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

Интерфейс и протокол

Интерфейс

... - физический/логический способ взаимодействия между компонентами
Определяет:

- Аппаратные средства средства: линии связи (провода), уровни напряжения (TTL, LVDS), разъёмы
- Логическую структуру: как передются данные (последовательно/параллельно), синхронизацию (тактовый сигнал), роли устройств (мастер/ведомый)

Примеры интерфейсов:

- SPI (Serial Peripheral Interface) - аппаратная реализация
- I2C (Inter-Integrated Circuit) - сочетает аппаратные и логические правила
- USB - стандарт разъёмов, аппаратные и логические правила

Протокол

... - набор правил, определяющих формат и порядок обмена данными
Описывает:

- Структуру сообщений
- Последовательность действий
- Семантику данных

Примеры протоколов:

- HTTP
- Modbus - формат сообщений для промышленных систем
- TCP //GETOUTOFMYHEADGETOUTOFMYHEADGETOUTOFMYHEADGETOUTOFMYHEADGETOUTOFMYHEADGETOUTOFMYHEADGETOUTOFMYHEADGETO

Схуяли что то типа SPI называют и интерфейсом, и протоколом?

SPI - интерфейс, включающий в себя протокол
Как интерфейс:

- Определяет физические линии
- Задаёт уровни напряжения
- Устанавливает роли

Как протокол:

- Определяет порядок передачи битов
- Задаёт режимы синхронизации
- Описывает интерпретацию данных

Отсюда и смешение терминов

Ключевое отличие интерфейс/протокол

- Интерфейс отвечает на вопрос "Как подключить устройства?"
- Протокол - на "Как они обмениваются данными?"

Протоколы UART, USART

UART (Universal Asynchronous Receiver-Transmitter)

... - Асинхронный последовательный протокол передачи данных
Работает без тактового сигнала
Использует два основных сигнала - TX (передача), RX (приём)
Данные передаются последовательно:

- Начиная со стартового бита
- За ним - биты данных
- Бит чётности (optional)
- Один/два стоповых бита

Особенности UART

- Асинхронная передача - не требует тактирования
- Формат кадра - 1 стартовый бит, 5-9 бит данных, 0/1 бит чётности, 1/2 стоповых бита
- Скорость передачи - задаётся битрейтом
- Дуплексность - полудуплексный/полный дуплекс
- Применение - широко используется в МК, GPS-модулях, Bluetooth, RS-232

Временная диаграмма UART

Диаграмма передачи байта 0b01001101
Pasted image 20250402104226%20darkmode.png
UART - асинхронный протокол, но для правильного приёма/передачи данных нужны битовая и кадровая синхронизации

Кадровая синхронизация UART

Кадровая синхронизация - корректное определение начала и конца каждого кадра

- Обнаружение стартового бита (0):
    - Линия UART висит в 1
    - Появился 0 - новый кадр
    - Приёмник запускает таймер для отсчёта битов
- Приём данных:
    - Приёмник фиксирует значения битов в заранее установленные моменты времени
- Проверка чётности (optional):
    - Если используется бит чётности - проверка его на соответствие с переданными данными
- Обнарузение стопового бита:
    - Наличие стопового бита (1) - завершение кадра
    - Если обнаружен 0 - ошибка кадрирования

Контроль чётности UART

- even parity -> число единиц чётное
- odd parity -> число единиц нечётное
- Ошибка - в слове состояния UART выставляет признак ошибки

Битовая синхронизация

Обеспечивает правильное определение границ каждого бита
Особенно важно ибо приёмник и передатчик работают на одном битрейте, но без тактвого сигнала
Как достигается:

- Фиксированный битрейт
- Приёмник использует стартовый бит для начала отсчёта времени **середины** каждого бита, дабы захватить значение строго в этот момент
- Для формирования временных битовых интервалов приёмник/передатчик имеют свои источники тактирования
- Точность источников в сумме погрешностей не должна превышать половины (в идеале четверти) битового интервала

Ошибки синхронизации UART

Битовой:

- Неравные битрейты - всё поехало
- Шум/дрожжание сигнала

Кадровой:

- Ошибка кадрирования
- Ошибка чётности
- Ошибки синхронизации - потеряешь данные нахуй

USART (Universal Syncronous/Asynchronous Receiver-Transmitter)

USART - более универсальный протокол, поддерживающий и синхронную и асинхронную передачу данных
В синхронном режиме юзает тактовый сигнал (duh)

Особенности USART

- Асинхронный режим - работает как UART
- Синхронный режим - добавляется линия тактирования SCK, синхронизация по фронту/спаду сигнала
- Выше скорость передачи - нет необходимости в стартовых/стоповых битах
- Применение - высокоскоростные системы, SPI-подключения, некоторые виды RS-485

Временная диаграмма USART, синхронный режим

Pasted image 20250402110523%20darkmode.png

Схемы поключения

Pasted image 20250402110600%20darkmode.png

Примеры работы с UART

AVR

```
// UART init
#include <avr/io.h>

#define F_CPU 16000000UL
#define BAUD 9600
#define UBRR_VAL ((F_CPU / 16 / BAUD) - 1)

void UART_init() {
    // Transmission speed
    UBRR0H = (uint8_t)(UBRR_VAL >> 8);
    UBRR0L = (uint8_t)UBRR_VAL;

    // Turn on i/o
    UCSR0B = (1 << TXEN0) | (1 << RXEN0);

    // No parity control
    UCSR0C = (1 << UCSZ01) | (1 << UCSZ00);
}

// Data output (TX)
void UART_SendChar(uint8_t data) {
    // Wait for buffer to be freed
    while (!(UCSR0A) & (1 << UDRE0)));

    // Write to buffer
    UDR0 = data;
}

void UART_SendString(const char* str) {
    while (*str) {
        UART_SendChar(*str++);
    }
}
// ^ Sends a whole ass string daymn

// Data input (RX)
uin8_t UART_ReceiveChar() {
    // Wait for buffer
    while (!(UCSR0A & (1 << RXC0)));

    return UDR0;
}

// Usage example
int main() {
    UART_init();
    UART_SendString("Hello, UART!\r\n");

    while (1) {
        uint8_t received = UART_ReceiveChar();
        UART_SendChar(received); // echo
    }
}
```