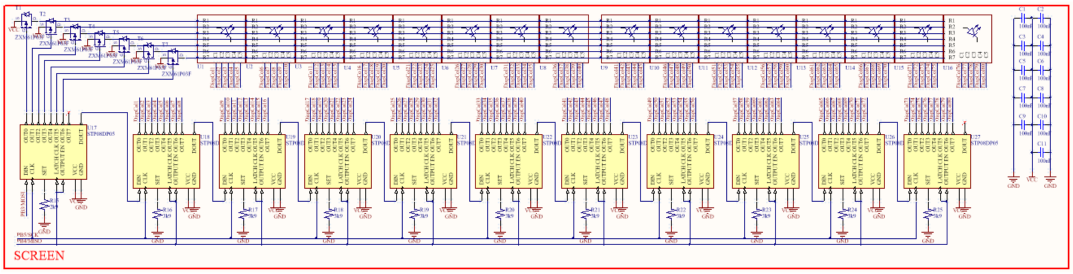
Sensors & Microsystem Electronics: microcontrollers

PART 5: SCREEN

Screen

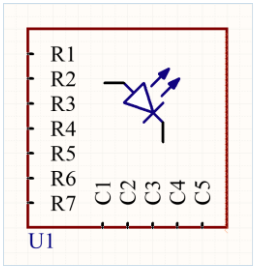


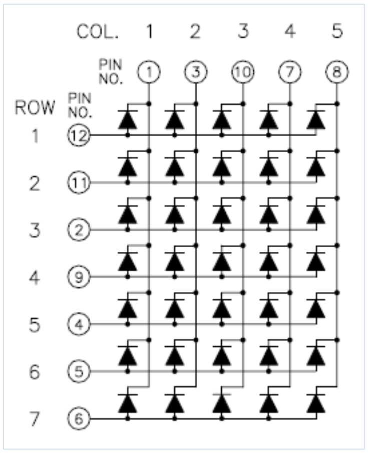
This is the schematic of the screen, A more detailed version can be found in the PDF with the board schematic,

The display is connected to a series of shift registers both in columns and in rows,

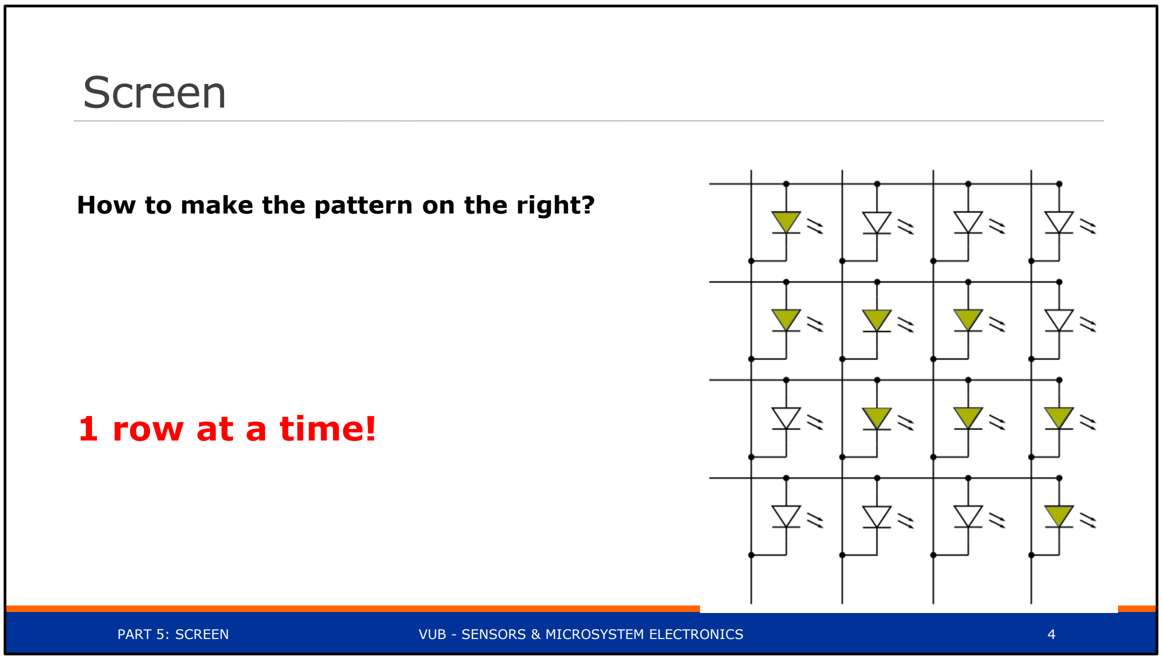
Screen = LED-matrix

- 7x5 matrix of LEDs
- “Column cathode”
- LED on when corresponding row is HIGH





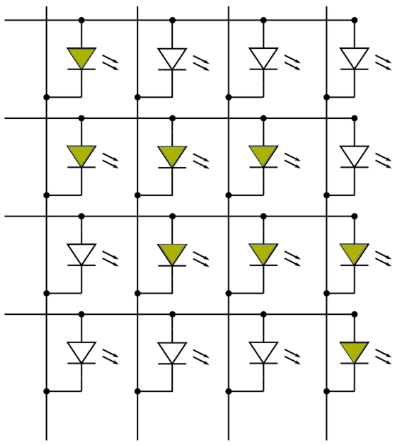
Each display is a 7 by 5 matrix of leds with a column cathode, this means that the cathodes of the leds are connected in the columns. To turn on a LED a current from anode to cathode: this means the respective row is high, and the column is low.



Screen

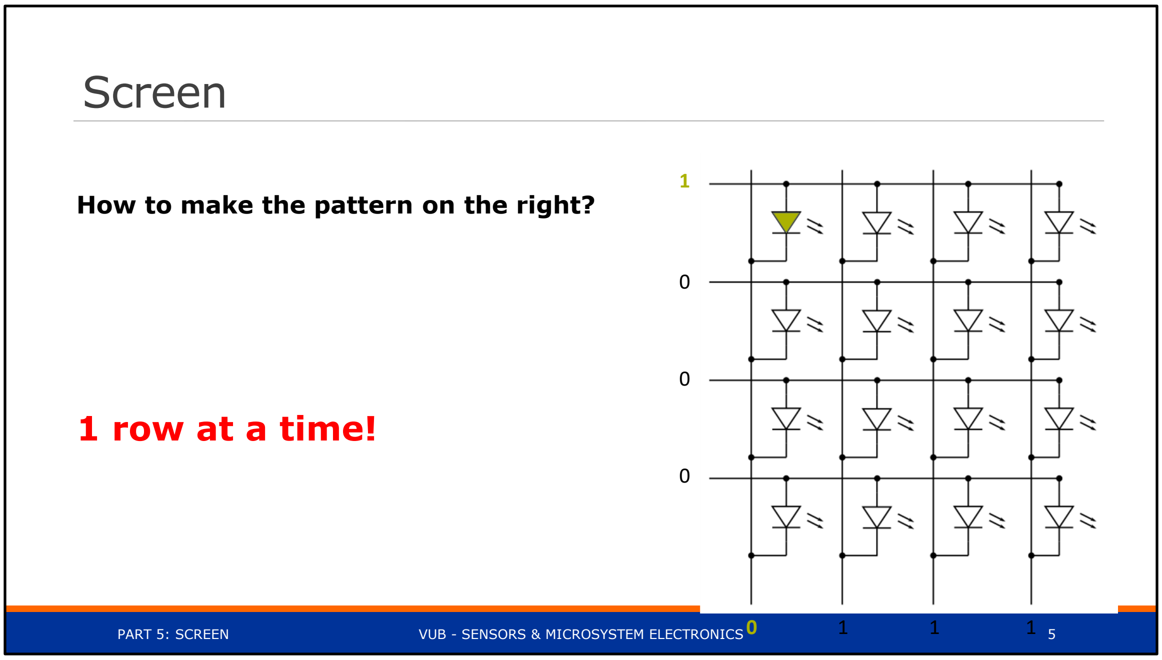
How to make the pattern on the right?

1 row at a time!



How can we create the pattern on the right side?

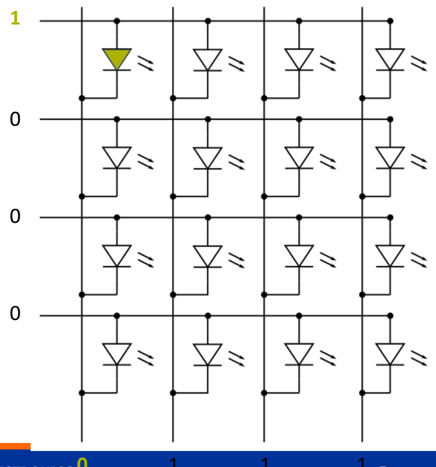
Because the rows and columns are common to anode and cathode respectively, we cannot make this pattern all at once. The only way to do this is to do it line by line.



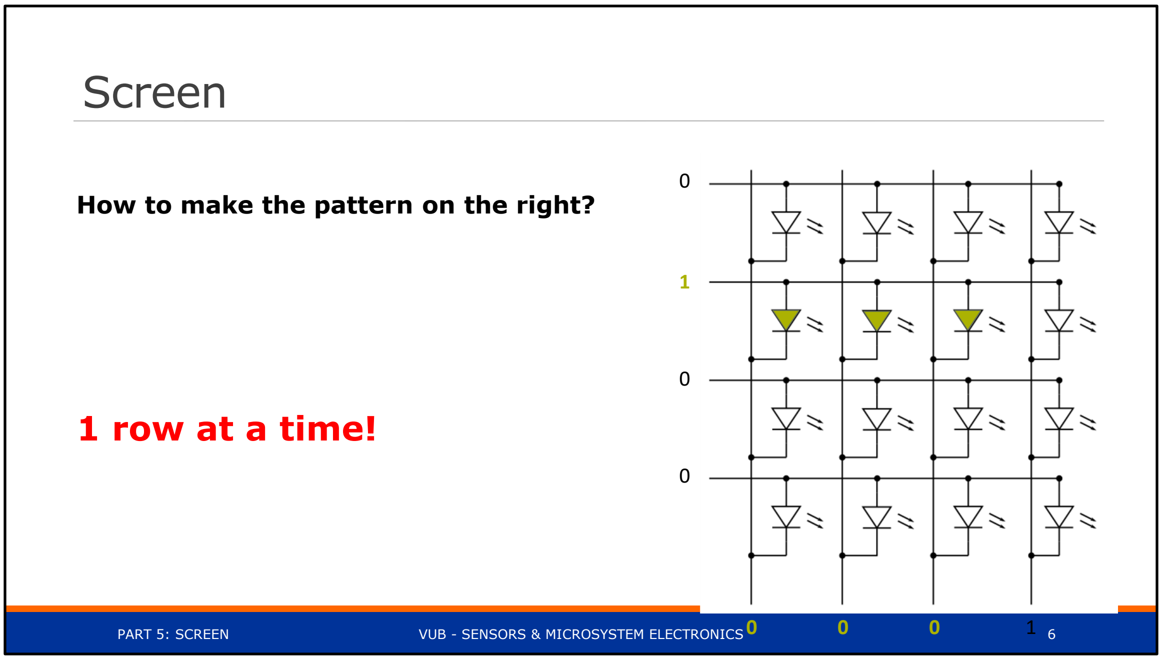
Screen

How to make the pattern on the right?

1 row at a time!



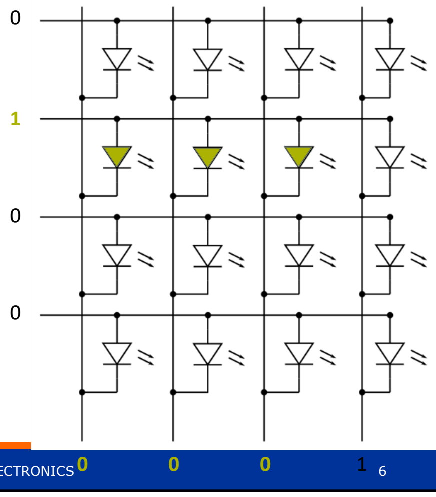
Top Row: set row_1 high and the other rows low AND column_1 low and the other columns high. This allows a current to flow from row_1 to column_1, and no other leds are on due to reverse biasing the LEDs



Screen

How to make the pattern on the right?

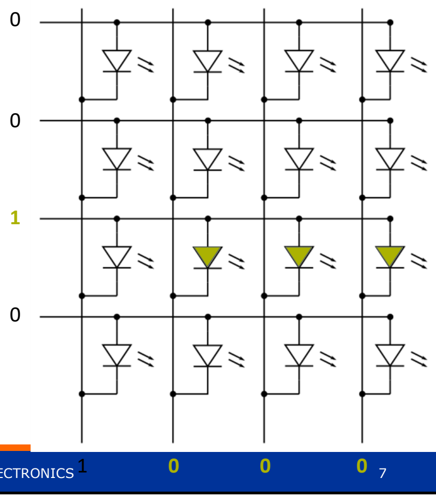
1 row at a time!



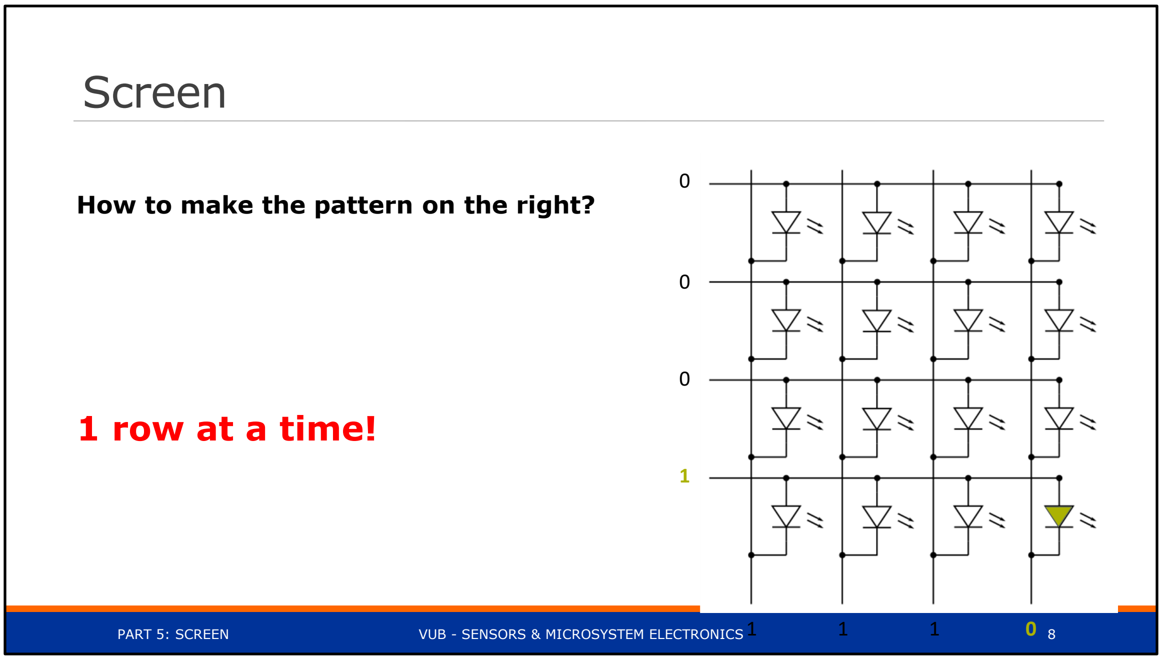
Row 2

Screen

How to make the pattern on the right?



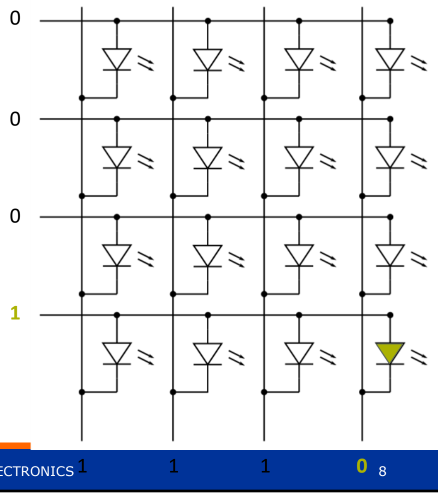
1 row at a time!



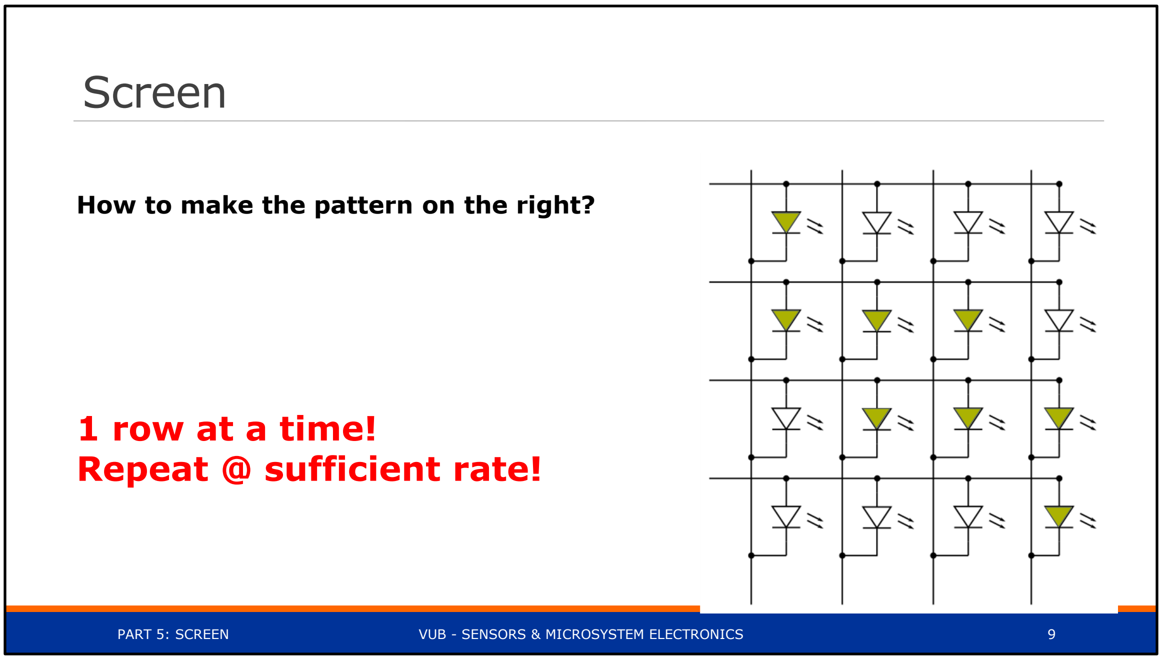
Screen

How to make the pattern on the right?

1 row at a time!

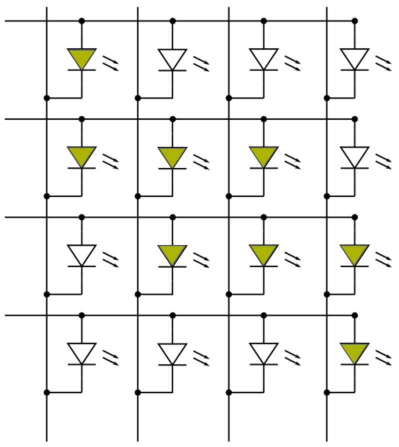


Row 3



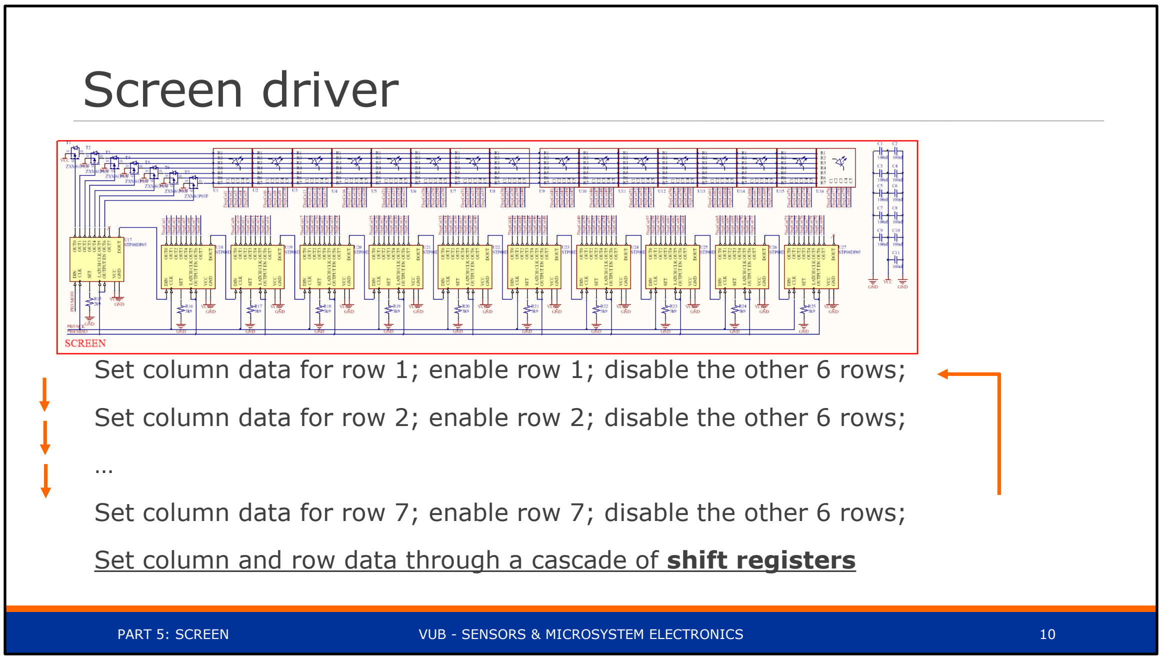
Screen

How to make the pattern on the right?

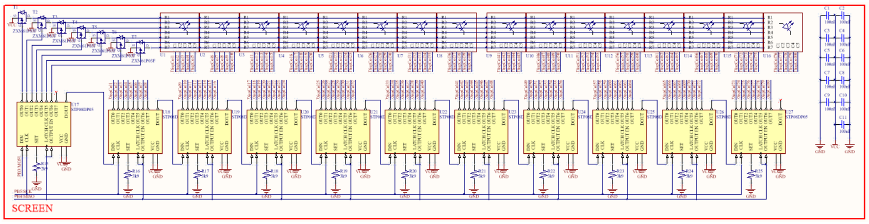


1 row at a time!
Repeat @ sufficient rate!

Row 4



Screen driver



Set column data for row 1; enable row 1; disable the other 6 rows;

Set column data for row 2; enable row 2; disable the other 6 rows;

…

Set column data for row 7; enable row 7; disable the other 6 rows;

Set column and row data through a cascade of **shift registers**

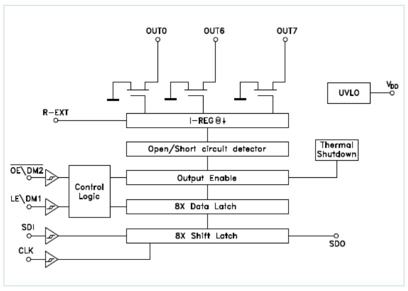
PART 5: SCREEN VUB - SENSORS & MICROSYSTEM ELECTRONICS 10

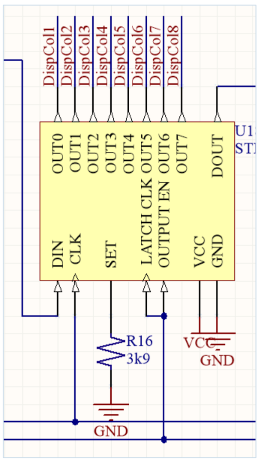
We take this principle now to the display on the demo board, First we set the column data for the first row, enabling the right column. Then we enable only row 1. Then we do the same thing for row 2 using the column data for row 2 and we disable row 1 but enable row 2.

Since the amount of pins of the microcontroller are limited. The circuit designed uses shift registers to be able to address more pins. If not one would need #rows + #columns of pins on the microcontroller to drive the display. Using this circuit the same can be done using only 3 pins!

Shift registers (STP08DP05)

- First In First Out (FIFO) buffer
- BIT 7 is copied to SDO to cascade multiple FIFOs
- Output latch & output enable
- R-EXT to configure the maximal output current



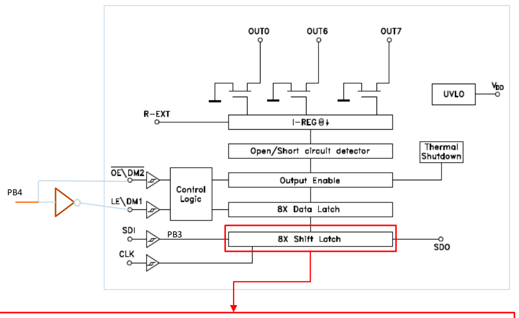


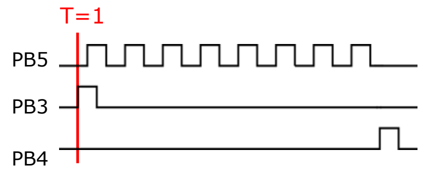
A shift register is a FIFO buffer for single bits, and is a serial to parallel convertor.

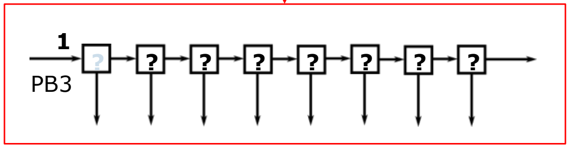
Using a clock signal, a single bit is inserted in the shift register, the other bits are shifted one position. Bit 7 is always mirrored on the cascade output so that multiple shift registers can be chained together by connecting the SDO to the SDI of the next one, The Latch and the clock are common between all shift registers,

Shift registers (STP08DP05)

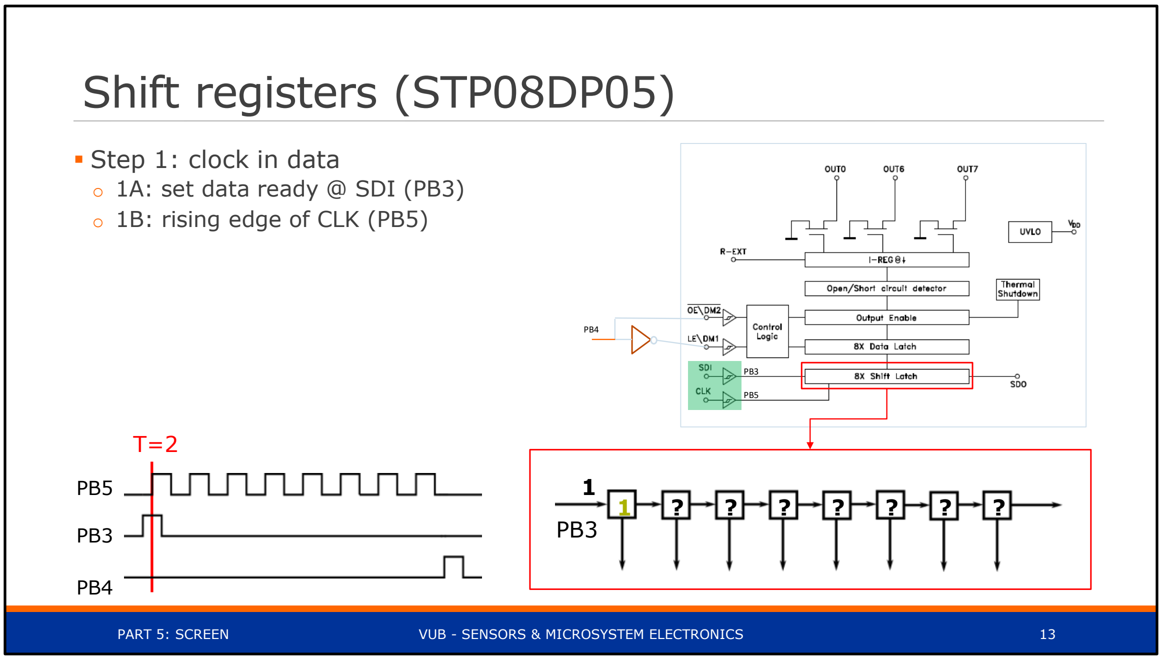
- Step 1: clock in data
  - 1A: set data ready @ SDI (PB3)





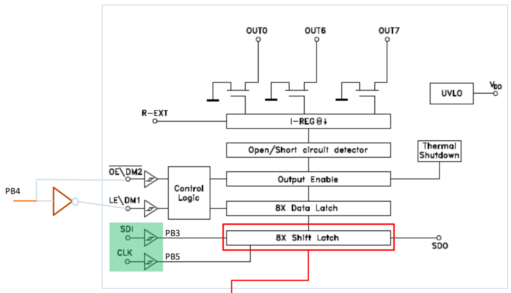


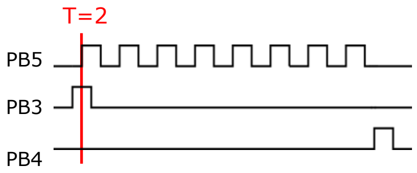
Data is clocked in by setting the data bit on PB3, and creating a rising edge on PB5, This shift in the value of PB3 in the shift register

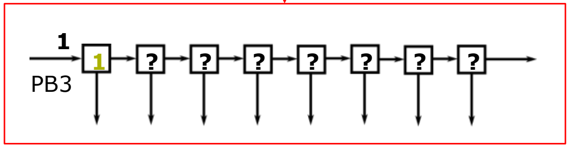


Shift registers (STP08DP05)

- Step 1: clock in data
  - 1A: set data ready @ SDI (PB3)
  - 1B: rising edge of CLK (PB5)



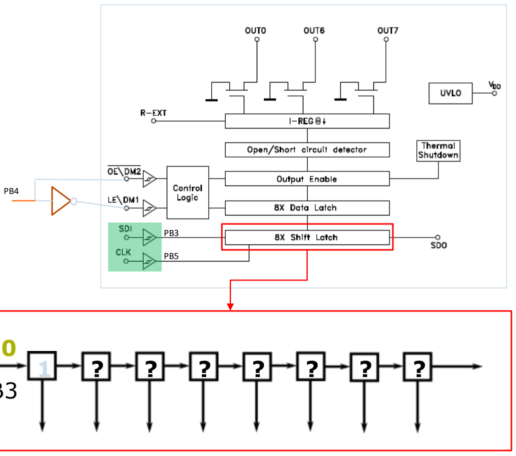


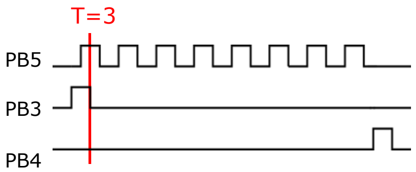


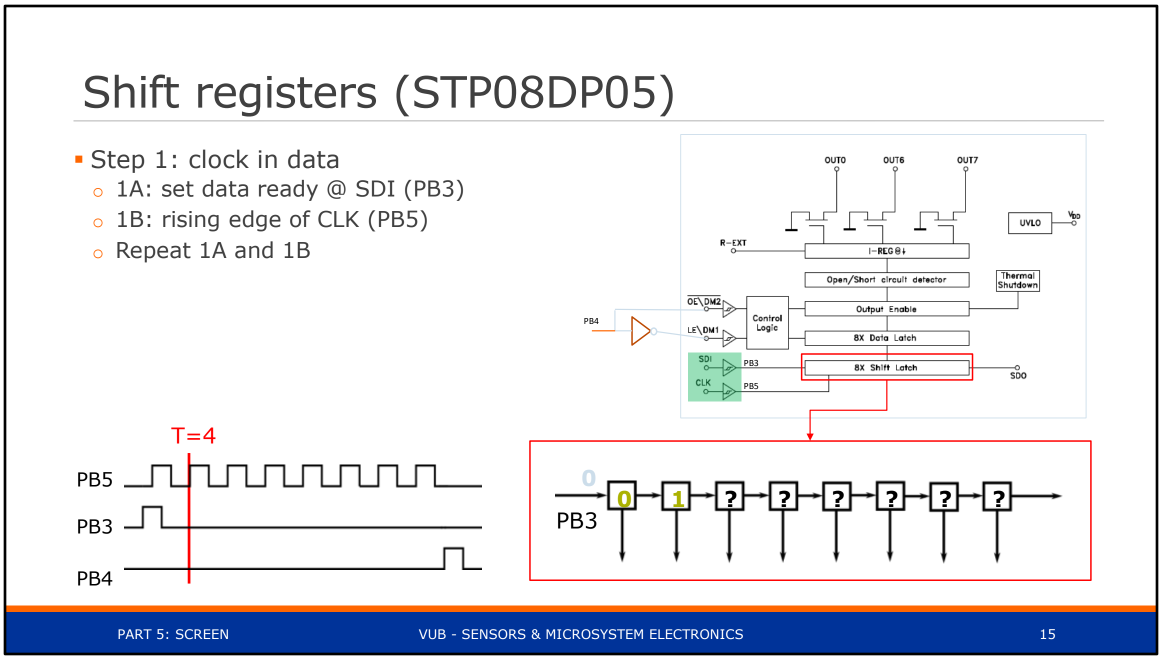
Data is clocked in by setting the data on PB3, and creating a rising edge on PB5, This shift in the value of PB3 in the shift register

Shift registers (STP08DP05)

- Step 1: clock in data
  - 1A: set data ready @ SDI (PB3)
  - 1B: rising edge of CLK (PB5)
  - Repeat 1A and 1B

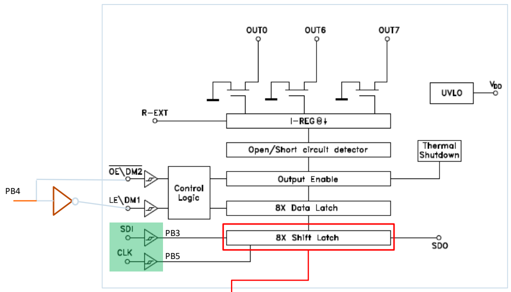


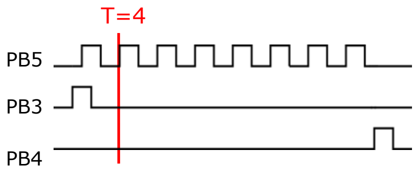


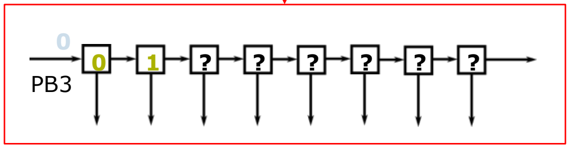


Shift registers (STP08DP05)

- Step 1: clock in data
  - 1A: set data ready @ SDI (PB3)
  - 1B: rising edge of CLK (PB5)
  - Repeat 1A and 1B



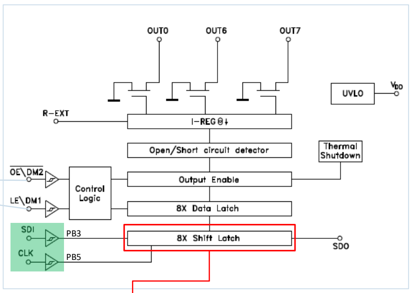


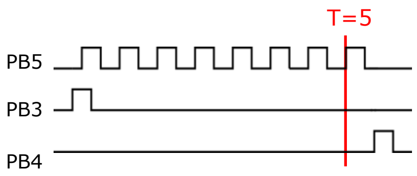


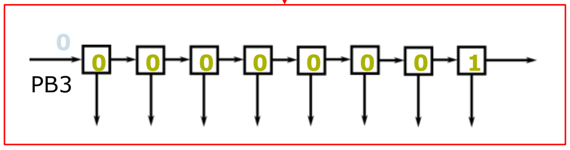
On each rising edge a new bit is shifted in the shift register and the bits propagate trough it.

Shift registers (STP08DP05)

- Step 1: clock in data
  - 1A: set data ready @ SDI (PB3)
  - 1B: rising edge of CLK (PB5)
  - Repeat 1A and 1B



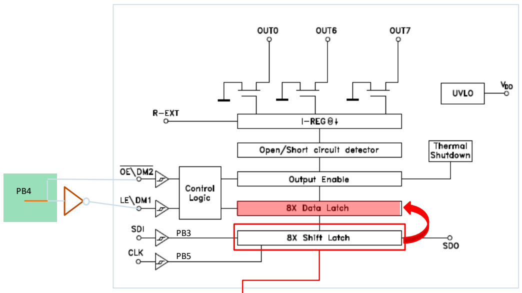


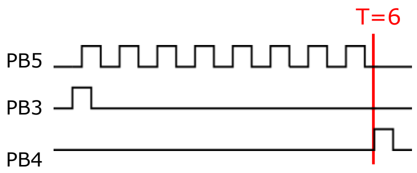


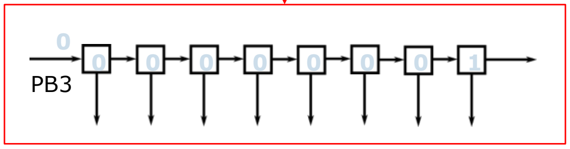
After 8 clock cycles the register is full, and any new bit that is shifted in propagates out trough the SDO (possibly into the next register)

Shift registers (STP08DP05)

- Step 1: clock in data
- Step 2: copy data to output Register
  - set LE/DM1 (PB4)



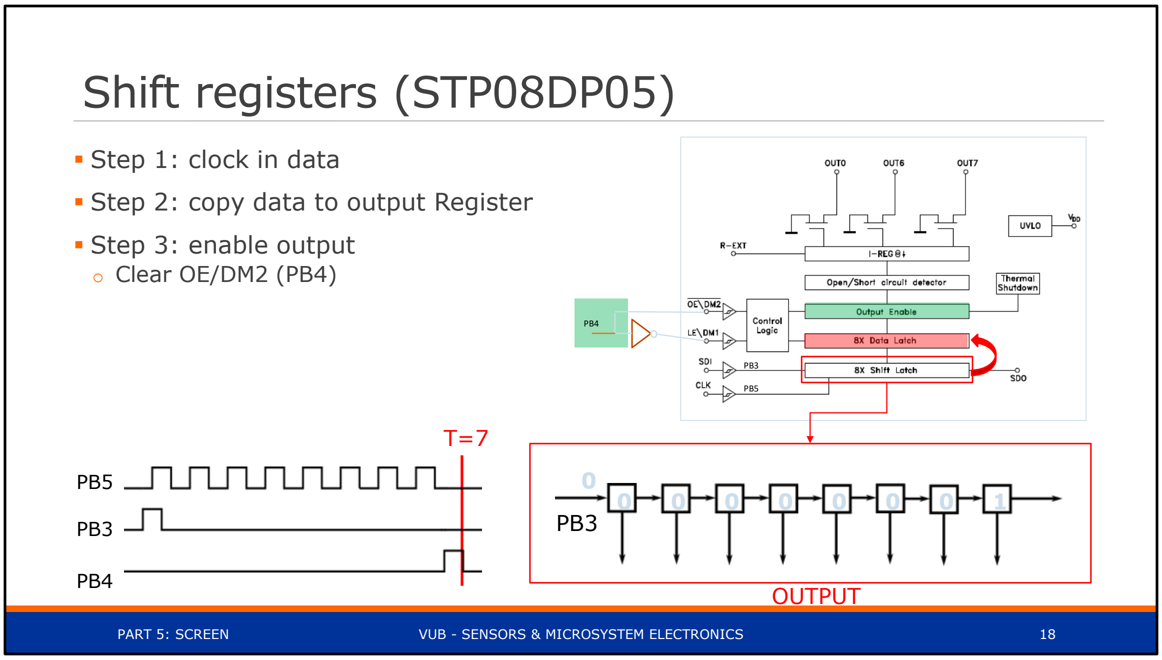




LATCHED

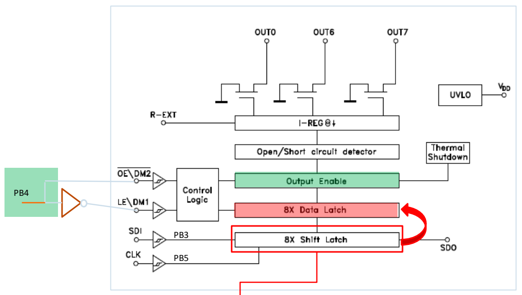
After shifting in all the bits, the data is latched into the 8 bit data latch register to become stored. The process is twofold: data is shifted trough the shift latch and only stored by latching the data in the data latch. The data in the data latch is not modified during shifting.

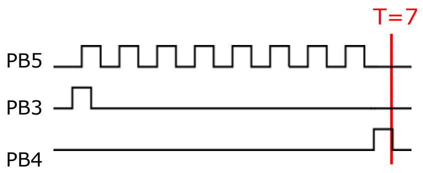
In this circuit the data latch and the output enable are tied together: a rising edge on PB4 latches the data, and a low value on PB4 enables the outputs based on the data in the data latch.

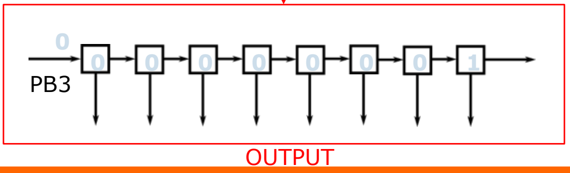


Shift registers (STP08DP05)

- Step 1: clock in data
- Step 2: copy data to output Register
- Step 3: enable output
  - Clear OE/DM2 (PB4)

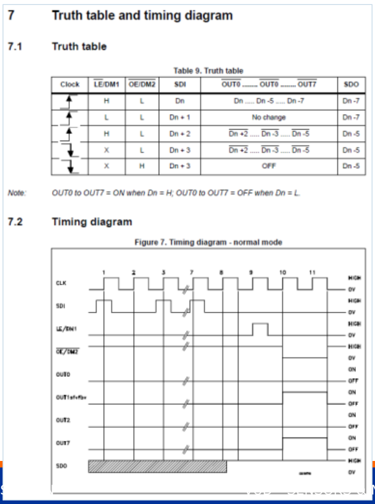






In this circuit the data latch and the output enable are tied together: a rising edge on PB4 latches the data, and a low value on PB4 enables the outputs based on the data in the data latch.

Shift registers (STP08DP05)



7 **Truth table and timing diagram**

7.1 **Truth table**

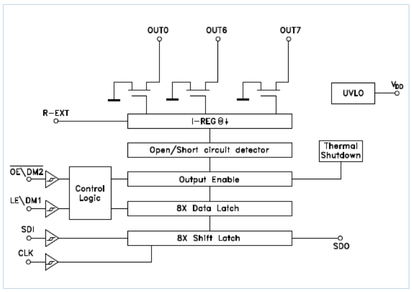
Table 9. Truth table

Clock	$\overline{LE}$/DM1	$\overline{OE}$/DM2	SDI	$\overline{OUT0}$ .... $\overline{OUT0}$ .... $\overline{OUT7}$	SDO
↑	H	L	Dn	Dn ..... Dn -5 ..... Dn -7	Dn -7
↑	L	L	Dn + 1	No change	Dn -7
↑	H	L	Dn + 2	$\overline{Dn+2}$ ..... $\overline{Dn-3}$ ..... $\overline{Dn-5}$	Dn -5
↓	X	L	Dn + 3	$\overline{Dn+2}$ ..... $\overline{Dn-3}$ ..... $\overline{Dn-5}$	Dn -5
↓	X	H	Dn + 3	OFF	Dn -5

Note: OUT0 to OUT7 = ON when Dn = H; OUT0 to OUT7 = OFF when Dn = L.

7.2 **Timing diagram**

Figure 7. Timing diagram - normal mode



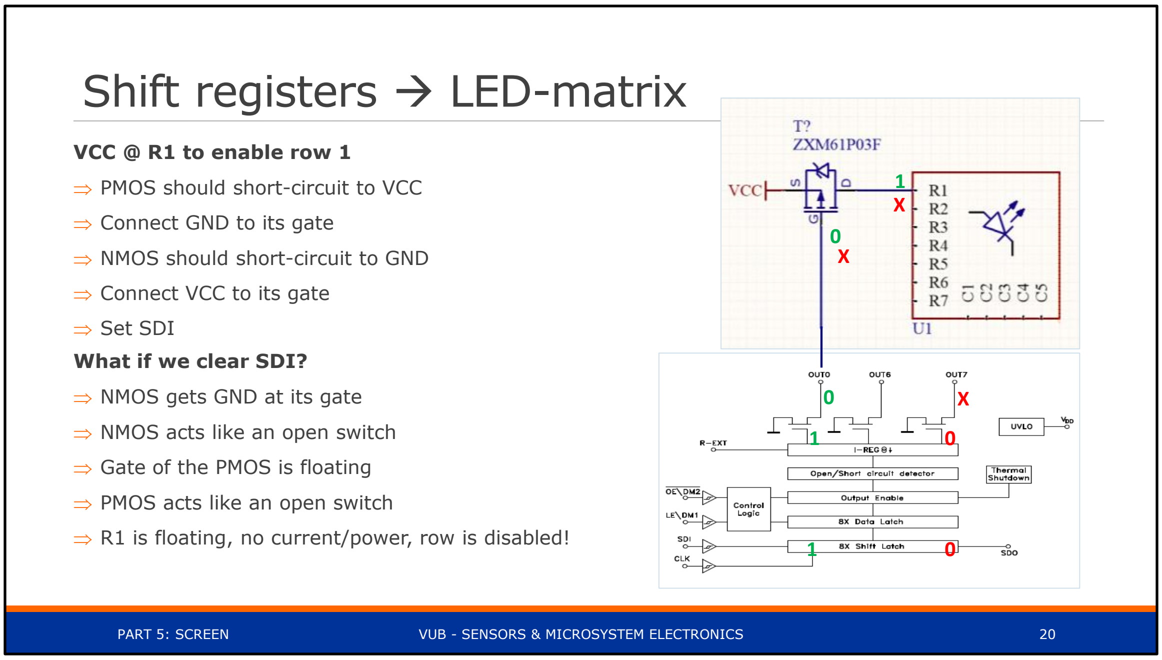
READ DATASHEET!

The truth table and timing diagram shows more detailed usage of the shift register

On the right image one can see that the output stage of the chip are NMOS (open drain) transistors. By setting a bit to ‘1’ in the data latch the gate of the transitor is set high, and thus the transistor will conduct from drain to ground.

To select a certain column we need to connect it to ground (see slide at the beginning – common cathode), We can connect a column to ground by enabling the NMOS transistors by setting the gate to ‘1’

In short: to enable a certain column, we set that bit to ‘1’ in the shift register.



Shift registers → LED-matrix

VCC @ R1 to enable row 1

⇒ PMOS should short-circuit to VCC

⇒ Connect GND to its gate

⇒ NMOS should short-circuit to GND

⇒ Connect VCC to its gate

⇒ Set SDI

What if we clear SDI?

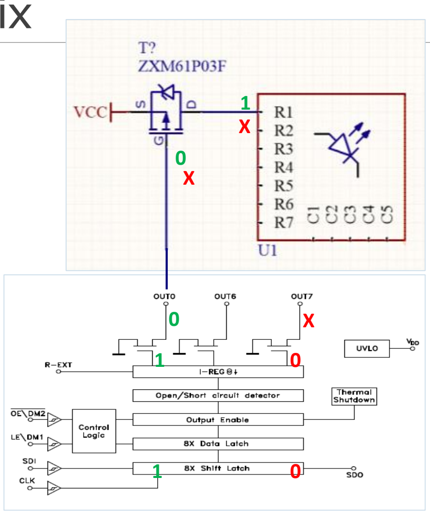
⇒ NMOS gets GND at its gate

⇒ NMOS acts like an open switch

⇒ Gate of the PMOS is floating

⇒ PMOS acts like an open switch

⇒ R1 is floating, no current/power, row is disabled!



The rows are also part of the full 88 bit shift register with PMOS transistors to enable the rows as shown on the schematic. The PMOS gates are connected to the outputs which in turn are connected to the chips NMOSs transistors Drains.

Now reasoning trough this circuit we get the following: To enable the row the PMOS needs to conduct since it is connected to VCC with the source. To get the PMOS in conduction the gate needs to be Vth lower than the source node. By connecting it to ground trough the shift-registers NMOS transistors the PMOS is in conductance.

In short: to enable a specific row, we need to set a ‘1’ to the corresponding bit in the shift register connected to the rows.

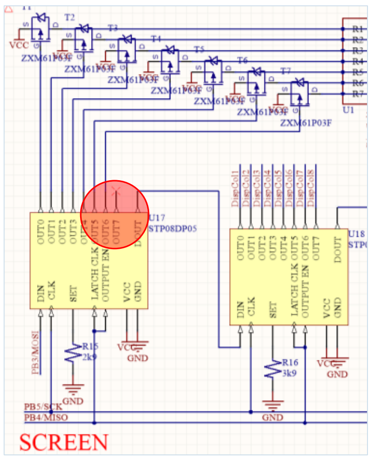
Shift registers → LED-matrix

Enable LED @ row X and column Y

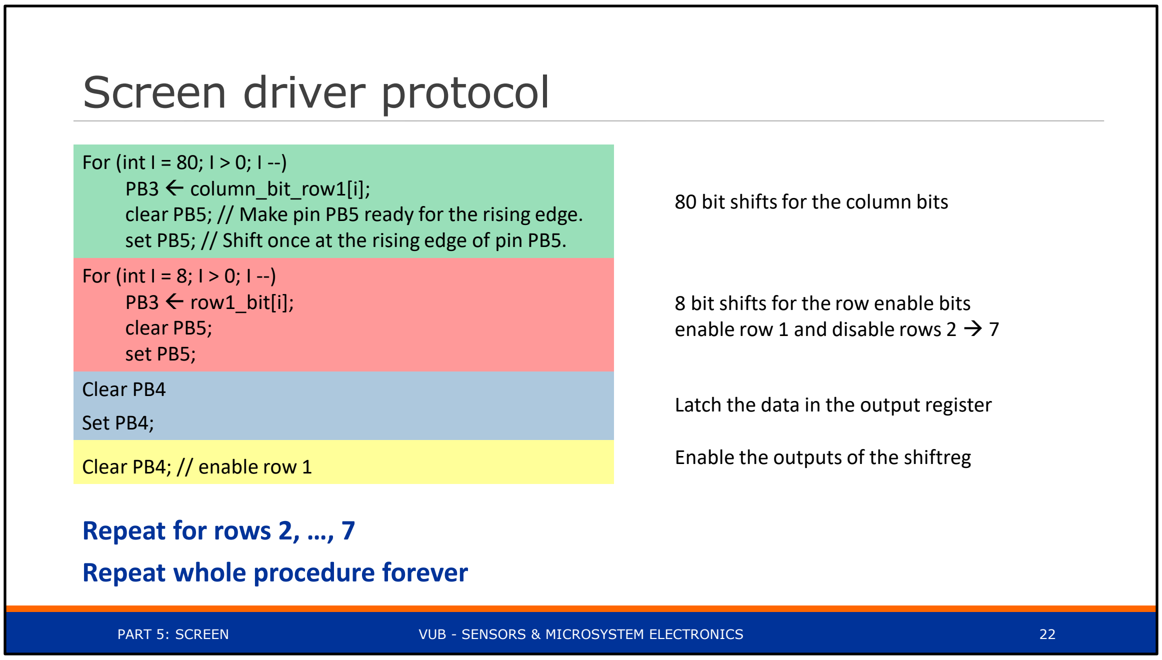
- Shiftregister:bitX ← HIGH
- Shiftregister:bitY ← HIGH

7 physical rows <-> 8 physical outputs

- OUT7 is not connected
- Value of OUT7 does not matter
- VALUE REQUIRED! DO NOT OMIT 1 SHIFT!!!



The rowselect and column select shift registers are all connected in one big chain of 11 shift registers (or 88 bit shiftreg). The 7th bit of the first shiftreg(rowselect) is not connected to anything since we have only 7 rows (0-6). But this pin is still part of the shift register. Because of this it cannot be omitted during the shifting, but its value is not relevant.



Screen driver protocol

```
For (int I = 80; I > 0; I --)
      PB3 ← column_bit_row1[i];
      clear PB5; // Make pin PB5 ready for the rising edge.
      set PB5; // Shift once at the rising edge of pin PB5.
```

80 bit shifts for the column bits

```
For (int I = 8; I > 0; I --)
      PB3 ← row1_bit[i];
      clear PB5;
      set PB5;
```

8 bit shifts for the row enable bits
enable row 1 and disable rows 2 → 7

```
Clear PB4
Set PB4;
```

Latch the data in the output register

```
Clear PB4; // enable row 1
```

Enable the outputs of the shiftreg

Repeat for rows 2, …, 7

Repeat whole procedure forever

PART 5: SCREEN   VUB - SENSORS & MICROSYSTEM ELECTRONICS   22

This shows the screen writing protocol in a c-style pseudocode. You will need to think on how to implement this yourself.

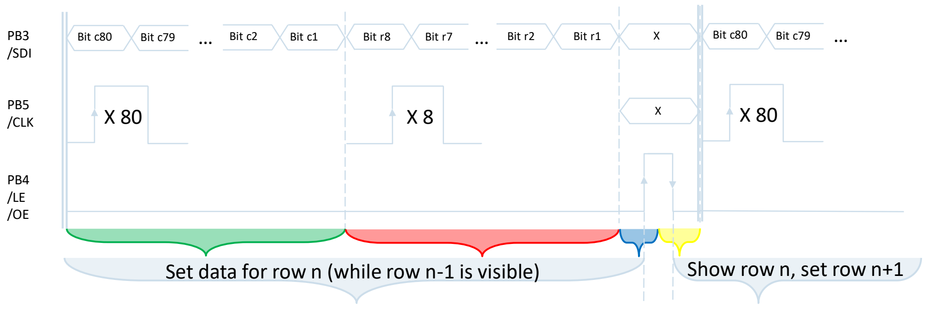
Due to the principle of the shift register, the last column bit is shifted in first, then the before last,…. Same with the row select bits. The last row (nonexistant row 8) is shifted in first, then row 7,….

In the end it takes 88 shifts to completely fill the shiftregister.

After all the shifts, the data is latched into the data register by creating a rising edge on PB4, and then the outputs are enabled by setting PB4 low (OE is active low)

Screen driver protocol

80 column bits for row 1
8 row bits to enable row 1 and disable rows 2 → 7
set PB4 to latch the data
Clear PB4 to make row 1 visible
Repeat for row 2,…,7 and restart

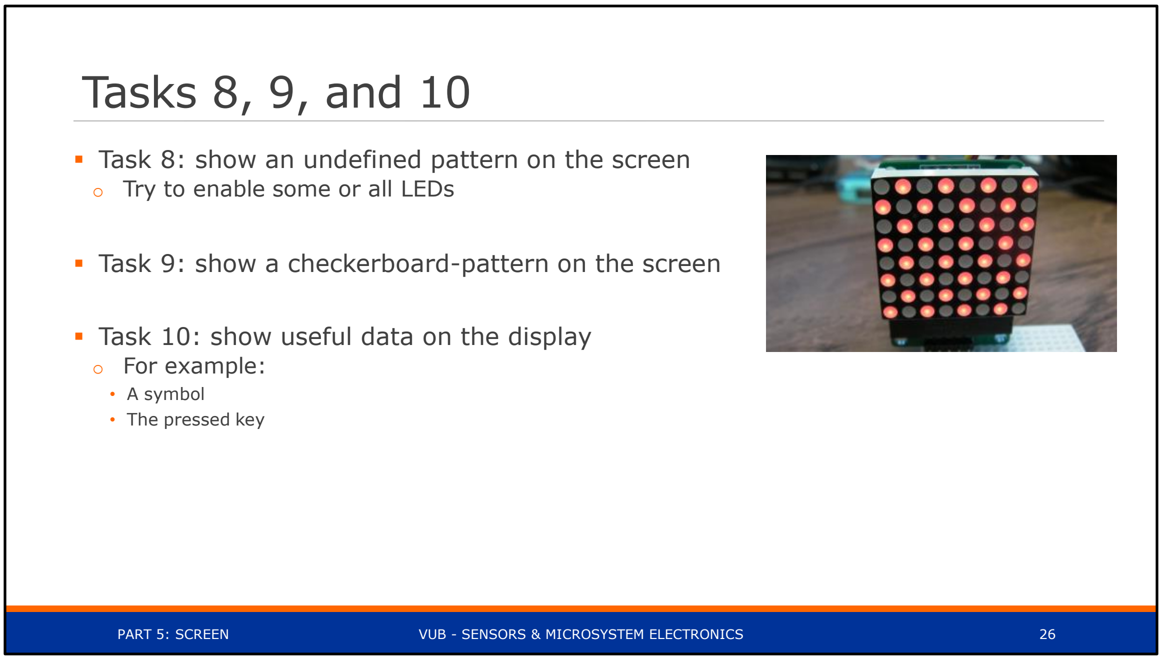


This is the same algorithm in a timing diagram, split in the different parts.

More info for a clean implementation:

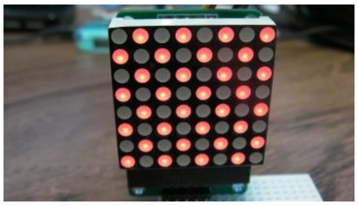
SCREENTIPSANDTRICKS.PDF

Tasks 8, 9, and 10



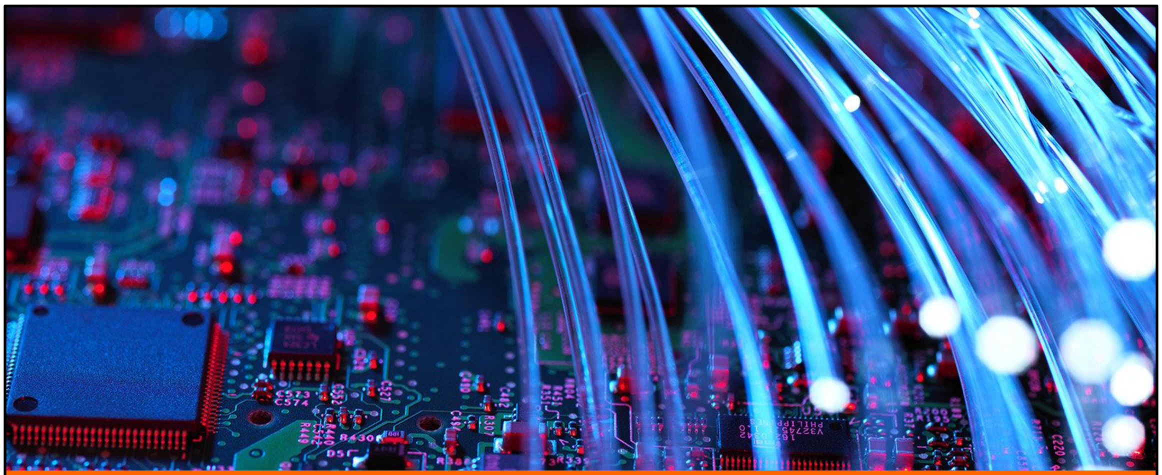
Tasks 8, 9, and 10

- Task 8: show an undefined pattern on the screen
  - Try to enable some or all LEDs
- Task 9: show a checkerboard-pattern on the screen
- Task 10: show useful data on the display
  - For example:
    - A symbol
    - The pressed key



The tasks are made so that you can start simple: just shifting any value in the shift register and enableing some or all LEDs. The next one you already need to multiplex line by line, but no logic yet for choosing which columns to enable. The last task you need to implement the logic to also address the columns.

Each task builds on the next one.



End of Part 5: Screen