forward together · saam vorentoe · masiye phambili

Computer Systems / Rekenaarstelsels 245 - 2020

Lecture 9

Inputs and Outputs – Part 1

Intrees en Uittrees – Deel 1

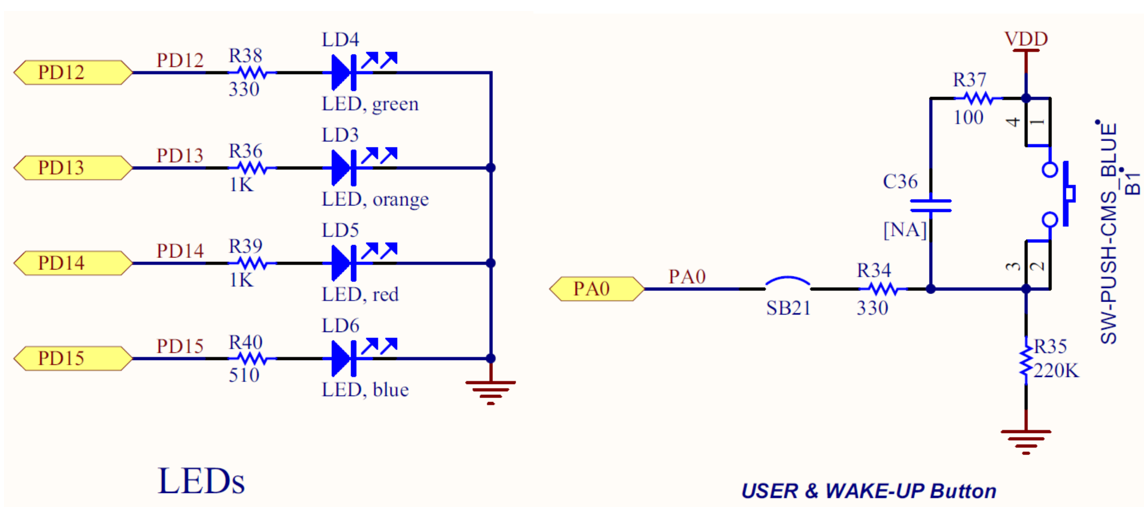
Dr Rensu Theart & Dr Lourens Visagie

Lecture Overview

- General-purpose Input/Output
- Digital signals
  - Outputs
  - Inputs
    - Schmitt triggers
- Debouncing

General-purpose Input/Output (GPIO) / Algemene Intree/Uittree

- **General-purpose Input/Output** (GPIO) pins are used to read or write digital signals from and to external devices.
- These pins are commonly connected to LEDs and switches but can be used in a wide variety of situations, even basic communication.
- LEDs are usually wired to glow when driven with a 1 and to turn off when driven with a 0.
  - Current-limiting resistors are placed in series with the LEDs to set the brightness and avoid overloading the current capability of the GPIO.
- Switches are usually wired to produce a 1 when closed and a 0 when open.

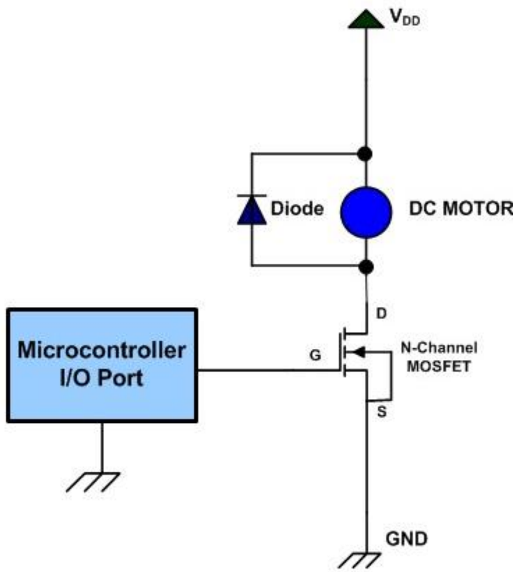


Output Examples / Uittree Voorbeelde

LED/display

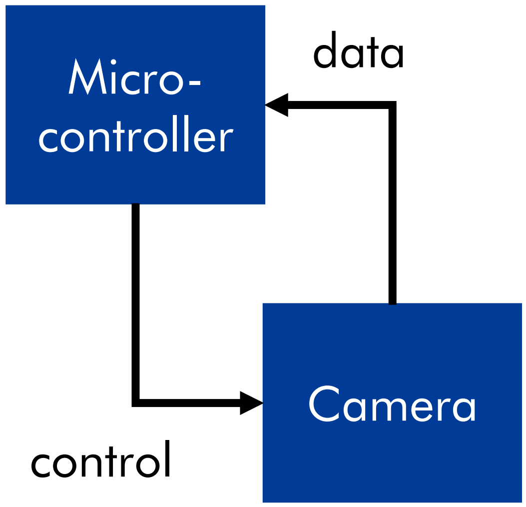
Our application wants to present user with feedback

Output switch



Our application wants to switch something on

Inter-device signaling



Our application must signal another device

Input Examples / Intree Voorbeelde

Push-button

Our application must know that a button was pushed/released

Contact switch

Our application must know the state of switch

Inter-device signaling

A device outputs a low or a high signal to represent a state. Our application wants to know the state, count number of pulses, detect transitions etc.

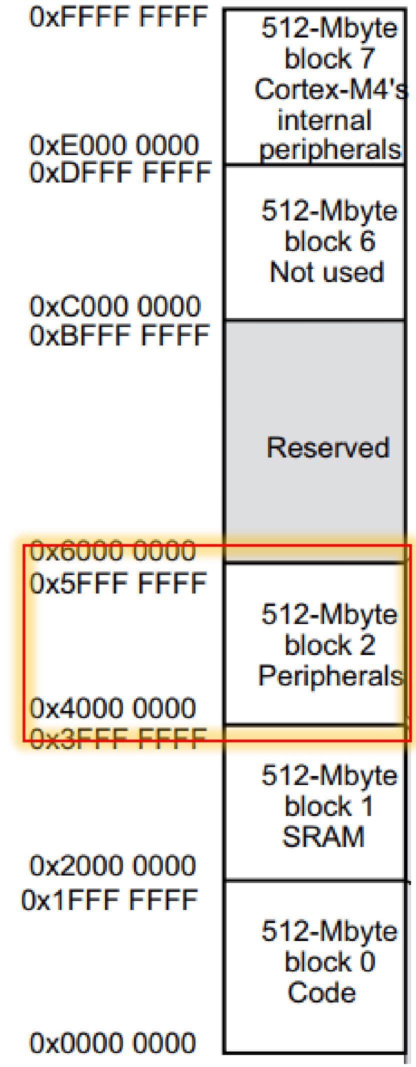
General-purpose Input/Output (GPIO) / Algemene Intree/Uittree

- At a minimum, any GPIO pin requires **registers** to:
  - read input pin values
  - write output pin values
  - set the direction of the pin.
- In many embedded systems, the GPIO pins can be shared with one or more special-purpose peripherals, so additional configuration registers are necessary to determine whether the pin is general or special purpose.
- Furthermore, the processor may generate interrupts when an event such as a rising or falling edge occurs on an input pin, and configuration registers may be used to specify the conditions for an interrupt.

MCU pin	
Main function	**Alternate functions**
PA4	SPI1_NSS, SPI3_NSS/I2S3_WS, USART2_CK, ADC1_4
PA5	TIM2_CH1/TIM2_ETR, SPI1_SCK, ADC1_5
PA6	TIM1_BKIN, TIM3_CH1, SPI1_MISO, ADC1_6

General-purpose Input/Output (GPIO) / Algemene Intree/Uittree

- Recall that a portion of the address space is dedicated to I/O devices rather than memory.
  - In our case the addresses in range 0x4000 0000 to 0x5FFF FFFF are used for I/O.
- Each I/O device is assigned one or more memory addresses in this range.
- A store to the specified address sends data to the device. A load receives data from the device.
- This method of communicating with I/O devices is called memory-mapped I/O.
  - A load or store may access either memory or an I/O device.



General-purpose Input/Output (GPIO) / Algemene Intree/Uittree

- The figure shows the hardware needed to support two memory-mapped I/O devices.
- An **address decoder** determines which device communicates with the processor.
- It uses the *Address* and *MemWrite* signals to generate control signals for the rest of the hardware.
- The *ReadData* multiplexer selects between memory and the various I/O devices.
- **Write-enabled registers** hold the values written to the I/O devices.

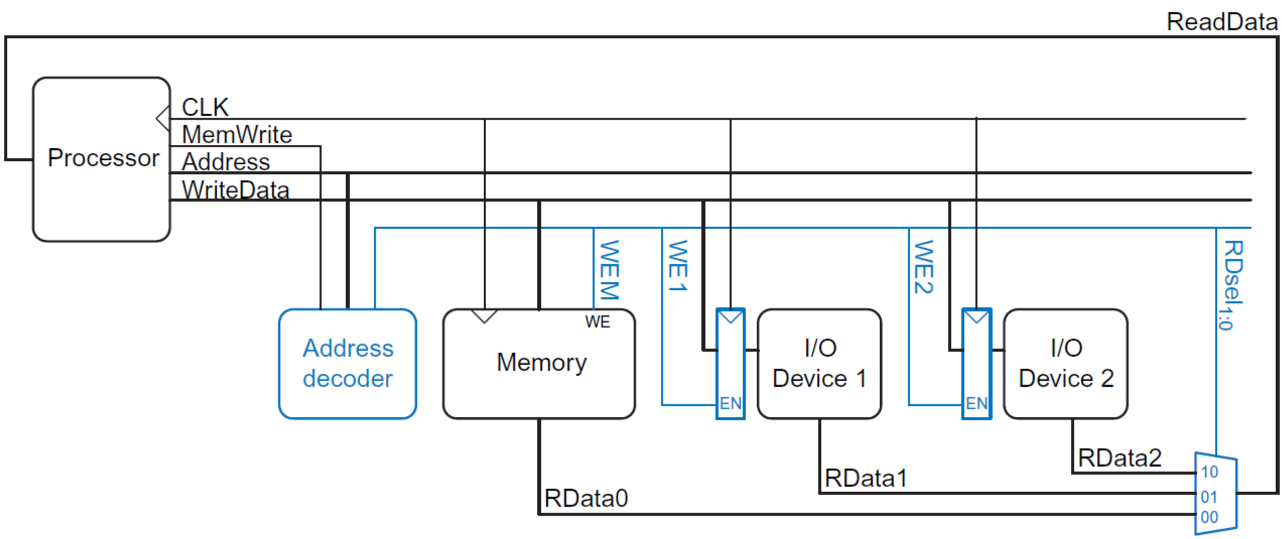


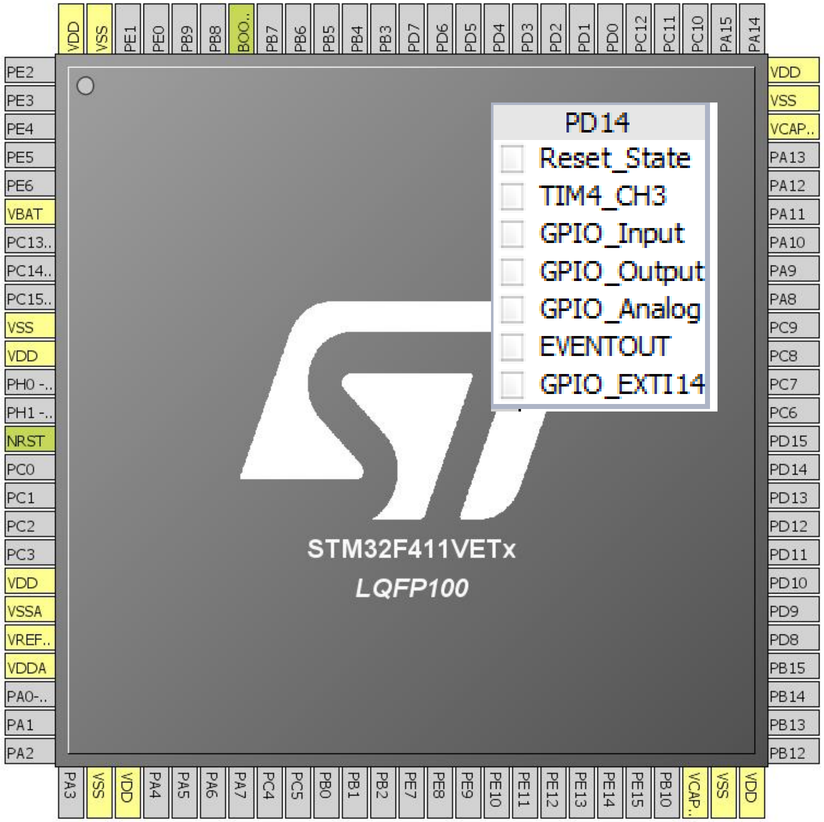
Figure e9.1 Support hardware for memory-mapped I/O

General-purpose Input/Output (GPIO) / Algemene Intree/Uittree

- GPIO = General Purpose Input/Output

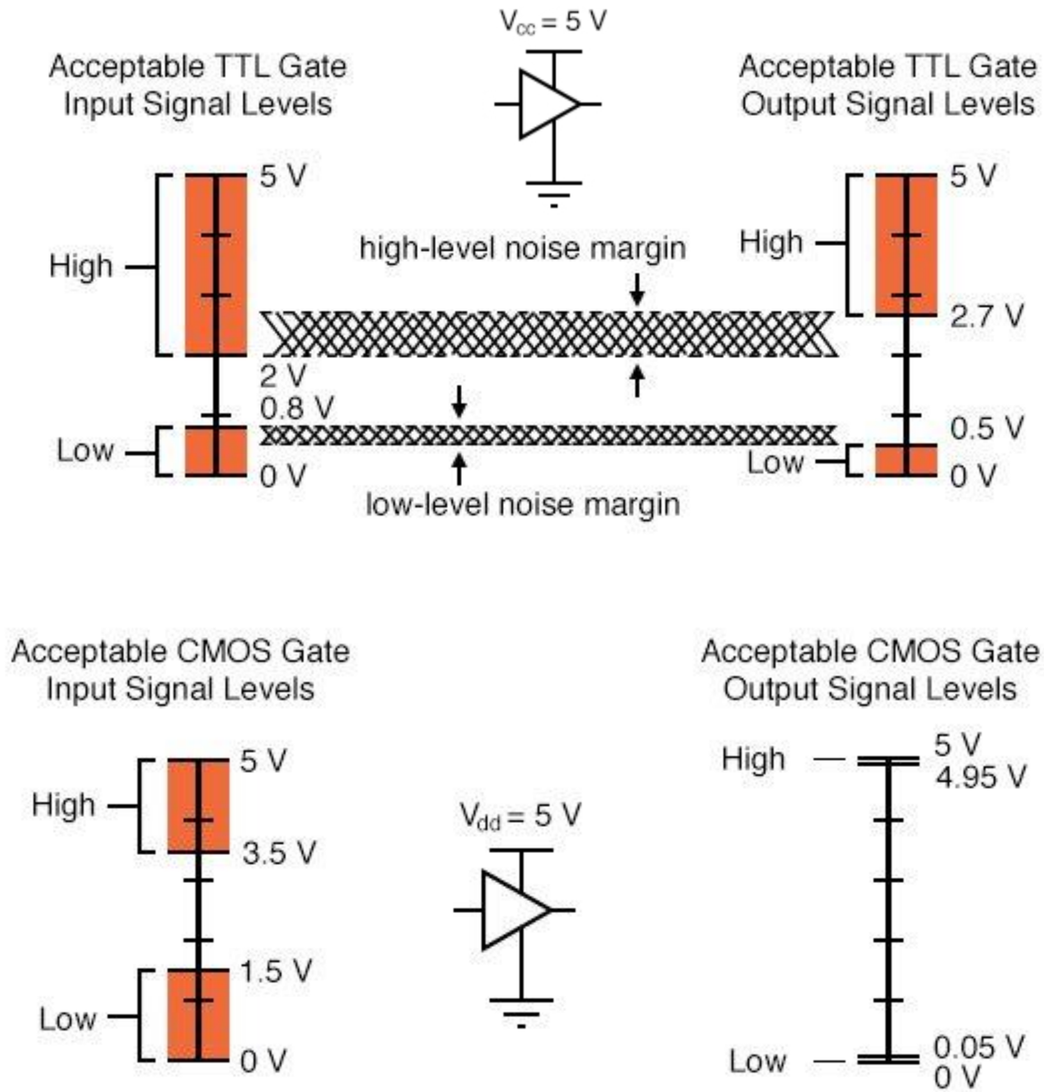
- Generally, pins on the microcontroller can be configured as
- Input
  - Floating (high-impedance)
  - Pull-down/pull-up
  - Analog
  - Alternate function

- Output
  - Open-drain or push-pull
  - Pull-up/pull-down
  - Alternate function



Digital Signals / Digitale Seine

- When considering GPIOs, we are talking about digital signal levels.
- Logical 0 = signal is at ground level (0V)
- Logical 1 = signal is at VDD level
- TTL signal levels vs. CMOS signal levels.
- For STM32F411 all I/Os are CMOS and TTL compliant.



https://www.allaboutcircuits.com/textbook/digital/chpt-3/logic-signal-voltage-levels/

Signal Levels for the STM32F4 processor / Seinvlakke vir die STM32F4 Verwerker

- From the STM32F411VE Data Sheet:

Table 54. Output voltage characteristics

Symbol	Parameter	Conditions	Min	Max	Unit
V_{OL}[1]	Output low level voltage for an I/O pin	CMOS port[2] I_{IO} = +8 mA 2.7 V ≤ V_{DD} ≤ 3.6 V	-	0.4	V
V_{OH}[3]	Output high level voltage for an I/O pin		V_{DD}–0.4	-	
V_{OL}[1]	Output low level voltage for an I/O pin	TTL port[2] I_{IO} =+8 mA 2.7 V ≤ V_{DD} ≤ 3.6 V	-	0.4	V
V_{OH}[3]	Output high level voltage for an I/O pin		2.4	-	
V_{OL}[1]	Output low level voltage for an I/O pin	I_{IO} = +20 mA 2.7 V ≤ V_{DD} ≤ 3.6 V	-	1.3[4]	V
V_{OH}[3]	Output high level voltage for an I/O pin		V_{DD}–1.3[4]	-	
V_{OL}[1]	Output low level voltage for an I/O pin	I_{IO} = +6 mA 1.8 V ≤ V_{DD} ≤ 3.6 V	-	0.4[4]	V
V_{OH}[3]	Output high level voltage for an I/O pin		V_{DD}–0.4[4]	-	
V_{OL}[1]	Output low level voltage for an I/O pin	I_{IO} = +4 mA 1.7 V ≤ V_{DD} ≤ 3.6 V	-	0.4[5]	V
V_{OH}[3]	Output high level voltage for an I/O pin		V_{DD}–0.4[5]	-	

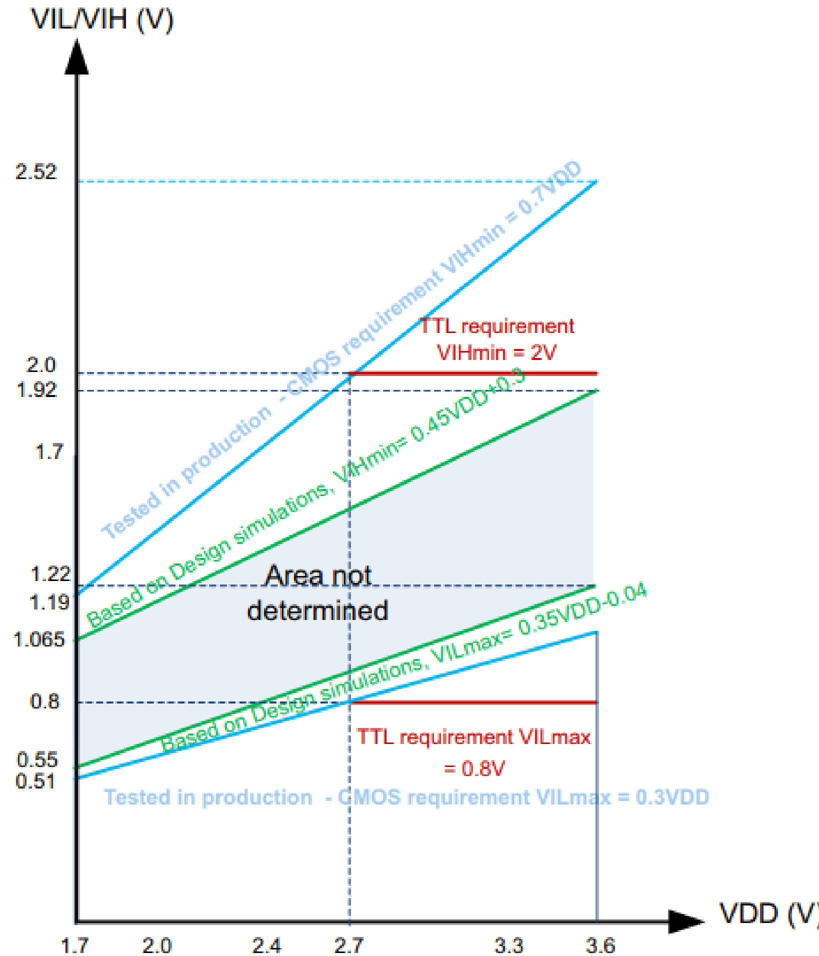
Signal Levels for the STM32F4 processor / Seinvlakke vir die STM32F4 Verwerker

- From the STM32F411VE Data Sheet:
- TTL requirements limit supply voltage
  - FT = 5V tolerant I/O
  - TC = Standard 3.3V I/O

Table 53. I/O static characteristics

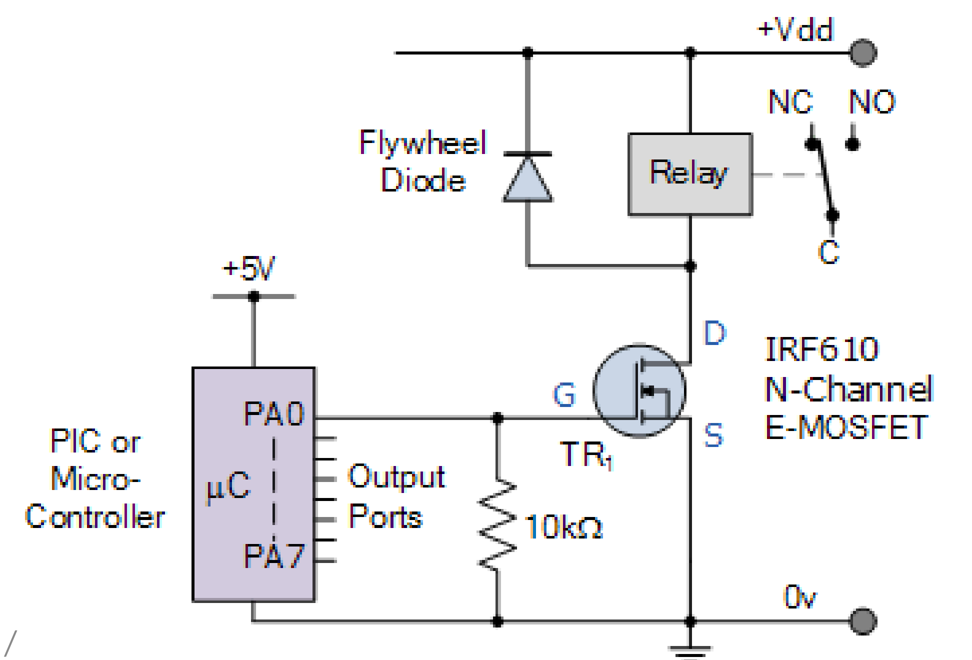
Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IL}	FT, TC and NRST I/O input low level voltage	1.7 V≤V_{DD}≤3.6 V	-	-	$0.3V_{DD}$[1]	V
	BOOT0 I/O input low level voltage	1.75 V≤V_{DD} ≤3.6 V, -40 °C≤T_A ≤125 °C	-	-	$0.1V_{DD}$+0.1[2]	
		1.7 V≤V_{DD} ≤3.6 V, 0 °C≤T_A ≤125 °C	-	-		
V_{IH}	FT, TC and NRST I/O input high level voltage[5]	1.7 V≤V_{DD}≤3.6 V	$0.7V_{DD}$[1]	-	-	V
	BOOT0 I/O input high level voltage	1.75 V≤V_{DD} ≤3.6 V, -40 °C≤T_A ≤125 °C	$0.17V_{DD}$ +0.7[2]	-	-	
		1.7 V≤V_{DD} ≤3.6 V, 0 °C≤T_A ≤125 °C				

Figure 30. FT/TC I/O input characteristics



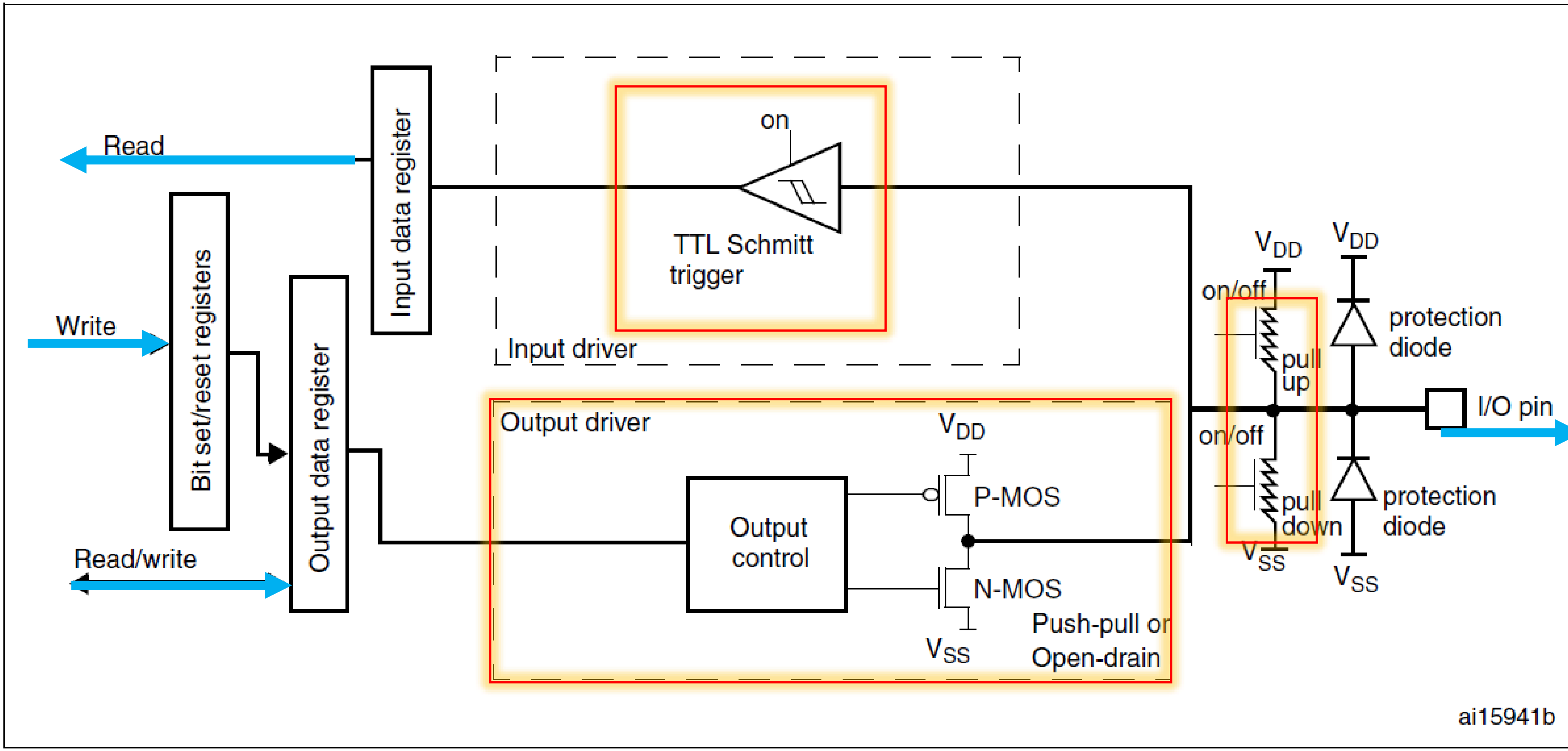
Digital Output / Digitale Uittree

- Change pin output state to a 0 or 1, i.e. drive the voltage level on the pin to GND (logical 0) or VDD (logical 1).
- Pin cannot deliver lots of current - sufficient to switch on a LED, or drive a transistor
  - The GPIOs can sink or source up to ±8 mA and sink or source up to ±20 mA (with a relaxed V_{OL}/V_{OH}).
- For higher current, we must build a switch using the microcontroller output as enable line.
  - Drive a motor
  - Switch on a power line



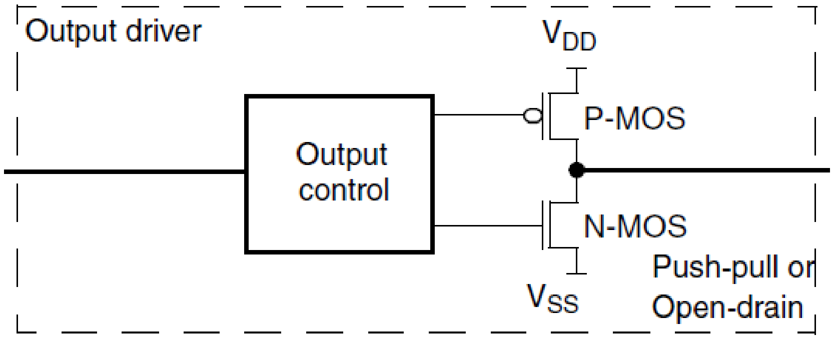
Digital Output / Digitale Uittree

Figure 19. Output configuration



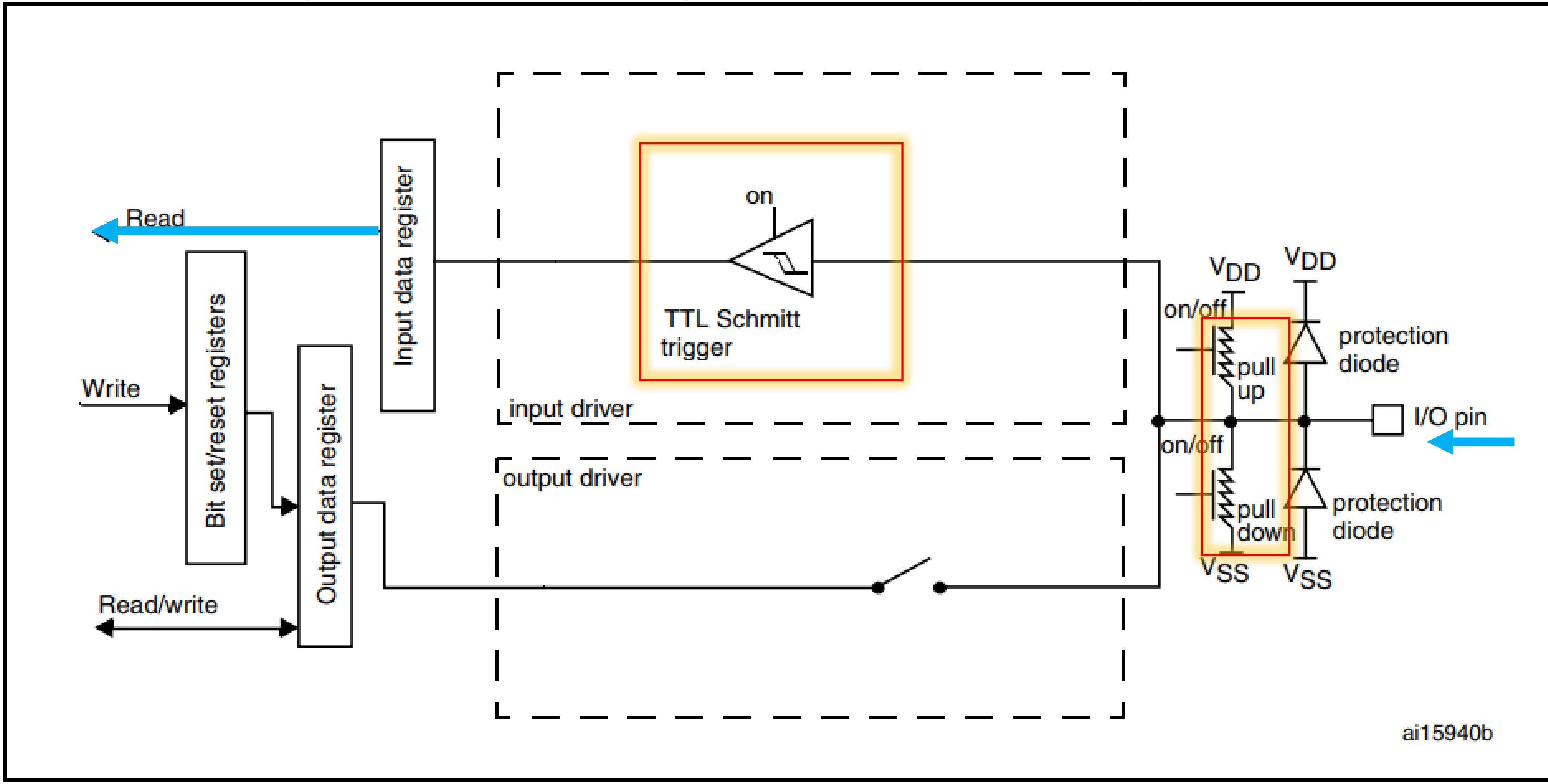
Digital Output / Digitale Uittree

- **Push-Pull configuration**: A "0" in the Output register activates the N-MOS whereas a "1" in the Output register activates the P-MOS
- **Open-drain configuration**: A "0" in the Output register activates the N-MOS whereas a "1" in the Output register leaves the port in Hi-Z (the P-MOS is never activated)
  - You can think of an open-drain GPIO as behaving like a switch which is either connected to ground or disconnected.
  - Often connected to pull-up resistor.
  - Used for communication on a bus (e.g. I2C).
- Option to enable internal weak **pull-up** or **pull-down resistors.**
- The data present on the I/O pin are sampled into the input data register.



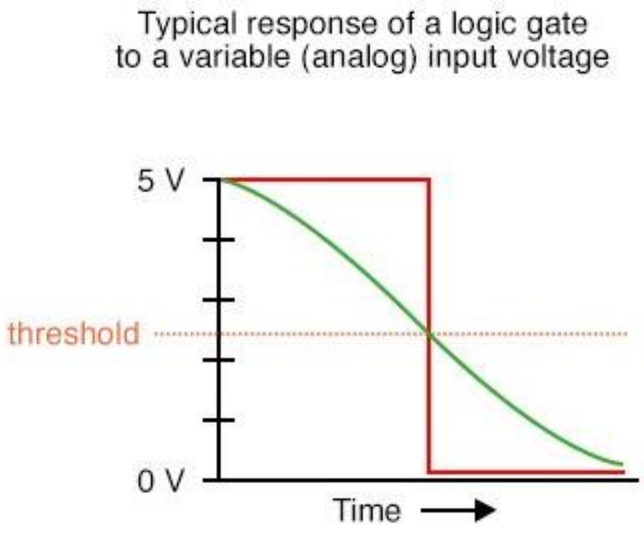
Digital Input / Digitale Intree

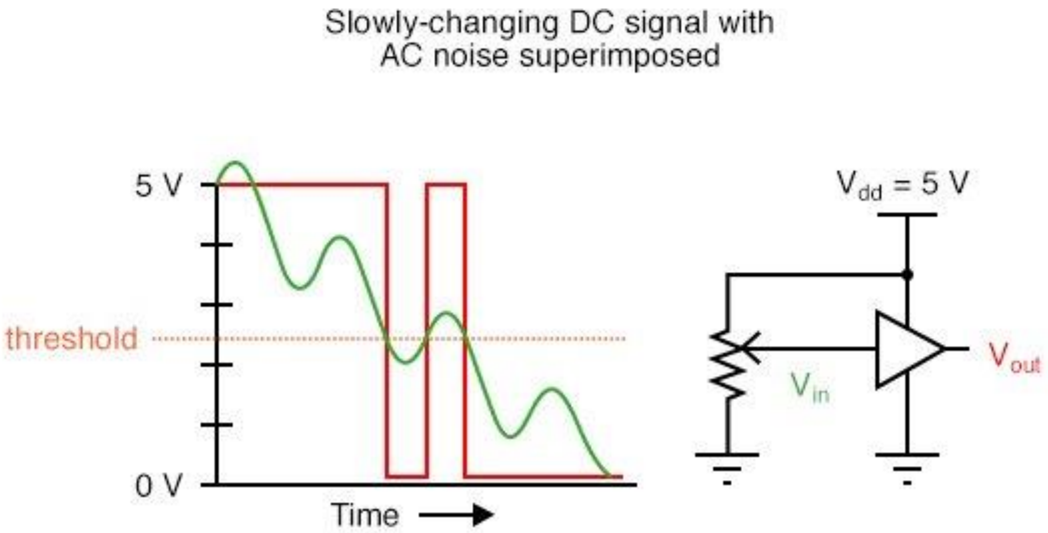
- Read the state of a pin as a 0 or 1, depending on voltage at the pin.
- Input pin is configured as high-impedance (option to also use internal pull-up or pull-down resistors).



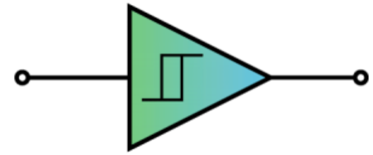
Digital Input / Digitale Intree

- Within the "uncertain" range for any gate input, there will be some point that will be separate the interpretation of an input as either HIGH or LOW.
  - For most gate circuits, this unspecified voltage is a single point
  - Manufacturer only specify V_{IL} and V_{IH} that it guarantees correct interpretation.
- In the presence of AC "noise" voltage superimposed on the DC input signal, a single threshold point at which the gate alters its interpretation of logic level will result in an erratic output.

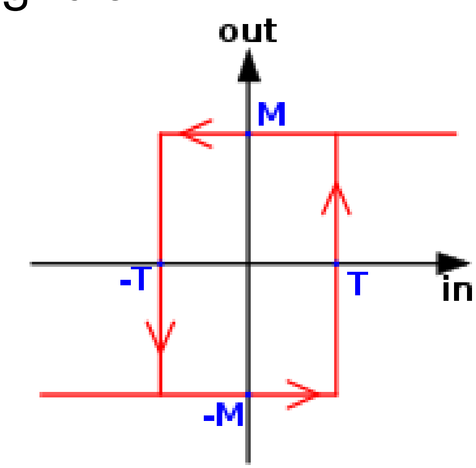


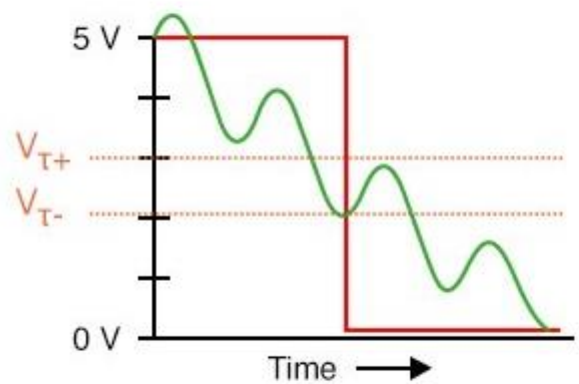


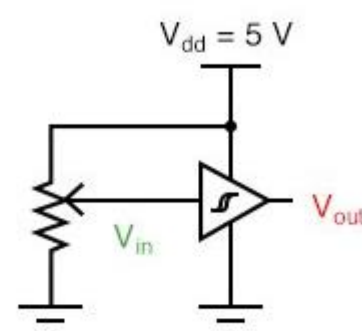
Schmitt trigger



- The solution is to use a **Schmitt trigger**.
- Schmitt trigger is a comparator circuit with **hysteresis** implemented by applying **positive feedback** to the noninverting input of a comparator or differential amplifier.
- Schmitt triggers interpret varying/analog input voltages according to two threshold voltages: a positive-going threshold (VT+), and a negative-going threshold (VT-).
- Converts analog input voltage to digital output – either 0 or 1.
  - Output retains its value until a large enough input to trigger a change in the output
- For a digital input, this removes noise from the signal, producing "square" signals

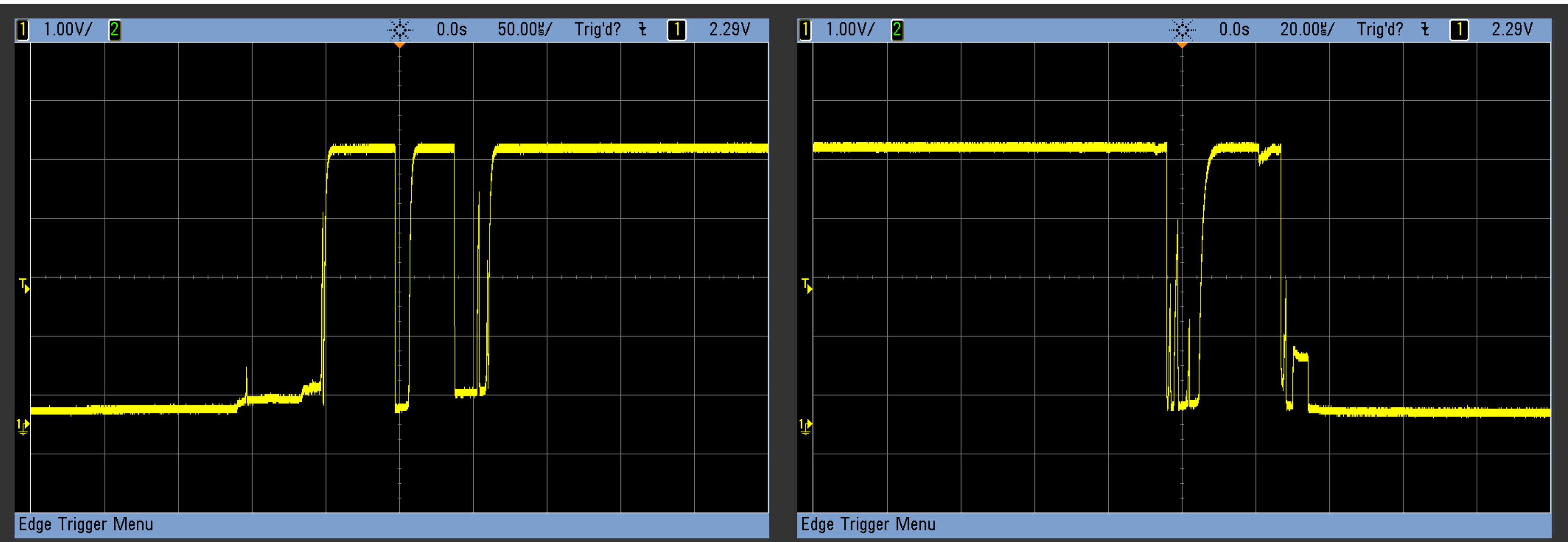






Switch bounce / Skakelaar bons

- When a button is pressed, it doesn't simply go from an open circuit to a closed circuit, there is some mechanical bounce that occurs.
- This can cause the microcontroller to detect multiple button presses – especially when using interrupts.
  - It is most important when the number of button presses matter, not whether a button was pressed or not.



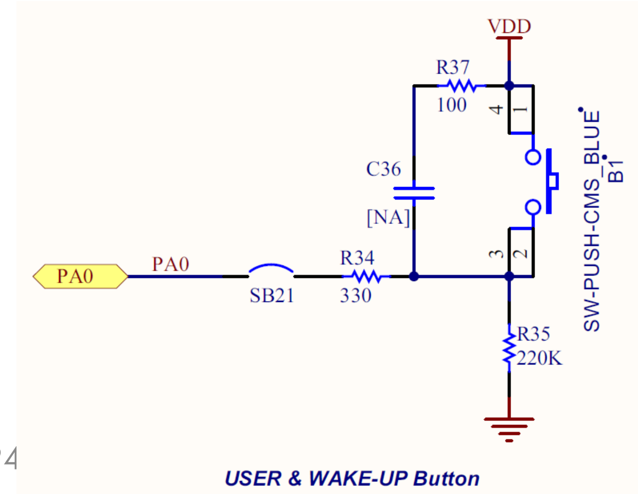
https://hackaday.com/2015/12/09/embed-with-elliot-debounce-your-noisy-buttons-part-i/

Debouncing / Ontbonsing

Buttons can be debounced either with hardware or software.

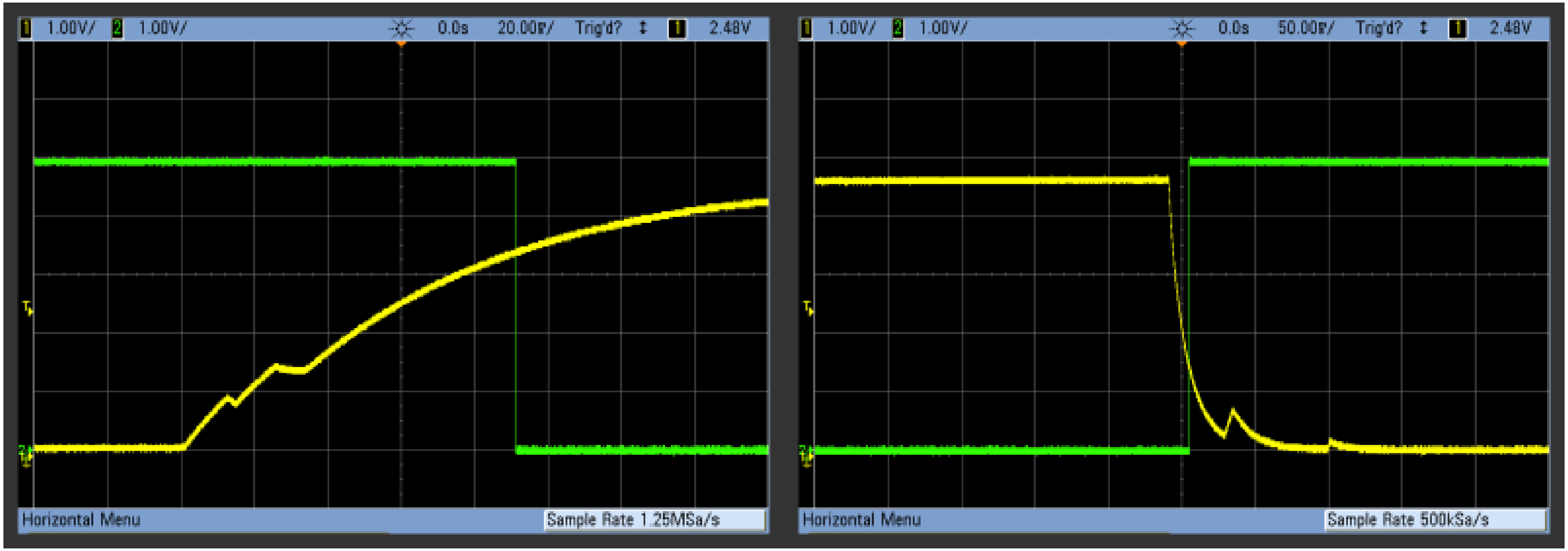
Hardware solution

- Use the **GPIO pull-down** resistor provided by the microcontroller
  - This lets you establish a definite logic state (in this case LOW) on a button when it's not pressed.
- To smooth out fluctuations, we can use a **lowpass filter** to filter the high frequency jitter out.
  - We do this by adding a resistor with a capacitor.
  - R and C are chosen so that their product gives you roughly the time you would like to debounce for.
  - Pressing the button will charge the capacitor. After releasing, the capacitor will keep the pin high for a little while.

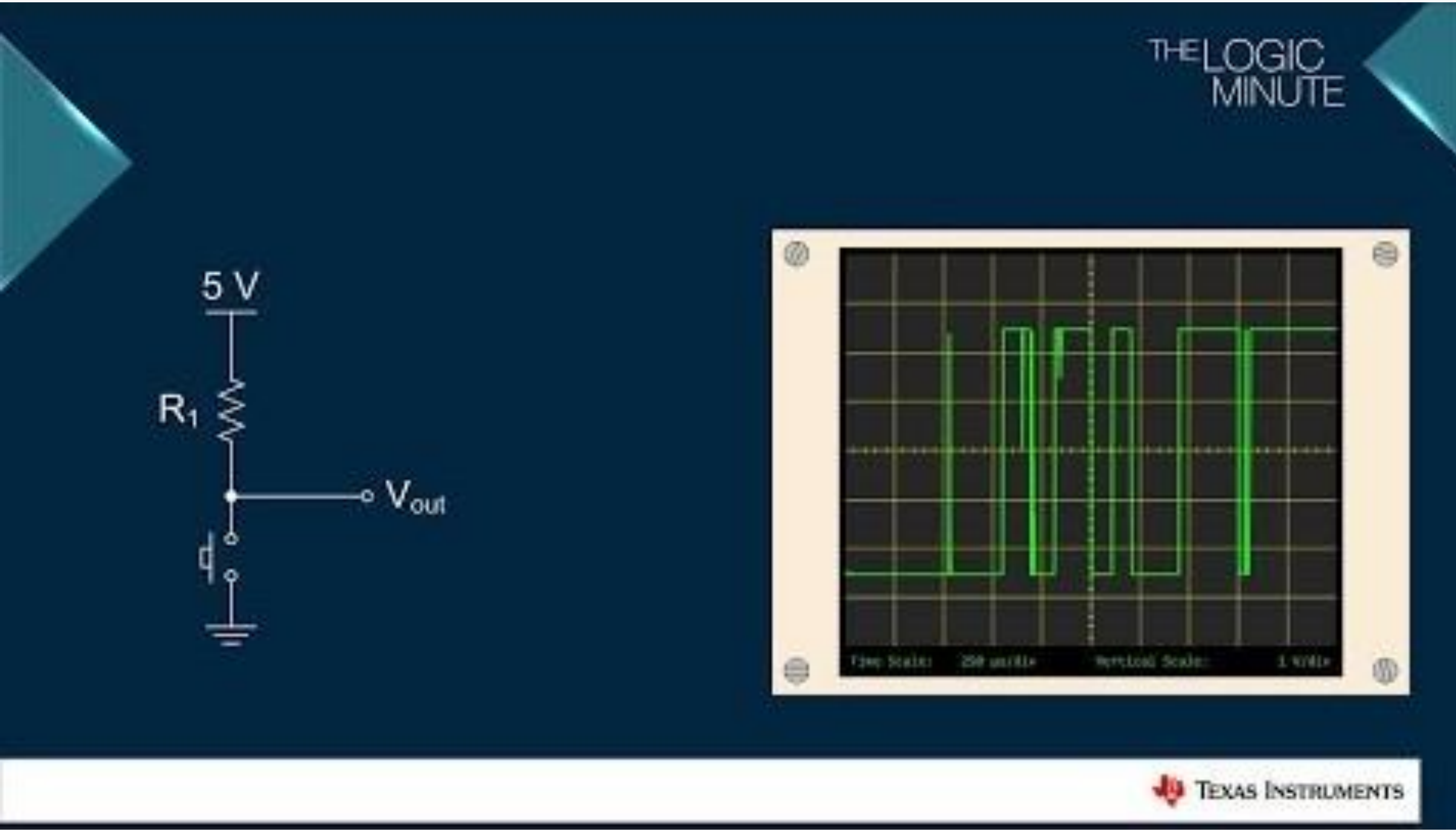


Debouncing / Ontbonsing

- Adding a Schmitt trigger, the gradual changed can be set to 1 or 0.



Debouncing / Ontbonsing



https://www.youtube.com/watch?v=e1-kc04jSE4

Debouncing / Ontbonsing

Software solution

- It is usually best to solve hardware issues in hardware, and there exist several approaches to solve debouncing in software.
- The most basic, and very limited, solution is to simply 'pause' the microcontroller while you wait for the bouncing to finish.
- It is usually best not to use HAL_Delay() inside an interrupt, since it depends on the SysTick interrupt.

```c
void EXTI0_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI0_IRQn 0 */
  int delay = 0xFFFF; // 65535 decrements = about 50ms
  while(delay--);

  buttonPressCount++;
  /* USER CODE END EXTI0_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
  /* USER CODE BEGIN EXTI0_IRQn 1 */

  /* USER CODE END EXTI0_IRQn 1 */
}
```