Аппаратное обеспечение IoT/CPS
Лекция 4

А. А. Подшивалов
apodshivalov@miem.hse.ru

Внутрисхемные и внешние интерфейсы

Внутрисхемные интерфейсы:

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

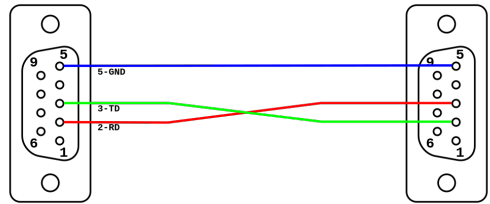
Внешние интерфейсы:

- Большая дальность
- Специализированные внешние микросхемы-драйверы
- Примеры:
  - USB
  - Ethernet
  - RS-232
  - RS-485
  - CAN
  - LIN (K-Line)
  - HDMI
  - and many others

UART и все-все-все

RS-232, или COM-порт

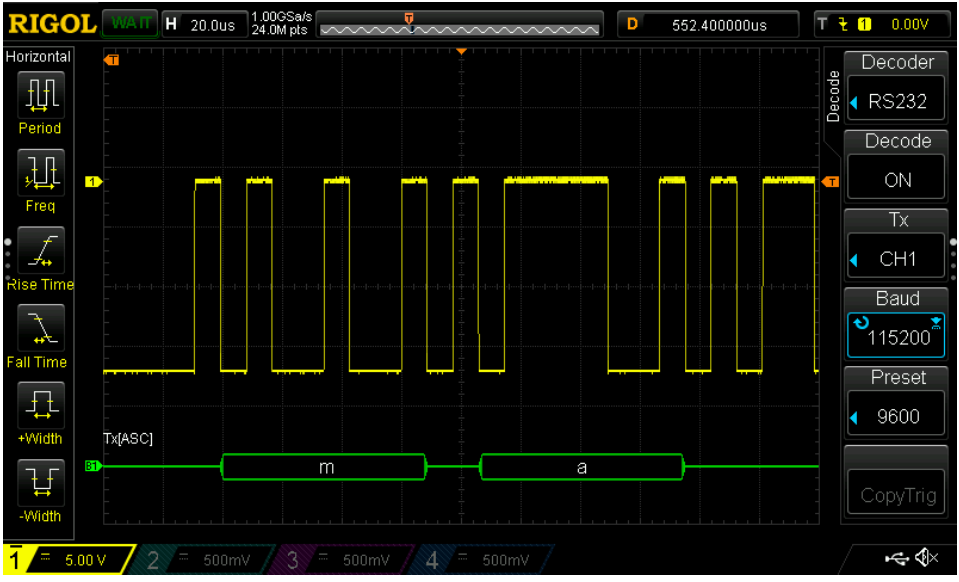
- Использовался для различного коммуникационного оборудования
- Присутствовал в персональных компьютерах до начала 2000-х годов
- Для двунаправленной передачи данных достаточно трех проводов



RS-232, электрические параметры

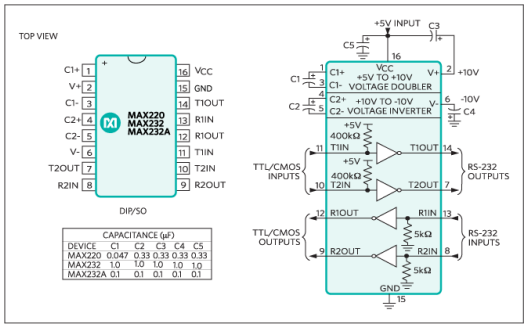
- $-15 \ldots -3$ В — логическая «1», mark
- $3 \ldots 15$ В — логический «0», space
- По умолчанию в линии поддерживается уровень, соответствующий логической «1»
- Данные передаются последовательно — стартовый бит (space), биты данных от LSB к MSB, бит четности (если есть), стоп-бит (mark)
- Параметры:
  - 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)

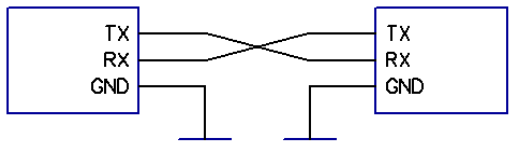
RS-232 на экране осциллографа



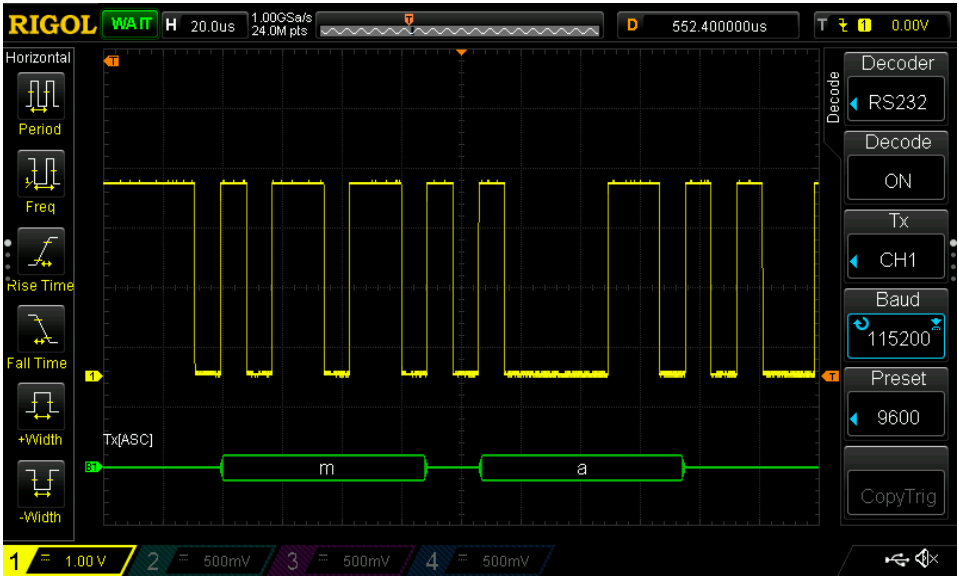
UART

- Universal Asynchronous Receiver/Transmitter
- Временные параметры аналогичны RS-232
- Логические уровни — стандартные 3,3 или 5 В
- Микросхемы преобразователей уровней распространены и дешевы
- Подключаем Rx к Tx, Tx к Rx... но иногда наоборот!





UART на экране осциллографа

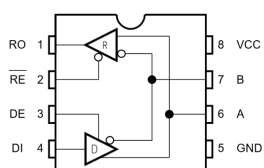


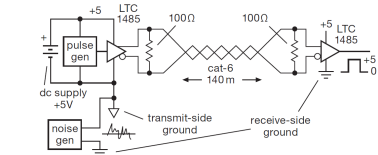
Внешние интерфейсы на основе UART

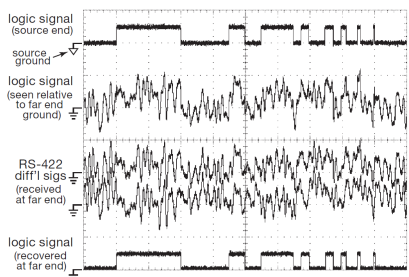
- UART — 3,3 или 5 В, три провода, полный дуплекс, 2 устройства на шине, до 1 м
- RS-232 — $\pm$12 В, три провода, полный дуплекс, 2 устройства на шине, единицы метров
- RS-485 — $\pm 1,5$ В, два (четыре) провода, полудуплекс или полный дуплекс, много устройств на шине, до сотен метров
- LIN — 12 В, два провода, ведущий-ведомый, много устройств на шине, десятки метров
- 1-wire — 3,3 или 5 В, два провода, ведущий-ведомый, много устройств на шине, десятки метров

RS-485

- Дифференциальный (полу)дуплексный интерфейс
- Временные параметры идентичны RS-232
- На шине может присутствовать множество устройств



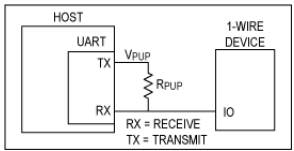


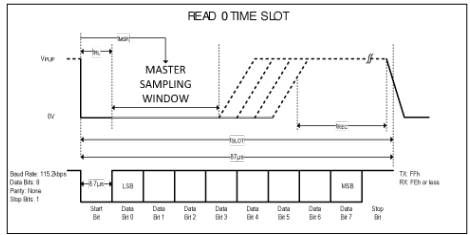


The Art of Electronics, Third Edition, fig. 12.121, 12.122

1-wire

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

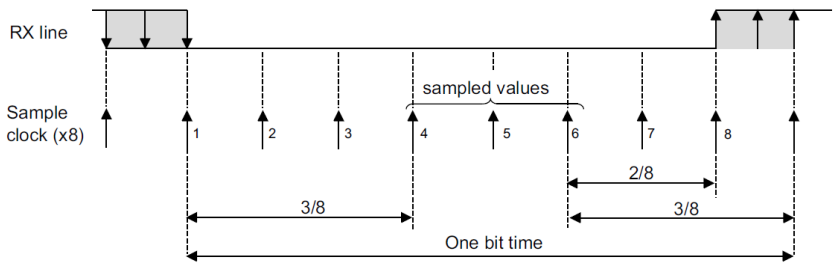




https://www.maximintegrated.com/en/design/technical-documents/tutorials/2/214.html

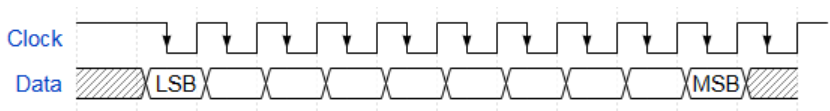
UART как асинхронный интерфейс

- Оцифровка входящего сигнала происходит с оверсемплингом (x8 или x16, в зависимости от скорости передачи данных)
- Приемник и передатчик синхронизируются по стартовому биту
- Типичный приемник UART использует мажорирование по 3 или 5 «центральным» отсчетам
- Допустимая рассинхронизация между приемником и передатчиком — 2 периода тактового сигнала на 10 бит данных (80 периодов), то есть 2,5%



Синхронные интерфейсы

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



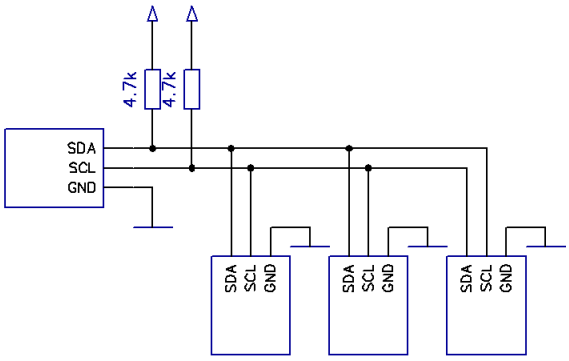
Интерфейсы I^2C и SPI

Адресация устройств на общей шине

- Точка-точка, без адресации (RS-232)
- Программная адресация (протоколы поверх RS-485, I^2C)
  - Необходимо принимать специальные меры, чтобы одновременная передача данных парой устройств была безопасна
- Аппаратная адресация (SPI)
  - Большой расход GPIO

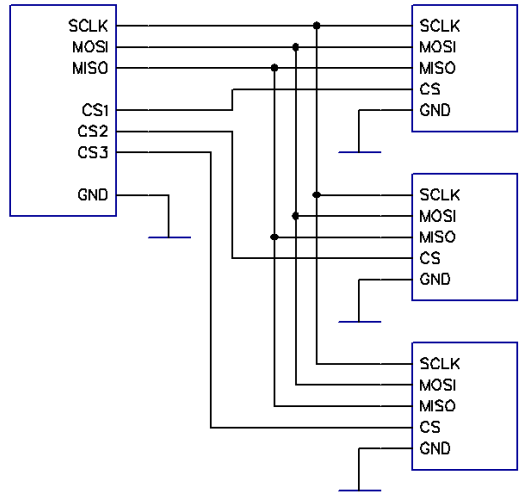
I²C, он же TWI

- Последовательный синхронный интерфейс
- Две линии — SDA (данные), SCL (тактирование), open-drain
- Адрес 7 или 10 бит
- Скорость 100 или 400 кбит/с
- В спецификации предусмотрен режим multi-master



SPI

- Последовательный синхронный интерфейс
- Четыре линии — MISO/MOSI (данные), SCLK (тактирование), CS (chip select), push-pull
- Одно ведущее устройство на шине, остальные ведомые
- Скорость до 20 Мбит/с
- Параллельные варианты — QuadSPI и подобные



Работа с встроенными в микроконтроллер интерфейсами

Программная реализация

- Bit-banging, «ногодрыг» — управление GPIO напрямую для реализации более сложных интерфейсов
- Крайне неэффективно тратит процессорное время
- По возможности используйте CMSIS или подобные средства, а не «универсальный» драйвер GPIO
- Soft-UART, Soft-SPI, драйвер дисплея HD47780 и подобных, CMSIS-DAP

Пример: CMSIS-DAP

```
static __inline void PIN_SWCLK_SET (void) {
    GPIO_BSRR(SWCLK_GPIO_PORT) = SWCLK_GPIO_PIN;
}
static __inline void PIN_SWCLK_CLR (void) {
    GPIO_BRR(SWCLK_GPIO_PORT) = SWCLK_GPIO_PIN;
}
static __inline uint32_t PIN_SWDIO_IN  (void) {
    return (GPIO_IDR(SWDIO_GPIO_PORT) & SWDIO_GPIO_PIN) ? 0x1 : 0x0;
}
#define SW_READ_BIT(bit)                \
  PIN_SWCLK_CLR();                      \
  PIN_DELAY();                          \
  bit = PIN_SWDIO_IN();                 \
  PIN_SWCLK_SET();                      \
  PIN_DELAY()

for (n = 32; n; n--) {
    SW_READ_BIT(bit);               /* Read RDATA[0:31] */
    val >>= 1;
    val  |= bit << 31;
}
```

Аппаратная реализация

- Большинство микроконтроллеров содержат аппаратные блоки, реализующие популярные интерфейсы (UART/USART, I^2C, SPI, возможно, что-то еще)
- Обычно возможна довольно гибкая настройка (коэффициенты деления тактового сигнала, некоторые параметры интерфейса, ...)
- При окончании передачи или приеме очередного байта, либо при какой-либо ошибке может быть вызвано прерывание

Периферийные устройства и ОС Riot

- Файлы `board.h` и `periph_conf.h` содержат описания доступных на конкретной плате интерфейсов
- Соответствие альтернативных функций GPIO и аппаратных модулей микроконтроллера описывается в этих файлах

Пример: I^2C для STM32...

```
static const i2c_conf_t i2c_config[] = {
    {
        .dev            = I2C1,
        .speed          = I2C_SPEED_NORMAL,
        .scl_pin        = GPIO_PIN(PORT_B, 6),
        .sda_pin        = GPIO_PIN(PORT_B, 9),
        .scl_af         = GPIO_AF4,
        .sda_af         = GPIO_AF4,
        .bus            = APB1,
        .rcc_mask       = RCC_APB1ENR_I2C1EN,
        .clk            = CLOCK_APB1,
        .irqn           = I2C1_EV_IRQn
    }
};
#define I2C_0_ISR           isr_i2c1_ev
#define I2C_NUMOF           ARRAY_SIZE(i2c_config)
```

...и I^2C для nRF52

```
static const i2c_conf_t i2c_config[] = {
    {
        .dev = NRF_TWIM1,
        .scl = 27,
        .sda = 26,
        .speed = I2C_SPEED_NORMAL
    }
};
#define I2C_NUMOF           ARRAY_SIZE(i2c_config)
```

Драйверы периферийных устройств в ОС Riot

```
// Инициализируем устройство
i2c_init(I2C_DEV(0));
...
// Захватываем шину
i2c_acquire(I2C_DEV(0));
// Записываем адрес регистра и продолжаем обмен
i2c_write_byte(I2C_DEV(0), device_addr, reg_addr, I2C_NOSTOP);
// Считываем данные после рестарта
i2c_read_byte(I2C_DEV(0), device_addr, &reg_data, 1, 0);
// Освобождаем шину
i2c_release(I2C_DEV(0));
```

UART и stdio

- Драйвер UART использует функцию `uart_write` для блокирующего вывода и позволяет определить обработчик прерывания при получении очередного байта
- По умолчанию `UART_DEV(0)` используется для стандартного ввода-вывода
- Реализация стандартной библиотека языка C — newlib-nano, модули `sys/stdio_uart` и `sys/newlib_syscalls_default` содержат реализации функций, необходимых для функционирования стандартного ввода-вывода

Драйверы внешних устройств

- Для работы с внешними устройствами используем только аппаратно-независимые интерфейсы
- Дескриптор устройства и его параметры

```
typedef struct {
    tmpabc_params_t p;
    int scale;
} tmpabc_t;
typedef struct {
    i2c_t bus;
    uint8_t addr;
} tmpabc_params_t;
```

- Функция инициализации (`int tmpabc_init(tmpabc_t *dev, const tmpabc_params_t *params)`) копирует параметры в дескриптор
- Все остальные функции, работающие с устройством, получают дескриптор в качестве первого параметра

Пример: датчик температуры/давления/влажности

```
bmx280_t bmx280;
const bmx280_params_t bmx280_params = {
    .i2c_dev = I2C_DEV(0),
    .i2c_addr = 0x76,
    .t_sb = BMX280_SB_0_5,
    .filter = BMX280_FILTER_OFF,
    .run_mode = BMX280_MODE_FORCED,
    .temp_oversample = BMX280_OSRS_X1,
    .press_oversample = BMX280_OSRS_X1,
    .humid_oversample = BMX280_OSRS_X1,
};

bmx280_init(&bmx280, &bmx280_params);
int t = bmx280_read_temperature(&bmx280);
```