Лабораторна робота №4

UART. УНІВЕРСАЛЬНИЙ ПОСІЛІДОВНИЙ ІНТЕРФЕЙС ПЕРЕДАЧІ ДАНИХ

Мета роботи:

1. Практичне ознайомлення з універсальним послідовним портом передачі даних UART.
2. Підключення та налаштування UART.
3. Передача даних від плати Arduino до комп’ютера за допомогою інтерфейсу UART.

1 Короткі теоретичні відомості

1.1 Основні поняття

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

До складу UART входять:

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

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

Керуючі регістри задають режим роботи послідовного порту і його переривань.

У статусному регістрі встановлюються прапори за різними подіями.

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

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

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

Електричний сигнал кадру посилки виглядає так:

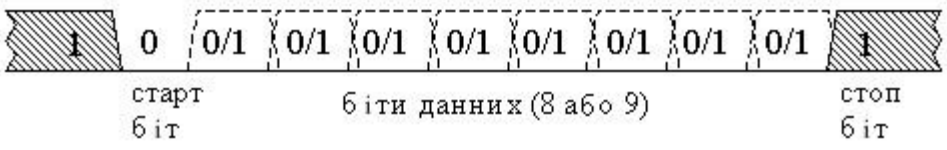

Рисунок 1 – Кадр посилки UART

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

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

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

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

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

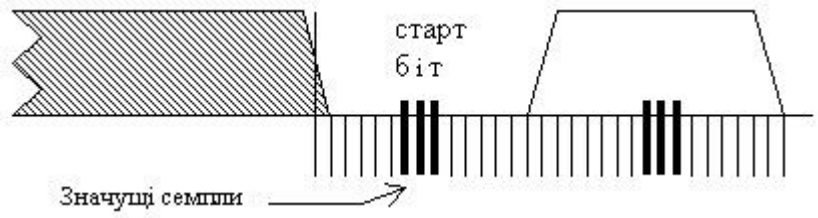

Рисунок 2 – Визначення бітів

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

1.2 Організація обміну даними між платою Arduino і комп'ютером через інтерфейс UART

Для зв'язку з платою Arduino можна використовувати спеціальну програму моніторингу послідовного порту (Serial Monitor), вбудовану в програмне забезпечення Ардуіно. Монітор послідовної шини відображає дані посилаються в платформу Arduino (плата USB або плата послідовної шини). Для відправки даних вибирається швидкість передачі зі списку, відповідна значенню Serial.begin в скетчі. Потім необхідно ввести текст і натиснути кнопку Send або Enter.

Плати Arduino в яких основний мікроконтролер ATmega328 мають один послідовний порт (також відомий як UART або USART): Serial. Він пов'язаний з цифровими виводами 0 (RX) і 1 (TX), а також використовується для зв'язку з комп'ютером через USB. Таким чином, під час використання послідовного порту, виводи 0 і 1 не можуть використовуватися в якості цифрових входів або виходів.

Для забезпечення зв'язку плати Ардуіно з комп'ютером або іншими пристроями використовується клас Serial. Клас - це абстрактний тип даних. За допомогою класу описується деяка сутність (її характеристики і можливі дії). Наприклад, клас може описувати змінні, параметри і функції послідовного порту. Клас Serial містить близько 20 функцій. Розглянемо деякі з них.

1.3 Функції для роботи з послідовним портом плати Arduino

if (Serial)

Параметри – Немає. **Значення, що повертаються** - boolean: повертає true, якщо вказаний послідовний порт готовий до роботи.

Опис: Дозволяє перевірити готовність певного послідовного порту.

Serial.begin(*speed****)***

Параметри:

speed: швидкість в бітах на секунду (бодах) – long

Опис: Задає швидкість передачі даних по послідовному інтерфейсу в бітах в секунду (бодах). Для взаємодії з комп'ютером слід використовувати одну з попередньо встановлених швидкостей обміну: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 або 115200.

Serial.end ()

Параметри – Немає. **Значення, що повертаються** - немає

Опис: Функція розриває послідовний зв'язок, після чого виводи RX і TX знову можуть використовувати як виводи загального призначення. Для відновлення послідовного з'єднання необхідно використовувати функцію *Serial.begin ().*

Serial.available()

Параметри – Немає. **Значення, що повертаються**: кількість байт, доступних для зчитування.

Опис: Повертає кількість байт (символів) доступних для зчитування з буфера послідовного порту. Під символами розуміються дані, які вже прийняті і

зберігаються в послідовному приймальному буфері (який може зберігати максимум 64 байти).

Serial.read()

Параметри – Немає. **Значення, що повертаються**: Перший байт прийнятих даних (або -1, якщо таких нема) - int

Опис: Зчитує дані, що надходять по послідовному інтерфейсу.

Serial.print(*val****)***

Serial.print(*val, format****)***

Параметри:

val: значення, яке необхідно вивести - будь-який тип даних

format: визначає систему числення (для цілочисельних типів), а також кількість десяткових знаків після коми (для чисел з плаваючою крапкою).

Значення, що повертаються: size_t (long): функція *print()* повертає кількість виведених байт. Зчитування цього значення не обов'язково.

Опис: Функція виводить через послідовний порт заданий ASCII текст у

вигляді, зрозумілому для людини. Ця команда може мати кілька різних форм.

Serial.println(*val****)***

Serial.println(*val, format****)***

Параметри:

val: значення, яке необхідно вивести - будь-який тип даних

format: визначає систему числення (для цілочисельних типів), а також кількість десяткових знаків після коми (для чисел з плаваючою точкою).

Значення, що повертаються:

size_t (long): функція *println()* повертає кількість виведених байт.

Зчитування цього значення не обов'язково.

Опис: Виводить через послідовний порт ASCII-текст в зрозумілому для людини вигляді з символами повернення каретки (ASCII 13 або '\ r') і нового рядка (ASCII 10 або 'n'). Ця команда має такі ж форми, як і *Serial.print()*.

При виведенні числа кожній його цифрі відповідає один ASCII-символ. Дробові числа теж виводяться у вигляді ASCII-цифр, при цьому після коми за замовчуванням залишається два десяткових знака. Байти виводяться у вигляді окремих символів, а символи і рядки виводяться без змін - "як є". Наприклад:

*Serial.print(*78*)* - виведе "78"

*Serial.print(*1.23456*)* - виведе "1.23"

Serial.print('N'*)* - виведе "N"

Serial.print("Hello world."*)* - виведе "Hello world."

Другий параметр необов'язковий. Він задає формат виведення; цей параметр може набувати таких значень: BIN (двійкова система з основою 2), OCT (восьмерична система з основою 8), DEC (десяткова система з основою 10), HEX (шістнадцятирічна система з основою 16). Для чисел із плаваючою крапкою (комою) цей параметр визначає кількість десяткових знаків після коми.

Наприклад:

Serial.print (78, BIN) - виведе "1001110"

Serial.print (78, OCT) - виведе "116"

Serial.print (78, DEC) - виведе "78"

Serial.print (78, HEX) - виведе "4E"

Serial.println (1.23456, 0) - виведе "1"

Serial.println (1.23456, 2) - виведе "1.23"

Serial.println (1.23456, 4) - виведе "1.2346"

2 Лабораторна установка

Лабораторна установка складється з плати Arduino (в разі відсутності, емулятор плати) ти персонального компютера з програмним забезпеченням Arduino IDE.

3 Хід виконання роботи

3.1 Ознайомтесь з лабораторною установкою та додатковими вказівками по роботі з приладами.

3.2 Підключіть плату Arduino до комп’ютера та запустіть Arduino IDE .

3.3 Налаштуйте порт на роботу при швидкості 9600 бод/с.

3.4 Напишіть програму яка буде виводити в монітор порту ваше прізвище, ім’я та групу в якій ви навчаєтесь, використовуючи наведені вище функції.

3.5 Намалюйте фрейм який буде передавати символ першої літери вашого імені, параметри такі: 1- стартовий біт, 1- стоповий, парність – відсутня. Як приклад можете використати зображений фрейм нижче, в якому виводиться символ велика англійська буква 'K' з таблиці ASCII, яка має порядковий номер 75.

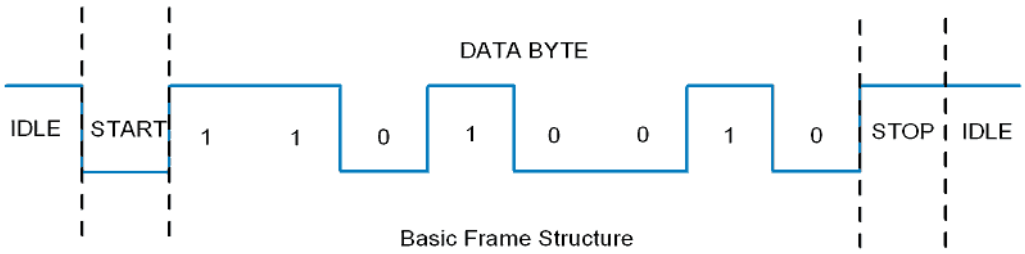

Рисунок 3 – Фрейм який передає значення 75.

3.6 Намалюйте фрейм який буде передавати символ першої літери вашого прізвища, параметри такі самі як і в попередньому пункті 3.5.

3.7 Намалюйте фрейм який буде передавати ваш порядковий номер в журналі груп в якій ви навчаєтесь, параметри такі самі як і в попередньому пункті 3.5.

4 Розрахункове завдання

Розрахункове завдання – див. п. 3.5, п. 3.6, п. 3.7.

5 Вимоги до звіту

Звіт з лабораторної роботи повинен містити:

1. Коротке описання мети і методики проведення роботи.

2. Перелік використаних приладів та матеріалів.
3. Таблиці результатів вимірювань, графічне оформлення.
4. Програмний код із середовища Arduino IDE з коментарями.
5. Розрахункове завдання.
6. Висновки.

6 Контрольні питання

1. Що означає термін UART?
2. Якою може бути швидкість передачі даних через UART?
3. Яка структура пакету даних?
4. Скільки інтерфейсів UART має плата Arduino?
5. Як організовано обмін між платою Arduino та комп'ютером?
6. Що означає поняття baudrate (бодрейт)?
7. Де використовується інтерфейс UART?
8. Який клас використовується в Arduino для роботи з UART?
9. Чи потрібно якось налаштовувати приймаючий пристрій, як і чому?
10. Чи можливе використання протоколу передачі даних UART в інших інтерфейсах передачі даних, яких?