STM32CubeIDE+MX part 1

IDE installation

- Go to
- https://www.st.com/content/st_com/en/stm32cubemx.html#st-get-software
- download (I suggest version 6.16.1; registration required), extract and install

- Go to
- https://www.st.com/en/development-tools/stm32cubeide.html
- download (I suggest version 1.19.0;), extract and install

- Note: it might be better to avoid strange characters and/or spaces in the installation directory path

Blinky project setup: CubeMX

- STM32CubeMX… Access to Board Selector… Commercial Part Number… NUCLEO-H7A3ZI-Q… click
  - don’t click on the device
  - Board Project Options: Unselect All… OK
- Pinout & Configuration:
  - PB0: GPIO_Output
    - green led is connected to pin 0 of port B, PB0
  - System Core… GPIO… GPIO… User Label: LD1
- Clock Configuration: nothing

Blinky project setup: CubeMX

- Project Manager… Project
  - set project name and location
  - Application Structure… Basic
  - Toolchain / IDE: STM32CubeIDE
- Tools: nothing
- Top right: GENERATE CODE
  - Do you want to continue?... Yes
  - Do you want to download… Yes
    - downloads libraries which are specific to H7 MCUs
  - Allow access… Yes
- Open Project
  - Launches CubeIDE
- *Do these steps before coming to the lab (downloads needed)*

Blinky project: CubeIDE

- To launch STM32CubeIDE: Open Project
  - Or, alternatively,
    - Open Folder
    - Open the file *.project*
  - Opening <*project_name*>*.ioc* opens the file in MX instead

Blinky project: CubeIDE

- Top left, Project explorer: open the project
  - Inc/main.h: we had defined LD1 for PB0 in MX
    - #define LD1_Pin GPIO_PIN_0
    - #define LD1_GPIO_Port GPIOB
  - Scr/main.c: our code *blinky.c* (user code 3) has to be inserted at line 104
  - Use
    - HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0,…);

    instead if you had not defined LD1
- Build (hammer symbol)
- Run (green arrow)
  - Allow access… Yes

- An update of the ST-LINK firmware might be needed
  - Open in update mode… Update

Digital input

- Example *pushbutton*
  - in MX, set PC13 to GPIO_Input
  - test *Pushbutton.c*

Serial communication

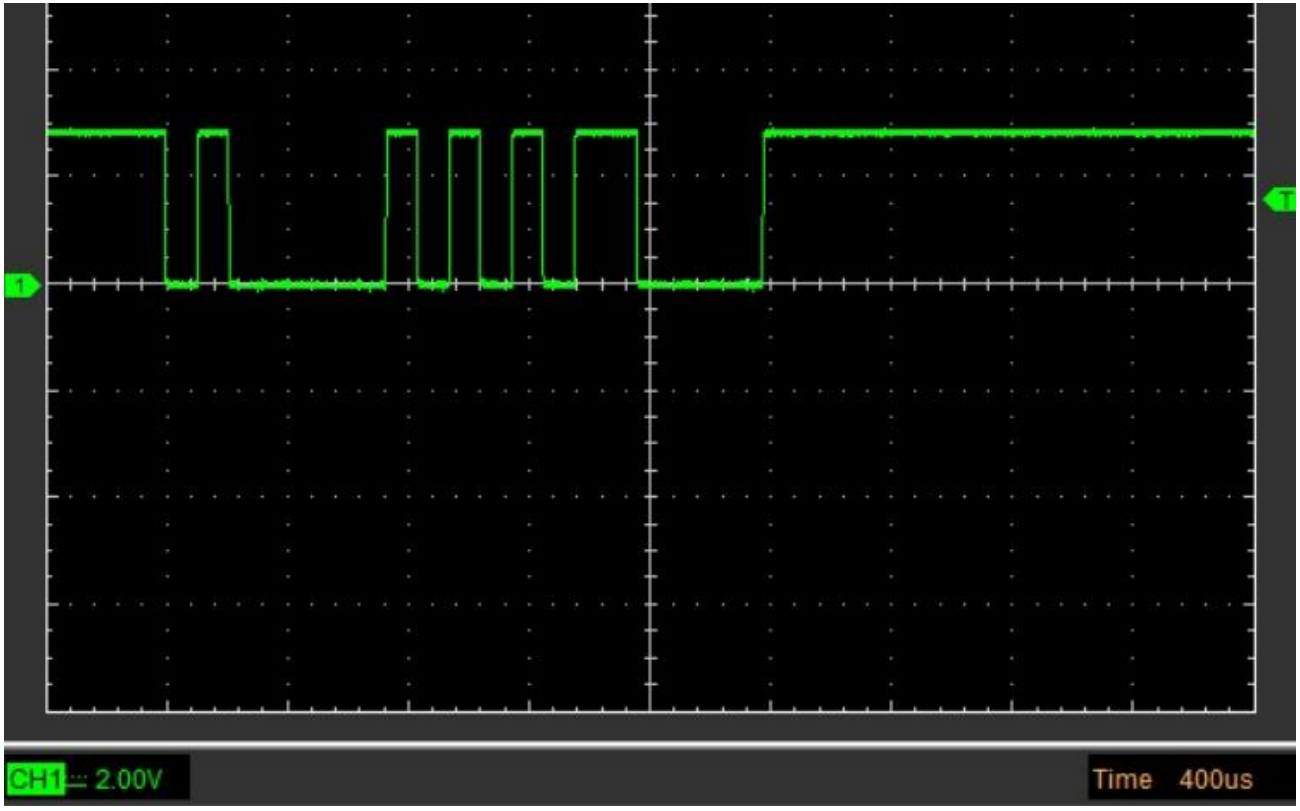
- USART3 interface available on MCU's PD8 and PD9 can be connected (by setting the related solder bridges (SB) to
  - ST-LINK (default), or
  - Zio connector
  - ST morpho connector
- By default, USART3 communication between the target STM32H7 and the STLINK is enabled, to support the Virtual COM port (SB16 and SB17 ON)

Pin name	Function	Virtual COM port (Default configuration)	ARDUINO® D0 and D1	ST morpho connection
PD8	USART3 TX	SB103 OFF, **SB16 ON** and SB15 OFF	SB103 OFF, SB16 OFF and **SB15 ON**	**SB103 ON**, SB16 OFF, SB15 OFF
PD9	USART3 RX	SB104 OFF, **SB17 ON** and SB94 OFF	SB104 OFF, SB17 OFF and **SB94 ON**	**SB104 ON**, SB17 OFF and SB94 OFF

Serial communication

Example:

- A<cr>, i.e.
- 65,13, i.e.
- 01000001, 00001101
- LSB first, start bit is low, stop bit is high



Serial port

Example *printf.c* (output only)

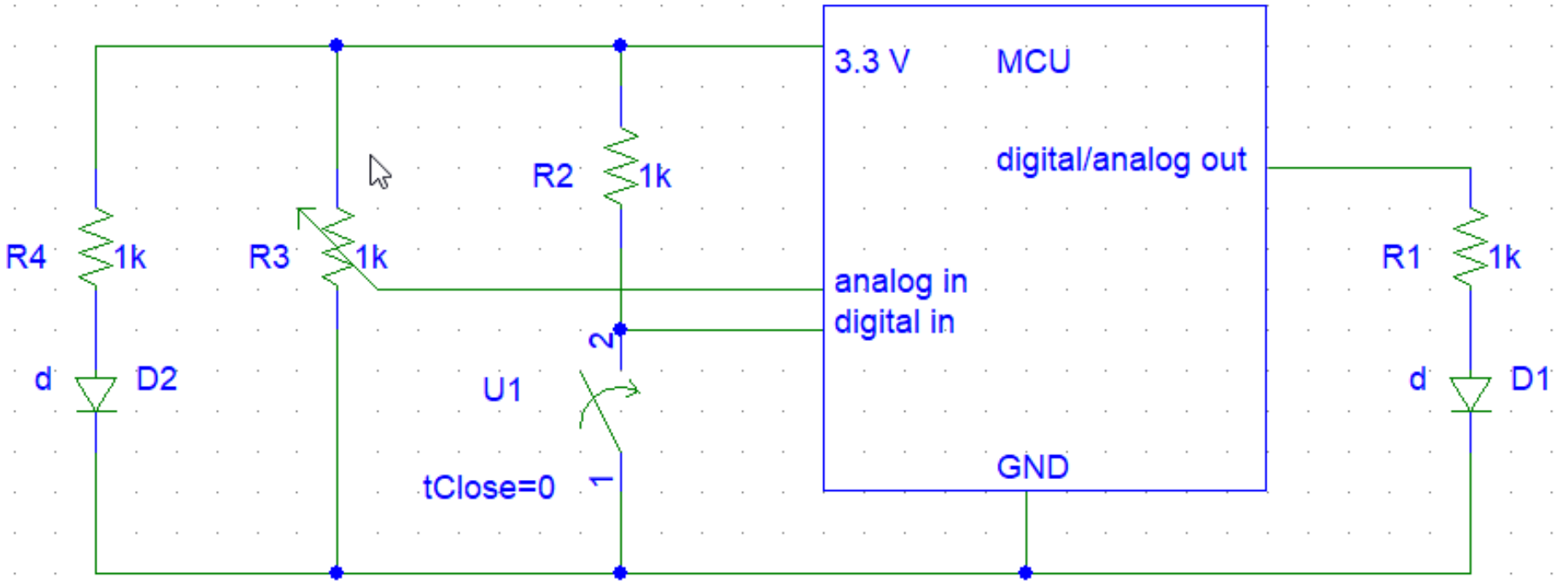
- The PC USB port is connected to UART3, using PD8 and PD9
- In MX
  - PD8 and PD9 are alread defined (see also *main.h* in IDE):
    - GPIO… Single Mapped Signals
  - Connectivity… USART3… Mode: Asynchronous
- Generate code, then compile and launch *printf.c*
- *Printf()* is a very flexible function; in its stack there is a call to *__IO_putchar()* which we implement with a call to our UART
- Use *Teraterm* as terminal emulator on the PC
  - Setup… Serial port.. Baud rate: 115200

Example *serial_io.c* (input and output)

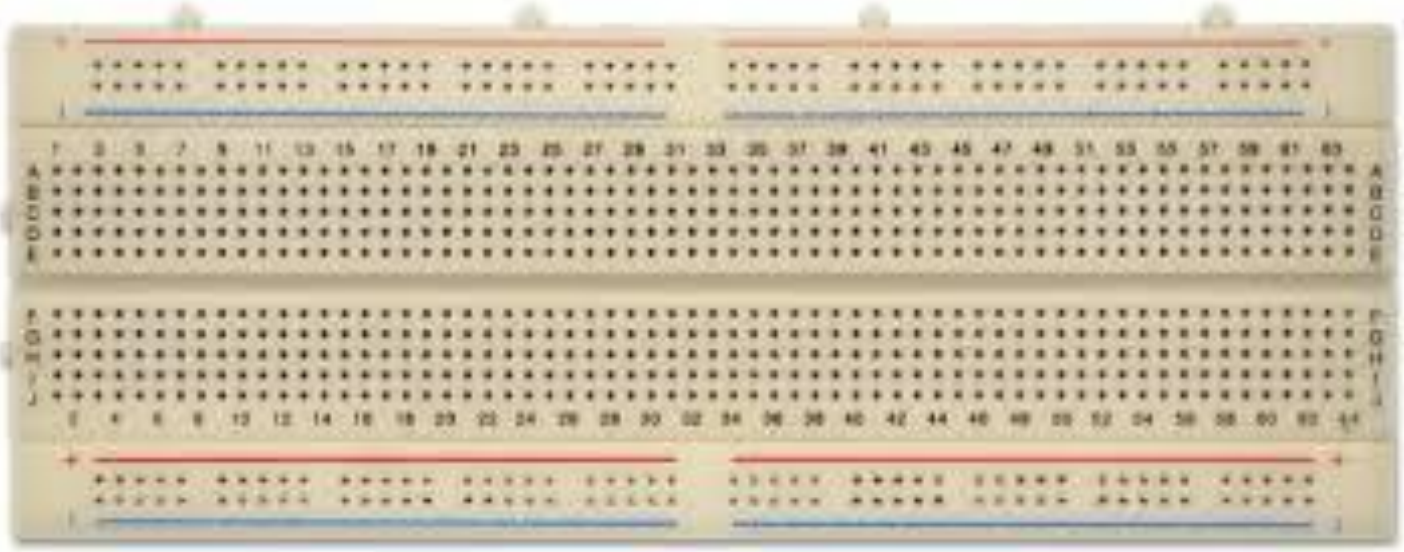
Digital in/out (external): *digital_io*

- Let's use PD14 as input and PB1 as output
- First we have to configure them in MX
  - …access to board selector…
  - set PD14 to GPIO_input and PB1 as GPIO_output
  - … project manager…
- Then test *digital_io.c* in IDE
  - Note the lines around 430 in *main.c*

Main electrical connections



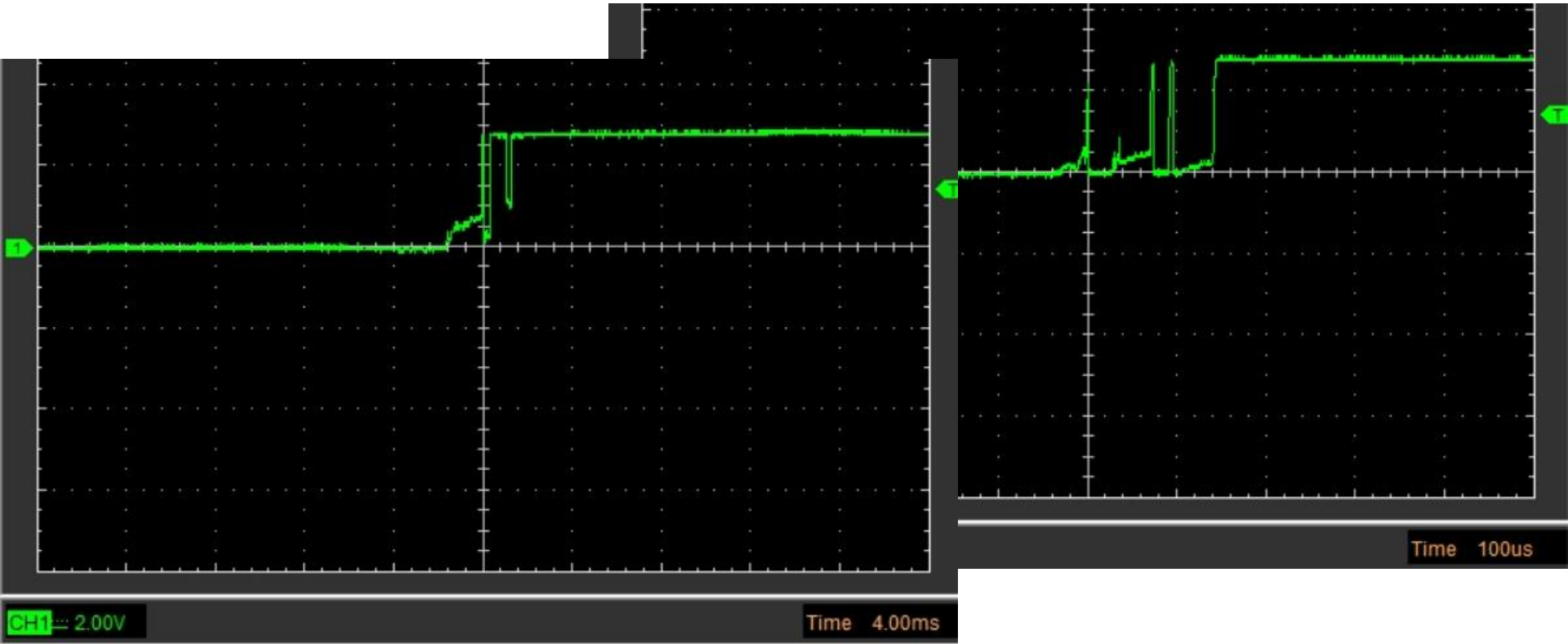
Breadboard



- 4 rows
  - normally used for (from top to bottom)
    - +5 V
    - +3.3 V
    - GND
    - n.c.
  - possibly split into two independent halves
- other holes: connected in two separate groups of columns

Exercises / homework

- Read a pushbutton to increment a counter
  - debouncing (h/w or s/w) is needed!
- auto-increment the counter if button is held down
- increment or decrement the counter according to the pulse duration



Analog in/out: *analog_io*

- Let's use PA6 as input and PA4 as output
- We first set the ADC and DAC in MX
  - ADC1: IN3 Single-ended
    - GPIO Settings shows only PA6
  - DAC1: OUT1 only external pin
    - GPIO Settings shows only PA4
  - Clock Configuration… Automatically solve clock issues: yes
- Then we test
  - *analog_out.c*
  - *analog_io.c*, using a potentiometer to set analog input

Serial port: UART1 and loopback

Example *serial_usart1_loopback.c*

- USART1 has to be setup in CubeMX
  - (to avoid conflicts) USB_OTG_HS:
    - Internal FS Phy: Disable
  - also reset PB0, PA11 and PA12 (pinout view on the right)
  - USART1
    - Mode: Asynchronous:
    - (USART1_TX already on PB6)
    - USART1_RX on PB7
    - Parameter Settings… Baud Rate: 9600
- Pin configuration then appears in *stm32h7xx_hal_msp.c*
- Connect PB6 to PB7 to create h/w loopback

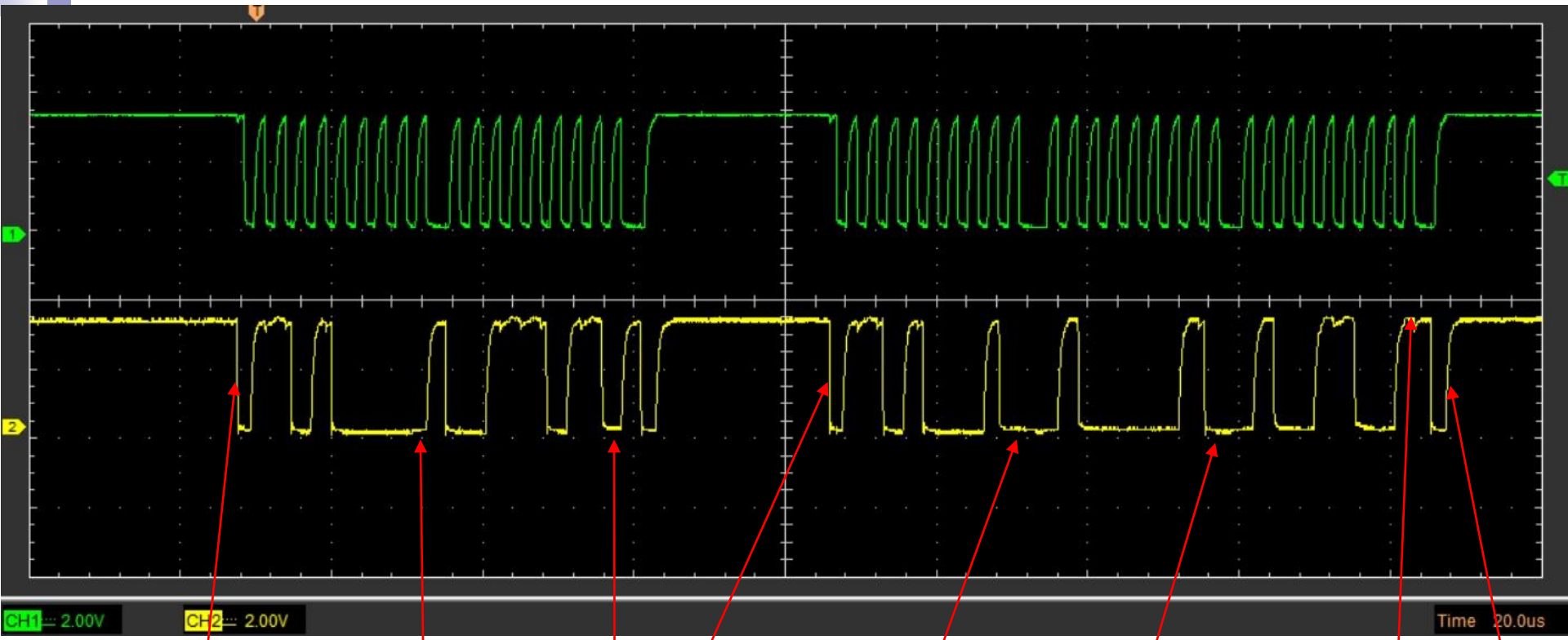
I2C

- I2C4 has to be setup in CubeMX
  - set I2C Speed Mode to Fast Mode
    - frequency should be 400 kHz
  - pull-up resistors are already present on the accelerometer board
  - Maximum Output Speed: very high

- Then, test *I2C.c*

I2C

- I2C slave device (the 6050 accelerometer) is at address 0x68
- to write into a register:
  - write(register address); write(data)
- to read a register:
  - write(register address); read(data)
- 6050 register addresses we use:
  - 0x6b: PWR_MGMT_1
  - 0x3b…: accelerometers
  - 0x41: temperature
  - 0x43…: gyros
- Physical connections for the 6050: pins 1 to 4 are VCC, GND, SCL, SDA respectively (AD0 already connected to GND)
- 5 V are needed to power the module



Master	S	AD+W		RA		S	AD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

Signal	Description
S	Start Condition: SDA goes from high to low while SCL is high
AD	Slave I^2C address
W	Write bit (0)
R	Read bit (1)
ACK	Acknowledge: SDA line is low while the SCL line is high at the 9^{th} clock cycle
NACK	Not-Acknowledge: SDA line stays high at the 9^{th} clock cycle
RA	MPU-60X0 internal register address
DATA	Transmit or received data
P	Stop condition: SDA going from low to high while SCL is high

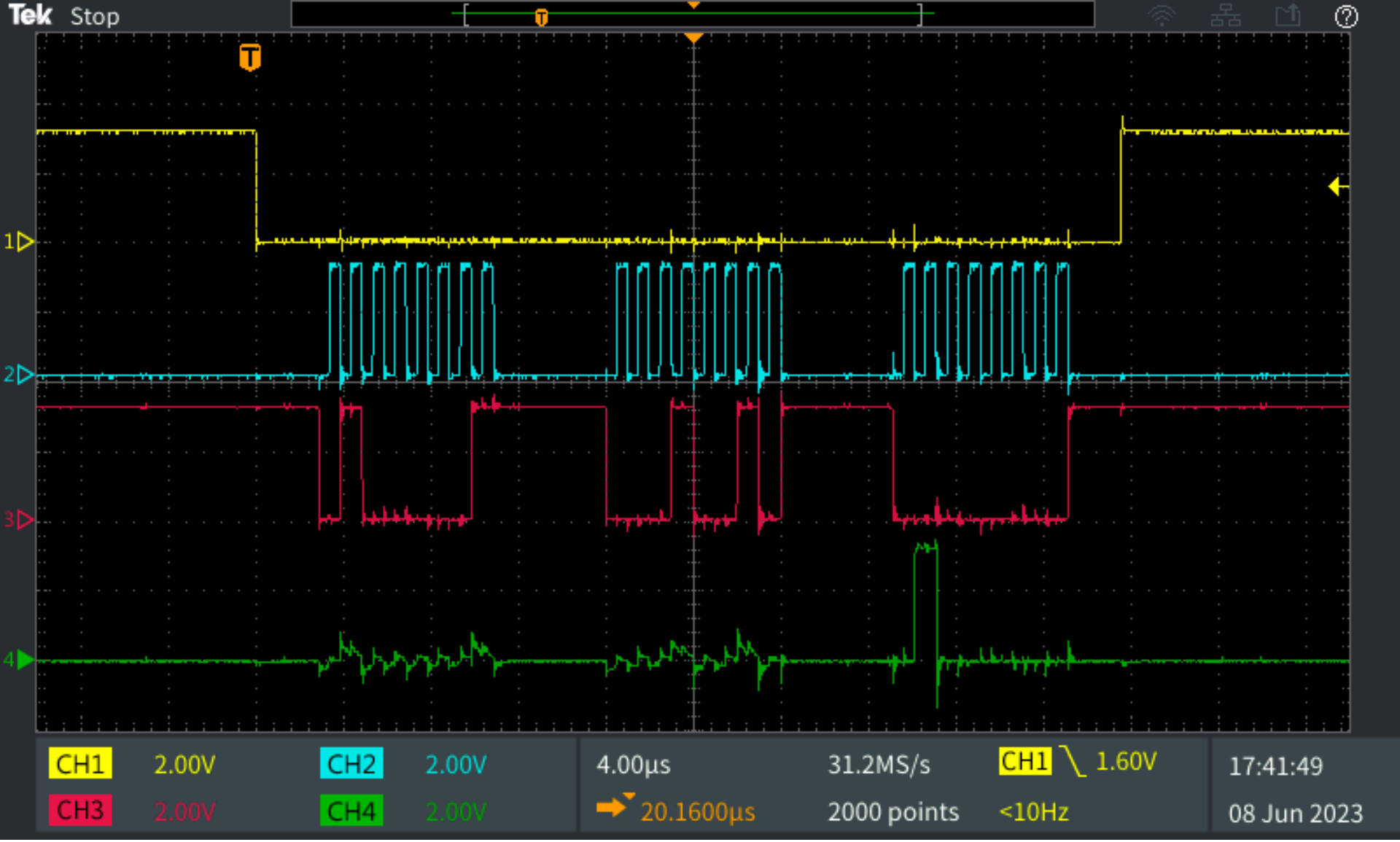
0x68<<1=11010000…0x3b=00111011
0x68<<1+1 = 11010001…byte…byte

SPI

- SPI slave device (the MCP23S17 I/O expander) is at address 0x40 +0x00<<1(if A0, A1 and A2 are set to GND)
- to write into a register:
  - CS low
  - write(device address)
  - write(register address)
  - write(value)
  - CS high
- to read a register:
  - CS low
  - write(device address+0x01)
  - write(register address)
  - value=write(dummy_byte)
  - CS high

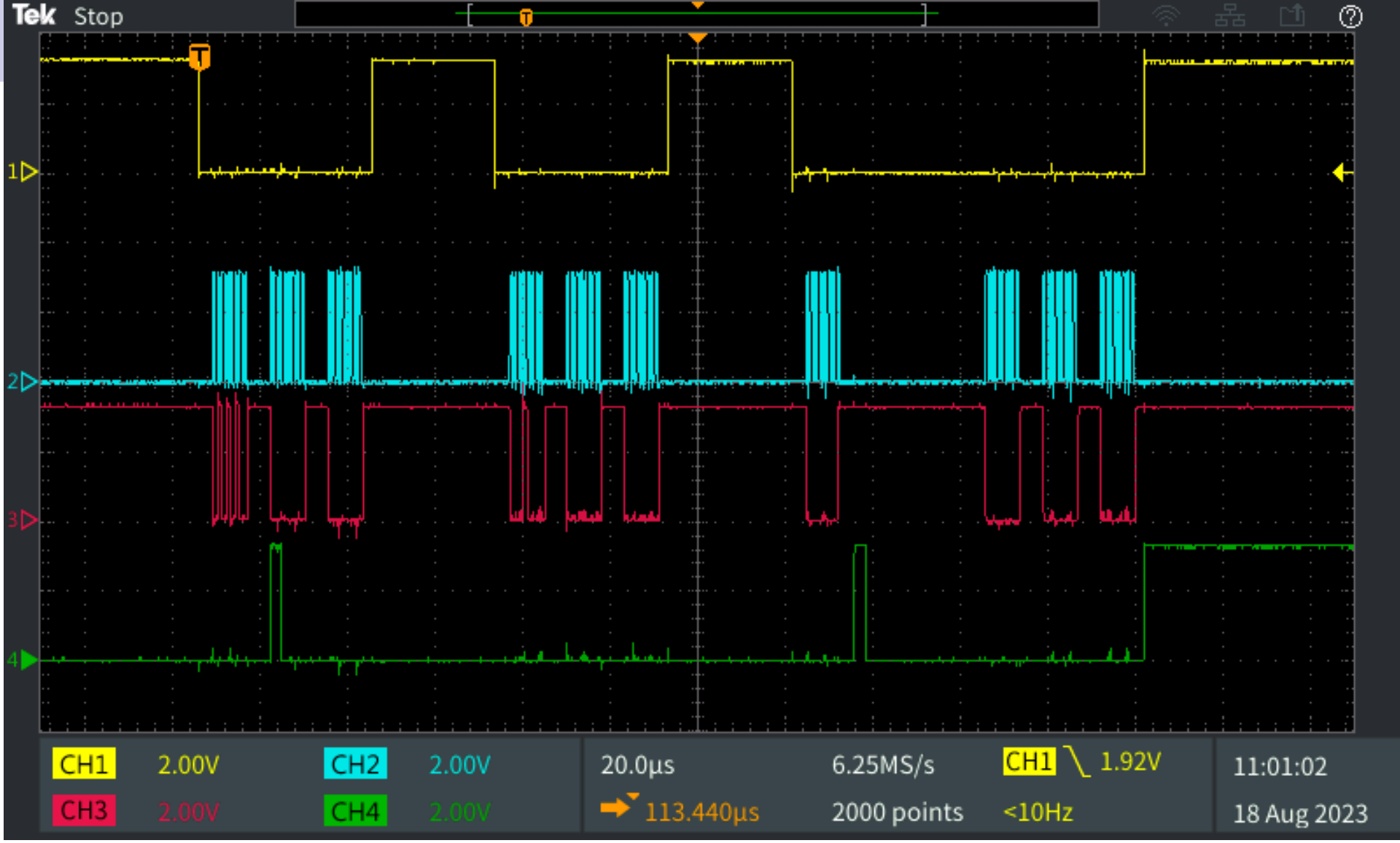
SPI

- During each SPI clock cycle, a full-duplex data transmission occurs:
  - the master sends a bit on the MOSI line and the slave reads it, while the slave sends a bit on the MISO line and the master reads it
  - this sequence is maintained even when only one-directional data transfer is intended
- MCP23S17 register addresses we use
  - 0x00 and 01: IODIRA and B
  - 0x12 and 13: GPIOA and B
- Physical connections for the MCP23S17:
  - pin 11 to 14 are ~CS, SCK, SI, SO
  - pin 15 to 17 are A0 to A2 and ~RESET



Top to bottom: ~CS, CLK, MOSI, MISO

Write 0x41, write 0x42, read 0x64



Top to bottom: ~CS, CLK, MOSI, ~DRDY

This is another example, with the ADS1256 ADC.

USB [UM1734]

- USB support is provided in middleware [UM2217, p. 1]
  - RTOS, TCP/IP and graphics are middleware
- USB can be used for Human Interface Devices (HID), mass storage, audio, communications… let's try communications
- In MX:
  - USB_OTG_HS… Internal FS Phy: Device_Only
  - Middleware and Software Packs… USB_DEVICE…
    - Class For HS IP: Communication Device Class (Virtual Port Com)
- In IDE: test *USB-COM*
  - long compilation time…
  - open a Teraterm connection each time you recompile ☹