Программно-аппаратные платформы Интернета вещей и встраиваемые системы

Лекция 1

Несколько вводных слов

- Александр Анатольевич Подшивалов

  apodshivalov@hse.ru

  apodshivalov@miem.hse.ru

- Критерии оценки:

  0,4*_Экзамен_ + 0,4*_Практика_+0,2*_Проект_

Содержание курса

- Программирование микроконтроллеров
  - Cortex-M (семейства STM32, nRF52, CC26xx), RISC-V
  - Язык Си
  - Операционная система RIOT OS
- Сетевые технологии (LoRaWAN, 6LoWPAN)
- «Экзотика» (энергосбережение, безопасность и тому подобное)

Лекции

- Микроконтроллерные системы в IoT.
- GPIO, таймеры, прерывания. Многозадачность.
- Внешние интерфейсы микроконтроллера: UART, I2C, SPI.
- ОС RIOT - модули, драйверы, HAL. Процессы и IPC.
- АЦП и ЦАП.
- Беспроводные технологии IoT. LoRaWAN как пример MAC-уровня.
- Безопасность в беспроводных сетях на примере LoRaWAN.
- IEEE 802.15.4, 6LoWPAN, mesh-сети.
- Энергосберегающие режимы работы.
- Отладка микроконтроллерных систем.
- Память - Flash/EEPROM, загрузчик (bootloader). Модуль DMA.
- Внешние датчики.
- Цифровая обработка сигналов, библиотека CMSIS-DSP.
- Определение местоположения.
- Программно-аппаратные проекты - некоторые нюансы.

Практические занятия

- Аудитория 234; лучше приносить свой ноутбук
- Несколько блоков заданий
- Баллы за задания:

  1 – типовой пример из лекций

  10 – существенный вклад в opensource-проект
- Github или LMS?

«Проект»

- Доклад по результатам «проектной деятельности» (если укладывается в тематику курса)
- Прототип устройства с использованием «конструктора» в лаборатории 234

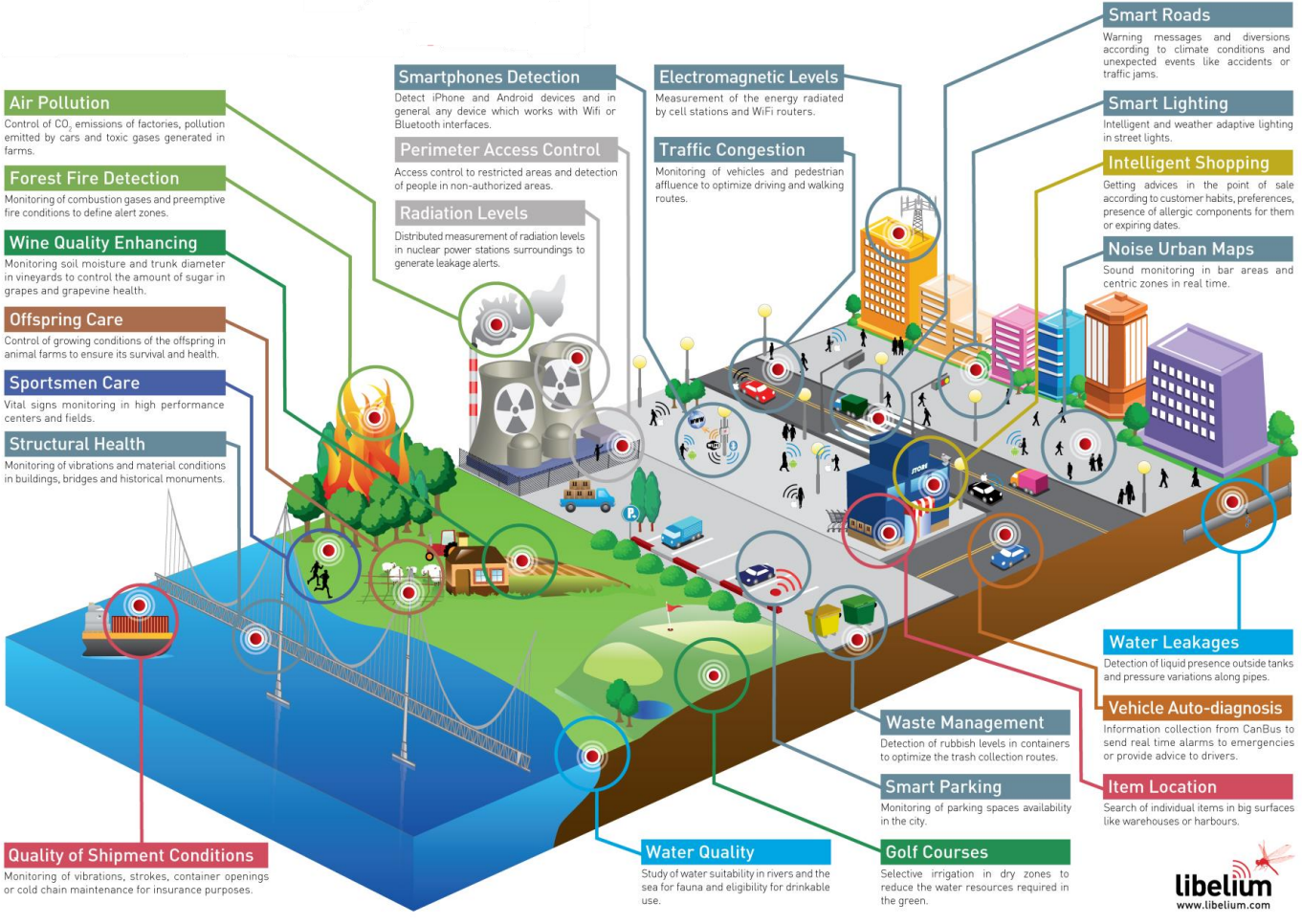
ИНТЕРНЕТ ВЕЩЕЙ – ЧТО ЭТО

Интернет вещей - определения

Интернет вещей - определения

- Википедия: «концепция вычислительной сети физических предметов («вещей»), оснащённых встроенными технологиями для взаимодействия друг с другом или с внешней средой, рассматривающая организацию таких сетей как явление, способное перестроить экономические и общественные процессы, исключающее из части действий и операций необходимость участия человека»
- То же самое простыми словами: **обеспечение сбора и передачи данных там, где раньше это было технически невозможно или экономически неэффективно**

Примеры систем IoT



Место систем сбора данных

Перри Ли — Архитектура интернета вещей, 2019

Требования к системам сбора данных

- Дешевизна – 10$ за устройство
- Компактность – размеры не больше спичечного коробка
- Экономичность – месяцы/годы на одной батарейке
- Дальность связи
- Безлицензионность

МИКРОКОНТРОЛЛЕРЫ: ОСНОВНЫЕ АРХИТЕКТУРЫ

Микроконтроллер vs. микропроцессор

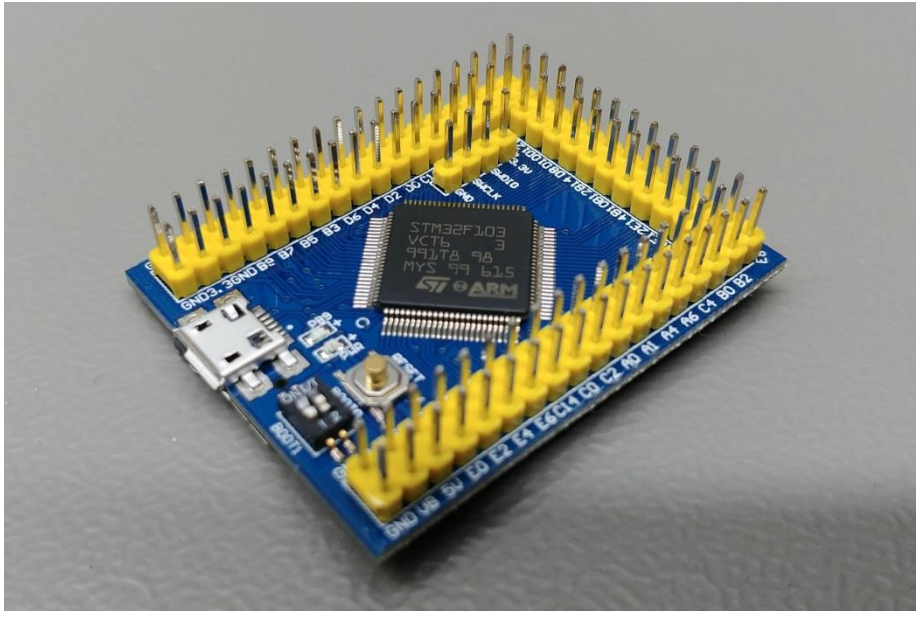
- Микроконтроллер – «однокристальная микро-ЭВМ», CPU, память и периферия на одном чипе
- «Промежуточные» варианты – SoC, SiP

	MCU	CPU
ОЗУ	встроенное	внешнее
Объём ОЗУ	< 1 МБ	>> 1 МБ
Постоянная память	встроенная	внешняя
Объём памяти	< 1 МБ	>> 1 МБ
Периферийные устройства	в основном встроенные	в основном внешние

Типичные характеристики микроконтроллера

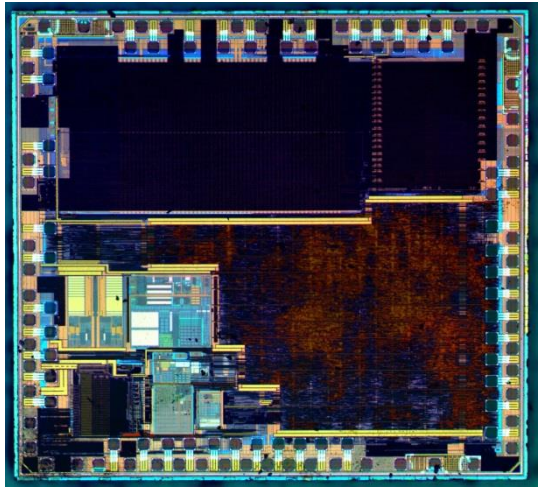
- STM32L151CC
  - Процессорное ядро ARM Cortex M3, 32 МГц
  - 256 кБ Flash-памяти (ROM), 8 кБ EEPROM
  - 32 кБ оперативной памяти
- На 3-6 порядков хуже «настольного» компьютера по производительности и объему памяти

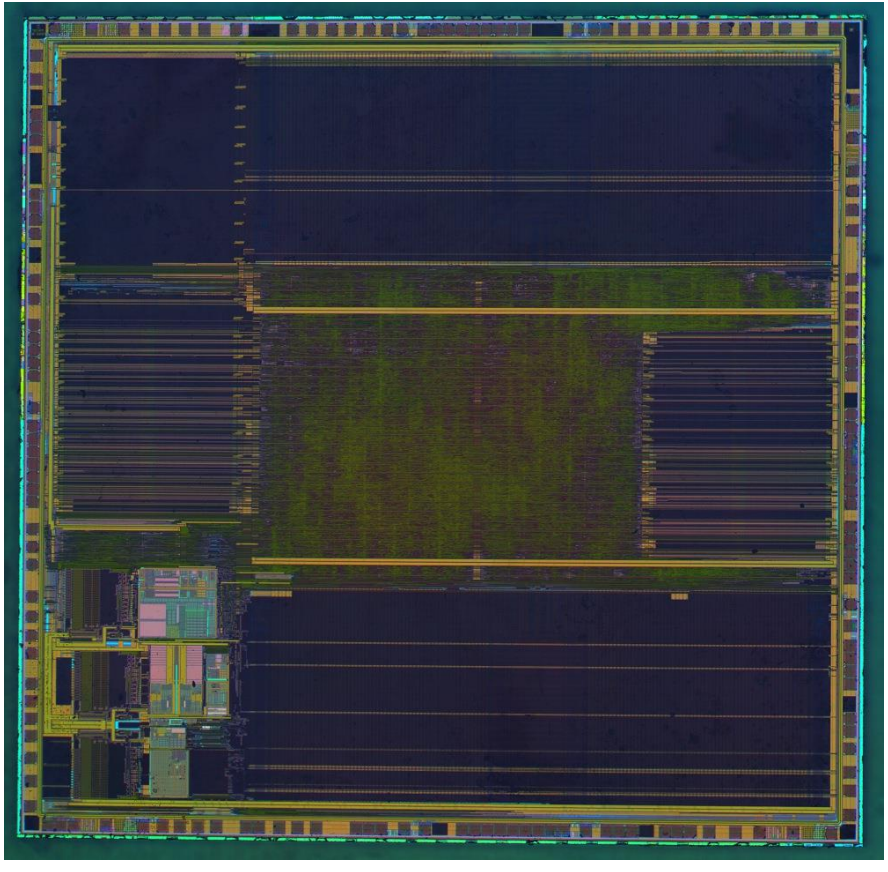
Микроконтроллер: фото «вживую»



Сверху – STM32F100C4T6B (ядро Cortex M3, 16 кБ FLASH, 4 кБ RAM, корпус LQFP-48), справа – STM32F103VCT6 (ядро Cortex M3, 256 кБ FLASH, 48 кБ RAM, корпус LQFP-100)

Микроконтроллер: фото кристалла



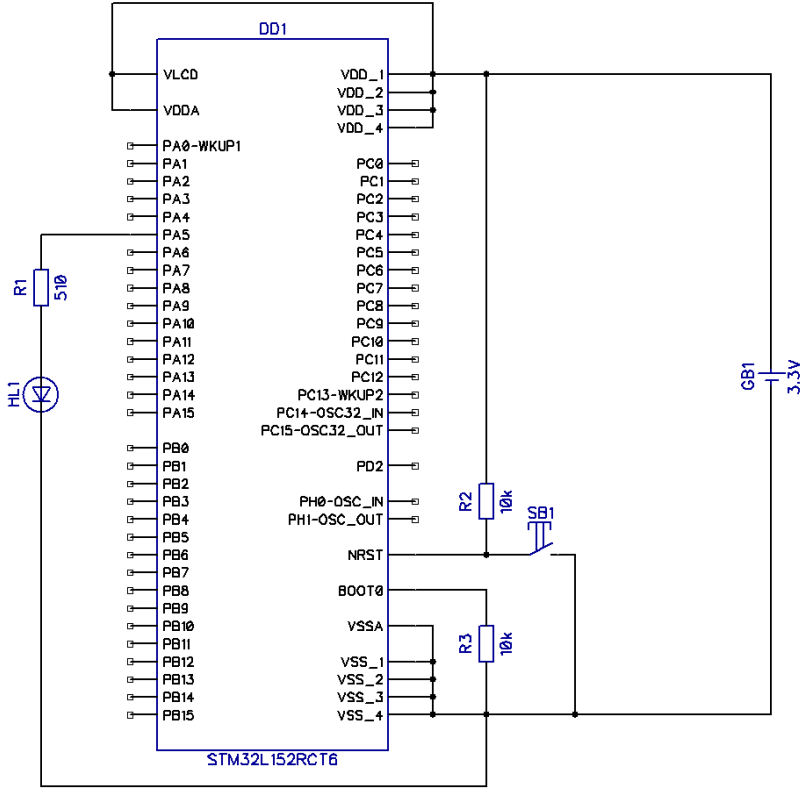


Сверху – STM32F100C4T6B (ядро Cortex M3, 16 кБ FLASH, 4 кБ RAM, 2854x3123 µм), справа – STM32F103VGT6 (ядро Cortex M3, 1 МБ FLASH, 96 кБ RAM, 5339x5188 µм), техпроцесс 180 нм

https://zeptobars.com/

Микроконтроллер – минимальная схема

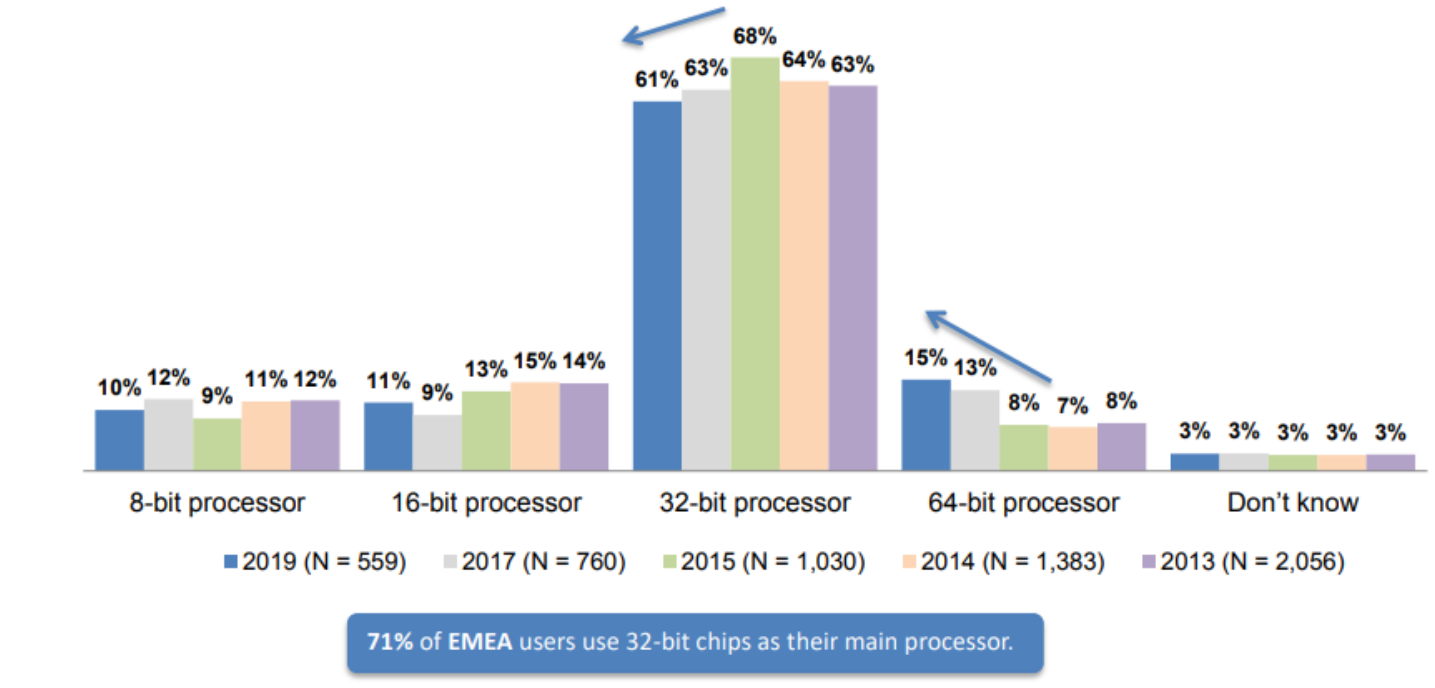
- Нужно только питание (3,3 В)
- Внутренний тактовый генератор может работать без внешних компонентов



Распространенные архитектуры микроконтроллеров

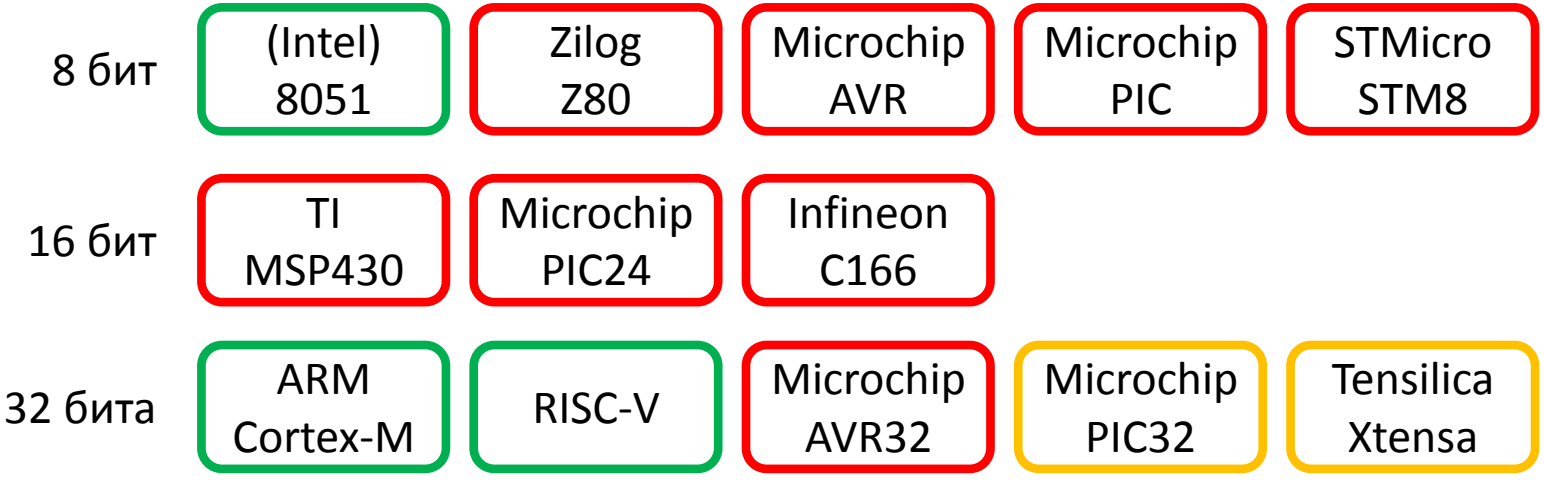
8 бит	(Intel) 8051	Zilog Z80	Microchip AVR	Microchip PIC	STMicro STM8
16 бит	TI MSP430	Microchip PIC24	Infineon C166		
32 бита	ARM Cortex-M	RISC-V	Microchip AVR32	Microchip PIC32	Tensilica Xtensa

Популярные архитектуры микроконтроллеров



Опрос Embedded.com, EETimes и Aspencore, 2019

Популярные архитектуры микроконтроллеров



В 2021 году 8- и 16-битные архитектуры имеет смысл использовать лишь для некоторых специфических задач

ARM Cortex-M	ARM Cortex-R	ARM Cortex-A
Микроконтроллерные системы	Системы жёсткого реального времени	Системы высокой производительности

ARM Cortex M0

ARM Cortex M0+

ARM Cortex M3

ARM Cortex M4

ARM Cortex M4F

ARM Cortex M7

ARM Cortex M23

ARM Cortex M33/M35P

ARM Cortex M55

ARM Cortex-M – некоторые производители

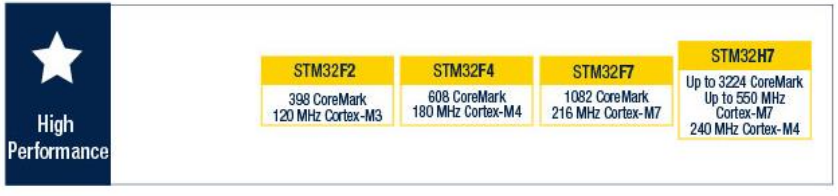
Почему Cortex-M?

- Отличная производительность (1,25 DMIPS/МГц)
- Огромный выбор различных моделей
- Очень богатый набор периферийных устройств
- Программная конфигурация процессора «на лету»
- Низкое энергопотребление и продвинутое управление питанием
- Низкая стоимость

life.augmented

STM32 MCUs
32-bit Arm® Cortex®-M

LONGEVITY 10 YEARS COMMITMENT



Почему не Cortex-M?

- Предыдущие слайды были сделаны в 2018 году

Почему не Cortex-M?

- Предыдущие слайды были сделаны в 2018 году
- «Кризис полупроводников»

Альтернативы

- RISC-V

  Открытая архитектура, множество производителей, в том числе российские (Миландр, Микрон)

С чем будем работать

- STM32
  - STM32L151CC – «конструктор» Unwired Devices
  - STM32WL55JC – микроконтроллер с LoRa
- Nordic Semiconductor nRF51/nRF52 – ядро Cortex-M и приемопередатчик BLE
- TI CC2650 – микроконтроллер с поддержкой IEEE 802.15.4
- SiFive FE310 – недорогой RISC-V

ПРОБУЕМ ПРОГРАММИРОВАТЬ МИКРОКОНТРОЛЛЕРЫ

Документация на микроконтроллер

- Datasheet

  Описывает в основном «электрические» параметры, 100-200 страниц

- Reference Manual

  Описание для программиста, около 1000 страниц

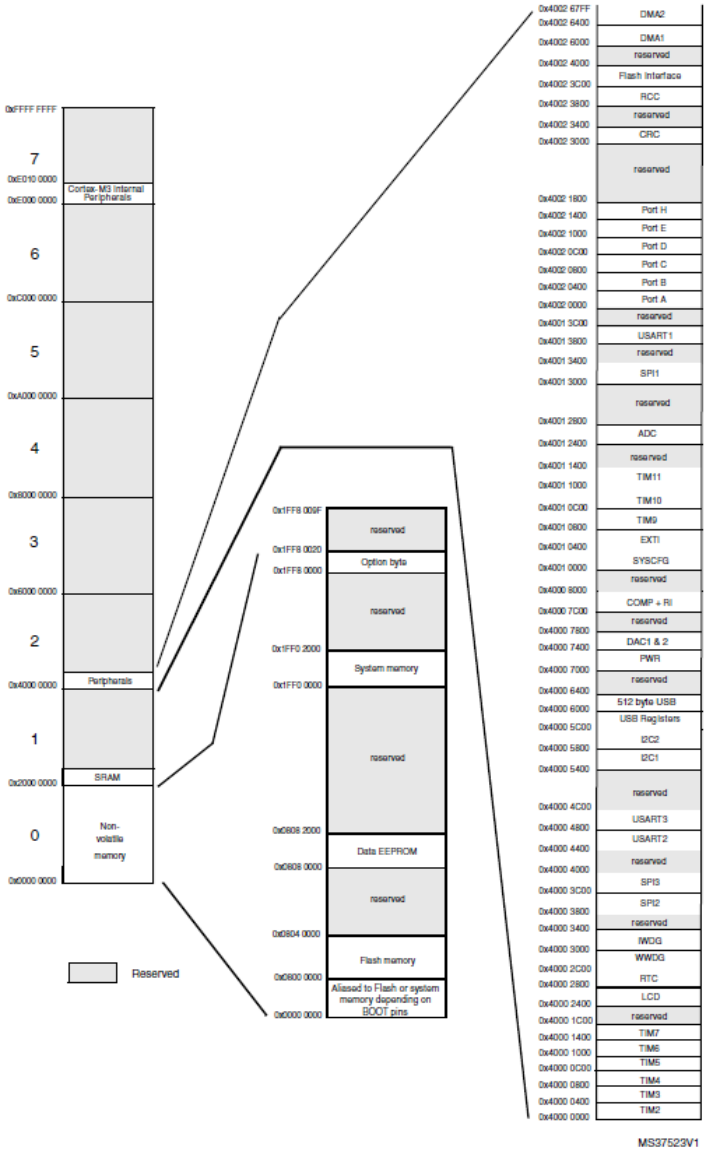
- Errata

Регистры и адреса

- Вся периферия микроконтроллера управляется с помощью записи в память по определенным адресам (регистры периферии)
- Состояние периферийных устройств можно определить с помощью чтения регистров

Memory Map

- Общий размер адресного пространства у 32-битного ядра – 4 Гб
- Для управления встроенной периферией микроконтроллера используются специальные адреса в памяти
- Встроенная память и ОЗУ проецируется в это же адресное пространство
- Обращение к памяти по несуществующему адресу приводит к остановке работы микроконтроллера



Попробуем включить и выключить светодиод

- В Reference Manual находим описание регистров GPIO
- Чтобы включить вывод («пин») микроконтроллера «на выход», надо записать единичку в определенный бит регистра MODER
- Чтобы установить на выходе определенный логический уровень, надо записать некоторое значение в регистр BSRR
- Вопросы настройки тактирования и тому подобные пока опустим

Регистр MODER

7.4.1 GPIO port mode register (GPIOx_MODER) (x = A..H)

Address offset: 0x00

Reset values:

- 0xA800 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **MODERy[1:0]:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O direction mode.
00: Input (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

Регистр BSRR

7.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..H)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x reset bit y (y = 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Resets the corresponding ODRx bit

Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BSy:** Port x set bit y (y= 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Sets the corresponding ODRx bit

Первая попытка

```
*((uint32_t*) 0x40020000) = (1UL << (5*2));
while(1) {
    delay_ms(500);
    *((uint32_t*) 0x40020018) = (1UL << 5);
    delay_ms(500);
    *((uint32_t*) 0x40020018) = (1UL << (5 + 16));
}
```

- GPIOA находится по адресу 0x40020000
- Регистр MODER – первый регистр GPIO, смещение отсутствует
- Регистр BSRR расположен со смещением 0x18, то есть по адресу 0x40020018

CMSIS

- Работать напрямую с регистрами неудобно и совершенно непереносимо между МК даже в пределах одного семейства
- Макросы **CMSIS (Cortex Microcontroller Software Interface Standard)** – набор определений, описывающих регистры микроконтроллера (их имена и адреса)

Тот же код (почти) с CMSIS

```
uint32_t tmpreg;
tmpreg = GPIOA->MODER;
tmpreg &= ~GPIO_MODER_MODER5;
tmpreg |= GPIO_MODER_MODER5_0;
GPIOA->MODER = tmpreg;
while(1) {
    delay_ms(500);
    GPIOA->BSRR = GPIO_BSRR_BS_5;
    delay_ms(500);
    GPIOA->BSRR = GPIO_BSRR_BR_5;
}
```

Hardware Abstraction Layer

- Для переносимости кода между разными МК необходимы дальнейшие уровни абстракции
  - STM32 HAL, LL
  - TI driverlib, Simplelink
  - libopencm3
  - И даже Arduino
- С помощью этих средств можно обеспечить переносимость исходного кода даже между разными семействами МК

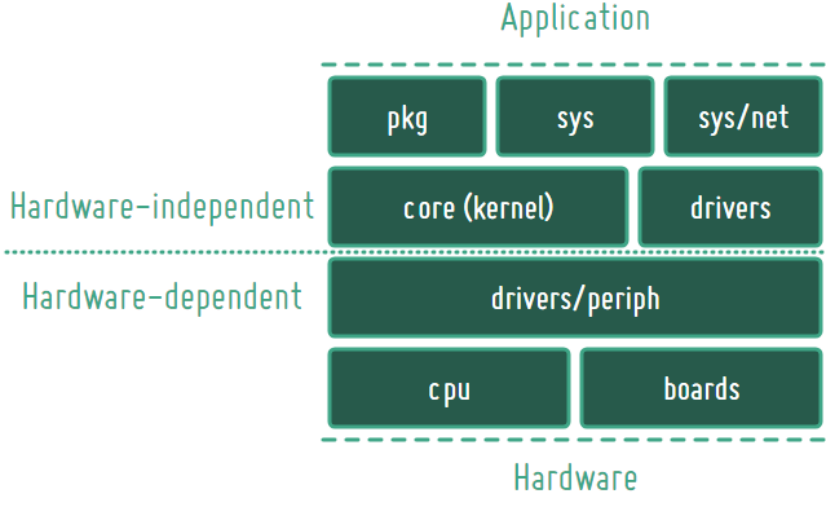
ОПЕРАЦИОННЫЕ СИСТЕМЫ ДЛЯ МИКРОКОНТРОЛЛЕРОВ

Операционные системы реального времени

- Управление ресурсами (процессорное время, память, периферия)
- Унифицированный HAL, переносимость между разными семействами МК

Операционная система RIOT OS

- Открытый исходный код, лицензия LGPL
- Разработана FU Berlin, INRIA и HAW Hamburg
- Ядро ОС
- Поддержка различных архитектур (ARM, AVR, RISC-V, …)
- Поддержка различных семейств микроконтроллеров
- Драйверы внешних устройств



Простейшая программа для RIOT

```c
#include <board.h>
#include <periph/gpio.h>

int main(void) {
    gpio_init(LED0_PIN, GPIO_OUT);
    while(1) {
        delay_ms(500);
        gpio_set(LED0_PIN);
        delay_ms(500);
        gpio_clear(LED0_PIN);
    }
    return 0;
}
```

На практическом занятии

- Как скомпилировать эту программу?
- Как загрузить ее в микроконтроллер?
- «Нулевой» блок заданий
  - Опрос стоимостью 1 балл
  - Настройка среды сборки для RIOT OS
- Консультации 03.11.2021