

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

ДО ВИКОНАННЯ

РОЗРАХУНКОВО-ГРАФІЧНОГО ЗАВДАННЯ

„Розробка пристрою на базі контролера з архітектурою 8051“

з дисципліни

„МІКРОПРОЦЕСОРНА ТЕХНІКА“

для студентів спеціальності

152 „Метрологія та інформаційно-вимірювальна техніка“

153 „Мікро- та наносистемна техніка“

денної й заочної форм навчання

Методичні вказівки до виконання розрахунково-графічного завдання „Розробка пристрою на базі контролера з архітектурою 8051“ з дисципліни „Мікропроцесорна техніка“ для студентів спеціальності 152 „Метрологія та інформаційно-вимірювальна техніка“ 153 „Мікро- та наносистемна техніка“ денної й заочної форм навчання / Укл.: В.І.Рева. – Запоріжжя: ЗНТУ, 2019. – 63 с.

Укладач:

В.І.Рева, ст. викл

Рецензент:

О.В. Василенко, доц., канд.техн.наук

Відповідальний за випуск:

А.В. Коротун, доц., канд.фіз.-мат.наук

Затверджено
на засіданні кафедри МіНЕ
Протокол №5
від “ 20 “ лютого 2019 р.

на засіданні МНК ФРЕТ
Протокол №5
від “ 21 “ лютого 2019 р.

ЗМІСТ

Передмова.	5
1 Загальні вказівки і рекомендації.	6
1.1 Етапи виконання розрахунково-графічного завдання.	6
1.2 Рекомендації до виконання розрахунково-графічного завдання.	7
1.3 Вимоги до оформлення текстової версії розрахунково-графічного завдання.	8
1.4 Вимоги до змісту та оформлення презентації.	9
1.5 Найбільш поширені помилки студентів при написанні та оформленні розрахунково-графічного завдання.	11
2 Опис елементної бази.	11
2.1 Загальні характеристики мікроконтролера i8051.	11
2.1.1 Структура мікроконтролера.	13
2.1.2 Арифметико-логічний пристрій.	14
2.2 Резидентна пам'ять програм / даних і регістри загального призначення.	15
2.2.1 Пам'ять програм.	16
2.2.2 Регістри спеціальних функцій.	18
2.3 Пристрій управління і синхронізації.	22
2.4 Організація портів введення/виводу мікроконтролера 8051	23
2.4.1 Особливості роботи портів.	25
2.5 Протокол I2C.	27
2.5.1 Стан СТАРТ і СТОП.	28
2.5.2 Підтвердження.	30
2.5.3 Адресація в шині I2C.	31
2.5.4 Розширення I2C.	33
2.6 Годинники реального часу.	36
2.6.1 Характеристики.	36
2.6.2 Опис.	36
2.6.3 Робота.	37
2.6.4 Точність годин.	39
2.6.5 RTC і карта адресації пам'яті.	40
2.6.6 Керуючий регістр.	41
2.6.7 Двохпровідна послідовна шина даних.	42
3 Завдання до розрахунку.	47
3.1 Перелік тем реферативної частини розрахунково-графічних завдань.	47

3.2 Завдання для практичної частини розрахунково-графічних завдань.	49
Рекомендована література.	49
Додаток А. Приклад оформлення титульної сторінки.	51
Додаток Б. Приклад виконання завдання.	52

ПЕРЕДМОВА

Мікропроцесори (МП) і мікропроцесорні системи (МПС) є в даний час найбільш масовими засобами обчислювальної техніки. Після появи перших МП у 1971 р., число нових розробок мікропроцесорних БІС продовжує безупинно збільшуватися.

Досвід розробки різних систем показав, що очікувана від МП універсальність є в значній мірі обмеженою. Розмаїтість мікропроцесорних БІС, що виражається в їх різній базовій технології, архітектурі, технічних характеристиках поставило розроблювачів цифрових систем перед складними проблемами, а саме: які особливості цього класу пристроїв і тенденції його розвитку, чим відрізняється використання могутніх мікро-ЕОМ від застосування міні-ЕОМ, як проектувати системи на основі МП, у чому полягають особливості підходів до розробки програмного забезпечення мікро-ЕОМ і багато інших.

Рішення цих проблем вимагає від розробника систем глибоких знань цифрової обчислювальної техніки (ОТ), включаючи технічні й алгоритмічні питання, представлення особливостей МП БІС, а головне досвіду розробки систем на основі МП.

Курс "Мікропроцесорна техніка" вивчається протягом одного семестру.

В результаті вивчення дисципліни студент повинний знати:

- характеристики і параметри, умовні графічні зображення і принцип дії основних різновидів мікропроцесорів і мікроконтролерів;
- принципи побудови мікро-ЕОМ на основі різних мікропроцесорних комплектів;
- методику аналізу і вибору оптимального технічного рішення;
- системи команд мікропроцесорів і правила складання програмного забезпечення РЕА на основі мікропроцесорних систем;
- сучасний стан мікропроцесорної і комп'ютерної техніки і перспективи їхнього розвитку.

1 ЗАГАЛЬНІ ВКАЗІВКИ І РЕКОМЕНДАЦІЇ

Розрахунково-графічне завдання (РГЗ) – це самостійне дослідження студента. Виконуючи РГЗ, студент удосконалює знання і вміння, отримані в процесі вивчення дисципліни „Мікропроцесорна Техніка“, а саме: визначати мету, виділяти задачі, формулювати проблеми і знаходити шляхи їх вирішення. Працюючи над РГЗ, студент отримує вміння і навички, які будуть корисні в майбутньому – при виконанні більш складних задач (курсовий проект, дипломна робота, дисертація, наукове дослідження).

Написання розрахунково-графічної роботи є одним із видів самостійної роботи з дисципліни „Мікропроцесорна Техніка“.

Метою написання РГЗ є:

- ❖ систематизація, закріплення і розширення теоретичних знань та практичних вмінь студентів;
- ❖ набуття досвіду роботи з літературою та іншими джерелами інформації, вміння узагальнювати і аналізувати наукову інформацію, формувати власне ставлення до проблеми;
- ❖ набуття вміння застосовувати інформаційні та комп'ютерні технології для вирішення прикладних задач;
- ❖ розвиток навичок оволодіння спеціалізованим програмним забезпеченням;
- ❖ проведення глибокого аналізу результатів досліджень і формування змістовних висновків щодо якості отриманих результатів.

1.1 Етапи виконання розрахунково-графічного завдання

Розрахунково-графічне завдання виконується в два етапи відповідно до кредитно-модульної структури дисципліни.

Перший етап РГЗ виконується впродовж першого модуля і включає:

- ❖ вибір теми РГЗ;
- ❖ визначення актуальності та мети роботи;
- ❖ підбір літератури і джерел інформації згідно з обраною темою;
- ❖ збір даних, необхідних для розкриття обраної теми;
- ❖ систематизацію і структурування даних;

❖ оформлення результату виконання першого етапу в текстовому редакторі.

Результат виконання першого етапу надається в роздрукованому вигляді.

Другий етап РГЗ виконується впродовж другого модулю і включає:

- ❖ вибір методу обробки інформації;
- ❖ обґрунтування і опис обраного методу;
- ❖ обробку інформації;
- ❖ отримання результатів;
- ❖ інтерпретацію результатів;
- ❖ формулювання висновків;
- ❖ оформлення звіту;
- ❖ підготовка до публічного захисту виконаної роботи.

Звіт про виконання РГЗ оформлюється в електронному і друкованому вигляді. До звіту додаються електронні файли з результатами обробки інформації. Для захисту РГЗ створюється електронна презентація.

1.2 Рекомендації до виконання розрахунково-графічного завдання

Тема РГЗ обирається студентом відповідно до його варіанту.

Актуальність проблеми висвітлюється стисло. Достатньо декількома реченнями висловити головне – суть проблеми.

Мета РГЗ полягає в тому, щоб проаналізувати певну проблему. У ній сконцентрована головна ідея роботи, її кінцевий пізнавальний і практичний результат. Мета має носити практично-прикладний характер. Мета розкриває те, задля чого здійснюється дослідження, і повинна описувати конкретні кроки, які необхідно здійснити студенту для отримання результатів. Мета дослідження конкретизується в задачах, сукупність яких дає уявлення про те, що потрібно зробити для її досягнення. Задач не повинно бути багато, як правило, 4–5. Задачі формулюються дієсловами у формі переліку дій: „вивчити...“, „проаналізувати...“, „встановити...“, „побудувати модель...“, „з’ясувати...“, „оцінити...“, „ідентифікувати...“, „порівняти...“, „обґрунтувати...“, „визначити чинники...“ тощо.

Анотація до кожного джерела інформації має бути обсягом не менше 10 друкованих рядків.

У кінці формулюються висновки. Вони по логіці і кількісно мають співпадати з поставленими в меті задачами, тобто одна задача – *один висновок* (!). Кожний з висновків повинний „перекликатися“ із задачами роботи. Отже, висновки повинні ґрунтуватися на проведенню студентом власному аналізі проблеми і відповідати задачам РГЗ.

Розрахунково-пояснювальна записка до РГЗ має містити:

- титульний лист (друкується у вигляді бланку (див. Додаток А);
- реферат обсягом 0,5–1 стор. (включає інформацію про обсяг РГЗ, ключові слова, короткий зміст роботи і методики проведення дослідження);

- зміст;

- вступ обсягом 1–2 стор. (відображає інформацію про актуальність дослідження, мету і задачі РГЗ);

- основний розділ, який включає детальний розв’язок задачі та фізичний аналіз отриманих результатів;

- перелік використаної літератури;

- додатки (за необхідності).

Розв’язок задачі має бути викладений зрозуміло і детально, з проведенням усіх проміжних викладок та перетворень. Основні формули повинні бути пронумеровані. У додатки включають допоміжний матеріал, який, на думку автора, необхідний для розуміння ходу розв’язку, але загромаджує основний текст роботи.

Слід мати на увазі, що оформлення РГЗ впливає на оцінку.

1.3 Вимоги до оформлення текстової версії розрахунково-графічного завдання

При оформленні текстової версії розрахунково-графічного завдання слід дотримуватися таких вимог:

- формат сторінок А4;

- шрифт Times New Roman;

- розмір шрифту 14;

- міжрядковий інтервал – полуторний;

- вирівнювання – за шириною сторінки;

- поля – 2 см;

- назви структурних частин і розділів – прописними літерами, напівжирним шрифтом, по центру, підрозділів – малими (крім першої букви), напівжирним шрифтом, по центру;
- відстань між заголовком та основним текстом – один рядок;
- кожен структурну частину (розділ) починати з нової сторінки;
- нумерація сторінок зверху, праворуч;
- номер на титульній сторінці, рефераті та змісті не ставиться, але враховується;
- таблиці повинні мати заголовки та номеруватися;
- рисунки, графіки і діаграми повинні мати підписи та номеруватися.

1.4 Вимоги до змісту та оформлення презентації

Вимоги до змісту та оформлення презентації наступні:

- презентація виконується в форматі Microsoft Power Point XP/2003 за темою РГЗ;
- презентація має бути розрахована на п'ятихвилинну доповідь за темою РГЗ;
- зміст презентації має бути чітко структурований. Кожний новий слайд має логічно впливати з попереднього й одночасно готувати появу наступного. Найкращий спосіб перевірити, чи вірно побудована презентація, – швидко прочитати лише заголовки. Якщо після цього стане зрозуміло, про що презентація – структура побудована вірно;
- оптимальним об'ємом презентації вважається не менше, ніж 15 традиційних слайдів;
- кількість тексту в презентаціях має складати не більше 35%, решта – ілюстративний матеріал (рисунки, графіки, діаграми, схеми тощо);
- слайд №1 повинний містити:
 - 1) назву навчального закладу і кафедру, де виконувалася робота (розмір шрифту – не менше 24);
 - 2) дисципліну, тему (розмір шрифту – не менше 28, напівжирний);
 - 3) прізвище, ім'я, по-батькові студента, факультет, групу (розмір шрифту – не менше 24);

- 4) прізвище, ім'я, по-батькові викладача (розмір шрифту – не менше 24);
 - 5) рік створення презентації (розмір шрифту – не менше 24);
- слайди №2 – 3 повинні описувати актуальність теми і мету, яку необхідно досягти в ході виконання роботи (загальний обсяг одного слайду – не більше 15 рядків тексту);
 - останній слайд (слайди) повинний містити висновки по роботі (не більше 15 рядків тексту на один слайд);
 - усі слайди (крім першого) повинні містити порядковий номер, розташований в правому верхньому куті (розмір шрифту – не менше 20);
 - кожен слайд (крім першого) повинний мати назву, набрану шрифтом не менше 24;
 - оптимальне оформлення презентації – застосування кольорових схем „світлий текст на темному фоні“ або „темний текст на світлому фоні“. У випадку співпадіння кольорів фону і тексту рекомендується оформлювати текст із використанням ефекту „тіні“;
 - фонові рисунки не повинні відволікати увагу від змісту слайда;
 - припустимий розмір шрифту – не менше 20;
 - рекомендований розмір шрифту – 24;
 - максимальна кількість текстової інформації на одному слайді – 15 рядків тексту, набраних Arial або Verdana, розмір – 28;
 - максимальна кількість графічної інформації на одному слайді – 2 рисунка (фотографії, схеми тощо) з текстовими коментарями (не більше 2-х рядків до кожного);
 - бажано, щоб на слайдах залишалися поля, не менше 1 см з кожного боку;
 - використовувати єдиний стиль оформлення;
 - використовувати шрифти без засічок (їх легше читати) і не більше 2-х варіантів шрифту на одному слайді;
 - використовувати не більше трьох кольорів на одному слайді (наприклад, один для фону, другий для заголовків, третій для тексту);
 - використовувати ефекти анімації, звукові та відеофайли лише у разі, коли це виправдане і дійсно необхідне для покращення сприйняття матеріалу;

– файл презентації має бути записаний на CD-ROM або DVD-ROM.

1.5 Найбільш поширені помилки студентів при написанні та оформленні розрахунково-графічного завдання

- зміст роботи не відповідає темі або не розкриває її;
- мета не пов'язана з проблемою, що досліджується, сформульована абстрактно і не відображає специфіки предмету дослідження;
- не проведений глибокий та всебічний аналіз джерел, нової спеціальної літератури за темою дослідження;
- список джерел за темою роботи не анотований; не вказано, з яких саме джерел взяті вихідні дані;
- кінцевий результат не відповідає меті дослідження, висновки не відповідають поставленим задачам;
- об'єм і оформлення роботи не відповідають вимогам;
- РГЗ виконана неохайно, з граматичними й стилістичними помилками.

2 ОПИС ЕЛЕМЕНТНОЇ БАЗИ

2.1 Загальні характеристики мікроконтролера i8051

Класичний мікроконтролер i8051 (MCS51) і вітчизняний аналог КМ1816BE51 виконані на основі високорівневої n-МОН технології і випускалися в корпусі БІС, що має 40 зовнішніх висновків. Цоколювання корпусу MCS51 і найменування висновків показані на рис. 1. Для роботи MCS51 потрібно одне джерело електроживлення +5. Через чотири програмованих порту введення / виведення MCS51 взаємодіє із середовищем в стандарті TTL-схем з трьома станами виходу.

Корпус MCS51 має два висновки для підключення кварцового резонатора, чотири виводу для сигналів, керуючих режимом роботи МК, і вісім ліній порту 3, які можуть бути запрограмовані користувачем на виконання спеціалізованих (альтернативних) функцій обміну інформацією з середовищем. Призначення висновків мікроконтролера 8051.

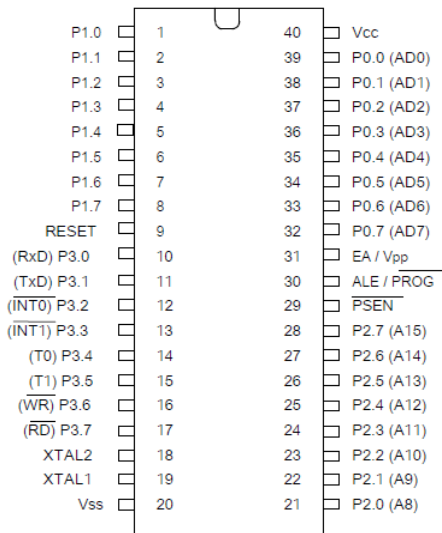


Рисунок 2.1. – Призначення виводів 8051

Позначення на Рис. 2.1:

- ❖ Vss – потенціал загального проводу ("землі");
- ❖ Vcc – основна напруга літання +5 В;
- ❖ XTAL1, XTAL2 - висновки для підключення кварцового резонатора;
- ❖ RESET (RST) – вхід загального скидання мікроконтролера;
- ❖ PSEN – дозвіл зовнішньої пам'яті програм; видається тільки при зверненні до зовнішнього ПЗУ;
- ❖ ALE – строб адреси зовнішньої пам'яті;
- ❖ EA – відключення внутрішньої програмної пам'яті; рівень 0 на цьому вході змушує мікроконтролер виконувати програму тільки з зовнішнього ПЗУ; ігноруючи внутрішнє (якщо останнє є);
- ❖ P0 – восьми бітний двонаправлений порт введення-виведення інформації: при роботі з зовнішніми ОЗУ і ПЗУ по лінії порту в режимі тимчасового мультиплексування видається адреса зовнішньої пам'яті, після чого здійснюється передача або прийом даних;

- ❖ P1 – восьми бітний квазі двонаправлений порт введення / виводу: кожен розряд порту може бути запрограмований як на введення, так і на висновок інформації, незалежно від стану інших розрядів;
- ❖ P2 – восьми бітний квазі двонаправлений порт, аналогічний P1; крім того, висновки цього порту використовуються для видачі адресної інформації при зверненні до зовнішньої пам'яті програм або даних (якщо використовується 16-бітова адресація останньої).
- ❖ P3 – восьми бітний квазі двонаправлений порт, аналогічний P1; крім того, висновки цього порту можуть виконувати ряд альтернативних функцій, які використовуються при роботі таймерів, порту послідовного введення-виведення, контролера переривань, і зовнішньої пам'яті програм і даних.

2.1.1 Структура мікроконтролера

Основу структурної схеми MCS51 (рис. 2.2) утворює внутрішня двонаправлена 8-бітна шина, яка пов'язує між собою всі основні вузли і пристрої: резидентну пам'ять програм (RPM), резидентную пам'ять даних (RDM), арифметико-логічне пристрій (ALU), блок регістрів спеціальних функцій, пристрій управління (CU) і порти введення / виводу (P0-P3).

Розглянемо основні елементи структури і особливості організації обчислювального процесу в MCS51.

Найпростіша операція додавання використовується в ALU для інкрементирования вмісту регістрів, просування регістра-показчика даних (RAR) і автоматичного обчислення наступної адреси резидентної пам'яті програм. Найпростіша операція віднімання використовується в ALU для декрементирования регістрів і порівняння змінних.

Найпростіші операції автоматично утворюють "тандеми" для виконання таких операцій, як, наприклад, інкрементирование 16-бітних реєстрових пар. У ALU реалізується механізм каскадного виконання найпростіших операцій для реалізації складних команд. Так, наприклад, при виконанні однієї з команд умовної передачі управління по результату порівняння в ALU тричі інкрементується лічильник команд (PC), двічі проводиться читання з RDM, виконується арифметичне порівняння двох змінних, формується 16-бітову адресу переходу і приймається рішення про те, робити чи не робити перехід за програмою. Всі перераховані операції виконуються лише за 2 мкс.

Важливою особливістю ALU є його здатність оперувати не тільки байтами, але і битами. Окремі програмно-доступні біти можуть бути встановлені, скинуті, інвертовані, передані, перевірені та використані в логічних операціях. Ця здатність досить важлива, оскільки для управління об'єктами часто застосовуються алгоритми, що містять операції над вхідними та вихідними булевими змінними, реалізація яких засобами звичайних мікропроцесорів пов'язана з певними труднощами.

Таким чином, ALU може оперувати чотирма типами інформаційних об'єктів: булевими (1 біт), цифровими (4 біта), байтними (8 біт) і адресними (16 біт). У ALU виконується 51 різна операція пересилання або перетворення цих даних. Так як використовується 11 режимів адресації (7 для даних і 4 для адрес), то шляхом комбінування операцій і режиму адресації базове число команд 111 розширюється до 255 з 256 можливих при однобайтному коді операції.

2.2 Резидентна пам'ять програм / даних і регістри загального призначення

Резидентні (розміщені на кристалі) пам'ять програм (RPM) і пам'ять даних (RDM) фізично і логічно розділені, мають різні механізми адресації, працюють під управлінням різних сигналів і виконують різні функції.

2.2.1 Пам'ять програм.

Пам'ять програм RPM має ємність 4 Кбайта і призначена для зберігання команд, констант, керуючих слів ініціалізації, таблиць перекодування вхідних і вихідних змінних і т.п. Пам'ять має 16-бітну шину адреси.

При зверненні до зовнішньої пам'яті програм (EPM) всі мікроконтролери сімейства 8051 завжди використовують 16-разрядний адреса, що забезпечує їм доступ до 64 Кбайт ПЗУ. Мікроконтролер звертається до програмної пам'яті при читанні коду операції та операндів (використовуючи лічильник команд PC), а також при виконанні команд перенесення байта з пам'яті програм в акумулятор. При виконанні команд перенесення даних адресація комірки пам'яті програм, з якої будуть прочитані дані, може здійснюватися з використанням як лічильника PC, так і спеціального двухбайтового реєстра-показчика даних DPTR.

Пам'ять даних і реєстри загального призначення. Пам'ять даних RDM призначена для зберігання змінних в процесі виконання прикладної програми, адресується одним байтом і має ємність 128 байт. Крім того, до її адресного простору примикають адреси реєстрів спеціальних функцій, які перераховані в таблиці 1.2.

Пам'ять даних, так само як і пам'ять програм, може бути розширена до 64 Кбайт шляхом підключення зовнішніх мікросхем. Перші 32 байта організовані в чотири банки реєстрів загального призначення (РОН), позначаються відповідно банк 0 - банк 3 (див. таблицю 1.1). Кожен з них складається з восьми реєстрів R0 - R7. У будь-який момент програмі доступний тільки один банк реєстрів, номер якого міститься в третьому і четвертому бітах слова стану програми PSW (див. Нижче).

Час, що залишився адресний простір може конфігуруватися розробником на свій розсуд: в ньому розташовуються стек, системні і призначені для користувача області даних. Звернення до осередків пам'яті даних можливо двома способами. Перший спосіб - пряма адресація комірки пам'яті. У цьому випадку адреса осередку є операндом відповідної команди. Другий спосіб - непряма адресація з допомогою реєстрів R0 або R1: перед виконанням 17 відповідної команди в один з них повинен бути занесений адреса комірки, до якої необхідно звернутися. Для звернення до зовнішньої пам'яті даних (EDM) використовується тільки непряма адресація з допомогою реєстрів R0 і R1 або за

допомогою 16-розрядного регістра-показчика DPTR. Він належить до групи регістрів спеціальних функцій, і з його допомогою можна адресувати всі 64 Кбайта зовнішньої пам'яті. Частина пам'яті даних являє собою так звану бітову область, в ній є можливість за допомогою спеціальних бітових команд адресовуватися до кожного розряду осередків пам'яті. Адреса прямо адресованих бітів може бути записаний або у вигляді (адреса Байта). (Розряд), наприклад вираз 21.3 означає третій розряд комірки пам'яті з адресою 21H, або у вигляді абсолютного бітового адреси. Відповідність цих двох способів адресації можна визначити по таблиці.

Таблиця 2.1 – Адреси бітових областей пам'яті мікроконтролера 8051 і регістрів загального призначення.

[illegible]

Примітка. Адреса прямо адресованих бітів може бути записаний або у вигляді виразу (адреса Байта). (Розряд), наприклад вираз 21.3 означає адресу третього розряду комірки пам'яті з адресою 21H, або у вигляді абсолютного бітового адреси, який для даного біта дорівнює (див. Таблицю 1.1) 0B.

2.2.2 Регістри спеціальних функцій

До адресного простору пам'яті даних примикає адресний простір регістрів спеціальних функцій SFR (Special Function Register).

Таблиця 2.2 – Розміщення регістрів спеціальних функцій в просторі SFR Адреса Символ

Адрес	Символ	Найменування
0E0H	*ACC	Акумулятор (Accumulator)
0F0H	*B	Регистр расширитель аккумулятора (Multiplication Register)
0D0H	*PSW	Слово состояния программы (Program Status Word)
080H	*P0	Порт 0 (SFR P0)
090H	*P1	Порт 1 (SFR P1)
0A0H	*P2	Порт 2 (SFR P2)
0B0H	*P3	Порт 3 (SFR P3)
081H	SP	Регистр указатель стека (Stack Pointer)
083H	DPH	Старший байт регистра указателя данных DPTR (Data Pointer High)
082H	DPL	Младший байт регистра указателя данных DPTR (Data Pointer Low)
08CH	TH0	Старший байт таймера 0 ()
08AH	TL0	Младший байт таймера 0 ()
08DH	TH1	Старший байт таймера 1 ()
08BH	TL1	Младший байт таймера 1 ()
089H	TMOD	Регистр режимов таймеров счетчиков (Timer/Counter Mode Control Register)
088H	*TCON	Регистр управления статуса таймеров (Timer/Counter Control Register)
0B8H	*IP	Регистр приоритетов (Interrupt Priority Control Register)
0A8H	*IE	Регистр маски прерывания (Interrupt Enable Register)
087H	PCON	Регистр управления мощностью (Power Control Register)
098H	*SCON	Регистр управления приемопередатчиком (Serial Port Control Register)
099H	SBUF	Буфер приемопередатчика (Serial Data Buffer)

Примітка. Регістри, символ яких відзначений знаком (*), допускають адресацію своїх окремих біт при використанні команд з групи команд операцій над бітами. Адреси, за якими розташовані ці регістри, наведені в таблиці 2.3.

Таблиця 2.3 – Карта адресованих бітів в блоці регістрів спеціальних функцій

Адреса байту	Адреси бітів за розрядами								Ім'я регістру
Adr	D7	D6	D5	D4	D3	D2	D1	D0	Name
F0H	F7	F6	F5	F4	F3	F2	F1	F0	B
...	...								...
E0H	E7	E6	E5	E4	E3	E2	E1	E0	ACC
...	...								...
D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW
...	...								...
B8H	-	-	-	BC	BB	BA	B9	B8	IP
...	...								...
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
...	...								...
A8H	AF	-	-	AC	AB	AA	A9	A8	IE
...	...								...
A0H	A7	A6	A5	A4	A3	A2	A1	A0	P2
...	...								...
98H	9F	9E	9D	9C	9B	9A	99	98	SCON
...	...								...
90H	97	96	95	94	93	92	91	90	P1
...	...								...
88H	8F	8E	8D	8C	8B	8A	89	88	TCON
...	...								...
80H	87	86	85	84	83	82	81	80	P0

Примітка. Адреса прямо адресованих бітів може бути записаний або у вигляді виразу (Назва Регістру). (Розряд), наприклад вираз SCON.3 означає адресу третього розряду регістра SCON, або у вигляді абсолютного бітового адреси, який для даного біта дорівнює (див. Таблицю 2.3) 9B. Крім того, деякі біти регістрів мають власні назви, так наприклад даний біт має назву TB8. Відзначимо, що регістри займають лише частину 128-байтового адресного простору. Тобто осередки пам'яті з адресами 80H-0FFH, які не зайняті регістрами, фізично відсутні, на кристалах 20 мікроконтролерів сімейства 8051 при зверненні до них можна прочитати лише код команди повернення.

- ❖ Регістри спеціальних функцій керують роботою блоків, що входять в мікроконтролер.
- ❖ Регістри-засувки SFR паралельних портів P0 ... P3 - служать для введення-виведення інформації.

- ❖ Дві реєстрові пари з іменами TH0, TL0 і TH1, TL1 являють собою регістри, двох програмно-керованих 16-бітових таймерів-лічильників.
- ❖ Режим таймерів-лічильників задаються з використанням регістра TMOD, а управління ними здійснюється за допомогою регістра TCON.
- ❖ Для управління режимами енергоспоживання мікро-ЕОМ використовується
- ❖ регістр PCON.
- ❖ регістри IP і IE керують роботою системи переривань мікро-ЕОМ,
- ❖ регістри SBUF і SCON - роботою приймача послідовного порту.
- ❖ Регістр-показчик стека SP в мікро-ЕОМ розглянутого сімейства - восьми бітний. Він може адресувати будь-яку область внутрішньої пам'яті даних. На відміну від мікропроцесора KP580BM80, у мікро-ЕОМ сімейства 8051 стек «росте вгору», тобто перед виконанням команди PUSH або CALL вміст SP інкрементується, після чого проводиться запис інформації в стек. Відповідно при добуванні інформації з стека регістр SP декрементується після вилучення інформації. В процесі ініціалізації мікро-ЕОМ після сигналу скидання або при включенні напруги живлення в SP заноситься код 07H. Це означає, що перший елемент стека буде розташовуватися в комірці пам'яті з адресою 08H.
- ❖ Регістр-показчик даних DPTR найчастіше використовують для фіксації 16-бітної адреси в операціях звернення до зовнішньої пам'яті програм і даних. З точки зору програміста він може виступати як у вигляді одного 16-бітного регістра, так і у вигляді двох незалежних регістрів DPL і DPH.
- ❖ акумулятор (ACC) є джерелом операнда і місцем фіксації результату при виконанні арифметичних, логічних операцій і ряду операцій передачі даних. Крім того, тільки з використанням акумулятора можуть бути виконані операції зрушень, перевірка на нуль, формування прапора паритету і т.п. У розпорядженні користувача є 8 регістрів загального призначення R0-R7 одного з чотирьох можливих банків. При виконанні багатьох команд в АЛУ формується ряд ознак операції (прапорів), які фіксуються в регістрі PSW.

- ❖ Регістр В використовується як джерело і як приймач при операціях множення і ділення, звернення до нього, як до регістру SFR, виробляється аналогічно акумулятору.
- ❖ При виконанні ряду команд в арифметико-логічному пристрої (АЛП) формуються ознаки операцій - прапори, які фіксуються в регістрі PSW. У таблиці 4 наводиться перелік прапорів PSW, даються їх символічні імена і описуються умови їх формування.

Таблиця 2.4 – Формат слова стану програми PSW.

Символ	Розряд	Ім'я та призначення																				
C	PSW.7	Флаг приоритета. Устанавливается и сбрасывается аппаратно в каждом цикле команды и фиксирует нечетное/четное число единичных бит в аккумуляторе																				
AC	PSW.6	Не используется																				
F0	PSW.5	Флаг переполнения. Устанавливается и сбрасывается аппаратно при выполнении арифметических операций																				
RS1 RS0	PSW.4 PSW.3	Биты выбора используемого банка регистров. Могут быть изменены программным путем <table border="1"><tr><td>RS0</td><td>RS1</td><td>Банк</td><td>Границы адресов ОЗУ</td></tr><tr><td>0</td><td>0</td><td>0</td><td>00H - 07H</td></tr><tr><td>1</td><td>0</td><td>1</td><td>08H - 0FH</td></tr><tr><td>0</td><td>1</td><td>2</td><td>10H - 17H</td></tr><tr><td>1</td><td>1</td><td>3</td><td>18H - 1FH</td></tr></table>	RS0	RS1	Банк	Границы адресов ОЗУ	0	0	0	00H - 07H	1	0	1	08H - 0FH	0	1	2	10H - 17H	1	1	3	18H - 1FH
RS0	RS1	Банк	Границы адресов ОЗУ																			
0	0	0	00H - 07H																			
1	0	1	08H - 0FH																			
0	1	2	10H - 17H																			
1	1	3	18H - 1FH																			
OV	PSW.2	Флаг пользователя. Может быть установлен, сброшен или проверен программой пользователя																				
-	PSW.1	Флаг вспомогательного переноса. Устанавливается и сбрасывается только аппаратными средствами при выполнении команд сложения и вычитания и сигнализирует о переносе или заеме в бите 3 аккумулятора																				
P	PSW.0	Флаг переноса. Устанавливается и сбрасывается как аппаратно, так и программным путем																				

Найбільш "активним" прапором PSW є прапор переносу, який бере участь і модифікується в процесі виконання безлічі операцій, включаючи додавання, віднімання ізрушення. Крім того, прапор переносу (C) виконує функції "булева акумулятора" в командах, що маніпулюють з битами. Прапор переповнення (OV) фіксує арифметичне переповнення при операціях над цілими числами зі знаком і робить можливим використання арифметики в додаткових кодах. ALU не керує прапорами селекції банку регістрів (RS0, RS1), їх значення повністю визначається прикладною програмою і використовується для вибору одного з чотирьох реєстрових банків.

У мікропроцесорах, архітектура яких спирається на акумулятор, більшість команд працюють з ним, використовуючи неявну адресацію. В Intel 8051 інша справа. Хоча процесор має в своїй основі акумулятор, він може виконувати безліч команд і без його участі. Наприклад, дані можуть бути передані з будь-якого елементу RDM в будь-який регістр, будь-який регістр може бути завантажений безпосереднім операндом і т.д. Багато логічні операції можуть бути виконані без участі акумулятора. Крім того, змінні можуть бути інкрементіровані, декрементіровані і перевірені без використання акумулятора. Пропори та керуючі біти можуть бути перевірені і змінені аналогічно.

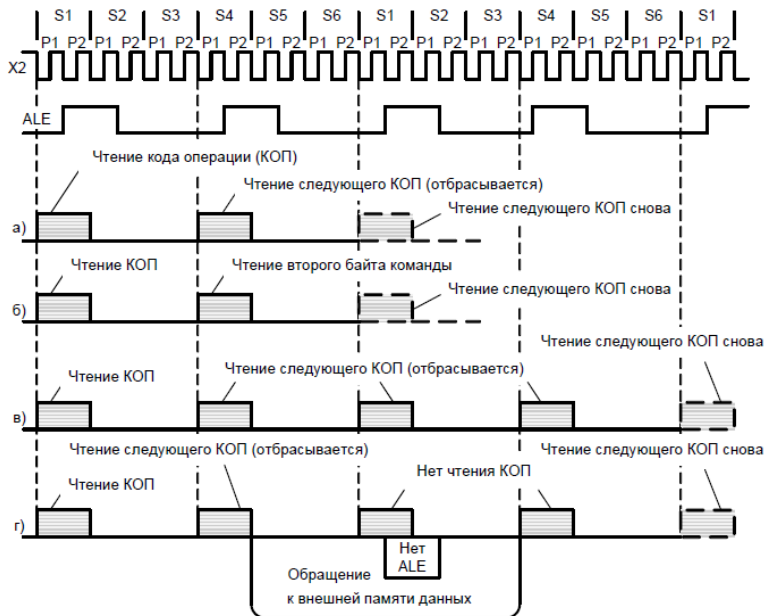
2.3 Пристрій управління і синхронізації

Кварцовий резонатор, що підключається до зовнішніх висновків мікроконтролера, керує роботою внутрішнього генератора, який в свою чергу формує сигнали синхронізації.

Пристрій управління (CU) на основі сигналів синхронізації формує машинний цикл фіксованої тривалості, рівної 12 періодам резонатора або шести станів первинного керуючого автомата (S1 - S6). Кожне стан керуючого автомата містить дві фази (P1, P2) сигналів резонатора. У фазі P1, як правило, виконується операція в АЛП, а в фазі P2 здійснюється міжрегістрових передача. Весь машинний цикл складається з 12 фаз, 23 починаючи з фази S1P1 і закінчуючи фазою S6P2, як показано на рис. 3. Ця тимчасова діаграма ілюструє роботу пристрою управління MCS51 при вибірці і виконанні команд різного ступеня складності. Все заштриховані сигнали є внутрішніми і недоступні користувачеві MCS51 для контролю. Зовнішніми, які спостерігаються сигналами є тільки сигнали резонатора і стрибає адреси зовнішньої пам'яті (ALE). Як видно з тимчасової діаграми, сигнал ALE формується двічі за один машинний цикл (S1P2 - S2P1 і S4P2 - S5P1) і використовується для управління процесом звернення до зовнішньої пам'яті [1, 4,5].

Більшість команд мікроконтролера виконується за один машинний цикл. Деякі команди, які оперують з 2-байтними словами або пов'язані зі зверненням до зовнішньої пам'яті, виконуються за два машинних циклу. Тільки команди ділення і множення вимагають чотирьох машинних циклів. На основі цих особливостей роботи пристрою управління, проводиться розрахунок часу виконання прикладних про-

грам.



а - команда 1 байт / 1 цикл, наприклад INC A; б - команда 2 байта / 1 цикл, наприклад ADD A, # d; в - команда 1 байт / 2 циклу, наприклад INC DPTR; г - команда 1 байт / 2 циклу, наприклад MOVX

Рисунок 2.3 – Послідовності вибірки і виконання команд в MCS51

2.4 Організація портів введення/виводу мікроконтролера 8051

Всі чотири порти (P0-P3) призначені для введення або виведення інформації побайтно.

Кожен з портів містить регістр-засувку (SFR P0 - SFR P3), вхідний буфер і вихідний драйвер. Кожен з розрядів регістра-засувки SFR є D-тригером, інформація в який заноситься з внутрішньої шини даних мікроконтролера за сигналом «Запис в SFR Px» (x = 0, 1, 2, 3) від центрального процесорного елемента (CPU). З прямого виходу D-тригера інформація може бути виведена на внутрішню шину по сигналу «Читання SFR Px» від CPU, а з виведення мікросхеми («із зовніш-

нього світу») за сигналом «Читання висновків P_x». Одні команди активізують сигнал «Читання SFR P₁», інші - «Читання висновків P₁». Вихідні драйвери портів 0 і 2, а також вхідний буфер порту 0 використовуються при зверненні до зовнішньої пам'яті (ВП). При цьому через порт 0 в режимі тимчасового мультиплексування спочатку виводиться молодший байт адреси ВП, а потім видається або приймається байт даних. Через порт 2 виводиться старший байт адреси в тих випадках, коли розрядність адреси дорівнює 16 біт.

Всі висновки порту 3 можуть бути використані для реалізації альтернативних функцій, перерахованих в таблиці 6. Ці функції можуть бути задіяні шляхом запису 1 у відповідні біти регістра-засувки (P3.0-P3.7) порту 3.

Порт 0 є двонаправленим, а порти 1-3 - квазідвонаправлені. Кожна лінія портів може бути використана незалежно для введення або виведення.

За сигналом RST в регістри-засувки всіх портів автоматично записуються одиниці, які налаштовують їх тим самим на режим введення.

Всі порти можуть бути використані для організації введення / виведення інформації по двонаправленим лініях передачі. Однак порти 0 і 2 не можуть бути використані для цієї мети в разі, якщо система має зовнішню пам'ять, зв'язок з якою організовується через загальну поділювану шину адреси / даних, що працює в режимі тимчасового мультиплексування.

Особливості електричних характеристик портів

Вихідні каскади тригерів SFR портів P₁ - P₃ виконані на польових транзисторах з внутрішньою навантаженням, в той час як аналогічні каскади тригерів SFR P₀ – на транзисторах з відкритим стоком. Кожна лінія будь-якого з портів може незалежно використовуватися як для введення, так і для виведення інформації (для ліній портів P₀ і P₂ це справедливо тоді, коли вони не використовуються для звернення до зовнішньої пам'яті).

Для перекладу будь-якої лінії портів P₁ - P₃ в режим введення інформації необхідно в відповідний розряд SFR занести 1. При цьому вихідний польовий транзистор відключається. Внутрішній навантажувальний резистор як би «підтягує» потенціал виведення до напруги харчування, в той час як зовнішнє навантаження може зробити його нульовим. Вихідні каскади порту P₀ мають іншу структуру. Наванта-

жувальний польовий транзистор лінії порту включений тільки тоді, коли порт виводити при зверненні до зовнішньої пам'яті. В інших випадках навантажувальний транзистор відключений. Таким чином, при роботі в режимі звичайного введення-виведення інформації (як, наприклад, порт P1) вихідні каскади порту P0 є ступені на транзисторах з відкритим стоком. Запис 1 в відповідний біт SFR відключає і другий транзистор, що призводить до того, що висновок БІС виявляється під «плаваючим» потенціалом.

Оскільки вихідні каскади портів P1 – P3 мають внутрішню навантаження, при перекладі в режим введення інформації вони стають джерелами струму для мікросхеми або транзистора, навантажених на даний висновок.

Таблиця 2.5 – Альтернативні функції виводів порту P3

Символ	Розряд	Ім'я та призначення
RD	P3.7	Чтение. Активный сигнал низкого уровня формируется аппаратно при обращении к внешней памяти данных
WR	P3.6	Запись. Активный сигнал низкого уровня формируется аппаратно при обращении к внешней памяти данных
T1	P3.5	Вход таймера/счётчика 1 или тест-вход
T0	P3.4	Вход таймера/счётчика 0 или тест-вход
INT1	P3.3	Вход запроса прерывания 1. Воспринимается сигнал низкого уровня или срез
INT0	P3.2	Вход запроса прерывания 0. Воспринимается сигнал низкого уровня или срез
TXR	P3.1	Выход передатчика последовательного порта в режиме UART. Выход синхронизации в режиме регистра сдвига
RXD	P3.0	Вход приёмника последовательного порта в режиме UART. Ввод/вывод данных в режиме регистра сдвига

2.4.1 Особливості роботи портів

Звернення до портів введення/виводу можливо з використанням команд, що оперують з байтом, окремим бітом, довільною комбінацією бітів. При цьому в тих випадках, коли порт є одночасно операндом і місцем призначення результату, пристрій управління автоматично реалізує спеціальний режим, який DC T P3.X Лінія внутр. шини Запис в SFR + Усс читання SFR читання виведення & Альтернативний сигнал виходу Альтернативний вхідний сигнал.

Навантаження називається "читання-модифікація-запис". Цей режим звертання припускає введення сигналів не з зовнішніх виснов-

ків порту, а з його регістра-засувки, що дозволяє виключити неправильне зчитування раніше виведеної інформації [1]. Цей механізм звернення до портів реалізований в командах:

- ❖ ANL - логічне І, наприклад, ANL P1, A;
- ❖ ORL - логічне АБО, наприклад, ORL P2, A;
- ❖ XRL - виключає АБО, наприклад, XRL P3, A;
- ❖ JBC - перехід, якщо в адресується біте одиниця, і подальший скидання біта, наприклад, JBC P1.1, LABEL;
- ❖ CPL - інверсія біта, наприклад, CPL P3.3;
- ❖ INC - інкремент порту, наприклад, INC P2;
- ❖ DEC - декремент порту, наприклад, DEC P2;
- ❖ DJNZ - декремент порту і перехід, якщо його вміст не дорівнює нулю, на-приклад, DJNZ r, LABEL;
- ❖ MOV PX.Y, C - передача біта перенесення в біт X порту Y;
- ❖ SET PX.Y - установка біта X порту Y;
- ❖ CLR PX.Y - скидання біта X порту Y.

Зовсім не очевидно, що останні три команди в наведеному списку є командами "читання-модифікація-запис". Однак це саме так. За цими командами спочатку зчитується байт з порту, а потім записується новий байт в регістр-засувку [1].

Причиною, через яку команди "читання-модифікація-запис" забезпечують роздільне доступ до регістру-клямці порту і до зовнішніх висновків порту, є необхідність виключити можливість неправильного прочитання рівнів сигналів на зовнішніх висновках.

Припустимо для прикладу, що лінія X порту Y з'єднується з базою потужного транзистора і вихідний сигнал на ній призначений для його управління. Коли в даний біт записана 1, то транзистор включається. Якщо для перевірки стану виконавчого механізму (в нашому випадку – потужного транзистора) прикладної програми потрібно прочитати стан вихідного сигналу в тому ж біте порту, то зчитування сигналу (наприклад, командою MOV ACC, PY) з зовнішнього виведення порту, а не з D-тригера регістра-засувки порту призведе до неправильного результату: одиничний сигнал на базі транзистора має відносно низький рівень і буде інтерпретований в МК як сигнал 0. Команди "читання-модифікація-запис" реалізують зчитування з регістру-засувки, а не з зовні на виведення порту

2.5 Протокол I2C

I2C шина є однією з модифікацій послідовних протоколів обміну даних. У стандартному режимі забезпечується передача послідовних 8-бітних даних зі швидкістю до 100 кбіт/с, і до 400 кбіт/с в "швидкому" режимі. Для здійснення процесу обміну інформацією по I2C шині, використовується всього два сигнали лінія даних SDA лінія синхронізації SCL. Для забезпечення реалізації двобічної шини без застосування складних арбітрів шини вихідні каскади пристроїв, підключених до шини, мають відкритий стік або відкритий колектор для забезпечення функції монтажного "И".

Проста двохпровідна послідовна шина I2C мінімізує кількість сполуки між ІС, ІС мають менше контактів і потрібно менше доріжок. Як результат – друковані плати стають простішими і технологічними при виготовленні. Інтегрований I2C-протокол усуває необхідність в дешифратор адреси та іншої зовнішньої логіці узгодження.

Максимальна допустима кількість мікросхем, приєднаних до однієї шини, обмежується максимальним обсягом до шини 400 пФ.

Вбудований в мікросхеми апаратний алгоритм помехоподавлення забезпечує цілісність даних в умовах перешкод значної величини.

Усе I2C-сумісні пристрої мають інтерфейс, який дозволяє їм зв'язуватися один з одним по шині навіть у тому випадку, якщо їх напруга живлення істотно відрізняється. На наступному малюнку представлений принцип підключення декількох ІМС з різними напругами живлення до однієї шини обміну.

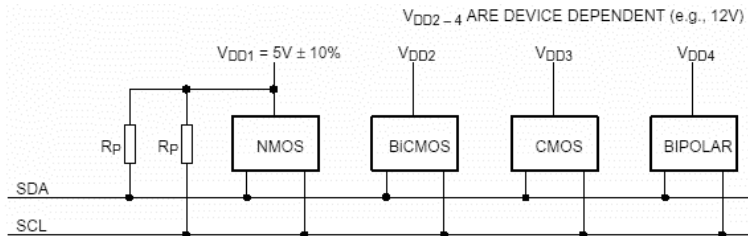


Рисунок 2.4 – Приклад підключення пристроїв до I2C шини

Кожен пристрій розпізнається за унікальною адресою і може працювати як передавач або приймач, в залежності від призначення пристрою.

Крім того, пристрої можуть бути класифіковані як провідні і ведені при передачі даних. Ведучий – це пристрій, який ініціює передачу даних і виробляє сигнали синхронізації. При цьому будь-який пристрій, що адресується вважається відомим по відношенню до ведучого.

Виходячи з специфікації роботи шини, в кожен окремий момент в шині може бути тільки один ведучий, а саме той пристрій, що забезпечує формування сигналу SCL шини. Ведучий може виступати як в ролі ведучого-передавача, так і ведучого-приймача. Проте – шина дозволяє мати кілька ведучих, накладаючи певні особливості їхньої поведінки в формуванні сигналів управління і контролю стану шини. Можливість підключення більше одного ведучого до шини означає, що більш ніж один ведучий може спробувати почати пересилання в один і той же момент часу. Для усунення "зіткнень", який може виникнути в даному випадку, розроблена процедура арбітражу – поведінки ведучого при виявленні "захоплення" шини іншим провідним.

Процедура синхронізації двох пристроїв. Ця процедура заснована на тому, що всі I2C- пристрої підключаються до шини за правилом монтажного И. У початковому стані обидва сигналу SDA і SCL знаходяться у високому стані.

2.5.1 Стан СТАРТ і СТОП

Процедура обміну починається з того, що ведучий формує стан СТАРТ – провідний генерує перехід сигналу лінії SDA з ВИСОКОГО стану в НИЗЬКИЙ при ВИСОКОМУ рівні на лінії SCL. Цей перехід сприймається усіма пристроями, підключеними до шини як ознака початку процедури обміну.

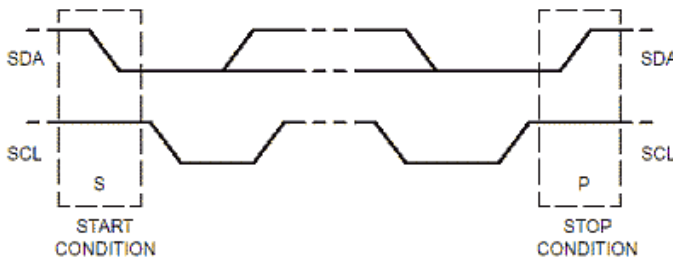


Рисунок 2.5 – Часова діаграма сигналів СТАРТ і СТОП I2C шини

Генерація синхросигналу – це завжди обов'язок ведучого; кожен провідний генерує свій власний сигнал синхронізації при пересиланні даних по шині.

Процедура обміну завершується тим, що ведучий формує стан СТОП – перехід стану лінії SDA з низького стану в ВИСОКУ при ВИСОКОМУ стані лінії SCL.

Стану СТАРТ і СТОП завжди виробляються провідним. Вважається, що шина зайнята після фіксації стану СТАРТ. Шина вважається звільнилася через деякий час після фіксації стану СТОП.

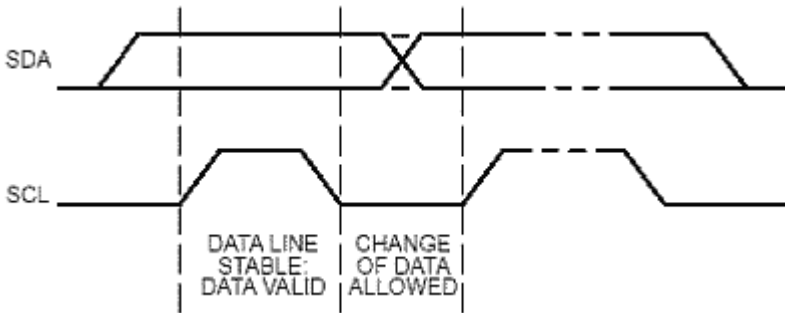


Рисунок 2.6 – Часова діаграма сигналів у момент передачі даних I2C шини

При передачі посилки по шині I2C кожен провідний генерує свій синхросигнал на лінії SCL.

Після формування стану СТАРТ, провідний опускає стан лінії SCL в НИЗЬКИЙ стан і виставляє на лінію SDA старший біт першого байта повідомлення. Кількість байт в повідомленні не обмежена.

Специфікація шини I2C дозволяє зміни на лінії SDA тільки при НИЗЬКОМУ рівні сигналу на лінії SCL.

Дані дійсні і повинні залишатися стабільними тільки під час ВИСОКОГО стану синхроімпульса.

Для підтвердження прийому байта від ведучого – передавача веденим – приймачем в специфікації протоколу обміну по шині I2C вводиться спеціальний біт підтвердження, який виставляється на шину SDA після прийому 8 біта даних.

2.5.2 Підтвердження

Таким чином передача 8 біт даних від передавача до приймача завершуються додатковим циклом (формуванням 9-го тактового імпульсу лінії SCL), при якому приймач виставляє низький рівень сигналу на лінії SDA, як ознака успішного прийому байта.

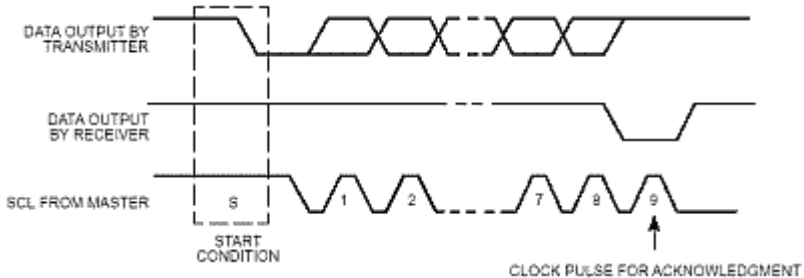


Рисунок 2.7 – Часова діаграма сигналів підтвердження передачі даних I2C шини

Підтвердження при передачі даних обов'язково. Відповідний імпульс синхронізації генерується провідним. Передавач відпускає (ВИСОКА) лінію SDA на час синхроімпульса підтвердження. Приймач повинен утримувати лінію SDA протягом ВИСОКОГО стану синхроімпульса підтвердження в стабільному НИЗЬКОМУ стані.

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

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

Синхронізація виконується з використанням підключення до лінії SCL за правилом монтажного И.

Це означає, що ведучий не має монопольного права на управління переходом лінії SCL з НИЗЬКОЇ стану ВИСОКОГО. У тому випадку, коли відомому необхідний додатковий час на обробку прийнятого

біта, він має можливість утримувати лінію SCL в низькому стані до моменту готовності до прийому наступного біта. Таким чином, лінія SCL буде перебувати в НИЗЬКОМУ стані протягом найдовшого НИЗЬКОЇ періоду синхросигналов.

Пристрої з більш коротким низькими періодом будуть входити в стан очікування на час, поки не скінчиться довгий період. Коли у всіх задіяних пристроїв скінчиться НИЗЬКИЙ період синхросигналу, лінія SCL перейде в ВИСОКУ стан. Всі пристрої почнуть проходити ВИСОКИЙ період своїх синхросигналов. Перший пристрій, у якого скінчиться цей період, знову встановить лінію SCL в НИЗЬКИЙ стан. Таким чином, НИЗЬКИЙ період синхролінії SCL визначається наїдлиннішим періодом синхронізації з усіх задіяних пристроїв, а ВИСОКИЙ період визначається найкоротшим періодом синхронізації пристроїв.

Механізм синхронізації може бути використаний приймачами як засіб управління пересиланням даних на байтовому і бітовому рівнях.

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

На рівні бітів, пристрій таке як мікроконтролер без вбудованих апаратних ланцюгів I2C або з обмеженими ланцюгами може уповільнити частоту синхроімпульсів шляхом продовження їх НИЗЬКОЇ періоду. Таким чином швидкість передачі будь-якого провідного адаптується до швидкості повільного пристрою.

2.5.3 Адресація в шині I2C

Кожен пристрій, підключений до шини, може бути програмно адресовано по унікальному адресу.

Для вибору приймача повідомлення провідний використовує унікальний адресну компоненту в форматі посилки. При використанні однотипних пристроїв, IC часто мають додатковий селектор адреси, який може бути реалізований як у вигляді додаткових цифрових входів селектора адреси, так і у вигляді аналогового входу. При цьому адреси таких однотипних пристроїв виявляються рознесені в адресному просторі пристроїв, підключених до шини.

У звичайному режимі використовується 7-бітова адресація.

Процедура адресації на шині I2C полягає в тому, що перший байт після сигналу СТАРТ визначає, який ведений адресується провідним для проведення циклу обміну. Виняток становить адреса "Загального виклику", який адресує всі пристрої на шині. Коли використовується ця адреса, всі пристрої в теорії повинні послати сигнал підтвердження. Однак, пристрої можуть обробляти "загальний виклик" на практиці зустрічаються рідко.

Перші сім бітів першого байта утворюють адреса веденого. Восьмий, молодший біт, визначає напрямок пересилки даних. "Нуль" означає, що провідний буде записувати інформацію в обраного веденого. "Одиниця" означає, що провідний буде зчитувати інформацію з веденого.

Після того, як адресу посланий, кожен пристрій в системі порівнює перші сім біт після сигналу СТАРТ зі своєю адресою. При збігу пристрій вважає себе обраним як ведений-приймач або як ведений-передавач, в залежності від біта напрямки.

Адреса веденого може складатися з фіксованою і програмованої частини.

Часто трапляється, що в системі буде кілька однотипних пристроїв (наприклад ІМС пам'яті, або драйверів LED-індикаторів), тому за допомогою програмованої частини адреси стає можливим підключити до шини максимально можливу кількість таких пристроїв. Кількість програмованих біт в адресі залежить від кількості вільних висновків мікросхеми. Іноді використовується один висновок з аналогової установкою програмованого діапазону адрес, як це, наприклад, реалізовано в ІМС SAA1064. При цьому в залежності від потенціалу на цьому адресному виведенні ІМС, можливий зсув адресного простору драйвера так, щоб однотипні ІМС не конфліктували між собою на загальній шині.

Всі ІМС, що підтримують роботу в стандарті шини I2C, мають набір фіксованих адрес, перелік яких зазначений виробником в описах контролерів.

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

У загальному вигляді процес обміну по шині від моменту формування стану СТАРТ до стану СТОП можна проілюструвати наступним рисунком:

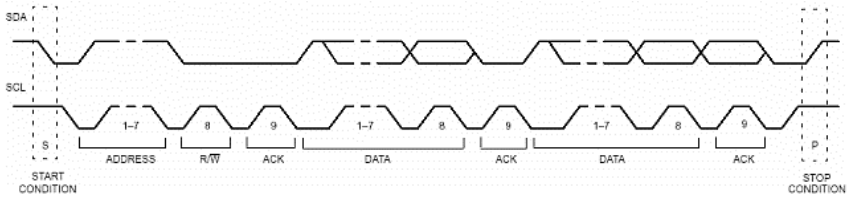


Рисунок 2.8 – Часова діаграма передачі даних I2C шини

Як впливає з специфікації шини, допускаються як прості формати обміну, так і комбіновані, коли в проміжку від стану СТАРТ до стану СТОП провідний і ведений можуть виступати і як приймач і як передавач даних. Комбіновані формати можуть бути використані, наприклад, для управління послідовної пам'яттю.

Під час першого байта даних можна передавати адресу в пам'яті, який записується у внутрішній регістр-засувку. Після повторення сигналу старту і адреси веденого видаються дані з пам'яті. Всі рішення про авто-інкремент або декремент адреси, до якого стався попередній доступ, приймаються конструктором конкретного пристрою. Тому, в будь-якому випадку кращий спосіб уникнути неконтрольованої ситуації на шині перед використанням нової (або раніше яка не використовується) ІМС слід ретельно вивчити її опис (datasheet), отримавши його з сайту виробника. Більш того, виробники часто розміщують поруч більш докладні інструкції по застосуванню.

У будь-якому випадку, за специфікацією шини всі розробляються пристрої повинні скидати логіку шини при отриманні сигналу СТАРТ або повторний СТАРТ і готуватися до прийому адреси.

Проте основні проблеми з використанням I2C шини виникають саме через те, що розробники, "початківці" працювати з I2C шиною не враховують того факту, що ведучий (часто – мікропроцесор) не має монопольного права ні на одну з ліній шини.

2.5.4 Розширення I2C

Стандартна шина I2C зі швидкістю передачі даних 100 кбіт/с і 7-бітовим адресою існує вже протягом більше 10 років в незмінному вигляді. Стандартна шина I2C прийнята повсюдно як стандарт для сотень типів мікросхем, що випускаються фірмою Philips і іншими пос-

тачальниками. В даний час специфікація шини I2C розширена в двох напрямках: збільшення швидкодії і розширення адресного простору для розширення номенклатури розроблених нових пристроїв.

Введення специфікації "швидкого" режиму, що дозволяє в чотирі рази збільшити швидкість передачі даних до 400кБіт/сек. Необхідність в цьому "розширенні" стандарту треба було через необхідність пересилання великих обсягів інформації, і, як наслідок, необхідність збільшення пропускну здатності каналу.

Введення специфікації "10-бітної адресації", що дозволяє використовувати 1024 додаткових адрес, тому що більшість з 112 адрес, допустимих при 7-бітної адресації, вже були використані більш ніж один раз. Для запобігання проблем з розміщенням адрес нових пристроїв, бажано мати більшу кількість адресних комбінацій. Приблизно десятикратне збільшення кількості доступних адрес отримано при використанні нової 10-бітної адресації.

Всі нові пристрої з I2C інтерфейсом працюють в швидкому режимі. Переважно, вони повинні вміти приймати і/або передавати дані на швидкості 400 кБіт/с. Як мінімум вони повинні бути здатні входити в синхронізацію в швидкому режимі, з тим щоб знизити швидкість передачі (шляхом подовження НИЗЬКОЇ періоду SCL) до допустимої величини.

Швидкі пристрої як правило сумісні знизу-вгору, що означає їх здатність працювати зі стандартними пристроями по повільній шині. Очевидно, що стандартні пристрої не здатні працювати в швидкій шині, тому що вони не можуть синхронізуватися на високій швидкості і їх стан стане непередбачуваним. Ведені швидкі пристрої можуть мати як 7-бітовим, так і 10-бітовим адресою. Однак, 7-бітову адресу більш кращий, так як його апаратна реалізація більш проста і довжина послілки менше. Пристрої з 7-бітовим і 10-бітовим адресами можуть одночасно використовуватися на одній шині, незалежно від швидкості передачі. Як існуючі, так і майбутні провідні зможуть генерувати і 7-бітні, і 10-бітні адреси.

У швидкому режимі протокол, формат, логічні рівні і максимальна місткість навантаження ліній шини залишається незмінними. Алгоритм синхронізації ліній SDA і SCL не змінений. Однак, від "швидких" пристрій може потребувати сумісності з CBUS-пристроями, так як вони не можуть працювати на високих швидкостях. Вхідні ланцюги швидких пристроїв повинні мати вбудоване придушення викидів і

тригер Шмітта на обох лініях Вихідний буфер швидких пристроїв повинен мати каскад з керуванням часом заднього фронту ліній SDA і SCL. Як правило при зникненні напруги живлення швидких пристроїв висновки, підключені до ліній I2C шини повинні переходити в третій стан. Зазнали змін схемотехнічні рішення вихідних каскадів для забезпечення часу наростання переднього фронту (перехід з НИЗЬКОЇ стану в ВИСОКУ). Якщо для навантажень шини до 200 пФ цю роль виконують підтягує резистори, то для навантажень від 200 пФ до 400 пФ цю функцію виконує джерело струму або схема на перемикаються резисторах, що забезпечує "форсоване" перемикання ліній I2C шини.

10-бітна адресація також не змінює формат шини. Для цього використовується зарезервована адресна комбінація 1111XXX перших семи біт першого байта. 10-бітна адресація не впливає на існуючу 7-бітну адресацію. Пристрої з 7-бітної і 10-бітної адресацією можуть бути підключені до однієї шини. Хоча є вісім можливих комбінацій послідовності 1111XXX, з них використовуються тільки чотири – 11110XX. Комбінації типу 11111XX зарезервовані для подальших поліпшень шини. Призначення бітів перших двох байтів 10-бітову адресу формується з перших двох байтів. Перші сім біт першого байта є комбінацією виду 11110XX, де два молодших біта (XX) є двома старшими (9 і 8) бітами 10-бітного адреси; восьмий біт першого байта – біт напрямку.

"Нуль" в тому бите означає, що провідний збирається записувати інформацію в веденого, а "одиниця" – що провідний буде зчитувати інформацію з веденого. Якщо біт напрямку дорівнює "нулю", то другий байт містить залишилися 8 біт 10-бітного адреси. Якщо біт напрямку дорівнює "одиниці", то наступний байт містить дані, передані з веденого ведучому.

На закінчення слід зазначити, що стандарт I2C шини досить просто реалізує арбітраж зіткнень – вирішує проблему одночасної ініціалізації обміну по шині декількома провідними, при цьому без втра-ти даних.

2.6 Годинники реального часу

2.6.1 Характеристики

- годинник реального часу (RTC) відраховують секунди, хвилини, години, дату місяця, місяць, день тижня і рік з компенсацією високосного року, дійсної до 2100 року;
- 56- байтове енергонезалежне ОЗУ з живленням від батареї для зберігання призначених для користувача даних;
- двохпровідний послідовний інтерфейс;
- програмований вихідний сигнал з прямокутними імпульсами (для тактирування зовнішніх пристроїв);
- автоматичне виявлення падіння напруга і схема перемикання на батарею;
- споживання менш 500 нА в режимі батарейної підтримки при працюючому тактовом генераторі;
- промисловий діапазон температур: від - 40 °C до + 85 °C;
- мікросхема проводиться в 8- вивідних корпусах DIP і SOIC.

Призначення висновків

- V_{CC} – первинне джерело живлення;
- X1, X2 – підключення 32.768 кГц кварцового резонатора;
- V_{BAT} – вхід батареї +3 В;
- GND – загальний мінус;
- SDA – послідовні дані;
- SCL – послідовні синхроімпульсів;
- SQW/OUT – вихідний сигнал з прямокутними імпульсами.

2.6.2 Опис

Годинник реального часу з послідовним інтерфейсом DS1307 – це малоспоживають повні двійковий-десяткові годинник-календар, що включають 56 байтів енергонезалежної статичної ОЗУ. Адреси та дані передаються послідовно по двохпровідний двобічної шині. Годинник-календар відраховують секунди, хвилини, години, день, дату, місяць і рік. Остання дата місяця автоматично коригується для місяців з кількістю днів менше 31, включаючи корекцію високосного року. Годинники працюють як в 24- годинному, так і в 12- годинному режимах з

індикатором АМ/PM. DS1307 має вбудовану схему спостереження за харчуванням, яка виявляє перебої харчування і автоматично перемикається на живлення від батареї.

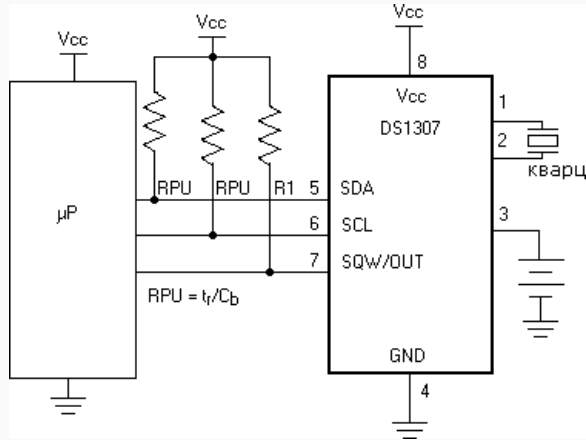


Рисунок 2.9 – Типова схема включення

2.6.3 Робота

DS1307 на послідовній шині працює як ведений пристрій. Доступ до нього досягається установкою умови START і передачею пристрою ідентифікаційного коду, за яким слід адреса регістра. До наступних за ним регістрів доступ здійснюється послідовно, поки не буде виконана умова STOP.

Якщо V_{CC} падає нижче $1.25 * V_{BAT}$, DS1307 перериває процес доступу і скидає лічильник адреси, причому в цей час зовнішні сигнали не сприймаються (щоб запобігти запис помилкових даних).

Якщо V_{CC} падає нижче V_{BAT} , DS1307 перемикається в низькоточних режимі батареї підтримки.

При включенні живлення DS1307 перемикається від батареї до V_{CC} , коли значення V_{CC} перевищує $V_{BAT} + 0.2 \text{ В}$. Вхідні сигнали починають сприйматися тоді, коли V_{CC} перевищує $1.25 * V_{BAT}$.

Блок-схема на **Рис. 1** показує основні елементи RTC з послідовним інтерфейсом.

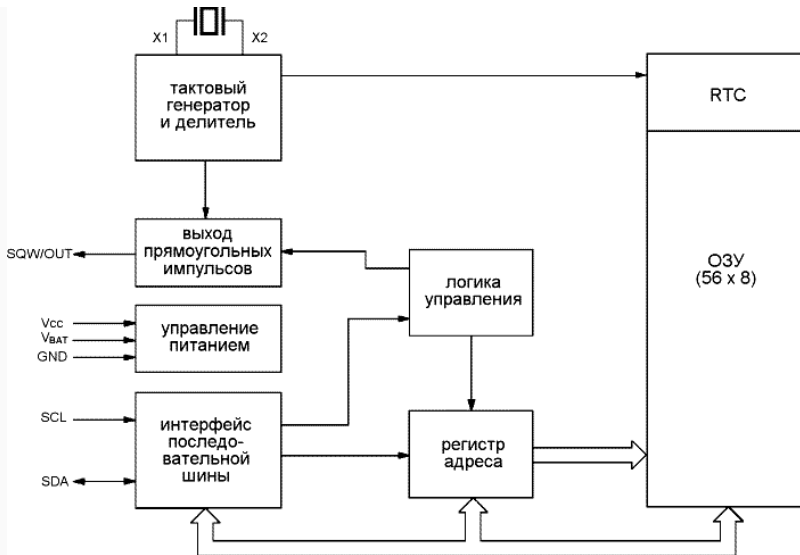


Рисунок 2.10 – Блок-схема DS1307

Опис висновків

V_{CC}, GND – на ці висновки підключений до джерела живлення.

V_{CC} – це вхід +5 В. Коли напруга живлення вище $1.25 * V_{BAT}$, пристрій повністю доступний, і можна виконувати читання і запис даних. Коли до пристрою приєднано батарея на 3 В, і **V_{CC}** нижче, ніж $1.25 * V_{BAT}$, читання і запис заборонені, проте функція відліку часу продовжує працювати. Як тільки **V_{CC}** падає нижче **V_{BAT}**, ОЗУ і RTC переключаються на батарейне живлення **V_{BAT}**.

V_{BAT} – вхід для будь-якої стандартної трьохвольтової літієвої батареї або іншого джерела енергії. Для нормальної роботи DS1307 необхідно, щоб напруга батареї було в діапазоні 2.0 ... 3.5 В. Літієва батарея з ємністю 48 мА/ч або більше при відсутності живлення буде підтримувати DS1307 протягом більш 10 років при температурі 25°C.

SCL (Serial Clock Input – вхід послідовних синхроімпульсів) – використовується для синхронізації даних по послідовному інтерфейсу.

SDA (Serial Data Input/Output – вхід/вихід послідовних даних) – висновок входу/виходу для двохпроводового послідовного інтерфейсу.

су. Висновок SDA – з відкритим стоком і вимагає зовнішнього підтягує резистора.

SQW/OUT (Square Wave/Output Driver – сигнал з прямокутними імпульсами) – коли включений, тобто біт SQWE встановлений в 1, висновок SQW/OUT видає прямокутні імпульси з однієї з чотирьох частот (1 Гц, 4 кГц, 8 кГц, 32 кГц). Висновок SQW/OUT – з відкритим стоком і вимагає зовнішнього підтягує резистора. SQW/OUT буде працювати як при живленні від V_{CC} , так і при живленні від V_{BAT} .

X1, X2 – висновки для стандартного кварцового резонатора з частотою 32.768 кГц. Схема внутрішнього тактового генератора розроблена для роботи з кварцовим резонатором, що має вхідну ємність 12.5 пФ.

Більш детальну інформацію щодо вибору та розташуванню кварцового резонатора ви зможете знайти в прикладі застосування **№ 58** "Розгляд кварцового резонатора для годин реального часу від Dallas" (Application Note 58, "Crystal Considerations with Dallas Real-Time Clocks"). DS1307 також може тактіроваться зовнішнім тактовим генератором з частотою 32.768 кГц. У цьому випадку висновок X1 підключається до сигналу зовнішнього тактового генератора, а висновок X2 залишається непідключеним.

2.6.4 Точність годин

Точність годин залежить від точності кварцового резонатора і точності відповідності між *ємнісний навантаженням схеми тактового генератора і внутрішньої ємністю кварцового резонатора*. Додаткова похибка буде вноситися дрейфом частоти кварцового резонатора, що відбувається через температурних перепадів. Перешкоди і шум зовнішньої схеми можуть привести до прискорення синхронізації.

Більш детальну інформацію дивіться в прикладі застосування **№ 58** "Розгляд кварцового резонатора для годин реального часу від Dallas" (Application Note 58, "Crystal Considerations with Dallas Real-Time Clocks"). Додаткову інформацію ви можете подивитися в прикладі застосування **№ 95** "Сполучення DS1307 з 8051 - сумісним мікроконтролером" (Application Note 95, "Interfacing the DS1307 with a 8051-Compatible Microcontroller").

2.6.5 RTC і карта адресації пам'яті

Карта адрес для RTC і регістрів ОЗУ представлена на **Рис. 2**. Регістри RTC розташовані в осередках адрес від 00h до 07h. Регістри ОЗУ розташовані в осередках адрес від 08h до 3Fh. В процесі багатобайтового доступу, коли адресний показчик досягає 3Fh (кінець простору ОЗУ), він переміщається на осередок 00h – початок простору RTC.

00H	СЕКУНДЫ
	МИНУТЫ
	ЧАСЫ
	ДЕНЬ НЕДЕЛИ
	ДАТА
	МЕСЯЦ
	ГОД
07H	УПРАВЛЕНИЕ
08H	ОЗУ
3FH	56 x 8

Рисунок 2.11 – Карта адресов DS1307

Годинник і календар

Інформація від годин і календаря виходить читанням відповідних байтів регістра. Регістри RTC показані на **Рис. 3**. Час і календар встановлюються або не започатковано записом відповідних байтів регістра. Вміст регістрів часу і календаря має двійковий-десятковий формат. Біт 7 регістра 0 – це біт зупинки годин (clock halt – CH). Коли цей біт встановлений в 1, тактовий генератор *вимкнений*. Коли скинутий в 0, – тактовий генератор *включений*.

Майте на увазі, що при включенні харчування початковий стан всіх регістрів не визначене. Тому необхідно примусово включати тактовий генератор (біт CH = 0) під час початкової ініціалізації.

DS1307 може працювати і в 12- годинному, і в 24- годинному режимах. Біт 6 регістра годин відповідає за вибір 12- або 24- годинно-

го режиму. Коли він встановлений в 1, обраний 12- годинний режим. В цьому режимі біт 5 – це біт АМ/РМ, при цьому високий логічний рівень означає РМ. В 24- годинному режимі біт 5 – це біт другого десятка годин (20 – 23 години).

При появі на двохпровідний шині умови START, поточний час копіюється в другий набір регістрів. Інформація про час читається з цих допоміжних регістрів, в той час як годинник може продовжувати працювати. Це усуває необхідність перерахувати регістри DS1307 в разі поновлення їх вмісту під час читання.

	Бит7							Бит0	
00H	CH	ДЕСЯТКИ СЕКУНД			СЕКУНДЫ				00 - 59
	0	ДЕСЯТКИ МИНУТ			МИНУТЫ				00 - 59
	0	12 24	ВТОРОЙ ДЕСЯТОК Д/Р	ПЕРВЫЙ ДЕСЯТОК ЧАСОВ	ЧАСЫ				01 - 12 00 - 23
	0	0	0	0	0	ДЕНЬ			1 - 7
	0	0	ДЕСЯТКИ ДАТЫ		ДАТА				01 - 28/29 01 - 30 01 - 31
	0	0	0	ДЕСЯТОК МЕСЯЦА	МЕСЯЦ				01 - 12
	ДЕСЯТКИ ЛЕТ				ГОД				00 - 99
07H	OUT	0	0	SQWE	0	0	RS1	RS0	

Рисунок 2.12 – Регистры хронометра DS1307

2.6.6 Керуючий регістр

Керуючий регістр використовується для управління роботою виведення SQW/OUT.

БІТ 7	БІТ 6	БІТ 5	БІТ 4	БІТ 3	БІТ 2	БІТ 1	БІТ 0
OUT	0	0	SQWE	0	0	RS1	RS0

OUT (Output control – управління виходом) – цей біт управляє логічним рівнем на виведення SQW/OUT, коли вихід сигналу з прямо-

кутними імпульсами відключений. Якщо $SQWE = 0$, то логічний рівень на виводі SQW/OUT дорівнює 1, якщо $OUT = 1$, і 0, якщо $OUT = 0$.

SQWE (Square Wave Enable – включення сигналу з прямокутними імпульсами) – коли цей біт встановлений в 1, включається генерація прямокутних імпульсів. Їх частота залежить від значення бітів $RS0$ і $RS1$. (Якщо частота вихідного сигналу встановлена в 1Гц, регістри годин оновлюються по задньому фронту цих імпульсів).

RS (Rate Select – вибір частоти) – ці біти керують частотою прямокутних імпульсів, коли біт $SQWE=1$. В таблиці 1 наведені частоти, які можна вибрати за допомогою бітів RS .

Таблиця 1. Частота виходу SQW

RS1	RS0	Частота
0	0	1 Гц
0	1	4.096 кГц
1	0	8.192 кГц
1	1	32.768 кГц

2.6.7 Двохпровідна послідовна шина даних

DS1307 підтримує обмін даними по протоколу I2C по двохрановідний двобічної шині. Пристрій, який передає дані на шину, є передавачем, а пристрій, що приймає дані, – приймачем. Пристрій, що управляє передачею даних, називається ведучим. Пристрій, яким керує ведучий, називається веденим. Провідний пристрій генерує синхроімпульсів (serial clock – SCL), управляє доступом до шини і генерує умови START і STOP. DS1307 працює на шині як ведене пристрій. Типова конфігурація шини з використанням протоколу I2C показана на **Рис. 4**.

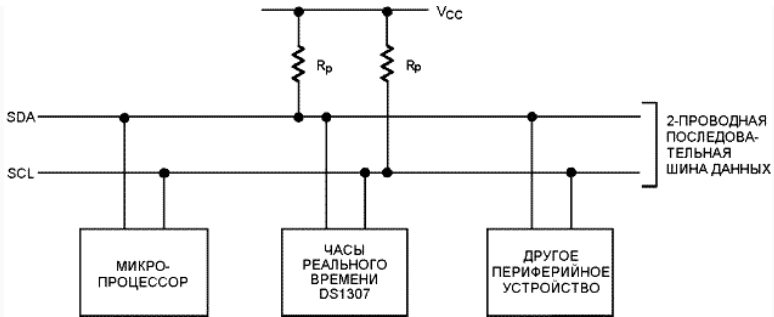


Рисунок 2.13 – Типова конфігурація двохпроводної шини

На Рис. 5, 6 і 7 детально показаний процес передачі даних по двохпроводній шині.

Основні принципи передачі даних по шині I2C:

а) Передача даних може бути ініційована лише коли шина вільна.

б) Під час передачі дані на лінії SDA можуть змінюватися тільки коли на лінії SCL низький рівень, в іншому випадку зміна даних буде інтерпретуватися як керуючий сигнал.

Таким чином, можливі наступні стани шини:

– **Шина не зайнята** – на лініях SDA і SCL зберігається високий рівень.

– **Початок передачі даних (умова START)** – зміна стану лінії SDA з високого на низький, в той час як на лінії SCL високий рівень.

– **Закінчення передачі даних (умова STOP)** – зміна стану лінії SDA з низького на високий, в той час як на лінії SCL високий рівень.

– **Коректні дані** – стан лінії SDA представляє коректні дані, якщо після умови START стан лінії SDA не змінюється протягом високого рівня тактового сигналу. Дані на лінії повинні змінюватися протягом періоду низького рівня тактового сигналу. На один біт даних доводиться один тактовий імпульс.

Кожна передача даних ініціюється умовою START і завершується умовою STOP. Число байтів даних, що передаються між умовами START і STOP, необмежена і визначається провідним пристроєм. Інформація передається побайтово, і кожен байт приймач підтверджує дев'ятим бітом (біт підтвердження – ACK). У специфікації двохпроводного інтерфейсу визначені звичайний режим (з тактовою частотою

100 кГц) і швидкий режим (з тактовою частотою 400 кГц). DS1307 працює тільки в звичайному режимі (100 кГц).

– **Підтвердження** – кожне приймаючий пристрій, коли є адресуються, зобов'язана генерувати підтвердження після прийому кожного байта. Провідне пристрій повинен генерувати додатковий тактовий імпульс, який призначений для біта підтвердження.

Підтверджує пристрій має підтягнути до низького рівня лінію SDA під час тактового імпульсу підтвердження таким чином, щоб на лінії SDA залишався стабільний низький рівень протягом періоду високого рівня тактового імпульсу, що відноситься до підтвердження. Звичайно, настройка і часи утримання повинні бути прийняті до уваги. Ведучий повинен сигналізувати відомому закінчення даних, які не генеруючи біт підтвердження на останньому байті, який був отриманий від веденого. В цьому випадку ведений повинен залишити лінію даних в стані високого рівня, щоб дозволити ведучому згенерувати умову STOP.

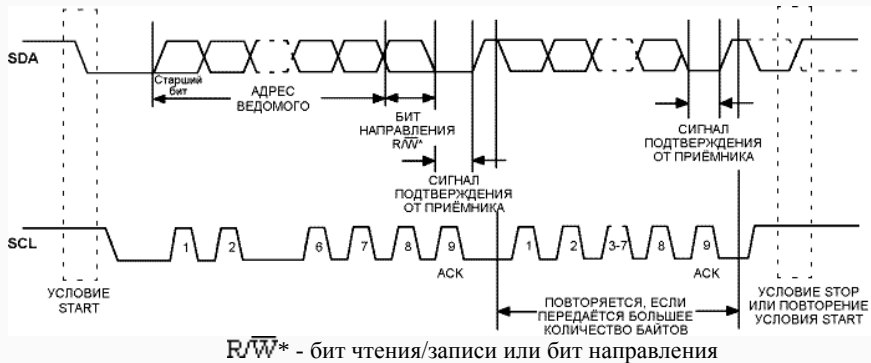


Рисунок. 2.14 – Передача даних по двохпроводній послідовній шині

Залежно від стану біта $R/\bar{W}^*$ можливі два типи передачі даних:

а) Передача даних від провідного передавача відомому прийма-
ча.

Перший байт, який передається провідним – це адреса веденого. Далі слід кілька байтів даних. Ведений повертає підтверджує біт після кожного прийнятого байта. Дані передаються, починаючи зі старшого біта (most significant bit – MSB).

б) Дані передаються від веденого передавача ведучому приймача.

Перший байт (адреса веденого) передається провідним. Ведений повертає підтверджує біт. За ним слідує передане веденим кілька байтів даних. Ведучий повертає підтверджує біт після всіх прийнятих байтів крім останнього байта. В кінці останнього прийнятого байта повертається "непідтвердження". Провідний пристрій генерує все послідовні синхроімпульсів, а також умови START і STOP. Передача закінчується умовою STOP або повтором умови START. Оскільки повторення умови START також є і початком наступної послідовної передачі, шина не звільняється. Дані передаються, починаючи зі старшого біта.

DS1307 може працювати в наступних двох режимах:

1) Режим веденого приймача (режим запису DS1307).

Послідовні дані приймаються по SDA і синхронізуються по SCL. Після кожного прийнятого байта передається біт підтвердження. Умови START і STOP розпізнаються як початок і кінець послідовної передачі. Розпізнавання адреси виконується апаратно після прийому адреси веденого і біта напрямку (див. **Рис. 6**).

Байт адреса – це перший байт, прийнятий після генерації провідним умови початку. Він містить 7 бітів адреси DS1307, який має значення **1101000**, і наступний за ним біт напрямку ($R/\overline{W}$), який для запису дорівнює 0. Після прийому і декодування адресного байта DS1307 видає на лінію SDA підтвердження, після чого ведучий передає в пристрій адресу регістру. Це встановить реєстровий показник. Потім ведучий починає передачу байтів даних, кожен з яких буде підтверджуватися DS1307. Щоб завершити запис даних, провідний генерує умова закінчення.

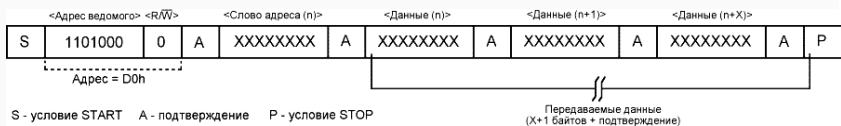


Рисунок 2.15 – Запис даних - режим веденого приймача

2) Режим веденого передавача (режим читання DS1307).

Перший байт приймається і обробляється так само, як і в режимі веденого приймача. Однак в цьому режимі біт напрямку буде вказувати, що напрямок передачі інвертовану. DS1307 передає послідовні дані по лінії SDA поки на лінію SCL надходять послідовні синхроімпульсів. Умови START і STOP розпізнаються на початку і в кінці послідовної передачі (див. **Рис. 2.16**).

Байт адреса – перший байт, що приймається після стартового умови, генерується провідним. Байт адреси складається з 7- бітного адреса DS1307, який дорівнює **1101000**, і наступного за ним біта напрямки передачі ($R/\overline{W}$), який для читання дорівнює 1. Після прийому і декодування байта адреси, пристрій видає підтвердження на лінію SDA. Потім DS1307 починає передавати дані, починаючи із зазначеної адреси, що знаходиться в регістровому покажчику. Якщо перед початком читання покажчик регістра не збережено, то перший читається адреса – це адреса, який був збережений в ньому останнім. DS1307 повинен прийняти "непідтвердження" для закінчення читання.



Рисунок 2.16 – Читання даних - режим веденого передавача

Примітки:

1. I_{CCS} вказується при $V_{CC}=5.0$ В і SDA, SCL = 5.0 В.
2. $V_{CC} = 0$ В, $V_{BAT} = 3$ В.
3. Після цього періоду генерується перший синхроімпульс.
4. Пристрій внутрішньо має забезпечувати час утримання принаймні 300 нс для сигналу SDA (враховуючи параметр V_{IHMIN} сигналу SCL), для того щоб подолати невизначену область заднього фронту SCL.
5. Максимальне значення $t_{HD:DAT}$ повинно з'являтися тільки принтер узагалі не розтягує період низького рівня (t_{LOW}) сигналу SCL.
6. C_B – повне ємкісне опір лінії шини в пФ.
7. I_{CCA} – синхронізація SCL на максимальній частоті (100 кГц).
8. V_{PF} виміряна при $V_{BAT} = 3.0$ В.

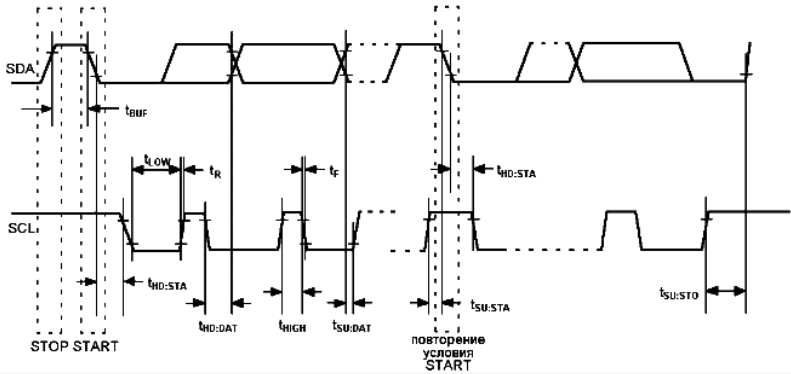


Рисунок 2.17 – Тимчасова діаграма

3 ЗАВДАННЯ ДО РОЗРАХУНКУ

3.1 Перелік тем реферативної частини розрахунково-графічних завдань

1. Історія розвитку інтегральних мікросхем. Фактори прогресу технології їх виробництва і основні віхи історії застосування мікросхем.
2. Історія становлення і розвитку засобів цифрової обчислювальної техніки.
3. Історія розвитку техніки мікропроцесорів і мікропроцесорної обчислювальної техніки.
4. Хронологія і еволюція програмного забезпечення та сервісних послуг користувача обчислювальної техніки.
5. Мікроконтролери сімейства 8051 (Intel).
6. Мікроконтролери сімейства C8051F34X.
7. Мікроконтролери сімейства MSP430 (TI).
8. Мікроконтролери AVR (Atmel).
9. Мікроконтролери PIC.
10. Мікроконтролери STM8 (STMicroelectronics).
11. Мікроконтролери STM32 (STMicroelectronics).
12. Мікроконтролери RL78 (Renesas Electronics).
13. Мікроконтролери з ядром ARM.
14. Мікроконтролери DSP.

15. Середовище розробки IAR Ebedded Workbench IDE.
16. Середовище розробки Keil.
17. Компілятори C/C++ для мікроконтролерів.
18. Платформа Arduino.
19. Платформа RASPBERRY PI ZERO.
20. Платформа NODEMCU.
21. Платформа STM32.
22. Платформа PARTICLE PHOTON.
23. Платформа ESP8266.
24. Платформа TEENSY.
25. Платформа BEAGLEBONE.
26. Платформа MSP430.
27. Класифікація програматоров.
28. Еволюція мережі Internet.
29. Історія створення і розвитку суперкомп'ютерів.
30. Історія становлення і розвитку кібернетики.
31. Історія розвитку периферійних пристроїв ЕОМ.
32. Історія розвитку носіїв інформації.
33. Історія розвитку найбільших компаній-виробників ЕОМ.
34. Мова програмування Сі: етапи еволюції та сучасний стан.
35. Мова програмування Асемблер: етапи еволюції та сучасний стан.
36. Мова програмування FORTRAN, FORTRAN 77: етапи еволюції та сучасний стан.
37. Мова програмування Algol 60, Algol 68, PL/1: етапи еволюції та сучасний стан.
38. Мова програмування Java: етапи еволюції та сучасний стан.
39. Мова програмування Perl: етапи еволюції та сучасний стан.
40. Мова програмування Basic і VB: етапи еволюції та сучасний стан.
41. Мови веб-програмування: етапи еволюції та сучасний стан.
42. Історія розвитку технологій програмування, ООП.
43. Історія розвитку технологій програмування, структурне програмування.
44. Процесори Intel Pentium. Еволюція.
45. Процесори AMD типу Pentium. Еволюція.
46. Елементна база РС – транзистор. Транзистор в сучасному CPU.

47. Елементна база РС – мікросхема. Мікросхема в сучасному CPU.

3.2 Завдання для практичної частини розрахунково-графічних завдань

1. На основі контролера з архітектурою 8051, годинника реального часу, та 4 рядкового LCD індикатора розробити принципову схему пристрою, який реалізовував би наступні функції:

А) У першому рядку відображає номер групи у якій навчається студент;

Б) У другому рядку відображає Прізвище та Ім'я студента;

В) У третьому рядку відображає поточний час з годинника реального часу (час виконання роботи), з інтервалом оновлення в 1 секунду;

Г) У четвертому рядку відображає поточну дату з годинника (час виконання роботи) реального часу, з інтервалом оновлення в 1 секунду.

2. Намалювати алгоритм роботи пристрою.

3. Створити програму для даного пристрою у відповідності з алгоритмом.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. **Майоров, В.Г.** Практический курс программирования микропроцессорных систем /В.Г.Майоров, А.И.Гаврилов.-М.: Машиностроение,1989.-272 с.
2. Микропроцессоры. В 3-х кн. Кн.1. Архитектура и проектирование микро-ЭВМ. Организация вычислительных процессов: учеб. для втузов. /П.В.Нестеров, В.Ф.Шаньгин, В.Л.Горбунов и др.; под ред. Л.Н.Преснухина. М.: Высш. шк., 1986. - 495 с.: ил.
3. **Самофалов, К.Г.** Микропроцессоры – Библиотека инженера /К.Г.Самофалов, О.В.Викторов.– К.: Техника, 1989. – 312 с., глава 3.

4. **Корячко, В.П.** Микропроцессоры и микро-ЭВМ в радиоэлектронных средствах: учеб. для вузов по спец. "Конструирование и технология радиоэлектронных средств.- М.: Высш. шк., 1990.-407 с.
5. **Каган, Б.М.** Основы проектирования микропроцессорных устройств автоматики /Б.М.Каган, В.В.Сташин.- М.: Энергоатомиздат, 1987.-304 с.
6. **Балашов, Е.П.** Микропроцессоры и микропроцессорные системы: учеб. пособие для вузов /Е.П.Балашов, Д.В.Пузанков; под ред. В.Б.Смолова. - М.: Радио и связь, 1981.- 328 с.: ил.
7. **Токхайд, Р.** Микропроцессоры: Курс и упражнения; пер. с англ.; под ред. В.Н.Грасевича.- М.: Энергоатомиздат, 1988. - 336 с.
8. **Григорьев, А.И.** Микропроцессоры и микропроцессорные системы: Руководство к практическим занятиям и лабораторным работам /А.И.Григорьев, В.В.Сысоев. - Харьков: УИПА, 1996. - 165 с.
9. **Коффон, Дж.** Расширение микропроцессорных систем /Дж.Коффон, В.Лонг. - М.: Машиностроение, 1987. - 320 с.
10. Микропроцессорное управление технологическим оборудованием микроэлектроники: учебное пособие. /А.А.Сазонов, Р.В.Корнилов, Н.П.Кохан и др.; под ред. А.А.Сазонова. - М.: Радио и связь, 1983. - 264 с.
11. Сопряжение датчиков и устройств ввода данных с компьютерами IBM PC: пер. с англ.; под ред. У.Тошинса, Дж. Уэбстера. - М.: Мир, 1992. - 592 с., ил.

Додаток А
Приклад оформлення титульної сторінки

Міністерство освіти та науки України

Запорізький національний технічний університет

Кафедра МiHE

РОЗРАХУНКОВО-ГРАФІЧНЕ ЗАВДАННЯ

з дисципліни

„МІКРОПРОЦЕСОРНА ТЕХНІКА“

(тема роботи)

Виконав:

студент ____ курсу, група РТ—__

(Ініціали, Прізвище)

Прийняв:

(посада)

(Ініціали, Прізвище)

20__ р.

Додаток Б

Приклад виконання завдання

1 Відкрити Proteus 8 Professional та створити новий проект (New Project). Обрати робочу папку та ввести ім'я проекту. Next.

2 Обрати Create a schematic from the selected template. Натиснути Next.

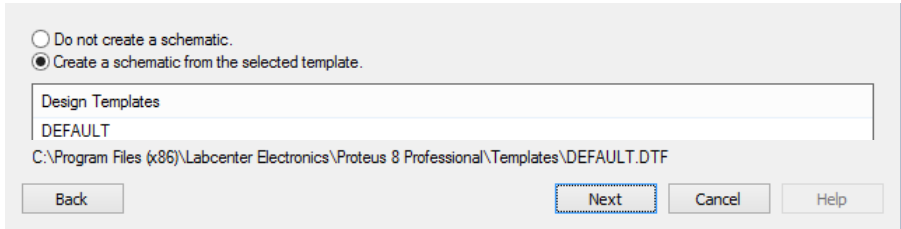


Рисунок Б.1 – Print screensпроцесу створення проекту

3 Обрати Do not create a PCB layout. Натиснути Next.

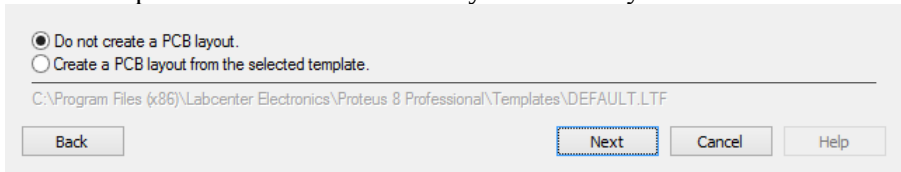


Рисунок Б.2 – Print screensпроцесу створення проекту

4 Обрати Create Firmware Project.8051. 80C51. Keil for 8051. Натиснути Next.

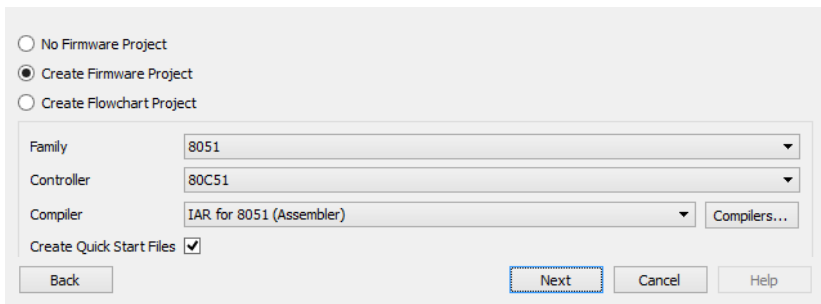


Рисунок Б.3 – Print screensпроцесу створення проекту

5 Перевірити налаштування та натиснути Finish.

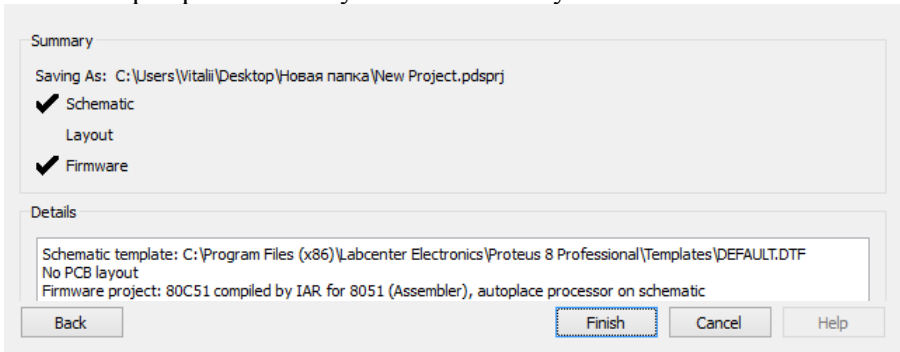


Рисунок Б.4 – Print screenпроцесу створення проекту

6 Проект створено, тепер використовуючи інструменти програми у вкладці Schematic Capture збираємо схему як на рис Б.5

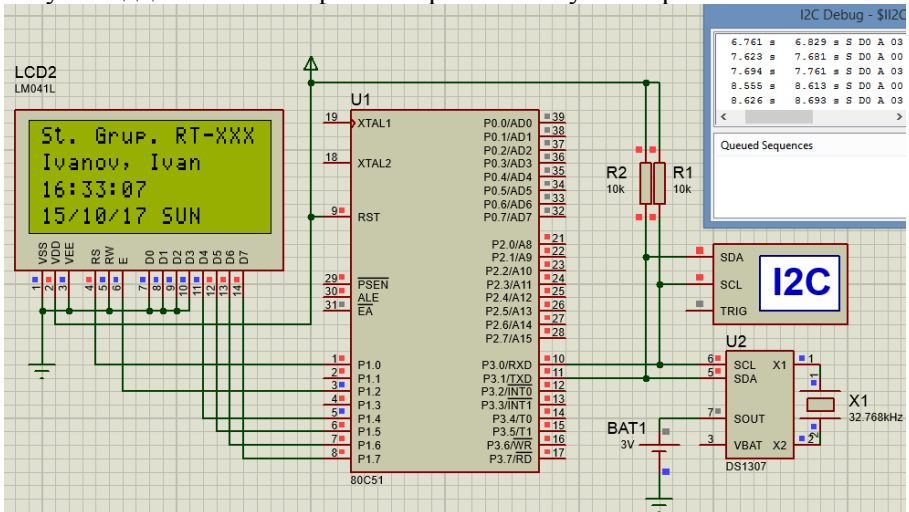


Рисунок Б.5 – Схема підключення пристроїв до мікроконтролера

7 Далі переходимо у вкладку Source Code. І створюємо програму.

```
#include <reg51.h>
#include <stdio.h>
// Визначте адресу пристрою DS1307 на шині i2c
#define Device_Address_DS1307_EEPROM 0xD0
```

```

// Визначення пінів i2c
sbit SCL = P3^0;          // Тактовий пін для i2c
sbit SDA = P3^1;          // Пін даних i2c
sbit RS = P1^0;
sbit E = P1^2;
// Визначення швидкості i2c
#define I2C_SPEED_FACTOR 1
#define Crystal_Value 12 // MHz
#define HalfBitDelay
(500*Crystal_Value)/(12*I2C_SPEED_FACTOR)
// Визначення макросів
#define Set_SDA_Low SDA = 0
#define Set_SDA_High SDA = 1
#define Set_SCL_Low SCL = 0
#define Set_SCL_High SCL = 1
// Визначення часових режимів
#define AM_Time 0
#define PM_Time 1
#define TwentyFourHoursMode 2
// Визначення днів
#define Monday 1
#define Tuesday 2
#define Wednesday 3
#define Thursday 4
#define Friday 5
#define Saturday 6
#define Sunday 7
// константи
#define E_Delay 250
unsigned char pRTCArray[4]; // RTC Масив
unsigned char Temp;

//Декларація функцій
void __delay_us(unsigned int);
void InitI2C(void);
void I2C_Start(void);
void I2C_ReStart(void);
void I2C_Stop(void);
void I2C_Send_ACK(void);
void I2C_Send_NACK(void);
bit I2C_Write_Byte(unsigned char);
unsigned char I2C_Read_Byte(void);
void delay(unsigned int);
void Write_Byte_To_DS1307_RTC(unsigned char, unsigned char);
unsigned char Read_Byte_From_DS1307_RTC(unsigned char);
void Write_Bytes_To_DS1307_RTC(unsigned char,unsigned
char*,unsigned char);

```

```

void Read_Bytes_From_DS1307_RTC(unsigned char,unsigned
char*,unsigned int);
void Set_DS1307_RTC_Time(unsigned char,unsigned char,unsigned
char,unsigned char);
unsigned char* Get_DS1307_RTC_Time(void);
void Set_DS1307_RTC_Date(unsigned char,unsigned char,unsigned
char,unsigned char);
unsigned char* Get_DS1307_RTC_Date(void);
void InitLCD(void);
void WriteCommandToLCD(int);
void WriteDataToLCD(char);
void ClearLCDScreen(void);
void WriteStringToLCD(const char*);
void DisplayTimeToLCD(unsigned char*) ;
void DisplayDateOnLCD( unsigned char*);

void main(void){
    InitLCD(); // Ініціалізація LCD
    InitI2C(); // Ініціалізація пінів i2c
    Set_DS1307_RTC_Time(TwentyFourHoursMode, 16, 32, 59);
    // Встановлення часу 08:32:59 AM
    Set_DS1307_RTC_Date(15, 10, 17, Sunday);// Встановлення
дати 02-11-2012 @ Friday
    ClearLCDScreen(); // Очистка дисплею і встанов-
лення курсору у 1 позицію
    WriteCommandToLCD(0x80); // Встановлення курсору у
першу строку
    WriteStringToLCD("St. Grup. RT-XXX");//Виведення першої
сторони
    WriteCommandToLCD(0xC0); // Встановлення курсору у
другу строку
    WriteStringToLCD("Ivanov, Ivan");//Виведення другої сто-
роки
    while(1)
    {
        WriteCommandToLCD(0x90);
        DisplayTimeToLCD(Get_DS1307_RTC_Time());// Виведення
часу з RTC у третю строку LCD
        WriteCommandToLCD(0xD0);
        DisplayDateOnLCD(Get_DS1307_RTC_Date());// Виведення
дати з RTC у четверту строку LCD
        delay(65000); // затримка в 1 секунду
    }
}
void ToggleEpinOfLCD(void){
    E = 1; // Тробоющий
    _delay_us(E_Delay); // імпульс

```

```

    E = 0; // E
    __delay_us(E_Delay);
}
void WriteDataToLCD(char t){
    RS = 1; // Передача даних
    P1 &= 0x0F; // Вивід першої
    P1 |= (t&0xF0); // тетради
    ToggleEpinOfLCD(); // стробуючий імпульс
    P1 &= 0x0F; // Вивід другої
    P1 |= ((t<<4)&0xF0); // тетради
    ToggleEpinOfLCD(); // стробуючий імпульс
}
void WriteCommandToLCD(int z){
    RS = 0; // Передача команди
    P1 &= 0x0F; // Вивід першої
    P1 |= (z&0xF0); // тетради
    ToggleEpinOfLCD(); // стробуючий імпульс
    P1 &= 0x0F; // Вивід другої
    P1 |= ((z<<4)&0xF0); // тетради
    ToggleEpinOfLCD(); // стробуючий імпульс
}
void InitLCD(void){
    RS = 0;
    E = 0;
    __delay_us(15000);
    P1 &= 0x0F;
    P1 |= 0x30;
    ToggleEpinOfLCD();
    __delay_us(4500);
    P1 &= 0x0F;
    P1 |= 0x30;
    ToggleEpinOfLCD();
    __delay_us(300);
    P1 &= 0x0F;
    P1 |= 0x30;
    ToggleEpinOfLCD();
    __delay_us(650);
    P1 &= 0x0F;
    P1 |= 0x20;
    ToggleEpinOfLCD();
    __delay_us(650);
    WriteCommandToLCD(0x28); //function set
    WriteCommandToLCD(0x0c); //display on,cursor off,blink
off
    WriteCommandToLCD(0x01); //clear display
    WriteCommandToLCD(0x06); //entry mode, set increment
}

```

```

void ClearLCDScreen(void) {
    WriteCommandToLCD(0x01);
    __delay_us(2000);
} // Очистка екрану у повернення курсору у початкову позицію
void WriteStringToLCD(const char *s) {
    while(*s)
        WriteDataToLCD(*s++);
}
void DisplayTimeToLCD( unsigned char* pTimeArray) { // Відобра-
ження часу у форматі HH:MM:SS AM/PM
    WriteDataToLCD( (pTimeArray[2]/10)+0x30 ); // Відображення
годин
    WriteDataToLCD( (pTimeArray[2]%10)+0x30 ); // Відображення
годин
    WriteDataToLCD(':'); // ':'
    WriteDataToLCD( (pTimeArray[1]/10)+0x30 ); // Відображення
хвилин
    WriteDataToLCD( (pTimeArray[1]%10)+0x30 ); // Відображення
хвилин
    WriteDataToLCD(':'); // ':'
    WriteDataToLCD( (pTimeArray[0]/10)+0x30 ); // Відображення
секунд
    WriteDataToLCD( (pTimeArray[0]%10)+0x30 ); // Відображення
секунд
    WriteDataToLCD(' '); // ' '
    switch(pTimeArray[3]) // Відображення режиму
    {
        case AM_Time: WriteStringToLCD("AM"); break;
        case PM_Time: WriteStringToLCD("PM"); break;
        default: WriteDataToLCD(' '); WriteDataToLCD(' '); break;
    }
}
void DisplayDateOnLCD( unsigned char* pDateArray) { // Відобра-
ження дати у форматі DD:MM:YY @ Day format
    WriteDataToLCD( (pDateArray[1]/10)+0x30 ); // Відображення
дня
    WriteDataToLCD( (pDateArray[1]%10)+0x30 ); // Відображення
дня
    WriteDataToLCD('/'); // '/'
    WriteDataToLCD( (pDateArray[2]/10)+0x30 ); // Відображення
місяця
    WriteDataToLCD( (pDateArray[2]%10)+0x30 ); // Відображення
місяця
    WriteDataToLCD('/'); // '/'
    WriteDataToLCD( (pDateArray[3]/10)+0x30 ); // Відображення
року

```

```

WriteDataToLCD( (pDateArray[3]%10)+0x30 );// Відображення
поку
WriteDataToLCD(' ');
switch(pDateArray[0])
{
case Monday:    WriteStringToLCD("MON");    break;
case Tuesday:   WriteStringToLCD("TUE");    break;
case Wednesday: WriteStringToLCD("WED");    break;
case Thursday:  WriteStringToLCD("THU");    break;
case Friday:    WriteStringToLCD("FRI");    break;
case Saturday:  WriteStringToLCD("SAT");    break;
case Sunday:    WriteStringToLCD("SUN");    break;
default:        WriteStringToLCD("???");    break;
}
}
void __delay_us(unsigned int d){
    unsigned int i, limit;
    limit = d/15;
    for(i=0;i<limit;i++);
}
void InitI2C(void){// Встановлення пінів SDA і SCL у початко-
вий стан
    SDA = 1;
    SCL = 1;
}
void I2C_Start(void){// Старт передачі
    Set_SCL_High;
    Set_SDA_High;
    __delay_us(HalfBitDelay);
    Set_SDA_Low;
    __delay_us(HalfBitDelay);
}
void I2C_ReStart(void){// Рестарт передачі
    Set_SCL_Low;
    __delay_us(HalfBitDelay/2);
    Set_SDA_High;
    __delay_us(HalfBitDelay/2);
    Set_SCL_High;
    __delay_us(HalfBitDelay/2);
    Set_SDA_Low;
    __delay_us(HalfBitDelay/2);
}
void I2C_Stop(void){//стоп передачі
    Set_SCL_Low;
    __delay_us(HalfBitDelay/2);
    Set_SDA_Low;
    __delay_us(HalfBitDelay/2);
}

```

```

    Set_SCL_High;
    __delay_us(HalfBitDelay/2);
    Set_SDA_High;
    __delay_us(HalfBitDelay/2);
}
void I2C_Send_ACK(void){//Відправка ACK біту
    Set_SCL_Low;
    __delay_us(HalfBitDelay/2);
    Set_SDA_Low;
    __delay_us(HalfBitDelay/2);
    Set_SCL_High;
    __delay_us(HalfBitDelay);
}
void I2C_Send_NACK(void){//Відправка NACK біту
    Set_SCL_Low;
    __delay_us(HalfBitDelay/2);
    Set_SDA_High;
    __delay_us(HalfBitDelay/2);
    Set_SCL_High;
    __delay_us(HalfBitDelay);
}
bit I2C_Write_Byte(unsigned char Byte){// I2C відправка байту
    unsigned char i;
    for(i=0;i<8;i++)
    {
        Set_SCL_Low;
        __delay_us(HalfBitDelay/2);
        if((Byte<<i)&0x80)
            Set_SDA_High;
        else
            Set_SDA_Low;
        __delay_us(HalfBitDelay/2);
        Set_SCL_High;
        __delay_us(HalfBitDelay);
    }
    Set_SCL_Low;
    Set_SDA_High;
    __delay_us(HalfBitDelay);
    Set_SCL_High;
    __delay_us(HalfBitDelay);
    return SDA;
}
unsigned char I2C_Read_Byte(void){// Читання біту з шини
    unsigned char i, d, RxData = 0;
    for(i=0;i<8;i++)
    {
        Set_SCL_Low;

```

```

        Set_SDA_High;
        __delay_us(HalfBitDelay);
        Set_SCL_High;
        __delay_us(HalfBitDelay/2);
        d = SDA;
        RxData = RxData | (d << (7-i));
        __delay_us(HalfBitDelay/2);
    }
    return RxData;
}

void delay(unsigned int d){//Функція затримки
    unsigned int i;
    for(i=0;i<d;i++);
}

void Write_Byte_To_DS1307_RTC(unsigned char Address, unsigned
char DataByte){//Запис байту у пам'ять DS1307, Адреси у
діапазоні 0 до 0xFF
    I2C_Start(); // Початок передачі
данних
    while(I2C_Write_Byte(Device_Address_DS1307_EEPROM + 0) ==
1)// очікування готовності пристрою
    { I2C_Start(); }

    I2C_Write_Byte(Address); // Запис адреси
    I2C_Write_Byte(DataByte); // Запис байту
данних
    I2C_Stop(); // Припинення передачі
данних
}

unsigned char Read_Byte_From_DS1307_RTC(unsigned char
Address){ //Читання байту у пам'ять DS1307, Адреси у діапа-
зоні 0 до 0xFF
    unsigned char Byte = 0;
    I2C_Start(); // Початок передачі
данних
    while(I2C_Write_Byte(Device_Address_DS1307_EEPROM + 0) ==
1)// очікування готовності пристрою
    { I2C_Start(); }
    I2C_Write_Byte(Address); // Запис адреси
    I2C_ReStart(); // Рестарт передачі
    I2C_Write_Byte(Device_Address_DS1307_EEPROM + 1); //
Передача адреси DS1307 RTC для читання команди
    Byte = I2C_Read_Byte(); // читання байту
EEPROM
    I2C_Send_NACK(); // Підтвердження успішного читання байту
    Set_SDA_Low;
    __delay_us(HalfBitDelay);
}

```

```

    Set_SDA_High;
    __delay_us(HalfBitDelay);
    return Byte;
}

void Write_Bytes_To_DS1307_RTC(unsigned char Address, unsigned
char* pData, unsigned char NoOfBytes){// Запис байту у DS1307
RTC
    unsigned int i;
    I2C_Start();
    while(I2C_Write_Byte(Device_Address_DS1307_EEPROM + 0) ==
1)// Очікування готовності пристрою
    {    I2C_Start();    }
    I2C_Write_Byte(Address);          // запис байту адреси
    for(i=0; i<NoOfBytes; i++)
        I2C_Write_Byte(pData[i]);    // запис байту даних
    I2C_Stop();
}

void Read_Bytes_From_DS1307_RTC(unsigned char Address, un-
signed char* pData, unsigned int NoOfBytes){
    unsigned int i;
    I2C_Start();
    while(I2C_Write_Byte(Device_Address_DS1307_EEPROM + 0) ==
1)
    {    I2C_Start();    }
    I2C_Write_Byte(Address);
    I2C_ReStart();
    I2C_Write_Byte(Device_Address_DS1307_EEPROM + 1);
    pData[0] = I2C_Read_Byte();          // Читання першо-
го байту з EEPROM
    for(i=1; i<NoOfBytes; i++)
    {
        I2C_Send_ACK();
        pData[i] = I2C_Read_Byte();
    }
    I2C_Send_NACK();
    Set_SDA_Low;
    __delay_us(HalfBitDelay);
    Set_SDA_High;
    __delay_us(HalfBitDelay);
}

void Set_DS1307_RTC_Time(unsigned char Mode, unsigned char
Hours, unsigned char Mins, unsigned char Secs){
    pRTCArray[0] = (((unsigned char) (Secs/10))<<4) | ((unsigned
char) (Secs%10));
    pRTCArray[1] = (((unsigned char) (Mins/10))<<4) | ((unsigned
char) (Mins%10));

```

```

    pRTCArray[2] = (((unsigned
char) (Hours/10))<<4) | ((unsigned char) (Hours%10)));
    switch (Mode)
    {
    case AM_Time:    pRTCArray[2] |= 0x40;    break;
    case PM_Time:    pRTCArray[2] |= 0x60;    break;
    default:        break;
    }
    Write_Bytes_To_DS1307_RTC(0x00, pRTCArray, 3); // запис
масиву у RTC
}
unsigned char* Get_DS1307_RTC_Time(void){
    Read_Bytes_From_DS1307_RTC(0x00, pRTCArray, 3); // Читання
годин, хвилин і секунд з RTC
    Temp = pRTCArray[0]; // Конвертація секунд з формату BCD в
число
    pRTCArray[0] = ((Temp&0x7F)>>4)*10 + (Temp&0x0F);
    Temp = pRTCArray[1]; // Конвертація хвилик з формату BCD в
число
    pRTCArray[1] = (Temp>>4)*10 + (Temp&0x0F);
    if(pRTCArray[2]&0x40) // Конвертація годин з формату BCD в
число
    {
        if(pRTCArray[2]&0x20)
            pRTCArray[3] = PM_Time;
        else
            pRTCArray[3] = AM_Time;
        Temp = pRTCArray[2];
        pRTCArray[2] = ((Temp&0x1F)>>4)*10 + (Temp&0x0F);
    }
    else
    {
        Temp = pRTCArray[2];
        pRTCArray[2] = (Temp>>4)*10 + (Temp&0x0F);
        pRTCArray[3] = TwentyFourHoursMode;
    }
    return pRTCArray;
}

void Set_DS1307_RTC_Date(unsigned char Date, unsigned char
Month, unsigned char Year, unsigned char Day){
    pRTCArray[0] = (((unsigned char) (Day/10))<<4) | ((unsigned
char) (Day%10));
    pRTCArray[1] = (((unsigned char) (Date/10))<<4) | ((unsigned
char) (Date%10));
    pRTCArray[2] = (((unsigned
char) (Month/10))<<4) | ((unsigned char) (Month%10));

```

```

    pRTCArray[3] = (((unsigned char)(Year/10))<<4)|((unsigned
char)(Year%10));
    Write_Bytes_To_DS1307_RTC(0x03, pRTCArray, 4); // Запис
RTC масиву у DS1307
}
unsigned char* Get_DS1307_RTC_Date(void){
    Read_Bytes_From_DS1307_RTC(0x03, pRTCArray, 4); // Читання
годин, хвилин, секунд з регістрів RTC
    Temp = pRTCArray[1]; // Конвенція дати з формату BCD у
число
    pRTCArray[1] = (Temp>>4)*10 + (Temp&0x0F);
    Temp = pRTCArray[2]; // Конвенція місяця з формату BCD у
число
    pRTCArray[2] = (Temp>>4)*10 + (Temp&0x0F);
    Temp = pRTCArray[3]; // Конвенція року з формату BCD у
число
    pRTCArray[3] = (Temp>>4)*10 + (Temp&0x0F);
    return pRTCArray;
}

```