

## Практична робота №5. Робота з інтерфейсом TWI (I<sup>2</sup>C) та годинником реального часу DS1307

**Мета:** ознайомитись з принципом роботи інтерфейсом TWI (I<sup>2</sup>C) Arduino та обіну даними з годинником реального часу DS1307; закріпити навички виведення інформації на семигментний індикатор з використанням регістру зсуву 74HC595 та LCD-індикатор.

**Завдання:** написати програму годинника / будильника з відображенням інформації на LED або LCD індикаторі.

**Обладнання:** лабораторний макет/віртуальний стенд «Arduino Learner Kit»; USB – кабель; провідники-з'єднувачі.

### Загальні відомості

Двопроводовий послідовний інтерфейс TWI ідеально підходить для типових додатків на мікроконтролерах і вимагає тільки дві лінії зв'язку. Протокол TWI дозволяє проектувальнику системи зовні зв'язати до 128 різних пристроїв через одну двухпроводную двосторонню шину, де одна лінія – лінія синхронізації SCL і одна – лінія даних SDA. В якості зовнішніх апаратних компонентів, які потрібні для реалізації шини, потрібен лише підтягуючий до плюса живлення резистор на кожній лінії шини. Всі пристрої, які підключені до шини, мають свої індивідуальні адреси, а механізм визначення вмісту шини підтримується протоколом.

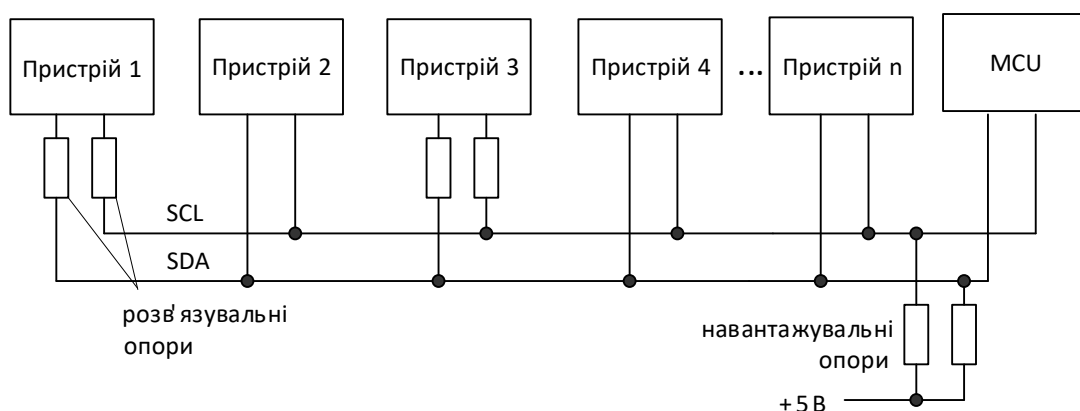


Рисунок 3.3 – Система протоколу TWI

Як показано на рис. 3.3, обидві лінії шини підключені до шини живлення через навантажувальні (pull up) резистори. У всіх сумісних з TWI пристроях, як драйвер шини, використовуються транзистор або з відкритим стоком, або з відкритим колектором. Так реалізована функція, яка дуже важлива для двобічної роботи інтерфейсу. Низький логічний рівень на лінії шини TWI генерується, якщо один або більше з TWI-пристроїв виводить логічний 0. Високий рівень на лінії присутній, якщо всі TWI-пристрої перейшли у третій високоімпедансний стан, дозволяючи підтягуючим резисторам задати рівень логічної 1. Зверніть увагу, що при підключенні до шини TWI декількох AVR- мікроконтролерів, для роботи шини, усі ці мікроконтролери повинні бути підключені до живлення.

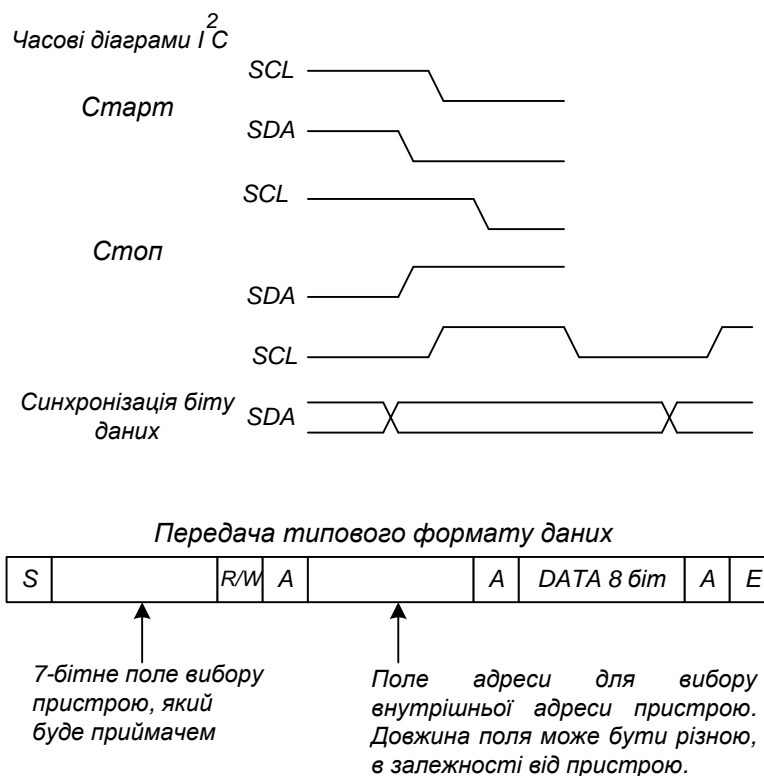


Рисунок 3.4 – Часові діаграми роботи шини I<sup>2</sup>C

Кількість пристроїв, яка може бути підключено до однієї шини обмежується гранично допустимою ємністю шини (400 пФ) і 7-розрядним простором адрес. Підтримуються два різних набори технічних вимог, де один набір - для шин зі швидкістю передачі даних нижче 100 кГц, а інший дійсний для швидкостей понад 400 кГц.

Уся передача даних складається із стартової послідовності, бітів і стопової послідовності. Початок передачі визначається Start послідовністю - провал SDA при високому рівні SCL (рис. 3.4).

При передачі інформації від Master до Slave, Master генерує такти на SCL і видає біти на SDA. Які Slave зчитує коли SCL стає 1.

При передачі інформації від Slave до Master, Master генерує такти на SCL і дивиться, що там Slave робить з лінією SDA – зчитує дані. А Slave, коли SCL йде в 0, виставляє на SDA біт, який Master зчитує коли підніме SCL назад.

Закінчується все STOP послідовністю. Коли при високому рівні на SCL лінія SDA переходить з низького на високий рівень.

Перший пакет від Master до Slave – це фізична адреса пристрою і біт напрямку (рис. 3.4).

Сама адреса складається з семи біт (ось чому до 127 пристроїв на шині), а восьмий біт означає, що буде робити Slave на наступному байті – приймати або передавати дані. Дев'ятим бітом йде біт підтвердження ACK. Якщо Slave розпізнав свою адресу повністю, то на дев'ятому такті він переведе лінію SDA в 0, згенерувавши ACK – тобто зрозумів. Тоді Master продовжить передачу даних. Якщо Slave не відповів, тобто SDA на дев'ятому такті не буде переведено в 0 (не буде ACK), то майстер припинить свої спроби під'єднатися.

Після адресного пакета йдуть пакети з даними в ту або іншу сторону, в залежності від біта R/W в заголовному пакеті.

На Arduino UNO, Nano, Pro Mini виводи I<sup>2</sup>C: контакт SDA – A4, контакт SCL – A5.

Бібліотека Wire.h дозволяє Arduino взаємодіяти з різними пристроями по інтерфейсу I<sup>2</sup>C / TWI. Згідно з протоколом I<sup>2</sup>C, адреса пристрою може складатися як з 7, так і з 8 біт. Як правило, 7 біт ідентифікують пристрій, в той час, як восьмий біт задає напрям передачі даних: від пристрою (читання) або до нього (запис). Всі функції бібліотеки Wire.h використовують 7-бітну адресацію, тим самим обмежуючи діапазон

можливих адрес в межах 0 – 127.

### **Функції I<sup>2</sup>C / TWI на платах Arduino:**

***Wire.begin(address)*** ініціалізує бібліотеку Wire і підключає Arduino до шини I<sup>2</sup>C в ролі ведучого (master) або веденого (slave) пристрою, де *address*: 7-бітова адреса Slave-пристрою (необов'язковий параметр); якщо адреса не вказана, то Arduino виступає в ролі Master-пристрою.

***Wire.requestFrom (address, quantity)*** запрошує дані у веденого пристрою (slave); як правило, використовується тільки ведучим пристроєм (Master). Після виклику requestFrom () запитувані дані повинні бути зчитані за допомогою функцій available () і read (), де *address*: 7-бітова адреса відомого пристрою, у якого запитуються дані; *quantity*: кількість запитуваних байт.

***Wire.beginTransmission (address)*** починає процедуру передачі даних по інтерфейсу I<sup>2</sup>C веденому пристрою з вказаною адресою. Для подальшої відправки даних, необхідно спершу поставити їх в чергу за допомогою функції write (), після чого здійснити, безпосередньо, передачу функцією endTransmission ().

***Wire.endTransmission ()*** завершує процедуру передачі даних веденому пристрою, ініційовану функцією beginTransmission(). При цьому функція відправляє байти, поставлені в чергу функцією write ().

***Wire.write(value)*** ***Wire.write(string)*** ***Wire.write(data, length)*** повертає кількість записаних байт, де *value*: значення, яке необхідно відправити у вигляді одиночного байта; *string*: рядок, який необхідно відправити у вигляді послідовності байт; *data*: масив даних, який необхідно відправити у вигляді декількох байт; *length*: кількість переданих байт.

***Wire.available ()*** повертає кількість байт, доступних для зчитування функцією read (). На ведучому пристрої (Master), ця функція має викликатися після функції requestFrom (), а на веденому (Slave) - всередині обробника onReceive ().

***Wire.read ()*** зчитує байт даних, отриманий ведучим пристроєм від веденого (або навпаки) в результаті виконання функції requestFrom ().

DS1307 – це годинник реального часу з екстремально точним ходом, завдяки вбудованому кварцовому резонатору з температурною компенсацією. Інтерфейс передачі даних – I<sup>2</sup>C. У мікросхемі є також вхід для підключення резервної батареї (рис. 1.10). При відключенні основного живлення мікросхема автоматично перемикається на роботу від резервної батареї, точність ходу від резервної батареї не порушується.

У DS1307 підтримується підрахунок секунд, хвилин, годин, днів місяця (дати), днів тижня, місяців і років (з урахуванням високосного року для місяців). Підтримується робота в 12 і 24 годинному форматі.

Для підключення RTC годинника реального часу DS1307 було розроблено декілька бібліотек: Wire.h, TimeLib, DS1307RTC.h, DS1307.h, iarduino\_RTC.h. Варто звернути увагу на iarduino\_RTC.h. Бібліотеки універсальні (підходить для DS3231, DS1302). У додатку Л наведені приклади роботи з DS1307 та виводом інформації на LED та LCD індикатор. У прикладах використовується бібліотека DS1307.h, яку потрібно встановити на комп'ютер або додати в директорію з файлом проекту. Використовуються такі функції бібліотеки:

***DS1307.begin()*** – ініціалізація роботи RTC модуля.

***DS1307.getDate(clock)*** – отримання часу.

***DS1307.setDate (P, M, D, DT, Г, X, C)*** – встановлення часу (рік, місяць, день, день тижня, години, хвилини, секунди).

### **Хід виконання роботи**

1. Підключити схему до комп'ютера через USB порт плати Arduino та/або запустити віртуальний стенд у середовищі Proteus 8.

2. Завантажити програму DS1307\_LCD (додаток Л) до лабораторного макета / віртуального стенду, попередньо виконати з'єднання DS1307, LCD-індикатора, тактових кнопок та Arduino у відповідності до програми. Встановити бібліотеку DS1307.h в середовище Arduino IDE (див. п. 1.5) Дослідити роботу програми.

3. Завантажити програму DS1307\_SEG (додаток Л) до лабораторного макета / віртуального стенду, попередньо виконати з'єднання DS1307, LCD-індикатора, тактових кнопок та Arduino у відповідності до програми. Дослідити роботу програми.

### **Завдання**

1. Установити поточне значення часу на годинник. Реалізувати програму, яка виводить на LCD-індикатор дату, час, прізвище автора та виводить ці значення в Монітор послідовного порту кожні 5 сек.

2. Реалізувати програму, яка виводить на LCD-індикатор поточний час та час будильника. Виставити значення часу та будильник. При спрацьовуванні будильника звучить тональний сигнал та загоряється світлодіод.

3. Реалізувати програму, яка виводить на семисегментний індикатор час та дату (по черзі) з можливістю установки дати та часу.

**Підготувати звіт** згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань).

### **Контрольні питання**

1. Що таке DS1307, які має характеристики та який інтерфейс використовує?
2. Коротко опишіть принцип підключення годинника до Arduino.
3. Які бібліотеки існують для роботи DS1307?
4. Які основні функції має, описана в роботі, бібліотека DS1307?
5. Які значення та в яких діапазонах повертаються годинником?
6. Особливості обміну інформацією по шині I<sup>2</sup>C.
7. Навести часові діаграми роботи шини I<sup>2</sup>C для передавання інформації від Arduino до DS1307.