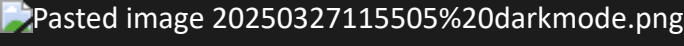
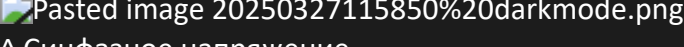


Хохлов умоляет ходить на лекции ибо подробные материалы - прикольно, но понимать тоже нихуёво было бы

## Временные диаграммы LVDS

^ Сигналы LVDS (односторонние и дифференциальные)  
^ Синфазное напряжение

## Реализация выходов на МК AVR

```
// push-pull
DDRB |= (1 << PB0)
PORTB &= ~(1 << PB0) // 0
PORTB |= (1 << PB0) //1

// open-drain emulation
#include <avr/io.h>

// Set pin as open-drain
#define OPEN_DRAIN_SET_LOW(pin) \
do { DDRB |= (1 << pin); PORTB &= ~(1 << pin); } \
while(0)
#define OPEN_DRAIN_SET_HIGH(pin) \
do { DDRB &= ~(1 << pin); PORTB |= (1 << pin); } \
while(0)

int main(void) {
    OPEN_DRAIN_SET_HIGH(0);
    OPEN_DRAIN_SET_LOW(0);
    return 0;
}
```

Приколюха про код выше:

**Q:** Почему do while(0) в define а не просто код, взятый в {}?

**A:** Кривой синтаксис define + приколы с препроцессором (он уберёт цикл моментально), так что есть общее соглашение о том, что многокомандные макросы обёртываются в do while(0)

**Q:** Почему не функция а макрос?

**A:** Экономия тактов. Вызов функции требует всего, что связано с вызовом функции, но макрос просто подставит код и похер. В C++ есть директива inline, которая помогла бы функцию сразу вставить в код, но:

- В C её нет (только в последних версиях с которыми мы не работаем сейчас лмао)
- В самих плюсах она иногда криво работает 🙄

## Реализация выходов на МК STM32

```
// push-pull
void GPIO_Init(void) {
    // Enable GPIOA clicking
    __HAL_RCC_GPIOA_CLK_ENABLE();

    GPIO_InitTypeDef GPIO_InitStructure = {0};
    GPIO_InitStructure.Pin = GPIO_PIN_5;
    // Push-Pull
    GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStructure.Pull = GPIO_NOPULL;
    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_HIGH;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStructure);
}

// Usage
// Enable
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
// Disable
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
```

**Q:** По кой хер нужен HAL?

**A:** Обратная совместимость. Всегда может что-то незнаительно поменяться по типу используемых регистров или такой херни. В статично написанном коде всё ломанётся нахуй, но при наличии HAL может даже не понадобится что-либо делать при переносе Legacy кода на новые МК

```
// open-drain
void GPIO_Init(void) {
    __HAL_RCC_GPIOB_CLK_ENABLE();
    GPIO_InitTypeDef GPIO_InitStructure = {0};
    GPIO_InitStructure.Pin = GPIO_PIN_6;
    // Output Open Drain Mode
    GPIO_InitStructure.Pull = GPIO_PULLUP;
    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_HIGH;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStructure);
}

// OD
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, GPIO_PIN_SET);
// 0
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, GPIO_PIN_RESET);

// tri-state
// Logical 1
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
// Logical 0
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
// Z-state
HAL_GPIO_DeInit(GPIOA, GPIO_PIN_5);
```

## Реализация выходов на МК ESP32

```
// push-pull
#include "driver/gpio.h"

void app_main() {
    gpio_config_t io_conf = {
        .pin_bit_mask = (1ULL << GPIO_NUM_2),
        .mode = GPIO_MODE_OUTPUT,
        .pull_up_en = GPIO_PULLUP_DISABLE,
        .pull_down_en = GPIO_PULLDOWN_DISABLE,
        .intr_type = GPIO_INTR_DISABLE
    };
    gpio_config(&io_conf);

    gpio_set_level(GPIO_NUM_2, 1); // high level
    gpio_set_level(GPIO_NUM_2, 0); // low level
}

// open-drain
#include "driver/gpio.h"

void app_main() {
    gpio_config_t io_conf = {
        .pin_bit_mask = (1ULL << GPIO_NUM_2),
        .mode = GPIO_MODE_OUTPUT_OD,
        .pull_up_en = GPIO_PULLUP_ENABLE,
        .pull_down_en = GPIO_PULLDOWN_DISABLE,
        .intr_type = GPIO_INTR_DISABLE
    };
    gpio_config(&io_conf)

    gpio_set_level(GPIO_NUM_2, 1); // OD
    gpio_set_level(GPIO_NUM_2, 0); // low level
}
```

## Входы GPIO МК

- Без подтяжки
- С внутренне подтяжкой к питанию
- С внутренней подтяжкой к земле

## Реализация входа с подтяжкой на МК AVR

```
DDRB &= ~(1 << PB0);
PORTB |= (1 << PB0);
if (PINB & (1 << PINB0)) {
    ...
}
```

## Реализация входа с подтяжкой на МК ESP32

```
gpio_config_t io_conf = {
    .pin_bit_mask = (1ULL << GPIO_NUM_5),
    .mode = GPIO_MODE_INPUT,
    .pull_up_en = GPIO_PULLUP_ENABLE,
    .pull_down_en = GPIO_PULLDOWN_DISABLE,
    .intr_type = GPIO_INTR_DISABLE
};
gpio_config(&io_conf);

if (!gpio_get_level(GPIO_NUM_5)) {
    ...
}
```

## Схема порта МК AVR и альтернативные функции

<https://www.microchip.com/en-us/product/atmega328p> - I/O порты - секция 13  
<https://www.arxterra.com/8-atmega-gpio/> - #8: ATmega GPIO - Arxterra  
<https://microsin.net/programming/avr/gpio-and-alternate-port-functions.html> - GPIO и альт функции порта AVR  
RM0090 глава 6  
AN4899 раздел 4  
<https://we.easyelectronics.ru/STM32/prakticheskiy-kurs-stm32-urok-1---gpio-porty-vvoda-vyvoda.html> - кусок  
практического курса STM32  
ESP32 Technical Reference Manual Version 5.3 раздел 4  
<https://microsin.net/programming/arm/esp32-pinout-reference.html> - справочник по выводам ESP32