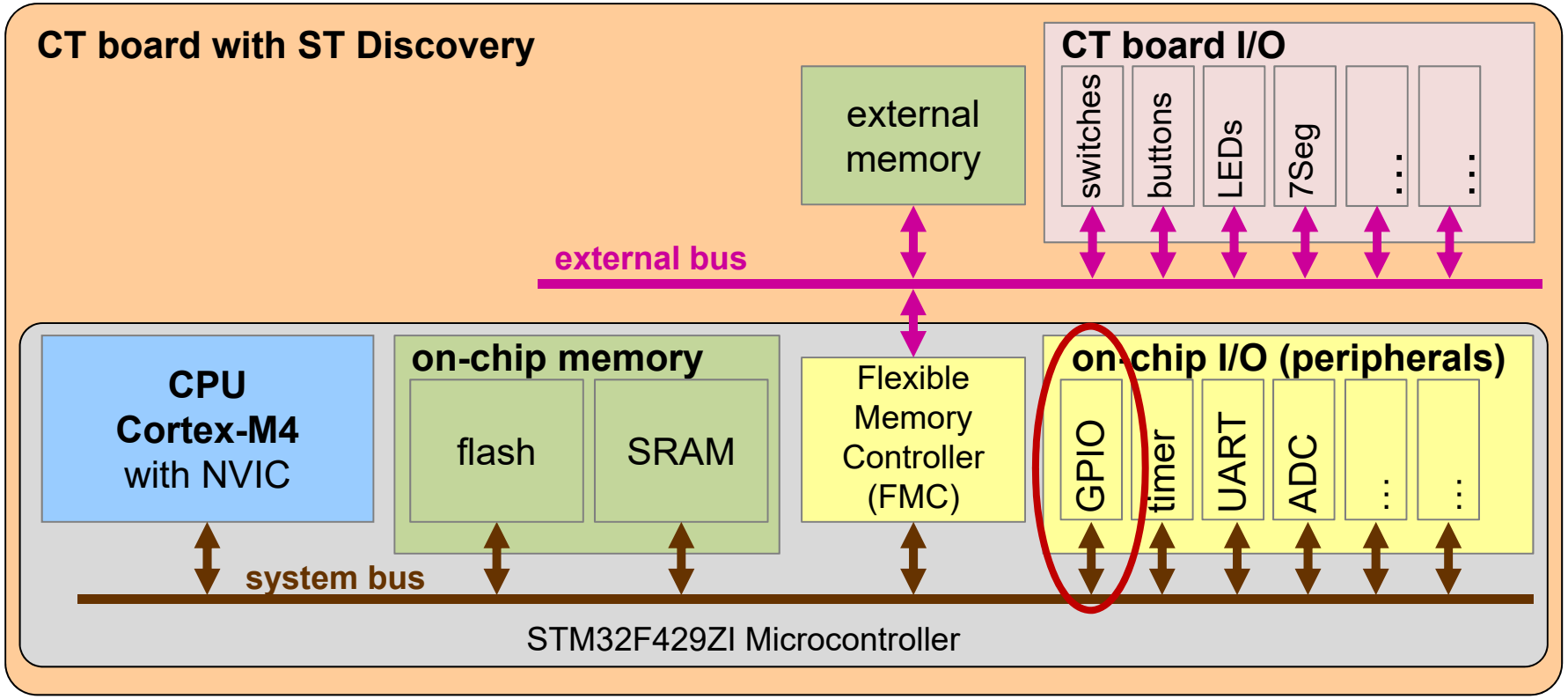
General Purpose I/O (GPIO)

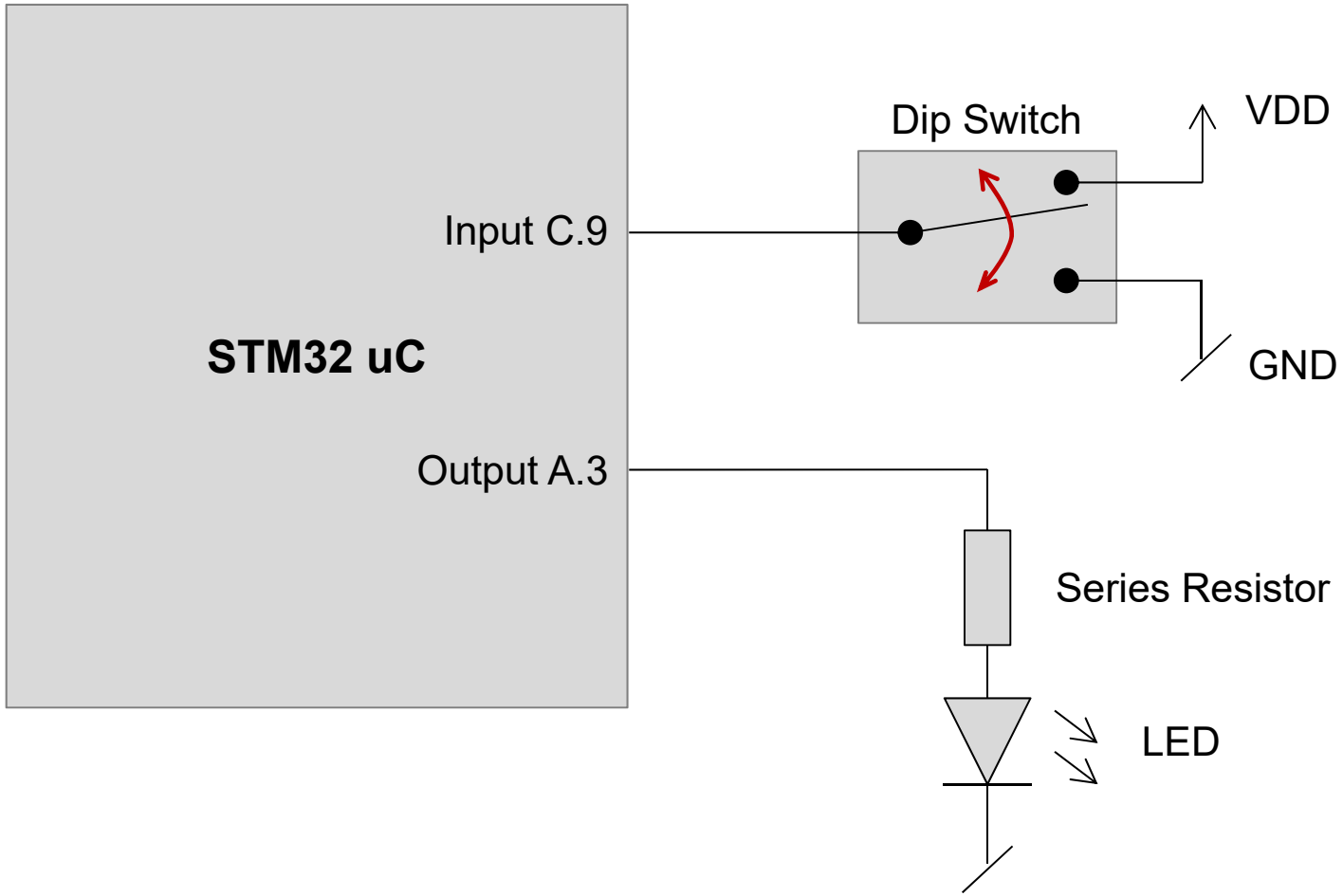
Computer Engineering 2

Overview

- **General Purpose Input / Output GPIO**
  - Reference Manual Pages 278 - 285



Why GPIOs?


Dip Switch
VDD
Input C.9
STM32 uC
GND
Output A.3
Series Resistor
LED

Learning Objectives

At the end of this lesson you will be able

- to work with register descriptions in reference manuals
- to explain the concept and implementation of GPIOs
- to explain the differences between open-drain and push-pull
- to use GPIOs in your own programs
- to explain the idea of a HAL

Agenda

School of Engineering
InES Institute of Embedded Systems

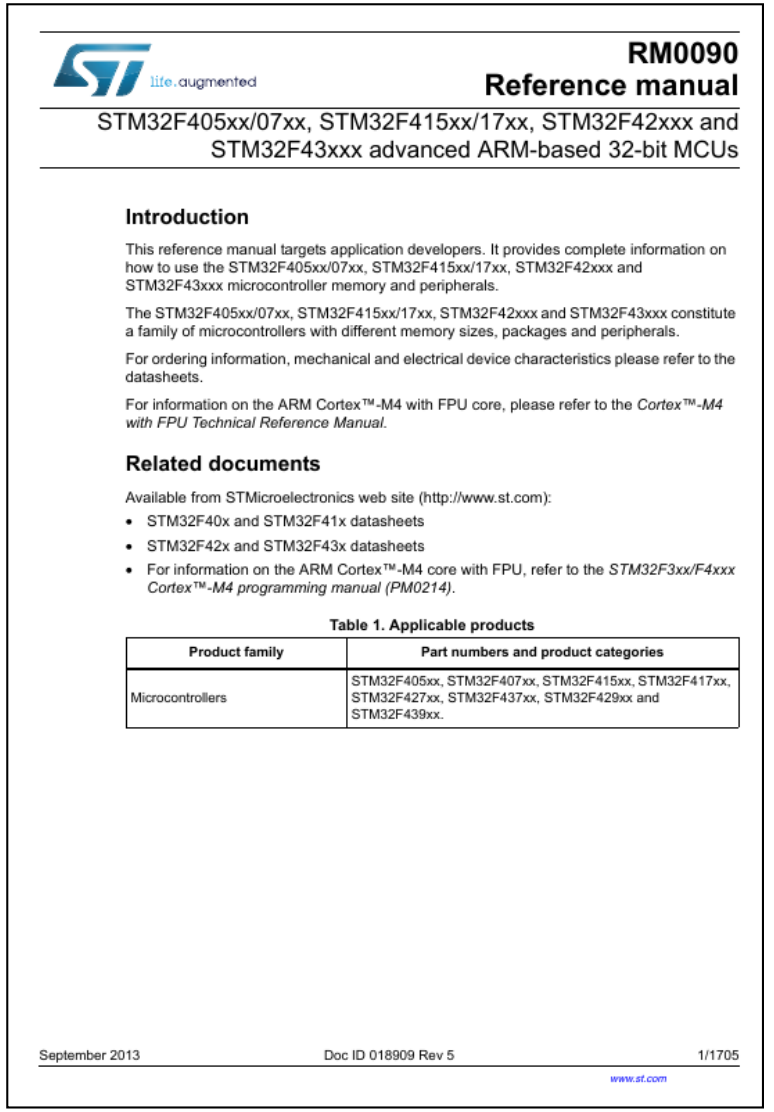
- **Working with documents**
- **General Purpose Input / Output (GPIO)**
- **GPIO Structure**
- **Configuring**
  - Direction
  - Output Type
  - Pull-up / Pull-down
  - Speed
- **Data Registers**
  - Reading Input Data
  - Writing Output Data, Setting and Clearing Bits
- **GPIO Cookbook**
- **Hardware Abstraction Layer (HAL)**

Working with Documents

School of Engineering
InES Institute of Embedded Systems

- **F4 Datasheet**
  - Pin out
  - Block diagram
  - Etc.

- **F4 Reference Manual**
  - Chapter “General-purpose I/Os (GPIO)”



life.augmented

RM0090
Reference manual

STM32F405xx/07xx, STM32F415xx/17xx, STM32F42xxx and STM32F43xxx advanced ARM-based 32-bit MCUs

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32F405xx/07xx, STM32F415xx/17xx, STM32F42xxx and STM32F43xxx microcontroller memory and peripherals.

The STM32F405xx/07xx, STM32F415xx/17xx, STM32F42xxx and STM32F43xxx constitute a family of microcontrollers with different memory sizes, packages and peripherals.

For ordering information, mechanical and electrical device characteristics please refer to the datasheets.

For information on the ARM Cortex™-M4 with FPU core, please refer to the *Cortex™-M4 with FPU Technical Reference Manual*.

Related documents

Available from STMicroelectronics web site (http://www.st.com):

- STM32F40x and STM32F41x datasheets
- STM32F42x and STM32F43x datasheets
- For information on the ARM Cortex™-M4 core with FPU, refer to the *STM32F3xx/F4xxx Cortex™-M4 programming manual (PM0214)*.

Table 1. Applicable products

Product family	Part numbers and product categories
Microcontrollers	STM32F405xx, STM32F407xx, STM32F415xx, STM32F417xx, STM32F427xx, STM32F437xx, STM32F429xx and STM32F439xx.

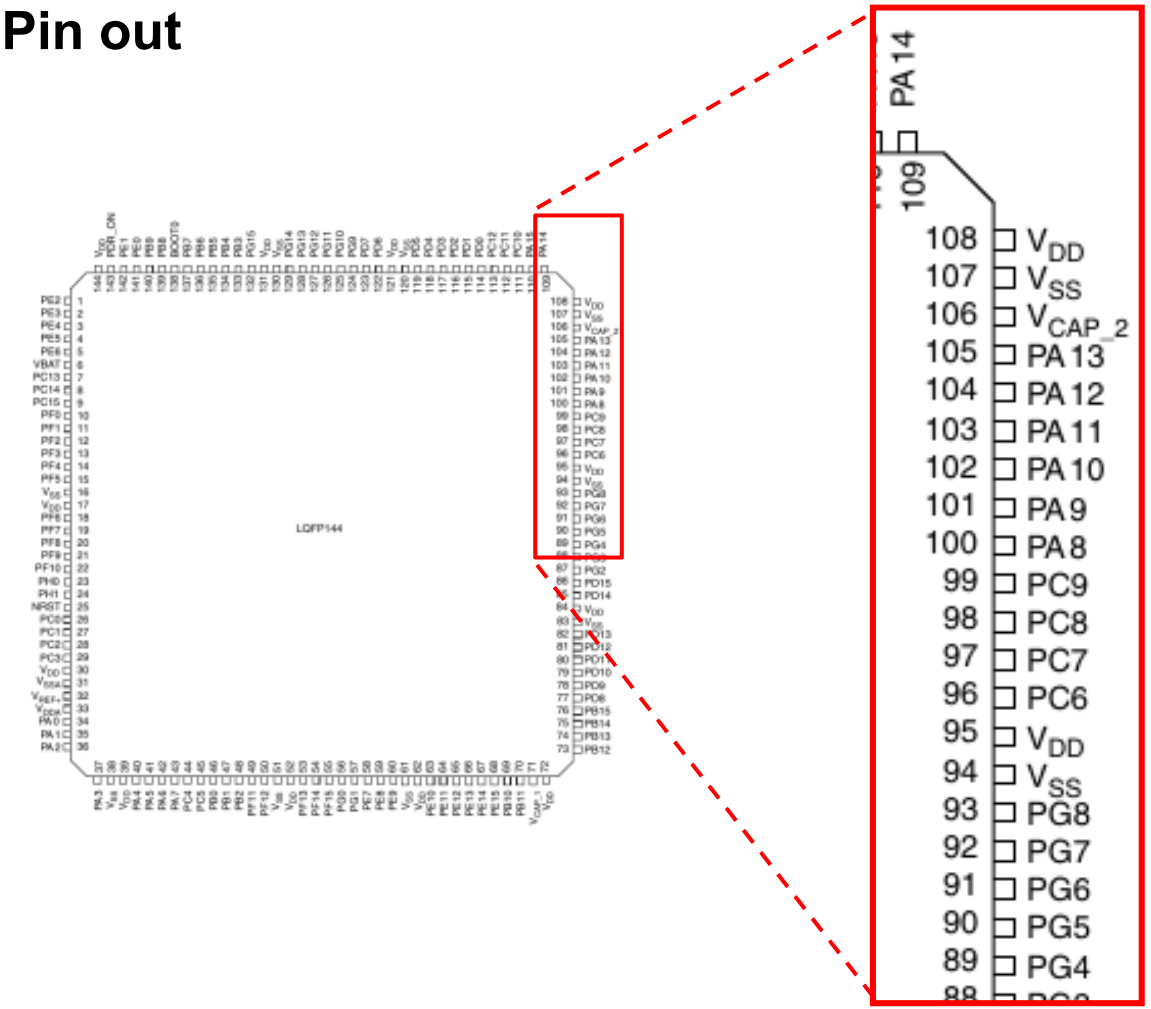
September 2013 Doc ID 018909 Rev 5 1/1705

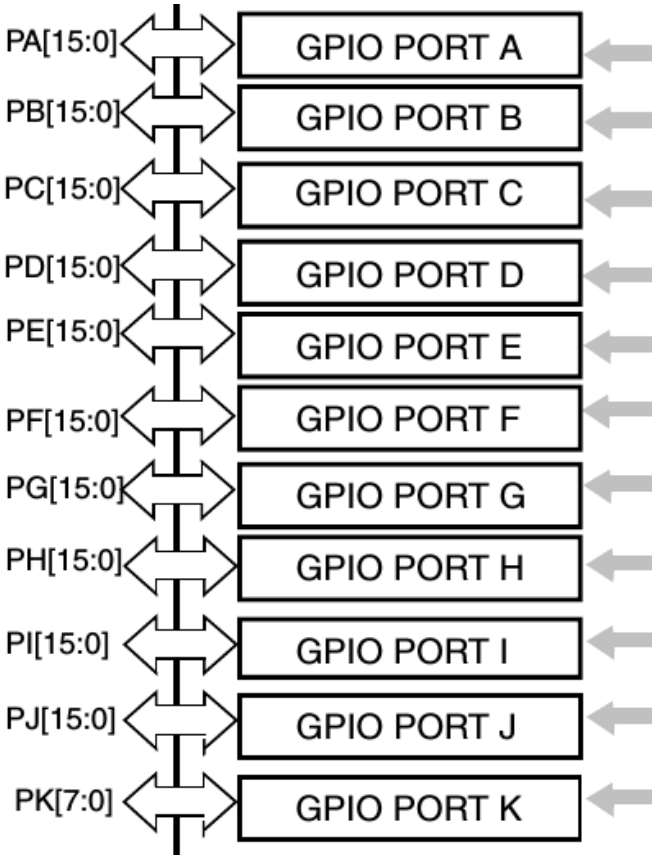
www.st.com

Working with Documents

School of Engineering
InES Institute of Embedded Systems

- **Pin out**





Sources: STM F4 Reference Manual
STM F4 Data Sheet

Working with Documents

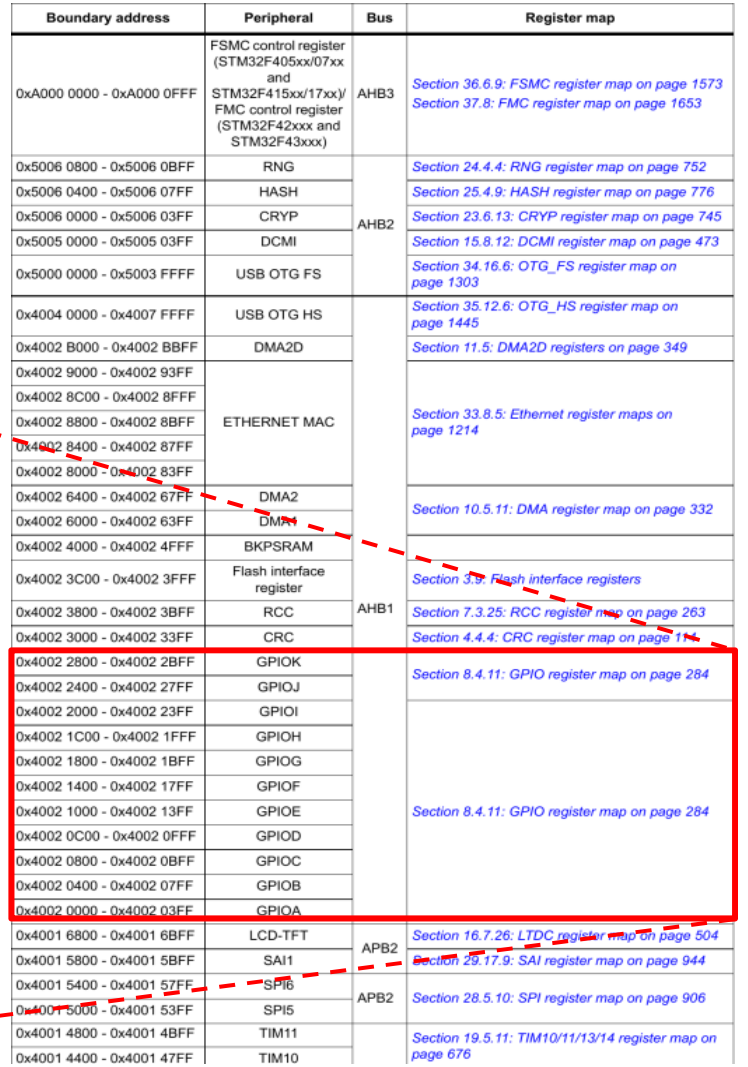
Note: The reference manual of ST shows memory maps with the low address at the bottom.

School of Engineering
InES Institute of Embedded Systems

- **Register address = Base address + Offset**
  - Offset is given for each register in reference manual
  - Base address defined in memory map
    → Reference Manual

base address

Boundary address	Peripheral
0x4002 2800 - 0x4002 2BFF	GPIOK
0x4002 2400 - 0x4002 27FF	GPIOJ
0x4002 2000 - 0x4002 23FF	GPIOI
0x4002 1C00 - 0x4002 1FFF	GPIOH
0x4002 1800 - 0x4002 1BFF	GPIOG
0x4002 1400 - 0x4002 17FF	GPIOF
0x4002 1000 - 0x4002 13FF	GPIOE
0x4002 0C00 - 0x4002 0FFF	GPIOD
0x4002 0800 - 0x4002 0BFF	GPIOC
0x4002 0400 - 0x4002 07FF	GPIOB
0x4002 0000 - 0x4002 03FF	GPIOA



Boundary address	Peripheral	Bus	Register map
0xA000 0000 - 0xA000 0FFF	FSMC control register (STM32F405xx/07xx and STM32F415xx/17xx)/ FMC control register (STM32F42xxx and STM32F43xxx)	AHB3	Section 36.6.9: FSMC register map on page 1573 Section 37.8: FMC register map on page 1653
0x5006 0800 - 0x5006 0BFF	RNG	AHB2	Section 24.4.4: RNG register map on page 752
0x5006 0400 - 0x5006 07FF	HASH		Section 25.4.9: HASH register map on page 776
0x5006 0000 - 0x5006 03FF	CRYP		Section 23.6.13: CRYP register map on page 745
0x5005 0000 - 0x5005 03FF	DCMI		Section 15.8.12: DCMI register map on page 473
0x5000 0000 - 0x5003 FFFF	USB OTG FS		Section 34.16.6: OTG_FS register map on page 1303
0x4004 0000 - 0x4007 FFFF	USB OTG HS	AHB1	Section 35.12.6: OTG_HS register map on page 1445
0x4002 B000 - 0x4002 BBFF	DMA2D		Section 11.5: DMA2D registers on page 349
0x4002 9000 - 0x4002 93FF 0x4002 8C00 - 0x4002 8FFF 0x4002 8800 - 0x4002 8BFF 0x4002 8400 - 0x4002 87FF 0x4002 8000 - 0x4002 83FF	ETHERNET MAC		Section 33.8.5: Ethernet register maps on page 1214
0x4002 6400 - 0x4002 67FF	DMA2		Section 10.5.11: DMA register map on page 332
0x4002 6000 - 0x4002 63FF	DMA1		
0x4002 4000 - 0x4002 4FFF	BKPSRAM		
0x4002 3C00 - 0x4002 3FFF	Flash interface register		Section 3.9: Flash interface registers
0x4002 3800 - 0x4002 3BFF	RCC		Section 7.3.25: RCC register map on page 263
0x4002 3000 - 0x4002 33FF	CRC		Section 4.4.4: CRC register map on page 114
0x4002 2800 - 0x4002 2BFF	GPIOK		Section 8.4.11: GPIO register map on page 284
0x4002 2400 - 0x4002 27FF	GPIOJ		
0x4002 2000 - 0x4002 23FF	GPIOI		
0x4002 1C00 - 0x4002 1FFF	GPIOH		
0x4002 1800 - 0x4002 1BFF	GPIOG		
0x4002 1400 - 0x4002 17FF	GPIOF		
0x4002 1000 - 0x4002 13FF	GPIOE		Section 8.4.11: GPIO register map on page 284
0x4002 0C00 - 0x4002 0FFF	GPIOD		
0x4002 0800 - 0x4002 0BFF	GPIOC		
0x4002 0400 - 0x4002 07FF	GPIOB		
0x4002 0000 - 0x4002 03FF	GPIOA		
0x4001 6800 - 0x4001 6BFF	LCD-TFT	APB2	Section 16.7.26: LTDC register map on page 504
0x4001 5800 - 0x4001 5BFF	SAI1		Section 29.17.9: SAI register map on page 944
0x4001 5400 - 0x4001 57FF	SPI6	APB2	Section 28.5.10: SPI register map on page 906
0x4001 5000 - 0x4001 53FF	SPI5		
0x4001 4800 - 0x4001 4BFF	TIM11		Section 19.5.11: TIM10/11/13/14 register map on page 676
0x4001 4400 - 0x4001 47FF	TIM10		

Working with Documents

School of Engineering
InES Institute of Embedded Systems

- **Exercise**

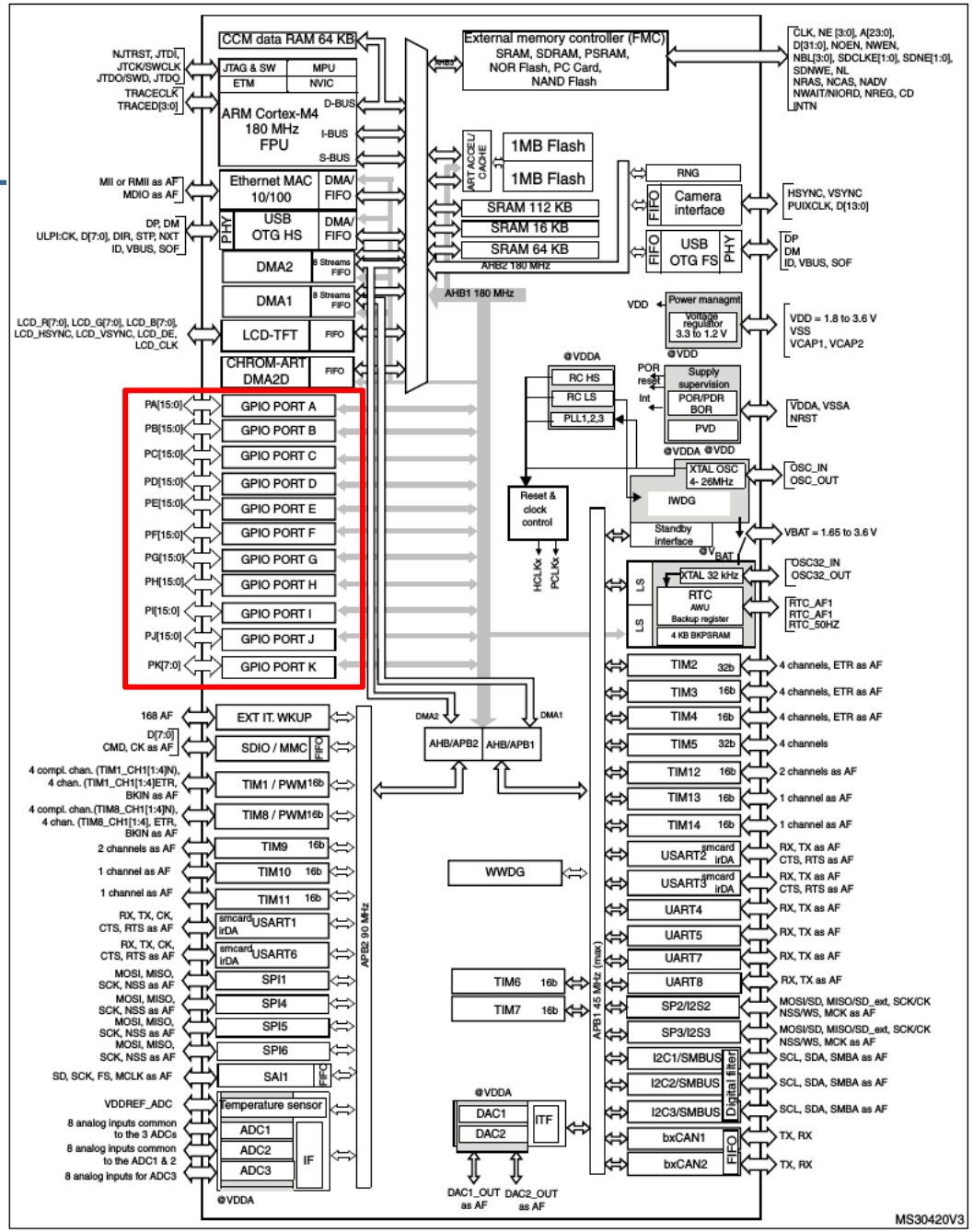
  Use the data sheet and reference manual to answer the following questions for an STM32F429 LQFP144

  - What are the pin numbers (0-144) for GPIO ports A.3 and E.1?

  - What is the address of register GPIOA_OTYPER?

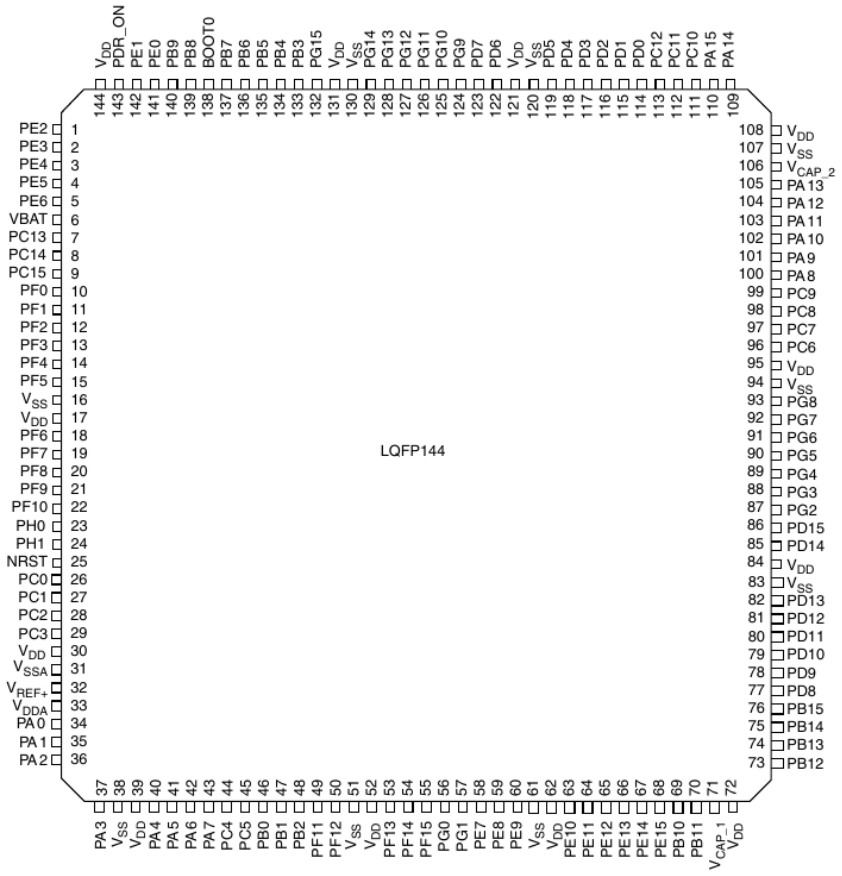
GPIO

- **General Purpose Input / Output**



General Purpose Input / Output

- **Situation**
  - Microcontroller as general purpose device
  - Many functional blocks included
- **Problem**
  - Limited number of pins
  - For a specific configuration, not all functions can be routed to I/O pins
- **Solution**
  - Many (all) pins configurable
  - Select the needed I/O pins / functions
  - „pin sharing“
  - Output multiplexer needs to be configured



Pinout STM32F429 LQFP144
Source: STM F4 Reference Manual

General Purpose Input / Output

School of Engineering
InES Institute of Embedded Systems

- **Multiple functions „share“ a single pin (pin sharing)**
  - Digital inputs / outputs (GPIO)
  - Serial interfaces
  - Timers / Counters
  - ADC (A/D conversion)

- **Consequences**
  - Not all functions externally available at the same time
  - Programming of internal registers defines pin use
  - Pin / function configuration is usually static, i.e. set once at startup

→ Not all combinations possible simultaneously

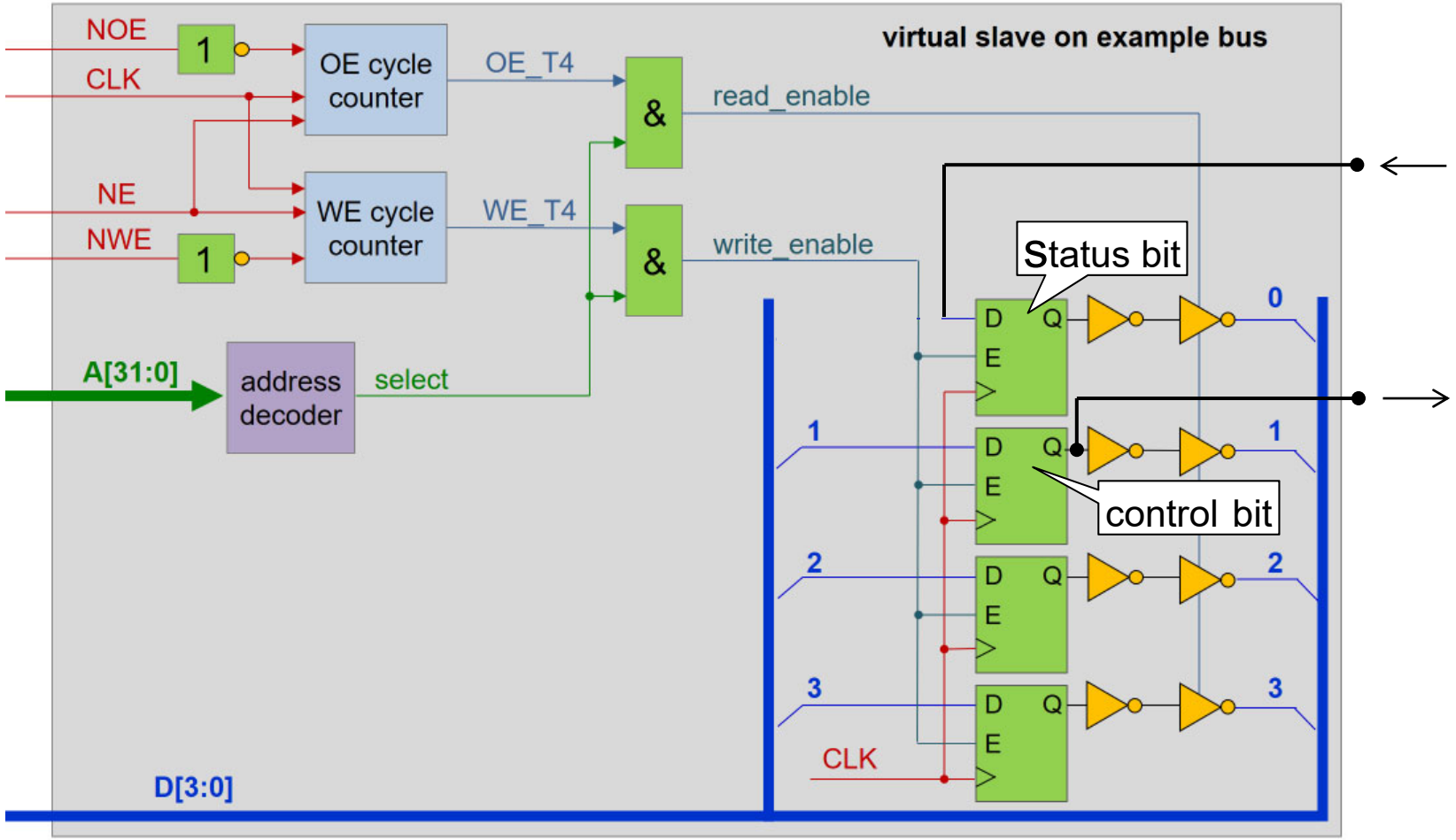
STM32F4xx GPIO

School of Engineering
InES Institute of Embedded Systems

- **Features**
  - GPIO pins configurable by software
    - output (push-pull or open-drain; with or without pull-up or pull-down)
    - input (floating, with or without pull-up or pull-down)
    - peripheral alternate function
  - High-current-capable
  - Speed selection
  - Maximum I/O toggling up to 90 MHz
  - Most of the GPIO pins are shared with alternate digital or analog functions

GPIO Structure

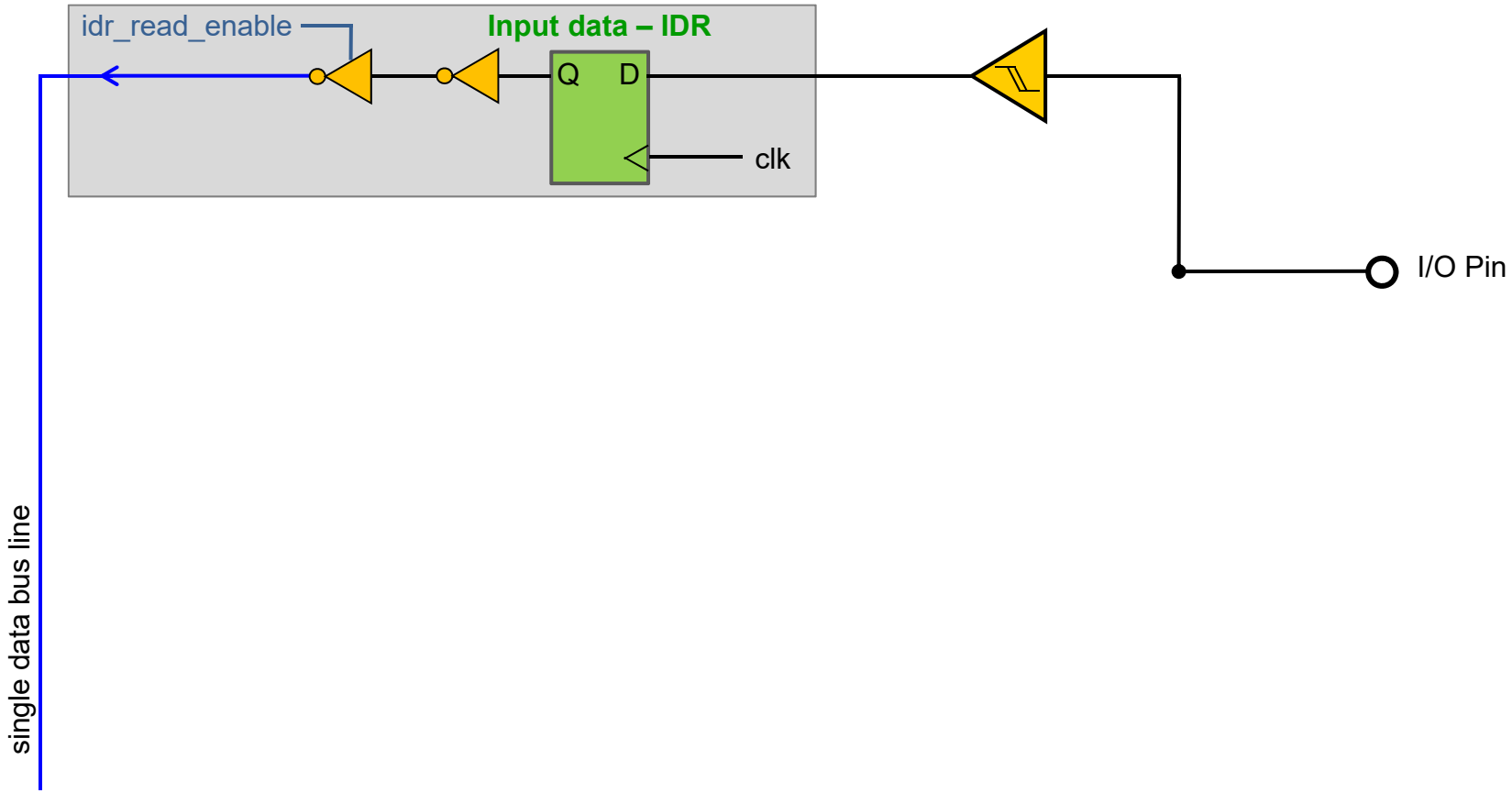
- **Repetition: Hardware slave on synchronous bus**



GPIO Structure

- **Input**

Address decoding: Register accesses activate the corresponding enable signals

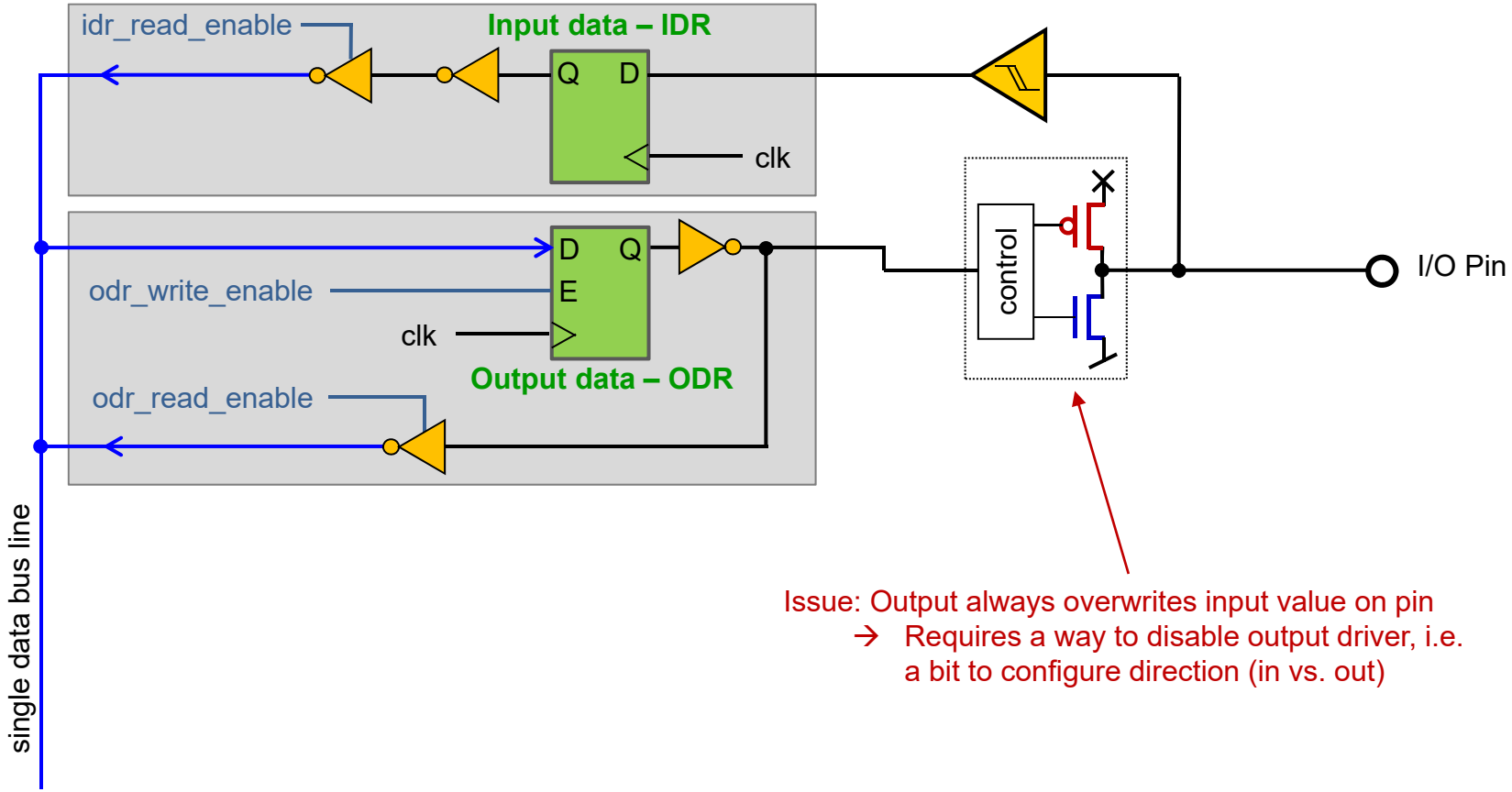


GPIO Structure

School of Engineering
InES Institute of Embedded Systems

Adding output

Address decoding: Register accesses activate the corresponding enable signals

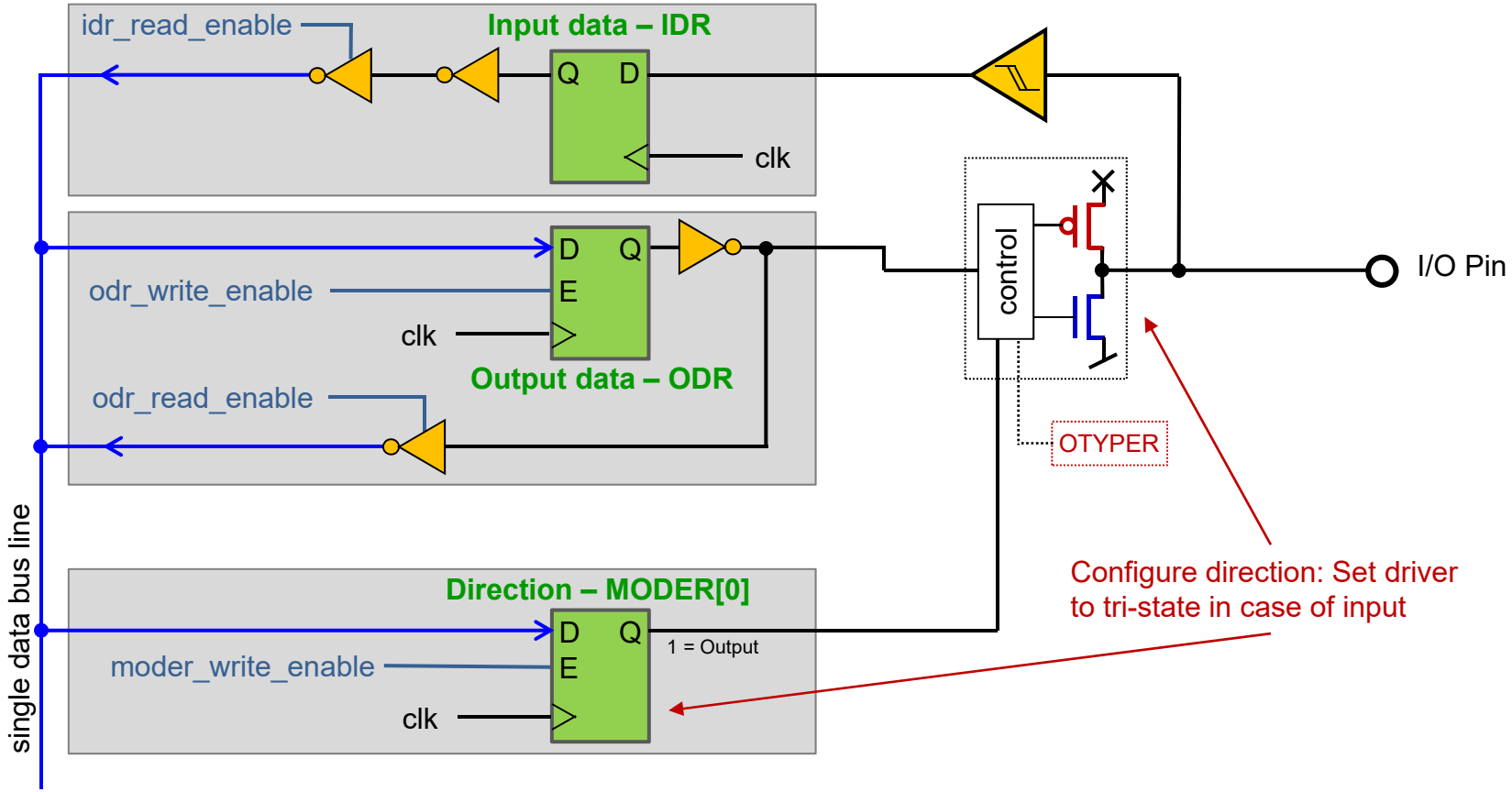


Issue: Output always overwrites input value on pin
→ Requires a way to disable output driver, i.e. a bit to configure direction (in vs. out)

GPIO Structure

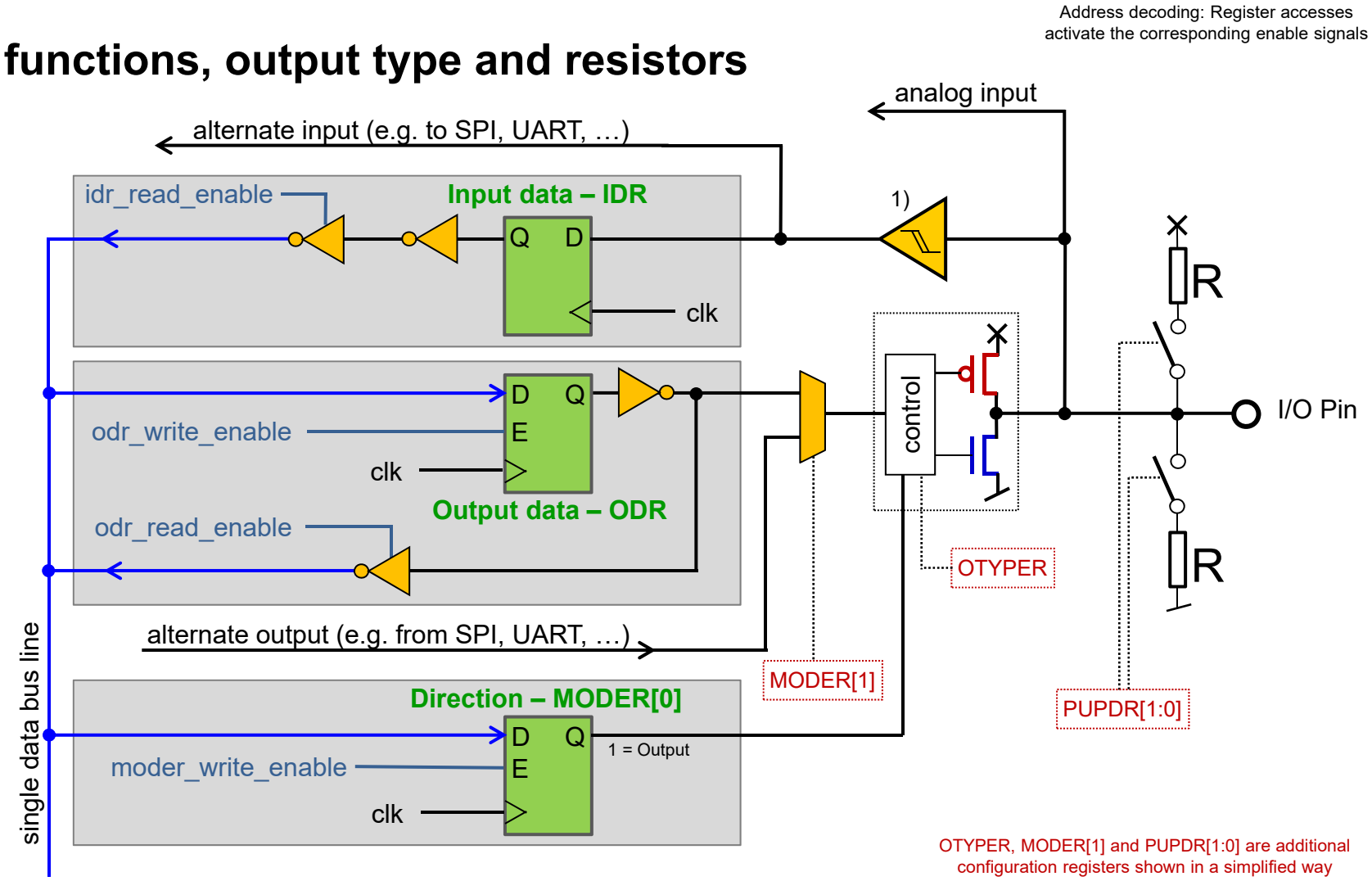
- **Choosing between input and output**

Address decoding: Register accesses activate the corresponding enable signals



GPIO Structure

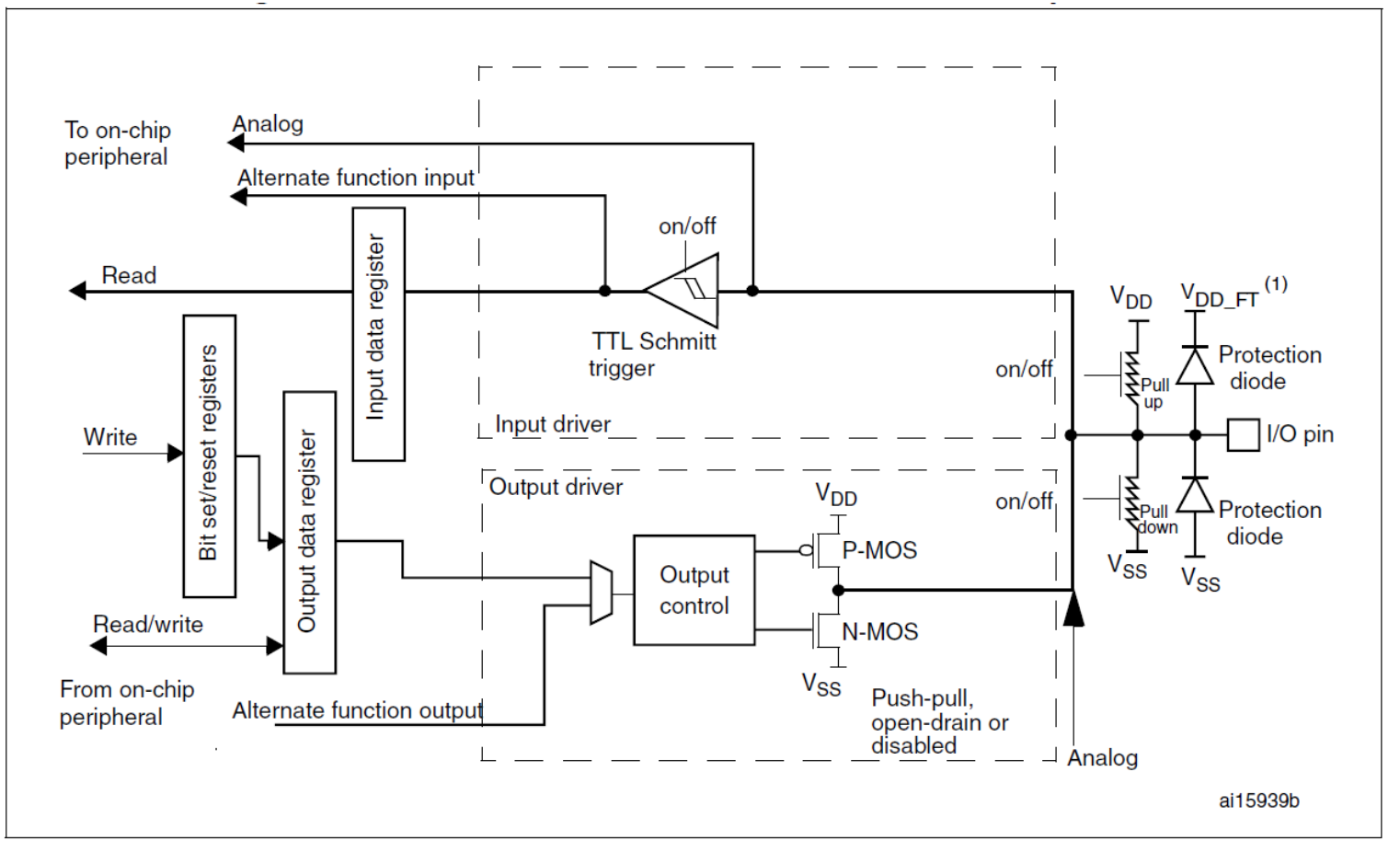
- **Alternate functions, output type and resistors**



1) Schmitt-Trigger off in case of analog input

GPIO Structure

- **GPIO as shown in ST Reference Manual**



Source: ST F4 Reference Manual

Configuring Direction

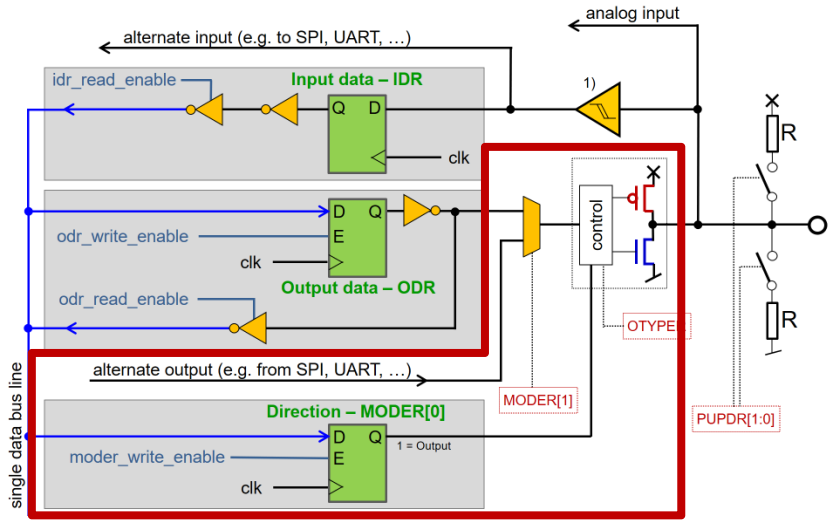
School of Engineering
InES Institute of Embedded Systems

- **Mode register (GPIOx_MODER)**

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

- MODER[1:0]
  - 00: Input
  - 01: General purpose output mode
  - 10: Alternate function mode
  - 11: Analog mode

In case of alternate function:
The individual alternate function (SPI, UART, etc.) can be selected
by programming GPIOx_AFRL/H (i.e. Alternate function register low / high)



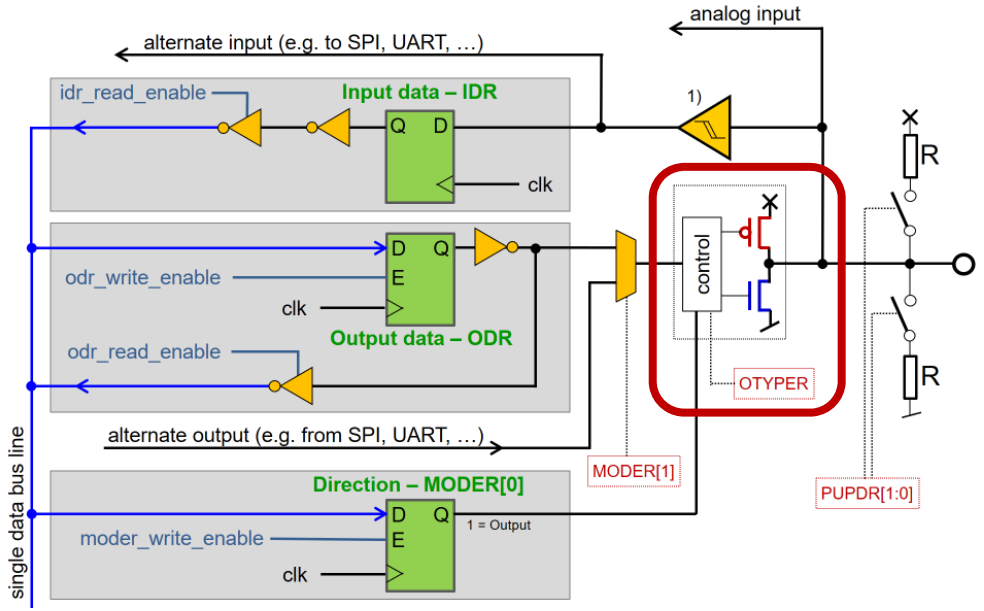
See reference manual

Configuring Output Type

- **Output type register (GPIOx_OTYPER)**

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
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

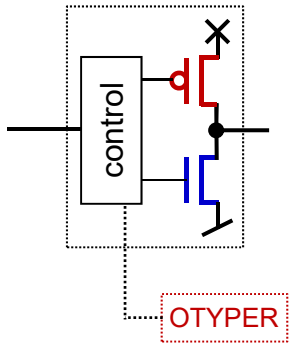
- OT
  - 0: Output push-pull
  - 1: Output open-drain



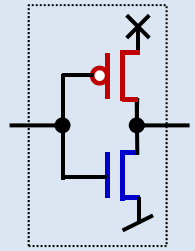
See reference manual

Output Type

- **Push-pull vs. Open-drain**
  - Case output i.e. MODER[0] = ‘1’

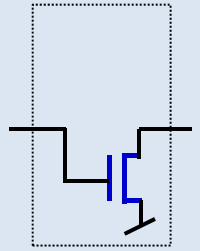


OT = ‘0’ → push-pull



Output stage can drive output ‘high’ or ‘low’

OT = ‘1’ → open-drain



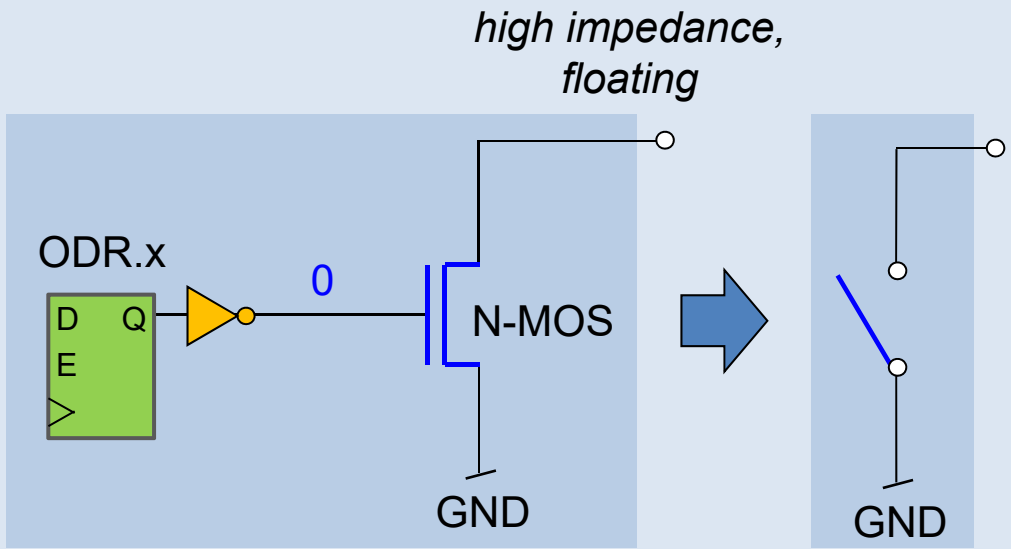
Output stage can only drive output ‘low’

Output Type

- **Open-drain**
  - Pull-down transistor only → no pull-up transistor

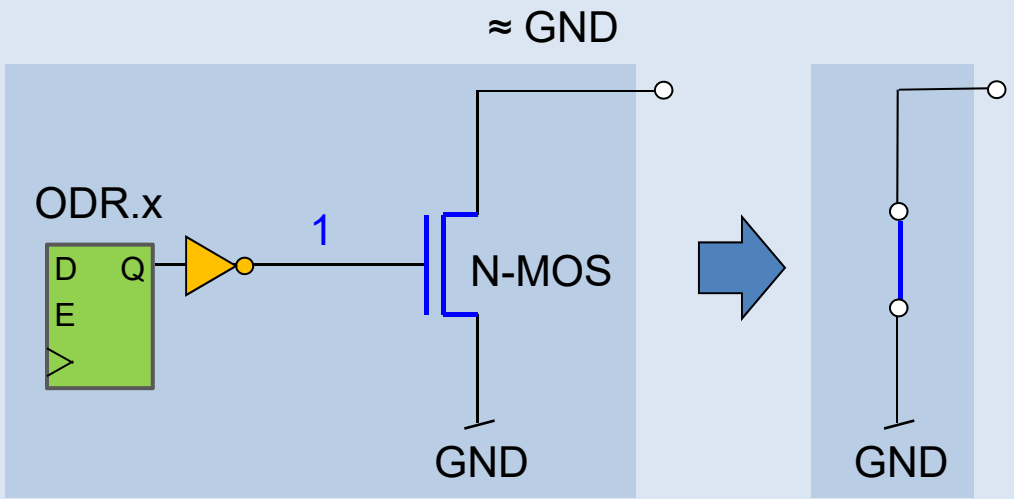
ODR.x = '1'

- R_{DS} is high
- Transistor is blocking → switch is open
- Output high impedance, floating



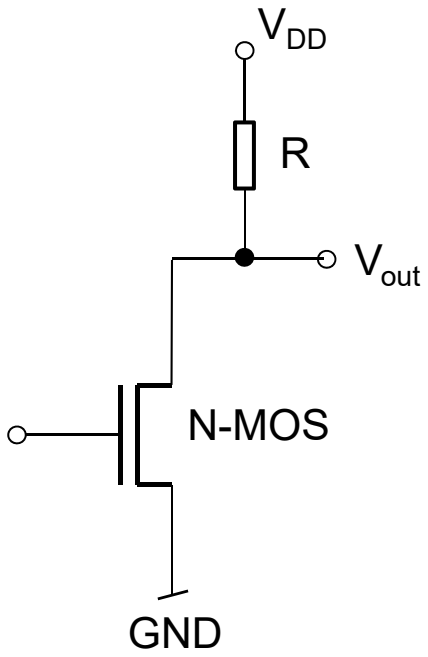
ODR.x = '0'

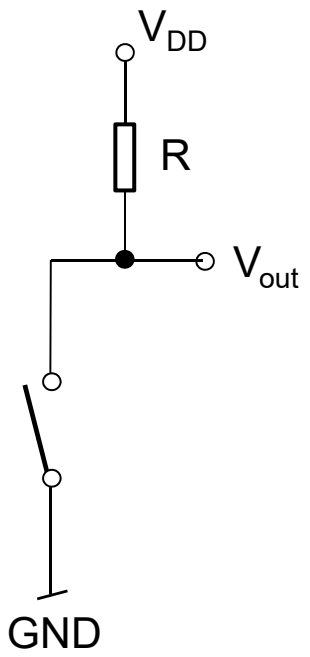
- R_{DS} is low
- Transistor is conducting → switch closed
- Output pulled to GND



Output Type

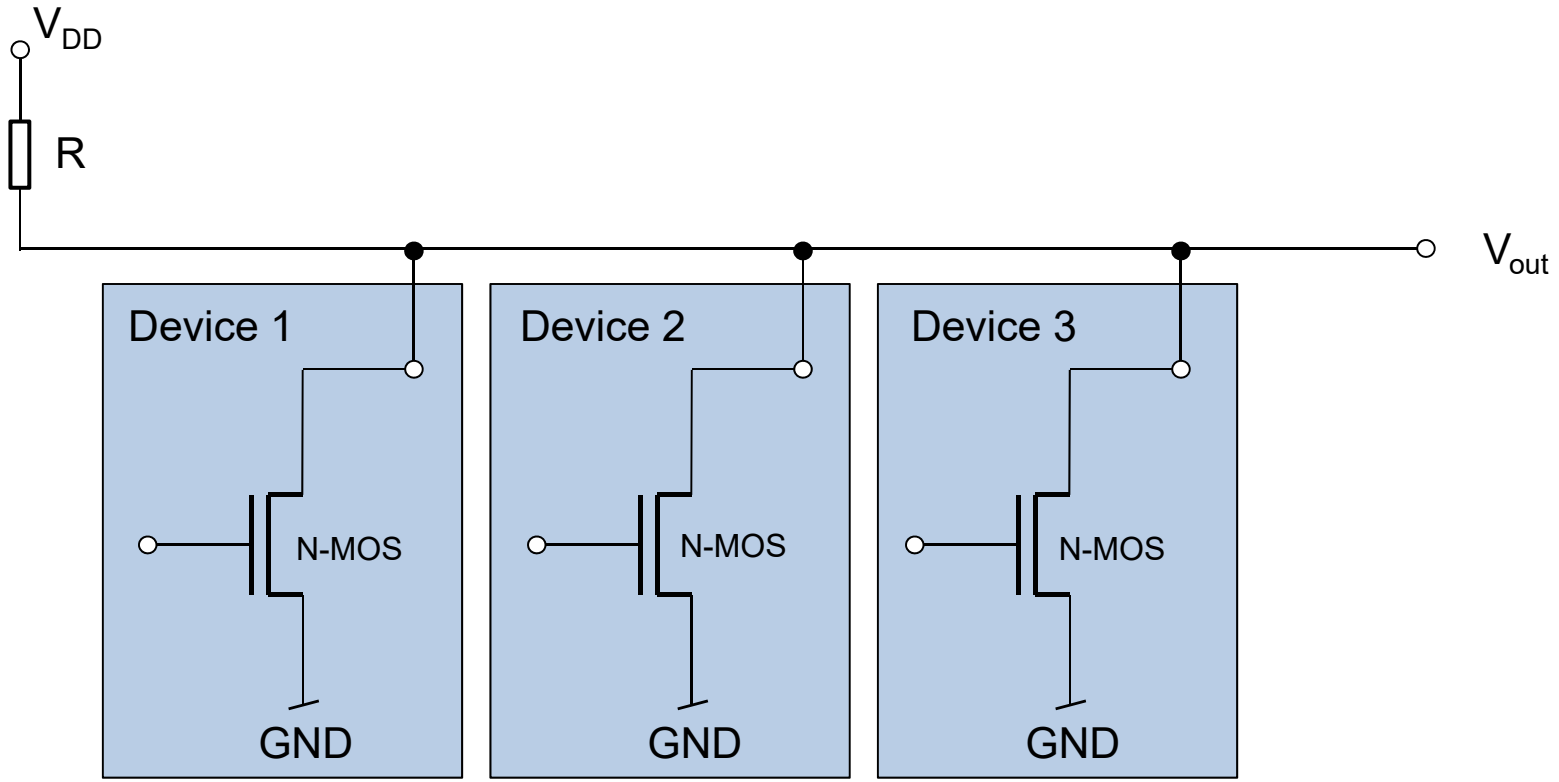
- **Open-drain with pull-up resistor**
  - Transistor blocking (= switch open) → V_{out} is pulled up to level of VDD
  - Transistor conducting (= switch closed) → V_{out} goes to GND





Output Type

- **Multiple open-drain outputs on a bus line**
  - No electrical signal conflicts possible
  - Any device can pull signal low at any time



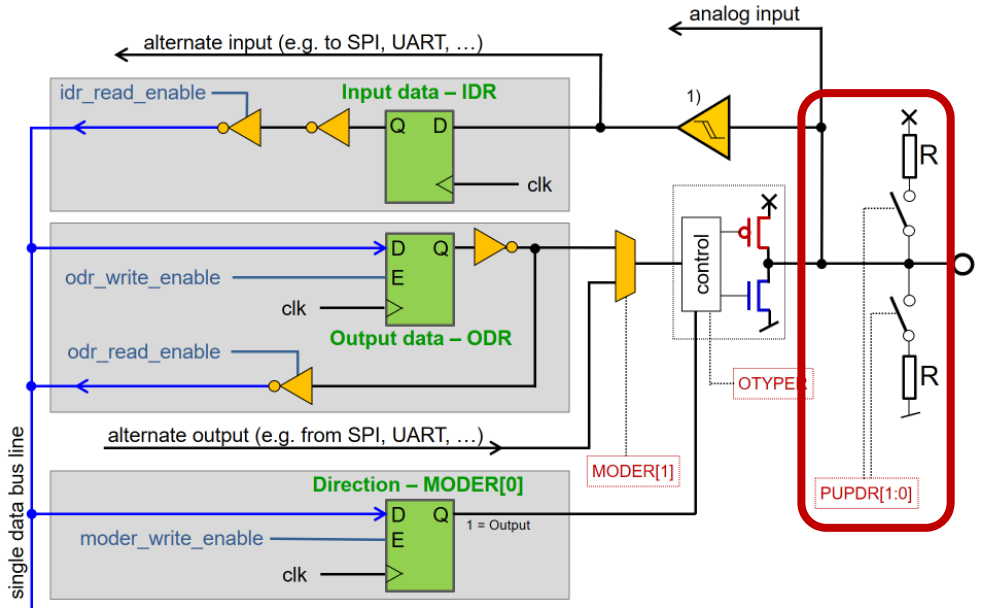
Configuring Pull-up / Pull-down

School of Engineering
InES Institute of Embedded Systems

- **Pull-up/pull-down register (GPIOx_PUPDR)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[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
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- PUPDR[1:0]
  - 00: No pull-up, no pull-down
  - 01: Pull-up
  - 10: Pull-down
  - 11: Reserved



See reference manual

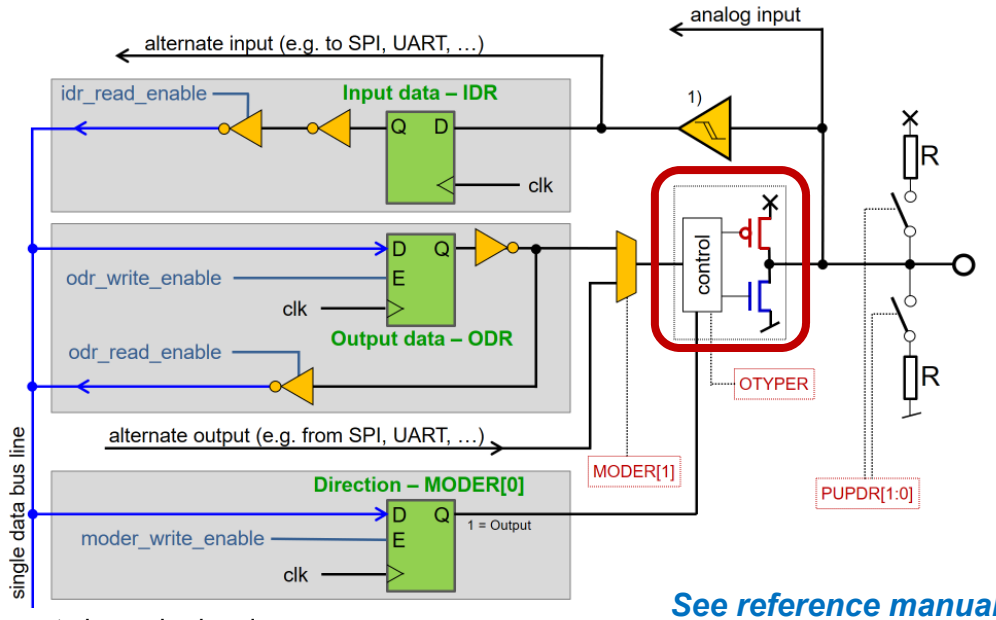
Configuring Speed

Output speed register (GPIOx_OSPEEDR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15 [1:0]		OSPEEDR14 [1:0]		OSPEEDR13 [1:0]		OSPEEDR12 [1:0]		OSPEEDR11 [1:0]		OSPEEDR10 [1:0]		OSPEEDR9 [1:0]		OSPEEDR8 [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
OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1 [1:0]		OSPEEDR0 1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- OSPEEDR[1:0]
  - 00: Low speed
  - 01: Medium speed
  - 10: Fast speed
  - 11: High speed

Motivation: Match output stage to impedance of transmission line.
Control steepness of edge → e.g for EMC reasons



OSPEEDR register not shown in drawing

See reference manual

I/O Port Configuration

School of Engineering
InES Institute of Embedded Systems

- **Overview**

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

	MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]		PUPDR(i) [1:0]		I/O configuration	
GP output	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	
GP input	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		

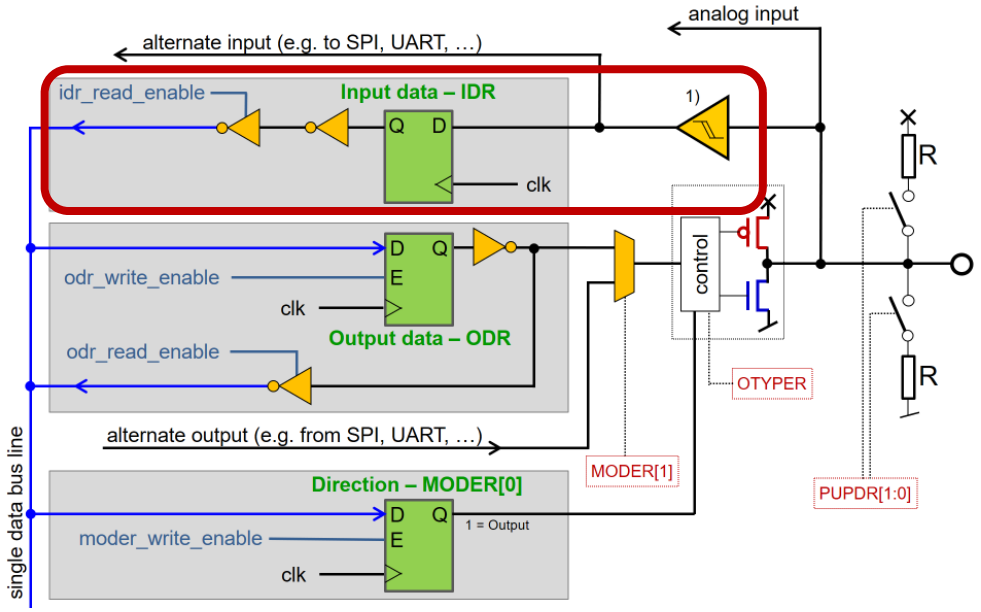
Source: ST F4 Reference Manual

Reading Input Data

- **GPIO port input data register (GPIOx_IDR)**

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

- IDR: Port input data
  - contain input value of corresponding I/O port
  - read-only



See reference manual

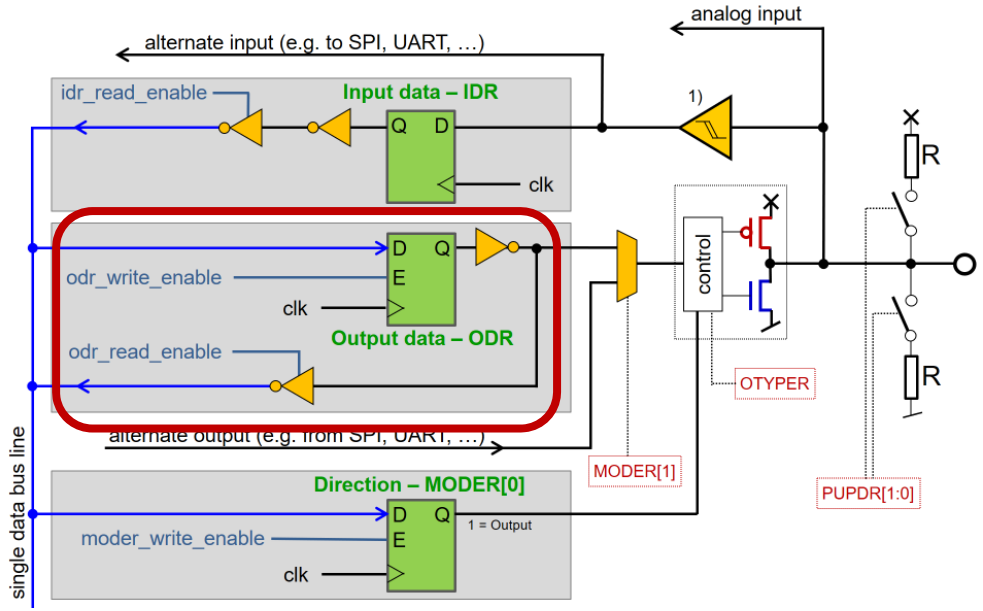
Writing Output Data

School of Engineering
InES Institute of Embedded Systems

- **GPIO port output data register (GPIOx_ODR)**

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
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

  - ODR: Port output data
    - read and write by software



See reference manual

Setting and Clearing Bits

School of Engineering
InES Institute of Embedded Systems

- **GPIO port bit set/reset register (GPIOx_BSRR)**

Clear bits →

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

Set bits →

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

- Clear port bit by writing a ‘1’ to BSRR[bit+16]
- Set port bit by writing a ‘1’ to BSRR[bit]
- Ensures atomic access in software (no interruption possible)
  - Setting a bit through ODR requires ‘read ODR’, ‘OR operation with bit mask’ and ‘write ODR’

See reference manual

GPIO Register Map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPIOA_MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[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]	
	Reset value	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	GPIOB_MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[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]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x00	GPIOx_MODER (where x = C..I/J/K)	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[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]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	GPIOx_OTYPER (where x = A..I/J/K)	Reserved																OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPIOx_OSPEEDER (where x = A..I/J/K except B)	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPIOB_OSPEEDER	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
0x0C	GPIOA_PUPDR	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOB_PUPDR	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x0C	GPIOx_PUPDR (where x = C..I/J/K)	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	GPIOx_IDR (where x = A..I/J/K)	Reserved																IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
	Reset value																	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
0x14	GPIOx_ODR (where x = A..I/J/K)	Reserved																ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	GPIOx_BSRR (where x = A..I/J/K)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	GPIOx_LCKR (where x = A..I/J/K)	Reserved															LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	GPIOx_AFRL (where x = A..I/J/K)	AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]				AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	GPIOx_AFRH (where x = A..I/J/K)	AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]				AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

See reference manual

GPIO Cookbook

School of Engineering
InES Institute of Embedded Systems

- **Basic input / output configuration**
  - Find pin numbers related to GPIOx or vice versa
  - Configure through configuration registers
    - GPIOx_MODER
    - GPIOx_OTYPER
    - GPIOx_OSPEEDR
    - GPIOx_PUPDR
- **Data operations**
  - Input → Read register GPIOx_IDR
  - Output → Write register GPIOx_ODR or GPIOx_BSRR

Exercise: GPIO Configuration

School of Engineering
InES Institute of Embedded Systems

- **On a STM32F429 LQFP144 Pin 37 should be configured as low speed output with open-drain and pull-up**

  - What registers (names and addresses) must be configured?
  - Indicate the bits to be configured and their values!
  - No code required.

Hardware Abstraction Layer (HAL)

School of Engineering
InES Institute of Embedded Systems

- **Accessing a register**

```
#define GPIOA_MODER     (*((volatile uint32_t *)(0x40020000)))

GPIOA_MODER = 0x55555555;     // all output
```

- **However**
  - Each GPIO port has the same 10 registers
  - There are 11 GPIO ports → GPIOA – GPIOK

  } → Results in 110 macros with repetitive code

- **We want an abstraction similar to base address and offset**

Hardware Abstraction Layer (HAL)

School of Engineering
InES Institute of Embedded Systems

reg_stm32f4xx.h

Base addresses

Pointers to struct of type reg_gpio_t

```
#define GPIOA                           ( (reg_gpio_t *) 0x40020000 )
#define GPIOB                           ( (reg_gpio_t *) 0x40020400 )
#define GPIOC                           ( (reg_gpio_t *) 0x40020800 )
#define GPIOD                           ( (reg_gpio_t *) 0x40020c00 )
#define GPIOE                           ( (reg_gpio_t *) 0x40021000 )
#define GPIOF                           ( (reg_gpio_t *) 0x40021400 )
#define GPIOG                           ( (reg_gpio_t *) 0x40021800 )
#define GPIOH                           ( (reg_gpio_t *) 0x40021c00 )
#define GPIOI                           ( (reg_gpio_t *) 0x40022000 )
#define GPIOJ                           ( (reg_gpio_t *) 0x40022400 )
#define GPIOK                           ( (reg_gpio_t *) 0x40022800 )
```

base addresses

Offset

Typedef for reg_gpio_t

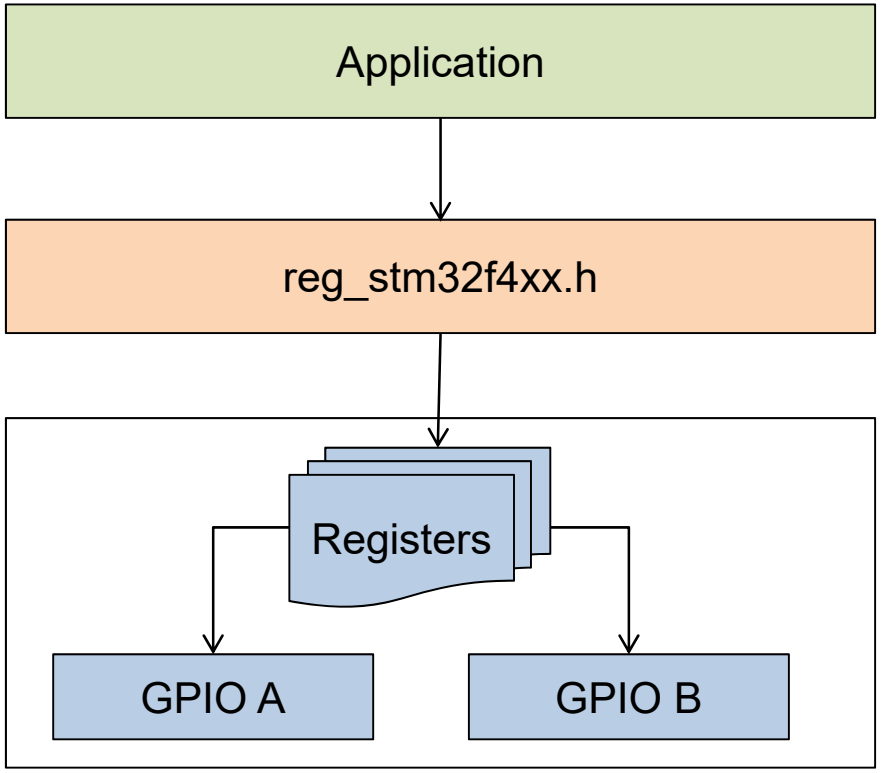
```
/**
 *  \struct reg_gpio_t
 *  \brief  Representation of GPIO register.
 *
 *  Described in reference manual p.265ff.
 */
typedef struct {
    volatile uint32_t MODER;    /**< Port mode register. */
    volatile uint32_t OTYPER;   /**< Output type register. */
    volatile uint32_t OSPEEDR;  /**< Output speed register. */
    volatile uint32_t PUPDR;    /**< Port pull-up/pull-down register. */
    volatile uint32_t IDR;      /**< Input data register. */
    volatile uint32_t ODR;      /**< output data register. */
    volatile uint32_t BSRR;     /**< Bit set/reset register */
    volatile uint32_t LCKR;     /**< Port lock register. */
    volatile uint32_t AFRL;     /**< AF low register pin 0..7. */
    volatile uint32_t AFRH;     /**< AF high register pin 8..15. */
} reg_gpio_t;
```

register names as in reference manual

size of registers

```
GPIOA->MODER = 0x55555555;    // all output
```

Hardware Abstraction Layer (HAL)



Lowest level of hardware abstraction layer, contains

- Base addresses
- Structs → members correspond to hardware registers
- Helper macros

Exercise: GPIO Configuration

School of Engineering
InES Institute of Embedded Systems

- **Write the code to configure the bits from the last exercise**
  - GPIOA_MODER[7:6] → MODER3 = 01 → GP output
  - GPIOA_OTYPER[3] → OT3 = 1 → open-drain
  - GPIOA_OSPEEDR[7:6] → OSPEEDR3 = 00 → low speed
  - GPIOA_PUPDR[7:6] → PUPDR3 = 01 → pull up

- Use the base addresses and structs in **`reg_stm32f4xx.h`**
- Do not change the other bits

```
#include "reg_stm32f4xx.h"

void config_gpioa_pin3(void)
{









}
```