій - ПК1 та ПК2, на другому - мікропроцесорна система керування (МСК) на базі однокристальної мікроЕОМ фірми ATMEL AT89C2051, на третьому - мікроконтролери окремих каналов типу AT89C1051.

ПК1 та ПК2 дозволяють з двох різних приміщень керувати конфігурацією та рівнем освітлення окремих ламп або встановлювати розроблені раніш сценарії освітлення. При одночасній роботи пріоритет має ПК1. Мікропроцесорна система керування передає на мікроконтролери окремих каналів сигнали керуючого впливу у вигляді значення скважності імпульсів. Технічні характеристики системи наведено в табл. 1.2

Таблиця 1.2 – Технічні характеристики

Технічні характеристики	Числове значення
Кількість каналів:	64
Діапазон регулювання вихідної напруги канала	60-220 В
Вихідний струм (Івих при Uвих=220 В)	1-5 А
Потужність Pmax каналу	1 кВт
Нестабільність вихідної напруги канала	<3%
Час виходу на режим	<3 с

Кількість программ освітлення	>=10

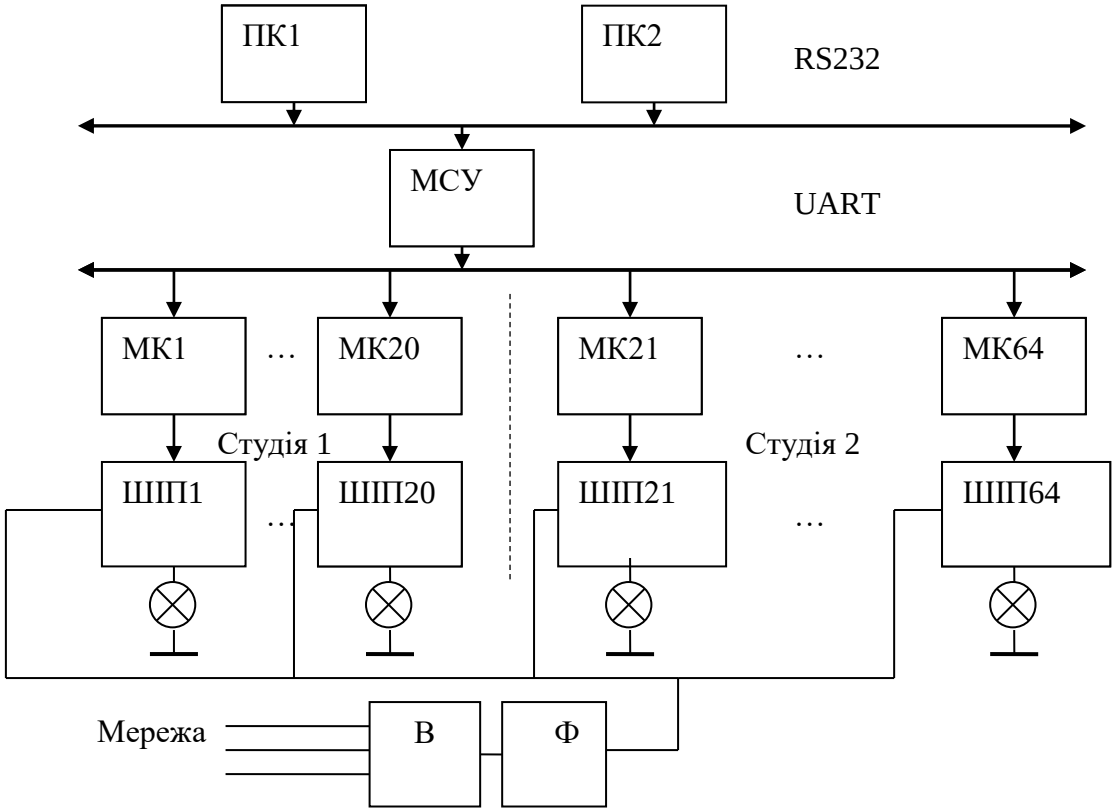


Рис. 1.4.

ЛЕКЦІЯ 2

Послідовний периферійний інтерфейс SPI

Характеристика та структура інтерфейса SPI (Serial Peripheral Interface). Послідовний периферійний інтерфейс SPI розроблений, як повнодуплексний чотирипровідний інтерфейс з шинною конфігурацією вузлів (пристроїв), що підключаються, для систем з одним головним вузлом. Первинна базова версія інтерфейсу SPI дозволяє підключати до одного головного (або ведучого - Master) вузла декілька ведених (Slave) вузлів через загальну шину. Окремий сигнал вибору веденого пристрою NSS (Slave Select signal) використовується для вибору веденого пристрою при здійсненні з ним обміну даними. Схема підключення двох ВІС МК по інтерфейсу SPI наведена на рис.1.5.

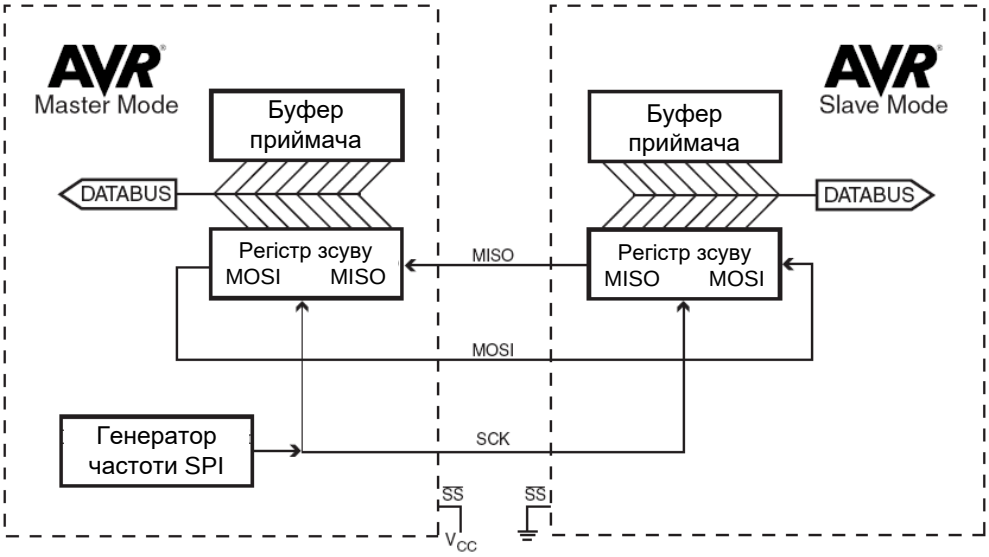


Рис. 1.5. Схема підключення двох ВІС МК по інтерфейсу SPI

Інтерфейс SPI має чотири сигнальні лінії:

- Лінія MOSI (Master-Out, Slave-In) - вихідна лінія даних ведучого інтерфейсу і вхідна лінія даних веденого інтерфейсу. З назви виходить, що лінія призначена для передачі даних від ведучого (Master) інтерфейсу (або вузла мережі) до веденого (Slave) інтерфейсу (або вузлу мережі).
- Лінія MISO (Master-In, Slave-Out) - вхідна лінія даних ведучого інтерфейсу і вихідна лінія даних веденого інтерфейсу. Лінія призначена для передачі даних від веденого інтерфейсу до ведучого. Дані передаються байтами, побітно, починаючи із старшого біта. Слід пам'ятати, що вивід MISO веденого інтерфейсу знаходиться в стані високоімпедансу, якщо ведений інтерфейс не вибраний по лінії NSS
- Лінія NSS (Slave Select) – ($\overline{SS}$) лінія вибірки веденого.
- Лінія SCK (Serial Clock) - вихідна лінія тактових імпульсів ведучого вузла і вхідна лінія тактових імпульсів веденого вузла. Лінія SCK використовується для синхронізації передачі даних між ведучим і веденим інтерфейсами по лініях MOSI і MISO.

Діаграмі обміну по інтерфейсу SPI (рис 1.8) пояснюють вибір полярності (CPOL) та фази (CPHA) тактових імпульсів.

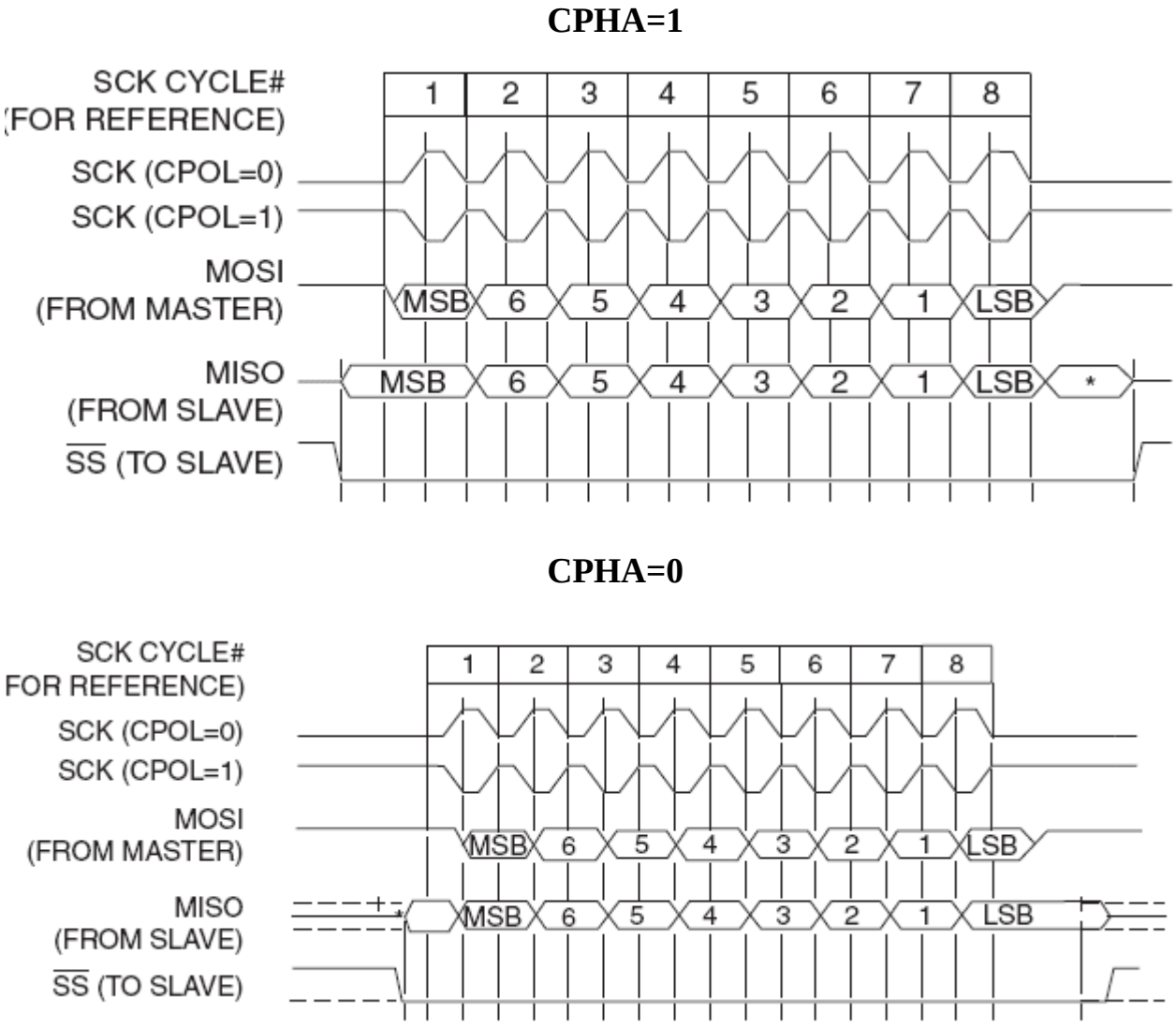


Рис. 1.6. Діаграмі обміну по інтерфейсу SPI

MSB – Most Significan Bit, LSB – Less Significan Bit

Регістри SPI. Зазвичай при апаратній реалізації інтерфейсу SPI, він доступний програмістові через чотири спеціалізованих функцій регістра SFR (Special function registers). Назви цих регістрів можуть трохи відрізнятися для мікроконтроллерів різних виробників, проте їх функціональне призначення при цьому залишається незмінним. Надалі використовуватимемо найменування SFR регістрів, вживаних в документації фірми Silicon Laboratories Corp., яка в даний час є лідером в розвитку інтерфейсу SPI.

Інтерфейсу SPI містить всього чотири SFR регістра:

1. SPIODAT - регістр даних;

2. SPIOCKR - регістр управління швидкістю;

3. SPIOCFG - регістр конфігурації;

4. SPIOCN - регістр управління шиною SPI.

Перші два регістри, призначення яких вочевидь з назви, не представляють особливого інтересу при розгляді архітектурних особливостей інтерфейсу. Зазначимо, що код, записуваний в регістр SPIOCKR, - управління швидкістю, дозволяє визначити швидкість роботи інтерфейсу SPI (частоту тактових імпульсів F SCLK) по наступній формулі:

F SCLK = 0.5 * SYSCLK / (SPIOCKR +1)

З приведеної формули виходить, що максимальна тактова частота SPI інтерфейсу може бути рівна половині системної тактової частоти SYSCLK в режимі ведучого (майстера). У режимі веденого швидкість передачі інтерфейсу SPI визначається тактовою частотою ведучого інтерфейсу SPI.

Третій регістр - регістр конфігурації інтерфейсу – SPIOCFG, містить біти CKPHA (SPI Clock Phase) - управління фазою тактування, і CKPOL (SPI Clock Polarity) - управління полярністю тактуючих імпульсів. Ці біти дозволяють вибрати фазу і полярність імпульсів тактування (Див. рис. 1.6).

У базовій версії мережі на базі SPI інтерфейсів лише один інтерфейс може бути ведучим. Інтерфейс встановлюється в режим ведучого установкою прапора MSTEN (Master Enable Flag) - біта SPIOCN.l. Якщо інтерфейс встановлений в режим ведучого, то запис байта даних в регістр даних SPIODAT приводить до початку передачі. Ведучий інтерфейс негайно побітно зсуває дані і видає їх на лінію MOSI у супроводі тактових імпульсів на лінії SCK. Після завершення передачі встановлюється прапор SPIF (SPIOCN.7). Якщо дозволені переривання, видається відповідне переривання. Крім того, інтерфейс може бути запрограмований на видачу від одного до восьми бітів для здійснення зв'язку по SPI з приладами, що мають різну довжину слова. Довжина передачі (кількість передаваних бітів) може бути

задана бітами SPIFRS в регістрі конфігурації SPIOCFG.[2:0] (SPI Configuration Register).

Режими роботи

Інтерфейс може працювати в повнодуплексному режимі, тобто можлива одночасна передача даних по лініях MOSI від ведучого до веденого, і MISO від веденого до ведучого. Дані, отримані від веденого інтерфейсу, замінюють дані в регістрі даних ведучого інтерфейсу. Цей регістр двічі буферизован на введення, але не на виведення. Тобто якщо в регістр даних SPIODAT відбувається спроба запису даних під час передачі попереднього байта, встановлюється прапор WCOL (SPIOCN.6) і спроба запису ігнорується. Таким чином, поточна передача даних продовжується безперервно. Читання з регістра даних SPIODAT приводить до читання приймального буфера. Якщо прийом не закінчений, встановлюється прапор RXOVRN (SPIOCN.4). Нові дані не передаються в регістр читання, поки попередній прийнятий байт не буде прочитаний. Вочевидь, що при затримці читання прийнятих байтів може статися втрата даних. Якщо SPI інтерфейс не налагоджений, як ведучий, він працюватиме в режимі веденого.

Не дивлячись на те, що базова версія інтерфейсу SPI розроблена для мікроконтроллерних систем з одним ведучим, можливий все ж режим мережі з багатьма ведучими. Прапор MODF (SPIOCN.5 - Mode Fault flag) встановлюється в логічну одиницю, якщо інтерфейс визначений як ведучий (MSTEN=1) і вивід NSS переведений в низький логічний рівень, тобто SPI інтерфейс намагаються використовувати як ведений. Якщо при цьому встановлений прапор MODF, біти MSTEN і SPIEN в регістрі управління SPI скидаються апаратно, переводячи інтерфейс в автономний стан. Таким чином, в системі з багатьма ведучими, ядро може визначити чи вільна шина шляхом опиту прапора SLVSEL (SPIOCN.2) перед тим, як встановити MSTEN прапор (тобто визначити інтерфейсу режим ведучого) і ініціалізувати обмін.

Окрім первинного базового інтерфейсу в багатьох сучасних мікроконтролерах використовується також і розширений інтерфейс SPI.

У *базовому* варіанті інтерфейсу SPI регістр конфігурації SPIOCFG містить також біти:

BC2- BCO (SPI Bit Count) – що відображають номер поточного передаваного біта;

SPIFRS2- SPIFRSO (SPI Frame Size), що визначають розмір фрейма (довжини ередаваного слова).

У *розширеному* інтерфейсі ці біти відсутні, оскільки при сучасних високих швидкостях передачі визначення поточного передаваного біта і зміна формату фрейма втрачає всякий сенс. Замість цих бітів розширений інтерфейс SPI має наступні конфігуруючі біти, що дозволяють оптимізувати роботу систем з багатьма ведучими інтерфейсами:

MSTEN (Master Mode Enable) – біт дозволу режиму ведучого;

NSSIN (NSS Instantaneous Pin Input) – біт, що відображає стан входу NSS у момент читання;

RXBMT (Receive Buffer Empty) – біт, що відображає, що регістр читання порожній;

SLVSEL (Slave Selected Flag) – біт стану веденого;

SPIBSY (SPI Busy) – біт зайнятості інтерфейсу SPI, що в деякій мірі замінює біти SPI Bit Count базового інтерфейсу;

SRMT (Shift Register Empty) - біт, що відображає звільнення регістра зсуву даних.

Приведені біти дозволяють діагностувати інтерфейс при шинній організації системи з багатьма ведучими.

Четвертий регістр SPIOCN - регістр управління інтерфейсу SPI. Він має ряд бітів управління, загальних як для базової, так і для розширеної версій:

- MODF (Mode Fault Flag) - прапор помилки режиму, що відображає невідповідність призначеного режиму (біта ведучого MSTEN=1) і стану входу NSS=0;
- RXOVRN (Receive Overrun Flag) – біт, що відображає випадок, якщо розпочато прийом наступного байта при непрочитаному попередньому байті, тобто біт відображає розсинхронізацію читання;
- WCOL (Write Collision Flag) - прапор розсинхронізації запису, що встановлюється в разі, якщо здійснена спроба запису байта в регістр під час незавершеної передачі попереднього байта;
- SPIF (SPI Interrupt Flag) - прапор переривання інтерфейсу SPI, що встановлюється апаратно після завершення передачі;
- SPIEN (SPI Enable) - біт дозволу роботи інтерфейсу SPI.
- Окрім цього, *базовий* інтерфейс має ряд бітів, які в розширеному інтерфейсі були переведені в регістр конфігурації SPI0CFG:
- TXBSY (Transmit Busy Flag) - біт зайнятості, - еквівалентний новому біту SPIBSY (SPI Busy); а також біти SLVSEL і MSTEN.

Розширений інтерфейс SPI має наступні конфігуруючі біти:

NSSMD1- NSSMD 0 (Slave Mode Select) –біти вибору режиму веденого інтерфейсу: 3-хпроводний режим ведучого або веденого; 4-х проводний режим веденого або режим з багатьма ведучими (NSS – завжди в режимі входу); 4-х проводний режим одного ведучого.

Таким чином, очевидна основна відмінність розширеного інтерфейсу SPI від базового. У розширеному варіанті передбачена робота шини SPI в багатоконтролерному режимі, що виразилося в розширенні засобів діагностики помилок і можливості управління сигналом NSS.

Класична архітектура SPI шини для базового варіанту інтерфейсу приведена на рис.1.7. Це так звана класична 4х – провідна структура. Один ведучий в ній управляє декількома (N) веденими. Всі ведені підключені паралельно на лініях SCLK, MISO і MOSI шини SPI. Вибірка одного з

ведених відбувається за допомогою однієї з ліній портів введення /виведення, яка з'єднується з входом NSS відповідного веденого. Зазвичай така архітектура використовується для побудови микроконтроллерных систем з одним мікроконтроллером, що виконує роль ведучого і рядом периферійних мікросхем, що виконують роль ведених. Як периферійні мікросхеми може бути використаний ряд сучасних мікросхем, оснащених інтерфейсом SPI: таймерів реального часу RTC, аналого-цифрових перетворювачів ADC, цифро-аналогових перетворювачів DAC, різних мікросхем пам'яті і тому подібне

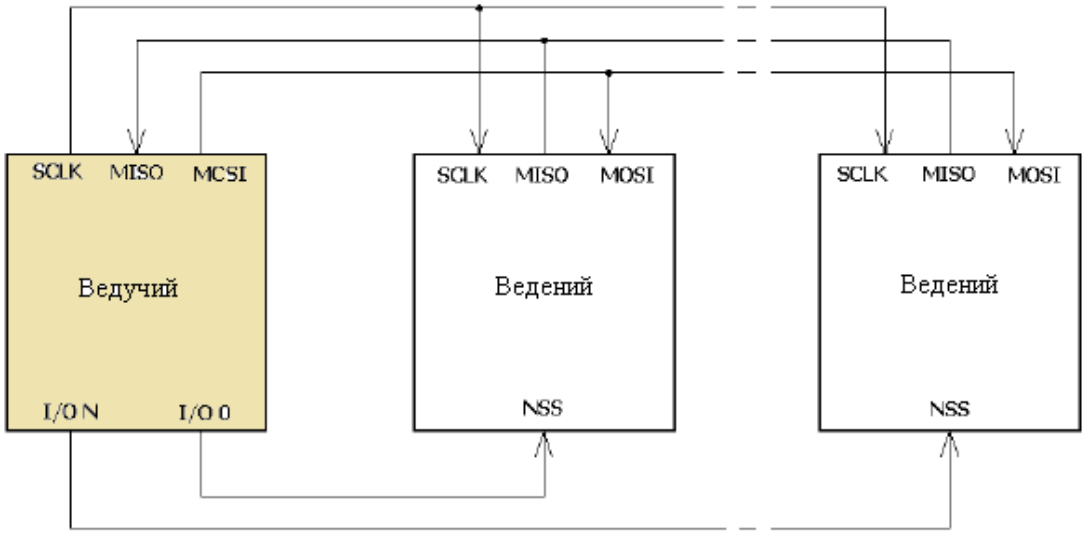


Рис.1.7. Архітектура 4х–проводной SPI шини з одним ведучим і декількома веденими

Частним випадком попередньої архітектури є так звана 3х-провідна структура, приведена на рис.1.8. Її відмінність полягає в тому, що ведена мікросхема постійно вибрана, і отже, необхідність в четвертій лінії інтерфейсу SPI відсутня.

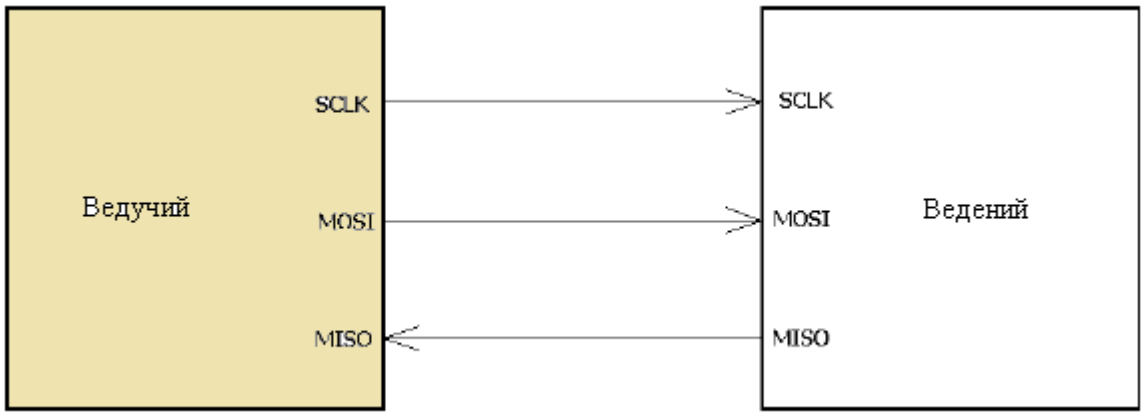


Рис.1.8. Архітектура 3х-провідної шини SPI з ведучим і веденим.

Розширений інтерфейс дозволяє організувати двопроцесорну 4х-проводную архітектуру, приведену на рис.1.9. В цьому випадку кожен з двох вузлів може виконувати роль ведучого. На практиці така архітектура використовується досить рідко, зважаючи на алгоритмічну складність діагностики станів колізій (конфліктів, викликаних одночасною передачею на шину двох провідних вузлів). Ширше використовується простіша в розумінні і діагностиці конфліктів, так звана архітектура з 4х-провідною «перехресною» вибіркою, приведена на рис.1.10. В цьому випадку для вибірки кожного з вузлів використовується лінія введення /виведення іншого вузла, що працює лише на вивід, а не на введення /виведення, як в попередній структурі.

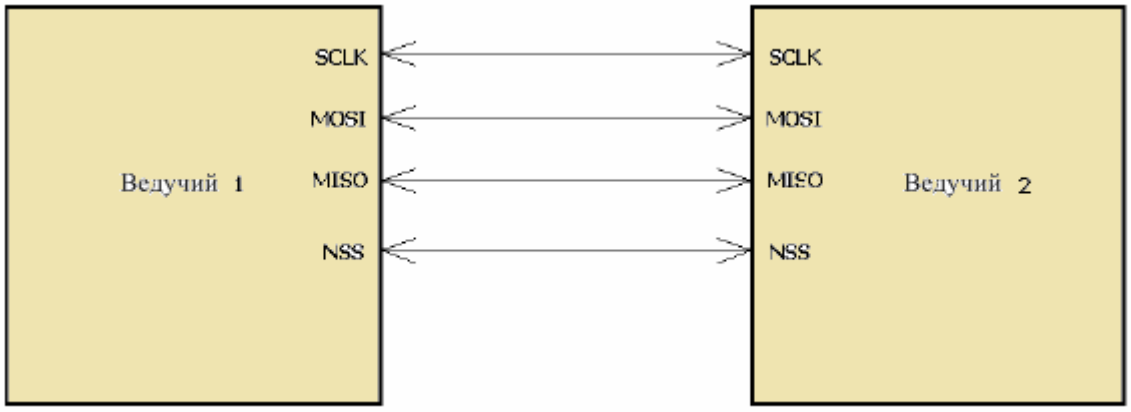


Рис.1.9. Архітектура 4х–провідної SPI шини з двома ведучими

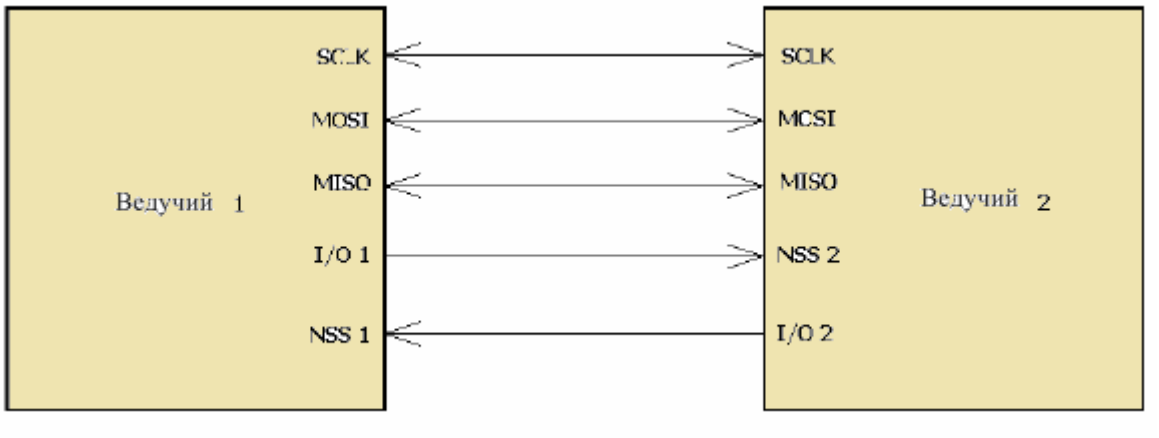


Рис.1.10. Архітектура 4х–проводной SPI шини з двома ведучими і «перехресною» вибіркою

Відмітимо, що двопроцесорна архітектура використовується досить рідко, наприклад, в разі, якщо один з мікроконтроллерів призначений для введення і обробки сигналів, а другий – для організації інтерфейсу з персональним комп'ютером або локальною мережею. В цьому випадку і один, і інший мікроконтроллери повинні мати можливість ініціювати процедуру обміну даними, а отже, повинні мати можливість функціонування в режимі ведучого. Більш поширений випадок використання двопроцесорної структури з ресурсом, що розділяється, наприклад пам'яттю, наведений на рис.1.11.

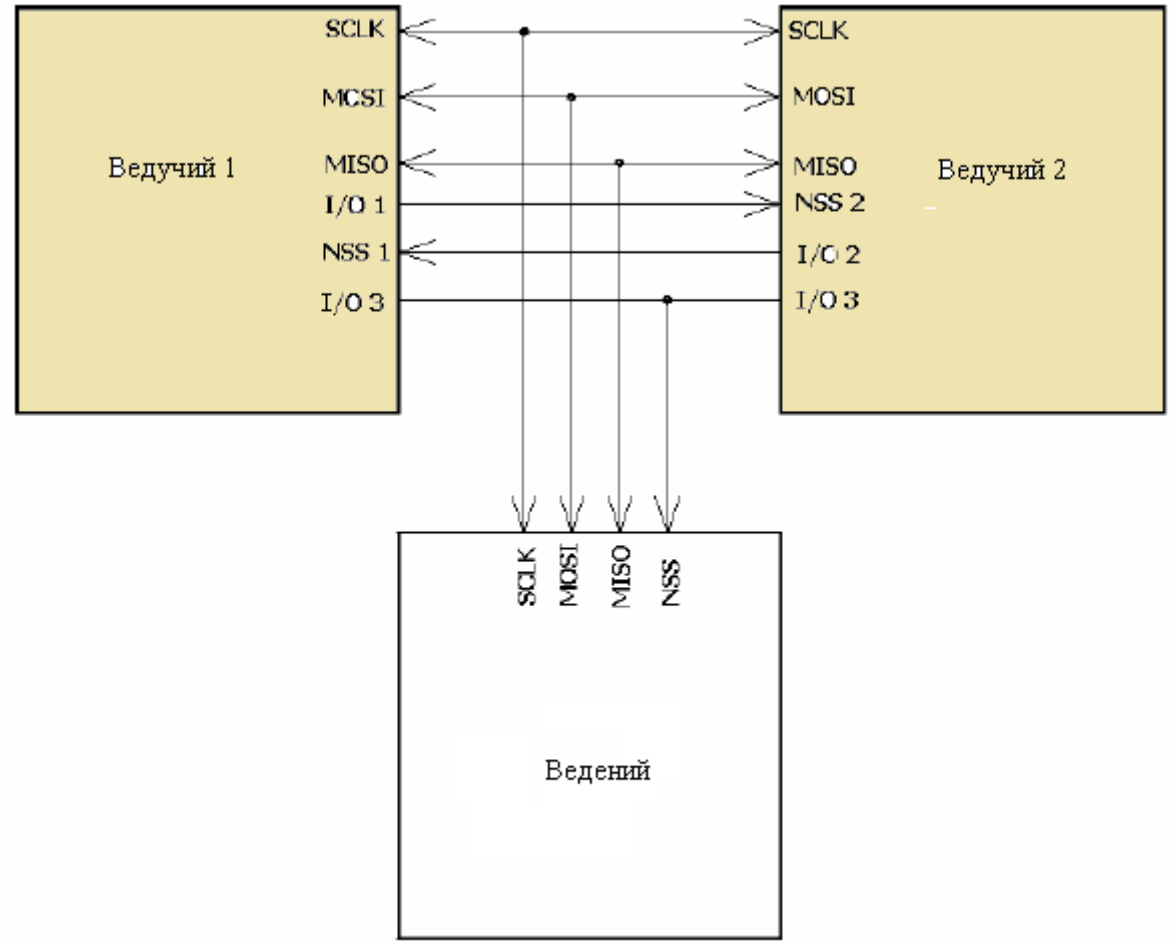


Рис.1.11. Двопроцесорна архітектура з ресурсом, що розділяється

У приведеній на рис.1.11. архітектурі є два ведучих і один ведений. Зв'язки між ведучими мінімізовані і відповідають архітектурі 4х–провідної SPI шини з двома ведучими і «перехресною» вибіркою. Лінії основних сигналів SPI шини підключені також до загального веденого, а для вибірки загального веденого використовується по одній додатковій лінії введення/виведення від кожного з ведучих, об'єднаних за «монтажним АБО». Архітектура в принципі працездатна, проте аналіз зайнятості SPI шини і діагностики колізій в ній ще більш утруднений. Це пояснюється необхідністю послідовного опиту як мінімум двох ліній вибірки (індивідуальною контроллера, що ініціює обмін, і загального ресурсу) перед заняттям шини, а отже і збільшенням вірогідності виникнення колізії. Для

спрощення процедур обміну рекомендується ввести ще дві лінії прапорів запиту шини ID1 і ID2, як показано на рис.1.12.

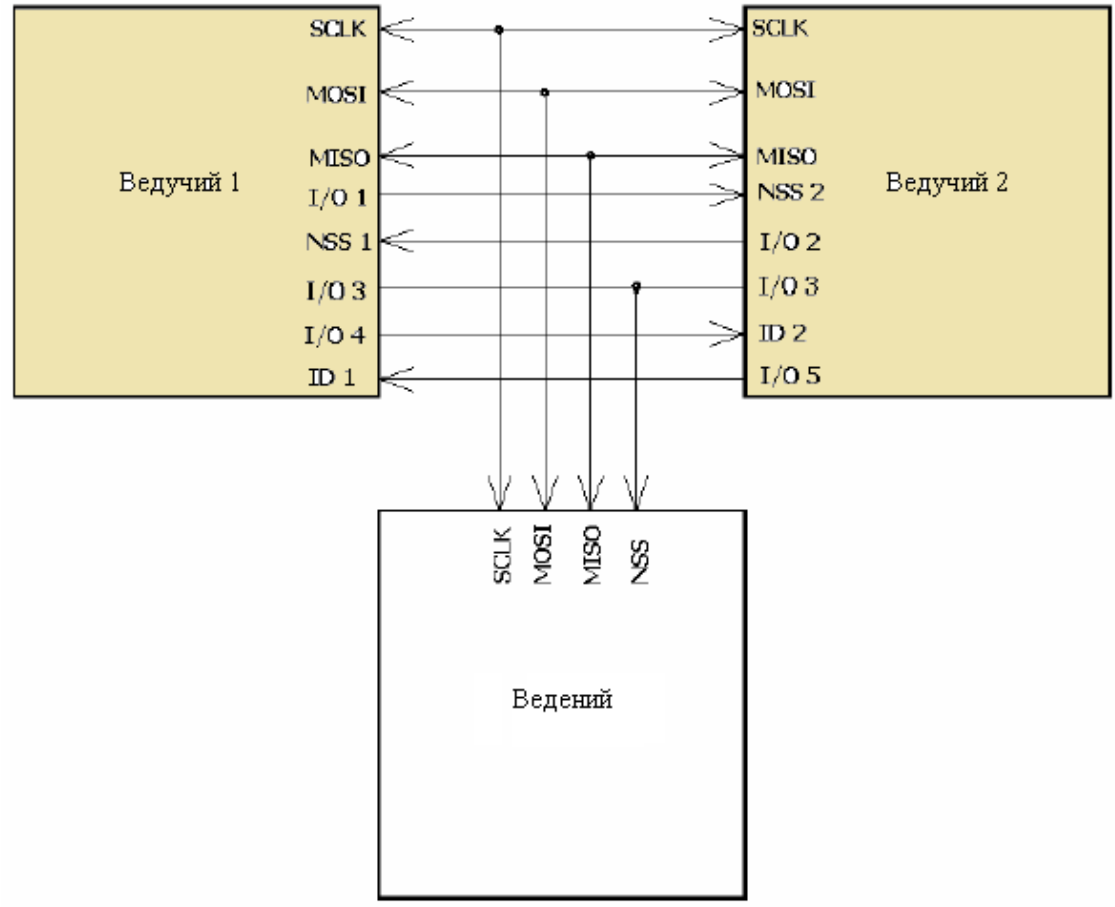


Рис.1.12. Двопроцесорна архітектура з розділеним ресурсом і лініями запиту шини

В цьому випадку при необхідності заняття шини контроллер виставляє свій прапор запиту шини, і чекає у відповідь прапора від другого контроллера. Другий контролер, отримавши прапор запиту, або відразу виставляє свій прапор, підтверджуючи, що шина вільна, або закінчує поточні операції обміну і тільки тоді виставляє свій прапор. Такий механізм значно спрощує діагностику зайнятості шини.

Також відзначимо як мінімум два варіанти вкористання останніх двох архітектур.

Варіант перший - двопроцесорна архітектура, що містить перший швидкодіючий мікроконтроллер C8051F121 і другий мікроконтроллер C8051F320 з інтерфейсом USB для зв'язку з персональним комп'ютером. Як загальний ресурс, що розділяється, використовується DataFlash пам'ять великого об'єму для накопичення даних, наприклад AT45DB642. Перший мікроконтролер вимірює за допомогою вбудованих аналого-цифрових перетворювачів ADC дані з датчиків об'єкту спостереження або управління, обробляє їх і записує в DataFlash пам'ять, а другий контроллер за певних умов передає накопичені дані в персональний комп'ютер по інтерфейсу USB. Другий варіант - аналогічний першому, але замість швидкодіючого першого мікроконтролера використовується мікроконтроллер C8051F350 зі вбудованими 24-розрядним аналого-цифровим перетворювачем.

Приведена архітектура необхідна лише тоді, поки перший мікроконтролер (надшвидкодіючий або з високоточним ADC) не буде оснащений власним інтерфейсом USB.

ЛЕКЦІЯ 3

Розподілена мікропроцесорна система на базі шини I2C.

Загальні відомості

У побутовій техніці, телекомунікаційному устаткуванні і промисловій електроніці часто зустрічаються схожі рішення, Наприклад, практично кожна система включає:

- Деякий “розумний” вузол управління, зазвичай однокристальна МІКРОЕОМ.
- Вузли загального призначення, такі як буфери PKI, порти введення/виведення, ОЗП, EEPROM або перетворювачі даних.

- Специфічні вузли, такі як схеми цифрового налаштування і обробки сигналу для радио- і відео- систем, або генератори тонального набору для телефонії.

Для того, щоб використовувати ці загальні рішення, а також для збільшення ефективності апаратури і спрощення рішень схемотехнік. Фірма Philips розробила просту двонаправлену двопровідну шину для ефективного "міжмікросхемного" (inter-IC) управління.. В даний час асортимент продукції Philips включає більше 150 КМОП і біполярних пристроїв I2C-сумісних, функціонально призначених для роботи у всіх трьох вищеперелічених категоріях електронного устаткування.

Переваги шини I2C наступні:

- Потрібно лише дві лінії - лінія даних (SDA) і лінія синхронізації (SCL) Кожен пристрій, підключений до шини, може бути програмно адресований за унікальною адресою. У кожен момент часу існує просте відношення введучій /ведений: ведучі можуть працювати як ведучий-передавач і ведучий-приймач.
- Шина дозволяє мати декілька ведучих, надаючи засоби для визначення колізій і арбітраж для запобігання пошкодженню даних за ситуації, коли два або що більш ведучих одночасно починають передачу даних В стандартному режимі забезпечується передача послідовних 8-бітових даних з швидкістю до 100 кбіт/с, і до 400 кбіт/с в "швидкому" режимі.
- Вбудований в мікросхеми фільтр пригнічує сплески, забезпечуючи цілісність даних.
- Максимальна допустима кількість мікросхем, приєднаних до однієї шини, обмежується максимальною ємкістю шини 400 пФ.

До недоліка можна віднести недостатню швидкість передачі даних. Дійсно, шина I2C є послідовною шиною і повинна застосовуватися тоді, коли система, що здійснює функції управління, не вимагає високошвидкісної передачі даних.

Хоча послідовні шини не мають пропускної спроможності паралельних шин, вони вимагають менше з'єднань і менше контактів мікросхем.

Поняття шини складається не лише із призначення сполучаючих дротів, вона також включає всі формати і процедури для зв'язку усередині системи.

Пристрої, що зв'язуються по шині, повинні мати деякий протокол, який попереджує всі можливості зіткнень, втрати даних і блокування інформації. Швидкі пристрої мають бути в змозі зв'язатися з повільними пристроями. Система не має бути залежна від пристроїв, підключених до неї, інакше модифікації і поліпшення стануть неможливими. Також має бути розроблена процедура, що встановлює, який пристрій управляє шиною і коли. Крім того, якщо різні пристрої з різними тактовими частотами підключені до шини, має бути визначене джерело синхронізації шини. Всім цим критеріям задовольняє шина I2C.

Приклад конфігурації шини I2C

Шина I2C підтримує будь-яку технологію виготовлення мікросхем (nМОП, КМОП, біполярну). Дві лінії даних (SDA) і синхронізації (SCL) служать для перенесення інформації. Кожен пристрій розпізнається за унікальною адресою - будь то мікроконтроллер, PKI буфер, пам'ять або інтерфейс клавіатури - і може працювати як передавач або приймач, залежно від призначення пристрою. Зазвичай PKI буфер - лише приймач, а пам'ять може як приймати, так і передавати дані. Крім того, пристрої можуть бути класифіковані як що ведучі і ведені при передачі даних. Ведучий - це пристрій, який ініціює передачу даних і виробляє сигнали синхронізації. При цьому будь-який пристрій, що адресується, вважається веденим по відношенню до ведучого. Терміни шини I2C наведено в таблиці 1.3., приклад конфігурації системи з 2 мікроконролерами по шині I2C – на рис 1.13.

Таблиця 1.3 - Терміни шини I2C

Термін (англ.)	Термін (укр)	Опис
Transmitter	Передавач	Пристрій, що посилає дані в шину
Receiver	Приймач	Пристрій, що приймає з шини
Master	Ведучий	Починає пересилку даних, виробляє синхроімпульси, закінчує пересилку даних
Slave	Ведений	Пристрій, що адресується ведучим
Multi-master	-	Декілька ведучих можуть намагатися захопити шину одночасно, без порушення передаваної інформації
Arbitration	Арбітраж	Процедура, забезпечуючи режим Multi-master
Synchronization	Синхр.	Процедура синхронізації двох пристроїв

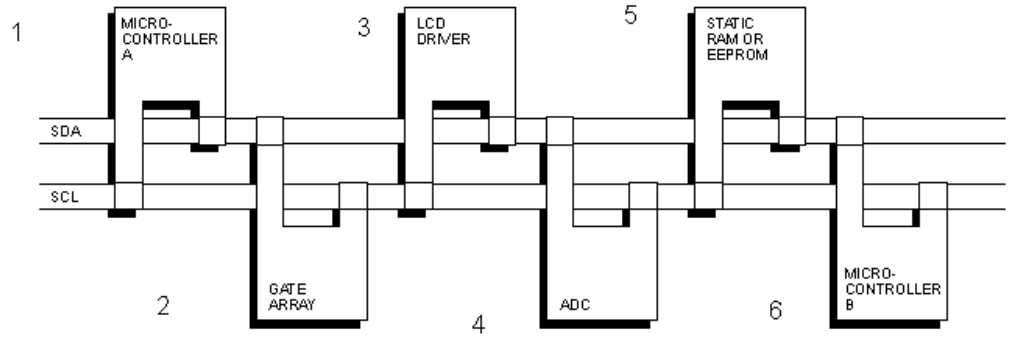


Рис. 1.13. Приклад конфігурації шини I2C з двома мікроконтроллерами

Схема (рис 1.13) містить:

- Microcontroller A -Мікроконтроллер А ;
- GATE ARRAY - Масив ключів;
- LCD-Driver – рідинно-кристаличний драйвер;
- ADC – АЦП;
- STAT IC RAM or EEPROM - Статична ОЗП або ППЗП;

- Microcontroller B - Мікроконтроллер В.

Шина I2C допускає декілька ведучих. Це означає, що більш ніж один пристрій, який здатний управляти шиною, може бути підключений до неї.

Розглянемо взаємини передавач-приймач і ведучий-ведений, що існують в шині I2C на прикладі пересилки даних між двома мікроконтролерами А та В, підключеними до шини (рис 1.13.). Необхідно відмітити, що ці стосунки не постійні, а залежать лише від напряму пересилки даних в даний момент часу. Пересилка даних відбуватиметься таким чином:

Хай мікроконтролер А бажає послати інформацію в мікроконтролер В, тоді послідовність дій наступна:

- мікроконтролер А (ведучий) адресує мікроконтроллер В (ведений);
- мікроконтролер А (ведучий-передавач) посилає дані мікроконтролеру В (ведений приймач);
- мікроконтролер А закінчує пересилку.

Хай мікроконтролер А бажає прийняти інформацію від мікроконтролера В тоді послідовність дій наступна:

- мікроконтролер А (ведучий) адресує мікроконтролер В (ведений);
- мікроконтролер А (ведучий-приймач) приймає дані від мікроконтролера В (ведений передавач);
- мікроконтролер А закінчує пересилку.

У обох випадках ведучий (мікроконтролер А) генерує синхроімпульси і закінчує пересилку. Генерація синхросигналу - це завжди обов'язок ведучого; кожен ведучий генерує свій власний сигнал синхронізації при пересилці даних по шині. Сигнал синхронізації може бути змінений лише якщо він “витягується” повільним веденим пристроєм (шляхом утримання лінії в низькому стані), або іншим ведучим, якщо відбувається зіткнення.

Можливість підключення більш одного мікроконтроллера до шини означає, що більш ніж один ведучий може спробувати почати пересилку в

один і той же момент часу. Для усунення хаосу, який може виникнути в даному випадку, розроблена процедура арбітражу.

Підключення I2C-устройств до шини

Як SDA, так і SCL є двонапрямленими лініями, приєднаними до додатнього джерела живлення через підтягуючий резистор (рис. 1.14.). Коли шина вільна, обидві лінії знаходяться у ВИСОКОМУ положенні. Вихідні каскади пристроїв, підключених до шини, повинні мати відкритий стік або відкритий колектор для забезпечення функції монтажного І.

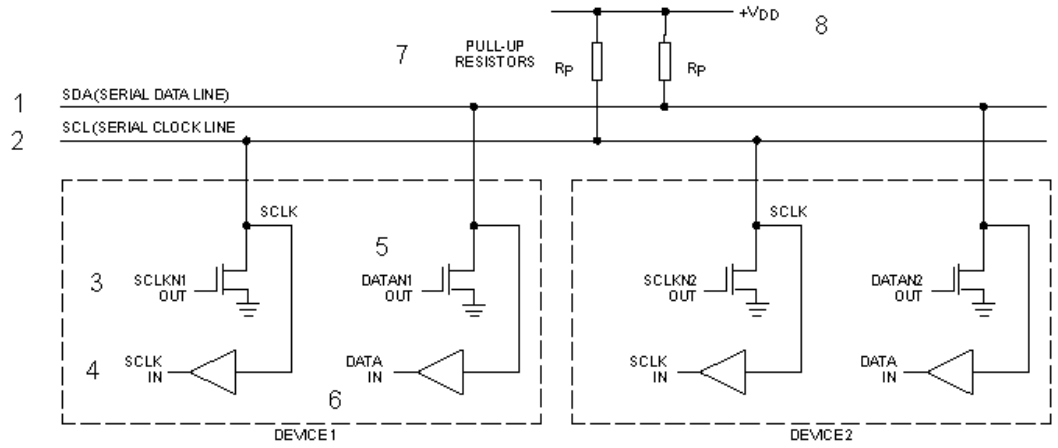


Рис. 1.14. Підключення I2C-приладів до шини

На рис 1.14 позначено:

SDA - лінія даних;

SCL - лінія синхронізації;

SCLKOUT - вихід синхронізації;

SCLKIN - вхід синхронізації;

DATA OUT - вихід даних;

DATA IN - вхід даних;

Rp - підтягуючі резистори;

Vdd - напруга живлення.

Пересилка біта даних

Унаслідок різних технологій мікросхем (КМОП, nМОП, біполярна), які можуть бути підключені до шини, рівні логічного нуля (“НИЗЬКИЙ”) і логічної одиниці (“ВИСОКИЙ”) не фіксовані і залежать від відповідного рівня напруги Vdd .

Один синхроімпульс генерується на кожен біт, що пересилається.

Дані на лінії SDA повинні бути стабільними протягом ВИСОКОГО періоду синхроімпульсу. ВИСОКИЙ або НИЗЬКИЙ стан лінії даних повинне мінятися, тільки якщо лінія синхронізації в стані НИЗЬКИЙ (рис 1.20).

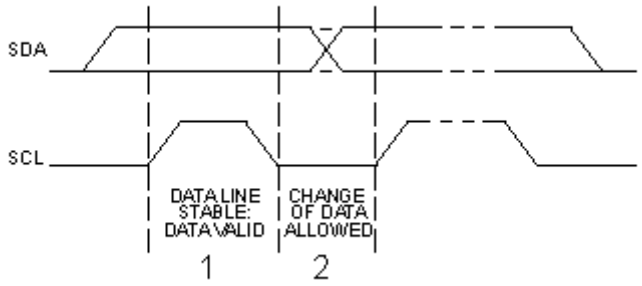


Рис. 1.15. Пересилка біта в шині I2C

Лінія даних знаходиться в стабільному стані, дані визначені

Допускається зміна даних

Сигнали START и STOP

Початок та кінець передачы визначають спеціальні ситуації - СТАРТ і СТОП (рис 1.16).

Перехід лінії SDA з ВИСОКОГО стану в НИЗЬКИЙ, тоді як SCL знаходиться у ВИСОКОМУ стані означає START.

Перехід лінії SDA з НИЗЬКОГО стану у ВИСОКЕ при SCL у ВИСОКОМУ стані означає STOP.

Сигнали СТАРТ і СТОП завжди виробляються ведучим. Вважається, що шина зайнята після сигналу СТАРТ. Шина вважається такою, що звільнилася через певний час після сигналу СТОП.

Визначення сигналів СТАРТ і СТОП пристроями, підключеними до шини, відбувається досить легко, якщо в них вбудовані необхідні ланцюги.

Проте мікроконтролери без таких ланцюгів повинні здійснювати прочитування значення лінії SDA як мінімум двічі за період синхронізації для того, щоб визначити перехід стану.

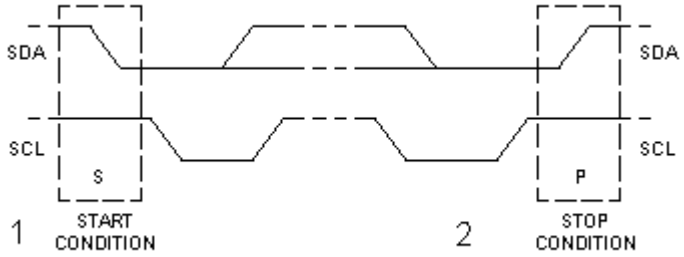


Рис. 1.16. Сигнали СТАРТ(1) і СТОП(2)

*Формат байта***.**

Кількість байт, передаваних за один сеанс зв'язку по лінії SDA необмежена. Кожен байт повинен закінчуватися бітом підтвердження. Дані передаються, починаючи з найбільш значущого біта MSB (рис. 1.17.). Якщо приймач не може прийняти наступний байт, поки він не виконає яку-небудь іншу функцію (наприклад, обслужить внутрішнє переривання), він може утримувати лінію SCL в НИЗЬКОМУ стані, переводячи передавач в стан чекання. Пересилка даних продовжується, коли приймач буде готовий до наступного байта і відпустить лінію SCL. В деяких випадках, необхідно використовувати інший формат даних (наприклад, CBUS). Посилка, яка передається з такою адресою, може бути закінчена видачею сигналу СТОП, навіть якщо це відбувається під час передачі байта. В цьому випадку підтвердження не генерується.

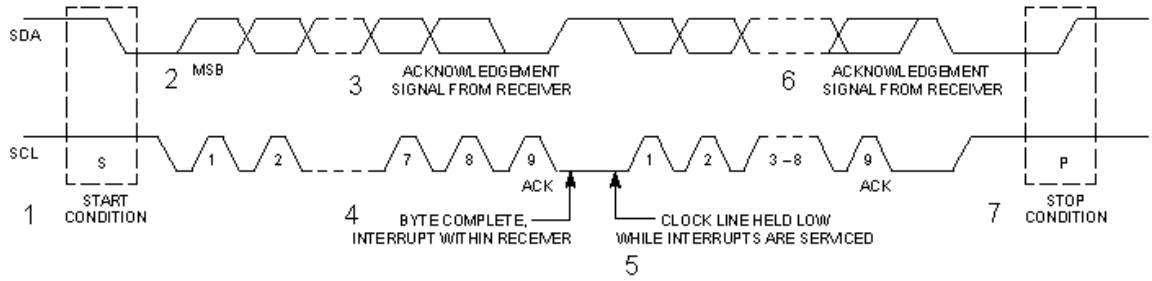


Рис.1.17. Пересилка даних по шині I2C

1. Сигнал СТАРТ (S)
2. Старший розряд байта (MSB –most significan bit)
3. Сигнал підтвердження від приймача (ACK)
4. Прийом байта завершений. Переривання усередині приймача
5. Синхролінія утримується в низькому стані, поки обслуговується переривання
6. Сигнал підтвердження від приймача (ACK)
7. Сигнал СТОП (P).

Підтвердження

Підтвердження при передачі даних обов'язково. Для цього ведучий генерує додатковий (9-й) імпульс синхронізації. Передавач відпускає (переводить у ВИСОКИЙ стан) лінію SDA протягом синхроімпульса підтвердження. Приймач повинен утримувати лінію SDA протягом ВИСОКОГО стану синхроімпульса підтвердження в стабільно НИЗЬКОМУ стані (рис. 1.18.). Зазвичай, приймач, який був адресований, зобов'язаний генерувати підтвердження після кожного прийнятого байта, виключаючи ті випадки, коли посилка починається з адреси CBUS.

У тому випадку, коли ведений приймач не може підтвердити свою адресу (наприклад, коли він виконує в даний момент які-небудь функції реального часу), лінія даних має бути залишена у ВИСОКОМУ стані. Після цього ведучий може видати сигнал СТОП для переривання пересилки даних.

Якщо ведений приймач підтвердив свою адресу, але через деякий час більше не може приймати дані, ведучий також повинен перервати пересилку.

Для цього ведений не підтверджує наступний байт, залишає лінію даних у ВИСОКОМУ стані і ведучий генерує сигнал СТОП.

Якщо в пересилці бере участь ведучий-приймач, то він повинен повідомити про закінчення передачі веденому передавачу шляхом не підтвердження останнього байта. Ведений передавач повинен звільнити лінію даних для того, щоб дозволити ведучому видати сигнал СТОП або повторити сигнал СТАРТ.

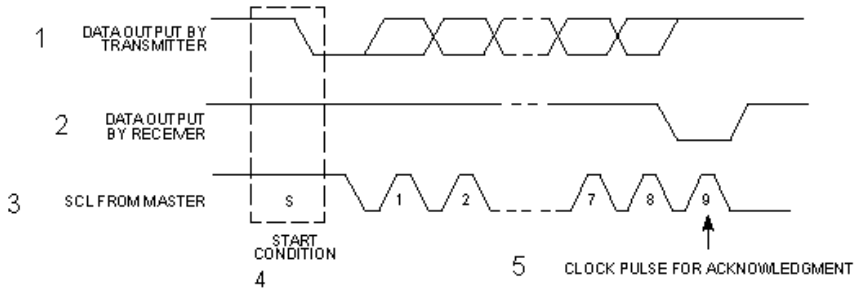


Рис. 1.18. Підтвердження

1. Дані, передані передавачем
2. Дані, передані приймачем
3. Синхроімпульси SCL від ведучого
4. Сигнал СТАРТ
5. Синхроімпульс підтвердження

Синхронізація

При передачі посилок по шині I2C кожен ведучий генерує свій синхросигнал на лінії SCL. Дані дійсні лише під час ВИСОКОГО стану синхроімпульса.

Синхронізація виконується з використанням підключення до лінії SCL за правилом монтажного І. Це означає, що унаслідок переходу лінії SCL з ВИСОКОГО стану в НИЗЬКИЙ, викликаного переходом синхросигналу одного з пристроїв в НИЗЬКИЙ стан, відбудеться також перехід синхросигналу іншого пристрою в НИЗЬКИЙ стан.

Цей стан лінії SCL утримується доти, поки не буде встановлено ВИСОКИЙ стан внутрішнього синхросигналу одного з пристроїв (рис.1.19). Проте, перехід з НИЗЬКОГО стану у ВИСОКИЙ синхросигналу може не викликати аналогічний перехід на лінії SCL, якщо синхросигнал іншого пристрою все ще знаходиться в НИЗЬКОМУ стані. Таким чином, лінія SCL знаходитиметься в НИЗЬКОМУ стані впродовж щонайдовшого НИЗЬКОГО періоду з двох синхросигналів. Пристрої з коротшим НИЗЬКИМ періодом входитимуть в стан чекання на якийсь час, поки не кінчиться довгий період.

Коли у всіх задіяних пристроїв кінчиться НИЗЬКИЙ період синхросигналу, лінія SCL перейде у ВИСОКИЙ стан. Всі пристрої почнуть проходити ВИСОКИЙ період своїх синхросигналів. Перший пристрій, в якого кінчиться цей період, знову встановить лінію SCL в НИЗЬКИЙ стан.

Таким чином, НИЗЬКИЙ період синхролинии SCL визначається найдовшим періодом синхронізації зі всіх задіяних пристроїв, а ВИСОКИЙ період визначається найкоротшим періодом синхронізації пристроїв.

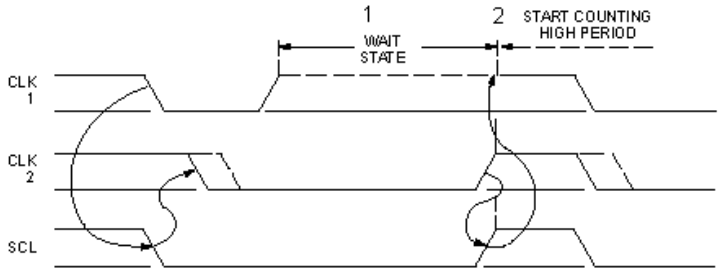


Рис. 1.19. Синхронізація під час арбітражу

1. Стан чекання
2. Початок відліку ВИСОКОГО періоду синхроімпульса

Арбітраж

Ця процедура заснована на тому, що всі I2C-устройства підключаються до шини SDA за правилом монтажного І.

Ведучий може починати пересилку даних, лише якщо шина вільна. Два і більше ведучих можуть згенерувати сигнал СТАРТ, що приведе до сигналу СТАРТ на шині.

Арбітраж відбувається на шині SDA в періоди, коли шина SCL знаходиться у ВИСОКОМУ стані. Якщо один ведучий передає на лінію даних НИЗЬКИЙ рівень, тоді як інший - ВИСОКИЙ, то останній відключається від лінії, оскільки стан SDL (НИЗЬКИЙ) не відповідає ВИСОКОМУ стану його внутрішньої лінії даних.

Арбітраж може продовжуватися впродовж декількох біт. Оскільки спочатку передається адреса, а потім дані, то арбітраж може тривати до закінчення адреси, а якщо ведучі адресують один і той же пристрій, то в арбітражі братимуть участь і дані. Унаслідок такої схеми арбітражу при зіткненні дані не втрачаються.

Ведучому, програвшому арбітраж, дозволяється видавати синхроімпульси на шину SCL до кінця байта, протягом якого був втрачений доступ.

Якщо в пристрій ведучого також вбудовані і функції веденого і він програв арбітраж на стадії передачі адреси, то він негайно повинен перемкнутися в режим веденого, оскільки ведучій, що виграв арбітраж, міг адресувати його.

Рис. 1.20 іллюструє процедуру арбітражу два ведучих. В момент часу, коли виявляється відмінність між рівнем внутрішньої лінії даних і SDA, вихід першого ведучого набуває ВИСОКОГО значення, не впливаючи таким чином на пересилку даних ведучого, що виграв.

Унаслідок того, що арбітраж залежить лише від адреси і даних, передаваних ведучими, що змагаються, не існує центрального ведучого, а також пріоритетного доступу до шини.

Особливу увагу слід звернути на ситуацію, коли під час арбітражної процедури на шину передається повторний сигнал СТАРТ або сигнал СТОП. Якщо існує можливість виникнення такої ситуації, то ведучі повинні послати

повторний сигнал СТАРТ або сигнал СТОП в одних і тих же позиціях кадру. Іншими словами, арбітраж заборонений між:

- повторним сигналом СТАРТ і бітом даних;
- сигналом СТОП і бітом даних;
- повторним сигналом СТАРТ і сигналом СТОП.

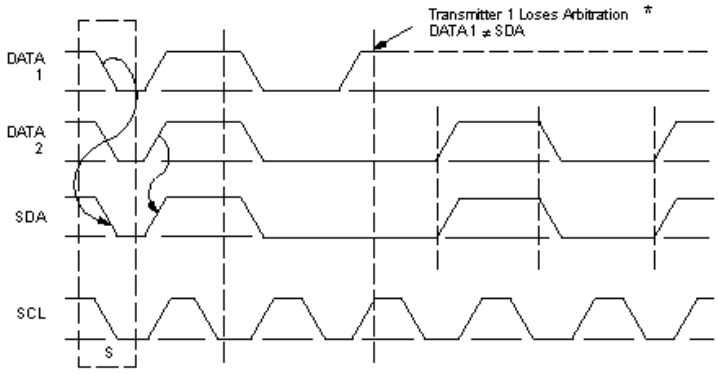


Рис 1.20. Арбітраж між двома ведучими

* передавач 1 програє арбітраж - його лінія даних не збігається з SDA

Використання механізму синхронізації як процедури управління зв'язком

Окрім використання в процедурі арбітражу, механізм синхронізації може бути використаний приймачами як засіб управління пересилкою даних на байтовому і бітовому рівнях.

На рівні байта, якщо пристрій може приймати байти даних з великою швидкістю, але вимагає певний час для збереження прийнятого байта або підготовки до прийому наступного, то він може утримувати лінію SCL в НИЗЬКОМУ стані після прийому і підтвердження байта, переводячи таким чином передавач в стан чекання.

На рівні бітів пристрій, такий як мікроконтроллер без вбудованих апаратних ланцюгів I2C або з обмеженими ланцюгами, може уповільнити частоту синхроімпульсів шляхом продовження їх НИЗЬКОГО періоду. (рис. 1.21). Таким чином швидкість передачі будь-якого ведучого адаптується до швидкості повільного пристрою.

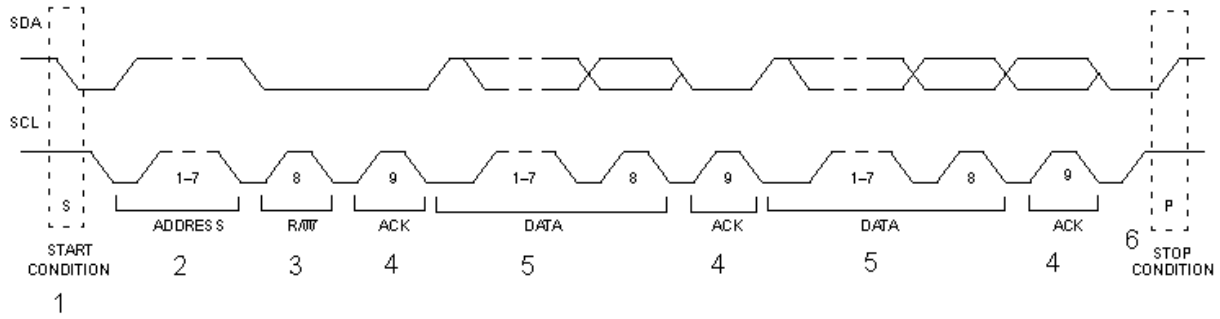


Рис. 1.21. Посилка даних

1. Сигнал СТАРТ
2. Адреса
3. Біт напряму ($R/\bar{W}$)
4. Підтвердження (ACK)
5. Дані
6. Сигнал СТОП

Формати з 7-бітовою адресою

Посилки даних відбуваються у форматі, показаному на рис. 1.22. Після сигналу СТАРТ посилається адреса веденого. Після 7 біт адреси слідує біт напрямку даних ($R/\bar{W}$), “нуль” означає передачу (запис), а “одиниця” - прийом (читання). Пересилка даних завжди закінчується сигналом СТОП, що генерується ведучим. Проте, якщо ведучий бажає залишатися на шині далі, він повинен видати повторний сигнал СТАРТ і потім адреса наступного пристрою. При такому форматі посилки можливі різні комбінації читання/запису:

Ведучий-передавач передає дані веденому приймачу. Напрям пересилки даних не змінюється (рис 1.22);

Ведучий читає дані веденого негайно після пересилки першого байта (рис. 1.23). У момент першого підтвердження ведучий-передавач стає ведучим-приймачем і ведений приймач стає веденим передавачем. Підтвердження проте генерується веденим. Сигнал СТОП генерується ведучим.

Комбінований формат (рис. 1.24). При зміні напряму пересилки даних повторюється сигнал СТАРТ і адреса веденого, але біт напряму даних інвертується. Якщо ведучий-приймач посилає повторний сигнал СТАРТ, він зобов'язаний заздалегідь послати сигнал непідтвердження.

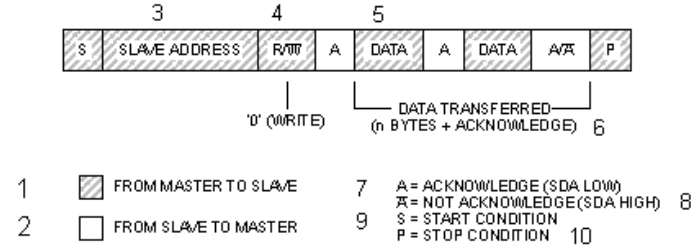


Рис. 1.22. Формат посилки від ведучого-передавача, що адресує ведений-приймача 7-бітовою адресою.

1. Напрям пересилки не змінюється
2. Напрямок передачі від ведучого до веденого
3. Напрямок передачі від веденого до ведучого
4. Адреса веденого
5. Біт напряму (в даному випадку $R/\bar{W}$=0 - запис)
6. Дані, що пересилаються (n байт + підтвердження)
7. A - Підтвердження
8. $A/\bar{A}$ - Непідтвердження
9. Сигнал СТАРТ
10. Сигнал СТОП

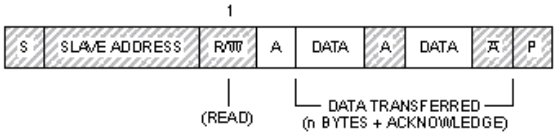


Рис. 1.23. Формат посилки від ведучого, що читає з веденого безпосередньо після першого байта

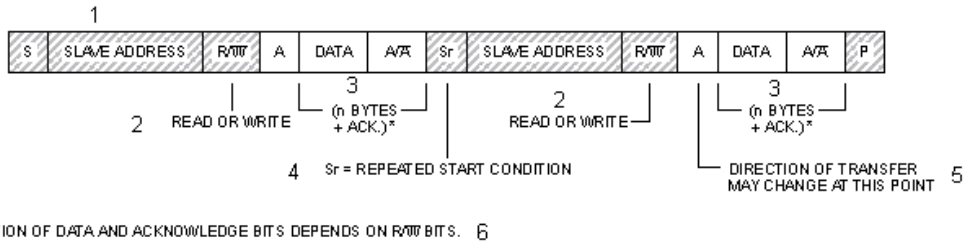


Рис. 1.24. Комбінований формат

1. Адреса веденого
2. Читання або запис
3. n байтів + підтвердження
4. Сигнал повторного СТАРТУ Sr
5. Напрям пересилки може змінитися в цій точці
6. Напрям пересилки даних і бітів підтвердження залежить від бітів напряму

Комбіновані формати можуть бути використані, наприклад, для управління послідовною пам'яттю. Під час першого байта даних можна передавати адресу в пам'яті, яка записується у внутрішній буфер. Після повторення сигналу СТАРТу (Sr) і адреси веденого дані видаються з пам'яті.

Всі рішення про авто-інкремент або декремент адреси, до якої стався попередній доступ, приймаються розробником пристрою.

Кожен байт завершується бітом підтвердження/непідтвердження, позначеним A або Ā відповідно.

I2C-сумісні пристрої повинні скидати логіку шини при виявленні сигналу СТАРТ або повторний СТАРТ і готуватися до прийому адреси.

7-бітова адресація

Процедура адресації на шині I2C полягає в тому, що перший байт посилки від ведучого після сигналу СТАРТ визначає, який ведений буде вибирається. Виключенням є адреса “Загального виклику”, що адресує всі пристрої на шині.

Перші сім бітів першого байта утворюють адресу веденого (рис 1.29). Восьмий, молодший біт, визначає напрям пересилки даних. “Нуль” означає, що ведучий записуватиме інформацію у вибраний ведений. “Одиниця” означає, що ведучий прочитуватиме інформацію з веденого.

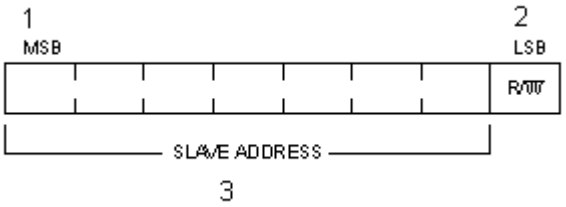


Рис. 1.25. Перший байт після сигналу СТАРТ

1. Старший розряд
2. Молодший розряд
3. Адреса веденого

Після того, як адреса послана, кожен пристрій в системі порівнює перші сім біт, що надійшли після сигналу СТАРТ зі своєю адресою. При збігу адрес пристрій вважає себе вибраним як ведений приймач або як ведений передавач, залежно від біта напряму $R/\bar{W}$.

Адреса веденого може складатися з фіксованої і програмованої частин. Ймовірно, що в системі буде декілька таких однакових пристроїв, тому за допомогою програмованої частини адреси стає можливим підключити до шини максимально можливу кількість таких пристроїв. Кількість програмованих біт в адресі залежить від кількості вільних виводів мікросхеми. Наприклад, якщо пристрій має 4 фіксованих і 3 програмованих адресних бітів, всього 8 однакових пристроїв може бути підключені до шини.

Комітет I2C координує виділення I2C адрес, що зведені в табл. 1.4. Комбінація біт 11110XX адреси зарезервована для 10-бітової адресації.

Таблиця 1.4 - Адресація I2C приладів

Адреса	$R/\bar{W}$	Опис
0000000	0	Адреса загального виклику
0000000	1	Байт СТАРТУ
0000001	X	Адреса CBUS
0000010	X	Адреса, зарезервована для шин іншого формату
0000011	X	Зарезервована для подальшого використання
00001XX	X	Зарезервована для подальшого використання
11111XX	X	Зарезервована для подальшого використання
11110XX	X	10-бітова адресація

Пристроям забороняється підтверджувати прийом байта СТАРТУ.

Адреса CBUS зарезервована для того, щоб можна було використовувати CBUS-сумісні і I2C-сумісні пристрою в одній системі. I2C-сумісні пристроям забороняється реагувати на прийом цієї адреси.

Адреса, зарезервована для шин іншого формату також призначеня для змішаного використання різних протоколів. Відповідати на прийом цієї адреси можуть лише пристрої, що працюють з цим форматом.

Комбінація біт 11110XX адреси зарезервована для 10-бітової адресації.

Адреса загального виклику

Адреса загального виклику відноситься до всіх пристроїв на шині для одночасного звернення до них. Проте, якщо пристрою не потрібні дані, які можуть бути передані по загальному виклику, він може ігнорувати звернення шляхом невидачі підтвердження. Якщо пристрою потрібні дані загального виклику, він генерує підтвердження і стає веденим приймачем. Другий і подальший байти повинні підтверджуватися кожним веденим приймачем, здатним обробити ці дані. Якщо ведений не може обробити один з байтів, він не генерує підтвердження.

Значення посилки загального виклику завжди визначається другим байтом (рис. 1.26).

Можливі наступні варіанти.

I. Біт напрямку $R/\bar{W}$ дорівнює “0”, другий байт приймає наступні значення:

1) 00000110, що означає: «Скинути пристрій і записати програмовану частину адреси». При виявленні цього байту всі пристрої скидаються і перечитують програмовану частину їх адрес. Перед видачею команди необхідно переконатися, що пристрої після подачі живлення не утримують лінії шини в низькому стані

2)00000100 - «Записати програмовану частину адреси». Всі пристрої, що мають можливість зміни програмованої частини адреси, запам'ятовують поточне значення адреси при прийнятті цієї команди. Пристрої не скидаються.

3)00000000. Цей код недопустимий для використання в якості другого байту.

4)Інші коди не встановлені і пристрої повинні ігнорувати їх.

II. Біт напрямку $R/\bar{W}$ дорівнює "1" , а двобайтова послідовність являє собою так званий "Апаратний загальний виклик". Він використовується у тих випадках, коли апаратний ведучий пристрой (такий, як сканер клавіатури), який не може бути запрограмован на видачу конкретної адреси веденого. Оскільки апаратний ведучий не знає, якому пристрою передається посилка, він може лише згенерувати апаратний загальний виклик і свою власну адресу - ідентифікуючи себе для системи (рис. 1.27).

Сім старших бітів другого байта містять адресу апаратного ведучого. Ця адреса розпізнається розумним пристроєм (мікроконтролером), який потім отримуватиме інформацію від апаратного ведучого. Якщо останній також може працювати як ведений, його адреса збігається з адресою ведучого.

III. В деяких системах апаратний ведучий-передавач встановлюється в режим веденого приймача відразу після скидання. При цому ведучий, що конфігурує систему, може повідомити апаратного ведучого (який знаходиться в режимі ведений-приймач), на яку адресу повинні посилатися дані (рис. 1.28.) Після цієї процедури апаратний ведучий залишається в режимі ведучий-передавач.

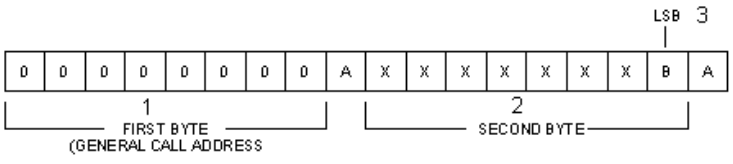


рис. 1.26. Формат адреси загального виклику

1. Перший байт

2. Другий байт
3. Молодший розряд

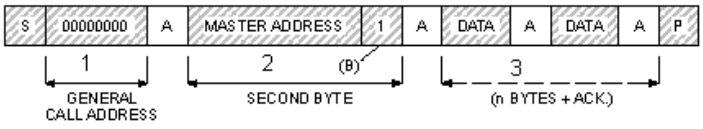


рис. 1.27. Пересилка даних з апаратного ведучого-передавача

1. Адреса загального виклику
2. Другий байт
3. N байт + підтвердження

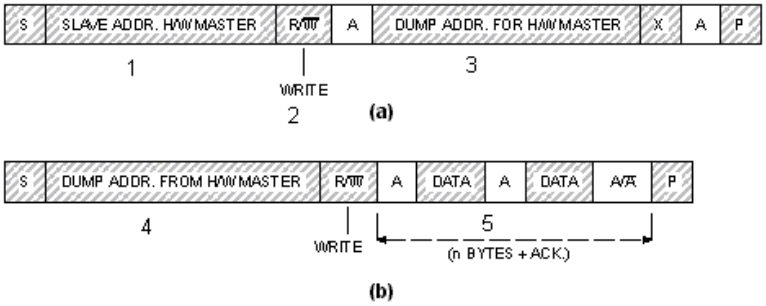


Рис. 1.28. Передача даних апаратним ведучим, здатним пересилати дані безпосередньо веденим пристроям.

(a) Конфігуруючий ведучий посилає адресу початку даних до апаратного ведучого;
(b) Апаратний ведучий посилає дані вибраному веденому.

1. Адреса апаратного ведучого
2. Запис
3. Адреса початку даних для апаратного ведучого
4. Адреса початку даних від апаратного ведучого
5. n байт + підтвердження

Байт СТАРТУ

Мікроконтролери можуть бути підключені до шини I2C двома способами:

Мікроконтролер зі вбудованими шинними ланцюгами реагує за допомогою переривань на події, що відбуваються на шині,

Мікроконтроллер без таких ланцюгів постійно відстежує стан шини програмним способом.

Вочевидь, що чим більше часу процесор витрачає на обслуговування шини, тим менше у нього залишається на основне завдання. Внаслідок цього виникає різниця між швидкими апаратними пристроями і повільними мікроконтролерами, що використовують програмне опитування. В цьому випадку посилка даних може починатися із стартової процедури, яка багато довше, ніж звичайний сигнал СТАРТ, але дає можливість мікроконтролеру другого типу підготуватися до початку передачі.

Процедура старту (рис. 1.29) складається з

- Сигналу СТАРТ
- Байта СТАРТУ
- Імпульсу підтвердження
- Повторного сигналу СТАРТ

Ведучий-передавач після звичайного сигналу СТАРТ передає байт СТАРТУ (00000001). Вкдений мікроконтролер може відстежувати лінію SDA з меншою частотою, поки не виявить послідовність з семи нулів, по суті - НИЗЬКИЙ рівень на SDA впродовж семи тактових імпульсів. Після виявлення цієї послідовності мікроконтролер може перемкнутися на вищу частоту опиту шини, для того, щоб виявити повторний сигнал СТАРТ.

Апаратний приймач I2C скинеться при прийомі повторного сигналу СТАРТ і тому проігнорує байт СТАРТУ.

Після байта СТАРТУ генерується тактовий імпульс для підтвердження. Він присутній лише для сумісності з форматом байта. Пристроям забороняється підтверджувати прийом байта СТАРТУ.

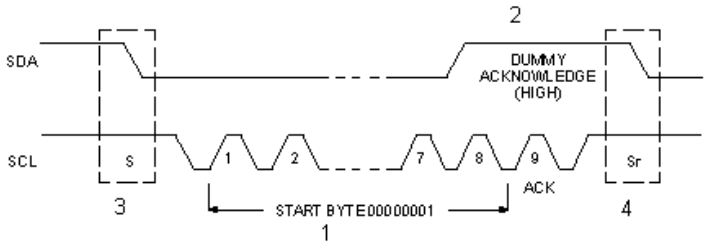


Рис. 1.29. Процедура байта СТАРТУ

1. Байт СТАРТУ
2. Фальшиве підтвердження (ВИСОКИЙ стан лінії SDA)
3. Сигнал СТАРТ (S)
4. Сигнал повторний СТАРТ (Sr)

Сумісність з CBUS

Приймачі CBUS можуть бути підключені до шини I2C, але при цьому має бути введена третя лінія DLEN, а біт підтвердження не використовується. Зазвичай посилки I2C складаються з 8-бітових слів, тоді як CBUS-сумісні пристрої мають інший формат.

У змішаній шині I2C-сумісні пристрої не повинні відповідати на посилки формату CBUS. Для цього зарезервована спеціальна адреса CBUS (0000001X). Після передачі адреса CBUS лінія DLEN стає активною і посилається дані формату CBUS (рис. 1.30). Після сигналу СТОП всі пристрої знову готові приймати дані.

Ведучі-передавачі можуть посилати дані в CBUS форматі після посилки адреси CBUS. Передача закінчується сигналом СТОП, розпізнаваним всіма пристроями.

Зазначимо, що якщо конфігурація шини CBUS відома і розширення CBUS-сумісних пристроїв не передбачається, конструктору дозволяється встановлювати час утримання лінії DLEN, керуючись конкретними вимогами використовуваних пристроїв.

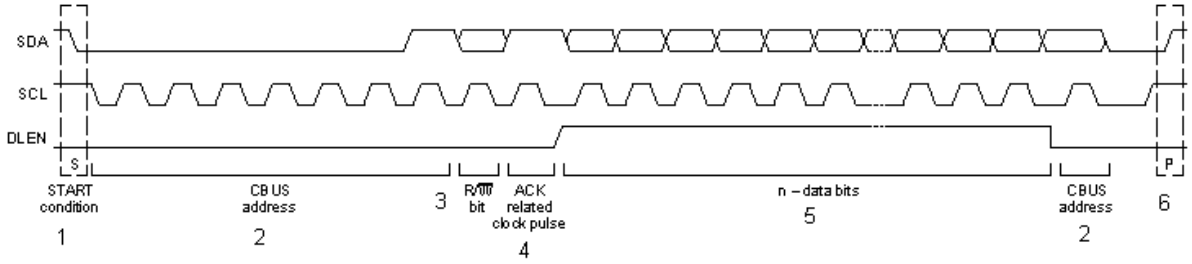


Рис. 1.30. Формат даних при посилках з CBUS передавачем/приймачем

1. Сигнал СТАРТ
2. Адреса CBUS
3. Біт напряму передачі $R/\bar{W}$
4. Синхроімпульс підтвердження
5. n біт даних
6. Сигнал СТОП

Підключення I2C пристроїв

Розрізняють I2C-пристрої з фіксованими та залежними від напруги вхідними рівнями.

Вхідні рівні визначаються таким чином:

- шумовий кордон НИЗЬКОГО рівня є 0.1 від напруги живлення
- шумовий кордон Високого рівня є 0.2 від напруги живлення

I2C-пристрої з *фіксованими* вхідними рівнями 1.5 В і 3 В можуть мати свою власну живлячу напругу. Підтягуючі резистори мають бути підключені до джерела 5 В ± 10% (рис. 1.31).

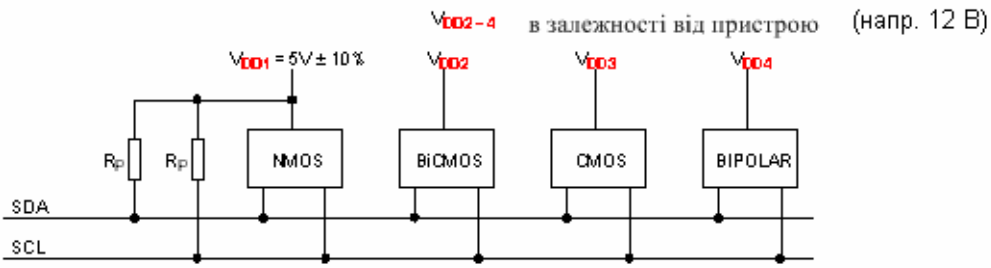


Рис. 1.31. Підключення пристроїв з фіксованим вхідним рівнем до шини I2C

I2C-устройства з вхідними рівнями, *залежними від напруги* живлення повинні мати одну загальну лінію живлення, до якої також має бути підключений підтягуючий резистор (рис. 1.32).

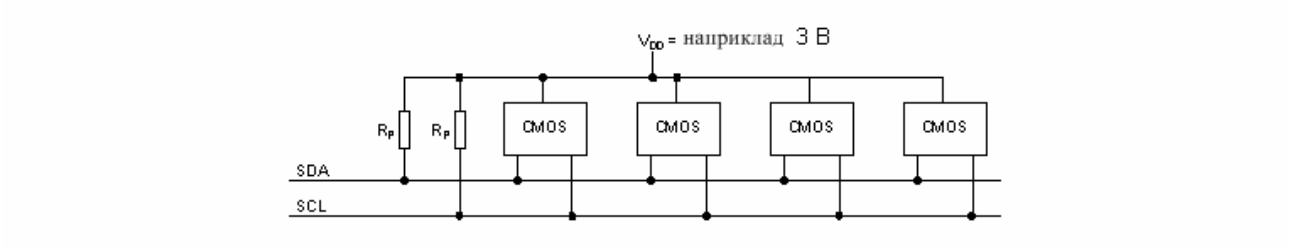


Рис 1.32. Підключення пристроїв з широким діапазоном живлення до шини I2C

Якщо пристрої з фіксованим вхідним рівнем використовуються сумісно з пристроями з відносним вхідним рівнем, останні мають бути підключені до однієї загальної лінії живлення 5 В ±10% і повинні мати підтягуючі резистори, підключені до SDA і SCL контактам як показано на рис 1.37.

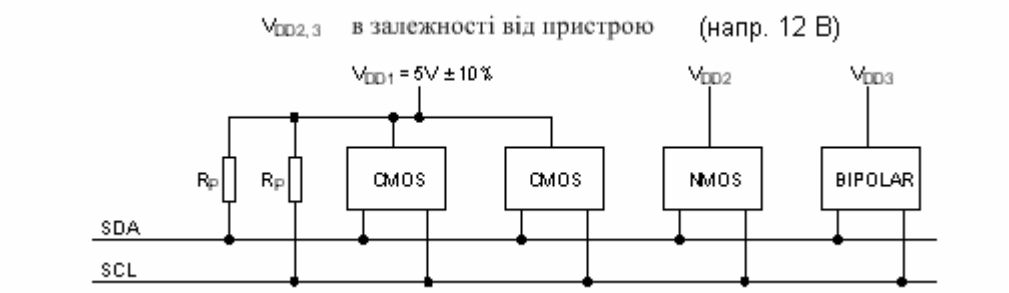


Рис. 1.33. Підключення пристроїв з відносним (Vdd1) рівнем вхідної напруги і фіксованим вхідним рівнем (Vdd2-4) до шини I2C

Для захисту від високовольтних викидів напруги на лініях шини рекомендовано включати послідовно резистори Rs (наприклад, 300 Ом), рис. 1.34

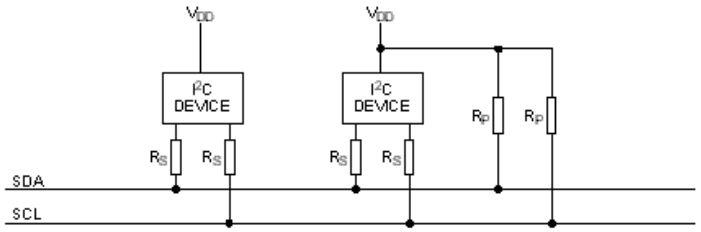


Рис. 1.34. Послідовні резистори Rs для захисту від високовольтних викидів

Доповнення до специфікації шини I2C

Шина I2C із швидкістю передачі даних 100 кбіт/с і 7-бітовою адресою існує вже впродовж 10 років в незмінному вигляді. Концепція прийнята повсюдно як стандарт для сотень типів мікросхем, що випускаються фірмою Philips і іншими постачальниками. Однак розвиток I2C пристроїв призвів до того, що:

Нові застосування потребують пересилки великих об'ємів інформації, отже потрібна велика пропускна спроможність шини. Покращувана технологія виробництва мікросхем дозволила в чотири рази збільшити швидкість передачі даних без зміни собівартості виробу.

Більшість з 112 адрес, допустимих при 7-бітовій адресації, вже були використані більш ніж один раз. Для запобігання проблемам з розміщенням адрес нових пристроїв, бажано мати більшу кількість адресних комбінацій. Десятиразове збільшення кількості доступних адрес отримано при використанні нової 10-бітової адресації.

В даний час специфікація шини I2C доповнена наступними положеннями:

- Швидкий режим, що дозволяє в чотири рази збільшити швидкість передачі даних
- 10-бітова адресація, що дозволяє використовувати 1024 додаткових адрес

Все нові пристрої з I2C інтерфейсом працюють в швидкому режимі. Переважно, вони повинні вміти приймати і передавати дані на швидкості 400 кбіт/с. Як мінімум вони мають бути здатні входити в синхронізацію в швидкому режимі, з тим аби понизити швидкість передачі (шляхом подовження НИЗЬКОГО періоду SCL) до допустимої величини. Швидкі пристрої мають бути сумісні «знизу», що означає їх здатність працювати із стандартними пристроями по повільній шині.

Вочевидь, що стандартні пристрої не здатні працювати в швидкій шині, тому що вони не можуть синхронізуватися на високій швидкості і їх стан стане непередбачуваним.

Ведені швидкі пристрої можуть володіти як 7-бітовою, так і 10-бітовою адресою. Проте, 7-бітова адреса переважніша, оскільки його апаратна реалізація простіша і довжина посилки менша.

Пристрої з 7-бітовою і 10-бітовою адресами можуть одночасно використовуватися на одній шині, незалежно від швидкості передачі.

. *Швидкий режим*

У швидкому режимі протокол, формат, логічні рівні і максимальне ємкісне навантаження ліній шини залишається незмінним. Зміни в специфікації такі:

- Максимальна швидкість передачі зросла до 400 кбіт/с;
- Не вимагається сумісності з CBUS-пристроями, оскільки вони не можуть працювати на високих швидкостях;
- Вхідні ланцюги швидких пристроїв повинні мати вбудоване придушення викидів і тригер Шмітта на обох лініях;
- Вихідний буфер швидких пристроїв повинен мати каскад з управлінням часом заднього фронту ліній SDA і SCL;

Якщо джерело напруги живлення швидких пристроїв вимикається, лінії повинні переходити в третій стан.

Зовнішні підтягуючі пристрої, підключені до ліній шини мають бути змінені для забезпечення допустимого часу наростання переднього фронту. Для навантажень шини до 200 пФ це підтягуючий пристрій може бути простим резистором, а для навантажень від 200 пФ до 400 пФ це має бути джерело струму (3 мА максимум) або схема на перемиканих резисторах.

10-бітова адресація.

10-бітова адресація не міняє формат шини. Для цього використовується зарезервована адресна комбінація 1111XXX перших семи біт першого байта. 10-бітова адресація не впливає на існуючу 7-бітову адресацію. Пристрої з 7-бітовою і 10-бітовою адресацією можуть бути підключені до однієї шини.

Хоча є вісім можливих комбінацій послідовності 1111XXX, з них використовуються лише чотири - 11110XX. Комбінації типа 11111XX зарезервовані для подальших удосконалень шини.

10-бітова адреса формується з перших двох байтів. Перші сім біт першого байта є комбінацією вигляду 11110XX, де два молодших біта (XX) є двома старшими (9 і 8) бітами 10-бітової адреси; восьмий біт першого байта - біт напряму. “Нуль” в цьому біті означає, що ведучий збирається записувати інформацію у веденого, а “одиниця” - що ведучий прочитуватиме інформацію з веденого.

Якщо біт напряму дорівнює “нулю”, то другий байт містить молодші 8 біт 10-бітової адреси. Якщо біт напряму дорівнює “одиниці”, то наступний байт містить дані, передані з веденого ведучому.

Можливі різні комбінації форматів 10-бітових посилок:

1) *Передача даних від ведучого до веденому.* (рис 1.35.). Напрям пересилки не міняється. Коли за сигналом СТАРТ з'являється початок 10-бітової адреси, кожен ведений на шині порівнює перші сім біт першого байта зі своєю власною адресою і упевняється, що біт напряму дорівнює “нулю”.

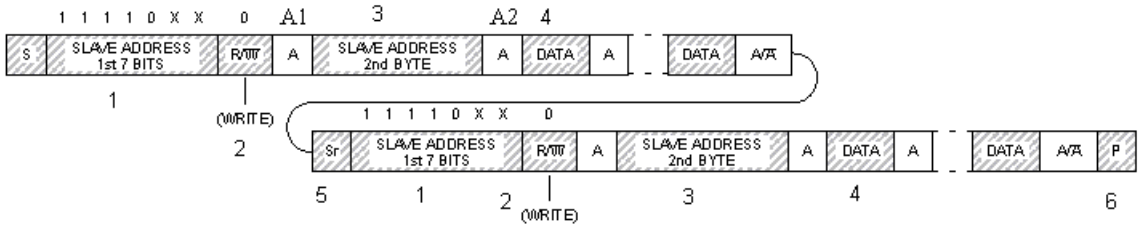


Рис. 1.35. Ведучий-передавач адресує веденого приймача 10-бітовою адресою

1. Адреса веденого (перші 7 біт)
2. Запис $R/\bar{W}=0$
3. Адреса веденого (другий байт)

Допустимо, що більш, ніж один пристрій виявили збіг і згенерували сигнал підтвердження (A1). Всі ведені з адресами, що збіглися, продовжують порівнювати подальші 8 біт адреси, і лише один пристрій виявляє збіг і генерує підтвердження (A2). Цей пристрій залишається вибраним, поки

ведучий не пошле сигнал СТОП або сигнал повторного СТАРТУ з іншою адресою.

2) *Прийом даних ведучим від веденого*. Ведучий-приймач приймає дані від веденого передавача. Напрям передачі міняється після другого біта напряму (рис. 1.36). Процедура ідентична вищеописаній до моменту другого підтвердження (A2). Далі передається сигнал повторного СТАРТУ. Вибраний ведений пам'ятає, що був адресований раніше. Цей ведений порівнює перші сім біт адреси зі своєю адресою, а також упевняється, що біт напряму дорівнює “одиниці”. При збігу ведений вважає, що він адресований як передавач і генерує підтвердження (A3). Ведений передавач залишається адресованим до приходу сигналу СТОП або сигналу повторного СТАРТУ з іншою адресою. Після сигналу повторного СТАРТУ всі інші пристрої також порівнюють перші сім біт зі своєю адресою і перевіряють біт напряму. Проте, жодне з них не адресується, оскільки біт напряму дорівнює “одиниці” (для 10-бітових пристроїв), або такої адреси не існує (для 7-бітових пристроїв).

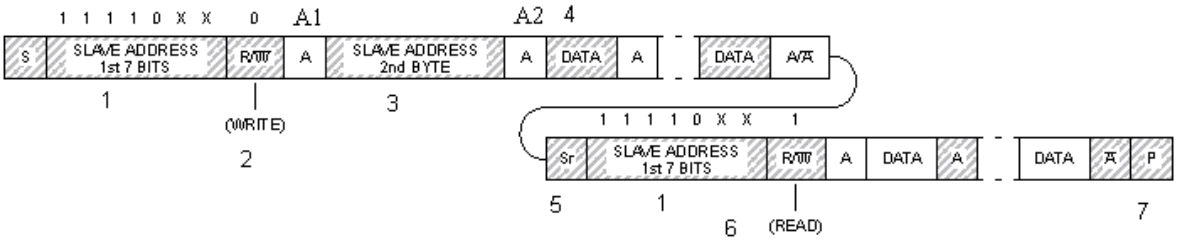


Рис. 1.36. Ведучий-приймач адресує веденого передавача 10-бітовою адресою

1. Адреса веденого (перші 7 біт)
2. Запис $R/\bar{W}=0$
3. Адреса веденого (другий байт)
4. Дані
5. Сигнал повторний СТАРТ
6. Адреса веденого (перші 7 біт та біт читання $R/\bar{W}=1$)
7. Сигнал СТОП

3) *Комбінований формат.* Ведучий передає дані веденому, а потім читає дані з цього ж веденого (рис 1.37). Один ведучий займає шину на весь час пересилки. Напрям пересилки міняється після другого біта напряму

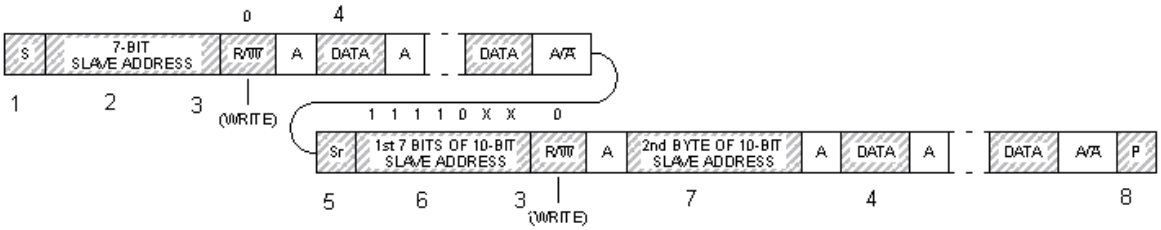


Рис. 1.37. Комбінований формат. Ведучий адресує веденого 10-бітовою адресою, потім передає йому дані і приймає з нього дані.

1. Старт
2. Адреса веденого (перші 7 біт)
3. Запис $R/\bar{W}=0$
4. Дані
5. Повторний СТАРТ
6. 7 чи 10 біт веденого
7. Другий байт адреси веденого
8. СТОП

Комбіновані формати можуть бути використані, наприклад, для управління послідовною пам'яттю. Під час першого байта даних можна передавати адресу пам'яті, яка записується у внутрішній регістр. Після повторення сигналу СТАРТу і адреси веденого видаються дані з пам'яті.

ЛЕКЦІЯ 4

Двопровідний послідовний інтерфейс TWI

Основні характеристики інтерфейсу TWI (Two Wire Interface)

Двопровідний послідовний інтерфейс (TWI) сумісний з протоколом I2C компанії Philips. Цей шинний інтерфейс був розроблений для організації простого, надійного і недорогого зв'язку між інтегральними схемами в електронних пристроях. Сильними сторонами шини TWI є можливість

адресації до 128 мікросхем на одній шині, арбітраж і можливість роботи декількох ведучих пристроїв (майстрів) на одній шині.

Апаратний модуль TWI входить до складу багатьох мікроконтролерів AVR.

Основні характеристики двопровідного інтерфейсу наступні:

- Гнучкий, простий, але при цьому ефективний послідовний комунікаційний інтерфейс, що вимагає лише дві лінії зв'язку;
- 7-розрядна адресація, щодозволяє підключити до шини до 128 підлеглих пристроїв;
- Підтримка багатомайстерного арбітрирування;
- Можливість обмеження швидкісті зміни сигналів на виходах драйверів ;
- Підвищує стійкість до викидів на лініях шини завдяки роботі схеми завадопридушення;
- Програмована адреса для підлеглого режиму з підтримкою загального виклику;
- Пробудження мікроконтроллера з режиму сну при визначенні заданої адреси на шині;
- Сі-код драйвера для ведучого TWI;
- Сумісність з протоколом I2C компанії Philips;
- Використання апаратного модуля TWI;
- Передача з використанням переривань;
- Підтримка стандартного 100 кбод і прискореного 400 кбод режимів.

Визначення шини TWI

Двопровідний послідовний інтерфейс TWI вирішує типові задачі зв’язку в мікроконтролері. Протокол TWI дозволяє проектувальникові системи зовні зв'язати до 128 різних пристроїв через двопровідну шину, де одна лінія - лінія синхронізації SCL і одна - лінія даних SDA. Для реалізації шини необхідний резистор, що підтягує до плюса живлення, на кожній лінії шини (рис 1.38.).

Всі пристрої, які підключені до шини, мають свою індивідуальну адресу, а механізм визначення вмісту шини підтримується протоколом TWI.

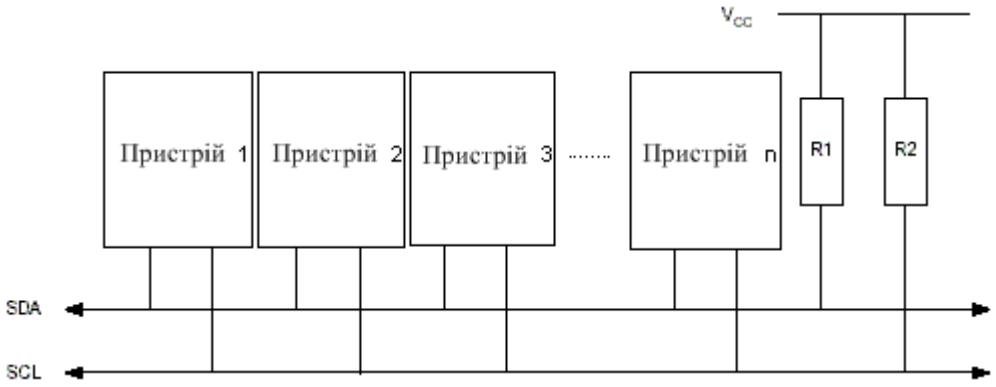


Рис. 1.38. Зовнішні підключення до шини TWI

Використання модуля TWI як ведучого інтерфейсу I2C

Модуль TWI складається з декількох підмодулів (рис. 1.39). Всі регістри, які виділені жирною лінією, доступні через шину даних мікроконтролера.

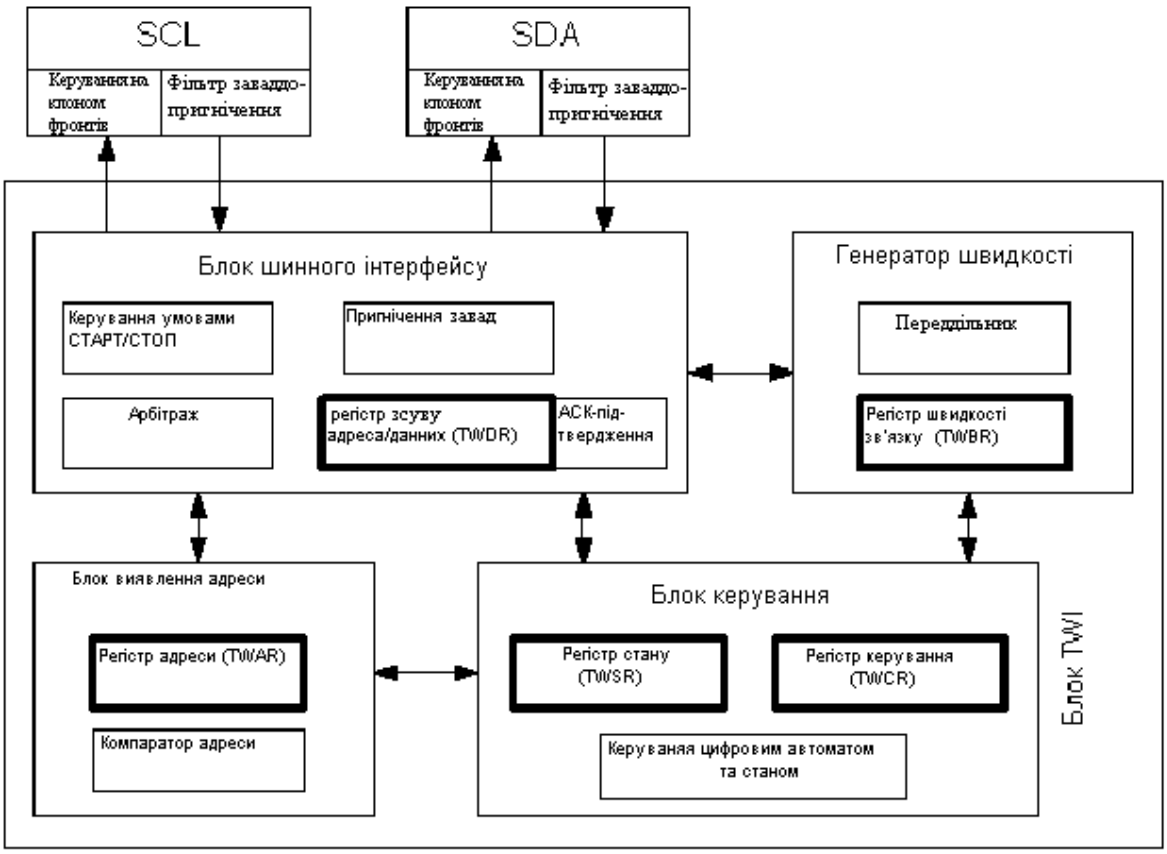


Рис. 1.39. Функціональна схема модуля TWI

Схема містить:

- драйвери виводів SCL й SDA;
- блок генератора швидкості зв‘язку;
- блок шинного інтерфейсу;
- блок виявлення адреси;
- блок керування.

Драйвери виводів SCL й SDA

Виводи SCL та SDA зв'язують двопровідний інтерфейс мікроконтролера з іншими мікроконтролерами в системі. Драйвери виходів містять обмежувач швидкості зміни фронтів для виконання вимог до TWI. Вхідні каскади містять блок придушення завад, завдання якого полягає в ігноруванні імпульсів тривалістю менше 50 нс. До кожної з цих ліній можна підключити

внутрішній підтягуючий резистор шляхом установки розрядів PORTD.0 (SCL), PORTD.1 (SDA). Використання вбудованих підтягуючих резисторів у ряді випадків дозволяє відмовитися від вживання зовнішніх.

Блок генератора швидкості зв'язку.

Даний блок управляє періодом імпульсів SCL в режимі ведучого пристрою. Період SCL задається регістром швидкості TWI (TWBR) і значенням біту управління переддільником в регістрі стану TWI (TWSR). У підлеглому режимі значення швидкості або установки переддільника не впливають на роботу, але частота синхронізації підлеглого пристрою (МК) має бути мінімум в 16 разів вище за частоту SCL. Підлеглі пристрої можуть подовжувати тривалість низького рівня на лінії SCL, тим самим зменшуючи середню частоту синхронізації шини TWI. Частота SCL визначається як:

$$f_{SCL} = \frac{f_Q}{16 + 2 \cdot (TWBR) \cdot 4^{TWPS}},$$

f_Q - частота кварцового генератора МК;

TWBR - значення регістра швидкості TWI;

TWPS - значення бітів переддільника в регістрі стану TWI.

Блок шинного інтерфейсу.

Даний блок містить регістр зсуву адреси і даних (TWDR), контролер умов СТАРТа/СТОПа і схему арбітражу. Регістр TWDR містить передаваний байт адреси або даних, або прийнятий байт адреси або даних. Окрім 8-розрядного регістру TWDR, до складу блоку шинного інтерфейсу також входить регістр, що зберігає значення передаваного або прийнятого біта підтвердження/не підтвердження (A/Ā). До даного регістра немає прямого доступу з боку програмного забезпечення. Проте під час прийому він може встановлюватися або скидатися шляхом маніпуляцій з регістром управління TWCR. У режимі передавача значення прийнятого біта A/Ā можна визначити за значенням регістра TWSR.

Контролер СТАРТа/СТОПа відповідає за генерацію і детектування умов СТАРТ, ПОВТОРНИЙ СТАРТ і СТОП. Контролер СТАРТа/СТОПа дозволяє виявити умови СТАРТ і СТОП, навіть якщо мікроконтролер знаходиться в одному з режимів «сну». Цим забезпечується можливість пробудження мікроконтролера за запитом шини від веденого.

Якщо TWI ініціював передачу як ведучий, то схема арбітражу безперервно контролює передачу, визначаючи можливість подальшої передачі. Якщо TWI втрачає арбитраж, то блок формує відповідний сигнал блоку управління, який виконує адекватні дії і генерує відповідний код стану.

Блок виявлення адреси.

Блок виявлення адреси перевіряє чи рівна прийнята адреса значенню 7-розрядної адреси з регістра TWAR. Якщо встановлений біт дозволу виявлення загального виклику TWGCE в регістрі TWAR, то всі вхідні адресні біти додатково порівнюватимуться з адресою загального виклику. При адресному збігу подається сигнал блоку управління, що дозволяє виконати йому необхідні дії. Залежно від установки регістра TWCR підтвердження адреси TWI може відбувається або ні. Блок виявлення адреси здатний функціонувати навіть, коли мікроконтроллер переведений в режим «сну», тим самим дозволяючи відновити нормальну роботу мікроконтролера по запиту від майстра шини. Якщо при адресному збігу TWI в економічному режимі мікроконтролера, виникає інше переривання (наприклад, INT0), то TWI припиняє роботу і повертається до стану холостого ходу (Idle). Якщо виникнення даного ефекту небажане, то необхідно стежити, аби під час виявлення адресації, коли мікроконтролер знаходиться в режимі виключення (Power-down), було дозволено лише одне переривання.

Блок керування

Блок керування спостерігає за шиною TWI і генерує відгуки відповідно до установок регістра управління TWCR. Якщо на шині TWI виникає подія, яка вимагає уваги з боку програми, то встановлюється прапор переривання

TWINT. Наступним тактом оновлюється вміст регістра статусу TWI - TWSR, в якому буде записаний код, що ідентифікує виниклу подію. Даная інформація зберігається в TWSR лише тоді, коли встановлений прапор переривання TWI. В регістрі TWSR міститься спеціальний код стану. До тих пір, поки встановлений прапор TWINT лінія SCL залишається в низькому стані. Цим забезпечується можливість завершити програмі всі завдання перед продовженням сеансу зв'язку.

- Прапор TWINT встановлюється в наступних ситуаціях:
- Після передачі умови СТАРТ/ПОВТОРНИЙ_СТАРТ;
- Після передачі адр_веденого+читання/запис;
- Після передачі адресного байта;
- Після втрати арбітражу;
- Після того, як TWI адресований власною підлеглою адресою або загальним викликом;
- Після прийому байта даних;
- Після прийому умови СТОП або ПОВТОРНИЙ_СТАРТ в режимі адресації веденого;
- Після виникнення помилки унаслідок некоректної умови СТАРТ або СТОП.

Опис регістрів TWI

Формат регістру швидкості зв'язку TWBR наступний:

Розряд	7	6	5	4	3	2	1	0
	TWBR 7	TWBR 6	TWBR 5	TWBR 4	TWBR 3	TWBR 2	TWBR 1	TWBR 0
Чит./запис	Чит./З п.	Чит./З п.	Чит./З п.	Чит./З п.	Чит./З п.	Чит./З п.	Чит./З п.	Чит./З п.
Вих. значення	0	0	0	0	0	0	0	0

TWBR задає коефіцієнт ділення частоти генератора швидкості зв'язку. Генератор частоти швидкості зв'язку - дільник частоти, який формує сигнал синхронізації SCL в режимах "Ведучий".

Формат регістру управління шиною TWCR наступний:

Розряд	7	6	5	4	3	2	1	0
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Чит./запис	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт.	Чт./Зп.	Чт.	Чт./Зп.
Вих. значення	0	0	0	0	0	0	0	0

Регістр TWCR призначений для управління роботою TWI. Він використовується для дозволу роботи TWI, для ініціації сеансу зв'язку; умови, що веде до генерації СТАРТ на шині; для генерації підтвердження прийому; для генерації умови СТОП і для остановки шини під час запису в регістр TWDR. Він також сигналізує про спробу помилкового запису в регістр TWDR, коли доступ до нього був заборонений.

Розряд 7 - TWINT: Прапор переривання TWI.

Даний біт встановлюється апаратно, якщо TWI завершує поточне завдання і чекає реакції програми. Якщо біт I в SREG і біті TWIE в TWCR встановлені, то мікроконтроллер переходить на вектор переривання TWI. Лінія SCL залишається в низькому стані, поки встановлений прапор TWINT. Прапор TWINT скидається програмно шляхом запису в нього логічні 1. Даний прапор скидається не автоматично при переході на вектор переривання. Також потрібно врахувати, що скидання даного прапора приводить до відновлення роботи TWI. З цього виходить, що програмне скидання даного прапора необхідно виконати після завершення опиту регістрів TWAR, TWSR і TWDR.

Розряд 6 - TWEA: Біт дозволу підтвердження.

Біт TWEA керує генерацією імпульсу підтвердження. Якщо в біт TWEA записана лог. 1, то імпульс ПІДТБ генерується на шині TWI, якщо виконується одна з наступних умов:

- Прийнята власна підлегла адреса;

- Прийнятий загальний виклик, коли встановлений біт TWGCE в регістрі TWAR;
- Прийнятий байт даних в режимі ведучого приймача або веденого приймача.

Запис лог. 0 в біт TWEA дозволяє тимчасово відключитися від двопровідної послідовної шини. Для відновлення розпізнавання адреси необхідно записати в даний біт лог.1.

Розряд 5 - TWSTA: Біт умови СТАРТ.

Програміст повинен встановити даний біт при необхідності стати ведучим на двопровідній послідовній шині. TWI перевіряє доступність шини і генерує умову СТАРТ, якщо шина вільна. Проте якщо шина зайнята, то TWI чекає появи умови СТОП, а потім генерує нову умову СТАРТ для перехоплення стану ведучого шини. TWSTA необхідне скидати програмно після передачі умови СТАРТ.

Розряд 4 - TWSTO: Біт умови СТОП.

Установка біта TWSTO в режимі ведучого призводить до генерації умови СТОП на двопровідній послідовній шині. Якщо на шині виконується умова СТОП, то біт TWSTO скидається автоматично. У режимі ведучого установка біта TWSTO може використовуватися для виходу з умови помилки. В цьому випадку умова СТОП не генерується, але інтерфейс TWI повертається до безадресного режиму ведучого і переводить лінії SCL і SDA в високоімпедансний стан.

Розряд 3 - TWWC: Прапор помилкового запису.

Біт TWWC встановлюється при спробі запису в регістр даних TWDR, коли TWINT має низький рівень. Прапор скидається при записі регістра TWDR, коли TWINT = 1.

Розряд 2 - TWEN: Біт дозволу роботи TWI.

Біт TWEN дозволяє роботу TWI і активізує інтерфейс TWI. Якщо біт TWEN встановлений, то TWI бере на себе функції управління лініями

введення-виведення SCL і SDA. При цьому дозволяється робота обмежувачів швидкості зміни фронтів і фільтрів завадопридушення. Якщо даний біт дорівнює нулю, то TWI відключається і всі передачі припиняються незалежно від стану роботи.

Розряд 1 - Резервний біт

Даний біт є резервним і прочитується як 0.

Розряд 0 - TWIE: Дозвіл переривання TWI

Якщо в даний біт записана лог. 1 і встановлений біт I в регістрі SREG, то запит на переривання TWI генеруватиметься доти, поки встановлений прапор TWINT.

Формат регістру стану TWSR наступний:

Розряд	7	6	5	4	3	2	1	0
	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0
Чит./запис	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	Чит./Зп.	Чит./Зп.
Вих. значення	1	1	1	1	1	0	0	0

Розряди 7..3 - TWS: Стан TWI

Дані 5 біт відображають стан логіки блоку TWI і двопровідної послідовної шини. Розряд 2 - біт є резервним і прочитується як 0.

Розряди 1..0 - TWPS: Біти переддільника TWI

Дані біти відрізняються повним доступом (Чит./запис) і дозволяють управляти переддільником швидкості зв'язку, згідно табл. 1.5.

Таблиця 1.5. - Переддільник швидкості зв'язку TWI

TWPS 1	TWPS 0	Значення передділення
0	0	1
0	1	4
1	0	16
1	1	64

Значення кодів статусу залежать від конкретних режимів роботи модуля TWI. Однак є два коди стану ($ F8 і $ 00), не пов'язані з яким-небудь

режимом роботи (табл. 1.6).

Стан з кодом $ F8 є проміжним. Наявність цього коду означає відсутність будь-якої інформації у зв'язку з тим, що прапор TWINT не встановлений. Код статусу $ 00 сигналізує про помилку на шині. Така помилка виникає при формуванні ведучим станів СТАРТ або СТОП в невідповідному місці, наприклад, під час передачі байта адреси, байта даних або біта підтвердження. Для виходу з таких ситуацій необхідно записати лог. 1 в розряди TWSTO і TWINT. В результаті модуль перейде в режим неадресованного ведучого, звільнить лінії SDA і SCL і скине прапор TWSTO. Стан СТОП в цій ситуації сформовано не буде.

Таблиця 1.6. - Коди стану, що не залежать від режиму TWI

Код статусу	Стан шини імодуля TWI	Дії програми					Наступна дія, що виконується модулем TWI
		в/із TWDR	в регістр TWCR				
			STA	STO	TWINT	TWEA	
$F8	Немає інформації; TWINT = «0»	Немає дій	Немає дій				Чекання встановлення прапору TWINT
$00	Помилка на шині в результаті некоректного формування станів СТАРТ або СТОП	Немає дій	0	1	1	X	Всі дії виконуються апаратно. Шина звільняться, а прапорTWSTO скидається в «0»

Інші коди стану залежать будуть розлянуті при опису режимів роботи модуля.

Формат регістру даних шини TWDR наступний:

Розряд	7	6	5	4	3	2	1	0
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0
Чит./запис	Чит./Зп.	Чит./Зп.	Чит./Зп.	Чит./Зп.	Чит./Зп.	Чит./Зп.	Чит./Зп.	Чит./Зп.
Вих. значення	1	1	1	1	1	1	1	1

У режимі передавача регістр TWDR містить наступний байт для передачі. У режимі приймача регістр TWDR містить останній прийнятий байт. Запис в регістр можливий лише коли TWI не виконує процес зсуву даних. Такий стан

настає, коли відбувається апаратна установка прапора переривання TWINT. Зазначимо що регістр даних не може ініціалізуватися користувачем перед виникненням першого переривання. Дані в регістрі TWDR залишаються стабільними доки встановлений біт TWINT. Під час зсуву даних послідовної передачі одночасно відбувається зсув для послідовного введення. Регістр TWDR завжди містить останній байт представлений на шині, виключаючи ситуацію відновлення нормальної роботи мікроконтролера за перериванням TWI. В цьому випадку стан TWDR є невизначеним. В разі втрати арбітражу шини дані, передавані від ведучого до веденого, не втрачаються. Управління бітом відбувається автоматично під управлінням схеми TWI, а МК безпосереднього доступу до біта підтвердження АСК не має.

Вміст регістру МК складає байт даних, який необхідно передати наступним, або останній прийнятий байт по двопровідній послідовній шині.

Регістр адреси веденого шини TWAR має формат:

Розряд	7	6	5	4	3	2	1	0
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Чит./запис	Чит./Зп.	Чит./Зп.	Чит./Зп.	Чит./Зп.	Чит./Зп.	Чит./Зп.	Чит./Зп.	Чит./Зп.
Вих. значення	1	1	1	1	1	1	1	1

Якщо TWI налаштований на режим веденого передавача або приймача, то реагуватиме лише на адресу, записану в цей регістр (у 7 старших розрядів TWAR). У режимах ведучого даний регістр не використовується. У багатомайстерних системах регістр TWAR встановлюється в тому ведучому, до якого адресуються як до веденого інші ведучі шини.

Молодший розряд регістра TWAR використовується для дозволу виявлення адреси загального виклику ($00). Спеціальний компаратор виконує порівняння підлеглої адреси (або адреси загального виклику) з прийнятою адресою. При збігу адрес генерується запит на переривання.

Послідовність обслуговування TWI при типовій передачі

TWI орієнтований на передачу даних в байтному форматі з управлінням за перериванням. Переривання виникають після виявлення одного з подій на шині, наприклад, прийом байта або передача умови СТАРТ. Управління TWI за перериванням дозволяє звільнити програмне забезпечення на виконання інших завдань під час передачі байта даних.Зазначимо, що установка прапора TWINT призводить до генерації запиту на переривання лише у тому випадку, коли встановлений біт дозволу переривання TWIE в регістрі TWCR, а також дозволена робота переривань установкою біта в регістрі SREG. Якщо біт TWIE скинутий, то стан TWINT повинен відстежуватися програмно для оцінки ситуації на шині TWI.

Після установки прапора TWINT інтерфейс TWI припиняє роботу і чекає реакції програми. В цьому випадку регістр статусу TWI (TWSR) містить значення, яке відображає поточний стан шини TWI. Виходячи з цього програма задає подальшу поведінку шини TWI, маніпулюючи регістрами TWCR і TWDR.

На рис. 1.40. показаний простий приклад підключення до шини TWI. Тут передбачається, що ведучій (майстер) бажає передати один байт даних веденому. Даний опис вельми загальний, а детальніше пояснення наводиться далі в цьому розділі.

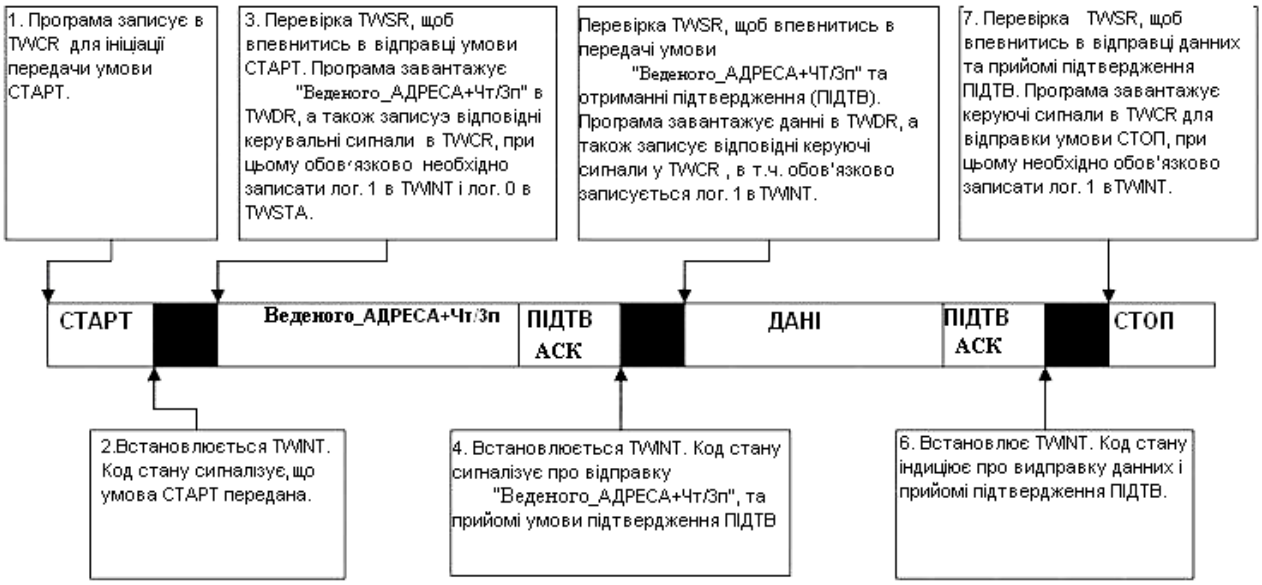


Рис. 1.40. Послідовність обслуговування TWI при типовій передачі

На рис 1.40 блоки 1,3,5,7 відображують дії програми, пункти 2,4,6 – стан пристрою I2C.

1) Першим кроком роботи TWI є передача умови СТАРТ. Це ініціюється шляхом запис

запису певного значення в TWCR. Зазначимо, що необхідно стежити, аби в записуваному в регістрі TWICR був скинутий біт TWINT. Запис лог. 1 в TWINT скидає цей прапор. TWI не почне роботу до тих пір, поки буде встановлений прапор TWINT в регістрі TWCR. Відразу після скидання TWINT починається передача умови СТАРТ.

2)Після передачі умови СТАРТ встановлюється прапор TWINT в регістрі TWCR, а вміст TWSR оновлюється значенням коду стану, що відображає про успішну передачу умови СТАРТ.

3)У програмі необхідно виконати перевірку значення TWSR, аби переконається в тому, що умова СТАРТ була успішно передана. Якщо TWSR відображає іншу ситуацію, то програма виконує особливі дії, наприклад, викликає процедуру обробки помилкових ситуацій. Якщо код стану має очікуване значення, то виконується завантаження умови ВЕДЕН_АДР +

ЗАПИС в TWDR. Необхідно пам'ятати, що TWDR використовується для зберігання як адреси, так і даних. Після завантаження в TWDR бажаного значення ВЕДЕН _АДР + ЗАПИС в регістр TWCR має бути записане специфічне значення, яке служить командою для передачі значення ВЕДЕН _АДР + ЗАПИС, що зберігається в TWDR. Проте необхідно стежити, аби в записуваному в регістр значенні був встановлений біт TWINT. Запис лог. 1 в TWINT наводить до скидання цього прапора. TWI не почне роботу до тих пір, поки встановлений біт TWINT в регістрі TWCR. Відразу після скидання прапора TWINT ініціюється передача адресного пакету.

Після передачі адресного пакету встановлюється прапор TWINT в регістрі TWCR, а вміст регістра TWSR оновлюється кодом стану, що відображає успішність передачі адресного пакету. У коді стану також відбивається чи було підтвердження прийому адресного пакету з боку веденого чи ні.

4)Виконується програмна перевірка значення TWSR, аби переконатися в успішності передачі адресного пакету і що біт підтвердження ПІДТБ має очікуване значення. Якщо TWSR відображає іншу ситуацію, то при необхідності виконуються особливі дії, наприклад, викликається процедура обробки помилкових ситуацій. Якщо ж код стану має очікуване значення, то програма записує пакет даних в TWDR. Згодом в регістр TWCR записується специфічне значення, яке служить командою для TWI і викликає апаратну передачу даних, записаних в TWDR. В записуваному значенні має бути встановлений біт TWINT. Запис лог. 1 в TWINT наводить до скидання цього прапора. TWI не почне роботу до тих пір, поки буде встановлений біт TWINT в регістрі TWCR. Відразу після скидання TWINT починається передача пакету даних.

5)Після передачі пакету даних встановлюється прапор TWINT в регістрі TWCR, а вміст регістра TWSR оновлюється значенням коди стану, яка

сигналізує про успішної передачі пакету даних. У коді стану також відбивається чи було прийнято підтвердження від підлеглого чи ні.

6)Виконується програмна перевірка значення в TWSR, аби переконатися в успішності передачі пакету даних і в тому, що біт ПІДТБ має очікуване значення. Якщо TWSR відображає іншу ситуацію, то програма виконує особливі дії, в т.ч. викликає процедуру обробки переривання. Якщо код стану має очікуване значення, то виконується запис спеціального значення в TWCR, яке служить командою для TWI і ініціює передачу умови СТОП. Під час запису має бути виконана установка біта TWINT. Запис лог. 1 в TWINT наводить до скидання цього прапора. TWI не почне роботу до тих пір, поки встановлений біт TWINT в регістрі TWCR. Відразу після скидання прапора TWINT ініціюється передача умови СТОП. Зверніть увагу, що прапор TWINT НЕ встановлюється після закінчення передачі умови СТОП.

Не дивлячись на простоту викладеного прикладу, він показує принципи, покладені в основу будь-якої передачі через TWI. З вищевикладеного можна зробити наступні виводи:

Після закінчення роботи TWI встановлюється прапор TWINT і далі очікується реакція з боку програми. Лінія знаходиться в низькому стані, поки скинутий прапор TWINT.

Якщо прапор TWINT встановлений, то користувач може оновлювати будь-який з регістрів TWI значенням, яке відноситься до наступного етапу роботи шини TWI. Наприклад, в TWDR завантажується значення, яке необхідно передати на наступному циклі шини.

Після оновлення всіх регістрів TWI і завершенні інших завдань виконується запис в TWCR. Під час запису TWCR необхідне, аби був встановлений біт TWINT. В цьому випадку запис лог. 1 в TWINT приведе до скидання даного прапора. TWI виконує дії у відповідності установкою регістра TWCR.

Приклад на Асемблері та Сі

Дії програми згідно рис. 1.40 наведено в прикладі на Асемблері та Сі.

Передбачається, що всі символьні позначення визначені в приєднаному файлі.

№	Програма на Асемблері	Програма на Сі	Коментар
1	ldi r16 (1<<TWINT)|(1<<TWSTA)|(1<<TWEN) out TWCR, r16	TWCR = (1<<TWINT)| (1<<TWSTA)|(1<<TWEN)	Передача умови СТАРТ
2	wait1: in r16,TWCR sbrs r16,TWINT rjmp wait1	while (!(TWCR & (1<<TWINT))) ;	Чекання установки прапора TWINT. Цим відображається завершення передачі умови СТАРТ
3	in r16,TWSR andi r16, 0xF8 cpi r16, START brne ERROR	if ((TWSR & 0xF8) != START) ERROR();	Перевірка коди стану TWI. Маскування біту переддільника.Якщо код стану не рівний СТАРТ, то перехід на ERROR
	ldi r16, SLA_W out TWDR, r16 ldi r16 (1<<TWINT) | (1<<TWEN) out TWCR, r16	TWDR = SLA_W; TWCR = (1<<TWINT) | (1<<TWEN);	Завантаження ВЕДЕН _АДР + ЗАПИС в регістр TWDR. Скидання біта TWINT в TWCR для початку передачі адреси
4	wait2: in r16,TWCR sbrs r16,TWINT rjmp wait2	while (!(TWCR & (1<<TWINT))) ;	Чекання установки прапора TWINT. Цим сигналізируется завершення передачі ВЕДЕН _АДР + ЗАПИС і отримання/не отримання підтвердження (ПІДТВ/НЕ ПІДТВ).
5	in r16,TWSR andi r16, 0xF8 cpi r16, MT_SLA_ACK brne ERROR	if ((TWSR & 0xF8) != MT_SLA_ACK) ERROR();	Перевірка значення регістра стану.Маскування біту переддільника.Якщо стан відрізняється від MT_SLA_ACK, то перехід на ERROR
	ldi r16, DATA out TWDR, r16 ldi r16 (1<<TWINT) | (1<<TWEN) out TWCR, r16	TWDR = DATA; TWCR = (1<<TWINT) | (1<<TWEN);	Завантаження даних в TWDR скидання прапора TWINT в TWCR для початку передачі даних
6	wait3: in r16,TWCR sbrs r16,TWINT rjmp wait3	while (!(TWCR & (1<<TWINT))) ;	Чекання установки прапора TWINT Цим відображається, що дані були передані і прийнято/не прийнято

			підтвердження (ПІТДВ/НЕ ПІДТВ).
7	in r16,TWSR andi r16, 0xF8 cpi r16, MT_DATA_ACK brne ERROR	if ((TWSR & 0xF8) != MT_DATA_ACK) ERROR();	Перевірка значення регістра стану TWI. Маскування біту переддільника. Якщо стан відрізняється від MT_DATA_ACK, то перехід на ERROR
	ldi r16 (1<<TWINT)|(1<<TWEN)| (1<<TWSTO) out TWCR, r16	TWCR = (1<<TWINT)|(1<<TWEN)| (1<<TWSTO);	Передача умови СТОП

Режими роботи

Модуль TWI може працювати в наступних режимах:

- Ведучий – передавач(Master Transmitter);
- Ведучий – прийомач (Master Receiver);
- Ведений - прийомач(Slave Transmitter) ;
- Ведений – передавач (Slave Receiver).

Вибір конкретного режиму визначається логікою роботи програми і, відповідно, виконуваними діями.

У всіх режимах будь-який етап взаємодії прикладної програми з модулем I2C складається з трьох частин:

• після завершення модулем виконання якої-небудь операції, він встановлює прапор TWINT регістра TWCR і чекає реакції програми. Поки прапорець встановлений в «1», на лінії SCL утримується НИЗЬКИЙ рівень;

• після установки прапора TWINT користувач повинен занести в регістр модуля значення, відповідні наступному етапу обміну;

• після оновлення вмісту регістрів модуля, користувач повинен сформувати в регістрі TWCR команду на виконання наступного етапу обміну. При завантаженні в регістр нового значення необхідно скинути прапор TWINT записом в нього лог. 1. Після скидання прапора модуль почне виконання операції, яка визначається вмістом регістра TWCR.

Значення кодів статусу залежать від конкретних режимів роботи модуля TWI, крім двох кодів стану ($ F8 і $ 00). (Див. табл. 1.6)

Стан з кодом $ F8 є проміжним. Наявність цього коду означає відсутність будь-якої інформації у зв'язку з тим, що прапор TWINT не встановлений. Код статусу $ 00 сигналізує про помилку на шині. Така помилка виникає при формуванні ведучим станів СТАРТ або СТОП в невідповідному місці, наприклад, під час передачі байта адреси, байта даних або біта підтвердження. Для виходу з таких ситуацій необхідно записати лог. 1 в розряди TWSTO і TWINT. В результаті модуль перейде в режим неадресованного ведучого, звільнить лінії SDA і SCL і скине прапор TWSTO. Стан СТОП в цій ситуації сформовано не буде.

Режим «Ведучий – передавач»

У режимі «Ведучий передавач» (Master Transmitter) здійснюється передача даних від ведучого пристрою до веденого. Для перемикання пристрою в режим ведучого модуль I2C повинен сформувати на шині стан СТАРТ. Формат адресного пакета, переданого потім, визначає в якому з режимів буде працювати ведучий. При передачі пакета SLA + W модуль переходить в режим «Ведучий передавач», а при передачі пакету SLA + R - в режим «Ведучий приймач».

Формування стану СТАРТ почнеться після запису в регістр TWCR наступного значення:

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN		TWIE
1	x	1	0	x	1		x
Скинути прапор TWINT		Сформувати стан СТАРТ			Дозволити роботу I2C		

В результаті запису вказаного значення модуль I2C почне контролювати стан шини і сформує стан СТАРТ відразу ж, як тільки вона стане вільною. Після закінчення формування стану СТАРТ встановлюється прапор TWINT; код статусу повинен при цьому мати значення, рівне $ 08. Для перемикання

модуля в режим «Ведучий передавач" необхідно передати по шині пакет SLA + W. Для цього вміст пакету завантажується в регістр TWDR, а в регістр TWCR заноситься таке значення:

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	x	0	0	x	1	0	x
Скинути прапор TWINT					Дозволити роботу I2C		

Після передачі адресного пакета і прийому біта підтвердження прапор TWINT знову встановлюється в «1». Код статусу на цьому етапі може мати одне з наступних значень: $ 18, $ 20 або $ 38.

Після передачі адресного пакету повинні бути передані пакети даних. Значення байта даних завантажується в регістр TWDR. Передача пакету даних починається після запису в регістр TWCR наступного значення:

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	x	0	0	x	1	0	x
Скинути прапор TWINT					Дозволити роботу I2C		

Описана процедура використовується для передачі всіх пакетів даних. Після передачі останнього байта даних, ведучий повинен сформувати на шині стан СТОП або ПОВСТАРТ Формування стану СТОП почнеться після запису в регістр TWCR наступного значення:

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	x	0	1	x	1	0	x
Скинути прапор TWINT			Сформувати стан СТОП				

А для формування стану ПОВСТАРТ в регістр TWCR необхідно занести значення:

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	x	0	1	x	1	0	x
Скинути прапор TWINT		Сформувати стан СТАРТ			Дозволити роботу I2C		

Після формування на шині стану ПОВСТАРТ (код статусу $ 10) ведучий може адресувати того ж або іншого веденого не формуючи стану СТОП.

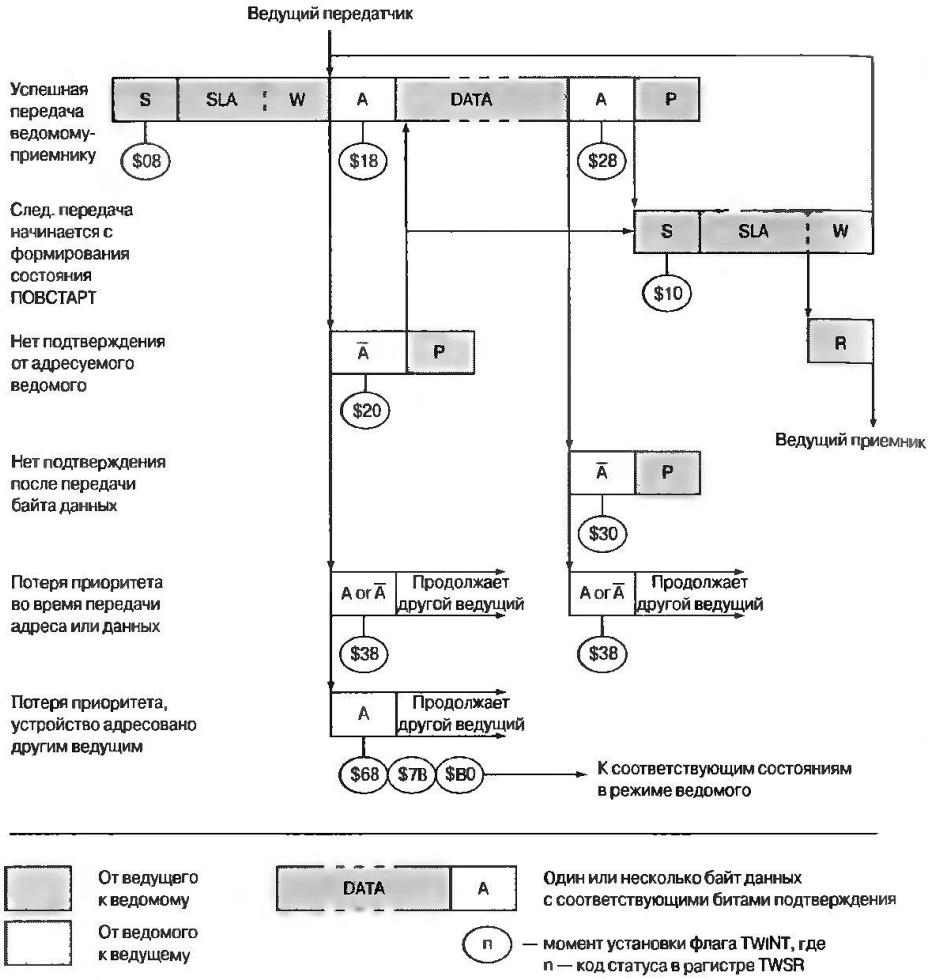
Іншими словами, використання стану ПОВСТАРТ дозволяє здійснювати зміну ведених пристроїв, а також перемикатися між режимами «Ведучий передавач» і «Ведучий приймач» без втрати контролю над шиною.

Коди статусу для режима «Ведучий передавач» наведено в табл. 1.7., а послідовність дій в цьому режимі ілюструється рис. 1.41.

Таблиця *1.7* - Коди статусу для режима «Ведучий передавач»

Код статусу	Стан шини і модуля TWI	Дії программы					Наступна дія, що виконується модулем TWI
		в/з TWDR	в регістр TWCR				
			STA	STO	TWINT	TWEA	
$08	Був сформований стан СТАРТ	Завантажити SLA + W	X	0	1	X	Буде переданоSLA + W. Буде прийнято АСК або NACK
$10	Був сформований	Завантажити SLA + W	X	0	1	X	Буде передано SLA + W. Буде прийнятий АСК або NACK
	стан ПОВСТАРТ	Завантажити SLA + R	X	0	1	X	Буде передано SLA+R. Модуль переключиться в режим "Ведучий приймач»
$18		Завантажити дані	0	0	I	x	Буде передано байт даних. Буде прийнятий АСК або NACK
	Був переданий пакет	Немає дій	1	0	1	X	Буде сформовано стан ПОВСТАРТ
	SLA+W і прийнято підтвердження (АСК)	Немає дій	0	1	I	X	Буде сформований стан СТОП (прапор TWSTO буде скинуто)
		Немає дій	1	1	1	X	Буде сформовано стан СТОП, а потім стан СТАРТ (прапор TWSTO буде скинуто)
$20	Був переданий пакет SLA+W, а підтвердження не було при-йнято (NACK)	Завантажити дані	0	0	1	X	Буде передано байт даних. Буде отримано АСК або NACK
		Немає дій	1	0	1	X	Буде сформовано стан ПОВСТАРТ
		Немає дій	0	1	1	X	Буде сформовано стан СТОП (прапор TWSTO буде скинуто)
		Немає дій	1	I	1	X	Буде сформовано стан СТОП, а потім стан СТАРТ (прапор TWSTO буде скинуто)
$28	Був переданий пакет даних і прийнято підтвердження	Завантажити дані	0	0	1	X	Буде передано байт даних. Буде отримано АСК або NACK
		Немає дій	1	0	1	X	Буде сформовано стан ПОВСТАРТ

	(АСК)	Немає дій	0	1	1		Буде сформовано стан СТОП (прапор TWSTO буде скинуто)
		Немає дій	1	I	1	X	Буде сформовано стан СТОП, а потім стан СТАРТ (прапор TWSTO буде скинуто)
$30	Був переданий пакет даних, а підтвердження не було прийнято (NACK)	Завантажити дані	0	0	1	X	Буде передано байт даних. Буде отримано АСК або NACK
		Немає дій	1	0	1	X	Буде сформовано стан ПОВСТАРТ
		Немає дій	0	1	1	X	Буде сформовано стан СТОП (прапор TWSTO буде скинуто)
		Немає дій	1	1	1	X	Буде сформовано стан СТОП, а потім стан СТАРТ (прапор TWSTO буде скинуто)
$38	Втрата приорітету при передачі пакету адреса або даних	Немає дій	0	0	1	X	Пристрій звільнить шину і перейде в режим неадресованого веденого
		Немає дій	1	0	I	X	Після звільнення шини буде сформовано стан СТАРТ



Состояния модуля TWI в режиме «Ведущий передатчик»

Рис. 1.41

Режим «Ведучий приймач»

У режимі «приймач» здійснює прийом даних від веденого пристрою. Як звичайно, для перемикання пристрою в режим ведучого модуль I2C повинен сформувати на шині стан СТАРТ. Формат адресного пакета, переданого слідом, визначає, в якому з режимів буде працювати ведучий. При передачі пакета SLA + W модуль переходить в режим «передавач», а при передачі пакету SLA + R переходить в режим «Ведучий приймач».

Формування стану СТАРТ починається після запису в регістр TWCR наступного значення

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	x	1	0	x	1	0	x
Скинути прапор TWINT		Сформувати стан СТАРТ			Дозволити роботу I2C		

В результаті запису вказаного значення модуль I2C почне контролювати стан шини і сформує стан СТАРТ відразу ж, як тільки вона стане вільною. Після закінчення формування стану СТАРТ встановлюється прапор TWINT; код статусу повинен при цьому мати значення, рівне $ 08 (див. табл. 1.6). Для перемикання модуля в режим «Ведучиий передавач» необхідно передати по шині пакет SLA + R. Для цього вміст пакету завантажується в регістр TWDR, а в регістр TWCR заноситься таке значення:

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	x	0	0	x	1	0	x
Скинути прапор TWINT					Дозволити роботу I2C		

Після передачі адресного пакета і прийому біта підтвердження прапор TWINT знову встановлюється в «1». Отриманий від веденого байт даних знаходиться при цьому в регістрі TWDR. Код статусу на цьому етапі може мати одне з наступних значень: $ 38, $ 40 або $ 48.

Описана процедура використовується для передачі всіх пакетів даних. Після прийому останнього байта даних ведучий повинен проінформувати про це веденого передавача, пославши сигнал непідтвердження (NACK). Потім ведучий повинен сформувати на шині стан СТОП або ПОВСТАРТ. Формування стану СТОП почнеться після запису в регістр TWCR наступного значення:

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	x	0	1	x	1	0	x
Скинути прапор TWINT			Сформувати стан СТОП		Дозволити роботу I2C		

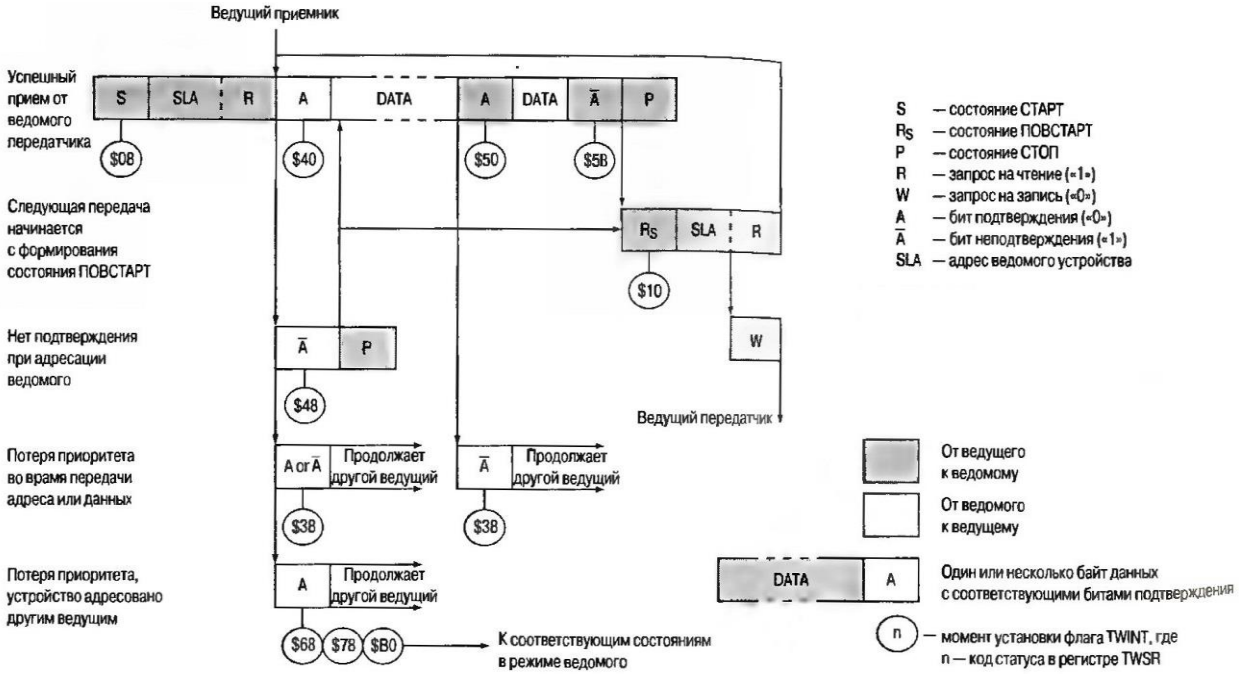
Для формування стану ПОВСТАРТ в регістр TWCR необхідно занести значення:

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	x	1	0	x	1	0	x
Скинути прапор TWINT		Сформувати стан СТАРТ			Дозволити роботу I2C		

Коди статусу для режима «Ведучий приймач» наведено в табл. 1.8., а послідовність дій в цьому режимі ілюструється рис. 1.42.

Таблиця 1.8 -Коди статуса для режима «Ведучий приймач»

Код статусу	Стан шини і модуля TWI	Дії програми					Наступна дія, що виконується модулем TWI
		в/з TWDR	в регістр TWCR				
			STA	STO	TWIN	TWEA	
$08	Був сформований стан СТАРТ	Завантажити SLA + R	X	0	1	X	Буде передано SLA+R. Буде отримано АСК абоNACK
$10	Був сформований стан ПОВСТАРТ	Завантажити SLA + R	X	0	1	X	Буде передано SLA+R. Буде отримано АСК абоNACK
		Завантажити SLA + W	X	0	1	X	Буде передано SLA+W Модуль переключиться в режим «Ведучий приймач»
$38	Втрата приорітету при передачі пакету адреса або даних	Нет действий	0	0	1	X	Пристрій звільнить шину і перейде в режим неадресованого веденого
		Нема дій	1	0	1	X	Після звільнення шини буде сформовано стан СТАРТ
$40	Був переданий пакет SLA+R і прийнято підтвердження (АСК)	Нема дій	0	0	1	0	Буде прийнято байт даних і передано непідтвердження (NACK)
		Нема дій	0	0	1	1	Буде прийнято байт даних і передано підтвердження (АСК)
$48	Був переданий пакет SLA+R і прийнято непідтвердження (NACK)	Нема дій	1	0	1	X	Буде сформовано стан ПОВСТАРТ
		Нема дій	0	1	1	X	Буде сформовано стан СТОП (прапор TWSTO буде скинуто)
		Нема дій	1	1	1	X	Буде сформовано стан СТОП, а потім стан СТАРТ (прапор TWSTO буде скинуто)
$50	Був принятий байт даних і передано підтвердження (АСК)	Прочитати дані	0	0	1	0	Буде прийнято байт даних і передано непідтвердження (NACK)
		Прочитати дані	0	0	1	1	Буде прийнято байт даних і передано підтвердження (АСК)
$58	Був принятий байт даних і передано непідтвердження (NACK)	Прочитати дані	1	0	1	X	Буде сформовано стан ПОВСТАРТ
		Прочитати дані	0	1	1	X	Буде сформовано стан СТОП (прапор TWSTO буде скинуто)
		Прочитати дані	1	1	1	X	Буде сформовано стан СТОП, а потім стан СТАРТ (прапор TWSTO буде скинуто)



Состояния модуля TWI в режиме «Ведущий приемник»

Рис. 1.42

Режим «Ведений приймач»

В режимі «Ведений приймач» ведений пристрій здійснює прийом даних від ведучого. Перед тим як перемкнути модуль в режим «Ведений приймач», слід занести в старші розряди регістра TWAR адресу пристрою і відповідно до логіки роботи програми встановити або скинути молодший розряд регістра (TWGCE). Потім необхідно записати в регістр TWCR таке значення:

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
0	1	0	0	0	1	0	x
	Дозволити підтвердження				Дозволити роботу I2C		

Після ініціалізації регістрів TWAR і TWCR модуль буде очікувати адресного пакета з адресою, зазначеною в регістрі TWAR, або адресу загального виклику (якщо її розпізнавання дозволено). Значення наступного за адресою біта напряму визначить режим, в який переключиться модуль. Якщо цей біт буде скинутий в «0» (запис), модуль I2C переключиться в

режим «Ведений приймач». В іншому випадку модуль переключиться в режим «Ведений передавач».

Після прийому пакету SLA + W встановиться прапор TWINT і стан обміну по шині можна буде, як звичайно, визначити за кодом статусу. Щоб перервати потік даних, слід скинути в «0» розряд TWEA регістру TWCR (це можна зробити і під час обміну, тобто при скинутому прапорі TWINT). В результаті після передачі наступного байта на лінію SDA буде виданий сигнал непідтвердження, що сигналізує ведучому про те, що ведений не може більше здійснювати прийом даних. Обробка адресних пакетів при скинутому розряді TWEA припиняється, але може бути відновлена в будь-який момент часу повторною установкою цього розряду.

Якщо адресація пристрою відбувається при знаходженні мікроконтролера в «сплячому» режимі і розряд TWEA буде при цьому встановлений, модуль I2C переведе мікроконтролер в робочий режим.

Коди статусу для режима «Ведений приймач» наведено в табл. 1.9., а послідовність дій в цьому режимі ілюструється рис. 1.43.

Таблиця 1.9 - Коди статусу для режима «Ведений приймач»

Код статусу	Стан шини і модуля TWI	Дії програми					Наступна дія, що виконується модулем TWI
		в/з TWDR	в регістр TWCR				
			STA	STO	TWIN	TWEA	
$60	Був прийнятий SLA+W з власною адресою та вислано підтвердження (ACK)	Немає дій	X	0	1	0	Буде прийнято байт даних і передано непідтвердження (NACK)
		Немає дій	X	0	1	1	Буде прийнято байт даних і передано підтвердження (ACK)
$68	Втрата приорітету в режимі ведучого під час передачі SLA+R/W; був	Немає дій	X	0	1	0	Буде прийнято байт даних і передано непідтвердження (NACK)
		Немає дій	X	0	1	1	Буде прийнято байт даних і передано підтвердження (ACK)

	прийнятий SLA+W з власною адресою та вислано підтвердження (ACK)						
$70	Був прийнятий загальний виклик і послано підтвердження (ACK)	Немає дій	X	0	1	0	Буде прийнято байт даних і передано непідтвердження (NACK)
		Немає дій	X	0	1	1	Буде прийнято байт даних і передано підтвердження (ACK)
$78	Втрата приорітету в режимі ведучого під час передачі SLA+R/W; був прийнятий загальний виклик і послано підтвердження (ACK)	Немає дій	X	0	1	0	Буде прийнято байт даних і передано непідтвердження (NACK))
		Немає дій	X	0	1	1	Буде прийнято байт даних і передано підтвердження (ACK)
$80	Пристрій вже адресований: був прийнятий байт даних і послано підтвердження (ACK)	Прочитати дані	X	0	1	0	Буде прийнято байт даних і передано непідтвердження (NACK)
		Прочитати дані	X	0	1	1	Буде прийнято байт даних і передано підтвердження (ACK)
$88	Пристрій вже адресований: був прийнятий байт даних і послано непідтвердження (NACK)	Прочитати дані	0	0	1	0	Перемкнутися в режим неадресованого веденого; розпізнавання будь-яких викликів заборонено.
		Прочитати дані	0	0	1	1	Перемкнутися в режим неадресованого веденого; дозволено розпізнавання SLA з власною адресою; дозволено розпізнавання загальних викликів, якщо TWGCE = «1»
		Прочитати дані	1	0	1	0	Перемкнутися в режим неадресованого веденого; заборонено розпізнавання будь-яких викликів; після звільнення шини буде сформовано стан СТАРТ
		Прочитати дані	1	0	1	1	Перемкнутися в режим неадресованого веденого; дозволено розпізнавання SLA з

							власною адресою; дозволено розпізнавання загальних викликів, якщо TWGCE = «1» ; після звільнення шини буде сформовано стан СТАРТ
$90	Пристрій вже адресований: (загальний виклик): був прийнятий байт даних і послано підтвердження (ACK)	Прочитати дані	X	0	1	0	Буде прийнято байт даних і передано непідтвердження (NACK)
		Прочитати дані	X	0	1	1	Буде прийнято байт даних і передано підтвердження (ACK)
$98	Пристрій вже адресований: (загальний виклик): був прийнятий байт даних і послано непідтвердження (NACK)	Прочитати дані	0	0	1	0	Перемкнутися в режим неадресованого веденого; розпізнавання будь-яких викликів заборонено.
		Прочитати дані	0	0	1	1	Перемкнутися в режим неадресованого веденого; дозволено розпізнавання SLA з власною адресою; дозволено розпізнавання загальних викликів, якщо TWGCE = «1»
		Прочитати дані	1	0	1	0	Перемкнутися в режим неадресованого веденого; заборонено розпізнавання будь-яких викликів; після звільнення шини буде сформовано стан СТАРТ
		Прочитати дані	1	0	1	1	Перемкнутися в режим неадресованого веденого; дозволено розпізнавання SLA з власною адресою; дозволено розпізнавання загальних викликів, якщо TWGCE = «1» ; після звільнення шини буде сформовано стан СТАРТ
$A0	Був виявлений стан СТАРТ або ПОВСТАРТ в той час, коли пристрій був адресований в якості веденого	Прочитати дані	0	0	1	0	Перемкнутися в режим неадресованого веденого; розпізнавання будь-яких викликів заборонено.
		Прочитати дані	0	0	1	1	Перемкнутися в режим неадресованого веденого; дозволено розпізнавання SLA з власною адресою; дозволено розпізнавання загальних викликів, якщо TWGCE = «1»
		Прочитати дані	1	0	1	0	Перемкнутися в режим неадресованого веденого; заборонено розпізнавання будь-яких викликів; після звільнення

							шини буде сформовано стан СТАРТ
		Прочитати дані	1	0	1	1	Перемкнутися в режим неадресованого веденого; дозволено розпізнавання SLA з власною адресою; дозволено розпізнавання загальних викликів, якщо TWGCE = «1» ; після звільнення шини буде сформовано стан СТАРТ

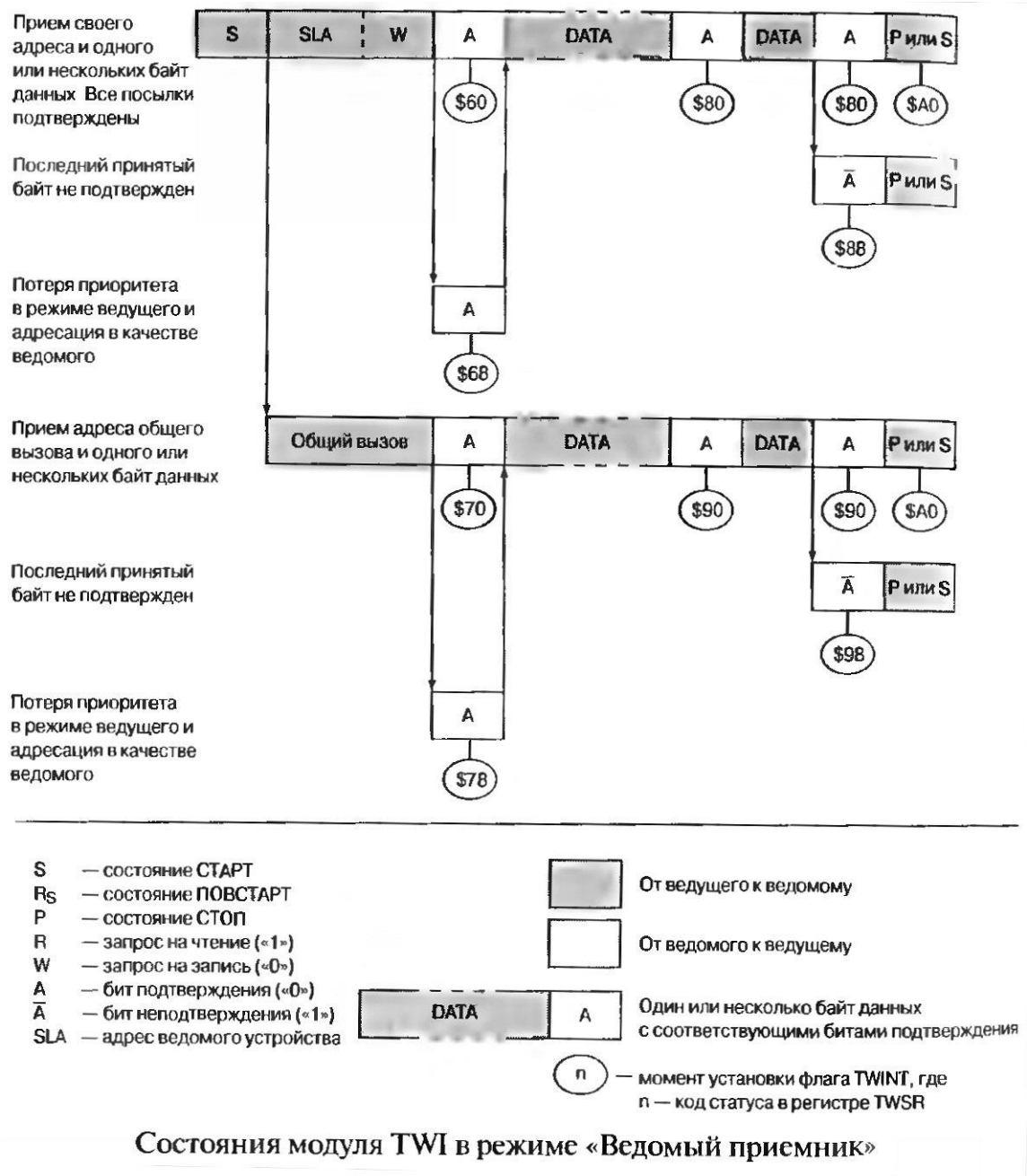


Рис. 1.43

Режим «Ведений передавач»

В режимі «Ведений передавач» ведений пристрій здійснює передачу даних ведучому, який в цьому випадку є приймачем. Перед тим, як перемкнути модуль в цей режим, необхідно занести в старші розряди регістра TWAR адресу пристрою і відповідно до логіки роботи програми встановити або скинути молодший розряд регістра Потім необхідно записати в регістр TWCR наступне значення:

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
0	1	0	0	0	1	0	x
	Дозволити підтвердження				Дозволити роботу I2C		

Після ініціалізації регістрів TWAR і TWCR модуль буде очікувати адресний пакет з адресою, зазначеною в регістрі TWAR, або адресу загального виклику (якщо його розпізнавання дозволено). Значення наступного за адресою біта напряму визначить режим, в який переключиться модуль. Якщо цей біт буде встановлений в «1» (читання), модуль I2C переключиться в режим "Ведений передавач». В іншому випадку модуль переключиться в режим "Ведений приймач». Після прийому пакету SLA + R встановиться прапор TWINT, і стан обміна можна буде, як звичайно, визначити за кодом статусу.

Коди статусу для режима «Ведений передавач» наведено в табл. 1.10., а послідовність дій в цьому режимі ілюструється рис. 1.44.

Таблиця 1.10 - Коди статусу для режима «Ведений передавач»

Код статусу	Стан шини і модуля TWI	Дії програми					Наступна дія, що виконується модулем TWI
		в/з TWDR	в регістр TWCR				
			STA	STO	TWIN	TWEA	
$A8	Був прийнятий SLA+W з власною	Завантажити дані	X	0	1	0	Буде передано останній байт даних; повинно бути отримано непідтвердження(NACK)

	адресою та вислано підтвердження (ACK)	Завантажити дані	X	0	1	1	Буде передано останній байт даних; повинно бути отримано підтвердження(ACK)
$B0	Втрата приорітету в режимі ведучого під час передачі SLA+R/W; був прийнятий SLA+W з власною адресою та вислано підтвердження (ACK)	Завантажити дані	X	0	1	0	Буде передано останній байт даних; повинно бути отримано непідтвердження(NACK)
		Завантажити дані	X	0	1	1	Буде передано останній байт даних; повинно бути отримано підтвердження(ACK)
$B8	Був переданий байт даних і отримано підтвердження (ACK)	Завантажити дані	X	0	1	0	Буде передано останній байт даних; повинно бути отримано непідтвердження(NACK)
		Завантажити дані	X	0	1	1	Буде передано останній байт даних; повинно бути отримано підтвердження(ACK)
$C0	Був переданий байт даних і отримано непідтвердження (NACK)	Немає дій	0	0	1	0	Перемкнутися в режим неадресованого веденого; розпізнавання будь-яких викликів заборонено.
		Немає дій	0	0	1	1	Перемкнутися в режим неадресованого веденого; дозволено розпізнавання SLA з власною адресою; дозволено розпізнавання загальних викликів, якщо TWGCE = «1»
		Немає дій	1	0	1	0	Перемкнутися в режим неадресованого веденого; заборонено розпізнавання будь-яких викликів; після звільнення шини буде сформовано стан СТАРТ
		Немає дій	1	0	1	1	Перемкнутися в режим неадресованого веденого; дозволено розпізнавання SLA з власною адресою; дозволено розпізнавання загальних викликів, якщо TWGCE = «1» ; після звільнення шини буде сформовано стан СТАРТ
$C8	Був переданий байт даних і отримано підтвердження	Немає дій	0	0	1	0	Перемкнутися в режим неадресованого веденого; розпізнавання будь-яких викликів заборонено.

	(ACK)	Немає дій	0	0	1	1	Перемкнутися в режим неадресованого веденого; дозволено розпізнавання SLA з власною адресою; дозволено розпізнавання загальних викликів, якщо TWGCE = «1»
		Немає дій	1	0	1	0	Перемкнутися в режим неадресованого веденого; заборонено розпізнавання будь-яких викликів; після звільнення шини буде сформовано стан СТАРТ
		Немає дій	1	0	1	1	Перемкнутися в режим неадресованого веденого; дозволено розпізнавання SLA з власною адресою; дозволено розпізнавання загальних викликів, якщо TWGCE = «1» ; після звільнення шини буде сформовано стан СТАРТ

При передачі останнього байта даних необхідно скинути в «0» розряд TWEA. Після цього модуль перейде в стан з кодом статусу $C0 або $C8 в залежності від того, який сигнал (ACK або NACK) передасть ведений у відповідь. У стан з кодом статусу $C8 модуль I2C перейде в разі, якщо ведучий зажадав додаткові дані, передавши підтвердження (ACK), незважаючи на те що ведений передавач передав останній байт і очікував сигналу NACK.

Рис. 2.126. Состояния модуля TWI в режиме «Ведомый передатчик»

Рис. 1.44

Після переходу в будь-який з вказаних станів модуль I2C буде ігнорувати звернення до нього ведучого. Відповідно, якщо ведучий буде продовжувати обмін по шині, він буде постійно приймати значеченіе «1». Обробка адресних пакетів при скинутому розряді TWEA також припиняється, але може бути відновлена в будь-який момент часу повторною установкою цього розряду.

Комбінування різних режимів

На практиці для виконання якої-небудь операції по шині I2C кожному пристрою доводиться використовувати декілька режимів, перемикаючись між ними при необхідності. В якості прикладу розглянемо операцію читання даних із зовнішнього EEPROM.

Всі операції такого роду можна розбити на чотири етапи:

1) ініціювання обміну;

2) передача адреси, за якою потрібно прочитати дані;

3) виконання читання;

4) завершення передачі.

Таким чином при виконанні операції відбувається передача інформації як від ведучого до веденого, так і навпаки. Після ініціювання обміну по шині ведучий повинен знаходитися в режимі «Ведений передавач», щоб повідомити веденому адресу, за якою він буде здійснити читання. Для виконання наступного етапу операції (читання даних) ведучий повинен перемкнутися в режим «Ведучий приймач». При цьому він повинен зберігати контроль над шиною під час виконання всіх етапів операції. Для цього використовується стан ПОВСТАРТ. У розглянутому випадку ведучий формує цей стан між передачею адреси і прийомом даних, як показано на рис. 1.45.

Рис. 1.45

Арбітраж

У разі присутності на шині декількох ведучих можлива ситуація, при якій кілька ведучих одночасно почнуть процес обміну. Для таких випадків специфікацією I2C передбачений процес розподілу пріоритетів (арбітраж), в результаті виконання якого на шині залишаються тільки один ведучий пристрій. Виконання арбітражу може розвиватися за різними сценаріями:

1.Два або більше ведучих здійснюють однотипний обмін з одним і тим же веденим. У цьому випадку ніхто з них не зможе розпізнати конфлікт на шині.

2. Два або більше ведучих звертаються до одного і того ж веденого з різними даними або різними типами обміну (читання / запис). У цьому випадку

розподіл пріоритетів відбудеться під час передачі бітів даних або біта напряму. Ведучий, який втратив пріоритет, може або переключитися в режим неадресованного веденого, або дочекатися звільнення шини і сформувати стан СТАРТ для її захоплення.

3. Два або більше ведучих звертаються до різних ведених. У цьому випадку розподіл пріоритетів почнеться під час передачі бітів адресного пакета. Ведучий, який втратив пріоритет, перемкнеться в режим веденого для перевірки, чи не був він адресований провідним. Якщо колишній ведучий був адресований, він переключиться в режим «Ведений передавач» або «Ведений приймач» в залежності від значення біта напряму. Якщо ж він не був адресований, він переключиться в режим неадресованного веденого, або дочекається звільнення шини і сформує новий стан СТАРТ.

Процес арбітража показано на рис. 1.46.

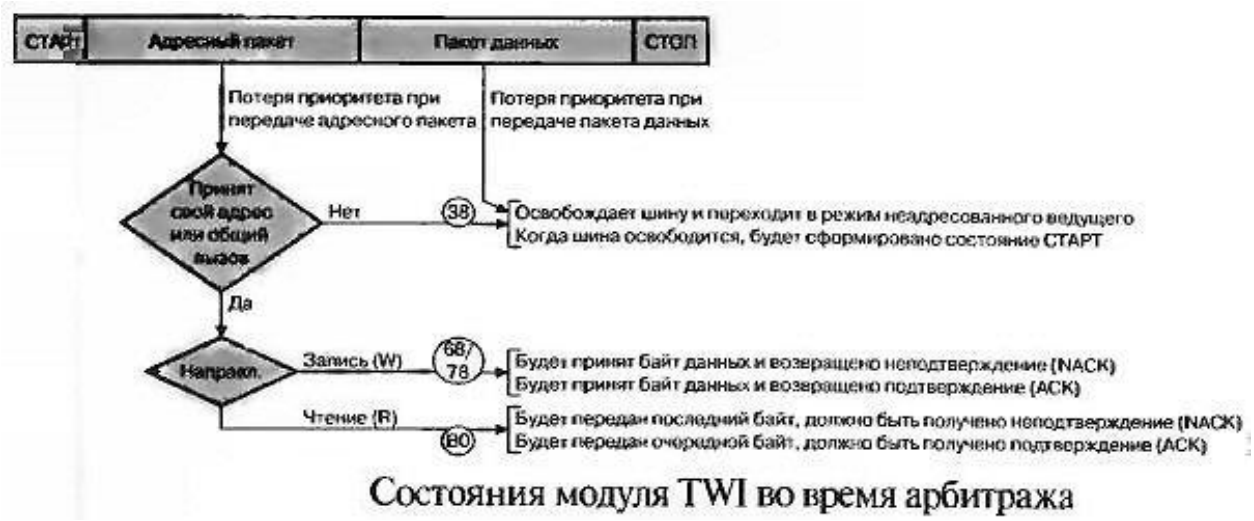


Рис. 1.46

Приклад 1. Виведення даних з мікроконтролера ATmegaSx в паралельний порт PCA9554 для індикації. Схема сполучення мікроконтролера МК і паралельного порту (ППП) наведена на рис. 1.47.

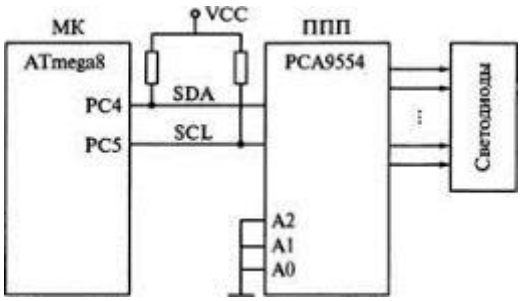


Рис. 1.47. Схема пристрою для обміну даними із ППП по інтерфейсу TWI

Приведена нижче програма виконує вивід в порт інвертіруемого 8-розрядного коду даних. При передачі пакетів з адресами і даними використаний механізм програмної перевірки умов установки в 1 прапора TWINT і станів регістра TWSR.

Програма має вигляд:

```
; Тестова програма для виводу даних в програмовний паралельний порт по інтерфейсу TWI
                .org 0×000
                rjmp init
                .include "mSdef.inc"            ;файл визначень ATmegaS
                .def SLA_W = rl7                ;адресний байт
                .def DATA = rl8                 ;байт даних (команди)
                .equ i2crd = 1                  ;1-читання
                .equ i2cwr = 0                  ;0-запис
                                                ;Коди статуса в режимі «Ведучий
                                                передавач»
                .equ START= 0×08                ;після СТАРТу
                .equ MT_SLA_ACK = 0x18          ;після передачі адреси
                .equ MT_DATA_ACK = 0x28         ;після передачі даних
init:           ldi rl6,HIGH(RAMEND)            ;ініціалізація покажчика стека
                out sph,rl6
                ldi r16,LOW(RAMEND)
                out spl,rl6
                                                ;Підготовка TWI до роботи
                ldi rl6,12                      ; Для частоти Fclk = 4 МГц
                out TWBR,rl6                    ; TWBR=12 TWPS=0
                ldi rl6, (1«TWEN)               ; Fscl = 100 КГц
                out TWCR,rl6
                ldi r20,0xAA                    ; дані для виводу
```

```
                                    ;Настройка мікросхеми ППП
        rcall Send_Start            ;генерація стартового біта
        ldi SLA_W,$40+i2cwr         ;посилка адреса+запис
        rcall Send_Adr
        ldi DATA,$03                ;команда запису в порт конфігурації
        rcall Send_Com_Data
        ldi DATA,0x00               ;настройка ППП на вивод
        rcall Send_Com_Data
        rcall Stop                  ;генерація стопового біта
                                    ;Вивод даних
loop:   rcall Send_Start            ;генерація стартового біта
        ldi SLA_W,$40+i2cwr         ;посилка адресу+запису
        rcall Send_Adr
        ldi DATA,$01                ;вихідной регістр
        rcall Send_Com_Data
        mov DATA,r20                ;вивід даних
        rcall Send_Com_Data
        rcall Stop                  ;генерація стопового біта
        com r20                     ;інвертування
        rjmp loop                   ;повтор виводу
                                    ; Функції режима ведучего передавача
Send_Start: ldi r16, (1«TWINT) I (1«TWSTA) |
        (1«TWEN)
        out TWCR,rl6                ;умова стану START
        waitl:
        in rl6,TWCR
        sbrs rl6,TWINT              ; очікування встановлення прапора
                                    TWINT
        rjmp waitl
        in rl6, TWSR
        andi rl6,0xF8
        cpi rl6,START               ;перевірка виконання СТАРТа
        brne error
        ret

Send_Adr:   out TWDR,SLA_W          ;завантаження пакета з адресою
        ldi r16, (1«TWINT) I (1«TWEN)
        out TWCR, rl6               ;передача
        wait2:
        in rl6,TWCR
        sbrs rl6,TWINT              ; очікування встановлення прапора
                                    TWINT
        rjmp wait2
        in rl6,TWSR
        andi rl6,0xF8
        cpi rl6,MT_SLA_ACK          ;перевірка виконання передачі
        brne error
        ret

Send_Com_   out TWDR,DATA           ;завантаження байта даних
```

```
Data:
            ldi r16, (1«TWINT) I (1«TWEN)
            out TWCR,rl6                    ;передача
            waits:
            in rl6,TWCR
            sbrs rl6,TWINT                  ; очікування встановлення прапора
                                            TWINT
            rjmp waits
            in rl6,TWSR
            andi rl6,0xF8
            cpi r16,MT_DATA_ACK             ;перевірка виконання передачі
            brne error
            ret

Stop:       ldi rl6, (1«TWINT) | (1«TWEN) |
            (1«TWST0)
            out TWCR,rl6                    ; умови для стану STOP
            ldi rl6,0xlf
            wait4:
            dec rl6                         ;затримка для стану STOP
            brne wait4
            ret
error:      ret
```

Приклад 2. Обмін даними між ведучим і веденим мікроконтролерами по інтерфейсу TWI (рис. 1.48).

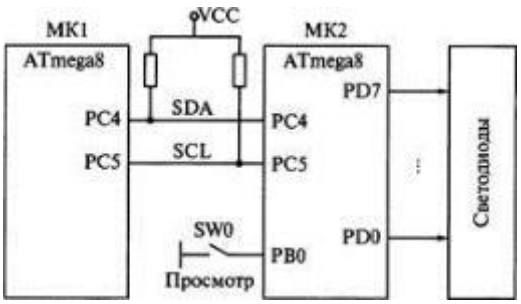


Рис. 1.48. Схема пристрою для обміну даними по інтерфейсу TWI між двомя мікроконтролерами

Замість зовнішніх резисторів, необхідних специфікацією інтерфейсу, використовуються внутрішні підтягувальні резистори виводів PC4, PC5 веденого мікроконтролера. У ролі ведучого виступає мікроконтролер МК1, в ролі веденого - мікроконтролер МК2. Робочий режим МК2 - режим очікування

Програма передачі ведучого мікроконтролера будується подібно до попереднього прикладу. Кожен переданий пакет містить службові біти, байти адреси і байт даних. Нижче наведено основний модуль програми для передачі трьох символів повідомлення по інтерфейсу TWI. Функції Send Start, Send_Ack, SendjCom Data, Stop ті ж, що і в попередньому прикладі:

```
; Основний модуль програми для передачі повідомлення ведучим мікроконтролером МК1 по інтерфейсу TWI
; Викликаються функції Send_Start, Send_Adr, Send_Com_Data,; Stop ті ж, що й у прикладі 1
; з'єднання: PC4мк1-PC4мк2, PC5мк1-PC5мк2
            .org 0×000
            rjmp init
            .include "mSdef.inc"            ;файл визначень ATmegaS
            .def temp = rl6
            .def SLA_W = rl7                ;адресний байт
            .def DATA = rl8                 ;байт даних (команди)
            .def count = rl9                ;лічильник даних
            .equ i2cwr = 0                  ;0-запис
                                            ;Коди статуса в режимі «Ведучий передавач»
            .equ START = 0x08               ;після СТАРТу
            .equ MT_SLA_ACK = 0x18          ;після передачі адреси
            .equ MT_DATA_ACK = 0x28         ;після передачі даних
init:       ldi r16,HIGH(RAMEND)            ;ініціалізація покажчика стека
            out sph,rl6
            ldi rl6,LOW(RAMEND)
            out spl,rl6
            ldi ZL,low(text*2)              ; завантаження адреси тексту повідомлення
            ldi ZH,high(text*2)             ; в регістр Z
            ldi count,3                     ;встановлення лічильника передач
                                            ;Підготовка TWI до роботи
            ldi r16,12                      ;для частоти Fclk = 4 МГц
            out TWBR,rl6                    ; TWBR=12, TWPS=0
            ldi rl6, (1«TWEN)               ; Fscl=100 КГц
            out TWCR,rl6
output:                                     ;вивід даних


            rcall Send_Start                ;генерація стартового біта
            ldi SLA_W,$44+i2cwr             ;посилка адреси+запис
            rcall Send_Adr
            lpm                             ;читання байту з flash-пам‘яті в r0
            mov DATA,r0                     ;вивід даних
            rcall Send_Com_Data
            rcall Stop                      ;генерація стопового біта
            adiw zl,l                       ;збільшення покажчика адреси на 1
            dec count                       ;зменшення лічильника на 1
```

```
        brne output
loop:   rjmp loop                   ;передача виконена
text:   .db ‘A’,'V’,'R’             ;текст повідомлення
                                    ; (ASCII – коди $41, $56, $52)
```

Програма веденого мікроконтролера має наступну структуру:

а) початкова ініціалізація мікроконтролера, включаючи визначення покажчика стека, лічильника прийнятих даних, налаштування портів, підготовку модуля TWI;

б) дозвіл переривань і перехід в режим очікування переривань;

в) обробка запитів переривань від модуля TWI, що забезпечує прийом пакетів з байтами адреси, команди і даних; збереження байтів повідомлення у внутрішній пам'яті SRAM; підрахунок числа байтів прийнятого повідомлення; по закінченні прийому - послідовний вивід байтів повідомлення при натисканні кнопки SWO.

Нижче наведено фрагмент програми веденого мікроконтролера, що включає процедуру ініціалізації, і обробник переривань від модуля TWI, що забезпечує прийом повідомлення по інтерфейсу TWI і збереження в пам'яті.

```
; Програма прийому по інтерфейсу TWI для веденого мікроконтролера МК2 (основний фрагмент програми)
; з'єднання: SWO-PB0, порт PD-LED
                    .org 0×000
                    rjmp init
                    .org 0×011
                    rjmp TWI_INT                ;перехід до обробки переривань
                    .include "mSdef.inc"        ;файл визначень ATmegaS
                    .def count = r22            ;лічильник даних
                    .def temp = rl6
                                                ;Коди статуса в режимі веденого приймача
                    .equ TW_SR_SLA_ACK =0x60    ;принято байт з адресою
                    .equ TW_SR_DATA_ACK =       ;принято байт даних
                    0x80
                    .equ TW_SR_STOP = 0xA0      ;виявлений стан STOP
                                                ;Ініціалізація веденого МК
```

```
init:           ldi r16,HIGH(RAMEND)        ;ініціалізація покажчика стека
                out sph,rl6
                ldi rl6,LOW(RAMEND)
                out spl,rl6
                clr temp                    ;настройка вивода порта PB0 на
                                            ввод
                out DDRB,temp
                sbi PORTE,0
                ser temp                    ;настройка виводов порта PD на
                                            вивод
                out DDRD,temp
                out PORTD,temp
                ldi count,3                 ;встановлення лічильника байтів
                ldi XL,0x80                 ;в регістре X адреса, за якій
                ldi XH,0x01                 записуються прийняті дані
                                            ;Підготовка TWI до роботи
                ldi temp,0x30               ; підключення підтягаючих
                                            резисторів на лініях SCL,SDA
                out PORTC,temp
                ldi r16,12                  ;для Fclk = 4 МГц, TWBR=12
                out TWBR,rl6                ; TWPS=0 Fscl=100 КГц
                ldi rl6,0x44                ;адреса пристрою I2C
                out TWAR,rl6
                di rl6, (1«TWINT) I (1«TWEA) |
                (1«TWIE) | (1«TWEN)
                out TWCR,rl6                ;запуск обміну по TWI
                sei
loop:           rjmp loop                   ; нескінченний цикл очікування
                                            ;Обробник переривання від
                                            модуля
TWI_INT:        in rl6,TWSR
                cpi r16,TW_SR_SLA_ACK       ;Перша перевірка
                brne ierror
                ldi rl6, (1«TWINT) I (1«TWEA) |
                (1«TWEN)
                out TWCR,rl6
TW_WAIT_DATA:   in rl6,TWSR
                cpi rl6,TW_SR_DATA_ACK      ;Друга перевірка
                brne TW_WAIT_DATA
                in rl6,TWDR
                St X+,rl6                   ; збереження байта даних в
                                            пам'яті
                ldi r16, (1«TWINT) I (1«TWEA) |
                (1«TWEN)
                out TWCR,rl6
TW_WAIT_STOP:   in rl6,TWSR                 ; очікування стану STOP
                cpi rl6,TW_SR_STOP          ;Третя перевірка
                brne TW_WAIT_STOP           ; Байт даних прийнято
                dec count                   ;зменшення лічильника даних
                brne ierror                 ; якщо не все, продовжуємо,
```

```
                rcall OUTLED                ; Інакше виводимо на індикатори
                reti
ierror:         ldi rl6, (1«TWINT) I (1«TWEA) |
                (1«TWIE) | (1«TWEN)
                out TWCR,rl6                ;перезапуск інтерфейса
                reti

                                            .Вивод на індикатори
OUTLED:         cr temp                     ;сигналізація - прийом
                                            завершений
                out PORTD,temp
                ldi XL,0x80                 ; установка початкової адреси
                ldi count,3                 ;встановлення лічильника байтів
WAIT_SHOW:      sbic PINB,SHOW              ; очікування натискання кнопки
                                            SHOW
                rjmp WAIT_SHOW
                Id temp,X+                  ; зчитування байта з пам'яті
                com temp                    ; інвертування
                out PORTD,temp              ; і вивід на світлодіоди
                rcall DELAY                 ;затримка
                dec count                   ;якщо показані не все дані,

                brne WAIT_SHOW              ; продовження по натисканню
                                            SHOW
                ret
```

ЛЕКЦІЯ 5

Розподілена мікропроцесорна система на базі CAN інтерфейсу

Визначення

Мережевий інтерфейс CAN (Controller Area Network) був розроблений в 1987г. (версія 1.0) фірмами BOSCH і INTEL для створення бортових мультипроцесорних систем реального часу. Остання специфікація інтерфейсу 2.0, розроблена фірмою BOSCH в 1992г., є доповненням попередньої версії. У міжнародній організації по стандартизації зареєстровано два стандарти : ISO 11898 (для високошвидкісних застосувань) і ISO 11519-2 (для низькошвидкісних застосувань).

CAN інтерфейс являє собою послідовну магістраль та призначений для організації високонадійних недорогих каналів зв'язку в розподілених

системах управління. Інтерфейс широко застосовується в промисловості, енергетиці і на транспорті. Дозволяє будувати як дешеві мультиплексні канали, так і високошвидкісні мережі.

CAN інтерфейс забезпечує ув'язку в мережу "інтелектуальних" пристроїв введення/виведення, датчиків і виконавчих пристроїв деякого механізму або навіть підприємства. Характеризується протоколом, що забезпечує можливість знаходження на магістралі декількох ведучіх пристроїв, забезпечує передачу даних в реальному масштабі часу і корекцію помилок, високою завадостійкістю.

У відповідності до моделі ISO/OSI CAN підрозділяэться на рівні:

- Рівень передачі даних (Data Link Layer)
- Фізичний рівень (Physical Layer)

Рівень передачі даних інакше 'об'єктний рівень або канальний рівень, призначений для забезпечення взаємодії мереж на фізичному рівні й контролю за помилками, які можуть виникнути. Отримані з фізичного рівня дані він упаковує в кадри, перевіряє на цілісність, якщо потрібно, виправляє помилки (формує повторний запит пошкодженого кадру) і відправляє на мережний рівень. Канальний рівень може взаємодіяти з одним або декількома фізичними рівнями, контролюючи й управляючи цією взаємодією. Специфікація IEEE 802 розділяє цей рівень на два підрівня:

- Підрівень логічного керування лінією (LLC logical link control)
- Підрівень керування доступом до середовища передачі (MAC media access control)

MAC підрівень являє собою ядро протоколу CAN. Він передає повідомлення, отримані від LLC підрівня, і приймає повідомлення, які будуть передані до LLC підрівня. MAC підрівень відповідальний за арбітраж, підтвердження, виявлення помилок і їх сигналізацію.

LLC підрівень - має відношення до фільтрації повідомлень, повідомленню про перевантаження та управлінню відновленням.

Фізичний рівень визначає, як сигнали фактично передаються і, отже, має справу з описом бітової синхронізації і кодування бітів. Усередині цієї специфікації характеристики передавача / приймача фізичного рівня не визначені, щоб дозволити середовищі передачі та реалізації рівня сигналу бути оптимізованими для конкретних систем.

Безпосередньо стандарт CAN від Bosch визначає передачу у відриві від фізичного рівня - він може бути яким завгодно, наприклад, радіоканалом або оптоволокном. Але на практиці під CAN-мережею зазвичай мається на увазі мережу топології «шина» з фізичним рівнем у вигляді диференціальної пари, визначеним в стандарті ISO 11898. Передача ведеться кадрами, які приймаються усіма вузлами мережі.

Рецесивні і домінантні біти

Для абстрагування від середовища передачі специфікація CAN уникає описувати двійкові значення як «0» і «1». Замість цього застосовуються терміни «рецесивний» і «домінантний», при цьому мається на увазі, що при передачі одним вузлом мережі рецесивного біта, а іншим домінантного, прийнятий буде домінантний біт. Наприклад, при реалізації фізичного рівня на радіоканалі відсутність сигналу означає рецесивний біт, а наявність - домінантний; тоді як в типової реалізації провідної мережі рецесії буває при наявності сигналу, а домінант, відповідно, за відсутності. Стандарт мережі вимагає від «фізичного рівня», фактично, єдиного умови: щоб домінантний біт міг придушити рецесивний, але не навпаки. Наприклад, в оптичному волокні домінантним біту повинен відповідати «світло», а рецесивним - «темрява». В електричному дроті може бути так: рецесивне стан - висока напруга на лінії (від джерела з великим внутрішнім опором), домінантне - низька напруга (всі вузли мережі «підтягують» лінію на землю). Якщо лінія знаходиться в рецесивним стані, перевести її в домінантне може будь-який вузол мережі (включивши світло в оптоволокні або закоротив висока напруга). Навпаки - не можна (включити темряву не можна).

Технічні характеристики

Характеристики CAN інтерфейсу зведено в таблицю 1.11

Таблиця 1.11. - Характеристики CAN інтерфейсу

Стандарт	ISO 11898
Швидкість передачі	1 Мбіт/с (максимум)
Відстань передачі	1000 м (максимум)
Характер сигналу, лінія передачі	диференціальна напруга, скручена пара
Кількість драйверів	64
Кількість приймачів	64
Схема з'єднання	напівдуплекс, багатоточкова

Швидкість передачі задається програмно і може бути до 1 Мбіт/с. Користувач вибирає швидкість, виходячи з відстаней, числа абонентів і ємкості ліній передачі, табл. 1.12.

Таблиця 1.12 - Залежність швидкості CAN інтерфейсу від відстані.

Відстань, м	25	50	100	250	500	1000	2500	5000
Швидкість, Кбіт/с	1000	800	500	250	125	50	20	10

Максимальне число абонентів, підключених до даного інтерфейсу фактично визначається здатністю навантаження застосованих приймачів. Наприклад, при використанні трансивера фірми PHILIPS PCA82C250 вона дорівнює 110.

Протокол CAN використовує оригінальну систему адресації повідомлень. Кожне повідомлення забезпечується ідентифікатором, який визначає призначення передаваних даних, але не адресу приймача. Будь-який приймач може реагувати як на один ідентифікатор, так і на декілька. На один ідентифікатор можуть реагувати декілька приймачів.

Протокол CAN має розвиненою системою виявлення і сигналізації помилок. Для цих цілей використовується порозрядний контроль, пряме заповнення бітового потоку, перевірка пакету повідомлення CRC-поліномом, контроль форми пакету повідомлень, підтвердження правильного прийому пакету даних. Хеммінговий інтервал d=6. Загальна вірогідність невиявленої помилки $4.7x10^{-11}$.

Система арбітражу протоколу CAN виключає втрату інформації і часу при "зіткненнях" на шині.

Інтерфейс із застосуванням протоколу CAN легко адаптується до фізичного середовища передачі інформації.

Організація мережі на основі CAN інтерфейсу

Пристрої в CAN-системі з'єднуються по шині (рис. 1.49).

Далі будемо розглядати випадок, коли сигнали в CAN шині є електричними. В цьому випадку пасивний (рецесивний) стан шини відповідає рівню логічної 1, а активний (домінуюче) стан відповідає рівню логічного 0. Коли по шині не передаються повідомлення, вона перебуває в пасивному стані. Передача повідомлення завжди починається з домінантного біта.

CAN шина представлена витою парою (екранованою або неекранованою) і загальним дротом. Плоска пара (телефонний тип кабелю) також працює добре, але чутливіша до зовнішніх джерел шуму. Усі пристрої, що входять в шину.з'єднані за "монтажним І".

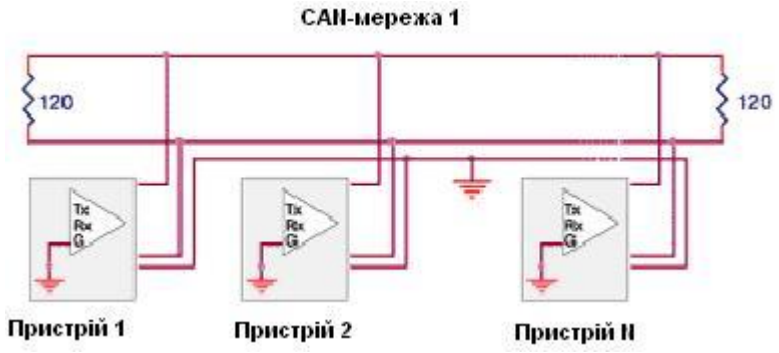


Рис. 1.49. З'єднання пристроїв по CAN-шині

Повідомлення даних, передавані з будь-якого вузла по CAN-шині, можуть містить від 1 до 8 байт. Кожне повідомлення помічене ідентифікатором, який в мережі є унікальним (наприклад: "Нагрівати до 240", "Відмова нагріву","Бункер завантажено", і так далі). При передачі інші вузли мережі отримують повідомлення і кожен з них перевіряє ідентифікатор. Якщо повідомлення має відношення до даного вузла, то воно обробляється, інакше – ігнорується. CAN-контролер кожного з пристроїв може обробляти одночасно декілька ідентифікаторів (наприклад, контролери SIEMENS і INTEL можуть обробляти до 15). Таким чином, в кожному з пристроїв можна легко організувати декілька "віртуальних" каналів обміну інформацією з різними пристроями, включаючи канали одночасного здобуття повідомлень.

У CAN- мережі є можливість одночасної передачі повідомлень відразу декільком пристроям. Ця особливість дозволяє передавати по ній синхросигнали.

Нові пристрої, призначені для прийому даних, можуть додаватися до мережі без зміни вже існуючих програмних засобів, якщо їх підключення не призводить до перевищення здатності навантаження і максимальної довжини шини. При цьому нові мережеві пристрої здатні обмінюватися інформацією між собою, не порушуючи працездатність старої системи, якщо в протоколі обміну були використані нові ідентифікатори.

Таким чином, прийнята в CAN-мережі схема передачі повідомлень забезпечує великі можливості при створенні, розширенні і модернізації систем.

CAN - вузли являють собою контролери, які здійснюють процедуру прийому-передачі даних і мають логіку обслуговуючу помилки. Всі вузли спочатку включені на прийом інформації.

Типова структура CAN-мережі з мікроконтролерами наведена на рис. 1.50.

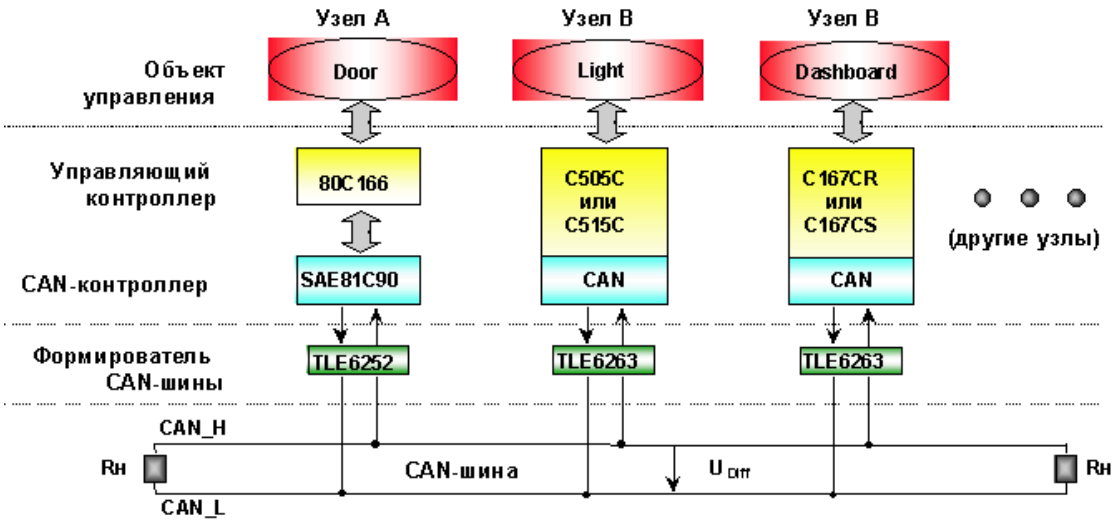


Рис 1. 50 Типова структура CAN-мережі

Вузол А являє собою з‘єднення керуючого контролера 80C166 (без вбудованного CAN-інтерфейса) з CAN-контролером SAE81C90 та шинним формувачем TLE 6252. Вузли Б та В містять CAN- контролер з формувачами.

Шинні формувачі призначені для перетворення логічних сигналів RxCAN і TxCAN контролера в рівні сигналів CAN-шини. Шинні формувачі являють собою надійні прийомо-передавачі і зберігають працездатність в тяжких умовах електромагнітних завад. Вони мають знижене енергоспоживання,

вбудований захист від перегріву і від короткого замикання, режим "засипання". Мікросхеми побудовані таким чином, що при виході їх з ладу вони не займають лінію передачі і не заважають нормальній роботі інших вузлів на лінії.

Робочий діапазон температур для цих мікросхем -40 .. 125 ° С.

Таблиця 1.13 - Технічні характеристики шинних формувачів

Тип	Макс.швидкість, кбод	Тип лінії	Опції
TLE6250G	1000	Вита пара	ISO/DIS 11898, 12/24 В лінія, защита от к.з.
TLE6252G	125	Вита пара, однопровідна лінія	Вбудований фільтр, захист від к.з., запобігання від заняття лінії
TLE6255G	40(100)	Однопровідна лінія	Захист від к.з., режим "засипання"
TLE6263G	125	Вита пара	Вбудований стабілізатор +5 В, "верхній" ключ 150 мА, SPI-інтерфейс

1.6.5. Формат CAN-повідомлення

Передача ведеться кадрами. Корисна інформація в кадрі складається з ідентифікатора довжиною 11 біт (стандартний формат) або 29 біт (розширений формат, надмножество попереднього) і поля даних довжиною від 0 до 8 байт. Ідентифікатор говорить про вміст пакету і служить для визначення пріоритету при спробі одночасної передачі декількома мережними вузлами. Ідентифікатор повідомлення 11- (2K повідомлень) або 29- бітовий (512M повідомлень) передається до кожного вузла системи, який визначає шляхом апаратного або програмного фільтру, чи є повідомлення призначеним для нього. Ідентифікатор також визначає пріоритет доступу до шини.

Нові пристрої, призначені для прийому даних, можуть додаватися до мережі без зміни вже існуючих програмних засобів, якщо їх підключення не призводить до перевищення здатності навантаження і максимальної довжини шини. При цьому нові мережеві пристрої здатні обмінюватися інформацією між собою, не порушуючи працездатність старої системи, якщо в протоколі обміну були використані нові ідентифікатори.

У CAN- мережі є можливість одночасної передачі повідомлень відразу декільком пристроям. Ця особливість дозволяє передавати по ній синхросигнали.

Стандартний CAN-протокол (версія 2.0A) підтримує формат повідомлення з 11-розрядними ідентифікаторами (Стандартне повідомлення).

Розширений CAN-протокол (версія 2.0B) підтримує 11-бітовий і 29-бітовий формати ідентифікаторів (Розширене повідомлення).

Більшість контролерів версії 2.0A передають і приймають лише повідомлення стандартного формату, хоча частина з них можуть лише отримувати повідомлення розширеного формату.

Контролери версії 2.0B можуть посилати і отримувати повідомлення в обох форматах.

Відмінності форматів наступні:

У версії 2.0B поле бітів ідентифікатора складається з двох частин.

Перша частина (основна частина ідентифікатора) має довжину одинадцять бітів для сумісності з версією 2.0A, друга частина - вісімнадцять бітів (розширення ідентифікатора), що дає загальну довжину ідентифікатора в двадцять дев'ять біт.

Для розрізнення форматів використовуються біти Identifier Extension (IDE) і Substitute Remote Request (SRR) в Полі Арбітражу.

Ідентифікатор визначає тип і пріоритет повідомлення. Нижчому числовому значенню ідентифікатора відповідає вище значення пріоритету. Повідомлення, що має вищий пріоритет, передається раніше повідомлення,

що має нижчий пріоритет. Після повідомлення з високим пріоритетом передається повідомлення з нижчим пріоритетом, якщо під час передачі не з'явиться повідомлення з вищим пріоритетом, потім передається повідомлення з ще нижчим пріоритетом і так далі

Види кадрів

Види кадрів наступні:

• Кадр даних (data frame);

• Кадр віддаленого запиту (remote frame)

• Кадр перевантаження (overload frame);

• Кадр помилки (error frame).

Кадр даних передає дані від передавача приймача.

Кадр віддаленого запиту даних передається вузлом, щоб запросити передачу кадру даних з тим же самим ідентифікатором.

Кадр помилки передається будь вузлом при виявленні помилки на шині.

Кадр перевантаження використовується, щоб забезпечити додаткову затримку між попереднім і наступним кадром даних або кадром віддаленого запиту даних.

Кадри даних і запиту відокремлюються від попередніх кадрів міжкадровим проміжком.

Кадри даних

Кадри даних можуть використовуватися і в стандартному і в розширеному форматі. Формати кадрів наведено на рис. 1.51**.** та в табл. 1.14 – 1.15.

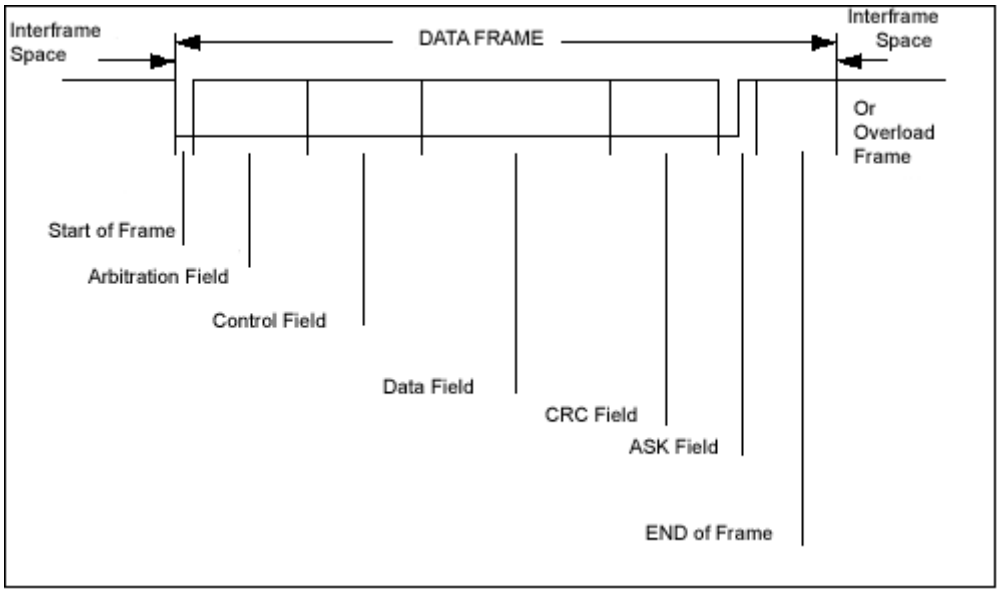


Рис. 1.51 Формат кадрів даних

Кадр даних складається з семи різних полів:

"Початок кадру" (start of frame), "поле арбітражу" (arbitration field), "поле управління" (control field), "поле даних" (data field), "поле CRC" (CRC field), "поле підтвердження" (ACK field), "кінець кадру" (end of frame). Поле даних може мати нульову довжину.

Таблиця 1.14 - Базовий формат кадра даних

Поле		Довжина (в бітах)	Опис
Початок кадру		1	Сигналізує початок передачі кадру
Поле арбітражу	Ідентифікатор	11	Унікальний ідентифікатор
	Запит на передачу (RTR)	1	Повинен бути домінантним
Поле управління	Біт розширення ідентифікатора (IDE)	1	Повинен бути домінантним
	Зарезервований біт (r0)	1	Резерв
	Довжина даних (DLC)	4	Довжина поля даних в байтах (0-8)
Поле даних		0-8 байт	Передані дані (довжина в поле DLC)
Поле CRC	Контрольна сума (CRC)	15`	Контрольна сума всього кадру
	Розмежувач контрольної суми	1	Повинен бути рецесивним
Поле підтвердження	Проміжок підтвердження (ACK)	1	Передавач шле рецесивний, приймач вставляє домінанту

	Розмежувач підтвердження	1	Повинен бути рецесивним
Кінець кадру (EOF)		7	Повинен бути рецесивним

Перші 7 біт ідентифікатора не повинні бути все рецесивними.

Таблиця 1.15 - Розширений формат кадру даних

Поле		Довжина (в бітах)	**Опис**
Початок кадру		1	Сигналізує початок передачі кадру
Поле арбітражу	Ідентифікатор A	11	Перша частина ідентифікатору
	Підміна запиту на передачу (SRR)	1	Повинен бути рецесивним
	Біт розширення ідентифікатора (IDE)	1	Повинен бути рецесивним
	Ідентифікатор B	18	Друга частина ідентифікатору
	Запит на передачу (RTR)	1	Повинен бути домінантним
Поле управління	Зарезервовані біти (r1 и r0)	2	Резерв
	Довжина даних DLC)	4	Довжина поля даних в байтах (0-8)
Поле даних		0-8 байт	Передані дані (довжина в поле DLC)
Поле CRC	Контрольна сума (CRC)	15	Контрольна сума всього кадру
	Розмежувач контрольної суми	1	Повинен бути рецесивним
Поле підтвердження	Проміжок підтвердження (ACK)	1	Передавач шле рецесивний, приймач вставляє домінанту
	Розмежувач підтвердження	1	Повинен бути рецесивним
Кінець кадру (EOF)		7	Повинен бути рецесивним

Ідентифікатор знаходять об'єднанням частин A і B.

Поле початку кадра (Start of frame) складається з одиночного нульового біта. Вузлу дозволено розпочати передачу, коли шина вільна Всі вузли повинні синхронізуватися по фронту, викликаному передачею поля "початок кадру" вузла, який розпочав передачу першим.

Поле арбітражу (Arbitration field) відрізняється для стандартного і розширеного форматів.

- В *стандартному форматі* полі арбітражу, складається з 11 розрядного ідентифікатора і RTR-біта. Біт RTR запиту віддаленої передачі в кадрах даних біт повинен бути переданий нульовим рівнем, в кадрах віддаленого запиту - одиничним.

- В *розширеному форматі* полі арбітражу складається з 29 розрядного ідентифікатора, SRR-біта, IDE-біта, і RTR-біта.

Замінник біта віддаленого запиту біт SRR, передається в розширених кадрах в позиції RTR біта. Таким чином, він замінює RTR - біт стандартного кадру.

Біт IDE розширення ідентифікатора в стандартному форматі передається нульовим рівнем, в той час як в розширеному форматі IDE біт - одиничний рівень.

Поле управління (Control field) складається з шести біт. Формат поля управління відрізняється для стандартного і розширеного формату.

Кадри в *стандартному форматі* включають: код довжини даних (DLC), біт IDE, який передається нульовим рівнем, і зарезервований біт r0.

Кадри в *розширеному форматі* включають код довжини даних і два зарезервованих біта r1 і r0. Зарезервовані біти повинні бути надіслані нульовим рівнем.

Код довжини даних (Data length code) розміром 4 біта, передається всередині поля управління.

Допустима кількість байт даних: {0,1, ..., 7,8}.

Інші величини використовуватися не можуть.

Поле даних (Data field) складається з даних, які будуть передані всередині кадру даних. Воно може містити від 0 до 8 байт, які передаються, починаючи з MSB.

Поле контрольної суми CRC (CRC field) містить послідовність CRC і CRC – роздільник, рис. 1.52

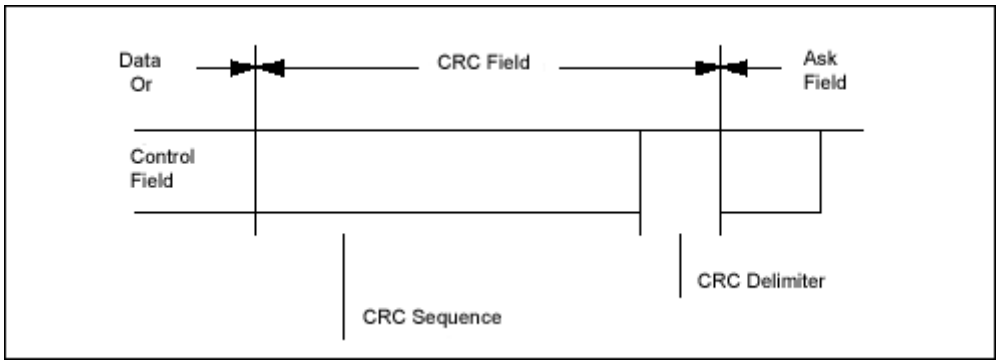


Рис. 1.52. Послідовність CRC (стандартний і розширений формати)

Послідовність контролю кадру CRC одержується з надлишкового циклічного коду. При обчисленні CRC спочатку визначається поліном, коефіцієнти якого задані послідовністю біт, що складається з полів: "початок кадру", "поле арбітражу", " поле управлыння", "поле даних" (якщо воно є) . 15 наймолодших коефіцієнтів приймаються рівними 0. Далі цей поліном ділеться на поліном виду:

X ^ 15 + X ^ 14 + X ^ 10 + X ^ 8 + X ^ 7 + X ^ 4 + X ^ 3 + 1.

Залишок від цього полиномиального ділення і є послідовність CRC.

Послідовність CRC супроводжується роздільником CRC(CRC Delimiter), який складається з одного одиничного біта.

Поле підтвердження (ACK field) має довжину два біти: "область підтвердження" і роздільник підтвердження. У полі підтвердження передавальний вузол посилає два біти з одиничним рівнем. Приймач, який отримав повідомлення правильно, повідомляє про це передавача, посилаючи біт з нульовим рівнем протягом прийому поля "область підтвердження".

Область підтвердження (ACK slot)

Всі вузли, які отримали відповідну послідовність CRC, повідомляють про це під час прийому поля "область підтвердження" шляхом заміни біта з одиничним рівнем на біт з нульовим рівнем.

Роздільник підтвердження (ACK delimiter):

Роздільник підтвердження - другий біт поля підтвердження, і він повинен бути представлений поодиноким рівнем.

Кінець кадру (End of frame) Кожен кадр даних і кадр віддаленого запиту даних обмежений послідовністю прапорів, що складається, з семи одиничних біт.

Кадри віддаленого запиту (Remote frame)

Збігаються з кадрами даних стандартного або розширеного формату за двома винятками:

• У полі RTR рецесії замість домінанти.

• Відсутня поле даних.

Вузол, що діє як приймач деяких даних, може ініціювати передачу відповідних даних вихідними вузлами, посилаючи кадр віддаленого запиту даних.

Кадр віддаленого запиту даних існує і в стандартному форматі і розширеному форматі. В обох випадках він складається з шести бітових полів:

"Початок кадру" (Start of frame), "поле арбітражу" (Arbitration field), "керуюче поле" (Control field), "поле CRC" (CRC - field), "поле підтвердження" (ACK field), "кінець кадру "(End of frame).

Кадр помилки (Error frame)

Кадр помилки складається з двох полів. Перше поле є суперпозиція прапорів помилки, отриманих від різних вузлів. Наступне поле - роздільник помилки (Error Delimiter) (Рис. 1.53)

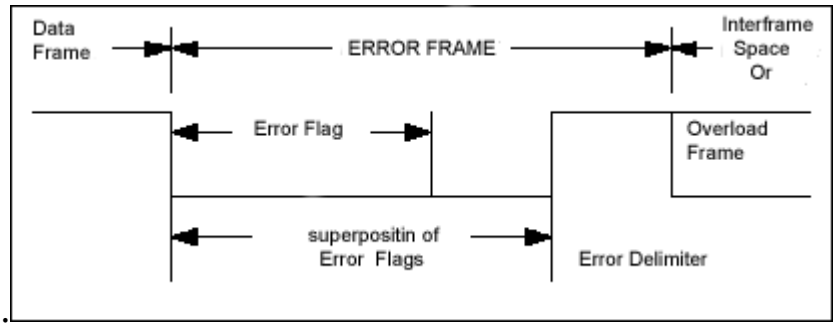


Рис 1.53- Формат кадра помилки

Для правильного завершення кадру помилки, вузол в стані "пасивної помилки" може потребувати доступу до шини, принаймні, на 3 бітових інтервали (якщо має місце локальна помилка приймача, що знаходиться в стані "пасивної помилки"). Отже, шина не повинна бути завантажена на 100%.

Прапор помилки (Error flag) може бути прапором активної або пасивної помилки.

1. Прапор активної помилки складається з шести послідовних нульових біт.

2. Прапор пасивної помилки складається з шести послідовних одиничних біт, якщо вони не перезаписані нульовими бітами інших вузлів.

Вузол у стані "активної помилки", виявивши умова помилки, повідомляє про це передачею прапора активної помилки. Форма прапора помилки порушує закон заповнення біта (див. кодування) вживаний до всіх полів від поля "початок кадру" до роздільника CRC або руйнує фіксовану форму поля підтвердження або поля "кінець кадру". Як наслідок, всі інші вузли виявляють умова помилки і в свою чергу починають передачу прапора помилки. Таким чином, послідовність домінантних бітів, яка фактично може з'явитися на шині, створюється суперпозицією різних прапорів помилки, переданих окремими вузлами. Сумарна довжина цієї послідовності змінюється від шести до дванадцяти біт.

Вузол, у стані "пасивної помилки", виявивши умову помилки, пробує повідомити про це передачею прапора пасивної помилки, він очікує появи шести послідовних однакових біт, початківців прапор пасивної помилки. Прапор пасивної помилки завершений, коли після виявлення цих 6 біт.

Роздільник помилки складається з восьми рецесивних бітів. Після передачі прапора помилки кожен вузол посилає рецесивні біти і перевіряє шину, поки не виявляє рецесивний біт. Далі він починає передачу ще семи рецесивних бітів.

Кадр перевантаження

Кадр перевантаження містить два бітові поля: прапор перевантаження і роздільник перевантаження. Є три причини перевантаження:

1. Внутрішній стан приймача, що вимагає затримки наступного кадру даних або кадру віддаленого запиту даних.

2. Виявлення домінантного біта при передачі першого і другого бітів перерви.

3. Якщо вузол виявляє домінантни біт на восьмому біті (останньому біті) роздільника помилки або роздільник перевантаження, це спричинить передачу кадру перевантаження (а не кадру помилки). Лічильники помилок не будуть збільшені.

Передачу кадру перевантаження, обумовленого 1-м видом перевантаження, дозволено починати в першому бітовому інтервалі перерви, в той час як кадр перевантаження, обумовлений 2-м і 3-м видами перевантаження, починає передаватися на один біт пізніше виявлення домінантого бита.

Не більше двох кадрів перевантаження може бути згенеровано, щоб затримати наступний кадр даних або кадр віддаленого запиту даних.

Прапор перевантаження складається з шести домінантних бітів. Повна форма відповідає прапору активної помилки. Форма прапора перевантаження порушує фіксовану форму поля перерви. Всі інші вузли також виявляють умову перевантаження і в свою чергу починають передачу прапора перевантаження. У разі якщо виявлений домінантний біт, під час 3-го біта перерви, то цей біт інтерпретується як початок кадру.

Роздільник перевантаження складається з восьми рецесивних біт. Роздільник перевантаження має таку ж форму, як і роздільник помилки. Після передачі прапора перевантаження вузол контролює шину, поки не виявить перехід від домінантного бита до рецессивного біту. У цей момент

часу кожен вузол закінчує передачу прапора перевантаження, і всі вузли починають передачу ще 7 рецесивних бітів.

Міжкадровий простір

Кадри даних і кадри віддаленого запиту даних відокремлюються від попередніх кадрів будь-якого типу (кадр даних, кадр віддаленого запиту даних, кадр помилки, кадр перевантаження) бітовим полем, званим міжкадрових простором. На відміну від них, кадрам перевантаження і кадрам помилки не передує міжкадровий простір (кратні кадри перевантаження також не відокремлюються інтервалами).

Міжкадровмй простір: містить бітові поля: перерва і простій шини.

Рис 1.57 а наведено для вузла, який був джерелом попереднього повідомлення.

Рис 1.57 в наведено для вузла в стані "пасивної помилки", який був джерелом попереднього повідомлення

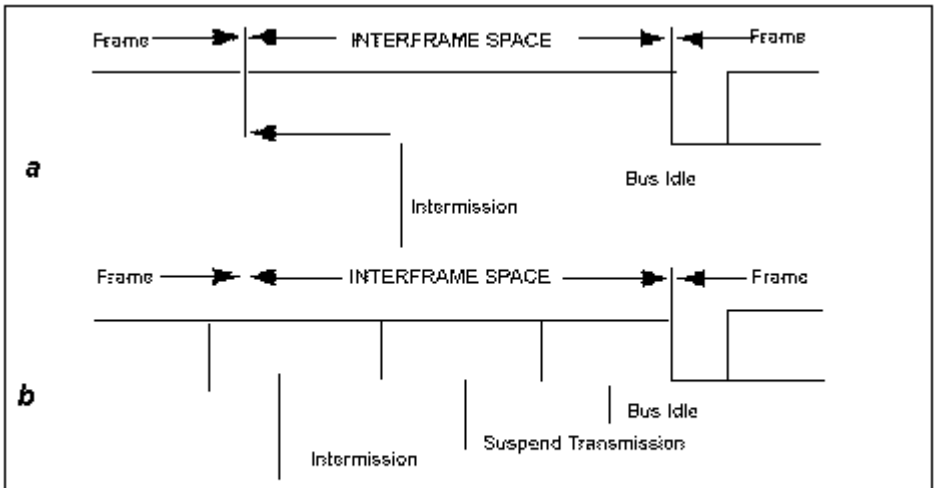


Fig 10. INTERFRAME SPACE
a) for not 'error passive' stations having been TRANSCEIVER of previous message
b) for 'error passive' stations having been TRANSMITTER of the previous message

.

Рис. 1.54 - Міжкадровий простір

Поле перерва складається з трьох рецесивних біт.

Протягом перерви єдина дія, що можна зробити - повідомлення про умови перевантаження і ніякої вузол не може почати передачукадру даних або кадру віддаленого запиту даних.

Період простою шини може мати довільну довжину. Шина вільна і будь-який вузол, якому потрібно щось передати, може звертатися до шини. Повідомлення, яке чекає передачі під час передачі іншого повідомлення, починає передаватися в першому бите після перерви. Виявлення домінантного біта на шині інтерпретується як "початок кадру".

призупинення передачі

Після того, як вузол в стані "пасивної помилки" передав повідомлення, він посилає вісім рецесивнбіт після яких йде перерву, перед передачею наступного повідомлення або розпізнавання простою шини.

Якщо тим часом починається передача, викликана іншим вузлом, вузол стає приймачем цього повідомлення.

Фільтрація повідомлень

Фільтрація повідомлення заснована на ідентифікації.

Існують спеціальні регістри масок, які дозволяють установку будь-якого біта ідентифікатора, можуть використовуватися для вибору групи ідентифікаторів, які будуть відображені в буферах.

Якщо регістри масок є, то кожен біт регістрів маски повинен бути програмованим, тобто його можна використовувати для фільтрації повідомлення. Довжина регістра маски може становити цілий ідентифікатор або тільки його частина.

Кодування послідовності бітів

Сегменти: "початок кадру", "поле арбітражу", "поле управління", "поле даних" і "послідовність CRC" кодуються методом розрядного заповнення.

Всякий раз, коли передавач виявляє в розрядному потоці, п'ять послідовних однакових біт, він автоматично вставляє додатковий біт в фактичний переданий розрядний потік.

Решту бітові поля кадру даних або кадру віддаленого запиту даних ("роздільник CRC", "поле підтвердження", і "кінець кадру") мають фіксовану

форму. Кадр помилки і кадр перевантаження мають фіксовану форму також і не кодуються методом розрядного заповнення.

Розрядний потік повідомлення кодується згідно NRZ методом (без повернення до нуля). Це означає, що протягом усього часу передачі бітів згенерований розрядний рівень є або "домінант" або "recessive".

Виявлення помилок

CAN містить 5-ступінчастий механізм виявлення помилок:

- циклічний контроль по надмірності (CRC)
- контроль передаваного поля бітів
- контроль сигналу "Підтвердження Прийому"
- поточний контроль логічного рівня бітів
- контроль заповнення бітів.

Циклічний контроль по надмірності (CRC)

Кожне передане повідомлення містить контрольний код (CRC), обчислений передавачем на основі вмісту передаваного повідомлення. Приймальні вузли виконують аналогічну операцію, позначають виявлені помилки і встановлюють відповідні прапори.

Поточний контроль логічного рівня бітів

Будь-який передавач автоматично контролює і порівнює фактичний логічний рівень бітів на шині з рівнем, який він передає. Якщо рівні не збігаються, позначається помилка логічного рівня бітів. Виняток - посилка рецесивнбита протягом заповненого бітового потоку поля арбітражу або протягом поля підтвердження.

Передавач, який посилає прапор пасивної помилки і виявив "домінант" біт не інтерпретує це як помилку біта.

(Примітка: цей механізм також використовується при арбітражі шини для визначення пріоритету повідомлення, проте помилка в цьому випадку, природно, не виникає).

Контроль передаваного поля бітів

У складі CAN-повідомлення передаються зумовлені бітові комбінації, які контролюються при прийомі. Якщо приймач виявляє недопустимий біт в одній з цих комбінацій, то встановлюється прапор помилки формату.

Контроль заповнення бітів

CAN використовує методику додавання заповнюючого біта для додаткового контролю передаваних повідомлень. Після передачі п'яти послідовних бітів з однаковим рівнем передавач автоматично вводить в розрядний потік біт протилежного значення. Приймачі повідомлення автоматично видаляють такі біти перед обробкою повідомлення. Якщо виявляється шостий біт однакової полярності, то позначається помилка заповнення бітів.

Контроль сигналу "Підтвердження Прийому"

Кожне передане повідомлення підтверджується приймачем, і якщо цього не сталося, тоді встановлюється прапор помилки підтвердження прийому.

Прапор помилки

У випадку якщо виявлена помилка, то вузол, що виявив помилку, перериває передачу посилкою прапора помилки. При цьому передавач автоматично реинициализирует передачу повідомлення, що запобігає всім вузлам від виникнення помилок і гарантує несуперечність даних в мережі.

З врахуванням дії всіх механізмів контролю, реальне значення виникнення невиявленої помилки в CAN- мережі – $10^{-11.}$

Вузол, що виявив помилку повідомляє про це, передаючи прапор помилки.

Типізація несправностей

Несправний вузол може бути в одному з трьох станів:

• активної помилки

• пасивної помилки

• відключення від шини

Вузол у стані "активної помилки" нормально приєднаний до шини і посилає прапор активної помилки при виявленні помилок.

Вузол у стані "пасивної помилки" не повинен посилати прапор активної помилки. Він підключений до шини, але при виявленні помилок, посилає прапор пасивної помилки. Після передачі, вузол в стані "пасивної помилки" буде чекати ініціалізації подальшої передачі Вузол у стані "відключення від шини" не може працювати з шиною.

Для типізації несправностей в кожному вузлі є два лічильника:

1) Лічильник помилок передачі
2) Лічильник помилок прийому

Значення лічильника помилок більше ніж 96 вказує на те, що шина сильно пошкоджена.

Якщо в CAN - мережі підключений тільки один вузол, і якщо цей вузол передає деяке повідомлення, він не отримає підтвердження, знайде помилку і повторить повідомлення. Через це він може перейти в стан "пасивної помилки", але не в стан "відключений від шини".

Арбітраж CAN-шини

При вільній шині будь-який вузол може починати передачу в будь-який момент. У разі одночасної передачі кадрів двома і більше вузлами проходить арбітраж доступу: передаючи адресу джерела, вузол одночасно перевіряє стан шини. Якщо при передачі рецесивного біта приймається домінантний - вважається, що інший вузол передає повідомлення з великим пріоритетом і передача відкладається до звільнення шини. Таким чином, на відміну, наприклад, від Ethernet в CAN не відбувається непродуктивної втрати пропускної здатності каналу при колізіях. Ціна цього рішення - ймовірність того, що повідомлення з низьким пріоритетом ніколи не будуть передані.

Оскільки всім вузлам дозволено починати передачу кадрів після того, як шина виявиться вільною, це може привести до того, що в один і той же час

відразу декілька вузлів почнуть передачу. Щоб запобігти руйнуванню одним вузлом кадру іншого вузла, вузол під час передачі індикатора і RTR біта контролює шину. Якщо при передачі рецесивного біта (recessive bit) він виявляє домінуючий біт (домінант bit), то звільняє шину, негайно зупиняє передачу і продовжує приймати кадр.

Часова діаграма приведена на рис.1.55 Початок кадру позначений домінантним стартовим бітом, за яким слідує 11 бітовий ідентифікатор і арбітражний біт RTR, яким розрізняють кадр даних і кадр запиту даних (видалений кадр).

У полі управління вказується тип формату повідомлення (стандарт / розширений) і довжина наступного поля даних (0-8 байт). Після даних слідує 15 бітовий сегмент CRC, який дозволяє одержуючому вузлу перевіряти придатність отриманих даних.

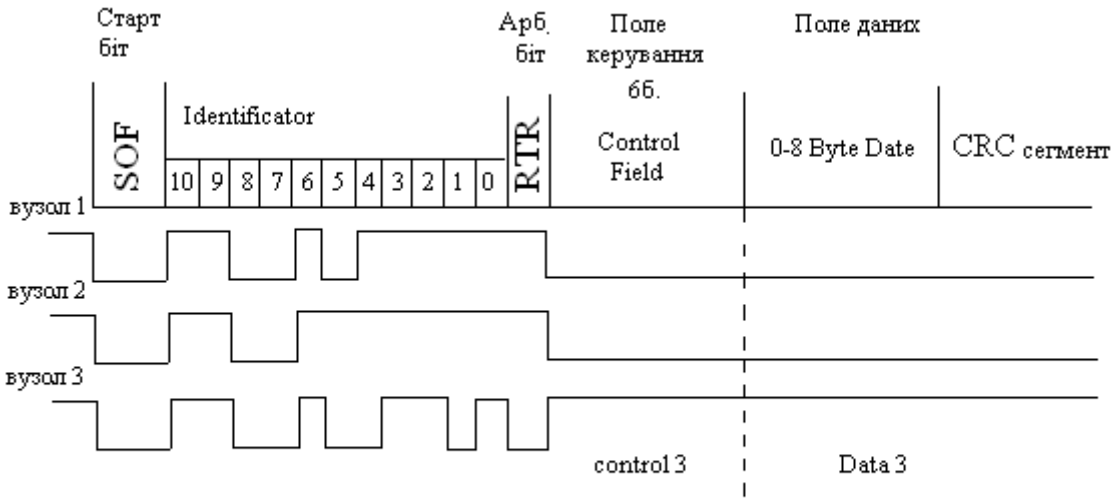


Рис. 1.55

У будь-якій системі деякі з параметрів змінюються швидше, ніж інші. Наприклад, швидкість ротора двигуна, як правило, змінюється за менший проміжок часу, чим температура його корпусу або положення заслінки. Параметри, що швидко змінюються, повинні передаватися частіше і, отже, вимагають вищого пріоритету. Під час роботи також можлива поява

аварійних повідомлень, які повинні передаватися з найвищим пріоритетом (наприклад, перевищення допустимої температури, обрив соленоїда, що управляє, коротке замикання в ланцюзі і так далі).

Вузли CAN- мережі є рівноправними при обміні, і кожен з них у будь-який момент часу може мати повідомлення, що вимагає невідкладної передачі. Вірогідність одночасної вимоги передачі від різних пристроїв не є чимось надзвичайним, а трапляється регулярно. Для вирішення подібного конфлікту потрібний швидкодіючий механізм розподілу черговості передачі повідомлень. Для цього в CAN- мережі використовується Неруйнівний Порозрядний Арбітраж.

Пріоритет CAN-повідомлення визначається двійковим значенням його ідентифікатора.

Числове значення кожного ідентифікатора повідомлення призначається в початковій фазі проектування системи. Ідентифікатор з найнижчим числовим значенням ідентифікатора має найвищий пріоритет. Передача логічного нуля по CAN-шині здійснюється струмовою посилкою, а стан логічної одиниці визначається по відсутності струму. В процесі передачі кожне з джерел повідомлень, який має необхідність в передачі, починає передавати свій ідентифікатор, одночасно перевіряючи його на лінії. Якщо в процесі передачі виявляється неспівпадання (тобто "зайвий" нуль), то передавач, що виявив це невідповідність, припиняє передачу свого ідентифікатора і перемикається на прийом. Конфлікту на шині при цьому немає, оскільки значення біта з рівнем логічної одиниці фактично не передається, і в результаті повідомлення з найвищим пріоритетом проходит по шині так, як ніби воно єдине. У наступному циклі шини буде передано повідомлення з нижчим пріоритетом, і так далі Таким чином досягається максимальна пропускна спроможність шини і мінімальна затримка для "гарячих" повідомлень.

Висока надійність

Для забезпечення безвідмовної роботи в тяжких умовах за стандартом ISO11898 CAN-контролер забезпечує роботу в мережі в наступних випадках:

- будь-який з 3-х дротів в шині обірваний;
- будь-який дріт - закорочений на живлення;
- будь-який дріт - закорочений на загальний дріт.

При обриві 2-х дротів частина функцій основної системи може бути реалізована в кожній з підсистем, створених обривом

Переваги та недоліки

Переваги

Можливість роботи в режимі жорсткого реального часу.

Простота реалізації і мінімальні витрати на використання.

Висока стійкість до перешкод.

Арбітраж доступу до мережі без втрат пропускної спроможності.

Надійний контроль помилок передачі і прийому.

Широкий діапазон швидкостей роботи.

Велике поширення технології, наявність широкого асортименту продуктів від різних постачальників.

Недоліки

Невелика кількість даних, яке можна передати в одному пакеті (до 8 байт).

Великий розмір службових даних в пакеті (по відношенню до корисних даними).

Відсутність єдиного загальноприйнятого стандарту на протокол високого рівня, проте ж, це і гідність. Стандарт мережі надає широкі можливості для практично безпомилкової передачі даних між вузлами, залишаючи розробникові можливість вкласти в цей стандарт все, що туди зможе помістится. У цьому відношенні CAN подібний простому електричний дріт. Туди можна «запхати» будь потік інформації, який зможе витримати пропускна здатність шини. Відомі приклади передачі звуку і зображення по

шині CAN (Росія). Відомий випадок створення системи аварійного зв'язку вздовж автодороги довжиною кілька десятків кілометрів (Німеччина). (У першому випадку потрібна була велика швидкість передачі і невелика довжина лінії, у другому випадку - навпаки). Виготовлювачі, як правило, не афішують, як саме вони використовують корисні байти в пакеті.

ЛЕКЦІЯ 6

Розподілена мікропроцесорна система на базі интерфейсу 1-Wire

Загальна характеристика

Однопровідний інтерфейс 1-Wire, розроблений в кінці 90-х років фірмою Dallas SemicoKTRWSZ 11nductor, регламентований розробниками для вживання в трьох основних сферах-застосуваннях:

- системи ідентифікації і контролю доступу (технологія iButton або Touch Memory);
- програмування вбудованої пам'яті інтегральних компонентів;
- системи автоматизації (технологія мереж MICROLAN).

Для прийому-передачі інформації використовується одна двонаправлена сигнальна лінія (другий дріт - земляний).

В системах ідентифікації і контролю доступу – (технології Touch Memory) обмін здійснюється в режимі напівдуплекса (або прийом, або передача). Взаємодія приладів по однопровідному інтерфейсу організована за принципом "веде-ведений" (master-slave). При цьому прочитуючий пристрій завжди веде, а один або декілька приладів Touch Memory - ведені. Взаємодія декількох приладів з прочитуючим пристроєм по одній двонаправленій лінії підтримується апаратними засобами Touch Memory.

Групу команд обміну з ПЗП складають чотири команди: читання ПЗП, пропуск, порівняння і пошук. Дві останні команди забезпечують взаємодію

по одній лінії декілька Touch Memory з прочитуючим пристроєм. Команда порівняння ініціює обмін з приладом, серійний номер якого вказаний. Команда пошуку дозволяє визначити серійний номер одного з приладів, підключених до двонапрямленої лінії.

Всі команди обміну мають фіксований розмір - один байт, дані представлені 8-розрядними цілими числами. Ведучій пристрій завжди ініціює обмін, посилаючи команди веденому пристрою.

Для забезпечення цілісності передаваної інформації протокол обміну на фізичному рівні строго регламентує часові параметри сигналів на лінії.

Програмування вбудованої пам'яті інтегральних компонентів забезпечує можливість легкої перебудови функцій напівпровідникових компонентів з малою кількістю зовнішніх виводів, вироблюваних фірмою Dallas Semiconductor

Системи автоматизації на базімереж MicroLAN.

MICROLAN є інформаційною мережею, що використовує для здійснення цифрового зв'язку одну лінію даних і один зворотний (або земляний) дріт. (рис. 1.56) Таким чином, для реалізації середовища обміну цієї мережі можуть бути використані, як кабелі, що містять неекрановану виту пару тієї або іншої категорії, так і звичайний телефонний дріт. Подібні кабелі при їх прокладці не вимагають, як правило, наявності спеціального устаткування. Обмеження максимальної довжини однопровідної лінії, що реалізовується без спеціальних додаткових допоміжних пристроїв (повторювачів), регламентоване на рівні 300м.

Основою архітектури мереж MICROLAN, є топологія загальної шини, коли кожен з пристроїв підключений безпосередньо до єдиної магістралі, без яких-небудь каскадних з'єднань або розгалужень. При цьому як базова використовується структура мережі з одним ведучим або майстром і багатьма веденими, хоча існує ряд прийомів організації роботи подібних мереж в режимі мультимастера.

Протокол обміну по цьому інтерфейсу дуже простий і легко реалізується програмно практично на будь-яких МК.

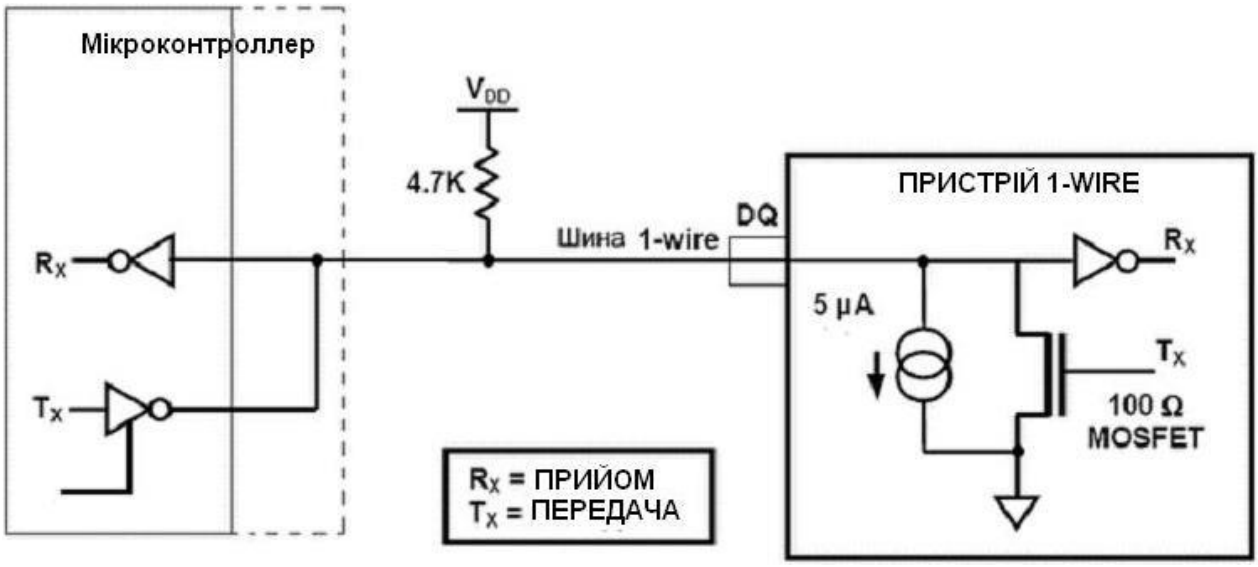


Рис. 1. 56. Схема обміну по 1-wire інтерфейсу

На рис.1.56 показана спрощена схема апаратної реалізації інтерфейсу 1-Wire. Виведення DQ пристрою є входом КМОП-логического елементу, який може бути зашунтірований (замкнутий на загальний дріт) польовим транзистором. Шина 1-Wire має бути підтягнута окремим резистором до напруги живлення пристроїв. Опір цього резистора 4.7 К, проте, це значення рекомендоване лише для досить коротких ліній. Якщо шина 1-Wire використовується для підключення віддалених на велику відстань пристроїв, то опір цього резистора слід зменшити. Мінімальний допустимий опір - близько 300 Ом, а максимальний - близько 20-30 кілоом.

Підключення шини 1-Wire до МК на рис. 1.56 показано умовно в двох варіантах: з використанням 2 окремих виводів МК (один як вихід, а інший як вхід), так і одного, що працює і на введення і на вивід.

Обмін інформацією по шині 1-Wire

1) Обмін завжди ведеться за ініціативою одного ведучого пристрою, який в більшості випадків є мікроконтролером (МК).Для інтерфейсу 1-Wire в

загальному випадку передбачається "гаряче" підключення і відключення пристроїв. Будь-який обмін інформацією починається з подачі імпульсу скидання ("Reset Pulse" або просто RESET) в лінію 1-Wire ведучім пристроєм.

2) Будь-який пристрій, підключений до 1-Wire після надання живлення видає в лінію DQ імпульс присутності, званий "Presence pulse" (далі я використовуватиму термін PRESENCE). Цей же імпульс пристрій завжди видає в лінію, якщо виявить сигнал RESET.

3). Поява в шині 1-Wire імпульсу PRESENCE після видачі RESET однозначно свідчить про наявність хоч би одного підключеного пристрою.

4). Обмін інформації ведеться так званими тайм-слотами: один тайм-слот служить для обміну одним бітом інформації.

5). Дані передаються побайтно, біт за бітом, починаючи з молодшого біта. Достовірність переданих/прийнятих даних (перевірка відсутності спотворень) гарантується шляхом підрахунку циклічної контрольної суми.

На рис.1.57. показана діаграма сигналів RESET і PRESENCE, з яких завжди починається будь-який обмін даними. Видача імпульсу RESET в процесі обміну служить для дострокового завершення процедури обміну інформацією.

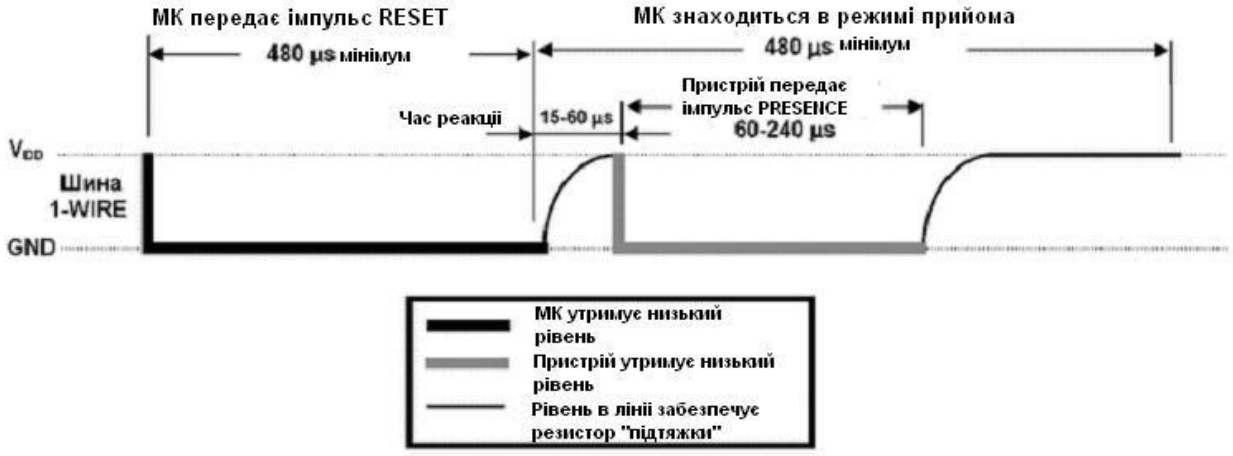


Рис. 1.57. Діаграма сигналів RESET і PRESENCE

Як бачимо, тривалість більшості часових інтервалів дуже приблизна і має лише обмеження лише по мінімуму (не менше вказаного). Умовні позначення ліній, показані на рис.1.57, використовуватимуться і далі.
Импульс RESET формує ведучий МК, переводячи в низький логічний рівень шину 1-Wire і утримуючи її в цьому стані мінімум 480 мікросекунд. Потім МК повинен "відпустити" шину. Через деякий час, залежний від ємкості лінії і опору підтягуючого резистора, в лінії встановиться високий логічний рівень. Протокол 1-Wire обмежує цей час "релаксації" діапазоном від 15 до 60 мікросекунд, що і є визначальним для вибору підтягуючого резистора (ємкість лінії міняти неможливо).

Після прийому імпульсу RESET, ведений пристрій приводить свої внутрішні вузли у вихідний стан і формує у відповідь імпульс PRESENCE, як випливає з рис.1.57. - не пізніше 60 микросекунд після завершення імпульсу RESET. Для цього пристрій переводить в низький рівень лінію DQ і утримує її в цьому стані від 60 до 240 мікросекунд. Конкретний час утримання залежить від багатьох параметрів, але завжди знаходиться у вказаному діапазоні. Після цього пристрій так само "відпускає" шину.
Після завершення імпульсу PRESENCE пристрою дається ще деякий час для завершення внутрішніх процедур ініціалізації, таким чином, МК повинен приступити до будь-якого обміну з пристроєм не раніше, чим через 480 мікросекунд після завершення імпульсу RESET.
Итак, процедура ініціалізації інтерфейсу, з якою починається будь-який обмін даними між пристроями, триває мінімум 960 мікросекунд, складається з передачі від МК сигналу RESET і прийому від пристрою сигналу PRESENCE. Якщо сигнал PRESENCE не виявлений, це означає що на шині 1-Wire немає готових до обміну пристроїв.

Розглянемо процедури обміну бітами інформації, які, здійснюються певними тайм-слотами. Тайм-слот - це жорстко лімітована за часом послідовність зміни рівнів сигналу в лінії 1-Wire. Далі будемо

використовувати термін МК, як синонім "провідного пристрою" і просто "пристрій", як синонім "ведений". Розрізняють 4 типи тайм-слотів: передача "1" від МК, передача "0" від МК, прийом "1" від пристрою і прийом "0" від пристрою.

Кожен тайм-слот завжди починає МК шляхом переведення шини 1-Wire в низький логічний рівень. Тривалість будь-якого тайм-слота повинна знаходитися в межах від 60 до 120 мікросекунд. Між окремими тайм-слотами завжди повинен передбачатися інтервал не менше 1 мікросекунди (конкретне значення визначається параметрами веденого пристрою).

Тайм-слоты передачі відрізняються від тайм-слотів прийому поведінкою МК: при передачі він лише формує сигнали, при прийомі, крім того, ще і опитує (тобто приймає) рівень сигналу в лінії 1-Wire. Рис.1. 58. демонструє часові діаграми тайм-слотів всіх 4-х типів: вгорі показані тайм-слоти передачі від МК, внизу - прийому від пристрою.

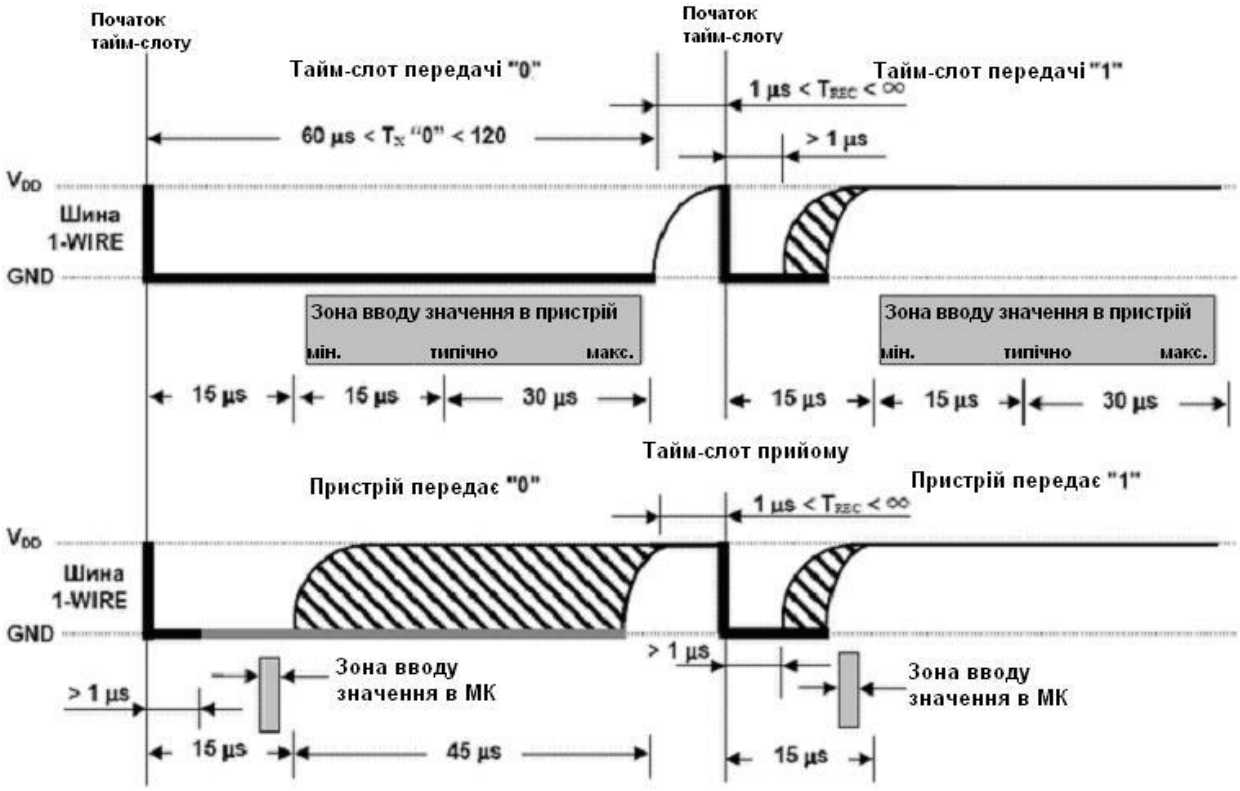


Рис.1.58. Тайм-слоти

Тайм-слот передачі "0" полягає просто в утриманні шини 1-Wire в низькому рівні протягом всієї тривалості тайм-слота. Передача "1" здійснюється шляхом "відпуску" шини 1-Wire з боку МК не раніше чим через 1 мікросекунду після початку тайм-слота, але не пізніше ніж через 15 мікросекунд. Ведений пристрій опитує рівень в шині 1-Wire протягом часового інтервалу, умовно показаного у вигляді сірого прямокутника, тобто починаючи з 15-ої мікросекунди від початку тайм-слота і закінчуючи 60-ою мікросекундою від початку. Типовий момент введення рівня в пристрій (тобто характерний для більшості пристроїв) - біля 30-ої мікросекунди від початку тайм-слота.

Заштрихована область - це область "наростання" рівня в шині 1-Wire, яка залежить від ємкості лінії і опору підтягуючого резистора, вона приведена для довідки.

Тайм-слоти прийому інформації відрізняються тим, що МК формує лише початок тайм-слота (абсолютно так само, як при передачі "1"), а потім управління рівнем шини 1-Wire бере на себе пристрій, а МК здійснює введення цього рівня так само в певній зоні часових інтервалів. Зона ця, як видно з рис.1.61., досить мала. Контролер повинен ввести рівень сигналу з лінії на 13-15-ій мікросекунді від початку тайм-слота.

Таким чином, МК починає тайм слот видачою в шину 1-Wire "0" протягом 1 мікросекунди. Подальший рівень залежить від типа тайм слота: для прийому і передачі "1" рівень повинен стати високим, а для передачі "0" - залишатися низьким до кінця тайм-слота, тобто не менше 60 і не більше 120 мікросекунд. Якщо МК приймає дані, то опит рівня в шині він повинен зробити на проміжку від 13-ої до 15-ій мікросекунді тайм-слота. МК повинен забезпечити інтервал між окремими тайм-слотами не менше 1 мікросекунди. Слід точно забезпечувати в шині 1-Wire необхідні часові інтервали, оскільки, наприклад, збільшення тривалості тайм-слота виведення "0" зверху рекомендованого значення може привести до помилкового сприйняття цього

тайм-слота, як сигналу RESET. Всі сигнали, які повинен формувати МК, слід формувати за принципом необхідного мінімуму тривалості (тобто трохи більше, чим вказана мінімальна тривалість), а від пристрою слід чекати сигналів за принципом найгіршого (тобто орієнтуватися на самі гірші варіанти часових параметрів сигналу).

Кожен пристрій 1-Wire має унікальний ідентифікаційний 64-бітовим номер, що програмується на етапі виробництва мікросхеми. Унікальний - це означає, що фірма-виробник гарантує, що не знайдеться двох мікросхем з однаковим ідентифікаційним номером (принаймні впродовж декількох десятків років при існуючих темпах виробництва).

Протокол обміну

При розгляді протоколу обміну вважаємо що на шині 1-Wire є більш ніж один пристрій. В цьому випадку перед МК встають 2 проблеми: визначення кількості наявних пристроїв і вибір (адресація) одного конкретного з них для обміну даними. Вирішення першої проблеми здійснюється двома шляхами: універсальним і гнучким, але вимагаючим досить складного алгоритму, що і простим, при якому відомі номери всіх використовуваних у вашій схемі 1-Wire-пристроїв. Номери деяких пристроїв нанесені на корпусі мікросхем (наприклад, для пристроїв iButton - всім відомих ключів-пігулок), а номери інших можна визначити за допомогою спеціальних програм або пристроїв.

Хай відомі номери всіх пристроїв 1-Wire на шині. Алгоритм наступний: МК посилає, імпульс RESET, і все наявні пристрої видають PRESENCE. Потім МК посилає в шину команду, яку приймають всі пристрої. Команд визначено декілька загальних для всіх типів 1-Wire-пристроїв, а так само можуть бути команди, унікальні для окремих типів. Серед загальних команд найбільш поширені наступні (див. табл. 1.16.).

Таблиця 1.16 - Команди інтерфейсу 1-Wire

Команда	Значення байта	Опис
SEARCH ROM	0xF0	Пошук адрес - використовується при універсальному алгоритмі визначення кількості і адрес підключених пристроїв
READ ROM	0x33	Читання адреси пристрою - використовується для визначення адреси єдиного пристрою на шині
MATCH ROM	0x55	Вибір адреси - використовується для звернення до конкретної адреси пристрою з багатьох підключених
SKIP ROM	0xCC	Ігнорувати адресу - використовується для звернення до єдиного пристрою на шині, при цьому адреса пристрою ігнорується (можна звертатися до невідомого пристрою)

Перша команда використовується в складному універсальному алгоритмі, друга дозволяє визначити адресу пристроїв, перед їх установкою в готовий виріб, а дві останні напевно є основними.

Після того, як МК видасть команду READ ROM, від пристрою поступить 8 байт його власної унікальної адреси - МК повинен їх прийняти. Будь-яка процедура обміну даними з пристроєм має бути завершена повністю або перервана посилкою сигналу RESET.

Якщо відправлена команда MATCH ROM, то після неї МК повинен передати так само і 8 байт конкретної адреси пристрою, з яким здійснюватиметься подальший обмін даними. Це рівносильно виставлянню адреси на паралельній шині в мікропроцесорних пристроях. Прийнявши цю команду, кожен пристрій порівнює передавану адресу зі своєю власною. Всі пристрої, адреса яких не збіглася, припиняють аналіз і видачу сигналів в лінії 1-Wire, а пристрій, що пізнав адресу, продовжує роботу. Тепер всі дані, передавані МК потраплятимуть лише до цього "адресованному" пристрою. То, які саме дані треба послати в пристрій або отримати від нього після його адресації, залежить від конкретного пристрою Наприклад, для згаданого термометра це

можуть бути команди запуску вимірювання або прочитування результату, для ключа-пігулки не визначені жодні інші команди, окрім основних, а для мікросхем АЦП додаткових команд може бути близько десятка.
Якщо пристрій один на шині то можна прискорити процес взаємодії з ним за допомогою команди SKIP ROM. Отримавши цю команду, пристрій відразу вважає адресу такою, що збіглася, хоча жодної адреси за цією командою не слідує. Деякі процедури не вимагають прийому від пристрою жодних даних, в цьому випадку команду SKIP ROM можна використовувати для передачі якоїсь інформації відразу всім пристроям. Це можна використовувати, наприклад, для одночасного запуску циклу вимірювання температури декількома датчиками-термостатами типа DS18S20.
Прийом і передача байтів завжди починається з молодшого біта. Порядок дотримання байтів при передачі і прийомі адреси пристрою так само ведеться від молодшого до старшого. Порядок передачі іншої інформації залежить від конкретного пристрою, тому слід звертатися до документації на вживані вами пристрої.

Унікальна 64-бітова номер-адреса пристроїв 1-Wire складається з 8 байт: одного байта ідентифікатора сімейства, шести байт (48 біт) власне унікальної адреси і одного байта контрольної суми всіх попередніх байтів.
Контрольна сума або CRC - це байт, значення якого передається найостаннішим і обчислюється по спеціальному алгоритму на основі значення всіх 7 попередніх байтів. Алгоритм підрахунку такий, що якщо всі байти передані-прийняті без спотворень, прийнятий байт контрольної суми обов'язково збіжиться з розрахованим в МК (або пристрої) значенням. Тобто при реалізації програмного алгоритму обміну інформацією необхідно при передачі і прийомі байтів підраховувати їх контрольну суму по строго певному алгоритму, а потім або передати набутого значення, або порівняти розрахункове значення з набутого значення CRC. Лише при збігу обох CRC МК або пристрій вважають прийняті дані достовірними. Інакше продовження

обміну неможливе.

Вочевидь, що алгоритм підрахунку CRC має бути однаковим як для МК, так і для будь-якого пристрою. Він "стандартизований" і описаний в документації.

"Understanding and Using Cyclic Redundancy Checks with Dallas Semiconductor iButtonTM Products"Приклад програмної реалізації CRC алгоритму на мові асемблера.

Ця підпрограма використовує один байт пам'яті CRC для зберігання результату. Перед першим викликом цей байт необхідно обнулити. У акумуляторі - черговий прийнятий або передаваний байт. Після того, як всі байти передані/прийняті у комірку пам’яті CRC вийде контрольна сума. Підпрограма не псує жодних регістрів, окрім регістра стану.

```
DO_CRC:
PUSH ACC    ; зберігаємо акумулятор
PUSH B    ; зберігаємо регістр B
push ACC    ; зберігаємо байт даних
MOV B, #8    ; кількість бітів (лічильник циклів)
CRC_LOOP:
XRL A, CRC    ; вимикаюче АБО (XOR) з попереднім значенням контрольної суми
RRC A    ; зсув праворуч через прапор перенесення
MOV A, CRC    ; беремо останнє значення CRC
JNC ZERO    ; перехід, якщо не було перенесення
XRL A, #18H    ; оновлюємо значення CRC шляхом XOR з константой
ZERO:
RRC A    ; знову зсув CRC
MOV CRC, A    ; зберігаємо нове значення CRC
POP ACC    ; відновлюємо байт даних
RR A    ; циклічно зсув праворуч
PUSH ACC    ; знову зберігаємо значення
```

```
DJNZ B, CRC_LOOP    ; повторюємо цикл 8 разів (для кожного біта)
POP ACC    ; очищаємо стек
POP B    ; відновлюємо колишні значення регістрів із стека
POP ACC
RET    ; завершення процедури
```

Використання цієї (та і подальшої) підпрограми дуже просте: перед початком прийому або передачі треба обнулити комірку CRC, а потім кожен прийнятий або переданий байт помістити в акумулятор і викликати цю підпрограму. Після того, як прийняті всі 8 (звернете увагу - саме 8!) байтів унікальної адреси пристрою, необхідно перевірити вміст комірки CRC: ненульове її значення свідчить про наявність спотворення прийнятих даних. Якщо ж CRC=0 - це означає, що дані прийняті без спотворень. Якщо ж МК вів передачу унікальної адреси пристрою, то вміст CRC має бути переданий 8-м байтом після попередніх семи.

Таким чином

- будь-який обмін інформацією починається з передачі імпульсу RESET і прийому имульса PRESENCE;
- якщо імпульсу PRESENCE не виявлено - на шині немає пристроїв;
- МК завжди ініціює обмін, починаючи кожен тайм-слот обміну бітом інформації;
- часові параметри кожного тайм-слота слід дотримувати з максимально можливою точністю;
- для вибору одного з багатьох пристроїв на шині 1-Wire МК повинен передати в шину команду MATCH ROM і потім 8 байт адреси пристрою, останній (8-й) байт цієї адреси - є контрольна сума попередніх семи;
- якщо пристрій на шині один - МК може взнати його адресу шляхом посилки команди READ ROM, після чого прийняти від пристрою 8 байтів адреси, останній з яких так само буде контрольною сумою перших семи;
- для роботи з єдиним пристроєм на шині можна відмовитися від вказівки

його адреси, для цього МК повинен передати пристрою команду SKIP ROM, після чого можна починати звичайний обмін даними;

- будь-яка почата процедура обміну може тривати скільки завгодно довго за рахунок пауз між окремими тайм-слотами, але завжди має бути завершена повністю;

- перервати початий обмін можна у будь-який момент шляхом видачі імпульсу RESET в шину 1-Wire (але це може порушити нормальну роботу деяких пристроїв).

ЛЕКЦІЯ 7

Інтерфейс USB

Інтерфейс USB призначений для легкого підключення різних типів пристроїв, як то - клавіатури, миші, джойстики, колонки, модеми, мобільні телефони, стрічкові, дискові, оптичні і магнітооптичні накопичувачі, флеш-диски, сканери і принтери, дігітайзери. Шина USB може виступати також як інтерфейс для підключення пристроїв цифрової мережі з інтегрованими послугами (ISDN) і цифрових пристроїв Private Branch eXchange (PBX) до комп’ютерів.

Специфікація периферійної шини USB розроблена лідерами комп'ютерної і телекомунікаційної промисловості -- Compaq, DEC, IBM, Intel, Microsoft, NEC і Northern Telecom -- для підключення комп'ютерної периферії поза корпусом машини за стандартом plug'n'play, в результаті відпадає необхідність в установці додаткових плат в слоти розширення і переконфігурації системи. Персональні комп'ютери, що мають шину USB, дозволяють підключати периферійні пристрої і здійснюють їх автоматичну конфігурацію, як тільки пристрій фізично буде приєднаний до машини, і при цьому немає необхідність перезавантажувати або вимикати комп'ютер, а так само запускати програми установки і конфігурації.

Шина USB дозволяє одночасно підключати послідовно до 127 пристроїв, таких, як монітори або клавіатури, що виконують роль додатково підключених компонентів, або хабов (тобто пристрій, через який підключається ще декілька).

USB визначає, ввімкнений пристрій або відключений, автоматично визначає який системний ресурс, включаючи програмний драйвер і пропускну спроможність, потрібний кожному периферійному пристрою і робить цей ресурс доступним без втручання користувача.

Операційна система Windows 95 (починаючи з версії OSR 2.1, випущеної 29 жовтня 1996р.) поставляється вже зі вбудованими драйверами, які дозволяє персональному комп'ютеру розпізнавати USB периферію. В результаті не потрібно купувати або інсталювати додаткове програмне забезпечення для кожного нового периферійного пристрою.

Технология USB 2.0

Відомо, що USB шина дозволяє підключати до 127 пристроїв. І в теж час на задній стінці зазвичай знаходиться 2 або 4 порти. Справа все в тому, що шина USB дозволяє багаторівневе каскадування (рис. 1.59).

Таким чином, перша архітектурна особливість шини USB: її логічна топологія - багаторівнева зірка, (рис. 1.60).

Самим верхнім рівнем є кореневий концентратор (хаб), який зазвичай поєднується з USB контроллером. Його функція така ж, що і концентраторів мереж передачі даних - додавання нових портів для підключення більшого числа пристроїв (розгалуджувач).

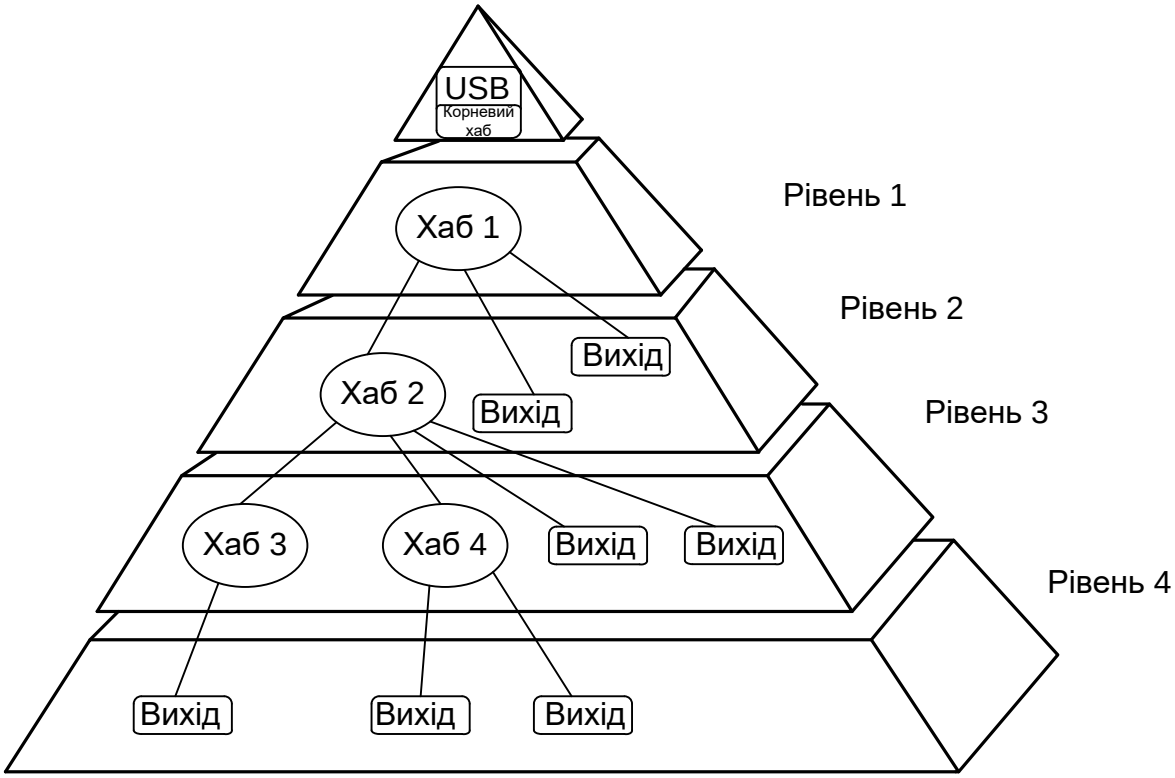


Рис.1.59.

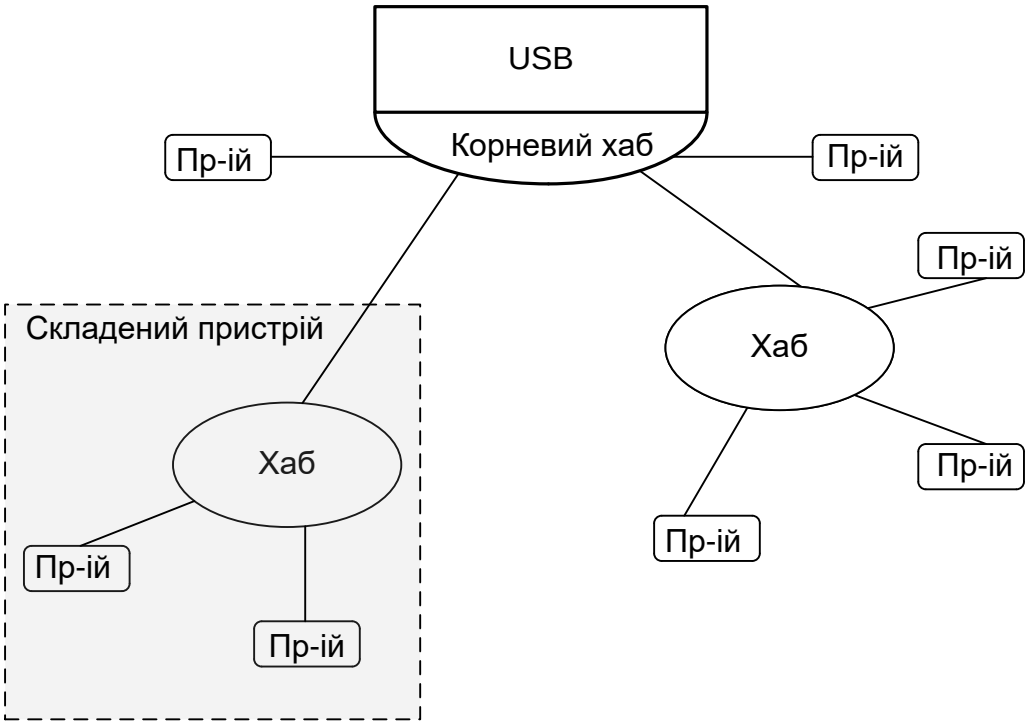


Рис.1.60.

До кореневого концентратора можуть бути підключені або пристрої, або ще концентратори, для збільшення числа доступних портів. Допускається організація до п'яти рівнів. Концентратор може бути виконаний у вигляді окремого пристрою, або бути вбудованим в якийсь інший. З цієї точки зору пристрої підключаються до USB можна підрозділити на функціональні пристрої, тобто ті які виконують якусь конкретну функцію і не беруть на себе жодних додаткових завдань (наприклад, миші). Пристрої-концентратори які виконують лише функцію розгалуження, і поєднані (комбіновані) пристрої, тобто що мають в своєму складі концентратор, розширюють набір портів дозволяють підключати інші пристрої (як приклади, що найчастіше зустрічаються, можна назвати монітори, що дозволяють по USB здійснювати налаштування параметрів, зазвичай мають ще декілька додаткових портів, для підключення інших пристроїв або клавіатури, з роз'ємами для підключення мишей).

На п'ятому рівні комбінований пристрій використовуватися не може. Крім того окремо варто згадати про хоста, що є швидше програмно-апаратним комплексом, ніж просто пристроєм. Логічна топологія шини - зірка (рис. 1.61)

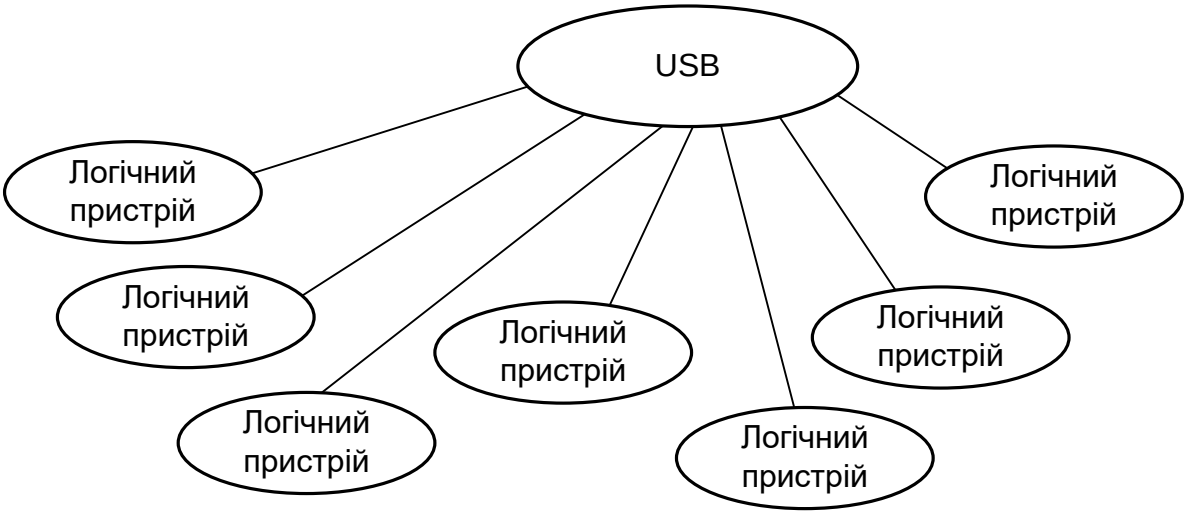


Рис.1.61.

Це пояснюється тим, що кожен концентратор забезпечує прозоре для хоста з'єднання з пристроєм.

На відміну від звичних старих інтерфейсів, де взаємодію можна було (і потрібно) здійснювати звертаючись до пристрою по конкретних фізичних адресах пам'яті і портів введення виводу, USB надає для взаємодії програмний інтерфейс і лише його, дозволяючи клієнтському ПЗ існувати у відриві від конкретного підключеного до шини пристрою і його конфігурації. Для клієнтської програми USB - це лише набір функцій.

Функції хоста, концентратора (хаба)

Хост – це програмно-апаратний комплекс, в обов'язки якого входить:

- Стеження за підключенням і відключенням пристроїв;
- Організація керуючих потоків, між USB-пиладом і хостом;
- Організація потоків даних між USB-приладом і хостом;
- Контроль стану приладів і ведення статистики активності;
- Постачання підключених пристроїв електроживленням.

Аппаратной частью является хост контролер - посередник між хостом и пристроями на шині.

Програмні функції (вибірка пристроїв і їх конфігурація, управління енергоспоживанням, процесами передачі, пристроями на шині і самою шиною) покладені на операційну систему. Першою популярною операційною системою, в якій підтримка USB реалізована була в повному об'ємі стала Windows 98 Second Edition. Деякі пристрої можуть бути працездатними і під ранішими версіями (98 без SE, і зрідка 95), але далеко не все і не завжди.

Концентратор (хаб). Дозволяє множинні підключення до одного порту, створюючи додаткові порти. Кожен хаб має один висхідний порт, призначений для підключення до вільного порту, що є в наявності, і декілька

низхідних, до яких можуть бути підключені або знову концентратори, або кінцеві пристрої, або камбіновані пристрої, рис. 1.62

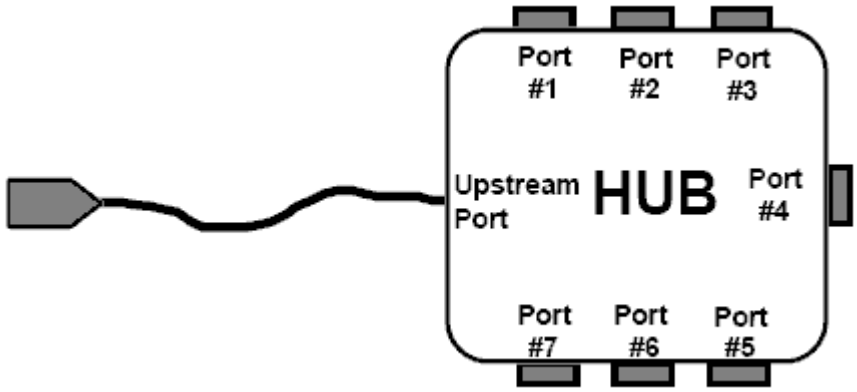


Рис.1.62.

Хаб повинен стежити за підключенням і відключенням пристроїв, повідомляючи хост про зміни, управляти живленням портів. У концентраторі стандарту USB 2.0 можна виділити 3 функціональних блоку: контролер, повторювач, транслятор транзакцій. Контролер відповідає за з'єднання з хостом. Поняття - повторювач в USB декілька відрізняється від прийнятого в мережах передачі даних. Його обов'язок - сполучати вхідний і якийсь потрібний з вихідних портів. Транслятор транзакцій з'явився лише в USB 2.0 і потрібний, як завжди, з міркувань сумісності з попередніми версіями. Коротко його суть в тому, що б забезпечувати максимальну швидкість з'єднання з хостом. Підключений до високошвидкісного (USB 2.0) порту старий повільний (USB 1.1) пристрій з'їдав би значну частину часу, а отже і корисній пропускній спроможності шини, ведучи обмін з хостом на низькій швидкості. Як метод боротьби транслятор транзакцій буферизує кадр, що поступає з повільного порту, а потім на максимальній швидкості передає його хосту, або ж буферизує отримуваний на максимальній швидкості кадр від хоста, передаючи його потім пристрою на меншій, прийнятнішій для

нього швидкості. Окрім розгалуження і трансляції транзакцій хаб повинен здійснювати конфігурацію портів і стеження за коректним функціонуванням підключених до них пристроїв. Потрібно сказати також, що при використанні старих і нових концентраторів разом можливе створення неоптимальних з точки зору продуктивності конфігурацій. Для того що б уникнути створення вузьких місць в своєму ланцюзі треба підключати низькошвидкісні пристрої до низькошвидкісних хабам, які у свою чергу останніми рівнями галуження і не підключати їх в середину високошвидкісного ланцюжка.

Функціональний пристрій.

З точки зору USB, пристрій - це набір кінцевих крапок з якими можливий обмін даними. Число і функції точок залежать від пристрою і виконуваних ним функцій, і визначаються при виробництві. В обов'язковому порядку присутня точка з номером 0 - для контролю стану пристрою і управління ним. До здійснення конфігурації пристрою через точку 0 інші канали не доступні. Кожна кінцева точка пристрою описується наступними параметрами:

- Частотою звернення до шини і вимогами до затримок
- Необхідною смугою пропускання
- Номером кінцевої точки
- Вимогами до обробки помилок
- Максимальним розміром кадру який може бути прийнятий або посланий.
- Типом підтримуваної передачі даних
- Напрямом здійснення передачі між кінцевою точкою і хостом.

Для низькошвидкісних (low-speed) пристроїв можливе існування до двох додаткових точок; для full-speed пристроїв їх число обмежується лише можливостями протоколу і може досягати 15-ти для введення і 15-ти для виводу.

Взагалі кінцева точка - це кінець логічного каналу даних між хостом і пристроєм. У свою чергу канал - це логічне з'єднання між хостом і пристроєм. Оскільки кінцевих крапок в пристрою передбачається декілька, то це означає, що обмін даними між хост-контролером і пристроєм на шині може виконуватись по декількох каналах, так званий багатоканальний режим. Смуга пропускання шини ділитися між всіма встановленими каналами.

Типи каналів

У розпорядження шина USB може надати канали наступних типів:

- Канали повідомлень. Є двонаправленими каналами і служать, для передачі повідомлень, що мають строго визначений в специфікації формат, необхідний для забезпечення надійної ідентифікації і передачі команд. Виникає канал при відсиланні хостом запиту в пристрій. Управляє передачею лише хост. Канали повідомлень використовується для передач, що лише управляють.
- Потокові канали. Є однонаправленими. На відміну від чітко певних повідомлень не мають певного закріпленого в стандарті формату, що означає можливість передачі даних будь-якого вигляду. Ці передачі можуть контролюватися не лише хостом, але і пристроєм. Використовується для передач даних типа переривання, групова пересилка, ізохронна. У специфікації залежно від типа передаваних даних, вимог, що пред'являються, до швидкості обробки, затримки доставки і тому подібне визначені наступні типи передач.
- Передачі, що управляють. Використовуються для конфігурації пристроїв під час підключення і виконання інших специфічних функцій над пристроєм, включаючи організацію нових каналів.
- Переривання. Використовуються для спонтанних, але гарантованих передач з гарантованими швидкостями і затримками. Використовуються зазвичай для передачі введених даних від

клавіатури або відомостей про зміну положення покажчика миші, в пристроях зворотного зв'язку, и.т.д

- Групова пересилка. Використовується для гарантованої передачі даних великих об'ємів без пред'явлених вимог до швидкостей і затримок. Займає під себе всю вільну пропускну спроможність шини. У будь-який момент доступна смуга може бути урізана при необхідності здійснення передач інших видів з вищим пріоритетом, або додана, при звільненні іншими пристроями. Зазвичай такі передачі використовується між принтерами, сканерами, накопичувачами і ін.
- Ізохронна передачі. Використовуються для потокових передач даних в реальному часі. Резервують певну смугу пропускання шини, гарантують певні величини затримок доставки, але не гарантують доставку (в разі виявлення помилки повторної передачі не відбувається. Передачі цього вигляду використовуються для передачі аудіо і відео трафіку.

Режими шини.

Обмін даними може здійснюватися в трьох швидкісних режимах:

- Low Speed. Низькошвидкісний режим. Швидкість передачі складає 1.5 Мбіт/с.
- Full Speed. Повношвидкісний режим. Швидкість передачі 12 Мбіт/с.
- High Speed. Високошвидкісний режим. З'явився лише в специфікації 2.0. Швидкість передачі 480 Мбіт/с.

Інформація по шині передається пакетами. Всього їх визначено 4 види:

- Маркерні пакети.
  - In - інформують USB пристрій, що хост хоче читати дані з пристрою
  - Out - інформує USB пристрій, що хост хоче передавати дані в пристрій

- Setup - використовуються для позначення початку типа передачі даних керуючого типу
- SOF - пакети початку кадру (Start of Frame Packets)

- Пакети даних.
  - Існують два типи пакетів даних - DATA0, DATA1, кожен з яких здатний містити до 1024 байтів даних. У високошвидкісних пристроїв для пакетів даних визначені два інших: DATA2 і MDATA.
- Пакети підтвердження.
  - ACK - підтвердження того, що пакет був успішно прийнятий
  - NAK - інформує, що пристрій в даний момент не може приймати або відправляти дані. А в Interrupt транзакціях повідомляє хосту, що пристрій не має нових даних для передачі.
  - STALL - вказує, що пристрій нездібний передавати або отримувати дані і потрібне втручання хоста.
- Спеціальні.
  - PRE - передує низькошвидкісній передачі даних.

Пристрої на шині USB діляться на ведучі і ведені. Фактично, ведучих пристроїв на шині може бути лише одне, і таким є хост. Всі передачі даних ініціюються хостом у відповідності з певною часовою програмою. Функціональні пристрої самі не можуть ініціювати передачу, а лише відповідають на запити хоста. Обмін даними можливий лише між хостом і пристроєм, і неможливий безпосередньо між пристроями підключеними до шини. Транзакції на USB шині складаються з двох-трьох актів: посилки пакету маркера, що визначає, що слідуватиме далі (тип транзакції, адресу пристрою і його кінцеву точку), пакету даних (опціонально), і пакету статусу транзакції (для підтвердження нормального виконання операції або повідомлення про помилку).

Фізичні канали зв'язку організовуються концентраторами і сполучними дротами. Дріт використовується для підключення USB пристроїв є екранованою витою парою. Для високошвидкісних пристроїв пред'являються високі вимоги до її якості. Низькошвидкісні до цього елементу фізичного інтерфейсу відносяться не критично, і без проблемно можуть функціонувати на неекранованому невитому дроті. Всього в USB кабелі використовується 4 дроти (рис. 1.63)

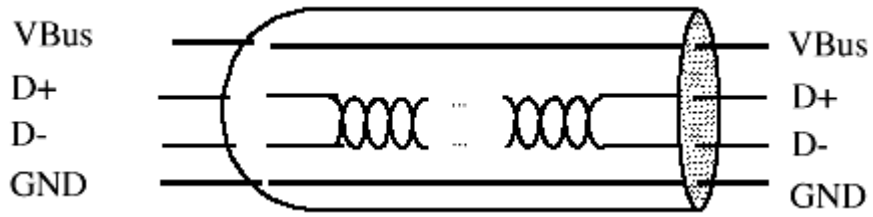


Рис. 1.63

Два для передачі сигналу і два для подачі напруги. Для підключення пристроїв призначені з'єднувачі двох типів: типа "А" і типа "В".

Конектори типу "А" використовуються для підключення до комп'ютера, забезпечують жорстке і надійне кріплення і не призначені для частого підключення/від'єднання.

З'єднувачі ж типа "В", навпаки, потрібні в тих місцях, де існує необхідність частого підключення/від'єднання, і застосовуються вони з боку периферії. Крім того в новій версії USB визначений коннектор miniUSB типа "В".

Він призначений для вживання на малогабаритних пристроях типа мобільних телефонів, фотоапаратів, плеєрів, де немає можливості розмістити стандартний повнорозмірний роз'єм. Від дрібної периферії, типа клавіатур, мишок, де розміщення сполучних роз'ємів незручне та і взагалі безглуздо, кабель може взагалі не від'єднуватися. Конструктивно роз'єми задумані так, що спочатку відбувається з'єднання шини живлення, потім шини даних.

Інтерфейс USB використовує диференціальну передачу. Застосовується кодування по методу NRZI (Non Return to Zero Invert to ones, без повернення

до нуля з інверсією для одиниць) і бітстаффинг для поліпшення властивостей потоку, для самосинхронізації.

Пристрої, що споживають невеликий струм, можуть живитися від шини USB. Максимальний струм, який може забезпечити шина дорівнює 500 мА. Для збільшення доступної потужності живлення на шині, концентратори можуть оснащуватися своїм власним блоком живлення, проте таке рішення не популярне.

Якщо до шини USB підключений новий пристрій, то виявлення пристрою, а також і його швидкісний режим, визначається по стрибку напруги, який має місце бути при включенні на шині даних. Цей стрибок створюється підключенням резистора до напруги 3.3 В. Для низькошвидкісних пристроїв цей резистор підключається до шини D-, для полно- і високошвидкісних - до шини D+. Зазвичай той резистор робиться програмно керованим для того, що б після виявлення пристрою його можна було відключити і збалансувати лінію. Отже, новий пристрій підключений і виявлений.

Конфігурація здійснюється через кінцеву точку з номером 0. Завантажуються необхідні драйвери. Пристрій готовий до роботи.

Обмін даними

Випадок перший: передача від хоста до пристрою. Як тільки така необхідність виникла, хост може ініціювати передачу. Для цього він посилає пристрою пакет out (на знак того, що дані передаватиме він), потім посилає самі дані, а потім приймає пакет ACK, підтверджуючий, що дані пристроєм отримані без помилок (якщо це не ізохронний тип передачі, для якого підтвердження не передається).

Випадок другий: від пристрою до хосту. В пристрою виникла необхідність передати дані. Але воно не може жодним чином дати знати про це хосту. Таких засобів в USB просто не передбачено. Для того, що б виконати таку передачу, хост повинен звернутися в пристрою з питанням, чи

не має воно бажання чого-небудь йому сказати (пославши пакет in). У відповідь на що пристрій вишле йому наявні дані і діждеться здобуття підтвердження (знову ж, якщо ведеться не ізохронна передача). Відповідно, якщо хост не поводитися з таким питанням, то дані ніколи не будуть передані.

Обмін керуючою інформацією. В принципі має ту ж логіку, але використовується передача типа управління і канал повідомлень і спеціальні пакети.

Під час простоїв в енергозбережних цілях пристрою переводяться в стан suspend (і вихід з цього стану, передача інформації про пробудження - єдиний випадок, коли пристрій може стати ініціатором транзакції).

USB 3.0

У вересні 2007 року компанія Intel у ході осіннього форуму IDF 2007 уперше розповіла громадськості про розробку нової версії універсального інтерфейсу USB – USB 3.0, відмінною рисою якого є збільшена пропускна здатність. Третє покоління Universal Serial Bus дозволить комп'ютеру обмінюватися інформацією з периферійними пристроями на швидкості до 4,8 Гбіт/с, тоді як нинішнє покоління інтерфейсу обмежене лише 480 Мбіт/с.

Головною особливістю USB 3.0 є, без сумніву, його пропускна здатність, яка не тільки вище, ніж у попередника, але й значно випереджає головного конкурента – Firewire. І навіть остання версія інтерфейсу Firewire - S3200, представлена в самому кінці 2007 року, дозволяє передавати дані на швидкості до 3,2 Гбіт/с. Якщо ж розглядати перспективи USB 3.0, то позитивно на популярності рішення повинна позначитися зворотна сумісність із USB 2.0 – останній сьогодні дуже затребуваний на ринку периферійних пристроїв, і можливість роботи нового покоління «заліза» зі старим інтерфейсом тільки прискорить перехід на повсюдне використання USB 3.0 (рис. 1.64)

3dnews.ru->novostey.com

Рис.1.64. Логотип USB 3.0

Superspeed швидше High-Speed

USB Implementers Forum фіналізував специфікації стандарту USB 3.0 наприкінці 2008 року. Як і можна було очікувати, новий стандарт збільшив пропускну здатність, хоча приріст не такий значний, як 40-кратне збільшення швидкості при переході від USB 1.1 на USB 2.0. У кожному разі, 10-кратне підвищення пропускної здатності можна вітати. USB 3.0 підтримує максимальну швидкість передачі 5 Гбіт/с. Пропускна здатність майже у два рази перевищує сучасний стандарт Serial ATA (3 Гбіт/із із урахуванням передачі інформації надмірності).

Відомо, що інтерфейс USB 2.0 є основним "вузьким місцем" сучасних комп'ютерів і ноутбуків, оскільки його пікова "чиста" пропускна здатність становить від 30 до 35 Мбайт/с. Але в сучасних 3,5" жорстких дисків для настільних ПК швидкість передачі вже перевищила 100 Мбайт/з (з'являються й 2,5" моделі для ноутбуків, що наближаються до даного рівня). Швидкісні твердотельные накопичувачі успішно перевершили поріг 200 Мбайт/с. А 5 Гбіт/з (або 5120 Мбіт/с) відповідає 640 Мбайт/с.

Сумнівно, що в недалекому майбутньому жорсткі диски наблизяться до рівня 600 Мбайт/, але наступні покоління твердотельных накопичувачів можуть перевищити це число вже через кілька років. Збільшення пропускної здатності стає усе більш важливим, оскільки кількість інформації збільшується, відповідно, росте й час її резервування. Чим швидше працює

сховище, тем менше буде час резервування, тем простіше буде зробити "вікна" у розкладі резервування (табл.1.17).

Таблиця 1.17 - Порівняння швидкостей передачі основних специфікацій USB

	Song / Pic	256 Flash	USB Flash	SD-Movie	USB Flash	HD-Movie
	4 MB	256 MB	1 GB	6 GB	16 GB	25 GB
USB 1.0	5.3 sec	5.7 min	22 min	2.2 hr	5.9 hr	9.3 hr
USB 2.0	0.1 sec	8.5 sec	33 sec	3.3 min	8.9 min	13.9 min
USB 3.0	0.01 sec	0.8 sec	3.3 sec	20 sec	53.3 sec	70 sec

Цифрові відеокамери сьогодні можуть записувати й зберігати гігабайти відеоданих. Частка Hd-Відеокамер збільшується, а їм потрібні більш ємні й швидкі сховища для запису великої кількості даних. Якщо використовувати USB 2.0, то на передачу декількох десятків гігабайт відеоданих на комп'ютер для монтажу буде потрібно значний час. USB Implementers Forum уважає, що пропускна здатність залишиться принципово важливої, і USB 3.0 буде досить для всіх споживчих пристроїв протягом найближчих п'яти років.

USB 3.0 не тільки швидкий, але й двонаправлений

На відміну від другої версії, де дані передаються тільки в одну сторону в ході однієї операції, USB 3.0 підтримує одночасне зчитування й запис.

Досягнута така функціональність завдяки додаванню чотирьох нових ліній до рознімання попереднього інтерфейсу: дві для передачі даних і дві – для приймання. Загальна кількість ліній у такий спосіб збільшене до 9 (в USB 2.0 – 4) (рис. 1.65).

В USB 3.0 використовується новий протокол фізичного рівня (PHY), з метою досягнення максимальної пропускної здатності, що передбачає використання двох окремих каналів – для передачі даних і для передачі службової інформації. Замість механізму опитувань і широкомовних передач, використовуваних у реалізаціях USB 2.0, у новім поколінні інтерфейсу застосовуватися техніка пакетної маршрутизації. Крім того, для кожного пристрою підтримуватися кілька потоків обміну з можливістю приоритезації,

що повинне, зокрема, усунути «ривки» при відтворенні відео з Usb-накопичувача.

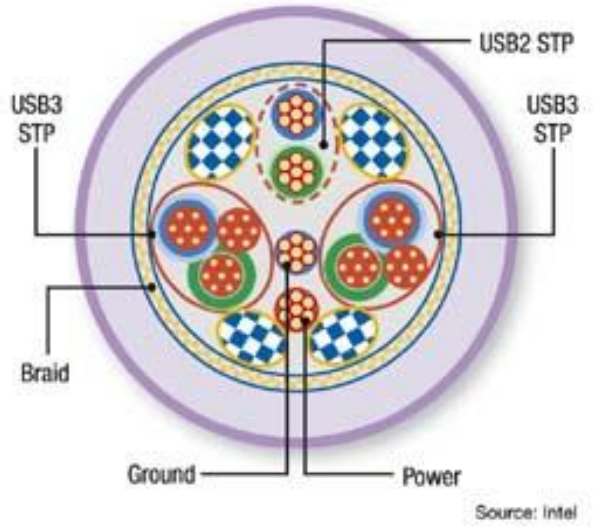


Рис. 1.65. Перетин кабелю USB 3.0.

Кодування 8/10 біт

Щоб гарантувати надійну передачу даних інтерфейс USB 3.0 використовує кодування 8/10 біт, наприклад, по Serial ATA. Один байт (8 біт) передається за допомогою 10-бітного кодування, що поліпшує надійність передачі на шкоду пропускної здатності. Тому перехід з битов на байти здійснюється зі співвідношенням 10:1 замість 8:1. (табл.1.18)

Таблиця 1.18 -. Огляд пропускний способности

Інтерфейс	**Номінальна пропускна здатність (Мбіт/с)**	**Номінальна пропускна здатність (Мбайт/с)**
USB 1.x	1,5	0,19
USB 1.x	12	1.5
USB 2.0	480	60
USB 3.0	5000	500*
Firewire 400	400	50
Firewire 800	800	80*
SATA / esata 150	1500	150*
SATA / esata 300	3000	300*

* - кодування 8/10 біт.

Режими енергозбереження

Основною метою інтерфейсу USB 3.0 є підвищення доступної пропускної здатності, однак новий стандарт ефективно оптимізує енергоспоживання. Інтерфейс USB 2.0 постійно опитує доступність пристроїв, на що витрачається енергія. Напроти, в USB 3.0 є чотири стани підключення, названі U0-U3. Стан підключення U0 відповідає активній передачі даних, а U3 занурює пристрій в "сон".

Якщо підключення не діє, то в стані U1 будуть відключені можливості приймання й передачі даних. Стан U2 іде ще на крок далі, відключаючи внутрішні тактові імпульси. Відповідно, підключені пристрої можуть переходити в стан U1 відразу ж після завершення передачі даних, що, як передбачається, дасть відчутні переваги по енергоспоживанню, якщо порівнювати з USB 2.0.

Більший струм

Крім різних станів енергоспоживання стандарт USB 3.0 відрізняється від USB 2.0 і більш високим підтримуваним струмом. Якщо USB 2.0 передбачав поріг струму 500 мА, то у випадку нового стандарту обмеження було зрушено до планки 900 мА. Струм при ініціації з'єднання був збільшено з рівня 100 мА в USB 2.0 до 150 мА в USB 3.0. Обоє параметра досить важливі для портативних жорстких дисків, які звичайно вимагають ледве більші струми. Раніше проблему вдавалося розв'язати за допомогою додаткової вилки USB, одержуючи живлення від двох портів, але використовуючи тільки один для передачі даних, нехай навіть це порушувало специфікації USB 2.0.

Розділ 2. КОМУНІКАЦІЙНІ ІНТЕРФЕЙСИ ARM ПРОЦЕСОРІВ

ЛЕКЦІЯ 8

Характеристики та архітектура ARM процесорів

Процесори ARM - нинішній безумовний лідер мікропроцесорного ринку мобільних і інтегрованих рішень. На рис. 2.1 зображено узагальнений шлях розвитку мобільних рішень (апаратів стільникового зв'язку, кишенькових обчислювальних пристроїв, тощо). Поряд із самим розвитком технічної бази мобільних рішень проілюстровано розвиток безпосередньо поколінь зв'язку – 2G -> 3G ->LTE.

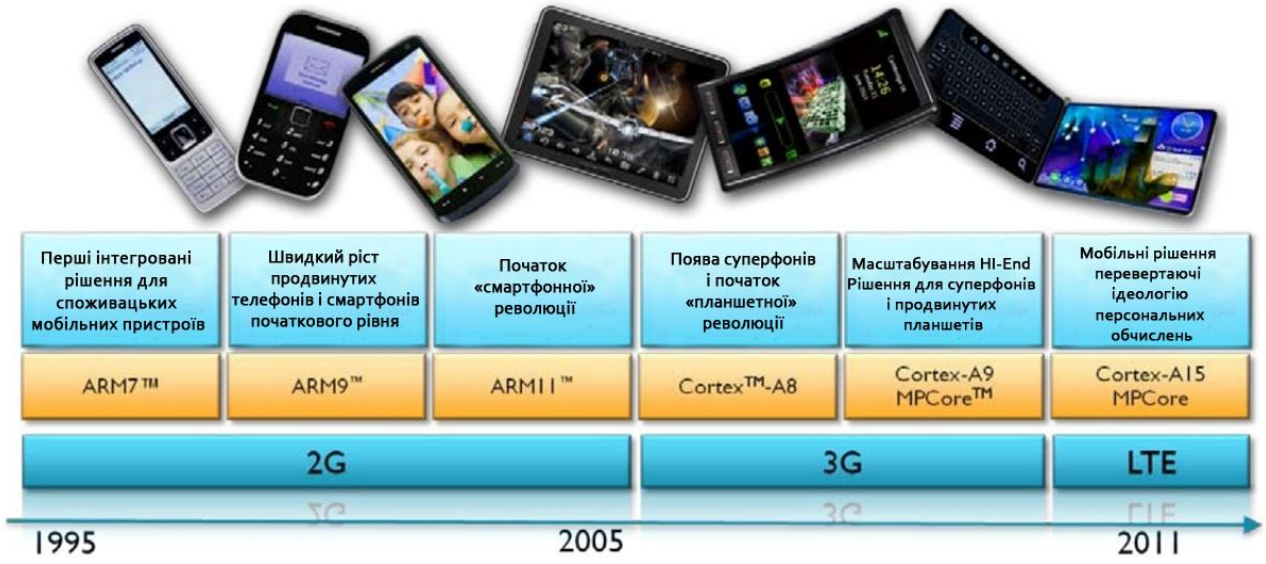


Рис. 2.1. Еволюція мобільних рішень.

Саме необхідність в процесорі підвищеної потужності, здатному працювати з графічним інтерфейсом користувача, призвела на початку 80-х років минулого сторіччя британську компанію AcornComputers до думки про необхідність відмови від готових, але малопотужних рішень, пропонованих партнерами MOS Technology і Motorola, і запуску розробки нового власного процесора. Учасники проекту BBC Micro створили для Acorn мікропроцесор

на архітектури RISC (Restricted (reduced) instructionsetcomputer - комп'ютер з скороченим набором команд), яка є альтернативою популярної архітектури CISC (Complexinstructionsetcomputer). RISC-архітектура пропонувала оптимізацію обчислювального процесу за рахунок реалізації складних функцій не за допомогою єдиної комплексної команди, як це робилося в CISC, а за допомогою набору більш простих команд. При такому підході арифметико-логічний пристрій істотно спрощується, що дозволяє додати в схему процесора більшу кількість регістрів. Збільшення кількості регістрів знижує необхідність частого звернення до повільною оперативної пам'яті.

У фірмі ARM обрали вектор розвитку мікропроцесорів спрямований у бік технологій ASIC і ASSP.

Технологія ASSP (Application-specific standard products) передбачає розробку простих, але в той же час універсальних по застосуванню компонентів - наприклад, апаратних декодерів звуку і відео.

Технологія ASIC (Application-specific integrated circuit) на противагу ASSP передбачає створення інтегральних мікросхем, що спеціалізуються на вирішенні деякого обмеженого кола завдань. До ASIC-рішень можна віднести роутери, мобільні телефони й ігрові консолі.

Завдяки поєднанню технологій ASIC та ASSP можна за дуже короткий термін можна створити новий пристрій довільної конфігурації, адаптований саме для використання під конкретно визначені задачі

Розробивши єдину архітектуру ARMv7 для всієї лінійки Cortex, компанія ARM вирішила розділити варіанти процесорних ядер Cortex по областях їх застосування. В результаті цього маркетингового кроку процесори лінійки Cortex були розділені на три класи.

1. Cortex-A (від application) - сімейство процесорів, орієнтованих на ринок споживчої електроніки і здатних вирішувати широкий спектр завдань, яким сучасні користувачі так люблять навантажувати свої гаджети. До

складу лінійки Cortex-A увійшли процесори: Cortex-A5, Cortex-A7, Cortex-A8, Cortex-A9 і Cortex-A15.

2. Cortex-R (від realtime) - серія мікропроцесорів, оптимізованих для виконання обчислень в режимі реального часу. Скрізь, де потрібно відгук системи в строго певний час, процесори серії Cortex-R забезпечать його. Найбільш бажаними галузями застосування цієї процесорної лінійки є, звичайно ж, вбудовані рішення, такі як контролери промислового та медичного обладнання, автомобільна електроніка та комунікаційні системи.

Втім, і в споживчій електроніці Cortex-R знаходиться гідне застосування. Контролери жорстких дисків, процесори обробки сигналів у фото / відеомодулем смартфонів і цифрових камер, набирає силу «розумне» телебачення - ось тільки мала частина областей, де «реальний» Cortex може застосувати свої здібності.

3. Cortex-M (від eMbedded) - лінійка Cortex-процесорів, які прийшли на зміну 8 - і 16-розрядних мікроконтролерів вбудованих систем. Низьке енергоспоживання, мінімальне тепловиділення і маленькі габарити поряд з незвичайною продуктивністю дозволяють Cortex-M стати на чолі «розумною» побутової техніки, а також інтелектуальних контролерів в областях автомобільної електроніки та ігрових консолей.

Кожна наступна лінійка процесорів ARM підтримує технологічні рішення попередників і включає в себе нові технології. Структура поділу процесорів ARM на класи представлена на рис. 2.2.

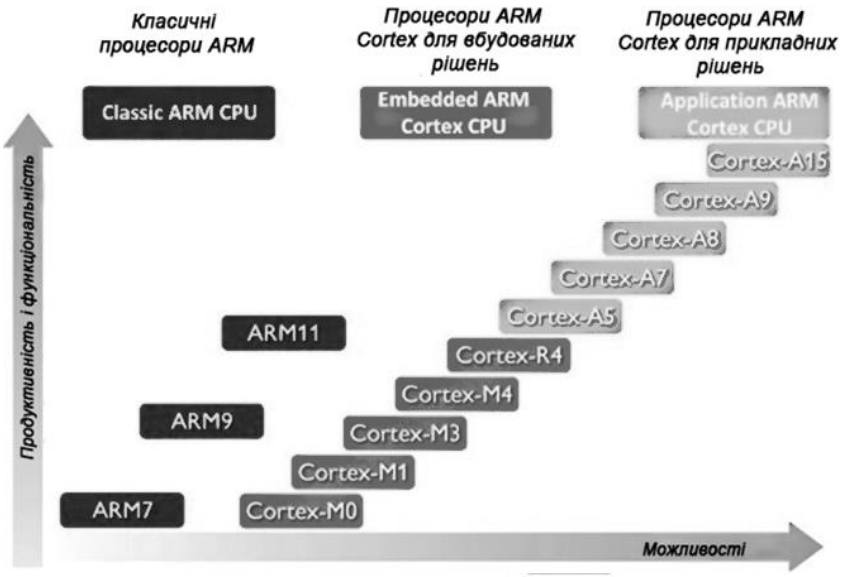


Рис. 2.2. Поділ процесорів ARM на класи.

*Архітектура мікроконтролера STM32F407VG, його характеристики***.**

Нова серія мікроконтролерів STM32F4 є розширенням платформи STM32 заснованим на останній версії ядра ARM Cortex-M4. У цій серії з'явилися нові можливості у сфері обробки сигналів і більш швидкі за часом виконання операції.

Структура мікроконтролера зображена на рис.2.3, нижче приведенні основні характеристики цього сімейства пристроїв:

Блок ядра (ARMCortex-M4 168 MHz):

- Ядро ARM 32-bit Cortex-M4 CPU;
- Частотатактування 168МГц, 210 DMIPS / 1.25 DMIPS / МГц (Dhrystone 2.1);
- Підтримка DSP-інструкцій;

Блок ART прискорювача (ARTAccelerator);

Блок високопродуктивної AHB-матриці шин (Multi-AHB bus matrix);

Рис. 2.3. Структура мікроконтролерів STM32F4xx

Блок пам’яті (Memory):

- До 1 МБайт Flash-пам'яті;
- До 196 кбайт SRAM-пам'яті;
- Контролер SDIO (карти SD, SDIO, MMC, CE-ATA);
- FSMC-контролер (Compact Flash, SRAM, PSRAM, NOR, NAND і LCD 8080/6800);

Системний блок (System):

- Напруга живлення 1,8 ... 3,6 (POR, PDR, PVD і BOR);
- Внутрішні RC-генератори на 16МГц і 32кГц (для RTC);

- Зовнішнє джерело тактування 4 ... 26МГц і для RTC - 32,768кГц;
- Апаратне обчислення CRC, 96-бітний унікальний ID;
- Зовнішній осцилятор 32kHz +16 MHz
- Xtal осцилятори 32kHz
- Регулятор POR/PDR/PVD
- 51/82/114/140 I/Os
- Clock control

Блок роботи з аналоговими сигналами (Analog):

- Три 12-бітних АЦП на 24 вхідних каналу (швидкість до 7,2 мегасемплів, температурний датчик);
- Два 12-бітових ЦАП;

Блок DMA-контролера на 16 потоків з підтримкою пакетної передачі (16-channelDMA);

Блок управління (Control):

- 17 таймерів (16 і 32 розрядні);
- Два сторожових таймера (WDG і IWDG);

Блок комунікації (Connectivity)

- Комунікаційніінтерфейси: I2C, USART (ISO 7816, LIN, IrDA), SPI, I2S;
- CAN (2,0 B Active);
- USB 2.0 FS / HS OTG;
- 10/100 Ethernet MAC (IEEE 1588v2, MII / RMII);
- Інтерфейс цифрової камери (8/10/12/14-бітові режими);
- Апаратний генератор випадкових чисел;

Блоккриптопроцесора (Crypto/hash processor)

- Модульшифрування AES 128, 192, 256, Triple DES, HASH (MD5, SHA-1), HMAC;
- Розширений температурний діапазон -40 ... 105 ° C.

Вбудовані інтерфейси комунікації

Блок комунікації (Рис.2.3 Connectivity) складається із наступних інтерфейсів:

Ethernet. Даний блок присутній не в усіх продуктах сімейства, а лише в контролерах STM32F407 / STM32F417. Блок виконаний за стандартом IEEE802.3. Можлива передача даних зі швидкістю 10/100 Мбіт/с. Доступна синхронізація годин для чого протокол IEEE1588 v2 реалізований апаратно. PHY-трансивер з'єднується безпосередньо з портом MII або RMI.

USB **(Universal Serial Bus)**. Мікропроцесор має два роздільних блоку USB. Перший - USB OTG full-speed, є повністю апаратною реалізацією і сумісний зі стандартами USB 2.0, а також OTG 1.0. Працює на швидкості до 12 Мбіт / с. Підтримується робота в режимі Host / Device / OTG. Присутній SRP (Session request protocol) і HNP (Host negotiation protocol).Другий - USB OTG high-speed працюєврежимі Host / Device / OTG звисокоюшвидкістю480 Мбіт / с, для чого необхідний блок передатчика, що працює на високій швидкості через спеціальний ULPI-інтерфейс.

SDIO **(Secure Digital Input/Output)**. Інтерфейс дозволяє працювати з картами SD / SDIO / MMC-картами пам'ятіі дисковими контролерами CE-ATA. У восьми бітному режимі несуча частот аобміну даними складає 48 MHz. Контролер відповідає такимс тандартам: SD Memory Card 2.0, MultiMediaCard System 4.2 (робота в режимах 1/4/8 біт), SD I / O Card 2.0 (режими 1гоі 4хбіт), CE-ATA 1.1.

SPI **(Serial Peripheral Interface)**. Пристрій включає три блоки SPI, кожен зяких працює в режимі Master (Multimaster) або в режимі Slave, передаючи дані у напів-дуплексному, дуплексному або сімплексному режимі. Підтримується апаратний розрахунок контрольних сум CRC дл япідвищення завадостійкості каналу зв'язку: так CRC може бути переданий останнім байтом слова в режимі Tx, присутня автоперевірка правильності CRC

останньо гобайта. Блок пристрої SPI1 працює нашвидкостях аж до 37,5 Мбіт / с. Інші обмежені максимальною швидкістю в 21 Мбіт / с.

USART **(Universal Synchronous Asynchronous Receiver Transmitter)**. У мікроконтролер вбудовано чотири блоки USART ідва UART. Блоки USART1 і USART6 допускають високошвидкісний обмін даними н ашвидкості до 10,5 Мбіт / с. Інші підтримують швидкість не більше 5,25 Мбіт / с. В будована підтримка передачі даних згідно стандарту NRZ (Non Return to Zero).

Обмін даними здійснюється з використанням 8- або 9-бітних блоків, один або два біти яких виділені як стоп-біти і біти перевірки парності. USART можна конфігурувати на режим SPI, блок USART при цьому виступає в ролі ведучого пристрою SPI. Використовуючи блок USART можна організувати підключення до інтерфейсу LIN, яка знайшла застосування в автомобільній промисловості, або налаштувати на енкодинг / декодинг ІЧ-сигналу IrDA. Можлива робота з модемами по лініях управління RTS і CTS. Підтримується робота зі смарт-картками.

I2C **(Inter-Integrated Circuit)**. На борту МК містить три блоки I2C, що підтримують роботу в режимі Master / Slave (ведучий або ведений), а також у режимі Multimaster (режим в якому на шині присутні кілька Master-пристроїв, які поділяють спільні ресурси Slave, або по черзі змінюють свій стан з Master на Slave і назад). У складі пристрою є модуль діагностики та виправлення пакетних помилок PEC. Використовується 7-бітний і 10-бітний режим адресації. Підтримуються загальноприйняті для протоколу швидкості обміну даними до 100 kHz в простому режимі і 400 kHz в режимі швидкого обміну даними. Модулі можуть бути сконфігурірованни на розширені протоколи SMBus 2.0 і PMBus.

I2S **(Inter-Integrated Sound)**. У мікроконтролері присутні два мультиплексованих блоку I2S з вбудованим SPI. Обидва модулі можуть бути сконфігурованими на роботу в режимі Master або Slave. Дані передаються по 16, 24 або 32 біта дуплексно або сімплексно.

Серед підтримуваних протоколів такі: Phillips I2S, PCM, MSB і LSB з вирівнюванням даних. Інтерфейс I2S був розроблений для обміну звуковими даними в цифровому форматі. Відтепер для тактування присутній окремий PLL, який робить можливим генерацію частот аудіосемплів від 8 до 192 kHz з похибкою не більше 0,01%.

CAN **(Controller Area Network)**. На борту знаходиться два CAN-модуля, що працюють за стандартами 2.0A і 2.0B, швидкість роботи при цьому досягає 1 Мбіт / с. Модулі можуть працювати зі стандартними, а також з розширеними кадрами. Модуль CAN містить три буфера передачі, трьохкаскадний FIFO-стек і 28 банків фільтрів повідомлень (розподілені і масштабуються).

DCMI **(Digital Camera Interface)**. Присутній в контролерах STM32F407 і STM32F417. За допомогою даного інтерфейсу можна організувати пряме підключення до камери або CMOS-матриці. Можлива внутрішня і зовнішня синхронізація, робота в безперервному режимі, автообрізка зайвих частин зображення. Серед підтримуваних форматів 8/10/12/14-бітове прогресивне відео, YCbCr 4: 2: 2 і RGB 565, JPEG.

FSMC (Flexible Static Memory Controller). Блок використовується для підключення рідкокристалічних дисплеїв якої зовнішньої пам'яті безпосередньо. Блок міститься лише в 100-, 144- або 176-пінових корпусах.

Можливе сполучення зпідключеною синхронної або асинхронної пам'яттю або PCMCIA- пристроями. В основному блок призначений для видачі даних МК у відповідному підключеним пристроям вигляді без зайвих витрат процесорного часу на перекодування даних.

Таким чином кожнтй зовнішній пристрій має власний адрес пулу, власні сигнали для управління. Подавши необхідний сигнал вибору мікросхеми можна отримати доступ до того чи іншого зовнішнього пристрою (одночасне використання не припустимо). Можливе підключення таких типів пам'яті як NAND Flash, Compact Flash, NOR Flash, SRAM і PSRAM.

Інтерфейс налаштований для роботи з LCD-контролерами Motorola 6800 і Intel 8080, однак може бути легко використаний для сполучення з контролерами інших виробників.

Робота з аналоговими сигналами

Блок роботи з аналоговими сигналами (Рис.2.3 Analog)містить три АЦП і два одноканальних ЦАП. АЦП має хорошу роздільну здатність 12 біт і високу швидкість перетворення - 2,4 МСемпла в звичайному режимі і 7,2 МСемпла - в потрійному режимі. Максимально доступне число аналогових каналів - 24. Як і в більшості сучасних МК, присутній генератор опорного напруги. Гнучка система налаштувань вбудованого аналогового мультиплексора дозволяє задавати будь послідовності перетворення аналогових каналів (за винятком одночасного перетворення одного каналу на декількох АЦП). Налаштування АЦП дозволяють виробляти одноразові і циклічні вимірювання. Для проведення перетворення на максимальних швидкостях необхідно дотримуватися діапазон напруги живлення 2,4 ... 3,6 В. При зниженні напруги до 1,8 (1,7) В швидкість перетворення знижується до 1,2 мегасемплов. Для контролю внутрішньої температури мікроконтролера вбудований температурний датчик. На його виході формується напруга в залежності від навколишньої температури. Вихід датчика через мультиплексор підключається до АЦП. Використовуючи температурний датчик, можна вимірювати температуру від -40 до 125 ° С з точністю ± 1,5 ° С.

ЦАП має роздільну здатність 12 біт, перетворення можливо в 8/12-бітовому форматі з вирівнюванням цього результату по лівому або правому краях. Так як ЦАП містить два канали, тобто можливість формування стереосигнала. Доступна функція автоматичної генерації шумового сигналу з мінливою амплітудою або трикутного сигналу.

ЛЕКЦІЯ 9

Периферія мікроконтролера STM32F407V. Інтерфейси I2C, SPI, USART, SDIO TA SAI

I2C, SPI, USART, SDIO TA SAI ІНТЕРФЕЙСИ.

Інтерфейс I2C

Inter-IntegratedCircuit (I2C) модуль - послідовний інтерфейс, призначений, підтримки зв'язку з периферійними приладами мікроконтролера.

МК містить три блоки I2C, що підтримують роботу в режимі Master / Slave (ведучий або ведений), а також у режимі Multimaster (режим в якому на шині присутні кілька Master-пристроїв, які поділяють спільні ресурси Slave, або по черзі змінюють свій стан з Master на Slave і назад). У складі пристрою є модуль діагностики та виправлення пакетних помилок PEC. Використовується 7-бітний і 10-бітний режим адресації. Підтримуються загальноприйняті для протоколу швидкості обміну даними до 100 kHz в простому режимі і 400 kHz в режимі швидкого обміну даними. Модулі можуть бути сконфігурірованни на розширені протоколи SMBus 2.0 і PMBus

Для початку слід включити тактування модуля I2C і налаштувати піни в режим альтернативної функції. (Див. [15], в розділ Device Overview) (рис.2.4). З нього видно, що I2C знаходяться на шині APB1.

Наступний крок - включення та налаштування GPIO, обираємо режим альтернативної функції (I2C відноситься до AF4), тип OpenDrain, а підтяжка повинна бути зовнішня. «Швидкість» пінов для 100кГц можна вибрати Low (2 MHz), а для 400 кГц ST рекомендують вибирати Medium або Fast (від 10 MHz). І, нарешті , можна налаштувати I2C.

До включення безпосередньо модуля I2C слід в регістр CR2 записати поточне значення частоти тієї шини, до якої підключений модуль I2C, в даному випадку це частота шини APB1. В рамках даташіта це значення називається PCLK.

Регістр CR2 керує перериваннями від даного модуля. Під цим розуміється то, що чи буде модуль I2C повідомляти в NVIC про те, що щось сталося, або ж просто поставить потрібні прапори в статусному регістрі. Варто зауважити, що в статусному регістрі завжди будуть ставитися подієві прапори, що логічно. В першу чергу цікаві переривання ITEVTEN і ITERREN, переривання подій і помилок відповідно. Можна обійтися цілком і тільки подіями, як найбільш загальним випадком.

```
I2C1->CR2 |= 48;          // Peripheral frequency 24MHz
I2C1->CR2 |= I2C_CR2_ITEVTEN;       // Enable events
```

Регістр CCR відповідає за тактування зовнішньої шини, тому в нього необхідно внести значення, яке розраховується за формулою PCLK / I2C_SPEED. Наприклад, для частоти шини 400 кГц, внутрішня шина APB1 тактується 48 МГц, відповідно в CCR запишемо значення, рівне 48 * 106/4 * 105 = 120. Так само в даному регістрі необхідно вказати режим роботи Slow / Fast, це останній, 16й біт

```
I2C1->CCR &= ~I2C_CCR_CCR;
I2C1->CCR |= 120;
I2C1->CCR |= I2C_CCR_FS;  // FastMode, 400 kHz
```

Регістр TRISE відповідає за фронти сигналів на SDA і SCL, сюди необхідно внести значення з невеликим запасом, яке розраховується так: TRISE = RISE / tPCLK. tPCLK = 1 / PCLK.

Константа RISE - це максимальний час наростання сигналу, по специфікації : 1000 нс для Slow Mode і 300 нс для Fast mode.

tPCLK - це період, що обчислюється по формулі 1 / F.

Оскільки обрано Fast Mode, то значення в TRISE буде наступне: 3.000 * 10-7 / 2.083 * 10-8 = 14.4, з невеликим запасом в більшу сторону, обираємо 15.

```
I2C1->TRISE = 24;
```

Після того, як дані дії будуть виконані, можна включати модуль і переривання в модулі NVIC (якщо потрібні).

```
I2C1->CR1 |= I2C_CR1_PE;          // Enable I2C block
NVIC_EnableIRQ(I2C1_EV_IRQn);
NVIC_SetPriority(I2C1_EV_IRQn, 1);
```

Після настройки модуля, з ним можна працювати або в режимі чекання появи прапора в циклі while (і контролер буде зайнятий тільки цим, що в більшості випадків погано), або - за перериванням. В другому випадку необхідно додати в код обробник переривань подій від модуля I2C. Для модуля I2C1 функція називається I2C1_EV_IRQHandler. Тому в необхідний .c файл додаємо таку функцію:

```
void I2C1_EV_IRQHandler(void) {
}
```

На цьому ініціалізація закінчується.

Далі розглянемо передачу даних slave-пристрою (рис. 2.5)

Figure 243. Transfer sequence diagram for master transmitter

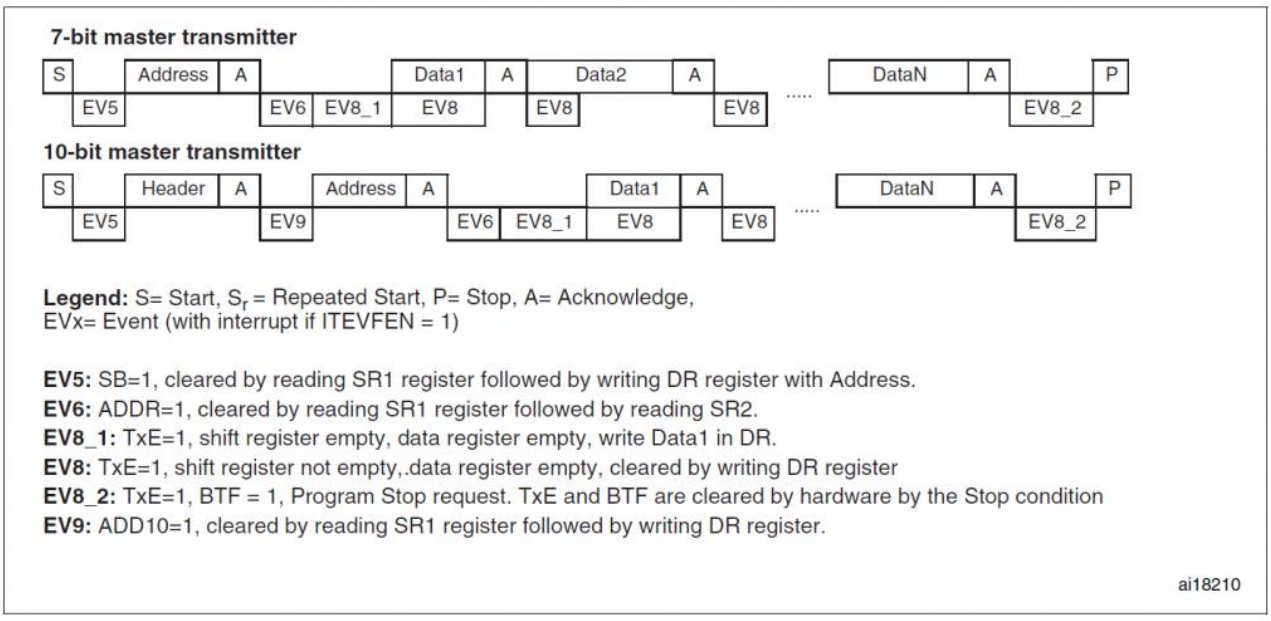


Рис.2.5. Передача даних slave-пристрою

Наприклад, хай необхідно відправити 1 байт даних пристрою і припинити передачу.

Для цього будемо використовувати стани EV5, EV6 та EV8. Десь в коді програми визначена глобальна змінна. Початок передачі ініціює послідовність START:

```
I2C1->CR1 |= I2C_CR1_START;
```

Даний рядок можна поставити по ходу програми там, коли потрібно починати передачу. Наприклад, після того, як ініціалізували змінну data. Далі вже буде код всередині функції-обробника переривань. Для початку необхідно в окремі змінні зберегти значення статусів

```
volatile uint32_t sr1 = I2C1->SR1, sr2 = I2C1->SR2;
```

Після відправки стартової послідовності відбудеться переривання за подією EV5. В статусному регістрі повинен виставитися біт SB. Якщо даний біт виставлений, то необхідно відправити адресу з бітом режиму читання або запису. Для спрощення можна зробити так #define I2C_MODE_READ 1

```
#define I2C_MODE_WRITE   0
#define I2C_ADDRESS(addr, mode) ((addr<<1) | mode)
```

Тепер можна написати обробник стану EV5:

```
 if( sr1 & I2C_SR1_SB ) {
    I2C1->DR = I2C_ADDRESS(0x14,I2C_MODE_READ);
}
```

Коли адресу відправиться і slave-пристрій відповість послідовністю ACK, то станеться подія EV6 і одночасно EV8: встановиться прапор ADDR і TXE. Прапор ADDR означає, що адреса відправлена і прийнята slave-пристроєм, а TXE означає, що буфер вільний для внесення даних для подальшої передачі. Прапор ADDR скинеться сам, як тільки буде прочитано SR1 і SR2, а прапор TXE опрацюємо окремим блоком коду. У обробнику TXE все, що потрібно - це передавати дані. Так як передавати будемо тільки 1 байт, то відразу ж відправимо і послідовність STOP

```
if(sr1 & I2C_SR1_TXE) {
```

```
    I2C1->DR = data;
    I2C1->CR1 |= I2C_CR1_STOP;
}
```

Таким чином, отримали наступну функцію-обробник:

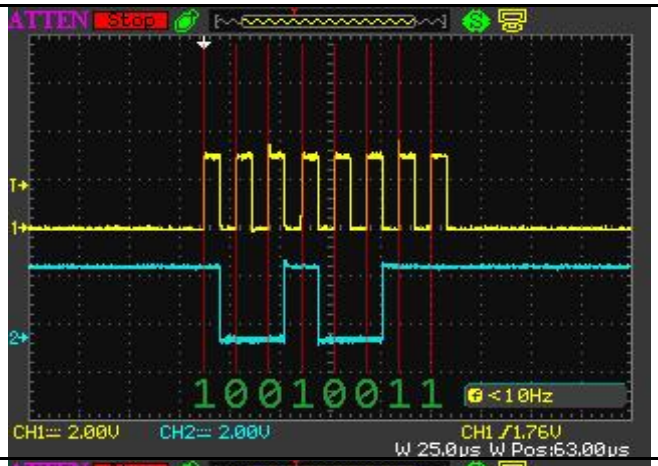
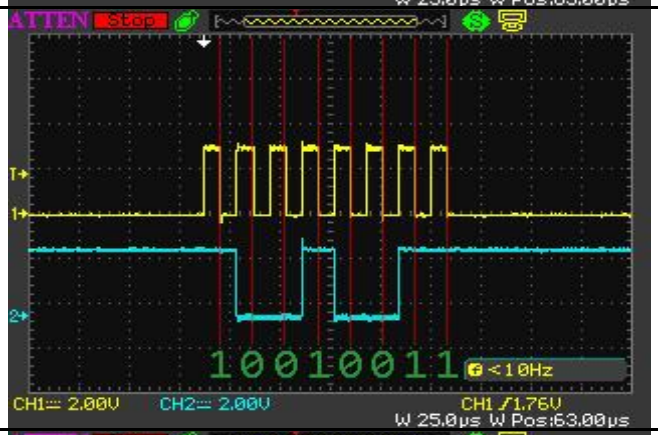
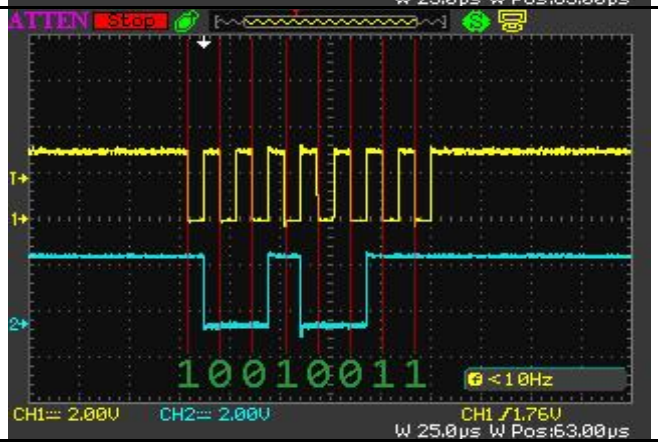
```
#define I2C_MODE_READ     1
#define I2C_MODE_WRITE    0
#define I2C_ADDRESS(addr, mode) ((addr<<1) | mode) uint8_t iter;
uint8_t data[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

void I2C1_EV_IRQHandler(void) {
    volatile uint32_t sr1 = module ->SR1, sr2 = module ->SR2;
    if( sr1 & I2C_SR1_SB ) {
        module ->DR = I2C_ADDRESS(0x14,I2C_MODE_READ);
    }
    if(sr1 & I2C_SR1_TXE) {
        if( iter < 10 ) {
            I2C1->DR = data[iter++];
        } else {
            I2C1->CR1 |= I2C_CR1_STOP;
        }
    }
}
```

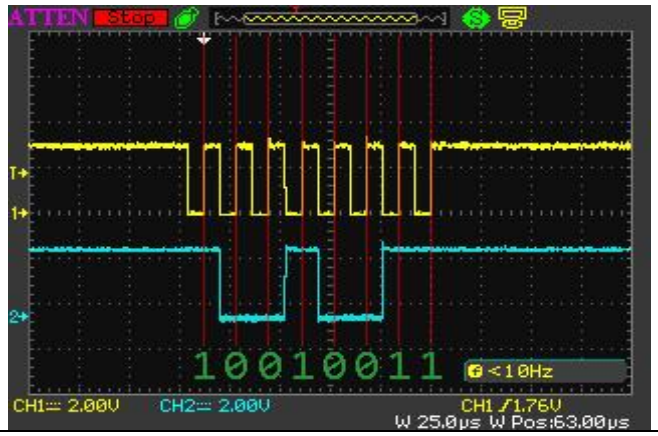
Інтерфейс SPI

Послідовний периферійний інтерфейс (SPI) – послідовний синхронний стандарт передачі даних в режимі повного дуплексу.

В таблиці 2.1 показані режими роботи SPI та їх відмінності один від одного. У всіх 4-х режимах Master посилає один і той же байт (0x93). Жовта лінія це SCK, а синя - MOSI.

Таблиця 2.1- Режими роботи SPI

Режими роботи SPI Режим	CPOL	CPHA	Осцилограма	Опис режиму
0	0	0		Вибірка по передьому наростаючому фронті
1	0	1		Вибірка по задньому спадаючому фронту
2	1	0		Вибірка по передньому спадаючому фронту

3	1	1		Вибірка по задньому наростаючому фронту

Як видно з табл. 2.1, номер режиму складається з двох біт - **CPOL** і **CPHA**. Біт **CPOL** визначає, в якому стані буде перебувати вивід SCL в той час коли нічого не передається. Якщо **CPOL** = 0, то в режимі простою на ніжці низький логічний рівень. Це означає, що передній фронтом буде вважатися перехід з 0 в 1. Якщо **CPOL** = 1, то в режимі простою на ніжці високий логічний рівень. Це означає, що передній фронтом буде вважатися перехід з 1 в 0 (а заднім фронтом, відповідно, навпаки з 0 в 1). Біт **CPHA** визначає по якому фронту потрібно робити вибірку 0 - по передньому фронту, 1 - по задньому фронту.

Наступний важливий параметр - порядок проходження біт. Зазвичай, першим передається старший біт, але іноді буває навпаки, якщо цього не врахувати, то можливі помилки при роботі інтерфейсу. Кількість біт може змінюватися, зазвичай це 8 біт. Далі розглянемо **SPI** на платі **STM32F4DISCOVERY**.

На рисунку 2.6 показані лінії SPI мікроконтроллера**STM32F407Vxxx**:

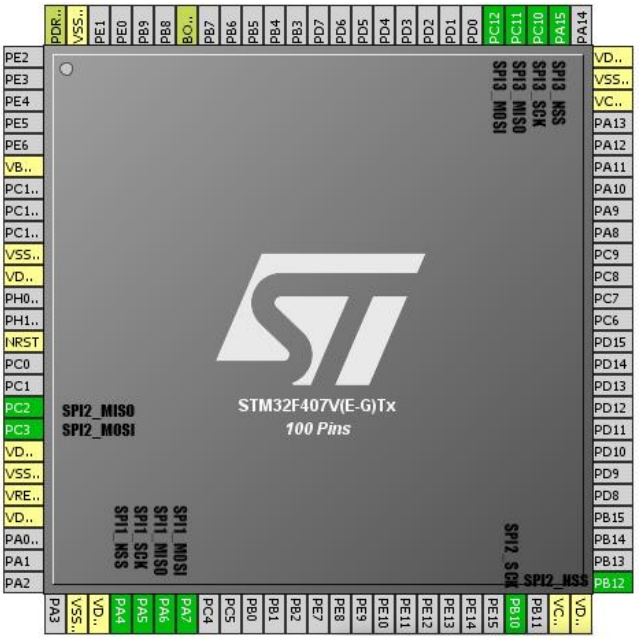


Рис. 2.6. Порти STM32F407Vxxx

Для спрощення роботи зі всією периферією мікроконтролера, (таймери, UART, SPI і т.д.) компанія ST розробила бібліотеку stdperiph_lib. З її використанням код стає більш зрозумілим і простішим для читання, також поліпшується перенесення коду з одного STM32 контролера на інший. Проте для кращого розуміння SPI розглянемо регістри:

Регістр SPI_CR1– регістр керування (рис.2.7)

BIDI MODE	BIDI OE	CRC EN	CRC NEXT	DFF	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Рис. 2.7

BIDIMODE – якщо цей біт встановлений, то прийом та передача даних здійснюються по одному проводу (не враховуючи SCK). При цьому, MOSI вивід Master підключається до MISO Slave.

BIDIOE – цей біт використовується в однопроводному режимі. Якщо він встановлений - SPI модуль тільки передає дані. Якщо скинутий - то приймає.

CRCEN – включає апаратний модуль розрахунку контрольної суми. 0 - викл, 1 - вкл. Змінювати стан цього біта можна тільки коли SPI модуль вимкнений (біт SPE = 0).

CRCNEXT – якщо цей біт встановлений, то після наступної передачі байту даних, буде відправлена контрольна сума.

DFF – якщо біт скинутий, то SPI модуль передає / приймає дані по 8 біт, інакше передається / приймається по 16 біт. Змінювати стан цього біта можна тільки коли SPI модуль вимкнений (біт SPE = 0).

RXONLY – якщо використовується 2-х провідний режим (див біт BIDIMODE) то установка цього біта забороняє передачу, SPI модуль працює тільки на прийом.

Відзначимо наступну особливість виводу NSS. Якщо SPI модуль налаштований в режимі Slave, то він може отримувати сигнал з вивода NSS або ж програмно. Якщо біт SSM скинутий, то сигнал SS буде зчитуватися з вивода NSS, а якщо він встановлений, то стан вивода NSS ігнорується. У такому випадку для керування сигналом SS переходить на біт SSI. Біт встановлений - є сигнал SS, інакше немає. Якщо ж SPI модуль працює в режимі майстра, то вивід NSS потрібно підтягнути до живлення або включити програмне управління (SSM = 1) і встановити біт SSI. В іншому випадку - SPI модуль вирішить, що з'явився новий Master і сам стане Slave.

LSBFIRST – задає порядок передачі біт: 0 - спочатку передається старший біт. 1 - спочатку передається молодший біт.

SPE – вмикає / вимикає SPI модуль

BR2,**BR1**,**BR0** – задають швидкість прийому / передачі (частоту SCK). Частота тактирования модуля SPI ділиться на число, яке задається комбінацією цих трьох біт.

Таблиця 2.3 - Задання швидкості прийому / передачі

BR2	BR1	BR0	Дільник
0	0	0	2
0	0	1	4
0	1	0	8
0	1	1	16
1	0	0	32
1	0	1	64
1	1	0	128
1	1	1	256

Змінювати стан цих біт можна тільки коли SPI модуль вимкнений (біт SPE = 0).

MSTR – якщо біт встановлений - SPI модуль є Master, інакше Slave.

CPOL – полярність сигналу SCK.

CPHA – фаза сигналу SCK.

Регістр SPI_CR2– регістр керування 2 (рис.2.8)

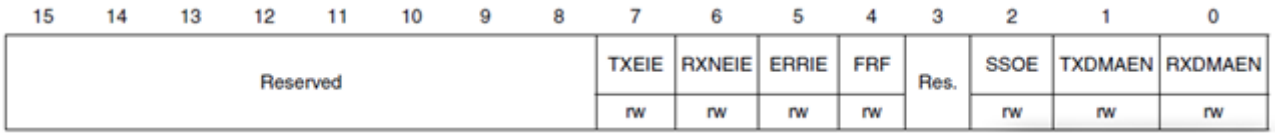


Рис.2.8

TXEIE –дозволяє переривання, коли буфер передачі порожній.

RXNEIE – дозволяє переривання, коли буфер заповнений даними.

ERRIE – дозволяє переривання у разі виникнення помилки. Їх три, щоб розібратися яка виникла, потрібно дивитися стан біт в регістрі статусу.

SSOE – Якщо цей біт виставлений, то SPI модуль сам керує виводом NSS. Тобто перед початком передачі виставляє нуль на цьому виводі, а після завершення - виставляє одиницю.

TXDMAEN – дозволяє / забороняє запит DMA по завершенню передачі

RXDMAEN – дозволяє / забороняє запит DMA по завершенню прийому

Статусний регістр SPI_SR (рис.2.9)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							FRE	BSY	OVR	MODF	CRC ERR	UDR	CHSID E	TXE	RXNE
							r	r	r	r	rc_w0	r	r	r	r

Рис. 2.9

FRE – Frame error flag, він використовується, коли SPI модуль працює в режимі «TI mode» (біт FRF = 1).

BSY – якщо цей біт встановлений, то модуль SPI зараз зайнятий передачею даних.

OVR – біт виставляється в тому випадку, якщо в SPI модуль надійшли нові дані і замінили старі, що не були прочитані.

MODF – виставляється в тому випадку якщо Master раптово перестав бути Master. Таке можливо, коли вивід Master NSS налаштована як вхід і на неї надійшов сигнал низького рівня.

CRCERR – помилка контрольної суми

UDR – флаг не використовується в режимі SPI

TXE – Передача даних закінчилась

RXNE – Прийом даних завершено

Регістр SPI_DR– регістр данних

Представляє собою 16-ти бітний регістр даних, який розбитий на два. Один для передачі, а інший для прийому, але робота з ними здійснюється через один регістр SPI_DR. Якщо щось в нього записати, то запис даних проводиться в регістр для передачі. Якщо прочитати, то дані зчитуються з регістра для прийому даних.

Регістр SPI_CRCPR

У цей регістр записують деяке число, яке повинно вплинути на розрахунок контрольної суми. За замовчуванням записано число 7.

Регістр SPI_TXCRCR

У цей регістр записується контрольна сума, яка була розрахована для переданих даних.

Регістр SPI_RXCRCR

У цей регістр записується контрольна сума, яка була розрахована для прийнятих даних.

Приклад використання SPI на платі STM32F4DISCOVERY, показує як налаштувати SPI1 в режимі Master і відправити дані. У нескінченному циклі відбувається передача одного і того ж байту (0x93).

```
01.#include "stm32f4xx.h"
02.#include "stm32f4xx_gpio.h"
03.#include "stm32f4xx_rcc.h"
04.#include "stm32f4xx_spi.h"
05.
06.int main(void) {
07.GPIO_InitTypeDef GPIO_InitStructure;
08.SPI_InitTypeDef SPI_InitStructure;
09.
10.// Тактуваннямодуля SPI1 іпортуA
11.RCC_APB2PeriphClockCmd(RCC_APB2Periph_SPI1, ENABLE);
12.RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
13.
14.// Налаштовуємо вивід SPI1 для роботи в режимі альтернативної функції
15.GPIO_PinAFConfig(GPIOA, GPIO_PinSource7, GPIO_AF_SPI1);
16.GPIO_PinAFConfig(GPIOA, GPIO_PinSource5, GPIO_AF_SPI1);
17.GPIO_PinAFConfig(GPIOA, GPIO_PinSource6, GPIO_AF_SPI1);
```

```
18.
19.GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
20.GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
21.GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
22.GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
23.GPIO_InitStructure.GPIO_Pin = GPIO_Pin_7 | GPIO_Pin_6 | GPIO_Pin_5;
24.GPIO_Init(GPIOA, &GPIO_InitStructure);
25.
26.//Заповнюємо структуру с параметрами SPI модуля
27.SPI_InitStructure.SPI_Direction= SPI_Direction_2Lines_FullDuplex; //повний дуплекс
28.SPI_InitStructure.SPI_DataSize = SPI_DataSize_8b; // передаємо по 8 біт
29.SPI_InitStructure.SPI_CPOL = SPI_CPOL_Low; // Полярність та
30.SPI_InitStructure.SPI_CPHA = SPI_CPHA_1Edge; // фаза тактовогосигналу
31.SPI_InitStructure.SPI_NSS = SPI_NSS_Soft; // Керувати станом сигналу NSS програмно
32.SPI_InitStructure.SPI_BaudRatePrescaler = SPI_BaudRatePrescaler_32; // Передільник SCK
33.SPI_InitStructure.SPI_FirstBit = SPI_FirstBit_MSB; // Першимвідправляєтьсястаршийбіт
34.SPI_InitStructure.SPI_Mode = SPI_Mode_Master; // Режим - Master
35.SPI_Init(SPI1, &SPI_InitStructure); //Налаштовуємо SPI1
36.SPI_Cmd(SPI1, ENABLE); // Включаємомодуль SPI1....
37.
38.// Оскільки сигнал NSS контролюжться програмно, встановимо його в одиницю
39.// Якщо скинути його в нуль, то SPI модуль вирішить, що
40.// використовується мультимастерна топологія і його перевели в Slave.
41.SPI_NSSInternalSoftwareConfig(SPI1, SPI_NSSInternalSoft_Set);
42.while(1) {
43.SPI_I2S_SendData(SPI1, 0x93); //Передаембайт 0x93 через SPI1
44.while(SPI_I2S_GetFlagStatus(SPI1, SPI_I2S_FLAG_BSY) == SET)
45.}}
```

Інтерфейс UART /USART

UART – вузол обчислювальних пристроїв, призначений для організації зв'язку з іншими цифровими пристроями. Перетворює передані дані в послідовний вид так, щоб було можливо передати їх по цифровій лінії інших аналогічних пристроїв. Метод перетворення добре стандартизований і широко застосовувався в комп'ютерній техніці, оскільки часто виникає

потреба під'єднати пристрій до комп'ютера для обміну даними з ним.Для обміну даними два пристрої повинні бути з'єднані так як показано на рисунку 2.10:

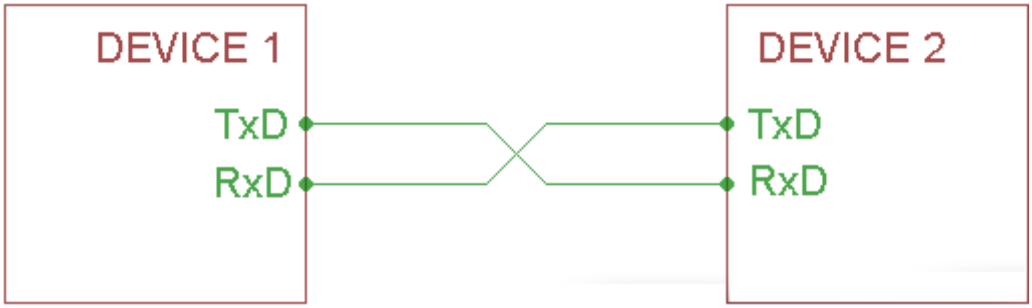


Рис.2.10. Зв’язок двох приладів через UART інтерфейс

Передавальний вивід одного пристрою з'єднується з приймаючим виводом іншого, і навпаки. Обмін даними можливий одночасно в обидві сторони. Коли передача не здійснюється, на виході передавача завжди присутня логічна одиниця. Перед початком передачі даних передавач встановлює на виході логічний нуль. Це називається стартовим бітом, після якого починається передача біт даних, яких зазвичай вісім, але може бути від 5 до 9. Потім слід біт перевірки парності, якщо вона використовується. Цей біт призначений для запобігання обробки некоректних даних після прийому. Потім, після відправлення всіх біт йде стоповий біт, зазвичай один, можливі два. Стоповий біт завжди логічна одиниця, як показано на рис. 2.11:

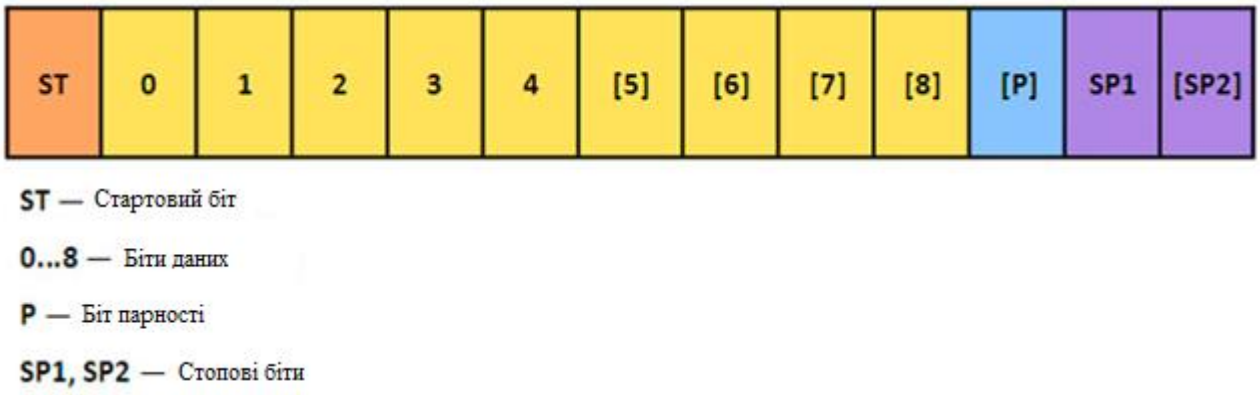


Рис.2.11. Формат передачі

Приклад передачі байту показано на рис. 2.12.

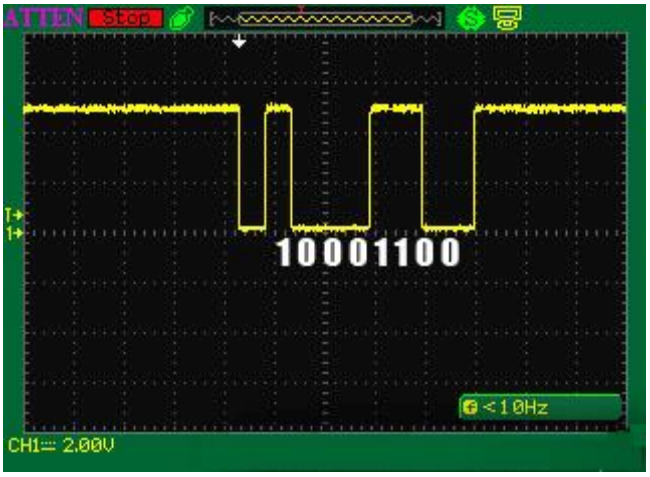


Рис. 2.12 Осцилограма передачі байту даних

Для успішної передачі даних, на обох пристроях, UART повинен бути налаштований з однаковими параметрами. Ще до початку передачі потрібно задати: Швидкість, кількість стопових біт, кількість біт даних, наявність перевірки парності. Швидкість передачі даних довільна, але як правило існує певний набір стандартних швидкостей: 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 56000, 57600 біт / сек і т.д. Чим нижче швидкість передачі даних - тим надійніше така передача.

У комп'ютера інтерфейсу UART як такого немає, але є COM порт. Різниця між COM портом і UART тільки в рівнях напруг. В COM порті логічний нуль - це приблизно +12 вольт, а логічна одиниця приблизно - 12 вольт. Якщо підключити мікроконтролер безпосередньо до виводів COM порту, то він може вийти з ладу. Для нього логічний нуль - це приблизно нуль вольт, а логічна одиниця - це приблизно повна напруга живлення. Для узгодження логічних рівнів використовують спеціальні мікросхеми, наприклад, max3232. Контролери STM32 працюють від 3.3 вольта, і напруга логічної одиниці не повинна перевищувати напруги живлення. Таким чином

застосовуючи max232 потрібно знизити напругу виходу до 3.3 вольт. Або можна використовувати max3232. Схема включення показана на рис. 2.12

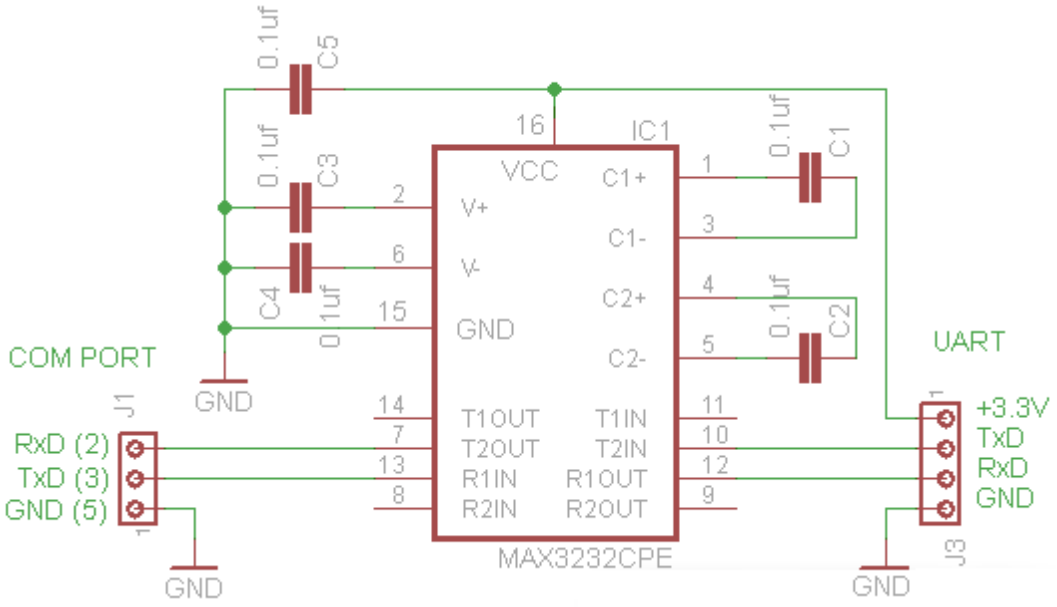


Рис. 2.13. Підключення мікросхеми max3232CPE

Розглянемо програмну реалізацію. Для того щоб відправити щось в UART використовують термінальні програми. Наприклад – стандартний HyperTerminal, або Bray's Terminal (рис. 2.14):

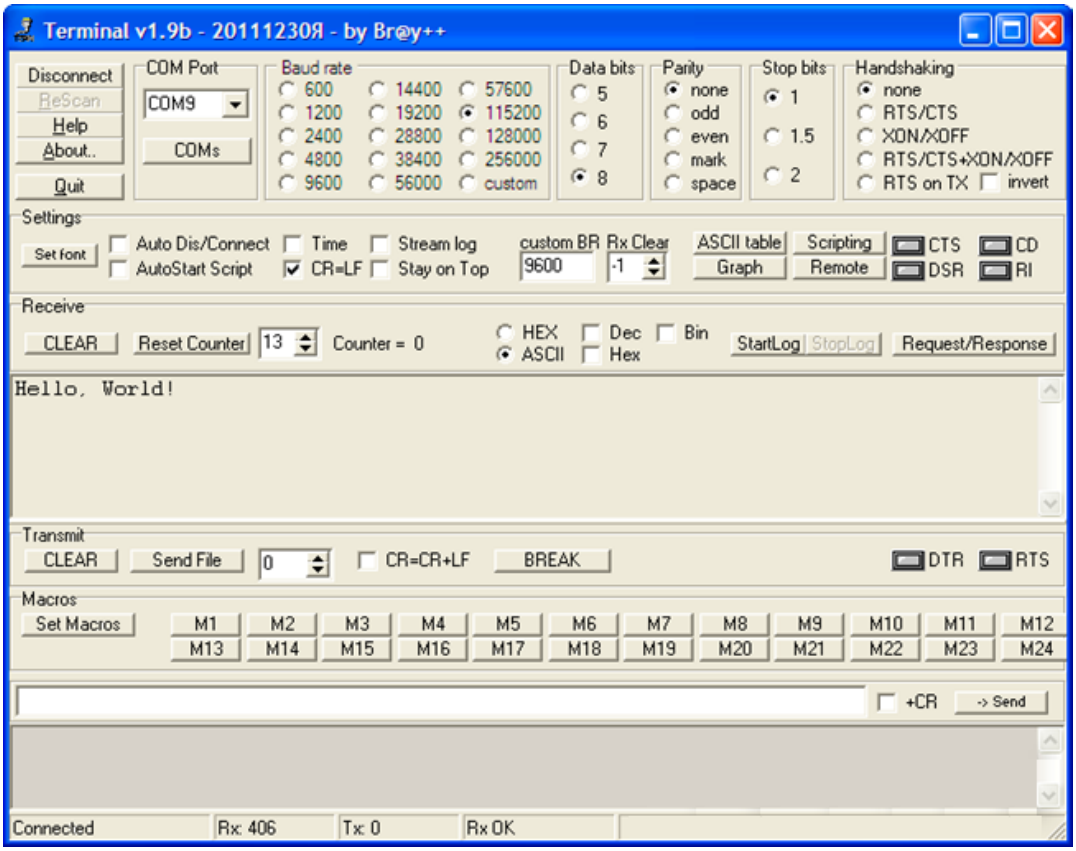


Рис.2.14. Вікно програми Bray`sTerminal

Регістри UART наступні.

USART_BRR– регістр швидкості прийому / передачі (рис. 2.15):

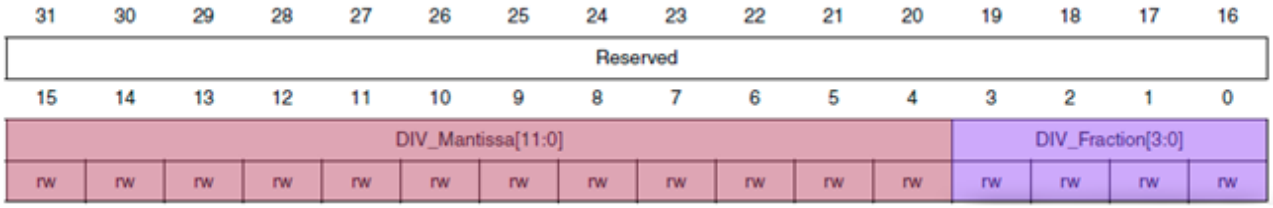


Рис. 2.15

Регістр ділиться на дві частини: Ціла (**DIV_Mantissa**) і дробова (**DIV_Fraction**). Для отримання значення, яке потрібно записати в цей регістр, потрібно скористатись формулою:

USART_BRR = (fck + baudrate /2) / baudrate

де **fck** це частота тактування UART1, а baudrate це бажана швидкість передачі / прийому. В біти **DIV_Mantissa** слід записати ціле число (без округлення) отримане в результаті виконання арифметичної операції:

fck / (16*baudrate)

Дробову частину потрібно округлити до сотих і помножити на 16. Потім ще раз округлити, але вже до цілого. Після цього записати її в біти **DIV_Fraction**. **USART_CR1**– регістр керування (рис. 2.16).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Рис. 2.16

UE - Біт призначений для включення інтерфейсу UART.

M - задає кількість біт даних, яке передаватиметься. Якщо біт дорівнює 0, тоді UART буде відправляти / приймати по 8 біт, якщо одиниці, то 9 біт.

PCE - Якщо біт дорівнює 1, то контроль парності включений, в іншому випадку вимкнений.

PS - Тип контролю парності: 0 - парне, 1 - непарне. **TE** - Якщо біт дорівнює 1, то дозволена робота передавача (вивід TxD)

RE - Якщо біт дорівнює 1, то дозволена робота приймача (вивід RxD)

Як видно по двох останнім бітам в цій таблиці - передавач і приймач один від одного не залежать. Якщо не потрібно приймати (або передавати) дані, то можна заощадити один вивід контролера (TxD або RxD) не включаючи не потрібну частину UART'а.

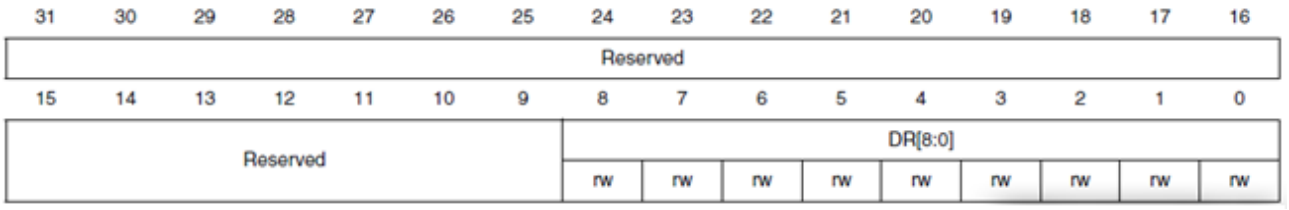
USART_CR2 - регістр налаштувань

В цьому регістрі лише два біти які відносяться до UART: **STOP1**, **STOP0**

За допомогою цих двох біт можна задати кількість стопових біт в передачі.

STOP1	STOP0	Кількість стопових біт
0	0	1 стоп біт
0	1	0.5 стоп біта
1	0	2 стоп біта
1	1	1.5 стоп біта

Регістр даних USART_DR: служить для прийому / передачі (рис.2.17)



У ньому використовуються молодші 8 або 9 біт (залежно від біта **M** в регістрі **USART_CR1**). Щоб відправити у UART дані записуємо їх в цей регістр. Щоб прочитати дані читаємо цей регістр. При цьому, щоб не виникало ситуацій, зчитування при передачі, або запису при прийомі використовується **регістр USART_SR**– статусний регістр (рис.2.18):

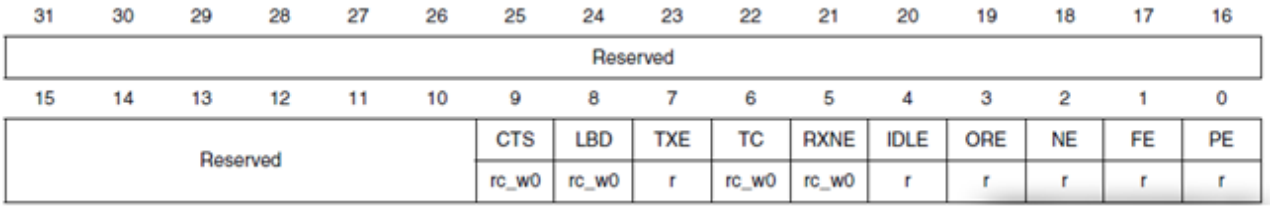


Рис. 2.18

RXNE - Цей біт встановлюється, коли в UART щось прийшло. Якщо не використати з **USART_DR** дані, то вони перезапишуться новими.

TC - Якщо цей біт встановлений в одиницю, то це означає що передача даних завершена і можна записувати в регістр (**USART_DR**).

Приклад

Передача на ПК рядка з вітанням“Hello:)”. На вивід PA9 в даному прикладі був підключений вхід (RxD) USB-UART перетворювача.

Налаштування UART: швидкість 9600 біт / сек, 1 стоп біт, без перевірки парності.

```
01.#include "stm32f10x.h"
02.#include "stm32f10x_gpio.h"
03.#include "stm32f10x_rcc.h"
04.
05.//Функція призначена для формування невеликої затримки
06.void Delay(void) {
07.volatile uint32_t i;
08.for (i=0; i != 0x70000; i++);
09.}
10.
11.//Функціящовідправляєбайтв UART
12.void send_to_uart(uint8_t data) {
13.while(!(USART1->SR & USART_SR_TC)); //Чекаємопокибіт TC врегістрі SR станет 1
14.USART1->DR=data; //Відправляємобайтчерез UART
15.}
16.
17.int main(void) {
18.GPIO_InitTypeDef PORTA_init_struct;
19.// Включаємо тактування порта A та USART1
20.RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_USART1, ENABLE);
21.// Налаштовуємо вивід TxD (PA9) як вихід push-pull з альтернативною функцією
22.PORTA_init_struct.GPIO_Pin = GPIO_Pin_9;
23.PORTA_init_struct.GPIO_Speed = GPIO_Speed_50MHz;
24.PORTA_init_struct.GPIO_Mode = GPIO_Mode_AF_PP;
25.GPIO_Init(GPIOA, &PORTA_init_struct);
26.//Налаштовуємо UART
27.USART1->BRR=0x9c4; //BaudRate 9600
28.USART1->CR1 |= USART_CR1_UE; //Дозволяємороботу USART1
29.USART1->CR1 |= USART_CR1_TE; //Включаємопередавач
30.while(1) {
31.//Відправляємочерез UART слово Hello
32.send_to_uart('H');
33.send_to_uart('e');
34.send_to_uart('l');
35.send_to_uart('l');
36.send_to_uart('o');
37.send_to_uart(' ');
```

```
38.send_to_uart(':');
39.send_to_uart(')');
40.send_to_uart('\n');
41.Delay(); //невеликазатримка
42.}}
```

Створюємо порожній проект, копіюємо код, компілюємо і прошиваємо. Вікно терміналу, після виконання програми прийме вигляд як на рисунку 2.19:

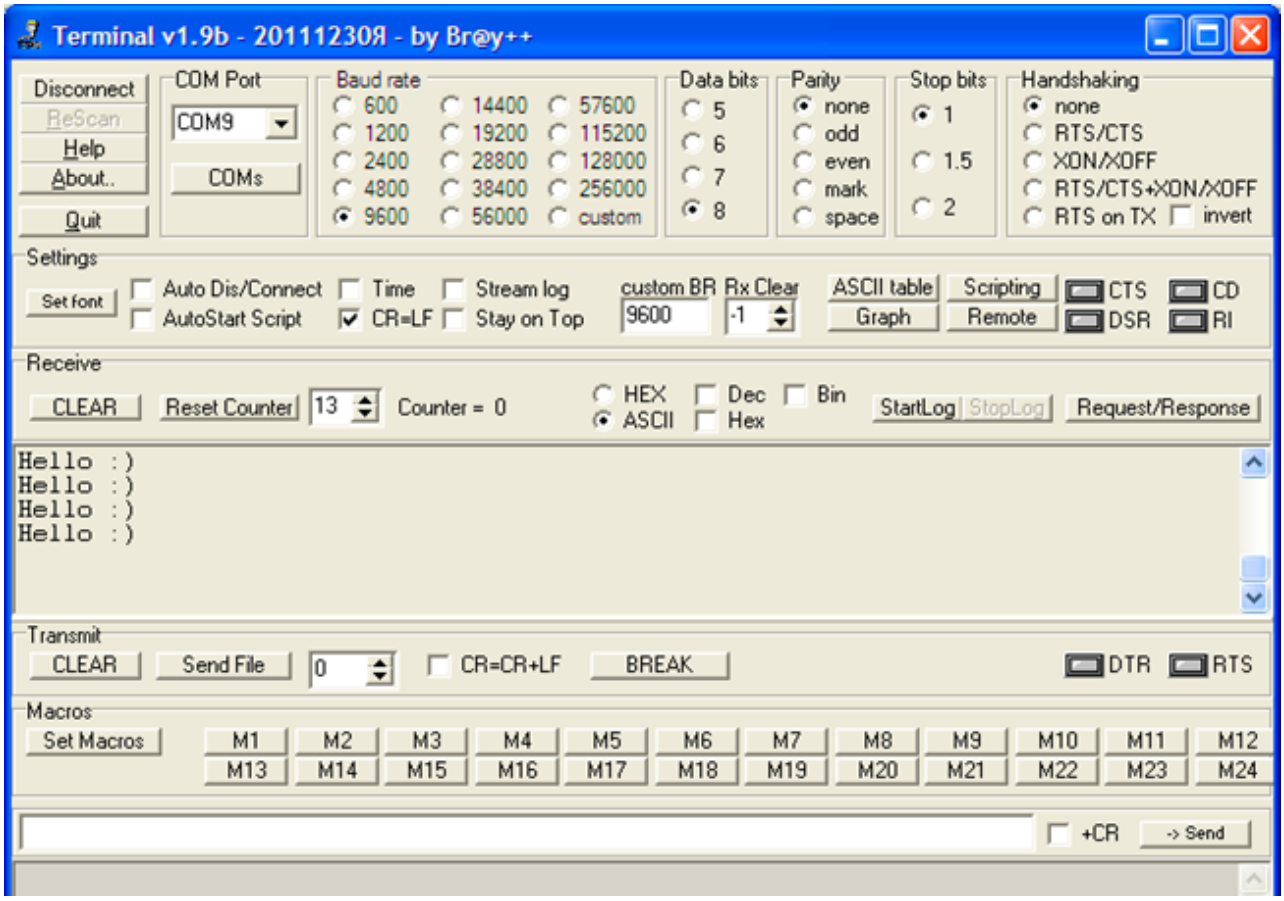


Рис.2.19. Прийом інформації з мікросхеми у вікні Bray`sTerminal

Інтерфейс SDIO

SDIO - це інтерфейс для передачі даних в / з карт пам'яті. Розглянемо роботу з microSD флеш картою пам'яті по SDIO, який є в контролері **stm32f407vgt6** плати **stm32f4discovery**.

Працювати з картами флеш пам'яті можна за допомогою SPI інтерфейсу, але сучасні 32 бітні контролери, які мають модуль, спеціально

призначений для роботи з картами пам'яті - SDIO, є значано дешевшими. Це також істотно спрощує і прискорює роботу. На рис. 2.14 представлені контакти і порівняння розмірів всіх SD карт:

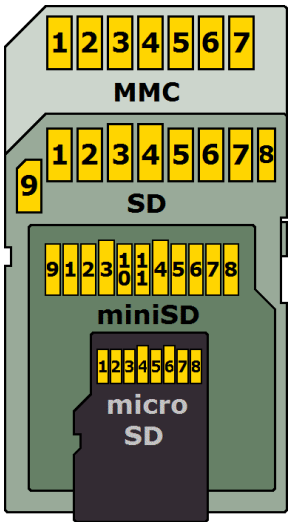


Рис.2.20. Контакти карти пам’яті

Призначення виводів наведено в табл. 2.4

Таблиця *2.4* Призначення виводів карт пам’яті

MMC Pin	SD Pin	miniSD Pin	microSD Pin	Имя	I/O	Logic	Описание
1	1	1	2	NC	.	.	Не используется
2	2	2	3	CMD	I/O	PP, OD	Command, Response
3	3	3		VSS	S	S	Ground
4	4	4	4	VDD	S	S	Power
5	5	5	5	CLK	I	PP	Serial Clock
6	6	6	6	VSS	S	S	Ground
7	7	7	7	DAT0	I/O	PP	SD Serial Data 0
	8	8	8	NC nIRQ	. O	. OD	Не используется (memory cards) Interrupt (SDIO cards) (Negative Logic)
	9	9	1	NC	.	.	Не используется
		10		NC	.	.	Зарезервировано
		11		NC	.	.	Зарезервировано

З картою можна працювати по SDIO в режимі однобітної та 4-х бітної шини даних. Далі мова йтиме про однобітну. Жодної принципової різниці, крім кількості використовуваних провідників, це не має. Як видно контактів

на флешці досить багато. З них, крім живлення та землі, в основному потрібні три:

- CLK - тактування карти.
- CMD - по цій лінії передаються команди.
- DAT0 - линія даних (у випадку 4х бітної їх буде 4).

З’єднуємо с контролером:

- PC8 --- SDIO_D0 (DAT0)
- PC12 --- SDIO_CK (CLK)
- PD2 --- SDIO_CMD (CMD)

Передача даних з / на карту пов'язана з обчисленням контрольних сум. Це відбувається при кожній передачі. Розглянемо функцію ініціалізації модуля SDIO і ліній GPIO:

```
voidSD_LowLevel_Init(void) {
  uint32_t tempreg;

  GPIO_InitTypeDef GPIO_InitStructure;

  // Дозволяємо GPIOC та GPIOD Periph clock
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC | RCC_AHB1Periph_GPIOD

  //Ініціалізуємопіни
  GPIO_PinAFConfig(GPIOC, GPIO_PinSource8, GPIO_AF_SDIO);
  GPIO_PinAFConfig(GPIOC, GPIO_PinSource9, GPIO_AF_SDIO);
  GPIO_PinAFConfig(GPIOC, GPIO_PinSource10, GPIO_AF_SDIO);
  GPIO_PinAFConfig(GPIOC, GPIO_PinSource11, GPIO_AF_SDIO);
  GPIO_PinAFConfig(GPIOC, GPIO_PinSource12, GPIO_AF_SDIO);
  GPIO_PinAFConfig(GPIOD, GPIO_PinSource2, GPIO_AF_SDIO);

  // Налаштовуємо PC.08, PC.09, PC.10, PC.11  піни: D0, D1, D2, D3
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8 ; // | GPIO_Pin_9 | GPIO_Pin_10 | GPIO_
 //розкоментуватидля 4хбітноїшини
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_25MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
```

```
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
  GPIO_Init(GPIOC, &GPIO_InitStructure);

  // Налаштовуємо PD.02 CMD line
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
  GPIO_Init(GPIOD, &GPIO_InitStructure);

  // Configure PC.12 pin: CLK pin
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOC, &GPIO_InitStructure);

  //Enable the SDIO APB2 Clock
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_SDIO, ENABLE);

  // Enable the DMA2 Clock
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_DMA2, ENABLE);

  //Ініціалізуємо SDIO зпочатковимтактуванням
  HW_Flow Disabled, Rising Clock Edge, Disable CLK ByPass, Bus Width=0, Power sav
  SDIO->CLKCR=tempreg;

  // Включаємо SDIO
  SDIO->>POWER = 0x03;
```

Функція передачі команди:

```
staticuint32_t SD_Command(uint32_t cmd, uint32_t resp, uint32_t arg) {
  //Якаможебутивідповідь:
  //0,2: Безвідповіді (cmdsent) ->NORESP
  //1:Короткавідповідь (cmdrend and ccrcfail) ->SHRESP
  //3:Довгавідповідь (cmdrend and ccrcfail) ->LNRESP

  //Очищуємофлаги
  SDIO->ICR=(SDIO_STA_CCRCFAIL | SDIO_STA_CTIMEOUT
 | SDIO_STA_CMDREND | SDIO_STA_CMDSENT);

  SDIO->ARG=arg;
SDIO->CMD=(uint32_t)(cmd & SDIO_CMD_CMDINDEX)
| (resp & SDIO_CMD_WAITRESP) | (0x0400);

  //Блокувати поки не отримаємо відповідь
```

```
  if(resp==NORESP) {
    //Чекаємона CMDSENT
    while(!(SDIO->STA & (SDIO_STA_CTIMEOUT | SDIO_STA_CMDSENT))) {};
  }
  else{//SHRESP or LNRESP or R3RESP
    //Чекаємо на CMDREND чи CCRCFAIL
    while(!(SDIO->STA & (SDIO_STA_CTIMEOUT | SDIO_STA_CMDREND | SDIO_
  }

  // Перевіряємо чи є відповідь правильною
  // Всі R3 відповіді без перевищення затримки вважаються правильними
  if(SDIO->STA & SDIO_STA_CTIMEOUT) {
    SD_Panic(cmd, "SDIO: Command Timeout Error\n");
  } elseif((SDIO->STA & SDIO_FLAG_CCRCFAIL) && (resp!=R3RESP)) {
    SD_Panic(cmd, "SDIO: Command CRC Error\n");
  }

  returnSDIO->STA;
}
```

Інтерфейсу SAI

SAI - (Serial audio interface) інтерфейс передачі звукових сигналів. Складається з двох звукових підблоків, незалежних один від одного. Підтримує I2S, PCM / DSP і AC`97 протоколи. Може працювати в режимах Master / Slave. Звукові підблоки можуть як приймати, так і передавати, синхронно чи ні. Може використовуватися з DMA контролером, який може отримувати доступ до системної шини незалежно від центрального процесора.

У табл. 2.5 наведені основні характеристики послідовного звукового інтерфейсу в **STM32F405xx**:

Таблиця 2.5- Характеристики послідовного звукового інтерфейсу

Symbol	Parameter	Conditions	Min	Max	Unit
f_{MCKL}	SAI Main clock output	-	256 x 8K	256xFs(2)	MHz
F_{SCK}	SAI clock frequency	Master data: 32 bits	-	64xFs	MHz
		Slave data: 32 bits	-	64xFs	
D_{SCK}	SAI clock frequency duty cycle	Slave receiver	30	70	%
$t_{v(FS)}$	FS valid time	Master mode	8	22	ns
$t_{su(FS)}$	FS setup time	Slave mode	2	-	
$t_{h(FS)}$	FS hold time	Master mode	8	-	
		Slave mode	0	-	
$t_{su(SD\_MR)}$	Data input setup time	Master receiver	5	-	
$t_{su(SD\_SR)}$		Slave receiver	3	-	
$t_{h(SD\_MR)}$	Data input hold time	Master receiver	0	-	
$t_{h(SD\_SR)}$		Slave receiver	0	-	
$t_{v(SD\_ST)}$ $t_{h(SD\_ST)}$	Data output valid time	Slave transmitter (after enable edge)	-	22	
$t_{v(SD\_MT)}$		Master transmitter (after enable edge)	-	20	
$t_{h(SD\_MT)}$	Data output hold time	Master transmitter (after enable edge)	8	-	

На рис.2.21 та рис.2.22 представлені часові діаграмми роботи SAI Master і Slave відповідно.

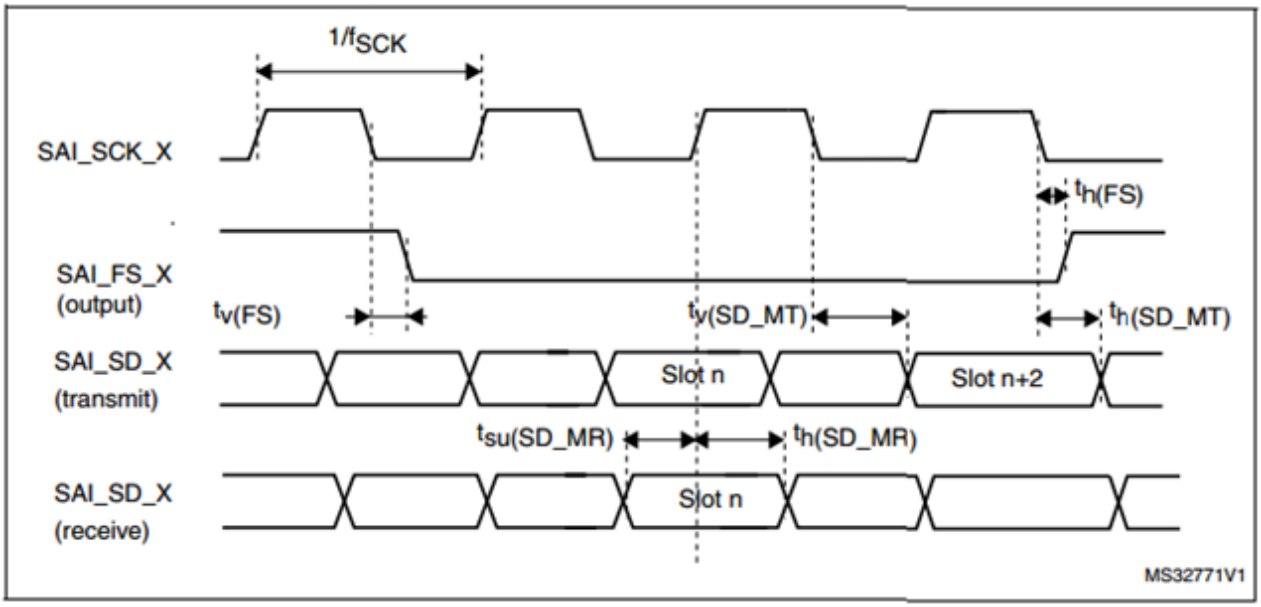


Рис.2.21 Часові діаграми SAI Master

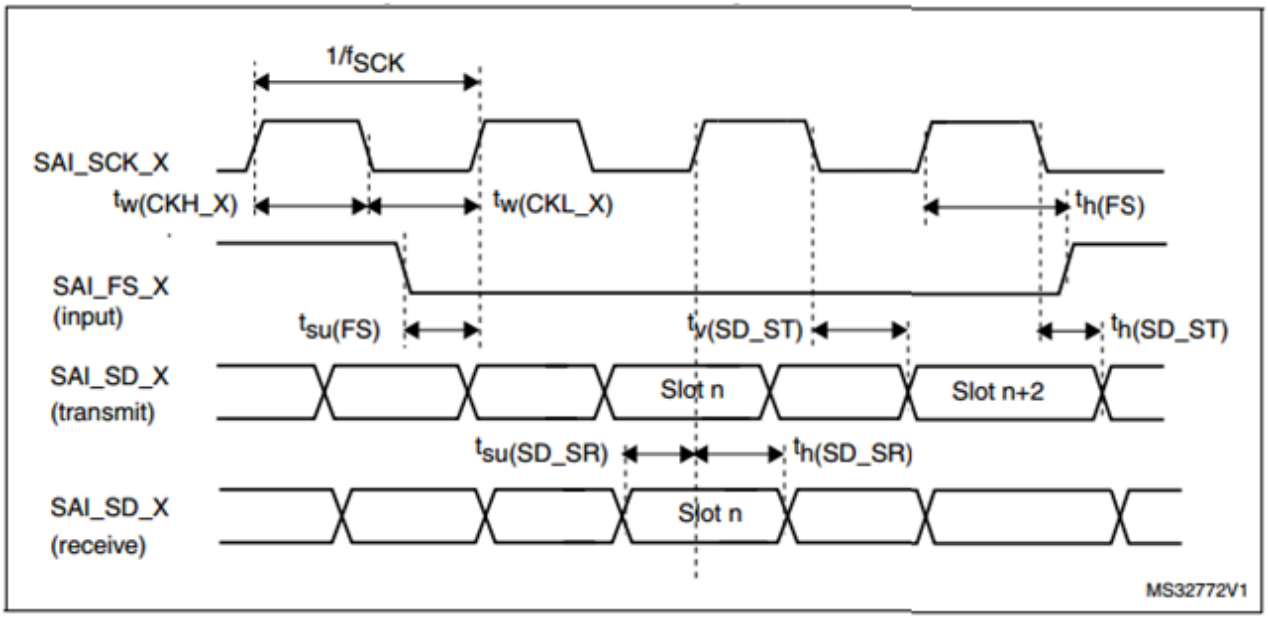


Рис.2.22. Часові діаграми SAI Slave

КОНТРОЛЬНІ ПИТАННЯ

1. Назвіть задачі, які вирішують інтелектуальні вузли розподіленої мікроконтролерної мережі. Вкажіть перевагими розподіленої системи перед централізованою системою
2. Назвіть інтерфейси. що використовуються в розподілених мікроконтролерних мережах?
3. Перерахуйте методи арбітражу в розподілених мікроконтролерних мережах.
4. Коротко опишіть режими роботи послідовного інтерфейсу мікроконтролера МК-51.
5. Опишіть регістри інтерфейсу SPI.
6. Коротко опишіть роботу інтерфейсу SPI.
7. Опишіть архітектуру 4х–проводної SPI шини з одним ведучим і декількома веденими. Як її можна вдосконалити?
8. Опишіть архітектуру 4х–проводної SPI шини з двома ведучими. Як її можна вдосконалити?
9. Опишіть двопроцесорну архітектуру SPI шини з ресурсом, що розділяється.
10. Опишіть двопроцесорну архітектуру SPI шини з розділеним ресурсом і лініями запиту шини.
11. Область застосування шини I2C? Вкажіть її переваги і недоліки.
12. Поясніть, яким чином здійснюється підключення I2C-приладів до шини?
13. Поясніть пересилку біта даних по шині I2C
14. Опишіть сигнали START и STOP шини I2C.
15. Опишіть формат байта шини I2C.
16. Поясніть, як здійснюється підтвердження при передачі даних шини I2C.
17. Поясніть, як здійснюється синхронізація по шині I2C.

18. Поясніть, як здійснюється арбітраж по шині I2C?
19. Поясніть, як використовується механізм синхронізації при процедурі управління по шині I2C?
20. Поясніть, як здійснюється 7-бітова адресація по шині I2C?
21. Опишіть загального виклику для шини I2C.
22. Опишіть призначення байту старту для шини I2C.
23. Опишіть основні характеристики двопровідного послідовного інтерфейсу TWI.
24. Опишіть блок генератора швидкості зв‘язку модуля TWI.
25. Опишіть блок шинного інтерфейсу модуля TWI.
26. Опишіть блок виявлення адреси модуля TWI.
27. Опишіть блок керування. модуля TWI
28. Коротко опишіть регістри TWI.
29. Яка послідовність обслуговування TWI при типовій передачі?
30. Дайте визначення CAN інтерфейсу.
31. Опишіть технічні характеристики CAN інтерфейсу.
32. Принцип роботи CAN інтерфейсу.
33. Ідентифікатори CAN інтерфейсу.
34. Фізична шина CAN інтерфейсу.
35. Поясніть, як досягається висока надійність CAN інтерфейсу?
36. Поясніть, як здійснюється арбітраж CAN шини?
37. Поясніть, як здійснюється виявлення помилок CAN шини?.
38. Опишіть формат CAN повідомлення.
39. Коротко опишіть однопровідний інтерфейс 1-Wire.
40. Для чого використовується система автоматизації на базі мереж MicroLAN?
41. Поясніть, як здійснюється обмін інформацією по шині 1-Wire?
42. Назвіть класи, на які були поділені процесори лінійки Cortex по областях їх застосування.

43. На основі якої архітектури побудовані мікроконтролери серії STM32?
44. Назвіть основні переваги мікроконтролера STM32F407.
45. Наведіть структуру мікроконтролерів серії STM32F4xx.
46. Назвіть, з яких інтерфейсів складається блок комунікації?
47. За допомогою якого інтерфейсу можна організувати пряме підключення до камери або CMOS-матриці мікроконтролера STM32F407?
48. Назвіть основні компоненти блоку роботи з аналоговими сигналами (Analog) мікроконтролера STM32F407та їх призначення.
49. Дайте визначення Inter-IntegratedCircuit (I2C) модуля мікроконтролера STM32F407
50. Назвіть режими передачі даних в інтерфейсі I2C мікроконтролера STM32F407
51. Як формується умова START (S) мікроконтролера STM32F407?
52. Як формується умова STOP(P) мікроконтролера STM32F407?
53. Поясніть роботу інтерфейсу передачі даних SPI мікроконтролера STM32F407?
54. Який існує стандартний набір швидкостей передачі даних інтерфейсом UART мікроконтролера STM32F407?
55. Дайте визначення інтерфейсу SDIO мікроконтролера STM32F407
56. Дайте визначення інтерфейсу SAI (Serial audio interface) мікроконтролера STM32F407

НАВЧАЛЬНО-МЕТОДИЧНІ МАТЕРІАЛИ

1. Дистанційний курс Сучасні напрямки комп'ютерної та мікропроцесорної техніки Сертифікат УЦДО від 25.04.2013; № НМП №3670 Режим доступу до ресурсу: http://moodle.ipo.kpi.ua/moodle/course/view.php?id=516

2. Дистанційний курс Спеціалізовані та промислові мікропроцесорні системи; Сертифікат УЦДО від 15.05.2012; № НМП №2536 Режим доступу до ресурсу: http://moodle.udec.ntu-kpi.kiev.ua/moodle/course/view.php?id=309

3. Конспект лекцій з дисципліни «Сучасні напрямки комп'ютерної та мікропроцесорної техніки. Розділ 3. Архітектура сучасних мікроконтролерів» для спеціальності 6.050802 – «Електронні пристрої та системи» (171 Електроніка)/ Укладачі: Терещенко Т.О., Ямненко Ю.С., Хохлов Ю.В.: НТУУ «КПІ ім. І. Сікорського», 2015. -230 с. Гриф «Рекомендовано» надано Вченою радою факультету електроніки НТУУ «КПІ»

4. Терещенко Т. О., Тодоренко В.А., Батрак Л.М.,Ямненко Ю. С. Мікропроцесорні пристрої. Навчальний посібник для аспірантів спеціальності «Електроніка». - К.: НТУУ «КПІ ім. Ігоря Сікорського», 2017. - 244с. Гриф надано Вченою радою КПІ ім.Ігоря Сікорського, протокол №6 від 12.06.2017 р.

5. Жуйков В.Я., Терещенко Т.О., Петергеря Ю.С. Електронний підручник «Мікропроцесори і мікроконтролери» - 2009 Гриф надано Міністерством освіти і науки України (лист № 1.4_18-Г-114 від 10.01.2009 р. - режим доступу до ресурсу:. http://www.kaf-pe.ntu-kpi.kiev.ua

6. UART и USART. COM-порт. Часть 1. - режим доступу до ресурсу: http://www.rotr.info/electronics/mcu/arm_usart.htm

7. Описание шины I2C - режим доступу до ресурсу: http://www.itt-ltd.com/reference/ref_i2c.html

8. Описание шины CAN- - режим доступу до ресурсу:: http://itt-ltd.com/reference/ref_can.html

9. Принцип действия шины TWI. режим доступу до ресурсу http://www.gaw.ru/html.cgi/txt/doc/micros/avr/arh_xmega_a/19_3.htm

10. 1-Wire-интерфейс - режим доступу до ресурсу: http://www.elin.ru/1-Wire/

11. Universal serial bus режим доступу до ресурсу: http://www.usb.org

12. Последовательный интерфейс SPI (3-wire) - режим доступу до ресурсу: http://www.gaw.ru/html.cgi/txt/interface/spi/index.htm

13. Дитрих Д., Артемов Н.И., Низамутдинов О.Б., Белковский С.В. Fieldbus-концепция построения систем промышленной автоматизации // Приборы и системы. Управление, Контроль, Диагностика, 11/2000. – С. 35-38.

14. Белковский С.В. Анализ протокола в системах полевых шин // Теоретические и прикладные аспекты информационных технологий: Сб. науч. тр. / НИИУМС. – Пермь, 1999. – Вып. 48. – С. 136-138.

15. STM32F407XX Datasheet, PDF Режим доступу до ресурсу: http://www.alldatasheet.com/view.jsp?Searchword=Stm32f407xx