Программно-аппаратные платформы Интернета вещей и встраиваемые системы

Лекция 4

ВНЕШНИЕ ИНТЕРФЕЙСЫ МИКРОКОНТРОЛЛЕРА

Внутрисхемные и внешние интерфейсы

Внутрисхемные интерфейсы

- Небольшая дальность (в пределах платы)
- Простая реализация
- Обычно встроены в микроконтроллер
- Примеры:
  - U(S)ART
  - SPI
  - I^2C
  - Параллельные интерфейсы памяти, дисплеев и т. п.
  - MII, RMII
  - JTAG, SWD

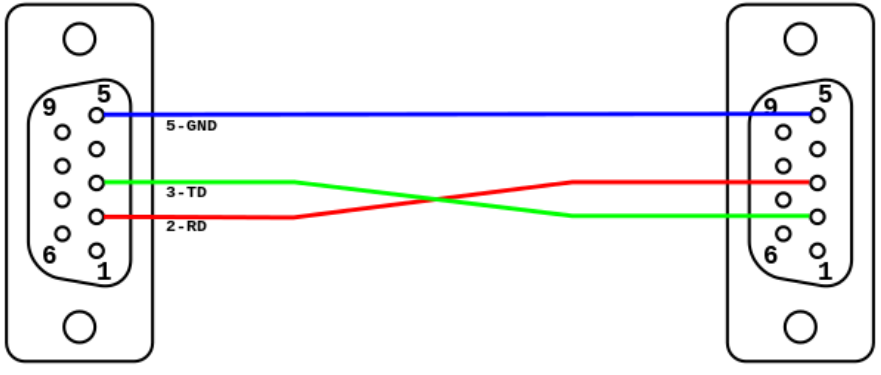
Внешние интерфейсы

- Большая дальность
- Специализированные внешние драйверы
- Примеры:
  - Ethernet
  - CAN
  - LIN (K-Line)
  - RS232
  - RS485
  - USB
  - HDMI
  - тысячи их…

UART И ВСЕ-ВСЕ-ВСЕ

RS232, он же COM-порт

- Использовался для различного коммуникационного оборудования
- Использовался в персональных компьютерах до начала 2000-х годов
- Схема нуль-модемного кабеля (для соединения двух компьютеров):



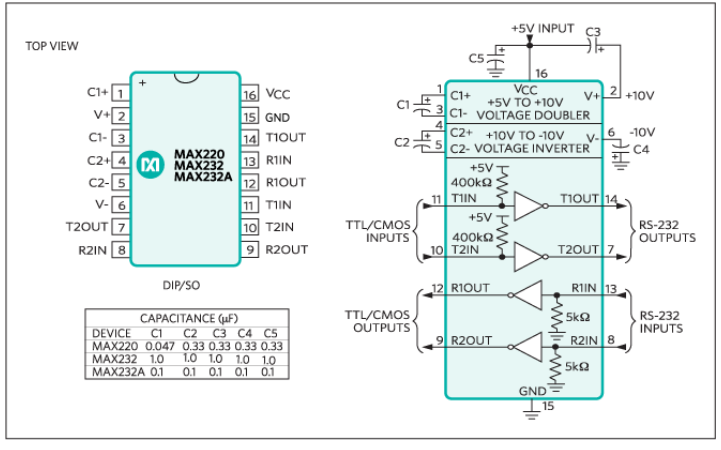
RS232, электрические параметры

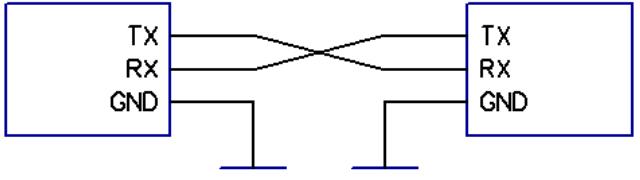
- -15…-3 В – логическая «1», mark
- 3…15 В – логический «0», space
- По умолчанию в линии поддерживается уровень, соответствующий логической «1»
- Последовательная передача – стартовый бит, биты данных (обычно от LSB к MSB), бит четности, стоп-бит
- Параметры:
  - Скорость передачи, обычно выбирается из стандартного ряда:
    300, 1200, 4800, 9600, 14400, 19200, 33400, 57600, 115200, … бит/с
  - Количество бит данных, от 5 до 8
  - Бит четности (опционально)
  - Длительность стоп-бита (1, 1,5, 2)

RS232 – попробуем прочитать

UART

- Universal Asynchronous Receiver/Transmitter
- Временные параметры полностью аналогичны RS-232
- Логические уровни – стандартные 3,3 или 5 В
- Микросхемы преобразователей уровня распространены и дешевы

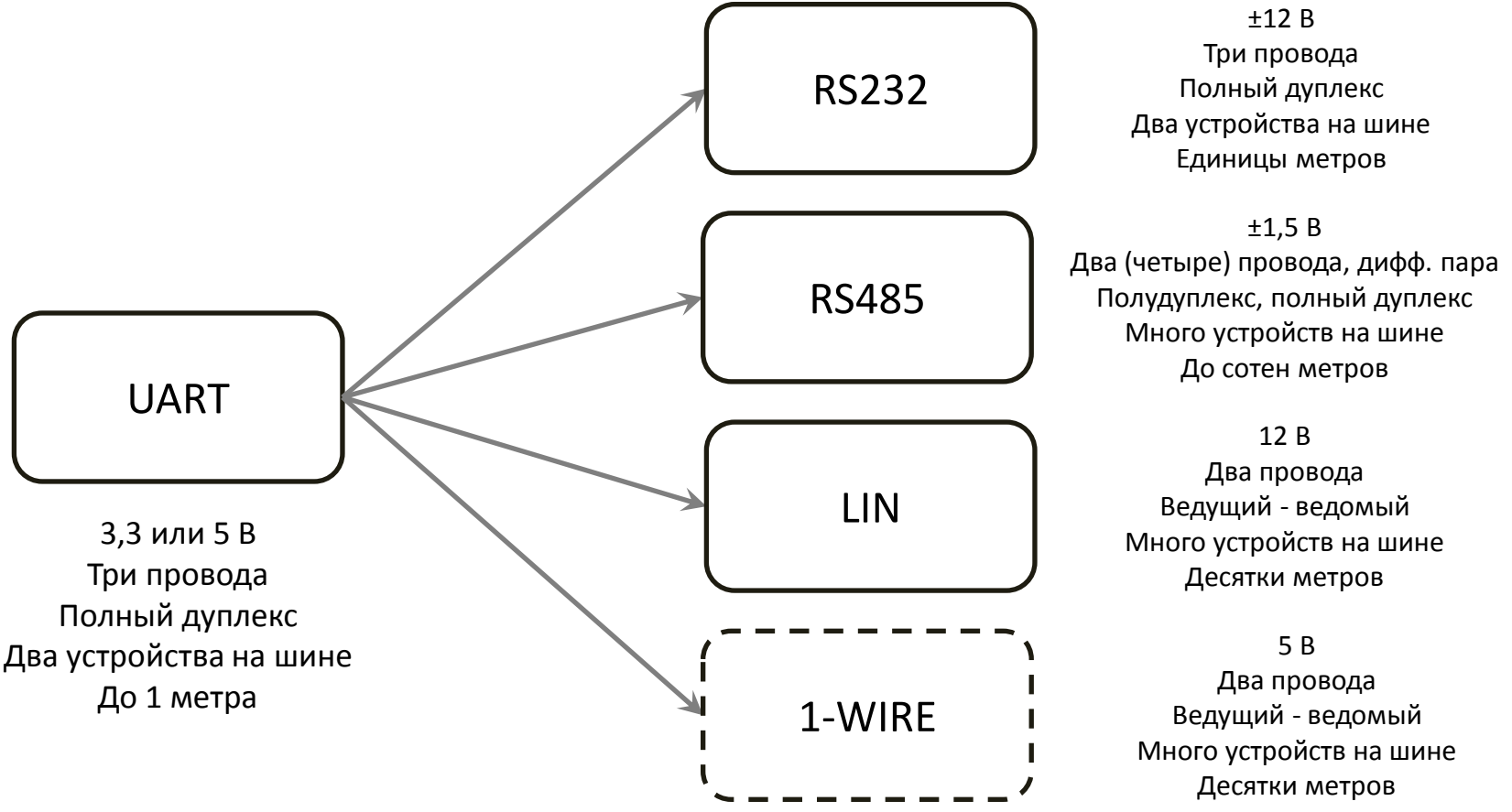




- Подключаем Rx к Tx, Tx к Rx… или наоборот?

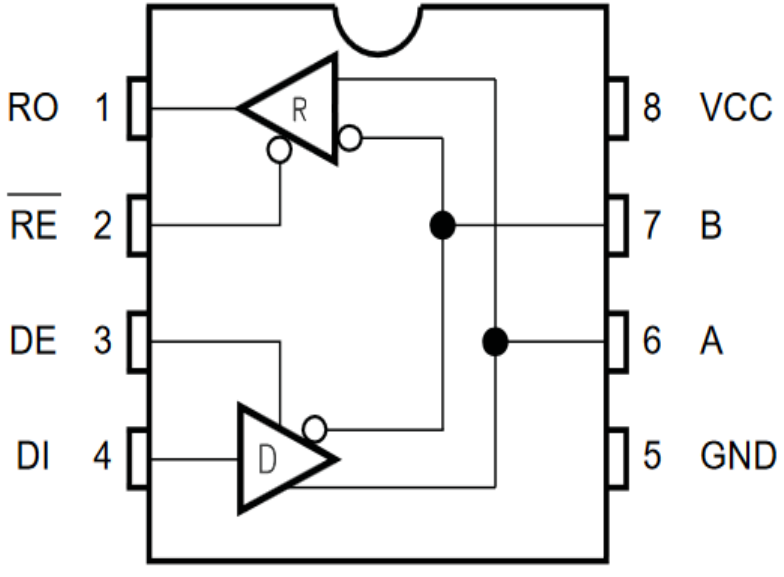
UART – попробуем прочитать

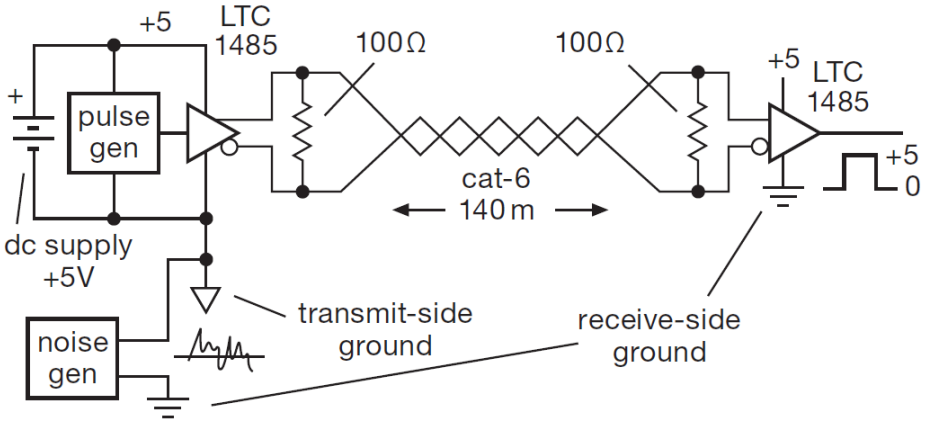
UART – не только внутрисхемный

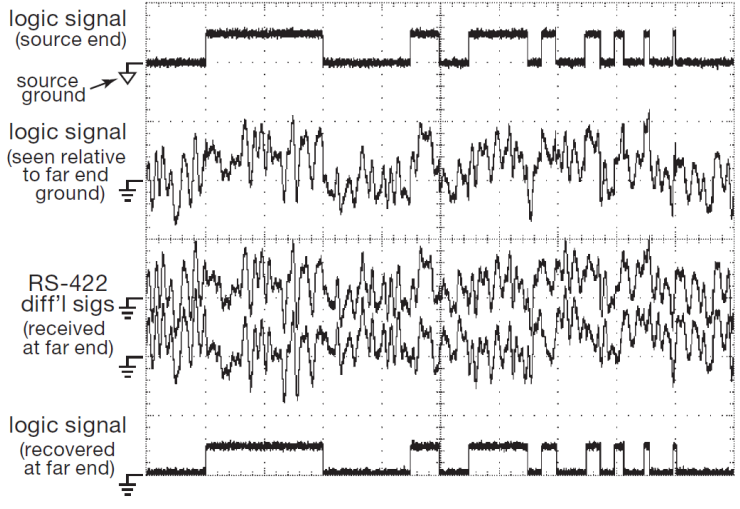


Пример: RS485

Драйвер RS485
ST3485





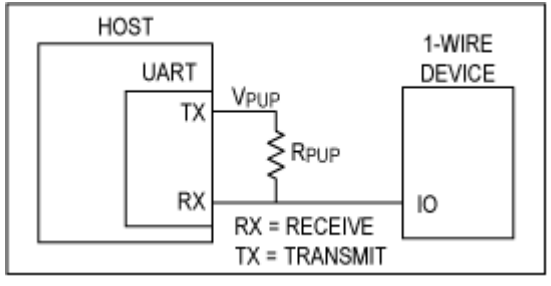


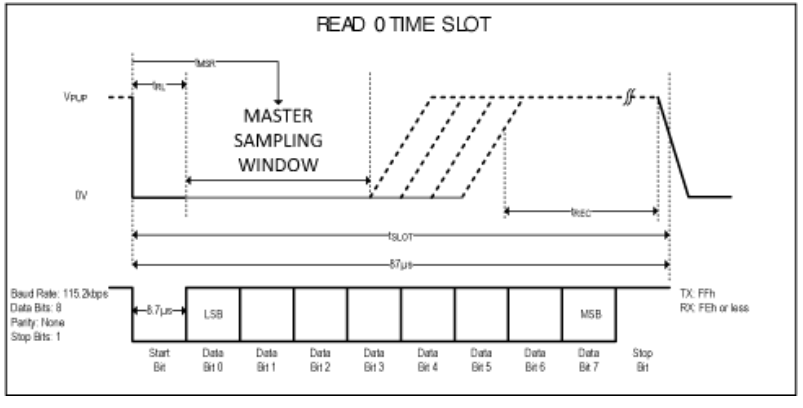
(The Art of Electronics, Third Edition, fig. 12.121, 12.122)

Пример: 1-Wire

- *1 провод*, открытый коллектор
  - Логическая «1» - включение низкого уровня на 1-15 мкс
  - Логический «0» – включение низкого уровня на 60 мкс
  - Сигнал сброса – 480 мкс, после чего «ведомое» устройство сигнализирует о своем присутствии, удерживая на шине низкий уровень
  - Протокол определения адресов устройств

https://www.maximintegrated.com/en/design/technical-documents/tutorials/2/214.html

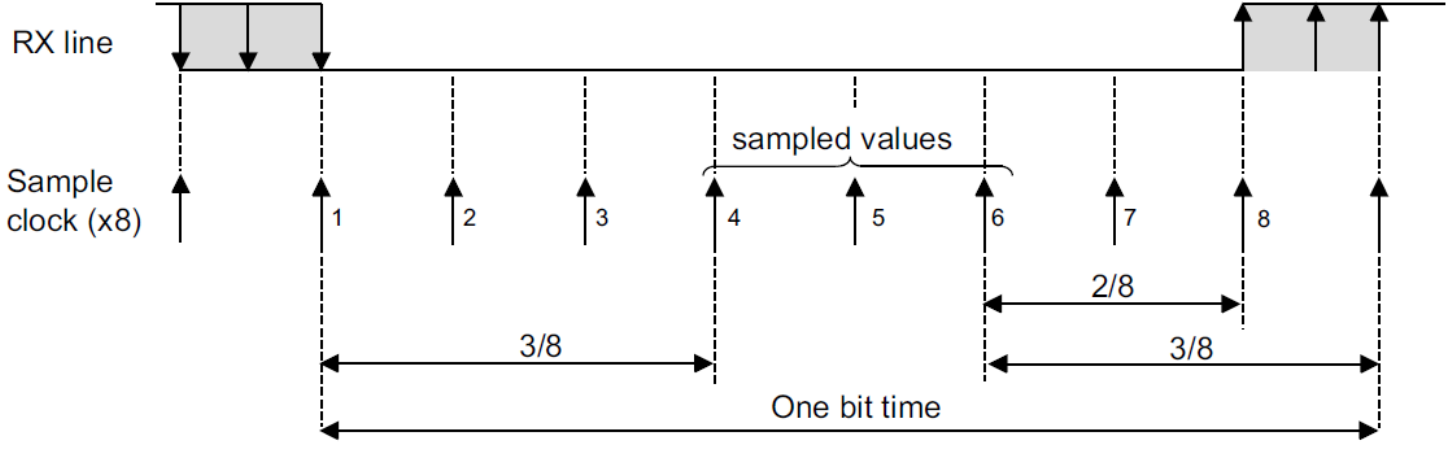




Важно! Вывод Tx у UART должен быть включен в режиме open-drain!

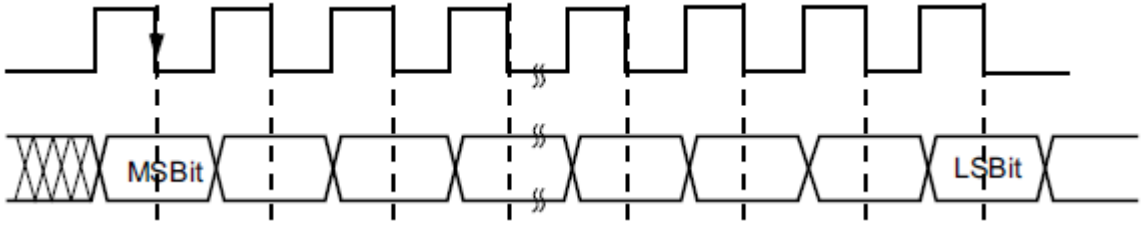
Асинхронные интерфейсы

- Длина бита зависит от скорости передачи
- Оцифровка с оверсемплингом
- Допустимое рассогласование скорости – около 2,5%



Синхронные интерфейсы

- USART – Universal Synchronous/Asynchronous Receiver/Transmitter
- К Rx и Tx добавляется тактовый сигнал (CK, clock), генерируется ведущим устройством на шине
- Измерения строго по фронтам тактового сигнала



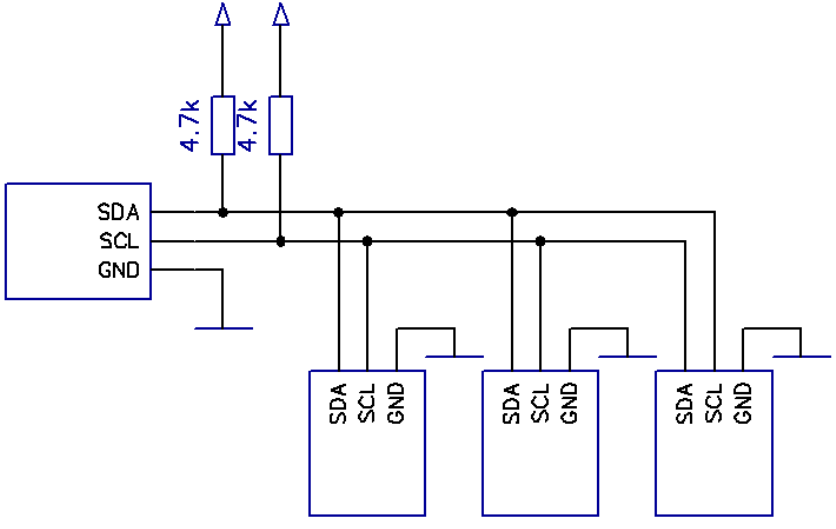
ШИНЫ I^2C И SPI

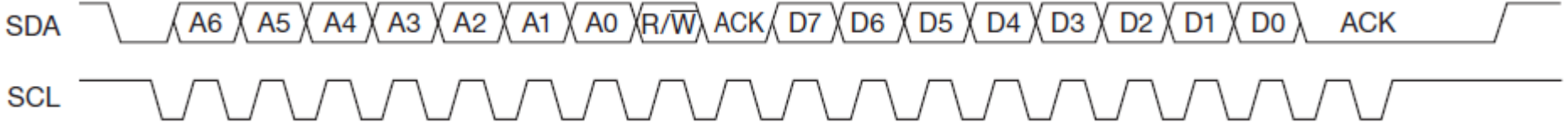
Адресация в интерфейсах

- Точка-точка, без адресации
  - UART
- Аппаратная адресация, сигнал CS (chip select)
  - SPI
  - Большой расход GPIO
- Программная адресация
  - 1-Wire, I^2C
  - Выходы на шине должны быть либо с открытым стоком/коллектором, либо 3-state

I²C (он же TWI)

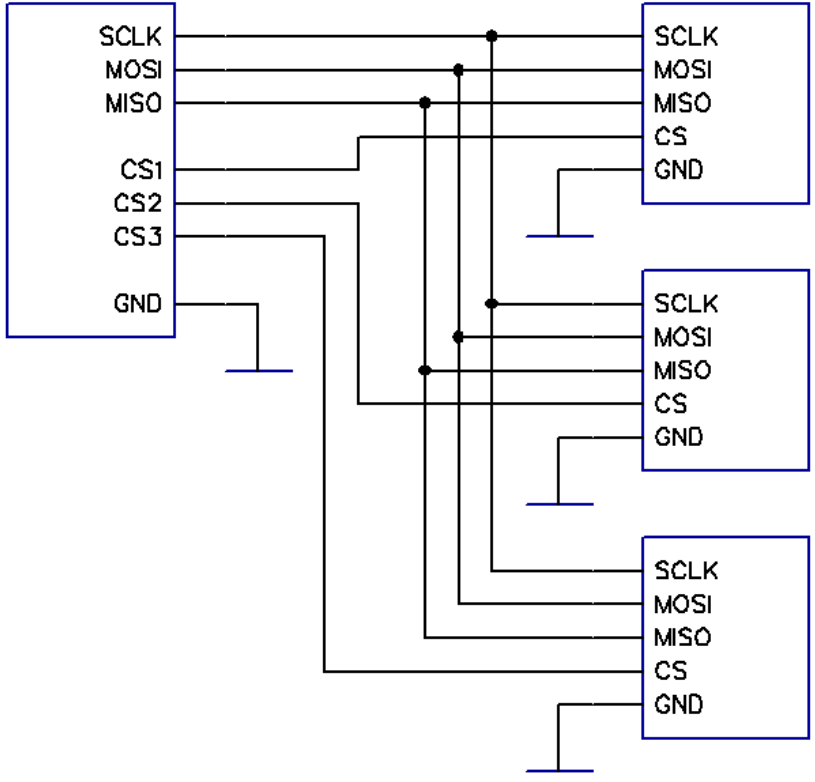
- Последовательный синхронный интерфейс
- 2 провода – SDA (данные), SCL (тактирование), open drain
- В спецификации предусмотрен режим multi-master
- Программная адресация, адрес 7 или 10 бит
- Скорость 100 или 400 кБит/с





SPI

- Последовательный синхронный интерфейс
- 4 провода – MISO/MOSI (данные), SCK (тактирование), CS (chip select), push-pull
- Одно ведущее устройство на шине, остальные ведомые
- Аппаратная адресация
- Скорость до 20 Мбит/с
- Параллельные варианты – QuadSPI и подобные



ИСПОЛЬЗОВАНИЕ ВНЕШНИХ ИНТЕРФЕЙСОВ В RIOT OS

Внешние интерфейсы в микроконтроллере

- Bit-banging – имитация сложных интерфейсов переключением GPIO
  - Так делать не надо, это медленно и тратит процессорное время на «коммуникации»
  - Примеры – Soft-UART, Soft-SPI, CMSIS DAP (отладчики UMDK-RF и UMDK-EMB)
  - Если все же так надо – возможно, надо использовать CMSIS/прямой доступ к регистрам, а не драйвер GPIO
- Используйте специальные периферийные модули
  - Несколько регистров для настройки
  - Регистры для передаваемых/принимаемых данных
  - Прерывание при приеме, окончании передачи, ошибке…
  - У «стандартной» периферии (UART, SPI, I^2C) – универсальные стандартизированные драйверы в ОС

Когда использовать прерывания?

- Стандартные драйверы RIOT почти никогда не используют прерывания для интерфейсов I^2C и SPI
- Вместо этого в цикле проверяется состояние флагов:

```
while (!(dev(bus)->SR & SPI_SR_TXE)) {}
```

- Сколько времени занимает переключение потоков, а сколько – передача одного байта?
- Если в микроконтроллере есть FIFO-буферы для внешних интерфейсов – использование прерывания по заполнению буфера на ½ или ¾ может быть полезно
- Модуль `sys/isrpipe` позволяет передавать данные из прерывания в контекст процесса с минимальными накладными расходами

Пример: драйвер I²C

1. Зарезервируйте права на нужный интерфейс (при этом захватится mutex)

```
i2c_acquire(&i2c);
```

2. Инициализируйте интерфейс

```
i2c_init_master(&i2c, I2C_SPEED_NORMAL);
```

3. Передайте данные

```
i2c_write_regs(&i2c, ADDRESS, REGISTER, &data, BYTES);
```

4. Получите данные

```
i2c_read_regs(&i2c, ADDRESS, REGISTER, &data, BYTES);
```

5. Отпустите интерфейс

```
i2c_release(&i2c);
```

Некоторые нюансы

- В SPI для *получения* данных надо *передать* данные (буквально что угодно)
- В I^2C надо обрабатывать ошибки
- В UART данные могут прийти в любой момент
  - в процессоре случится прерывание
  - драйвер UART позовёт указанную при настройке интерфейса функцию
  - микроконтроллер принимает по 1 байту за раз
- В любой непонятной ситуации — читай даташит