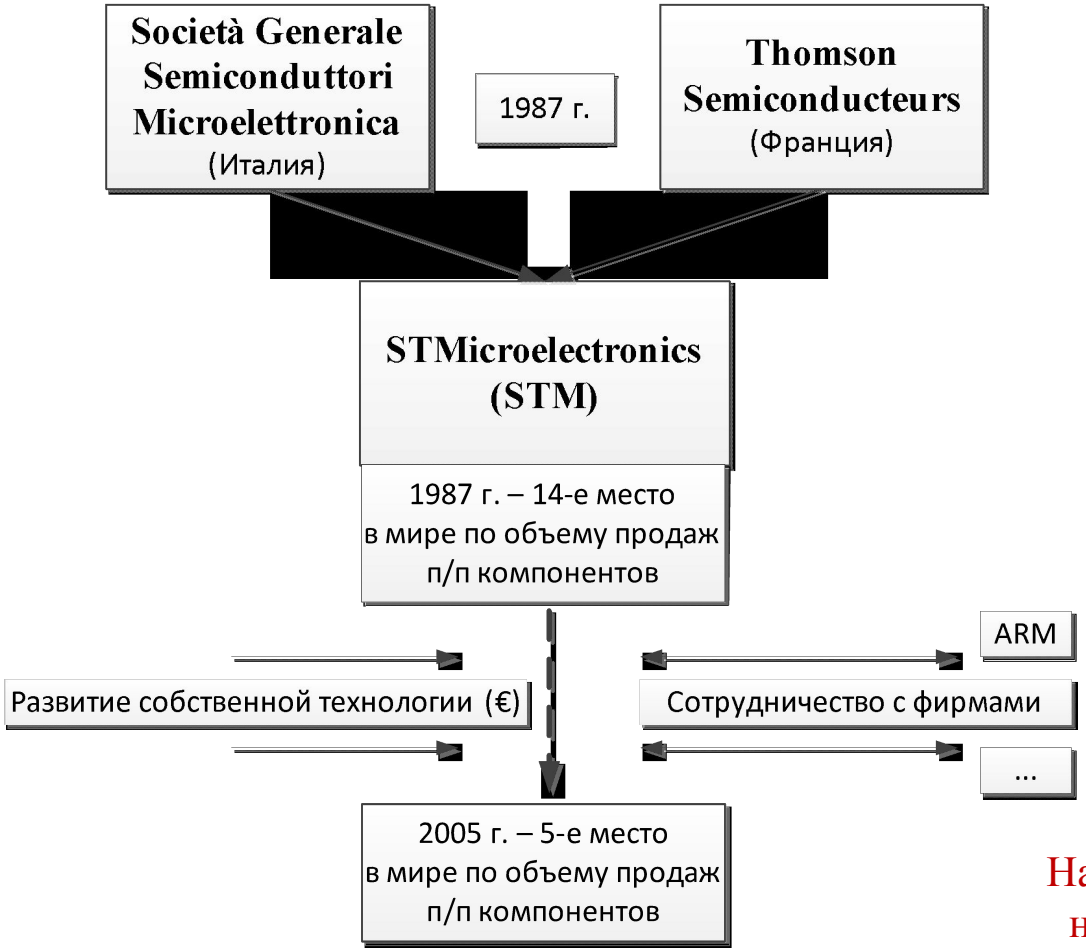
Новосибирский Государственный Технический Университет
Факультет Радиотехники и Электроники
И. А. Баховцев

ОМПТ-2

Фирма STMicroelectronics



В настоящее время фирма STM - ведущий мировой производитель п/п продукции для микроэлектроники. В компании работает более 53000 человек в 14 производственных предприятиях в 10 странах мира.

Ассортимент продукции: аналоговые и смешанные микросхемы, усилители и компараторы, диоды, устройства электромагнитной фильтрации, встроенные микропроцессоры, микроконтроллеры, модули памяти, преобразователи тока, модули питания, транзисторы, программное обеспечение и т.д.

Наиболее востребованной продукцией на рынке микроэлектроники данной фирмы являются ее МК!

Фирма ARM Limited

- **ARM Limited** - известная британская компания в области информационных технологий.
- **“ARM” - Advanced RISC Machines** (усовершенствованная **RISC**-машина).
- **ARM Limited** - один из крупных мировых разработчиков и лицензиаров 32- и 64-разрядной архитектуры RISC-процессоров, которыми оснащается большинство портативных устройств.
- **Особенность:** компания не занимается производством микропроцессоров, а лишь разрабатывает и лицензирует свою технологию другим сторонам. В частности ARM-архитектура микроконтроллеров закупается такими производителями: *Atmel, Intel, Apple, nVidia, HiSilicon, Marvell, Samsung, Sony Ericsson, Texas Instruments,* ***STMicroelectronics*** и др.

Особенности архитектуры x86 и ARM

- В МПТ, с точки зрения состава команд, существует две основные архитектуры: **CISC** (x86) и **RISC** (ARM), каждая из которых имеет свои преимущества и недостатки.
- **CISC - Complex Instruction Set Computer** (компьютер с полным [сложным] набором команд). Большое количество сложных по своей структуре команд сначала декодируются в простые, и только затем обрабатываются. На всю эту последовательность действий уходит немало энергии.
- **RISC - Reduced Instruction Set Computer** (компьютер с сокращенным набором команд). Т.е. имеется небольшой набор простых команд, которые обрабатываются с минимальными затратами энергии и с достаточно высокой производительностью.
- В практической реализации каждая из этих архитектур подвергается модификации с целью повышения характеристик процессора при решении конкретных задач.

В технической литературе **процессор**, являющийся частью цифрового устройства

Краткая характеристика серий микроконтроллеров STM32

Серии МК в группах	Наименование серии	STM32L0	STM32L1	STM32L4
Серия STM32L. Ориентирована на минимальный уровень потребления энергии с сохранением производительности.	**Ядро**	Cortex-M0+	Cortex-M3	Cortex-M4
	Частота работы	До 32 МГц	До 32 МГц	До 80 МГц
	Flash-память	16…192 кбайт	32…512 кбайт	256 кбайт…1 Мбайт
	Память RAM	До 20 кбайт	До 80 кбайт	До 320 кбайт
STM32F0, STM32F1 и STM32F3 - базовые серии. Обладают наиболее рациональными характеристиками и сбалансированным соотношением производительности, энергопотребления и цены.	**Наименование серии**	**STM32F0**	**STM32F1**	**STM32F3**
	Ядро	Cortex-M0	Cortex-M3	Cortex-M4
	Частота работы	До 48 МГц	До 72 МГц	До 72 МГц
	Flash-память	16…256 кбайт	16 кбайт…1 Мбайт	16…512 кбайт
	Память RAM	До 32 кбайт	До 96 кбайт	До 80 кбайт

Серии МК в группах	Наименование серии	STM32F2	STM32F4	STM32F7	STM32H7
STM32F2, STM32F4, STM32F7, STM32H7 Высокопроизводительные серии, предназначенные для получения максималь-ного быстродействия.	**Ядро**	Cortex-M3	Cortex-M4	Cortex-M7	Cortex-M7
	Частота работы	До 120 МГц	168…180 МГц	до 216 МГц	до 400 МГц
	Flash-память	128Кбайт… 1 Мбайт	до 2 Мбайт	до 2 Мбайт	до 2 Мбайт
	Память RAM	До 128 кбайт	До 384 кбайт	До 512 кбайт	До 1 Мбайт

Классификация серий микроконтроллеров STM32

В настоящее время STM32 производит около 300 вариантов МК, которые можно классифицировать по 2 признакам: критерию оптимизации и используемому ARM-ядру.

По критерию оптимизации:

- с пониженным энергопотреблением (энергосберегающие) **STM32L0/1/4**;
- с оптимальным соотношением производительности, энергопотребления и цены (базовые серии) **STM32F0, STM32F1 и STM32F3);**
- с высокой производительностью **STM32F2/F4/F7/H7.**

По используемому ARM-ядру:

- с ARM-ядром Cortex-M0 (**STM32L0/F0**);
- с ARM-ядром Cortex-M3 **(STM32L1/F1/F2**);
- с ARM-ядром Cortex-M4 **(STM32L4/F3/F4**);
- с ARM-ядром Cortex-M7 **(STM32F7/H7**).

Приведенные выше обозначения – это обозначение серии МК. В обозначении конкретной модели добавляются цифры и буквы.

Так, далее мы будем изучать МК типа **STM32F410RB**, относящийся к *высокопроизводительным* МК с *ARM-ядром Cortex-M4.*

Основные характеристики ядра Cortex-M4 микроконтроллеров STM32

Характеристика	Значение
Ширина слов для данных, разряд	32
Архитектура	Гарвард
Конвейер	3-ступенчатый (-уровневый)
Набор инструкций	RISC
Организация памяти программ, разряд	32
Буфер предвыборки, разряд	2x64
Средний размер инструкции, байт	2
Тип прерываний	Векторизированные
Задержка реагирования на прерывания	12 циклов
Режимы управления энергопотреблением	Sleep, Stop, Standby
Отладочный интерфейс	ST-LINK, JTAG

Микроконтроллер STM32F410RBT

1. Характеристики семейства STM32F4xx:

- Ядро ARM 32-битное **Cortex-M4**.
- Частота тактирования: 168 МГц.
- Поддержка **DSP-инструкций**.
- Новая высокопроизводительная многоуровневая **AHB-матрица шин** (переключение данных и команд, включая потоки данных от ПДП, между ЦП, ПУ и разными видами памяти).
- До 1 Мбайт флэш-памяти с ускорителем памяти (**ART-Accelerator**), обеспечивает работу с флэш-памятью с такой же скоростью, что и с ОЗУ.
- До 192 + **4 кбайт** SRAM-памяти (статической оперативной памяти), включая область резервного питания.
- Напряжение питания: 1,8–3,6 В. Режимы пониженного энергопотребления: **Sleep, Stop, Standby**. Резервное (батарейное) питание.
- Внутренние RC-генераторы на 16 МГц и 32 кГц (для **Real Time Clock - RTC**).

1. Характеристики семейства STM32F4xx (продолжение):

- Внешний источник тактирования 4–26 МГц и для RTC — 32,768 кГц.
- Модули отладки **SWD/JTAG**, модуль **ETM**.
- Аппаратный генератор случайных чисел (**Random number generator - RNG**).
- Часы реального времени (**Real Time Clock - RTC**).
- Модуль шифрования (позволяет «на лету» реализовать различные алгоритмы шифрования информации).
- Эффективная **система прерываний** по внутренним (от ПУ) и внешним событиям.
- До **трех 12-битных АЦП** на 24 входных канала.
- До двух 12-битных ЦАП.
- ПДП-контроллер на 16 потоков с поддержкой пакетной передачи (**DMA - Direct Memory Access**).

1. Характеристики семейства STM32F4xx (продолжение):

• До **17 таймеров** **(16 и 32 разряда).** Некоторые оснащены каналами входного захвата и выходного сравнения (ШИМ), соответственно IC – input capture, OC – Output compare (PWM). Один или два таймера способны реализовать функции "**Motion Control**".

• **Два сторожевых таймера** (WDG и IWDG).

• Коммуникационные интерфейсы: **I^2C, USART, SPI, I^2S, CAN (2,0), USB 2.0.**

• Интерфейс **Ethernet**.

• Контроллер карт памяти **SDIO**.

• Интерфейс цифровой камеры.

• Контроллер внешней статической памяти и графических дисплеев **FSMC**.

• **Порты ввода/вывода** общего назначения (**GPIO -** General-Purpose Input Output). Все ПУ связаны с внешним миром через порты (альтернативная функция портов).

Особенность: большая часть линий портов в режиме цифрового ввода может генерировать запрос на прерывание.

• Расширенный температурный диапазон: –40…105 °C.

1. Характеристики семейства STM32F4xx (продолжение):

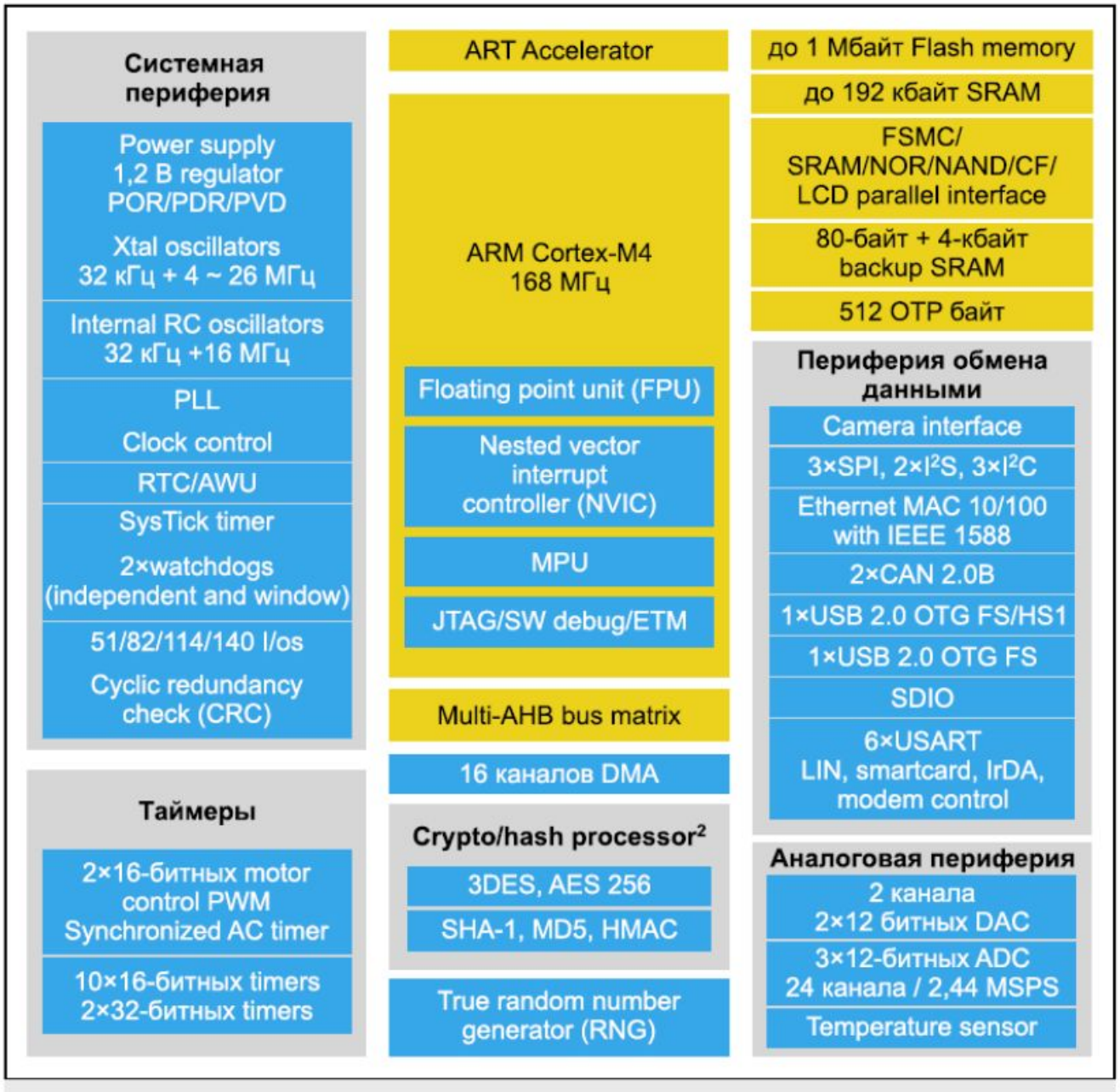
Режимы пониженного напряжения:

Режим Sleep. Только ядро останавливает свою работу. Вся периферия продолжает работать и пробуждает ЦП по наступлению определенного прерывания или события.

Режим Stop. Все тактирование в зоне 1,2 В останавливается. Все схемы высокочастотного тактирования отключаются. В данном режиме МК продолжает работать от низкоскоростных источников тактирования. Состояние SRAM и регистров при этом сохраняется. ЦП переходит в рабочий режим по заранее сконфигурированному событию.

Режим Standby. Обеспечивает самое низкое потребление. Питание 1,2 В полностью отключается. Данные SRAM и регистров не сохраняются, за исключением резервного домена и резервной SRAM. Для выхода из режима необходимо прерывание от часов RB, общий сброс или возрастающий фронт на ножке WKUP.

2. Функциональная схема STM32F4xx

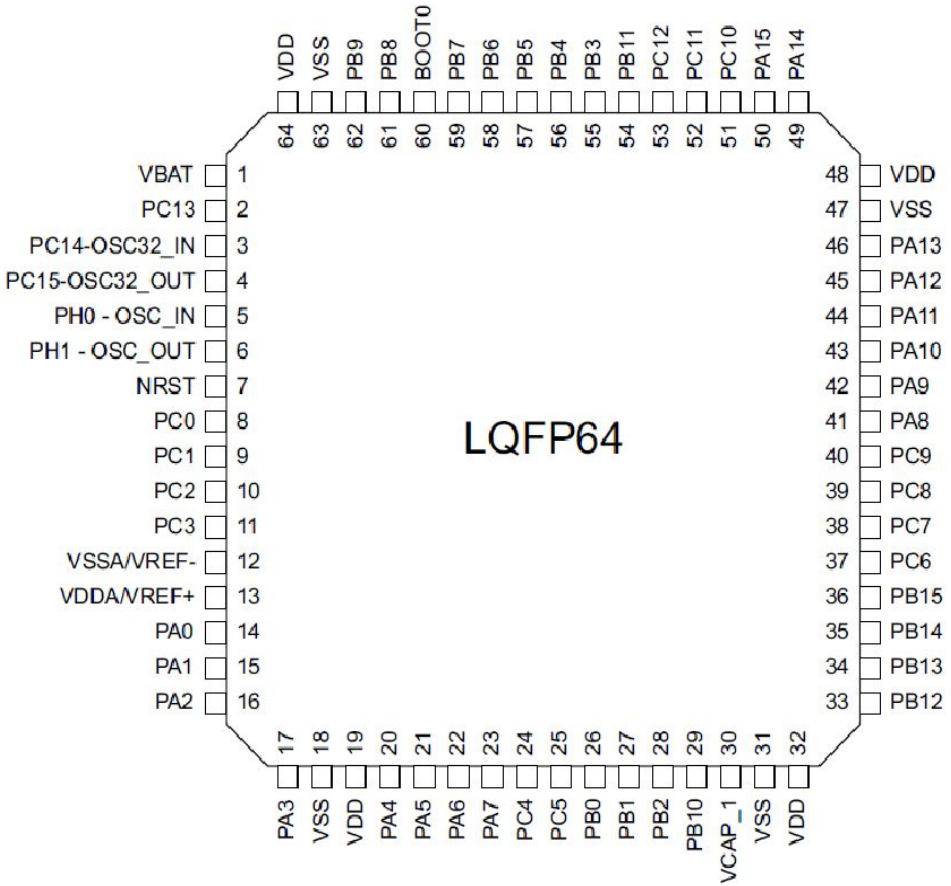


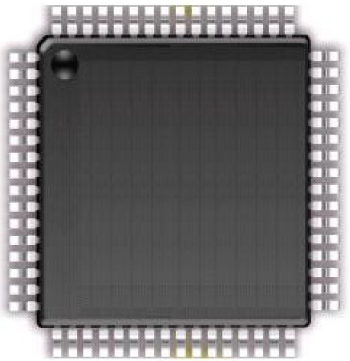
3. Порты ввода/вывода общего назначения

(GPIO – General Purpose Input/Output)

1) ***Порты I/O выполняют*** обмен информацией с внешними устройствами в параллельном коде.

2) В STM32F410RBT четыре порта I/O: **GPIOA, GPIOB, GPIOC, GPIOH.** Порты разной разрядности: **GPIOA, GPIOB, GPIOC** по 16 выводов, **GPIOH** – 2 вывода. Остальные 14 линий (из 64-х в корпусе)– питание, системные входы.





3) ***Назначение выводов, не относящихся к портам***:

- **VSS/VDD** – цифровая земля/цифровое питание;
- **VBAT** – дополнительное батарейное питания (для резервной области питания);
- **NRST** – двунаправленная линия системного сброса (RESET)
- **VSSA/VREF-** – аналоговая земля/минусовой вывод опорного напряжения АЦП и ЦАП;
- **VDDA/VREF+** – аналоговое питание/ плюсовой вывод опорного напряжения АЦП и ЦАП;
- **VCAP_1** – вывод для подключения внешнего конденсатора для внутреннего регулятора напряжения;
- **BOOT0** – вход, определяющий режим МК после сброса: запуск/загрузка ПО.

4) ***Основные режимы работы портов*** : "Цифровой ввод/вывод" (ЦВ/В), "Периферийный ввод/вывод" (ПВ/В).

ПВ/В - ***альтернативная функция портов*** – возможность использования линий портов как входных/выходных линий внутренних ПУ.

Альтернативная функция (АФ) может быть аналоговой и цифровой.

5) *Основные свойства портов ввода/вывода*

- До 16 линий ввода/вывода могут находиться под управлением порта;
- Для каждой линии любого порта можно выбрать свою скорость тактирования;
- «На лету» можно менять режим работы линии порта. На смену режима уходит около двух тактов.
- Механизм блокировки, приводящий к «замерзанию» конфигурации линии I/O, т.е. ее нельзя изменить в процесс работы;
- Гибкое мультиплексирование позволяет использовать линии I/O как GPIO или как одну из нескольких АФ;
- Большинство выводов I/O толерантны к + 5 В.
- Выходные цифровые состояния: двухтактное (push-pull), с открытым стоком (open drain) и с подтягивающими резисторами вверх/вниз (pull-up/pull-down);
- Входные цифровые состояния: плавающее (floating), или Z-, высокоимпедансное состояние; вход с подтягивающими резисторами вверх/вниз (pull-up/pull down);
- С помощью коротких команд и специальных регистров можно производить индивидуальную модификацию выходного состояния любой лини порта.
- Большая часть линий портов в режиме цифрового ввода может генерировать запрос на прерывание.

6) *ПЛМ портов I/O*:

- **GPIOx_MODER –** регистр режима работы;
- **GPIOx_OTYPER –** задания типа выхода;
- **GPIOx_OSPEEDR –** регистр задания частоты тактирования каждого вывода;
- **GPIOx_PUPDR –** регистр управления подтягивающими резисторами;
- GPIOx_IDR – регистр входных данных;
- GPIOx_ODR – регистр выходных данных;
- **GPIOx_BSRR –** регистр установки/сброса разрядов (линий) порта;
- **GPIOx_LCKR –** регистр блокировки конфигурации порта;
- **GPIOx_AFRL –** нижний регистр альтернативной функции;
- **GPIOx_AFRH –** верхний регистр альтернативной функции.

***x** – имя порта (= A, B, C, H).*

Количество регистров управления портами (8 шт.) говорит о сложности и многофункциональности портов ввода-вывода в МК.

7) Описание регистров ПЛМ портов I/O

Регистры GPIO 32-битные, но доступны байтами, полусловами (16 бит) и словами. Формат регистров позволяет настроить каждую линию или модифицировать ее состояние индивидуально. В состав ПЛМ портов входит 10 регистров. **После сброса АФ неактивны (кроме АФ отладки), а все линии портов настроены на цифровой ввод.**

- ***Регистр режима (GPIOx_MODER) (x = A, B, C, H)*** **–** задает режим работы линий (разрядов) порта **x**.

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

Биты (2y+1):2y – **MODERy**[1:0]: биты режима порта **x** (y=0…15).

00: цифровой вход (или АФ отладки); 10: режим АФ;

01: цифровой выход; 11: аналоговый режим (ДФ-доп. ф-ция).

Значение при сбросе: 0x0C00 0000 для порта A,

0x0000 0280 для порта B,

0x0000 0000 для портов C и H.

Ненулевые разряды регистра после сброса настраивают соответствующие линии портов A и B в АФ как выводы интерфейса отладки.

- ***Регистр типа выхода (GPIOx_OTYPER)***

  Биты 31:16 – резерв, всегда читаются как 0x0000.

  Биты 15:0 **– OTy**: биты конфигурации линий порта **x** (y=0…15)

  0: выход push-pull (PP); **1:** выход с открытым стоком (OD).

- ***Регистр скорости выхода (GPIOx_OSPEEDR)***

  Биты (2y+1):2y **– OSPEEDRy**[1:0]: Биты настройки скорости выходов порта **x** (y = 0..15).

  00: низкая скорость (8 МГц); 10: высокая скорость (50 МГц);

  01: средняя скорость (25 МГц); 11: очень высокая скорость (100 МГц).

  Конкретные цифры в datasheet. В скобках значения при VDD>2,7 V

  Значение при сбросе: 0x0C00 0000 для порта A,

  0x0000 00C0 для порта B,

  0x0000 0000 для портов C и H.

- ***Регистр подтягивания вверх/вниз (pull-up/pull-down) линий порта (GPIOx_PUPDR)***

  Биты (2y+1):2y **– PUPDRy[1:0]** Биты настройки порта **x** (y = 0..15)

  00: нет подтягивания; 10: подтягивание вниз;

  01: подтягивание вверх; 11: резерв.

  Значения при сбросе: 0x6400 0000 для порта A;

  0x0000 0100 для порта B;

  0x0000 0000 для портов C и H.

Таблица конфигурации разрядов портов I/O

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]		PUPDR(i) [1:0]		I/O configuration	
01	0	SPEED [B:A]		0	0	GP output	PP
	0			0	1	GP output	PP + PU
	0			1	0	GP output	PP + PD
	0			1	1	Reserved	
	1			0	0	GP output	OD
	1			0	1	GP output	OD + PU
	1			1	0	GP output	OD + PD
	1			1	1	Reserved (GP output OD)	
10	0	SPEED [B:A]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0		
	x	x	x	1	1		

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

- ***Регистр входных данных (GPIOx_IDR)***

Биты 31:16 – резерв. Биты 15:0 – **IDRy**: входные данные порта **x** (y = 0..15).

Эти биты только читаются и только в режиме слова. Они содержат **логические состояния на входе** соответствующего порта, которые могут определяться как внешними источниками (разряды настроены на цифровой ввод), так и состояниями выходного регистра данных порта (разряды настроены на цифровой вывод).

Значения при сбросе: 0x0000 XXXX (x - неопределенное значение).

Почему такое начальное состояние?

- ***Регистр выходных данных (GPIOx_ODR)***

Биты 31:16 – резерв. Биты 15:0 – **ODRy**: выходные данные порта **x** (y = 0..15).

Эти биты могут быть прочитаны и записаны в программе.

- ***Регистр установки/сброса (set/reset) битов (GPIOx_BSRR)***

Биты 31:16 – **BRy:** Бит сброса разряда **y** порта **x** (y = 0..15).

Эти биты только записываются: в режиме слова, полуслова или байта.

0/1 => Нет воздействия/сброс одноименного бита **y** в регистре вых. данных ODRx;

Биты 15:0 – **BSy:** Бит установки разряда **y** порта **x** (y= 0..15).

Эти биты только записываются: в режиме слова, полуслова или байта.

0/1 => Нет воздействия/установка одноименного бита **y** в регистре вых. данных ODRx.

Замечание - Если оба бита BSy и BRy установлены, то BSy имеет приоритет.

Чтение этих бит всегда возвращает 0x0000 0000 (одноразовое действие).

- ***Регистр блокировки конфигурации линии порта (GPIOx_LCKR)***

В процессе работы МК режимы каких-то линий портов могут меняться, Чтобы при этом случайно не изменился режим других линий, требуется их защита. Для этого - регистр блокировки конфигурации **GPIOx_LCKR).**

Каждому выводу порта соответствует (по номеру) бит блокировки LCK0 – LCK15. При установке бита **y** в 1 запрещается изменение заданного режима и соответствующей конфигурации линии **y** данного порта.

В процессе инициализации ПО после задания всех нужных битов регистра блокировки необходимо активизировать защиту. Для этого в 16-й бит регистра блокировки (LCKK) последовательно записывают 1, 0, 1. Запись должна производиться 32-битным словом. При этом состояние битов 15:0 не должно меняться, иначе блокировка сбрасывается. После этого защита будет действовать, и изменение конфигурации и режима защищенных битов (линий порта) будет игнорироваться вплоть до сброса МК.

Биты 31:17 - резерв;

Бит 16 **– LCKK**: ключ блокировки;

Этот бит можно прочитать в любой момент времени. Он может быть модифицирован, только используя последовательность записи ключа блокировки.

0: Ключ блокировки конфигурации порта не активен;

1: Ключ блокировки конфигурации порта активен.

Биты 15:0 **– LCKy:** бит блокировки разряда **y** порта **x** (y=0…15).

Эти биты можно прочитать/записать, но записаны они м.б, только когда бит LCKK=0.

0: Конфигурация разряда **y** порта **x** не защищена;

1: Конфигурация разряда **y** порта **x** защищена.

- ***Нижний/верхний регистры альтернативной функции GPIO (GPIOx_AFRL)/ (GPIOx_AFRH)***

В МК серии TMS320F4xx (с учетом дальнейших разработок) предусмотрено 16 АФ. Для задания номера этой функции требуется 4 двоичных разряда (тетрада). Теоретически к каждой линии порта м.б. подключена любая из 16 АФ. => Для задания каждой линии порта своей АФ необходимо 4*16=64 разряда. Эти разряды распределены между двумя 32-битными регистрами GPIOx_AFRL (нижним) и GPIOx_AFRH (верхним).

В нижнем регистре тетрады AFRL7:AFRL0 задают АФ линиям 7:0 порта, а в верхнем регистре тетрады AFRH7:AFRH0 задают АФ соответственно линиям 15:8 порта.

Подключение АФ к той или иной линии порта осуществляется двумя (на младший и старший байты порта), мультиплексорами 16*8, управляемыми соответственно нижним и верхним регистрами альтернативной функции.

В каждом конкретном МК – свой (в соответствии с набором ПУ) набор АФ.

Для обоих регистров значение при сбросе: 0x0000 0000 (т.е. настройка на АФ0).

Номер АФ	Содержание АФ	Номер АФ	Содержание АФ
0000: АФ0	системная	1000: АФ8	USART6
0001: АФ1	TIM1/LPTIM1	1001: АФ9	$I^2C2/4$
0010: АФ2	TIM5	1010: АФ10	резерв
0011: АФ3	TIM9/11	1011: АФ11	резерв
0100: АФ4	$I^2C1/2/4$	1100: АФ12	резерв
0101: АФ5	SPI1/2, $I^2S1/2$	1101: АФ13	резерв
0110: АФ6	SPI1/2/5, $I^2S1/2/5$	1110: АФ14	резерв
0111: АФ7	USART1/2	1111: АФ15	EVENTOUT

EVENTOUT - выходное событие ядра МК (Cortex®-M4 с FPU), для активации других ЦП в многопроцессорных МПСУ.

- Однако к настоящему времени данный мультиплексор еще не соответствует требованию «на любой вывод – любую АФ». К каждой линии портов через MUX подкреплены свой набор АФ, причем в разных портах эти наборы также отличаются.
- Чтобы правильно использовать АФ, необходимо смотреть техдокументацию на конкретный МК (*datasheet*), где подобная информация представлена. В нашем случае она – в раздаточных материалах к лабораторным работам.

Дополнительная функция

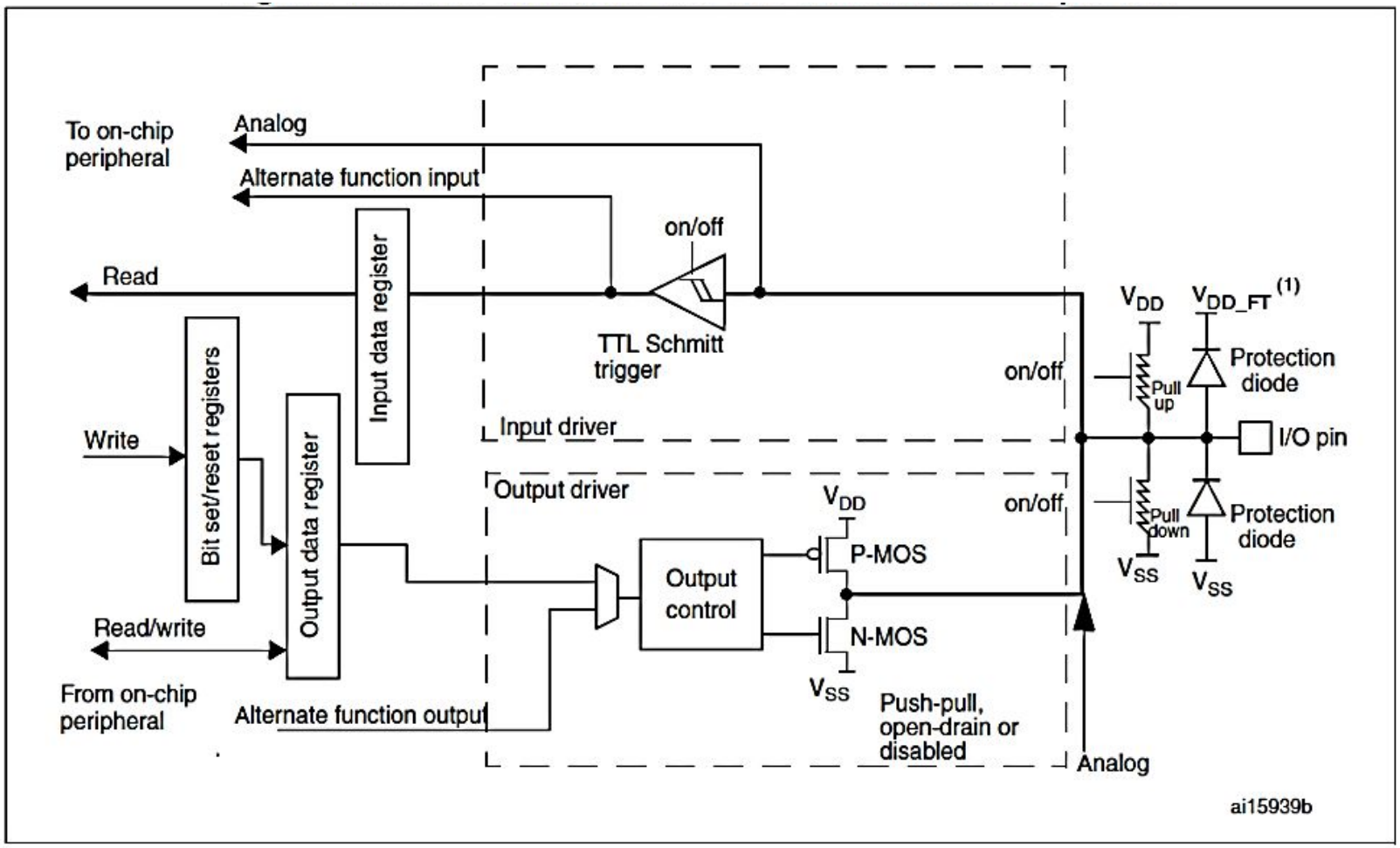
Дополнительная функция – аналоговый режим работы линий портов. В этом случае цифровой мультиплексор не задействован, и аналоговые каналы (входные для АЦП и выходной для ЦАП) жестко привязаны к соответствующим линиям портов:

Аналог. Канал	ADC1_0	ADC1_1	ADC1_2	ADC1_3	ADC1_4	ADC1_5, DAC_OUT1	ADC1_6	ADC1_7
Порт	PA0	PA1	PA2	PA3	PA4	PA5	PA6	PA7
Аналог. канал	ADC1_8	ADC1_9	ADC1_10	ADC1_11	ADC1_12	ADC1_13	ADC1_14	ADC1_15
Порт	PB0	PB1	PC0	PC1	PC2	PC3	PC4	PC5

Замечание 1 – 1-я цифра в наименовании аналогового канала говорит о номере АЦП. В данном семействе их может быть три. В данном МК есть только один АЦП – первый.

Замечание 2 - Использование и настройку линий PC13-PC15 и PH0, PH1, предусмотренных для организации внутренних и внешних ГТИ, опускаем. Их лучше в своих программах не использовать. Их настройка будет выполняться в системных библиотечных файлах, добавленных в ваш проект

Базовая структура разряда порта I/0, толерантного +5 В



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

10) Операции над отдельными разрядами регистров ПЛМ портов I/O. Рекомендации к программам.

- Часто в МПСУ приходится устанавливать или анализировать отдельные разряды: настройка режимов работы ПУ, оценка состояния ПУ или отдельных линий портов ввода.
- Однако прямых операций над битами в том компиляторе СИ и в том наборе библиотек, которые мы используем, нет.
- Но аналогичные операции можно выполнить, используя операции маск***и***рования.

  Маск*и*рование это:

  - модификация одного или нескольких битов в регистре без изменения состояния других его битов с помощью слова-маски;
  - выделение одного или нескольких битов в регистре с обнулением других его битов с помощью слова-маски.
- **Модификация бита**: установка в ноль, в единицу, инвертирование.
- В основе – логические операции.

Свойства логических операций, используемые при маскировании		
ИЛИ	И	Исключ. ИЛИ
$x+0=x$ $x+1=1$	$x\cdot 0=0$ $x\cdot 1=x$	$x\oplus 0=x$ $x\oplus 1=\overline{x}$

10) Операции над отдельными разрядами регистров ПЛМ портов I/O. Рекомендации к программам (*продолжение*).

- В файле «STM32f1410rx.h» для каждого разряда портов и регистров ПУ прописана своя 16-/32-битная **слово-маска**.
- **Пример 1: маски разрядов выходного регистра данных**

  Бит 0 - **GPIO_ODR_OD0** =0x0001

  Бит 1 - **GPIO_ODR_OD1** =0x0002

  Бит 2 - **GPIO_ODR_OD2** =0x0004

  Бит 15 - **GPIO_ODR_OD15** =0x8000

  Для входного регистра данных: ODR --> IDR, OD --> ID.

- **Пример 2: маски разрядов регистра установки/сброса (32 бита)**

  Бит BS0 - **GPIO_BSRR_BS0** =0x0000 0001

  Бит BS1 - **GPIO_BSRR_BS1** =0x0000 0002

  Бит BS2 - **GPIO_BSRR_BS2** =0x0000 0004

  Бит BS15 - **GPIO_BSRR_BS15** =0x0000 8000

  Бит BR0 - **GPIO_BSRR_BR0** =0x0001 0000

  Бит BR1 - **GPIO_BSRR_BR1** =0x0002 0000

  Бит BR2 - **GPIO_BSRR_BR2** =0x0004 0000

  Бит BR15 - **GPIO_BSRR_BR15**=0x8000 0000

10) Операции над отдельными разрядами регистров ПЛМ портов I/O. Рекомендации к программам (*продолжение*).

Обобщенные выражения для операций маскирования над одним разрядом:

- Установка в 1 бита **y** регистра REG порта **x**:

```
GPIOx->REG |= GPIO_REG_NAMEy;
```

- Инверсия бита **y** регистра REG порта **x**:

```
GPIOx->REG ^= GPIO_REG_NAMEy;
```

- Сброс в 0 бита **y** регистра REG порта **x**:

```
GPIOx->REG &=~ GPIO_REG_NAMEy;
```

В данных выражениях представлены слева направо имена: порта, регистра, разряда.

Формирование меандра программным способом на линии 5 порта GPIOC. Варианты

- использованием регистра установки и сброса:

```
while (1)
    {
    GPIOC->BSRR |= GPIO_BSRR_BS5; // установка в 1 бита BS5, т.е.  5-го бит порта C.
    for (i=0; i<800; i++)                  // программная задержка.
      asm("nop");
    GPIOC->BSRR |= GPIO_BSRR_BR5; // установка в 1 бита BR5, что сбрасывает
                                  // 5-й бит порта C.
    for (i=0; i<800; i++)                  // программная задержка.
      asm("nop");
    }
```

10) Операции над отдельными разрядами регистров ПЛМ портов I/O. Рекомендации к программам (*продолжение*).

- записью в порт соответствующего числа и нуля;

```
while (1)
        {
      GPIOC->ODR = 0x0020;    // недостаток?
      for (i=0; i<800; i++)
           asm("nop");
      GPIOC->ODR = 0x0000;    // недостаток?
      for (i=0; i<800; i++)
            asm("nop");
        }
```

- установкой бита в 1 и обнулением бита;

```
while (1)
        {
      GPIOC->ODR |= GPIO_ODR_OD5;
      for (i=0; i<800; i++)
           asm("nop");
      GPIOC->ODR &=~ GPIO_ODR_OD5;
      for (i=0; i<800; i++)
           asm("nop");
        }
```

10) Операции над отдельными разрядами регистров ПЛМ портов I/O. Рекомендации к программам (*продолжение*)..

- установкой бита в 1 и инверсией бита;

```
while (1)
        {
      GPIOC->ODR |= GPIO_ODR_OD5;
      for (i=0; i<800; i++)
            asm("nop");
      GPIOC->ODR ^= GPIO_ODR_OD5;
      for (i=0; i<800; i++)  {
            asm("nop");   }
        }
```

- одной инверсией бита (!)

```
while (1)
        {
      GPIOC->ODR ^= GPIO_ODR_OD5;
      for (i=0; i<800; i++)
           asm("nop");
        }
```

10) Операции над отдельными разрядами регистров ПЛМ портов I/O. Рекомендации к программам (*продолжение*).

Работа с кнопкой пользователя (PC13)

```
while (1)
      {
    if(GPIOC->IDR & GPIO_IDR_ID13)  //анализ маскирования разряда PC13
    {
            GPIOC->ODR ^= GPIO_ODR_OD5;
            for (i=0; i<800; i++)
    }
    else GPIOC->ODR &=~ GPIO_ODR_OD5;
      }
```

Продолжение следует