

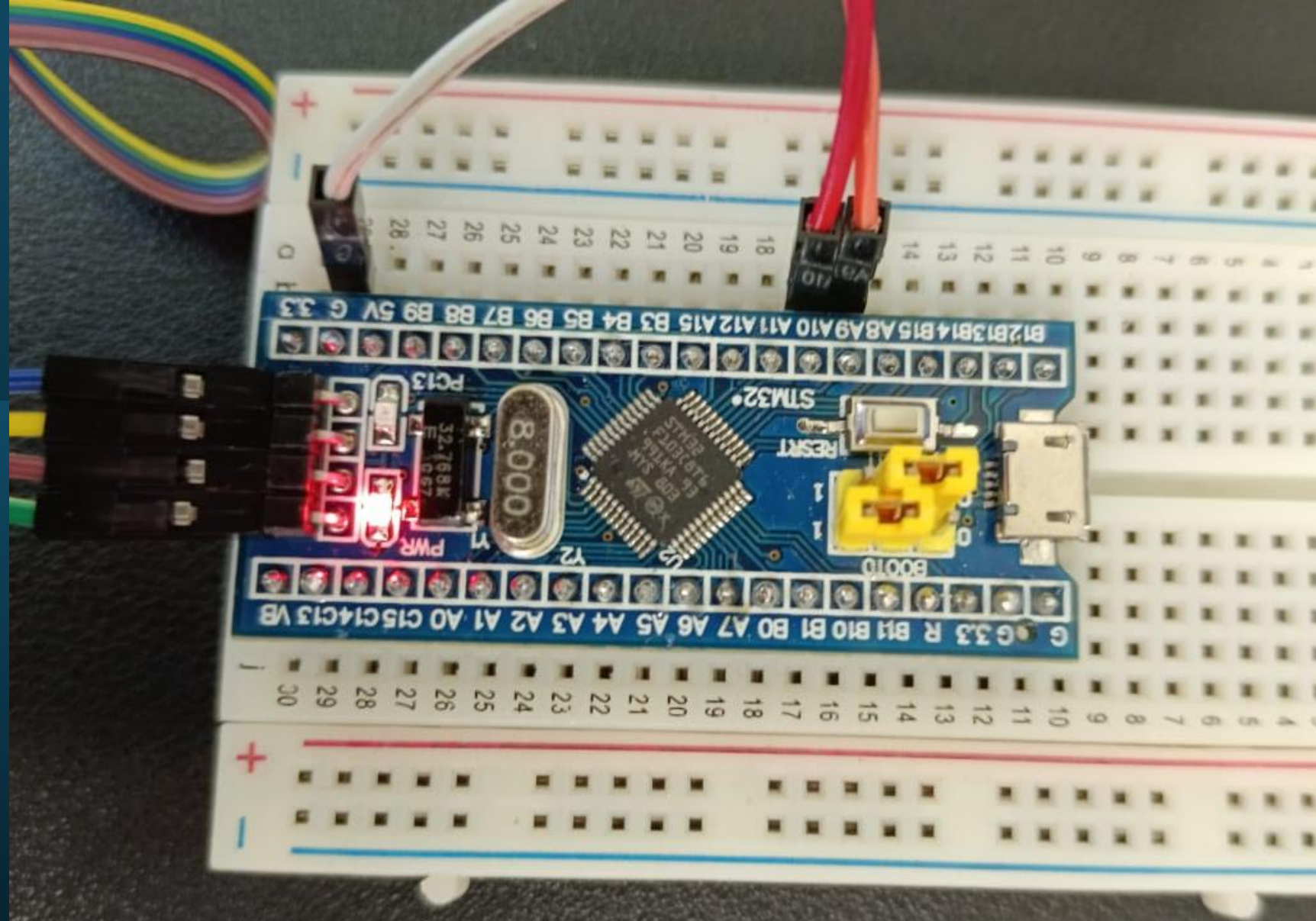
PM Ir Dr Tee Kian Sek

Faculty of Electrical and Electronic Engineering

Universiti Tun Hussein Onn Malaysia

Date: May 2024

Getting Start with STM32 using STM32CubeIDE and HAL



Instructors



PM Ir Dr Tee Kian Sek

<https://community.uthm.edu.my/staff/people/tee>



Dr Chew Chang Choon

<https://community.uthm.edu.my/chewcc>



Outlines

Cognition – Good to know!

- Objectives of this short course
- Arduino and STM32 – Learning scope
- Brief on ARM and STM32
- What is MCU?

This Short Course (Hands-on)

- The training kit
- The preparation
- First Project – very important
- Practices



GitHub – Shared Online

Shared Documents

https://github.com/rtlab1417/STM32_intro_shared



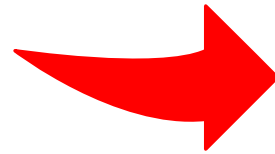
Objectives of
this short course



Arduino and STM32 – Learning scopes

Arduino UNO

- ATmega328P
- Which company? Who cares?
- Datasheet – anyone read?
- IDE and libraries? Many from third parties.
- Libraries – who read them?



STM32

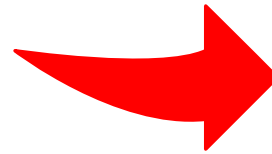
- STM32 MCU families
- ST
- Datasheet – we do care
- IDE and HAL.
- Copy customized .c and .h

All hardware engineers care about spec/datasheets on all MCUs and ICs.

Arduino and STM32 – Learning scopes

Arduino UNO

- Exploring registers?
- Is anyone trying to understand them?
- I2C/SPI electronic parts?
- Interrupts / Timer / PWM / UART / ADC ?
- Live debugging?



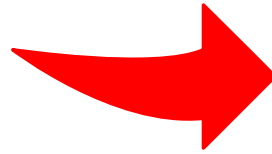
STM32

- We do care the registers
- We do care – Specification/datasheet of I2C/SPI electronic parts
- Registers for Interrupts / Timer / PWM / USART / DMA / USB / ADC / RTC
- Live debugging

Arduino and STM32 – Learning scopes

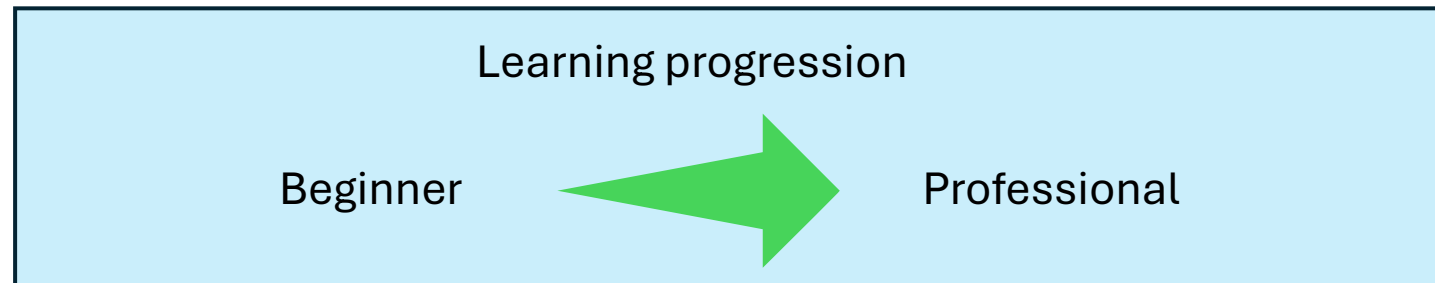
Arduino UNO

- Anyone care?
- Production?
- Flashing ? duplicates, erase, password protected?



STM32

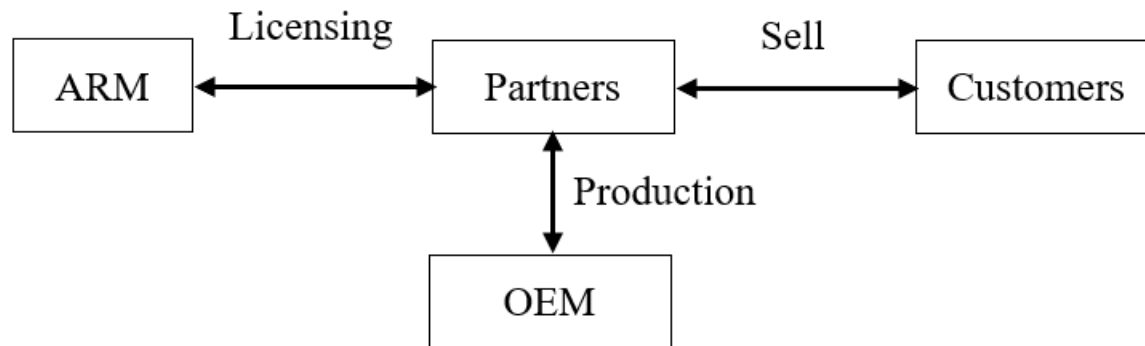
- Part of the exercises
- Tools for flashing



** ATmega328P could be flashed in production.*

Brief on ARM and STM32

ARM Licensing model (simplified)



Partners

1. Broadcom
2. Apple
3. ST
4. Microchip
5. Etc.

- ARM licenses IP to over 1,000 global partners (including Samsung, Apple, Microsoft).
- See modules

See the website for Licensing Model Options

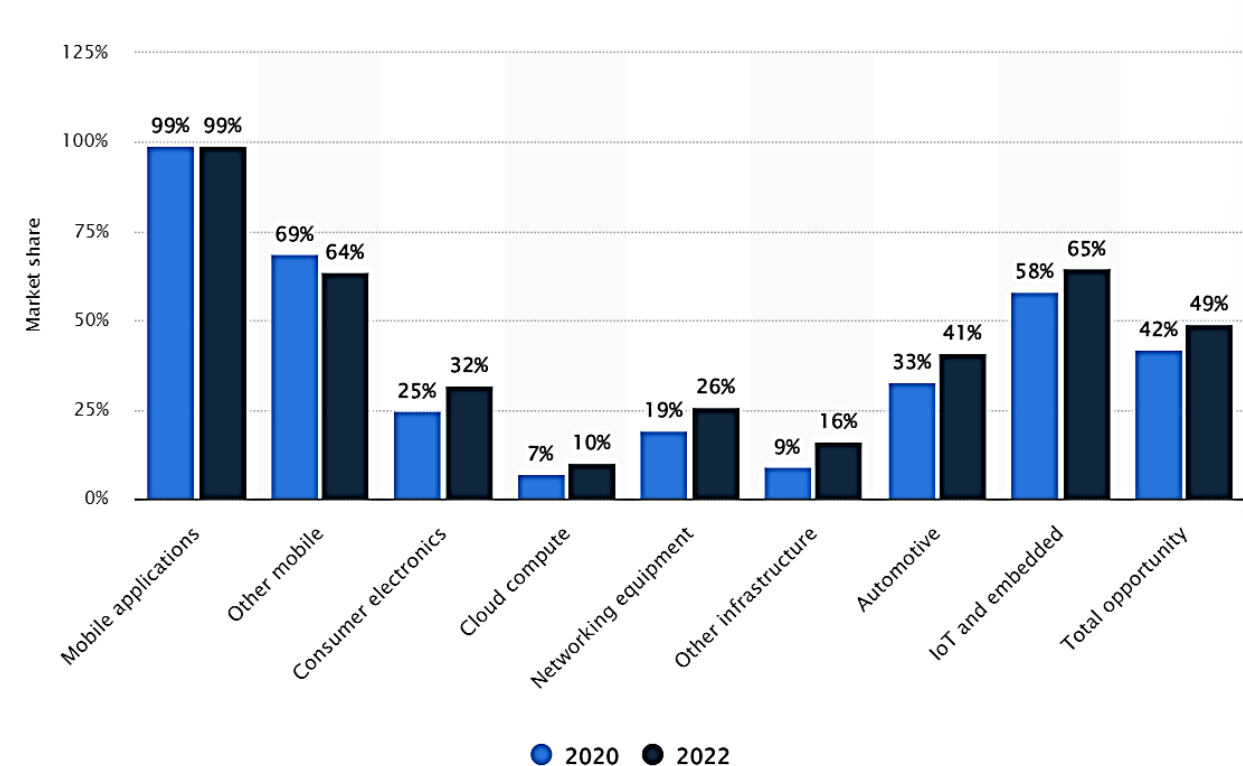
** MCU, MP, GPU, , AI, IoT, etc.*

Brief on ARM and STM32

Popular

1. Low energy consumption, low cost, high performance
2. Support 16/32 instruction sets
3. Many partners
4. Rich ecosystem
5. Many more reasons...

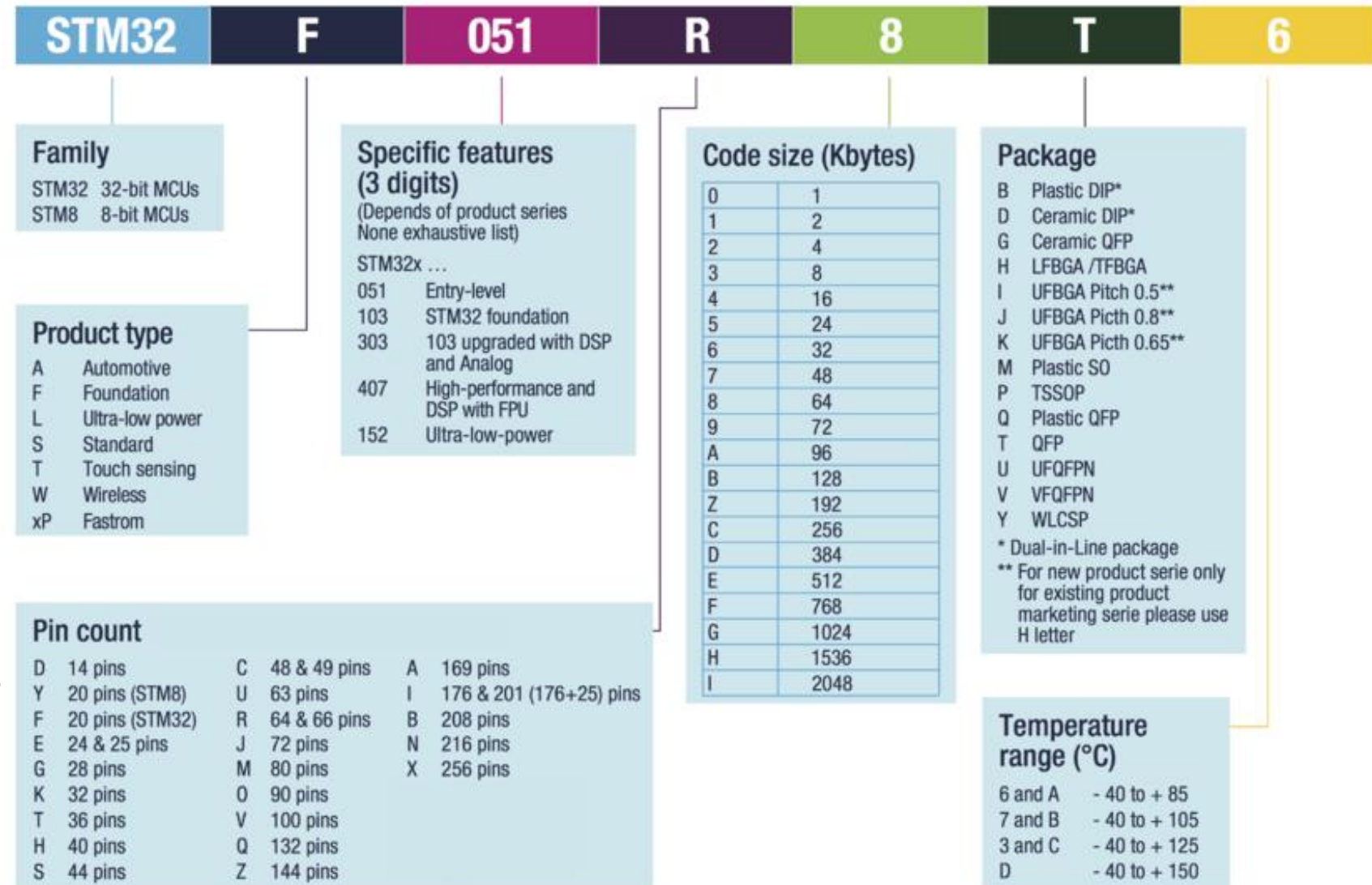
Market



ARM and STM32

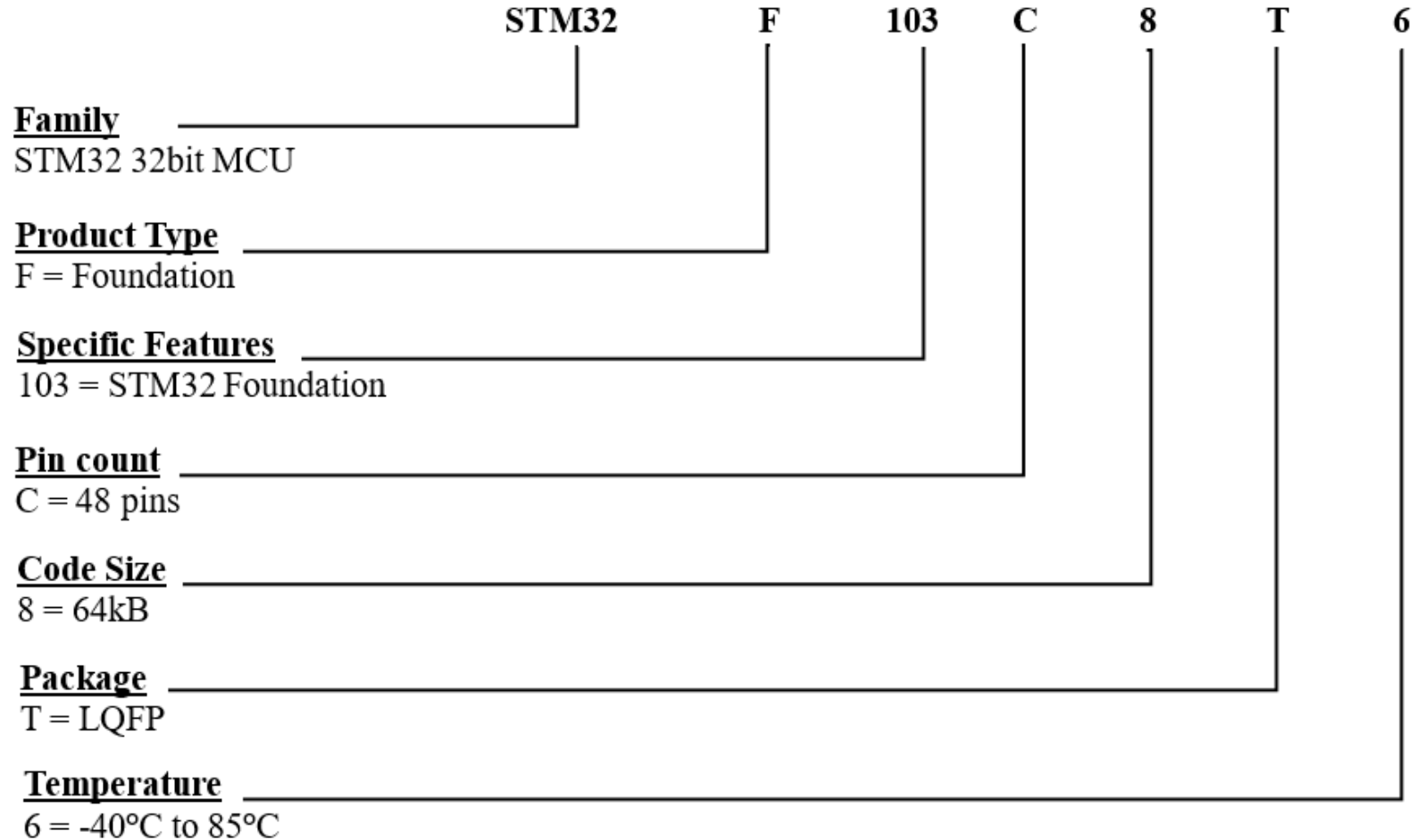
STM32 MCUs					32-bit Arm® Cortex®-M	<div>LONGEVITY 10 YEARS COMMITMENT</div>
<div>★</div> <div>High Performance</div>			<div>STM32F7</div> <div>1082 CoreMark 216 MHz Cortex-M7</div>		<div>STM32H7</div> <div>Up to 3224 CoreMark Up to 600 MHz Cortex-M7 240 MHz Cortex-M4</div>	
	<div>STM32F2</div> <div>398 CoreMark 120 MHz Cortex-M3</div>		<div>STM32F4</div> <div>608 CoreMark 180 MHz Cortex-M4</div>		<div>STM32H5</div> <div>Up to 1023 CoreMark 250 MHz Cortex-M33</div>	
<div>»</div> <div>Mainstream</div>	<div>STM32G0</div> <div>142 CoreMark 64 MHz Cortex-M0+</div>		<div>STM32G4</div> <div>569 CoreMark 170 MHz Cortex-M4</div>			
	<div>STM32C0</div> <div>114 CoreMark 48 MHz Cortex-M0+</div>	<div>STM32F0</div> <div>106 CoreMark 48 MHz Cortex-M0</div>	<div>STM32F1</div> <div>177 CoreMark 72 MHz Cortex-M3</div>	<div>STM32F3</div> <div>245 CoreMark 72 MHz Cortex-M4</div>	● Optimized for mixed-signal applications	
<div>🔋</div> <div>Ultra-low-power</div>			<div>STM32L4+</div> <div>409 CoreMark 120 MHz Cortex-M4</div>		<div>STM32U5</div> <div>651 CoreMark 160 MHz Cortex-M33</div>	
	<div>STM32L0</div> <div>75 CoreMark 32 MHz Cortex-M0+</div>	<div>STM32U0</div> <div>140 CoreMark 56 MHz Cortex-M0+</div>	<div>STM32L4</div> <div>273 CoreMark 80 MHz Cortex-M4</div>	<div>STM32L5</div> <div>443 CoreMark 110 MHz Cortex-M33</div>		
<div>📶</div> <div>Wireless</div>			<div>STM32WL</div> <div>162 CoreMark 48 MHz Cortex-M4 48 MHz Cortex-M0+</div>		<div>STM32WB</div> <div>216 CoreMark 64 MHz Cortex-M4 32 MHz Cortex-M0+</div>	
			<div>STM32WB0</div> <div>64 MHz Cortex-M0+</div>		<div>STM32WBA</div> <div>407 CoreMark 100 MHz Cortex-M33</div>	
						● Cortex-M0+ Radio co-processor

STM32 MCU



Example

STM32
F103C8T6



What is MCU?

- Quote from Wiki –

“A microcontroller (MC, UC, or μ C) or microcontroller unit (MCU) is a small computer on **a single integrated circuit**. A microcontroller contains one or more CPUs (**processor cores**) along with **memory** and **programmable input/output peripherals**.”

What is MCU?

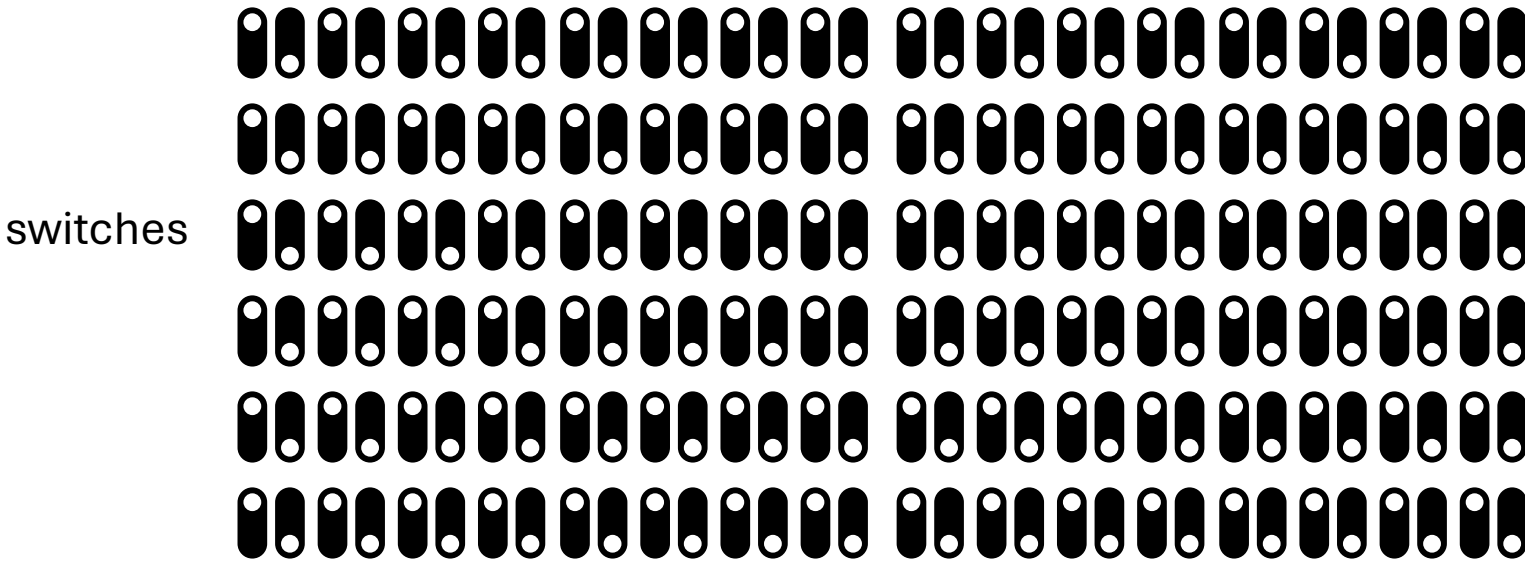
Embedded System

- Used in many applications
- home appliances, automotive systems, medical devices, and industrial control systems.
- consumer electronics products, such as gaming systems, digital cameras, and audio players.

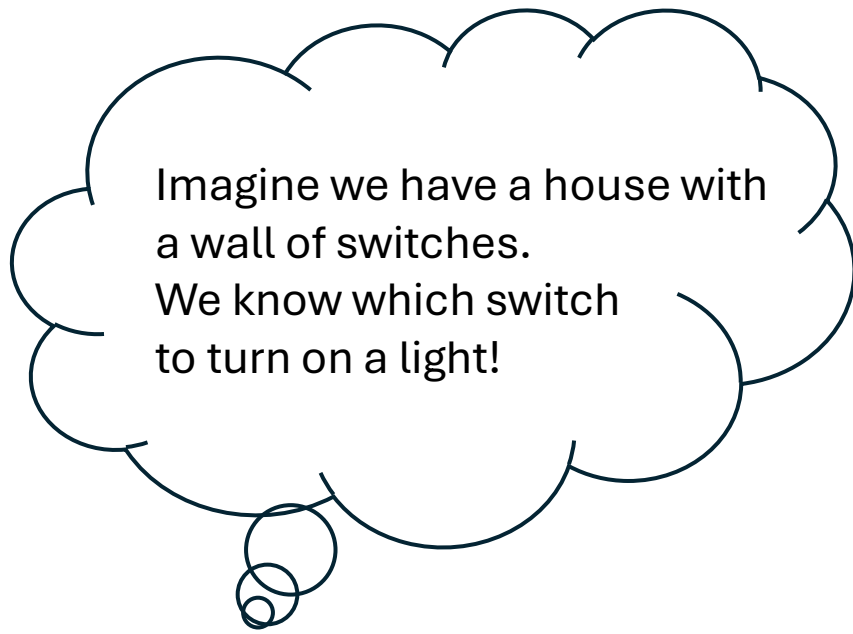
What is MCU?

Registers – Nontechnical concepts of “configurable switches”

RCC, BUS, GPIO, USART, TIMER, etc....



Many more....



What is MCU? – Register?

Example:

We want to blink an LED at PC13.

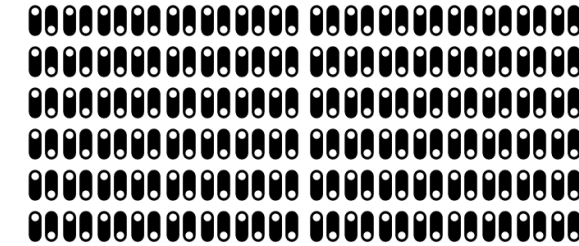
HAL is writing to a register
Namely BSRR,

Set ON/OFF at GPIOC, pin 13

```
HAL_GPIO_WritePin(GPIOC,  
GPIO_PIN_13, GPIO_PIN_RESET);
```



No worry! We are going to try this soon!



9.2.5 Port bit set/reset register (GPIOx_BSRR) (x=A..G)

Address offset: 0x10

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 mode only.

0: No action on the corresponding ODRx bit

1: Reset the corresponding ODRx bit

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

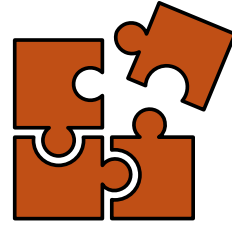
Bits 15:0 **BSy**: Port x Set bit y (y= 0 .. 15)

These bits are write-only and can be accessed in Word mode only.

0: No action on the corresponding ODRx bit

1: Set the corresponding ODRx bit

What is HAL?



- The HAL driver layer provides a simple, generic multi-instance set of **APIs (application programming interfaces)** to interact with the upper layer (application, libraries and stacks).
- The HAL driver **APIs** are split into two categories: generic APIs, which provide common and generic functions for all the STM32 series and extension APIs, which include specific and customized functions for a given line or part number. The HAL drivers include a complete set of ready-to-use **APIs** that simplify the user application implementation

Refer to: UM1850 User manual - Description of STM32F1 HAL and low-layer drivers

Training Kit and MCU

The training kit – the outlook

- Pinouts
- Schematic

MCU – STM32F103C8T6

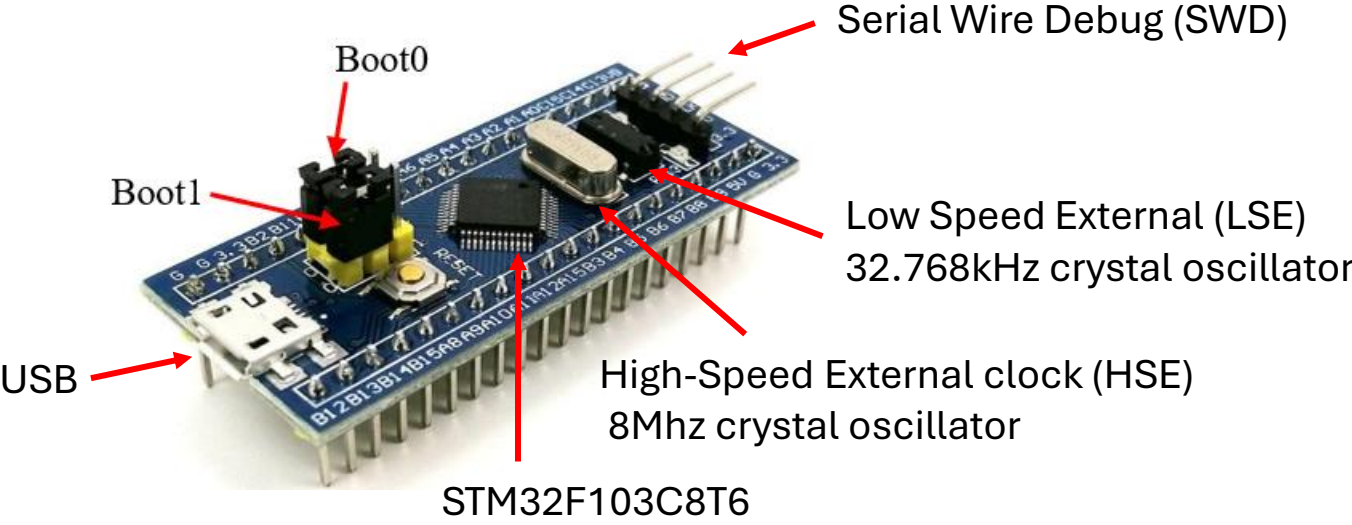
- Quick glimpse
- System Architecture
- Clock Tree
- GPIO

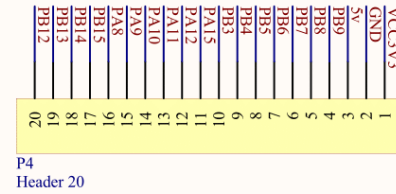
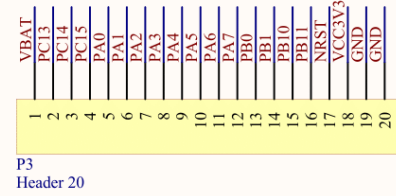
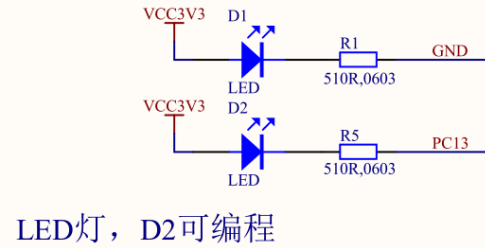
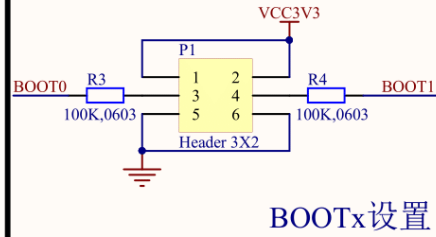
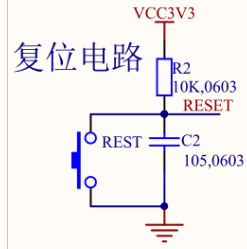
Many more on the datasheet and Manual as we explore further.

STM32F103C8T6 – datasheet

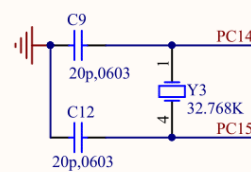
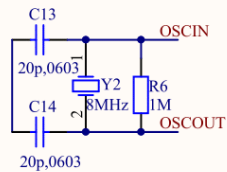
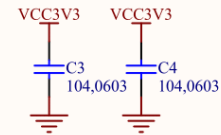
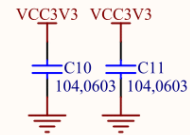
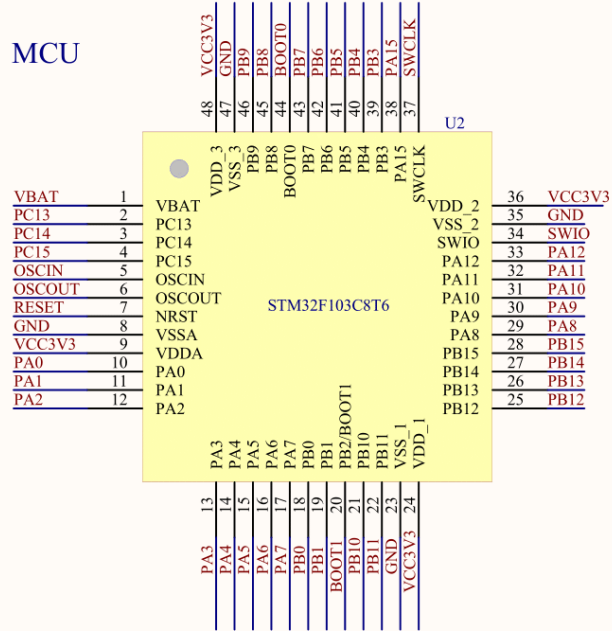
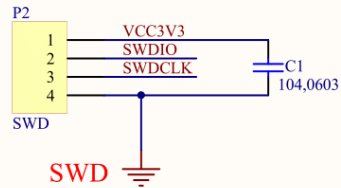
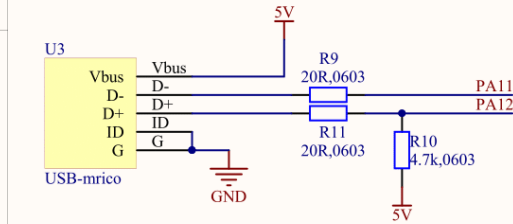
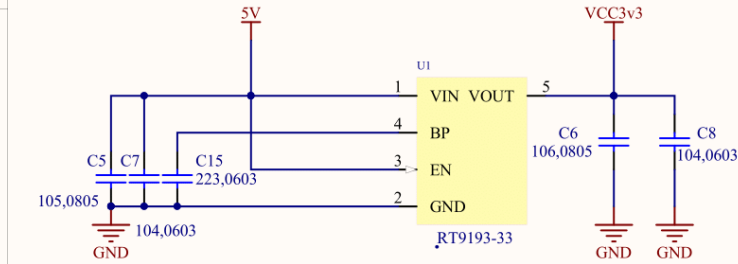
Table 9. Boot modes

Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
x	0	Main Flash memory	Main Flash memory is selected as boot space
0	1	System memory	System memory is selected as boot space
1	1	Embedded SRAM	Embedded SRAM is selected as boot space



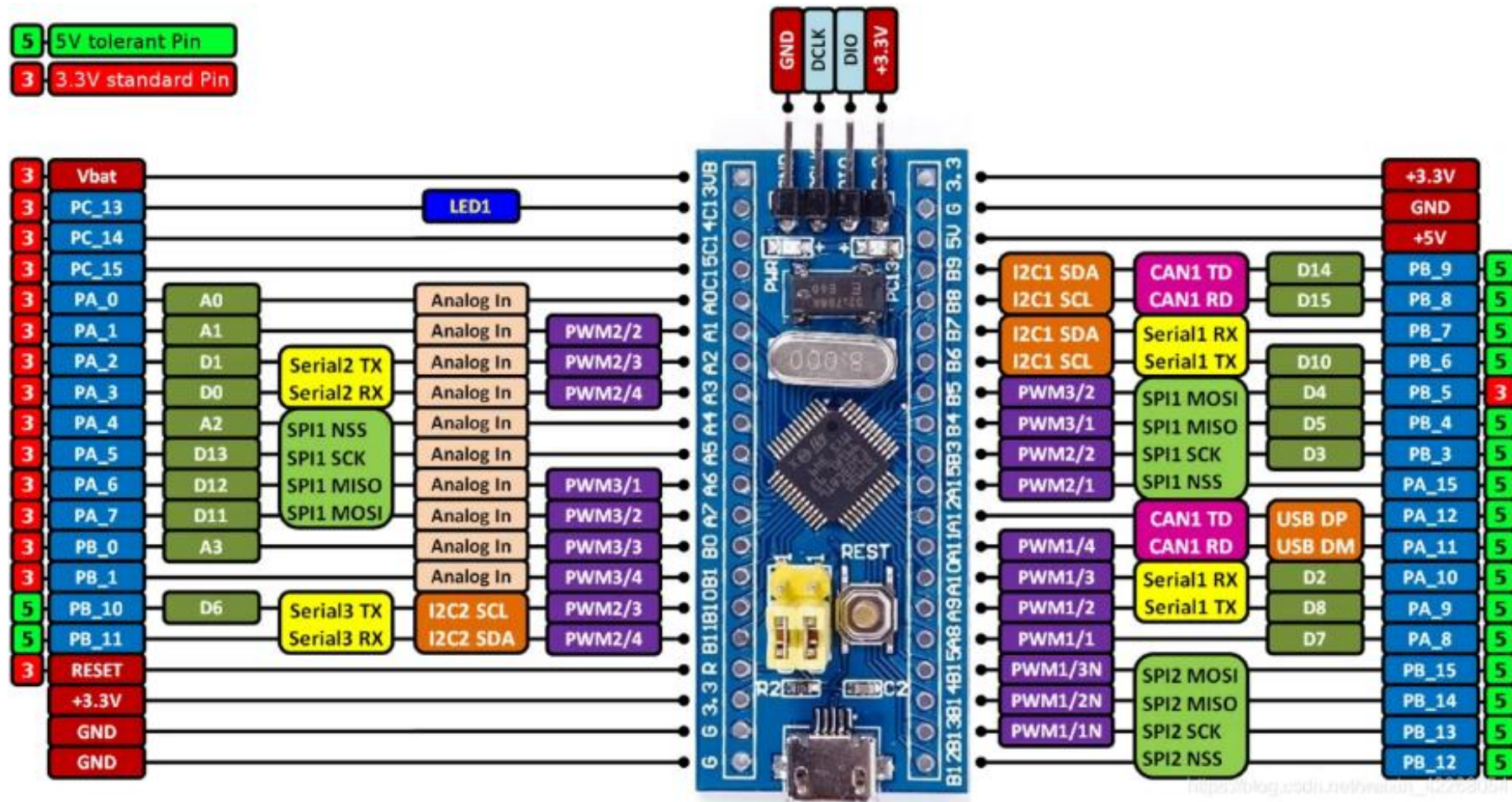


对外端子



Title			STM32F103C8T6核心板原理图		
Size	Number				Revision
A4	源地工作室www.vcc-gnd.com				
Date:	2016/1/26	Sheet of		10/10	
File:	E:\VCC-GND WORK FILE\STM32F103C8T6\STM32F103C8T6_SchDoc				

STM32F103C8T6 – pins



STM32F103C8T6 - pin definition

LQFP 48 pins




Find me!

[STM32F103C8T6 -pin definition.xlsx](#)

STM32F103C8T6 - pin definition
LQFP 48 pins

Pin No	Pin Name	Type	IO Level	Main Function	Alternate functions	
					Default	Remap
1	VBAT	S		VBAT		
2	PC13-TAMPER-RTC	I/O		PC13	TAMPER-RTC	
3	PC14-OSC32_IN	I/O		PC14	OSC32_IN	
4	PC15-OSC32_OUT	I/O		PC15	OSC32_OUT	
5	OSC_IN	I		OSC_IN		
6	OSC_OUT	O		OSC_OUT		
7	NRST	I/O		NRST		
8	VSSA	S		VSSA		
9	VDDA	S		VDDA		
10	PA0-WKUP	I/O		PA0	WKUP/USART2_CTS/ADC12_IN0/TIM2_CH1_ETR	
11	PA1	I/O		PA1	USART2_RTS/ADC12_IN1/TIM2_CH2	
12	PA2	I/O		PA2	USART2_TX/ADC12_IN2/TIM2_CH3	
13	PA3	I/O		PA3	USART2_RX/ADC12_IN3/TIM2_CH4	
14	PA4	I/O		PA4	SPI1_NSS/USART2_CK/ADC12_IN4	

Refer to the datasheet [Medium-density STM32F103xx pin definitions]



STM32F103C8T6

– Features and Peripherals

Term	Description	Term	Description
NVIC	Nested Vectored Interrupt Controller	CAN	CAN Comm.
SysTick	the Cortex® System Timer	USB	USB Comm.
RCC	Reset and Clock Control	RTC	Real-time clock
GPIO	General-purpose I/O	CRC	Cyclic Redundancy Check
AFIO	Alternate-function I/O	PWR	Power Control
EXTI	External interrupt/event controller	BKP	Backup registers
TIM	Timer	IWDG	Independent watchdog
ADC	Analog-to-Digital Converter	WWDG	Window watchdog
DMA	Direct memory access	DAC	Digital-to-analog converter
USART	USART Comm.	SDIO	SD Interface
I2C	I2C Comm.	FSMC	Flexible static memory controller
SPI	SPI Comm.	USB OTG	USB

STM32F103C8T6 – Important References

Documents

- Datasheets
- RM0008 Reference manual



Websites

- Official site - ST

<https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html>

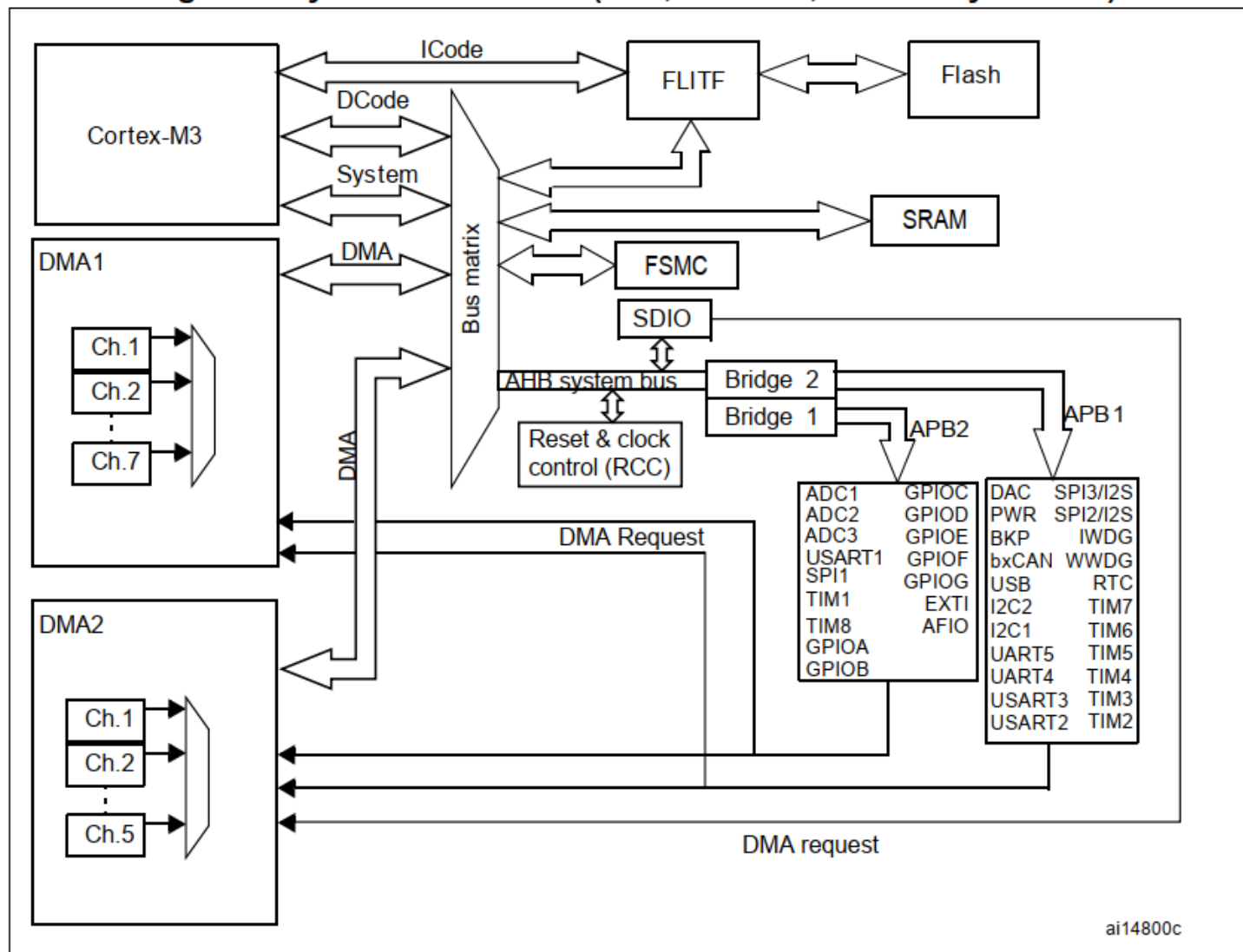
- Many learning sites...



Let me explain!

System architecture

Figure 1. System architecture (low-, medium-, XL-density devices)



Advanced High-performance **Bus (AHB)**
Advanced Peripheral **Bus (APB)**



What is this?

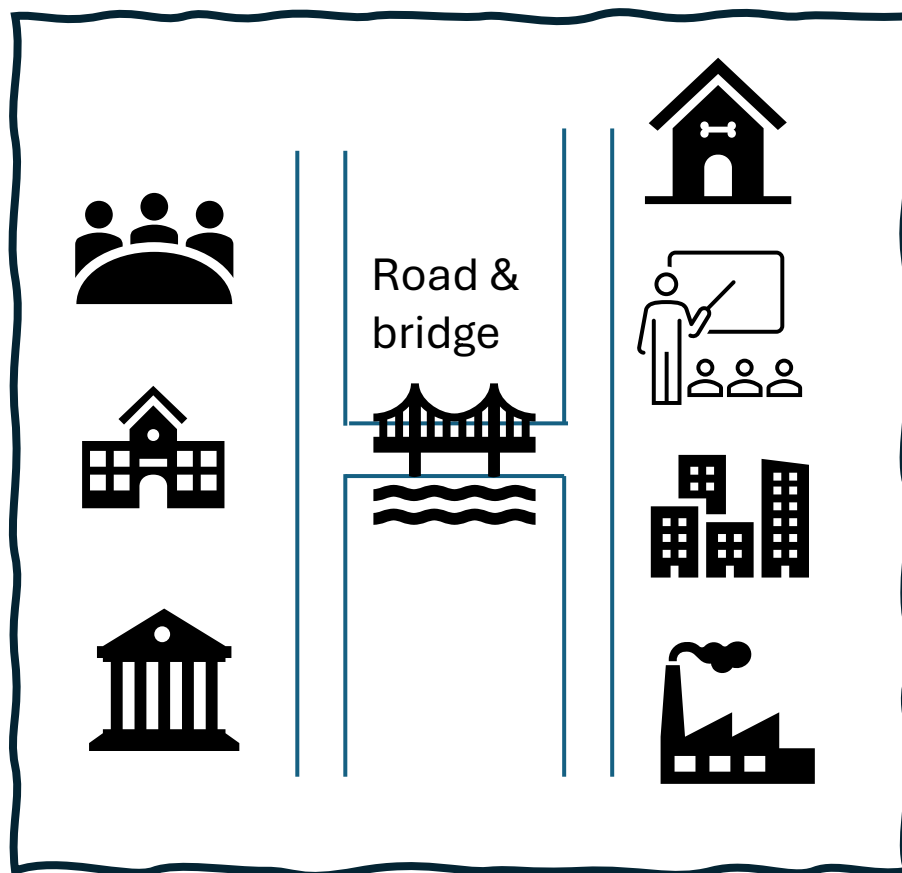


Let me explain!



System architecture

Administrators
Council, town
hall, bank, etc.



Simplified idea of “Town Plan”

House, factory,
commercial
buildings, school,
etc.

System clock

**Refer to
Clock tree**



Stay cool!
You do not have to understand
fully now!

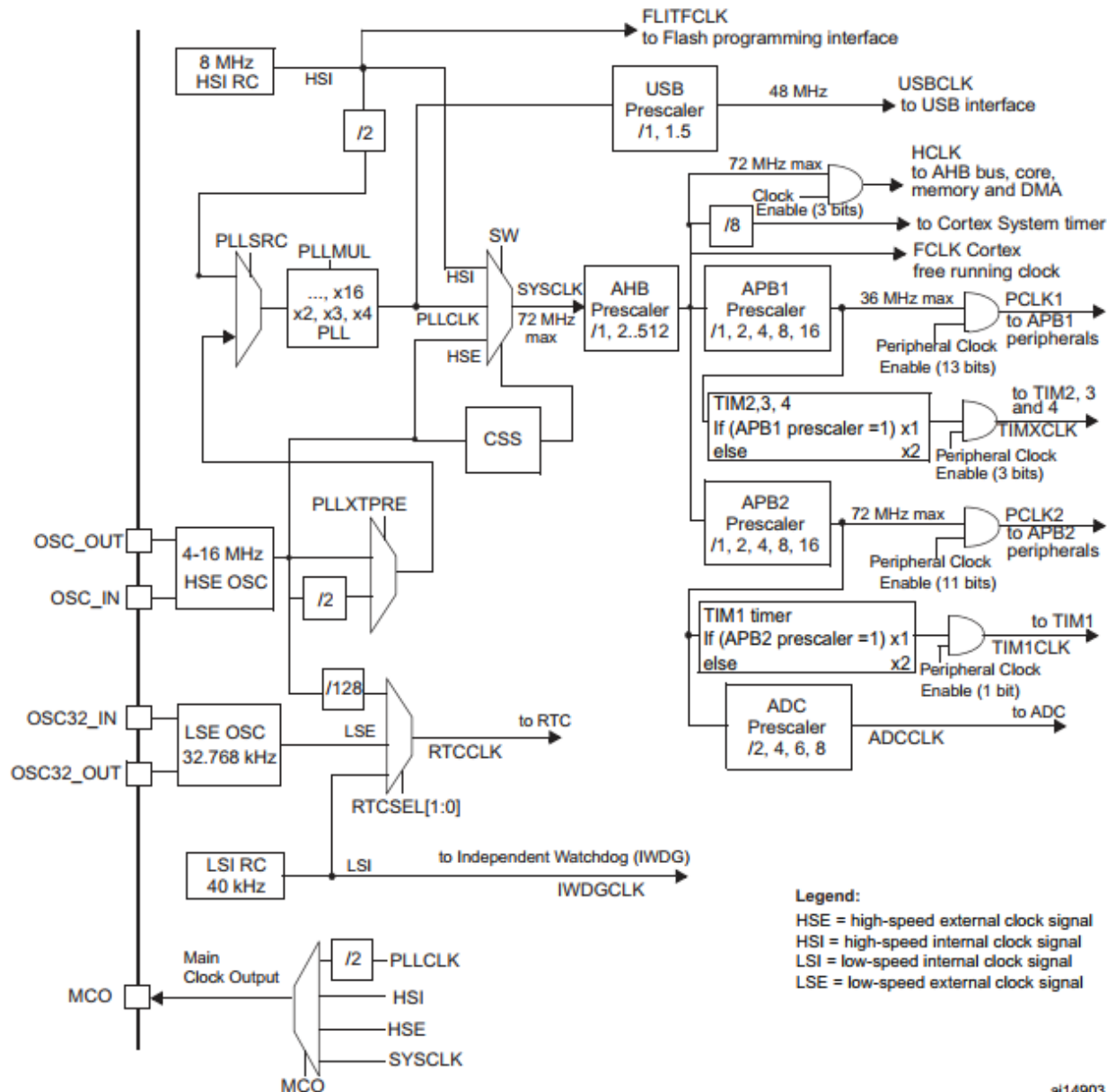
(Datasheet)

STM32F103x8

STM32F103xB

**Medium-density performance line Arm®-based 32-bit MCU with
64 or 128 KB Flash, USB, CAN, 7 timers, 2 ADCs, 9 com. interfaces**

Figure 2. Clock tree



Clock Tree



Let me explain!



High-Speed Internal clock (HSI) RC oscillator
 High-Speed External clock (HSE) crystal oscillator
 Phase-locked loop or phase lock loop (PLL) clock
 Low Speed External (LSE)
 Low Speed Internal (LSI)

STM32-F103C8T6 (See Schematic)
 Crystal oscillator – 8MHz and 32.768kHz



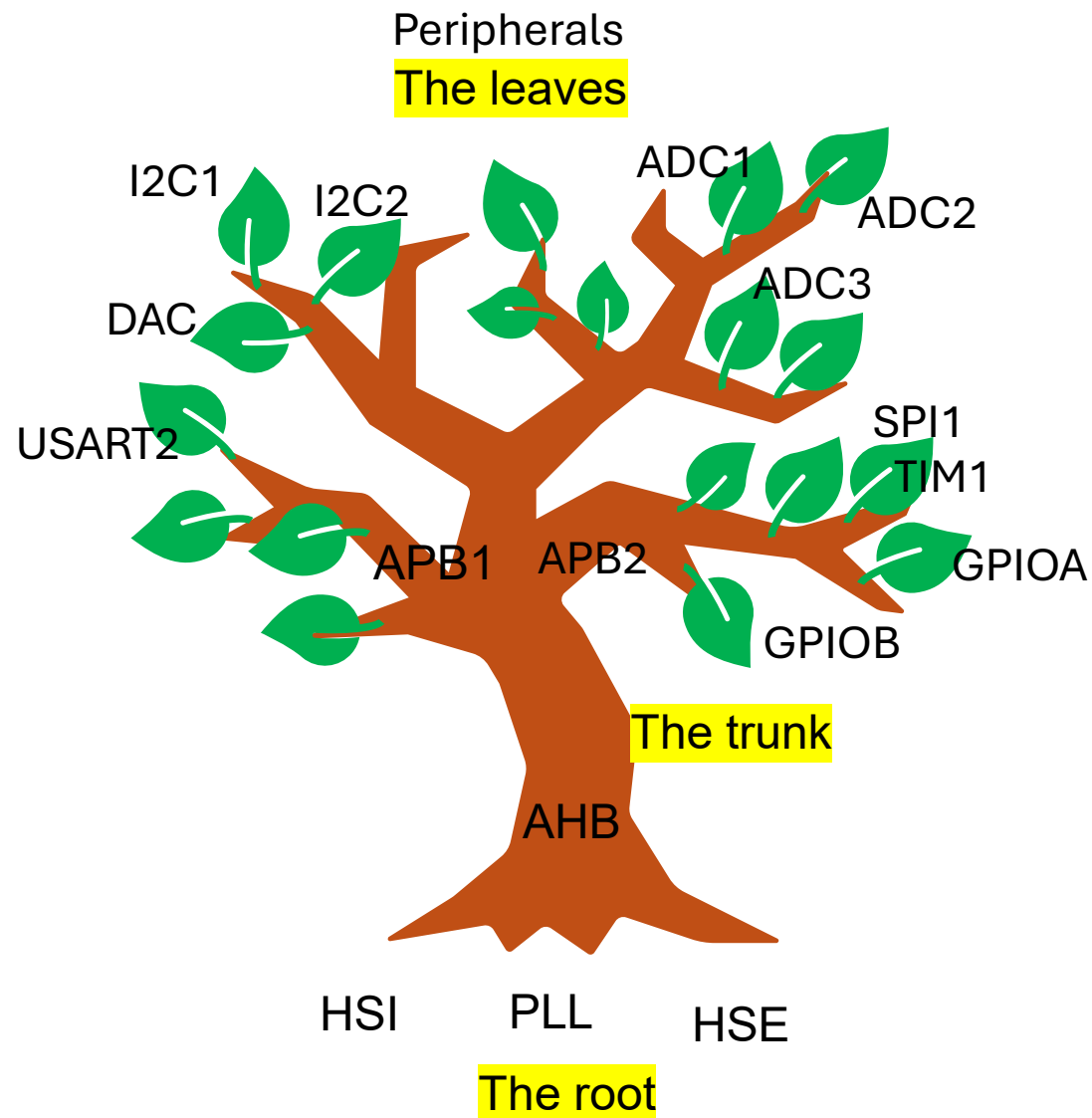
What is this?

STM32CubeIDE
 Try to read together *.ioc | Clock Configuration

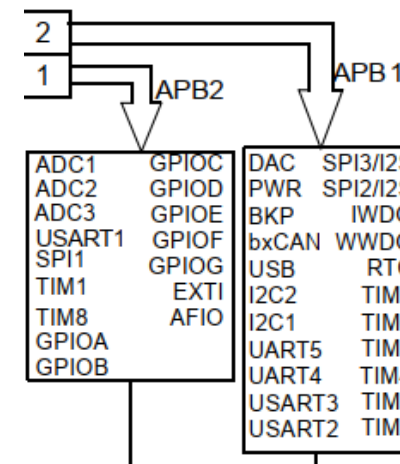


Clock Tree

Let me explain!



Simplified idea of a “tree”



STM32CubeIDE

Try to read together *.ioc | Clock Configuration

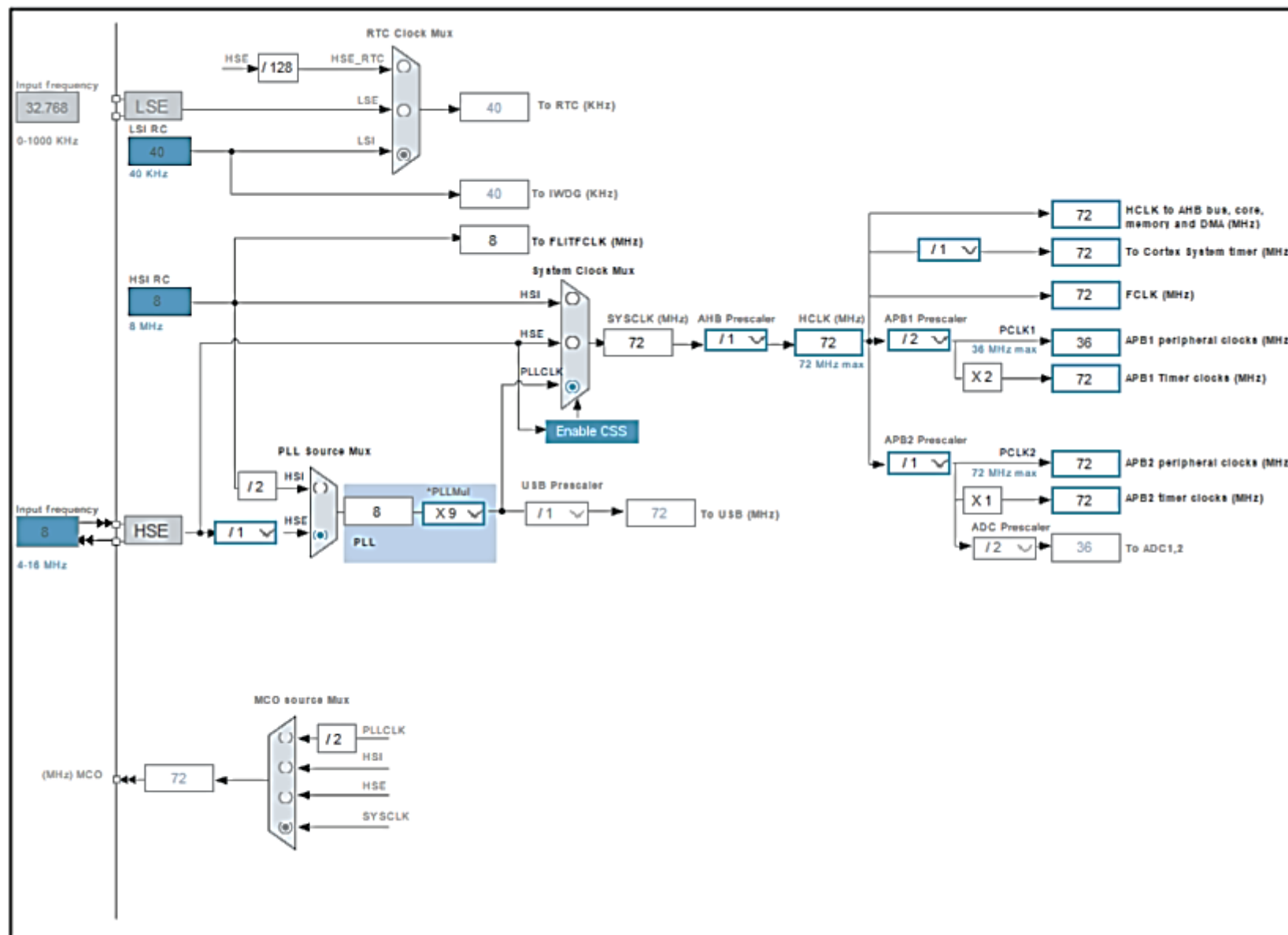


Let me explain!

Clock Tree

STM32CubeIDE

Try to read together *.ioc | Clock Configuration



General Purpose Input Output

Refer to

**9 General-purpose and alternate-function I/Os
(GPIOs and AFIOs)**

RM0008

Reference manual

STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and
STM32F107xx advanced Arm®-based 32-bit MCUs



Stay cool!
You do not have to understand
fully now!



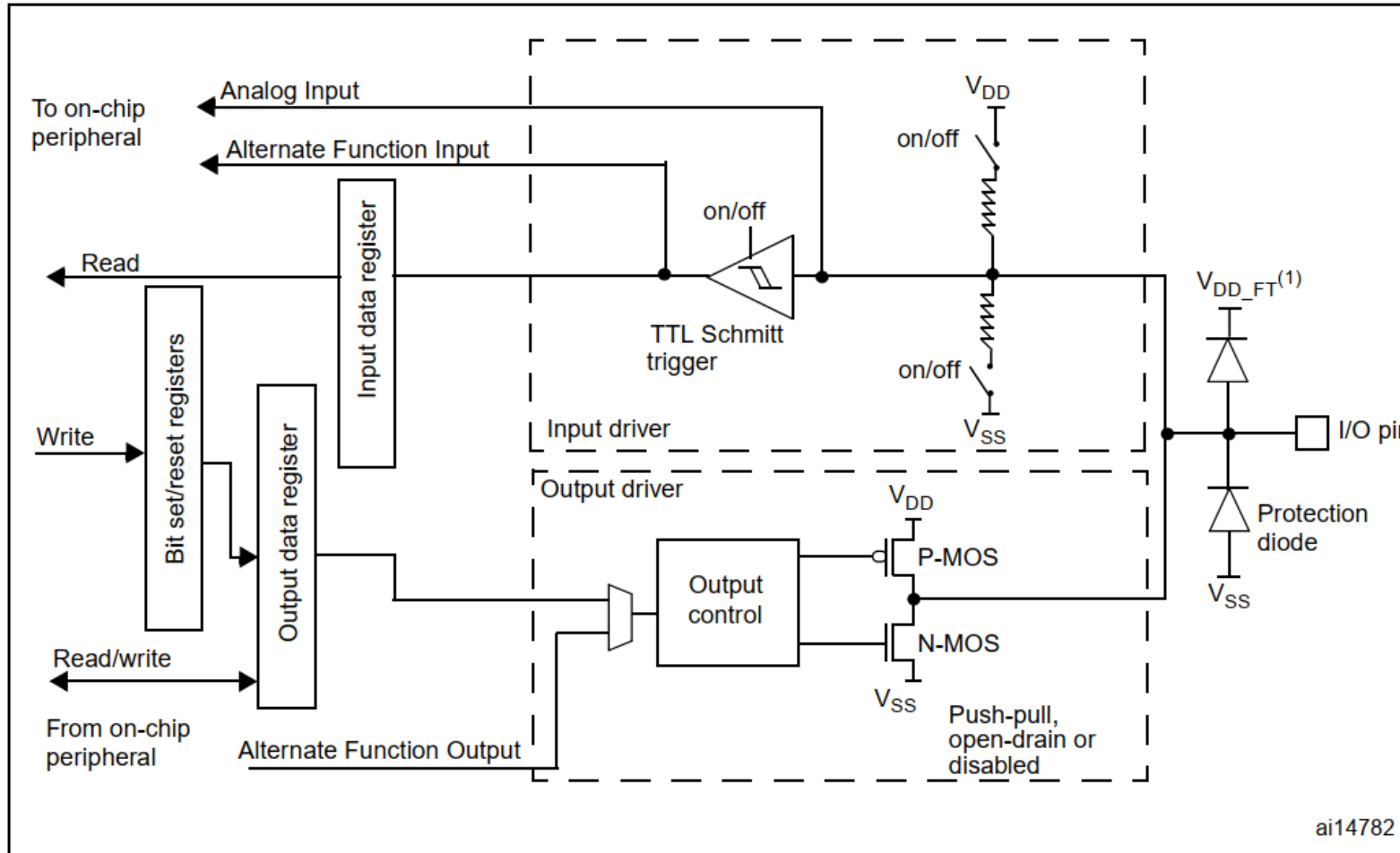
GPIO

Let me explain!

General Purpose IO (GPIO) Ports, can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain
- Output push-pull
- Alternate function push-pull
- Alternate function open-drain

Figure 14. Basic structure of a 5-Volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to 5-Volt tolerant I/Os, and different from V_{DD} .



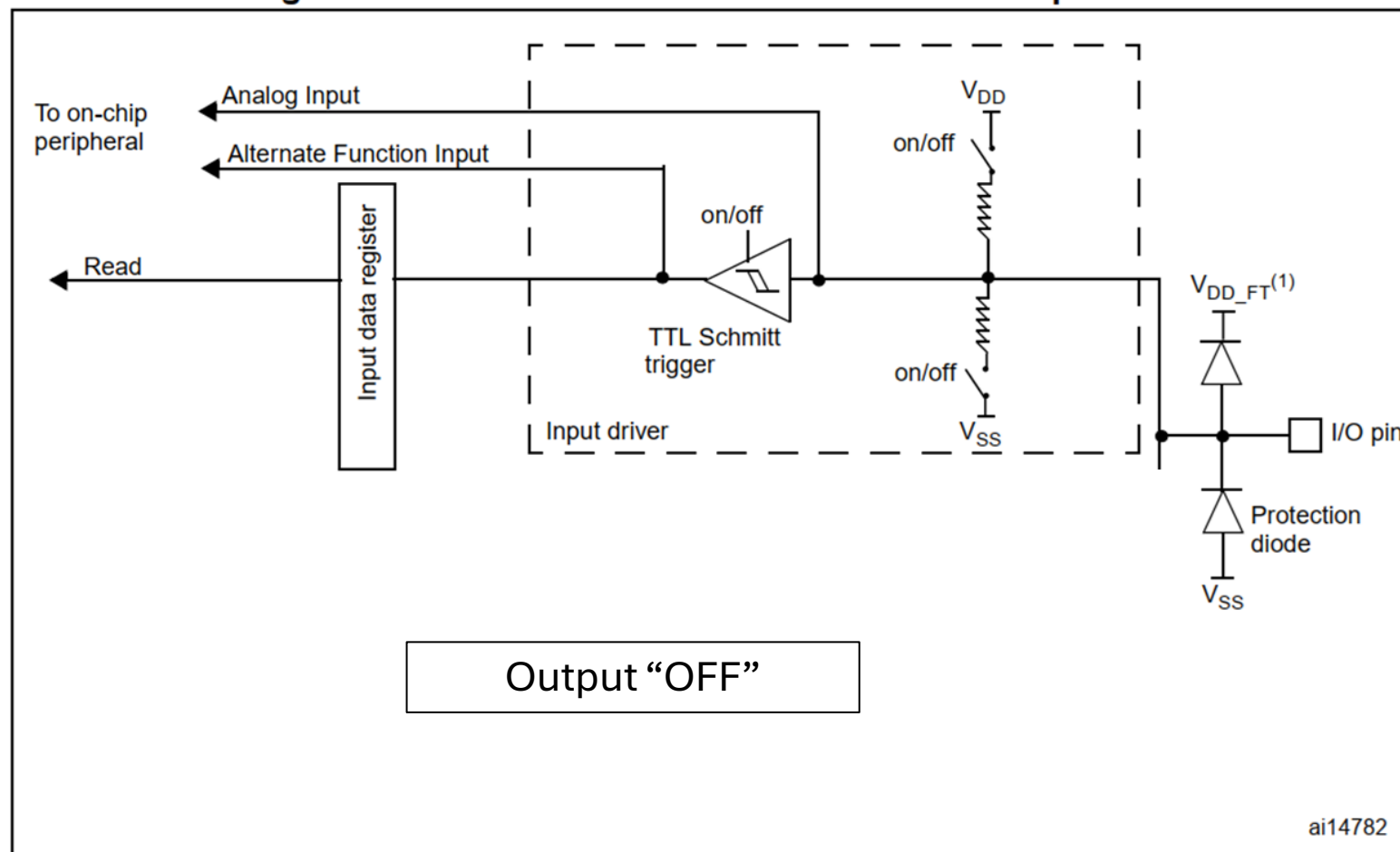
GPIO

Let me explain!

Input

- Input floating
- Input pull-up
- Input-pull-down
- Analog

Figure 14. Basic structure of a 5-Volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to 5-Volt tolerant I/Os, and different from V_{DD} .

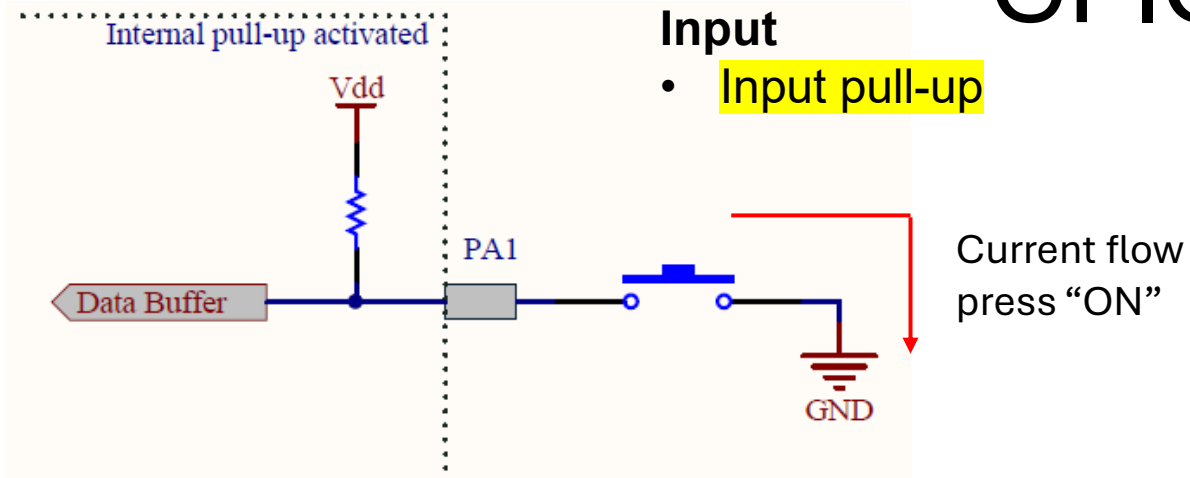
GPIO



Let me explain!

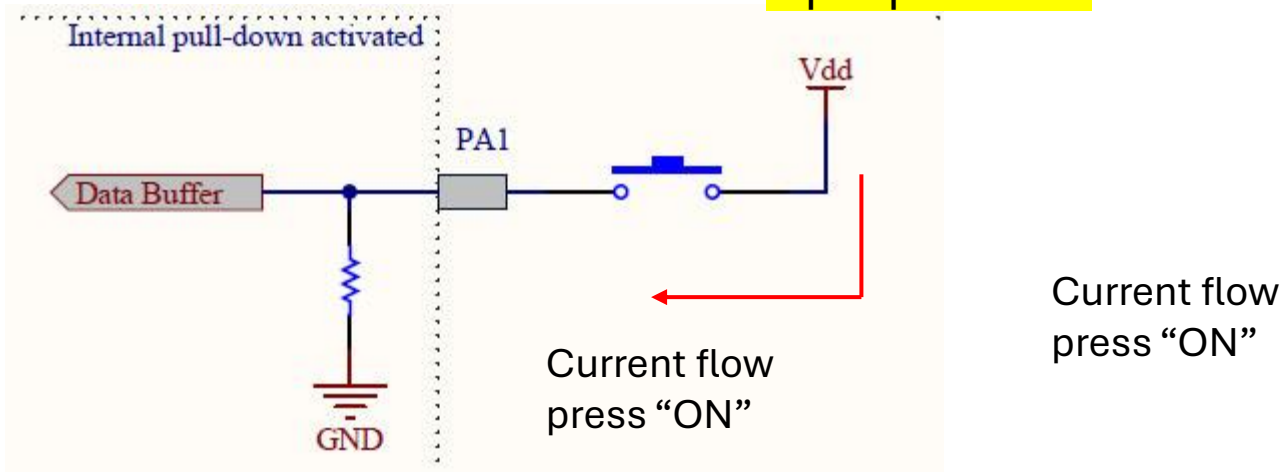
Input

- Input pull-up



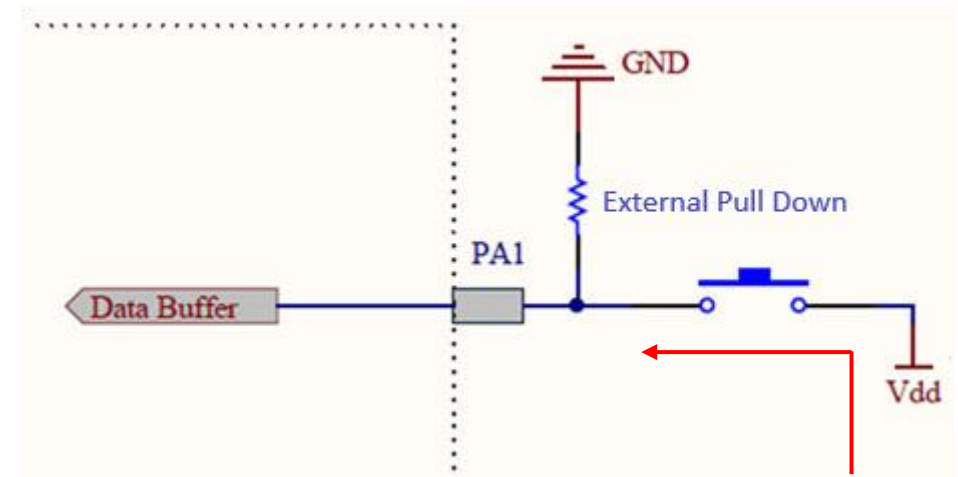
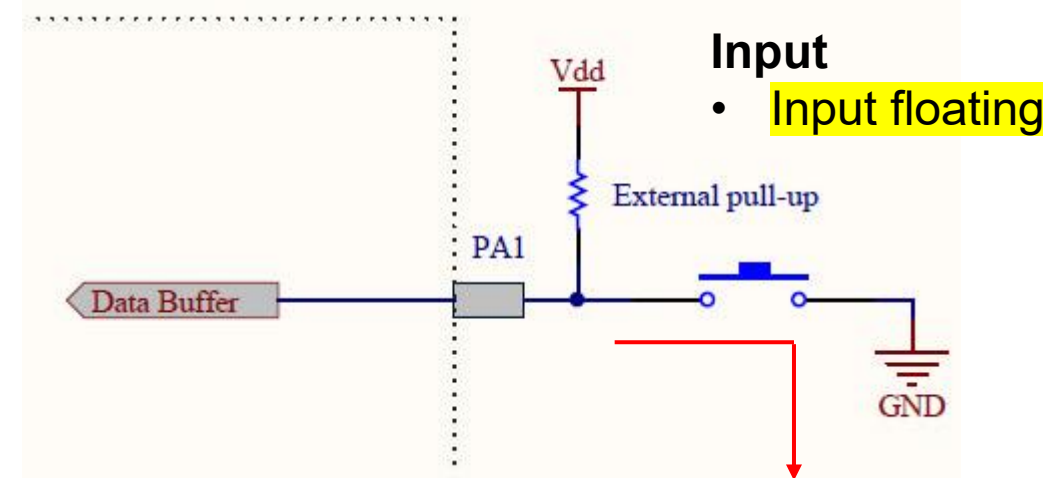
Input

- Input-pull-down



Input

- Input floating





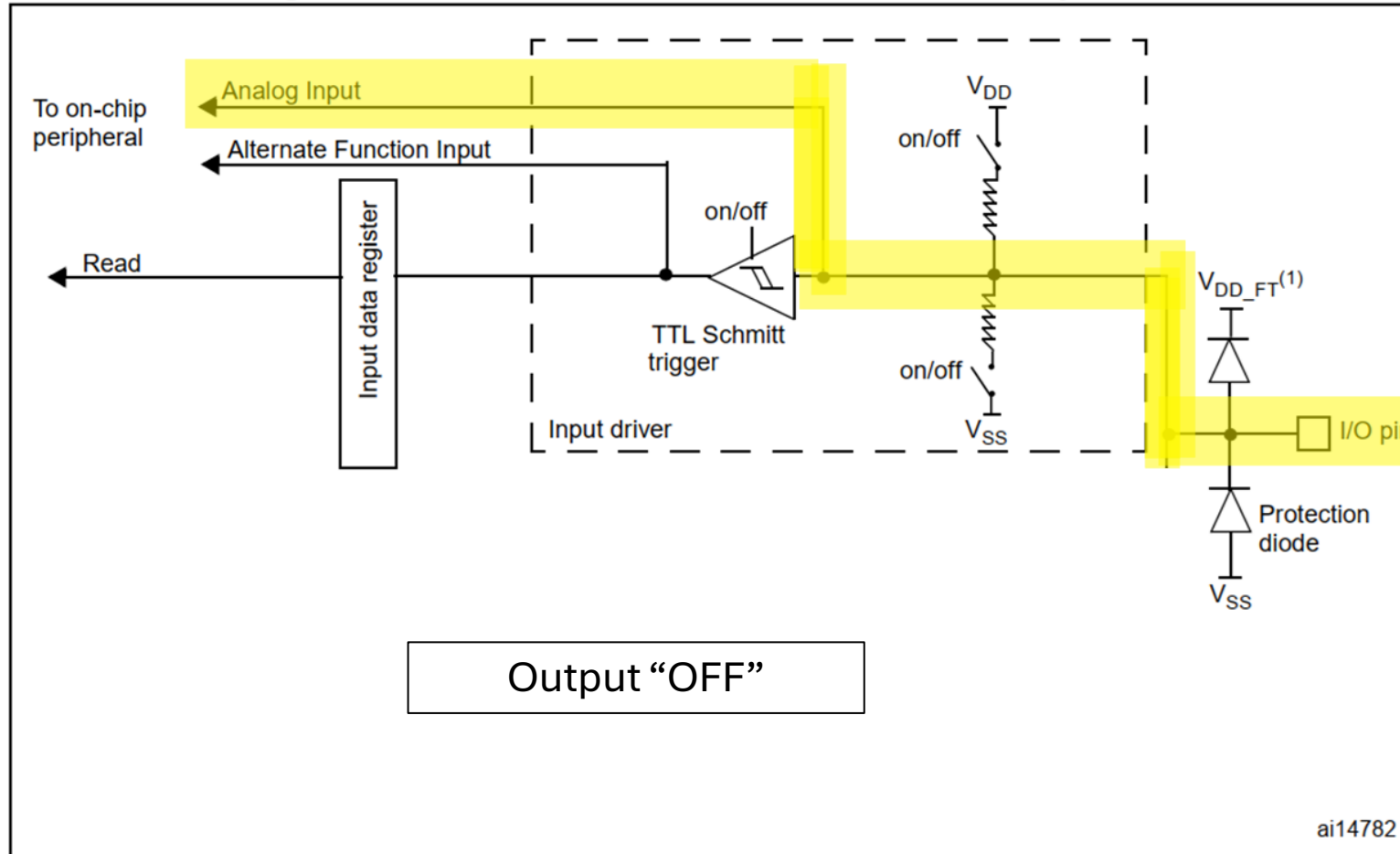
GPIO

Let me explain!

Input

- Analog

Figure 14. Basic structure of a 5-Volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to 5-Volt tolerant I/Os, and different from V_{DD} .



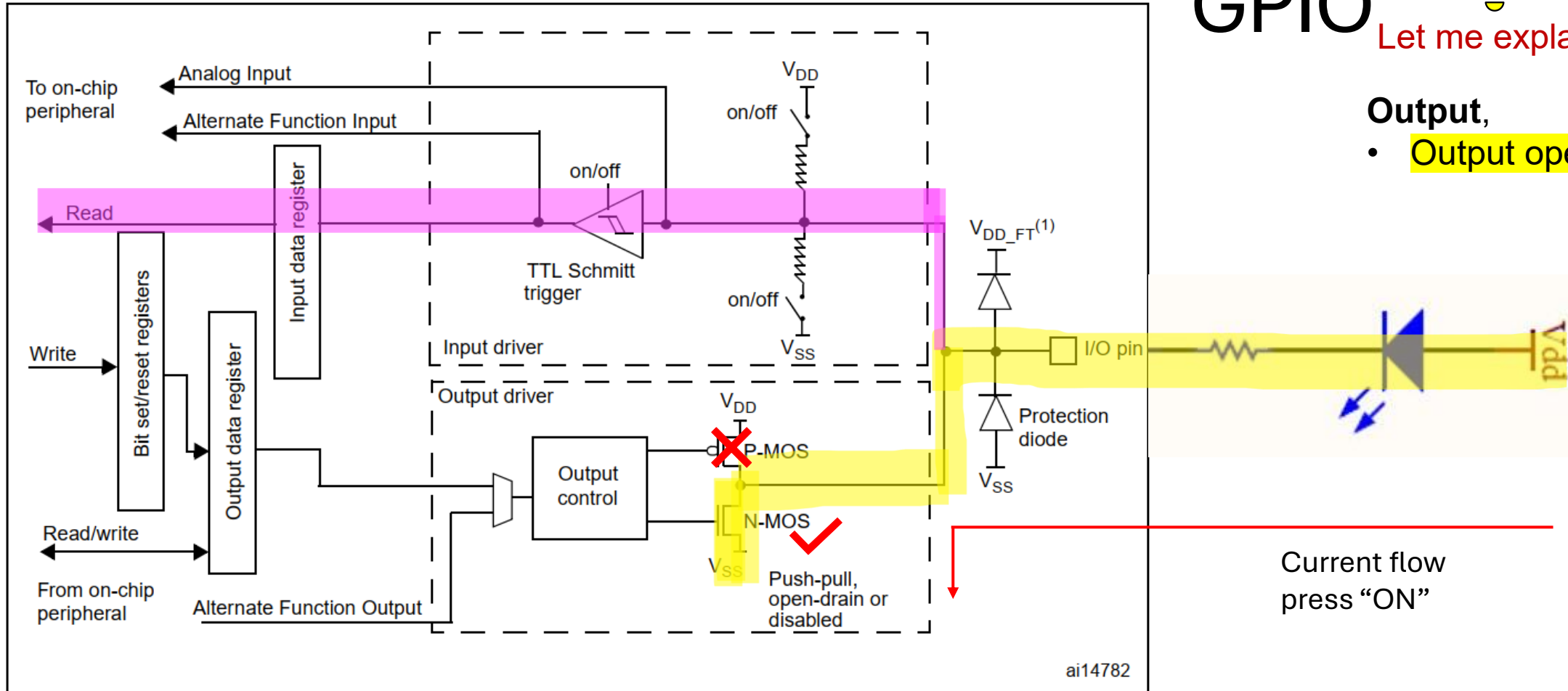
GPIO

Let me explain!

Output,

- Output open-drain

Figure 14. Basic structure of a 5-Volt tolerant I/O port bit



Current flow
press "ON"

ai14782

1. V_{DD_FT} is a potential specific to 5-Volt tolerant I/Os, and different from V_{DD} .



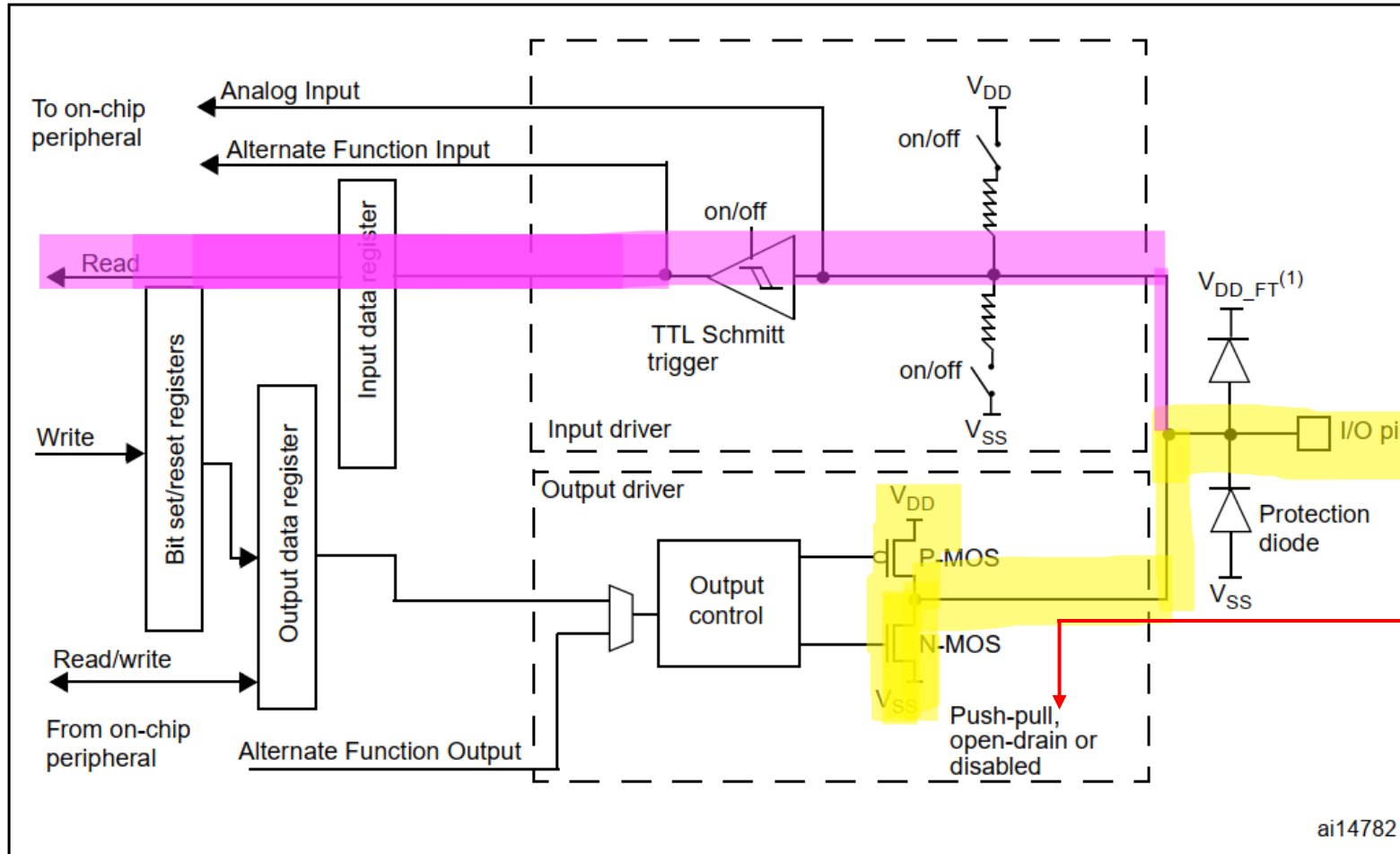
GPIO

Let me explain!

Output,

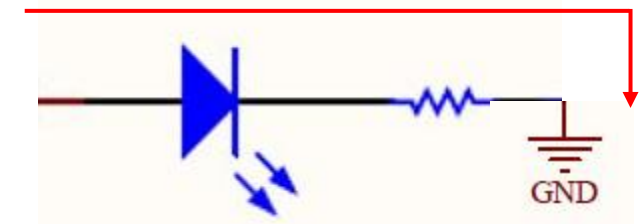
- Output push-pull

Figure 14. Basic structure of a 5-Volt tolerant I/O port bit



Current flow press "ON" if N-MOS "ON" and P-MOS "OFF"

Current flow press "ON" if P-MOS "ON" and N-MOS "OFF"



1. V_{DD_FT} is a potential specific to 5-Volt tolerant I/Os, and different from V_{DD} .

Can also wired this way!

Nested vectored interrupt controller (NVIC)



Refer to
10 Interrupts and events



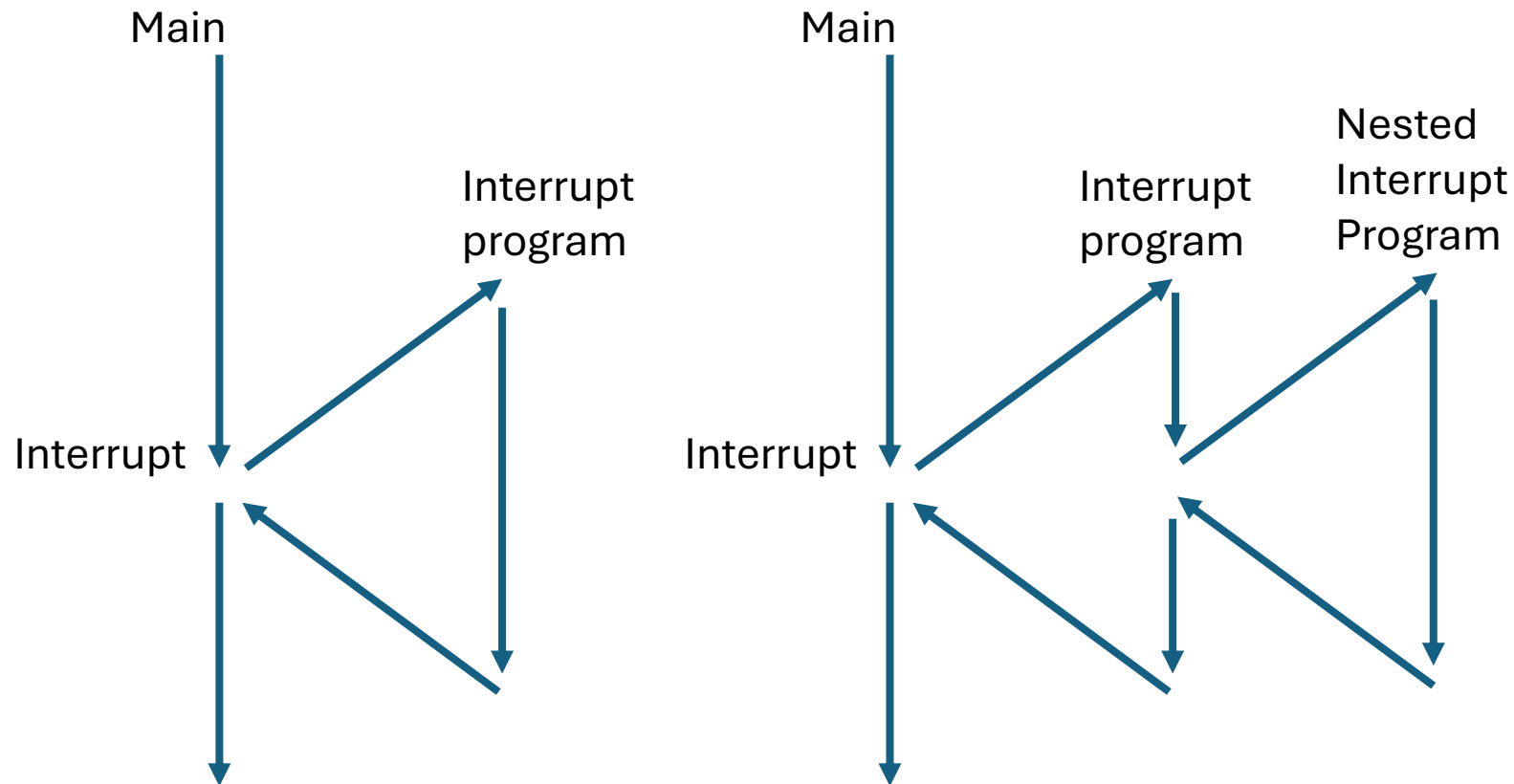
Stay cool!
You do not have to understand
fully now!

RM0008

Reference manual

STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and
STM32F107xx advanced Arm®-based 32-bit MCUs

Interrupts and events



```

1 int main(void)
2 {
3
4     while(1)
5     {
6         // main program
7     }
8 }
9
10 void EXTI15_10_IRQHandler(void) {
11
12     // interrupt program
13 }
  
```

See -> Nested vectored interrupt controller (NVIC)



Nested vectored interrupt controller (NVIC)

Table 62. Vector table for XL-density devices

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000_0000
-	-3	fixed	Reset	Reset	0x0000_0004
-	-2	fixed	NMI	Nonmaskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000_0008
-	-1	fixed	HardFault	All class of fault	0x0000_000C
-	0	settable	MemManage	Memory management	0x0000_0010
-	1	settable	BusFault	Prefetch fault, memory access fault	0x0000_0014
-	2	settable	UsageFault	Undefined instruction or illegal state	0x0000_0018
-	-	-	-	Reserved	0x0000_001C-0x0000_002B
-	3	settable	SVCall	System service call via SWI instruction	0x0000_002C
-	4	settable	Debug Monitor	Debug monitor	0x0000_0030
-	-	-	-	Reserved	0x0000_0034
-	5	settable	PendSV	Pendable request for system service	0x0000_0038
-	6	settable	SysTick	Systick timer	0x0000_003C
0	7	settable	WWDG	Window watchdog interrupt	0x0000_0040
1	8	settable	PVD	PVD through EXTI Line detection interrupt	0x0000_0044
2	9	settable	TAMPER	Tamper interrupt	0x0000_0048
3	10	settable	RTC	RTC global interrupt	0x0000_004C
4	11	settable	FLASH	Flash global interrupt	0x0000_0050
5	12	settable	RCC	RCC global interrupt	0x0000_0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000_0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000_005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000_0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000_0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000_0068
11	18	settable	DMA1_Channel1	DMA1 Channel1 global interrupt	0x0000_006C
12	19	settable	DMA1_Channel2	DMA1 Channel2 global interrupt	0x0000_0070
13	20	settable	DMA1_Channel3	DMA1 Channel3 global interrupt	0x0000_0074

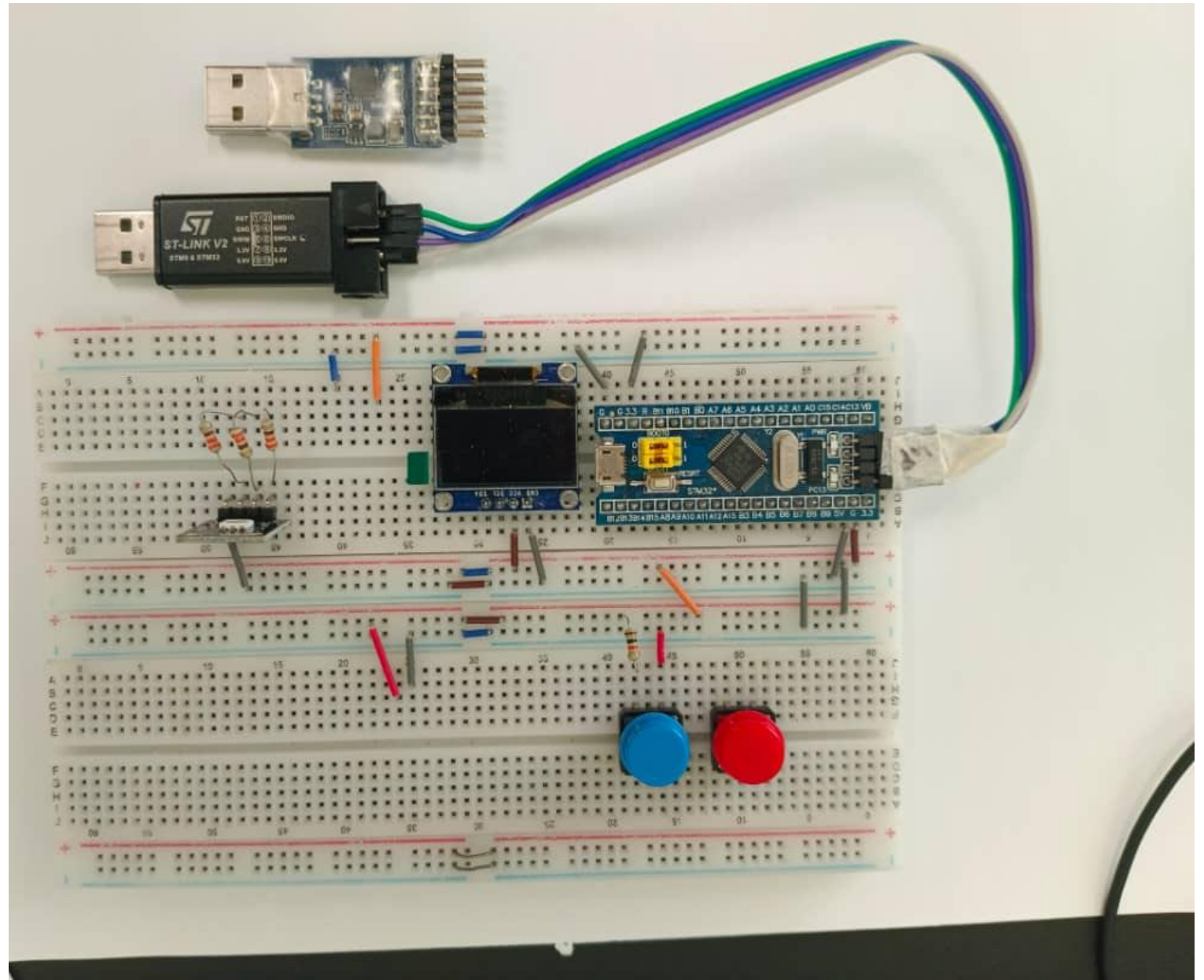
Table 62. Vector table for XL-density devices (continued)

Position	Priority	Type of priority	Acronym	Description	Address
14	21	settable	DMA1_Channel4	DMA1 Channel4 global interrupt	0x0000_0078
15	22	settable	DMA1_Channel5	DMA1 Channel5 global interrupt	0x0000_007C
16	23	settable	DMA1_Channel6	DMA1 Channel6 global interrupt	0x0000_0080
17	24	settable	DMA1_Channel7	DMA1 Channel7 global interrupt	0x0000_0084
18	25	settable	ADC1_2	ADC1 and ADC2 global interrupt	0x0000_0088
19	26	settable	USB_HP_CAN_TX	USB high priority or CAN TX interrupts	0x0000_008C
20	27	settable	USB_LP_CAN_RX0	USB low priority or CAN RX0 interrupts	0x0000_0090
21	28	settable	CAN_RX1	CAN RX1 interrupt	0x0000_0094
22	29	settable	CAN_SCE	CAN SCE interrupt	0x0000_0098
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000_009C
24	31	settable	TIM1_BRK_TIM9	TIM1 Break interrupt and TIM9 global interrupt	0x0000_00A0
25	32	settable	TIM1_UP_TIM10	TIM1 Update interrupt and TIM10 global interrupt	0x0000_00A4
26	33	settable	TIM1_TRG_COM_TIM11	TIM1 Trigger and Commutation interrupts and TIM11 global interrupt	0x0000_00A8
27	34	settable	TIM1_CC	TIM1 Capture Compare interrupt	0x0000_00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000_00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000_00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000_00B8
31	38	settable	I2C1_EV	I2C1 event interrupt	0x0000_00BC
32	39	settable	I2C1_ER	I2C1 error interrupt	0x0000_00C0
33	40	settable	I2C2_EV	I2C2 event interrupt	0x0000_00C4
34	41	settable	I2C2_ER	I2C2 error interrupt	0x0000_00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000_00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000_00D0
37	44	settable	USART1	USART1 global interrupt	0x0000_00D4
38	45	settable	USART2	USART2 global interrupt	0x0000_00D8
39	46	settable	USART3	USART3 global interrupt	0x0000_00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000_00E0

Table 62. Vector table for XL-density devices (continued)

Position	Priority	Type of priority	Acronym	Description	Address
41	48	settable	RTCAlarm	RTC alarm through EXTI line interrupt	0x0000_00E4
42	49	settable	USBWakeUp	USB wakeup from suspend through EXTI line interrupt	0x0000_00E8
43	50	settable	TIM8_BRK_TIM12	TIM8 Break interrupt and TIM12 global interrupt	0x0000_00EC
44	51	settable	TIM8_UP_TIM13	TIM8 Update interrupt and TIM13 global interrupt	0x0000_00F0
45	52	settable	TIM8_TRG_COM_TIM14	TIM8 Trigger and Commutation interrupts and TIM14 global interrupt	0x0000_00F4
46	53	settable	TIM8_CC	TIM8 Capture Compare interrupt	0x0000_00F8
47	54	settable	ADC3	ADC3 global interrupt	0x0000_00FC
48	55	settable	FSMC	FSMC global interrupt	0x0000_0100
49	56	settable	SDIO	SDIO global interrupt	0x0000_0104
50	57	settable	TIM5	TIM5 global interrupt	0x0000_0108
51	58	settable	SPI3	SPI3 global interrupt	0x0000_010C
52	59	settable	UART4	UART4 global interrupt	0x0000_0110
53	60	settable	UART5	UART5 global interrupt	0x0000_0114
54	61	settable	TIM6	TIM6 global interrupt	0x0000_0118
55	62	settable	TIM7	TIM7 global interrupt	0x0000_011C
56	63	settable	DMA2_Channel1	DMA2 Channel1 global interrupt	0x0000_0120
57	64	settable	DMA2_Channel2	DMA2 Channel2 global interrupt	0x0000_0124
58	65	settable	DMA2_Channel3	DMA2 Channel3 global interrupt	0x0000_0128
59	66	settable	DMA2_Channel4_5	DMA2 Channel4 and DMA2 Channel5 global interrupts	0x0000_012C

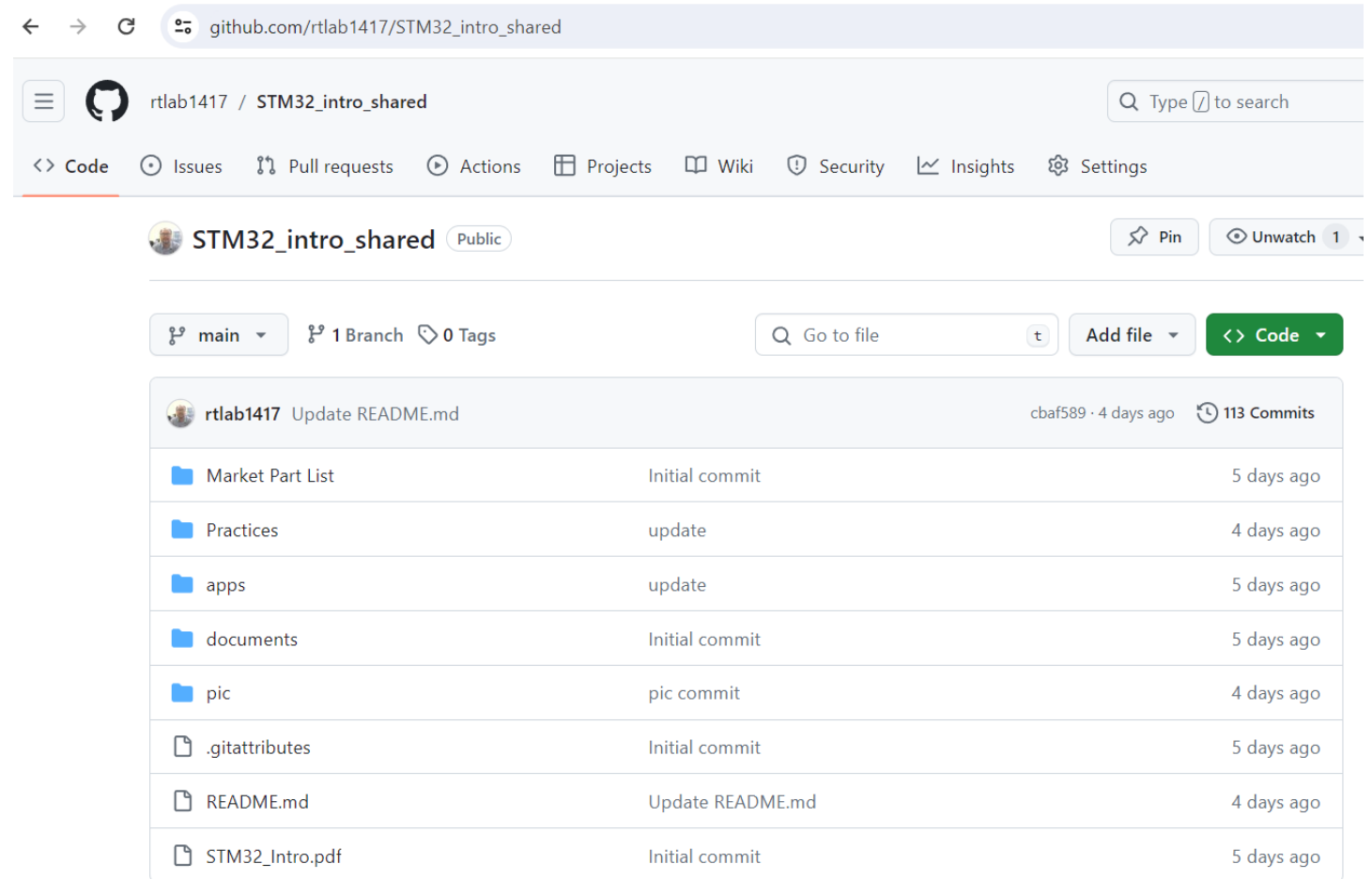
The Training Kit



The Training Kit

Shared information –

https://github.com/rtlab1417/STM32_intro_shared.git



github.com/rtlab1417/STM32_intro_shared

rtlab1417 / STM32_intro_shared

Type to search

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

STM32_intro_shared Public

main 1 Branch 0 Tags

Go to file Add file Code

rtlab1417 Update README.md cbaf589 · 4 days ago 113 Commits

Market Part List	Initial commit	5 days ago
Practices	update	4 days ago
apps	update	5 days ago
documents	Initial commit	5 days ago
pic	pic commit	4 days ago
.gitattributes	Initial commit	5 days ago
README.md	Update README.md	4 days ago
STM32_Intro.pdf	Initial commit	5 days ago

Programming Software

1. **STM32CubeIDE - free - explored in this training.**
 2. Keil MDK - paid service
 3. IAR Embedded Workbench – paid service
 4. Arduino IDE - free
 5. PlatformIO - free
 6. Matlab - Hardware support needed.
 7. Etc.
1. Bare metal programming
 - a. Call the registers directly and manually
 2. Standard peripheral library
 - a. Provided by ST
 3. **Hardware Abstract Layer (HAL)**
 - a. **ST provides HAL for its MCU family.**
 - b. **This is implemented in this short course.**

Programming Software - Language

C Language

- STM32CubeIDE deploy C-language for HAL and coding.
- Having a basic understanding of the C Language can be very useful.
- Some C elements: Variables, Data type, Operators, Loops, Struct, Pointer, Function, type cast, etc.
- *Note: C not equal to C++ however C could be implemented in C++*

Installing STM32CubeIDE

- Preparation – installed before attending the short course
- See module
- Remember where is your workspace.
- For example:
- D:\STM32CubeIDE\workspace\

First Project – looking silly yet very important

The motivation

- To be confirmed that the MCU is communicating with ST-Link
- To be confirmed that STM32CubeIDE is working

First Project

Repeated procedures for all projects

1. Create New Project
2. Target Selection
3. *.ioc
 - Pinout & Configuration
 - Clock Configuration
 - Project Managers
 - Tools
4. Save and auto-generate project – template
5. Edit the code
6. RUN or DEBUG

First Project

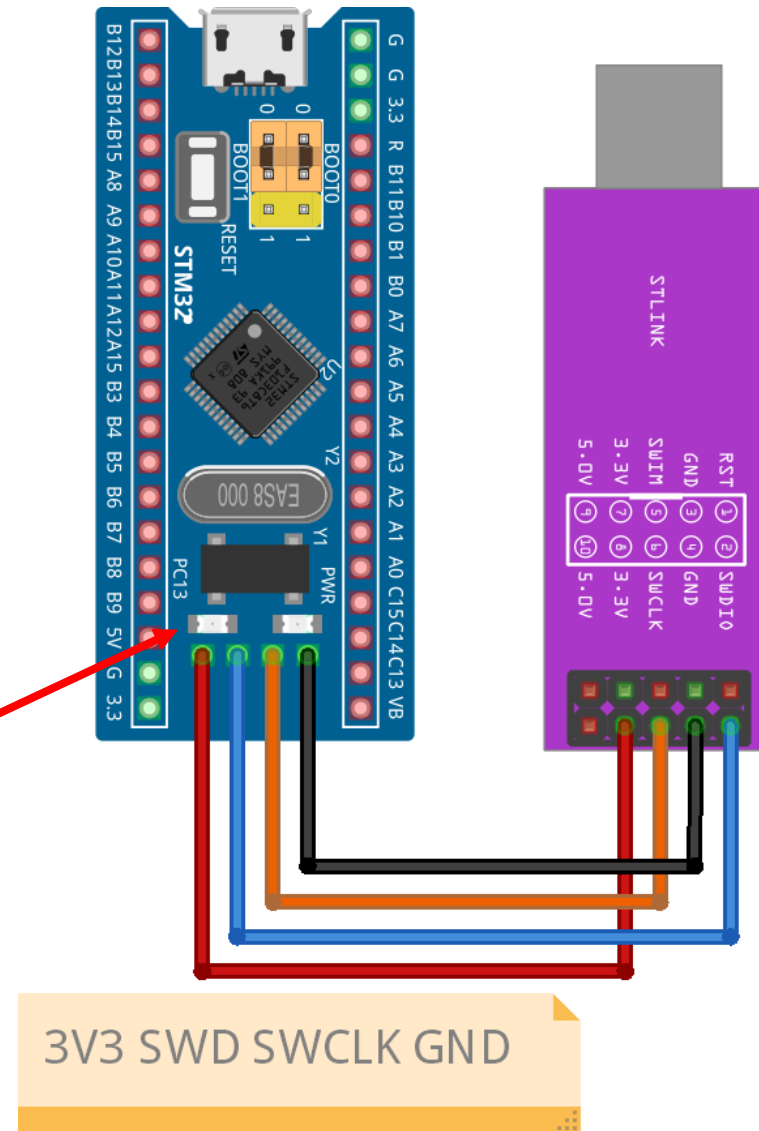
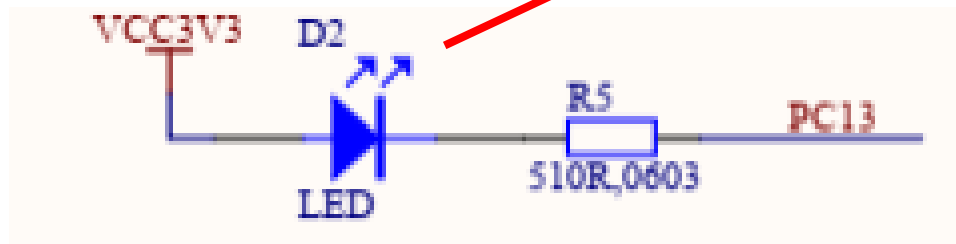
No external connection

No coding needed

Just configure GPIO

Lit up built-in LED – PC13

See module

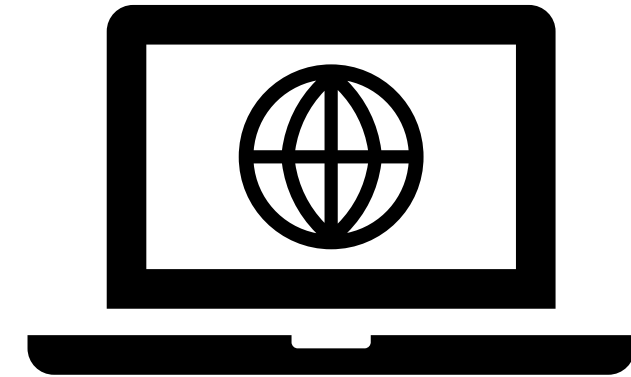


fritzing

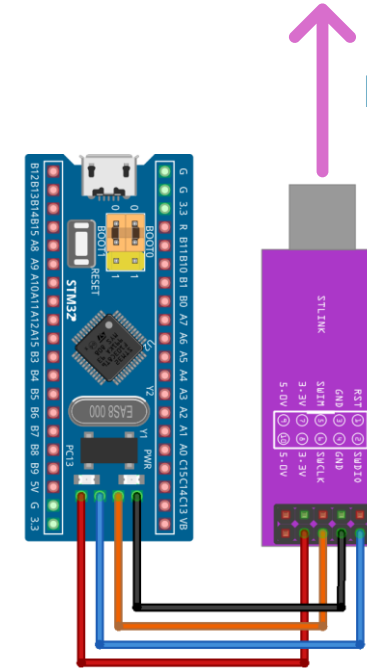
First Project

Reminder:

- **USB type-A**
 - Detected and listed in Device Manager
- **Keep online**
 - Update ST-Link firmware
 - Update app.



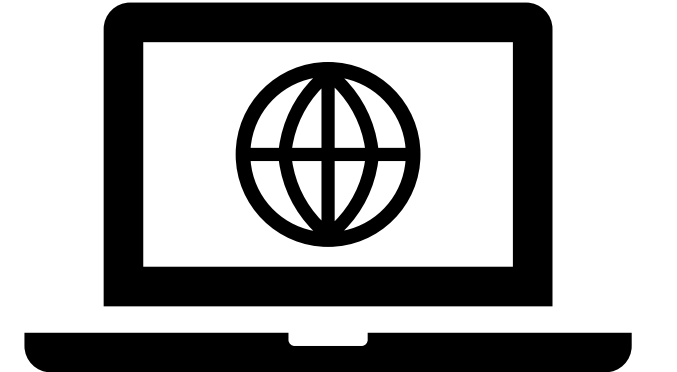
Plug into USB



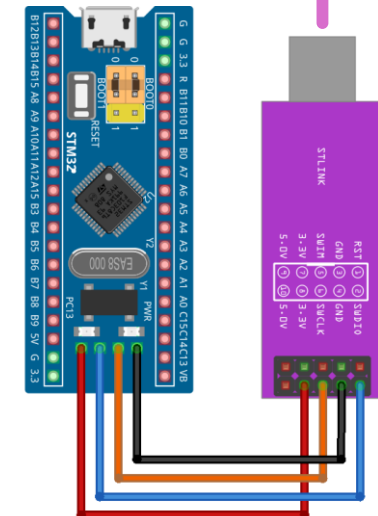
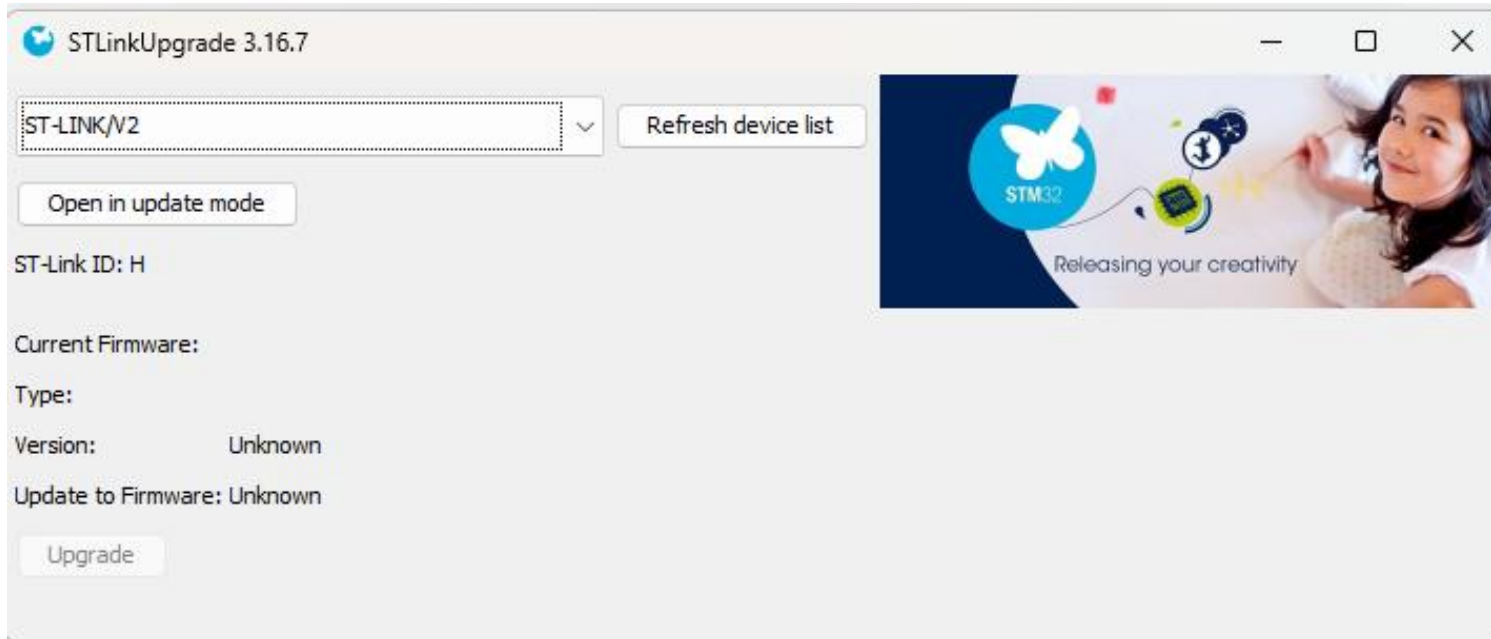
First Project

If you are prompted to update ST-Link firmware:

- Try! Rather intuitive.
- Make sure your laptop is connected to internet!



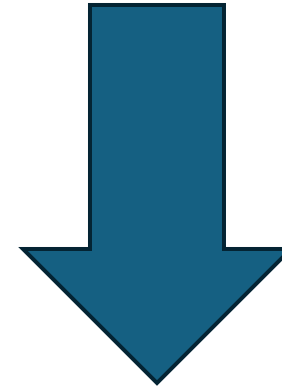
Plug into USB



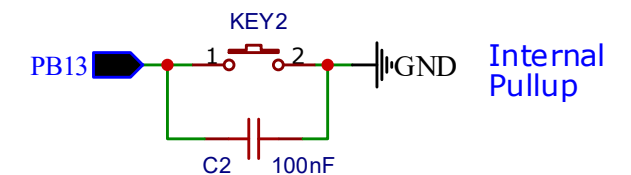
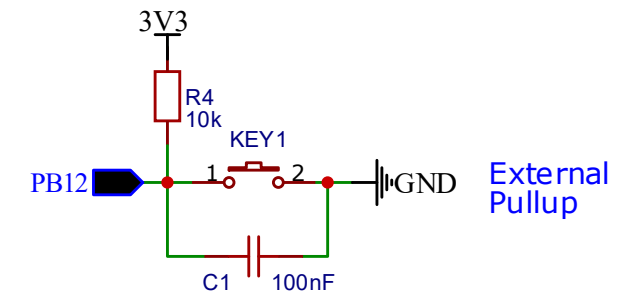
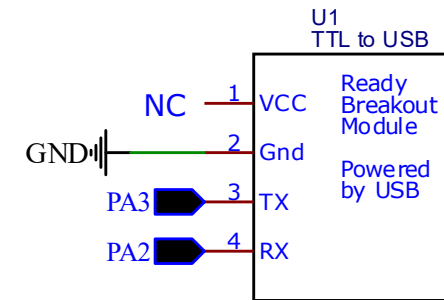
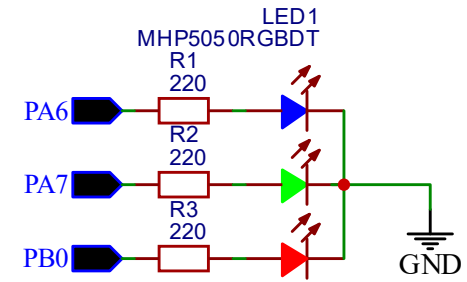
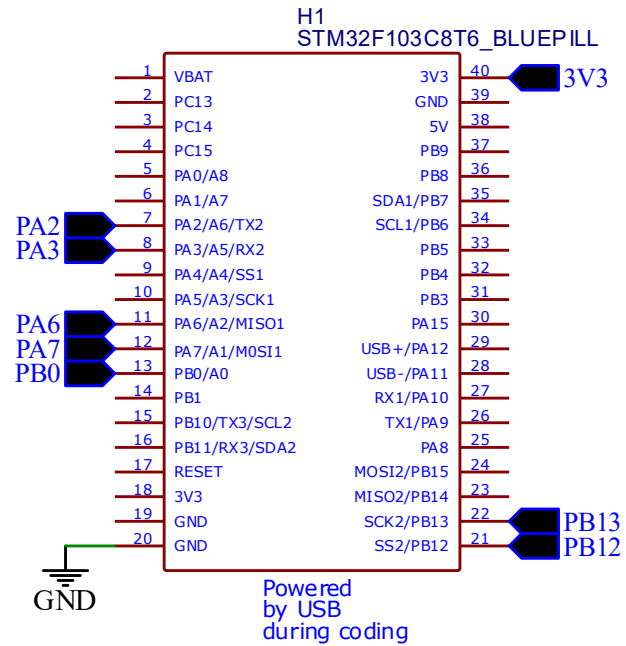
List of Practices

1. GPIO – LED
2. GPIO – LEDs Blink
3. GPIO – LED Buttons
4. GPIO – EXTI
5. UART in Polling Mode
6. UART With Interrupt
7. UART With DMA

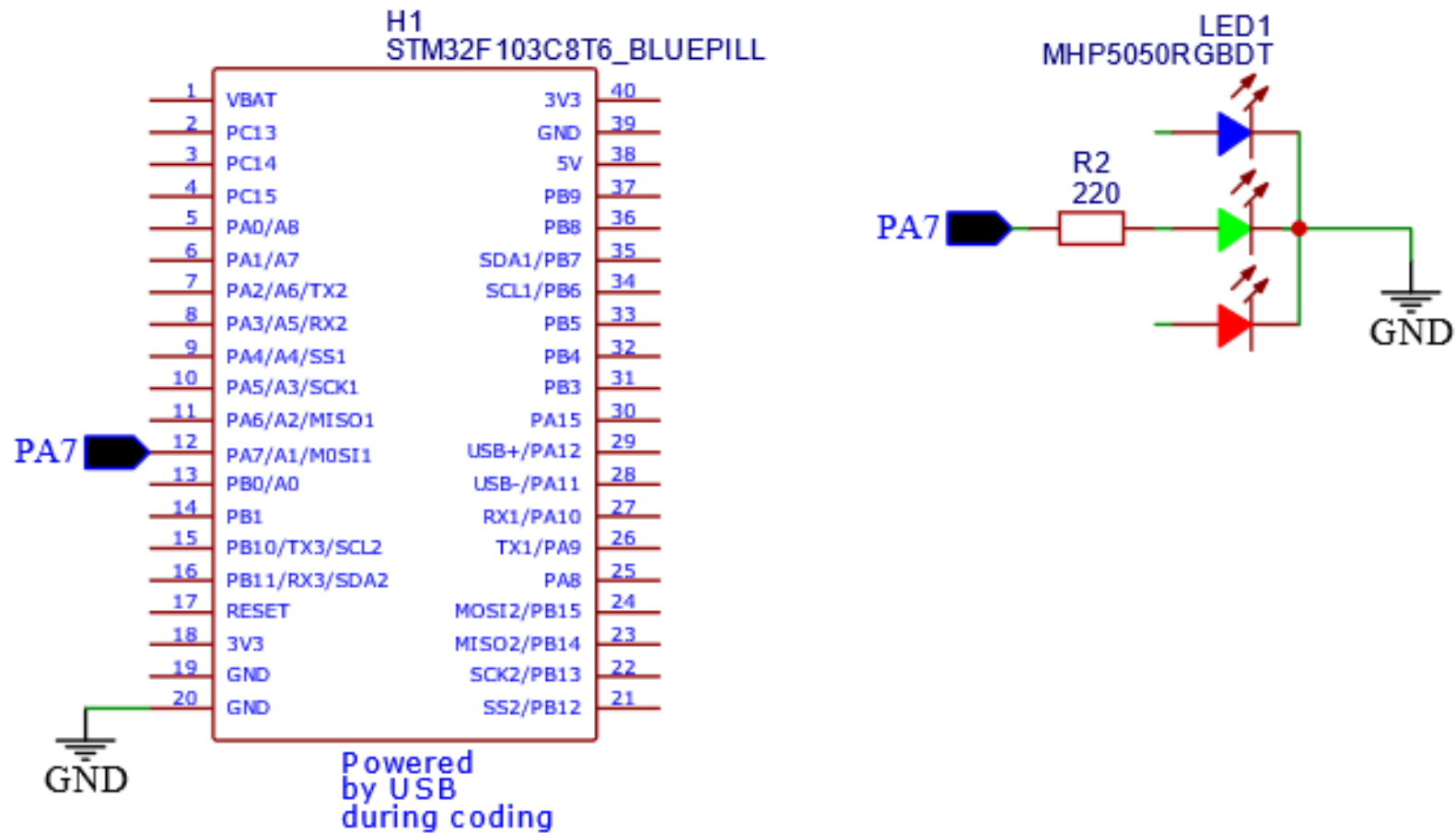
- More to go...



Schematic - Practice 1 to 7



Practice 1 – GPIO -LED



Practice 1 – GPIO -LED

Pin	Mode	User Label
PA7	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_GREEN



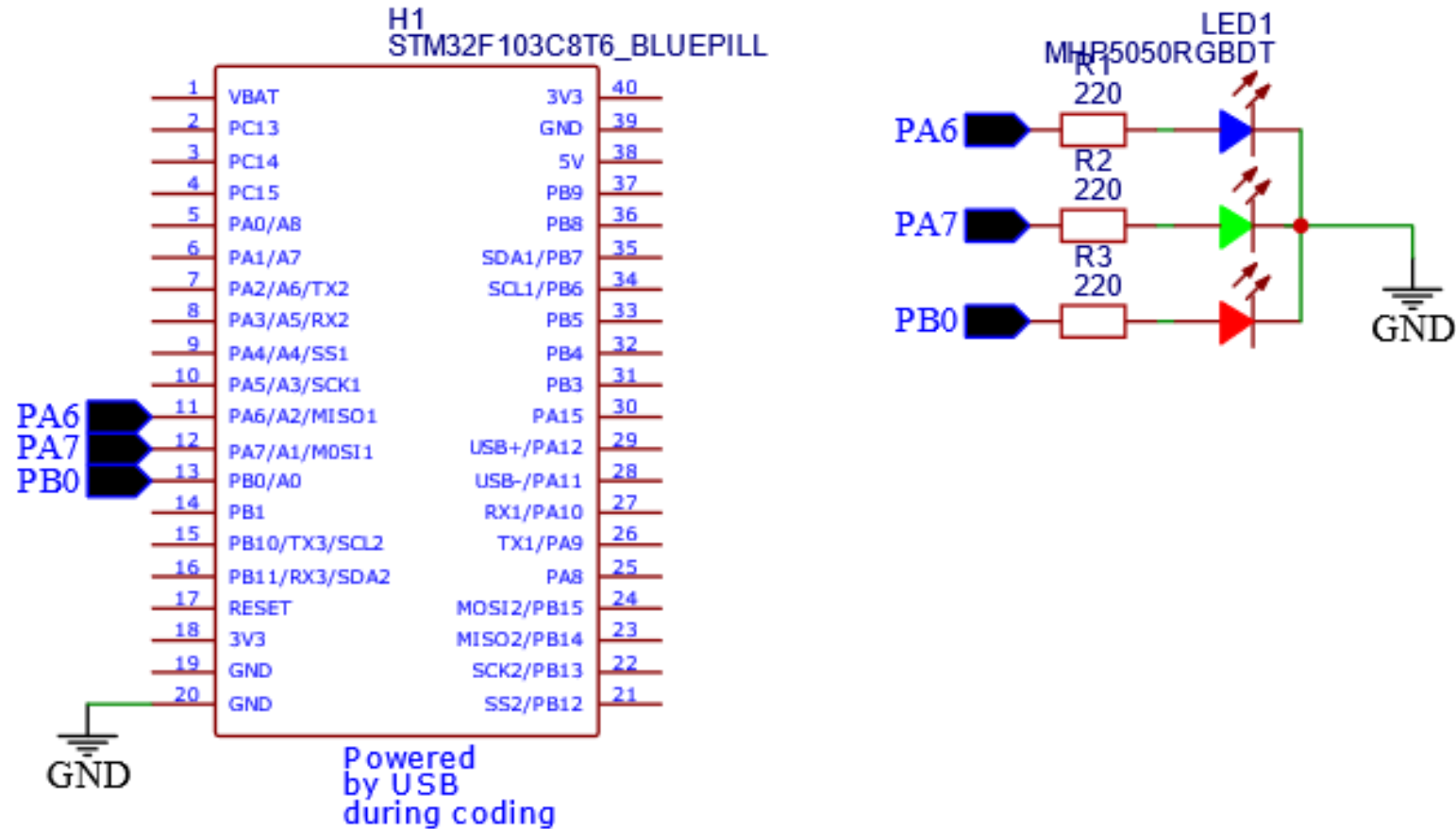
The configuration in xxxx.ioc
Example: 3-led-buttons.ioc

Practice 1 – GPIO -LED

Exploring GPIO

- See module
- Exploring the IDE and file structure
- main.c
- HAL and code accordingly – see module and demonstration

Practice 2 – GPIO –LEDs Blink



Practice 2 – GPIO –LEDs Blink

Pin	Mode	User Label
PA6	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_BLUE
PA7	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_GREEN
PB0	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_RED

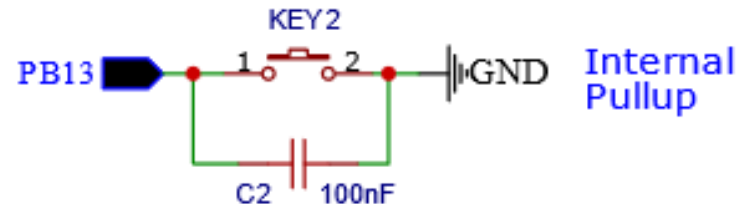
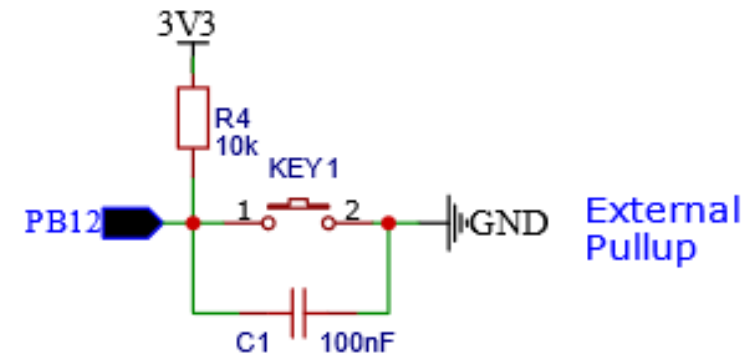
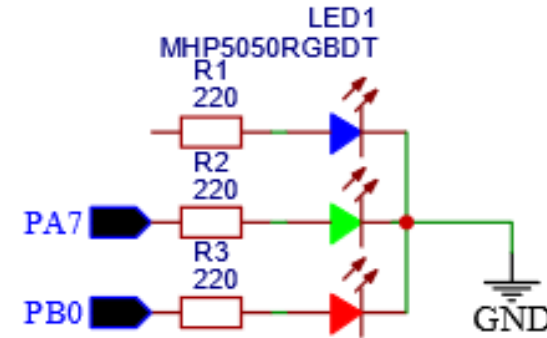
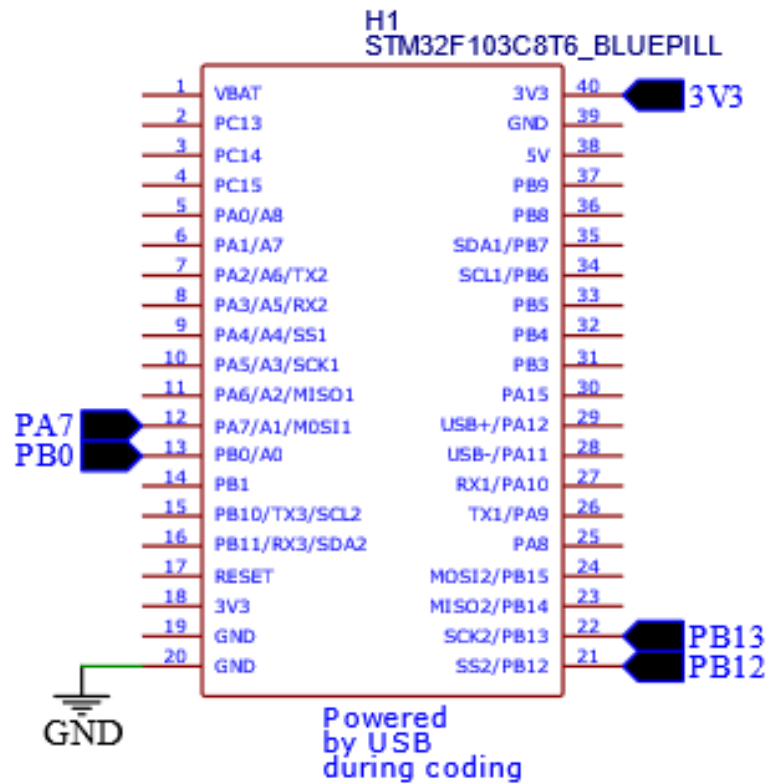
Requirement:

	State0	State1	State2	State3	State4	State5
LED_BLUE	0	0	1	1	1	1
LED_GREEN	0	1	1	1	0	0
LED_RED	1	1	1	1	0	1

Practice 2 – GPIO –LEDs Blink

- See module
- Update main.c
- Have fun.
- Change new delay time.
- **Challenge**
- What if we have a new requirement? Design your own!

Practice 3 – GPIO –LED - Buttons

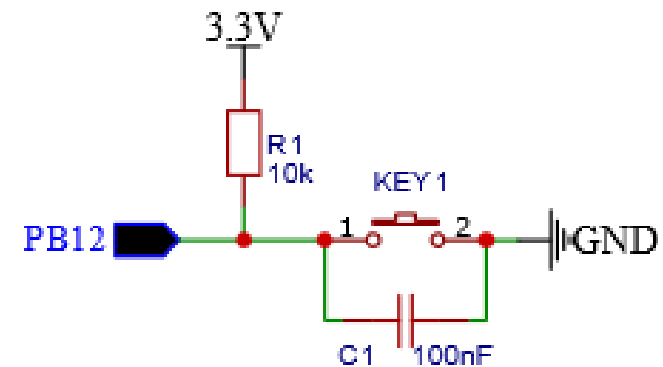
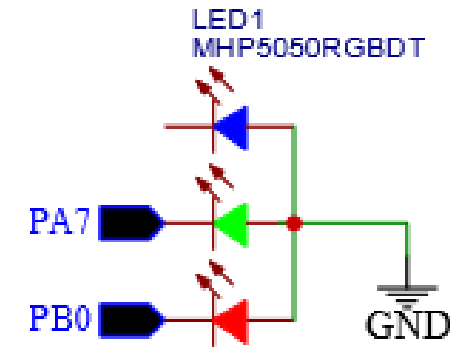
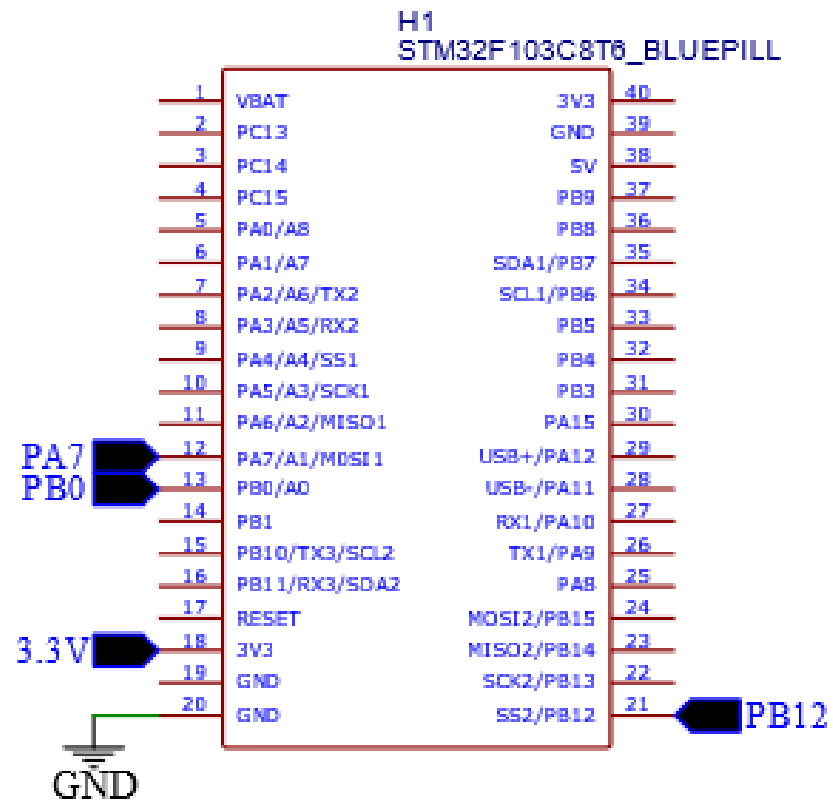


Practice 3 – GPIO –LED - Buttons

Pin	Mode	User Label
PA7	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_GREEN
PB0	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_RED
PB12	GPIO_Input, No pull-up and no pull-down	KEY1
PB13	GPIO_Input, Pull-up	KEY2

We will check out the common effect on mechanical buttons – debouncing!

Practice 3 – GPIO –EXTI

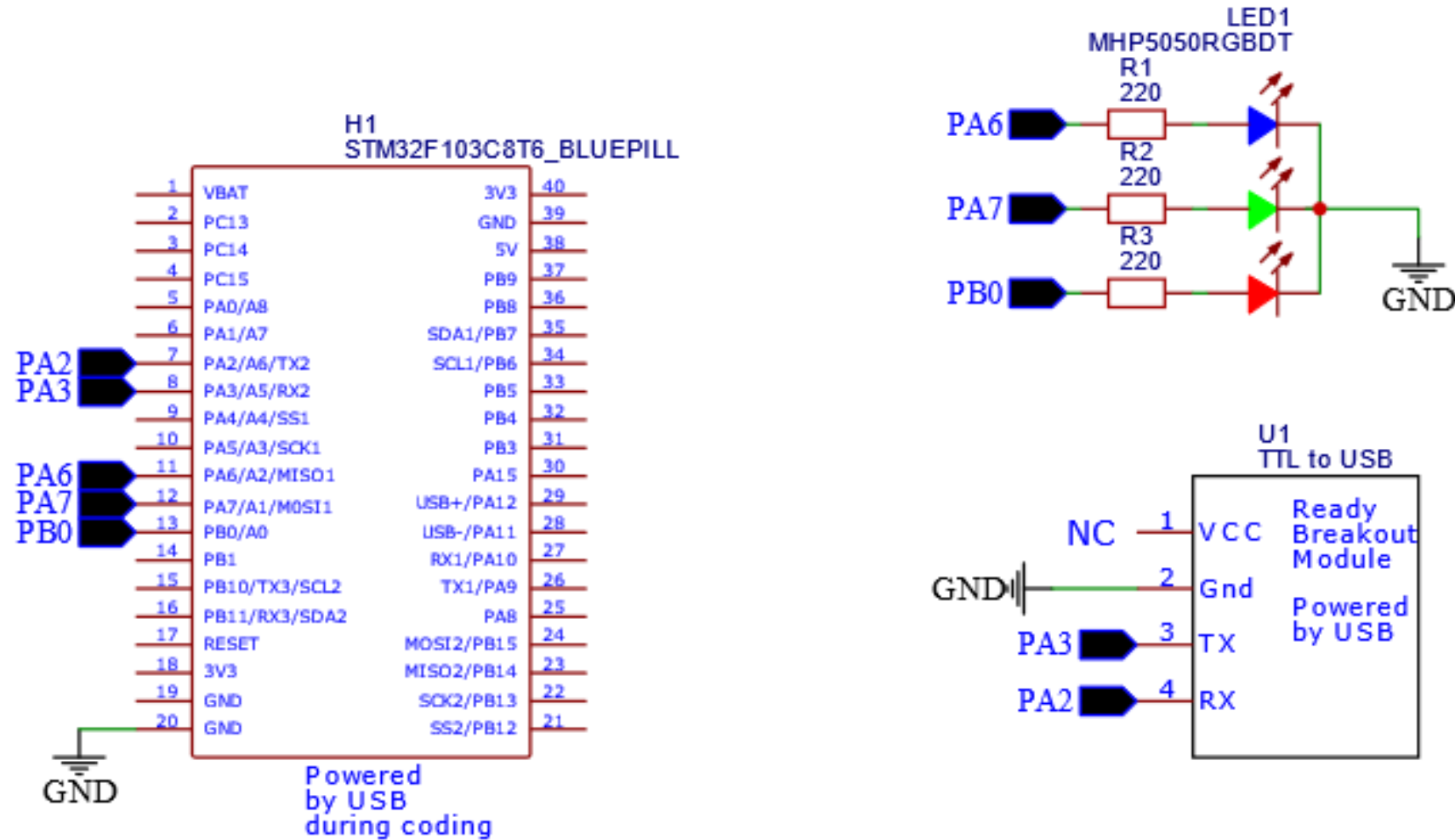


Practice 3 – GPIO –LED - Buttons

Pin	Mode	User Label
PA7	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_GREEN
PB0	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_RED
PB12	GPIO_Input, No pull-up and no pull-down	KEY1

We will explore interrupt!

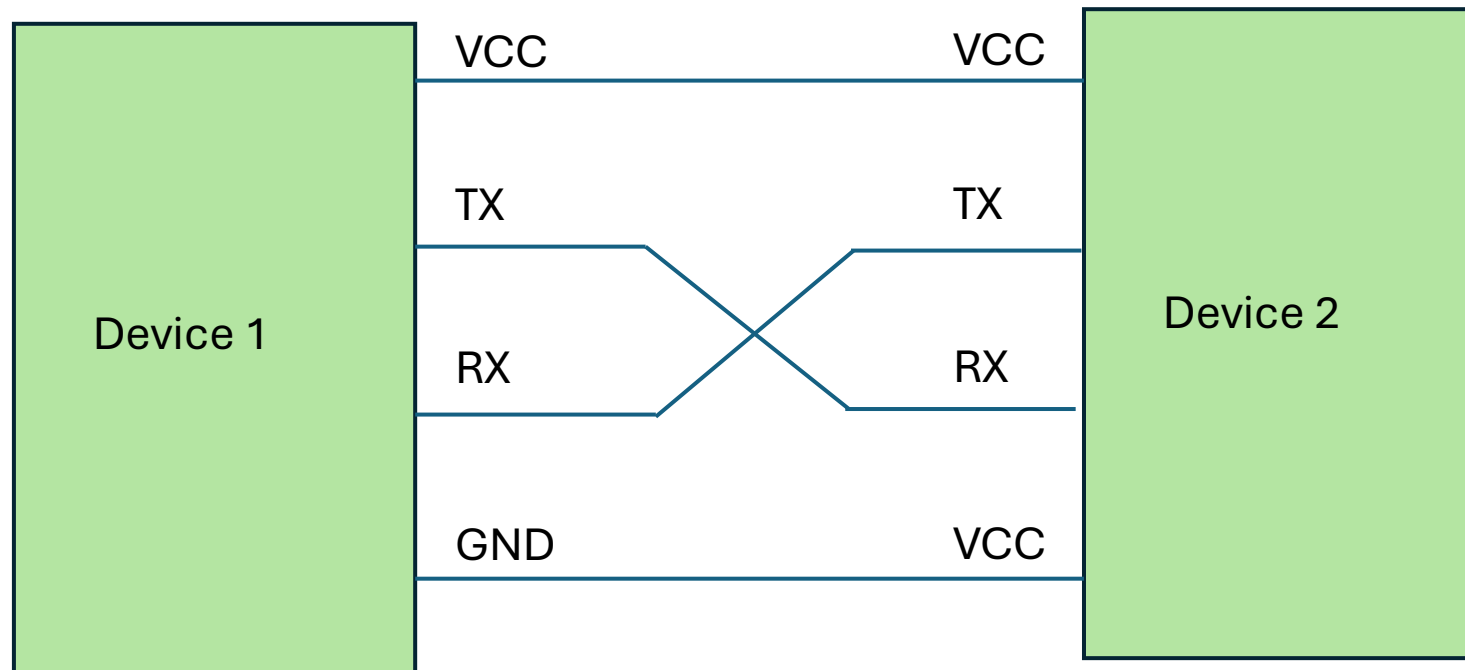
Practice 5 – UART in Polling Mode



Practice 5 – UART in Polling Mode

1:1 device

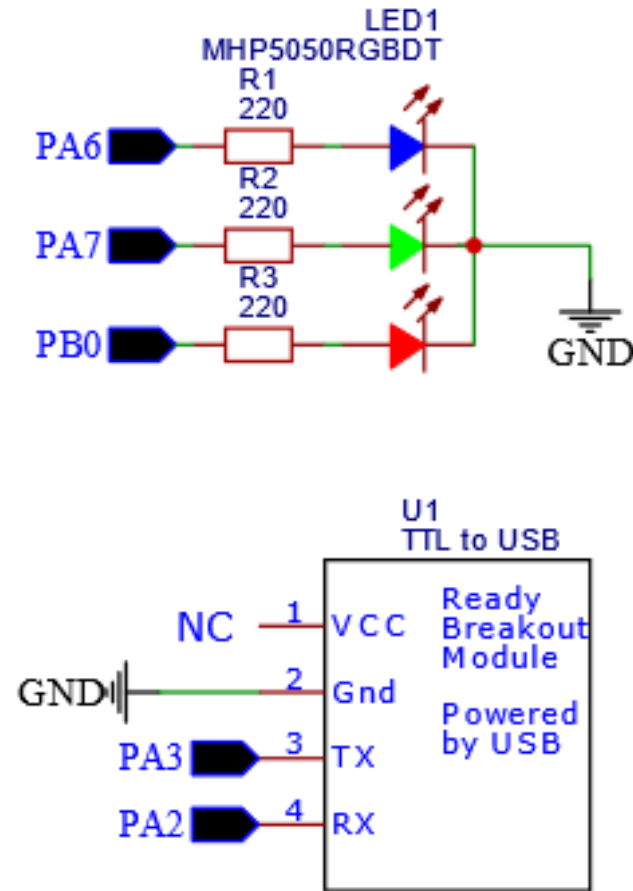
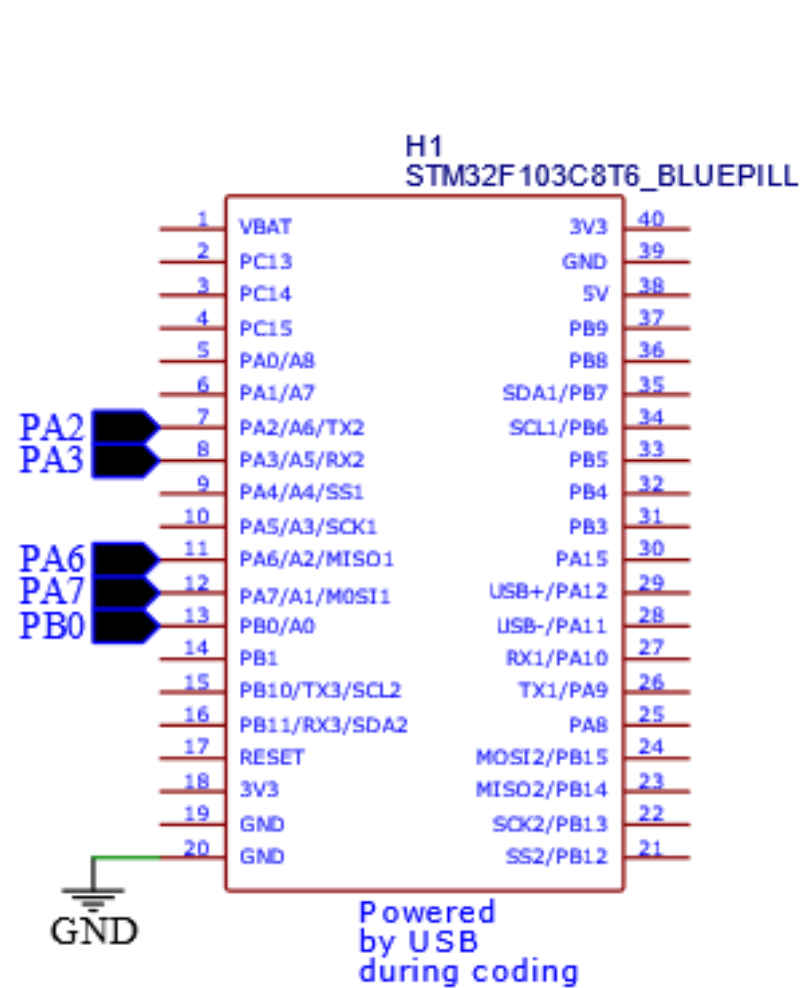
Protocol – Baudrate, Word Length, Stop Bit, Parity



Practice 5 – UART in Polling Mode

In UART polling mode, the microcontroller **constantly** checks (or "polls") the UART hardware for incoming data or to see if it's ready to transmit, rather than relying on interrupts. This method is straightforward to implement but can be inefficient as it requires the **CPU to continuously monitor the UART status**, potentially wasting processing time.

Practice 6 – UART With Interrupt



Same diagram as: UART in Polling Mode

See [Interrupts and events](#)

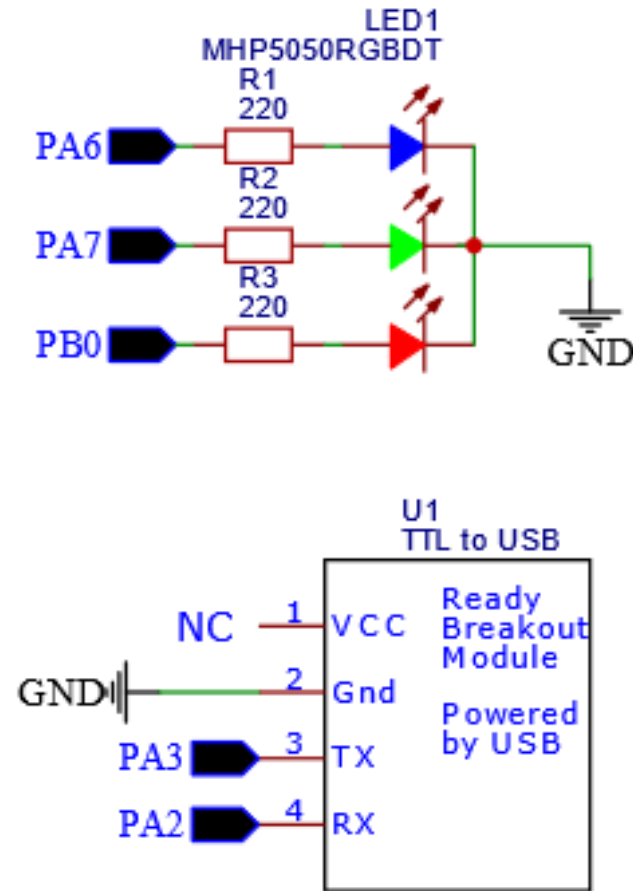
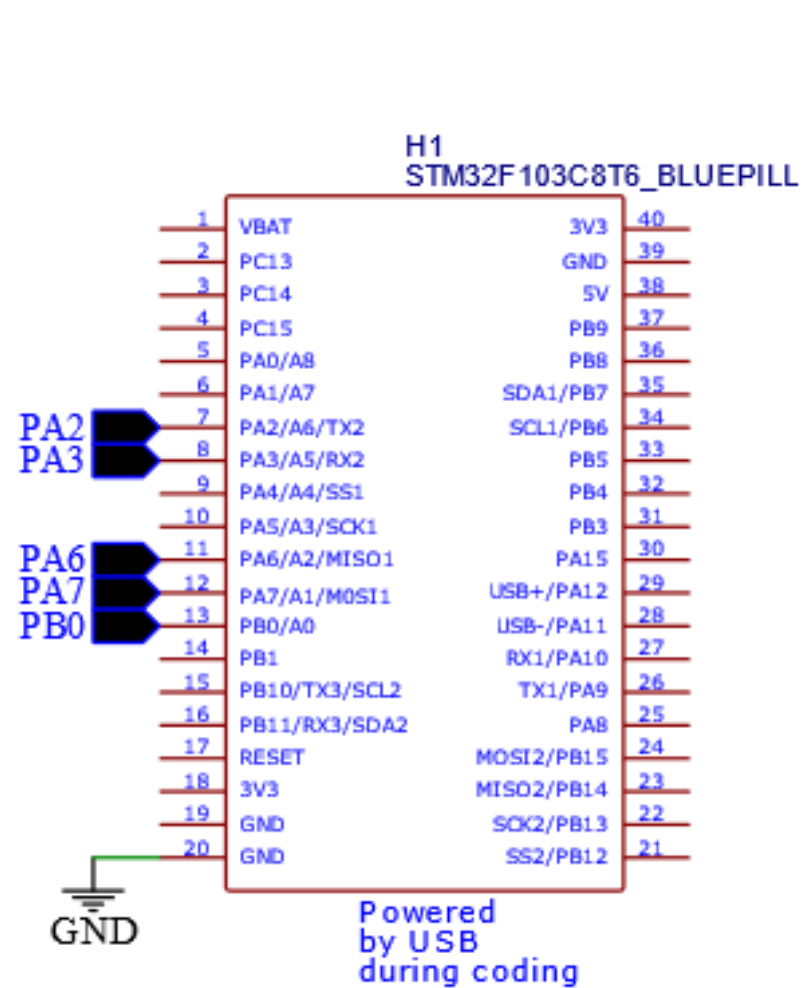
Practice 6 – UART With Interrupt

UART (Universal Asynchronous Receiver/Transmitter) in **interrupt mode** allows a microcontroller to handle incoming serial data **without continuously polling the UART hardware**. Instead, the UART triggers an interrupt when a byte is received, and the microcontroller's interrupt handler processes the data. **This frees up the processor to perform other tasks while waiting for data.**

Same diagram as: UART in Polling Mode

See [Interrupts and events](#)

Practice 7 – UART With DMA



Same diagram as: UART in Polling Mode

See [Interrupts and events](#)

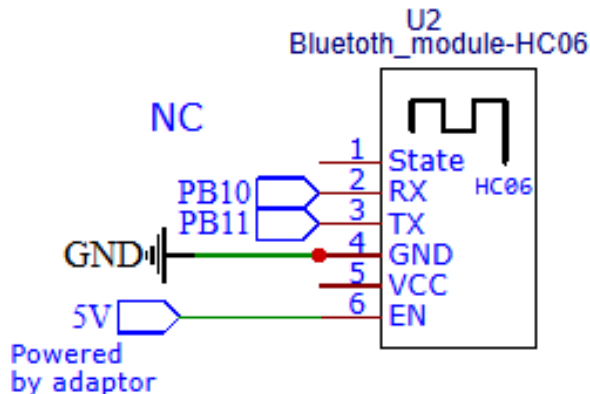
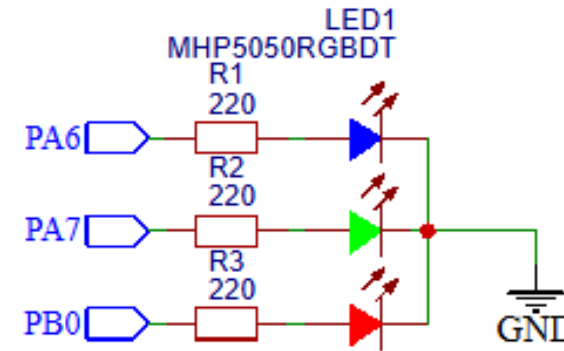
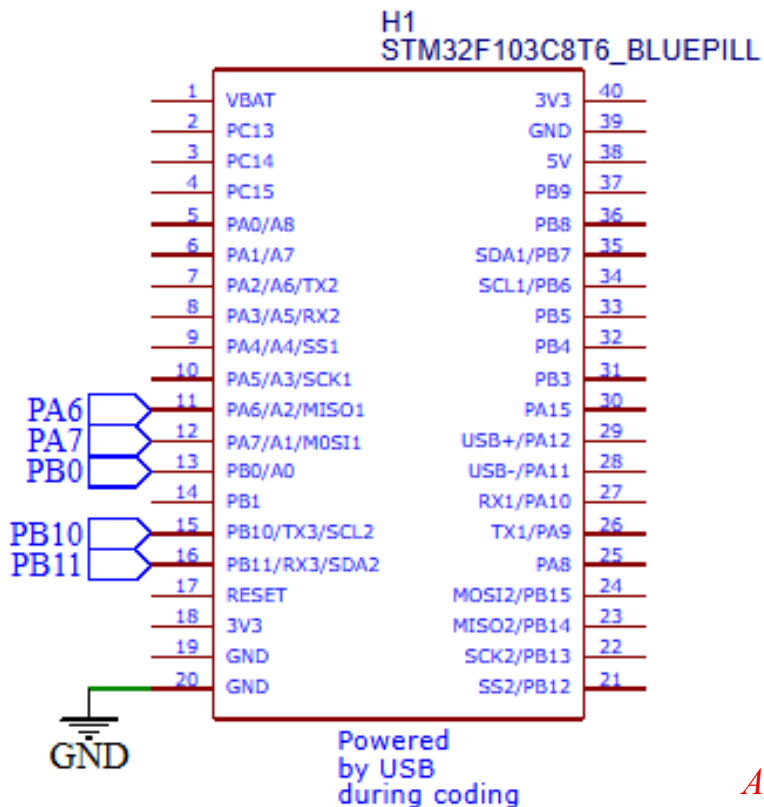
Practice 7 – UART With DMA

1. Compare with UART with Interrupt
2. Discuss
 - UART in Polling Mode
 - UART with Interrupt
 - UART with DMA
3. When to use these?

Same diagram as: UART in Polling Mode

See [Interrupts and events](#)

Practice 8 – UART With BLE_DMA



HC05 or HC06
Slave



Alternatively, tapped +5V from
a TTL-to-USB converter,
with common GND

Use the template in Practice 7 – UART With DMA

See [Interrupts and events](#)

Practice 8 – UART With BLE_DMA

Way 1

On the laptop,

- Enable Bluetooth, HC06/HC05. (scan for BLE devices)
- Open a serial terminal app. Configure port setting (9600, 8N1).
- Examine the LEDs. Use the protocols mentioned above.

Way 2

- Smart phone (Android)
- Play Store | Serial Bluetooth Terminal. Install it.
- HP -> Enable Bluetooth. Search for HC05 and pair it.
- Serial Bluetooth Terminal – connect and send data (HEX)

Use the template in Practice 7 – UART With DMA

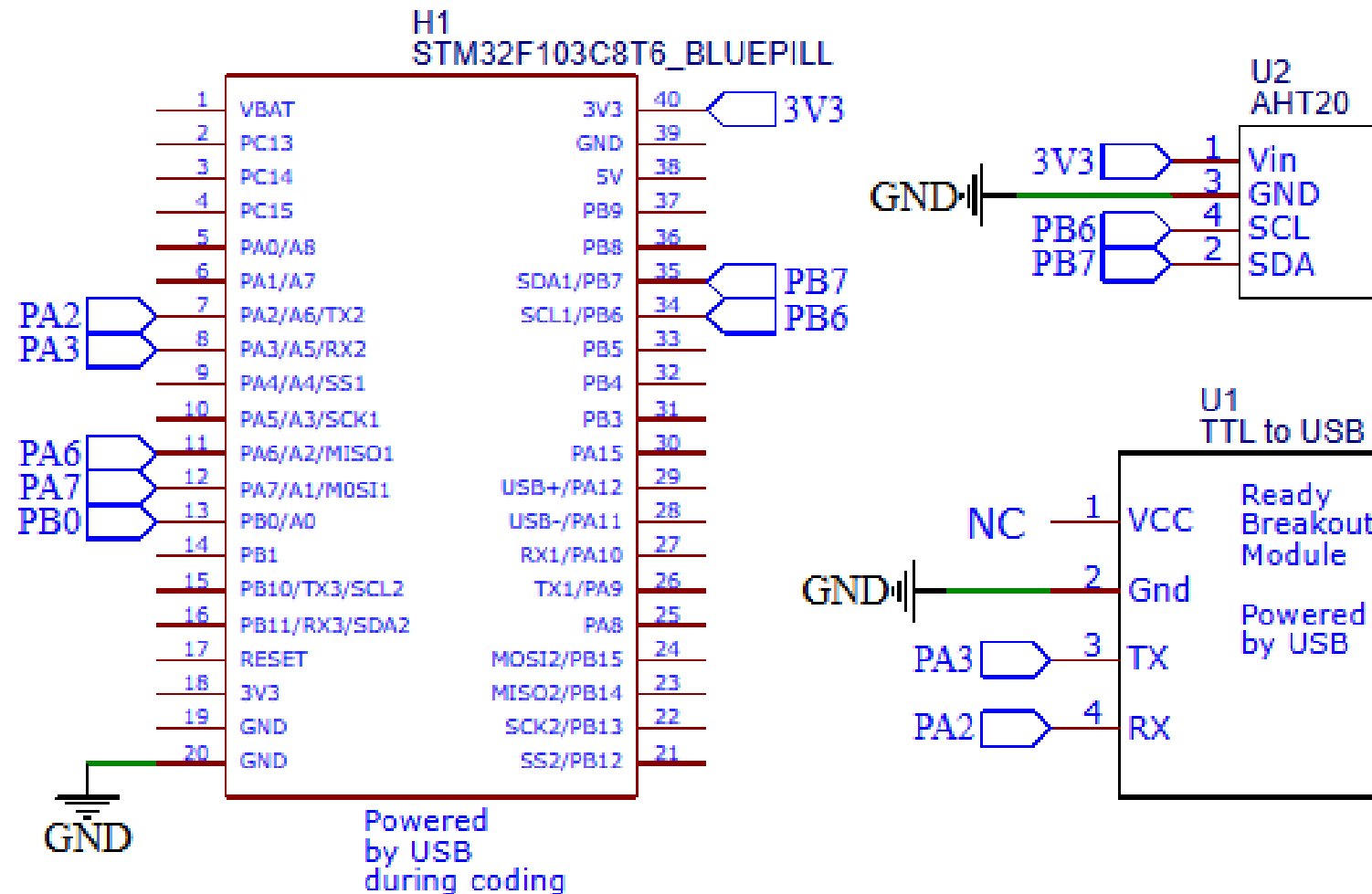
Practice 8 – UART With BLE_DMA

Send data (Bluetooth to STM32 via UART3)

- Red OFF == AA 05 01 00 B0
- Red ON == AA 05 01 FF AF
- Green OFF == AA 05 02 00 B1
- Green ON == AA 05 02 FF B0
- Blue OFF == AA 05 03 00 B2
- Blue ON == AA 05 03 FF B1

Use the template in Practice 7 – UART With DMA

Practice 9 – IIC_AHT20



Practice 9 – IIC_AHT20

- Temperature and Humidity will be posted to USART2.
- Open a serial app and read the data.
- Blow your breath on the sensor and observe the data change.

Note:

I2C1 is in polling mode. No interrupts or DMA are configured.



UART Serial Monitor

Serial Configuration

COM Port: COM8 Refresh

Baud Rate: 115200

Data Bits: 8

Parity: None

Stop Bits: 1

Flow Control: None

Disconnect

Communication Display

☐ Display HEX ☐ Show Timestamp ☒ Autoscroll ☐ DTR ☐ RTS Clear Display Save to File

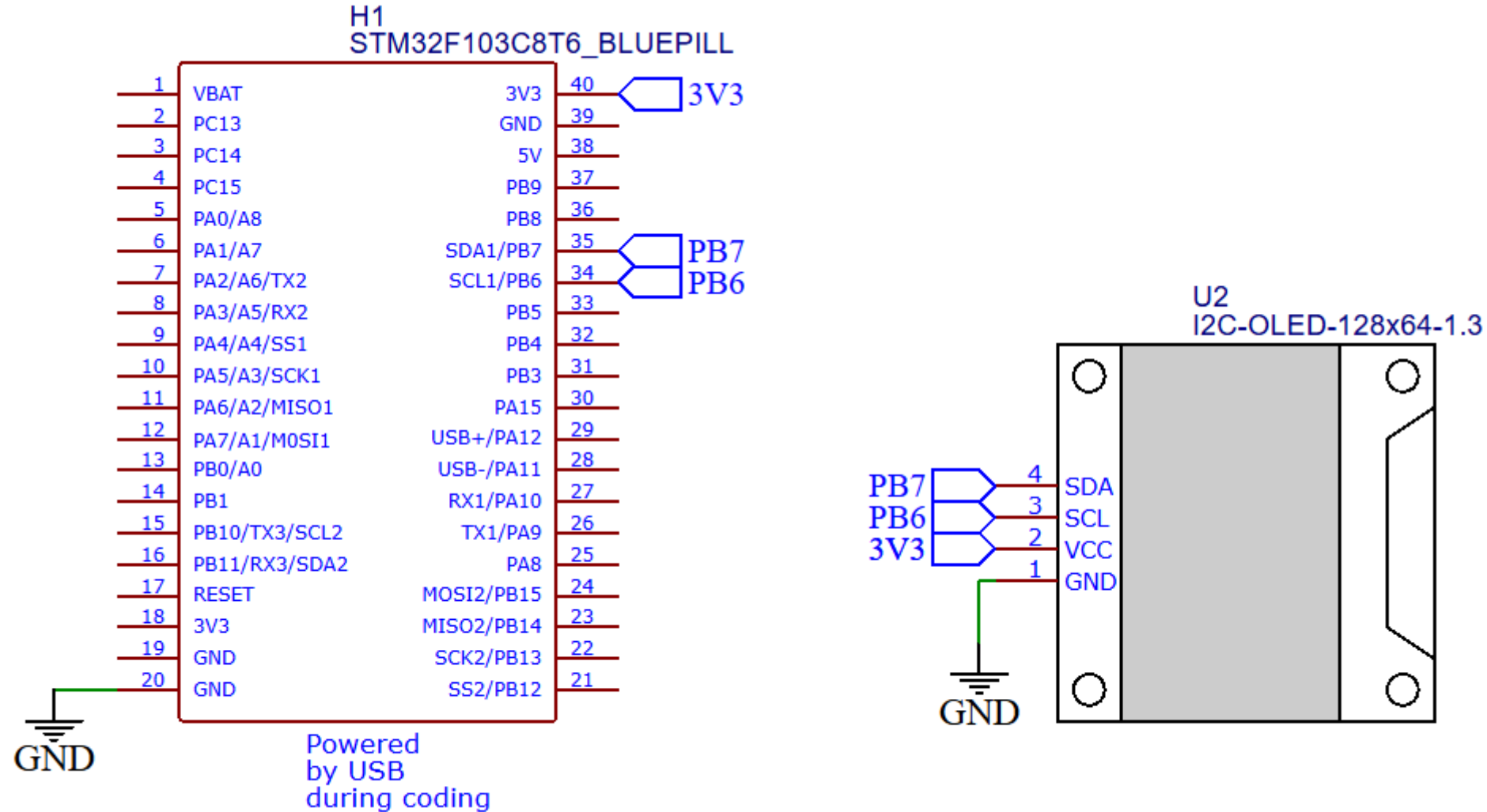
[RX] Temperature: 27.0 C, Humidity: 81.2 %
[RX] Temperature: 27.0 C, Humidity: 81.2 %
[RX] Temperature: 27.0 C, Humidity: 81.2 %
[RX] Temperature: 27.0 C, Humidity: 81.2 %
[RX] Temperature: 27.0 C, Humidity: 81.3 %

Transmit Data

☐ Send as HEX ☐ Auto Transmit (ms): 1000 ^ v

Transmit Clear Input

Practice 9 – IIC AHT20



Practice 9 – IIC_AHT20

- In this practice, we do not want to rewrite the code from scratch. Instead, we look for a pre-existing library created by someone on GitHub. This exercise illustrates the procedure for utilizing a third-party library
- GitHub
 - <https://github.com/afiskon/stm32-ssd1306/tree/master>
- Video
 - <https://youtu.be/z1Px6emHleg?si=2rF47ub8JtQxVgpg>