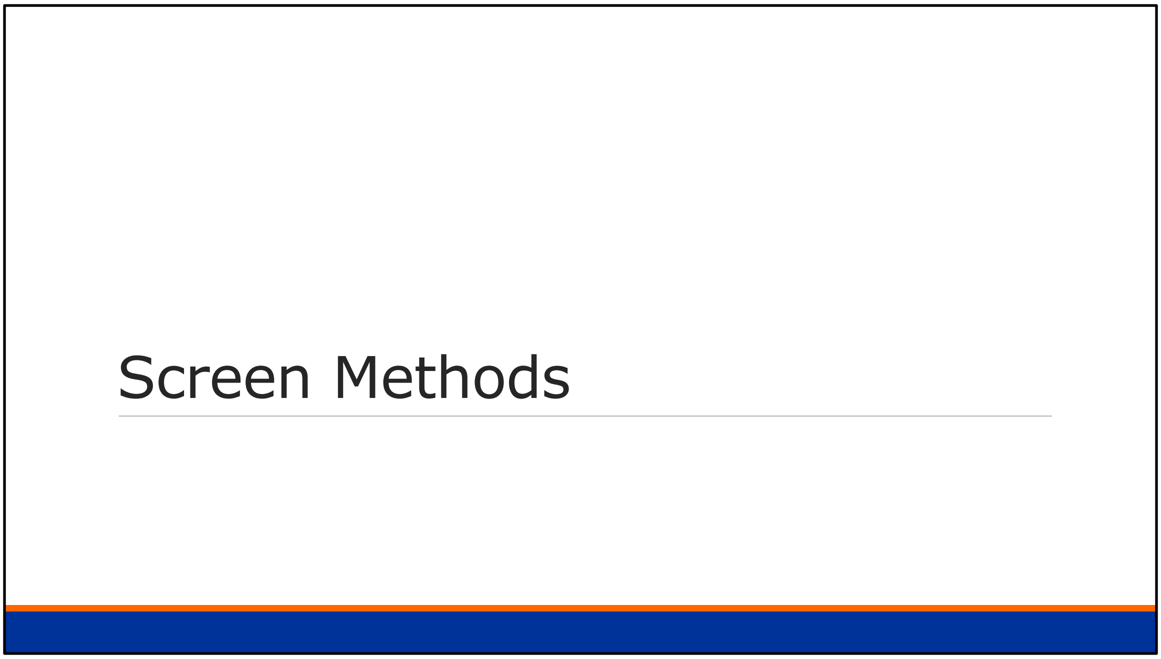
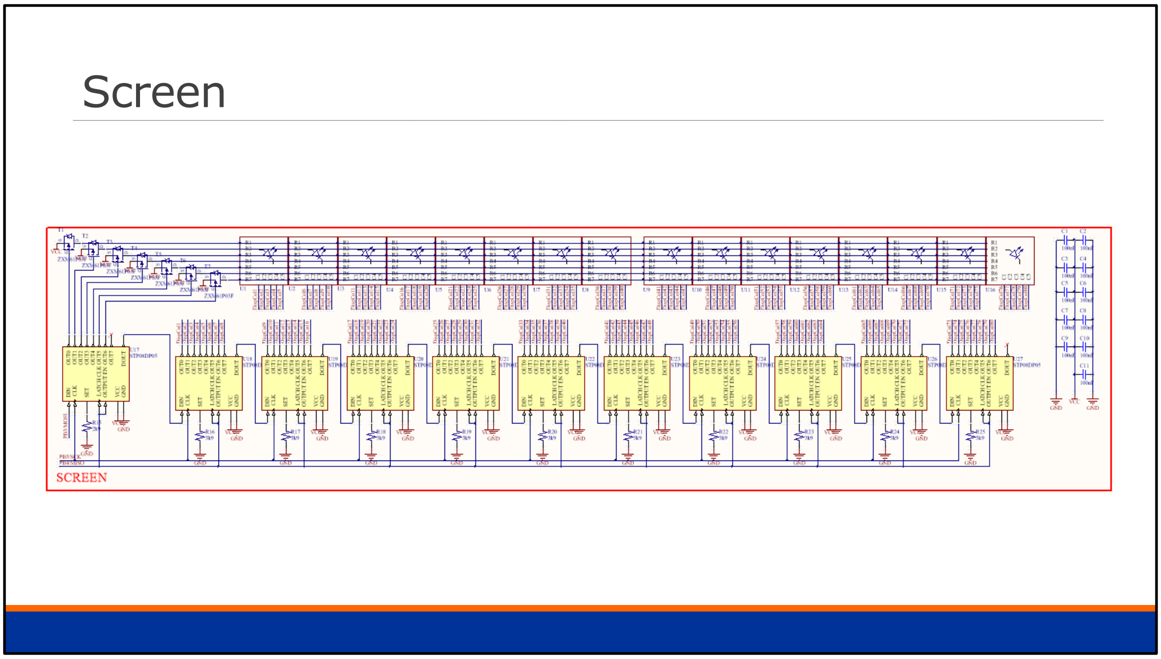
Sensors & Microsystem Electronics: microcontrollers

PART 5: SCREEN – DESIGNING A GOOD STRUCTURE



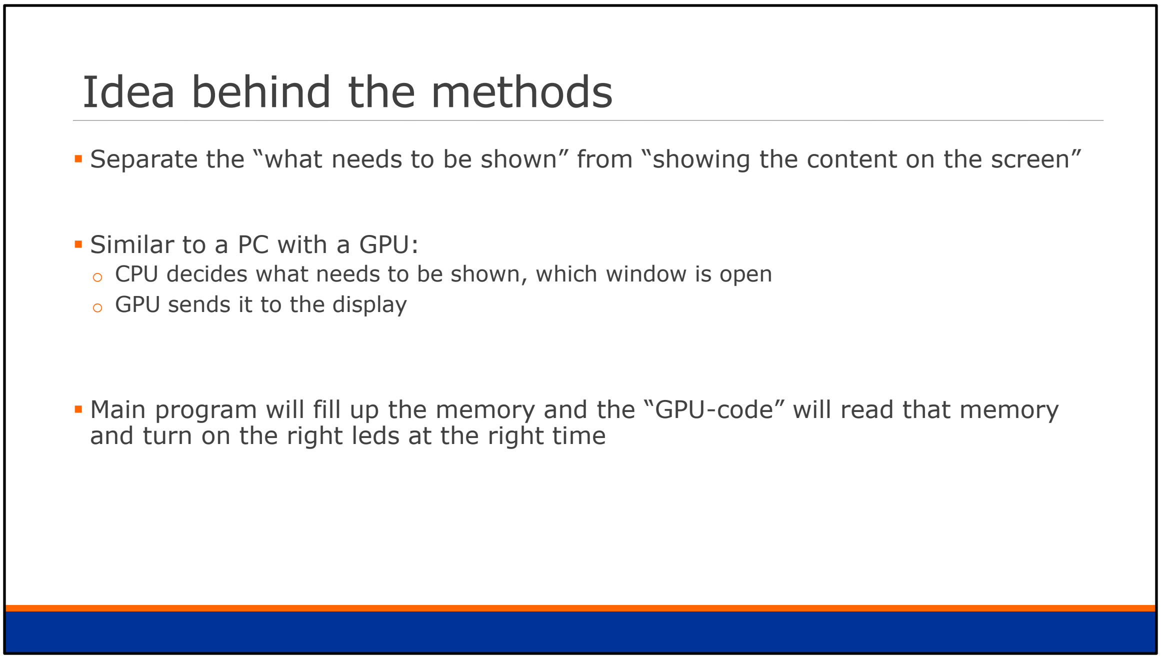
Screen Methods



Screen

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

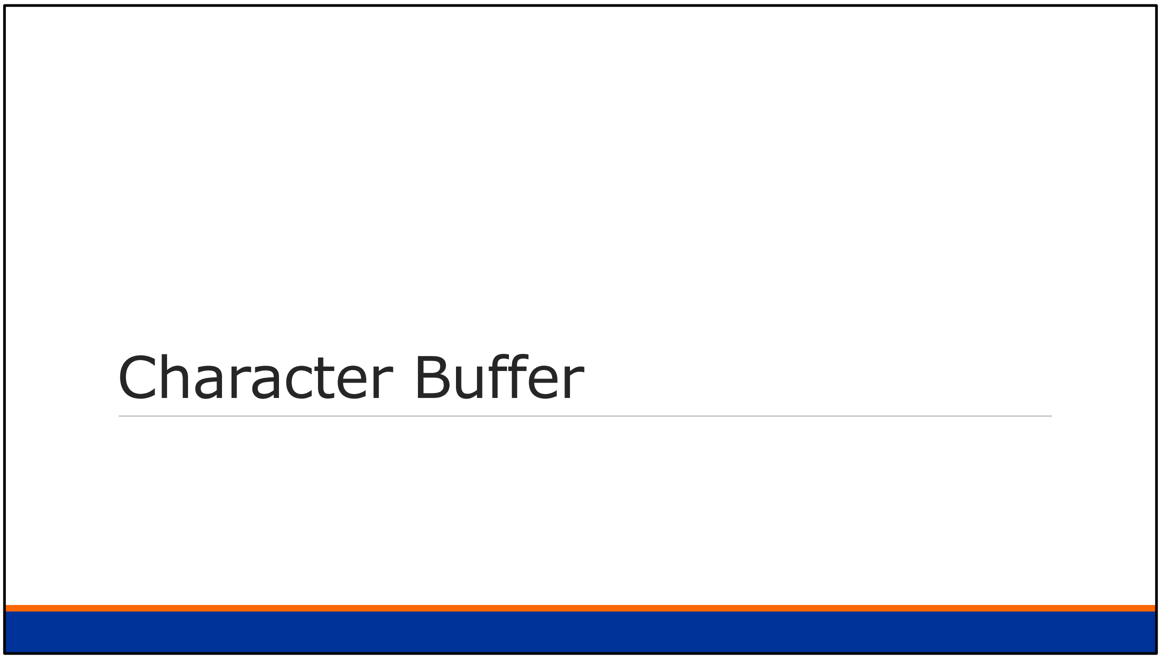
The display is connected to a series of shift registers both in columns and in rows,



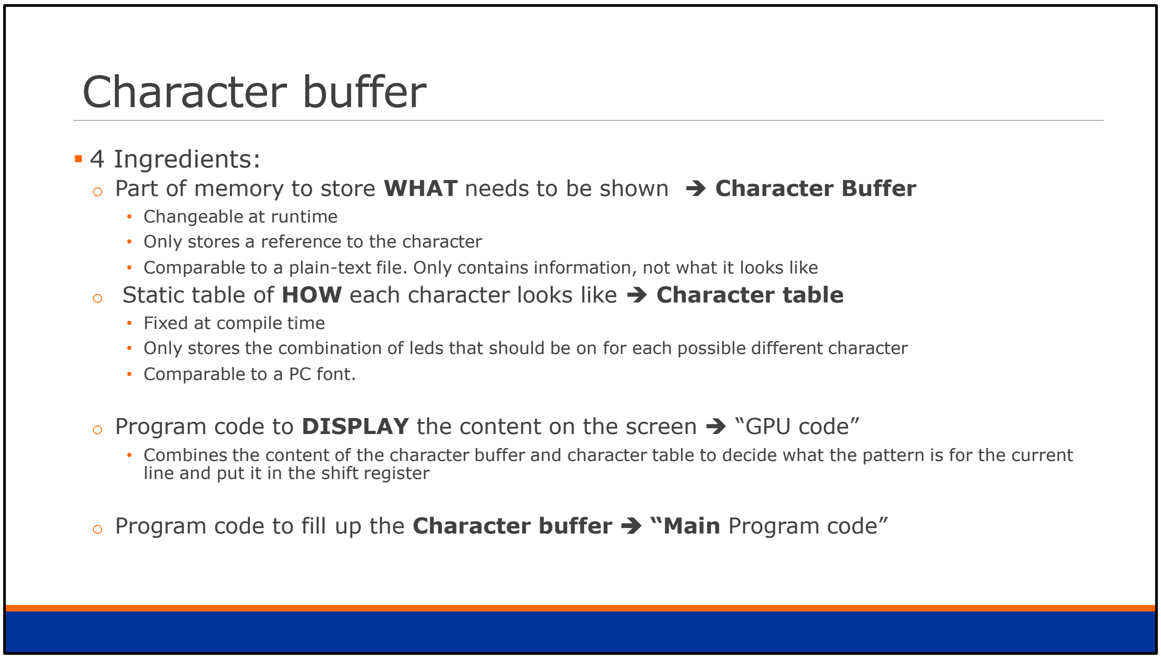
Idea behind the methods

- Separate the “what needs to be shown” from “showing the content on the screen”
- Similar to a PC with a GPU:
  - CPU decides what needs to be shown, which window is open
  - GPU sends it to the display
- Main program will fill up the memory and the “GPU-code” will read that memory and turn on the right leds at the right time

Row 2



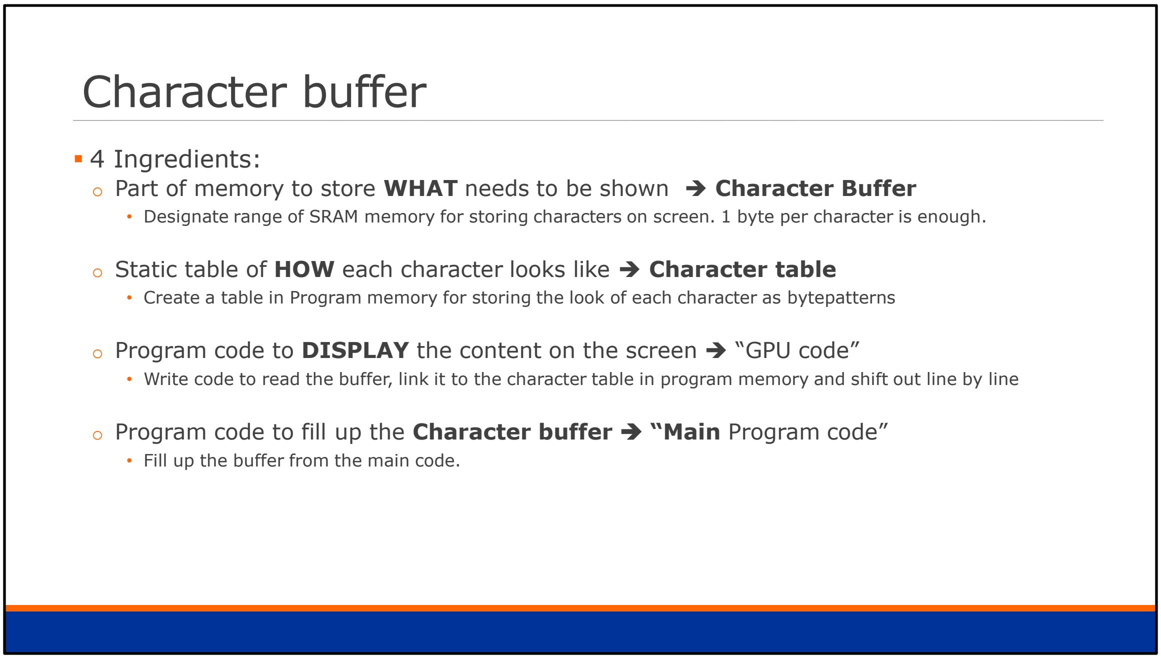
Character Buffer



Character buffer

- 4 Ingredients:
  - Part of memory to store **WHAT** needs to be shown ➔ **Character Buffer**
    - Changeable at runtime
    - Only stores a reference to the character
    - Comparable to a plain-text file. Only contains information, not what it looks like
  - Static table of **HOW** each character looks like ➔ **Character table**
    - Fixed at compile time
    - Only stores the combination of leds that should be on for each possible different character
    - Comparable to a PC font.
  - Program code to **DISPLAY** the content on the screen ➔ “GPU code”
    - Combines the content of the character buffer and character table to decide what the pattern is for the current line and put it in the shift register
  - Program code to fill up the **Character buffer** ➔ **“Main** Program code”

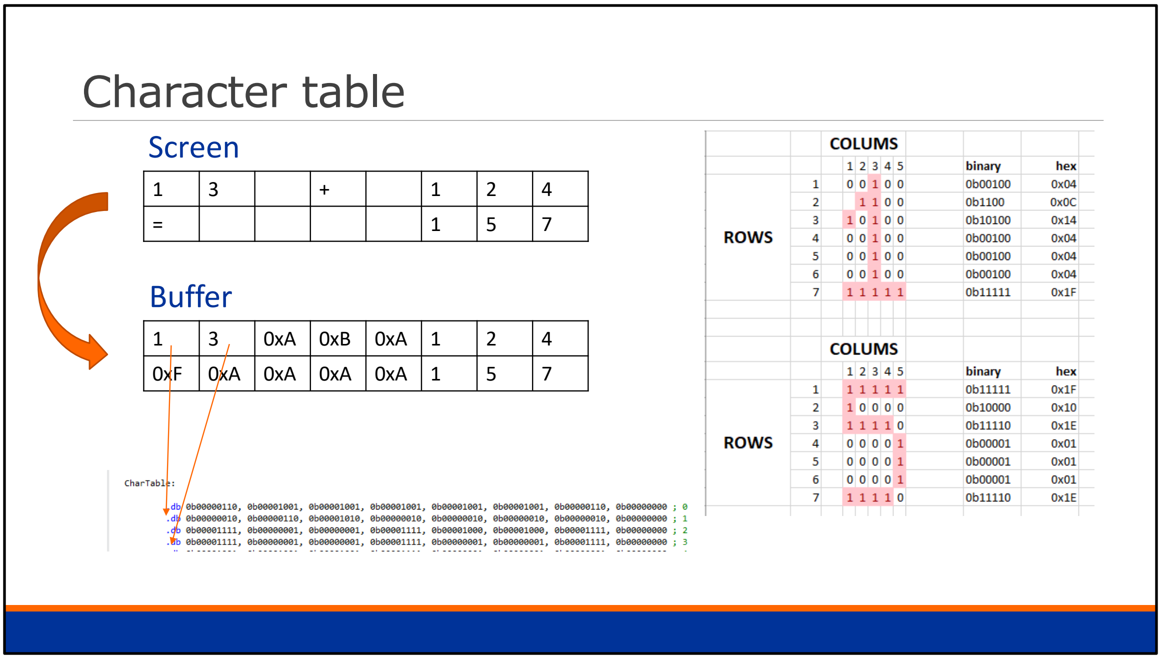
Row 2



Character buffer

- 4 Ingredients:
  - Part of memory to store **WHAT** needs to be shown ➔ **Character Buffer**
    - Designate range of SRAM memory for storing characters on screen. 1 byte per character is enough.
  - Static table of **HOW** each character looks like ➔ **Character table**
    - Create a table in Program memory for storing the look of each character as bytepatterns
  - Program code to **DISPLAY** the content on the screen ➔ “GPU code”
    - Write code to read the buffer, link it to the character table in program memory and shift out line by line
  - Program code to fill up the **Character buffer** ➔ **“Main** Program code”
    - Fill up the buffer from the main code.

Row 2



Character table

Screen

1	3		+		1	2	4
=					1	5	7

Buffer

1	3	0xA	0xB	0xA	1	2	4
0xF	0xA	0xA	0xA	0xA	1	5	7

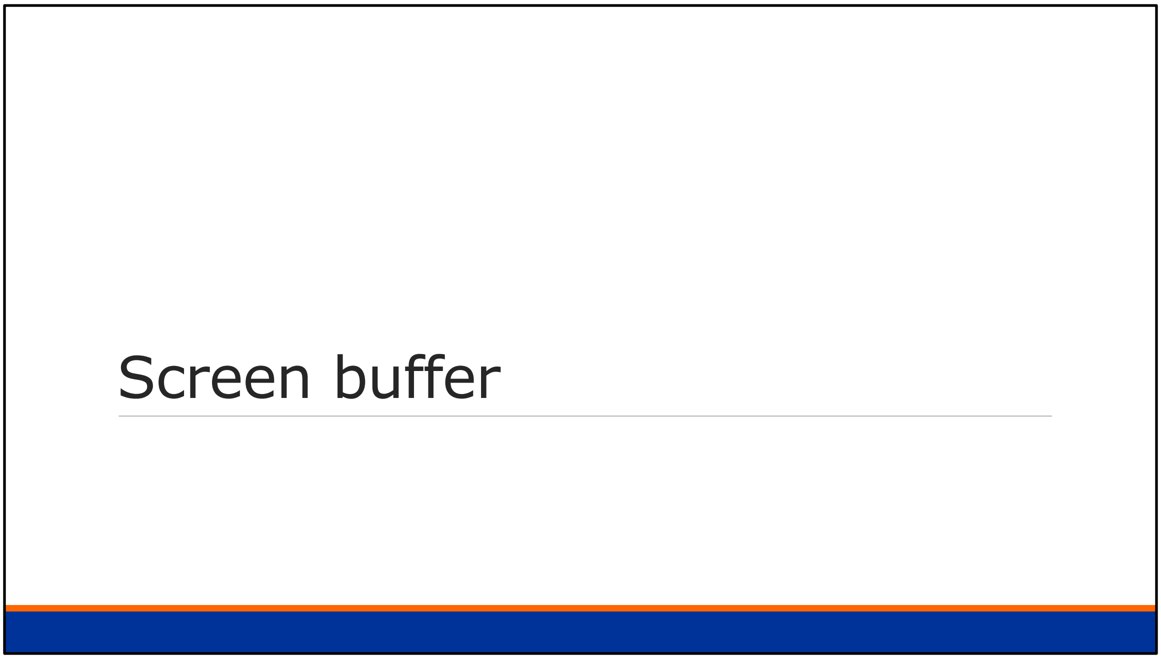
```
CharTable:

        .db 0b00000110, 0b00001001, 0b00001001, 0b00001001, 0b00001001, 0b00001001, 0b00000110, 0b00000000 ; 0
        .db 0b00000010, 0b00000110, 0b00001010, 0b00000010, 0b00000010, 0b00000010, 0b00000010, 0b00000000 ; 1
        .db 0b00001111, 0b00000001, 0b00000001, 0b00001111, 0b00001000, 0b00001000, 0b00001111, 0b00000000 ; 2
        .db 0b00001111, 0b00000001, 0b00000001, 0b00001111, 0b00000001, 0b00000001, 0b00001111, 0b00000000 ; 3
```

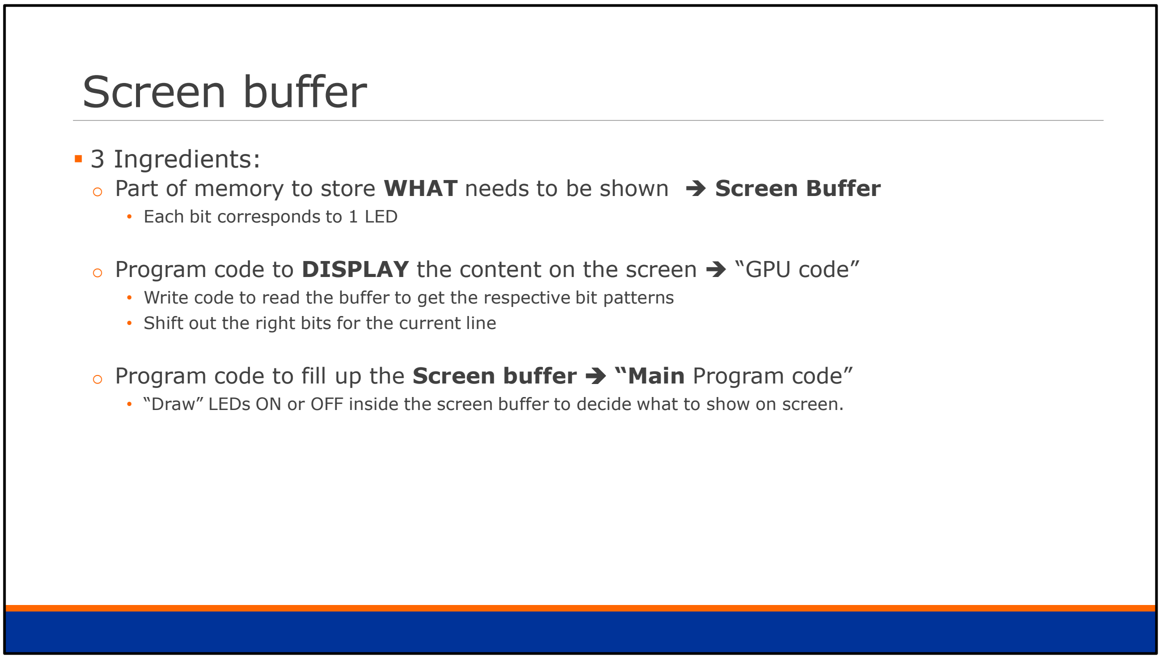
		COLUMS						
		1	2	3	4	5	binary	hex
ROWS	1	0	0	1	0	0	0b00100	0x04
	2		1	1	0	0	0b1100	0x0C
	3	1	0	1	0	0	0b10100	0x14
	4	0	0	1	0	0	0b00100	0x04
	5	0	0	1	0	0	0b00100	0x04
	6	0	0	1	0	0	0b00100	0x04
	7	1	1	1	1	1	0b11111	0x1F

		COLUMS						
		1	2	3	4	5	binary	hex
ROWS	1	1	1	1	1	1	0b11111	0x1F
	2	1	0	0	0	0	0b10000	0x10
	3	1	1	1	1	0	0b11110	0x1E
	4	0	0	0	0	1	0b00001	0x01
	5	0	0	0	0	1	0b00001	0x01
	6	0	0	0	0	1	0b00001	0x01
	7	1	1	1	1	0	0b11110	0x1E

Row 2



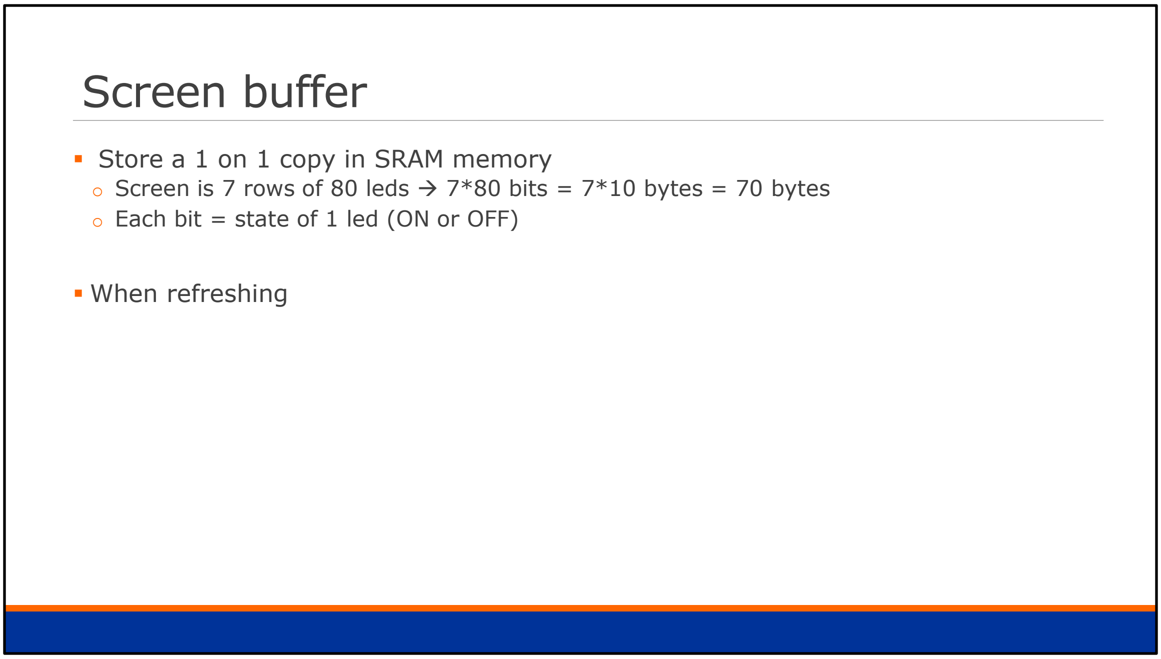
Screen buffer



Screen buffer

- 3 Ingredients:
  - Part of memory to store **WHAT** needs to be shown ➔ **Screen Buffer**
    - Each bit corresponds to 1 LED
  - Program code to **DISPLAY** the content on the screen ➔ “GPU code”
    - Write code to read the buffer to get the respective bit patterns
    - Shift out the right bits for the current line
  - Program code to fill up the **Screen buffer** ➔ **“Main** Program code”
    - “Draw” LEDs ON or OFF inside the screen buffer to decide what to show on screen.

Row 2



Screen buffer

- Store a 1 on 1 copy in SRAM memory
  - Screen is 7 rows of 80 leds → 7*80 bits = 7*10 bytes = 70 bytes
  - Each bit = state of 1 led (ON or OFF)
- When refreshing

Row 2

Common problems that have an easy solution

- Bright line at the top or the bottom
  - Duty cycle of the lines is different... Try to figure out why

- Ghost-line: leds light up dimly in the same pattern to the line above
  - Add some delay between the output-disable and the latch/OE

- Random leds are on, flickering, garbage data
  - Don’t forget to fill the buffers
  - If you have an interrupt don’t forget to push and pop the relevant registers

- Screen stops working when I press a button
  - The button code is blocking and screen does not refresh anymore
  - Check your code paths, or refresh the screen in an interrupt.