Микропроцессорные системы.

Цифровые интерфейсы МК. Часть I. UART. RS-232, RS-422, RS-485.

Хохлов С.А.

МГТУ им. Н.Э. Баумана

11 апреля 2025 г.

Интерфейс и протокол

Интерфейс

- Это физический или логический способ взаимодействия между компонентами.
- Он определяет:
  - Аппаратные средства: линии связи (провода), уровни напряжения (например, TTL, LVDS), разъемы.
  - Логическую структуру: как данные передаются (последовательно, параллельно), синхронизацию (тактовый сигнал), роли устройств (мастер/ведомый).
- Примеры интерфейсов:
  - SPI (Serial Peripheral Interface) — аппаратная реализация.
  - I²C (Inter-Integrated Circuit) — сочетает аппаратные линии (SDA, SCL) и логические правила.
  - USB (Universal Serial Bus) — стандарт разъемов, уровни сигналов, питание.

Протокол

- Это набор правил, определяющих формат и порядок обмена данными.
- Он описывает:
  - Структуру сообщений: стартовые/стоповые биты, адреса, контрольные суммы.
  - Последовательность действий: как устройства инициируют связь, обрабатывают ошибки, завершают передачу.
  - Семантику данных: что означают конкретные биты или байты (например, команды, статусы).
- Примеры протоколов:
  - HTTP (Hypertext Transfer Protocol) — правила запросов и ответов в вебе.
  - Modbus — формат сообщений для промышленных систем.
  - TCP (Transmission Control Protocol) — управление потоком данных в сетях.

Почему, например, SPI называют и интерфейсом, и протоколом?

- SPI (Serial Peripheral Interface) — это интерфейс, который включает в себя протокол.
- Как интерфейс:
  - Определяет физические линии (MOSI, MISO, SCK, SS).
  - Задает уровни напряжения (например, 3.3 В или 5 В).
  - Устанавливает роли (мастер инициирует обмен, ведомый отвечает).
- Как протокол:
  - Определяет порядок передачи битов (MSB-first или LSB-first).
  - Задает режимы синхронизации (CPOL и CPHA).
  - Описывает, как данные интерпретируются (например, команда → данные).
- Таким образом, SPI объединяет аппаратную часть (интерфейс) и логическую часть (протокол). Это и вызывает смешение терминов.

Другие примеры употребления

1. “UART”:
   - Интерфейс: линии TX и RX, уровни напряжения (TTL, CMOS, RS-232).
   - Протокол: стартовый/стоповый биты, скорость (например, 9600 бод), формат данных (8N1).
2. I²C:
   - Интерфейс: линии SDA и SCL, открытый сток, подтягивающие резисторы.
   - Протокол: адресация устройств, ACK/NACK, арбитраж шины.
3. “Ethernet”:
   - Интерфейс: разъем RJ-45, витая пара, уровни сигналов.
   - Протокол: MAC-адресация, фреймы (Ethernet II, 802.3), коллизии (CSMA/CD).

Ключевое отличие

- Интерфейс отвечает на вопрос: «Как подключить устройства?» (физически и логически).
- Протокол отвечает на вопрос: «Как они обмениваются данными?» (формат, правила, семантика).

Протоколы UART и USART

UART (Universal Asynchronous Receiver-Transmitter)

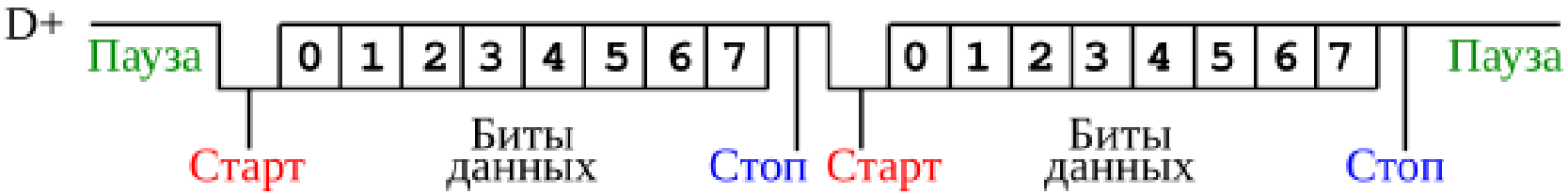
- UART – это асинхронный последовательный протокол передачи данных.
- Он работает без тактового сигнала (clock),
- используя только два основных сигнала: TX (передача) и RX (прием).
- Данные передаются бит за битом,
  - начиная со стартового бита,
  - за которым следуют биты данных,
  - необязательный бит четности и
  - один или два стоповых бита.

Особенности UART

- Асинхронная передача:
  - не требует отдельного сигнала тактирования.
- Формат кадра:
  - 1 стартовый бит, 5–9 бит данных, 0 или 1 бит четности, 1 или 2 стоповых бита.
- Скорость передачи:
  - задается битрейтом (300; 600; 1200; 2400; 4800; 9600; 19200; 38400; 57600; 115200; 230400; 460800; 921600 бод).
- Дуплексность:
  - полудуплексный или полный дуплекс (два провода TX/RX).
- Применение:
  - широко используется в микроконтроллерах, GPS-модулях, Bluetooth, RS-232.

Временная диаграмма UART

- Диаграмма передачи байта 0b01001101 (7 бит, четность отсутствует, 1 стоповый бит):



- UART — это асинхронный протокол, что означает отсутствие отдельного сигнала тактирования. Однако для правильной передачи и приема данных необходимо обеспечить битовую и кадровую синхронизацию.

Кадровая синхронизация UART

- Кадровая синхронизация в UART означает корректное определение начала и конца каждого передаваемого кадра (байта).
- Достигается за счет специальных битов, которые окружают полезные данные.

Структура кадра UART

Старт	Данные	Четность (опц.)	Стоп
1 бит	5–9 бит	0 или 1 бит	1–2 бита

Этапы кадровой синхронизации UART

1. Обнаружение стартового бита (0)
   - Линия UART обычно находится в 1 (логическая “1” - высокий уровень).
   - Появление 0 указывает на начало нового кадра.
   - Приемник запускает таймер для отсчета битов.
2. Прием данных
   - Приемник фиксирует значения битов в заранее установленные моменты времени.
3. Опциональная проверка четности
   - Если используется бит четности, приемник проверяет его на соответствие ожиданиям.
4. Обнаружение стопового бита (1)
   - Наличие стопового бита (1) подтверждает завершение кадра.
   - Если вместо 1 обнаружено 0, значит произошла ошибка кадрирования.

Контроль четности UART

- even parity -> число единиц четное
- odd parity -> число единиц нечетное
- при обнаружении ошибки чётности логика UART выставляет признак ошибки в слове состояния.

Битовая синхронизация UART I

- Битовая синхронизация обеспечивает правильное определение границ каждого передаваемого бита.
- Это особенно важно, поскольку передатчик и приемник работают на одной заранее установленной скорости (битрейте), но без общего тактового сигнала.

Выборка данных UART на стороне приемника

Как достигается битовая синхронизация?

- Фиксированный битрейт (например, 9600, 115200 бит/с) должен быть одинаковым у передатчика и приемника.
- Приемник использует стартовый бит (0) для начала отсчета времени середины (!) каждого бита, чтобы захватить значение строго в этот момент.
- Для формирования временных битовых интервалов передающий и приёмный UART имеют свой собственный источник точного времени (тактирования).
- Точность этих источников должна быть такой, чтобы сумма погрешностей (приёмника и передатчика) установки временного интервала от начала стартового импульса до середины стопового импульса не превышала половины (а лучше — четверти) битового интервала.

Ошибки синхронизации UART

Ошибки битовой синхронизации

- Неравные битрейты передатчика и приемника → биты “съезжают”, данные искажаются.
- Шум и дрожание сигнала → прием может происходить на неверных временных интервалах.

Ошибки кадровой синхронизации

- Ошибка кадрирования — стоповый бит не 1, а 0.
  - Возможные причины: разное число битов данных, сбой в передаче.
- Ошибка четности — при использовании четности не совпадает ожидаемое и фактическое значение.
- Ошибки синхронизации могут привести к потере данных, поэтому важен контроль тактовых частот и чистота сигнала.

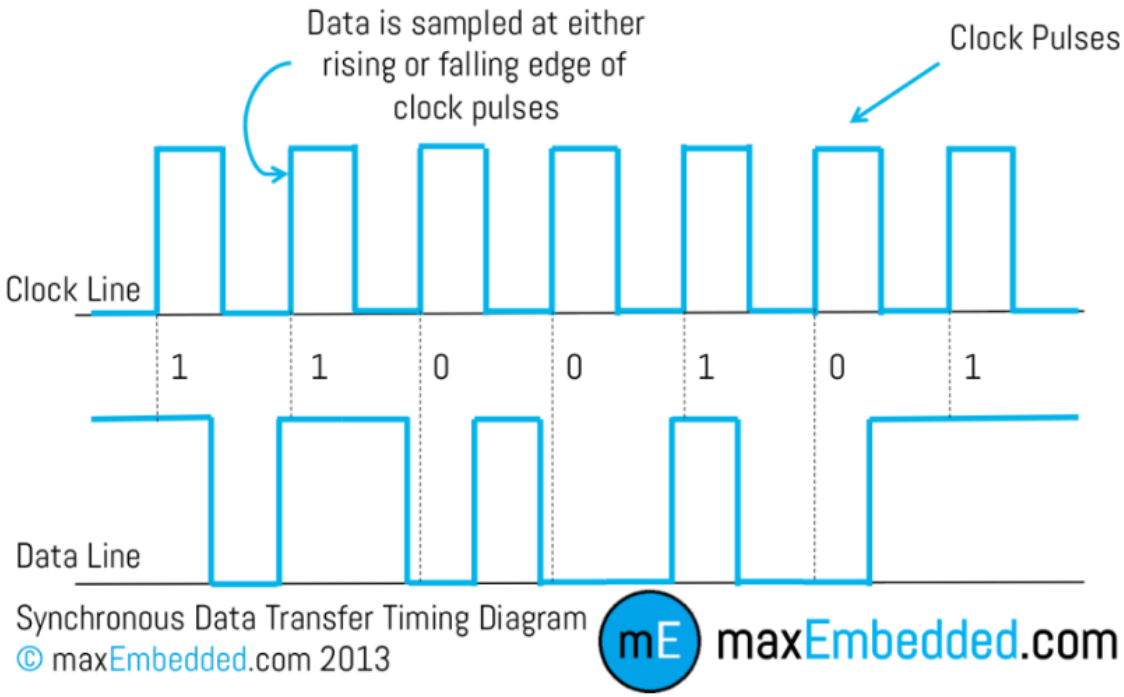
USART (Universal Synchronous/Asynchronous Receiver-Transmitter)

- USART – более универсальный протокол, поддерживающий как асинхронную, так и синхронную передачу данных.
- В синхронном режиме он использует тактовый сигнал (Clock) для синхронизации передатчика и приемника.

Особенности USART

- Асинхронный режим:
  - работает как обычный UART.
- Синхронный режим:
  - добавляется линия тактирования (SCK), синхронизация происходит по фронту или спаду тактового сигнала.
- Выше скорость передачи:
  - так как нет необходимости в стартовых и стоповых битах.
- Применение:
  - используется в высокоскоростных системах, SPI-подключениях, некоторых видах RS-485.

Временная диаграмма USART в синхронном режиме



- Данные передаются по фронту или спаду тактового сигнала.
- Скорость может быть выше, чем у UART, за счет синхронизации.

Режимы работы USART для МК семейства STM32H7

- UART
- SPI master/slave
- Smartcard
- IrDA SIR
- LIN
- Modbus

Схемы подключения

1. UART (Только TX и RX)

```
MK 1              MK 2
TX ------------> RX
RX <------------ TX
GND ------------ GND
```

(Общий GND обязателен!)

2. USART в синхронном режиме

```
MK 1              MK 2
TX ------------> RX
RX <------------ TX
SCK ------------> SCK
GND ------------ GND
```

Выводы

- UART – прост в использовании, не требует тактового сигнала, но скорость ограничена из-за стоповых битов.
- USART – более универсален, поддерживает синхронный режим, что позволяет передавать данные быстрее.

Примеры работы с UART

Схема UART-порта MK AVR и примеры работы

- Например, даташит ATmega328P, раздел 19
  - Схемы:
    - Figure 19-1. USART Block Diagram
    - Figure 19-2. Clock Generation Logic, Block Diagram
  - Примеры работы:
    - 19.5 USART Initialization
    - 19.6 Data Transmission – The USART Transmitter
    - 19.7 Data Reception – The USART Receiver

AVR

Инициализация UART

```
1  #include <avr/io.h>
2
3  #define F_CPU 16000000UL
4  #define BAUD 9600
5  #define UBRR_VAL ((F_CPU / 16 / BAUD) - 1)
6
7  void UART_Init() {
8      // Установка скорости передачи
9      UBRR0H = (uint8_t)(UBRR_VAL >> 8);
10     UBRR0L = (uint8_t)UBRR_VAL;
11
12     // Включение передатчика и приемника
13     UCSR0B = (1 << TXEN0) | (1 << RXEN0);
14
15     // Формат кадра: 8 бит данных, 1 стоповый бит,
16     // без контроля четности
17     UCSR0C = (1 << UCSZ01) | (1 << UCSZ00);
18 }
```

Передача данных (TX)

```
1  void UART_SendChar(uint8_t data) {
2      // Ждем, пока буфер передатчика не освободится
3      while (!(UCSR0A & (1 << UDRE0)));
4
5      // Записываем данные в буфер
6      UDR0 = data;
7  }
8
9  void UART_SendString(const char* str) {
10     while (*str) {
11         UART_SendChar(*str++);
12     }
13 }
```

Прием данных (RX)

```
1  uint8_t UART_ReceiveChar() {
2      // Ждем, пока данные не будут приняты
3      while (!(UCSR0A & (1 << RXC0)));
4
5      // Возвращаем принятый байт
6      return UDR0;
7  }
```

Пример использования

```
1  int main() {
2      UART_Init();
3      UART_SendString("Hello, UART!\r\n");
4
5      while (1) {
6          uint8_t received = UART_ReceiveChar();
7          UART_SendChar(received); // Эхо-ответ
8      }
9  }
```

STM32 (STM32F103, без HAL)

Инициализация UART (USART1, PA9-TX, PA10-RX)

```
#include "stm32f1xx.h"

#define BAUD 9600
#define APB2_CLK 72000000UL // Частота APB2 (72 МГц для STM32F103)

void UART_Init() {
    // Включение тактирования GPIOA и USART1
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN | RCC_APB2ENR_USART1EN;

    // Настройка PA9 (TX) как альтернативной функции Push-Pull
    GPIOA->CRH &= ~(GPIO_CRH_CNF9 | GPIO_CRH_MODE9);
    GPIOA->CRH |= GPIO_CRH_CNF9_1 | GPIO_CRH_MODE9;

    // Настройка PA10 (RX) как вход с подтяжкой
    GPIOA->CRH &= ~(GPIO_CRH_CNF10 | GPIO_CRH_MODE10);
    GPIOA->CRH |= GPIO_CRH_CNF10_0;
    GPIOA->ODR |= GPIO_ODR_ODR10; // Подтяжка к VCC

    // Расчет UBRR (для oversampling=16)
    uint32_t ubrr = (APB2_CLK / (16 * BAUD)) - 1;
    USART1->BRR = ubrr;

    // Включение передатчика и приемника
    USART1->CR1 |= USART_CR1_TE | USART_CR1_RE | USART_CR1_UE;
}
```

Передача данных (TX)

```
1  void UART_SendChar(uint8_t data) {
2      // Ждем, пока буфер передатчика не освободится
3      while (!(USART1->SR & USART_SR_TXE));
4
5      // Записываем данные в буфер
6      USART1->DR = data;
7  }
8
9  void UART_SendString(const char* str) {
10     while (*str) {
11         UART_SendChar(*str++);
12     }
13 }
```

Прием данных (RX)

```
1  uint8_t UART_ReceiveChar() {
2      // Ждем, пока данные не будут приняты
3      while (!(USART1->SR & USART_SR_RXNE));
4
5      // Возвращаем принятый байт
6      return USART1->DR;
7  }
```

Пример использования

```
1  int main() {
2      UART_Init();
3      UART_SendString("Hello, STM32 UART!\r\n");
4
5      while (1) {
6          uint8_t received = UART_ReceiveChar();
7          UART_SendChar(received); // Эхо-ответ
8      }
9  }
```

STM32 (STM32F1/F4, с использованием HAL-библиотеки (STM32CubeMX/STM32CubeIDE))

Код инициализации (HAL)

```c
#include "stm32f1xx_hal.h"  // или stm32f4xx_hal.h для F4

UART_HandleTypeDef huart2;

void SystemClock_Config(void);
void Error_Handler(void);

void UART2_Init(void) {
    huart2.Instance = USART2;
    huart2.Init.BaudRate = 9600;
    huart2.Init.WordLength = UART_WORDLENGTH_8B;
    huart2.Init.StopBits = UART_STOPBITS_1;
    huart2.Init.Parity = UART_PARITY_NONE;
    huart2.Init.Mode = UART_MODE_TX_RX;
    huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart2.Init.OverSampling = UART_OVERSAMPLING_16;

    if (HAL_UART_Init(&huart2) != HAL_OK) {
        Error_Handler();
    }
}

int main(void) {
    HAL_Init();
    SystemClock_Config();
    UART2_Init();

    while (1) {
        // Примеры передачи и приёма (см. ниже)
    }
}
```

Настройка в STM32CubeMX

1. Включите USART2 в режиме Asynchronous.
2. Настройте пины:
   - TX: PA2 (Alternate Function Push-Pull)
   - RX: PA3 (Input Pull-Up)
3. Установите параметры:
   - Baud Rate: 9600
   - Word Length: 8 бит
   - Stop Bits: 1
   - Parity: None
   - Hardware Flow Control: Disabled

- см. подробнее:
  - Уроки STM32. Работа с UART через библиотеку HAL. Инициализация интерфейса и передача данных в блокирующем режиме. Отладка программ с помощью UART. Функция sprintf. | Оборудование, технологии, разработки

Передача данных (TX)

```
1  //Отправка одного символа
2  void UART_SendChar(uint8_t data) {
3      HAL_UART_Transmit(&huart2, &data, 1,
4          HAL_MAX_DELAY);
5  }
1  //Отправка строки
2  void UART_SendString(const char* str) {
3      HAL_UART_Transmit(&huart2, (uint8_t*)str, strlen(str),
4          HAL_MAX_DELAY);
5  }
```

Пример использования

```
1  UART_SendString("Hello, STM32 UART!\r\n");
2  UART_SendChar('A');
```

Прием данных (RX)

```
//Прием одного символа (блокирующий режим)
uint8_t UART_ReceiveChar(void) {
    uint8_t data;
    HAL_UART_Receive(&huart2, &data, 1,
        HAL_MAX_DELAY);
    return data;
}
//Прием строки (с таймаутом)
void UART_ReceiveString(uint8_t* buffer, uint16_t length) {
    // Таймаут 1 сек
    HAL_UART_Receive(&huart2, buffer, length, 1000);
}
```

Пример использования

```
uint8_t received = UART_ReceiveChar();
if (received == 'A') {
    UART_SendString("Got 'A'\r\n");
}
```

ESP32 (на ESP-IDF)

Инициализация UART (UART1, GPIO9-TX, GPIO10-RX)

```c
#include "driver/uart.h"
#include "driver/gpio.h"

#define UART_PORT UART_NUM_1
#define BAUD_RATE 9600
#define TX_PIN GPIO_NUM_9
#define RX_PIN GPIO_NUM_10

void UART_Init() {
    uart_config_t uart_config = {
        .baud_rate = BAUD_RATE,
        .data_bits = UART_DATA_8_BITS,
        .parity = UART_PARITY_DISABLE,
        .stop_bits = UART_STOP_BITS_1,
        .flow_ctrl = UART_HW_FLOWCTRL_DISABLE
    };

    // Настройка UART1
    uart_param_config(UART_PORT, &uart_config);
    uart_set_pin(UART_PORT, TX_PIN, RX_PIN, UART_PIN_NO_CHANGE, UART_PIN_NO_CHANGE);
    uart_driver_install(UART_PORT, 1024, 0, 0, NULL, 0);
}
```

Передача данных (TX)

```
1 void UART_SendString(const char* str) {
2     uart_write_bytes(UART_PORT, str, strlen(str));
3 }
```

Прием данных (RX)

```
1 uint8_t UART_ReceiveChar() {
2     uint8_t data;
3     uart_read_bytes(UART_PORT, &data, 1,
4         portMAX_DELAY);
5     return data;
6 }
```

Пример использования

```
1 void app_main() {
2     UART_Init();
3     UART_SendString("Hello, ESP32 UART!\r\n");
4 
5     while (1) {
6         uint8_t received = UART_ReceiveChar();
7         UART_SendChar(received); // Эхо-ответ
8     }
9 }
```

RS-232, RS-485, RS-422

RS-232, COM-порт, UART – разве не одно и то же?

- RS-232, RS-485, RS-422 — стандарты физического уровня, определяющие электрические характеристики сигналов (уровни напряжения, разъемы, защиту от помех).
- Важно: если говорить строго, то UART — это протокол передачи данных, а RS-xxx — физическая реализация этого протокола.
- COM-порт, serial port – это название интерфейса стандарта RS-232 в IBM PC (“так исторически сложилось”).

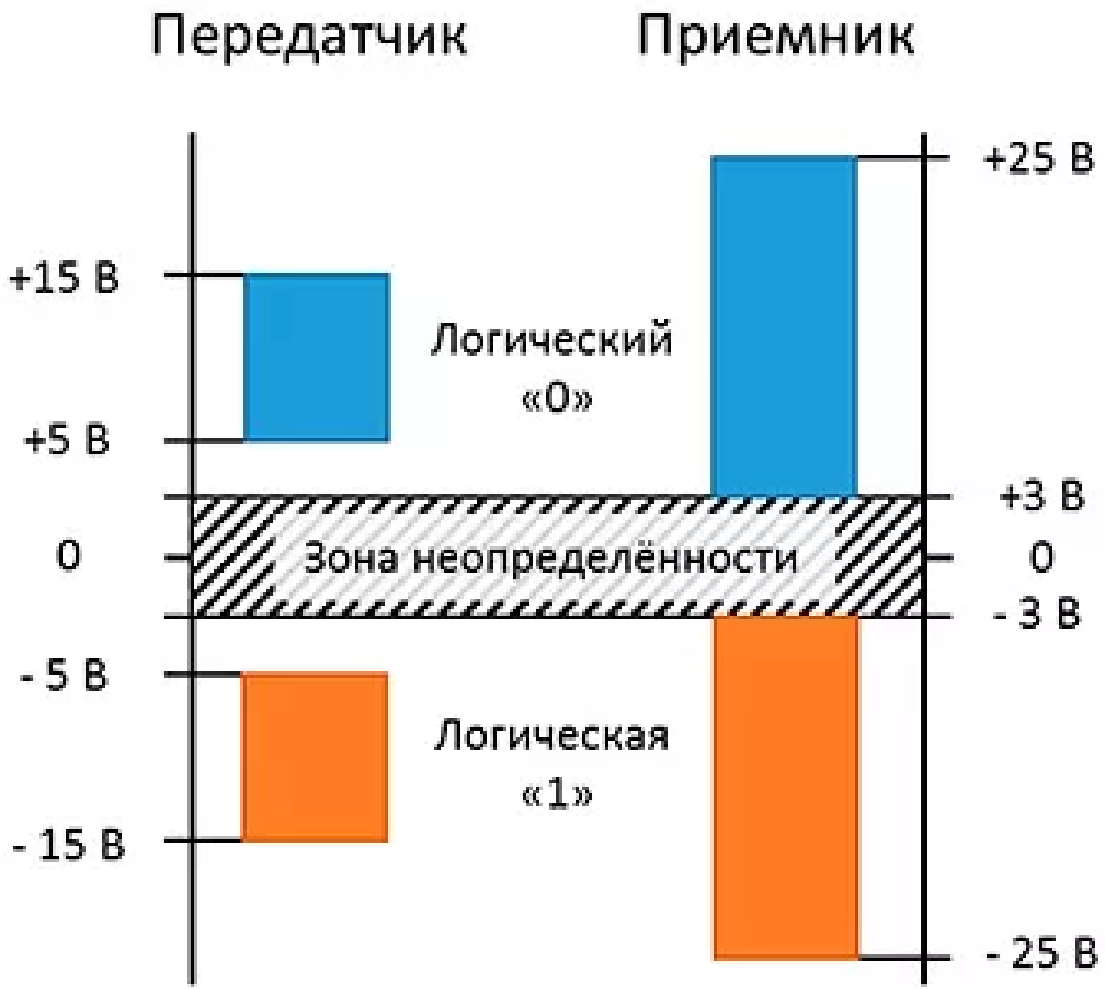
Стандарт RS-232

- Основные параметры
  - Уровни напряжения на передатчике:
    - Логический 0: +5 В до +15 В.
    - Логический 1: -5 В до -15 В.
  - Топология:
    - Точка-точка.
  - Скорость:
    - До 115.2 кбит/с (практически).
  - Расстояние:
    - До 15 м.
- Применение
  - Связь между ПК и периферией (мыши, модемы).
- Разъемы:
  - DB9, DB25.

Недостатки RS-232

- Низкая помехоустойчивость.
- Ограниченная скорость и расстояние.

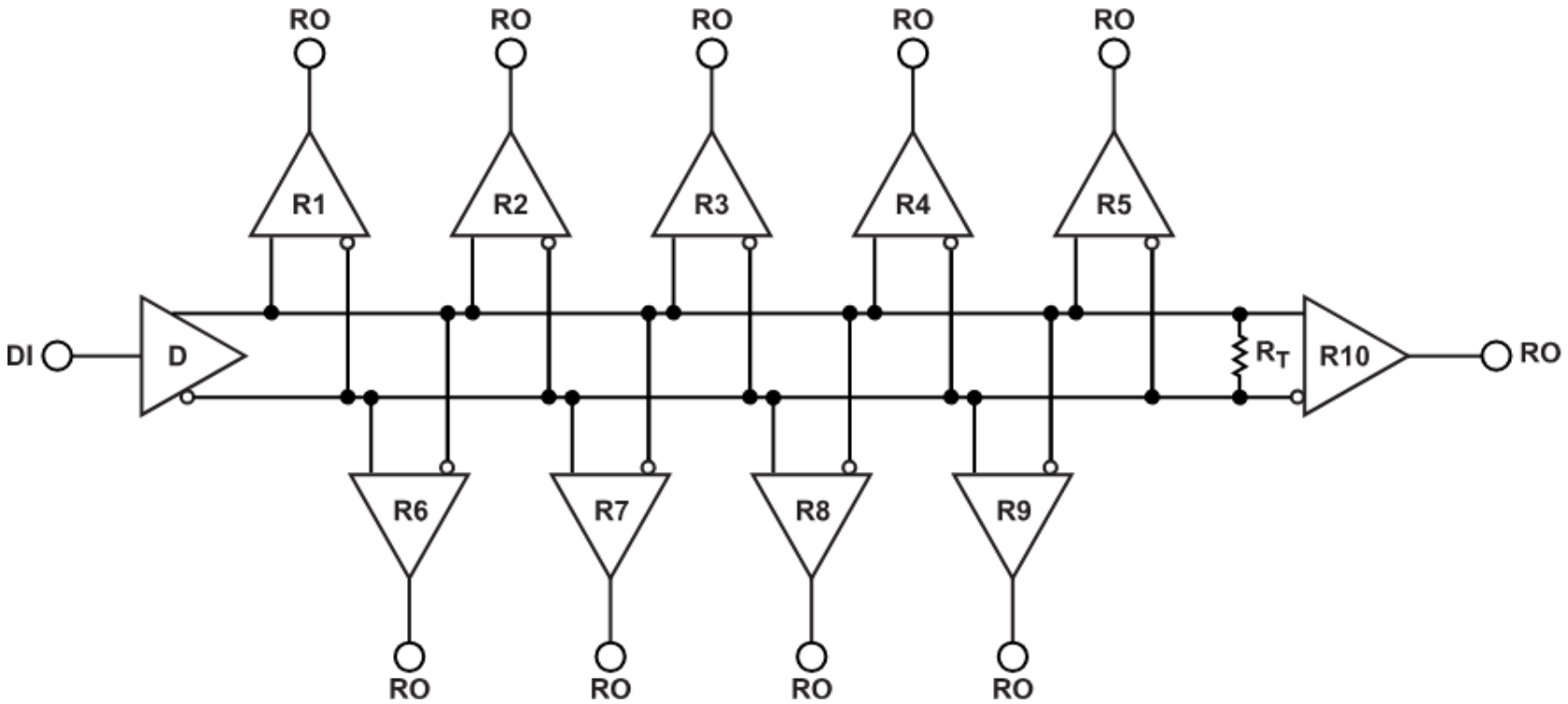
Уровни напряжений RS-232



Стандарт RS-422

- Основные параметры
  - Уровни напряжения на передатчике:
    - Дифференциальные сигналы (-2..-6В, +2..+6В).
  - Диапазон дифф. входного напряжения на приёмнике
    - –7 В … +7 В
  - Порог переключения приёмника
    - ≥ |200 мВ|
  - Синфазное входное напряжение (Vcm)
    - –7 В … +7 В макс, типичное ~2.0–3.0 В
  - Топология:
    - Точка-точка.
    - Многоточечная (1 передатчик, до 10 приемников).
  - Скорость:
    - До 10 Мбит/с (на коротких расстояниях).
  - Расстояние:
    - До 1200 м (на низких скоростях, 10-60кбит/с).
  - Применение
    - Промышленные сети с высокой помехозащищенностью.

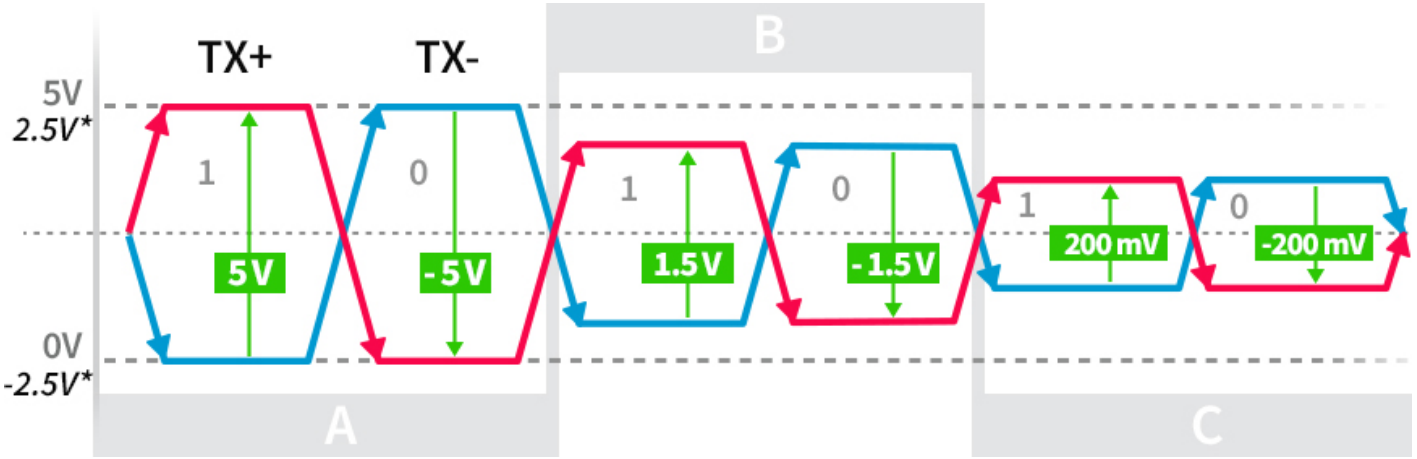
Типичная схема сети RS-422



Стандарт RS-485

- Основные параметры
  - Уровни напряжения на передатчике:
    - Дифференциальные сигналы (-1.5..-6В, +1.5..+6В).
  - Диапазон дифф. входного напряжения на приёмнике
    - –7 В … +12 В
  - Порог переключения приёмника
    - $\geq$ |200 мВ|
  - Синфазное входное напряжение (Vcm)
    - –7 В … +12 В макс, типичное ~1.5–3.0 В
  - В стандарте RS-485 для передачи и приёма данных используется одна или две витых пары проводов, сопровождаемые экранирующей оплеткой или общим проводом
  - Топология:
    - Многоточечная (до 32 устройств на шину).
  - Скорость:
    - До 10 Мбит/с.
  - Расстояние:
    - До 1200 м.
  - Двухпроводной и четырехпроводной варианты
    - двухпроводной – полудуплекс
    - четырехпроводной – полный дуплекс
  - Применение
    - Промышленные сети (Modbus, Profibus).
    - Системы автоматизации зданий.

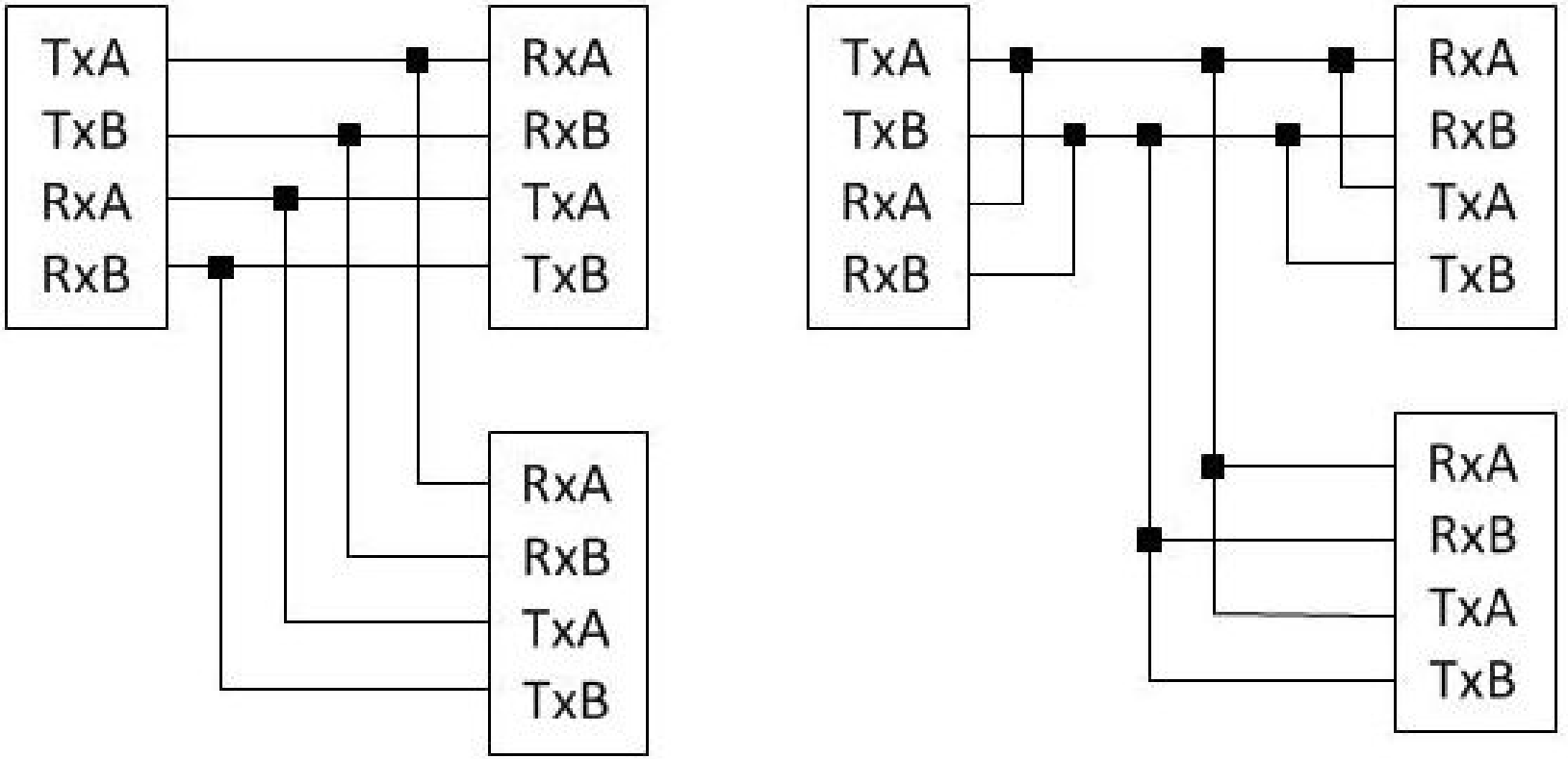
Уровни напряжений RS-485



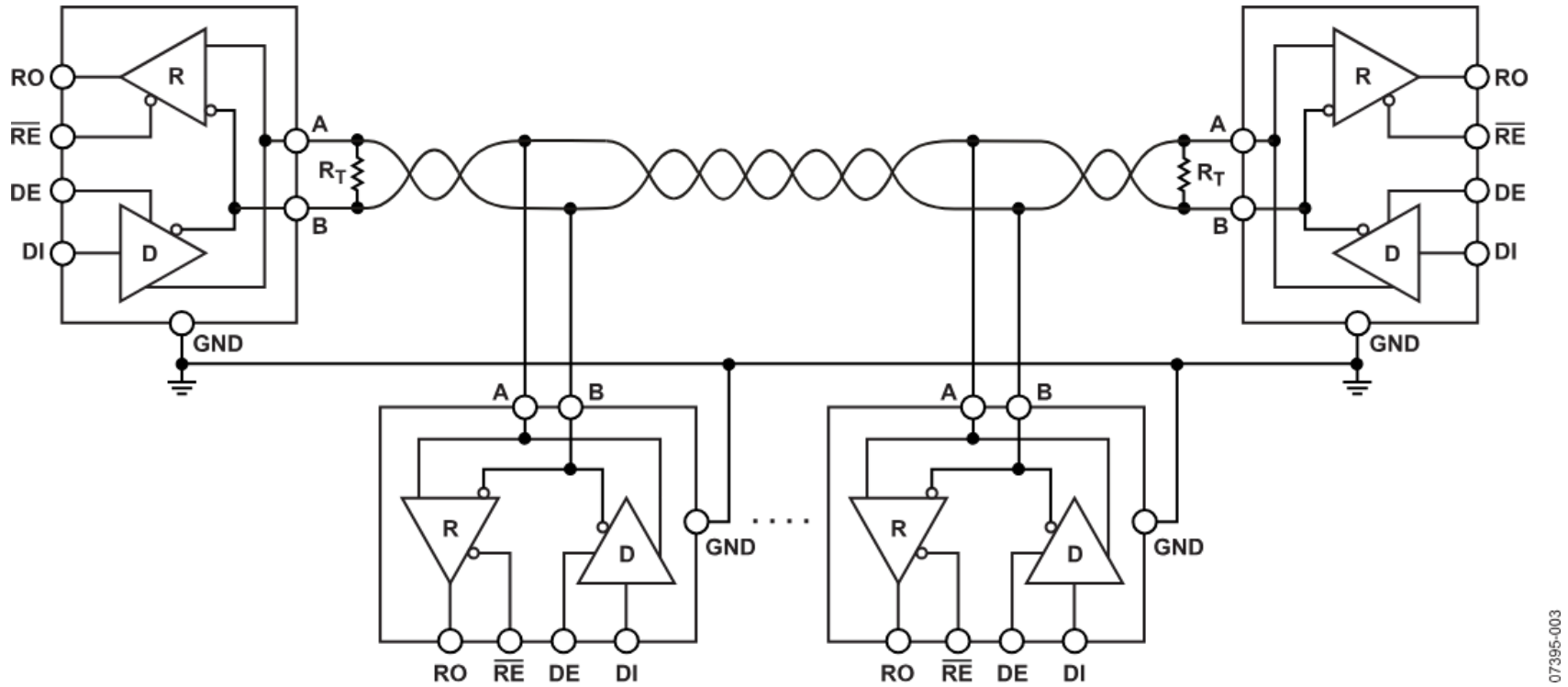
* 2.5 V .. -2.5V относительно синфазного напряжения(common mode voltage)

- A = Номинальное напряжение (5 V)
- B = Минимальное выходное напряжение на передатчике (1.5 V)
- C = Минимальное входное напряжение на приемнике (200mV)

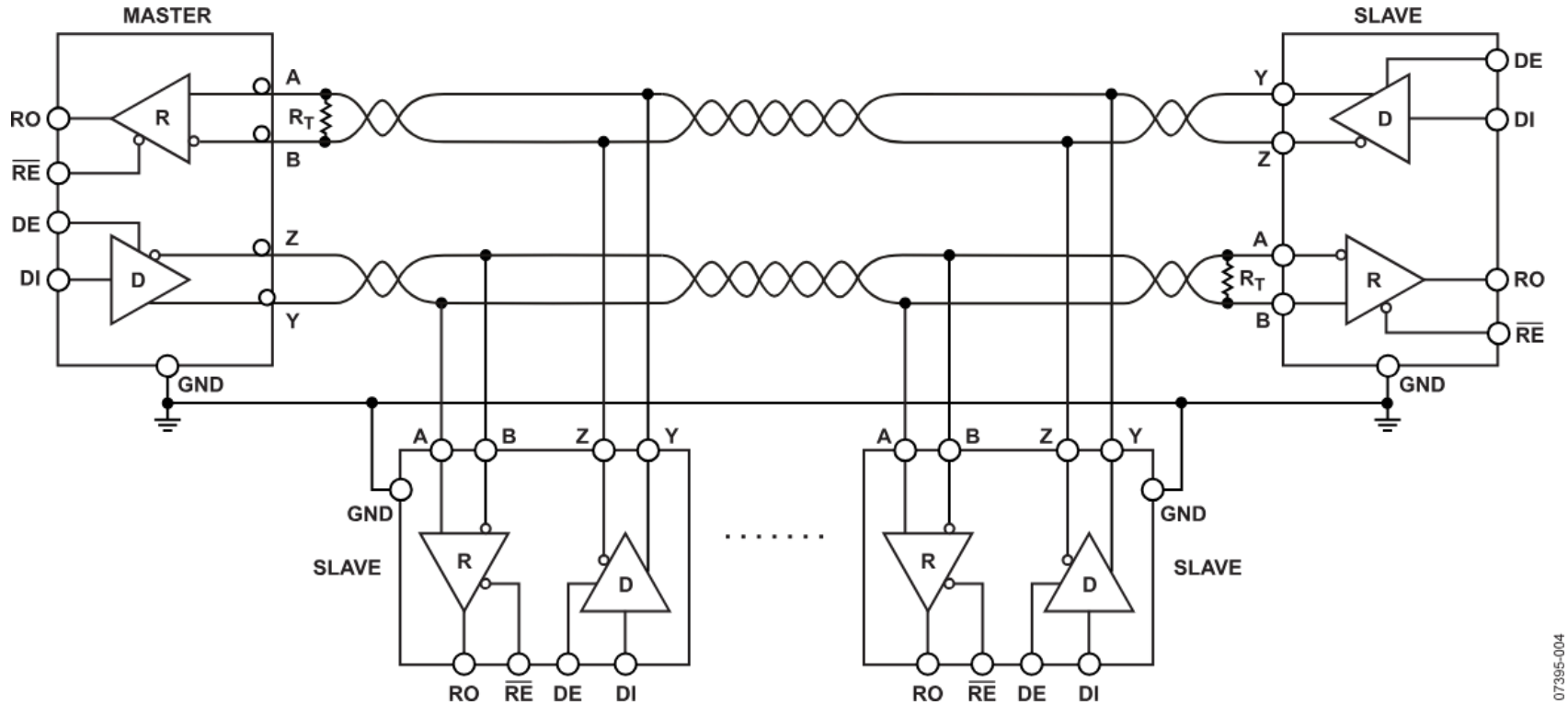
Четырехпроводная и двухпроводная версии RS-485



Типичная схема сети RS-485 (полудуплекс)



Типичная схема сети RS-485 (полный дуплекс)



Сравнение стандартов RS-xxx

Параметр	RS-232	RS-422	RS-485
Тип сигнала	Несиммет-ричный	Дифферен-циальный	Дифферен-циальный
Топология	Точка-точка	Точка-точка/ Много-точечная	Точка-точка/Много-точечная
Устройства на шине	2, дуплекс	1 передатчик, 10 приемников, симплекс	До 32, дуплекс, полудуплекс
Помехо-устойчивость	Низкая	Высокая	Высокая
Макс. расстояние	15 м	1200 м	1200 м
Максимальная скорость передачи	115,2 кбит/с, 460 max	10 Мбит/с	10 Мбит/с

- Подробнее:
  - В чем отличия интерфейсов RS-232, RS-422 и RS-485?
  - Энциклопедия АСУ ТП | 2.3. Интерфейсы RS-485, RS-422 И RS-232
  - AN-960: RS-485/RS-422 Circuit Implementation Guide | Analog Devices

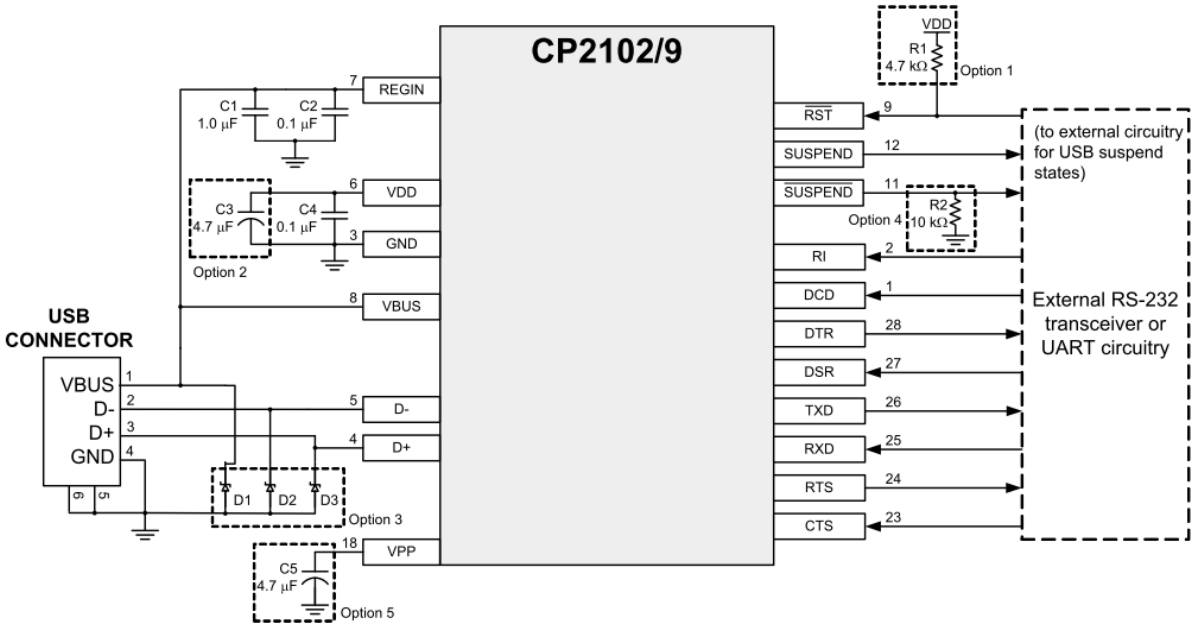
Связь UART/USART с RS-xxx

- UART/USART генерирует логические уровни (0/1), которые преобразуются в физические сигналы через драйверы:
  - MAX232: Преобразует UART в RS-232.
  - MAX485: Преобразует UART в RS-485.
- Пример схемы:

  UART_TX → MAX232 → RS-232_TX
  UART_RX ← MAX232 ← RS-232_RX

Приятная опция: конвертер USB-UART

- cp2102
- ch340g



Option 1: A 4.7 kΩ pull-up resistor can be added to increase noise immunity.
Option 2: A 4.7 µF capacitor can be added if powering other devices from the on-chip regulator.
Option 3: Avalanche transient voltage suppression diodes should be added for ESD protection. Use Littlefuse p/n SP0503BAHT or equivalent.
Option 4: 10 kΩ resistor to ground to hold $\overline{\text{SUSPEND}}$ low on initial power on or device reset.
Option 5: A 4.7 µF capacitor can be added for in-system programming (CP2109 only).

CP2102-9.pdf