МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт

з дисципліни
"Промислові інтерфейси"

для студентів спеціальності
172 Електронні комунікації та радіотехніка
174 Автоматизація, комп'ютерно інтегровані технології та робототехніка
усіх форм навчання

2024

Методичні вказівки до виконання лабораторних робіт з дисципліни "Промислові інтерфейси" для студентів спеціальностей 172 Електронні комунікації та радіотехніка, 174 Автоматизація, комп'ютерно інтегровані технології та робототехніка усіх форм навчання / Уклад.: Фарафонов О.Ю., Фурманова Н.І., Малий О.Ю., Гарачук С.А. – Запоріжжя: НУ «Запорізька політехніка», 2024. – 58 с.

Укладачі: Фарафонов Олексій Юрійович, канд. техн. наук, доцент, доцент каф. ІТЕЗ;
Фурманова Наталія Іванівна, канд. техн. наук, доцент, доцент каф. ІТЕЗ;
Малий Олександр Юрійович, канд. техн. наук, доцент, зав. каф. ІТЕЗ;
Гарачук Сергій Анатолійович, ст. викл. каф. ІТЕЗ.

Рецензент: Онищенко Вадим Федорович, к.ф.-м.н., доцент каф. ІТЕЗ

Відповідальний за випуск: Малий Олександр Юрійович, канд. техн. наук, доцент, зав. каф. ІТЕЗ

Затверджено
на засіданні кафедри ІТЕЗ
протокол № 1 від 10.09.24 р.

Рекомендовано до видання
НМК ФІБЕК
протокол № 2 від 19.09.2024 р.

Зміст

1. Лабораторна робота №1
ПЕРЕТВОРЮВАЧ USB –TTL. ДОСЛІДЖЕННЯ ПАКЕТУ ЗА ДОПОМОГОЮ ЛОГІЧНОГО АНАЛІЗАТОРУ

Мета роботи: отримати практичні навички роботи з логічним аналізатором.

1.1 UART та USART

UART (Universal Asynchronous Receiver/Transmitter) – універсальний асинхронний прийомопередавач, інтерфейс для зв'язку цифрових пристроїв, призначений для передачі даних в послідовній формі. Дуже поширений і дуже затребуваний, має апаратну реалізацію у багатьох мікроконтролерах. Наприклад, мікроконтролери STM32 із сімейства STM32F100xx, в залежності від варіанту виконання, містять 2 або 3 USART.

USART (Universal Synchronous-Asynchronous Receiver/Transmitter) – універсальний синхронно-асинхронний прийомопередавач – аналогічний до UART інтерфейс, але окрім можливостей UART, підтримує режим синхронної передачі даних із використанням додаткової лінії тактового сигналу. Синхронна передача використовується набагато рідше за асинхронну.

UART може використовуватися як для взаємодії компонентів всередині одного пристрою, так і для підключення пристроїв між собою. Для зовнішніх підключень сигнали з рівнями логіки ТТЛ або КМОП підходять мало через низьку завадостійкість. Поширеним стандартом фізичного рівня для UART, який підходить для підключення зовнішніх пристроїв, є RS-232. Цьому стандарту, зокрема, відповідає послідовний порт (COM-порт) комп'ютера. Таким чином, мікроконтролер за допомогою схеми перетворення рівнів може обмінюватися інформацією з COM-портом комп'ютера.

1.2 Варіанти підключення UART

У UART передача даних відбувається в послідовній формі, тобто по одному біту. Тому для передачі в одному напрямку потрібен один провідник; для повнодуплексного двонаправленого зв'язку знадобиться два провідники.

Вихід позначають TD або TX (transmitted data), вхід - RD або RX (received data). Для підключення двох пристроїв вихід першого пристрою підключають до входу другого і вхід першого – до виходу другого (рис. 1.1).

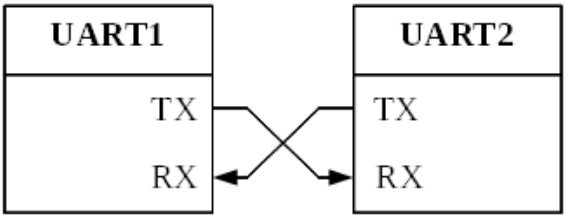


Рисунок 1.1 – Підключення для повнодуплексного зв’язку

Можливий варіант використання UART для напівдуплексного двонаправленого зв'язку по одному дроту. В цьому випадку виводи TX і RX кожного пристрою з’єднують разом. Увесь час, поки пристрій не передає даних, він тримає вихід у відключеному стані (переводить в Z-стан, стан з високим опором; можна використати режим роботи з відкритим стоком: під час паузи в передачі на виході UART формується 1, що є рівносильним переходу в Z-стан). Пристрій може мати апаратну підтримку напівдуплексного обміну даними, тоді потрібно лише вибрати режим роботи. Якщо апаратної підтримки немає, напівдуплексний режим легко реалізується програмно. Для цього треба відключати передавач, коли пристрій не передає даних, щоб звільнити лінію для здійснення передачі іншими пристроями, і відключати приймач під час роботи свого передавача, щоб не приймати власну передачу (або програмно відкидати дані, що передаються своїм передавачем).

До однопровідної лінії можна підключити декілька пристроїв, які утворюватимуть мережу для передачі даних. Арбітраж в цій мережі має бути реалізований програмно (рис. 1.2).

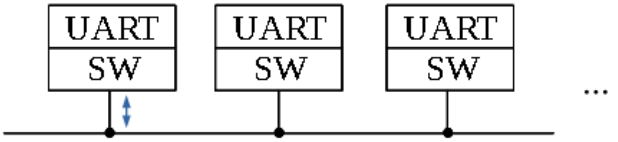


Рисунок 1.2 – Підключення для мережевої передачі даних

Для двонаправленого підключення потрібні тільки три провідники (з урахуванням загального дроту), а для однонапрямленого або двонаправленого напівдуплексного – всього два.

1.3 Формат передачі даних UART

За відсутності передачі на виході UART наявний рівень 1. Дані передаються у вигляді посилок (кадрів, фреймів), кожна з яких складається із стартового біта, бітів даних і одного або декількох стоп-бітів. Тривалість усіх бітів однакова, пов'язана зі швидкістю передачі співвідношенням T=1/S. Існує ряд стандартних швидкостей передачі: 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600 бод. Якщо всередині одного пристрою зв'язок можна здійснювати на довільній швидкості, то для зв'язку із зовнішніми пристроями слід дотримуватися стандартних величин (рис. 1.3).

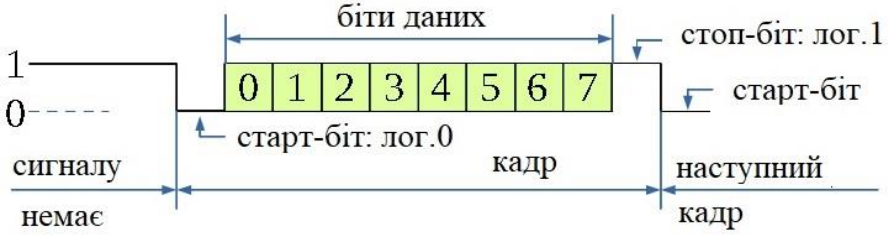


Рисунок 1.3 – Структура даних

Посилка розпочинається зі стартового біта, він завжди має значення логічного нуля (лог. 0). Після стартового біта передаються біти даних. Кількість бітів даних може складати 5-9 залежно від налаштувань UART. Зазвичай передається 8 біт даних або 9 біт (8 біт власне даних і один біт парності). Завершується посилка стоп-бітами, їх значення - завжди логічна одиниця (лог. 1), кількість зазвичай складає 1, 1.5 або 2. Під кількістю стоп-бітів розуміється тривалість одиничного імпульсу, що відповідає їм, по відношенню до тривалості бітів даних і старт-біта. Цим пояснюється можливість виражати кількість бітів дробовим числом. Відразу ж після стоп-бітів може починатися передача наступної посилки або може бути пауза

довільної тривалості, під час якої на виході також формується рівень лог. 1.

Оскільки під час передачі стоп-біта і доки ліня вільна, на виході є присутнім одиничне значення, а старт-біт має значення лог. 0, старт-біт дозволяє виявити момент початку передачі даних, розділити дві послідовні посилки і здійснити синхронізацію передавача і приймача. Якщо передавач і приймач працюють на одній швидкості, налаштовані на роботу з однаковою кількістю бітів даних, стоп-бітів, однаково конфігуровані відносно біта парності, то для обміну даними не потрібно передавати окремо тактовий сигнал – він може бути відновлений приймачем самостійно.

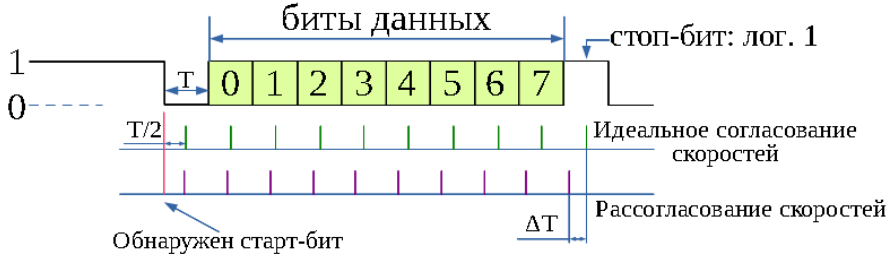


Рисунок 1.4 – Необхідність узгодження по частоті

Виявивши початок старт-біта, приймач чекає впродовж половини тривалості передачі біта, після чого починає зчитувати сигнал на вході з частотою, рівній швидкості передачі даних. В ідеальному випадку момент кожного зчитування припадає на середину біта, що приймається. У реальності генератори тактових імпульсів передавача і приймача мають розузгодження по частоті, в результаті кожне нове зчитування все більше зміщується відносно середини наступного біта (рис. 1.4).

Важливо, щоб за час передачі однієї посилки зміщення не перевищило половини тривалості біта, а з урахуванням перехідних процесів – зміщенню краще не перевищувати четвертої частини тривалості біта. Інакше замість зчитування біта відбудеться зчитування сусіднього біта (чи зчитування лінії під час перехідного процесу), і посилка буде прийнята невірно.

1.4 Керування потоком даних

Для керування потоком даних UART використовується програмний або апаратний метод. У разі програмного методу, інформація про готовність пристрою приймати дані або про необхідність зупинити передачу передається по тих же каналах, що і дані. Приймаюча сторона програмно розділяє дані та сигнали керування, відповідно до протоколу.

Інтерфейс UART передбачає можливість використання додаткових сигналів (CTS, RTS) для апаратного керування потоком даних. Апаратне управління може використовуватися деякими повільними пристроями або пристроями з простою схемною реалізацією. Проте воно потребує двох додаткових ліній для підключення пристрою (рис. 1.5).

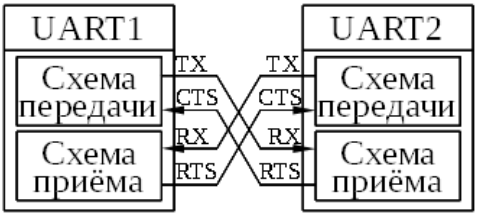


Рисунок 1.5 – Керування потоком даних

Якщо в UART включений контроль стану CTS, передавач перед відправкою чергового фрейма перевіряє вхід CTS. Якщо на CTS низький рівень, передача відбувається, в іншому випадку – ні. Якщо сигнал CTS буде встановлений під час передачі посилки (фрейма), поточна передача все одно буде завершена перед зупинкою. Приймач, у свою чергу, встановлює на виході RTS значення лог. 0, якщо він готовий приймати дані і встановлює лог. 1, вимагаючи від передавача зупинити передачу.

1.5 COM-порт (інтерфейс стандарту RS-232)

COM-порт – інтерфейс комп'ютера, що відповідає стандарту RS-232. Раніше комп'ютери, як правило, мали два COM-порти. Потім COM-порт почав витіснятися USB-інтерфейсом, і зараз у комп'ютера може бути всього один COM-порт або взагалі жодного. Якщо в комп'ютері відсутній COM-порт, то можна використати перехідник

USB - COM. Хоча він і вважається застарілим для використання в комп'ютерах, проте, інтерфейс стандарту RS-232 ще не втратив повністю свого значення, і існує устаткування, де він використовується. Інтерфейс не забезпечує високої швидкості передачі даних (максимум 115200 бод), але є простим, надійним і дешевим у реалізації.

RS-232 є стандартом фізичного рівня для інтерфейсу UART. Він визначає набір використовуваних сигнальних ліній і рівні для сигналів (табл. 1.1). Використовувані рівні значно відрізняються від традиційних ТТЛ або КМОП-рівней. По-перше, використовуються двополярні сигнали, по-друге, сигнал позитивної полярності відповідає логічному нулю. Для узгодження рівнів сигналів інтерфейсу RS-232 та іншої частини схеми на звичайній логіці використовуються спеціалізовані мікросхеми для перетворення рівнів.

Для сигналів використовуються наступні рівні.

Для драйвера (вихід):

+5…+15 В - лог. 0 (SPACE);

- 5…- 15 В - лог. 1 (MARK).

Вхід повинен мати опір в межах 3,7 кОм і має бути розрахований на сигнали:

+3…+25 В - лог. 0;

- 3…- 25 В – лог. 1.

Необхідно, щоб будь-якій вивід інтерфейсу витримував замикання на будь-якій інший вивід і на джерело напруги 5 В. У табл. 1.1 вказані позначення для сигналів прийняті для COM-порту, позначення відповідно до RS - 232, номери виводів в роз'ємах і короткий опис призначення сигналів.

Таблиця 1.1 – Сигнали інтерфейсу RS-232

COM	RS-232	DB-9P	DB-25P	I/O	Призначення
PG	AA	5	1	-	Protective Ground – захисна земля, з’єднується з корпусом пристрою.
SG	AB	5	7	-	Signal Ground – загальний провід для сигнальних ліній.
TD	BA	3	2	O	Transmitted data – передавання данних з порту.

RD	BB	2	3	I	Received Data – прийом даних у порт.
RTS	CA	7	4	O	Request to send – запит COM-порта на передавання даних (сигнал COM-порту щодо готовності приймати дані).
CTS	CB	8	5	I	Clear to send – вхід для дозволу COM-порту передавати дані.
DSR	CC	6	6	I	Data set ready – вхід сигнала готовності від підключеного до порту пристрою.
DTR	CD	4	20	O	Data terminal ready – сигнал готовності COM-порту до обміну даними.
CD	CF	1	8	I	Carrier Detected - сигнал знаходження несучої (від модему).
RI	CE	9	22	I	Ring indicator – сигнал від модема щодо отримання дзвінка.

Для підключення до інтерфейсу використовуються 25-контактні або 9-контактні роз'єми (DB25, DB9). Спочатку застосовувалися 25-контактні роз'єми, але багато сигналів не використовувалися пристроями. У зв'язку з цим стався перехід до 9-контактних роз'ємів. В устаткуванні використовуються роз'єми типу вилка (Pin): DB - 9P. У апаратурі передачі даних (модеми, наприклад) використовуються роз'єми типу розетка (Socket): DB - 9S.

1.6 Завдання на підготовку до роботи

1.6.1 Ознайомитись з методичними вказівками до лабораторної роботи.

1.6.1 Встановити драйвера для адаптера USB-TTL.

1.6.2 Підключити логичний аналізатор. Запустити відповідне програмне забезпечення (Saleae Logic 1.2.10) до аналізатора.

1.6.3 Підключити адаптер. Запустити среду розробки Arduino.

1.6.4 Отримати завдання для аналізу у викладача.

1.6.5 Використовуючи середовище Arduino, відправити діагностичний пакет та записати його за допомогою логічного аналізатору. Провести аналіз пакету.

1.6.6 Дати короткі письмові відповіді на контрольні питання.

1.7 Контрольні питання

1.7.1 Формат передачі даних в UART?

1.7.2 Що таке стоповий біт? Які вони бувають?

1.7.3 Як приймачем UART фіксується початок передачі даних?

1.7.4 Що таке апаратне керування потоком даних? Як воно відбувається?

1.7.5 Рівень сигналів RS-232?

1.7.6 Як побудувати гальванічну розв’язку для COM – порту?

1.7.7 Параметри послідовного інтерфейсу?

1.7.8 Як і чим визначена максимальна швидкість передачі даних для послідовного інтерфейсу?

1.7.9 Як в середовищі Arduino виконати зв’язок між двома COM-портами?

1.7.10 Вид пакету в UART та яким чином в ньому розміщені біти даних?

2. Лабораторна робота №2

РОБОТА з GSM МОДУЛЕМ SIM800. ПІДКЛЮЧЕННЯ. КЕРУВАННЯ.

Мета роботи: отримати навички керування пристроями за допомогою UART. Отримати знання щодо узгодження логічних рівнів різних пристроїв.

2.1 Загальні відомості

Об’єктом дослідження обрано мініатюрний модуль GSM/GPRS стільникового зв'язку на основі компонента Sim800l, розробленого компанією Simcom Wireless Solutions. Стандартний інтерфейс управління компонента Sim800l надає доступ до сервісів мереж GSM/GPRS 850/900/1800/1900МГц для відправки дзвінків, SMS повідомлень і обміну цифровими даними GPRS. Поставляється із вбудованою антеною, також можна підключити додаткові антени для поліпшення якості сигналу (рис. 2.1).

Рисунок 2.1 - Фрагмент плати модуля SIM800L

Керувати модулем можна за допомогою персонального комп'ютера через перетворювач інтерфейсу USB-UART або безпосередньо через UART модулем мікроконтролера самостійної розробки або Arduino, Raspberry Pi чи аналогічними.

Компонент Sim800l має реалізований стек протоколу TCP/IP. Містить мікросхему MT6260SA компанії Mediatek і мікросхему приймача RFMD RF7176. Завдяки функції відправки SMS повідомлень найчастіше модуль GSM GPRS Sim800 MICROSIM із

антеною використовується в диспетчеризації, безпровідній сигналізації і в охоронних системах. До модуля GSM GPRS Sim800 MICROSIM підключаються динамік і мікрофон. З модуля можна здійснювати дзвінки і приймати.

При ввімкненні модуля GSM GPRS на платі швидко блимає світлодіод. При установці з'єднання з мобільним оператором частота блимання знижується. Якщо зв'язок з мобільним оператором втрачений, то світлодіод знову блимає швидко. Для поліпшення якості сигналу під'єднується антена.

2.2. Підключення

Зовнішний вигляд модуля зі сторони слоту SIM-карти показано на рис. 2.2. Розщифровку позначень на платі модулю надано в табл. 2.1.

Таблиця 2.1 – Розшифрування позначень на платі

DTR додатковий сигнал UART	VCC жівлення
MICP з’єднується з мікрофоном	RST скидання
MICN з’єднується з мікрофоном	RXD до контакту TX мікроконтроллера
SPKP з’єднуеєься з динаміком	TXD до контакту RX мікроконтроллера
SPKN з’єднується з динаміком	GND загальний провід

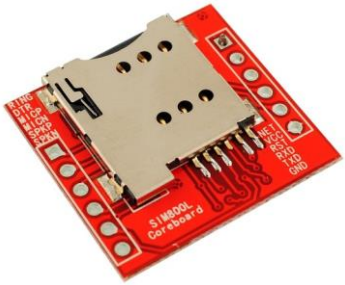


Рисунок 2.2 – Назви контактів модуля

Перевищення вхідної напруги інтерфейсу UART призведе до псування модуля Sim800. Не існує перетворювача інтерфейсу USB–

UART з вихідною напругою 2,8 В. Існуючі перетворювачі мають вищу напругу на виході UART. Тому між виходом перетворювача і входом модуля GSM GPRS Sim800 встановлюється резисторний подільник напруги (рис. 2.3).

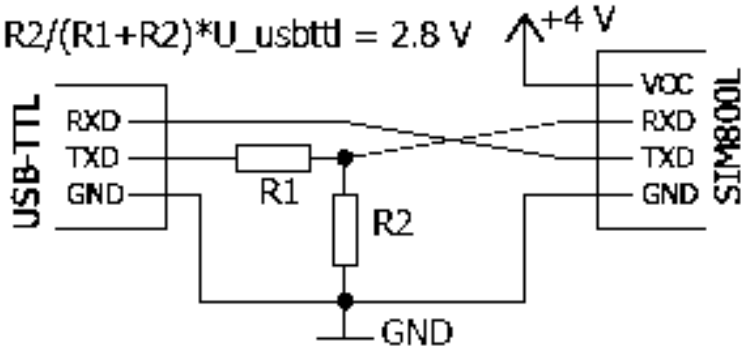


Рисунок 2.3 – Узгодження рівнів сигналів за допомогою резистивного подільника напруги.

Залежно від вихідної напруги U_usbttl наявного перетворювача інтерфейсу розраховуються номінали резисторів в подільнику за формулою, наведеною на рис. 2.3. Під час розрахунків слід прагнути до величин опорів порядка декількох кілоом.

2.3 Керування

Перевірку працездатності можна проводити за двома методами підключення модуля до ПК: через перетворювач інтерфейсів USB–UART або використовуючи Arduino UNO. Розглянемо метод перевірки без Arduino. Встановіть сім-карту в модуль GSM GPRS, дотримуючись розташування контактів. З'єднайте ПК через перетворювач інтерфейсів USB–UART з пристроєм за схемою. Підключіть живлення номінальною напругою.

Дочекайтесь підключення до мобільного оператора, орієнтуючись на блимання світлодіода. Ввімкніть на ПК термінальну програму. Дані в неї слід вводити великими буквами. Використовуючи термінальну програму, відправте до модуля GSM GPRS Sim800 через порт підключення наступні команди:

AT
Відповідь модуля OK

AT+CSQ
Відповідь модуля +CSQ: 18,0 OK

Остання команда дає інформацію про рівень сигналу. Перше число – рівень сигналу, величина 18 означає -78 dbm. Друге число – кількість помилково прийнятих біт, величина “0” говорить про долю помилок менше 0,2 %, що свідчить про хороший зв'язок.

Алгоритм встановлення зв’язку та отримання даних з мережі Internet наступний:

AT+SAPBR=3,1,"Contype","GPRS" // створюємо GPRS з’єднання
OK
AT+SAPBR=3,1,"APN","CMNET" // параметри точки доступу
OK
AT+SAPBR=1,1 // подключаємось
OK
AT+SAPBR=2,1 // перевірка підключення
+SAPBR: 1,1,"10.89.193.1"
OK
AT+CIPGSMLOC=1,1 // команда на запит щодо координат
+CIPGSMLOC: 0,111.667381,29.049339,2016/06/26,06:26:52 // 0 – з’єднання встановлено, 111.667381 – довгота, 29.049339 – широта, час.
OK
AT+CIPGSMLOC=2,1
+CIPGSMLOC: 0,2016/06/26,05:51:36 // запит часу
OK
AT+SAPBR=0,1 //Деактивувати GPRS з’єднання
 OK

Для відправлення SMS необхідно перевести модуль у текстовий режим:

AT+CMGF=1 – перевод модуля в текстовий режим;

AT+CMGS=”+38063XXXXXXX”

Після вводу тексту SMS повідомлення необхідно ввести символ завершення SMS повідомлення (mySerial.print((char)26) – в кодуванні ASCLL – char 26).

2.4 Порядок виконання

2.4.1 Отримати пристрої: перетворювач USB-TTL та GSM модуль.

2.4.2 Опрацювати документацію до пристроїв.

2.4.3 Провести підключення пристроїв за допомогою резистивного подільника напруги. Узгодити жівлення GSM модуля з стандартними напругами живлення.

2.4.4 Встановити у GSM модуль SIM картку.

2.4.5 Провести керування GSM модулем за допомогою базових AT команд.

2.4.6 Провести налагодженя GPRS каналу та отримати дані з мережі інтернет, використовуючи AT команди.

2.5 Контрольні питання

2.5.1 Що таке резистивний подільник? Як він використовується?

2.5.2 Що таке АТ команди? Як і де вони використовуються?

2.5.3 Яким чином забезпечено живлення модулю SIM800 у лабораторному стенді?

2.5.4 Що таке символ повернення каретки?

2.5.5 Яким символом завершується відправлення SMS у модулі SIM800?

2.5.6 Яким чином можна визначити, що модуль SIM800 зареєстровано у мережі?

2.5.7 Як прийняти вхідний дзвінок на модулі SIM800?

2.5.8 Що таке рівень сигналу у мережі GSM? Які одиниці вимірювань? Як його отримати на модулі SIM800?

3. Лабораторна робота №3

ВИДИ СХЕМОТЕХНІЧНОЇ РЕАЛІЗАЦІЇ АДАПТЕРІВ K-LINE

Мета роботи: отримати навички аналізу схем електричних принципових різних видів адаптерів. Отримати знання щодо узгодження логічних рівнів різних пристроїв.

3.1 Схеми адаптерів

K-line – одноканальна, але двонаправлена шина, яка застосовується в устаткуванні для автодіагностики, для зв'язку з електронними блоками керування (ЕБК). Використовується в системах з інжекторним впорскуванням палива двигунів внутрішнього згорання. Робота K-line забезпечена протоколами ISO 9141-2 та ISO 14230, які входять до відомого стандарту OBD II. До появи шини CAN, K-лінія сполучала електронні вузли автомобіля в єдине коло [1-5].

Швидкість обміну даними невелика – до 10 кБ за секунду. У протоколі ISO 9141-2 пакети передаються по 7 піну (K-лінія) сервісної колодки. L-line використовується лише для з'єднання ЕБК із сканером.

На сьогодні K-line адаптери представлені в основному з USB роз'ємом, а не COM-портом. Це пов'язано з тим, що діагностику зазвичай проводять з використанням ноутбуків, в яких немає COM-портів. Проте суть роботи адаптера не змінюється. Всередині адаптера встановлюють мікросхема-перетворювач з інтерфейсу USB в інтерфейс COM або в Bluetooth. Для кожного типу таких мікросхем необхідний драйвер, щоб в системі з'явився так званий віртуальний COM-порт, через який адаптер сполучатиметься з діагностичним програмним забезпеченням на комп’ютері [6].

Стандартна схема адаптера налічує три головних елементи:

- вхідний каскад – для приймання та узгодження рівней;
- контролер – для обробки потоку даних;
- коло живлення (живлення відбувається за допомогою штатної лінії USB каналу ноутбука або ПК).

Для узгодження сигналів використовуються спеціалізовані мікросхеми.

Мікросхема MC33199 використовується для узгодження з К-лінією і «розділення» та «змішування» сигналів (рис. 3.1).

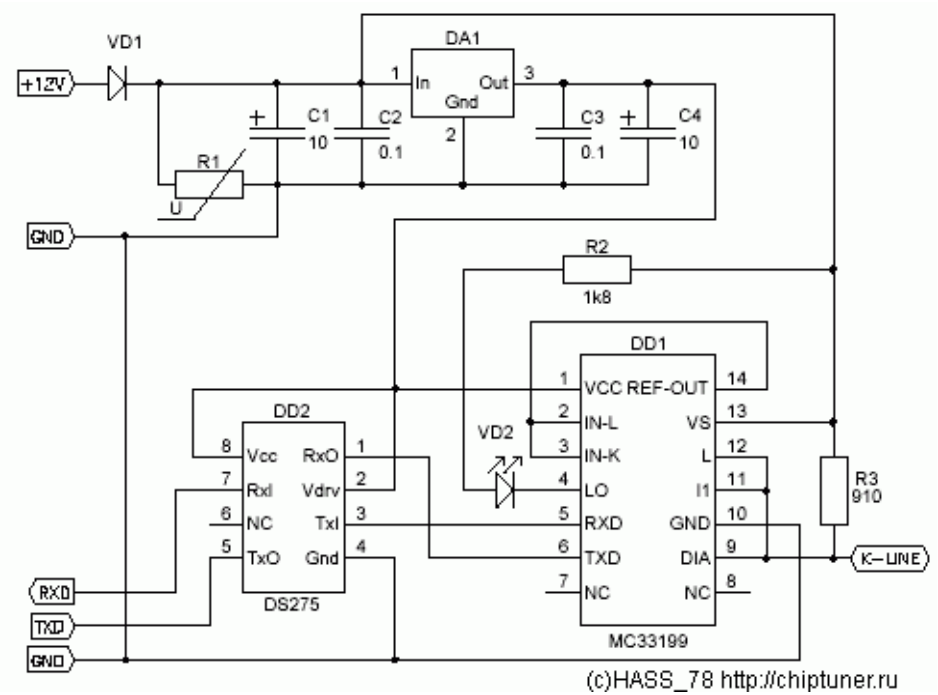


Рисунок 3.1 – Адаптер на MC33199

MAX232 – спеціалізована мікросхема для узгодження різних пристроїв з RS232 (стандарт COM-порту). MAX232 містить в собі інтегральні перетворювачі напруги, що дозволяє отримати потрібні для роботи порту +/-12V і приводить до необхідного рівня.сигнали, що надходять (рис. 3.2).

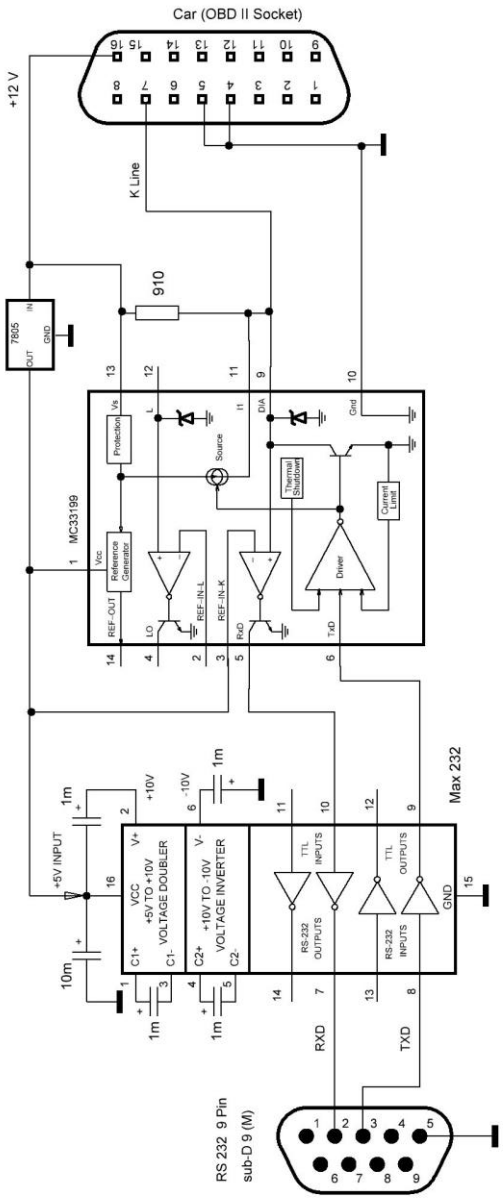


Рисунок 3.2 – Адаптер на MC33199 з використанням MAX 232

Більш спеціалізовані мікросхеми типу DS275, що виконують ті ж самі функції, що і MAX232, однак мають автоматичне настраювання вихідних сигналів відповідно до рівня вхідних та не потребують значної кількості ємностей для роботи (рис. 3.3).

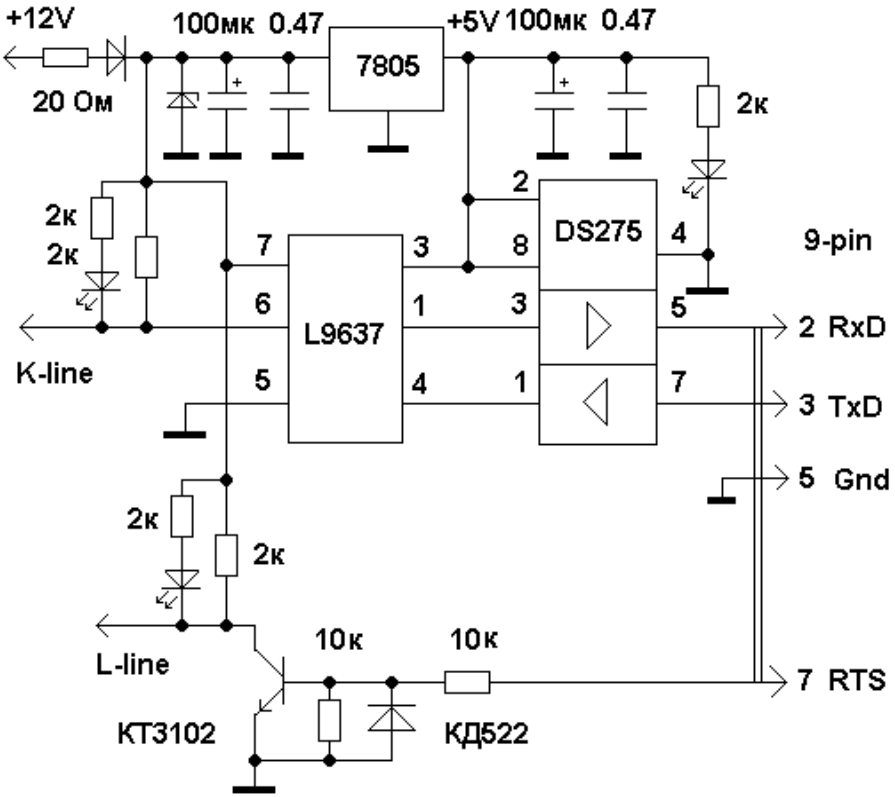


Рисунок 3.3 – Адаптер на L9637

На рис. 3.4 показано діод, що захищає схему адаптера від переполюсування, має бути з мінімальним падінням напруги, наприклад, діод Шоткі. В деяких випадках корисно підібрати номінал резистора R4 в межах 510 Ом – 1 кОм, шляхом вимірювання струму між K-лінією і загальним дротом в межах 15–20 мА.

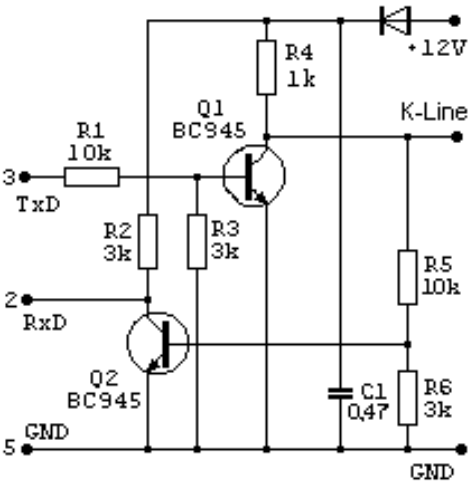


Рисунок 3.4 – Адаптер на двох транзисторах

Основна проблема адаптерів такого типу – транзистор, що передає сигнал від K-лінії на комп'ютер (Q1 на наведеній схемі), повільно закривається, що викликає необхідність підбору резисторів для запобігання перенасиченню транзистора. В іншому випадку фронт сигналу сильно запізнюється, що призводить до відсутності зв'язку.

3.2 OBD-команди

Стандарти вимагають відповідність даних, що надсилаються на автомобілей, OBD-формату. Перший байт (режим mode) завжди містить тип інформації запиту. Другий та наступні конкретизують дані. Ці байти називаються PID номером та описуються у стандартах SAE. У більшості автомобілів обмін з блоками OBD можливий тільки при увімкненому запаленні.

Режими (modes) OBD:

01 – show current data;

02 – show freeze frame data;

03 – show diagnostic trouble codes;

04 – clear troubles codes and store values;

05 – test results, oxygen sensors;

06 – test results, non-continuously monitored;

07 – show "pending" trouble codes;
08 – special control mode;
09 – request vehicle information.

У будь-якому режимі PID00 зазвичай використовують для виводу усіх значень PID, яки підтримуються на даному автомобілі. Режим 01, PID00 повинен підтримуватись усіми автомобілями. Якжо ввести команду >0100, то після можливих повідомлень ініціалізації шини OBD у відповідь отримаємо повідомлення 41 00 BE 1F B8 10.

Значення 41 00 вказує на те, що відповідь (4) на запит режиму (1) з PID 00 складається з чотирьох байт (BE 1F B8 10).

Запит >01 05 – це режим 01, PID 05. Відповідь може бути, наприклад, 41 05 7B. Відповідь 41 05 вказує, що відповідь (4) режиму (1) з PID 05 складається з одного байту (7B). Для визначення температури охолоджуючої рідини необхідно перевести це число з hex формату у десятковий формат та відняти 40:

7B hex = 123 T=123-40 = 83°C.

Адаптер K-лінії використовується для діагностичних робіт та визначення робочих параметрів двигунів на автомобілях зарубіжного виробництва, а також вітчизняних автомобілів, що оснащені електронними блоками керування систем двигуна.

Після підключення адаптера і запуску на комп'ютері відповідної програми можна здійснити наступні діагностичні операції:

– проводити поглиблену діагностику систем двигуна;
– визначати і зчитувати коди несправностей;
– використовувати весь об'єм обслуговуючих функцій (просушування свічок запалення, прогрівання двигуна).

Також можна проводити контроль та моніторинг систем під час руху:

– поточну напругу у бортовій мережі;
– оберти двигуна та його температуру;
– контроль споживання палива;
– швидкісні показники;
– поточний стан різних датчиків та причини їх несправностей;
– тривалість руху;
– кілометраж, що пройшов транспортний засіб.

3.3 Завдання на підготовку до роботи

3.3.1 Ознайомитися з методичними вказівками до лабораторної роботи.

3.3.2 Опрацювати літературу.

3.3.3 Дати короткі письмові відповіді на контрольні питання.

3.4 Контрольні питання

3.4.1 Як працює мікросхема L9637D?

3.4.2 Які вимоги до діагностичних тестерів відповідно до стандартів (перелік)?

3.4.3 Навіщо потрібна мікросхема MAX232? Як вона працює?

3.4.4 Який рівень сигналу у К-лінії? Як відбувається передача повідомлення?

3.4.5 Навіщо у схемі резистор 20 Ом (рис. 3.3)?

3.4.6 Для чого потрібен транзистор КТ3102 (рис. 3.3)?

3.4.7 Що таке SAE та ISO стандарти? Чим вони відрізняються?

3.5 Література

3.5.1 ISO/WD 14230-1 - Road Vehicles - Diagnostic Systems - Keyword Protocol 2000 - Physical layer. URL: https://www.iso.org/standard/55591/ (date of access 03.01.2024).

3.5.2 ISO14230-2 - Road vehicles — Diagnostic systems — Keyword Protocol 2000 - Part 2:Data link layer. URL: https://www.iso.org/standard/55592/ (date of access 03.01.2024).

3.5.3 ISO / DIS 14230 Road Vehicles - Diagnostic Systems - Keyword Protocol 2000 - Part 3: Implementation. URL: https://www.iso.org/obp/ui/#iso:std:iso:14230:-3:ed-1:v1:en/ (date of access 03.01.2024).

3.5.4 ISO/CD 14230-4 Road vehicles - Diagnostic systems - Keyword Protocol 2000 - Part 4 : Requirements for emission related systems. URL: https://www.iso.org/standard/28826/ (date of access 03.01.2024).

3.5.5 ISO 15031-5:2006 Road vehicles —Communication between vehicle and external equipment for emissions-related diagnostics — Part 5: Emissions-related diagnostic services. URL: https://www.iso.org/standard/29108/ (date of access 03.01.2024).

3.5.6 OBD-II_PIDs. URL: https://en.wikipedia.org/wiki/OBD-II_PIDs/ (date of access 03.01.2024).

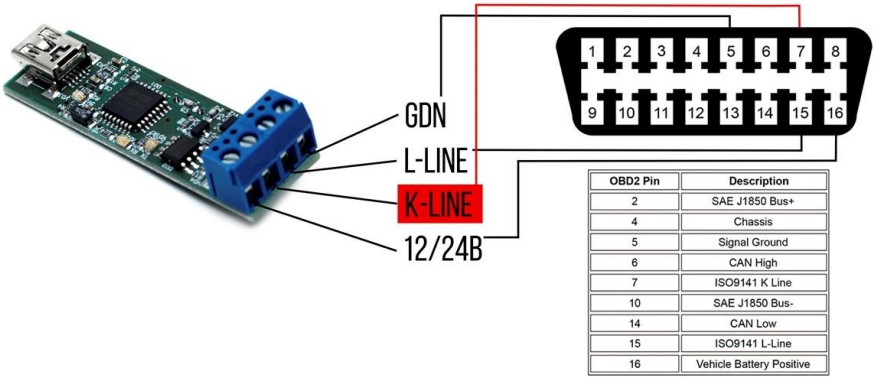
4. Лабораторна робота № 4

ПІДКЛЮЧЕННЯ ДО K-LINE. ПАРАМЕТРИ ДЛЯ ДІАГНОСТУВАННЯ

Мета роботи: отримати практичні навички роботи з адаптерами K-line. Ознайомитись з датчиками та виконуючими механізмами у автомобілі.

4.1 Підключення до K-line

Зовнішній вигляд стандартного роз'єму OBD-II та підключення до нього адаптеру USB-K-line наведено на рис. 4.1.



OBD2 Pin	Description
2	SAE J1850 Bus+
4	Chassis
5	Signal Ground
6	CAN High
7	ISO9141 K Line
10	SAE J1850 Bus-
14	CAN Low
15	ISO9141 L-Line
16	Vehicle Battery Positive

Рисунок 4.1 – Підключення до K-line

Адаптери USB-K-line потребують встановлення спеціалізованих драйверів у системі ПК.

Порядок підключення:

- підключити адаптер до порту ПК;
- встановити драйвери за необхідності;
- підключити адаптер до діагностичного роз'єму K-line;
- подати живлення на адаптер за необхідності;
- увімкнути запалення на автомобілі;
- запустити діагностичне ПЗ.

При запуску діагностичного програмного забезпечення (ПЗ) необхідно встановити відповідний порт до якого підключен адаптер K-line.

Послідовність відключення діагностичного обладнання:

– закрити діагностичне ПЗ;
– вимкнути запалення автомобіля;
– вимкнути додаткове живлення адаптеру (за наявності);
– від'єднати адаптер від діагностичного роз'єму автомобіля;
– від'єднати адаптер від ПК.

4.2 Параметри для діагностування

Запит щодо обертів двигуна має вигляд: >01 0C.

Відповідь – 41 0C 1A F8. 41 0C вказує, що відповідь (4) режиму (1) з PID 0C містить два байта (1A F8). Для отримання обертів необхідно перевести число з hex формату до десяткового та розділити результат на 4:

1AF8=6904/4=1726 (rpm).

Запит щодо серійного номеру автомобіля має вигляд: >09 02. Відповідь має вигляд послідовності, що містить декілька рядків:

49 02 01 00 00 00 31
49 02 02 44 34 47 50
49 02 03 30 30 52 35
49 02 04 35 42 31 32
49 02 05 33 34 35 36

Відповідь 49 02 у кожному рядку повторюється та розшифровується так само, як у попередньому прикладі. Третя група цифр – номер рядка (01…05). У першому ряку нулі відкидаються. Таким чином, отримуємо відповідь:

31 44 34 47 50 30 30 52 35 35 42 31 32 33 34 35 36.

Перетворюємо відповідь, користуючись таблицею ASCII: 1D4GP00R55B123456 (VIN-номер).

Отримання кодів помилок відбувається за допомогою запиту > 01 01 (режим 01, PID 01). У відповідь отримуємо 41 01 81 07 65 04. Третій байт у відповіді (81) несе інформацію щодо кількості помилок.

(07 65 04) інформація про типи тестів, які підтримуєть автомобіль (відповідно до SAE J1979). Перетворимо число 81 з формату hex до десяткового та віднімемо 128: отримуємо 1 помилку [1].

Для отримання даних щодо помилки необхідно надіслати запит про коди помилки: > 03. Відповідь 43 01 33 00 00 00 00 означає наступне:

(43) – відповідь 4 на режим 3.

Наступні байти аналізуємо попарно 0133 0000 0000. При цьому коди 0000 відкидаємо. Отриманий код 0133 аналізуємо згідно з табл. 4.1.

Таблиця 4.1 - Коди помилок

0	P0	Powetrain Codes(PC)-SAE defined
1	P1	PC-manufacturer defined
2	P2	PC- SAE defined
3	P3	P3 PC-jointly defined
4	C0	C0 Chassis Codes(CC) – SAE defined
5	C1	CC-manufacturer defined
6	C2	CC-manufacturer defined
7	C3	CC-reserved for future
8	B0	Bode Codes(BC) –SAE defined
9	B1	BC- manufacturer defined
A	B2	BC- manufacturer defined
B	B3	BC- reserved for future
C	U0	Network Codes(NC)- SAE defined
D	U1	NC- manufacturer defined
E	U2	NC- manufacturer defined
F	U3	NC- reserved for future

0133=P0133 де перша цифра в форматі hex замінюється на відповідний символ з табл. 4.1. P0133 – "oxygen sensor slow response" [1]. Для скидання кодів помилок існує запит >04.

Цей запит виконує:

– скидання кількості помилок;

– стирання кодів помилок;

– стирання freeze frame data;

– стирання кодів пов’язаних з freeze frame data;

– стирання усіх oxygen sensor test data;
– стирання результатів тестів режимів 06 та 07.

4.3 Порядок виконання

4.3.1 Отримати пристрої: перетворювач USB-TTL, плату Arduino, адаптер K-line та логічний аналізатор.

4.3.2 Опрацювати документацію до пристроїв.

4.3.3 Провести підключення пристроїв за допомогою резистивного подільника напруги.

4.3.4 Провести емуляцію підключення до K-line.

4.3.5 Сформувати діагностичний пакет за допомогою платформи Arduino та записати його за допомогою логічного аналізатора на виході адаптера K-line.

4.3.6 Отримати базові сигнали ініціалізації різних програм діагностування та розшифрувати їх у логічному аналізаторі.

4.4 Контрольні питання

4.4.1 Як розшифровується OBD-II?

4.4.2 Що таке протоколи OBD-II? Скільки їх?

4.4.3 Який формат коду несправності? Наведіть приклад розшифрування.

4.4.4 Як працює датчик кисню?

4.4.5 Що таке комбінація OBD-II –CAN?

4.4.6 Що таке kwp2000?

4.4.7 Що таке ініціалізація адаптера? Як вона відбувається?

4.5 Література

4.5.1 P0133 Oxygen Sensor Circuit Slow Response. URL: https://www.obd-codes.com/p0133/ (date of access 03.01.2024).

5. Лабораторна робота № 5

АНАЛІЗ ПАКЕТІВ ІНІЦІАЛІЗАЦІЇ ПРОГРАМ ДІАГНОСТУВАННЯ. КОМАНДИ ПРОТОКОЛУ OBD-II

Мета роботи: отримати практичні навички аналізу пакетів протоколу OBD-II.

5.1 Ініціалізація адаптерів

В більшості випадків при роботі через K-лінію за протоколами ISO 14230-1 і ISO 14230-2 використовується метод ініціалізації з'єднання під назвою fast initialization (швидка ініціалізація на противагу повільній 5-бодовій). У "вільному" стані лінія знаходиться в стані логічної одиниці. Для початку з'єднання потрібно на 25 мс перевести лінію в логічний нуль, потім на 25 мс назад в одиницю, і відразу після цього відправити стартове повідомлення (у звичайній конфігурації 8-N-1 зі швидкістю 10400 бод).

Потрібно відправити деякі дані, які б дали на виході 25 мс логічного нуля, а потім 25 мс логічної одиниці. Якщо відправляти дані в конфігурації 8-N-1, то кожен відправлений байт складається з послідовності бітів: один старт-біт (логічний нуль), вісім біт даних, один стоп-біт (логічна одиниця).

Якщо біти даних, що відправляються, складатимуться з 4 нульових бітів і 4 одиничних бітів, то виходить загальна послідовність з 5 нульових бітів і 5 одиничних. Кожні з цих 5 біт повинні передаватися потрібні 25 мс, що дає швидкість 1000/25*5=200 (бод). Оскільки дані передаються в "зворотному" порядку, то необхідним байтом з 4 нулями і 4 одиницями буде F0h. Тоді алгоритм ініціалізації спрощено виглядає так:

a) set 200 baud rate;
b) write F0h;
c) set 10400 baud rate;
d) write start message.

Пункти потрібно виконувати лише після фізичної передачі F0h. Практично для цього доводиться між кроками 2 і 3 додавати sleep 50 ms (у ідеальному випадку). Але тут виникає проблема із затримками у часі [1].

Після підключення сканера до роз'єму діагностики автомобіля та хост-комп'ютера відбувається ініціалізація у вигляді спеціальної посилки. Це є простим методом для визначення факту підключення. Перша операція – це посилка одиночного байта 20h. Це команда повідомляє мікроконтролер про початок комунікації. Мікроконтролер не повертає дану посилку, але передає у відповідь Ffh. З цієї миті мікроконтролер чекає здобуття даних.

Вибір протоколу здійснюється шляхом посилки контрольного байта 41h, за яким слідує байт вибору протоколу обміну: 0h=VPW, 1h=PWM, 2h=ISO 9141. Наприклад, 41h 02h означатиме вибір протоколу ISO 9141.

Мікроконтролер відповідає контрольним байтом і байтом статусу, який визначає внутрішні умови. Якщо ініціалізація успішна, то контрольний байт буде 01h, який показує, що слідуватиме один байт статусу. Цей байт по суті є байтом перевірки і визначається таким чином: для VPW або PWM цей байт є відлунням посланого байта (0 або 1); для ISO 9141 це буде «ключ», який повертає ЕБК і визначає одну з двох версій ISO 9141. Для автомобілів на протоколі ISO цей період може займати до 5 секунд. На цьому етап ініціалізації завершується.

Вигляд процесу ініціалізації відповідно до стандартів наведено на рис. 5.1.

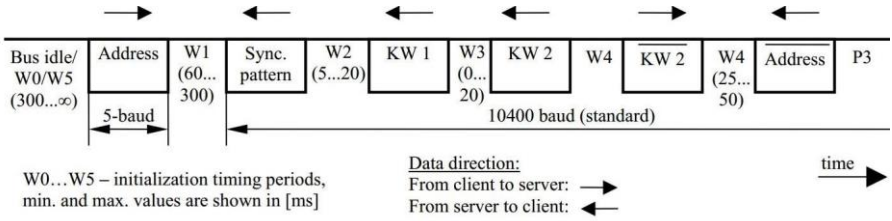


Рисунок 5.1 – Процес ініціалізації сигналів у протоколах ISO 9141 та ISO 14230.

Вигляд команди ініціалізації для протоколу KWP2000 (Keyword Protocol 2000) (ISO-14230) наведено на рис. 5.2.

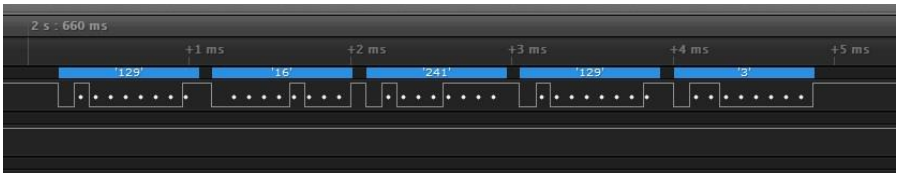


Рисунок 5.2 – Команда ініціалізації

5.2 Команди протоколу OBD-II

Кожен з OBD-II кодів несправностей складається з п’яти символів: букви і чотирьох цифр.

Нумерація помилок OBD-II [2]:

– P00xx – Контроль системи сумішеутворення та системи додаткового зниження токсичності викидів;

– P01xx – Контроль системи сумішеутворення;

– P02xx – Контроль системи сумішеутворення;

– P03xx – Система запалення і система контролю пропусків загоряння;

– P04xx – Допоміжні системи контролю емісії;

– P05xx – Контроль швидкісті автомобіля, системи холостого ходу та інших систем;

– P06xx – Блоки керування ECM / PCM / TCM та інші системи;

– P07xx – Трансмісія;

– P08xx – Трансмісія;

– P09xx – Трансмісія;

– P10xx – Коди, що встановлені виробником. Залежать від марки автомобіля;

– P20xx – Коди, що встановлені виробником. Залежать від марки автомобіля;

– B00xx – Кузов (подушки безпеки, центральний замок, електросклопідіймачі).

– C00xx – Шасі (ABS противопробуксовочна система, ESP, TCS-Traction Control System Система курсової стійкості);

– U10xx –Міжблокова шина обміну даних (CAN-bus) (CAN-II);

– U25xx – Міжблокова шина обміну даних (CAN-bus) (CAN-II).

Символи xx посилаються на окремі несправності всередині кожної підсистеми.

5.3 Порядок виконання

5.3.1 Отримати пристрої: адаптер K-line та логічний аналізатор.

5.3.2 Провести підключення пристроїв до автомобілю за допомогою резистивного подільника напруги.

5.3.3 Провести підключення за допомогою різних програм діагностування.

5.3.4 Провести запис відповідей ЕБК автомобіля на пакети ініціалізації.

5.3.5 Провести аналіз отриманних даних.

5.4 Контрольні питання

5.4.1 Як перепризначити COM-порт для програми діагностики?

5.4.2 Який рівень сигналу у К-лінії? Як відбувається передача повідомлення?

5.4.3 Які вимоги до рівня «логічного нуля»?

5.4.4 Формат StartCommunication service у kwp2000.

5.4.5 Що таке Timing для діагностики? У чому вони вимірюються? Які бувають?

5.4.6 Що таке режими PID? Наведіть приклад.

5.5 Література

5.5.1 Fast Init через K-Line (практика прикладных программ). URL: http://malykh.blogspot.com/2014/01/sdl-kwp/ (date of access 03.01.2024).

5.5.2 OBD-II Trouble Codes Home. URL: https://www.obd-codes.com/ (date of access 03.01.2024).

6. Лабораторна робота № 6

ПЕРЕДАЧА ТА ПРИЙОМ ПОВІДОМЛЕНЬ У CAN-ШИНІ

Мета роботи: отримати знання щодо змісту пакетів обміну інформацією у CAN- шині та принципами побудови CAN-мережі.

6.1 Загальні відомості

CAN характерізується наступними загальними властивостями:

– кожному повідомленню (а не пристрою) встановлюється свій пріоритет;

– гарантована величина паузи між двома актами обміну;

– гнучкість конфігурування та можливість модернізації системи;

– широкомовний прийом повідомлень з синхронізацією у часі;

– допустимість декількох ведучих пристроїв у мережі;

– здатність пошуку і знаходження помилок та сигналізація у разі їхньої наявності;

– автоматичне повторення передачі повідомлень, що доставлені з помилкою, відразу, як тільки мережа стане вільною;

– автоматичне розпізнавання збою та відмов із можливістю автоматичного відключення несправних модулів.

CAN мережа призначена для комунікації так званих вузлів. Кожен вузол складається з двох складових. Це власне CAN контролер, який забезпечує взаємодію з мережею і реалізує протокол, та мікропроцесор (CPU).

CAN контроллери з'єднуються за допомогою шини, яка має як мінімум два дроти Can_h і Can_l, якими передаються сигнали за допомогою спеціалізованих прийомо-передавачів ІМС. Крім того, ІМС прийомо-передавачів реалізують додаткові сервісні функції:

– вбудована схема обмеження струму захищає виходи передавачів від пошкодження при можливих замиканнях ліній CAN_H і CAN_L із колами живлення, а також від короткочасного підвищення напруги на цих лініях;

– внутрішній тепловий захист;

– режим зниженного енергоспоживання, у якому прийомник продовжує повідомляти контролеру про стан мережі для того, щоб при виявленні у мережі інформаційних сигналів він міг ввімкнути прийомо-передавачі у нормальний режим роботи.

Найбільш широке розповсюдження отримали два типа прийомо-передавачів (трансиверів):

– "High Speed" прийомо-передавачі (ISO 11898-2),

– "Fault Tolerant" прийомо-передавачі.

Трансивери, виконані відповідно до стандарту "High-speed" (ISO11898-2), найбільш прості, дешеві і дають можливість передавати дані з швидкістю до 1 Мбіт/с. "Fault-tolerant" приймачі (нечутливі до пошкоджень на шині) дозволяють побудувати високонадійну малоспоживаючу мережу зі швидкостями передачі даних не вище 125 кбіт/с.

Фізичний рівень (Physical Layer) протоколу CAN визначає опір кабелю, рівень електричних сигналів в мережі і тому подібне. Існує декілька фізичних рівнів протоколу CAN (ISO 11898, ISO 11519, SAE J2411). У переважній більшості випадків використовується фізичний рівень CAN, визначений в стандарті ISO 11898.

ISO 11898 як середовище передачі визначає двопровідну диференціальну лінію з імпедансом (термінатори) 120 Ом (допускається коливання імпедансу в межах від 108 Ом до 132 Ом [1].

Термінування відповідає наступним цілям:

1. Усунення відбиття сигналу від кінця шини.

2. Перевірка коректності рівнів постійного струму (DC).

Максимальна швидкість мережі CAN у відповідності з протоколом дорівнює 1 Мбіт/с. При швидкості в 1 Мбіт/с максимальна довжина кабелю дорівнює приблизно 40 м. Обмеження на довжину кабелю пов'язане з кінцевою швидкістю поширення сигналу і механізмом побітового арбітражу (під час арбітражу всі вузли мережі повинні отримувати поточний біт передачі одночасно, таким чином сигнал повинен встигнути поширитися по всьому кабелю за одиничний відлік часу в мережі.

Співвідношення між швидкістю передачі і максимальною довжиною кабелю швидкість передачі/максимальна довжина мережі: 1000 Кбіт/сек 40 метрів 500 Кбіт/сек 100 метрів 250 Кбіт/сек 200 метрів 125 Кбіт/сек 500 метрів 10 Кбіт/сек 6 кілометрів.

Логічний нуль реєструється, коли на лінії CAN_H сигнал вищий, ніж на лінії CAN_L, а логічна одиниця – у разі, коли сигнали CAN_H і CAN_L однакові (відрізняються менш ніж на 0.5 В). Використання такої диференціальної схеми передачі робить можливою роботу CAN мережі в дуже складних зовнішніх умовах. Логічний нуль називається домінантним бітом, а логічна одиниця – рецесивним. Ці назви відображають пріоритет логічної одиниці і нуля на шині CAN.

При одночасній передачі в шину логічного нуля і одиниці, на шині буде зареєстрований лише логічний нуль (домінантний сигнал), а логічна одиниця буде пригнічена (рецесивний сигнал) [1].

Кабель витої пари в мережі CAN повинен мати загальний (третій) дріт; на обох кінцях витої пари мають бути узгоджуючі резистори, опір яких дорівнює хвильовому опору кабелю. Максимальна довжина кабелю складає 1 км. Для збільшення довжини, кількості вузлів або гальванічної розв'язки можуть бути використані повторювачі інтерфейсу, мережеві мости та шлюзи.

Вита пара може бути в екрані або без, залежно від електромагнітної обстановки. Топологія мережі має бути шинною, максимальна довжина відведення від шини при швидкості передачі 1 Мбіт/с не повинна перевищувати 30 см. Довжину відведення можна розрахувати за формулою:

$$l = \frac{T_\Phi}{50 нс / м},$$

де T_Φ - тривалість переднього фронту передавача.

Основні вимоги до лінії передачі та її характеристик близькі до RS-485, проте в передавачах CAN є режим керування тривалістю фронтів імпульсів. Керування виконується шляхом заряджання ємностей затворів вихідних транзисторів від джерел струму, при цьому величина струму задається зовнішнім резистором. Збільшення тривалості фронту дозволяє знизити вимоги до узгодження лінії на низьких частотах, збільшити довжину відведень і послабити випромінювання від електромагнітних завад.

Виводи "землі" всіх передавачів мережі мають бути з’єднані (якщо інтерфейси гальванічно не ізольовані). При цьому різниця потенціалів між виводами заземлень не повинна перевищувати 2 В.

Гальванічна ізоляція рекомендується при довжині лінії більше 200 м, але не є обов'язковою вимогою стандарту.

6.2 Зміст повідомлень

Швидкодія CAN мережі (до 1 Mbit/s) досягається завдяки механізму недеструктивного арбітражу шини за допомогою порівняння біт конкуруючих повідомлень. Тобто якщо станеться так, що одночасно почнуть передачу декілька контролерів, то кожен з них порівнює біт, який збирається передати на шину, з бітом, який намагається передати на шину конкуруючий контролер. Якщо значення цих бітів рівні, обидва контролери намагаються передати наступний біт. Так відбувається до тих пір, поки значення переданих бітів не виявляться різними. Тоді контролер, який передавав логічний нуль (пріоритетніший сигнал) продовжуватиме передачу, а інший (інші) контролер перерве свою передачу до того часу, доки шина знову не звільниться. Звичайно, якщо шина в даний момент зайнята, то контролер не почне передачу до моменту її звільнення.

Ця специфікація CAN виходить із припущення, що все CAN контролери приймають сигнали з шини одночасно. Тобто в один і той же час один і той же біт приймається всіма контролерами в мережі. З одного боку, такий стан речей робить можливим побітовий арбітраж, а з іншого боку – обмежує довжину CAN bus.

Сигнал поширюється по CAN bus з величезною, але кінцевою, швидкістю, і для правильної роботи CAN потрібно, щоб всі контролери сприйняли його майже одночасно. Майже, тому що кожен контролер приймає біт в діапазоні певного проміжку часу. Таким чином, чим вища швидкість передачі даних, тим менша довжина CAN bus можлива.

Структура формату передачи даних.

Дані щодо CAN мережі пересилаються у вигляді окремих кадрів стандартного формату. Найбільш важливими полями є поле ідентифікатора (identifier) і дані (data).

Ідентифікатор служить унікальним ім'ям для типу повідомлення і визначає те, ким буде прийнято і як буде інтерпретовано наступне за ним поле даних. Чому саме (арифметично) дорівнює це число, в загальному випадку не має значення. Така контекстна адресація відрізняється рядом переваг для мереж невеликого масштабу. Вона забезпечує максимально можливу простоту модернізації. Оскільки децентралізовані контролери ніяк не зв'язані між собою логічно, додавання нового елементу в систему ніяк не вплине на поведінку всіх інших.

Цікавішим представляється використання ідентифікаторів як основних інструментів, що використовуються в процедурі вирішення колізій. У CAN як основний критерій для розбору колізій, для ухвалення рішення, кому віддати ефір, використовується пріоритет повідомлень. Якщо одночасні декілька станцій почали передачу, і при цьому сталася колізія, відбувається суперпозиція ідентифікаторів, що були передані.

Ідентифікатори послідовно, побітно (bitwise), починаючи зі старшого, накладаються один на одного, і в їхній "боротьбі" виграє той, у кого менше арифметичне значення ідентифікатора, а значить, вище пріоритет. Домінантний "нуль" подавить одиниці, і в будь-якому випадку до кінця передачі поля ідентифікатора воно дорівнюватиме більш пріоритетному значенню. Таким чином, система дозволяє на рівні проектування (і визначення ідентіфікатора) для будь-якого повідомлення в системі заздалегідь обумовити його приоритетність в обслуговуванні.

Пріоритетність повідомлення, таким чином, визначається значенням ідентифікатора. Пріоритет тим більший, чим ідентифікатор менший. Як правило, контролер дозволяє задавати лише ці два поля. Останні поля використовуються для передачі специфічних даних, необхідних для функціонування CAN.

Формати кадру.

Дані у CAN передаються короткими повідомленнями-кадрами стандартного формату. У CAN існують чотири типа повідомлень:

– Data Frame;
– Remote Frame;
– Error Frame;
– Overload Frame.

Data Frame – це тип повідомлення, що використовується найчастіше. Він складається з поля арбітражу (arbitration field), що визначає пріоритет повідомлення у разі, коли два або більше вузли одночасно намагаються передати дані в мережу.

Поле арбитражу складається, у свою чергу, з:

– для стандарту CAN-2.0A, 11-бітного ідентифікатора + 1 біт RTR (retransmit);

– для стандарту CAN-2.0B, 29-бітного ідентифікатора + 1 біт RTR (retransmit).

Слід ще раз відзначити, що поле ідентифікатора, недивлячись на свою назву, ніяк не ідентифікує сам по собі ні вузол в мережі, ні вміст поля даних.

Для Data кадру біт RTR завжди виставлено в логічний нуль (домінантний сигнал). Поле даних (data field) містить від 0 до 8 байт даних, поле CRC (CRC field), що містить 15-бітову контрольну суму повідомлення, яка використовується для виявлення помилок, слот підтвердження (Acknowledgement Slot) (1 біт).

Кожен CAN-контролер, який правильно прийняв повідомлення, посилає біт підтвердження в мережу. Вузол, який послав повідомлення, слухає цей біт, і у випадку, якщо підтвердження не прийшло, повторює передачу. В разі прийому слота підтвердження вузол, що передає, може бути упевнений лише в тому, що хоча би один з вузлів в мережі правильно прийняв його повідомлення.

Remote Frame – це Data Frame без поля даних і з виставленим бітом RTR (1 - рецесивні біт). Основне призначення Remote кадру – це ініціація одним із вузлів мережі передачі в мережу даних іншим вузлом. Така схема дозволяє зменшити сумарний трафік мережі. Проте, на практиці Remote Frame зараз використовується рідко (наприклад, в Devicenet Remote Frame зовсім не використовується).

Error Frame – це повідомлення, яке явно порушує формат повідомлення CAN. Передача такого повідомлення призводить до

того, що всі вузли мережі реєструють помилку формату CAN-кадру, і у свою чергу автоматично передають в мережу Error Frame. Результатом цього процесу є автоматична повторна передача даних в мережу передавальним вузлом. Error Frame складається з поля Error Flag, яке складається з 6 біт однакового значення (і таким чином Error frame порушує перевірку Bit Stuffing, див. нижчий), і поля Error Delimiter, що складається з 8 рецесивних бітів. Error Delimiter дає можливість іншим вузлам мережі, виявивши Error Frame, надіслати в мережу свій Error Flag.

Overload Frame – повторює структуру і логіку роботи Error кадру, з тією різницею, що він використовується переобтяженим вузлом, який в даний момент не може обробити повідомлення, що надходить, і тому просить за допомогою overload-кадру про повторну передачу даних. В даний час overload-кадр практично не використовується.

Механізм обробки похибок.

Надійність CAN мережі визначається також механізмами виявлення помилок. Стандарт CAN визначає наступні методи виявлення помилок в мережі CAN:

- Check Bit monitoring;
- Bit stuffing;
- Frame check;
- ACKnowledgement Check;
- Check CRC.

Check Bit monitoring – кожен вузол під час передачі бітів в мережу порівнює значення біта, що він передає, зі значенням біта, яке з'являється на шині. Якщо ці значення не збігаються, то вузол генерує помилку Bit Error. Природно, що під час арбітражу на шині (передача поля арбітражу в шину) цей механізм перевірки помилок вимикається.

Bit stuffing – коли вузол передає послідовно в шину 5 біт з однаковим значенням, то він додає шостий біт з протилежним значенням. Приймаючі вузли цей додатковий біт видаляють. Якщо вузол виявляє на шині більше 5 послідовних біт з однаковим значенням, то він генерує помилку Stuff Error.

Frame Check – деякі частини CAN-повідомлення мають однакове значення у всіх типах повідомлень. Тобто протокол CAN точно визначає які рівні напруги і коли повинні з'являтися на шині.

Якщо формат повідомлень порушується, то вузли генерують помилку Form Error.

ACKnowledgement Check – кожен вузол, отримавши правильне повідомлення з мережі, надсилає в мережу домінантний (0) біт. Якщо ж цього не відбувається, то передавальний вузол реєструє помилку Acknowledgement Error.

CRC Check – кожне повідомлення CAN містить CRC суму, і кожен приймаючий вузол підраховує значення CRC для кожного отриманого повідомлення. Якщо підраховане значення CRC суми не збігається зі значенням CRC в тілі повідомлення, що приймає вузол генерує помилку CRC Error.

Кожен вузол мережі CAN, під час роботи намагається виявити одну з п'яти можливих помилок. Якщо помилка виявлена, вузол передає в мережу Error Frame, руйнуючи тим самим весь поточний трафік мережі (передачу і прийом поточного повідомлення). Всі останні вузли виявляють Error Frame і приймають відповідні дії (скидають прийняте повідомлення).

Окрім цього, кожен вузол веде два лічильника помилок:

- **Transmit Error Counter** (лічильник помилок передачі);
- **Receive Error Counter** (лічильник помилок прийому).

Ці лічильники збільшуються або зменшуються у відповідності з декількома правилами. Самі правила управління лічильниками помилок досить складні, але зводяться до простого принципу: помилка передачі приводить до збільшення Transmit Error лічильника на 8, помилка прийому збільшує лічильник Receive Error на 1, будь-яка коректна передача/прийом повідомлення зменшують відповідний лічильник на 1. Ці правила призводять до того, що лічильник помилок передачі вузла, що передає, збільшується швидше, ніж лічильник помилок прийому приймаючих вузлів. Це правило відповідає припущенню про велику вірогідність того, що джерелом помилок є передавальний вузол.

Кожен вузол CAN мережі може знаходиться в одному з трьох станів. Коли вузол стартує він знаходиться в стані Error Active. Коли, значення хоч би одного з двох лічильників помилок перевищує межу 127, вузол переходить в стан Error Passive. Коли значення хоч би одного з двох лічильників перевищує межу 255, вузол переходить в стан Bus Off.

Вузол, що знаходиться в стані Error Active, в разі виявлення помилки на шині передає в мережу Active Error Flags. Active Error Flags складається з 6 домінантних біт, тому всі вузли його реєструють.

Вузол в стані Passive Error передає в мережу Passive Error Flags при виявленні помилки в мережі. Passive Error Flags складається з 6 рецесивних біт, тому останні вузли мережі його не помічають, і Passive Error Flags лише приводить до збільшення Error лічильника вузла. Вузол в стані Bus Off нічого не передає в мережу (не лише Error кадри, але взагалі будь-які інші). Формат кадру в сеті CAN наведено на рис. 6.1 та 6.2 [2].

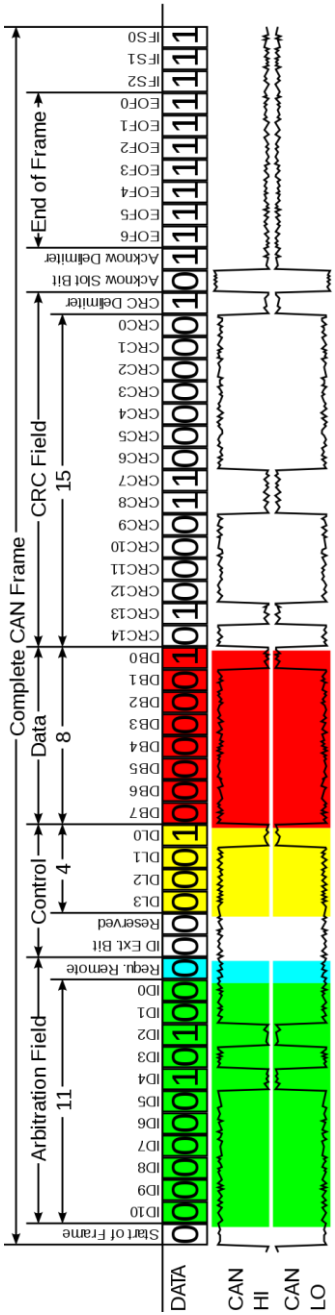
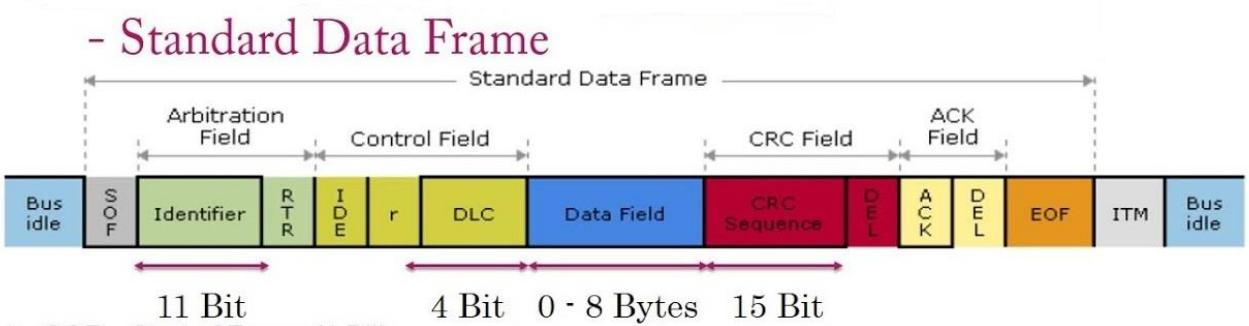


Рисунок 6.1 – Базовий формат CAN-фрейму



- SOF – Start of Frame (1 Bit)
- Identifier – A message identifier sets the priority of the data frame.
- RTR – Remote Transmission Request, defines the frame type (data frame or remote frame) (1 Bit).
- Control Field – User defined Functions.
- DLC – Data Length Code (4 Bits).
- Data Field – User defined Data (0 to 8 Bytes)
- CRC Field – cyclic redundancy check for Error (Data corruption) detection.
- ACK Field – Receivers Acknowledgement.
- EOF – End of Frame (7 Bits)

Рисунок 6.2 – Базовий формат CAN-фрейму

Стандарт CAN не регламентує, яким чином конкретні програми передаватимуть специфічні для себе дані мережею CAN. Таким чином, виникає потреба у використанні якого-небудь протоколу верхнього рівня. Можна придумати свій протокол, який дозволяв би додаткам працювати з CAN мережею просто і зручно, але навряд чи варто витрачати на це сили, якщо вже існує безліч високорівневих протоколів на основі CAN технології. Причому це відкриті протоколи, тобто можна отримати вже готові специфікації і навіть брати участь в подальшому розвитку даних систем.

Тому з початком масового випуску CAN-компонентів і широкого поширення CAN-додатків низкою незалежних компаній і некомерційних асоціацій в області систем промислової автоматизації, транспорту тощо проводилася (і продовжується до цього дня) робота зі створення і стандартизації специфікацій протоколів верхнього рівня HLP (Higher Level Protocol) для CAN-мереж.

Утилізація поля арбітражу і поля даних, і розподіл адресів вузлів, ідентифікаторів повідомлень і пріоритетів в мережі є предметом розглядів так званих протоколів високого рівня (HLP - Higher Layer Protocols).

Назва HLP відображає той факт, що протокол CAN описує лише два нижні рівні еталонної мережевої моделі ISO/OSI, а останні рівні описуються протоколами HLP.

До теперішнього часу відомо вже більше чотирьох десятків CAN HLP. Серед подібного різноманіття CAN HLP найбільше поширення, особливо в системах промислової автоматизації, отримали чотири, які підтримуються асоціацією CIA, а саме [1]:

– CAL/ CANopen,
– CAN Kingdom,
– DeviceNet та
– SDS (Smart Distributed System).

CANopen – поширений стандарт прикладного рівня. Для спрощення вживання стандарту вводяться декілька специфічних для CANopen понять. Всі функціональні можливості прикладного рівня розподіляються між так званими сервісами (елементами послуг). Програмні додатки взаємодіють між собою шляхом виклику відповідних сервісів прикладного рівня.

Сервіси обмінюються даними з рівними їм (одноранговими) сервісами через CAN-мережу за допомогою визначенного протоколу. Цей протокол описується в специфікації протоколу сервісу.

Вводиться поняття сервісного примітиву, який є засобом (мовна конструкція), за допомогою якого програмне застосування взаємодіє з прикладним рівнем. У CANopen існує чотири різні примітиви:

– *запит додатку* до прикладного рівня, який публікується додатком для виклику сервісу;

– *індикація*, що публікується прикладним рівнем додатка, щоб сповістити про внутрішні події, які виявлено прикладним рівнем;

– *відповідь*, що публікується для прикладного рівня, щоб відповісти на індикацію, що була отримана раніше;

– *підтвердження*, що публікується прикладним рівнем для додатку, щоб відзвітувати про результати раніше опублікованого запиту.

Сервіси також діляться на декілька *типів сервісів*:

– *локальний сервіс,* що виконує запит додатку без взаємодії з іншими сервісами того ж рангу;

– *непідтверджений сервіс,* що долучає до виконання запиту один або кілька однорангових сервісів. Додаток надсилає запит до локального сервісу. Цей запит передається іншому сервісу (або сервісам) того ж рангу;

– *підтверджений сервис,* що може залучити лише один сервісний об'єкт того ж рангу. Додаток видає запит до його локального сервісу. Цей запит передається сервісу того ж рангу, який передає його іншому додатку як індикацію. Інший додаток видає відповідь, що передається вихідному сервісу, який передає його як підтвердження запрошуючому додатку;

– *ініційований провайдером сервіс* – залучає тільки локальний сервіс [3].

Структуру фрейму CANopen показано на рис. 6.3 [4,5]. Групування параметрів CANopen наведено у табл. 6.1 [6,7].

Рисунок 6.3 – Структуру фрейму CANopen

Таблиця 6.1 – Діапазони груп параметрів у словнику об’єктів

Index (HEX)	Object
0000	Reserved
0001 – 001F	Static data types
0020 – 003F	Complex data types
0040 – 005F	Manufacturer specific data types
0060 – 007F	Device profile specific static data types
0080 – 009F	Device profile specific complex data types
00A0 – 0FFF	Reserved for further use
1000 – 1FFF	Communication profile area
2000 – 5FFF	Manufacturer profile specific area
6000 – 9FFF	Standardized device profile area
A000 – FFFF	Reserved for further use

6.3 Порядок виконання

6.3.1 Отримати пакет даних з CAN-шини.
6.3.2 Провести аналіз отриманих даних.
6.3.3 Оформити звіт.

6.4 Контрольні питання

6.4.1 Що таке побітовий арбітраж?
6.4.2 Які види передатчиків використовуються з CAN-шиною і чому?
6.4.3 Що таке «термінатори»? Навіщо вони потрібні? Де використовуються?

6.4.4 Яка максимальна довжина лінії CAN? Як вона розраховується?

6.4.5 Що таке домінантний та рецесивний сигнали у шині CAN?

6.4.6 Навіщо повідомленню «ідентифікатор»? Який вид він має?

6.4.7 Типи повідомлень у CAN-шині.

6.4.8 Як генерується помилка “Bit Error”?

6.4.9 Чому термінатори звичайно дорівнюють 120? Чи завжди це доречно?

6.4.10 Навіщо потрібні лічильники помилок?

6.4.11 Скільки провідників у кабелі потрібно для реалізації CAN-шини?

6.4.12 Що таке «Error Frame», як він формується?

6.4.13 Що відбувається, коли вузол на CAN-шині намагається відіслати повідомлення і при цьому на шині він єдиний?

6.5 Література

6.5.1 Физический уровень канала CAN. URL: http://itt-ltd.com/reference/ref_can/ (date of access 03.01.2024).

6.5.2 CAN Protocol Frame Format | CAN Protocol Basics URL: https://www.youtube.com/watch?v=nyef9xoZjqc/ (date of access 03.01.2024).

6.5.3 CANopen Explained - A Simple Intro [2022] URL: https://www.csselectronics.com/screen/page/canopen-tutorial-simple-intro. (date of access 03.01.2024).

6.5.4 CANopen protocol. URL: https://www.typhoon-hil.com/documentation/typhoon-hil-schematic-editor-library/References/canopen_protocol/ (date of access 03.01.2024).

6.5.5 Возможности CAN протокола. URL: https://www.cta.ru/cms/f/326789.pdf/ (date of access 03.01.2024).

6.5.6 CANopen and Kvaser. URL: https://www.kvaser.com/about-can/higher-layer-protocols/canopen/ (date of access 03.01.2024).

6.5.7 Промышленные сети URL: https://www.electrocentr.com.ua/files/documentation/SE/TechLibrary/SolutionGuide/9-Fieldbus.pdf/ (date of access 03.01.2024).

7. Лабораторна робота № 7

РОБОТА ІНТЕРФЕЙСУ RS-485 В УМОВАХ ЗАВАД

Мета роботи: отримати знання щодо змісту пакетів обміну інформацією інтерфейсу RS-485 та принципів побудови мережі з використанням RS-485.

7.1 Загальні відомості

Під позначеннями RS-232, RS-422 і RS-485 розуміють інтерфейси для цифрової передачі даних. Стандарт RS-232 відомий також як звичайний COM-порт комп'ютера або послідовний порт (хоча послідовним портом також можна вважати Ethernet, Firewire і USB). Інтерфейси RS-422 і RS-485 широко застосовуються в промисловості для з'єднання різного устаткування.

Інтерфейс RS-232 (TIA/EIA-232) призначений для організації прийому-передачі даних між передавачем або терміналом (англ. Data Terminal Equipment, DTE) і приймачем або комунікаційним устаткуванням (англ. Data Communications Equipment, DCE) за схемою крапка-крапка.

Швидкість роботи RS-232 залежить від відстані між пристроями, зазвичай на відстані 15 м швидкість рівна 9600 біт/с. На мінімальній відстані швидкість зазвичай рівна 115,2 кбіт/с, але є устаткування, яке підтримує швидкість до 921,6 кбіт/с.

Інтерфейс RS-232 працює в дуплексному режимі, що дозволяє передавати і приймати інформацію одночасно, тому що використовуються різні лінії для прийому і передачі. У цьому полягає відмінність від напівдуплексного режиму, коли використовується одна лінія зв'язку для прийому і передачі даних, що накладає обмеження на одночасну роботу, тому в напівдуплексному режимі в один проміжок часу можливий або прийом, або передача інформації.

Інформація щодо інтерфейсу RS-232 передається у цифровому вигляді логічними 0 та 1.

Логічній «1» (MARK) відповідає напруга в діапазоні від −3 до −15 В.

Логічному «0» (SPACE) відповідає напруга у діапазоні від +3 до +15 В (рис. 7.1).

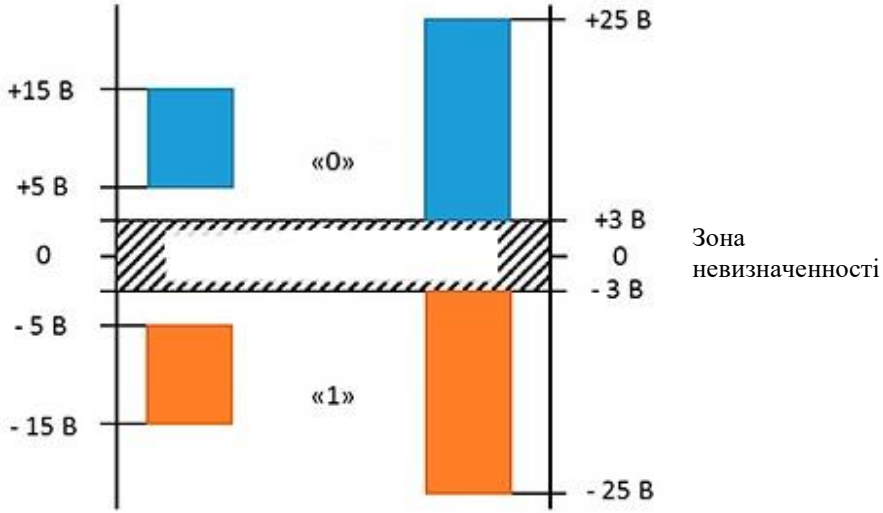


Рисунок 7.1 – Рівні сигналів RS-232

На додаток до двох ліній прийому і передачі, на RS-232 є спеціальні лінії для апаратного управління потоком і інших функцій.

Інтерфейс RS-422 схожий на RS -232, оскільки дозволяє одночасно відправляти і приймати повідомлення окремими лініями (повний дуплекс), але використовує для цього диференціальний сигнал, тобто різницю потенціалів між провідниками A і B.

Швидкість передачі даних в RS-422 залежить від відстані і може змінюватися в межах від 10 кбіт/с (1200 м) до 10 Мбіт/с (10 м).

В мережі RS-422 може бути лише один передавальний пристрій і до 10 приймаючих пристроїв.

Лінія RS-422 представляє собою 4 провідника для прийому-передачі даних (2 скручених провода для передачі і 2 скручених провода для прийому) та одним загальним провідником землі GND.

Скручування провідників (вита пара) між собою дозволяє позбавитися від наведень і перешкод, тому що наведення однаково діє на обидва дроти, а інформація визначається різницею потенціалів між провідниками A і B однієї лінії.

Напруга на лініях передачі даних може знаходиться в діапазоні від -6 В до +6 В.

Логічному 0 відповідає різниця між А та В більша, ніж +0,2 В.

Логічній 1 відповідає різниця між А та В менша, ніж -0,2 В.

У промисловості найчастіше використовується інтерфейс RS-485 (EIA-485), тому що в RS-485 використовується багатоточкова топологія, що дозволяє підключити декілька приймачів і передавачів.

7.2 Підключення до RS-485

Інтерфейс RS-485 схожий на RS-422 тим, що також використовує диференційний сигнал для передачі даних.

Існують два типа RS-485:

– RS-485 з 2 контактами, працює у режимі напівдуплекс;

– RS-485 з 4 контактами, працює у режимі повний дуплекс.

У режимі повний дуплекс можна одночасно приймати і передавати дані, а в режимі напівдуплекс або передавати, або приймати.

В одному сегменті мережі RS-485 може бути до 32 пристроїв, але за допомогою додаткових повторювачів і підсилювачів сигналів – до 256 пристроїв. В один проміжок часу активним може бути лише один передавач.

Швидкість роботи також залежить від довжини лінії і може досягати 10 Мбіт/с на 10 м.

Напруга на лініях знаходиться в діапазоні від−7 В до +12 В. Стандарт RS-485 не визначає конкретного типа роз'єму, але часто це клемна колодка або роз'єм DB9.

Розпіновка роз'єму RS-485 залежить від виробника пристрою і вказується в документації на нього.

Підключення до RS-485 наведено на рис. 7.2 [1].

Вхідний опір приймача зазвичай складає 12 кОм. Оскільки потужність передавача не безмежна, це створює обмеження на кількість приймачів, підключених до лінії. Згідно специфікації RS-485 з врахуванням резисторів узгодження, передавач може вести до 32 приймачів. Проте є ряд мікросхем з підвищеним вхідним опором, що дозволяє підключити до лінії значно більше 32 пристроїв.

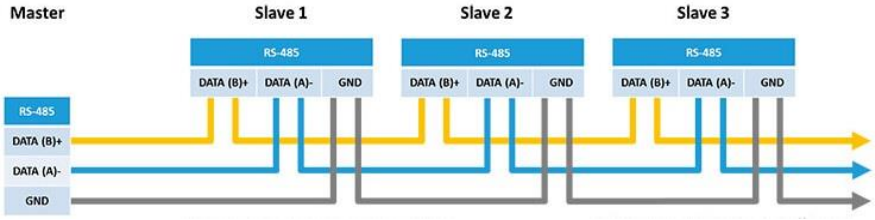


Рисунок 7.2 – Підключення до RS-485

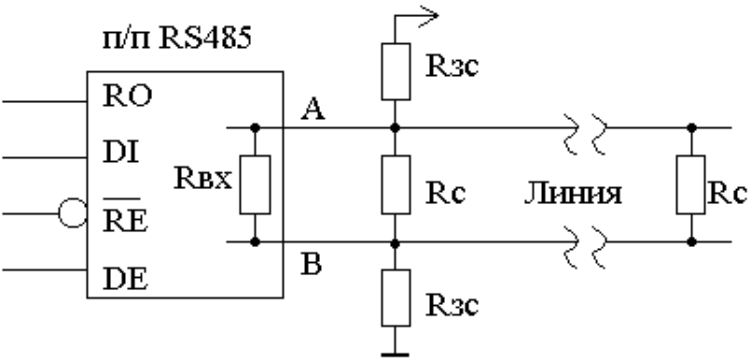
Максимальна швидкість зв'язку по специфікації RS-485 може досягати 10 Мбод/сек. Максимальна відстань - 1200 м. Якщо необхідно організувати зв'язок на відстані більш ніж 1200 м або підключити більше пристроїв, чим допускає здатність навантаження передавача - застосовують спеціальні повторітелі (репітери).

Лінія зв'язку має бути одним кабелем витої пари. До цього кабелю приєднуються всі приймачі і передавачі. Відстань від лінії до мікросхем інтерфейсу RS-485 має бути як можна коротшою, оскільки довгі відгалуження вносять розузгодження і викликають віддзеркалення.

В обидва найбільш віддалених кінця кабелю (Zв=120 Ом) включають резистори узгодження Rt по 120 Ом (0.25 Вт).

Приймачі більшості мікросхем RS-485 мають пороговий діапазон розпізнавання сигналу на входах A-B - ±200мВ. Якщо |Uab| менше порогового (близько 0), то на виході приймача RO можуть бути довільні логічні рівні із-за несинфазної перешкоди. Таке може статися або при від'єднанні приймача від лінії, або за відсутності в лінії активних передавачів, коли ніхто не задає рівень.

Щоб в цих ситуаціях уникнути видачі помилкових сигналів на приймач UART, необхідно на входах A-B гарантувати різницю потенціалів Uab > +200 мВ. Цей зсув за відсутності вхідних сигналів забезпечує на виході приймача логічну "1", підтримуючи, таким чином, рівень стопового біта. Добитися цього просто: прямий вхід (A) слід підтягнути до живлення, а інверсний (B) - до "землі". Виходить подільник напруги:



Rвх – вхідний опір приймача (звичайно 12 кОм);
Rс – узгоджуючи резистори (120 Ом);
Rзс - резисторы захисного зміщення.

Рисунок 7.3 – Захисний подільник напруги

Величини опорів для резисторів захисного зсуву (Rзс) неважко розрахувати за допомогою подільника напруги. Необхідно забезпечити Uab > 200 мВ. Напруга живлення – 5 В. Опір середнього плеча – 120 Ом//120 Ом//12 кОм на кожен приймач - приблизно 57 Ом (для 10 приймачів). Таким чином, виходить приблизно по 650 Ом на кожен з двох Rзс. Для зсуву із запасом - опір Rзс має бути менше 650 Ом. Традиційно ставлять 560 Ом.

Зверніть увагу: у розрахунку номінала Rзс враховується навантаження. Якщо на лінії є багато приймачів, то номінал Rзс має бути меншим. У довгих лініях передачі необхідно так само враховувати опір витої пари Для довгої лінії краще ставити два комплекти підтягуючих резисторів в обидва віддалені кінці поряд з термінаторами.

При роботі з напівдуплексним інтерфейсом RS-485 (прийом і передача по одній парі провідників з розділенням у часі) можна незважати, що UART контроллера - повнодуплексний, тобто приймає і передає незалежно і одночасно.

Зазвичай під час роботи приймача RS-485 на передачу, вихід приймача RO переводиться в третій стан і ніжка RX контроллера (приймач UART) "повисає в повітрі". В результаті, під час передачі на приймачі UART замість рівня стопового біта ("1") виявиться

невідомий сигнал, і будь-яка перешкода буде прийнята за вхідний сигнал (рис. 7.4).

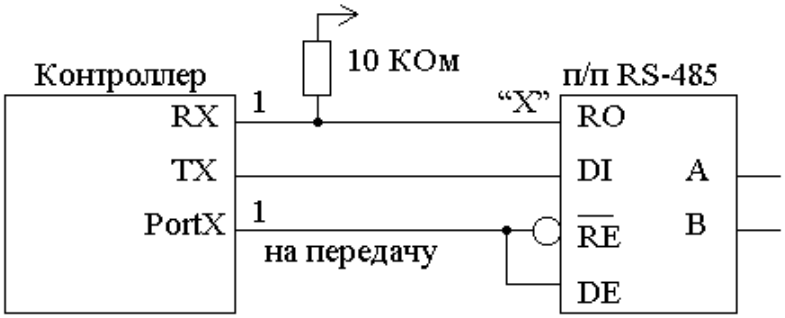


Рисунок 7.4 – Встановлення логичної одиниці на порту RX

Тому потрібно або на час передачі відключати приймач UART (через керуючий регістр), або підтягувати RX до одиниці. В деяких мікроконтроллерів це можна зробити програмно - активувати вбудовані підтяжки портів.

7.3 Порядок виконання

7.3.1 З'єднати USB-RS485 перетворювачі у мережу.
7.3.2 Підключити перетворювачі USB-RS485 до ПК.
7.3.3 Встановити на ПК термінальну програму.
7.3.4 Провести передачу даних у мережі.
7.3.5 Написати скрипт для постановки завад у мережі.
7.3.6 Перевірити роботу написаного скрипту.
7.3.7 Спробувати передавати повідомлення при роботі скріпту.

7.4 Контрольні питання

7.4.1 Чому у інтерфейсу rs485 висока стійкість до синфазної завади?

7.4.2 В чому відмінність інтерфейсів rs422 та rs485?

7.4.3 Що таке поріговий диапазон розпізнавання сигналу? Як він вимірюється?

7.4.4 Який мінімальний поріговий опір приймача у лінії rs485?

7.4.5 Яка максимальна довжина лінії зв'язку для rs485?

7.4.6 Який мінімальний поріговий опір приймача у лінії rs422?

7.4.7 Навіщо потрібні узгоджуючі резистори? Яки їх параметри? Де вони встановлюються?

7.4.8 Що таке захісне зміщення? Навіщо воно потрібно? Як його реалізувати?

7.4.9 Як виконується захист від розсинхронізування кадрів?

7.4.10 Навіщо потрібен "тайм-аут"?

7.4.11 Навіщо потрібен дренажний провід?

7.4.12 Навіщо потрібні та як працюють індуктивні фільтри?

7.4.13 Перелічіть видмінності у інтерфейсів rs485 та CAN?

7.4.14 Як виконати зв'язок по односигнальній линії?

7.5 Література

7.5.1 В чем отличия интерфейсов RS-232, RS-422 и RS-485? URL: https://ipc2u.ru/articles/prostye-resheniya/otlichiya-interfeysov-rs-232-rs-422-rs-485/ (date of access 03.01.2024).