Программирование микроконтроллеров STM32

FSM & Intro to OS

МФТИ, 2019

Спаггети-код. Миф или реальность?

Can someone please fix my spaghetti code

Credits: me.me

Что делать?

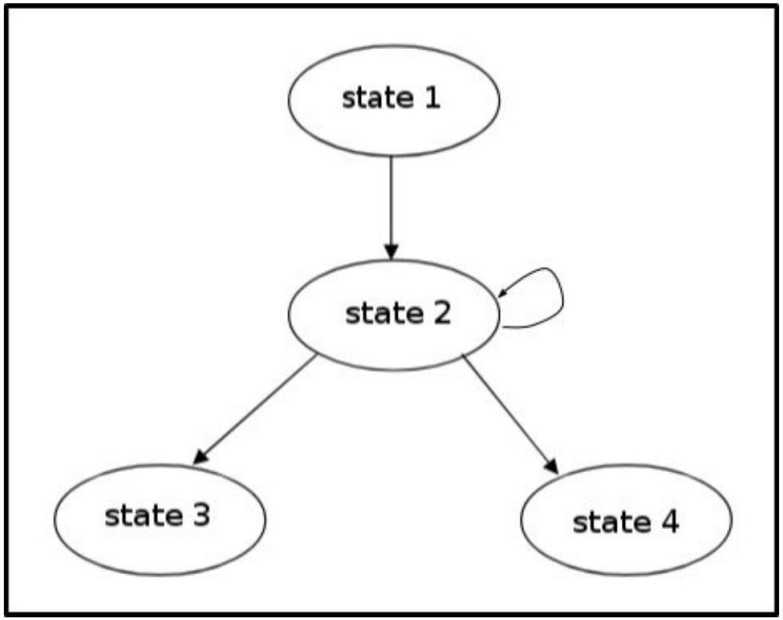
Конечный автомат

Операционная система

Жить с этим

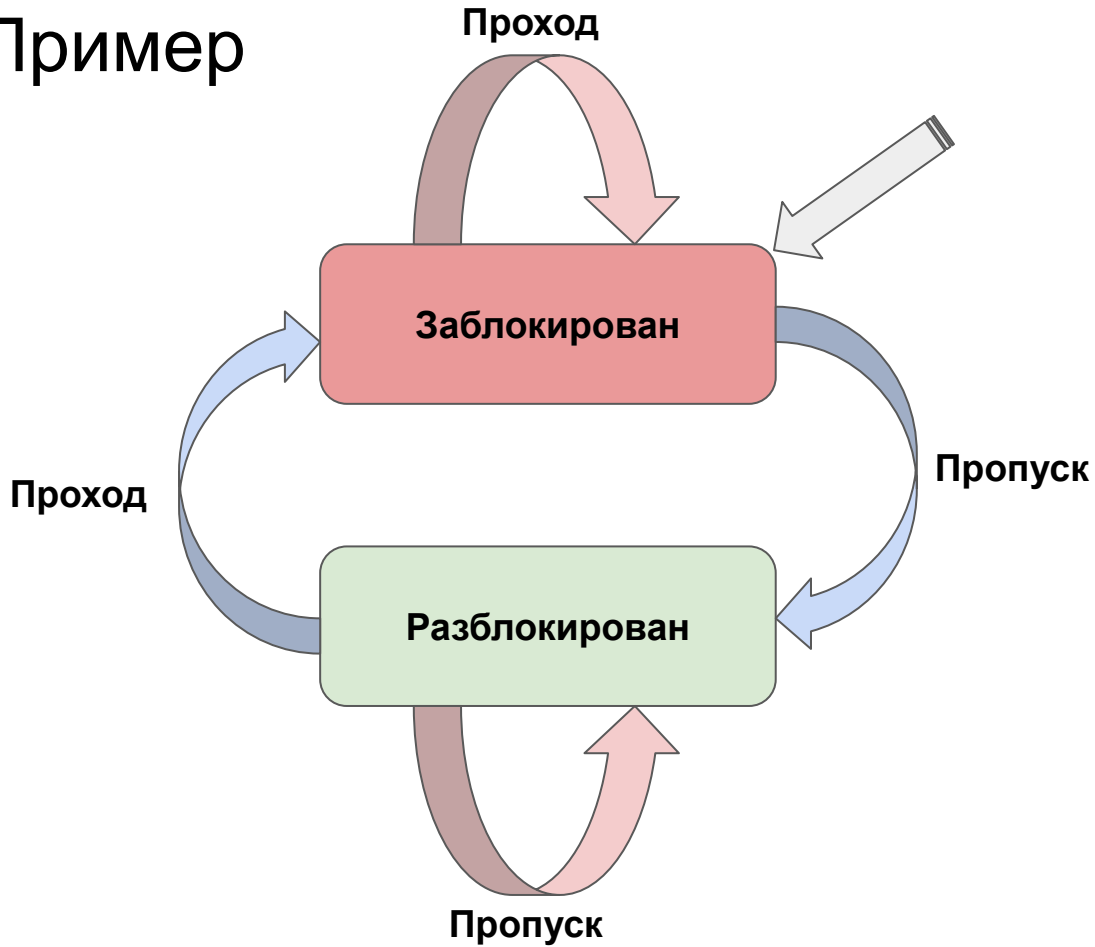
Конечный автомат

Конечный автомат - это машина состояний, которая может находиться только в одном состоянии каждый момент времени. Между состояниями могут осуществляться переходы.



Credits: Google

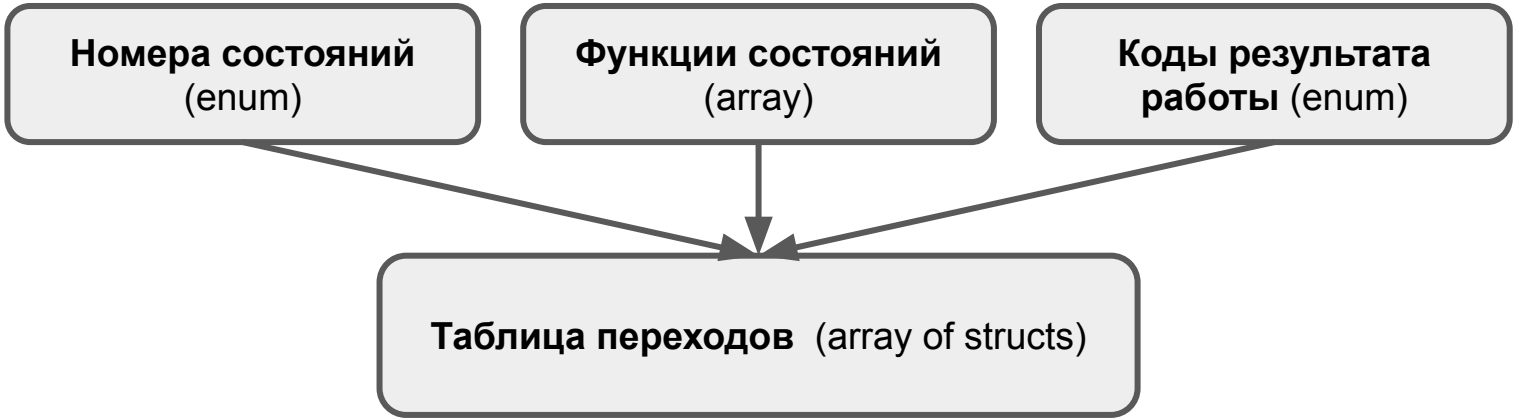
Конечный автомат. Пример



Конечный автомат. Таблица переходов

Текущее состояние	Результат работы	Следующее состояние
Заблокирован	Проход	Заблокирован
Заблокирован	Пропуск	Разблокирован
Разблокирован	Проход	Заблокирован
Разблокирован	Пропуск	Разблокирован

Конечный автомат в С




```
typedef struct {
    enum FSM_STATES src_state;
    enum RET_CODES ret_code;
    enum FSM_STATES dst_state;
} fsm_cell_t;
```

```
fsm_cell_t fsm_table[] = {
    {STARTUP, OK_FSM, BUTTON_POLL},
    {BUTTON_POLL, OK_FSM, LED_TURN_ON},
    {BUTTON_POLL, REPEAT_FSM, BUTTON_POLL},
    {LED_TURN_ON, OK_FSM, BUTTON_POLL}
};
```

Конечный автомат и ОС

Конечный автомат:

+ Простота реализации
+ Отсутствие накладных расходов
+ Эффективность
- Отсутствие механизма передачи данных между состояниями
- Кооперативное выполнение

Следующий уровень абстракции это использование **операционной системы реального времени**

ОС реального времени

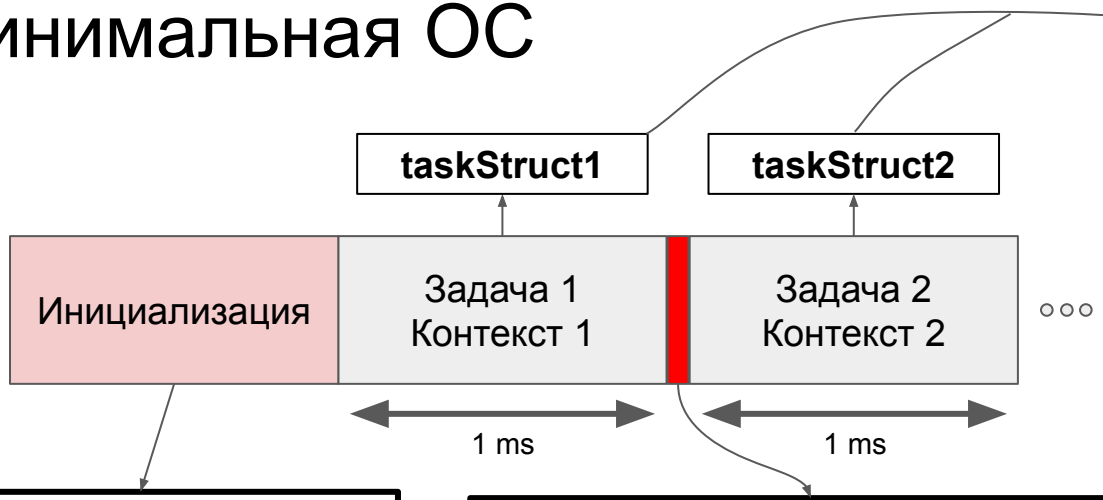
ОСРВ — это система, мультиплексирующая аппаратные ресурсы и реагирующая на внешние события в определенный промежуток времени

Компоненты ОСРВ:

1. Планировщик задач
2. Менеджер памяти
3. Примитивы межпроцессного взаимодействия
4. Драйвера различных устройств

Примеры ОСРВ: FreeRTOS, RIOT OS, RTLinux и т.д.

Минимальная ОС



taskStruct:
1. Значения всех регистров
2. Маш. состояние процессора
3. Указатель на стек
4. Сервисные данные

1. Создание структур
2. Переключение на PSP стек
3. Инициализация всех структур

Переключение контекста (MSP стек, привилегированный режим):
1. Сохранение маш. состояния предыдущего процесса
2. Выбор следующего процесса
3. Загрузка маш. состояния для след. процесса (контекст)
4. Выбор стека

Обычно происходит в SysTimer

Машинное состояние:
xPSR
PC, LR, PSP
R0-R12

Минимальная ОС. Пример

```
void task1(void) {
    while (1) {
        toggle_led(GPIOC, 7);
        delay_ms(1000);
    }
}
```

```
void task2(void) {
    while (1) {
        toggle_led(GPIOC, 8);
        delay_ms(1500);
    }
}
```

```
uint32_t mem1[512];
tcb_t tcb1;
uint32_t mem2[512];
tcb_t tcb2;

os_add_task(task1, mem1, tcb1, 512);
os_add_task(task2, mem2, tcb2, 512);

os_start_scheduler();
```

Минимальная ОС. Куда двигаться дальше?

1. Добавление менеджера памяти
2. Добавление средств межпроцессного взаимодействия
3. Добавление поддержки системных вызовов для удобной работы с драйверами (не имеет смысла для Cortex-M0)
4. Анализ таймингов

Репозиторий

https://github.com/edosedgar/stm32f0_ARM