Module 3

Serial Peripheral Interface
(SPI)

Tim Rogers 2022

Reading

- Textbook, Chapter 22, Serial Communication Protocols, pp. 527 – 598
  - It’s a long chapter.
  - Let’s first look at Section 22.3, SPI, pp. 568–577.
  - Next, we’ll look at Section 22.2, I2C, pp. 546–567.
  - Don’t worry so much about the USB section.
    - Read that only if you’re curious.
    - Not much we can do with that.
    - Other books are better for understanding USB.

Learning Outcome #3

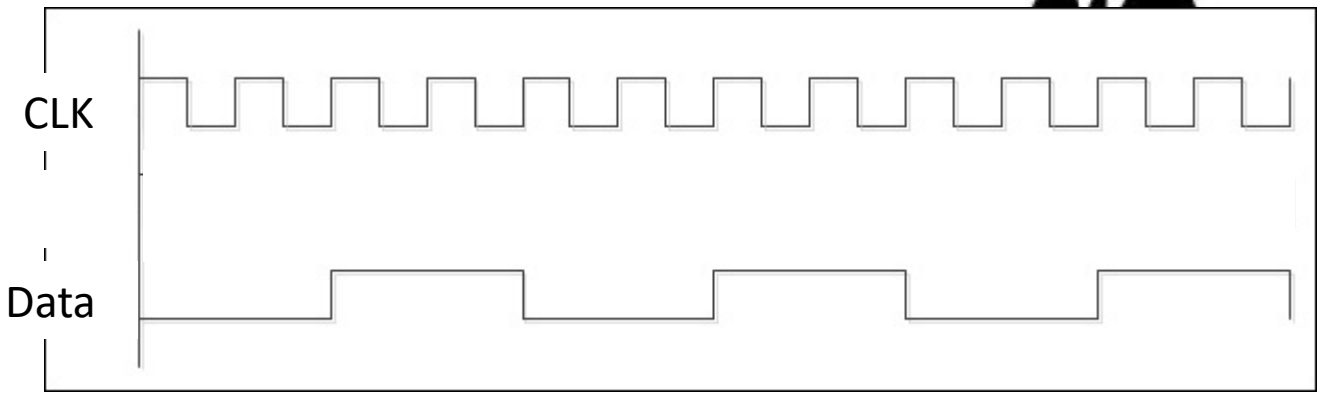
"Serial Peripheral Interface (SPI)"

Why?

Want to communicate with a peripheral

One options:
Send data in parallel

Better option:
Send multi-bit data serially

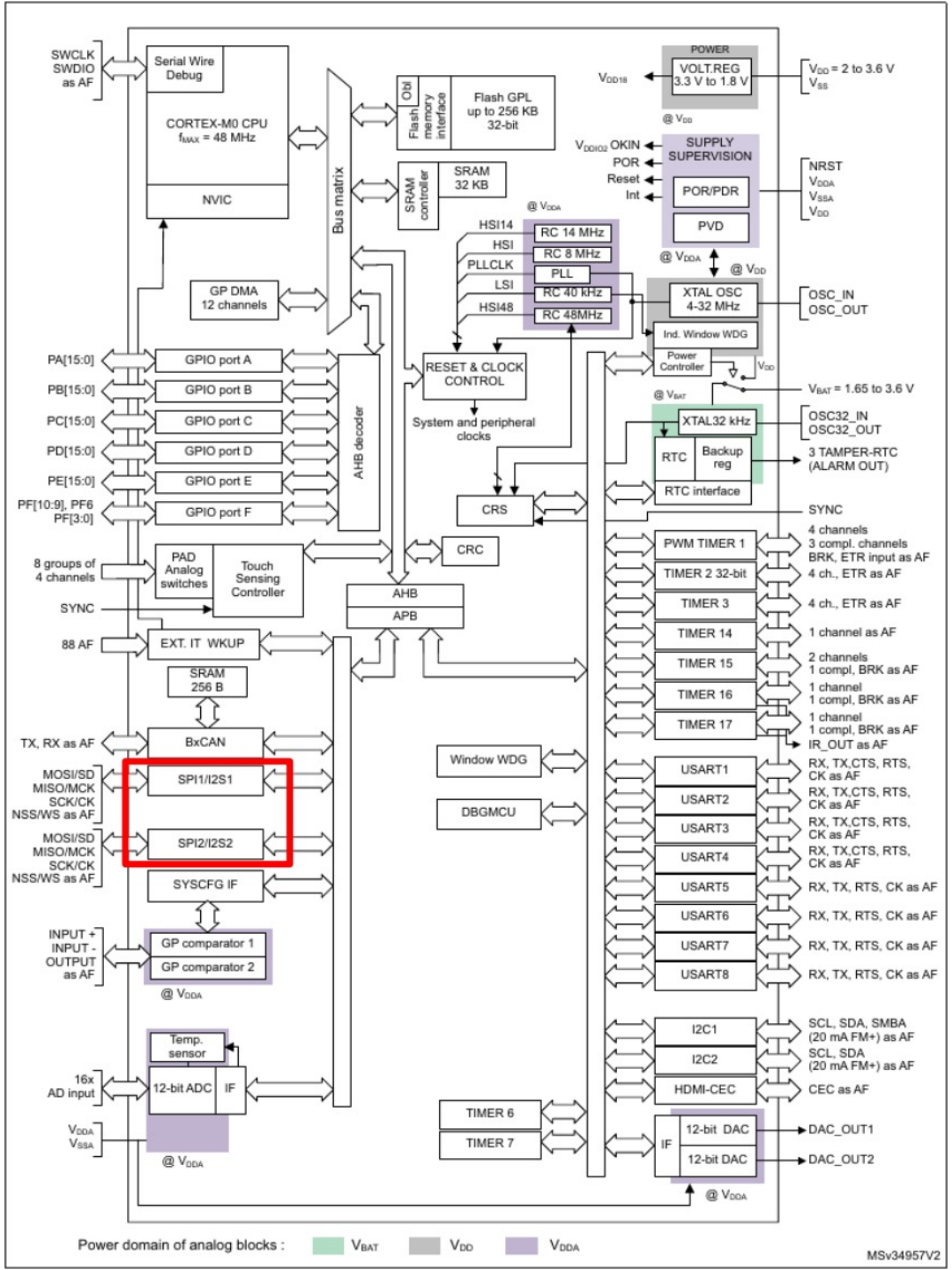


Basic Description

- Communication takes place serially
- Synchronous because a clock signal is involved
- Operates in Master/Slave Mode.
  - Needs to be at least one master – can be multiple slaves
  - Master defines the clock
- Operates via “shifting in” bits each clock cycle

Standardized and used for a variety of peripherals: LCDs, D/A converters, etc…

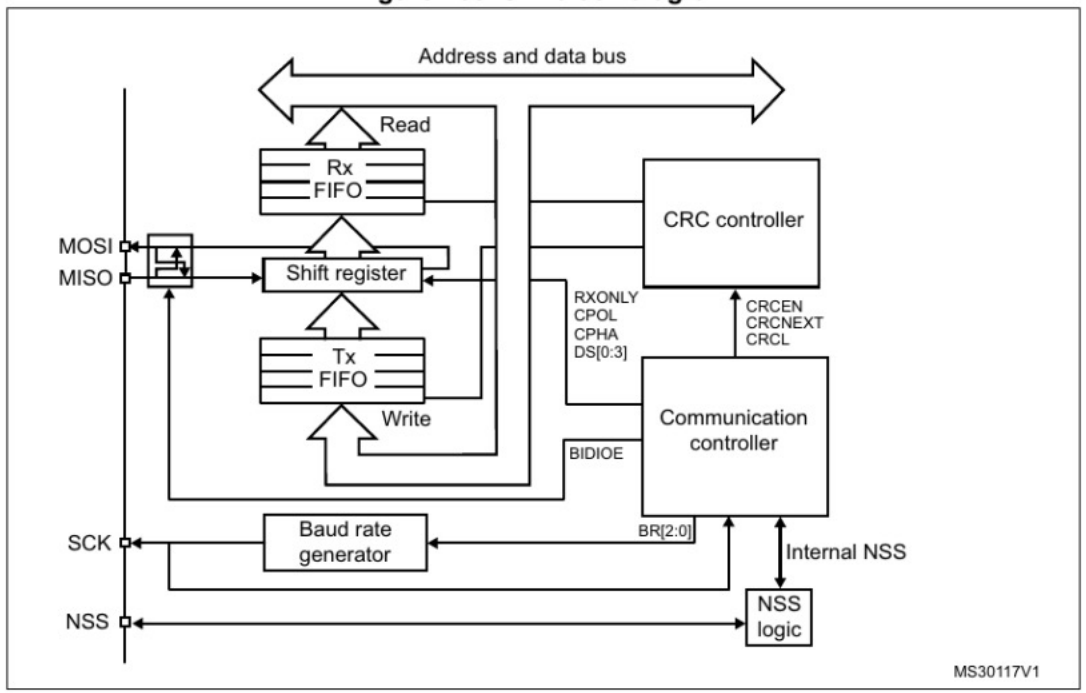
Can also use it to communicate between microcontrollers



STM32 SPI

- Two independent "channels".
  - Turn a parallel word (from 4 – 16 bits) into a serial output stream.
  - Turn a serial input stream into a 4 – 16 bit word.
  - Synchronous clock pulse for each bit.
  - "negative slave select" (NSS) to indicate that a master device is reading or writing.

Figure 266. SPI block diagram



Four I/O pins are dedicated to SPI communication with external devices.

- **MISO:** Master In / Slave Out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** Master Out / Slave In data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** Serial Clock output pin for SPI masters and input pin for SPI slaves.
- **NSS:** Slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - select an individual slave device for communication
  - synchronize the data frame or
  - detect a conflict between multiple masters

  See *Section 27.5.4: Slave select (NSS) pin management* for details.

Block Diagram

- It's a shift register.
- Assume that the microcontroller is the "master device".
  - Output is MOSI: Master Out, Slave In.
  - Input is MISO: Master In, Slave Out
  - Clock is SCK
  - Negative Slave Select is NSS
    - It goes low to indicate that a transaction is in progress

SPI is a synchronous protocol

- A clock pulse accompanies each data bit.
- A clock signal must be delivered to each data recipient.

- By comparison, an asynchronous protocol would require only a data line.

SPI is fairly fast

- STM32 can drive SPI at half the system clock speed (maximum): 24 MHz.
  - Many devices cannot handle this.
- Speed specified in bits/s called a ***Baud rate***
  - After French telegrapher Émile Baudot.
- Baud rate selection allows for up to 256 as a clock divisor. e.g. 48 MHz / 256 = ~ 187 kHz.
  - 187 kHz is the slowest the STM32 can clock SPI.

Master / Slave Devices

- SPI devices are designated as masters or slaves.
  - A master device initiates all data transfer operations.
    - It drives the clock pin and slave select pin(s).
  - A slave device responds to transfer operations.
- Signals are named according to the devices' standpoint:
  - e.g. MOSI: "Master Out, Slave In" is the data transmitted by a master device and received by a slave device.

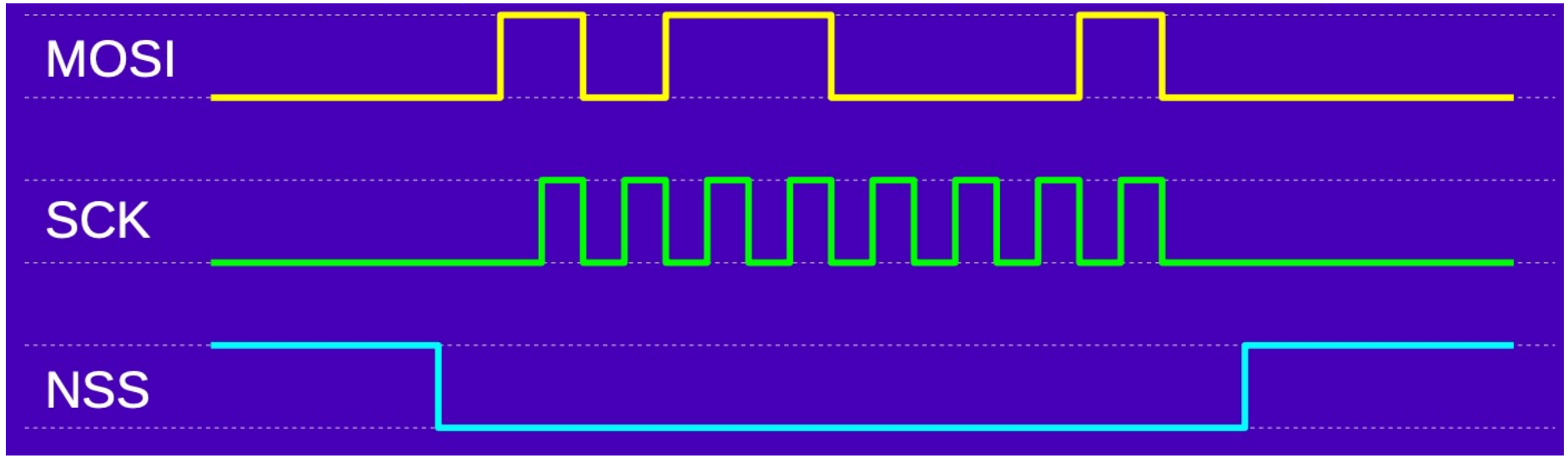
What does serial output look like?

- Note: For this example, we'll send the MSB first.
- Let's look at 8-bit data size first:
- Parallel word:

1	0	1	1	0	0	0	1

- MSB-first serial representation:

In this example, data sent from master to slave on MOSI is latched in to the slave on the rising edge of the each clock.

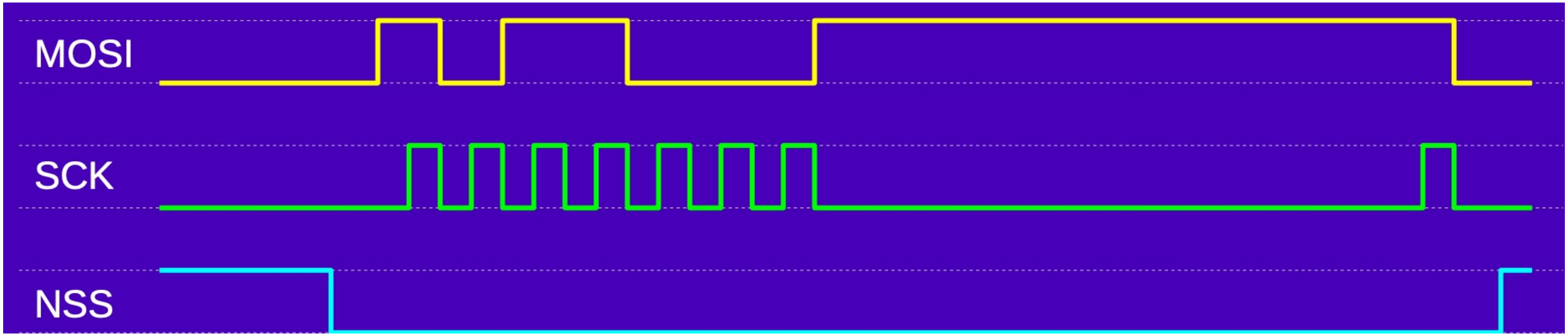


synchronous →clock

enable the transfer

Note about the clock

- Just because it’s called a clock doesn’t mean it must have special properties like periodicity.
- As long as the clock is not too fast, it can have enormous pauses at any point.
- This is a fringe benefit of a fully-synchronous protocol.

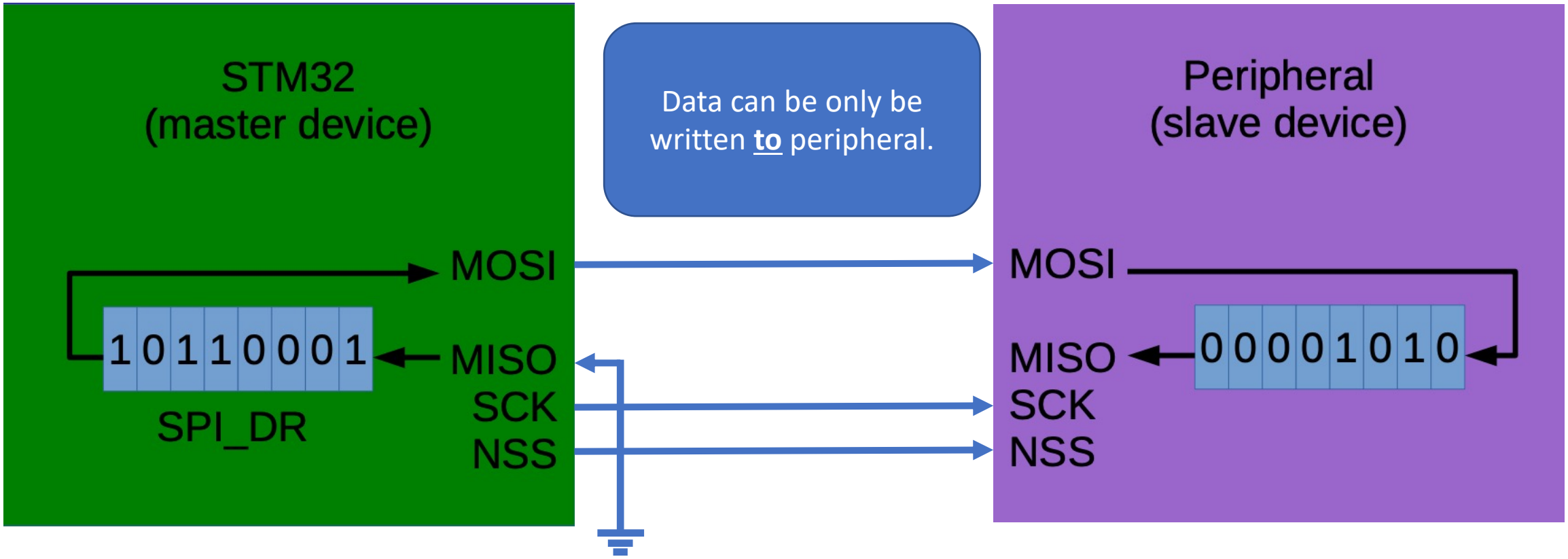


How do we use SPI?

- Single master device to single slave device configuration:

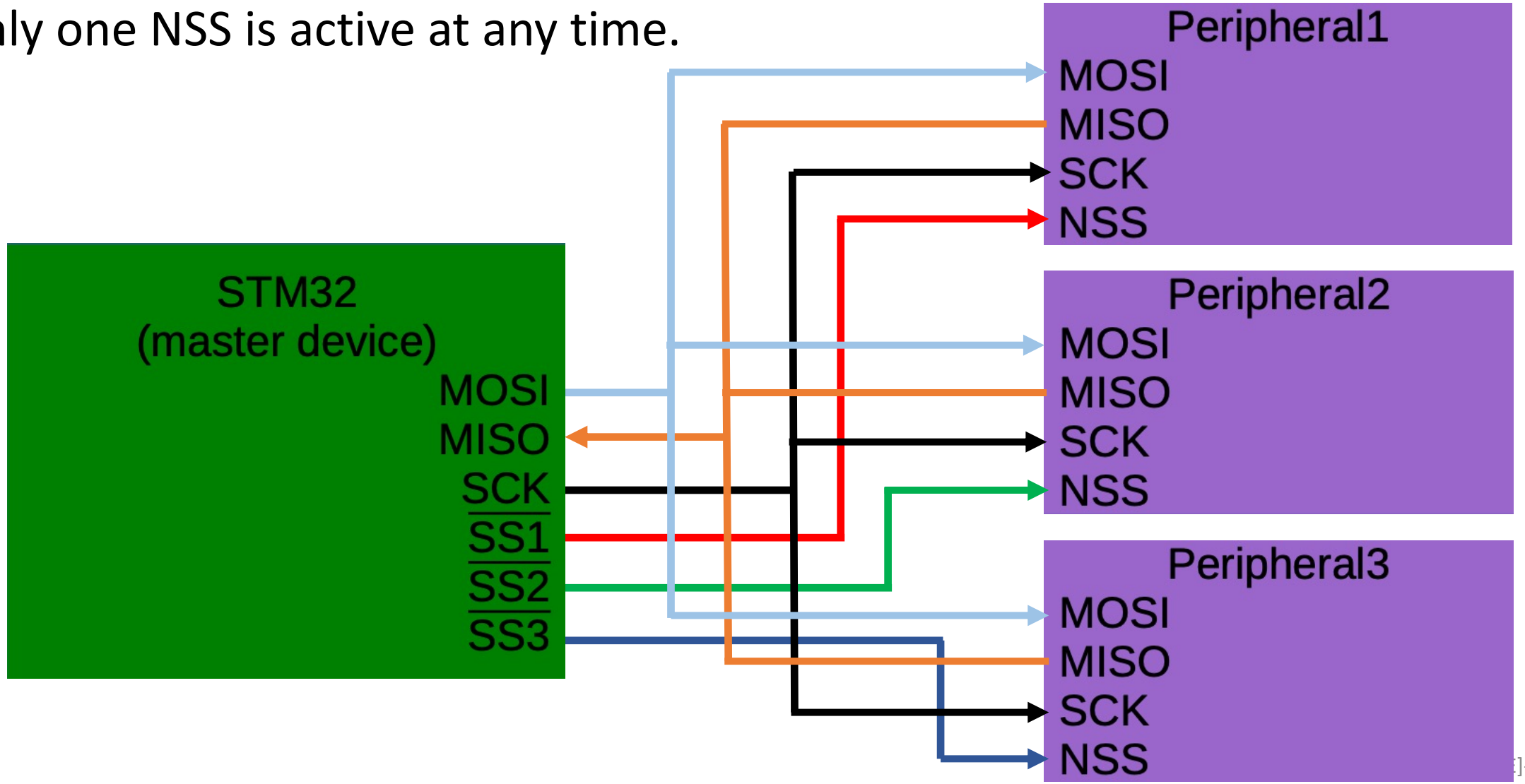
How do we use SPI?

- Single master device to single slave device configuration:
- (no read from slave device)



Multiple slave devices

- Only one NSS is active at any time.



Multiple master devices

- The NSS pin can be driven as well as monitored.
  - It is an error for multiple master devices to assert NSS at the same time.
  - Some coördination is needed to ensure that one master device reads/writes at any time.

Configuring SPI on the STM32

- Everything is more complicated than you want it to be.
  - Enable the appropriate GPIOx pins with alternate function.
  - Enable the clock to SPI1 in RCC_APB2ENR.
  - Enable the clock to SPI2 in RCC_APB1ENR.
  - Set the baud rate. (some fraction of the system clock)
  - Set the data size. (it can be from 4 – 16 bits)
  - Configure the protocol: (These things are interdependent!)
    - Configure the mode, e.g., bidirectional, but currently master.
    - Set the clock polarity and clock phase (default zero for both).
    - Enable output of NSS, and use NSSP to strobe NSS automatically.
  - Enable the SPI channel.

Not all combinations allow NSSP. See FRM 27.5.5 & 27.5.6

Configuration example

```
RCC->APB2ENR |= RCC_APB2ENR_SPI1EN;
SPI1->CR1 |= SPI_CR1_MSTR | SPI_CR1_BR;
SPI1->CR1 |= SPI_CR1_BIDIMODE | SPI_CR1_BIDIOE;
SPI1->CR2 = SPI_CR2_SSOE | SPI_CR2_NSSP | SPI_CR2_DS_3 | SPI_CR2_DS_0;
SPI1->CR1 |= SPI_CR1_SPE;
```

- BR[2:0] selects the SCK divisor. fSCK = fSYSCLK / 2(1+BR[2:0])
  - e.g. when BR is ‘111’ fSCK = 48 MHz / 256
- DS[3:0] selects the data size.
  - ‘1001’ selects a 10-bit word size.
  - **WARNING**: You must set this. Do not clear and ‘OR’ it. Why?...

CR2 DS Initialization Hazard

Bits 11:8 **DS [3:0]**: Data size

These bits configure the data length for SPI transfers:

0000: Not used
0001: Not used
0010: Not used
0011: 4-bit
0100: 5-bit
0101: 6-bit
0110: 7-bit
0111: 8-bit
1000: 9-bit
1001: 10-bit
1010: 11-bit
1011: 12-bit
1100: 13-bit
1101: 14-bit
1110: 15-bit
1111: 16-bit

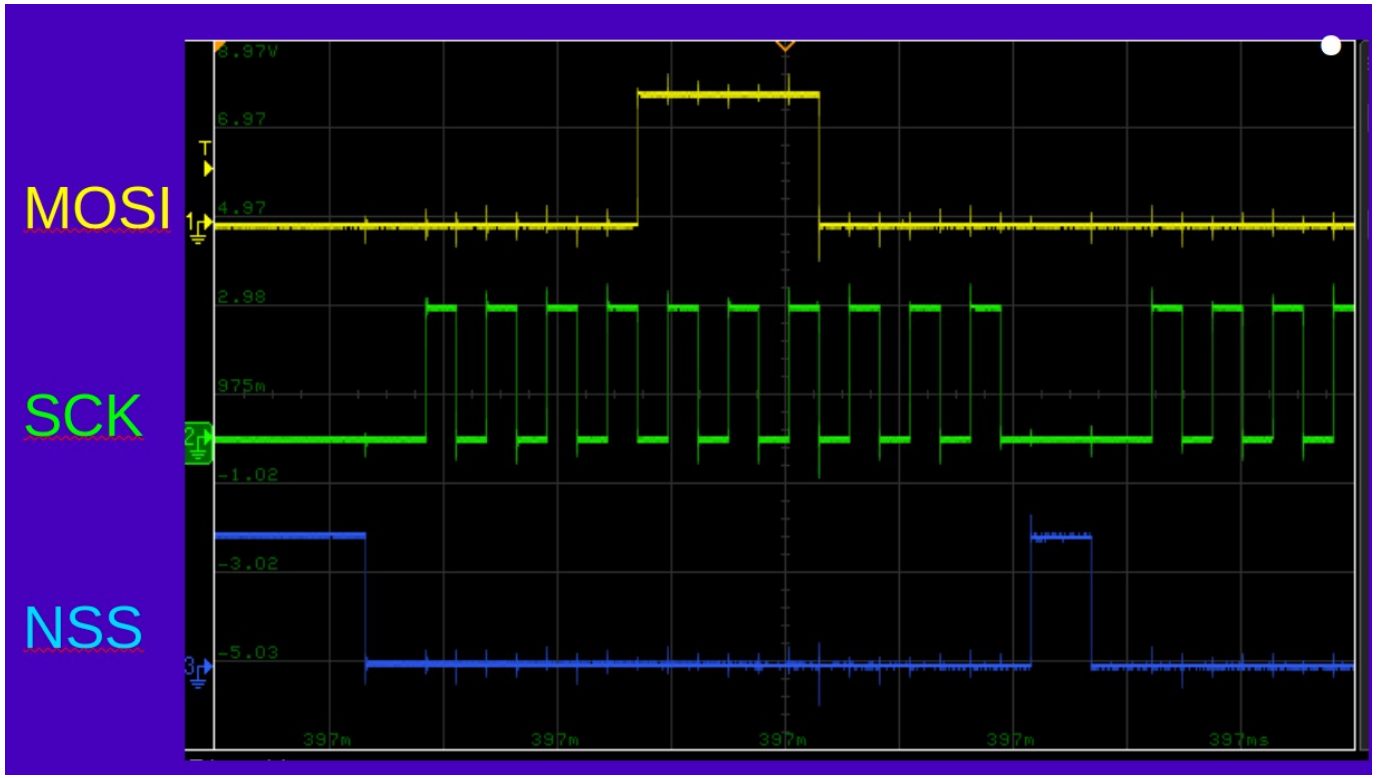
If software attempts to write one of the “Not used” values, they are forced to the value “0111”(8-bit).

- The DS (data size) field in SPIx_CR2.
- Pay attention to the note very carefully.

10-bit word size?

- Yes, this is strange.
- That’s what the OLED LCD display requires.
- You may not ever see anything else that uses a 10-bit SPI interface.

Example waveforms



- This works with the OLED display.
  - Initialization sending 0x038 prefaced with two ‘0’ bits.
  - MOSI changes on falling edge of SCK.
  - MOSI data is read by peripheral on rising edge of each clock. (that’s when data is stable)

Reading/Writing SPI

- Once SPIx peripheral is configured, data can be sent and received using the SPIx_DR register:
  - SPI1->DR = 0x038;
  - int x = SPI2->DR;
- Different configuration options (in SPIx_CR2) available to coördinate sending and receiving:
  - TXEIE: Tx buffer empty interrupt enable
  - RXNEIE: Rx buffer not empty interrupt enable
  - TXDMAEN: Tx buffer DMA enable
  - RXDMAEN: Rx buffer DMA enable

8-bit Usage Hazard

- When writing to the SPI output, you write a 16-bit word to SPIx_DR by default.
  - For an 8-bit word size, it puts two bytes into the transmitter buffer.
    - Which byte gets sent first?
  - What if you want to write only one byte?
    - You can do a one-byte store to SPIx_DR
    - (e.g. use the STRB assembly language instruction).
    - You can cast the SPIx_DR to an 8-bit integer:
    - e.g., ***(uint8_t *)&SPI1->DR = one_byte_value;**
    - See Appendix A.17.3.

Other interesting configurations

- Cyclic Redundancy Check (CRC)
  - After sending/receiving a group of words, compute a multi-byte checksum.
    - Uses mathematical field theory to do this.
  - If receiver does not compute the same value, using the same algorithm, it indicates an error in transmission.
  - Much better than parity check of RS-232.
    - Parity is a degenerate form of 1-bit CRC.
  - Does not handle error correction.

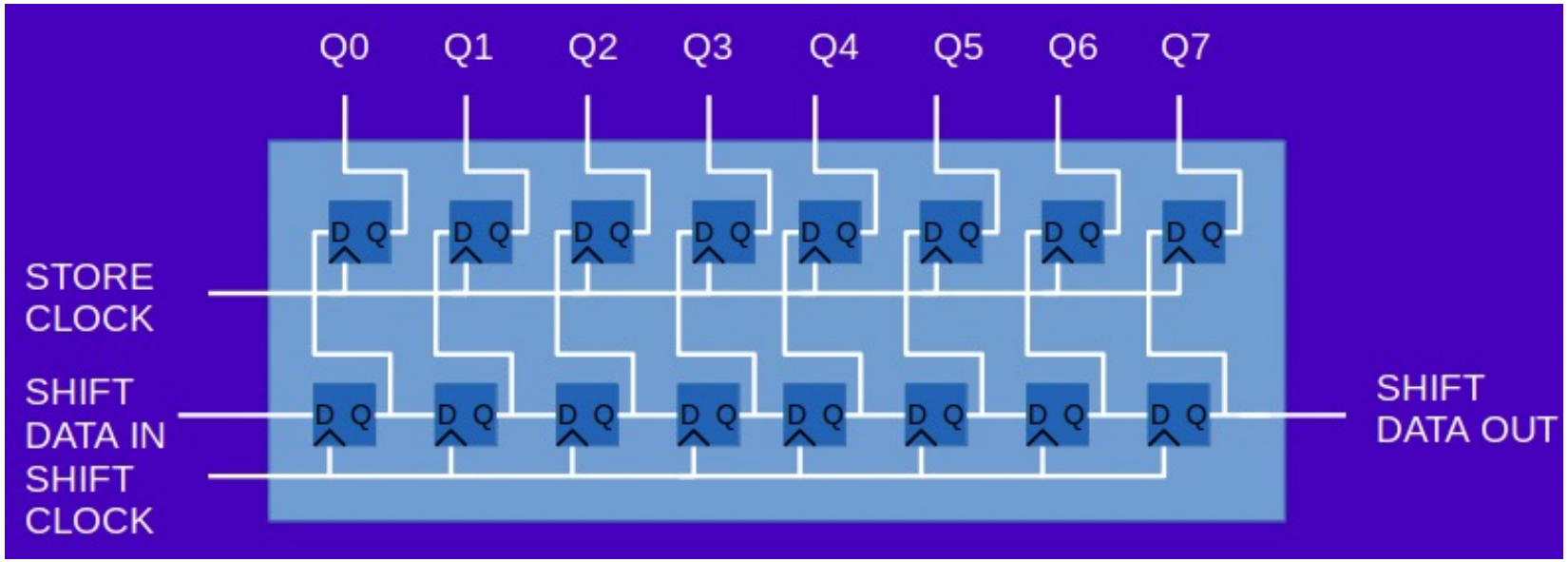
What do we use SPI for?

- There are hundreds of thousands of devices in the world that support SPI protocol:
  - The OLED LCD display in your lab kit
  - The graphical display in your lab kit
  - The SDcard interface on the graphical display in your lab kit
- Commercial/commodity SPI peripherals are not the only thing to use SPI for
- It is easy to construct your own SPI devices

How a Shift Register works

- One data input pin to specify the next bit.
- One clock pin shifts in the new data.
- One more pin to “store” or “output” the data newly shifted in.
- (One more pin to shift data out of the top bit of the internal shift register. This allows you to chain multiple shift registers.)

74HC595

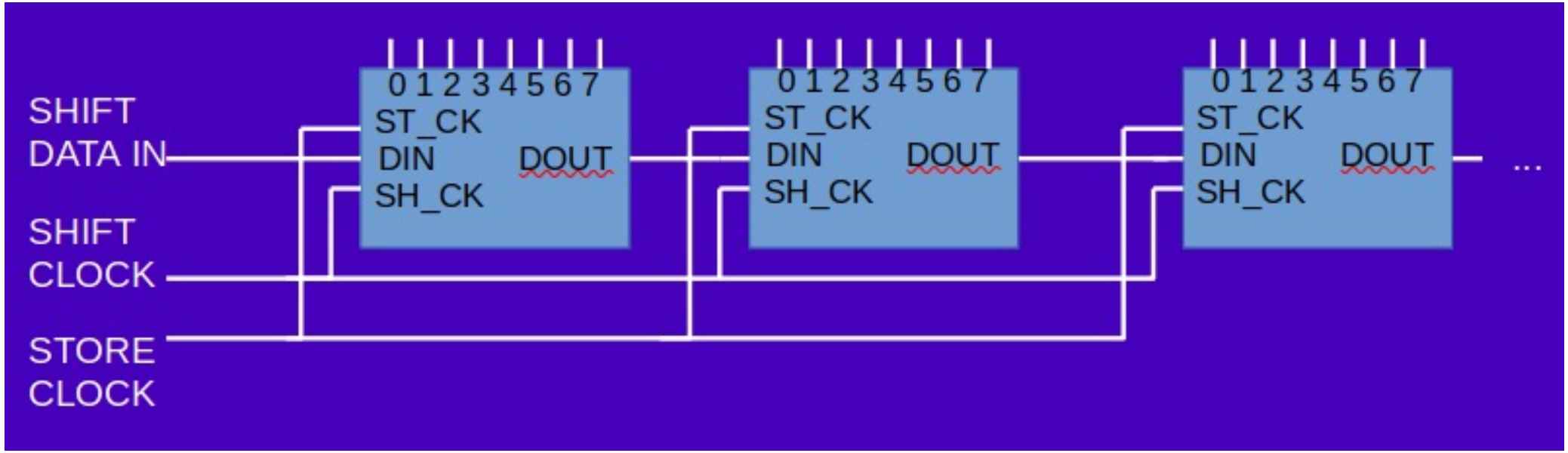

Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
D Q
STORE
CLOCK
SHIFT
DATA IN
SHIFT
CLOCK
SHIFT
DATA OUT

Shift Register Use

- Shift a pattern into the shift register with DIN and SH_CK.
- Store the pattern to the output registers with ST_CK.
- Leave it there while you shift in a new pattern.
- This provides 8*N outputs by using only 3 GPIO pins to drive the N shift registers.

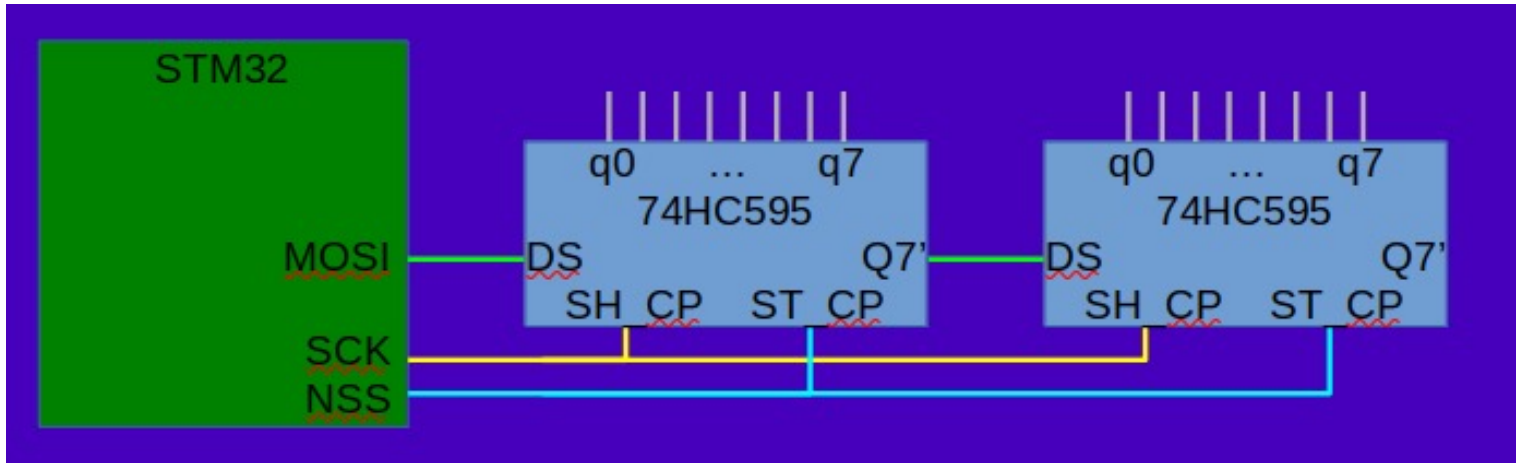
- This is the basis for simple serial peripherals.

Chaining 74HC595 Shift Registers



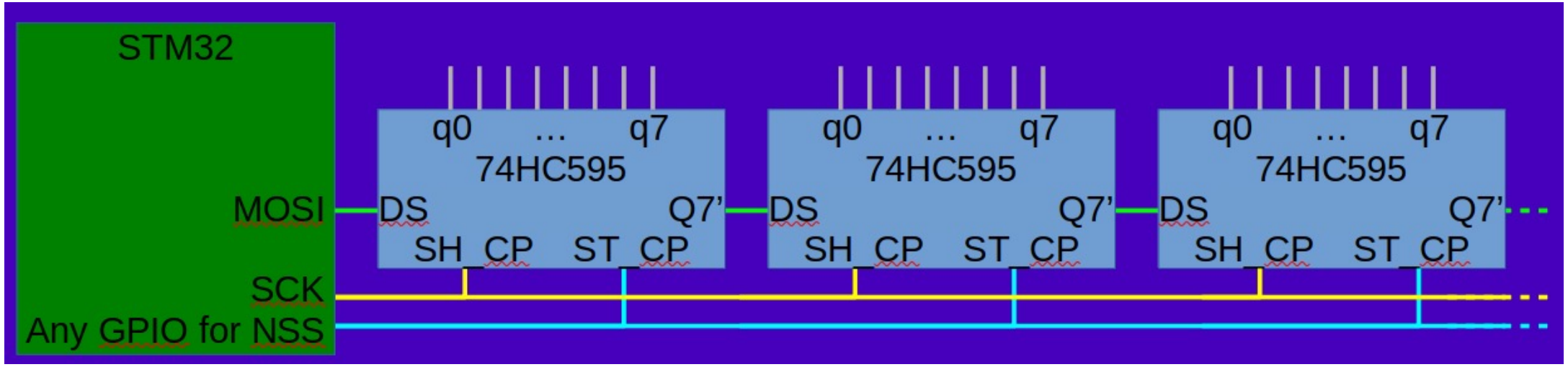
Applications of SPI

- If you wanted a simple means of loading a 16-bit shift register (e.g. two 74HC595s), you could use SPI for this. (But not for more than 16 bits if you want to use automatic NSS.)



Longer chains of shift registers

- If you use GPIO instead of an automatic NSS pin, then you can have an SPI chain of any length. Just issue multiple words of output.

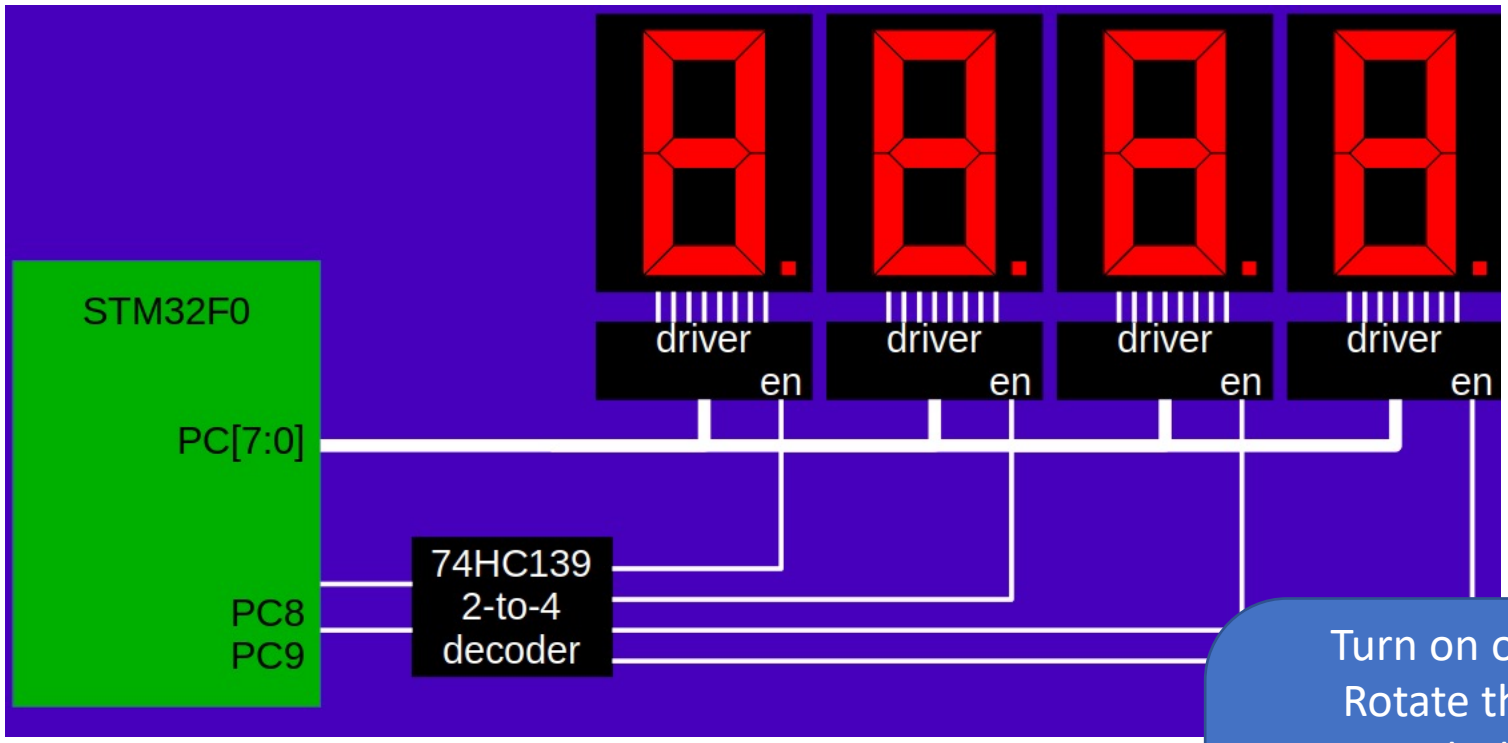


Must wait for SPI Tx buffer to be empty before setting GPIO NSS high.

The Gentle Art of Multiplexing

- Consider the seven-segment displays in your development kit.
  - 8 wires connected to 8 GPIO ports through a driver.
  - Not enough pins to drive lots of 7-segment LEDs.
- If you want an LED display on your project, you will need to multiplex the LED segments.

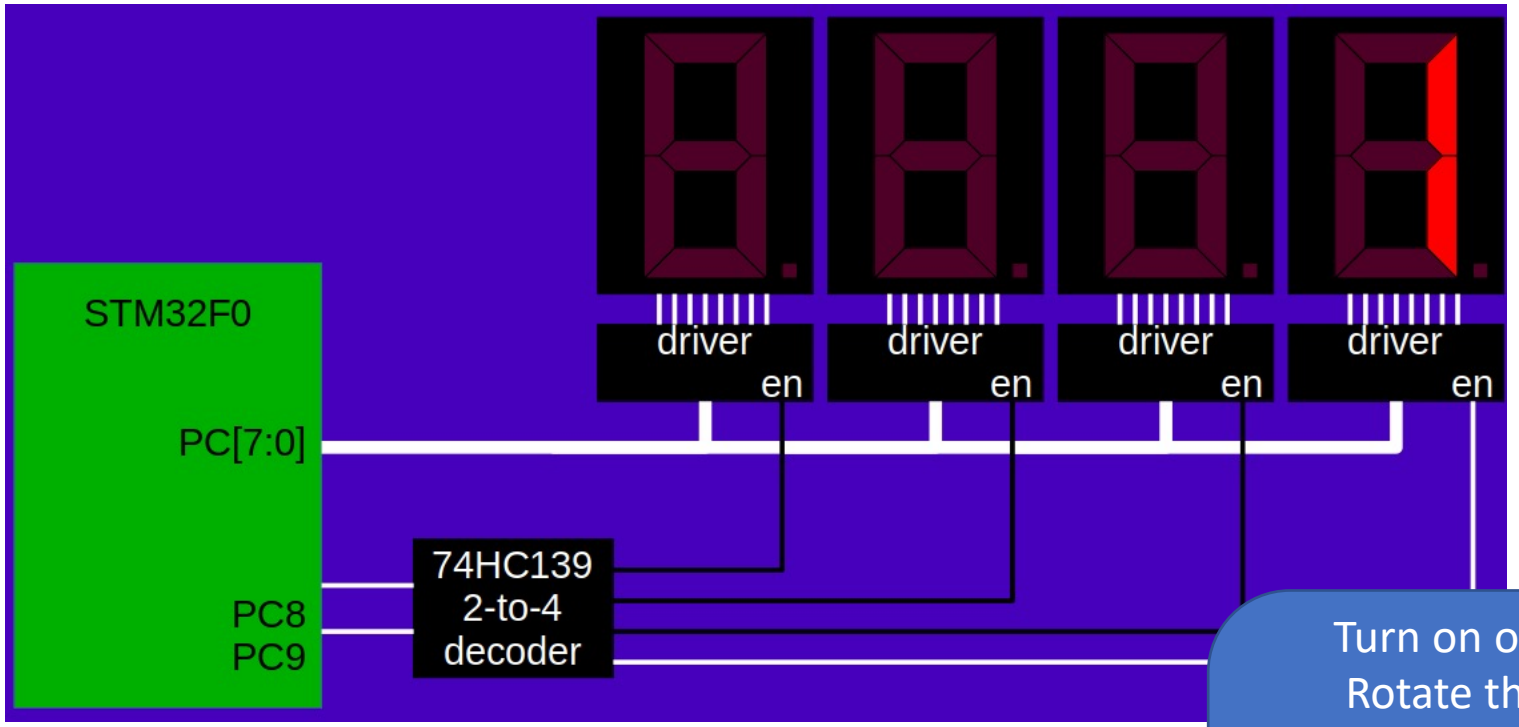
Example



Turn on one display at a time. Rotate through them rapidly enough that your "persistence of vision" makes it appear they are all on simultaneously and displaying different digits.

Four displays with 10 GPIO pins.

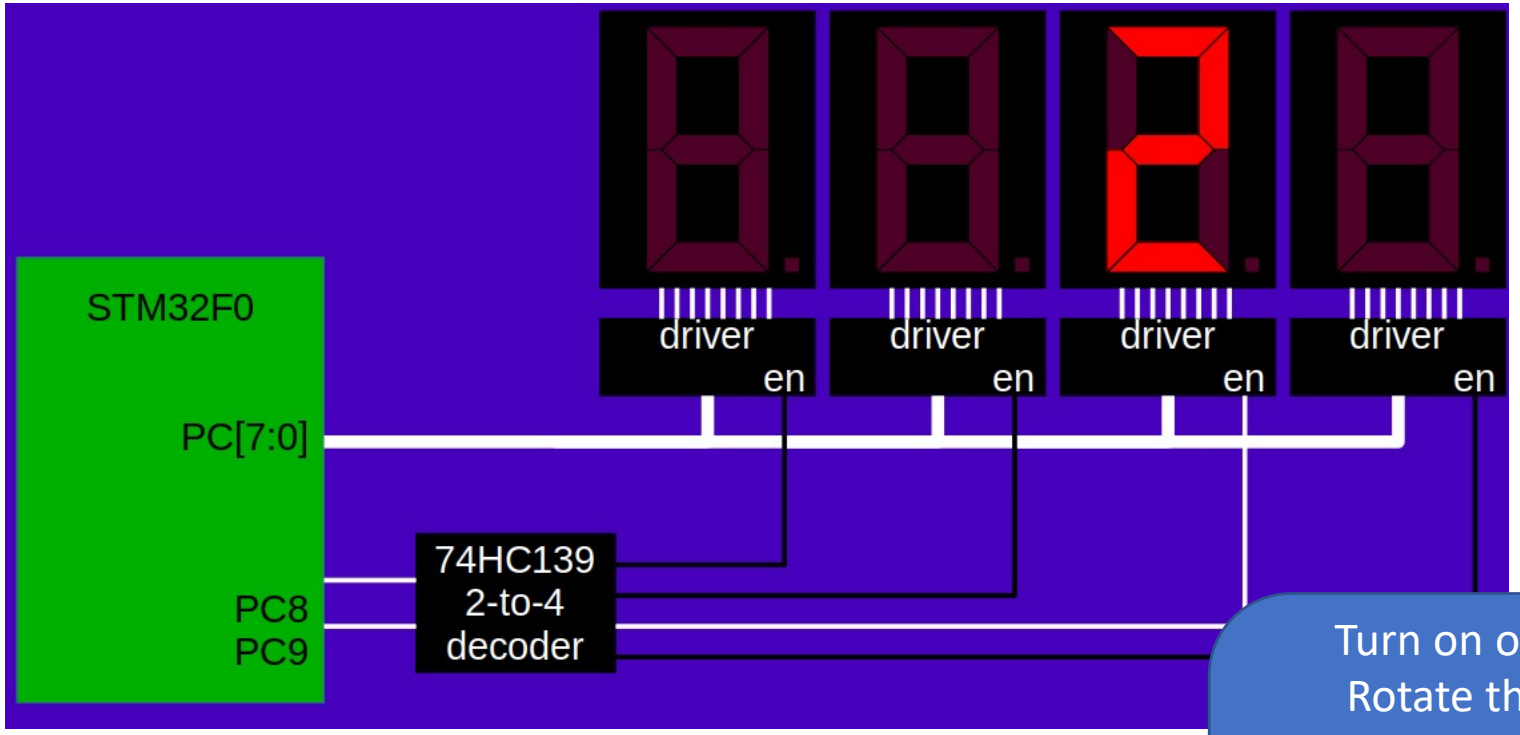
Example



Turn on one display at a time. Rotate through them rapidly enough that your "persistence of vision" makes it appear they are all on simultaneously and displaying different digits.

Four displays with 10 GPIO pins.

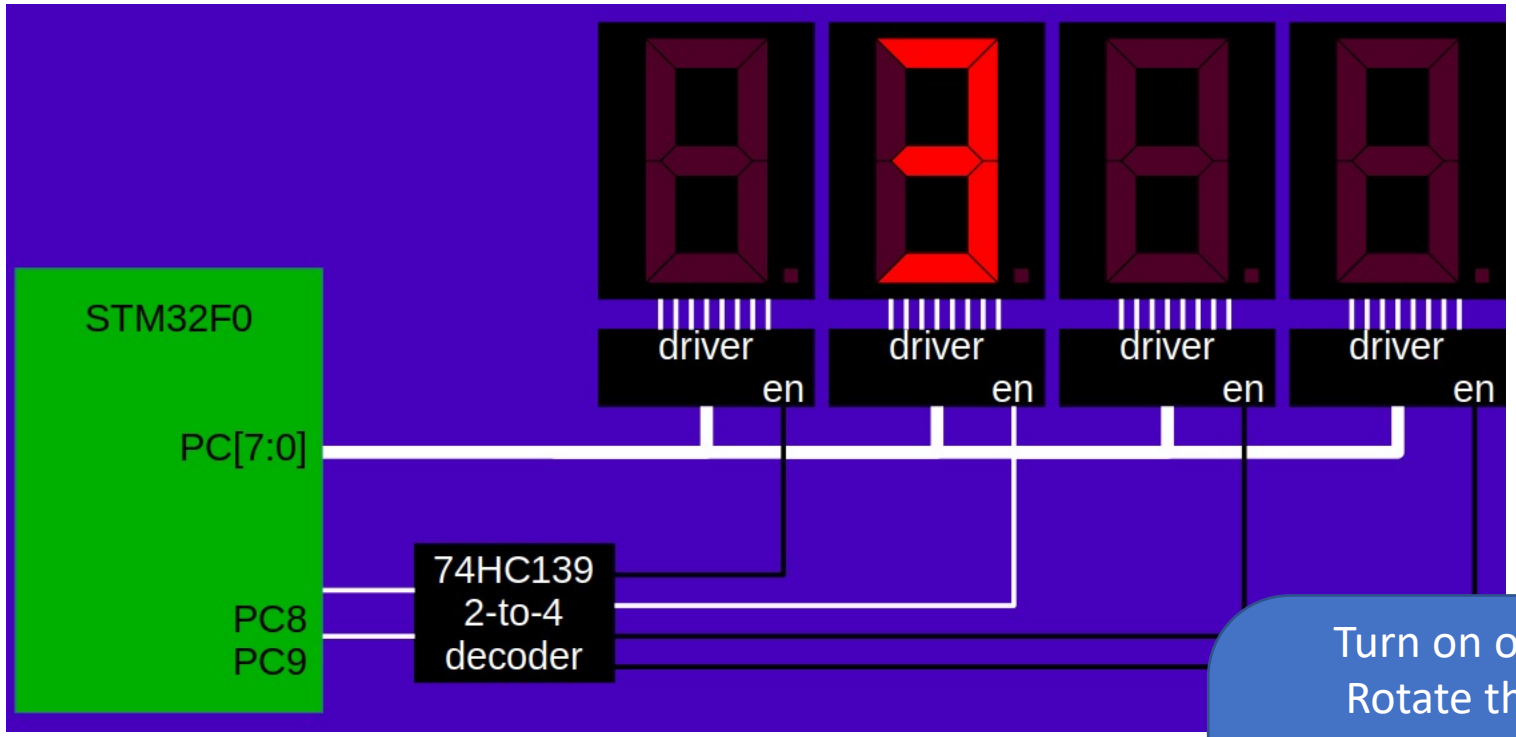
Example



Turn on one display at a time. Rotate through them rapidly enough that your "persistence of vision" makes it appear they are all on simultaneously and displaying different digits.

Four displays with 10 GPIO pins.

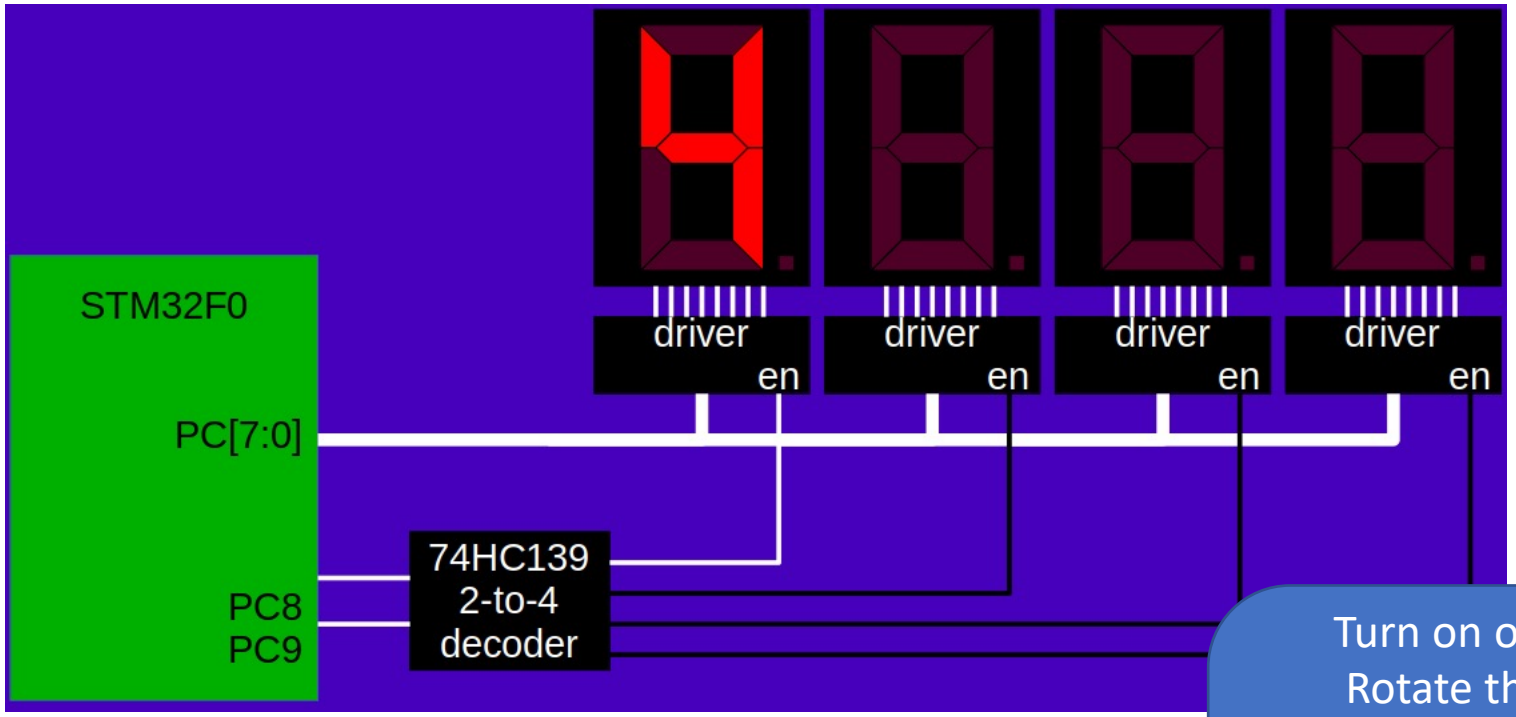
Example



Turn on one display at a time. Rotate through them rapidly enough that your "persistence of vision" makes it appear they are all on simultaneously and displaying different digits.

Four displays with 10 GPIO pins.

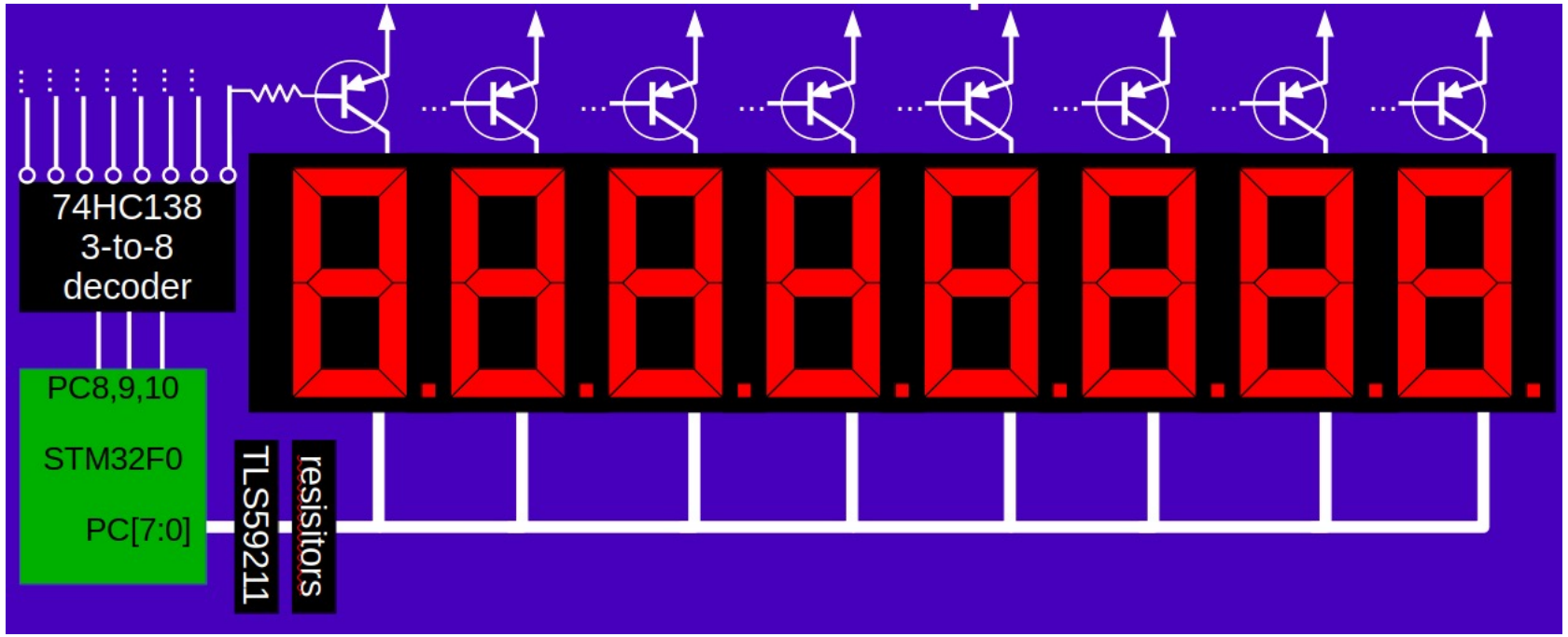
Example



Turn on one display at a time. Rotate through them rapidly enough that your "persistence of vision" makes it appear they are all on simultaneously and displaying different digits.

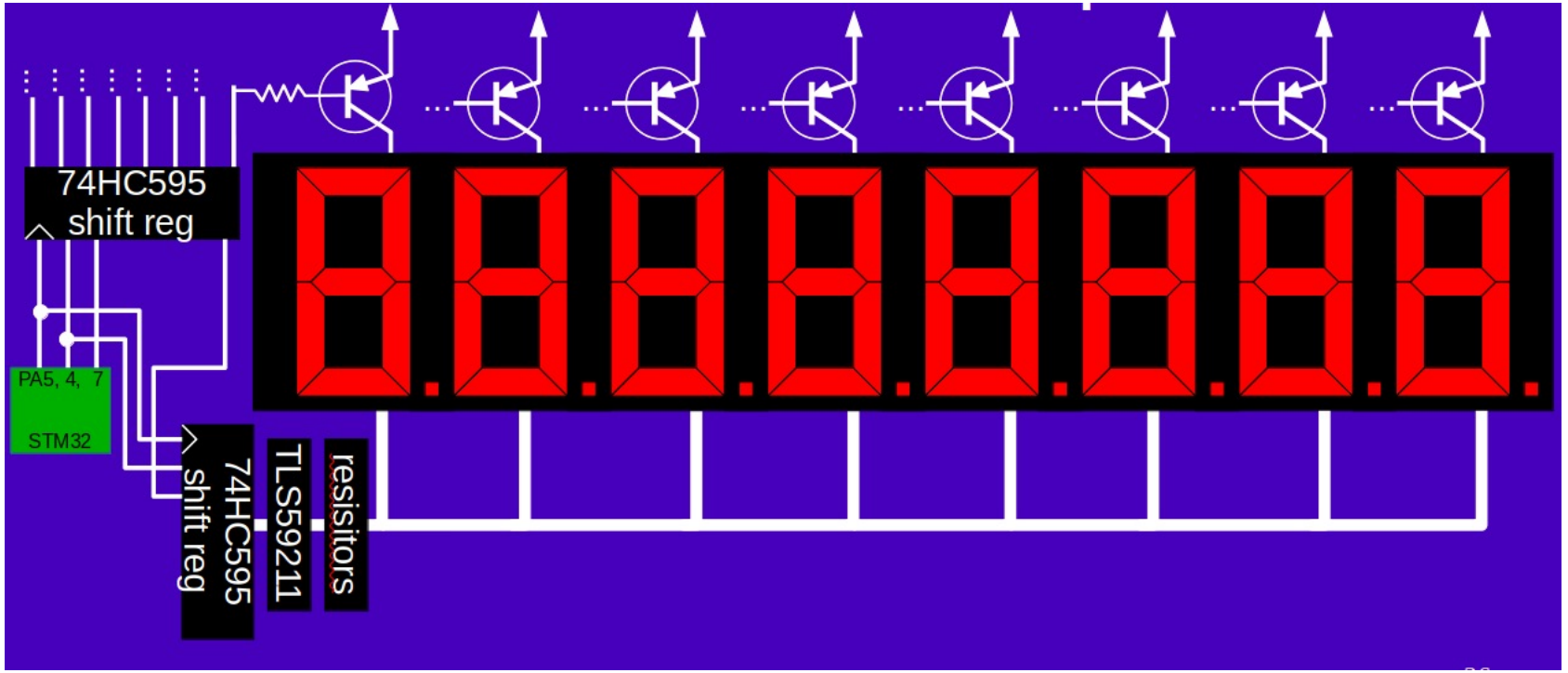
Four displays with 10 GPIO pins.

A Better Example



Only using 11 pins on the STM32

Even Better Example



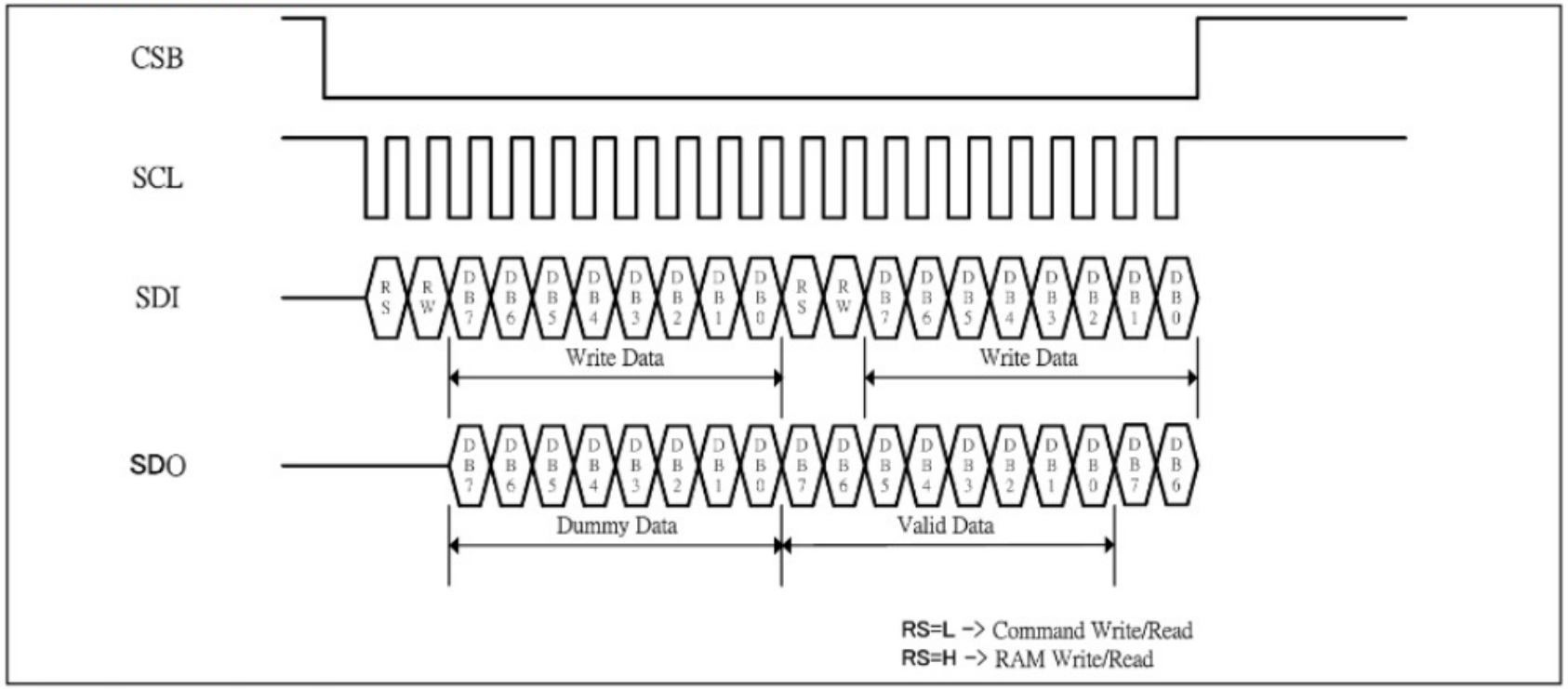
Only using 3 pins on the STM32

Bidirectional SPI Protocol

- Each SPI transaction exchanges two words (4-16 bits each) between two devices. Most devices do not independently and continually send data to the other. Usually,
  - the master device sends a command and ignores the response from the slave device
  - the slave device responds while the master sends nothing
- The master device must send something in order to strobe the SCK signal. It sends dummy data (a zero or 0xff word that the slave device does not interpret it as a new command)

OLED Bidirectional Protocol

Instruction transmission:

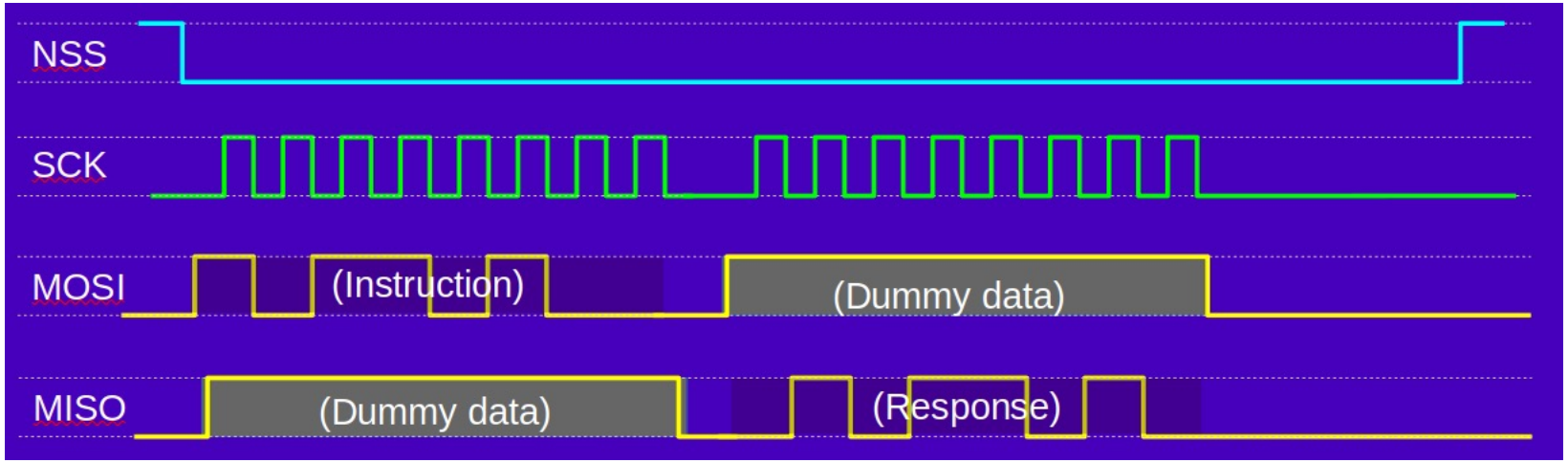


*Note: RS and RW should be used between each instruction.

Secure Digital (SD) Media Cards

- Consider the pins of a (full size) SD card:
- nCS (NSS)
- DI (MOSI)
- VSS (Gnd)
- VDD (3.3V)
- CLK (SCK)
- VSS
- DO (MISO)
- NC
- NC

Simplified SD Card Example



SD card commands

- The host (master device) sends 48-bit chunks to the card (slave device) that are:
  - 1 byte command
  - 4 byte argument
  - 1 byte CRC
- Commands do the following:
  - Initialize the card
  - Prepare to read a block.
  - Check if block is ready to read.
  - Read the block.
  - Prepare to write a block.
- These **blocks** are not files. They're just linearly addressed chunks of data on the storage device.

File systems

- You might not want to just write and read blocks.
  - FATFS: a library for reading/writing Microsoft FAT/exFAT filesystem on an SD card.
  - We'll have some more information about this on the course web page when we can.