POLITECNICO
DI MILANO

www.polimi.it





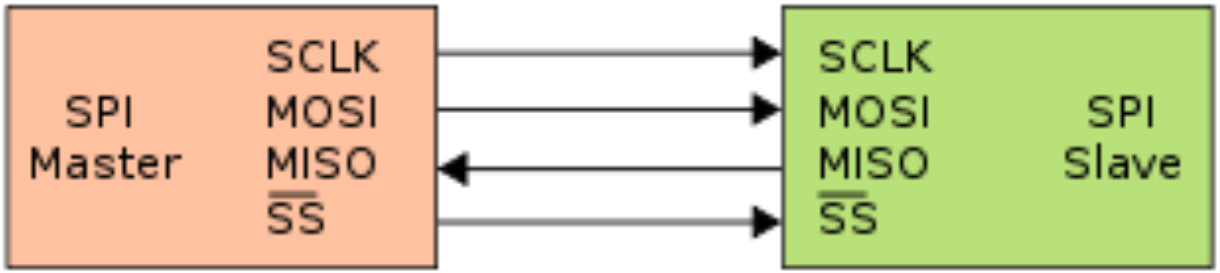
STM32 – SPI

Dr. Enrico Conca

Serial Peripheral Interface

=

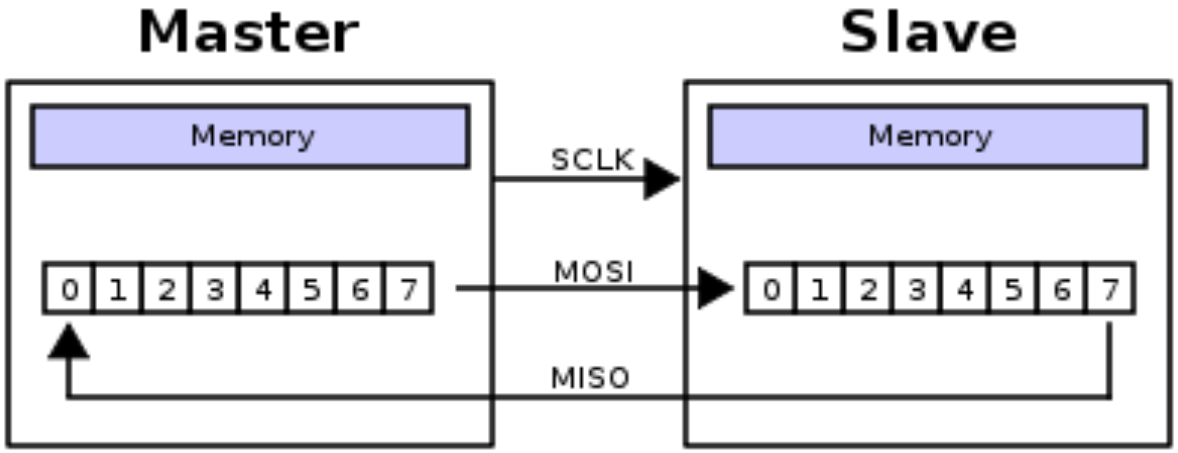
Synchronous serial communication interface used for short-distance communication



Full-duplex (bidirectional) communication using a single-master multi-slave architecture.

Slaves are individually selected by means of the **Slave Select (SS)** – also called **Chip Select (CS)** – line, asserted by the master. Often, these lines are active LOW, as indicated by the overbar: the line must be pulled LOW to enable the selected slave device.

SPI - Functionality



MOSI: Master Out, Slave In

MISO: Master In, Slave Out

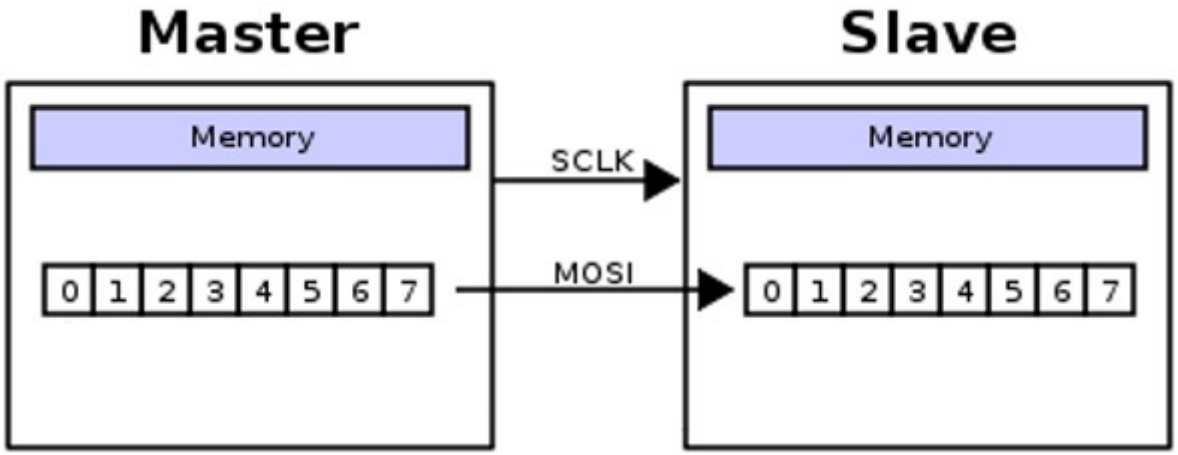
SCLK: Serial Clock

Transmissions normally involve two shift registers of some given word-size, such as eight bits. During each clock cycle, the master sends a bit on the MOSI line and the slave reads it. At the same time, the slave outputs a bit on the MISO line and the master reads it (“virtual ring topology”)

Data is shifted out MSB first.

How many wires?

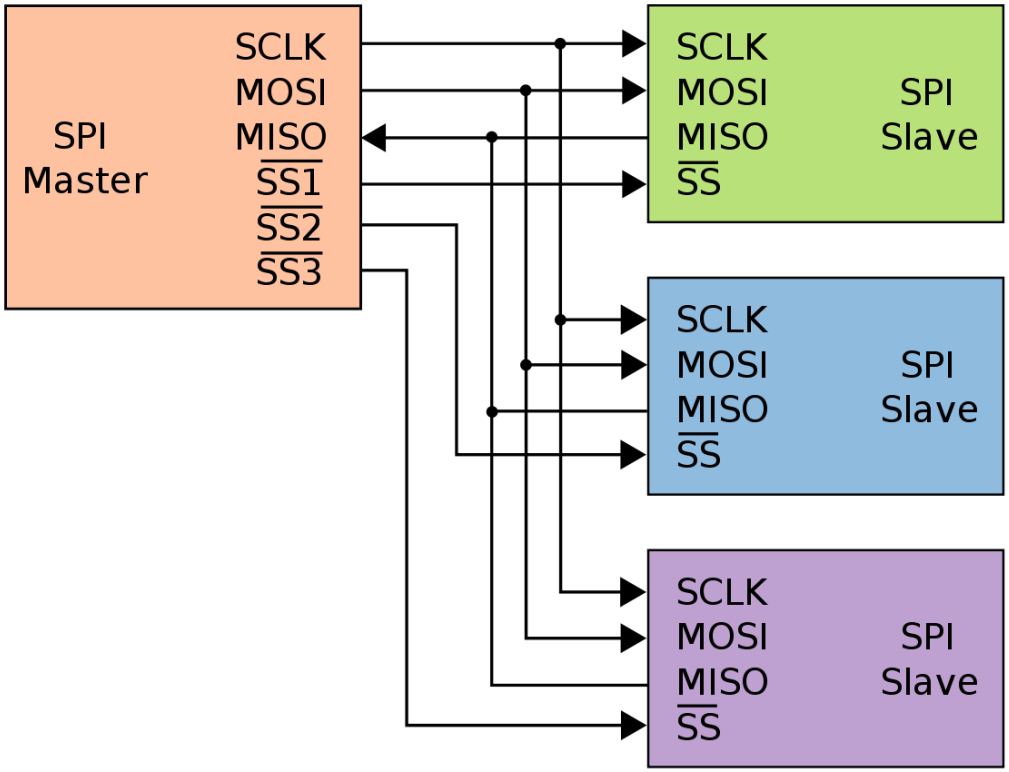
Often called ‘4-wires protocol’ → but you don’t always need 4 wires!



…but often components have other auxilary pins, that might be syncronous to the SCLK or not. E.g. :

- Reset (sync or async)
- Output enable (OE)
- …

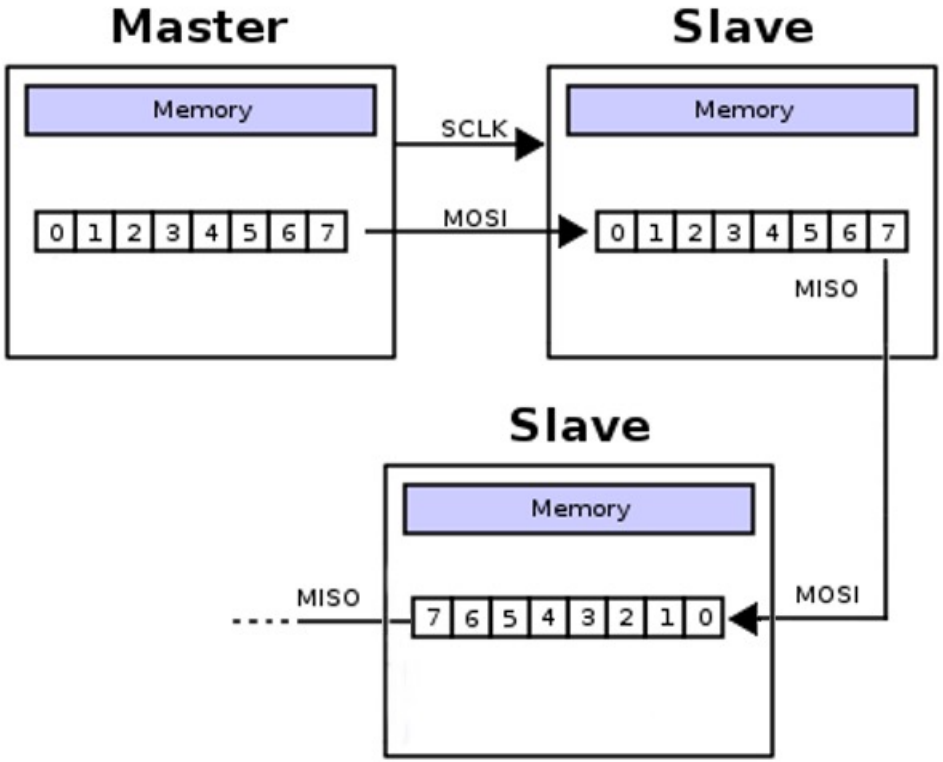
Multi-slave configurations

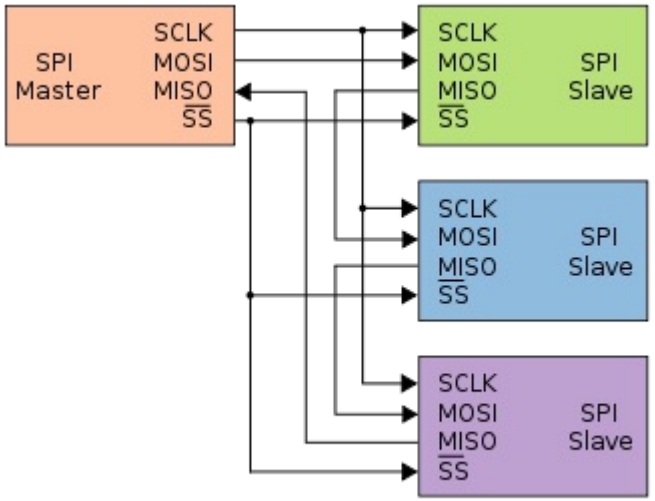


Independent slaves:

Every slave has an independent SS line. Requires the slave to enter a tri-state (high-impedance) mode of its MISO output when not selected.

Multi-slave configurations

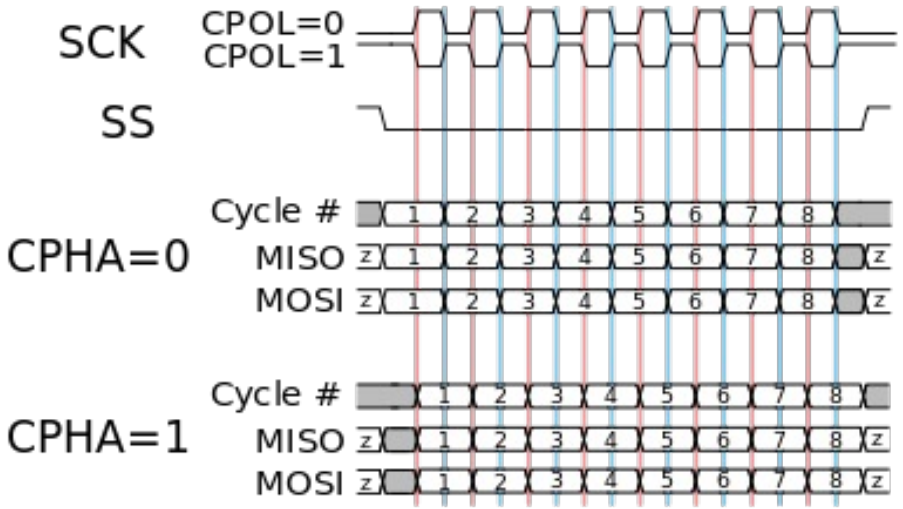




Daisy chain:

A single SS line is shared by all devices. Communication is handled like a shift register.

Clock POLarity & Clock PHAse



CPOL = 0 → Clock idles at '0'; the leading edge is a rising edge.

CPOL = 1 → Clock idles at '1'; the leading edge is a falling edge.

CPHA = 0 → Data changes on the trailing edge of the previous clock (i.e. is sampled on the leading edge, when its value is constant)

CPHA = 1 → Data changes on the leading edge of the current clock (i.e. is sampled on the trailing edge).

SPI mode

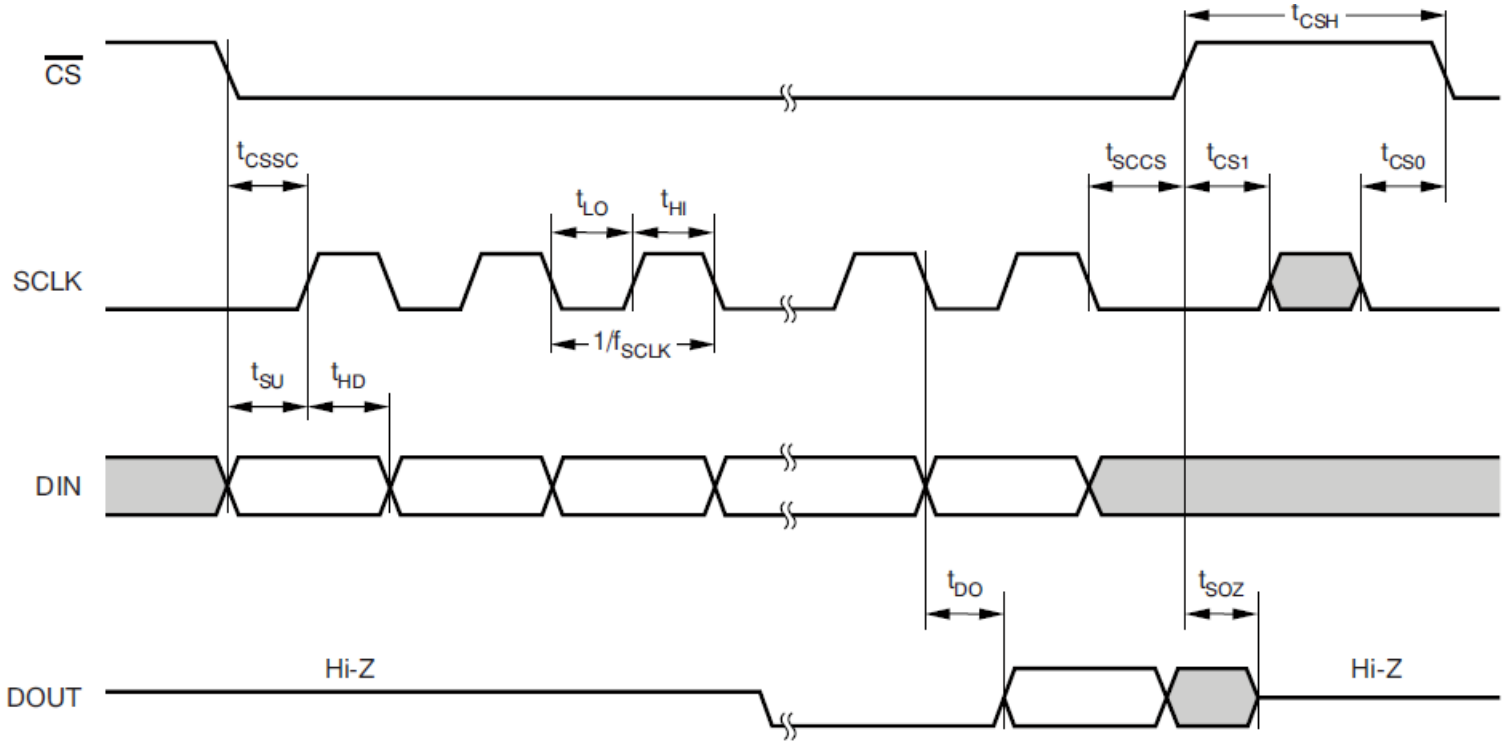


Figure 1. SPI Mode 0, 0

Pros and cons

Pros:

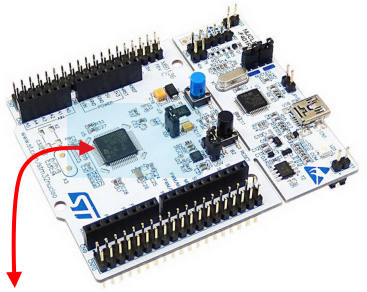
- Full duplex communication
- Push-pull drivers provide good signal integrity (and high speed)
- **Higher speed** compared to I2C
- Complete protocol flexibility (not limited to 8-bit words)
- Very simple hardware interface
  - Often less circuitry, hence less power dissipated
  - Slaves don’t need oscillators
- Very simple software implementation

Cons:

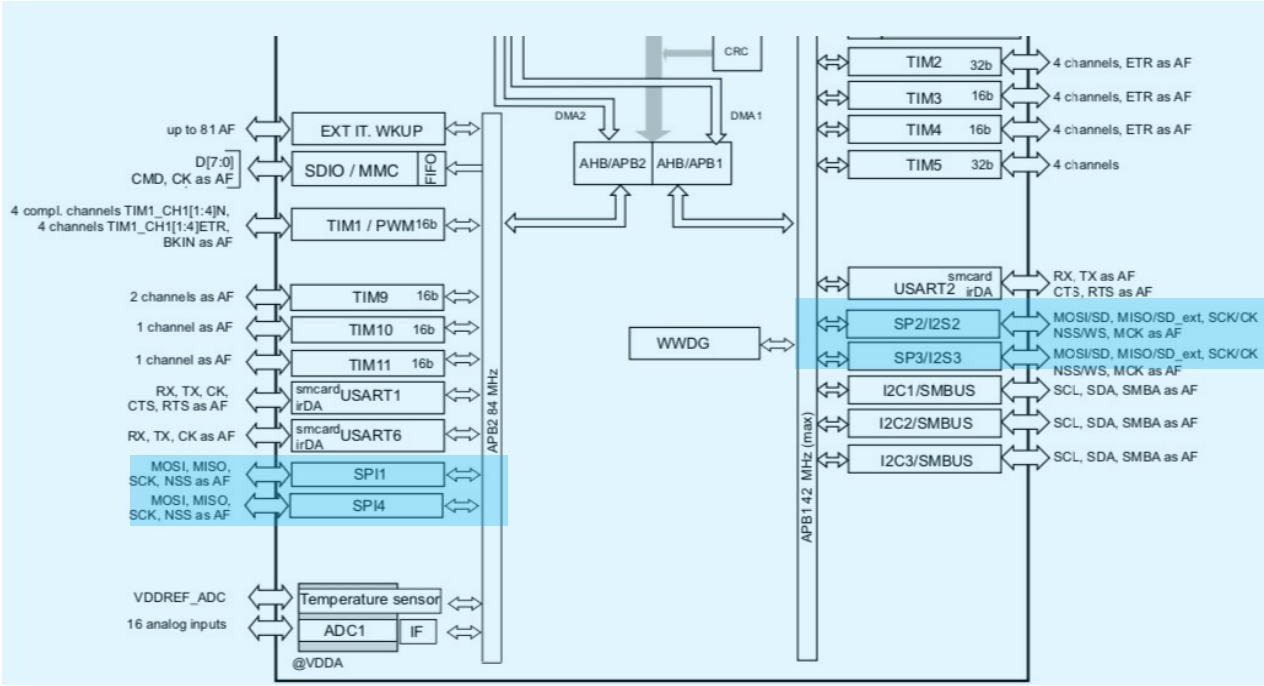
- Requires **more pins** on IC packages than I2C
- No hardware slave acknowledgment (the master could be transmitting to nowhere and not know it)
- Typically supports only one master device (depends on device's hardware implementation)
- Only handles short distances
- Many existing variations, making it difficult to find development tools like host adapters that support those variations

:

SPI – *STM32F401*

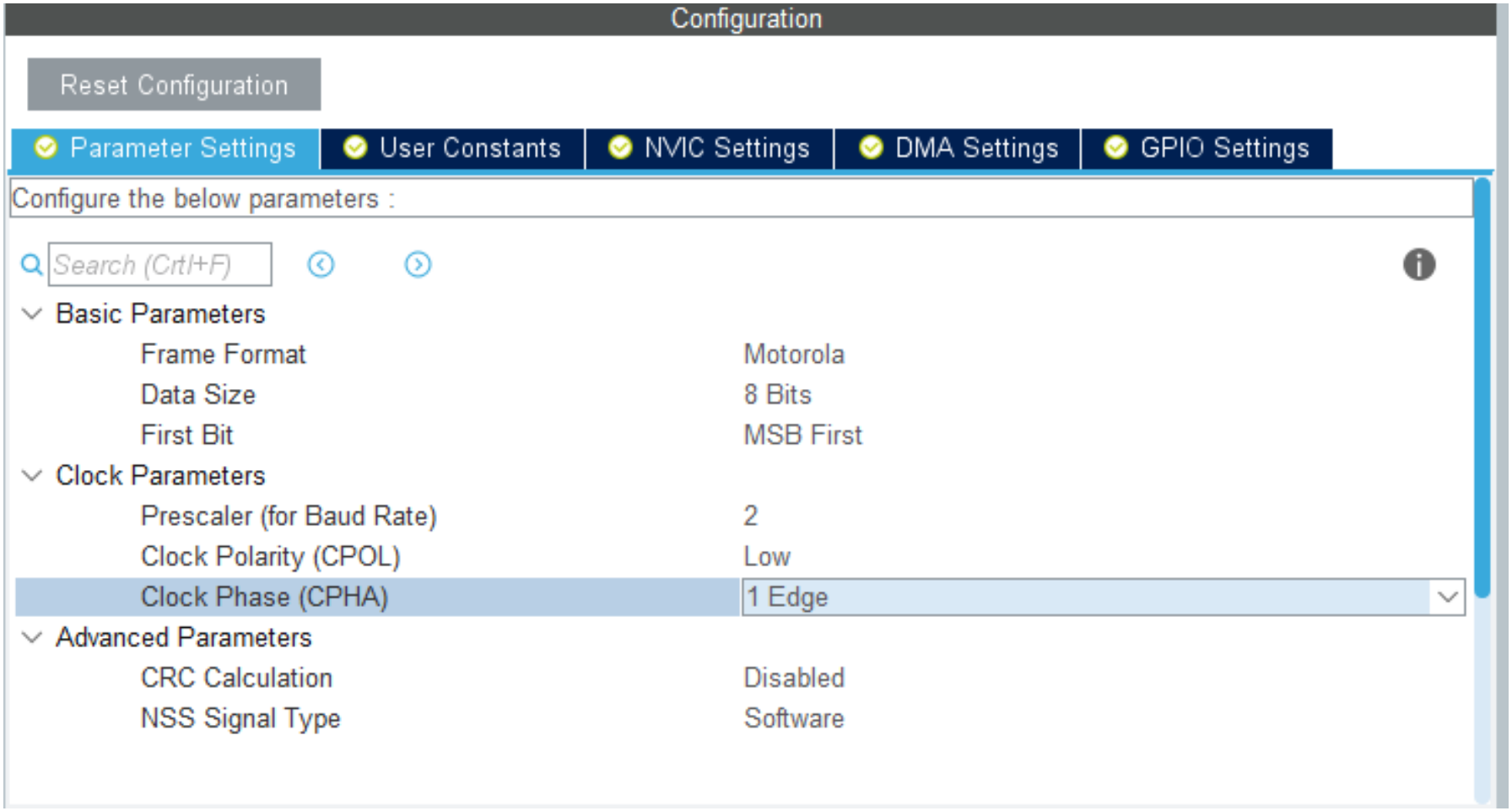


4 wire synchronous full-duplex communication
(up to 42Mbit/s at f_{CPU} = 84 MHz)



- SPI1 and SPI4 can communicate at up to 42 Mbit/s, SPI2 and SPI3 can communicate at up to 21 Mbit/s.
- The 3-bit prescaler gives 8 master mode frequencies and the frame is configurable to 8 bits or 16 bits.
- All SPIs can be served by the DMA controller.

SPI configuration



HAL functions

HAL_StatusTypeDef HAL_SPI_Transmit (SPI_HandleTypeDef *hspi, uint8_t * pData, uint16_t Size, uint32_t Timeout)

Transmit an amount of data in blocking mode.

Parameters:

hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.

pData: pointer to data buffer

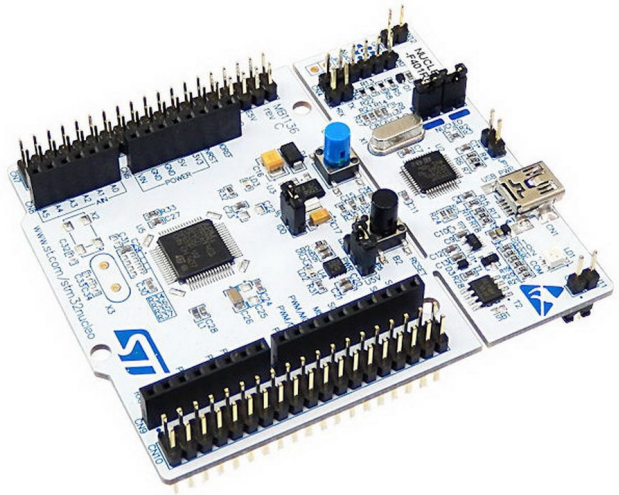
Size: amount of data to be sent

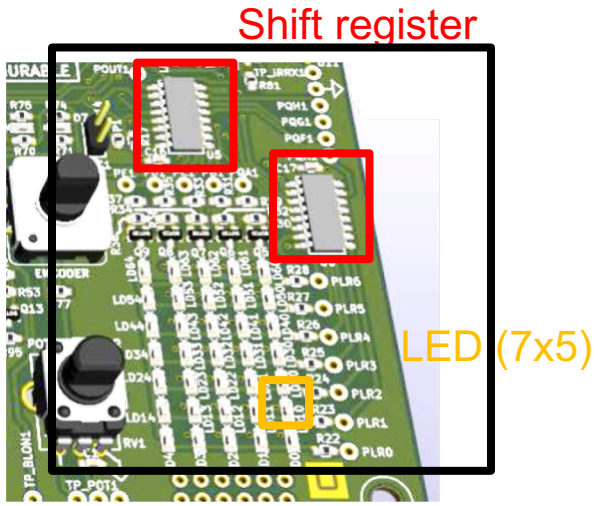
Timeout: timeout duration

HAL_StatusTypeDef HAL_SPI_Transmit_DMA (SPI_HandleTypeDef *hspi, uint8_t * pData, uint16_t Size)

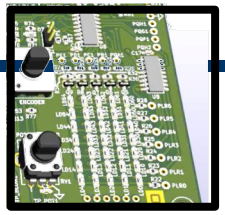
Transmit an amount of data in non-blocking DMA mode.

SPI – *LED Matrix*

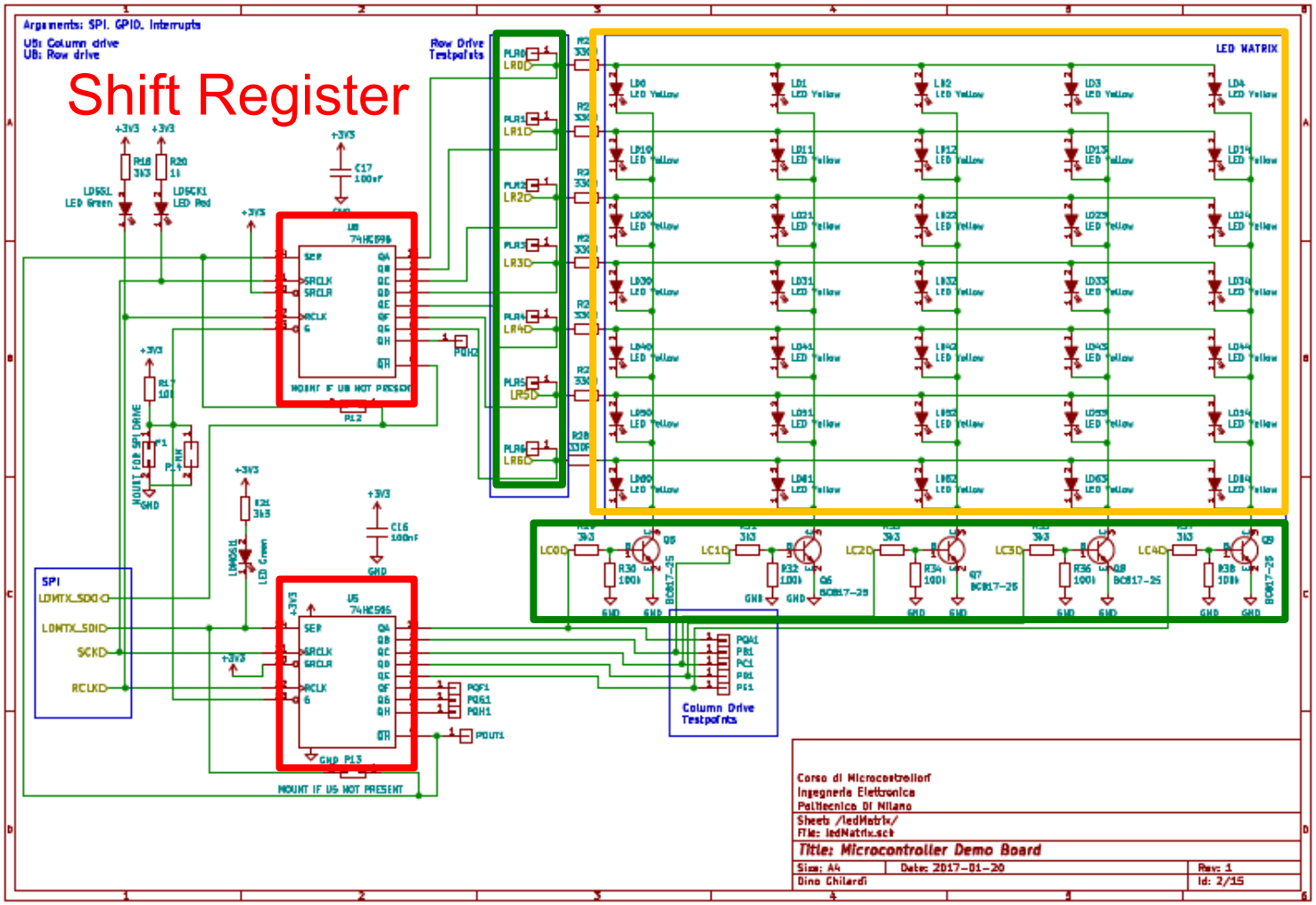




SPI – *LED Matrix Overview*



Row Driver

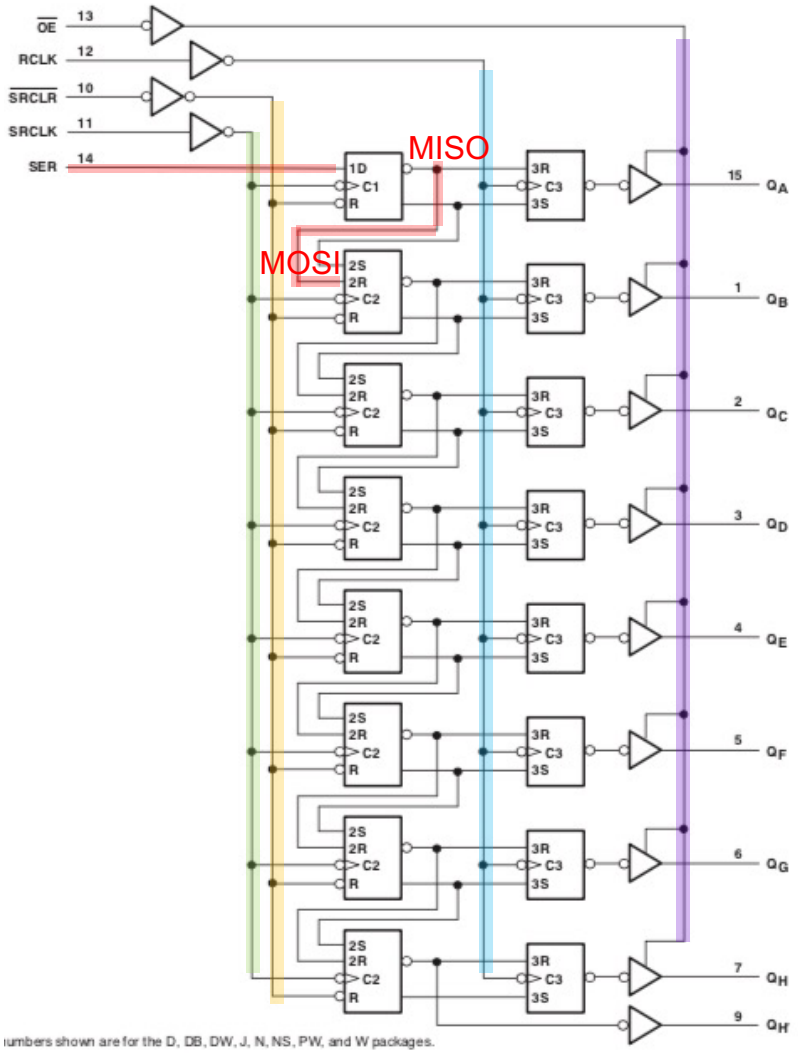


LED (7x5)

Column Driver

LED Matrix – *Shift Register*

- The SNx4HC595 is part of the HC family of logic devices intended for CMOS applications.



Numbers shown are for the D, DB, DW, J, N, NS, PW, and W packages.

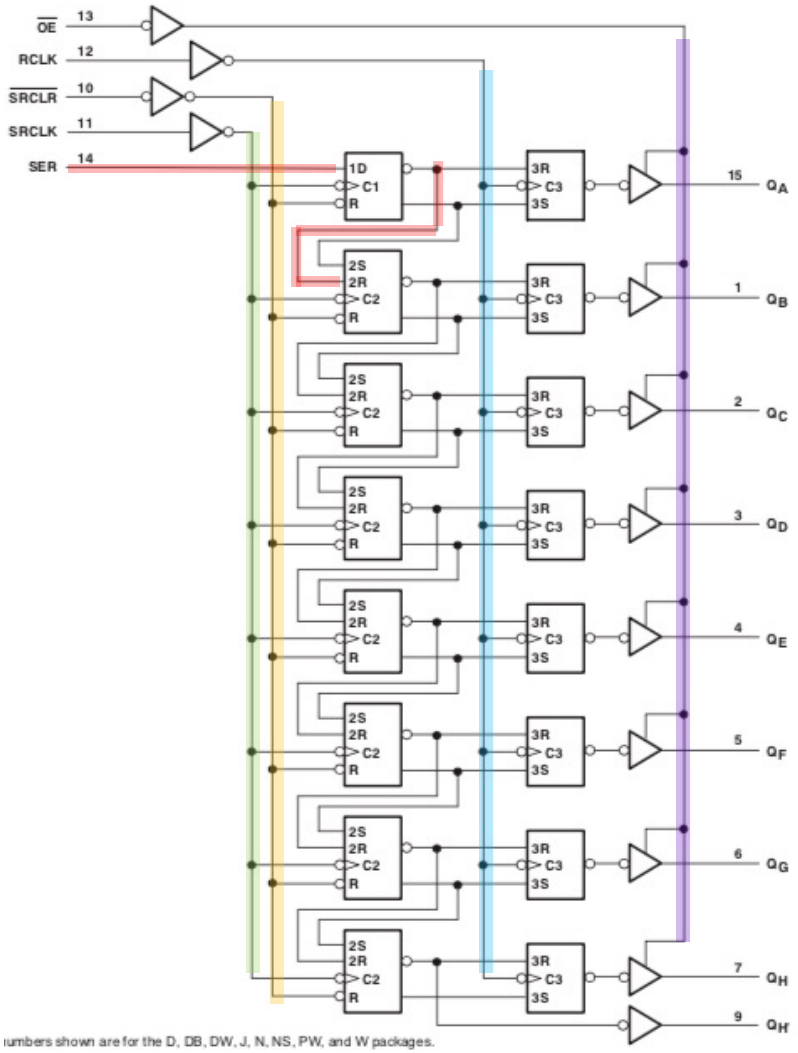
- Is an 8-bit **shift register** that feeds an 8-bit D-type **storage register**.
- Both the shift register clock (SRCLK) and storage register clock (RCLK) are positive-edge triggered.
- The storage register has parallel 3-state outputs. Separate clocks are provided for both the shift and storage register.
- The shift register has a direct overriding clear (SRCLR) input, serial (SER) input, and serial outputs for cascading. When the output-enable (OE) input is high, the outputs are in the high-impedance state.

Table 1. Function Table

INPUTS					FUNCTION
SER	SRCLK	SRCLR	RCLK	OE	
X	X	X	X	H	Outputs Q_A – Q_H are disabled.
X	X	X	X	L	Outputs Q_A – Q_H are enabled.
X	X	L	X	X	Shift register is cleared.
L	↑	H	X	X	First stage of the shift register goes low. Other stages store the data of previous stage, respectively.
H	↑	H	X	X	First stage of the shift register goes high. Other stages store the data of previous stage, respectively.
X	X	X	↑	X	Shift-register data is stored in the storage register.

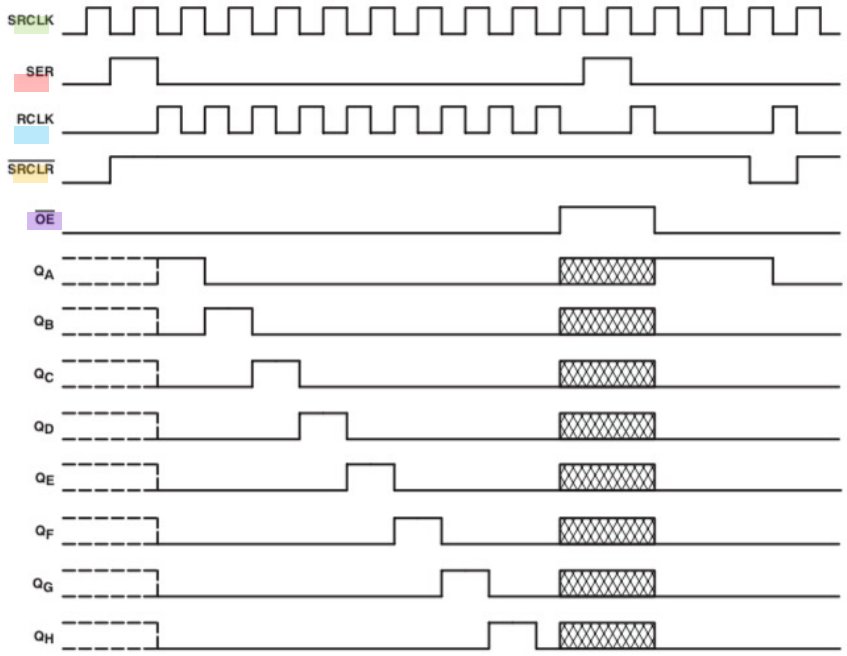
LED Matrix – *Shift Register*

- The SNx4HC595 is part of the HC family of logic devices intended for CMOS applications.



numbers shown are for the D, DB, DW, J, N, NS, PW, and W packages.

Timing Diagram



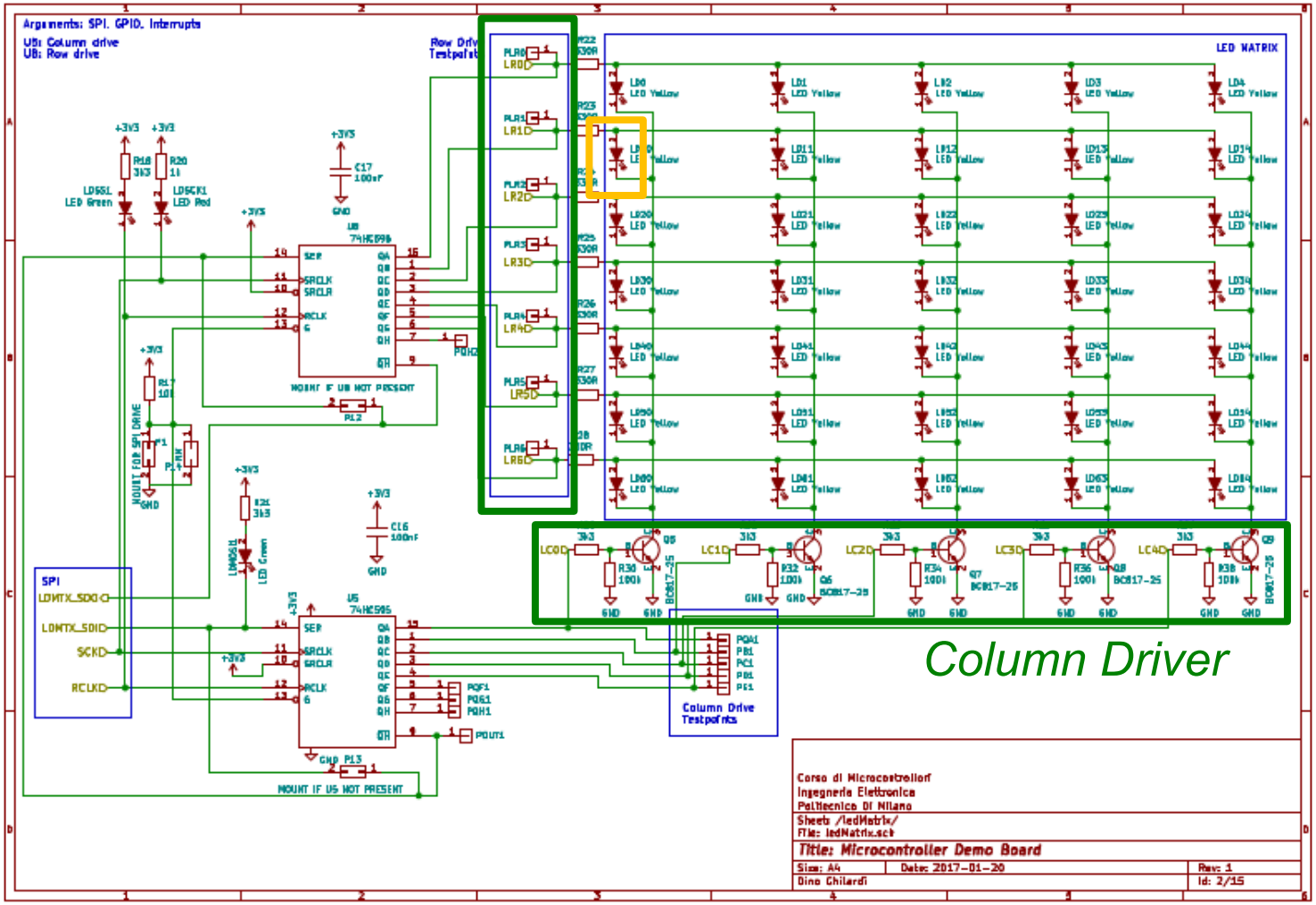
Function Table

INPUTS					FUNCTION
SER	SRCLK	SRCLR	RCLK	OE	
X	X	X	X	H	Outputs $Q_A - Q_H$ are disabled.
X	X	X	X	L	Outputs $Q_A - Q_H$ are enabled.
X	X	L	X	X	Shift register is cleared.
L	↑	H	X	X	First stage of the shift register goes low. Other stages store the data of previous stage, respectively.
H	↑	H	X	X	First stage of the shift register goes high. Other stages store the data of previous stage, respectively.
X	X	X	↑	X	Shift-register data is stored in the storage register.

shift register clock (SRCLK), storage register clock (RCLK), shift register overriding clear (SRCLR), serial (SER), output-enable (OE)

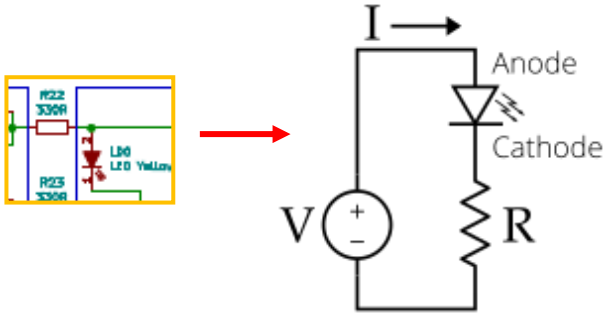
LED Matrix – *Overview*

Row Driver



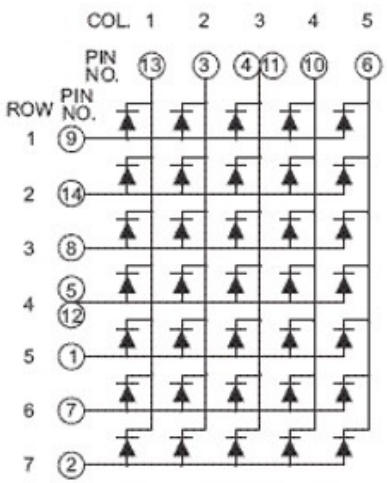
Column Driver

LED - *Light Emitting Diode*

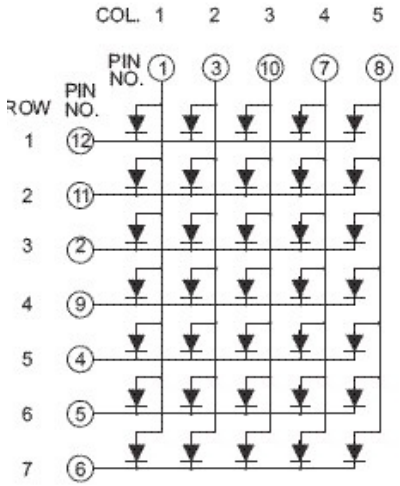


Limit the current to prevent damaging the LED

LED Matrix Configurations

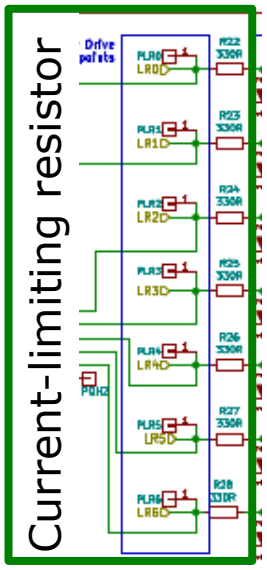


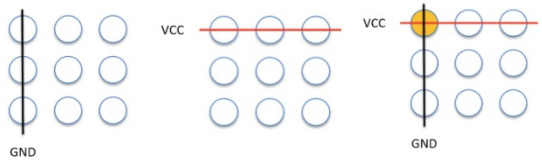
Columns sharing a common-cathode



Columns sharing a common-anode

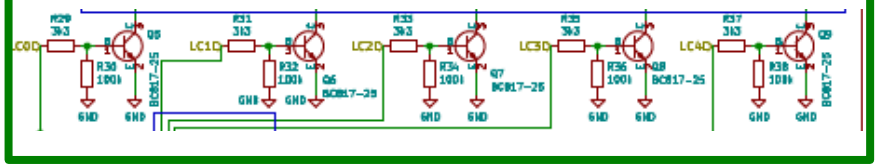
Row Driver





Column Driver

NPN transistors to drive the cathodes



- Transistors as voltage controlled switches
- Transistor is turned on, the common cathodes of the selected column are connected to ground and the chosen LEDs light up

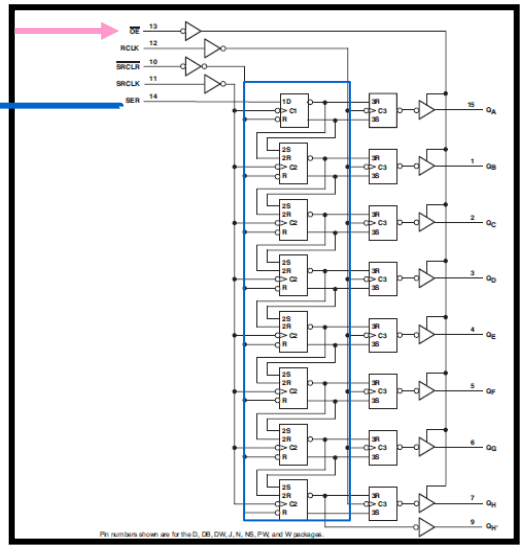
Data transfer to the led matrix(1)

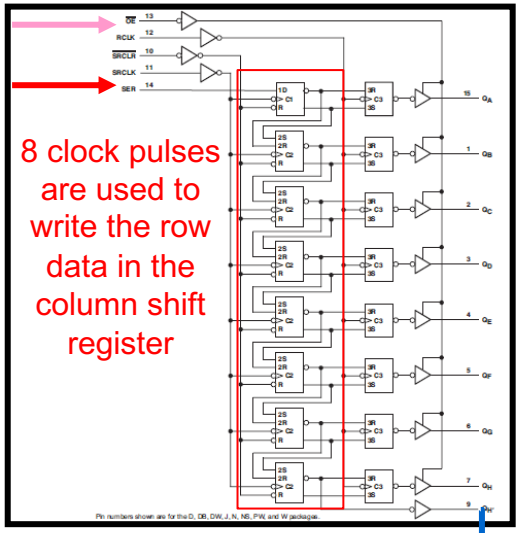
16 clock pulses to transfer the data to the led matrix (daisy chain configuration – Most significant bit)

8 clock pulses:

- Transfer the row data in the row shift register
- Write the column data in the column shift register

OE – OUTPUT ENABLE



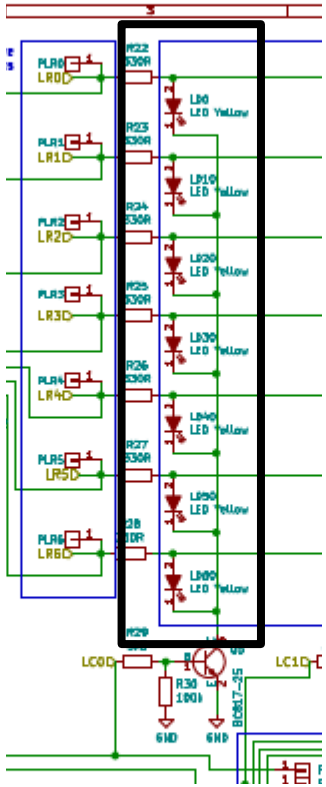


ROW

SHIFT REGISTER

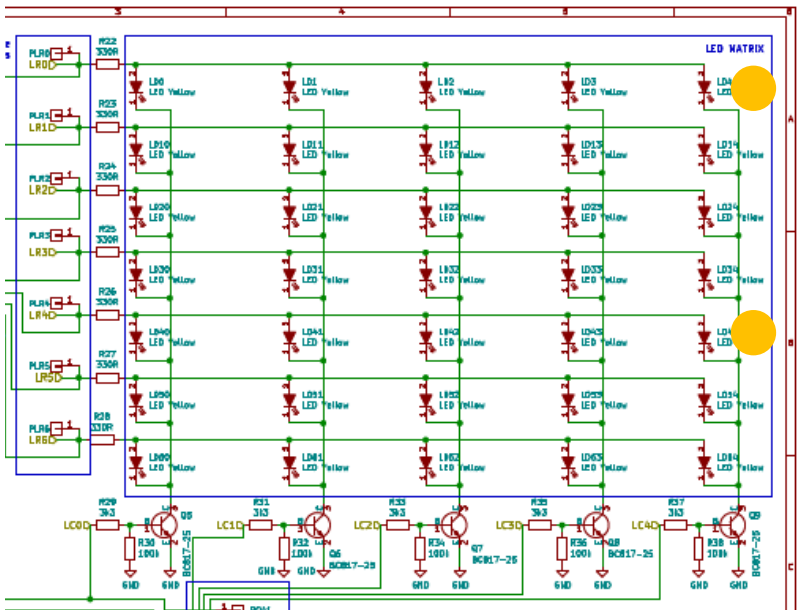
COLUMN

SHIFT REGISTER



For each column the correct row data are written

Data transfer to the led matrix(2)



1
0
0
0
1
0
0

{ 68 , 16 }

0 0 0 0 1

row	column
.	16
.	8
.	4
.	2
.	1

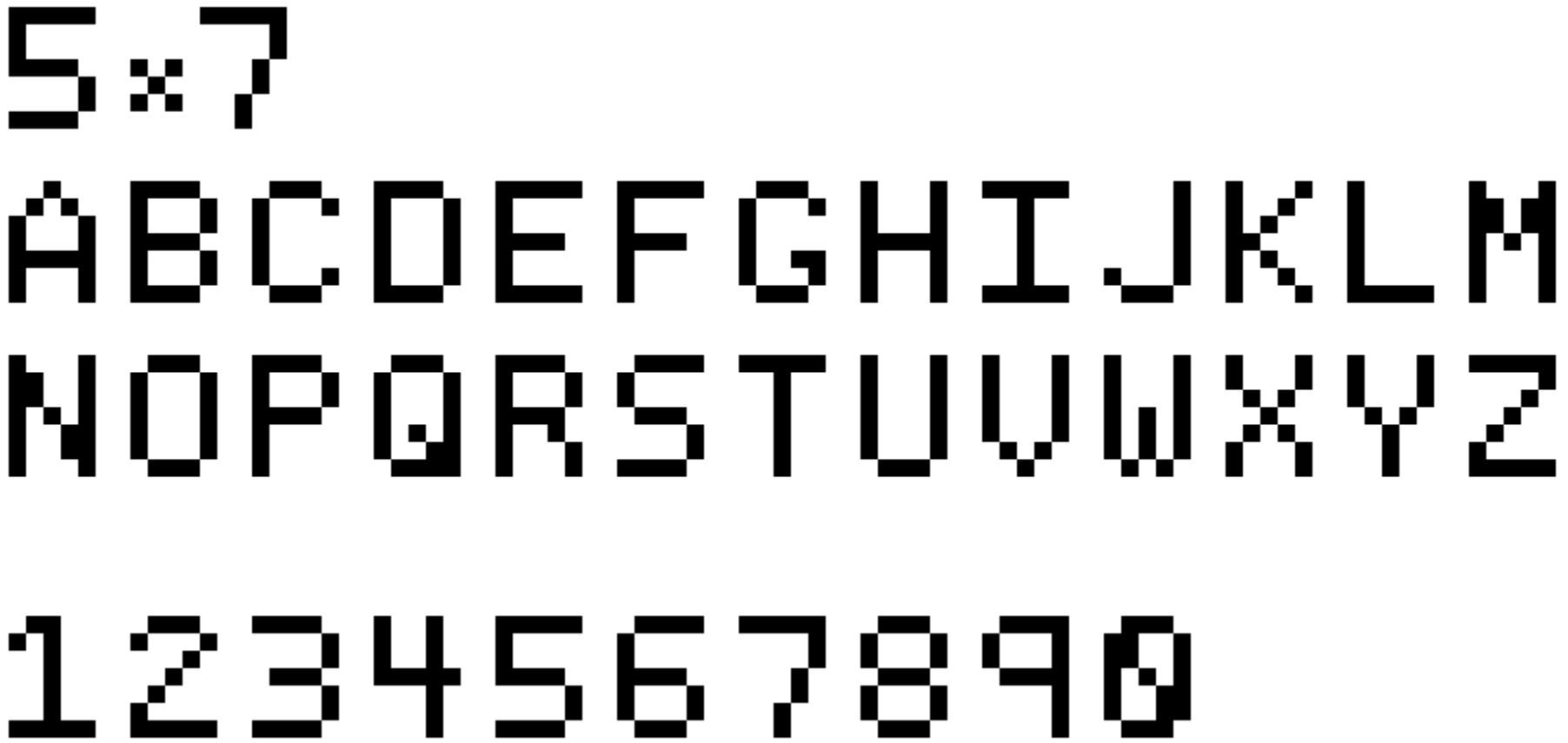
MULTIPLEXING:

You can only write ONE COLUMN AT A TIME!

To show a complete matrix, quickly **scan through all columns, writing one column at a time**, with a fast refresh rate (> 50 Hz)

i.e. one column every < 4 ms

5 × 7 font example



LED matrix computation

How to use the

LED matrix computation.xlsx

file

5 × 7 LED matrix design

16	8	4	2	1	
		X			64
	X		X		32
X				X	16
X				X	8
X	X	X	X	X	4
X				X	2
X				X	1

Instruction:

Write an 'x' (small letter) where you want the LED to light up.

The corresponding pairs of row and column data appears in the cells in RED (format: decimal)

Computed value

COL	16	8	4	2	1
ROW	31	36	68	36	31

Project 1a – Write a single letter

Objective

Transmit a letter to the LED matrix using SPI and a timer interrupt

Project hints

- Identify the SPI data and clock pins, as well as the RCLK of the register
- Setup a timer to provide an interrupt every 4 ms. In the callback, write one column of the LED matrix and pulse the RCLK. After writing the 5th column, roll over to the first one.
- Compile and debug the code.

Project 1b – Alternate between letters

Objective

Alternate between two letters (or one letter and one symbol)

Use the SPI DMA write and avoid the Hal_Delay function.

Project hints

- Start from the previous project. Enable the SPI_TX DMA transfer. Beware of how the SPI DMA transfer works.

- Within the main loop, find a way to change what is shown on the matrix. Keep the interrupt routine code as simple as possible.

- Compile and debug the code.