МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ОСНОВИ ПОБУДОВИ КОМП'ЮТЕРНО-ІНТЕГРОВАНИХ СИСТЕМ

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського як навчальний посібник для студентів, які навчаються за спеціальністю 151 "Автоматизація та комп'ютерно-інтегровані технології", освітньо-професійною програмою "Автоматизація та комп'ютерно-інтегровані технології кібер-енергетичних систем"

Київ
КПІ ім. Ігоря Сікорського
2020

Рецензент: *Баранюк О.В., к.т.н., доцент*

Відповідальний редактор: *Волощук В.А., докт. техн. наук, в.о. зав. кафедри*

Гриф надано Методичною радою КПІ імені Ігоря Сікорського (протокол № 9 від 30 . 04 .2020 р.) за поданням Вченої ради факультету (протокол № 9 від 29 . 04 .2020 р.)

Електронне мережне навчальне видання

Любицький Сергій Вікторович, ст. викладач
Новіков Павло Валерійович, асистент

ОСНОВИ ПОБУДОВИ КОМП'ЮТЕРНО-ІНТЕГРОВАНИХ СИСТЕМ

Основи побудови комп'ютерно-інтегрованих систем [Електронний ресурс] : навч. посіб. для студ. спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології», освітньо-професійна програма «Автоматизація та комп'ютерно-інтегровані технології кібер-енергетичних систем» /Укладачі: С. В. Любицький, П. В. Новіков ; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 1,5 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 77 с.

Посібник призначений для студентів, які навчаються за спеціальністю 151 «Автоматизація та комп'ютерно інтегровані технології», освітньо-професійна програма «Автоматизація та комп'ютерно інтегровані технології кібер-енергетичних систем», які вивчають курс «Основи побудови комп'ютерно-інтегрованих систем».

Метою посібника «Основи побудови комп'ютерно-інтегрованих систем» є висвітлення основних принципів побудови інформаційно-обчислювальних систем керування, електронно-обчислювальних пристроїв на базі мікропроцесорної техніки, вбудованих систем, отримання практичних навичок побудови апаратного забезпечення таких систем і програмування систем реального часу.

ЗМІСТ

Вступ

Бажання звільнити себе від одноманітної і тяжкої праці змушувало і змушує людей створювати все більш досконалі технічні пристрої і системи. Комп'ютерний бум в другій половині минулого сторіччя вивів промислове виробництво на принципово новий технологічний рівень, перетворивши автоматизацію з раціоналізаторської ідеї на необхідність для будь-якого підприємства. Цифровізація у виробничих цехах дозволила з'єднати рівень виробництва з рівнем управління підприємством, сформувавши піраміду автоматизованих систем управління. Швидкість обміну інформацією між рівнями збільшилася в рази. Настала епоха тотальної автоматизації. Об'єми масового виробництва зросли на стільки, що одна фабрика тепер може задовольнити потреби сотень мільйонів споживачів. Тим не менш, незважаючи на те, що більшість підприємств світу має технологічний уклад на рівні 3-ї ПР, вже 10 років як говорять про 4 ПР, так звану Industry 4.0. Відмінною ознакою даної революції, порівняно з попередніми, є домінуюча роль саме інформаційної складової і програмного забезпечення. Industry 4.0 – це не революція hardware, це в першу чергу software революція. Базовим поняттям Industry 4.0 є кіберфізична система. Це комплекс механізмів, що контролюються комп'ютерними програмами і алгоритмами на основі штучного інтелекту. Прикладами кіберфізичних систем можуть бути безпілотні автомобілі, розумні будинки, самокеровані дрони, робототехнічні системи автоматичні системи керування.

Чому ж програмне забезпечення стало відігравати таку ж роль, як і залізо? Справа в тому, що з розповсюдженням концепції Інтернету речей і зменшенням фізичних розмірів обчислювальних елементів, процесорів і комп'ютерних плат, роль вбудованих систем значно зросла. Від побутової техніки до промислових роботів – всі пристрої з мікропроцесором на борту можна назвати розумними, а від так запрограмувати на виконання деяких операцій.

У вбудованих системах апаратне і програмне забезпечення завжди були тими двома полюсами, між якими відбувалося перетягування канату. Так от в останні роки роль програмного забезпечення суттєво зросла. Не в останню чергу завдяки розширенню інформаційних мереж і хмарних сервісів.

Поєднання інформаційних технологій та інженерних знань є тією характерною ознакою, яка формує сучасні компенеції кваліфікованого спеціаліста з автоматизації та комп’ютено-інтегрованих технологій.

Лекція 1. Роль і місце вбудованих систем

1.1. Особливості вбудованих систем

Сучасні вбудовані системи управління реального часу (Embedded Systems, або в нашій термінології ВбС,) представляють собою результат міждисциплінарного проектування, в якому умовно можна виділити три основні складові.

- Етап рішення задачі на прикладному рівні, коли необхідно знайти правильні методи і алгоритми без деталей реалізації. Це сфера діяльності прикладних фахівців з відповідних областей (фізика, енергетика, медицина, лінгвістика, біологія та ін.).
- Процес програмування, в ході якого потрібно відобразити отримане прикладне рішення на технологічну базу інформатики та обчислювальної техніки (ОТ). Це робота фахівців з галузі інформатики, сьогодні все частіше її називають архітектурним, високорівневим або системним проектуванням.
- Фаза реалізації, в ході якої інженери, програмісти і прикладні фахівці забезпечують виконання раніше сформульованих вимог, таких як необхідна функціональність, динаміка поведінки, надійність і безпека функціонування, габарити, енергоспоживання, вартість і технологічність при тиражуванні.

До складу простий ВбС входять:

- мікропроцессорниі модуль з пам'яттю;
- периферійна система: датчики, виконавчі елементи і контролери вводу-виводу для зв'язку з об'єктом управління, пристрої людино-машинного інтерфейсу (при необхідності);
- система електроживлення;
- об'єднувальний конструктив (корпус);
- керуюче програмне забезпечення (ПЗ).

Пристрої зв'язку з об'єктом (ПЗО) зазвичай містять аналого-цифрові і цифро-аналогові перетворювачі, порти дискретного вводу-виводу, схеми гальванічної ізоляції та інші елементи. Все це може доповнюватися різноманітними комунікаціонними модулями і пристрями пам'яті.

Основними особливостями ВбС вважаються:

- робота в реальному масштабі часу (майже завжди);
- різноманітні, часто важкі, умови експлуатації;
- автономність роботи (обмеження електроживлення);
- високі вимоги надійності і безпеки функціонування;
- обмеженість ресурсів (низькострумові кола).

ВбС відносяться до категорії систем з переважно програмною реалізацією (Software-Intensive або Software-Dominated Systems). Це означає, більшість функціональності системи реалізується програмним способом. Програмованість і конфігурованість пронизують всі рівні і компоненти ВбС у все більшій мірі. Складність і питома вага програмної складової в ВбС стрімко зростає. З'явився навіть термін «вбудоване програмне забезпечення» (Embedded Software), що підкреслює особливі властивості такого ПЗ і технологій його створення.

Елементну базу ВбС складають електронні, оптичні, механічні та інші фізичні компоненти (елементи, модулі, блоки), з яких складається фізична реалізація ВбС. Сьогодні в переліку таких компонентів знаходяться складні мікросхеми процесорів (мікропроцесори), контролерів, акселераторів, системні плати обчислювачів. У свою чергу, до складу таких елементів входять програмні засоби (завантажувачі, стеки протоколів тощо), які розміщуються у вбудованих блоках постійної пам'яті. Таким чином, навіть традиційне уявлення обчислювальної елементної бази виходить далеко за межі опису тільки конструкції і схемотехніки, зачіпаючи все більше питання системотехніки, програмування, архітектури.

Найбільш широко використовуваними сьогодні в індустрії є наступні платформи:

- промислові ПК;
- програмовані логічні контролери (ПЛК, PLC) і програмовані контролери автоматизації (ПАК, PAC);
- мобільні та інтернет-пристрої (смартфони і планшети);
- мікроконтролери, сигнальні процесори (DSP);
- програмована логіка ПЛІС (PLD, FPGA);
- замовні НВІС (НадВеликі Інтегральні Схеми) (ASIC, ASIP, SoC, Network on Chip - HoC).

1.2. Кібер-фізичний підхід

Створення ВбС, тісно взаємодіючих з фізичними процесами, вимагає технічно складного проектування на низькому рівні. Розробники ВбС змушені «боротися» з контролерами переривань, архітектурою пам'яті, програмуванням рівня асемблера (щоб використовувати спеціалізовані команди чи точно управляти синхронізацією), проектуванням драйверів пристроїв, мережевих інтерфейсів і плануванням процесів, замість того, щоб зосередитися на визначенні необхідного поводження системи. Маса відхилень від основного курсу рішення кінцевої задачі і складність цих технологій змушують шукати нові підходи до проектування. Фахівці з Каліфорнійського університету (Берклі) вважають, що в основі таких підходів має лежати моделювання системи вцілому і технології спільного проектування апаратно-програмного забезпечення, мереж і фізичних процесів. Такий підхід названий кібер-фізичним від терміна «кібер-фізичні системи» (Cyber-Physical Systems, CPS), який був введений Хелен Гілл (Helen Gill) в NSF, США, приблизно в 2006 р, щоб позначити погляд на системи з позиції інтегрування обчислень і фізичних процесів.

1.3. Визначення, класифікація ВбС

Зародження вбудованих систем відбувалося на початку п'ятидесятих років. У той час комп'ютери виготовлялися на громіздкій елементній базі і

були вкрай ненадійними. Для нормальної роботи таких машин були потрібні ідеальні умови експлуатації. Клас обчислювальних систем, призначених для управління і максимально віддалених від об'єкта керування, називали інформаційно-керуючими системами (ІКС). З появою комп'ютерних мереж, приблизно в 70-х роках, з'явилася можливість будувати розподілені або мережеві ІКС. Поява інтегральних мікросхем, а також мікропроцесорів дала можливість наблизити ІКС безпосередньо до об'єкта управління, або навіть вбудувати в нього ЕОМ. Так з'явилися перші вбудовані системи (Embedded System). Поступово, у міру здешевлення елементної бази та збільшення ступеня її інтеграції та збільшення надійності обчислювальних пристроїв з'явилася можливість встановлювати ЕОМ в різні місця об'єкта керування, об'єднуючи всі обчислювальні вузли в єдину контролерну мережу. У процесі подальшого розвитку, завдяки ще більшої мініатюризації і дифузії з об'єктом керування з'явилися так звані кіберфізичні системи, (CPS, Cyber Physical System). CPS характеризуються глибоким зрощенням з механічними, оптичними, хімічними і біологічними системами. Отже, за ступенем проникнення обчислювальної системи в об'єкт керування можна виділити:

- Інформаційно-керуючі системи (ІКС).
- Розподілені інформаційно-керуючі системи (РІКС).
- Вбудовані системи (Embedded System, ES).
- Мережеві вбудовані системи (Networked Embedded System, NES).
- Кіберфізичні системи (Cyber Physical System, CPS).

Останнім часом, через прогрес в області обчислювальної техніки, зміст терміну вбудована система досить сильно видозмінився і розмився. Вміру розвитку техніки відбувалася еволюція позначення класу керуючих комп'ютерних систем: від інформаційно-керуючої системи до вбудованої, від вбудованої до вбудовано-мережевої, а від вбудовано-мережевої до кіберфізичної. У процесі розвитку, відбувалася плавна інтеграція обчислювальної системи і об'єкта керування. Якщо перші ІКС представляли

собою систему, практично не пов'язану з об'єктом керування, то сучасні кіберфізичні системи дуже і дуже тісно інтегровані з об'єктом керування.

Існує безліч визначень терміну «вбудована система» (embedded system), наведемо деякі з них:

Вбудовані обчислювальні системи (ВбОС) – спеціалізовані (замовні) обчислювальні системи, що безпосередньо взаємодіють з об'єктом контролю або управління і об'єднані з ним єдиною конструкцією.

Вбудована обчислювальна система – спеціалізована інформаційно-керуюча система (ІКС) для виконання певного набору функцій.

Вбудована обчислювальна система – будь-яка система, яка використовує комп'ютер як елемент, але чия основна функція не є функцією комп'ютера. Приклади ВбОС: DVD-програвач, світлофор, банкомат, паркомат і т.д.

Вбудованою системою можна вважати будь-яку обчислювальну систему, яка не є ПК, портативним комп'ютером (laptop) або великим універсальним комп'ютером (mainframe computer).

Вбудована обчислювальна система – пристрій, який включає в себе програмований комп'ютер, але не є при цьому комп'ютером загального призначення.

Вбудована обчислювальна система – практично будь-яка обчислювальна система, яка не є настільним комп'ютером.

Вбудована система – система спеціального призначення, в якій обчислювальний елемент повністю вбудовується у пристрій, яким вона керує. На відміну від універсального комп'ютера, вбудована система виконує одну або кілька визначених завдань, зазвичай з дуже конкретними вимогами. У технічному сенсі вбудована система взаємодіє з навколишнім середовищем контрольованим чином, задовольняючи ряд вимог на здатність реагувати в сенсі якості та своєчасності. Як правило, вона повинна задовольняти вимогам реалізації, таким як вартість, споживана потужність і використання

обмежених фізичних ресурсів. В ідеалі вона повинна взаємодіяти з середовищем протягом всього життя об'єкта.

Як правило, ВбС є частиною більшої системи або вбудовується безпосередньо в об'єкт керування. ВбС – це системи «глибоко інтегровані» з об'єктами фізичного світу. Їх елементи практично завжди обмежені в ресурсах. Це системи тривалого життєвого циклу, часто автономні. Масштаб цих систем за розмірами і складності змінюється в дуже широких межах. Ці системи розраховані на непрофесійних користувачів і разом з тим, часто виконують критично важливі функції.

Вбудовувані обчислювальні системи можна класифікувати:

- за областю застосування/призначенням;
- за співвідношенням інформаційних і керуючих функцій, тобто система переважно інформаційна (система збору даних) або керуюча (система автоматичного керування);
- за просторовою локалізацією апаратних блоків:
    - просторово локалізовані;
    - просторово розосереджені.
- за співвідношенням обчислювальної (обробка даних) і комунікаційної (функція вводу-виводу даних) складової;
- за ступенем участі людини:
    - автоматичні системи – системи, в яких оператор виконує тільки функції початкового налаштування і оперативного корегування параметрів і режимів роботи системи. Функції збору даних, передачі та виконання команд управління, оперативне вироблення команд керування відбуваються без участі людини;
    - автоматизовані системи – системи, в яких оператор частково або у повному обсязі забезпечує оперативну обробку даних і формування команд керування виконавчими пристроями (наприклад, телекерування).

- за організацією обробки даних, обчислень (централізовані, децентралізовані);
- за розподілом на рівні завдань і/або функцій між фізичними/логічними модулями системи.

1.4. Режим реального часу

Особливість роботи вбудованої системи полягає в наявності необхідності роботи в реальному масштабі часу (або просто в реальному часі).

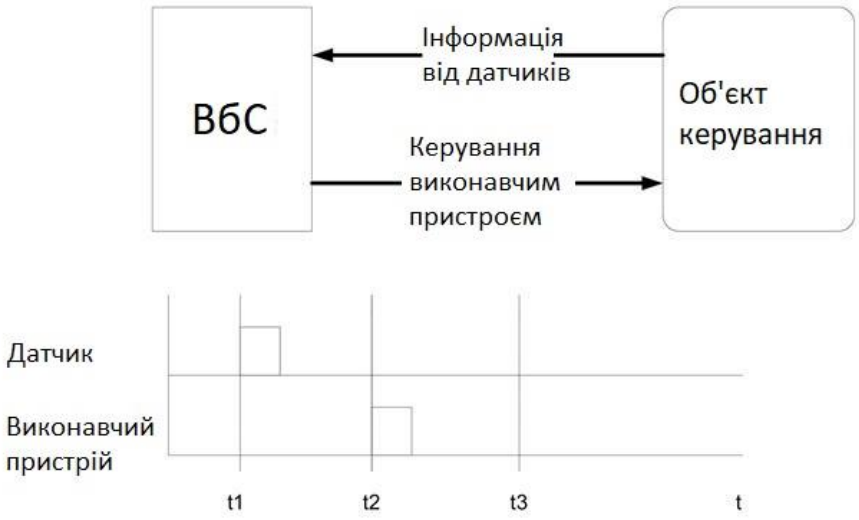


Рис. 1.1. Робота в реальному часі

На Рис. 1.1 показано три часи: t1 - час отримання сигналу з датчика, t2 - час видачі керуючого впливу на виконавчий пристрій. В крайній момент видачі керуючого впливу, якщо з якої-небудь причини видача керуючого сигналу затримається, сигнал буде вироблений після t3, керуючий сигнал буде марний або навіть шкідливий. Як приклад розглянемо систему управління склопідйомниками в автомобілі. Якщо ВбС ігнорує сигнал датчика положення скла, або скло, або механізм, що подає скло, можуть бути зіпсовані.

Система реального часу – обчислювальна система з гарантованим часом реакції на події.

Система реального часу (СРЧ) – будь-яка обчислювальна система, в якій час формування вихідного впливу є суттєвим. Приклади СРЧ – керування технологічними процесами, вбудовані обчислювальні системи, касові торгові системи і т.д.

Принципова відмінність інформаційних систем (lnformation Technology) від систем реального часу (Real-time) в трактуванні параметра «реакція вхід-вихід»:

«*The right answer late is wrong*»

«*Правильна відповідь пізно = неправильна*»

До особливостей вбудованих систем відноситься необхідність забезпечення надійності, безпеки і гарантованого часу реакції. Дотримання гарантованого часу відповіді зазвичай називають роботою в реальному часі.

ВбС отримує інформацію про об'єкт керування за допомогою датчиків. У відповідь на отриману інформацію ВбС виробляє керуючий вплив і передає його об'єкту керування через пристрій сполучення з об'єктом. Час, що протікає між отриманням інформації від об'єкта керування і видачі сигналу управління від ВбС, ми назвемо **часом реакції**.

Система реального часу не повинна бути обов'язково швидкою. Це поширена помилка. Система реального часу повинна видавати керуючі сигнали у відповідь на інформацію, що надходить від датчиків в гарантовані проміжки часу. За ступенем важливості наслідків недотримання часу реакції зазвичай виділяють дві групи систем реального часу:

- Система м'якого реального часу;
- Система жорсткого реального часу.

Система м'якого реального часу (soft real-time system) – затримки задаються середніми величинами. Має місце в організації бізнес-процесів і в торгівлі.

Система жорсткого реального часу – це система реального часу, невиконання тимчасових обмежень якої призводить до катастрофічних

наслідків для цільової функції системи. Має місце в разі військових, космічних, промислових застосувань.

Лекція 2. Піраміда автоматизованих систем управління

Піраміда автоматизації (Computer Integrated Manufacturing pyramid – CIM) – модель, яка об'єднує всі сфери діяльності сучасного підприємства в єдине інформаційне середовище.

Об'єднання всіх рівнів в єдину систему дозволяє:

1. Оптимізувати використовувану інформацію за рахунок централізації і упорядкування потоків даних.
2. Інтегрувати бізнес процеси, процеси управління матеріалами, розробками і виробництвом в єдину систему.
3. Отримати доступ до всіх даних, які існують на підприємстві для аналізу з метою оптимізації управлінських процесів.

Результатом впровадження систем комплексної автоматизації є зниження собівартості продукції, істотне збільшення якості та зниження тривалості циклів виробництва. Кількість рівнів піраміди автоматизації зазвичай коливається від трьох до п'яти. Існують різні трактування одних і тих же термінів у різних системах. У трирівневій моделі CIM є рівні стратегічного планування, тактичного виконання і управління операціями. У чотирирівневої моделі є рівні даних, управління, пристроїв і датчиків, і виконавчих пристроїв. Ми розглянемо CIM на прикладі п'ятирівневої моделі.

Верхні рівні піраміди автоматизації часто будуються за принципами, які можуть застосовуватися для звичайних інформаційних систем. Не всі системи містять всі п'ять рівнів. Прості системи можуть містити в собі один або кілька нижніх рівнів піраміди.

2.1. Рівень підприємства (1)

Верхній рівень відповідає за наступні речі:

- Управління ресурсами (ERP, MRP, MRP II, JIT)
- Управління виробництвом виробів (FMS)
- Управління продукцією (PDM)
- Планування та прогнозування (OLAP)

- Розробка нових виробів (CAD/CAM, CAE)

На цьому рівні ми маємо звичайну корпоративну мережу (на базі Ethernet, HSE FIELDBUS, ATM і т.п.), що складається з комплексу ЛОМ загального призначення, файлових серверів, СУБД і різних систем автоматизації. Рівень характеризується порівняно низькими вимогами до надійності і безпеки до обладнання. Можливе використання офісних комп'ютерів для робочих станцій і серверів. Устаткування функціонує в звичайних офісах.

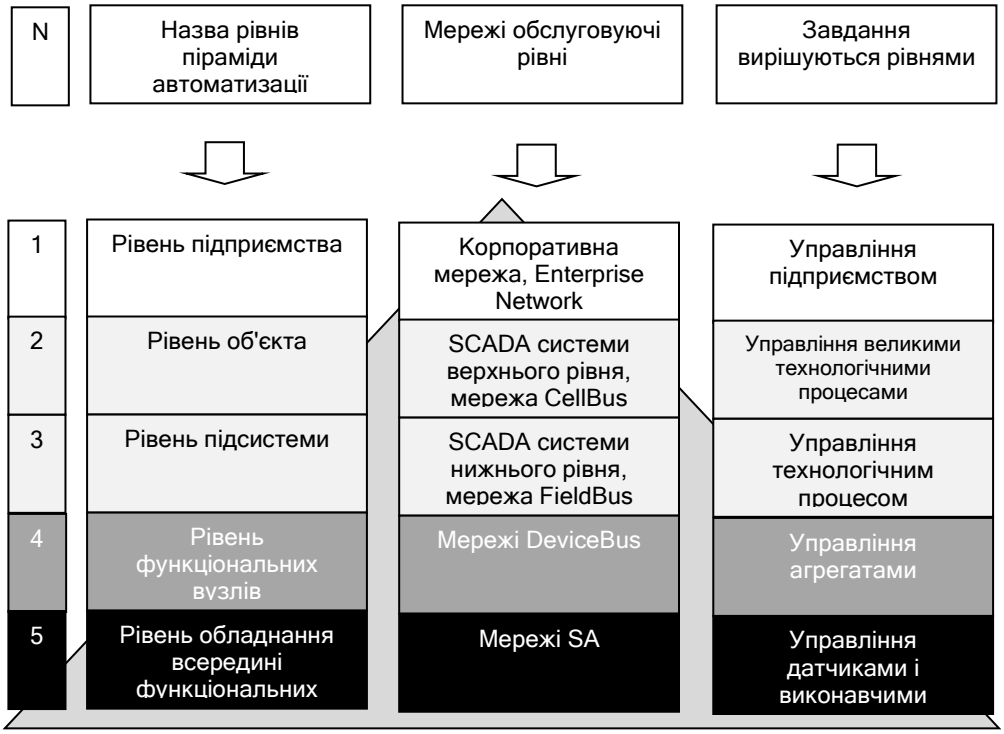


Рис. 2.1. Піраміда АСУ підприємства

2.2. Рівні об'єкта (2) і підсистеми (3)

Наступні два рівні відповідають за управління технологічними процесами. Більш верхній рівень відповідає за великі промислові об'єкти. Дрібніший – за порівняно невеликі технологічні процеси (складові частини великих). На невеликих підприємствах ці два рівня зливаються в один рівень управління. На рівні об'єкта і підсистеми систему управління будують на базі мережі CellBus. У більшості випадків це звичайний Ethernet або HSE FIELDBUS. У вузлах мережі CellBus знаходяться SCADA-системи

(Supervisory Control and Data Acquisition). SCADA-системи рівня об'єкта керують SCADA-системами рівня підсистеми. Комп'ютерне обладнання цих двох рівнів, як правило, знаходиться в захищених приміщеннях (наприклад, в диспетчерських).

2.3. Рівень функціональних вузлів (4)

Рівень функціональних вузлів призначений для управління агрегатами, тобто якимись складними, складовими механізмами. Головним елементом цього рівня є логічні контролери – обчислювальні керуючі модулі, що задають логіку функціонування і координують роботу цих агрегатів (в кінцевому рахунку, об'єктів керування). Як логічні контролери зазвичай виступають програмовані логічні контролери (ПЛК або PLC), промислові комп'ютери і мережеві модулі вводу-виводу. Алгоритм роботи задається програмним забезпеченням логічного контролера. Програмування зазвичай здійснюється кінцевим користувачем (НЕ розробником системи) на спеціальних «технологічних» мовами (наприклад, на мові релейних схем, функціональних блоків та інших). В межах цього рівня використовуються мережі, що входять в категорію DeviceBus, FieldBus.

2.4. Рівень обладнання функціональних вузлів (5)

Найнижчий рівень призначений для збору інформації з датчиків і управління виконавчими пристроями. На цьому рівні працюють інтелектуальні пристрої зв’язку з об'єктом (ПЗО) або мережеві модулі вводу-виводу. Як мережі використовуються мережі SA Bus (Sensor Actuator). До обладнання останніх двох рівнів пред'являються підвищені вимоги до надійності і безпеки. Контролери і мережі знаходяться в безпосередній близькості від об'єкту управління, схильні до дії різних перешкод, агресивних середовищ і вібрацій.

2.5. Пристрої вводу-виводу

Зазвичай, у ВбС пристрій вводу-виводу (ПВВ) містить деяку кількість дискретних і аналогових входів і виходів.

Система вводу-виводу є тим блоком, заради якого зазвичай робиться інформаційно-керуюча система. Якщо «обличчям» (інтерфейсом) офісної системи (якщо так можна висловитися) є дисплей з відповідною графічною оболонкою (десктопом), то «обличчям» ІКС є система вводу-виводу. Системи вводу-виводу можуть бути дуже складні по конструкції і дорогі за ціною. Дуже часто, в системи вводу-виводу вбудовуються додаткові електронні, електричні та електромеханічні пристрої для захисту об'єкта управління або ІКС в різних позаштатних ситуаціях.

2.6. Пристрій зв’язку з об'єктом

Пристрої зв’язку з об'єктом (ПЗО), які також називають модулями вводу-виводу, виконують функції адаптера датчиків і виконавчих пристроїв. Вони мають спеціальні апаратні каскади сполучення з кінцевими пристроями і підтримують алгоритми управління ними. ПЗО можуть виконувати функції первинної обробки даних з датчиків: фільтрацію, усереднення і накопичення. ПЗО є підлеглими по відношенню до логічних контролерів і самостійно не реалізують будь-яких алгоритмів контролю об'єкта управління. За способом взаємодії з логічним контролером ПЗО діляться на:

- Локальні ПЗО, конструктивно суміщені з логічним контролером;
- Дистанційні або мережеві ПЗО, які взаємодіють з логічним контролером з мережевого каналу, конструктивно незалежні від логічних контролерів.

На відміну від ПВВ, ПЗО працює з реальним об'єктом в умовах перешкод, високих напруг і великих струмів (наприклад, управління двигуном змінного струму). Як правило, цей пристрій гальванічно розв'язаний з об'єктом керування. Як ПЗО можуть виступати дискретні (бітові) входи і виходи, аналогові входи і виходи і т.п. Без гальванічної

розв'язки керуючий обчислювальний комплекс буде або постійно давати збої через перешкоди, або згорить при появі підвищеної напруги на вході ПЗО.

Пристрій зв’язку з об'єктом є межею, що відокремлює «тепличний» світ обчислювального пристрою від досить агресивного навколишнього середовища. ПЗО має не тільки сполучати різні рівні сигналів, наприклад 24 Вольт необхідні для включення реле і 5 Вольт (TTL рівень) на виході бітового порту вводу-виводу мікроконтролера, але і не пропускати високі напруги у внутрішні ланцюга контролера.

У деяких системах (наприклад, в медичних) головним є збереження об'єкта керування, тому що від цього безпосередньо залежить життя людини. Такі системи тестуються на ступінь захисту людини від електричної травми через некоректну роботу ПЗО.

2.7. Системи-на-кристалі

Система-на-кристалі (System-on-Chip, SoC) – в загальному випадку системи, на єдиному кристалі яких інтегровані процесор (процесори, в тому числі спеціалізовані), деякий обсяг пам'яті, ряд периферійних пристроїв і інтерфейсів, – тобто максимум того, що необхідно для вирішення завдань, поставлених перед системою. Вираз "система на кристалі" не є, строго кажучи, терміном. Це поняття відображає загальну тенденцію до підвищення рівня інтеграції за рахунок інтеграції функцій.

Продуктивність приладів класу "система-на-кристалі" в значній мірі залежить від ефективності взаємодії всіх встановлених компонентів і від ефективності їх взаємодії з зовнішнім, щодо приладу, світом. В першу чергу це пов'язано з різницею в швидкодії вбудованих компонентів, особливо організації інтерфейсів.

Системи на кристалі зазвичай складаються з трьох основних цифрових системних блоків: процесор, пам'ять і логіка. Процесорне ядро реалізує потік управління, коли кожній керуючій програмі однозначно встановлюються послідовність виконання операцій обробки даних, що дозволяє задавати один

з можливих алгоритмів роботи всієї інтегральної схеми. Пам'ять використовується по її прямому призначенню – зберігання коду програми процесорного ядра і даних. Нарешті, логіка використовується для реалізації функцій спеціалізованих апаратних пристроїв, обробки і проходження даних, склад і призначення яких визначаються кінцевим застосунком.

Реальна система на кристалі містить як мінімум всі три перерахованих блоки, що виключає застосування численних окремих інтегральних схем і реалізацію інтерфейсів зв'язку між ними. Однокристальне конфігуроване або програмоване рішення допускає оперативну зміну своєї внутрішньої апаратної структури і кінцевого призначення як на етапі виробництва, так і в польових умовах, безпосередньо в проекті. Такі інтегральні схеми були віднесені до групи виробів системного рівня інтеграції, але отримали іншу назву Configurable System on a Chip або CSoC. Оскільки термін CSoC не стандартизований, то існують і інші назви виробів цього класу System on Programmable Chip (SoPC), Programmable System on a Chip (PSoC) або просто SoC.

Типова вбудована система, побудована на базі SoC, містить різні набори наступних інтерфейсів і контролерів:

- Системна шина і контролери шин LPC/ISA, PCI, PCMCIA;
- Контролери управління NOR/NAND Flash, SDRAM, SRAM, DDR;
- Контролер Ethernet;
- Послідовні інтерфейси UART, SPI/SSP/uWire, RS-232, RS- 422/RS-485, CAN;
- Бездротові інтерфейси WiFi/IEEE802.11, ZigBee, Bluetooth, IrDA;
- Інтерфейси підтримки Flash-карт пам'яті: SD/MMC, CompactFlash, MemoryStick;
- Контролер LCD STN/TFT/OLED;
- Контролер матричної клавіатури;
- Модулі безпровідної передачі даних GSM/GPRS, CDMA;

- Модулі прийому сигналів супутникових навігаційних систем GPS, Glonass;
- Апаратні підтримки плаваючою точки, шифрування, DRM і т.п .;
- Аудіо- і відеоінтерфейси.

Лекція 3. Програмне забезпечення та інструментальні засоби вбудованих систем

Дана глава присвячена огляду програмного забезпечення (ПЗ) вбудованих систем, мов програмування, використовуваних для розробки ПЗ ВбС, спеціалізованих інструментальних засобів. Особлива увага приділяється обговоренню проблем проектування програмного проекту в загальному і в разі вбудованих систем, особливостям управління такого роду проектами.

3.1. Основні визначення

Програмне забезпечення – незафіксована (*soft* від англ. м'який) частина системи, яку можна змінити. Незмінні системи (*hard* від англ. твердий), наприклад, мережевий комутатор, який має в своєму складі ПЗ (навіть цілі ОС), проте вважається апаратним забезпеченням.

Операційна система реального часу (ОС РЧ) – це засіб розподілу і виділення ресурсів вбудованої системи.

Програмований логічний контролер (ПЛК, PLC) – контролер, програмований кінцевим користувачем, а не професіоналом в області програмування. ПЛК зазвичай випускаються у вигляді наборів модулів конструкторів, з яких користувач сам будує систему. До складу ПЛК входить, як правило, процесорний модуль і кілька модулів вводу-виводу.

3.2. Особливості ПЗ ВбС

До особливостей програмного забезпечення вбудованих систем, як вже говорилося, ми відносимо:

- Реальний час;
- Надійність;
- Безпека;
- Малі ресурси апаратури (пам'ять, швидкодія, електроживлення);
- Важкі умови експлуатації платформи.

Програмне забезпечення вбудованих систем може бути побудовано наступним чином:

- Спеціально під задачу (спеціалізоване ПЗ);
- На базі операційної системи реального часу;
- На базі ОС загального призначення;
- На базі віртуальної машини програмованого логічного контролера.

3.3. Операційні системи реального часу

ОС РЧ в проектуванні є деякою постійною складовою, яка винесена окремо після аналізу безлічі монолітних реалізацій програмного забезпечення ВбС.

Що, по суті, дає застосування ОС РЧ у ВбС? По-перше, це засіб розподілу ресурсів між прикладними процесами і засіб організації цих процесів. По-друге, це налагоджений (тобто з мінімальною кількістю помилок) програмний код з корисною функціональністю. По-третє, ОС РЧ, як правило, є архітектурою зі свідомо відомими плюсами і мінусами. По-четверте, це засіб для організації зв'язку з досить великою номенклатурою апаратних засобів (різних контролерів, периферійних пристроїв). Самостійна підтримка багатьох протоколів обміну, різних процесорів і контролерів, як правило, виявляється нерентабельною для більшості компаній, що створюють ВбС, що також визначає використання готових ОС РЧ.

Які мінуси може принести використання ОС РЧ у ВбС? Природно, більшість ОС РЧ, присутні на ринку, розроблялися як універсальні системи. Універсальність, як правило, означає надмірність функцій і, отже, необхідність в додаткових апаратних ресурсах для підтримки цих функцій. При використанні в проекті готової ОС РЧ існує можливість отримання закритої системи, тобто системи з прихованою внутрішньою структурою. Проти використання такого "чорного ящика" є багато аргументів. Найсильнішим з них є неможливість перевірки системи (наприклад, при

сертифікації) на відсутність серйозних помилок і різного роду неврахованого, "шпигунського" програмного коду.

Останнім часом популярний спосіб проектування систем на базі шаблонів. Так зокрема, в HW/SW CoDesign проектах використовують заготовки ОС РЧ (планувальники, перемикачі процесів та інші). Ці шаблони використовуються на етапі архітектурного проектування. В результаті на виході системи проектування розробник отримує монолітний код. Такий підхід позбавлений більшості недоліків, властивих використанню універсальних (або покупних) ОС РЧ.

Отже, основними причинами, що змушують застосовувати ОС РЧ в складі програмного забезпечення ВбС, будемо вважати:

- Необхідність використання готової, надійної і передбачуваною платформи (виділення з безлічі програм стандартної складової, що підтримує уніфікацію, стандартизацію, модульність);
- Необхідність забезпечення паралельного функціонування прикладних процесів;
- Необхідність забезпечення захисту процесів один від одного;
- Необхідність в готових драйверах периферійних пристроїв, обчислювальної мережі.

3.4. Програмовані логічні контролери

Програмований логічний контролер (ПЛК, PLC) – контролер програмований кінцевим користувачем, а не професіоналом в області програмування. ПЛК зазвичай випускаються у вигляді наборів модулів - конструкторів, з яких користувач сам будує систему. Як правило, до складу ПЛК входить процесорний модуль і кілька модулів вводу-виводу.

3.5. Варіанти побудови систем на базі ПЛК

Існує два основні варіанти побудови систем на базі ПЛК.

В першому варіанті в ПЛК передбачені спеціальні роз'єми розширення, в які можна вставляти пасивні (тобто без власного процесора) модулі вводу-

виводу. Такий варіант кращий, коли потрібно сконцентрувати більшу обчислювальну потужність і велику кількість входів-виходів в одному місці.

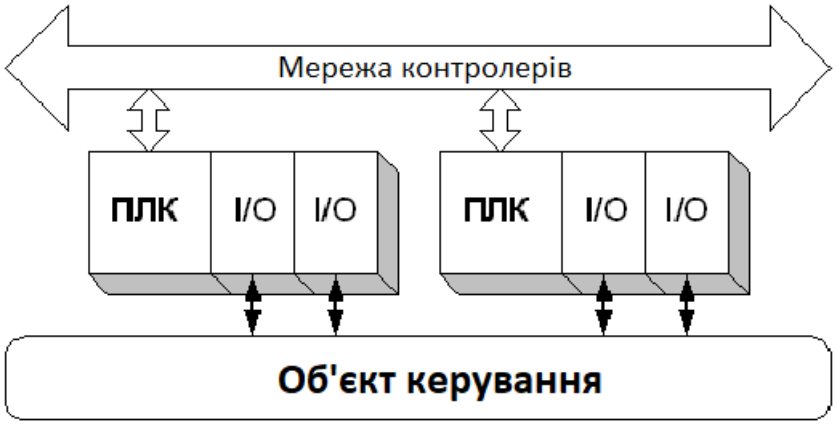


Рис. 3.1. Структура ПЛК зі сконцентрованими модулями

В другому варіанті ПЛК не має своїх входів виходів взагалі або їх обмежена кількість. Додаткова кількість виходів забезпечується за рахунок підключення модулів вводу-виводу через спеціальну промислову мережу. Останній варіант цікавий тим, що дозволяє досить гнучко змінювати масштаб системи управління, залишаючи розробникам значну свободу у виборі рішень.

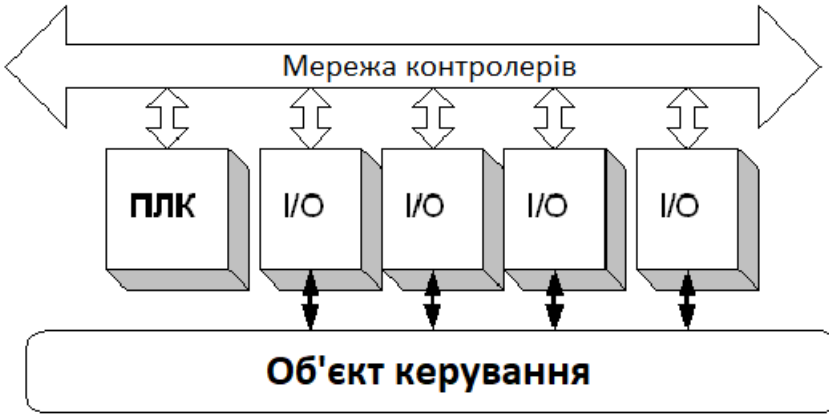


Рис. 3.2. Структура ПЛК з розподіленмиим модулями

3.6. Особливості програмування ПЛК

Як правило, люди, що програмують ПЛК, не є професійними програмістами. Найчастіше ПЛК використовуються в АСУ ТП як промислові контролери. Програмування ПЛК ведеться за допомогою спеціальних мов

програмування IEC1131-3, IEC61131-3, IEC-61499 та інших. Вони дозволяють повністю ізолювати рівень системного програмування від програміста, досягти досить високої надійності функціонування і роботи в реальному масштабі часу.

3.7. Варіанти реалізації ПЛК

Існує два полярних варіанти реалізації ПЛК.

Soft PLC

В першому випадку, як апаратна база береться звичайний промисловий комп'ютер і забезпечується операційною системою реального часу або DOS для индустріальних програм (для комп'ютерів на основі процесора Intel). Далі, на цьому промисловому комп'ютері запускається спеціальна програма – віртуальна машина ПЛК, що реалізує одну або кілька обчислювальних моделей, що використовуються в мовах програмування для ПЛК. В результаті ми отримуємо так званий Soft PLC. Цей варіант побудови ПЛК цікавий своєю гнучкістю. Кінцевий користувач може в широких межах змінювати характеристики програмного забезпечення. Недоліками такого рішення є висока ціна компонентів системи. Вам доведеться купувати промисловий комп'ютер, операційну систему і віртуальну машину ПЛК. Крім того, якщо ви не спеціаліст по операційним системам реального часу, ви можете отримати досить низькі показники системи.

Спеціалізований ПЛК

В другому випадку як апаратна база використовується не промисловий комп’ютер, а спеціалізований контролер. Все необхідне програмне забезпечення вже «зашите» в постійний запам’ятовучий пристрій (ПЗП) на заводі виробника. Користувачеві зазвичай залишаються тільки роботи з конфігурації мережі і розробки прикладної програми. Звичайний ПЛК можна реалізувати як Soft PLC, закритий для зміни користувача, або як спеціалізовану обчислювальну машину, з апаратною підтримкою моделей обчислень, використовуваних в мовах програмування ПЛК.

Перевагою системи на базі спеціалізованого ПЛК є низька вартість, простота використання і висока надійність. До недоліків можна меншу розширюваність апаратної частини спеціалізованого контролера і неможливість зміни системного програмного забезпечення кінцевим користувачем.

3.8. Цикл ПЛК

В основі роботи програмованого логічного контролера лежить циклічне виконання програми.

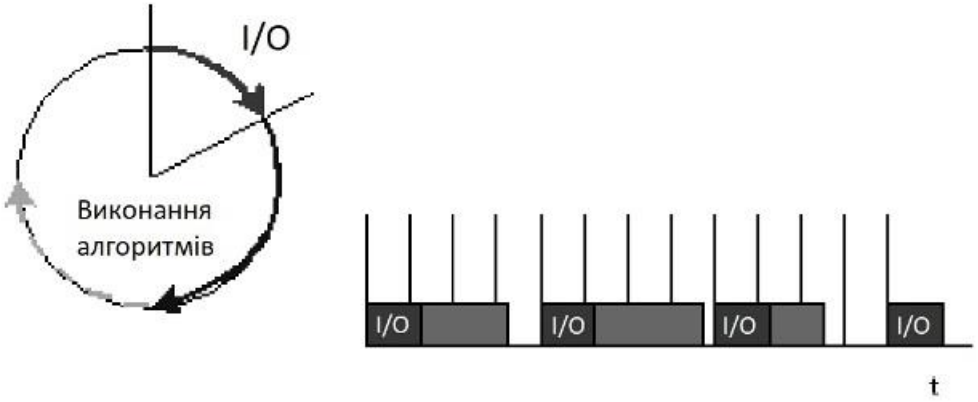


Рис. 3.3. Цикл (скан) ПЛК

На початку циклу виконується введення-виведення. Відбувається обмін між різними пристроями по мережі, отримання інформації з датчиків і виведення інформації на виконавчі пристрої. Після цього відбувається виконання алгоритмів керування. Через деякий час цикл повторюється знову. Особливість такого рішення полягає в тому, що обмін з пристроями вводу-виводу відбувається через строго певні проміжки часу. Це дає гарантований час реакції системи, тобто ми отримуємо систему реального часу досить простим і надійним способом. Звичайно, у будь-якого методу є свої недоліки. До недоліків такого підходу можна віднести достатньо великий час простою центрального процесора. Для того, щоб цикл ПЛК був постійним необхідно, щоб сумарний час вводу-виводу і час виконання алгоритмів керування був завжди менше періоду циклу.

3.9. Області застосування ПЛК

ПЛК добре пристосовані для широкого діапазону задач автоматики. Дуже часто це автоматизація промислових процесів на виробництві, де ціна розробки та підтримки автоматичних систем виявляється набагато більшою сумарної вартості самих систем автоматики, і, там де потрібні зміни в системах протягом періоду їх експлуатації. ПЛК містять пристрої вводу/виводу сумісні з промисловими регуляторами і приводами, тому практично не вимагають електротехнічного проектування, а більшість завдань укладається в опис необхідних наборів операцій в нотації релейної логіки або діаграм станів. ПЛК можуть бути легко переналаштовані на виконання різних завдань.

3.10. Порівняння з мікроконтролерами

Рішення на базі мікроконтролерів будуть ефективніше, в порівнянні з ПЛК, там, де випускаються сотні або тисячі пристроїв (підвищена ціна розробки окупається за рахунок великого обсягу продажів) і там, де кінцевому користувачеві не потрібно змінювати алгоритми управління. Прикладом можуть слугувати системи автомобільної автоматики, мільйони пристроїв випускаються щорічно і тільки їх виробники займаються програмуванням контролерів. Також ПЛК можуть виявитися непридатними в тих областях, де пред'являються підвищені вимоги до обчислювальних ресурсів. Висока швидкість і точність обчислень потрібна в бортових системах авіації і військової техніки.

Лекція 4. Архітектура системи команд та пам’ять

Системою команд обчислювальної машини називають повний набір команд, які може виконувати дана ОМ. В свою чергу, під *архітектурою системи команд* (АСК) прийнято визначати ті засоби обчислювальної машини, які видні та доступні програмісту. АСК можна розглядати як лінію узгодження потреб розробників програмного забезпечення з можливостями творців апаратури обчислювальної машини (рис. 4.1).

Мета одних та інших – реалізація обчислень найбільш ефективним способом, тобто за мінімальний час, і тут важливу роль грає правильний вибір архітектури систем команд.

В спрощеній трактовці час виконання програми (T) можна визначити через число команд в програмі (N), середню кількість тактів процесора, припадаючих на одну команду (CPI), та довжину тактового періоду τ;

$$T = N \times CPI \times \tau.$$

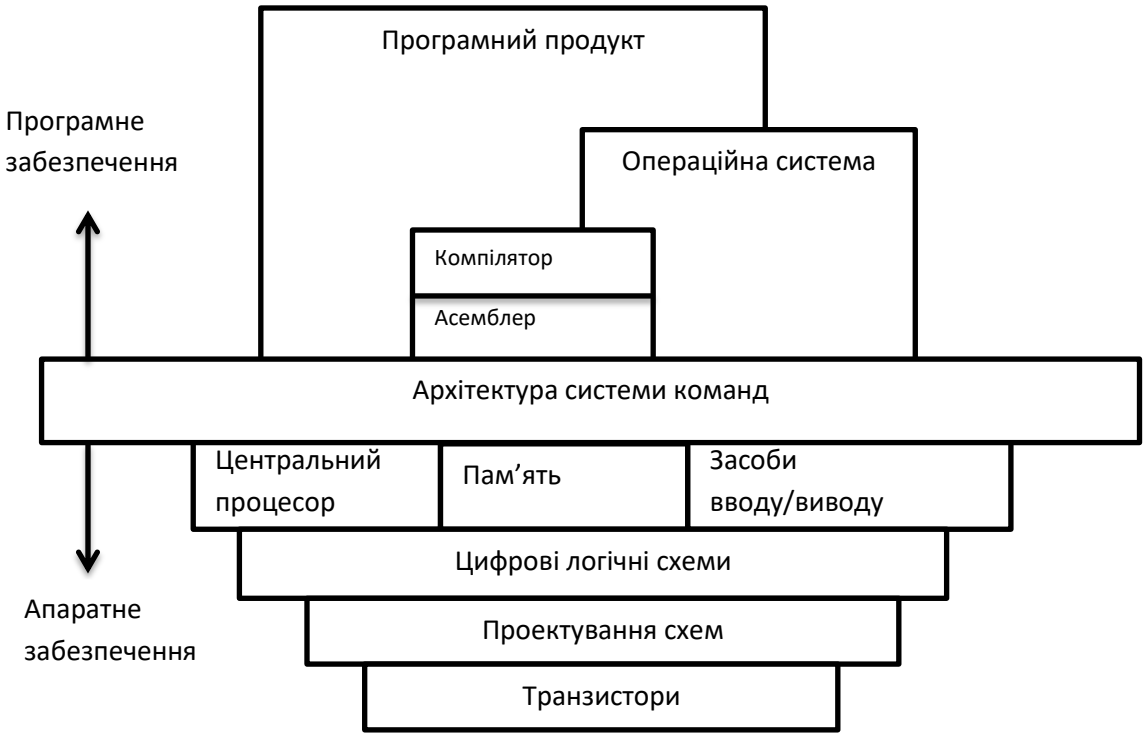


Рис. 4.1. Архітектура системи команд як інтерфейс між програмним і апаратним забезпеченням

4.1. Класифікація архітектури системи команд

В історії розвитку обчислювальної техніки як в дзеркалі відображуються зміни, що відбувалися в поглядах розробників на перспективність однієї чи іншої архітектури системи команд. Ситуацію, що склалася в даний час в області АСК, ілюструє рис. 4.2.

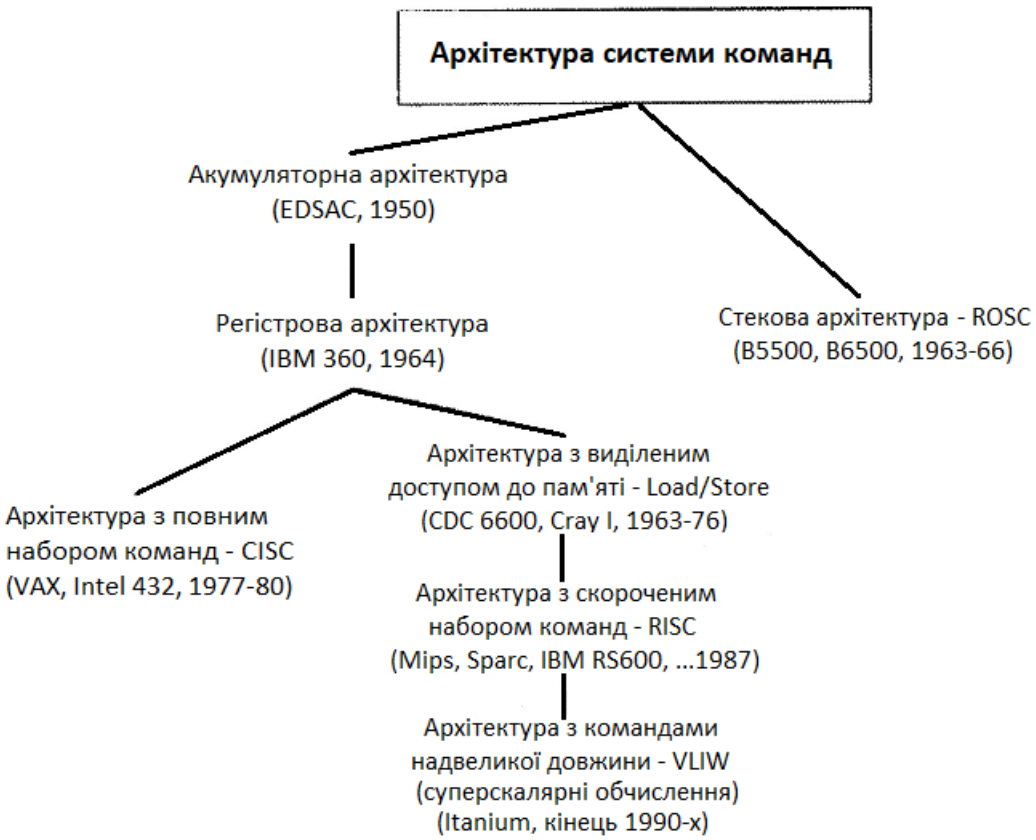


Рис. 4.2. Хронологія розвитку архітектур системи команд

Серед мотивів, що частіше за все зумовлюють перехід до нового типу АСК, зупинимося на двох найбільш суттєвих. Перший – це склад операцій, що виконуються обчислювальною машиною, та їх складність. Другий – місце зберігання операндів, що впливають на кількість та довжину адрес, вказаних в адресній частині команд обробки даних. Саме ці характеристики взяті як показники класифікації архітектур систем команд.

4.2. Класифікація по складу та складності команд

Сучасна технологія програмування орієнтована на мови високого рівня (МВР), головна мета яких – полегшити процес програмування. Але перехід до МВР породив серйозну проблему: складні оператори, характерні для МВР,

суттєво відмінні від простих машинних операцій, які реалізовані в більшості обчислювальних машинах. Наслідком такої невідповідності є недостатньо ефективне виконання програм на ОМ. Проблема отримала назву *семантичний розрив*, а для її розв'язку розробники обчислювальних машин в даний час вибирають один з трьох типів АСК:

- архітектура з повним набором команд: **CISC** (Complex Instruction Set Computer);
- архітектура з скороченим набором команд: **RISC** (Reduced Instruction Set Computer);
- архітектура з командними словами надвеликої довжини: **VLIW** (Very Long Instruction Word).

В CISC-архітектурі семантичний розрив долається за рахунок розширення системи команд, доповнення її складними командами, семантично аналогічними операторам МВР. Основоположником CISC-архітектури вважається компанія IBM, яка почала застосовувати даний підхід з родиною машин IBM 360 і продовжує його в своїх потужних сучасних універсальних ОМ (мейнфреймах). Аналогічний підхід характерний і для компанії Intel в її мікропроцесорах серії x86. Для CISC-архітектури характерні:

- наявність в процесорі порівняно невеликої кількості регістрів загального призначення;
- велика кількість машинних команд, частина яких апаратно реалізують складні оператори МВР;
- різноманітність способів адресації операндів;
- багато форматів команд різної розрядності;
- наявність команд, де обробка здійснюється зі зверненням до пам'яті.

До типу CISC можна віднести майже всі ОМ, які випускались до середини 1980-х років, і значну частину сучасних. Розглянутий спосіб рішення проблеми семантичного розриву разом з тим веде до ускладнення апаратури ОМ, головним чином, пристроїв керування, що, в свою чергу,

негативно впливає на продуктивність ОМ вцілому. Ця обставина спонукала більш уважно проаналізувати програми, отримані після компіляції з МВР. Було зроблено комплекс випробувань, в результаті яких з'ясувалося, що частина додаткових команд, еквівалентних операторам МВР, в загальному об'ємі програм не перевищує 10 – 20%, а для деяких найбільш складних команд навіть 0,2%. В той же час об'єм апаратних засобів, необхідних для реалізації таких додаткових команд, зростає досить суттєво. Так, ємність мікропрограмної пам'яті, що зберігає мікропрограми виконання всіх команд ОМ, із-за введення складних команд може збільшуватись на 60%.

Детальний аналіз результатів згаданих випробувань призвів до серйозного перегляду традиційних рішень, результатом чого стала поява RISC-архітектури. Термін RISC вперше був застосований Д. Патерсоном та Д. Дитцелем в 1880 році. Суть полягає в обмежені списку команд ОМ найчастіше використовуваними простими командами, що оперують даними, розміщеними лише в регістрах процесорів. Звернення до пам'яті допускається лише за допомогою спеціальних команд читання та запису. Різко зменшується кількість форматів команд і способів вказання адреси операндів. Це дозволило суттєво спростити апаратні засоби ОМ та підвищити їх швидкодію. RISC-архітектуру розробляли таким чином, щоб зменшити T за рахунок скорочення CPI та τ. Виявилося, що реалізація складних команд за рахунок послідовності із простих, але швидких RISC-команд не менш ефективна, ніж апаратний варіант складних команд в CISC-архітектурі.

Елементи RISC-архітектури вперше з'явилися в обчислювальних машинах CDC 6600 та суперЕОМ компанії Cray Research. Достатньо успішно реалізується RISC-архітектура і в сучасних ОС.

Відмітимо, що в останній час в мікропроцесорах компаній Intel та AMD широко використовуються ідеї, притаманні RISC-архітектурі, так що багато відмінностей між CISC та RISC поступово стираються.

Окрім CISC- та RISC-архітектур, в загальній класифікації був згаданий ще один тип АСК – архітектура з командними словами надвеликої довжини (VLIW). Концепція VLIW базується на RISC-архітектурі, але декілька простих RISC-команд об’єднуються в одну наддовгу команду та виконується паралельно. В плані АСК архітектура VLIW порівняно мало відмінна від RISC. З’явилося лише додатковий рівень паралельності обчислень, в результаті чого архітектуру VLIW логічніше адресувати не до обчислювальних машин, а до обчислювальних систем.

Таблиця 4.1

Порівняльна оцінка CISC-, RISC- та VLIW- архітектур

Характеристика	**CISC**	**RISC**	**VLIW**
Довжина команди	Варіюється	Єдина	Єдина
Положення полів в команді	Варіюється	Незмінне	Незмінне
Кількість регістрів	Декілька (часто спеціалізованих)	Багато регістрів загального призначення	Багато регістрів загального призначення
Доступ до пам’яті	Може виконуватися як частина команд різних типів	Виконується лише спеціальними командами	Виконується лише спеціальними командами

У будь-якій ЕОМ, незалежно від її архітектури, дані зберігаються в запам'ятовуючих пристроях (ЗП), які, з огляду на їх характеристики, місце розташування і спосіб взаємодії з процесором, відносять до внутрішньої або зовнішньої пам'яті. Внутрішня пам'ять розташовується частково на загальному кристалі з процесором (регістри і кеш-пам'ять), а частково – на системній платі (основна пам'ять і, можливо, кеш-пам'ять 3-го і більш високих рівнів). Повільні ЗП великої ємності (твердотільні, магнітні та оптичні диски, магнітні стрічки) називають зовнішньою пам'яттю, оскільки до ЕОМ вони підключаються аналогічно пристроям вводу-виводу. Основними функціями ЗП є прийом, зберігання і видача даних в процесі роботи ЕОМ. Процес прийому даних, в ході якого здійснюється їх занесення

в ЗП, називається записом, процес видачі дани – читанням або зчитуванням, а спільно їх визначають як процеси звернення до ЗП.

4.3. Характеристики запам'ятовуючих пристроїв внутрішньої пам'яті

Перелік основних характеристик, які необхідно враховувати, розглядаючиконкретний вид ЗП, включає в себе:

- ємність;
- одиницю пересилання;
- метод доступу;
- швидкодію;
- фізичний тип;
- фізичні особливості;
- вартість.

Ємність ЗП характеризують числом бітів, або байтів, які одночасно можуть зберігатися в запам'ятовуючому пристрої. На практиці застосовуються більші одиниці, і для їх позначення до слів «біт» або «байт» додають приставки: кіло, мега, гіга, тера, пета, екса, зетта, йотта (kilo, mega, giga, tera, peta, exa , zetta, yotta). Стандартно ці приставки означають множення основної одиниці вимірювань на 10^3, 10^6, 10^9, 10^{12}, 10^{15}, 10^{18}, 10^{21} і 10^{24} відповідно. У обчислювальної техніці, орієнтованої на двійкову систему числення, вони відповідають значенням, досить близьким до стандартних, але представляє собою цілу ступінь числа 2, тобто 2^{10}, 2^{20}, 2^{30}, 2^{40}, 2^{50},2^{60}, 2^{70}, 2^{80}. Щоб уникнути різночитань 2000 року МЕК – Міжнародна електротехнічна комісія (IEC – International Electrotechnical Comission) затвердила стандарт IEC 60027-2, що передбачає нові позначення, в яких до основної приставці додається склад bi (від англійського binary – двійковий). Так, якщо одиниця вимірювання ємності кратна байту, пропонуються наступні назви та позначення: kibibyte (KiB), mebibyte (MiB), gibibyte (GiB), tebibyte (TiB), pebibyte (PiB), exbibyte (EiB), zettabyte (ZiB), yottabvte (YiB). В

українській транскрипції – кібібайт (КіБ), мебібайт (МіБ), гібібайт (ГіБ), тебібайт (Тіб) і т. д.

Важливою характеристикою ЗП є одиниця пересилання. Для основної пам'яті одиниця пересилання визначається шириною шини даних, тобто кількістю бітів, що передаються по лініях шини паралельно. Зазвичай одиниця пересилання дорівнює довжині слова, але не обов'язково. Так, при пересиланні інформації між основною пам'яттю і кеш-пам’яттю дані передаються одиницями, що перевищують розмір слова, і такі одиниці називаються блоками.

При оцінці швидкодії необхідно враховувати застосовуваний в даному типі ЗП метод доступу до даних.

Розрізняють чотири основні методи доступу:

- послідовний;
- прямий;
- довільний;
- асоціативний,

з яких для внутрішньої пам'яті характерні два останніх. У ЗП з довільним доступом комірка має унікальну фізичну адресу. Звернення до будь-якого осередку займає один і той самий час і може проводитися в будь-якій послідовності (довільної чергованості). Прикладом можуть слугувати запам'ятовуючі пристрої основної пам'яті. Асоціативний доступ дозволяє звертатися до осередків ЗП відповідно до ознак збережених в них даних, він забезпечує пошук осередків, що містять таку інформацію, в якій значення окремих бітів збігається зі станом однойменних бітів в заданому зразку. Порівняння здійснюється паралельно для всіх осередків пам'яті. За асоціативним принципом побудовані деякі блоки кеш-пам'яті.

Швидкодія ЗУ є одним з найважливіших його показників. Для кількісної оцінки швидкодії зазвичай використовують чотири параметри:

- Час вибірки даних. Він відповідає інтервалу часу між початком операції зчитування і видачею зчитаних даних з ЗП.

- Час зберігання даних – інтервал часу, протягом якого пристрій в заданому режимі зберігає дані без регенерації.
- Цикл звернення до ЗП або період звернення. Їм називають мінімальний інтервал часу між двома послідовними доступами до ЗП. Період звернення включає в себе власне час доступу плюс деякий додатковий час. Додатковий час може вимагатися для загасання сигналів на лініях, а в деяких типах ЗП, де зчитування інформації призводить до її руйнування, – для відновлення прочитаної інформації.
- Швидкість передачі даних – кількість даних, що зчитуються (записуються) запам'ятовуючим пристроєм за одиницю часу.

Говорячи про фізичний тип ЗП, необхідно відзначити, що ЗП внутрішньої пам'яті сучасних обчислювальних машин базуються на напівпровідниковій технології.

Залежно від застосованої технології слід враховувати і ряд фізичних особливостей ЗП. Так, для напівпровідникової технології доводиться враховувати фактор енергозалежності. В енергозалежній пам'яті інформація може бути викривлена або втрачена при відключенні джерела живлення, в той час як в енергонезалежних ЗП записана інформація зберігається і при відключенні напруги живлення. Напівпровідникова пам'ять може бути як енергозалежною, так і навпаки, в залежності від її типу. Крім енергозалежності потрібно враховувати, чи приводить зчитування інформації до її руйнування.

Вартість ЗП прийнято оцінювати відношенням загальної вартості ЗП до його ємності в бітах, тобто вартістю зберігання одного біта інформації.

4.4. Ієрархія запам'ятовуючих пристроїв

Пам'ять часто називають «вузьким місцем» фон-неймановської ОМ через її значне відставання по швидкодії від процесорів. Так, якщо продуктивність процесорів зростає вдвічі приблизно кожні 1,5 року, то для мікросхем пам'яті приріст швидкодії не перевищує 9% в рік (подвоєння за 10

років), що виражається в збільшенні розриву в швидкодії між процесором і пам'яттю приблизно на 50% в рік .

Якщо проаналізувати використовувані в даний час типи ЗП, виявляється наступна закономірність:

- чим менше час вибірки, тим вище вартість зберігання біта;
- чим більше ємність, тим нижче вартість зберігання біта, але більше час вибірки.

При створенні системи пам'яті постійно доводиться вирішувати завдання забезпечення необхідної ємності і високої швидкодії за прийнятну ціну. Найбільш поширеним підходом тут є побудова системи пам'яті ЕОМ за ієрархічним принципом. Ієрархічна пам'ять складається з ЗП різних типів (рис. 4.3), які, в залежності від характеристик, відносять до певного рівня ієрархії. Більш високий рівень менше по місткості, швидше і має велику вартість в перерахунку на біт, ніж нижчий рівень. Рівні ієрархії взаємопов'язані: всі дані на одному рівні можуть бути також знайдені на більш низькому рівні, і всі дані на цьому нижчому рівні можуть бути знайдені на наступному нижчому рівні і т. д.

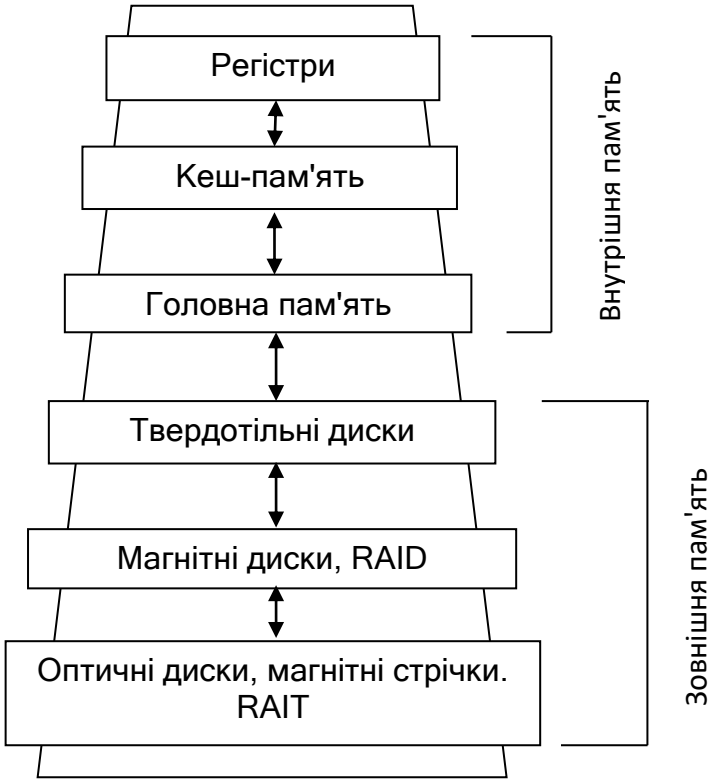


Рис. 4.3. Ієрархія запам'ятовуючих пристроїв

Три верхніх рівня ієрархії утворюють внутрішню пам'ять ЕОМ. а всі нижні рівні – це зовнішня або вторинна пам'ять. В міру руху вниз по ієрархічній структурі:

1. зменшується співвідношення «ціна/біт»;
2. зростає ємність;
3. росте час вибірки;
4. зменшується частота звернення до пам'яті з боку центрального процесора.

Якщо пам'ять організована відповідно до пунктів 1-3, а характер розміщення в ній даних задовольняє пункту 4, ієрархічна організація веде до зменшення загальної вартості при заданому рівні продуктивності.

Справедливість цього твердження випливає з принципу локальності за зверненням. Якщо розглянути процес виконання більшості програм, то можна помітити, що з дуже високою ймовірністю адреса чергової команди програми або слідує безпосередньо за адресою, за якою була зчитана поточна команда, або розташована поблизу неї. Таке розташування адрес називається просторовою локальністю програми. Оброблювані дані, як правило, структуровані, і такі структури зазвичай зберігаються в послідовності суміжних осередків пам'яті. Крім того, програми містять безліч невеликих циклів і підпрограм. Це означає, що невеликі набори команд можуть багаторазово повторюватися протягом деякого інтервалу часу, тобто має місце тимчасова локальність. Всі три види локальності об'єднує поняття локальність за зверненням. Принципу локальності часто надають чисельну форму і подають як так зване правило «90/10»: 90% часу роботи програми пов'язано з доступом до 10% адресного простору цієї програми.

З властивості локальності випливає, що програму розумно представити у вигляді послідовно оброблюваних фрагментів – компактних груп команд і оброблюваних ними даних. Помістивши такі фрагменти в більш швидку пам'ять, можна істотно знизити загальні затримки на звернення, оскільки команди і вихідні дані, будучи один раз передані з повільного ЗП в швидкий,

потім можуть використовуватися багаторазово, і середній час доступу до них в цьому випадку визначається вже більш швидким ЗП. Це дозволяє зберігати великі програми і масиви даних на повільних, ємних, але дешевих ЗП, а в процесі обробки активно використовувати порівняно невелику швидку пам'ять, збільшення ємності якої пов'язане з високими витратами.

На кожному рівні ієрархії інформація (дані) розбивається на блоки, які виступають як найменша інформаційна одиниця, що пересилається між двома сусідніми рівнями ієрархії. Розмір блоків може бути фіксованим або змінним. При фіксованому розмірі блоку ємність пам'яті зазвичай кратна його розміру. Розмір блоків на кожному рівні ієрархії найчастіше різний і збільшується від верхніх рівнів до нижніх.

При доступі до команд і вихідних даних, наприклад для їх зчитування, спочатку проводиться пошук в пам'яті верхнього рівня. Факт виявлення потрібної інформації називають попаданням (hit), в іншому випадку говорять про промах (miss). При промаху проводиться пошук в ЗП наступного нижчого рівня, де також можливі потрапляння або промах. Після виявлення необхідної інформації виконується пересилання блоку, що містить потрібну інформацію, з нижніх рівнів на верхні.

При оцінці ефективності подібної організації пам'яті зазвичай використовують такі характеристики:

- коефіцієнт попадання (hit rate) – відношення числа звернень до пам'яті, при яких відбулося потрапляння, до загальної кількості звернень до ЗП даного рівня ієрархії;
- коефіцієнт промахів (miss rate) – відношення числа звернень до пам'яті, при яких мав місце промах, до загальної кількості звернень до ЗП даного рівня ієрархії;
- час звернення при потраплянні (hit time) – час, необхідний для пошуку потрібної інформації в пам'яті верхнього рівня (включаючи з'ясування, чи є звернення потраплянням), плюс час на фактичне зчитування даних;

- витрати на промах (miss penalty) – час, необхідний для заміни блоку в пам'яті більш високого рівня на блок з потрібними даними, розташований в ЗП наступного (більш низького) рівня. Витрати на промах включають в себе: час доступу (access time) – час звернення до першого слова блоку при промаху і час пересилки (transfer time) – додатковий час для пересилки залишкових слів блоку. Час доступу обумовлений затримкою пам'яті нижчого рівня, в той час як час пересилки пов'язаний з пропускною здатністю каналу між ЗП двох суміжних рівнів.

Найшвидший, але і мінімальний по ємності тип пам'яті – це внутрішні регістри ЦП, які іноді об'єднують поняттям надоперативний пристрій (НОЗП) або регістровий файл. Як правило, кількість регістрів невелика, хоча в архітектурі зі скороченим набором команд їх число може доходити до декількох сотень. Основна пам'ять, значно більшої місткості, розташовується нижче. Між регістрами ЦП і основною пам'яттю часто розміщують кеш-пам'ять, яка по ємності відчутно програє основній пам’яті, але істотно перевершує останню за швидкодією, поступаючись в той же час НОЗП. Всі види внутрішньої пам'яті реалізуються на основі напівпровідникових технологій і в основному є енергозалежними. Довготривале зберігання великих обсягів даних забезпечується зовнішніми ЗП, серед яких найбільш поширені ЗП на основі магнітних і оптичних дисків. Останнім часом все більшої популярності набувають твердотільні диски на базі флеш пам'яті. Ще один рівень ієрархії може бути доданий між основною пам'яттю і магнітними дисками. Цей рівень має назву дискової кеш-пам'яті і реалізується у вигляді самостійного ЗП, що включається до складу магнітного диска. Дискова кеш-пам'ять суттєво підвищує продуктивність при обміні інформацією між дисками і основною пам'яттю.

4.5. Блокова організація основної пам'яті

Ємність основної пам'яті сучасних ЕОМ занадто велика, щоб її можна було реалізувати на базі єдиної інтегральної мікросхеми (ІМС). Необхідність об'єднання декількох ІМС ЗП виникає також, коли розрядність осередків в мікросхемі ЗП менше розрядності слів ЕОМ. Збільшення розрядності ЗП реалізується за рахунок об'єднання адресних входів ІМС ЗП, що об'єднуються. Інформаційні входи і виходи мікросхем є входами і виходами модуля ЗП збільшеною розрядності (рис. 4.4). Отриману сукупність мікросхем називають модулем пам'яті. Модулем можна вважати і єдину мікросхему, якщо вона вже має потрібну розрядність. Один або кілька модулів утворюють банк пам'яті.

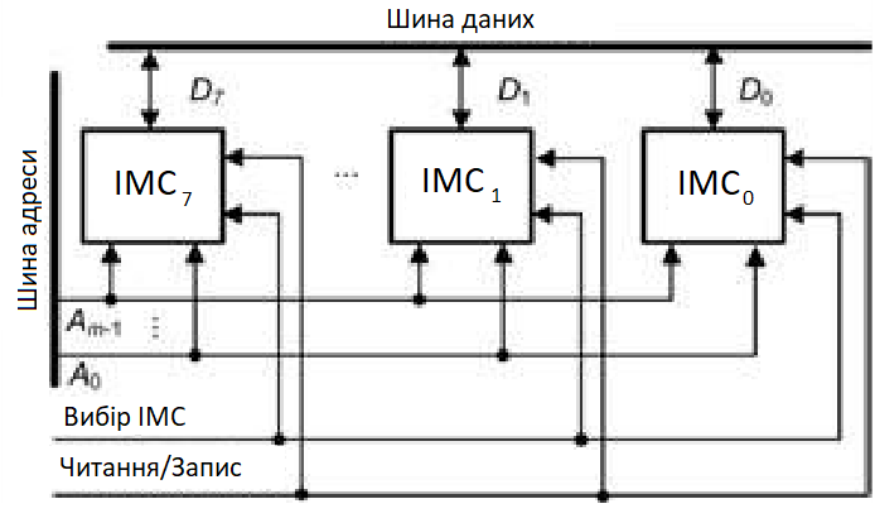


Рис. 4.4. Збільшення розрядності пам'яті

Для отримання необхідної ємності ЗП потрібно певним чином об'єднати кілька банків пам'яті меншої ємності. У загальному випадку основна пам'ять ЕОМ практично завжди має блочну структуру, тобто містить кілька банків.

Лекція 5. Організація шин

5.1. Загальні визначення

Сукупність трактів, які об’єднують між собою основні пристрої ЕОМ, утворють *систему з’єднань* обчислювальної машини. Система з’єднань повинна забезпечувати обмін інформації між:

- центральним процесором і пам’яттю;
- центральним процесором і модулями вводу/виводу;
- пам’яттю і модулями вводу/виводу.

З розвитком обчислювальної техніки система з’єднань пристроїв ЕОМ зазнавала змін (рис. 5.1). На початковому етапі переважали безпосередні зв’язки між пристроями ЕОМ. З появою міні ЕОМ і особливо перших мікроЕОМ більш популярною стає схема з однією загальною шиною. За цим пішов швидкий ріст продуктивності практично всіх пристроїв ЕОМ, який призвів до того, що одна шина не може впоратися зі збільшеним трафіком, і їй на зміну приходять структури на базі з декількома шинами.

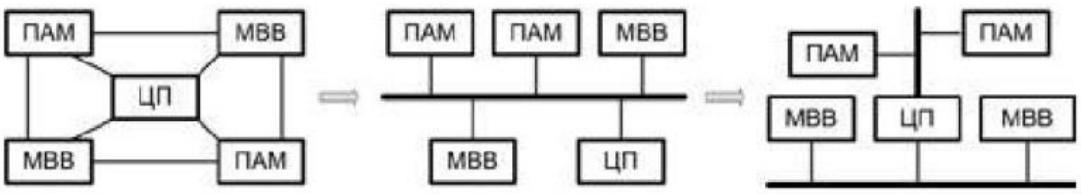


Рис. 5.1. Еволючція систем з’єднань (ЦП – центральний процесор, ПАМ – модуль основної пам’яті, МВВ – модуль вводу/виводу)

Взаємозв'язок частин обчислювальної машини і її «спілкування» із зовнішнім світом забезпечує система шин. Більшість ЕОМ містять кілька різних шин, кожна з яких оптимізована під певний вид комунікацій. Частина шин схована всередині інтегральних мікросхем або до неї існує доступ тільки в межах друкованої плати. Деякі шини мають доступні ззовні точки, щоб до них можна було легко під’єднати додаткові пристрої, причому більшість таких шин не просто доступні, але і відповідають певним стандартам, що дає

змогу під’єднати до шини пристрої різних виробників. Щоб охарактеризувати конкретну шину, потрібно описати:

- сукупність сигнальних ліній;
- фізичні, механічні і електричні характеристики шини;
- сигнали арбітражу, що використовуються, стани, управління і синхронізації;
- правила взаємодії підключених до шини пристроїв (протокол шини).

Шину утворює набір ліній зв’язку. Лінія зв’язку – це фізичне середовище, що забезпечує пересилання сигналів, найчастіше це двійкові цифри 1 і 0. По лінії може передаватися розгорнута в часі послідовність таких сигналів. При спільному використанні кілька ліній можуть забезпечити одночасну (паралельну) передачу двійкових чисел. Лініями зв'язку можуть бути дроти, смужки провідника на монтажній платі, оптоволокно, радіо та інфрачервоні канали.

Операції на шині називають транзакціями. Основні види транзакцій – транзакції читання і транзакції запису. Якщо в обміні бере участь пристрій вводу/виводу, можна казати про транзакції вводу і виводу, еквівалентні транзакціям читання й запису відповідно. Шинна транзакція включає в себе дві складові: посилку адреси та приймання (або посилку) даних.

Коли два пристрої обмінюються інформацією через шину, одне з них повинно ініціювати обмін і керувати ним. Такі пристрої будемо називати *ведучими* (bus master). У комп'ютерній термінології «ведучий» – це будь-який пристрій, який може взяти на себе управління шиною і керувати пересиланням даних. Ведучий не обов'язково використовує дані для своїх потреб. Він, наприклад, може захопити управління шиною в інтересах іншого пристрою. Пристрої, що не мають можливості ініціювати транзакції, звуться *веденими* (bus slave). Фактично, до шини може бути підключено декілька потенціальних ведучих, але в будь-який момент часу активним може бути тільки один із них: якщо кілька пристроїв передають інформацію одночасно, їх сигнали перекриваються і спотворюються. Для запобігання одночасної

активності декількох ведучих у шині передбачається процедура допуску до управління шиною тільки одного з претендентів (арбітраж). Водночас деякі шини допускають широкомовний режим запису, коли інформація одного ведучого передається відразу декільком веденим (тут арбітраж не вимагається). Сигнал, спрямований одним пристроєм, доступний усім іншим пристроям, які під’єднані до шини.

Англійський еквівалент терміну «шина» – «bus» – походить від латинського слова *omnibus*, що означає «для всього». Цим прагнуть підкреслити, що шина виступає як магістраль, що може забезпечити всілякі види трафіку.

У цьому розділі розглядаються тільки загальні питання, що стосуються організації, функціонування й застосування шин, без орієнтації на конкретні реалізації.

5.2. Системна шина

Для зниження вартості деякі ЕОМ мають загальну шину для пам'яті і пристроїв вводу/виводу. Така шина часто називається системною. *Системна шина* потрібна для фізичного й логічного об'єднання всіх пристроїв ЕОМ.

Системна шина зазвичай складається з декількох сотень ліній, які можна поділити на три функціональних групи (рис. 5.2): шину даних, шину адреси і шину керування. До останньої переважно відносять також лінії для подання напруги живлення на під’єднані до системної шини пристрої.

Рис. 5.2. Структура системної шини

5.3. Шина адреси

Будь-яка транзакція на системній шині починається з виставлення ведучим адресної інформації. Адреса дозволяє обрати ведений пристрій і з’єднати його з ведучим. Для передачі адреси використовується частина сигнальних ліній шини, сукупність яких часто називають шиною адреси (ША).

На ША можуть видаватися адреси комірок пам'яті, номера регістрів ЦП, адреси портів вводу/виводу і т. п. Різноманіття видів адреси передбачає наявність додаткової інформації, що описує її вид, який використовується в даній транзакції. Така інформація може міститися в самій адресі, але частіше передається спеціальними керуючими лініями.

Різноманітною може бути і структура адреси. В адресі старші біти можуть вказувати на один із модулів основної пам'яті, тоді як молодші біти визначають комірку всередині цього модуля.

У деяких шинах передбачені адреси спеціального виду, що забезпечують одночасний вибір певної групи ведених або всіх ведених відразу (*broadcast*). Зазвичай це практикується в транзакціях запису (від ведучого до веденого), проте існує також спеціальний вид транзакції читання (одночасно від декількох ведених загальному ведучому). Англійська назва такої транзакції читання *broadcall* можна перекласти як «широкомовне опитування». Інформація, яка повертається ведучому, являє собою результат побітового логічного додавання даних, що надійшли від усіх ведених.

Число сигнальних ліній, які виділені для передачі адреси (*ширина шини адреси*), визначає максимально можливий розмір адресного простору. Це одна з основних характеристик шини, оскільки від неї залежить потенційна ємність адресної пам'яті і кількість обслуговуваних портів вводу/виводу. В сучасних мікропроцесорах шина адреси складається з 36 сигнальних ліній, при цьому адресний простір складає 64 Гбайт. У перспективних розробках передбачається розширення ША до 40 ліній.

5.4. Шина даних

Сукупність ліній для пересилання даних між модулями системи, називають *шиною даних* (ШД). Найважливіші характеристики ШД – ширина і пропускна здатність.

Ширина шини даних визначається кількістю бітів інформації, яка може бути передана по шині за одну транзакцію *(цикл шини*). Цикл шини слід відрізняти від періоду тактових імпульсів – одна транзакція на шині може займати кілька тактових періодів. В середині 1970-х років типова ширина шини даних становила 8 бітів. У наш час це зазвичай 32, 64 або 128 бітів. У будь-якому випадку ширину ШД вибирають кратному цілому числу байтів, причому це число, як правило, являє собою цілу ступінь числа 2.

Елемент даних, який задіює всю ширину ШД, прийнято називати *словом*, хоча в архітектурі деяких ЕОМ поняття «слово» трактується по-іншому, тобто слово може мати розрядність, яке не збігається з шириною ШД.

У більшості шин використовуються адреси, що дозволяють вказати окремий байт слова. Це корисна властивість, коли бажано змінити в пам'яті лише частину повного слова.

При передачі по ШД частини слова пересилання зазвичай проводиться тими ж сигнальним лініях, що і в разі пересилання повного слова, проте в ряді шин «урізане» слово передається молодшими лініями ШД. Останній варіант може виявитися зручнішим при подальшому розширенні шини даних, оскільки в цьому випадку зберігається сумісність зі «старою» шиною.

Ширина шини даних істотно впливає на продуктивність ЕОМ. Якщо ширина шини даних вдвічі менше довжини команди, ЦП протягом кожного циклу команди змушений здійснювати доступ до пам'яті двічі.

Якщо адреса і дані в системній шині передаються через незалежні (виділені) сигнальні лінії, то ширина ША і ШД зазвичай вибираються самостійно. Найбільш часті комбінації: 16-8, 16-16, 20-8, 20-16, 24-32, 32-32 і 36-64.

У деяких ЕОМ адреса і дані пересилаються по одних і тих же лініях, але в різних тактах циклу шини. Цей прийом називається *тимчасовим мультиплексуванням*. В цьому випадку лінії адреси і даних об'єднані в *єдину мультиплексну шину адреси/даних*. Така шина функціонує в режимі розподілу часу, оскільки цикл шини розбитий на часовий інтервал для передачі адреси і часовий інтервал для передачі даних.

Мультиплексування адрес і даних передбачає наявність мультиплексора на одному кінці тракту пересилання інформації та демультиплексор на його іншому кінці. Мультиплексори і демультиплексори грають роль комутуючих пристроїв. Вони забезпечують перенаправлення інформації з одного зі своїх входів на загальний вихід (мультиплексор) або з загального входу на один з виходів (демультиплексор).

Мультиплексування дозволяє скоротити загальне число ліній, але вимагає ускладнення логіки зв'язку з шиною. Крім того, воно веде до потенційного зниження продуктивності, оскільки виключає можливість паралельної передачі адрес і даних, що можна було б використовувати в транзакціях запису, одночасно виставляючи на ША адреси, а на ШД – слово, яке записується.

5.5. Шина керування

Крім трактів пересилання адреси і даних, невід’ємним атрибутом будь-якої шини є лінії, за якими передається *керуюча інформація*, а також інформація про стан пристроїв, які беруть участь в транзакції . Сукупність таких ліній прийнято називати *шиною керування* (ШК). Сигнальні лінії, що входять в ШК, можна умовно розділити на кілька груп.

Першу групу утворюють лінії, за якими пересилаються *сигнали управління транзакціями*, тобто сигнали, які визначають:

- тип виконуваної транзакції (читання або запис);
- кількість байтів, переданих по шині даних, і якщо пересилається частина слова, то номери байтів;

- який тип адреси виданий на шину адреси;
- який протокол передачі повинен бути застосований.

На перераховані цілі зазвичай відводиться від двох до восьми сигнальних ліній.

До другої групи відносять лінії передачі *інформації стану (статусу)*. У цю групу входять від однієї до чотирьох ліній, за якими ведений пристрій може інформувати ведучого про свій стан або передати код помилки.

Третя група – *лінії арбітражу*. Арбітраж необхідний для вибору одного з декількох ведучих, які одночасно претендують на доступ до шини. Число ліній арбітражу в різних шинах варіюється від 3 до 11.

Четверту групу утворюють *лінії переривання*. По цих лініях передаються запити на обслуговування, що посилаються від ведених пристроїв до ведучого. Під запити зазвичай відводяться одна або дві лінії, проте при одночасному виникненні запитів від декількох ведених виникає проблема арбітражу, для чого можуть знадобитися додаткові лінії, якщо тільки з цією метою не використовуються лінії третьої групи.

П'ята група – лінії для організації *послідовних локальних мереж*. Наявність від 1 до 4 таких ліній є загальноприйнятою практикою в сучасних шинах. Обумовлено це тим, що послідовна передача даних протікає значно повільніше, ніж паралельна, і мережі значно вигідніше їх будувати без завантаження швидких ліній основних шин адреси і даних. Крім того, шини цієї групи можуть бути використані як повноцінний, хоча і повільний, надлишковий тракт для заміни ША і ШД в разі їх відмови. Іноді шини п'ятої групи призначаються для реалізації спеціальних функцій, таких, наприклад, як обробка переривань або сортування пріоритетів завдань.

У деяких ШК є **шоста група** сигнальних ліній – від 4 до 5 *ліній позиційного коду*, що приєднуються до спеціального виходу роз'єму. За допомогою перемичок на цих виходах можна задати унікальний позиційний код роз'єму на материнській платі або вставлений в цей роз'єм дочірньої

плати. Такий код може бути використаний для індивідуальної ініціалізації кожної окремої плати при включенні або перезавантаження системи.

В кожній шині обов'язково присутні лінії **сьомої групи**, яка по суті є однією з найважливіших. Це група ліній *тактування і синхронізації*. При проектуванні шини таким лініях приділяється особлива увага. До складу групи, в залежності від протоколу шини (синхронний або асинхронний), входять від двох до шести ліній.

Необхідно згадати лінії для підведення напруги живлення і лінії заземлення.

Функціонування системної шини можна описати таким чином. Якщо один з модулів хоче передати дані в інший, він повинен виконати дві дії: отримати в своє розпорядження шину і передати по ній дані. Якщо якийсь модуль хоче отримати дані від іншого модуля, він повинен отримати доступ до шини і за допомогою відповідних ліній управління і адреси передати в інший модуль запит. Далі він повинен очікувати, поки модуль, який отримав запит, передасть дані.

Фізично системна шина являє собою сукупність паралельних електричних провідників. Цими провідниками служать металеві смужки на друкованій платі. Шина підводиться до всіх модулів, і кожен із них приєднується до всіх або до деяких її лінії. Якщо ЕОМ конструктивно виконана на декількох платах, то всі лінії шини виводяться на роз'єми, які потім об'єднуються провідниками на загальному шасі.

Лекція 6. Інтерфейс 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 один від одного.

До складу UART входять:

- тактовий генератор зв’язку (бодрейт-генератор);
- керуючі регістри;
- статусні регістри;
- буфери;
- регістри зсуву приймача і 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/передачі 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:

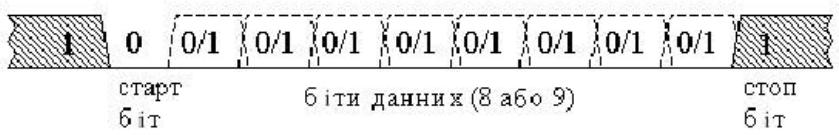


Рис. 1.1. Кадр посилки UART

Рис. 6.1. Кадр посилки UART

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

Швидкість зв'язку (baudrate) виміряється в бодах як число переданих бітів за секунду (включаючи старт і стоп-біт). 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 (семплює) порт RX. Це саме середина старт-бита. Якщо більшість значень сем плів – "0", старт-біт вважається прийнятим, інакше приймач приймає його за шум і чекає наступного падаючого фронту.

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

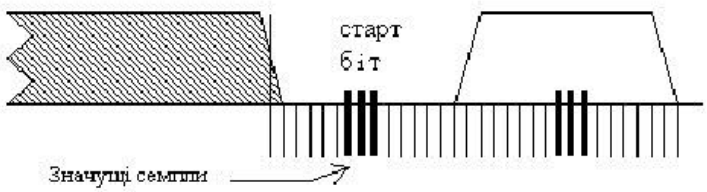


Рис. 1.2. Визначення бітів

Рис. 6.2. Визначення бітів

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

Лекція 7. Інтерфейс SPI

7.1. Характеристика та структура інтерфейса SPI.

Послідовний периферійний інтерфейс SPI (Serial Peripheral Interface) розроблений як повнодуплексний чотирипровідний інтерфейс з шинною конфігурацією під'єднуваних вузлів (пристроїв) 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 простроїв по інтерфейсу SPI наведена на рис. 7.1.

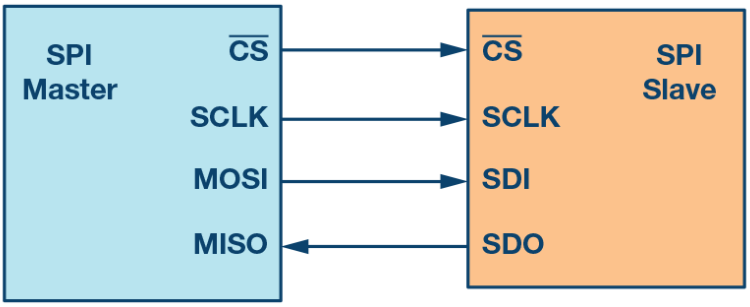


Рис. 7.1. Схема підключення двох пристроїв по інтерфейсу SPI

Інтерфейс SPI мас 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) – 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.

7.2. 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.

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.

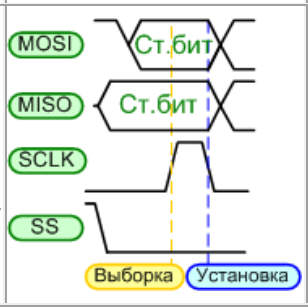
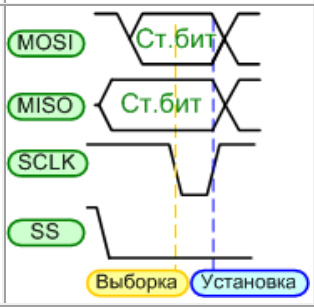
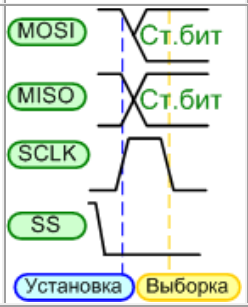
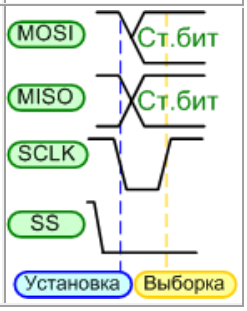
Третій регістр – регістр конфігурації інтерфейсу – SPIOCFG, містить біти CKPHA (SPI Clock Phase) – управління фазою тактування, і CKPOL (SPI Clock Polarity) – управління полярністю тактуючих імпульсів. Ці біти дозволяють вибрати фазу і полярність імпульсів тактування.

CKPOL – вихідний рівень сигналу синхронізації (якщо CKPOL = 0, то лінія синхронізації до початку циклу передачі і після його закінчення має низький рівень (тобто перший фронт наростаючий, а останній – спадаючий), інакше, якщо CKPOL = 1, – високий (тобто перший фронт спадаючий, а останній – наростаючий));

CKPHA – фаза синхронізації; від цього параметра залежить, в якій послідовності виконується установка і вибірка даних (якщо CKPHA = 0, то по передньому фронту в циклі синхронізації буде виконуватися вибірка даних, а потім, по задньому фронту, – встановлення даних; якщо ж CKPHA = 1, то встановлення даних буде виконуватися по передньому фронту в циклі синхронізації, а вибірка – по задньому). Інформація по режимам SPI узагальнена в таблиці 7.1.

Таблиця 7.1

Режими роботи SPI

Режим SPI	0	1	2	3
CKPOL	0	1	0	1
CKPHA	0	0	1	1
Часова діаграма першого циклу синхронізації				

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.1. Якщо інтерфейс встановлений в режим ведучого, то запис байта даних в регістр даних SPIODAT призводить 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).

Класична архітектура SPI тини для базового варіанту інтерфейсу приведена на рис. 7.2. 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 вводу/виводу, яка з'єднується з входом NSS відповідного веденого. Зазвичай така архітектура використовується для побудови мікроконтролерних систем з одним мікроконтролером, 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.

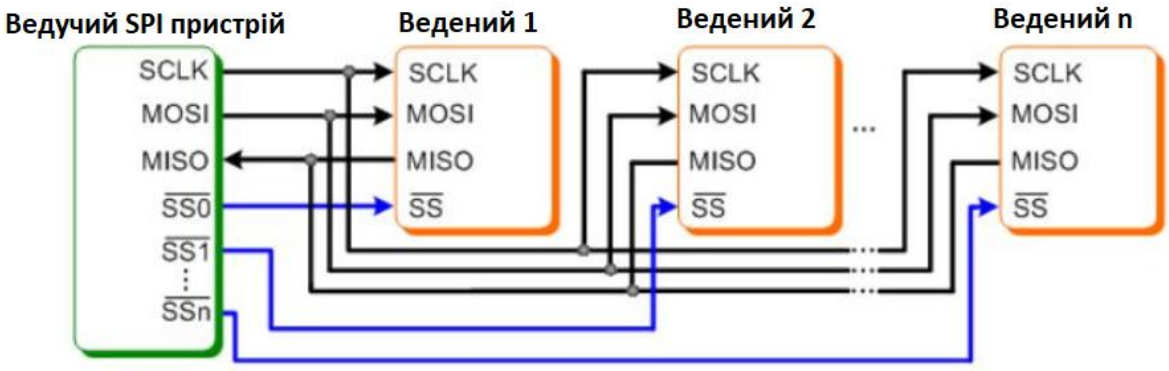


Рис. 7.2. Архітектура 4х-проводной SPI шини з одним ведучим і декількома веденими

Лекція 8. Інтерфейс I2C

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 ведучий/введений: 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 в ситуації, коли два або більше ведучих одночасно починають передачу даних. В стандартному режимі забезпечується передача послідовних 8-бітових даних з швидкістю до 100 кбіт/с, і до 400кбіт/с в "швидкому" режимі.
- Вбудований в мікросхеми фільтр пригнічує сплески, забезпечує цілісність даних.
- Максимальна допустима кількість мікросхем, приєднаних до однієї шини, обмежується максимальною ємкістю шини 400 пФ.

До недоліку 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 дротів з’єднання, воно також включає всі формати і процедури для зв’язку усередині системи.

Пристрої, що зв’язуються по шині, повинні мати деякий протокол, який попереджує всі можливі колізії, втратити даних і блокування інформації. Швидкі пристрої мають бути в змозі зв’затися з повільними пристроями. Система не має бути залежна від пристроїв, підключених до неї, інакше модифікації і поліпшення стануть неможливими. Також має бути розроблена процедура, яка встановлює, який пристрій керує шиною і коли. Крім того,

якщо різні пристрої з різними тактовими частотами підключення до шини, має бути визначене джерело синхронізації шини. Всім цим критеріям задовольняє шина I2C.

8.1. Приклад конфігурації шини I2C

Шина I2C підтримує будь-яку технологію виготовлення мікросхем (nМОП, КМОП, біполярну). Дві лінії, даних (SDA) і синхронізації (SCL), слугують для перенесення інформації. Кожен пристрій розпізнається за унікальною адресою, будь то мікроконтролер, РКІ буфер, пам'ять або інтерфейс клавіатури, і може працювати як передавач або приймач, залежно від призначення пристрою. Зазвичай РКІ буфер – лише приймач, а пам'ять може як приймати, так і передавати дані. Крім того, пристрої можуть бути класифіковані як 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 8.1, приклад конфігурації системи з 2 мікроконтролерами по шині I2C – на рис. 8.1.

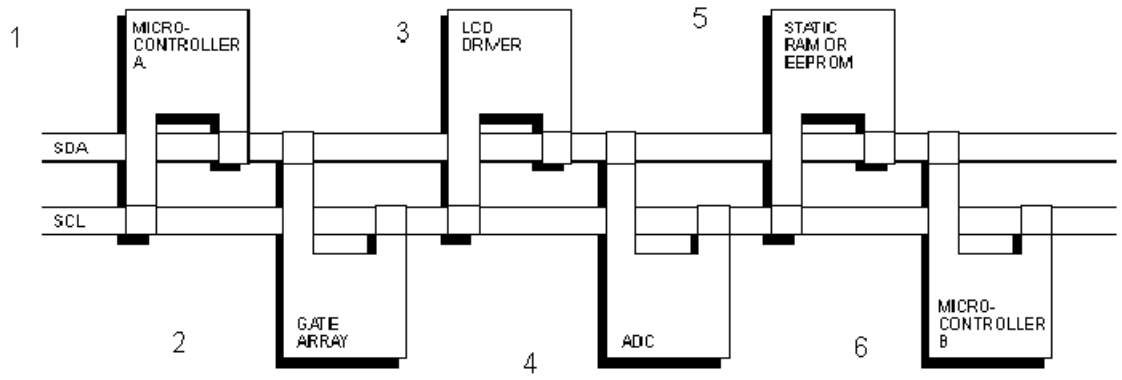


Рис. 8.1. Приклад конфігурації шини I2C з двома мікроконтролерами

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 A та B, підключення до шини (рис. 8.1). Необхідно відмітити, що ці взаємини не

постійні, а залежать лише від напряму пересилки даних в даний момент часу. Пересилка даних відбуватиметься таким чином:

Нехай 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.

Таблиця 8.1

Терміни шини I2C

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

Нехай 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, якшо відбувається зіткнення. Можливість підключення більше одного мікроконтролера 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.

8.2. Підключення I2C-пристроїв до шини

Як SDA, так і SCL є двонапрямленими лініями, приєднаними до додатнього джерела живлення через підтягуючий резистор (рис. 8.2). Коли шина вільна, обидві лінії знаходяться у ВИСОКОМУ положенні. Вихідні каскади пристроїв, підключених до шині, повинні мати відкритий стік або відкритий колектор для забезпечення функції монтажного І.

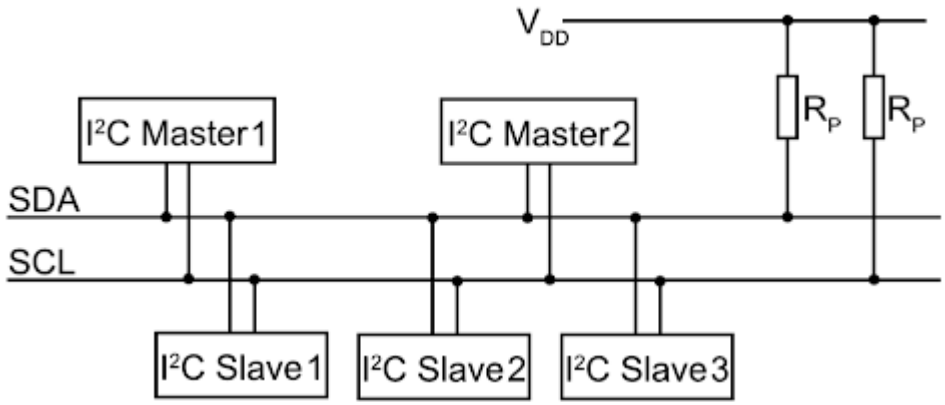


Рис. 8.2. Підключення пристроїв по I2C-шині

8.3. Пересилка біта даних.

Внаслідок різних технологій мікросхем (КМОП, nМОП, біополярна), які можуть бути підключенні 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 повинен змінюватися, тільки якщо лінія синхронізації в стані НИЗЬКИЙ (рис. 8.3).

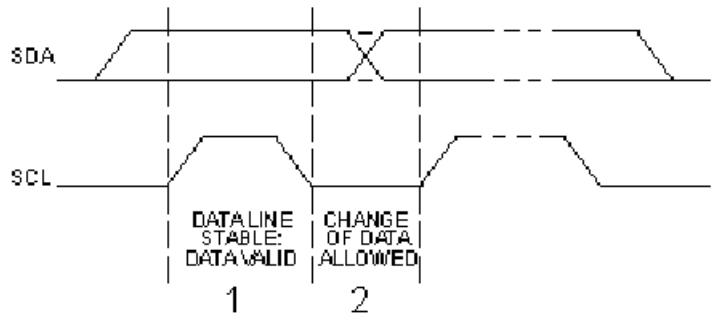


Рис. 8.3. Пересилка біта по шині I2C

8.4. Сигнали START і STOP

Початок та кінець передачі визначають спеціальні ситуації – СТАРТ і СТОП (рис. 8.4).

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 читання значення лінії SDA як мінімум двічі за період синхронізації для того, щоб визначити перехід стану.

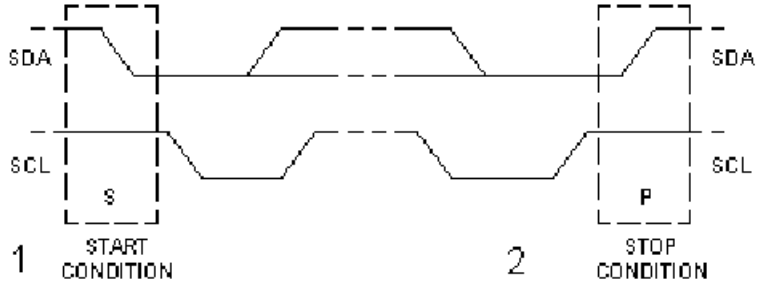


Рис. 8.4. Сигнали СТАРТ і СТРОП

8.5. Формат байта.

Кількість байт, переданих за один сеанс зв’язку по лінії SDA, необмежена. Кожен байт повинен закінчуватися бітом підтвердження. Дані передаються, починаючи з найбільш значущого біта MSB (рис. 8.5). Якщо приймач не може прийняти наступний байт, поки він виконує яку-небудь іншу функцію (наприклад, обслуговує внутрішнє переривання), він може утримувти 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.

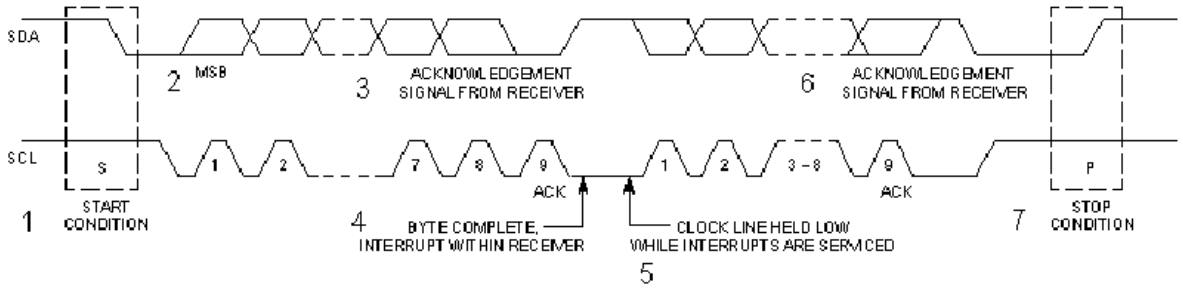


Рис. 8.5. Пересилка даних по шині I2C

1. Сигнал СТАРТ (S);
2. Старший розряд байта (MSB – most significant 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).

8.6. Підтвердження

Підтвердження при передачі даних обов’язкове. 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. 8.6). 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 шляхом не підтвердження останнього байта. Ведений передач повинен звільнити лінію даних для того, щоб дозволити ведучому видати сигнал СТОП або повторити сигнал СТАРТ.

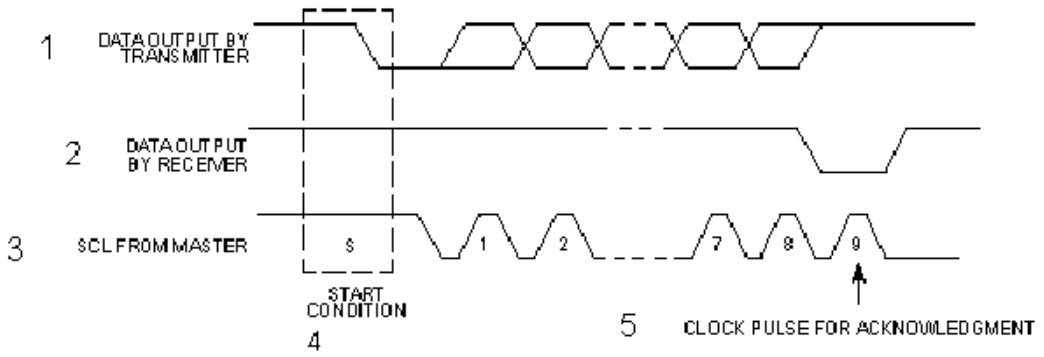


Рис. 8.6. Підтвердження

1. Дані, передані передавачем;

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

8.7. 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 НИЗЬКИЙ з пристроїв в стан, відбувається також перехід одного синхросигналу іншого пристрою в НИЗЬКИЙ стан.

Цей стан ліній SCL утримується доти, доки не буде встановлено ВИСОКИЙ стан внутрішнього синхросигналу одного з пристроїв (рис. 8.7). Проте перехід з НИЗЬКОГО стану у ВИСОКИЙ стан синхросигналу може не викликати аналогічний перехід на лінії SCL, якщо синхросигнал іншого пристрою все ще знаходиться в НИЗЬКОМУ стані. Таким чином, лінія SCL знаходиться 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 закінчиться НИЗЬКИЙ період синхросигналу, лінія SCL перейде у ВИСОКИЙ стан. Всі пристрої почнуть проходити ВИСОКИЙ період своїх синхросигналів. Перший пристрій, у якого закінчиться цей період знову встановить лінію SCL в НИЗЬКИЙ стан.

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

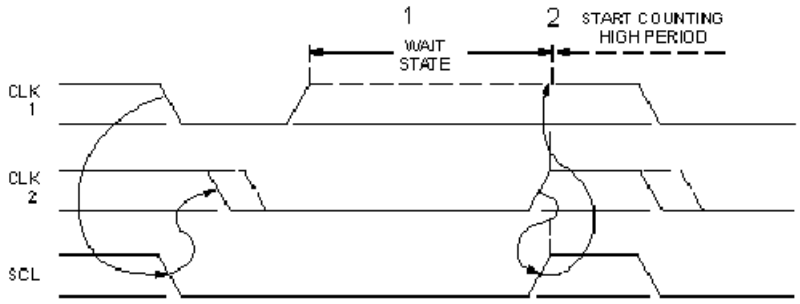


Рис. 8.7. Синхронізація під час арбітражу

1. Стан чекання.
2. Початок відліку ВИСОКОГО періоду синхроімпульса.

Лекція 9. Інтерфейс 1-Wire

9.1. Загальна характеристика

Однопровідний інтерфейс 1-Wire, розроблений в кінці 90-х років фірмою Dallas Semiconductor, регламентований розробниками для вживання в трьох основних сферах застосування:

- 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 линія (другий дріт – заземлення).

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

Групу команд обміну з ЗП складають чотири команди: читання ЗП, пропуск, порівняння і пошук. Дві останні команди забезпечують взаємодію по одній лінії декількох Touch Memory з читаючим пристроєм. Команда порівняння ініціює обмін з приладом за вазаним серійним номером. 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. Ведучий пристрій завжди ініціює обмін, посилаючи команди веденому пристрою. Для забезпечення цілісності передаваної інформації протокол обміну на фізичному рівні строго регламентує часові параметри сигналів на лінї.

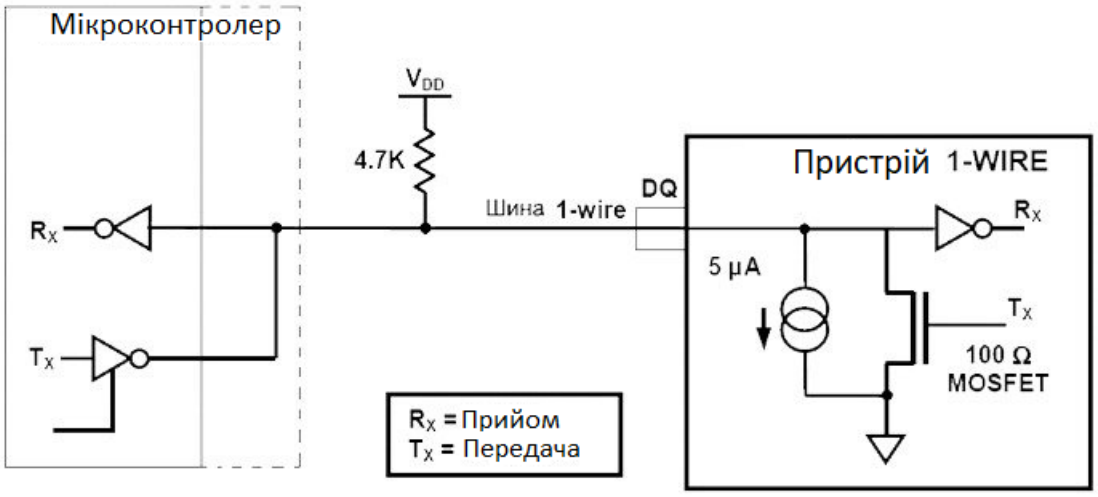


Рис. 9.1. Схема обміну по 1-wire інтерфейсу

На Рис. 9.1 показана спрощена схема апаратної реалізації інтерфейсу 1-Wire. Виводи DQ пристрою є входом КМОП-логічного елементу, який може бути зашунтований (замкнутий на загальний дріт) польовим транзистором. Шина 1-Wire має бути підтянута окремим резистором до напруги живлення пристроїв. Опір цього резистора 4.7 кОм, 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 кОм.

Підключениня шини 1-Wire до МК на рис. 9.1 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 ввід і на вивід.

9.2. Обмін інформацією по шині 1-Wire

- Обмін завжди ведеться за ініціативою одного ведучого пристрою, який в більшості виадків є мікроконтролером (МК). Для інтерфейсу 1-Wire в загальному випадку передбачається "гаряче" підключения і відключення пристроїв. Будь-який обмін інформацією починається з подачі імпульсу скидания ("Reset Pulse" або просто RESET) в лінію 1-Wire ведучім пристроєм.

- Будь-який пристрій, підключений до 1-Wire, після надання живлення видає в лінію DQ імпульс присутності, званий "Presence pulse" (PRESENCE) Цей же імпульс пристрій завжди видає в лінію, якщо виявить сигнал RESET
- Поява в шині 1-Wire імпульсу PRESENCE після видачі RESET однозначно свідчить про наявиість хоч би одного підключеного пристрою.
- Обмін інформацією ведеться так знаними тайм-слотами: один тайм-слот служить для обмну одним бітом інформації.
- 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. 9.2 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. 9.2. Діаграма сигналів RESET і PRESENCE

Як бачимо, тривалість більшості часових інтервалів дуже приблизна і має обмеження лише по мінімуму (не менше вказаного). Умовні позначення ліній, показані на рис. 9.2, 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, залежно від ємності 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.

Після прийому імпульсу RESET, ведений пристрій приводить свої внутрішні вузли у вихідний стан і формує у відповідь імпульс PRESENCE, як випливає з рис. 9.2, 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 пристроєм не раніше, ніж через 480 мікросекунд після завершения імпульсу RESET.

Отже, процедура інцалізації інтерфейсу, з якої починається буль-який обмін даними між пристроями, триває мнімум 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 "ведучого 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 переведения 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-слоти 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. 9.3 демонструє часові діаграми тайм-слотів всіх 4-х типів: вгорі показані тайм-слоти передачі від МК, внизу – прийому від пристрою.

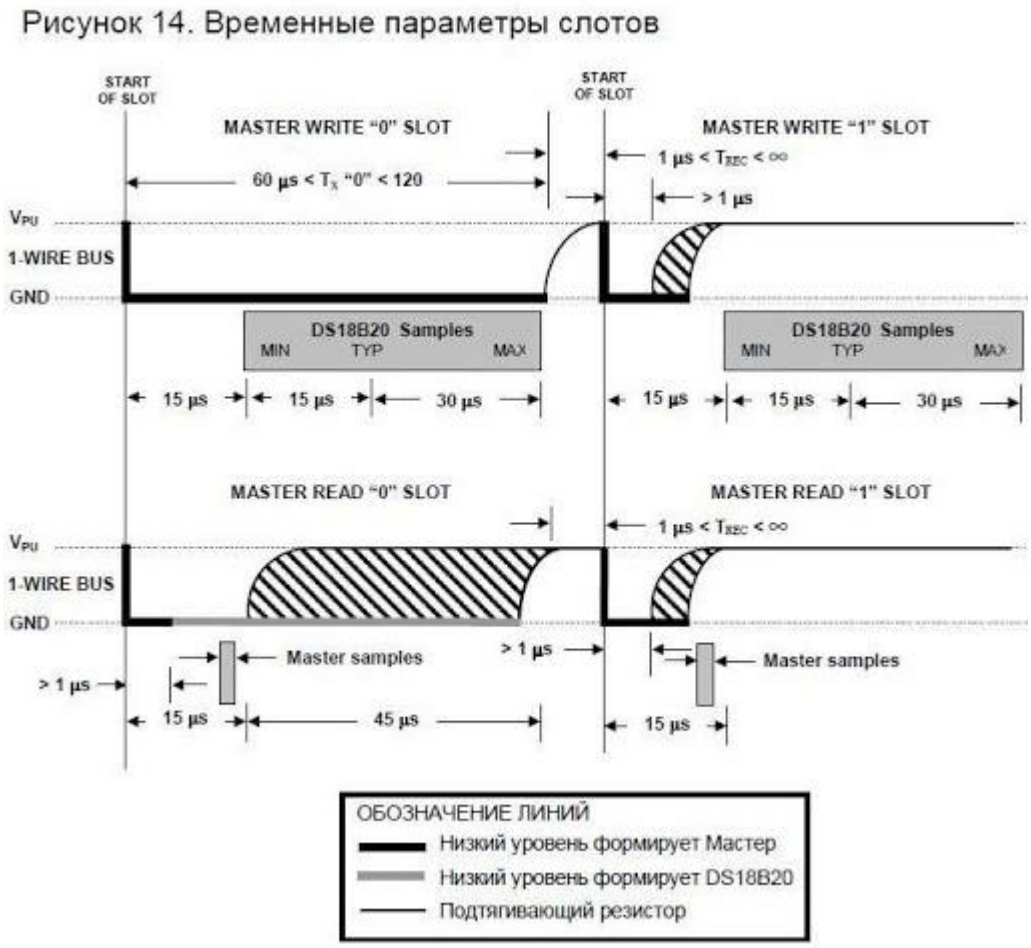


Рис. 9.3. Тайм-слоти

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 ніж 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-ю

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 ємності підтягуючого резистора, вона приведена для довідки.

Тайм-слоти прийому інформаії 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. 9.3, досить мала. Контролер повинен ввести рівень сигналу з лінії на 13-15-й мікросекунді від початку тайм-слота.

Таким чином, МК починає тайм слот видачею 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 опитування рівня в шині він повинен зробити на проміжку від 13-ї до 15-ї мікросекунди тайм-слота. МК повинен забезнечити 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" понад 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 найпершого (тобто орієнтуватися на найгірші варіанти часових параметрів сигналу).

Кожен пристрій 1-Wire має унікальний ідентифікаційний 64-бітовий 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).

9.3. 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, і простим, при якому відомі номери всіх використовуваних у схемі 1-Wire-пристроїв. Номери деяких пристроїв нанесені на корпусі мікросхем, 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 всі наявні пристрої приймають PRESENCE. Потім МК посилає в шину команду, яку приймають всі пристрої. Команд визначено як декілька загальних для всіх типів 1-Wire-пристроїв, так і для окремих типів. Серед загальних команд можуть бути команди, унікальні для для окремих типів. Серед загальних команд поширені наступні (див. таблицю 9.1).

Таблиця 9.1

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 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 встановленню адреси на паралельній шині в мікропроцесорних пристроях. Прийнявши цю команду, кожен пристрій порівнює передану адресу зі своєю власною. Всі пристрої, адреса яких не збіглася, припиняють аналіз і видачу сигналів в лінію 1-Wire, а пристрій, що впізнав адресу, продовжує роботу. Тепер всі дані, передавані МК потраплятимуть лише до цього "адресованого" пристрою. Те, які саме дані треба послати в пристрій або отримати від нього після його адресації, залежить від конкретного пристрою Наприклад, для 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 типу 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 останнім і обчислюється по спеціальному алгоритму на основі значення всіх 7 попередніх байтів. Алгоритм підрахунку такий, що якщо всі байти прийняті без спотворень, прийнятий байт контрольної суми обов'язково співпаде з розрахованим в МК (або пристрої) значенням. Тобто при реалізації програмного алгоритму обміну інформацією необхідно при передачі і прийомі байтів підраховувати їх контрольну суму за певним суворим алгоритмом, а потім або передати розраховане значения, або порівняти розрахункове значення з отриманого значення CRC. Лише при збігу обох CRC МК або пристрій вважають прийняті дані достовірними. Інакше продовження обміну неможливе.

Вочевидь, алгоритм підрахунку CRC має бути однаковим як для МК, так і для будь-якого пристрою. Він стандартизований і описаний в документації, наприклад "Understanding and Using Cyclic Redundancy Checks with Dallas Semiconductor iButtonTM Products".

Таким чином

- будь-який обмін інформацією починається з передачі імпульсу RESET і прийому імульса 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 контрольною сумою попередніх семи;

- якщо пристрій на шині один – МК може дізнатися 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).

Навчально-методичні матеріали

1. Платунов А. Встраиваемые системы управления [Текст] / А. Платунов // Control Ingineering, 2013, № 1 (43).– С. 16-24.
2. Ключев, А.О., Ковязина Д.Р., Кустарев, П.В., Платунов, А.Е. Аппаратные и программные средства встраиваемых систем. Учебное пособие [Текст] / А.О. Ключев, П.В. Кустарев, А.Е. Платунов. – СПб.: СПбГУ ИТМО, 2010. – 290 с.
3. Платунов А.Е, Постников Н.П. Высокоуровневое проектирование встраиваемых систем. – СПб.: НИУ ИТМО, 2011. – 121 с.
4. Орлов С. А., Цилькер Б. Я. Организация ЭВМ и систем: Учебник для вузов. 2-е изд. — СПб.: Питер, 2011. — 688 с.: ил.
5. 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.
6. UART и USART. COM-порт. Часть 1. - режим доступу до ресурсу: http://www.rotr.info/electronics/mcu/arm_usart.htm
7. Последовательный интерфейс SPI (3-wire) - режим доступу до ресурсу: http://www.gaw.ru/html.cgi/txt/interface/spi/index.htm
8. Описание шины I2C - режим доступу до ресурсу: http://www.itt-ltd.com/reference/ref_i2c.html
9. 1-Wire-интерфейс - режим доступу до ресурсу: http://www.elin.ru/1-Wire