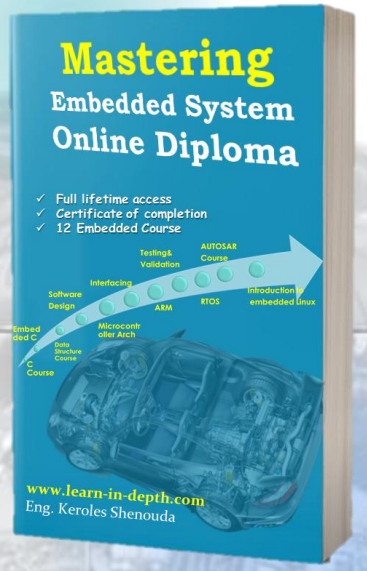
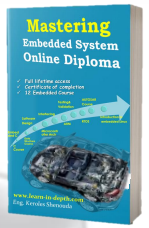
Mastering Embedded System

Online Diploma

- ✓ Full lifetime access
- ✓ Access on Android mobile and PC (Windows)
- ✓ Certificate of completion
- ✓ 12 Embedded Course



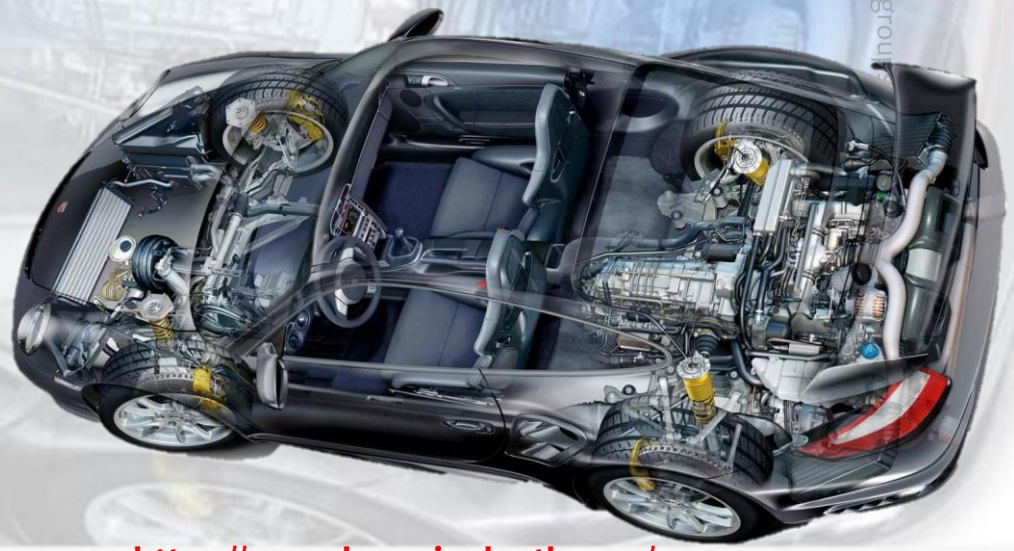


#LEARN_IN DEPTH

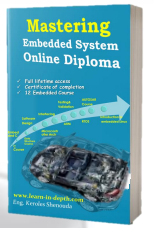
#Be_professional_in_embedded_system

Unit 7 (MCU Essential Peripherals) . lesson 1 GPIO Part1

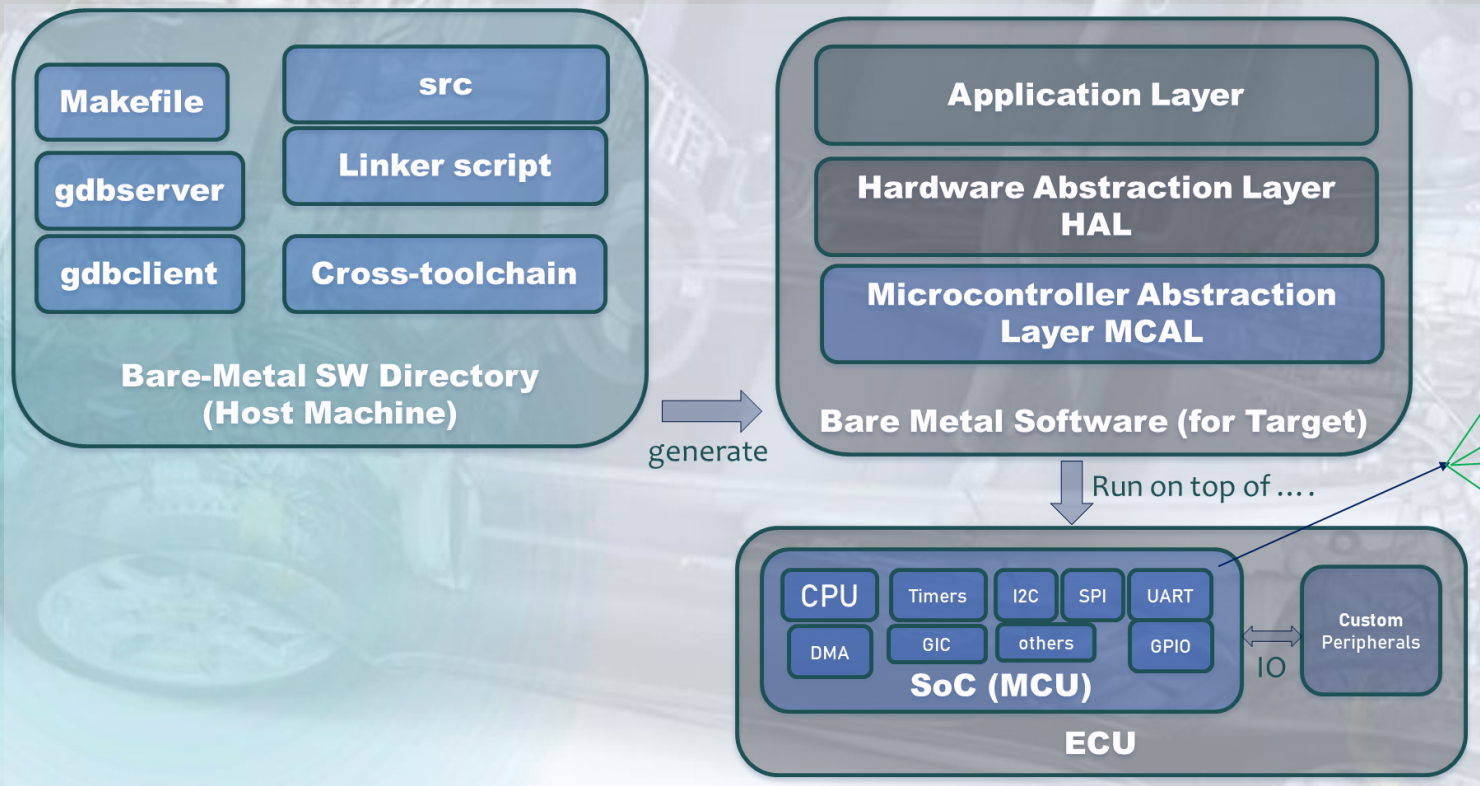
- on-chip/off-chip peripherals
- GPIO controllers
- GPIO Concepts
  - GPIO port consists of a group of GPIO pins in different SoCs
  - GPIO Internal Circuit
  - GPIO Output Modes: Push-Pull
  - GPIO Output Modes: Open-Drain
  - What is the important usage of open-drain outputs ?
  - open-drain Vs push-pull (output mode)
  - GPIO Output Speed: Slew Rate
  - GPIO Input buffer Circuit
  - High impedance (High-Z)/Floating pin
  - GPIO Input Modes: Pull Up and Pull Down
- How To Read GPIO module From TRM
  - Figure the GPIO Ports for SoC and For Board
  - SoC / Board IOs for Different SoCs/Boards
  - Read GPIO functional description from TRM
  - Lab1 GPIO: STM32F103XX
- Assignment

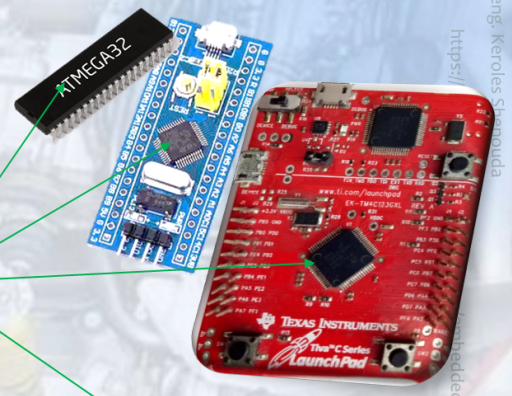


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Big Picture

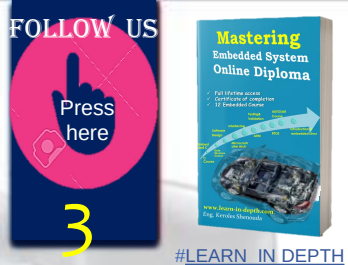




X – SoC

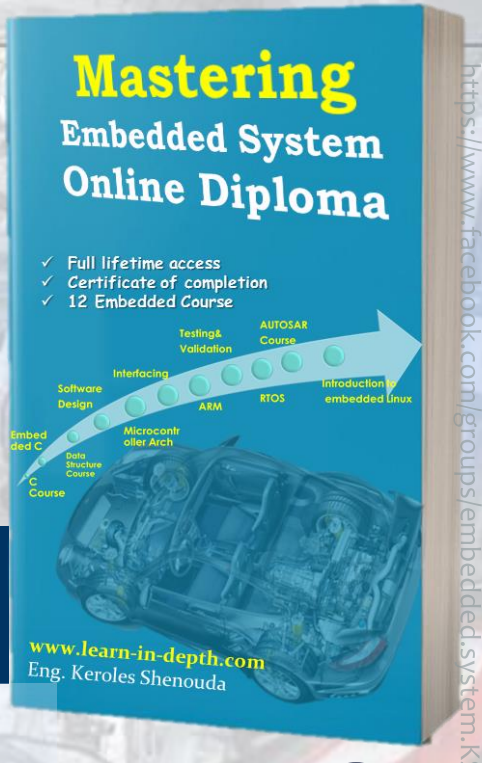
Read Specs and Program X - SoC

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

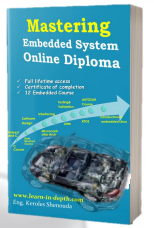
#Be_professional_in_embedded_system



on-chip/off-chip peripherals

LEARN-IN-DEPTH
Be professional in embedded system

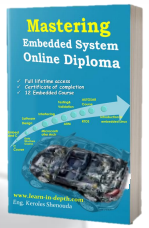
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



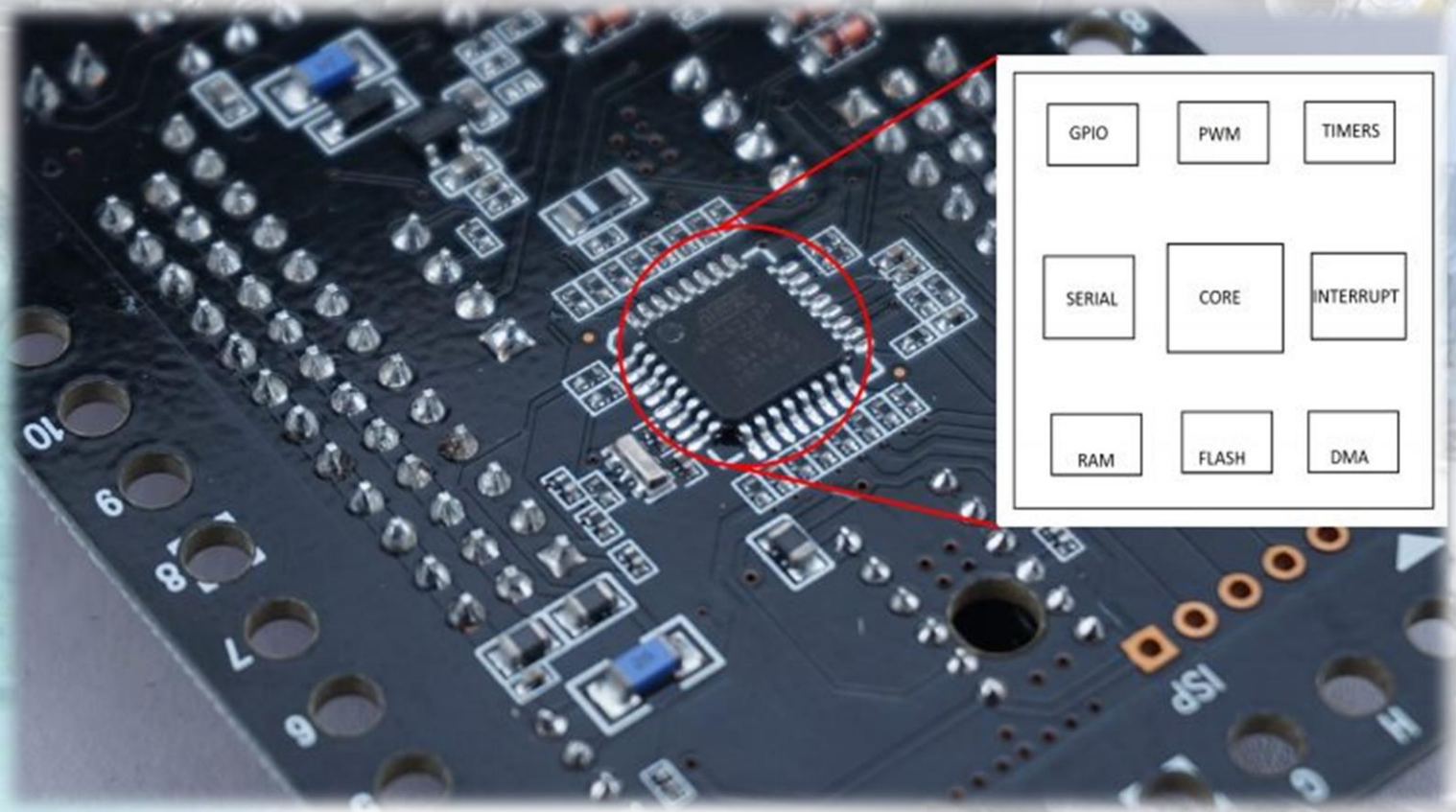
What are peripherals?

- Peripherals are devices that aid the microprocessor to accomplish a given job.
- if they are located inside the SoC (expands to System on Chip, in other words, its just the IC containing the microprocessor) of a micro-controller they are called as on-chip peripherals
- if they are located outside the SoC but on the same PCB they are called off-chip peripherals.
- Common embedded peripherals that most systems need include
  - GPIO controllers
  - Timers
  - PWM controllers
  - DACs
  - ADCs
  - Serial Communication Controllers for UART, SPI, I2C, and Ethernet
  - Memory
  - Interrupt controllers and
  - Direct Memory Access controllers (DMA)

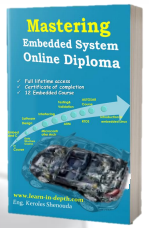
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



on-chip peripherals

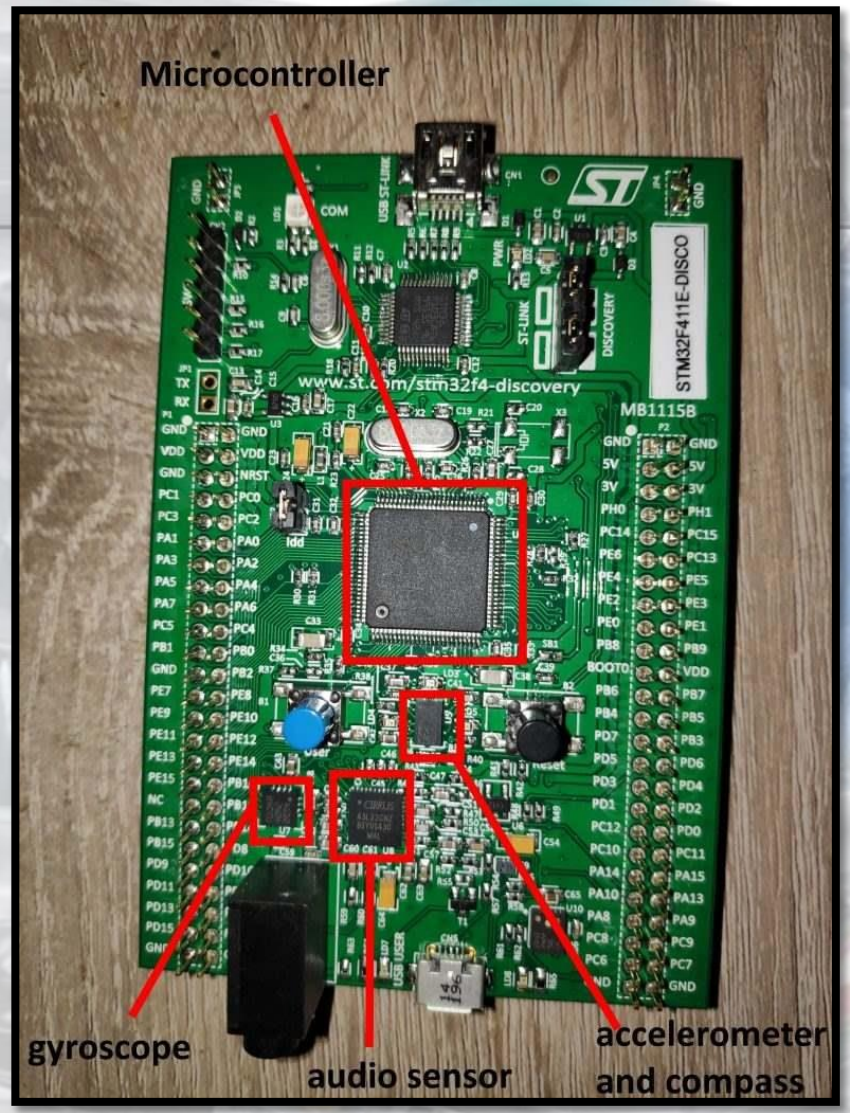


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

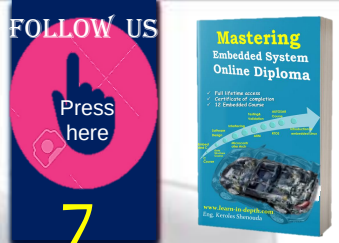


off-chip peripherals

- the discovery development board from STM microelectronics with its STM32 microcontroller and some external peripherals like an audio sensor, gyroscope sensor, compass, and accelerometer sensors.



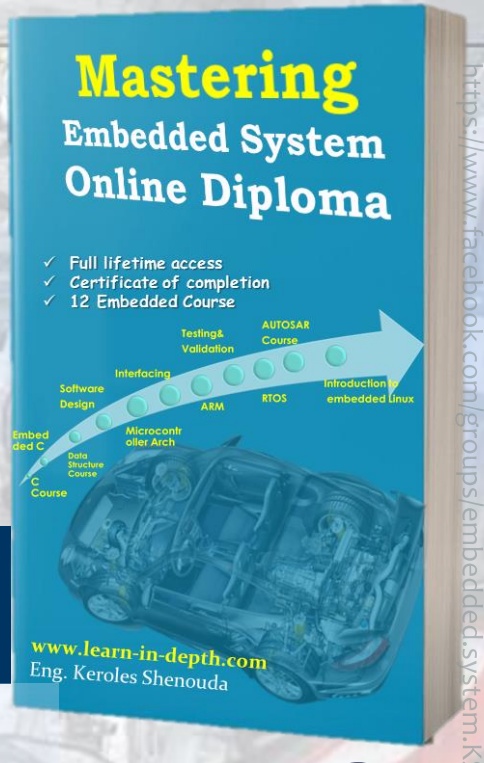
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

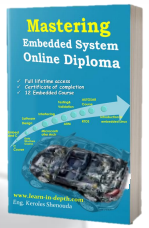
#Be_professional_in_embedded_system

GPIO controllers



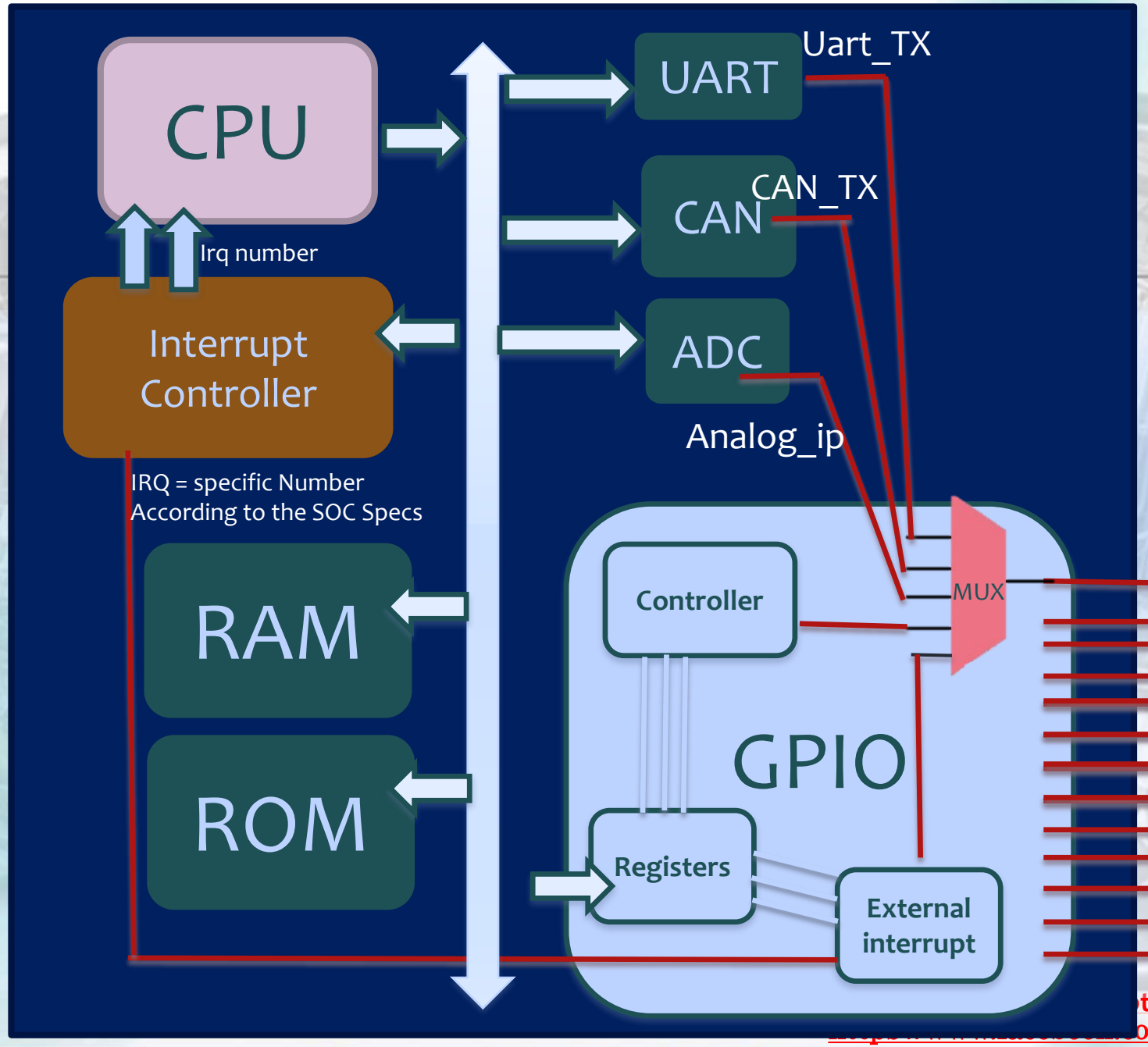
LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

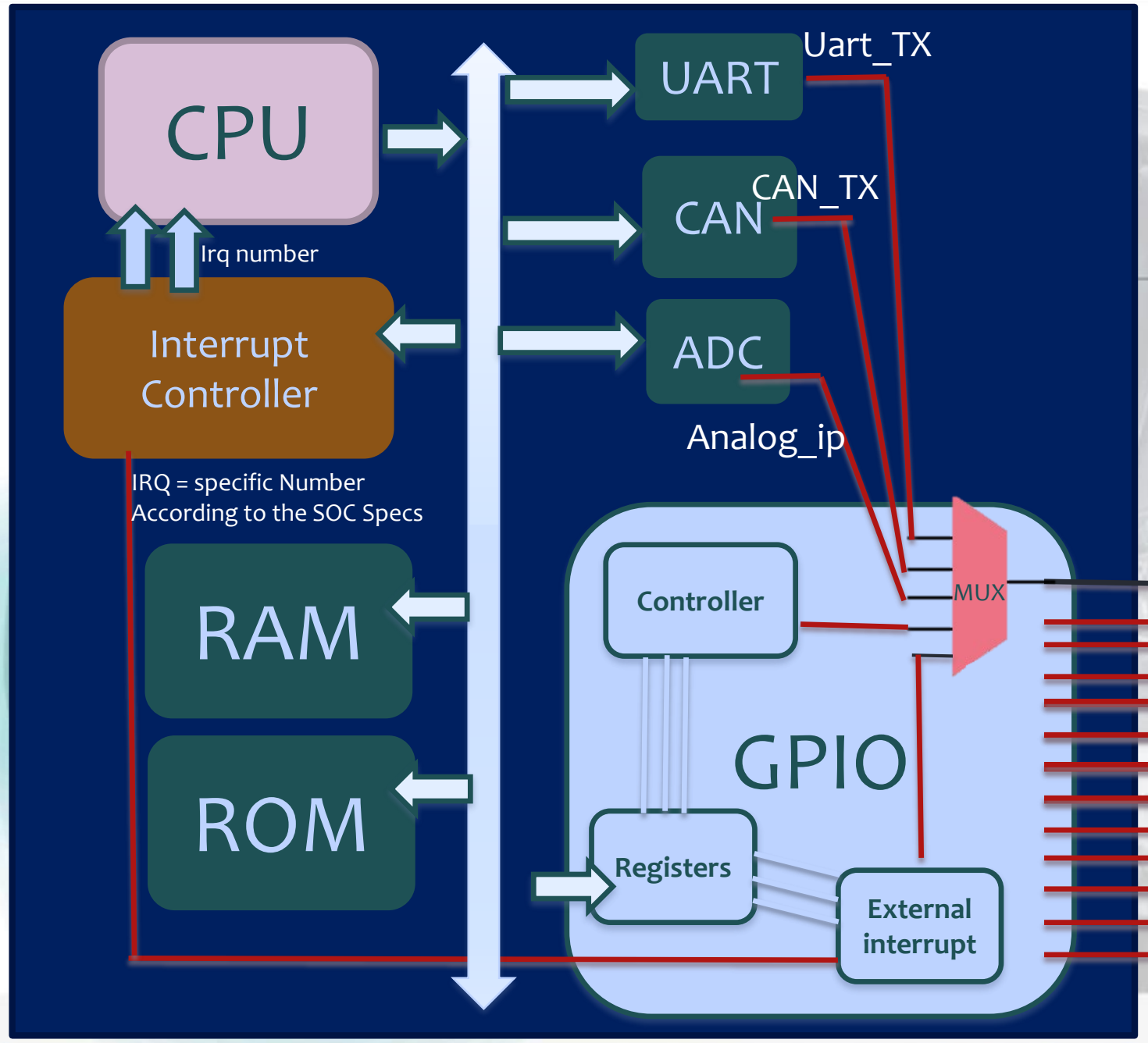


GPIO/DIO/IO controllers

- GPIO stands for **General Purpose Input and Output**. These refer to the pins that are coming out of the microcontroller SoC.
- The basic functionality of a GPIO controller includes
  - setting the GPIO pins as either input or output,
  - reading the voltage level, if they are set as input
  - writing the voltage to 0 or 1 if they are set as output.
  - can be routed to other peripherals such as ADCs and DACs to serve some alternate functions as needed by the peripherals.
- Generally, a single one of these GPIO controllers are able to control 8 IO pins and a microcontroller can have more than one of these. These units are usually referred to as Ports (Port A is controlled by GPIO controller A, port B is controlled by GPIO controller B,…)



th.com/
om/groups/embedded.system.KS/



Each PIN

Can be configured as

Alternative Signal

Like in this example

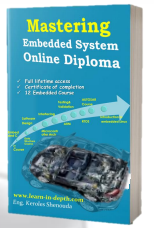
UART_TX

CAN_TX

Anlog_ip

GPIO Signal

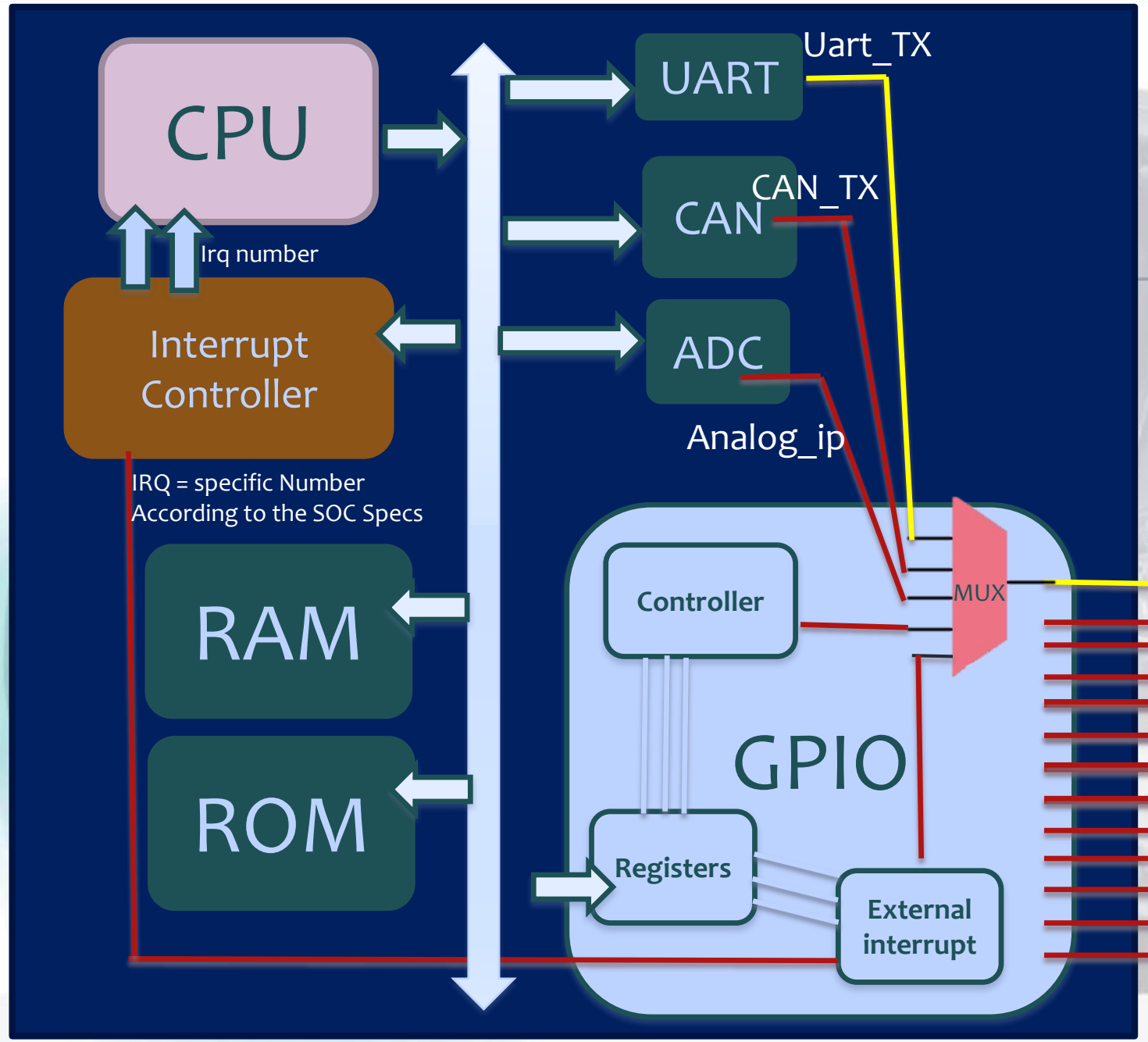
PIN



#LEARN_IN DEPTH

#Be_professional_in embedded_system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Each PIN

Can be configured as

Alternative Signal

Like in this example

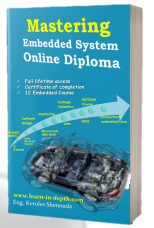
UART_TX
CAN_TX
Anlog_ip

PIN = UART_TX

GPIO Signal

You Can Configure PIN to be alternative Signal Then Write on the MUX register to choose the UART_TX

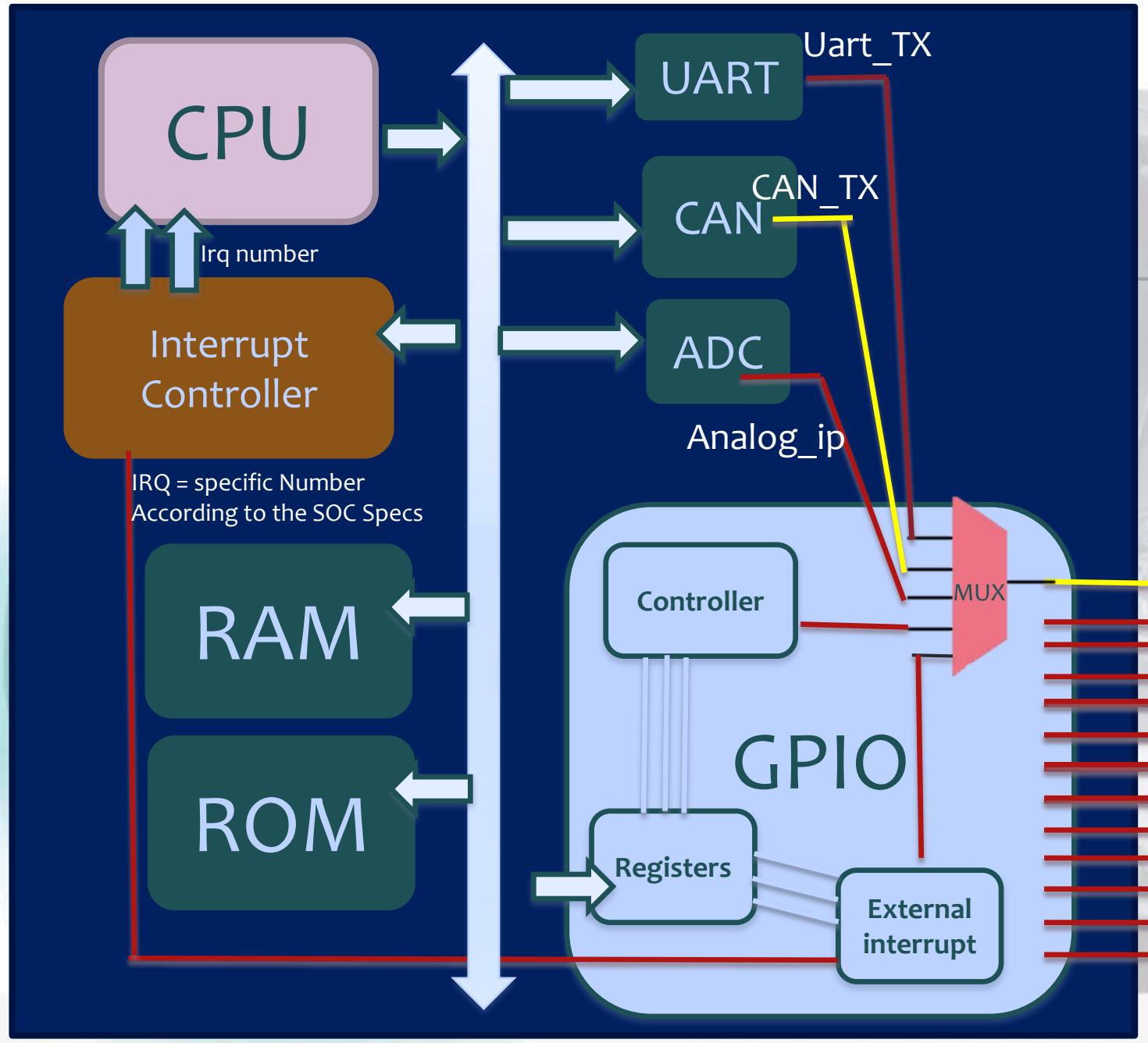
FOLLOW US
Press here
11



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Each PIN

Can be configured as

Alternative Signal

Like in this example

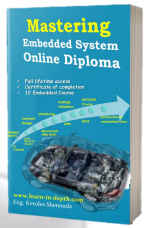
UART_TX
CAN_TX
Anlog_ip

PIN = CAN_TX

GPIO Signal

You Can Configure PIN to be alternative Signal Then Write on the MUX register to choose the CAN_TX

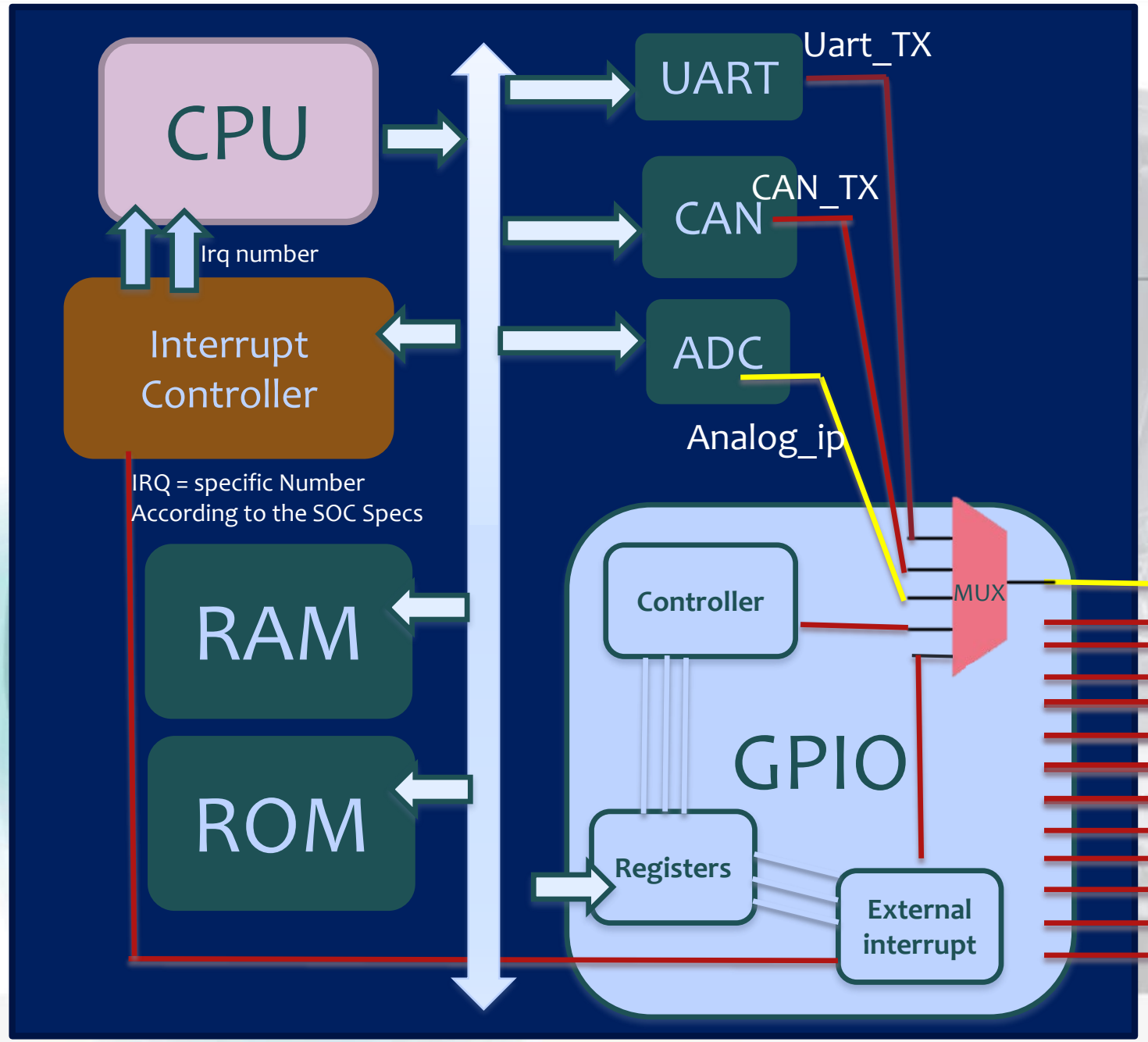
FOLLOW US
Press here
12



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Each PIN

Can be configured as

Alternative Signal

Like in this example

UART_TX

CAN_TX

Anlog_ip

GPIO Signal

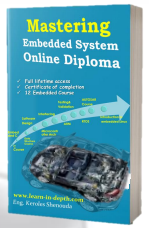
PIN = Anlog_ip

You Can Configure PIN to be alternative Signal Then Write on the MUX register to choose the Anlog_ip

FOLLOW US

Press here

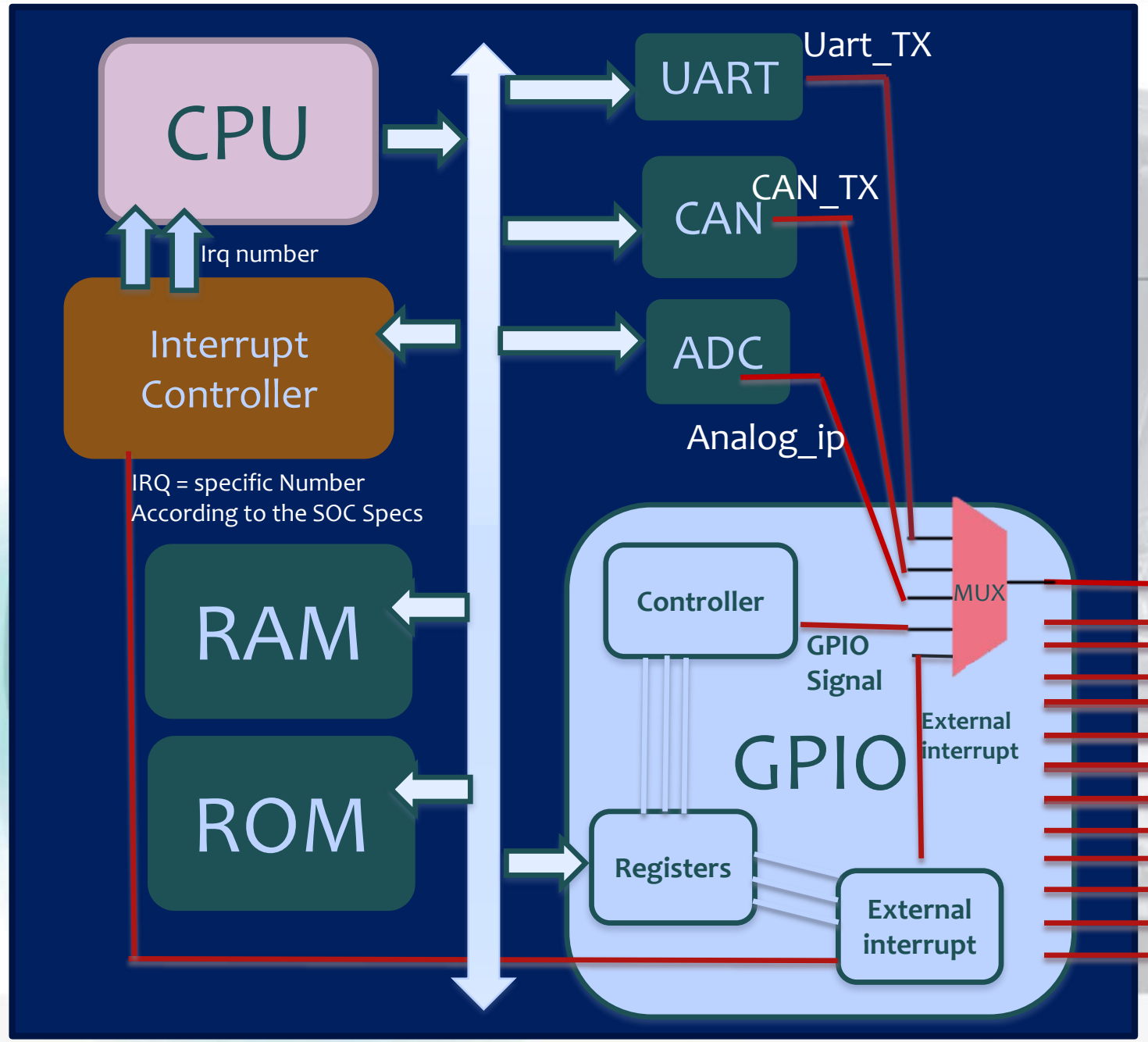
13



#LEARN_IN DEPTH

#Be_professional_in embedded_system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Each PIN

Can be configured as

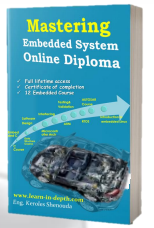
- Alternative Signal
- GPIO Signal
  - Output
  - Input
    - External Interrupt
    - Normal Input

PIN

FOLLOW US

Press here

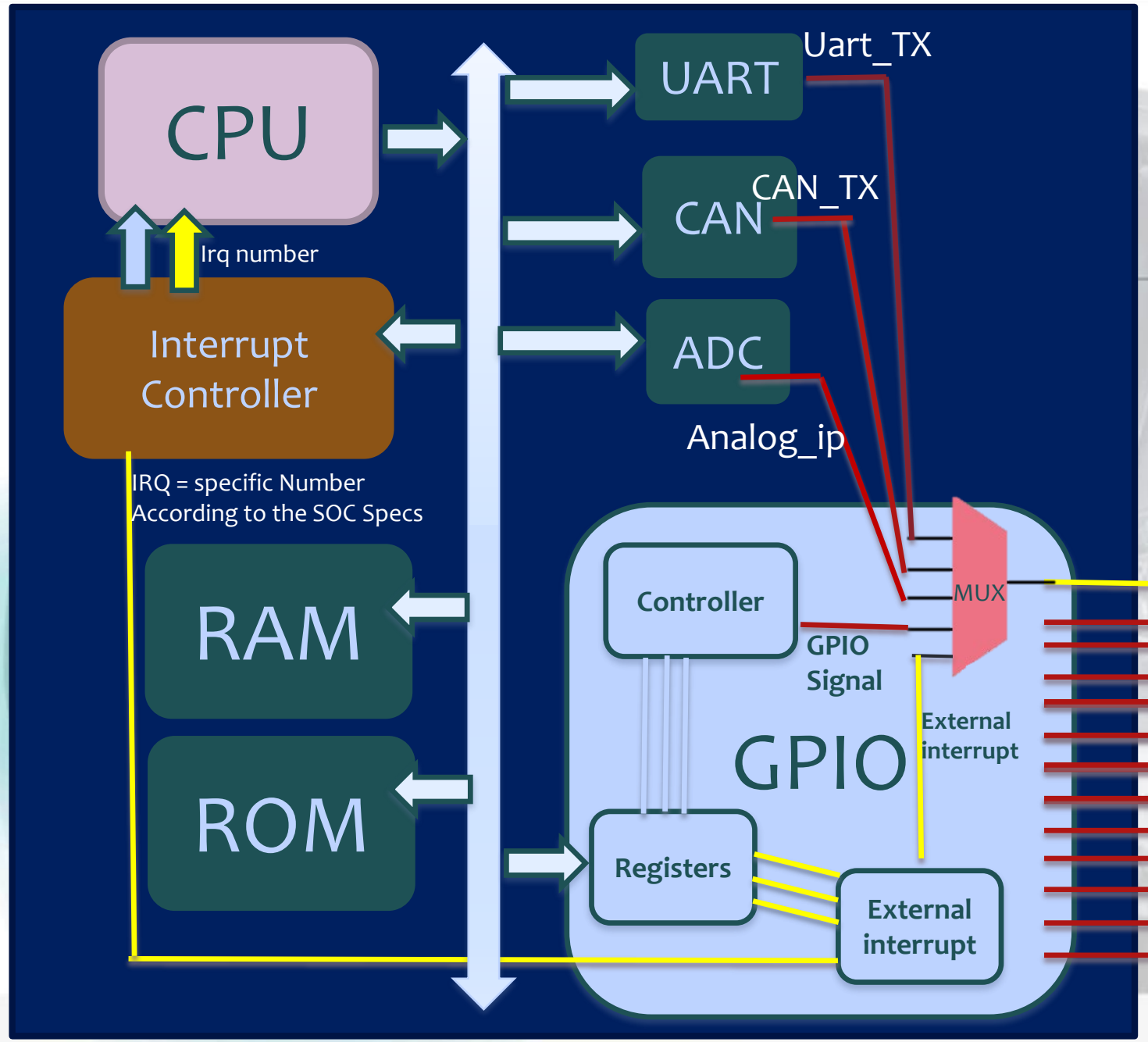
14



#LEARN_IN DEPTH

#Be_professional_in embedded_system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Each PIN

Can be configured as

- Alternative Signal
- GPIO Signal
  - Output
  - Input
    - External Interrupt
    - Normal Input

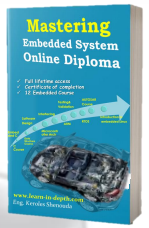
PIN = External interrupt

You Can Configure PIN to be GPIO Signal, Input and External interrupt

FOLLOW US

Press here

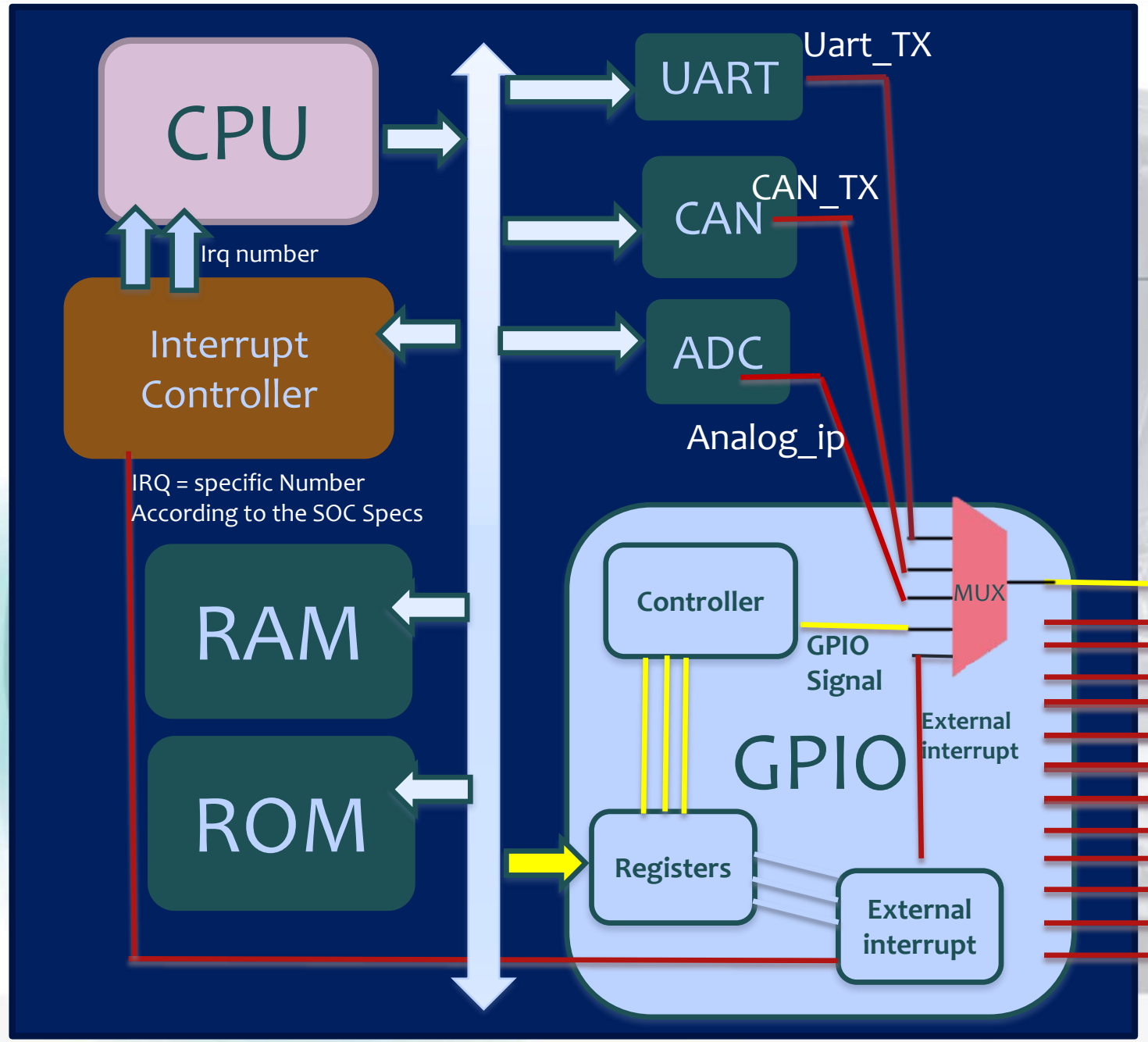
15



#LEARN_IN DEPTH

#Be_professional_in embedded_system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Each PIN

Can be configured as

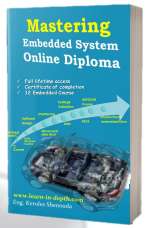
- Alternative Signal
- GPIO Signal
  - Output
  - Input
    - External Interrupt
    - Normal Input

PIN = INPUT

You Can Configure PIN to be GPIO Signal, Normal Input

The CPU will read the GPIO Data Register

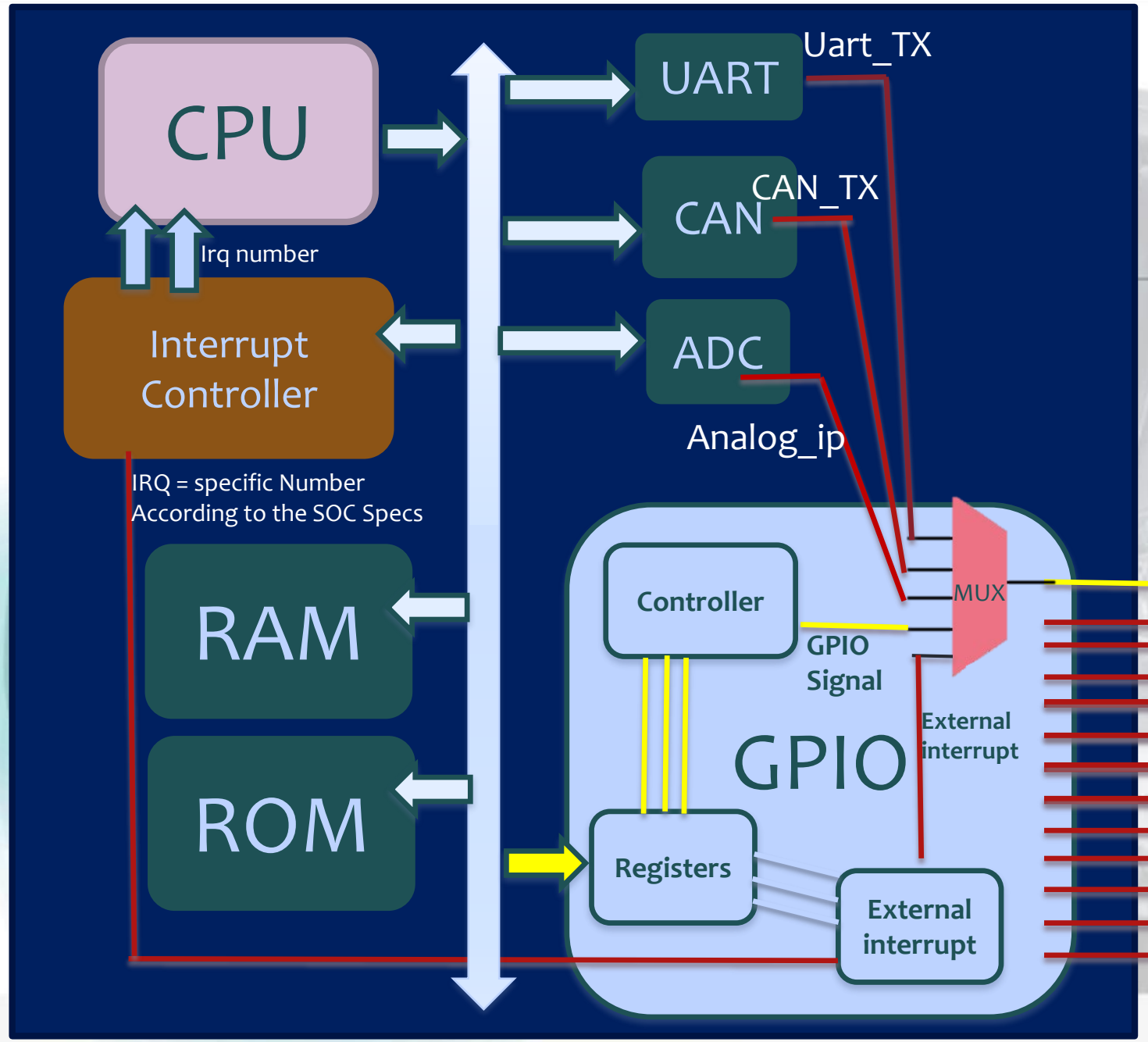
FOLLOW US
Press here
16



#LEARN_IN DEPTH

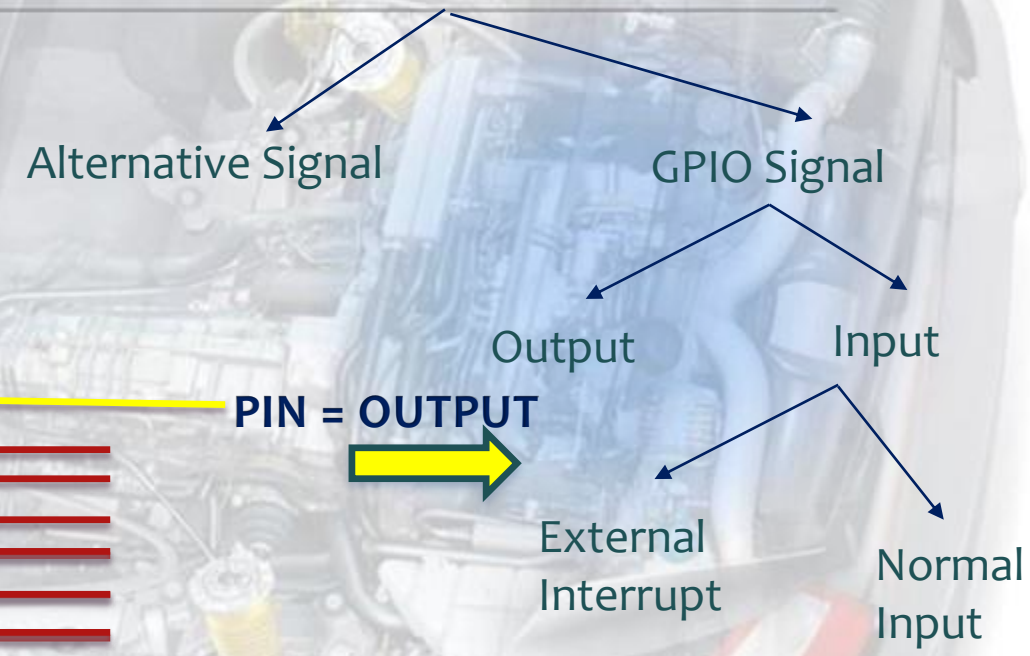
#Be_professional_in embedded_system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Each PIN

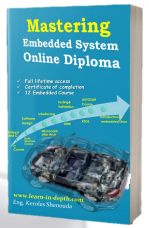
Can be configured as



You Can Configure PIN to be

GPIO Signal, Output

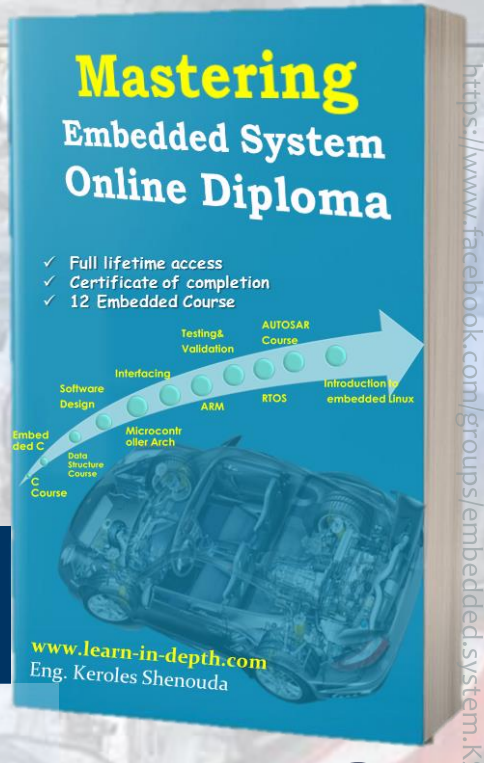
The CPU will write the GPIO Data Register



#LEARN_IN DEPTH

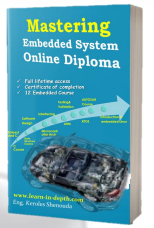
#Be_professional_in_embedded_system

GPIO Concepts



LEARN-IN-DEPTH
Be professional in embedded system

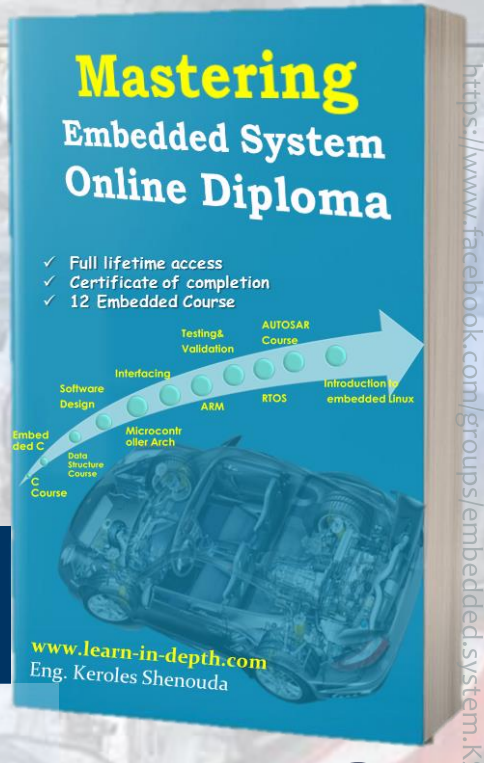
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

GPIO port consists of a group of GPIO pins in different SoCs

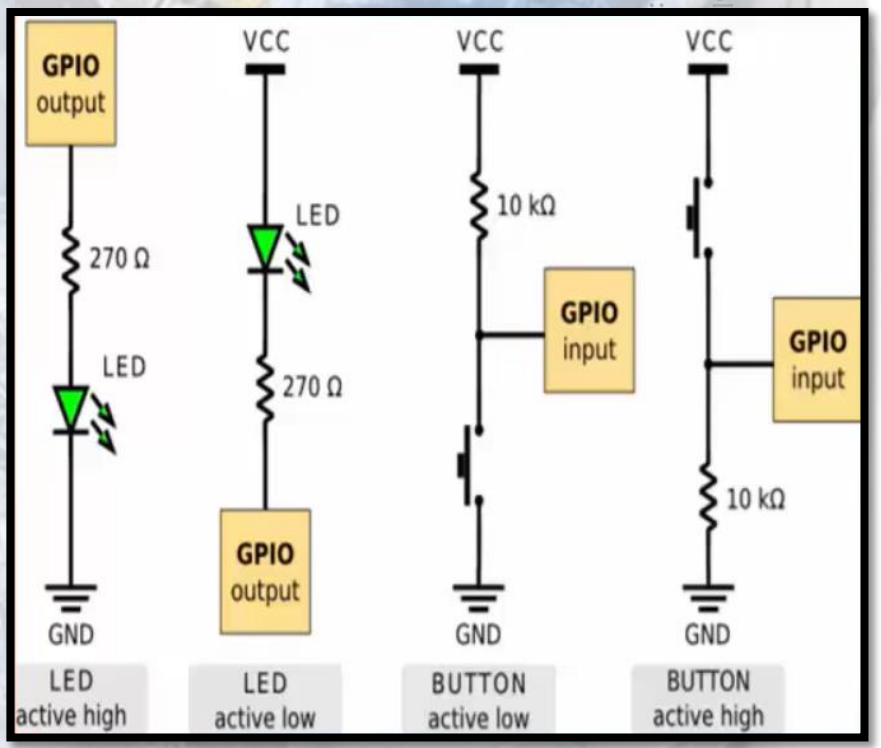


LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

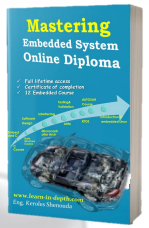
GPIO port consists of a group of GPIO pins

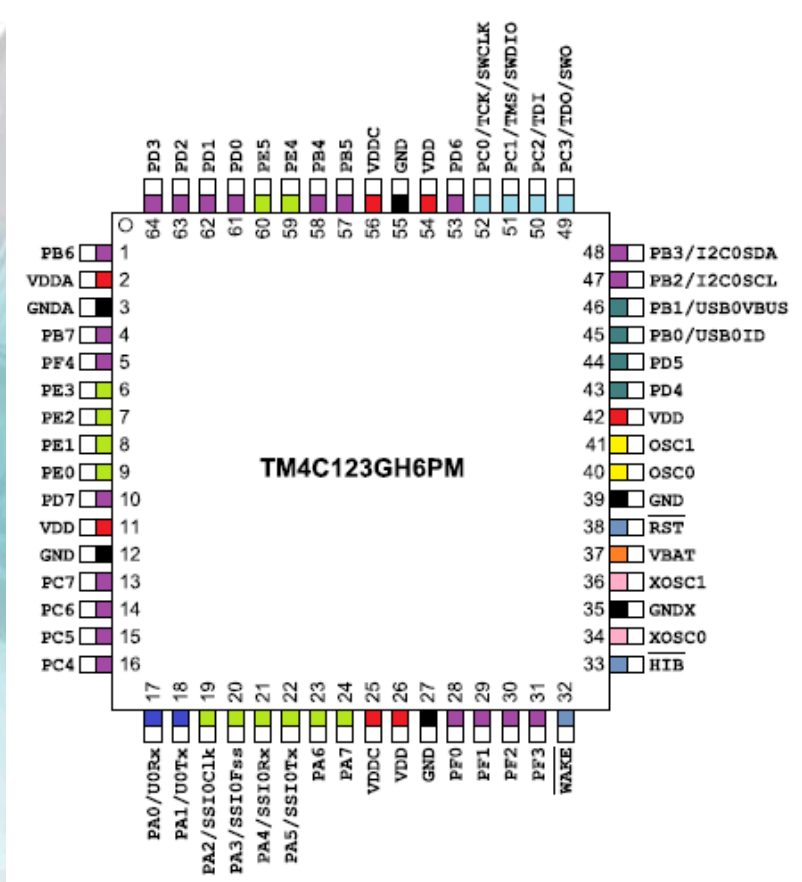
- A GPIO port **consists of a group of GPIO pins**, typically 8 or 16, which share the same data and control registers.
  - When a GPIO pin *i* is set as a *digital input*, the binary data read from this pin of this GPIO group is saved at bit *i* in the input data register (IDR). Each bit in IDR holds the digital input of the corresponding pin.
  - When a GPIO pin *i* is configured as a *digital output*, bit *i* in the output data register (ODR) holds the output of this pin. Therefore, when changing the output of a GPIO pin, the programmer should only alter the value of the corresponding bit of ODR, without affecting the other bits in ODR.
  - All GPIO pins in a GPIO port can be configured as input or output independently

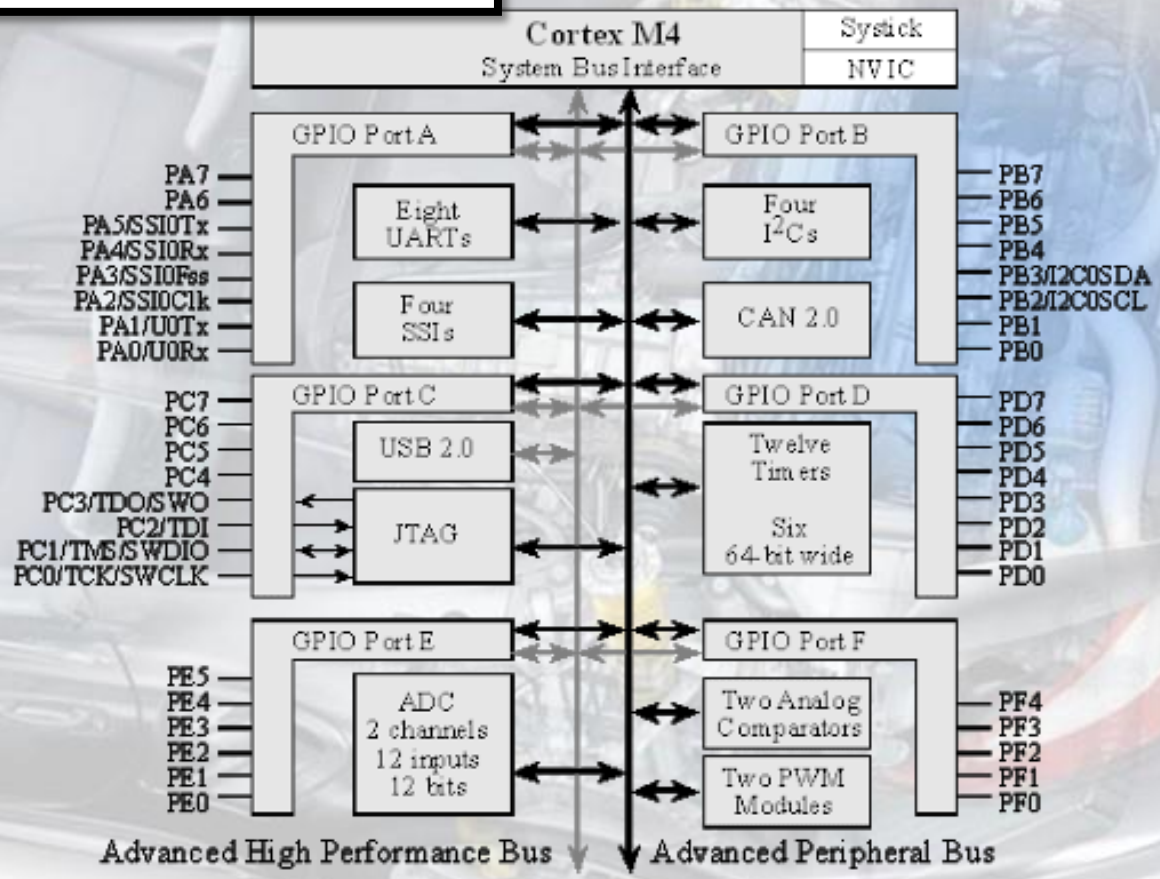


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

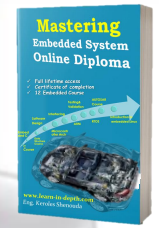
TM4C123



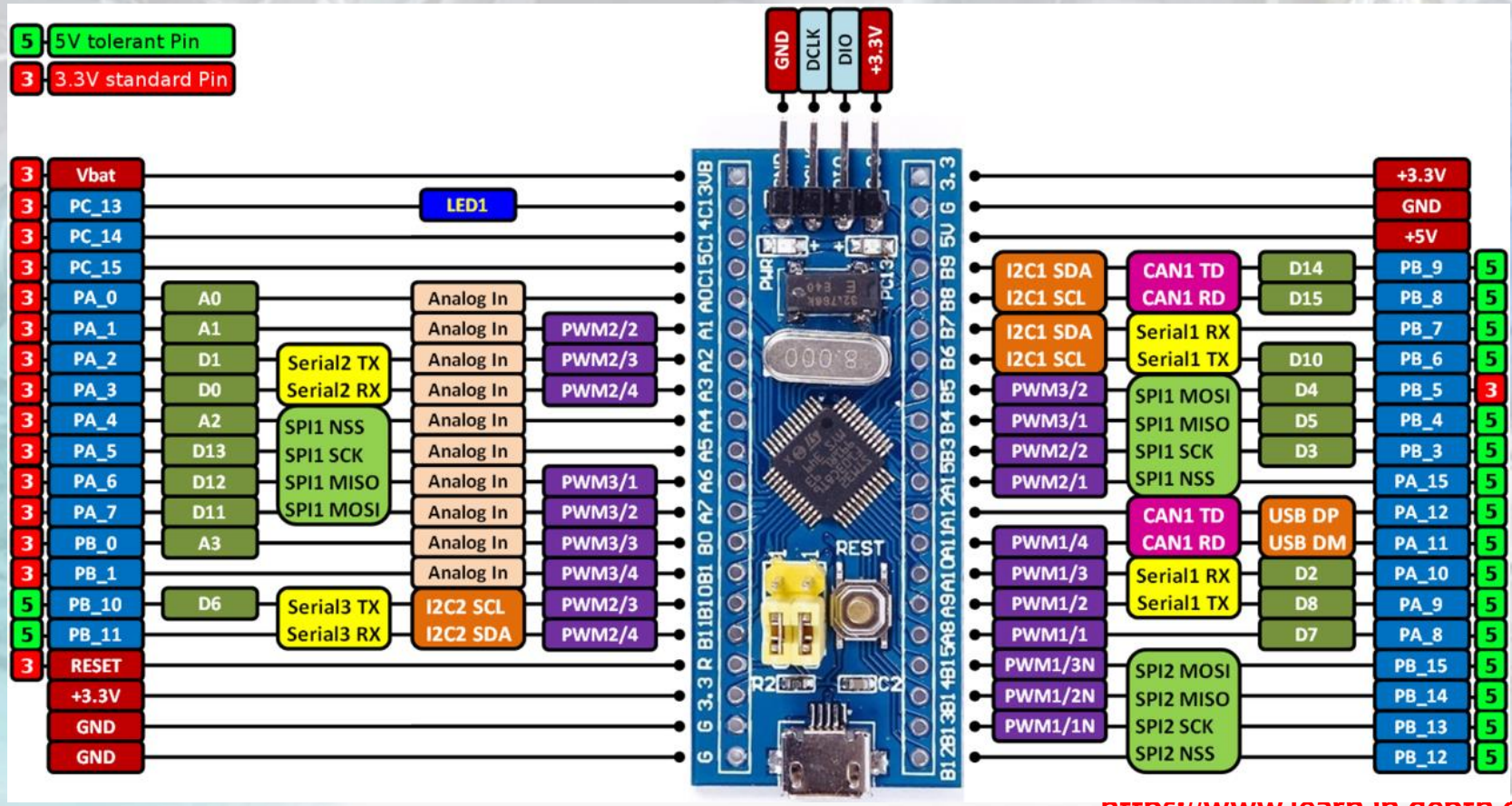




https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Stm32



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Atmega32

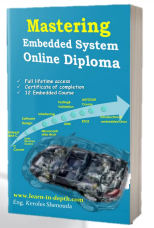
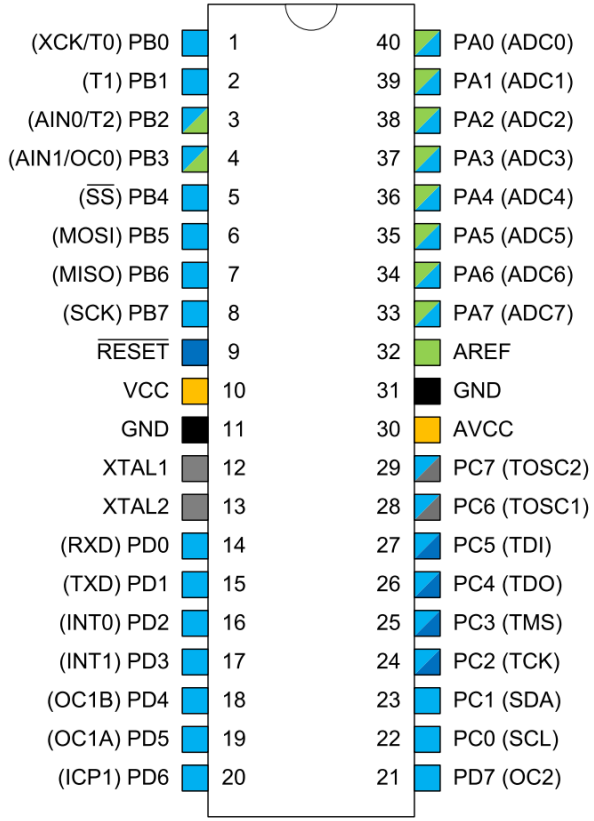
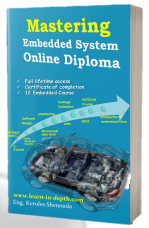


Figure 5-2 Pinout PDIP ATmega32A



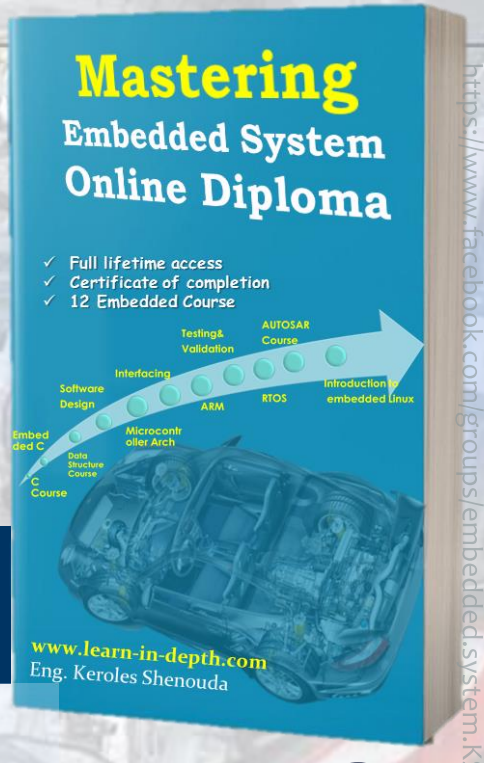
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

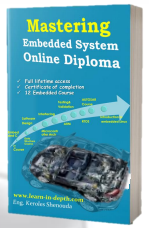
#Be_professional_in_embedded_system

GPIO Internal Circuit

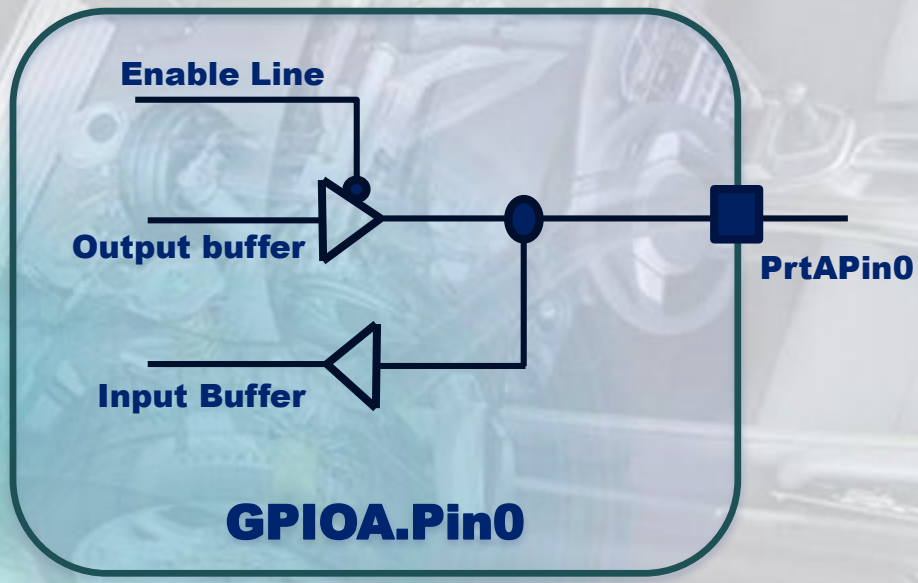


LEARN-IN-DEPTH
Be professional in embedded system

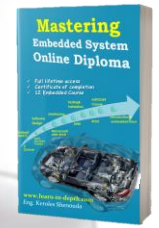
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



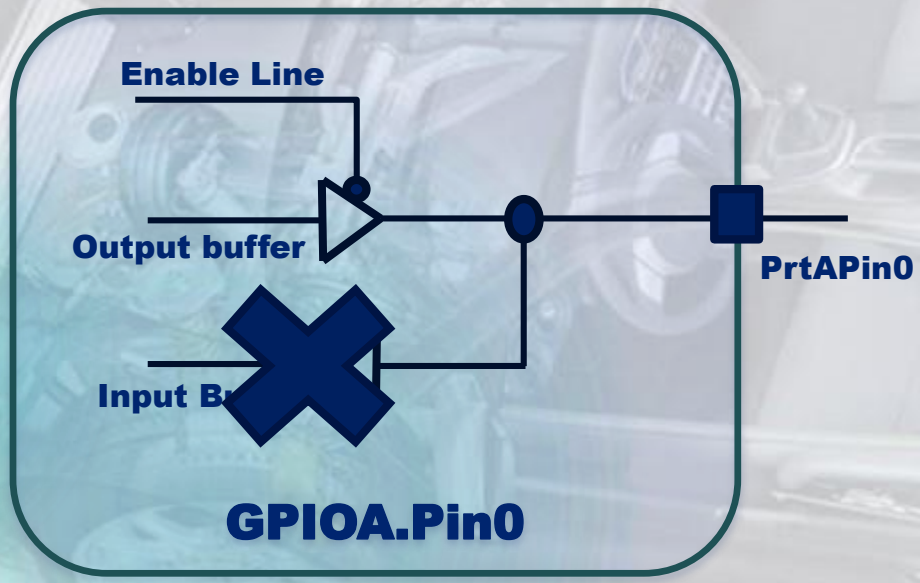
GPIO Internal Circuit

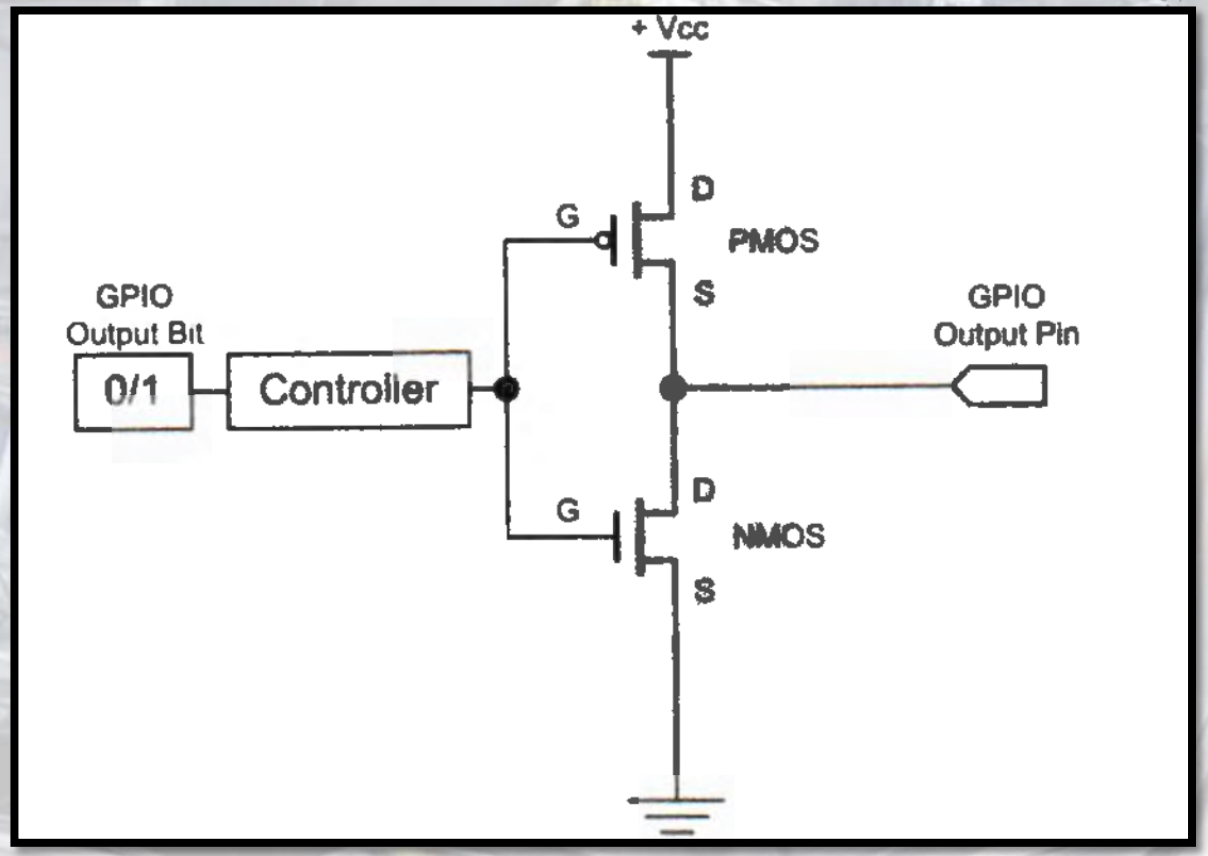


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

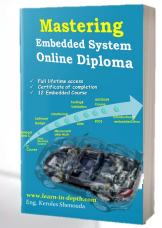


Output buffer

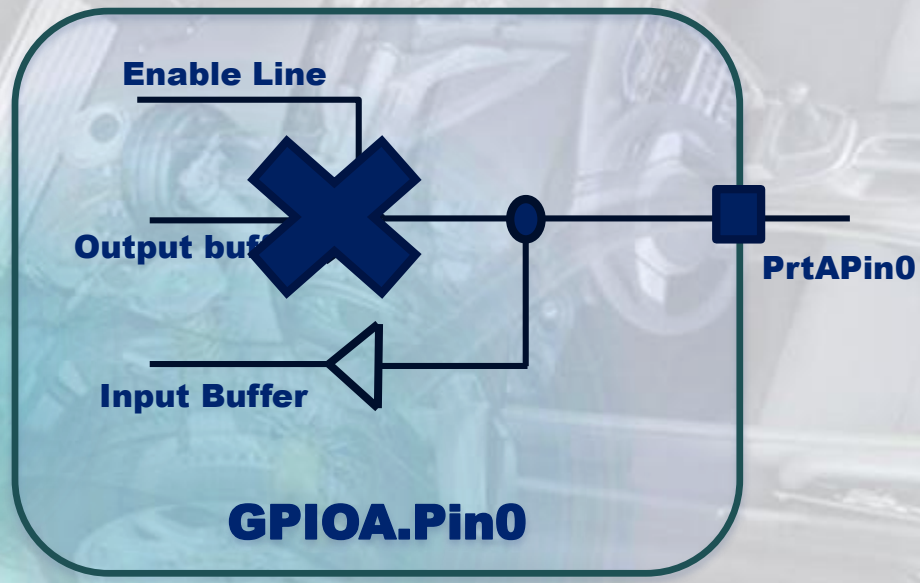


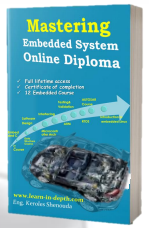


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Input buffer

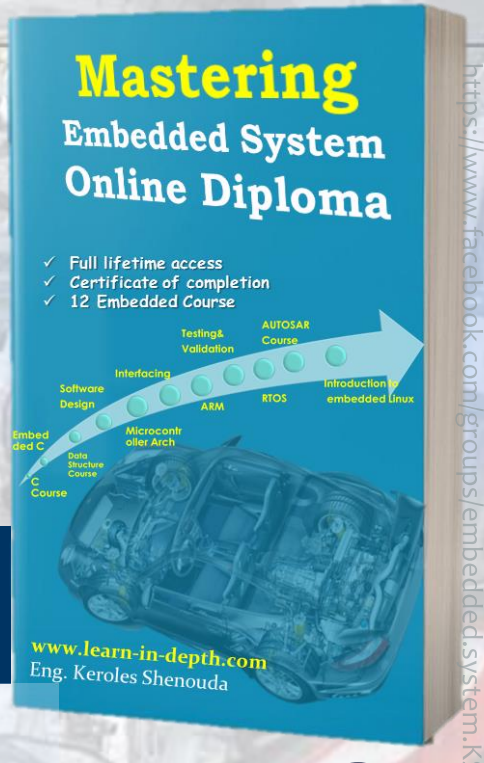




#LEARN_IN DEPTH

#Be_professional_in_embedded_system

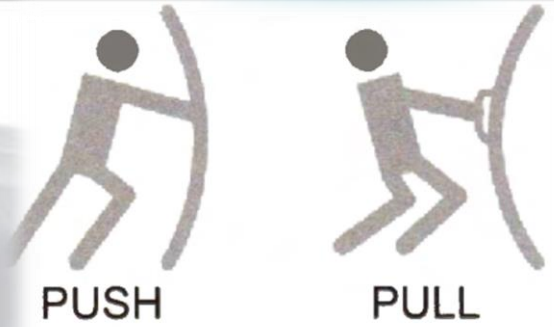
GPIO Output Modes: Push-Pull

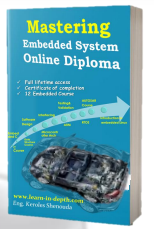


LEARN-IN-DEPTH
Be professional in embedded system

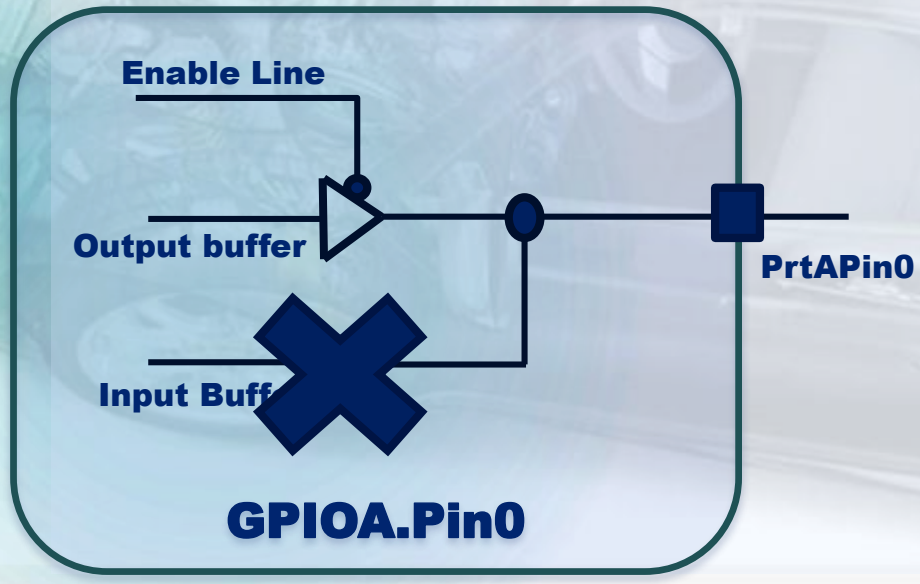
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

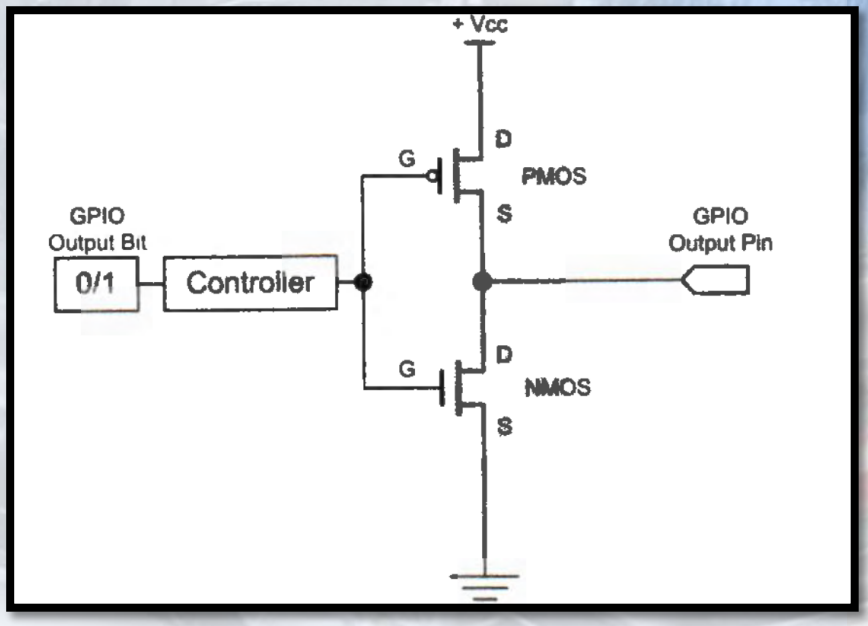
GPIO Push-Pull Output





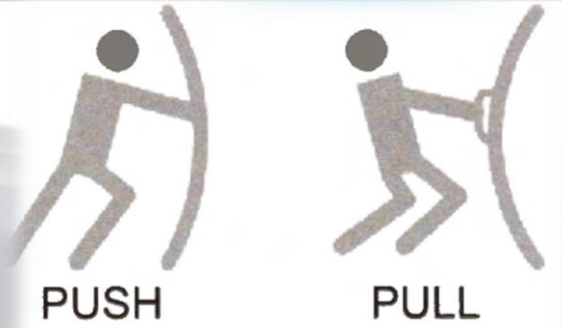
- Push-pull mode allows the **pin** to **supply** and **absorb** current.
- A push-pull output consists of a pair of complementary transistors

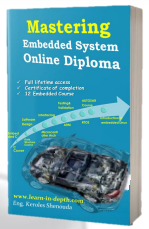




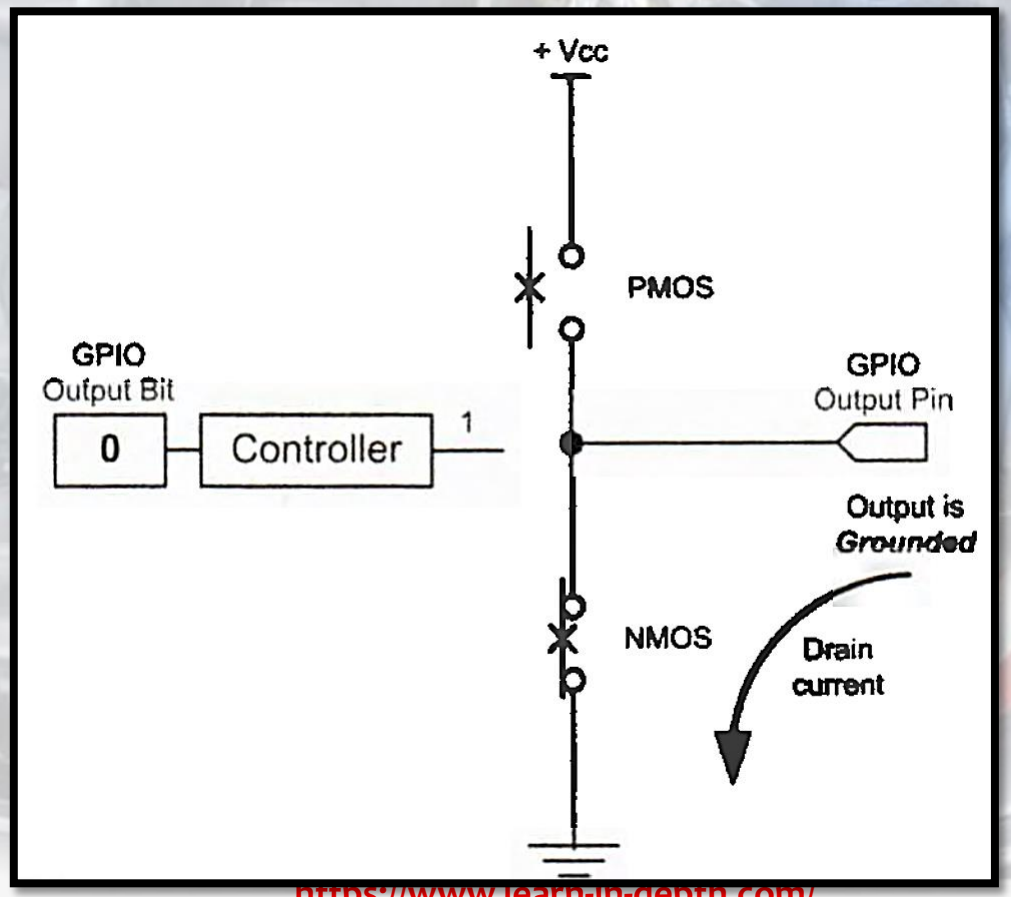
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

GPIO Push-Pull Output



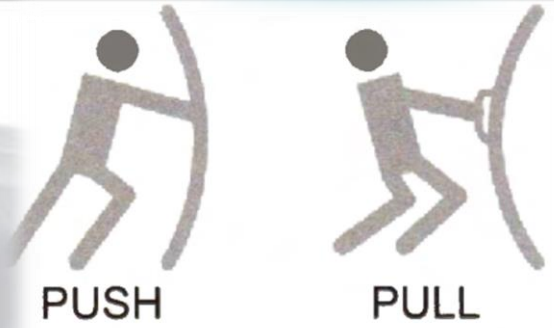


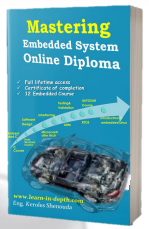
- When logic 0 is outputted, **the transistor connected to the ground is turned on** to **sink** an electric current from the external circuit



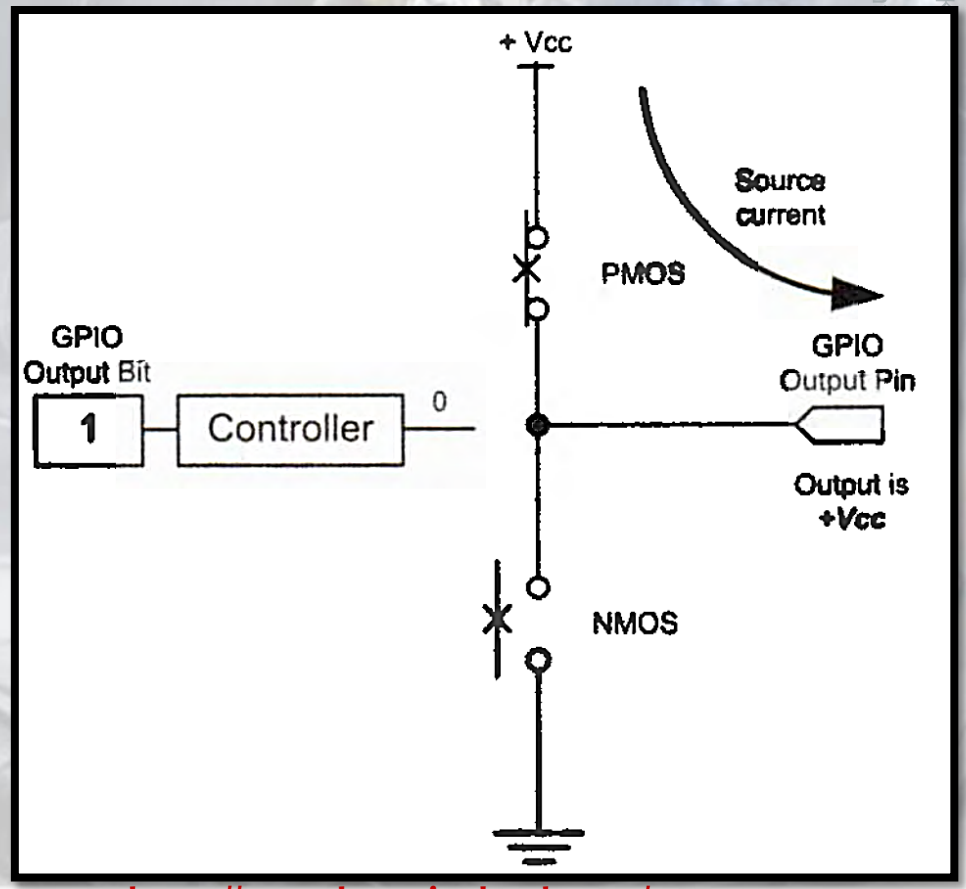
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

GPIO Push-Pull Output

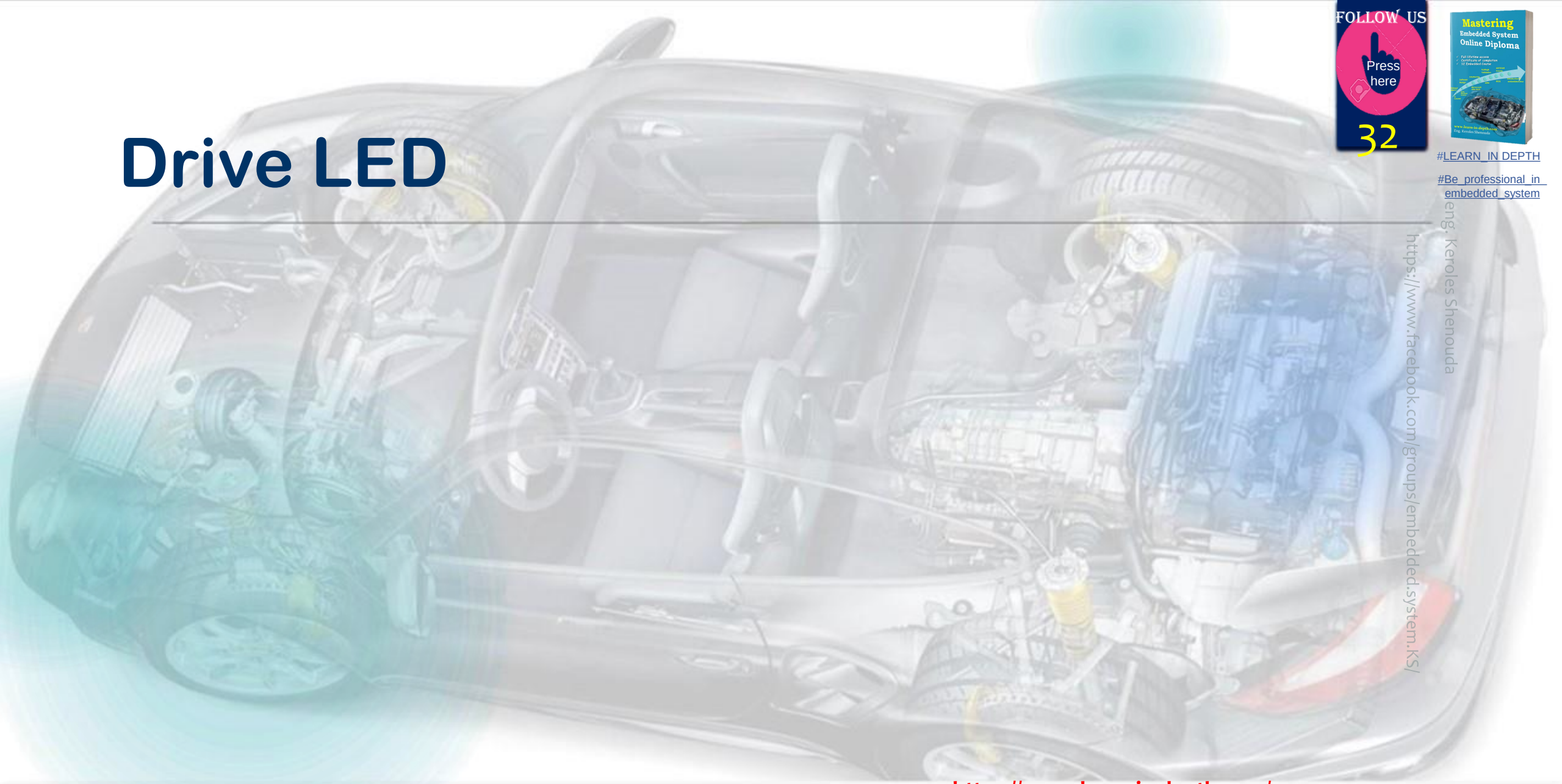




- When the pin outputs logic 1, **the transistor connected to the power supply is turned on**, and it **provides** an **electric current** to the external circuit connected to the output pin

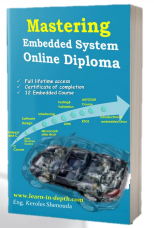


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Drive LED

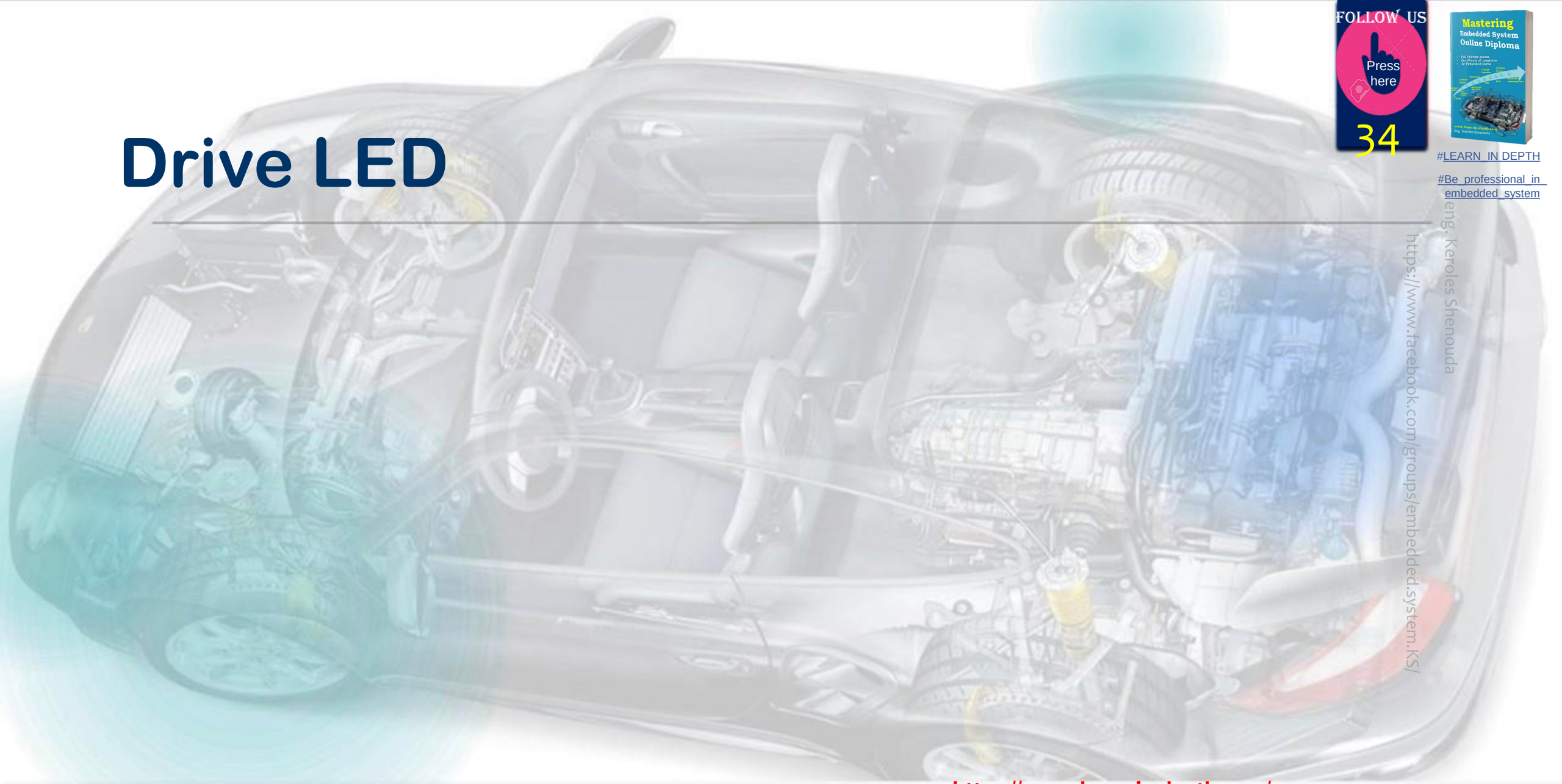
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



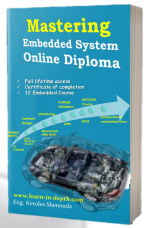
Drive LED

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Drive LED



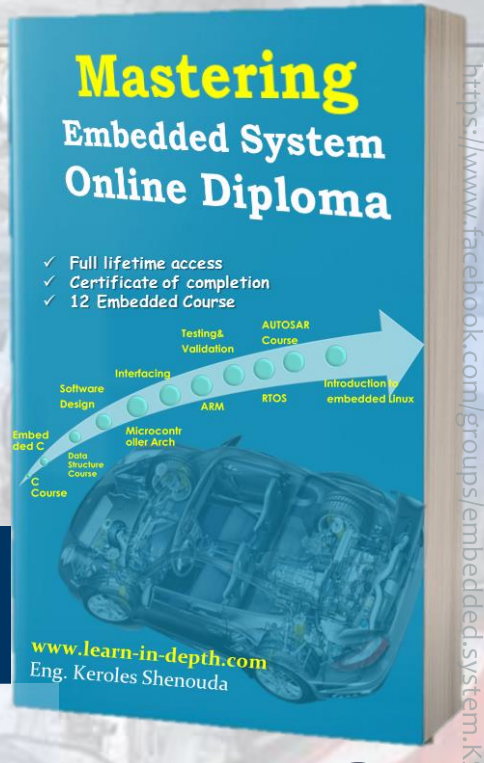
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

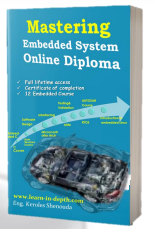
GPIO Output Modes: Open-Drain



LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

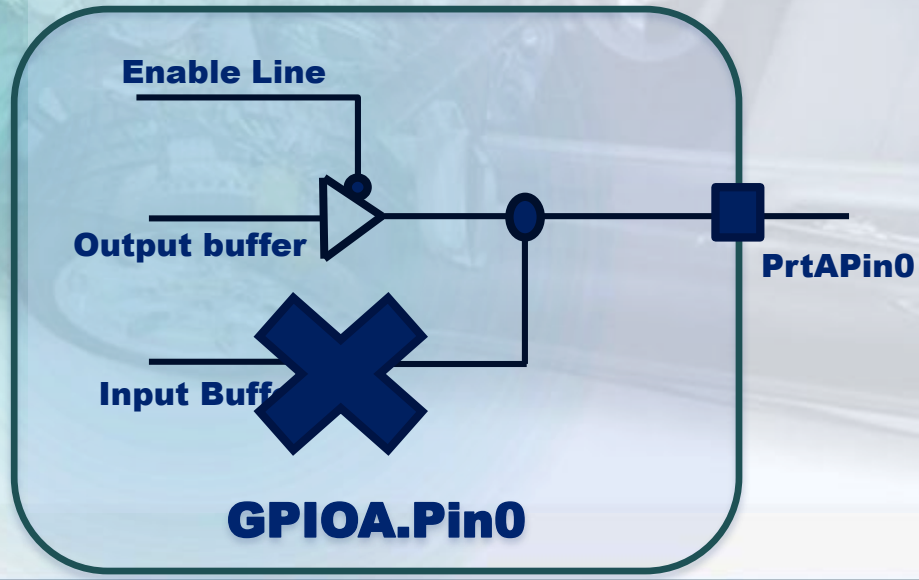
GPIO Output Modes: Open-Drain

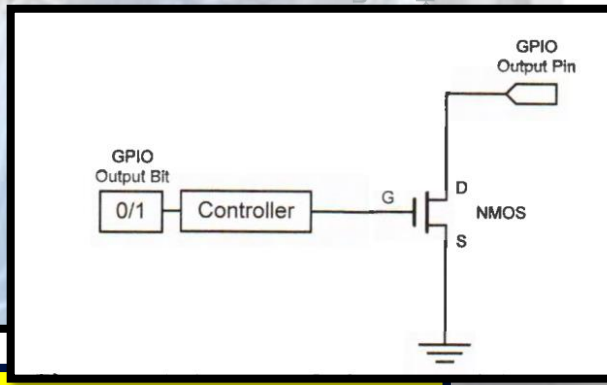


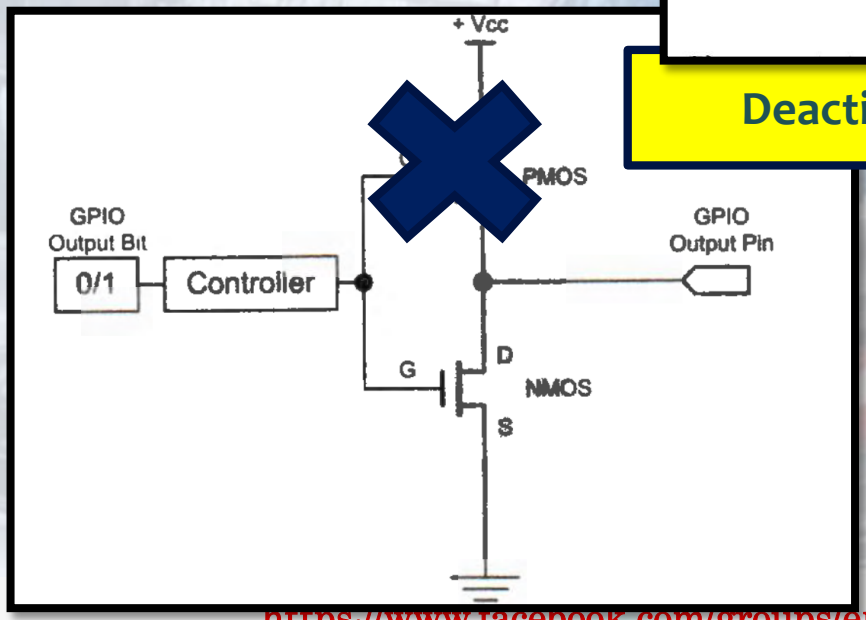
- GPIO pin in open-drain

  (also called collector) mode **can only absorb current**

- An open-drain output consists of a pair of the same type of CMOS or transistors



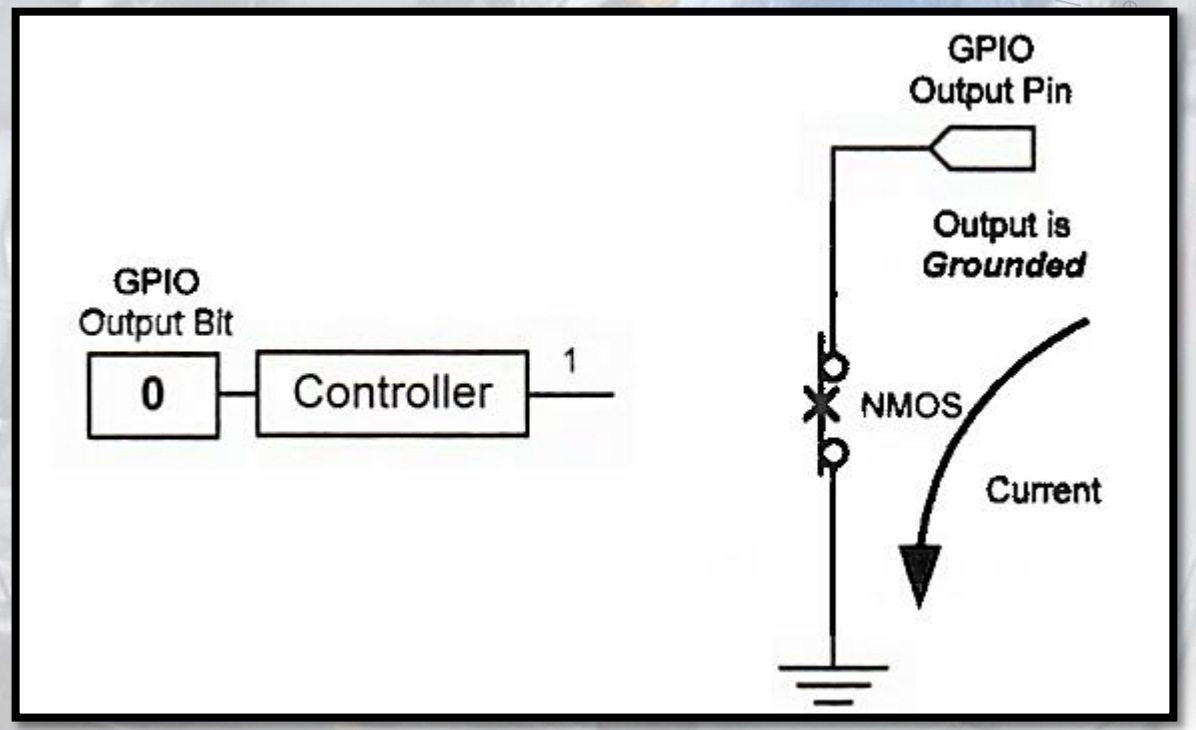




Deactivated

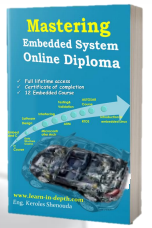
GPIO Output Modes: Open-Drain

- When software outputs a logic 0,
- the **open-drain circuit** can **sink** an electric current from the external load connected to the GPIO pin.

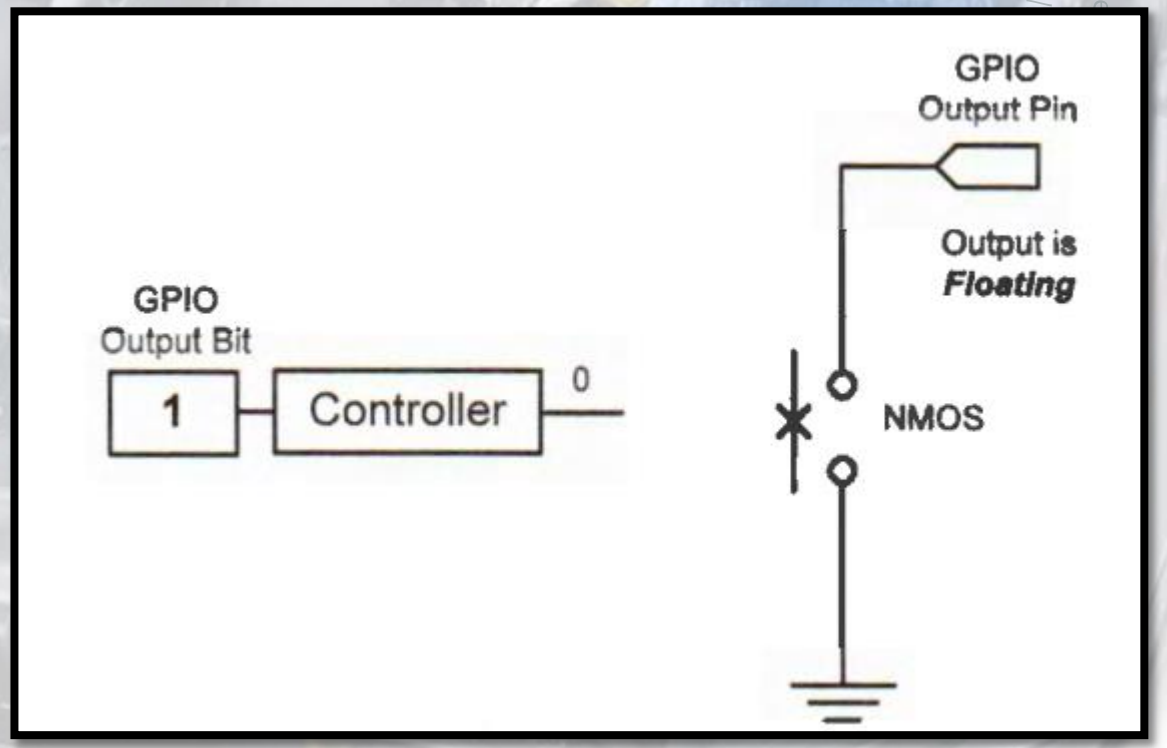


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

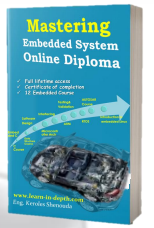
GPIO Output Modes: Open-Drain



- when software outputs a logic 1,
- it cannot supply any electric current to the external load because the output pin is **floating**, connected to neither the power supply nor the ground.



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

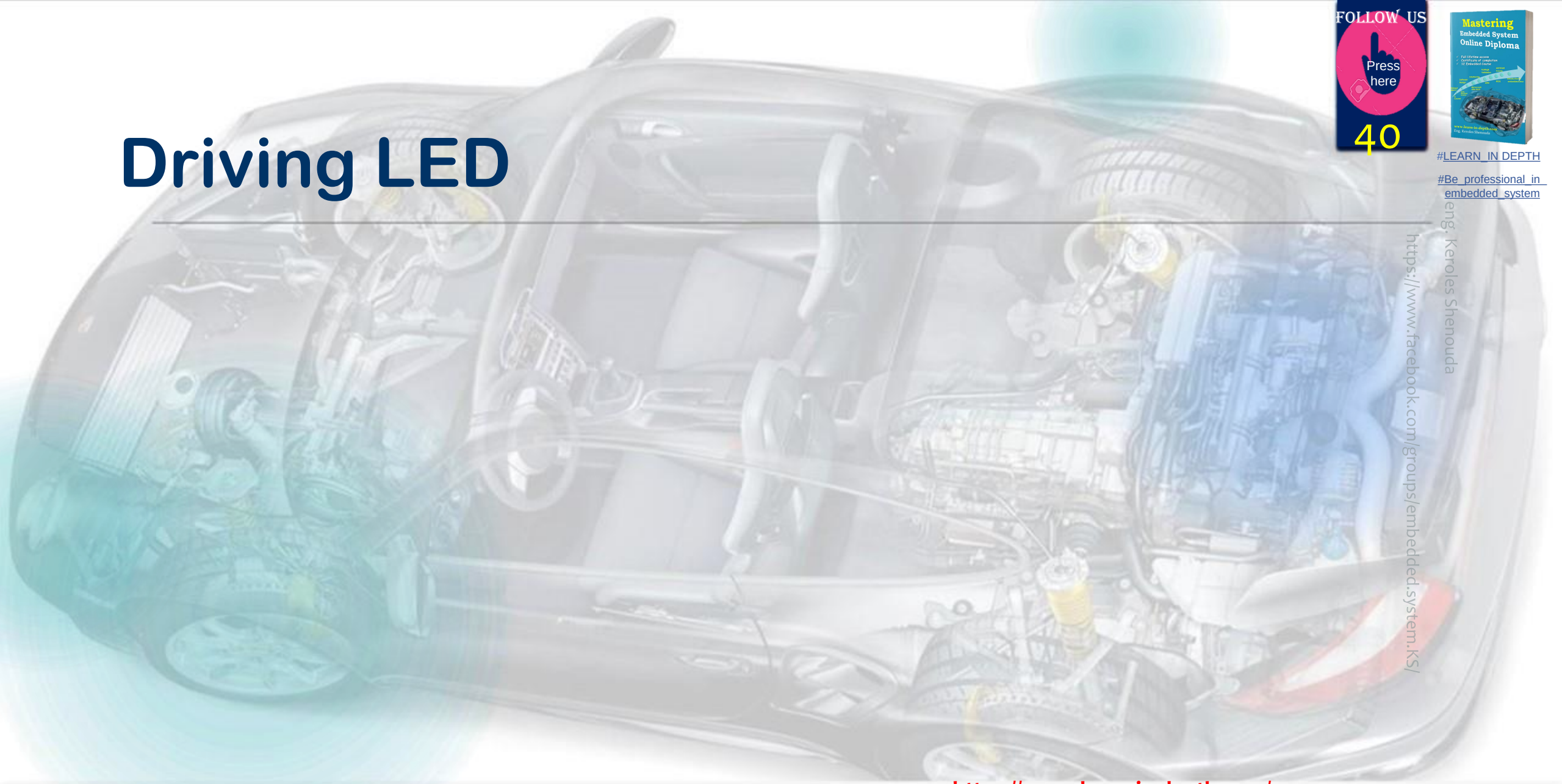


GPIO Output Modes: Open-Drain

- An open-drain output has only two states:
- low voltage (logic 0),
- and high impedance (logic 1). It often has an external pull-up resistor

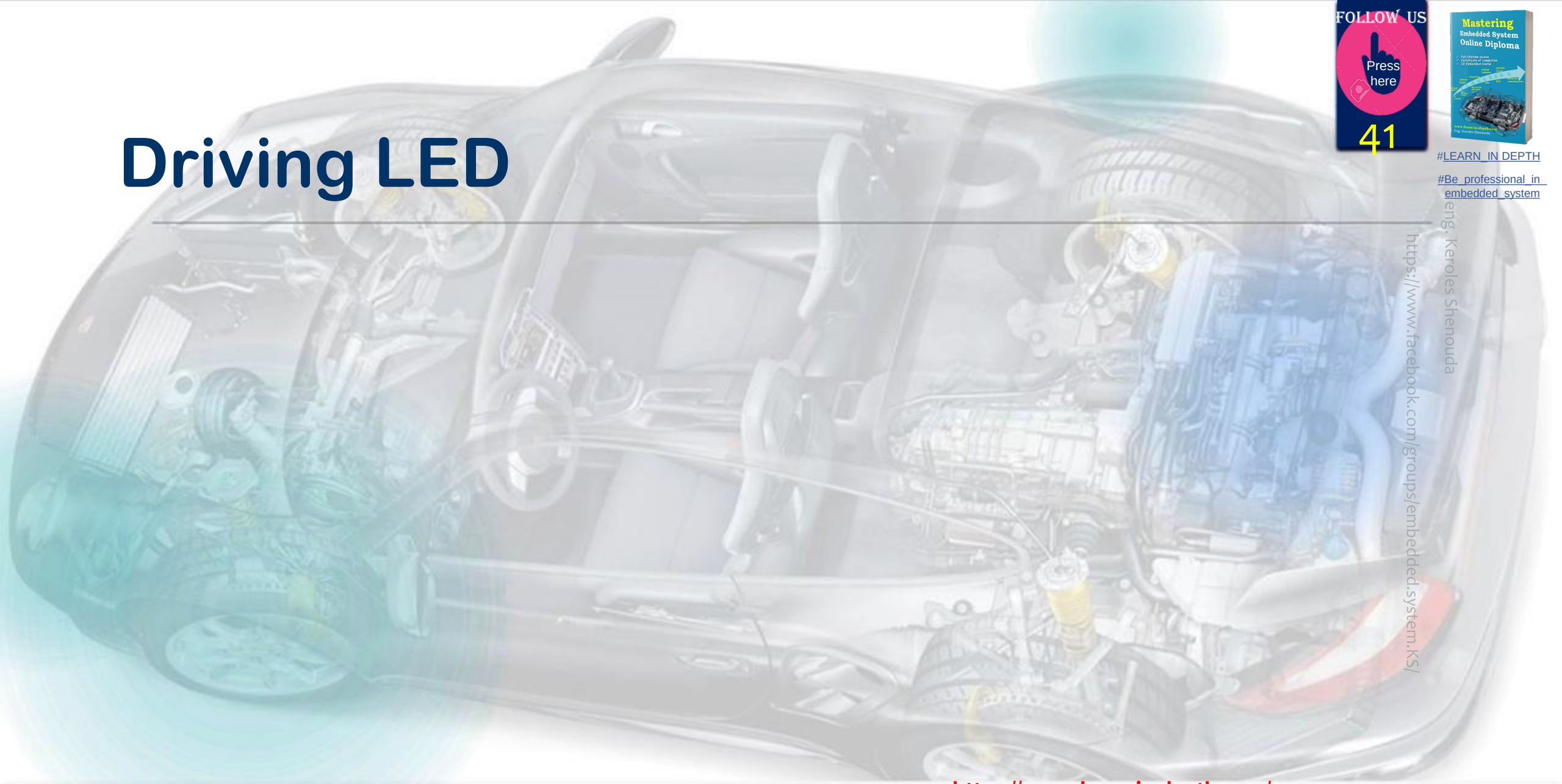
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Driving LED



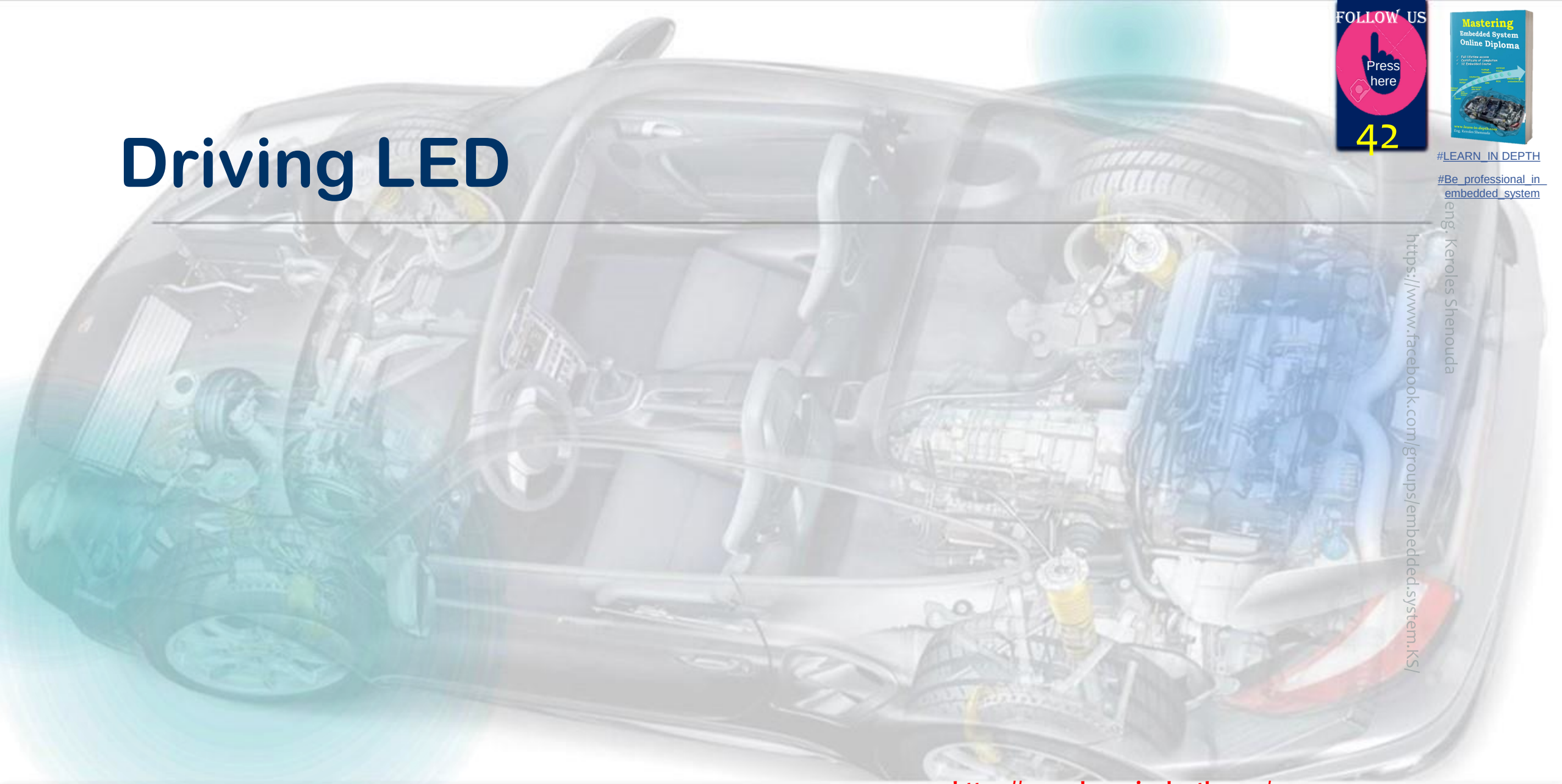
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Driving LED

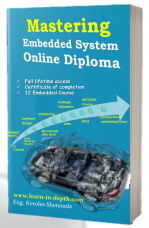


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Driving LED



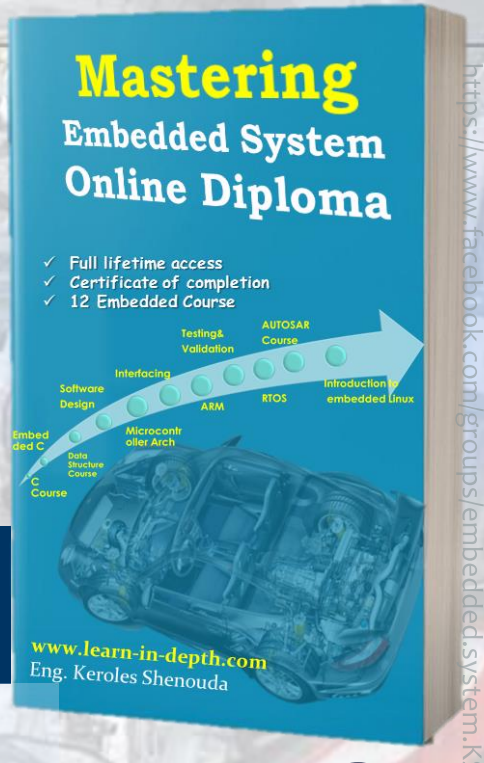
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

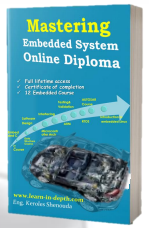
#Be_professional_in_embedded_system

What is the important usage of open-drain outputs ?



LEARN-IN-DEPTH
Be professional in embedded system

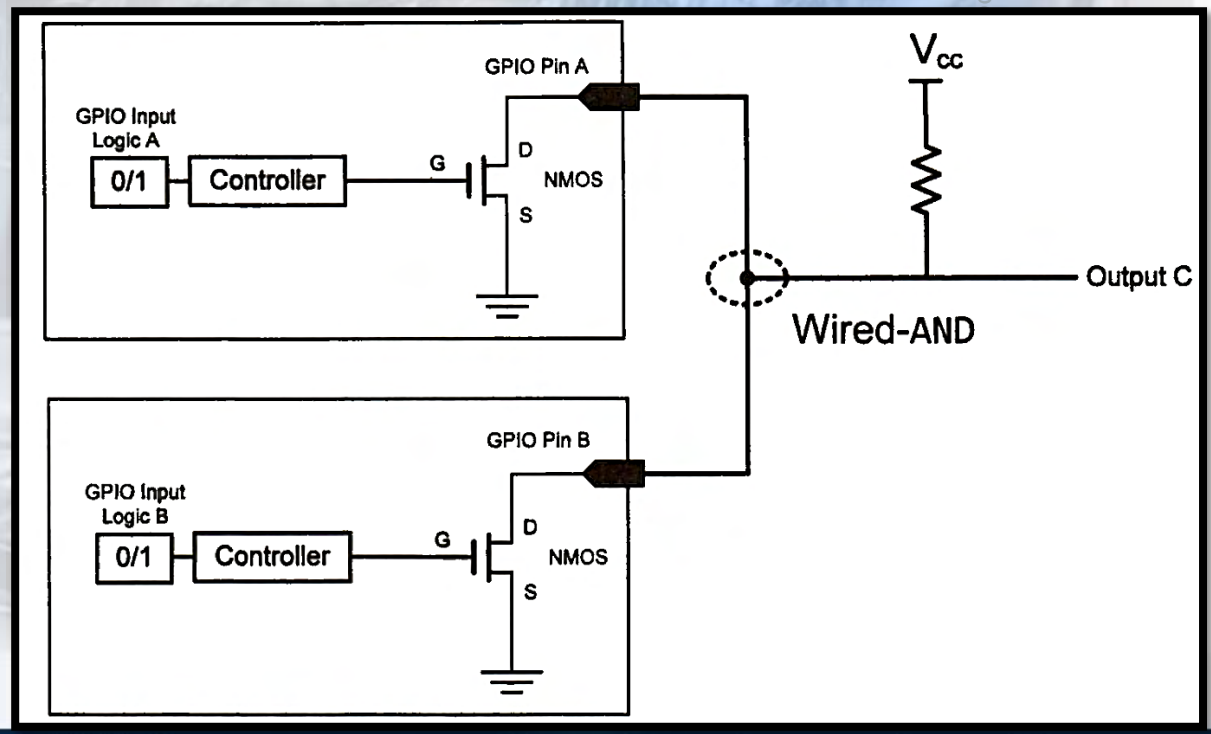
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



What is the important usage of open-drain outputs ?

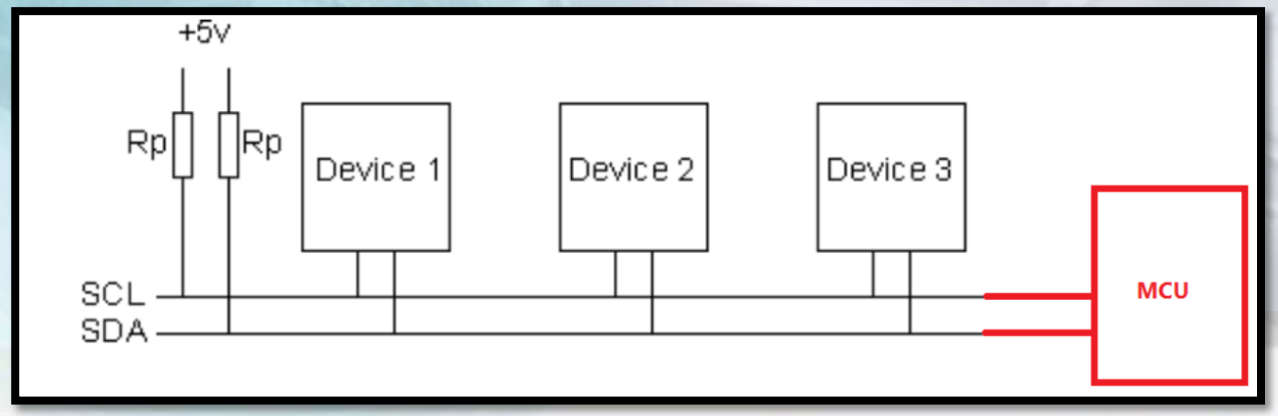
- One important usage of open-drain outputs is to connect directly **several outputs together** and **implement wired logic AND** (active high) or OR (active low) circuit in an easy way.
- If multiple open-drain output pins are connected and **are pulled up via a shared resistor**,
- any output pin can drive the output voltage low.
- The pin voltage is high if and only if all pins output a high voltage level.

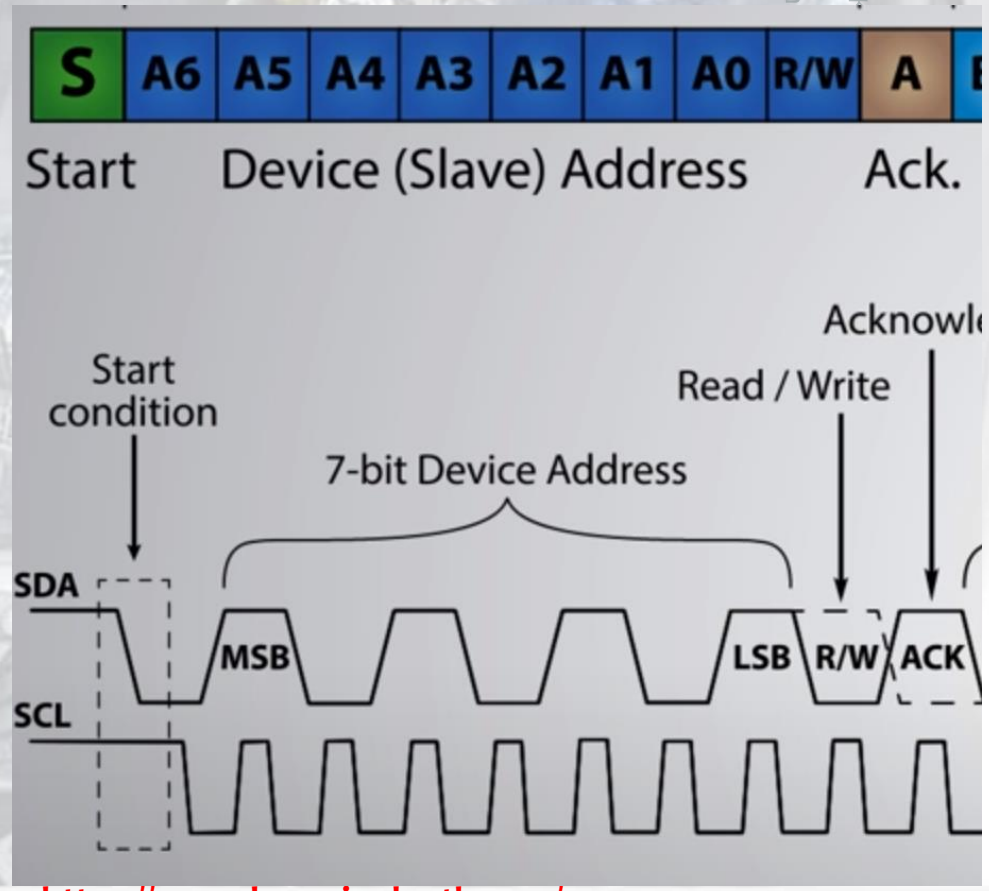
Inputs				Output
Logic A	Logic B	Circuit A	Circuit B	
0	0	Drain	Drain	0
0	1	Drain	Open	0
1	0	Open	Drain	0
1	1	Open	Open	1



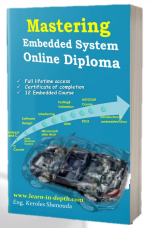
Driving I2C Bus

- *in this case both lines have pull-up resistors*
- The pull-up resistor connects to HIGH (which is the same as the supply voltage or Vcc) at one end and connects to one or more external pins of the **open-drain devices** (via the bus). **Thus, if any one of the open-drain devices is set to sink current, current flow for all of the devices sinks to ground.**





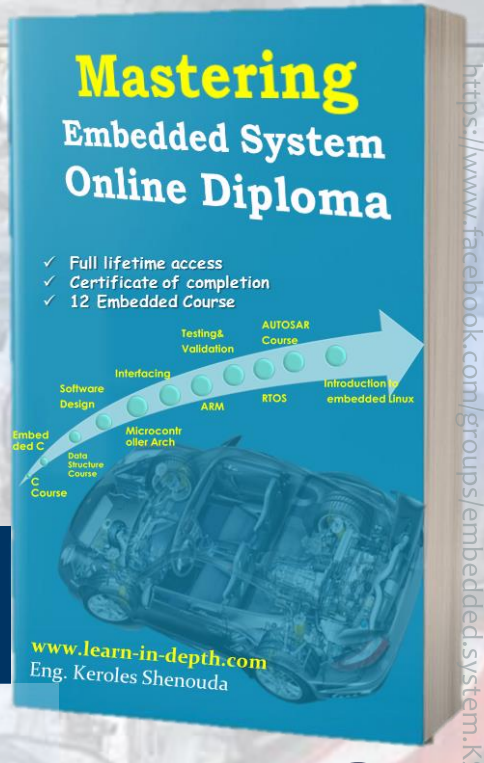
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

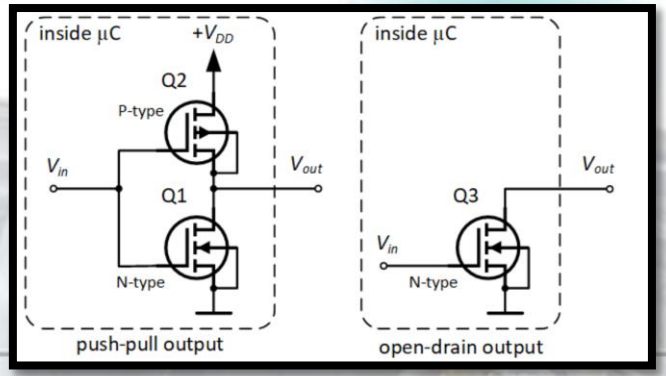
open-drain Vs push-pull (output mode)



LEARN-IN-DEPTH
Be professional in embedded system

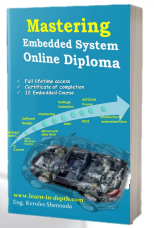
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

open-drain Vs push-pull (output mode)



- Push-pull output is best suited for communication interfaces that have **single direction lines** (e.g SPI, UART etc.).
- Open drain is commonly used for **bidirectional single line communication interfaces**, where more than two devices are connected on the same line(e.g I2C, One-Wire etc.)
- **Open drain** output **has higher power consumption** during active transfers due to the pull-up resistors that are used.
- In general, **the push-pull output** has **faster slopes** than the open drain output.
  - Another advantage is that **it can supply current and simplify the circuit.**
  - For example, a push-pull output can directly control an external LED while an open-drain output cannot light up an LED without external voltage source.

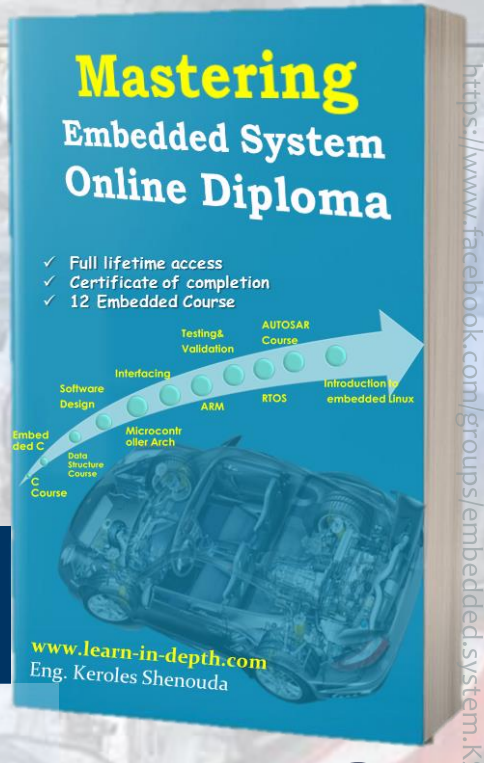
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

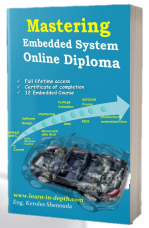
#Be_professional_in_embedded_system

GPIO Output Speed: Slew Rate



LEARN-IN-DEPTH
Be professional in embedded system

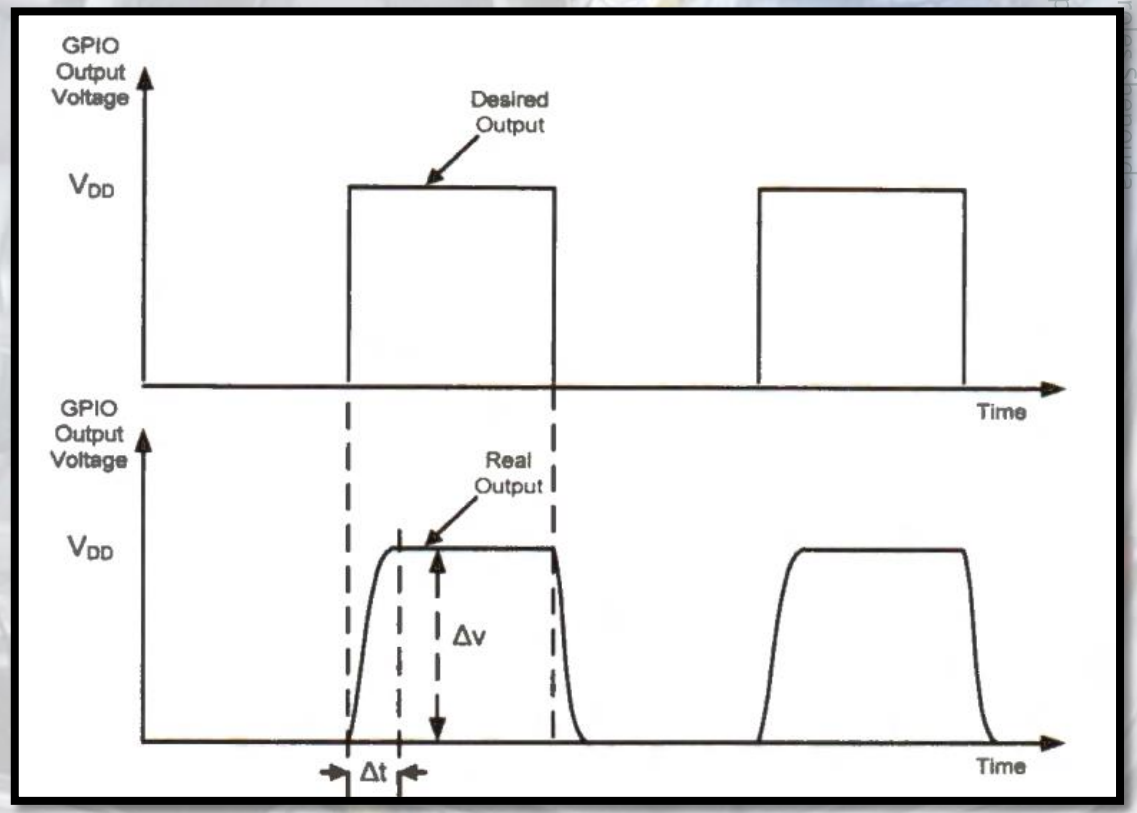
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



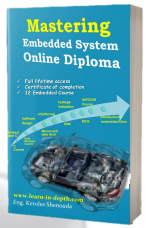
GPIO Output Speed: Slew Rate

$$Slew\ Rate = \frac{\Delta V}{\Delta t}$$

- The slew rate of a GPIO pin is the speed of change of its output voltage per unit of time, as defined as follows.
- If the logic output of a GPIO pin changes from 0 to 1 and accordingly the voltage output of this pin rises from OV to 3V in 3µs,
  then the slew rate is 1 volt per µs
- The higher the slew rate, the shorter time the output voltage takes to rise or fall to desired values.
- Therefore, a higher slew rate allows faster speed at which the processor can toggle the logic level of a GPIO pin.



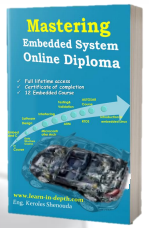
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



GPIO Output Speed: Slew Rate

- a large slew rate often causes high electromagnetic interference (EMI), also called radio frequency interference (RFI) to neighbor electronic circuits
- A fast rising and falling signal has large-amplitude and high-frequency harmonics, which can transfer to a victim circuit via radiation, conduction, or induction
- The slew rate of the GPIO circuit is programmable by setting the GPIO output speed. For example, the digital output speed of a GPIO pin can be low speed (400 kHz), medium speed (2 MHz), fast speed (10 MHz), or high speed (40 MHz)

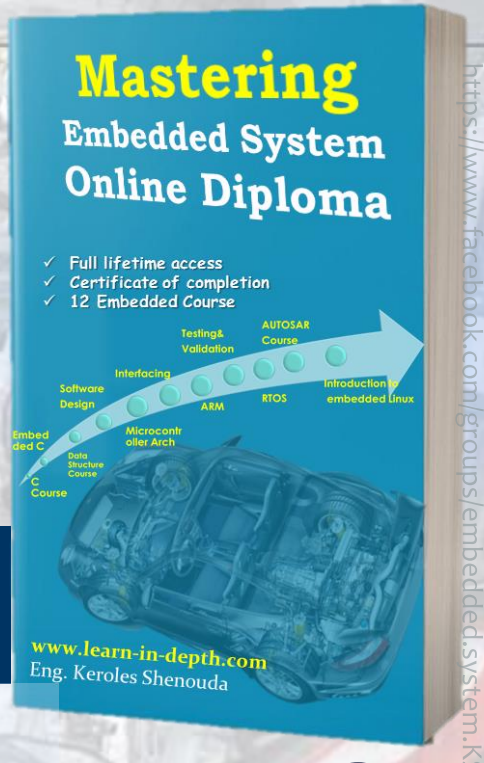
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

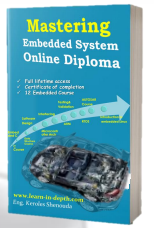
#Be_professional_in_embedded_system

GPIO Input buffer Circuit

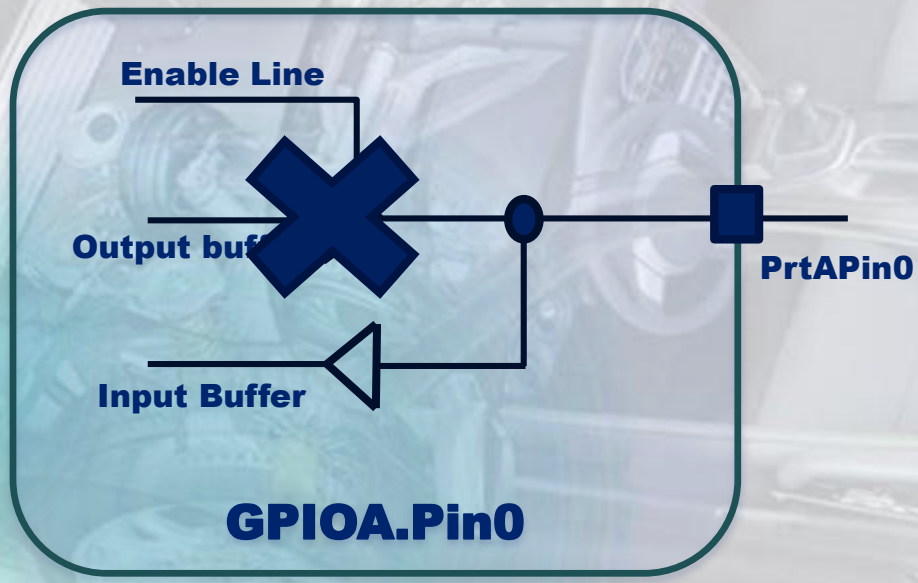


LEARN-IN-DEPTH
Be professional in embedded system

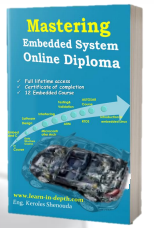
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Input buffer

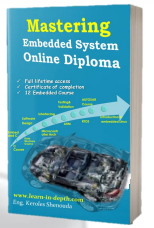


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



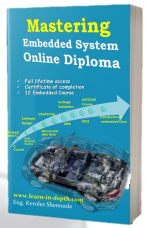
Input buffer

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Input buffer

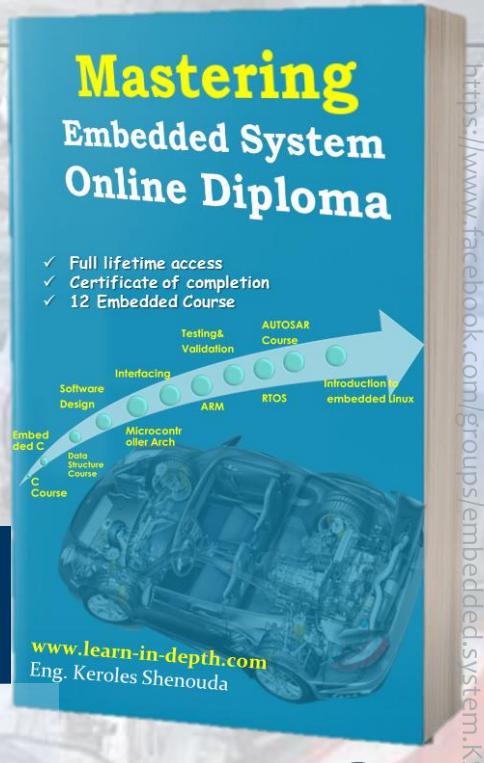
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

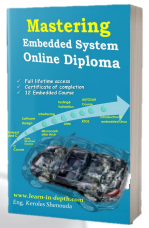
#Be_professional_in_embedded_system

High impedance (High-Z)/Floating pin



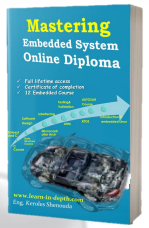
LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



High impedance (High-Z, Hi-Z) pin

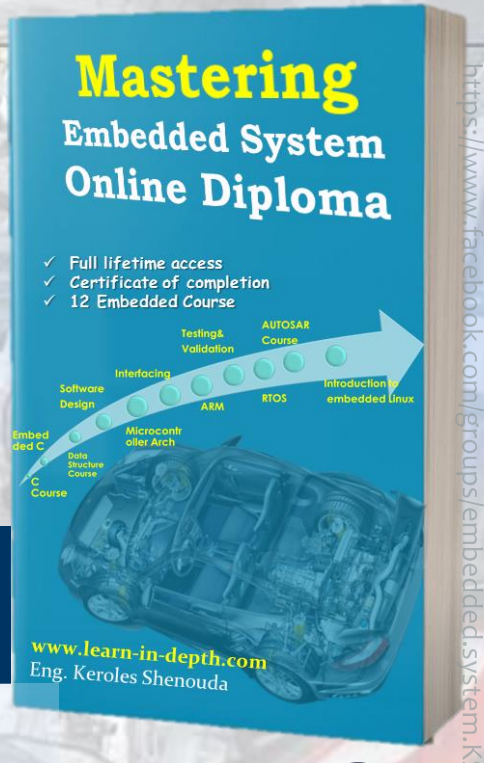
- A pin that is characterized by having a high impedance
- It is not actively driven and is "floating" unless another external device or circuitry (pull-up/pull-down) is driving it.
- A high impedance pin that is not driven by a pull-up or pull-down circuit is said to be **floating**.



#LEARN_IN DEPTH

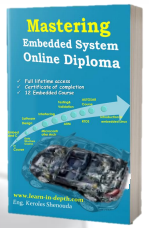
#Be_professional_in_embedded_system

GPIO Input Modes: Pull Up and Pull Down



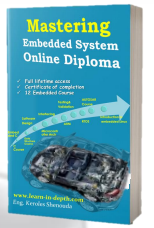
LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



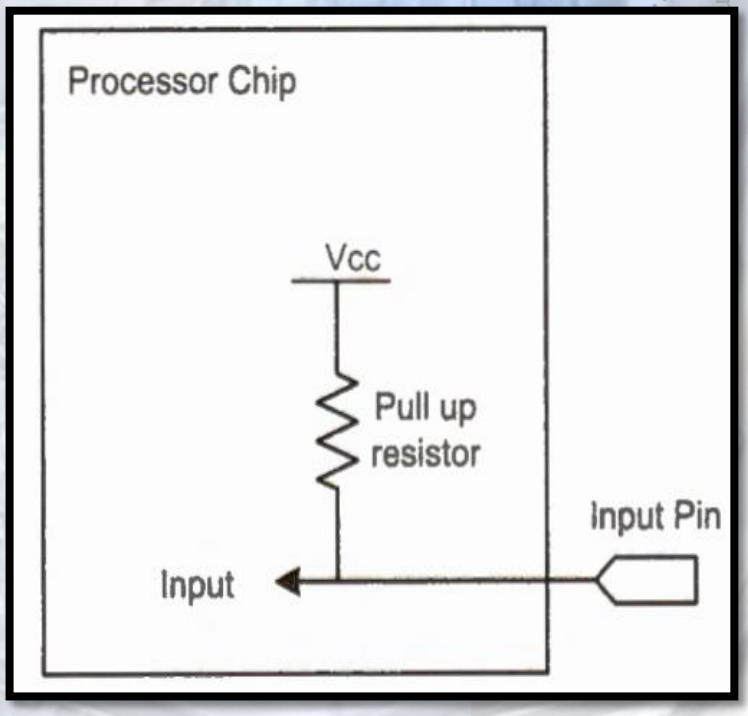
GPIO Input Modes: Pull Up and Pull Down

- When a GPIO pin is used as digital input, the pin has three states:
  - **high voltage**
  - **Low voltage**
  - **high impedance (also called floating or tri-stated)**
- Pull-up and pull-down are used to ensure the input pin has a valid high (logic 1) or a valid low (logic 0) when the **external circuit does not drive the pin**



Pull Up

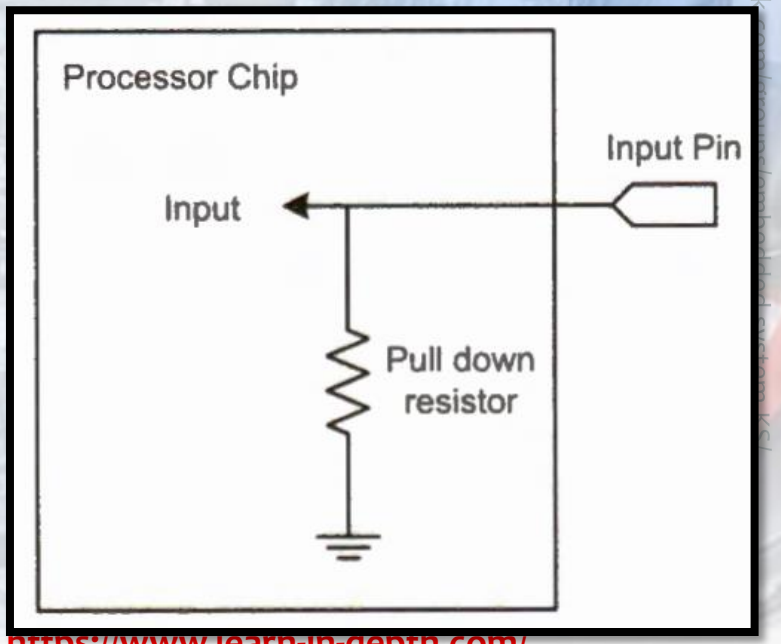
- When software configures a pin as pull-up, the pin is internally connected to the power supply via a resistor
- The pin is always read as high (logic 1) unless the external circuit drives this pin low



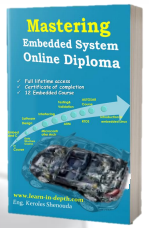
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Pull Down

- when a pin is configured as pull-down, the pin is then internally connected to the ground via a resistor
- The pin is always read as low (logic 0) unless the external circuit drives this pin high



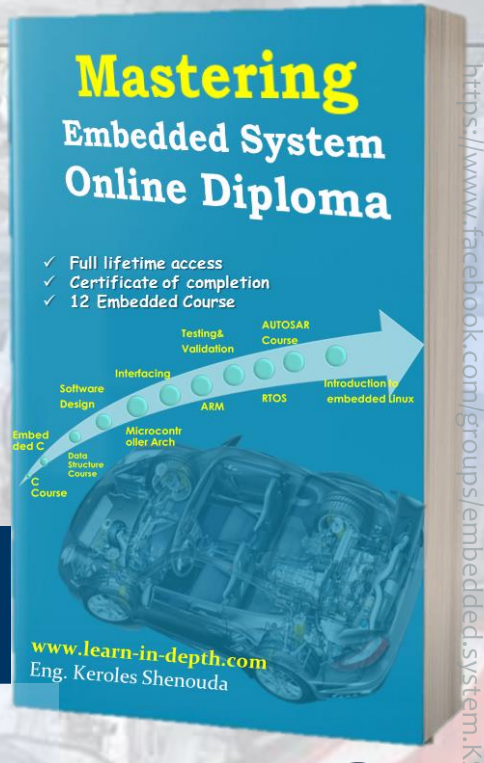
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

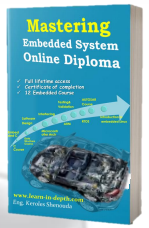
#Be_professional_in_embedded_system

How To Read GPIO module From TRM



LEARN-IN-DEPTH
Be professional in embedded system

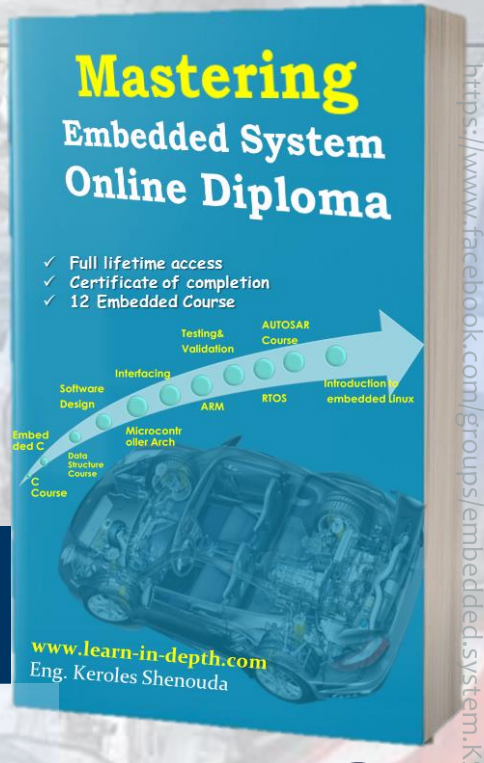
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

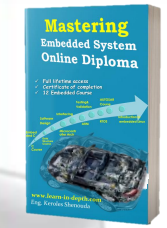
#Be_professional_in_embedded_system

Figure the GPIO Ports for SoC and For Board



LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



For Stm32F103X SoC (TRM)

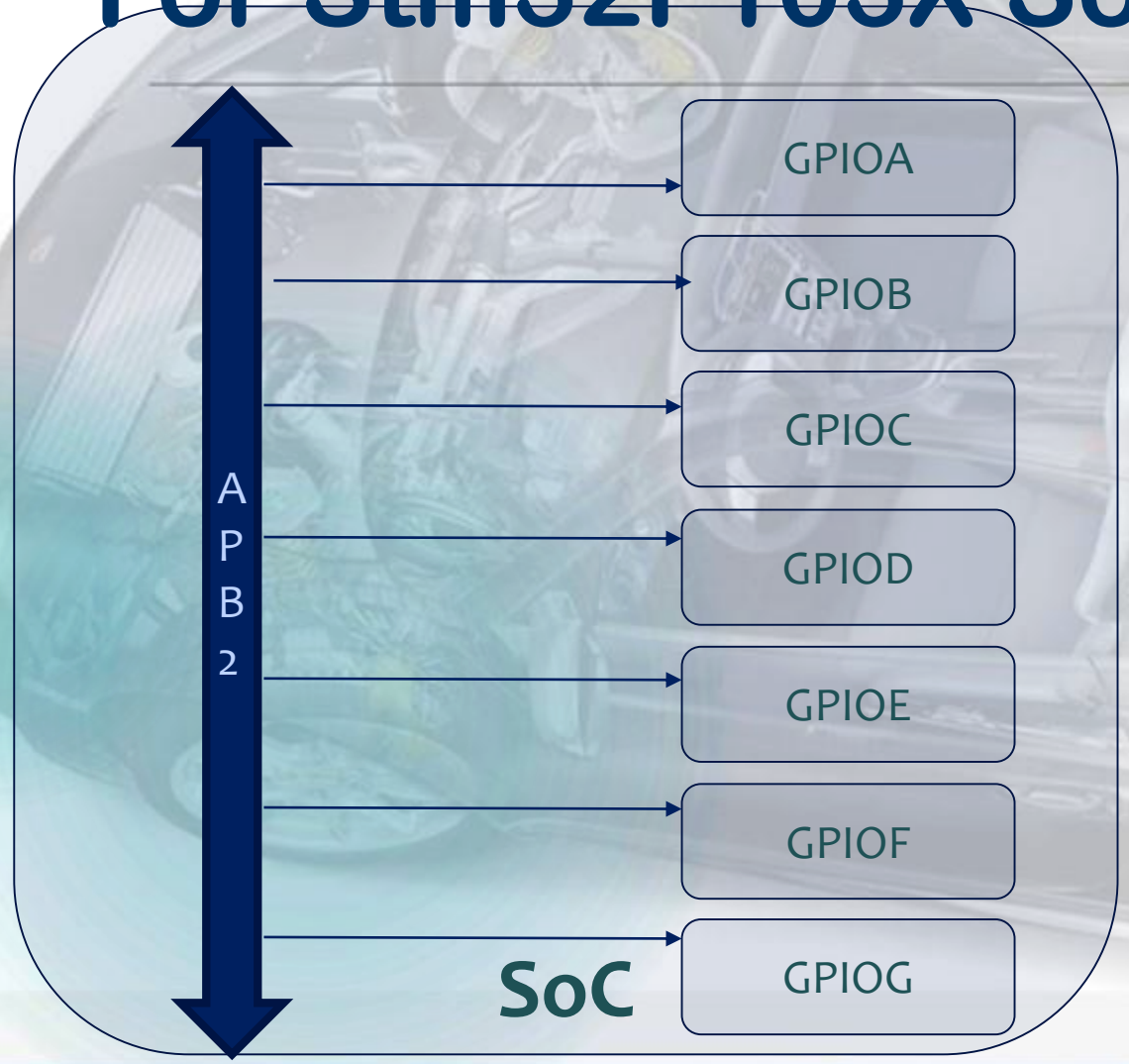
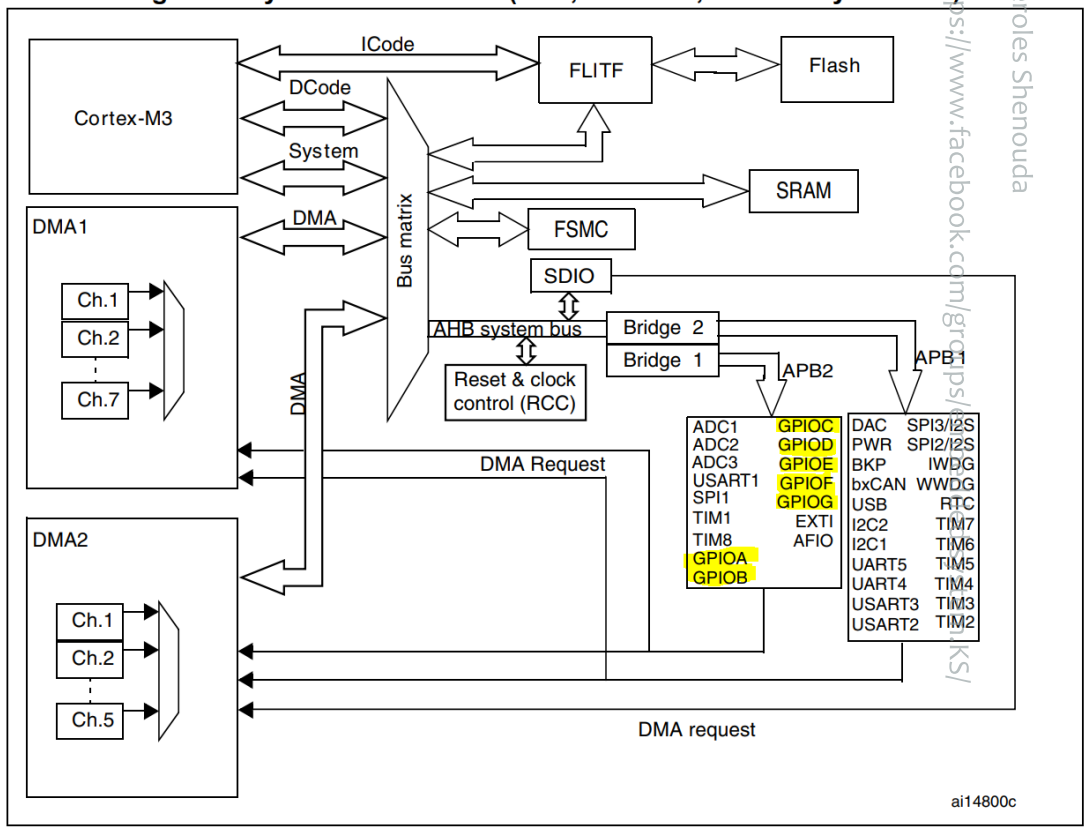
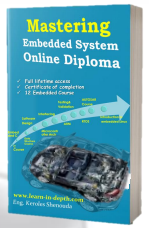


Figure 1. System architecture (low-, medium-, XL-density devices)



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



For Stm32F103X SoC (TRM)

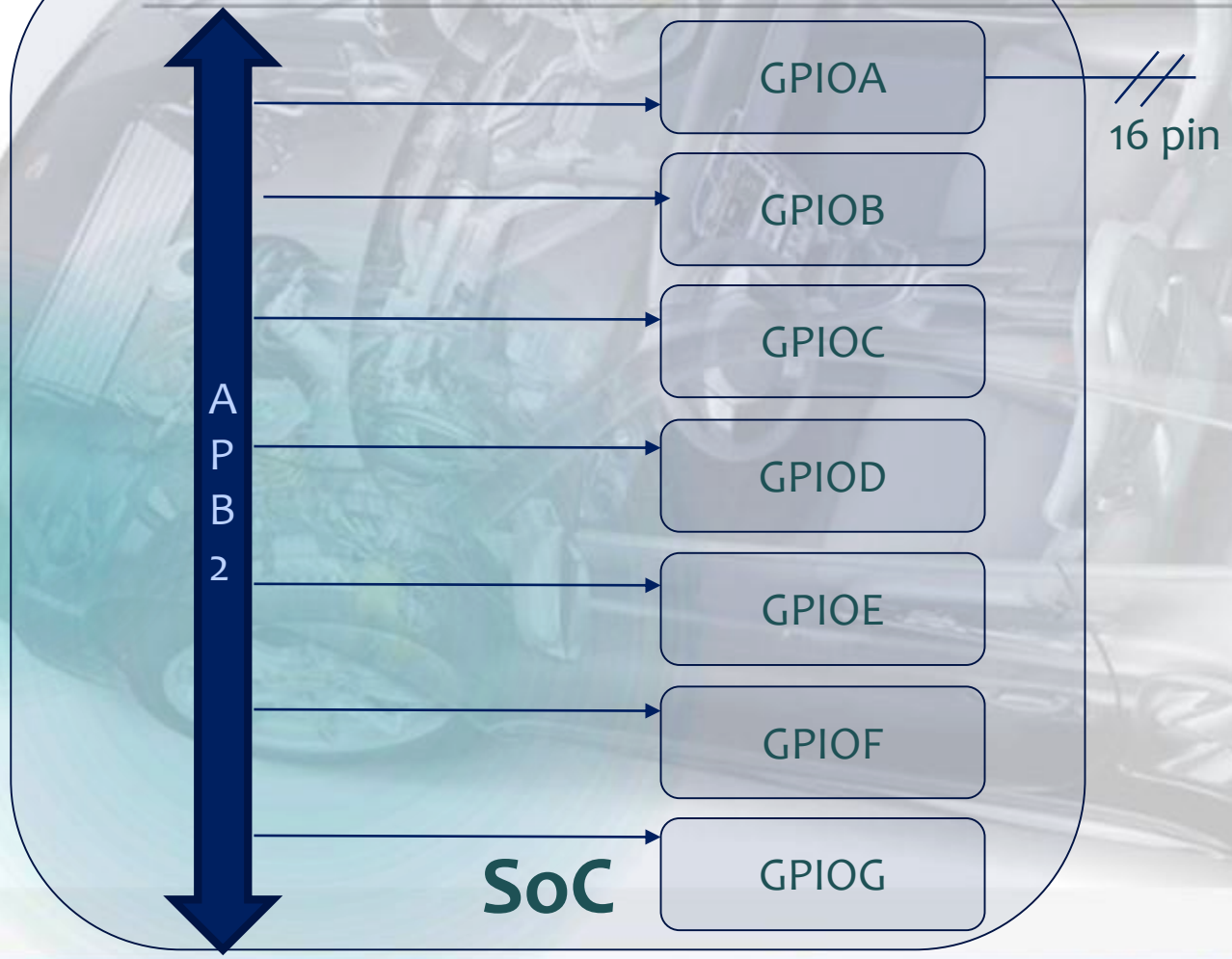
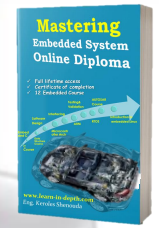


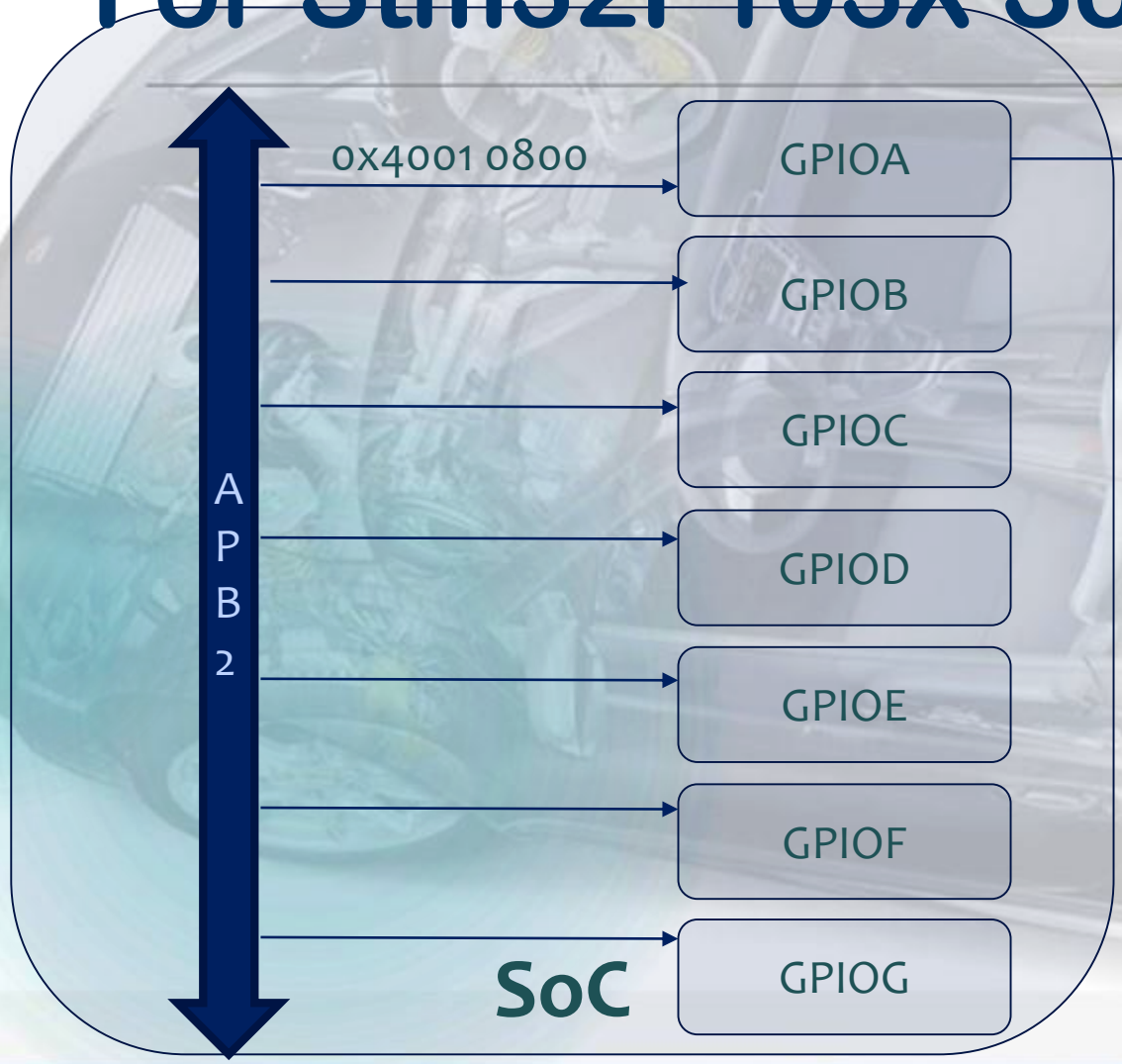
Table 59. GPIO register map and reset values

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	GPIOx_CRL	CNF7 [1:0]		MODE7 [1:0]		CNF6 [1:0]		MODE6 [1:0]		CNF5 [1:0]		MODE5 [1:0]		CNF4 [1:0]		MODE4 [1:0]		CNF3 [1:0]		MOD E3 [1:0]		CNF2 [1:0]		MODE2 [1:0]		CNF1 [1:0]		MOD E1 [1:0]		CNF0 [1:0]		MODE0 [1:0]	
	Reset value	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
0x04	GPIOx_CRH	CNF15 [1:0]		MODE15 [1:0]		CNF14 [1:0]		MODE14 [1:0]		CNF13 [1:0]		MODE13 [1:0]		CNF12 [1:0]		MODE12 [1:0]		CNF11 [1:0]		MOD E11 [1:0]		CNF10 [1:0]		MODE10 [1:0]		CNF9 [1:0]		MOD E9 [1:0]		CNF8 [1:0]		MODE8 [1:0]	
	Reset value	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
0x08	GPIOx_IDR	Reserved																IDRy															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOx_ODR	Reserved																ODRy															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	GPIOx_BSRR	BR[15:0]																BSR[15:0]															
	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
0x14	GPIOx_BRR	Reserved																BR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	GPIOx_LCKR	Reserved															LCKK	LCK[15:0]															
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

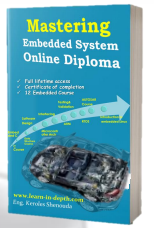
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



For Stm32F103X SoC (TRM)



0x4001 2000 - 0x4001 23FF	GPIO Port G
0x4001 1C00 - 0x4001 1FFF	GPIO Port F
0x4001 1800 - 0x4001 1BFF	GPIO Port E
0x4001 1400 - 0x4001 17FF	GPIO Port D
0x4001 1000 - 0x4001 13FF	GPIO Port C
0x4001 0C00 - 0x4001 0FFF	GPIO Port B
0x4001 0800 - 0x4001 0BFF	GPIO Port A



For Stm32f103c6 Board (Specs)

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

Low-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

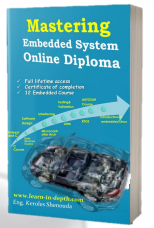
High-density devices are STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

XL-density devices are STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 768 Kbytes and 1 Mbyte.

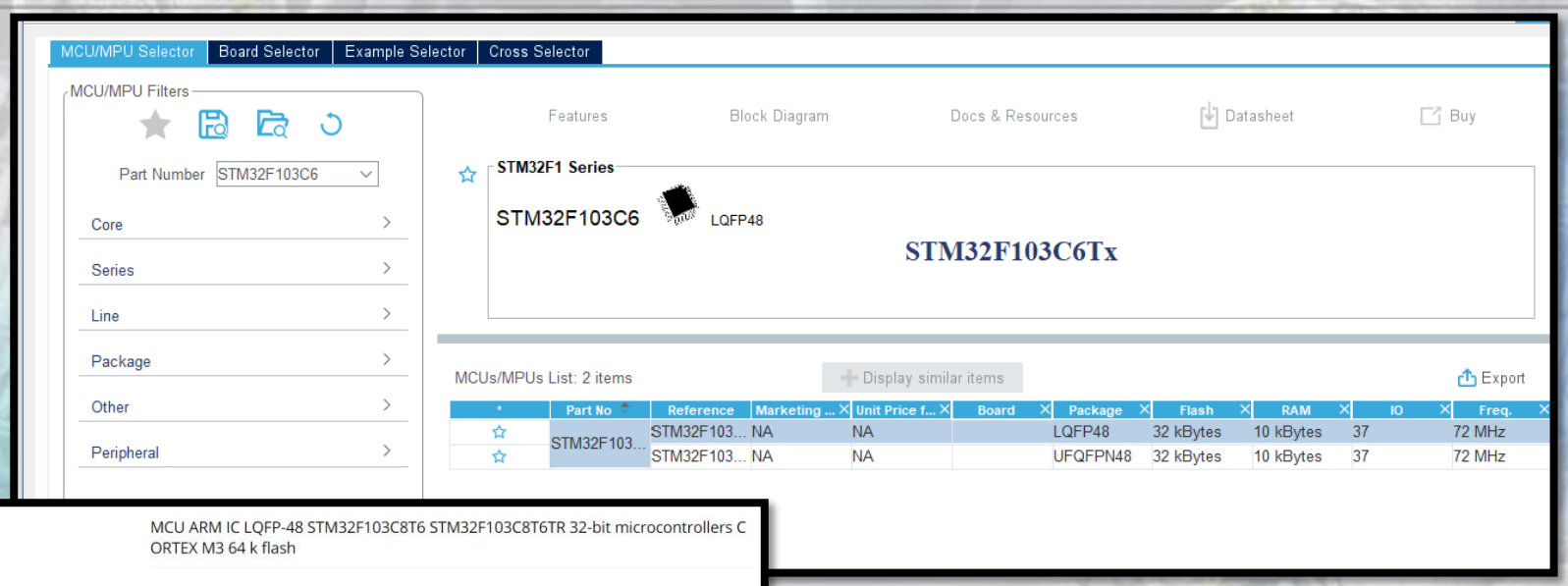
Connectivity line devices are STM32F105xx and STM32F107xx microcontrollers.

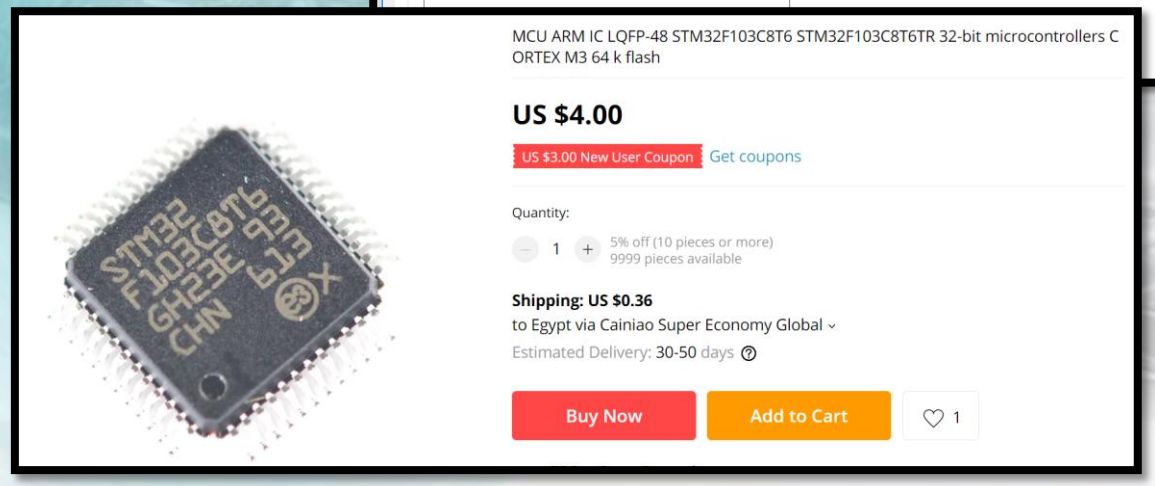
This section applies to the whole STM32F10xxx family, unless otherwise specified.

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

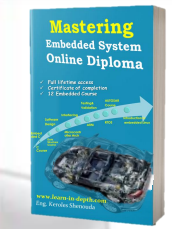


For Stm32f103c6 MCU





https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

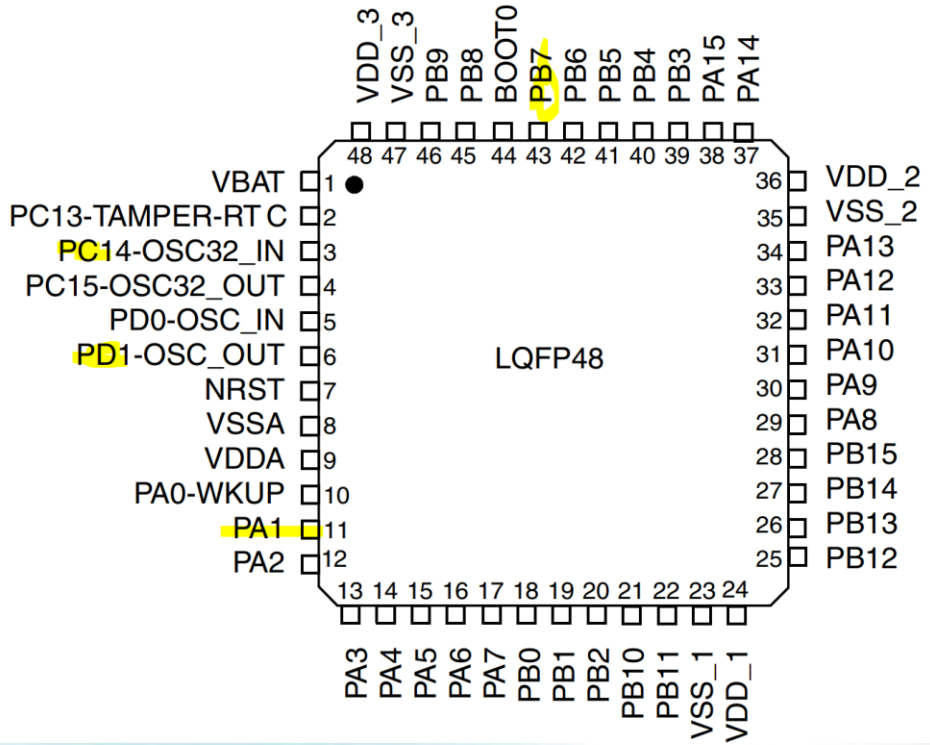


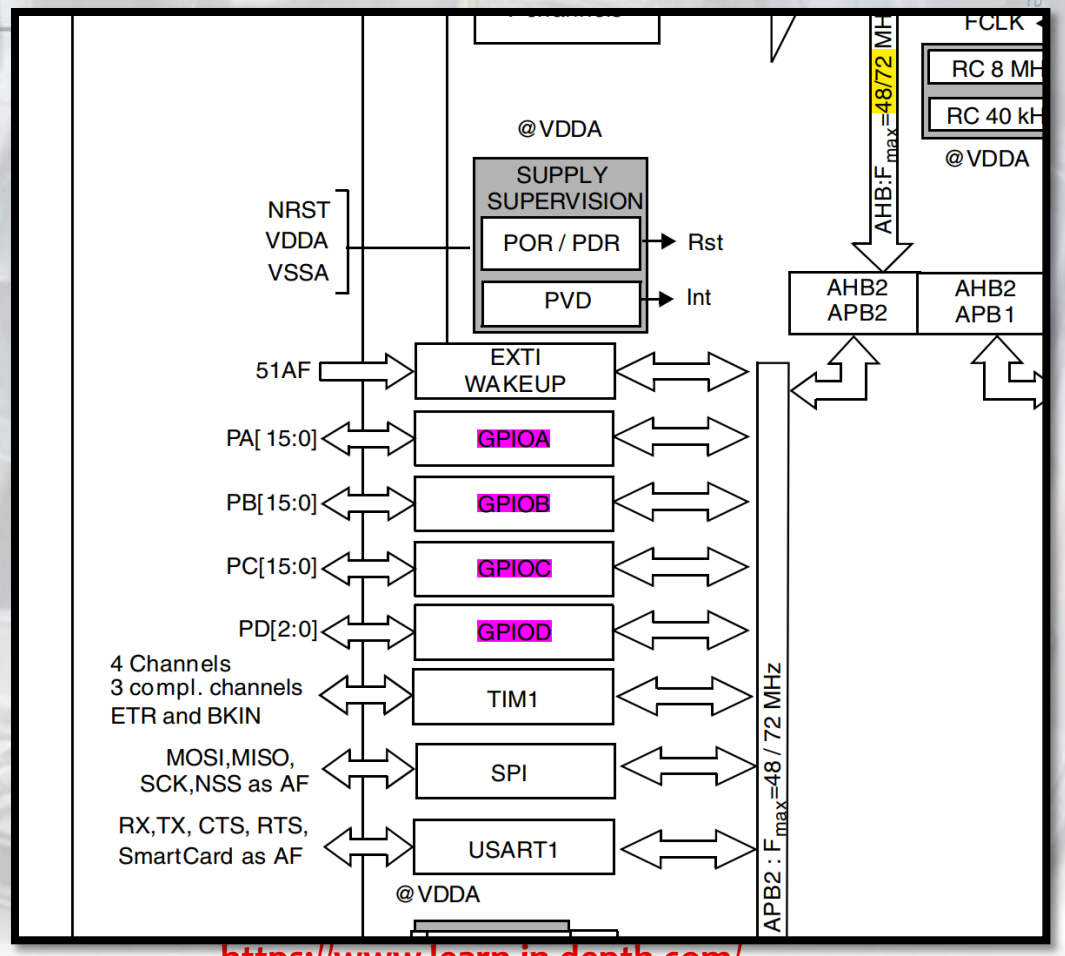
#LEARN_IN DEPTH

#Be_professional_in embedded_system

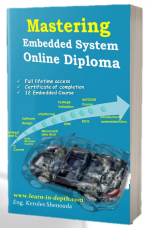
For Stm32f103c6 MCU

Figure 5. STM32F103xx performance line LQFP48 pinout

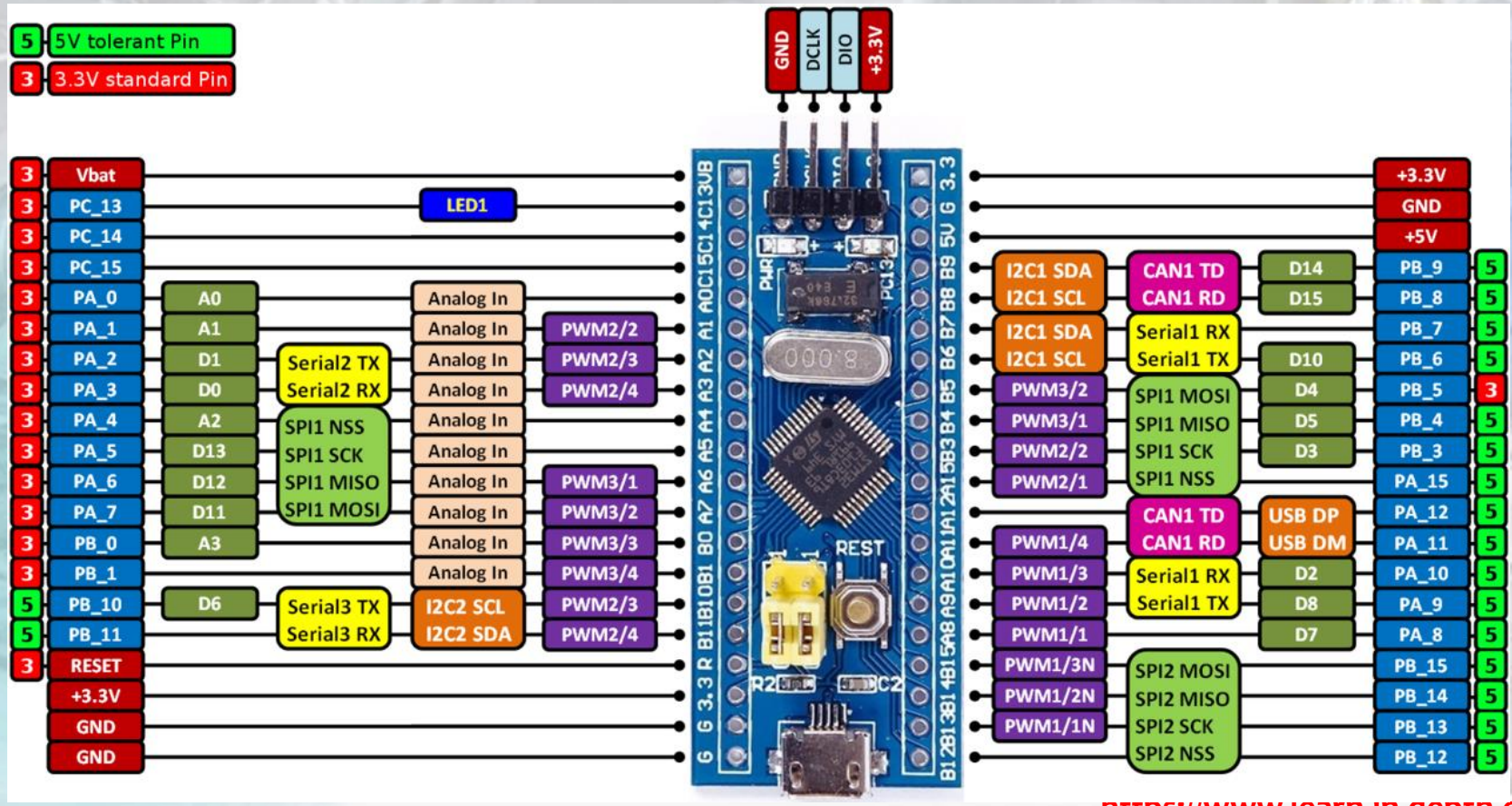




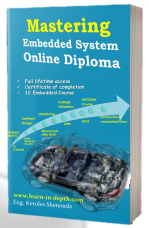
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



For Board

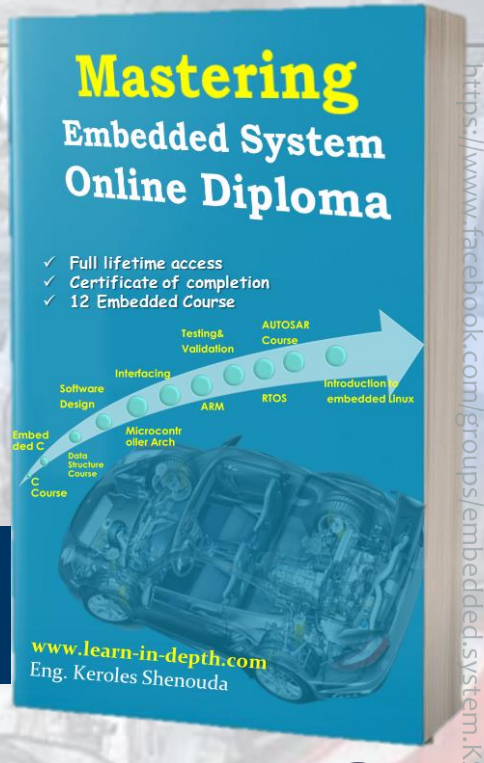


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

#Be_professional_in_embedded_system



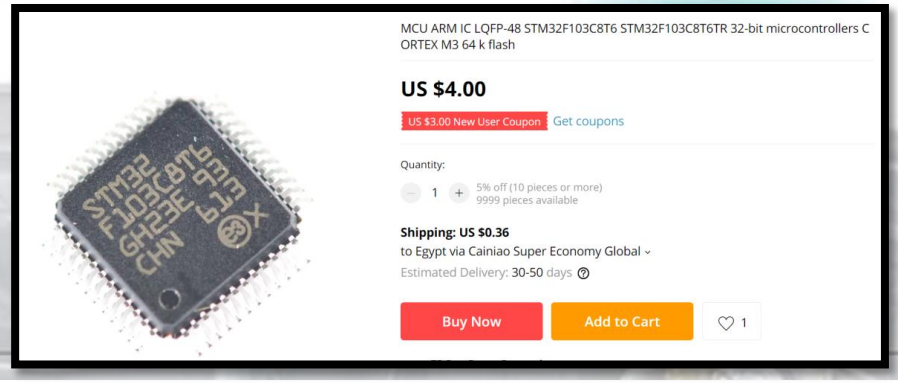
SoC / Board IOs for Different SoCs/Boards

STM32F103XX

LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

SoC IOs



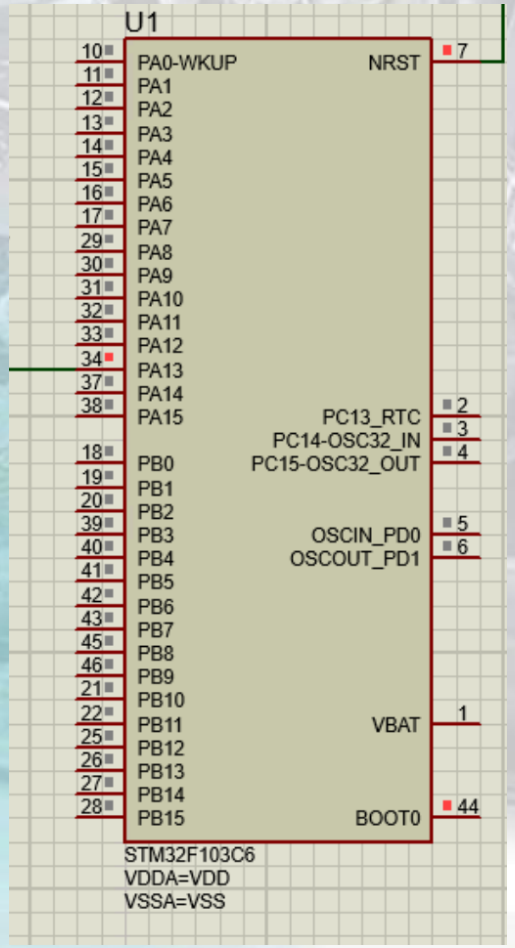
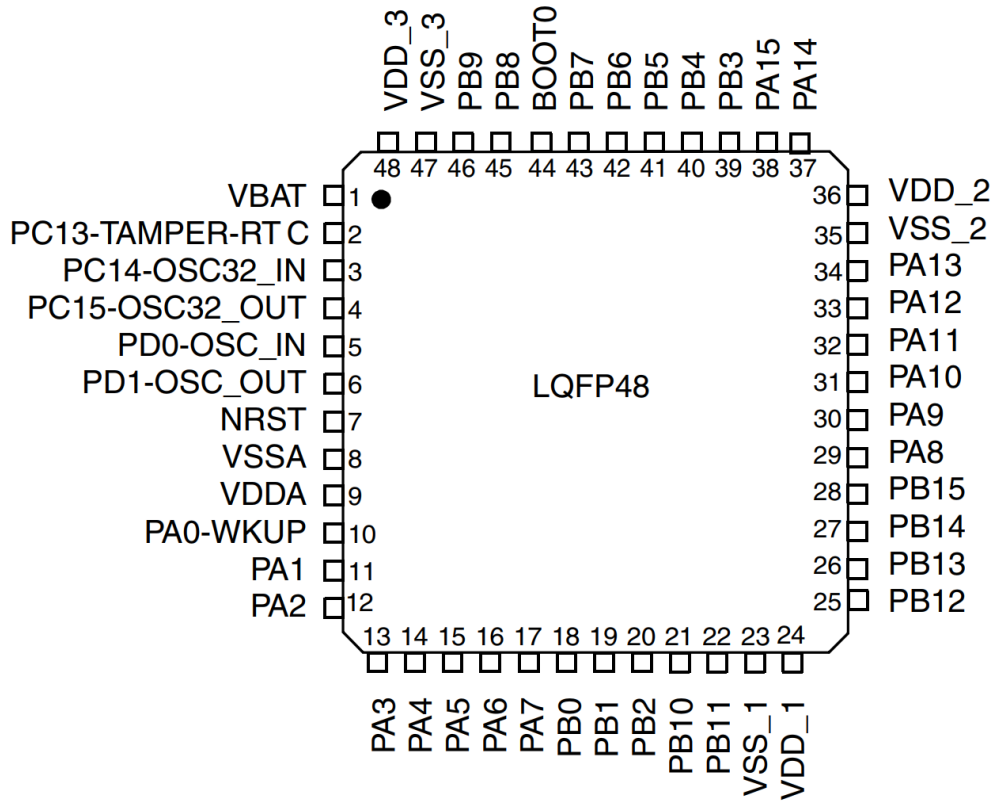
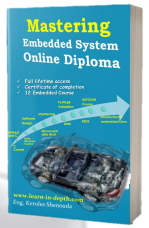


Figure 5. STM32F103xx performance line LQFP48 pinout



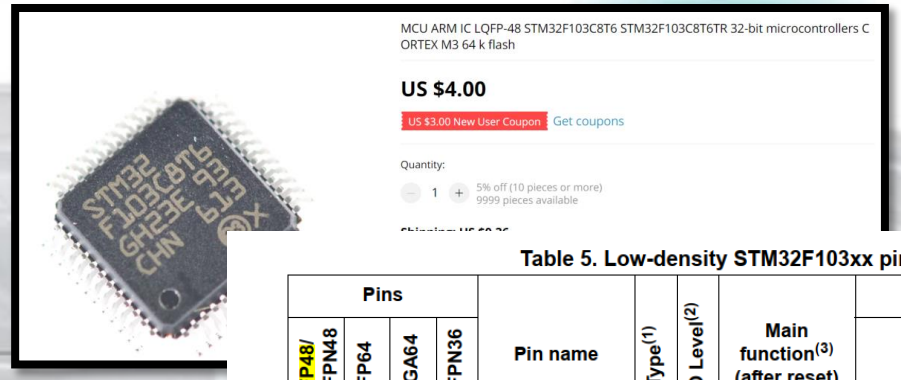


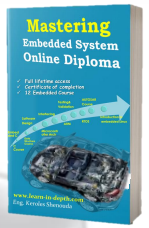
#LEARN_IN DEPTH

#Be_professional_in_embedded_system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

SoC IOs





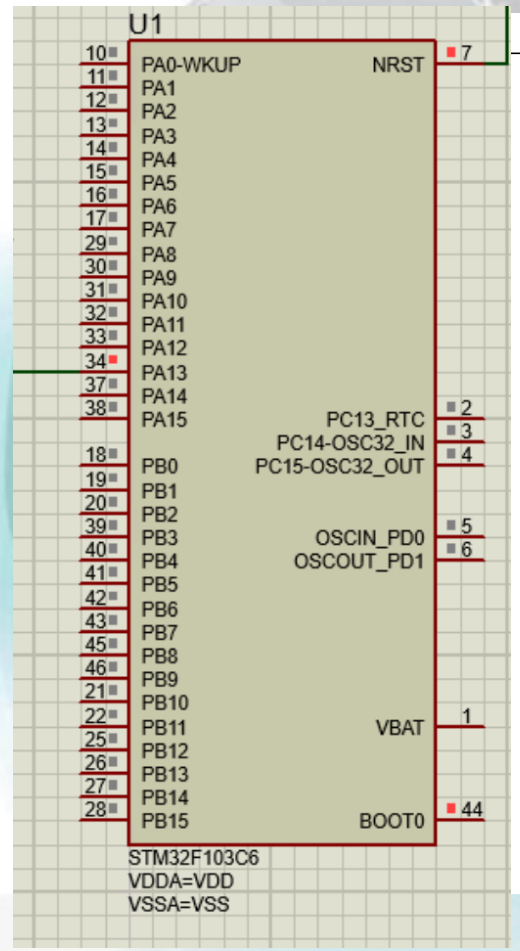


Figure 5. STM32F103xx performance line LQFP

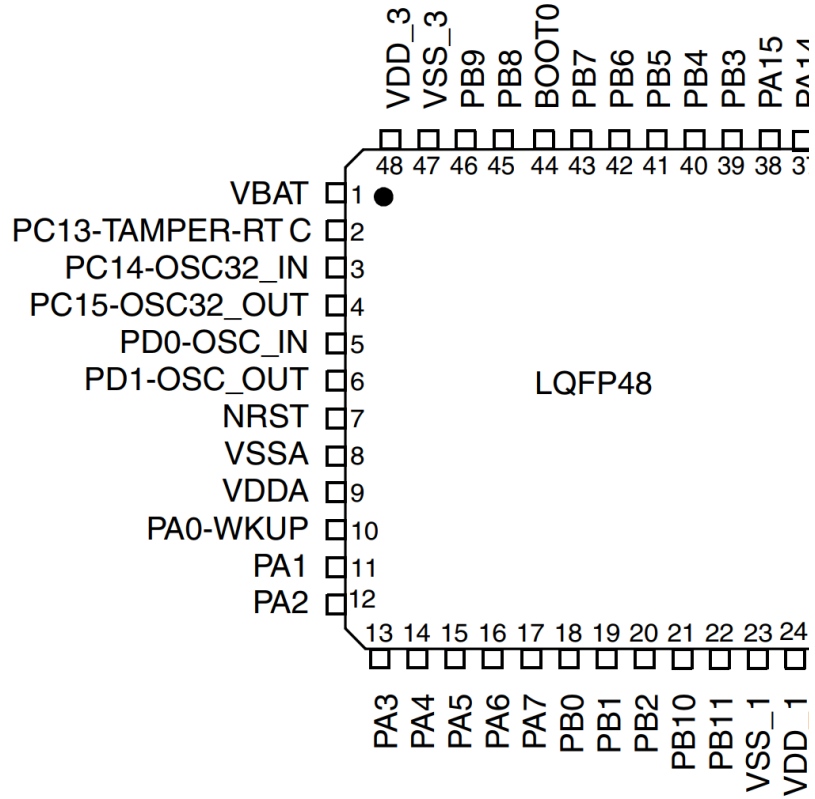
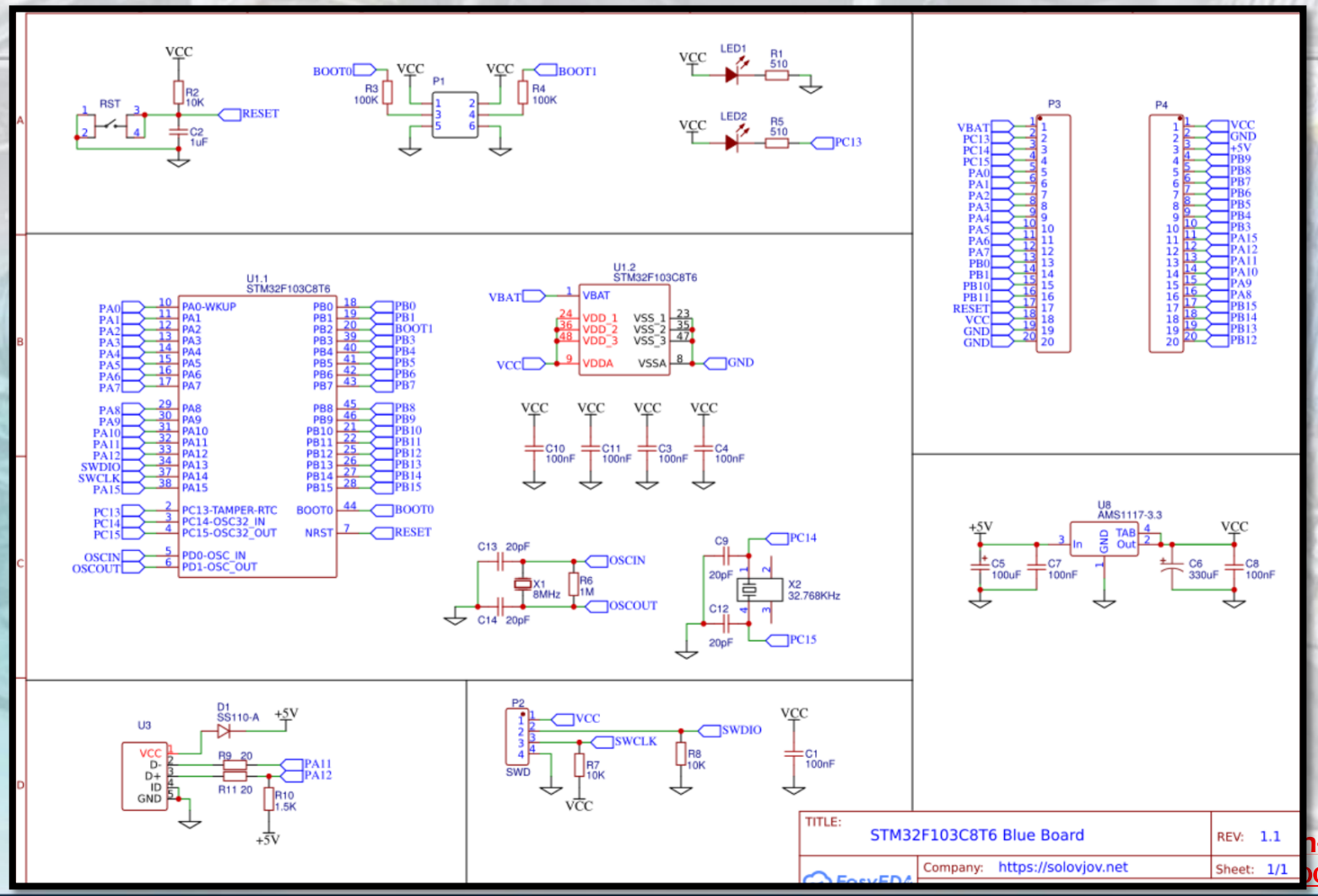
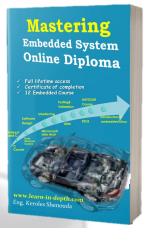


Table 5. Low-density STM32F103xx pin definitions

Pins				Pin name	Type(1)	I / O Level(2)	Main function(3) (after reset)	Alternate functions(4)	
LQFP48/ UFQFPN48	LQFP64	TFBGA64	VFQFPN36					Default	Remap
1	1	B2	-	V_{BAT}	S	-	V_{BAT}	-	-
2	2	A2	-	PC13-TAMPER-RTC(5)	I/O	-	PC13(6)	TAMPER-RTC	-
3	3	A1	-	PC14-OSC32_IN(5)	I/O	-	PC14(6)	OSC32_IN	-
4	4	B1	-	PC15-OSC32_OUT(5)	I/O	-	PC15(6)	OSC32_OUT	-
5	5	C1	2	OSC_IN	I	-	OSC_IN	-	PD0(7)
6	6	D1	3	OSC_OUT	O	-	OSC_OUT	-	PD1(7)
7	7	E1	4	NRST	I/O	-	NRST		-
-	8	E3	-	PC0	I/O	-	PC0	ADC12_IN10	-
-	9	E2	-	PC1	I/O	-	PC1	ADC12_IN11	-
-	10	F2	-	PC2	I/O	-	PC2	ADC12_IN12	-
-	11	-	-	PC3	I/O	-	PC3	ADC12_IN13	-
-	-	G1	-	V_{REF+}(8)	S	-	V_{REF+}	-	-
8	12	F1	5	V_{SSA}	S	-	V_{SSA}	-	-
9	13	H1	6	V_{DDA}	S	-	V_{DDA}	-	-
10	14	G2	7	PA0-WKUP	I/O	-	PA0	WKUP/USART2_CTS/ ADC12_IN0/ TIM2_CH1_ETR(9)	-
11	15	H2	8	PA1	I/O	-	PA1	USART2_RTS/ ADC12_IN1/ TIM2_CH2(9)	-
12	16	F3	9	PA2	I/O	-	PA2	USART2_TX/ ADC12_IN2/ TIM2_CH3(9)	-
13	17	G3	10	PA3	I/O	-	PA3	USART2_RX/ ADC12_IN3/TIM2_CH4(9)	-
-	18	C2	-	V_{SS_4}	S	-	V_{SS_4}	-	-
-	19	D2	-	V_{DD_4}	S	-	V_{DD_4}	-	-
								SPI1_NSS(9)/	

Board IOs



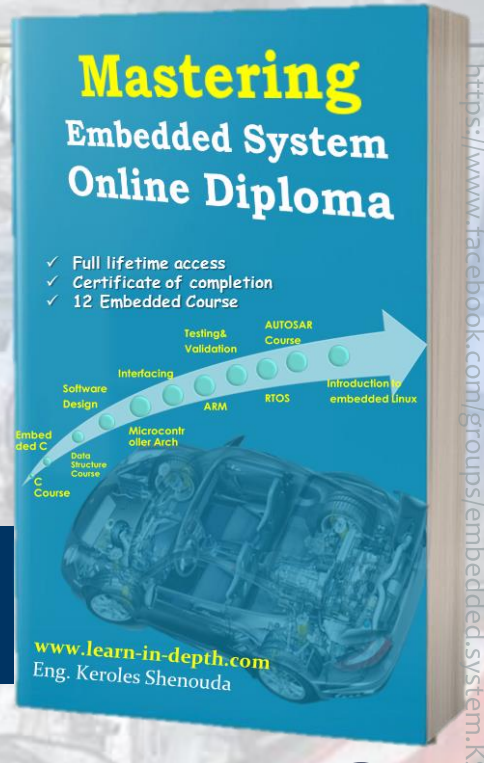


#LEARN_IN DEPTH

#Be_professional_in_embedded_system

SoC / Board IOs for Different SoCs/Boards

ATMEGA32



LEARN-IN-DEPTH

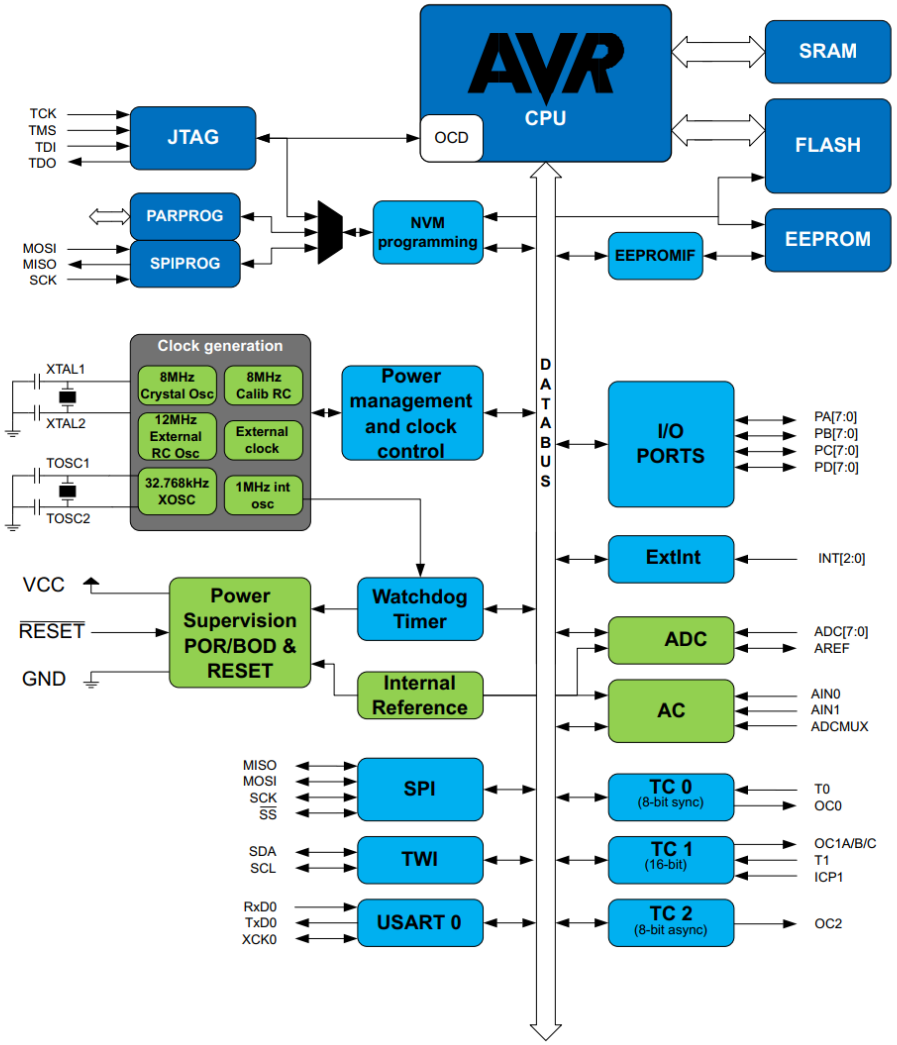
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

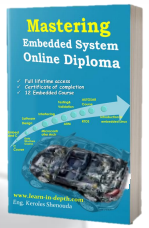
Atmega32 MCU

4. Block Diagram

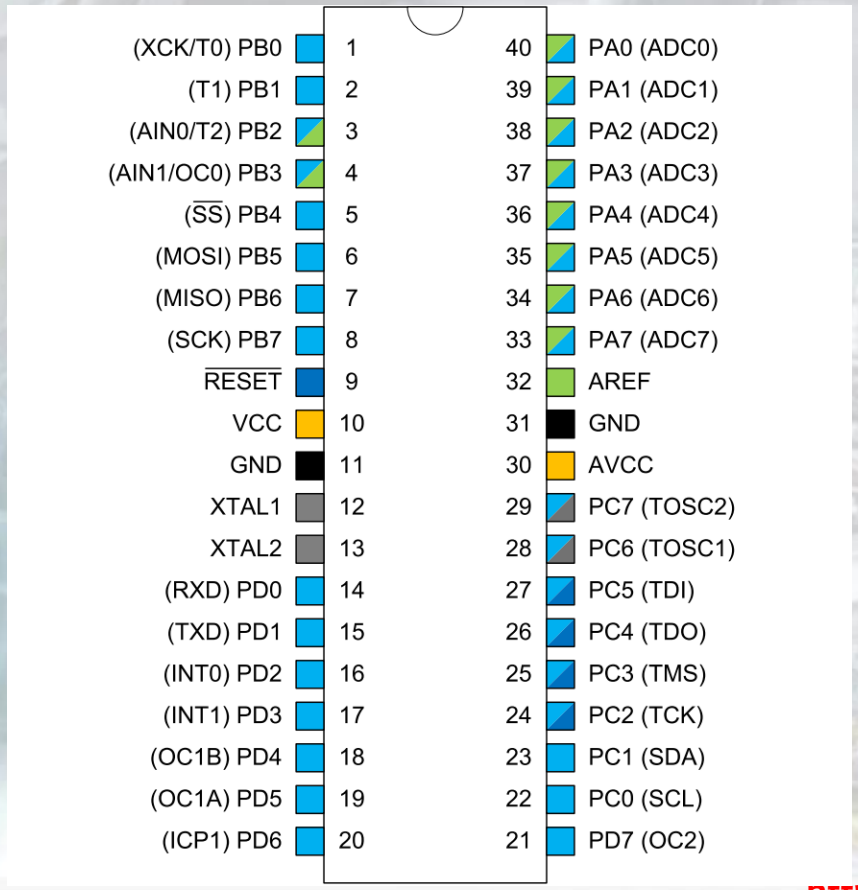
Figure 4-1 Block Diagram



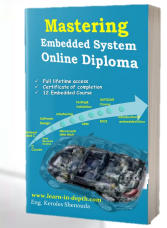
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



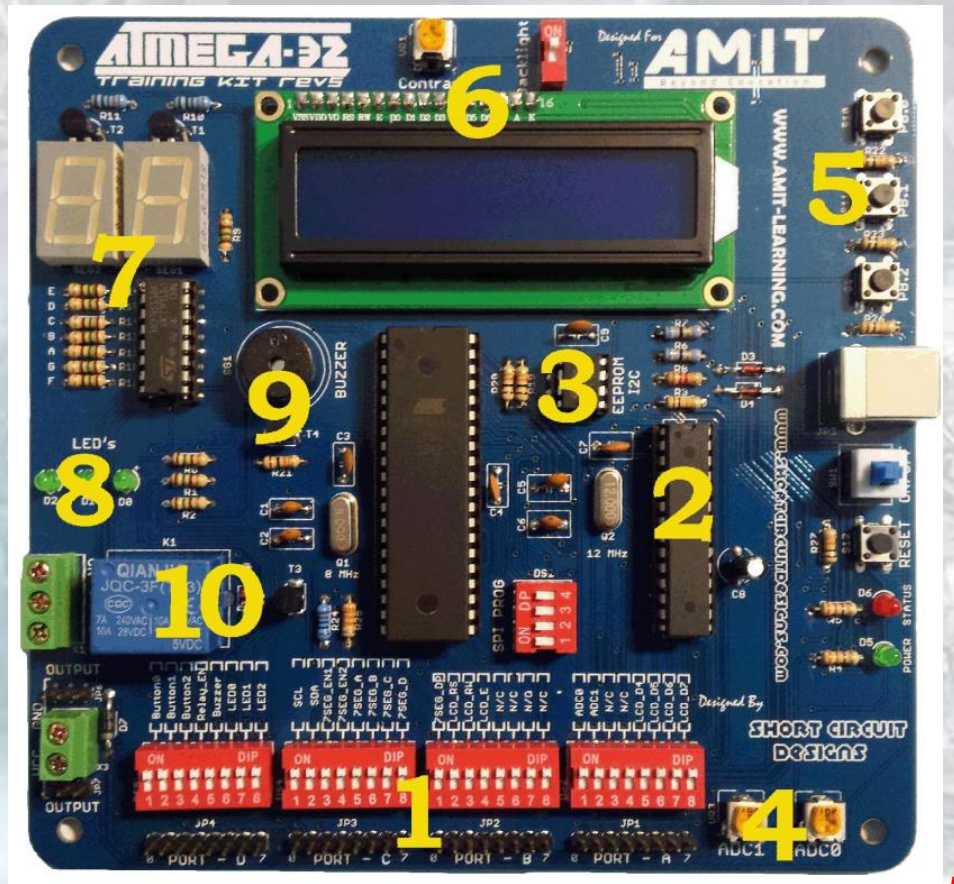
Atmega32 MCU



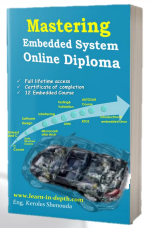
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



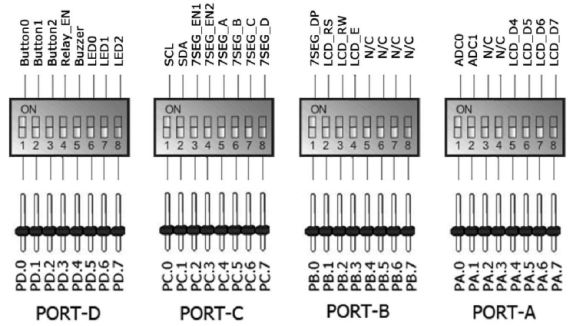
Atmega32 Board



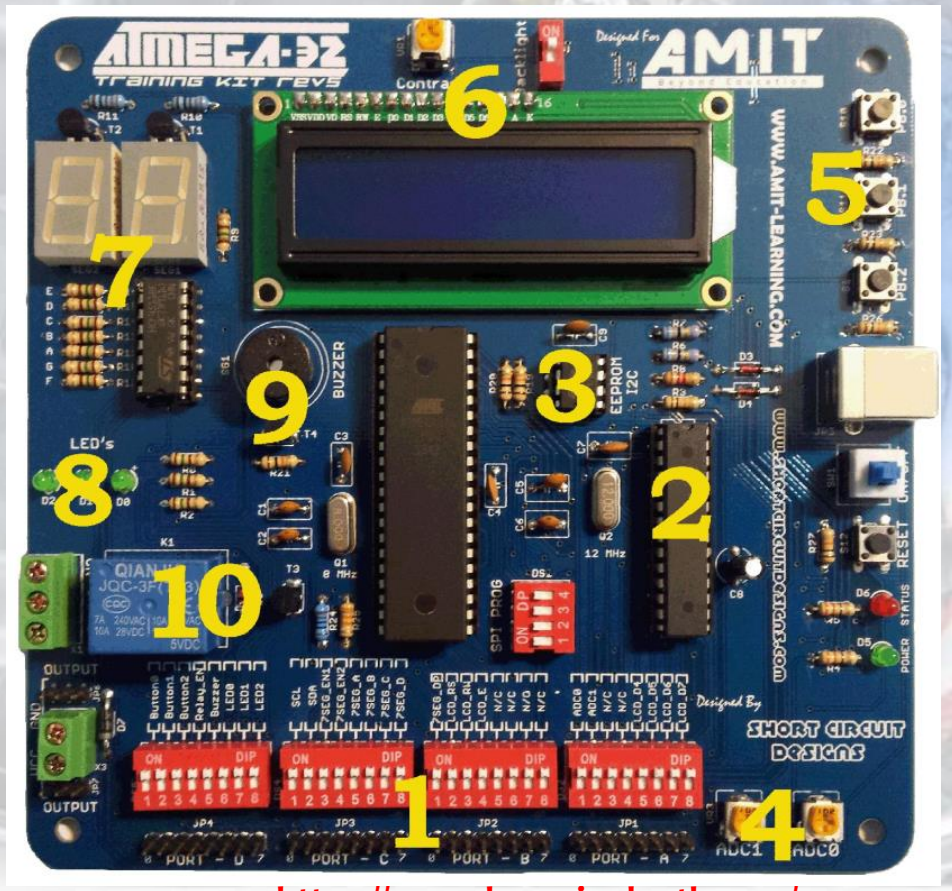
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



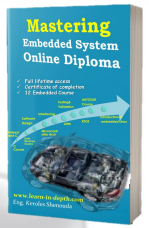
Atmega32 Board



- **Three-Bit LED**
  - LED0→PORTD .5
  - LED1→PORTD .6
  - LED2→PORTD .7
- **Buzzer**
  - Buzzer→PORTD .4
- **RELAY**
  - Relay_EN→PORTD .3
- **Pushbutton**
  - Button0→PORTD.0
  - Button1→ PORTD.1
  - Button2→ PORTD.2
- **7 Segment Display**
  - DATA LINES:
  - 7SEG_A→PORTC .4
  - 7SEG_B→PORTC .5
  - 7SEG_C→PORTC .6
  - 7SEG_D→PORTC .7
  - DECIMAL POINT:
  - 7SEG_DP→PORTB .0
  - ENABLE LINES
  - 7SEG_EN1→PORTC .2
  - 7SEG_EN2→PORTC .3



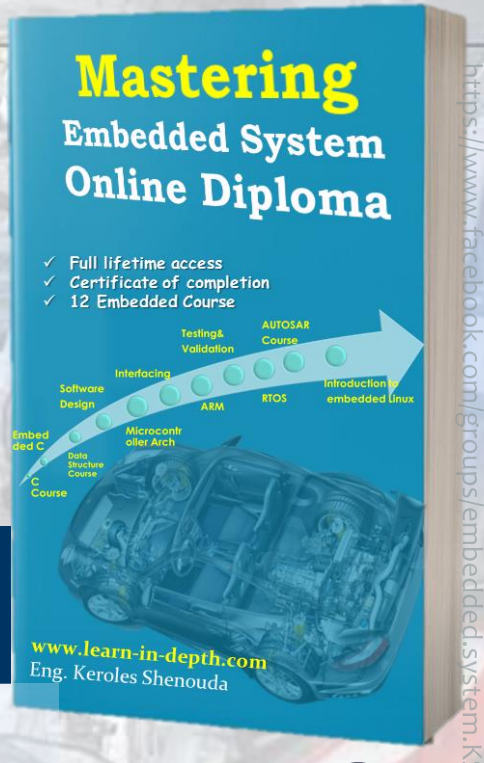
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

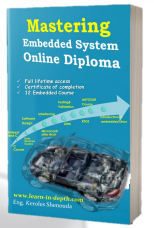
#Be_professional_in_embedded_system

Read GPIO functional description from TRM



LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



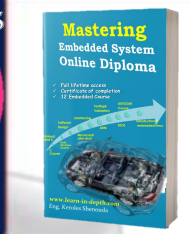
GPIO functional description

GPIO functional description

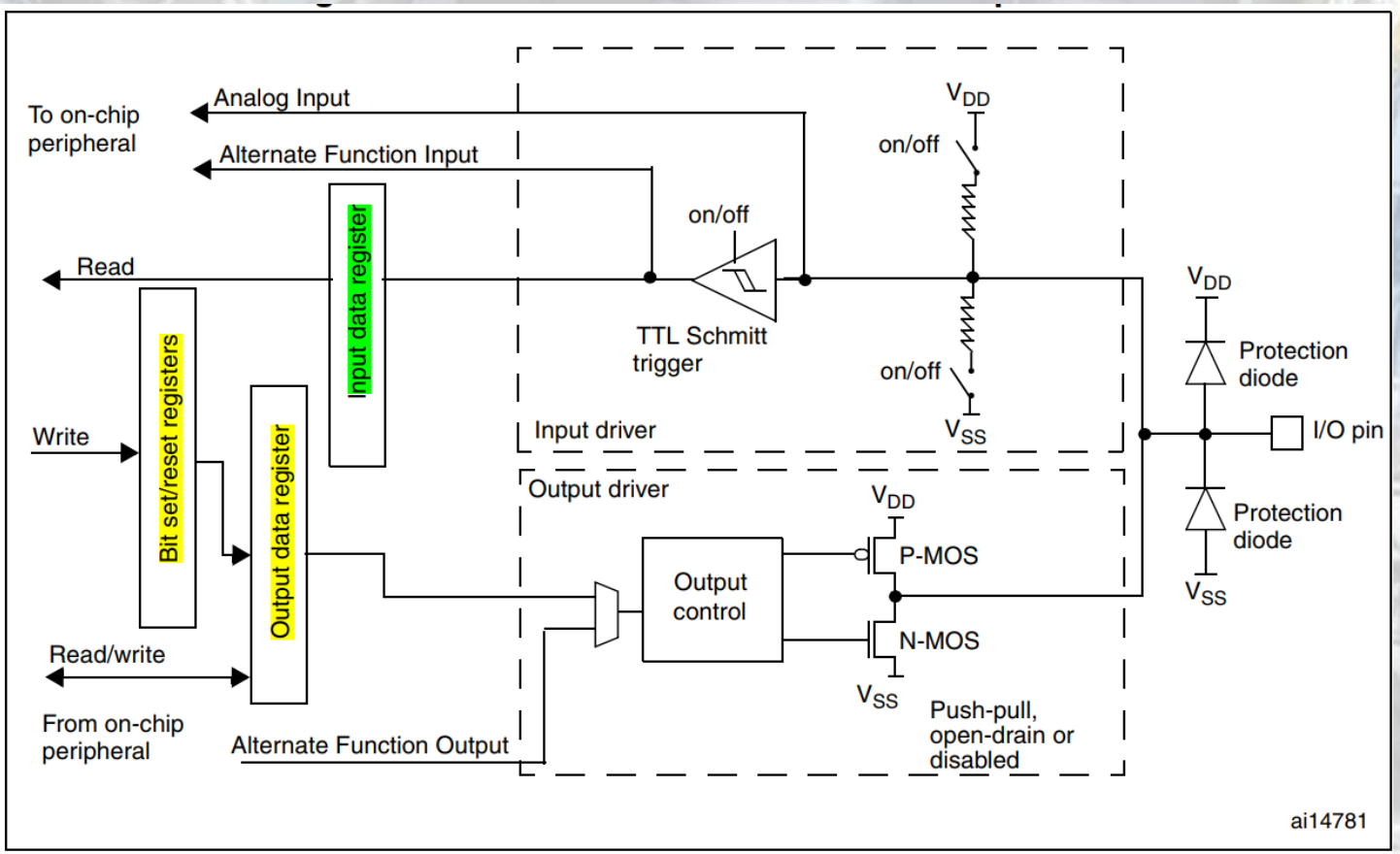
Each of the general-purpose I/O ports has two 32-bit configuration registers (GPIOx_CRL, GPIOx_CRH), two 32-bit data registers (GPIOx_IDR, GPIOx_ODR), a 32-bit set/reset register (GPIOx_BSRR), a 16-bit reset register (GPIOx_BRR) and a 32-bit locking register (GPIOx_LCKR).

Subject to the specific hardware characteristics of each I/O port listed in the *datasheet*, each port bit of the General Purpose IO (GPIO) Ports, can be individually configured by software in several modes:

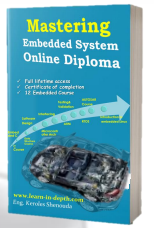
- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain
- Output push-pull
- Alternate function push-pull
- Alternate function open-drain



Basic structure of a standard I/O port bit



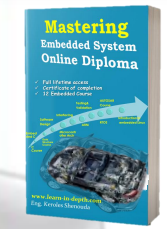
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Input configuration

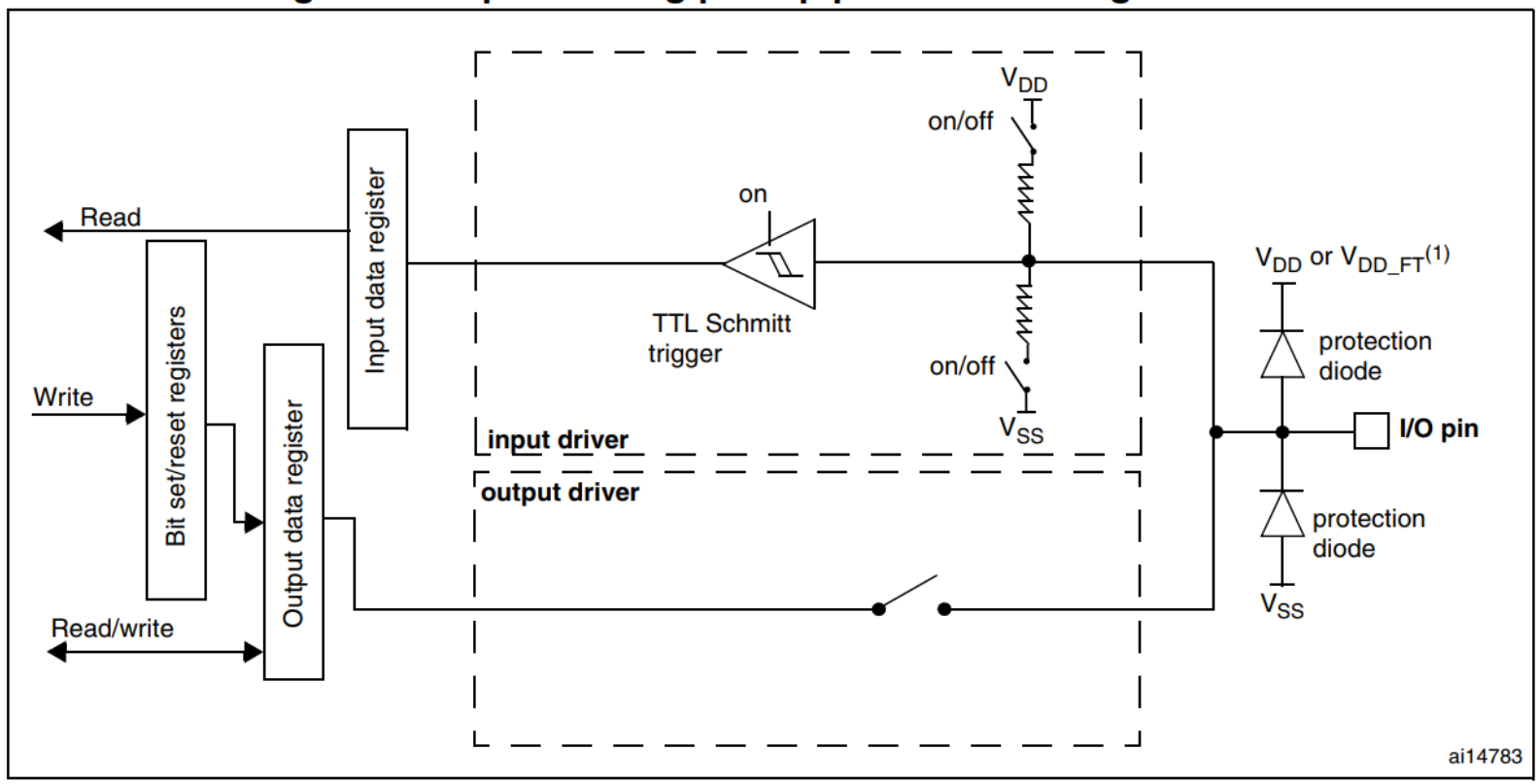
When the I/O Port is programmed as Input:

- The Output Buffer is disabled
- The Schmitt Trigger Input is activated
- The weak pull-up and pull-down resistors are activated or not depending on input configuration (pull-up, pull-down or floating):
- The data present on the I/O pin is sampled into the Input Data Register every APB2 clock cycle
- A read access to the Input Data Register obtains the I/O State.

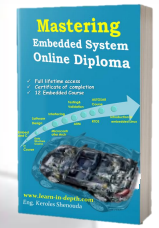


Input configuration

Figure 15. Input floating/pull up/pull down configurations

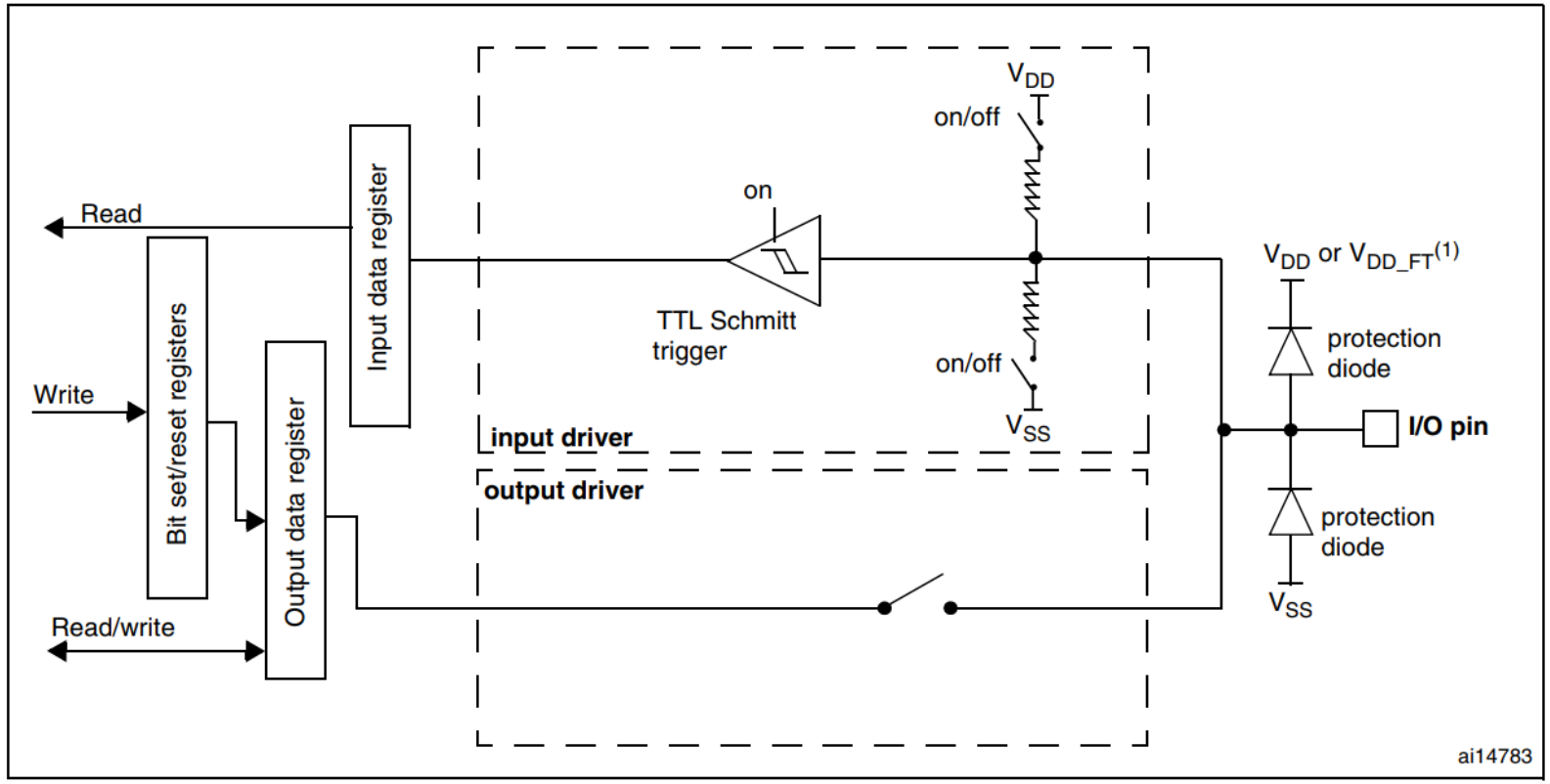


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

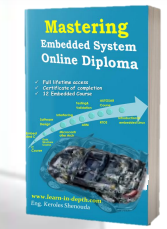


Input configuration floating

Figure 15. Input floating/pull up/pull down configurations

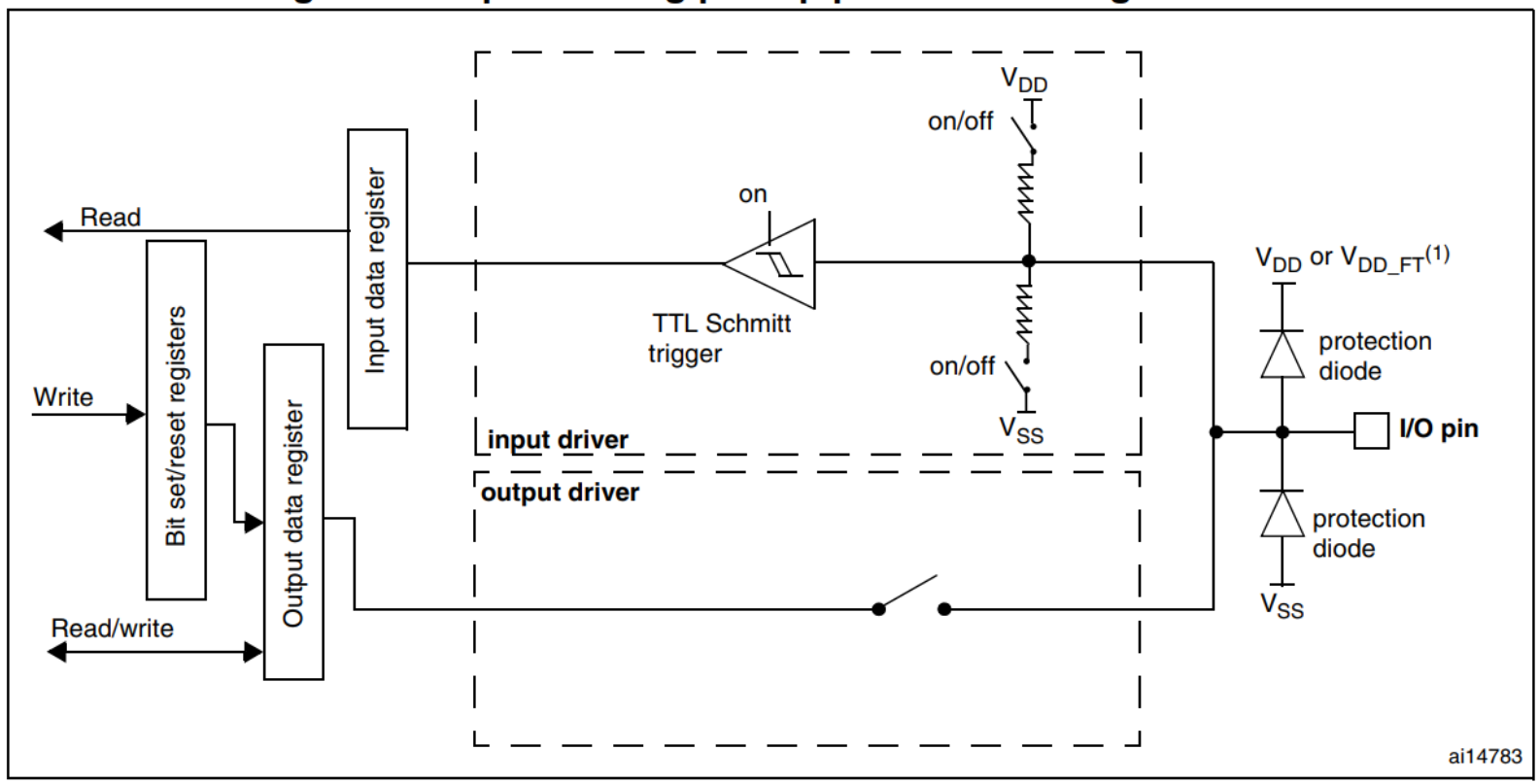


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Input configuration floating

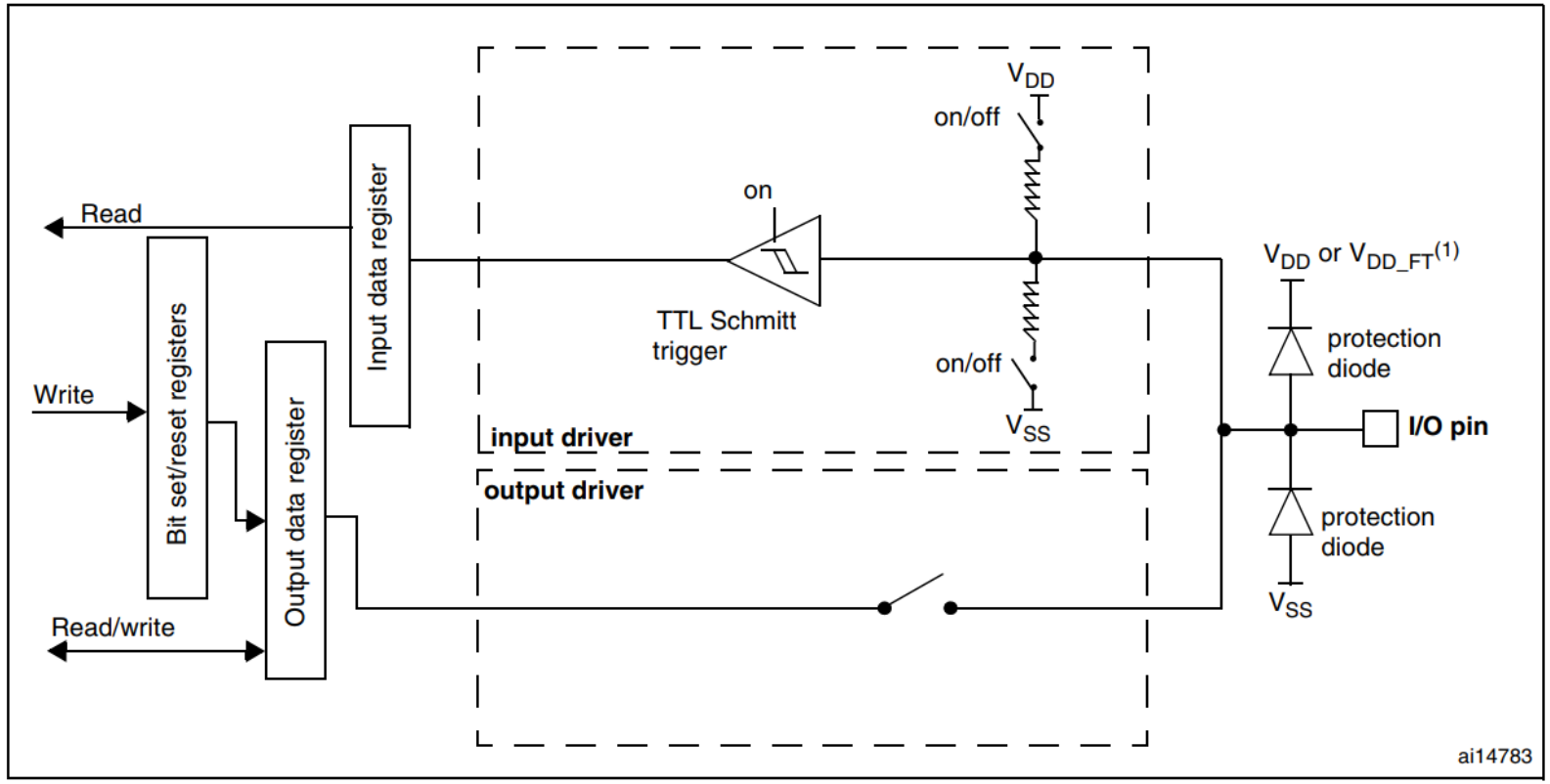
Figure 15. Input floating/pull up/pull down configurations



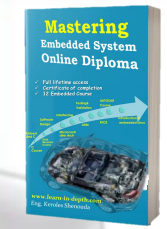
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Input configuration pull up

Figure 15. Input floating/pull up/pull down configurations

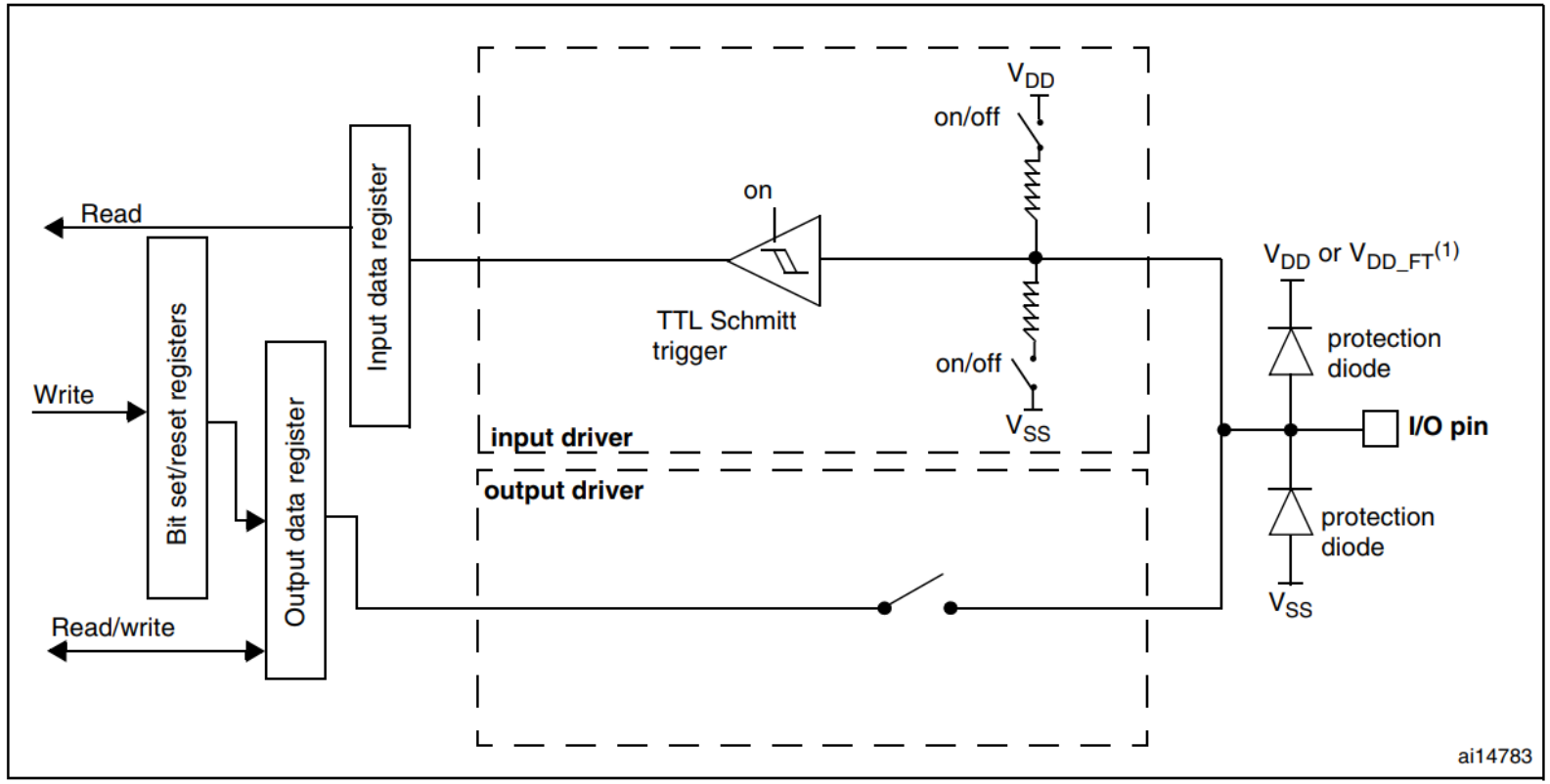


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

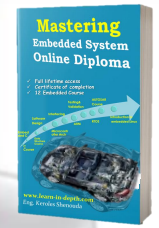


Input configuration pull up

Figure 15. Input floating/pull up/pull down configurations

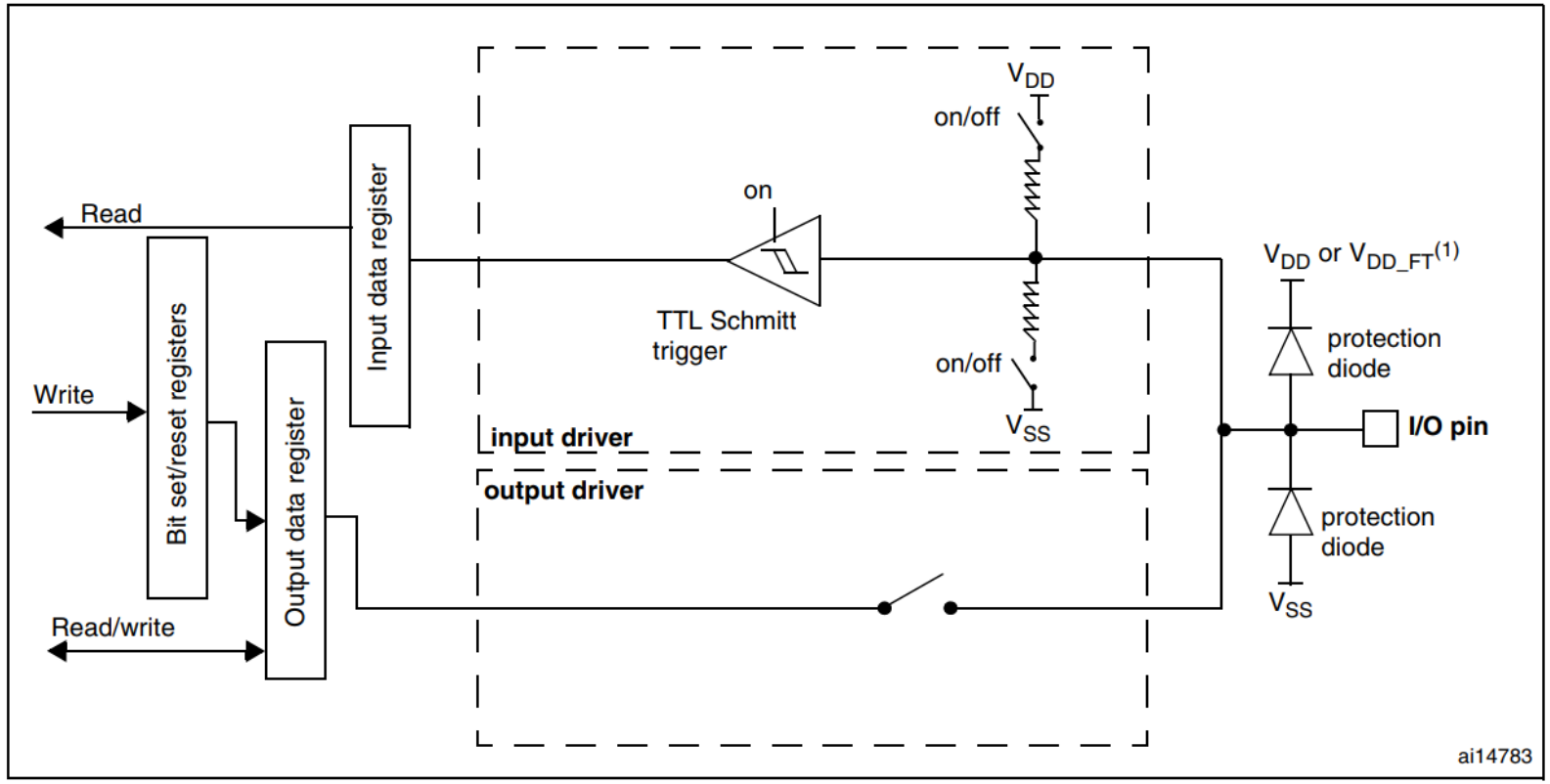


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

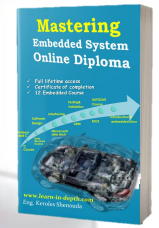


Input configuration pull down

Figure 15. Input floating/pull up/pull down configurations

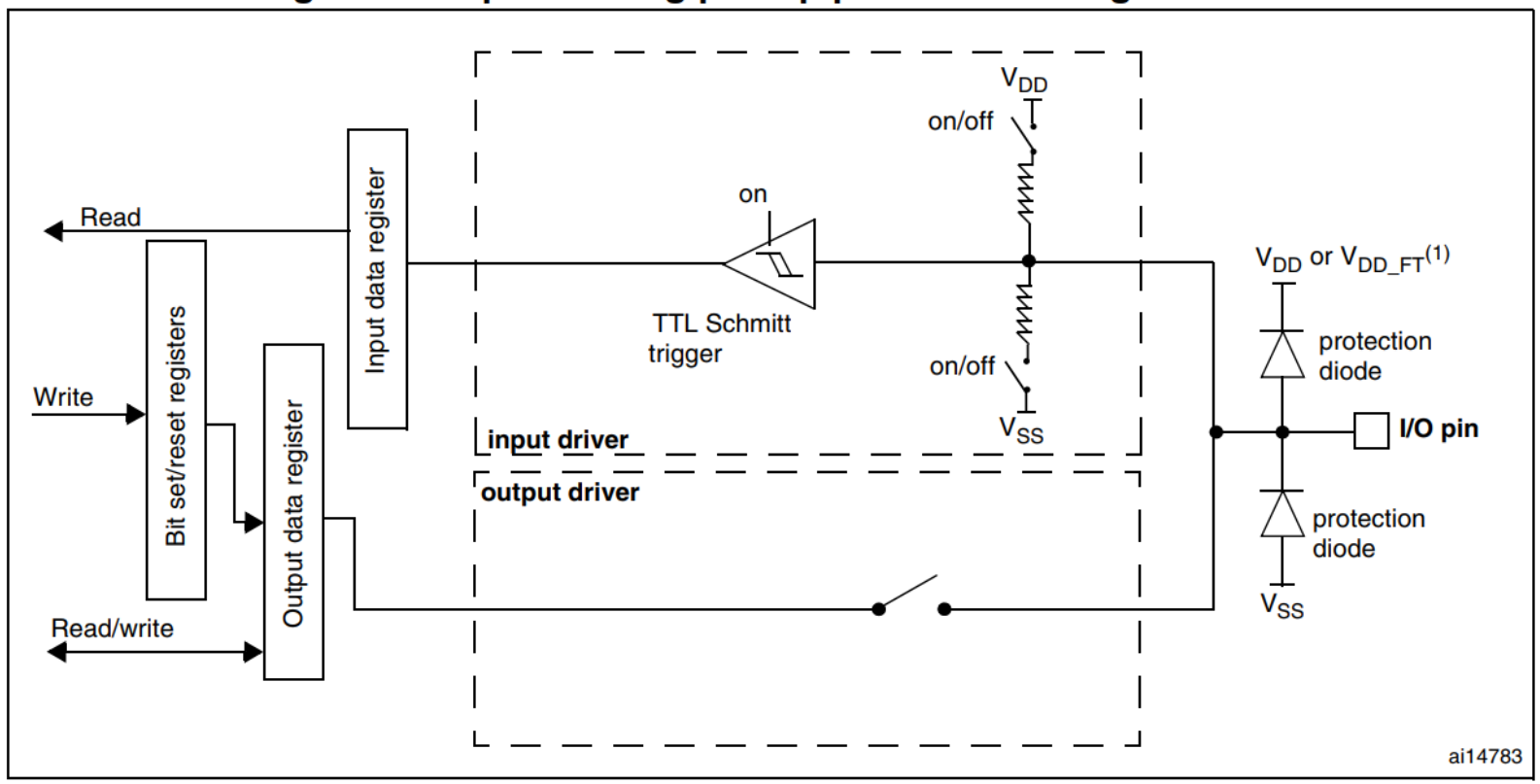


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

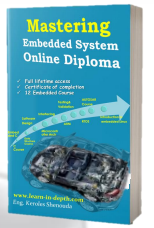


Input configuration pull down

Figure 15. Input floating/pull up/pull down configurations



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

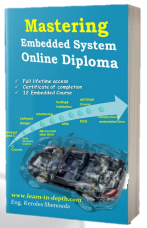


Output configuration

Output configuration

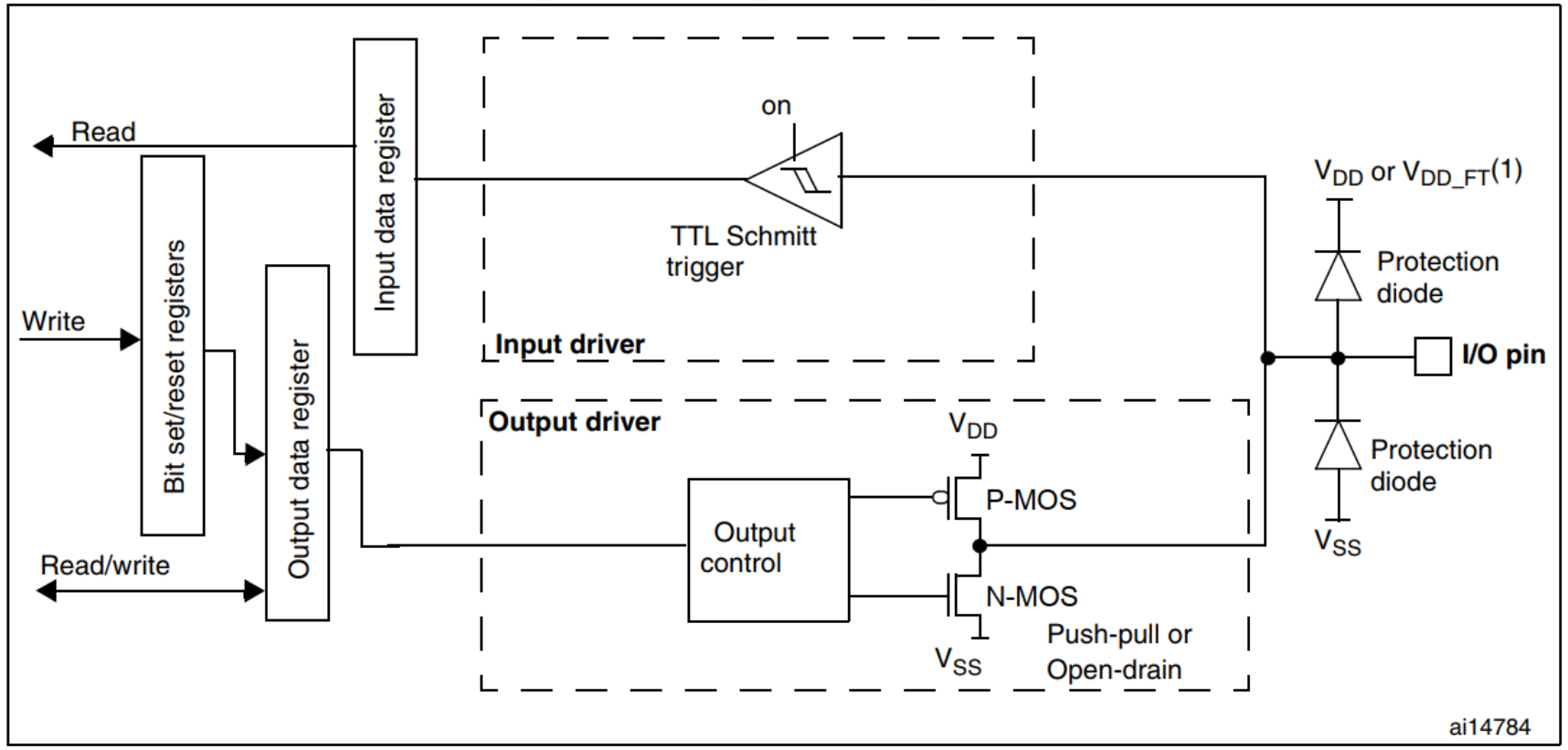
When the I/O Port is programmed as Output:

- The Output Buffer is enabled:
  - Open Drain Mode: A “0” in the Output register activates the N-MOS while a “1” in the Output register leaves the port in Hi-Z (the P-MOS is never activated)
  - Push-Pull Mode: A “0” in the Output register activates the N-MOS while a “1” in the Output register activates the P-MOS
- The Schmitt Trigger Input is activated.
- The weak pull-up and pull-down resistors are disabled.
- The data present on the I/O pin is sampled into the Input Data Register every APB2 clock cycle
- A read access to the Input Data Register gets the I/O state in open drain mode
- A read access to the Output Data register gets the last written value in Push-Pull mode

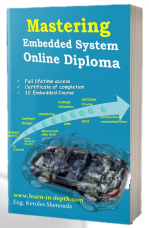


Output configuration

Figure 16. Output configuration



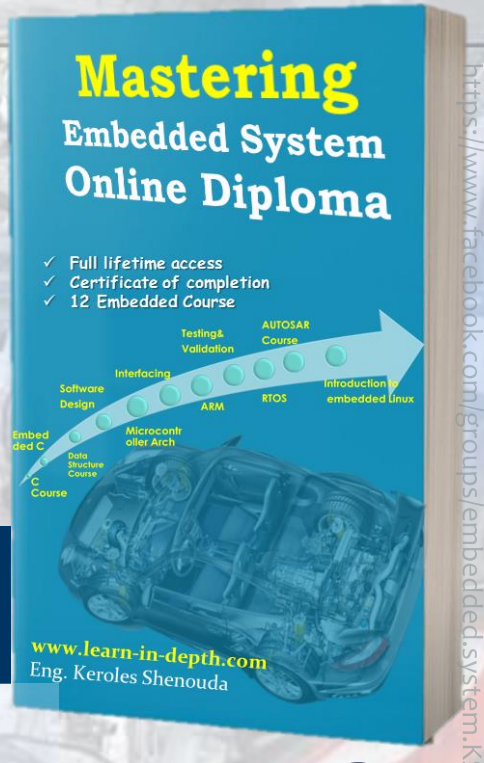
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

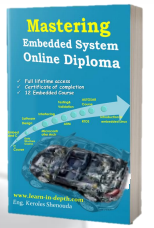
#Be_professional_in_embedded_system

GPIO registers



LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Port configuration register low/high (GPIOx_CRL) (x=A..G)

9.2.1 Port configuration register low (GPIOx_CRL) (x=A..G)

Address offset: 0x00

Reset value: 0x4444 4444

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

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.

9.2.2 Port configuration register high (GPIOx_CRH) (x=A..G)

Address offset: 0x04

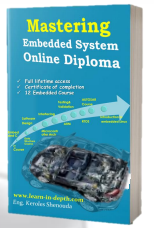
Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[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
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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= 8 .. 15)
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= 8 .. 15)
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.

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



9.2.3 Port input data register (GPIOx_IDR) (x=A..G)

Address offset: 0x08h

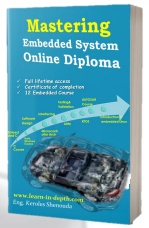
Reset value: 0x0000 XXXX

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDRy:** Port input data (y= 0 .. 15)

These bits are read only and can be accessed in Word mode only. They contain the input value of the corresponding I/O port.



9.2.4 Port output data register (GPIOx_ODR) (x=A..G)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

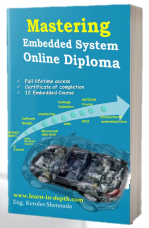
Bits 15:0 **ODRy:** Port output data (y= 0 .. 15)

These bits can be read and written by software and can be accessed in Word mode only.

Note: For atomic bit set/reset, the ODR bits can be individually set and cleared by writing to the GPIOx_BSRR register (x = A .. G).

Table 59. GPIO register map and reset values

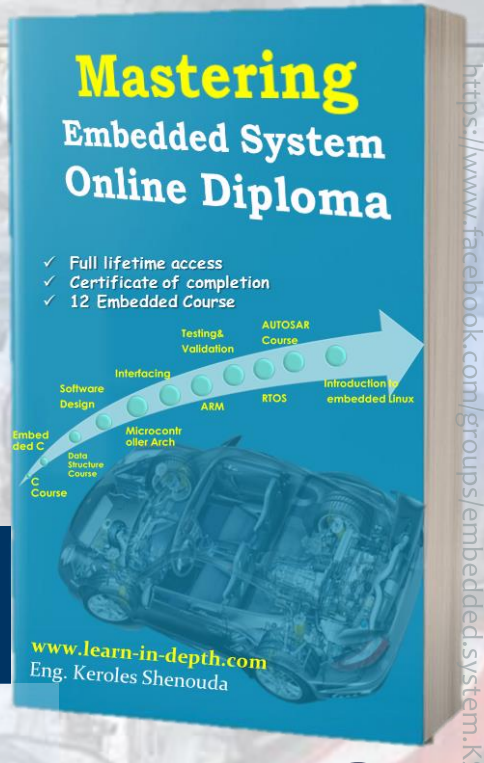
Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPIOx_CRL	CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]		CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
	Reset value	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
0x04	GPIOx_CRH	CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]		CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
	Reset value	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
0x08	GPIOx_IDR	Reserved																IDRy															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOx_ODR	Reserved																ODRy															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	GPIOx_BSRR	BR[15:0]																BSR[15:0]															
	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
0x14	GPIOx_BRR	Reserved																BR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	GPIOx_LCKR	Reserved															LCKK	LCK[15:0]															
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

Lab1 GPIO: STM32F103XX



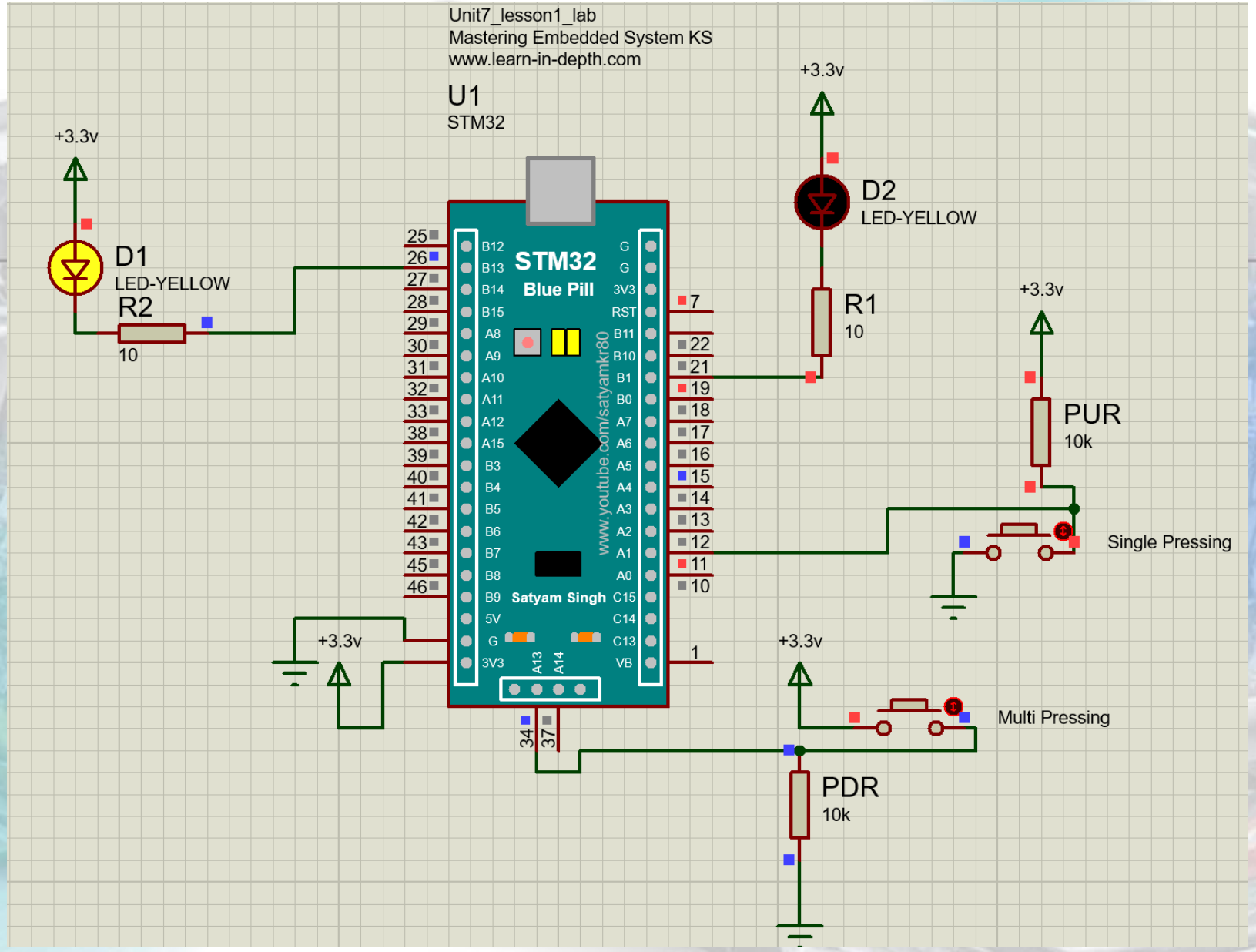
LEARN-IN-DEPTH
Be professional in embedded system

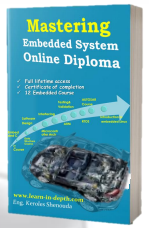
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

lab1

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Lab1

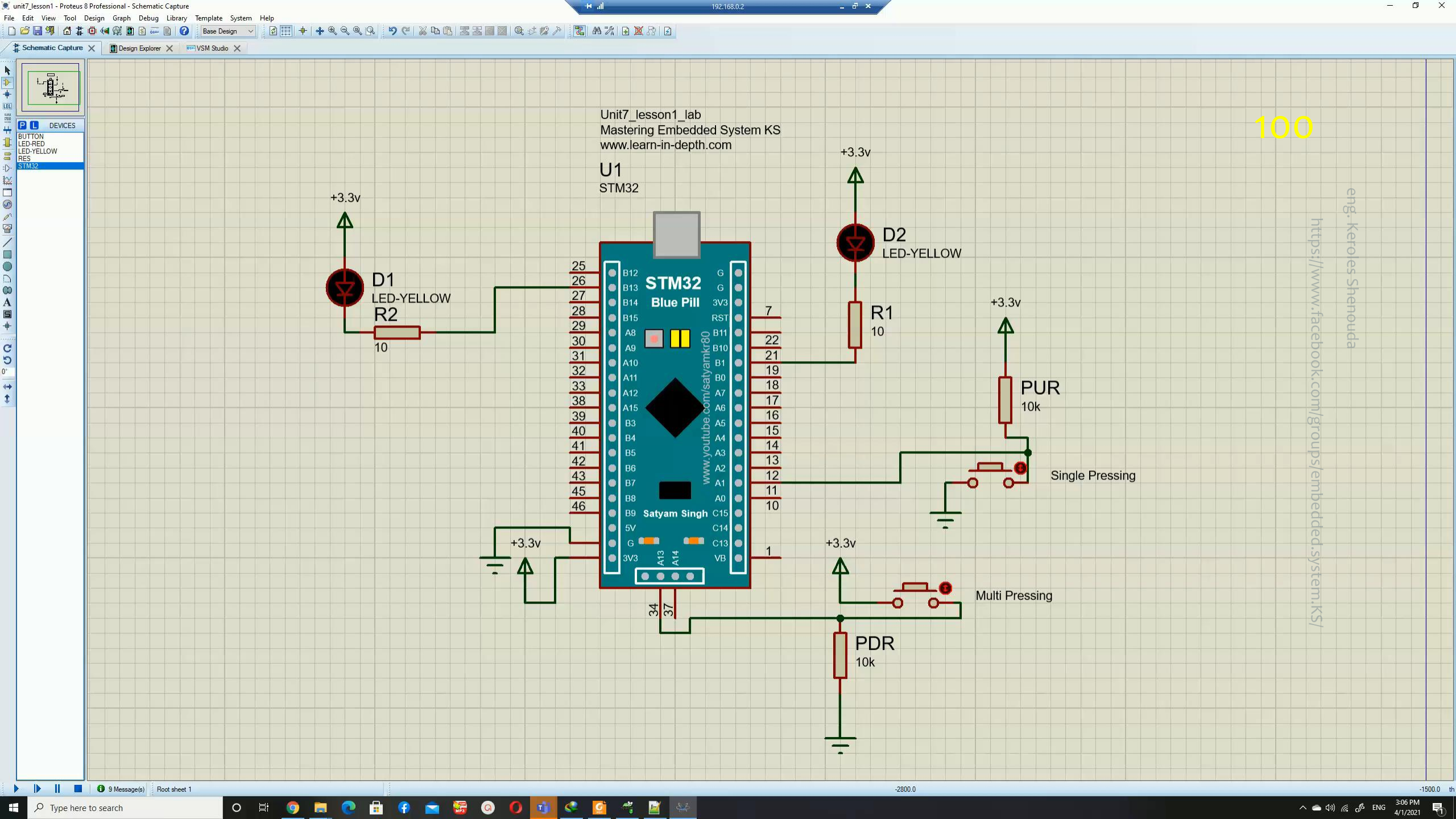


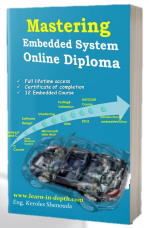


#LEARN_IN DEPTH

#Be_professional_in embedded_system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/


Unit7_lesson1_lab
Mastering Embedded System KS
www.learn-in-depth.com
100
U1
STM32
+3.3v
D1
LED-YELLOW
R2
10
D2
LED-YELLOW
R1
10
PUR
10k
Single Pressing
Multi Pressing
PDR
10k
STM32
Blue Pill
Satyam Singh
www.youtube.com/satyamkr80
eng. Keroles Shenouda
https://www.facebook.com/groups/embeddedsystemKS/



#LEARN_IN DEPTH

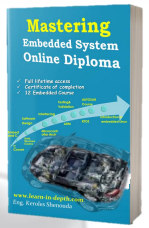
#Be_professional_in_embedded_system

```
/**
 * Mastering Embedded System Online Diploma
 * www.learn-in-depth.com
 */

#if !defined(__SOFT_FP__) && defined(__ARM_FP)
#warning "FPU is not initialized, but the project is compiling for an FPU. Plea
#endif


//Learn-in-depth
//Keroles Shenouda
//Mastering_Embedded System online diploma
typedef volatile unsigned int vuint32_t ;
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>

// register address
//RCC
#define RCC_BASE      0x40021000
#define RCC_APB2ENR   *(volatile uint32_t *)(RCC_BASE   + 0x18)
#define RCC_IOPAEN    (1<<2)

//GPIO
#define GPIOA_BASE    0x40010800
#define GPIOA_CRH     *(volatile uint32_t *)(GPIOA_BASE + 0x04)
#define GPIOA_CRL     *(volatile uint32_t *)(GPIOA_BASE + 0x00)
#define GPIOA_ODR     *(volatile uint32_t *)(GPIOA_BASE + 0x0C)
#define GPIOA_IDR     *(volatile uint32_t *)(GPIOA_BASE + 0x08)
#define GPIOA13       (1UL<<13)

#define GPIOB_BASE    0x40010C00
#define GPIOB_CRH     *(volatile uint32_t *)(GPIOB_BASE + 0x04)
#define GPIOB_CRL     *(volatile uint32_t *)(GPIOB_BASE + 0x00)
#define GPIOB_ODR     *(volatile uint32_t *)(GPIOB_BASE + 0x0C)
#define GPIOB_IDR     *(volatile uint32_t *)(GPIOB_BASE + 0x08)


```

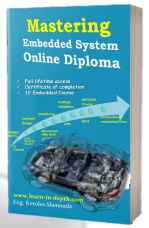
```
void clock_init()
{
    //Enable clock GPIOA
    RCC_APB2ENR |= RCC_IOPAEN;
    //Enable clock GPIOB Bit 3 IOPBEN: IO port B clock enable
    RCC_APB2ENR |= (1<<3) ;
}
void GPIO_init()
```

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

```c
void GPIO_init()
{

    //take care CNF by default  1  (check the reset value)
    GPIOA_CRL = 0 ;
    GPIOA_CRH = 0 ;
    GPIOA_ODR = 0 ;


    //00: Input mode (reset state)
    GPIOA_CRL &= ~( 0b11 << 4 )  ;
    //CNF 01: Floating input (reset state)
    //GPIOA_CRL = 0b10000000  ;
    GPIOA_CRL |= (0b01<<6) ;


    //PB1 (output PUSH pull Mode)
    //01: Output mode, max speed 10 MHz.
    GPIOB_CRL |= ( 0b01 << 4 )  ;
    //CNF 00: General purpose output push-pull
    GPIOB_CRL &= ~( 0b11 << 6 )  ;

    //============================



    //00: Input mode (reset state)
    GPIOA_CRH &= ~( 0b11 << 20 )  ;
    //CNF 01: Floating input (reset state)
    GPIOA_CRH |= ( 0b01 << 22 )  ;

    //PB7 (open drain)
    //01: Output mode, max speed 10 MHz.
    GPIOB_CRH |= ( 0b01 << 20 )  ;
    //CNF 00: General purpose output push-pull
    GPIOB_CRH &= ~( 0b11 << 22 )  ;

}
void wait_ms(uint32_t time){
    uint32_t i, j;
    for(i = 0; i < time; i++)
        for(j = 0; j < 255; j++);
}
```



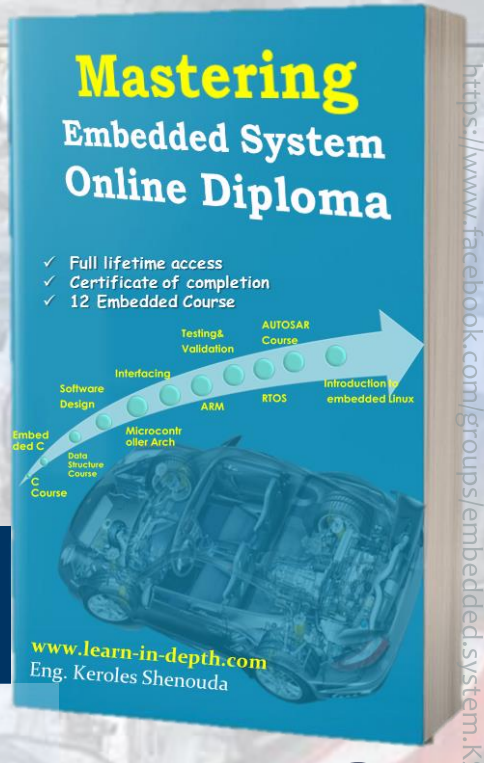

```
104 int main(void)
105 {
106
107     clock_init();
108     GPIO_init();
109
110
111
112     while(1)
113     {
114         if (((GPIOA_IDR & (1<<1)) >> 1 ) == 0)
115         {
116             GPIOB_ODR ^= 1<<1 ;
117             while ((((GPIOA_IDR & (1<<1)) >> 1 ) == 0)); //Single Pressing
118         }
119         if (((GPIOA_IDR & (1<<13)) >> 13 ) == 1) //Multi Pressing
120         {
121             GPIOB_ODR ^= 1<<13 ;
122         }
123         wait_ms(1);
124     }
125 }
```



#LEARN_IN DEPTH

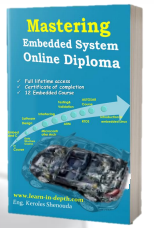
#Be_professional_in_embedded_system

Assignment



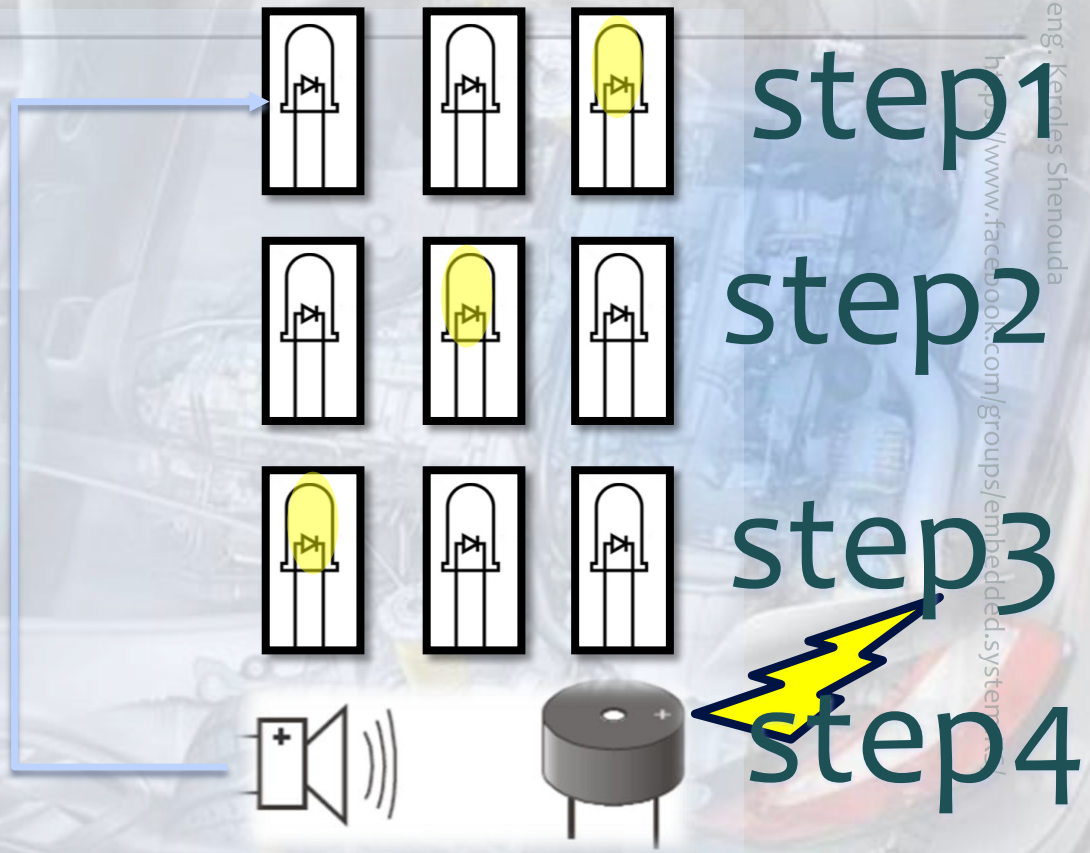
LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Assignment

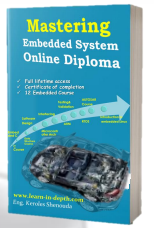
- Read the GPIO Data Sheet in Atmega32
- And Write A embedded C Code
- To toggle 3 Led on sequential way
- Then start a tick(pulse) by Buzzer then
- Rerun again



Port D pin 5,6,7 as output (LED)
Port D pin 4 (Buzzer)

Choose any delay you want
But the delay should be enough to be detect by human eyes
Also don't use avr/io.h and util/delay.h

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



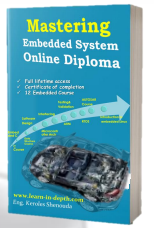
#LEARN_IN DEPTH

#Be_professional_in embedded_system

References

- https://docs.google.com/viewer?a=v&pid=sites&srcid=ZmtlLnV0bS5teXxyaWR6dWFuLXMtd2Vic2l0ZXxneDo2ODU0NzlkM2JkOTg4MjRk
- http://www.avrprojects.net/index.php/avr-projects/sensors/38-humidity-and-temperature-sensor-dht11?showall=&start=1
- http://www.cse.wustl.edu/~lu/cse467s/slides/dsp.pdf
- http://www.avr-tutorials.com/
- Microprocessor: ATmega32 (SEE3223-10) http://ridzuan.fke.utm.my/microprocessor-atmega32-see3223-10
- http://circuitdigest.com/article/what-is-the-difference-between-microprocessor-and-microcontroller
- http://cs4hs.cs.pub.ro/wiki/roboticsisfun/chapter2/ch2_7_programming_a_microcontroller
- Embedded Systems with ARM **Cortex-M Microcontrollers in** Assembly Language and C Dr. Yifeng Zhu Third edition June 2018

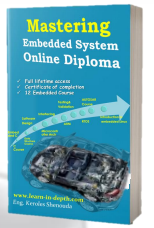
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

- http://techdifferences.com/difference-between-interrupt-and-polling-in-os.html
- http://www.bogotobogo.com/Embedded/hardware_interrupt_software_interrupt_latency_irq_vs_fiq.php
- Preventing Interrupt Overload Presented by Jiyong Park Seoul National University, Korea 2005. 2. 22. John Regehr, Usit Duogsaa, School of Computing, University.
- First Steps Embedded Systems Byte Craft Limited reference
- COMPUTER ORGANIZATION AND ARCHITECTURE DESIGNING FOR PERFORMANCE EIGHTH EDITION **William Stallings**
- **Getting Started with the Tiva™ TM4C123G LaunchPad Workshop**

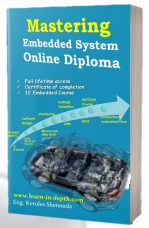
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

- Tiva™ TM4C123GH6PM Microcontroller DATA SHEET
- Interrupts and Exceptions COMS W6998 Spring 2010
- THE **AVR MICROCONTROLLER**. AND EMBEDDED SYSTEMS Using Assembly and C. Muhammad Ali **Mazidi**.

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

- https://docs.google.com/viewer?a=v&pid=sites&srcid=ZmtlLnV0bS5teXxyaWR6dWFuLXMtd2Vic2l0ZXxneDo2ODU0NzlkM2JkOTg4MjRk
- http://www.avrprojects.net/index.php/avr-projects/sensors/38-humidity-and-temperature-sensor-dht11?showall=&start=1
- http://www.cse.wustl.edu/~lu/cse467s/slides/dsp.pdf
- http://www.avr-tutorials.com/
- Microprocessor: ATmega32 (SEE3223-10) http://ridzuan.fke.utm.my/microprocessor-atmega32-see3223-10
- http://circuitdigest.com/article/what-is-the-difference-between-microprocessor-and-microcontroller
- AVR Microcontroller and Embedded Systems: Using Assembly and C (Pearson Custom Electronics Technology) 1st Edition https://www.amazon.com/AVR-Microcontroller-Embedded-Systems-Electronics/dp/0138003319

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/


FOLLOW US
Press here
111
Mastering
Embedded System
Online Diploma
#LEARN_IN DEPTH
#Be_professional_in_embedded_system
Thank You
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/