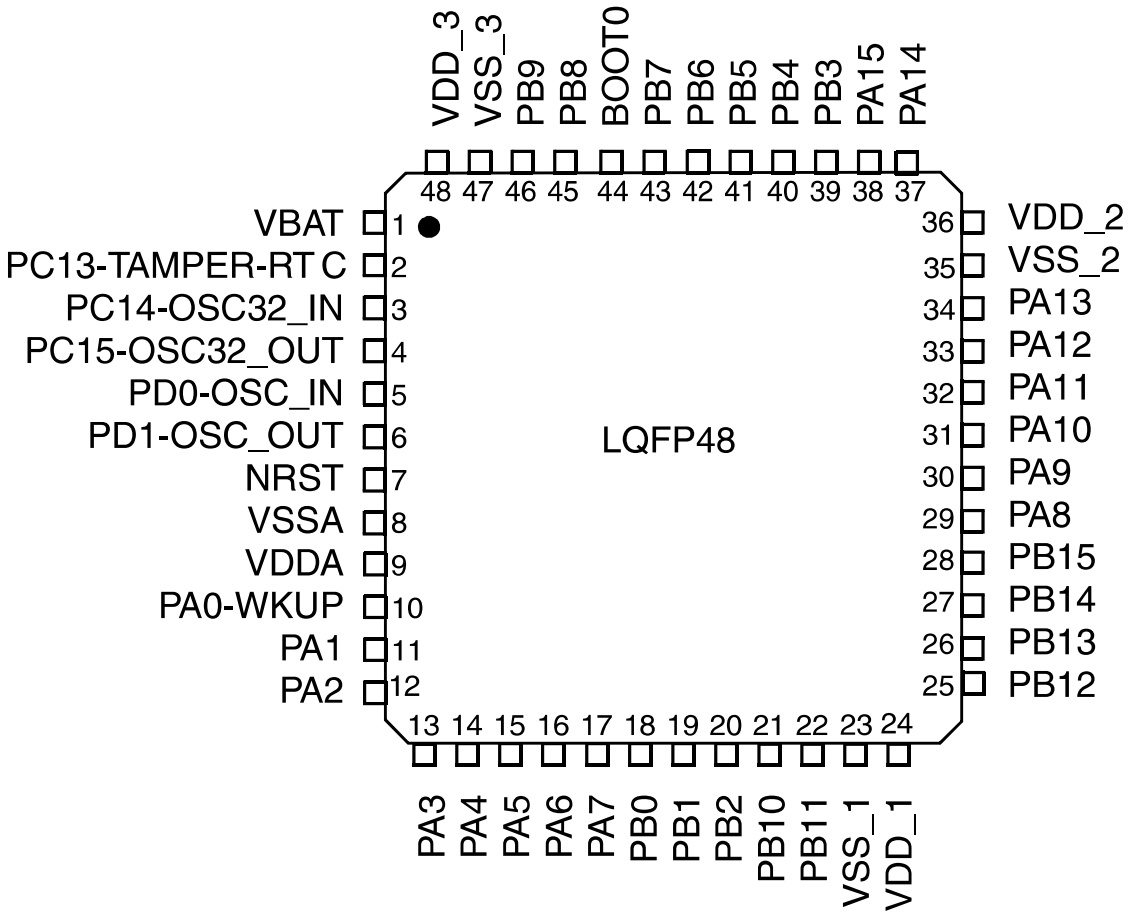
Порты ввода/вывода

Все внешние схемы, с которыми взаимодействует микроконтроллер подключены к портам ввода/вывода. Мы их уже рассматривали в разделе «**Микроконтроллер под микроскопом**». Сейчас же наша цель посмотреть регистры портов и разобраться как их настраивать.

В микроконтроллерах STM32 все ножки сгруппированы в порты и обозначаются буквами: A, B, C, D и т.д. В библиотеке CMSIS под них выделены структуры вида `GPIOx`. Например: третья ножка порта A. В каждом порте может быть до 16 ножке. Словосочетание «может быть» не случайно — физическое количество выводов МК ограничено, а значит не все они реализованы в железе. Конкретно в **stm32f103c8** имеется 48 выводов и присутствуют порты A, B, C и D.



Изображение из Datasheet, *раздел Pinouts and pin description, корпус LQFP48*

Наряду с обычными портами присутствуют и специальные выводы — питание, сброс и `boot0`. Их нельзя использовать для управления внешними цепями. Часть ножек микроконтроллера отведены для программатора, в нашем случае PA13 отвечает за SWDIO, а PA14 за SWCLK. Плюс ко всему по-умолчанию включен и другой интерфейс отладки, JTAG. Он задействует ножки PAx. Все эти выводы можно использовать как обычные порты ввода-вывода, т.е. ими можно управлять внешними цепями, но для этого их нужно переключить в «обычный» режим, а после этого перепрошить устройство у вас не получится, только изменив конфигурацию загрузки (ножки `boo0` и `boo1`).

> Если у вас не хватает ножек, вы вполне можете задействовать, например SWDIO, просто поставьте на время разработки пятисекундную задержу при старте, что бы у вас оставалась возможность прошить устройство до запуска основной программы. В наборе не используются такие ножек, но чуть ниже будет рассказано, как их переключить.

Откройте **Reference Manual** и найдите в оглавлении пункт **General-purpose I/Os (GPIO)**. С текстовым описанием вы можете ознакомиться самостоятельно, но перед тем как перейти к подразделу **GPIO Registers**, вернитесь к разделу «**Микроконтроллер под микроскопом**» или в **Reference Manual** и ещё раз рассмотрите структуру порта.

Первые два регистра, которые мы встречаем, называются `GPIOx_CRL` (`L`, нижний) и `GPIOx_CRH` (`H`, высокий).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[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
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Регистр `CRL` из Reference Manual

Каждой ножке порта ставится в соответствие 4 бита: два `MODE` и два `CNF`. Первые 8 описываются в нижнем регистре, вторые восемь в верхнем. Например нам нужно настроить пятую ножку порта A, тогда нужно записать некоторые значения в биты с 20 по 23 (нумерация с нуля). Понять что именно нужно записать, можно всё из того же **Reference Manual**. Под таблицей регистров располагается описание битов. Посмотрите внимательно и определите, что нужно записать в биты `MODE5` и `CNF5`, чтобы настроить ножку на вход с плавающим выходом.

Bits 31:30, 27:26, 23:22, 19:18, 15:14, 11:10, 7:6, 3:2 **CNFy[1:0]:** Port x configuration bits (y= 0 .. 7)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 20: Port bit configuration table*.
In input mode (MODE[1:0]=00):
00: Analog mode
01: Floating input (reset state)
10: Input with pull-up / pull-down
11: Reserved
In output mode (MODE[1:0] > 00):
00: General purpose output push-pull
01: General purpose output Open-drain
10: Alternate function output Push-pull
11: Alternate function output Open-drain

Bits 29:28, 25:24, 21:20, 17:16, 13:12, 9:8, 5:4, 1:0 **MODEy[1:0]:** Port x mode bits (y= 0 .. 7)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 20: Port bit configuration table*.
00: Input mode (reset state)
01: Output mode, max speed 10 MHz.
10: Output mode, max speed 2 MHz.
11: Output mode, max speed 50 MHz.

Описание настройки регистра `CRL` из Reference Manual

Сперва следует заполнить биты `MODE`. Так как нам нужен вход, туда следует установить `00`. Далее в биты `CNF` записать `01`. Не трудно заметить, что в скобочках у нужных нам значений указано «reset state», т.е. значение по-умолчанию, т.е. то, которое будет выставлено при сбросе МК. Если данную ножку вы будите использовать только в таком режиме, то настраивать её в не обязательно, однако ножка может быть настроена по другому и её состояние нужно будет изменить во время работы. Сделать это можно уже известным нам способом: применить логические операции и взять макросы из файла `stm32f10x.h`.

```
GPIOA->CRL &= ~GPIO_CRL_MODE5;  // reset MODE bits to 00
GPIOA->CRL &= ~GPIO_CRL_CNF5_0; // set CNF frist bit to 1
GPIOA->CRL &= ~GPIO_CRL_CNF5_1; // set CNF second bit to 0
```

Если ножка была настроена на вход, то считать значение логического уровня на ней можно через регистр `IDR` (input data register). Каждой ножке соответствует один бит. Так как мы говорили о первой ножке порта, то и реагировать нужно на первый (второй по счёту) бит.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Регистр `IDR` из Reference Manual

Применив операцию логического умножения значения регистра с маской мы гарантировано затерём все значения и оставим единицу, если она там была, на месте первого бита.

```
if (GPIOA->IDR & GPIO_IDR_IDR5) {
    // do something
}
```

Так как в Си всё что не равняется нулю считается истиной, то условный оператор сработает где бы бит не находился.

В случае когда ножка настроена на выход, нужно использовать другой регистр `ODR` (output data register). Его строение сходно с регистром входа. Записав `1` в нужной позиции, на соответствующей биту ножке будет выставлен высокий логический уровень.

> Здесь стоит отметить, что максимально допустимый выходной ток для одной ножки — 25 мА, а выходной ток на весь порт 160 мА. Таким образом, пытаться питать «прожорливую» нагрузку через МК может стать фатальным для системы питания порта, после чего он просто перестанет работать. Используйте транзисторный ключ для коммутации таких нагрузок как реле.

Если вы рассмотрели функциональную схему порта, то должны были заметить, что регистр `ODR` подключен через блок set/reset register. Следующие два регистра в документации `BSRR` и `BRR`. Может возникнуть резонный вопрос, — «Для чего ещё два регистра установки значений, когда есть `ODR`?». Ответ не совсем очевидный для новичка. Проблема здесь в том, что когда вы записываете в регистра `ODR` вам приходится его считывать. Если в это время произойдёт асинхронное к программе событие (о них мы поговорим позже), то операция будет на некоторое время прервана. В некоторых случаях такое поведение является недопустимым. Для этой цели ST предоставляет два отдельных регистра, в которые можно записывать значения, не считывая сам регистр.

```
GPIOA->BRR = GPIO_BSRR_BS3;
```

В данной строчке мы записываем маску сразу в регистр. Нули будут проигнорирована, а вот единица во второй (третьей) позиции скажет, что нужно переключить третью ножку порта A.

Последний регистр, `LCKR` довольно не обычный, он позволяет заблокировать конфигурацию каждой ножки в отдельности. Последний бит, шестнадцатый (семнадцатый) отводится под «замок». Подав на неё специальную последовательность вы можете активировать блокировку конфигурации порта. Мы не будем пользоваться данной функциональностью.

У ножки может быть альтернативная функция. При записи `00` в `CNFx` она автоматически работать не станет, нужно как минимум включить тактирование блока альтернативных функций.

```
RCC->APB2ENR |= RCC_APB2ENR_AFIOEN;
```

Затем тактирование нужного блока, допустим это будет `SPI2` .

```
RCC->APB2ENR |= RCC_APB2ENR_SPI2EN;
```

После этого настроить порт и периферию. Однако... такое сработает не всегда. Дело в том, что альтернативные функции ножек могут быть подключены хитрым способом и их нужно переназначить (англ. remap), т.е. переназначать. Например ножки PC14 и PC15 по-умолчанию работают как вход и выход низкочастотного кварцевого резонатор (LSE). Если вам не хватает ножке, то их можно использовать как обычные GPIO. Тоже касается ножек интерфейса отладчика. Все эти настройки совершаются через регистр `MAPR` , но мы не будем задевать его в данном устройстве.

Быстро определить что и куда можно переназначить можно через документ **Datasheet**, в разделе **Pinouts and pin description**.

Pins							Pin name	Type(1)	I / O Level(2)	Main function(3) (after reset)	Alternate functions(4)	
LFBGA100	UFBG100	LQFP48/UFQFPN48	TFBGA64	LQFP64	LQFP100	VFQFPN36					Default	Remap
A3	B2	-	-	-	1	-	PE2	I/O	FT	PE2	TRACECK	-
B3	A1	-	-	-	2	-	PE3	I/O	FT	PE3	TRACED0	-
C3	B1	-	-	-	3	-	PE4	I/O	FT	PE4	TRACED1	-

Таблица из Datasheet, *Table 5. Medium-density STM32F103xx pin definitions*

Как видите в отдельной колонке выписаны все возможные переназначаемые функции ножки.

Назад | Оглавление | Дальше