Topic 8: The LCD Screen

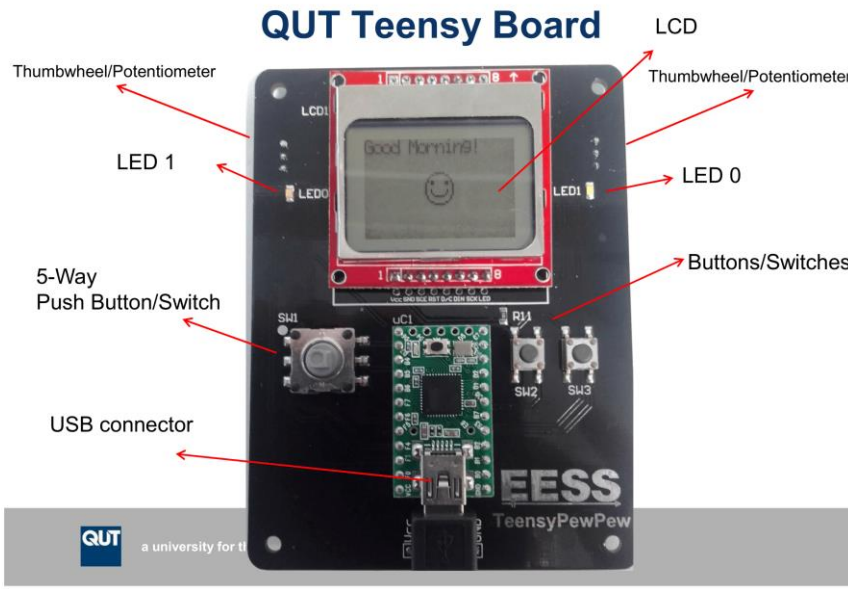
Bilal Arain

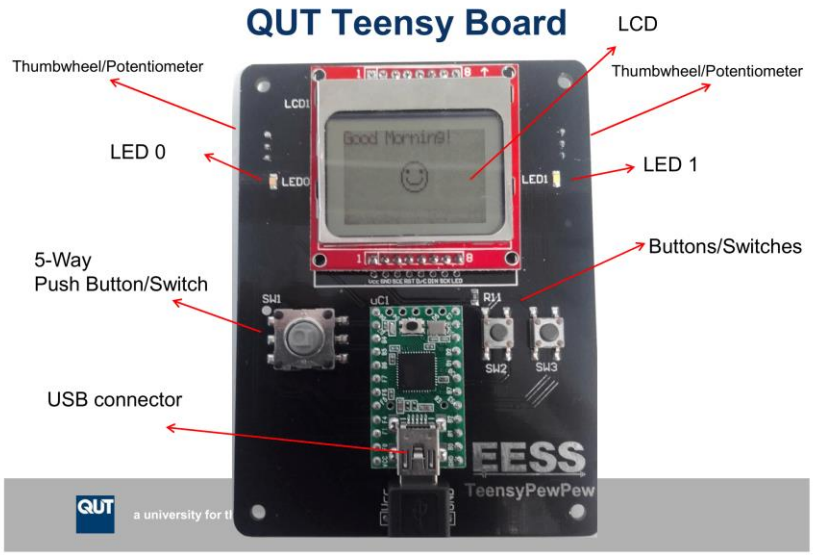
Reference: The slides were originally prepared by Dr Luis Mejias

Outline

- Revisit last week's lecture (Topic 7)
- Writing to the LCD screen in the teensy board
- Reviewing cab202_teensy graphics library

Revisiting Topic 7





Revisiting Topic 7

- Turning an LED on/off
- Used Buttons to turn LED on/off
- LED example and bitwise shifts in more details

LED=Light Emitting Diode

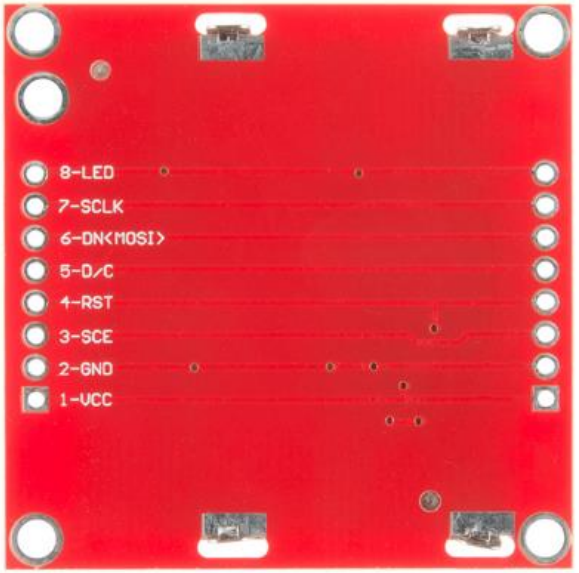
The LCD Screen

- Schematic
- Datasheets
- Programming the device

Nokia 5110 LCD screen
The PCD8544 LCD Controller/driver

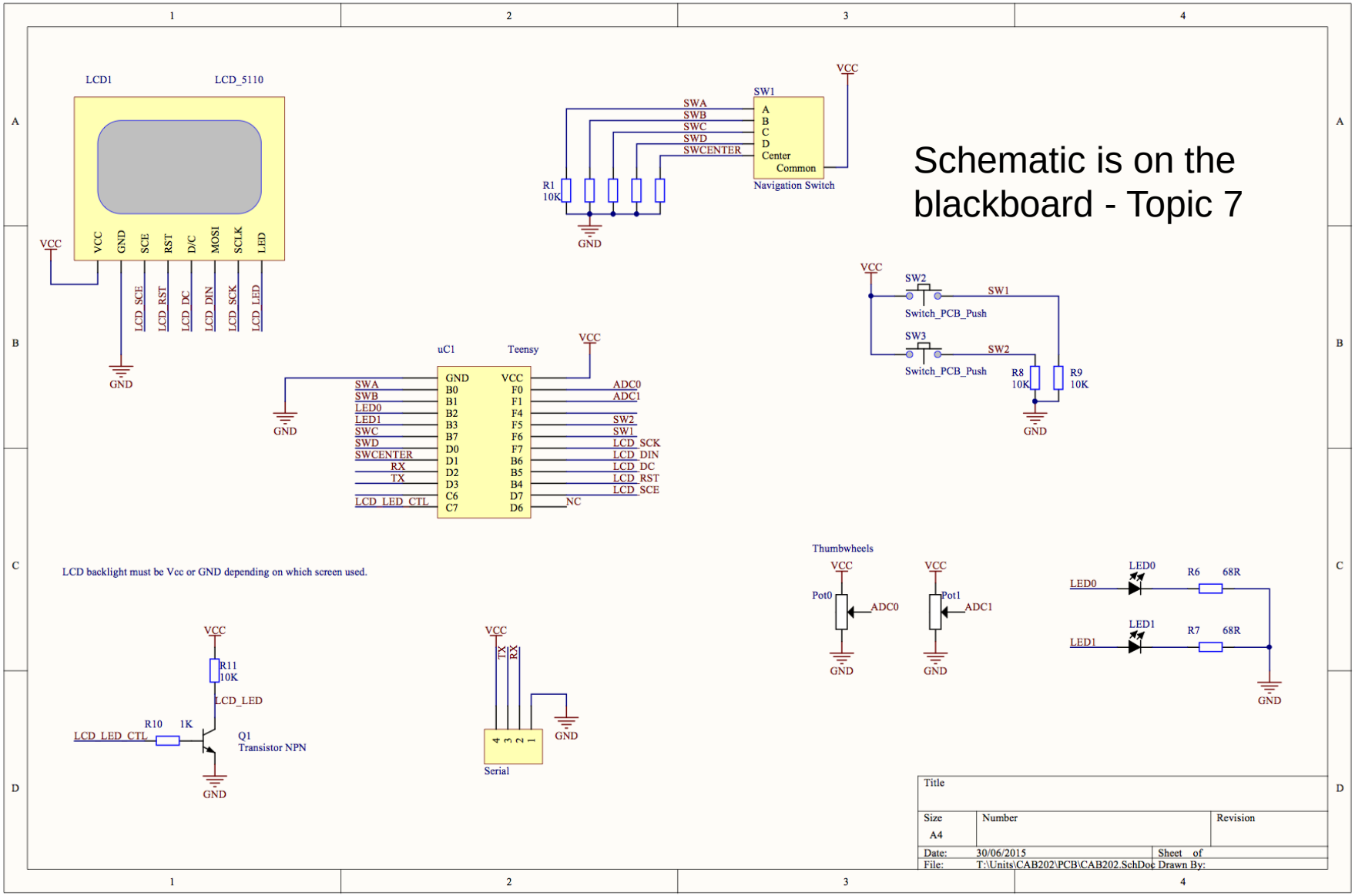
- 48×84 pixel monochrome display.
- Build-in back-light
- Interfaces to microcontrollers via serial bus interface
- The controller has a small amount of RAM which holds the pixel data for display
- Recommended reading
- Learning Resources→Microcontrollers→Nokia5110-LCD-Screen.pdf (PCD8544 Data Sheet).

The PCD8544 LCD Controller/driver



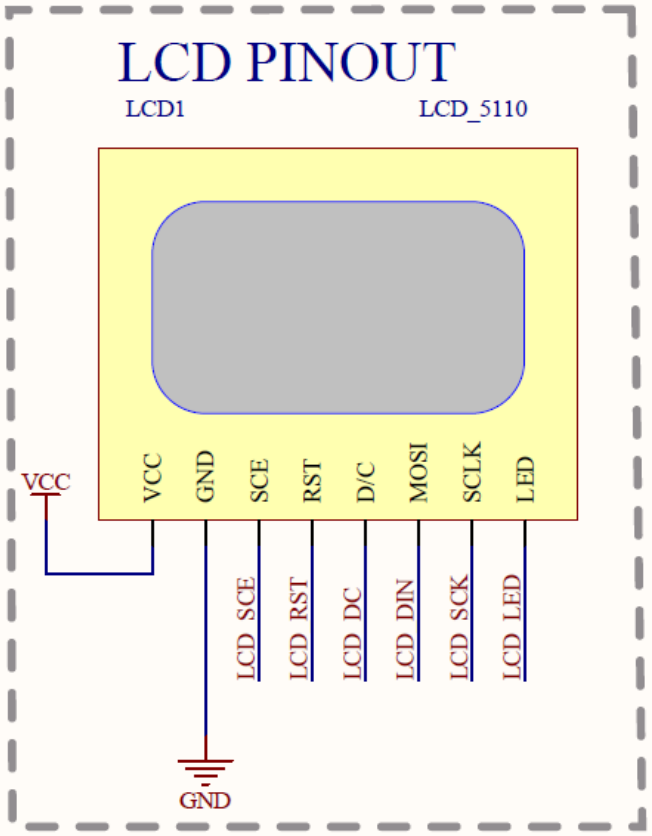
SCE = Chip Select
RST = Reset
DC = Mode select
DIN = Serial data in (MOSI)
SCLK = Serial clock
LED = backlight supply

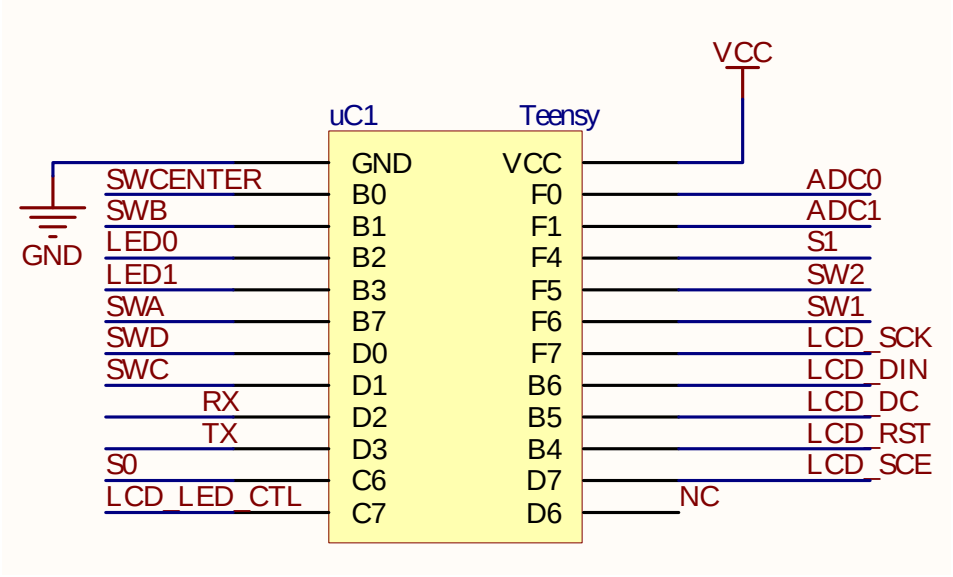
Teensy schematic



Schematic is on the blackboard - Topic 7

The LCD Screen





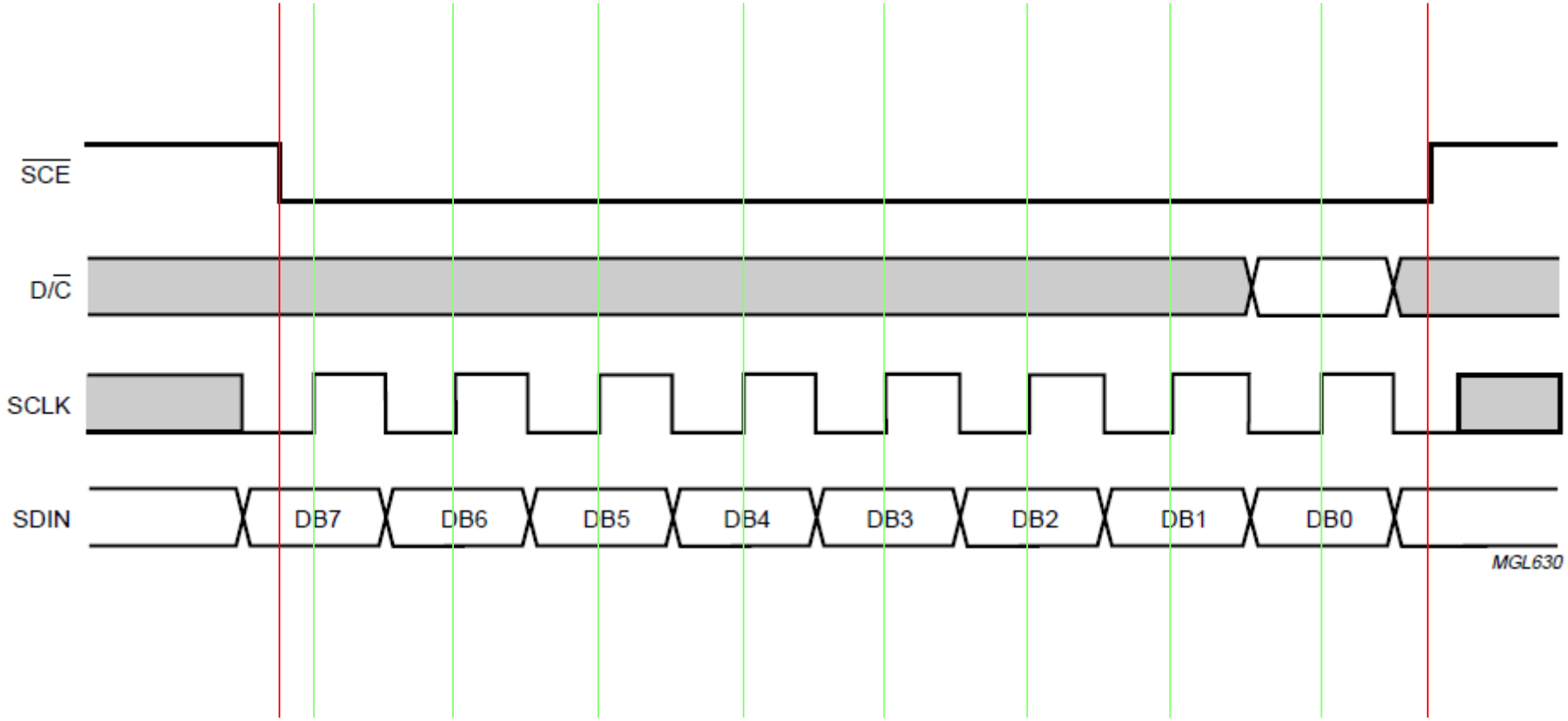
Port C, pin 7 → LCD backlight.
Port F, pin 7 → LCD Serial Clock pin.
Port B, pin 6 → LCD Serial Data Input pin.
Port B, pin 5 → LCD Serial Data/Command pin.
Port B, pin 4 → LCD Reset pin.
Port D, pin 7 → LCD Chip Select pin.

http://ww1.microchip.com/downloads/en/devicedoc/Atmel-7766-8-bit-AVR-ATmega16U4-32U4_Summary.pdf

The LCD Screen

LCD Pin	Notes
Vcc	3.3V Vcc
GND	GND
SCE	Chip select pin. Low: The start of data transmission High: SCLK clock pulses have no effect
RST	Reset to default configuration.
D/C	Data/Command Pin Low: means incoming data must be interpreted as a command. High: means incoming data is pixel data to be displayed
DIN	Serial Data input pin
SCK	Serial Clock Pin
LED	Backlight

The LCD Screen



Reference: LCD data sheet, Fig 10, p12

The LCD Screen

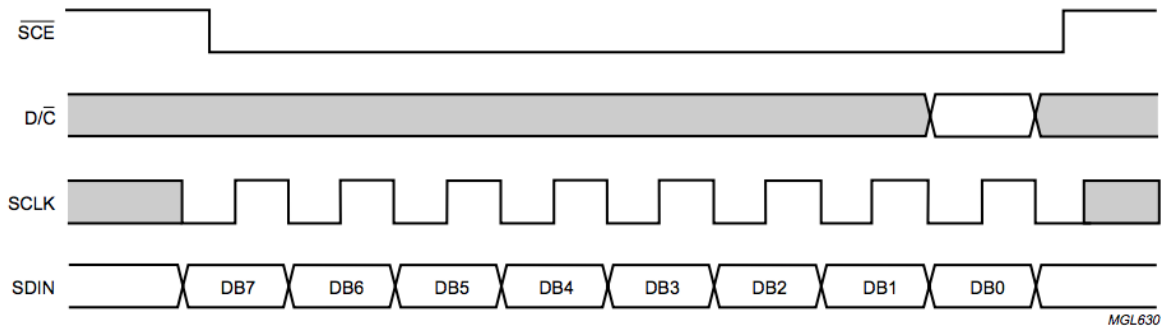


Fig.10 Serial bus protocol - transmission of one byte.

new byte

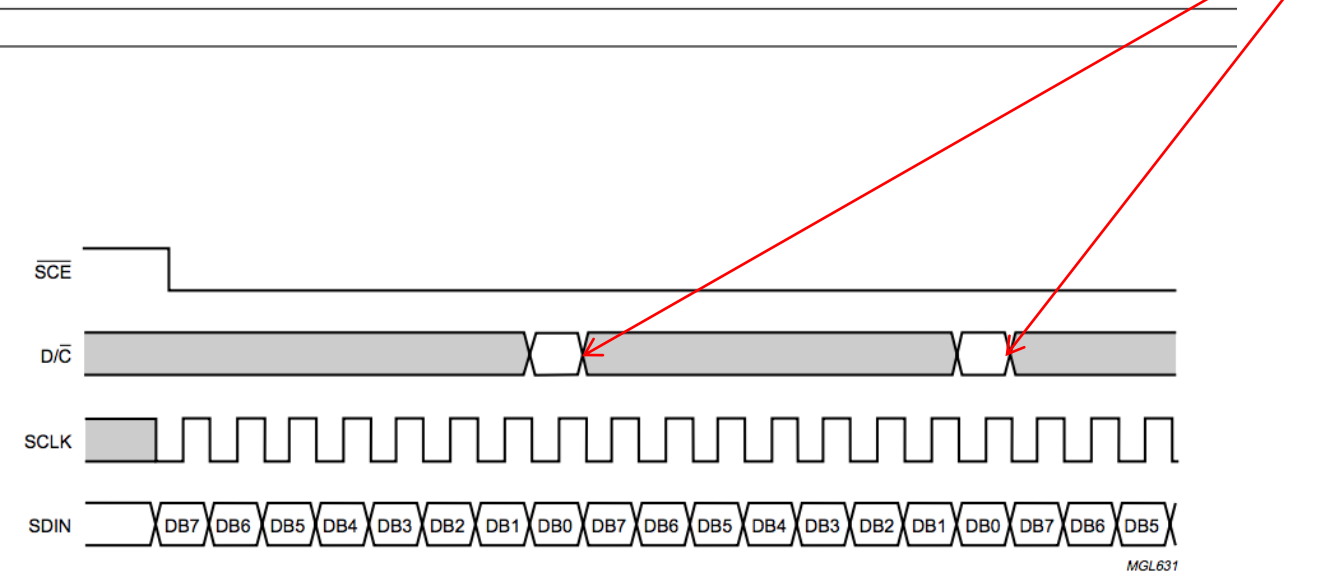


Fig.11 Serial bus protocol - transmission of several bytes.

We will use software routines (libraries) that take cake of the low level Interaction with the LCD

A library called: cab202_teensy.a

However, we will also show you how to write directly to the screen.

What is a library? cab202_teensy.a

- A library in C is a group of functions and declarations, exposed for use by other programs. The library therefore consists of an interface expressed in a .h file (named the "header") and an implementation expressed in a .c file.

- Like a .zip file, they're just a bag of object files — containing functions, of course — with a table of contents in front giving the address of each name.

- They can be static or dynamic.
  - Static are joined to the main program at compiling
  - Dynamic are referenced at compiling, but aren’t used until runtime

Our graphics library cab202_teensy.a

- Provides a high level interface (functions) to write characters on the LCD screen.
- Similar to zdk, it will allows us to draw characters on the screen.
- How it is used?
  - Include “graphics.h” and “lcd.h” in your main.
  - Include cab202_teensy directory in your makefile (see blackboard topic 8)
  - Use library functions

Our graphics library cab202_teensy.a

- Useful functions (graphics.h)
  - void show_screen(void);
    - Copy content entire screen buffer to the LCD
  - void clear_screen(void);
    - Clear/reset all the screen
  - void draw_pixel(int x, int y, colour_t colour);
    - Draw a pixel on the screen using FG or BG colours
  - void draw_line(int x1, int y1, int x2, int y2, colour_t colour);
    - Draw a line from (x1,y1) to (x2,y2) using FG or BG colours
  - void draw_char(int top_left_x, int top_left_y, char character, colour_t colour);
    - Draw a single character using FG or BG colours. Position is referenced to the top left corner of the character
  - void draw_string(int top_left_x, int top_left_y, char *text, colour_t colour) ;
    - Draw a string of character. Position is referenced to the top left corner of the character

Our graphics library cab202_teensy.a

- Useful functions (lcd.h)
  - Void lcd_init(uint8_t contrast);
    - Initialise the screen with a value for contrast
  - void lcd_write(uint8_t dc, uint8_t data);
    - Write a byte of data to the screen
  - void lcd_clear(void);
    - Clear the screen (clear the pixels)
  - void lcd_position(uint8_t x, uint8_t y);
    - Set the address (position cursor) at the address x,y (see lecture notes **Pixel data storage** section)

Programming the LCD

Port D, pin 7 → LCD Chip Select pin.
Port B, pin 4 → LCD Reset pin.
Port F, pin 7 → LCD Serial Clock pin.
Port B, pin 6 → LCD Serial Data Input pin.
Port B, pin 5 → LCD Serial Data/Command pin.

macros.h

```
#define SET_OUTPUT(portddr, pin)
#define SET_BIT(reg, pin)
#define CLEAR_BIT(reg, pin)
```

lcd.h

```
#define DCPIN  5   // PORTB
#define RSTPIN 4   // PORTB
#define DINPIN  6   // PORTB
#define SCKPIN 7   // PORTF
#define SCEPIN 7   // PORTD
```

```
void lcd_init(uint8_t contrast) {
        // Set up the pins connected to the LCD as outputs
        SET_OUTPUT(DDRD, SCEPIN);
        SET_OUTPUT(DDRB, RSTPIN);
        SET_OUTPUT(DDRB, DCPIN);
        SET_OUTPUT(DDRB, DINPIN);
        SET_OUTPUT(DDRF, SCKPIN);

        CLEAR_BIT(PORTB, RSTPIN);
        SET_BIT(PORTD, SCEPIN);
        SET_BIT(PORTB, RSTPIN);

        lcd_write(LCD_C, 0x21); // Enable LCD extended command set
        lcd_write(LCD_C, 0x80 | contrast ); // Set LCD Vop (Contrast)
        lcd_write(LCD_C, 0x04);
        lcd_write(LCD_C, 0x13); // LCD bias mode 1:48

        lcd_write(LCD_C, 0x0C); // LCD in normal mode.
        lcd_write(LCD_C, 0x20); // Enable LCD basic command set
        lcd_write(LCD_C, 0x0C);

        lcd_write(LCD_C, 0x40); // Reset row to 0
        lcd_write(LCD_C, 0x80); // Reset column to 0
}
```

Programming the LCD

Command bytes write to the LCD

// LCD Command and Data
#define LCD_C 0
#define LCD_D 1

```
void lcd_init(uint8_t contrast) {
        // Set up the pins connected to the LCD as outputs
        SET_OUTPUT(DDRD, SCEPIN);
        SET_OUTPUT(DDRB, RSTPIN);
        SET_OUTPUT(DDRB, DCPIN);
        SET_OUTPUT(DDRB, DINPIN);
        SET_OUTPUT(DDRF, SCKPIN);

        CLEAR_BIT(PORTB, RSTPIN);
        SET_BIT(PORTD, SCEPIN);
        SET_BIT(PORTB, RSTPIN);

        lcd_write(LCD_C, 0x21); // Enable LCD extended command set
        lcd_write(LCD_C, 0x80 | contrast ); // Set LCD Vop (Contrast)
        lcd_write(LCD_C, 0x04);
        lcd_write(LCD_C, 0x13); // LCD bias mode 1:48

        lcd_write(LCD_C, 0x0C); // LCD in normal mode.
        lcd_write(LCD_C, 0x20); // Enable LCD basic command set
        lcd_write(LCD_C, 0x0C);

        lcd_write(LCD_C, 0x40); // Reset row to 0
        lcd_write(LCD_C, 0x80); // Reset column to 0
}
```

Programming the LCD

INSTRUCTION	D/C	COMMAND BYTE								DESCRIPTION
		DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
(H = 0 or 1) *Either instruction set*										
NOP	0	0	0	0	0	0	0	0	0	no operation
Function set	0	0	0	1	0	0	PD	V	H	power down control; entry mode; extended instruction set control (H)
Write data	1	D7	D6	D5	D4	D3	D2	D1	D0	writes data to display RAM
(H = 0) *Basic instruction set*										
Display control	0	0	0	0	0	1	D	0	E	sets display configuration
Set Y address of RAM	0	0	1	0	0	0	Y2	Y1	Y0	sets Y-address of RAM; 0 ≤ Y ≤ 5
Set X address of RAM	0	1	X6	X5	X4	X3	X2	X1	X0	sets X-address part of RAM; 0 ≤ X ≤ 83
(H = 1) *Extended instruction set*										
Temperature control	0	0	0	0	0	0	1	TC1	TC0	set Temperature Coefficient (TCx)
Bias system	0	0	0	0	1	0	BS2	BS1	BS0	set Bias System (BSx). *Determined by number of multiplexed planes. This unit has MUX == 1:48, so the appropriate value for Bias is 3. See page 16 of data sheet.*
Set VOP	0	1	VOP6	VOP5	VOP4	VOP3	VOP2	VOP1	VOP0	write VOP to register; sets the operating voltage; 0 ≤ VOP ≤ 127 == 0x7f. *Provides adjustable contrast.*

Interpretation:

BIT	0	1
PD	chip is active	chip is in Power-down mode
V	horizontal addressing	vertical addressing
H	use basic instruction set	use extended instruction set
D and E 00 10 01 11	 display blank normal mode all display segments on inverse video mode	
TC1 and TC0 00 01 10 11	 VLCD temperature coefficient 0 VLCD temperature coefficient 1 VLCD temperature coefficient 2 VLCD temperature coefficient 3	

Function set
0b00100001 = 0x21

Section 8 of the data sheet (pages 14–16).

Programming the LCD

```
void lcd_write(uint8_t dc, uint8_t data) {
        // Set the DC pin based on the parameter 'dc' (Hint: use the WRITE_BIT macro)
        WRITE_BIT(PORTB,DCPIN,dc);

        // Pull the SCE/SS pin low to signal the LCD we have data
        CLEAR_BIT(PORTD,SCEPIN);

        // Write the byte of data using "bit bashing"
        for(int i = 7; i >= 0; i--) {
                CLEAR_BIT(PORTF, SCKPIN) ;
                if((data>>i) & (1 == 1)) {
                        SET_BIT(PORTB, DINPIN);
                } else {
                        CLEAR_BIT(PORTB, DINPIN);
                }
                SET_BIT(PORTF, SCKPIN);
        }

        // Pull SCE/SS high to signal the LCD we are done
        SET_BIT(PORTD, SCEPIN);
}
```

Example of using lcd_write

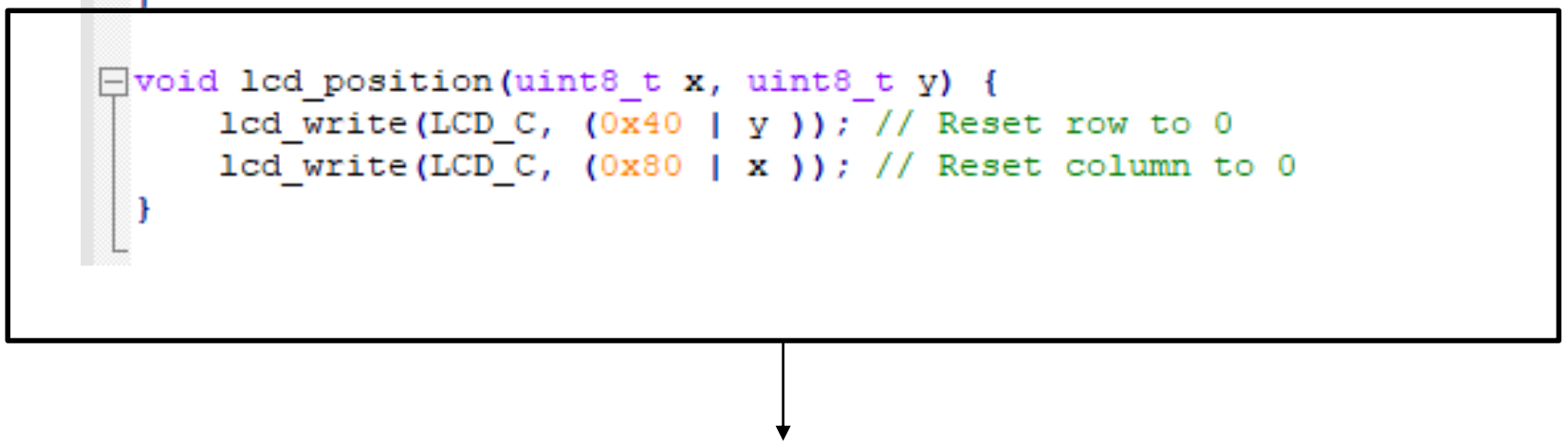
```
void show_screen(void) {
    // Reset our position in the LCD RAM
    lcd_position(0, 0);

    // Iterate through our buffer and write each byte to the LCD.
    for ( int i = 0; i < LCD_BUFFER_SIZE; i++ ) {
        lcd_write(LCD_D, screen_buffer[i]);
    }
}
```

Programming the LCD

```
void lcd_clear(void) {
    // For each of the bytes on the screen, write an empty byte
    for (int i = 0; i < LCD_X * LCD_Y / 8; i++) {
        lcd_write(LCD_D, 0x00);
    }
}
```

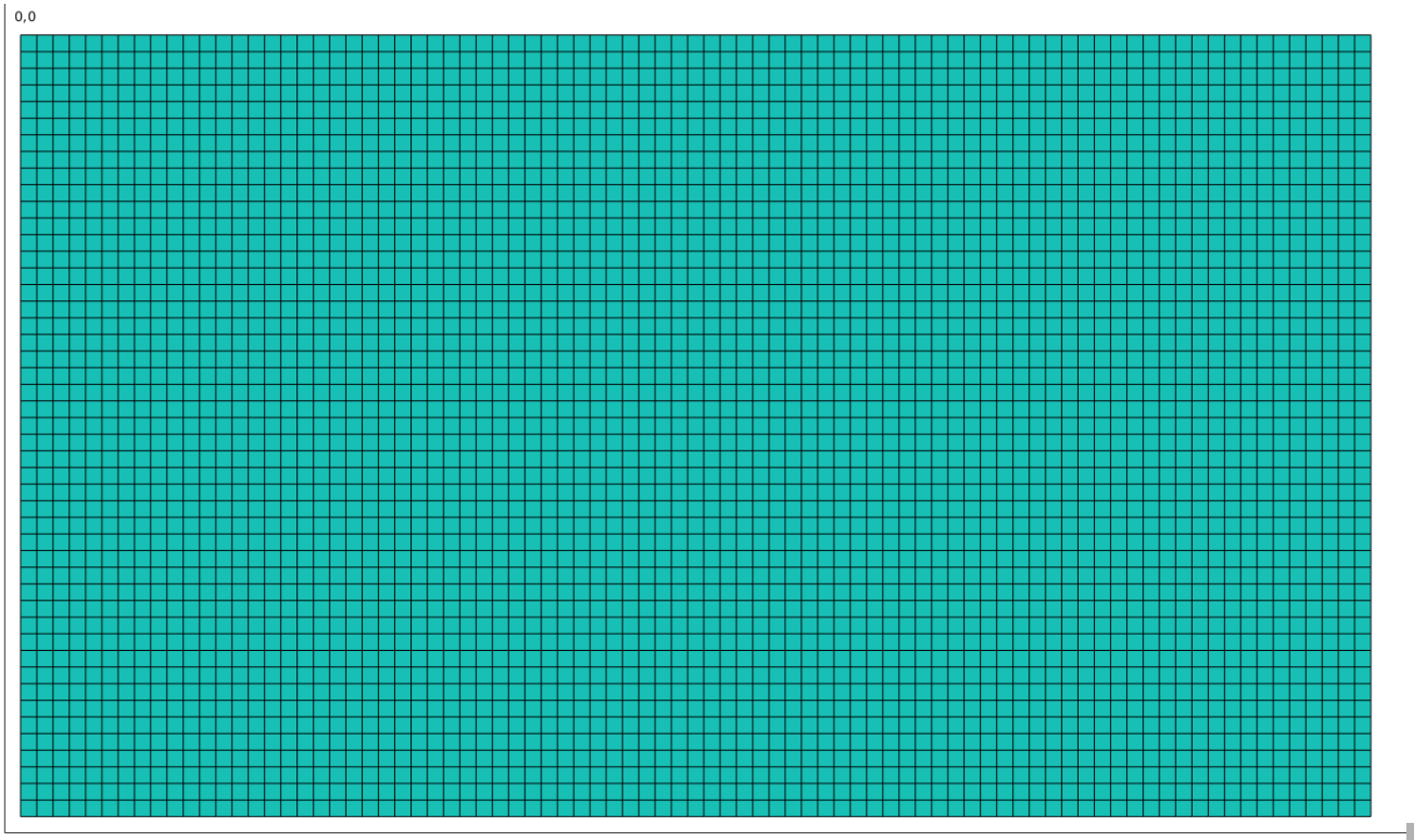
```
void lcd_position(uint8_t x, uint8_t y) {
    lcd_write(LCD_C, (0x40 | y )); // Reset row to 0
    lcd_write(LCD_C, (0x80 | x )); // Reset column to 0
}
```



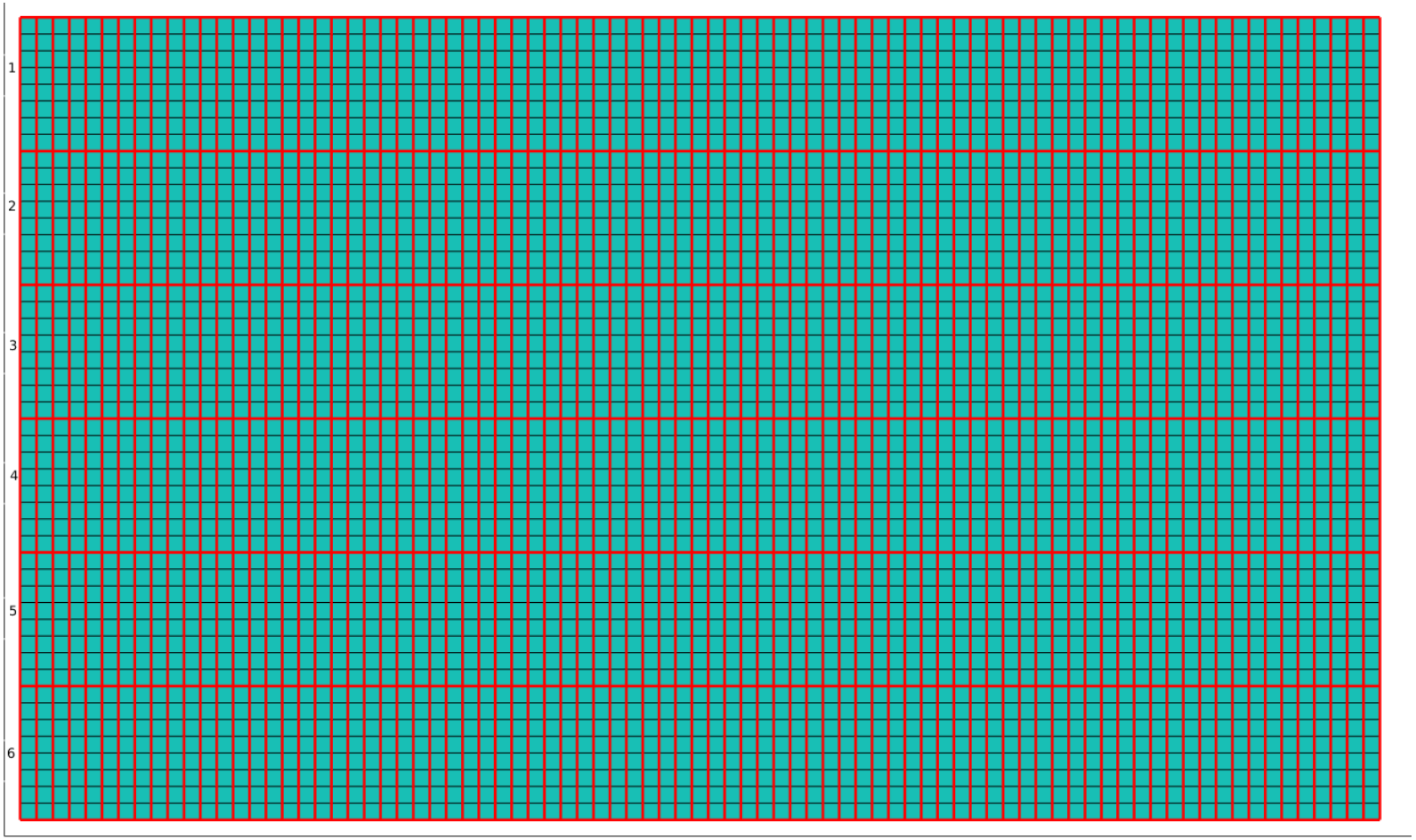
To understand lcd_position function, we need to know how pixel data is stored in RAM

Programming the LCD

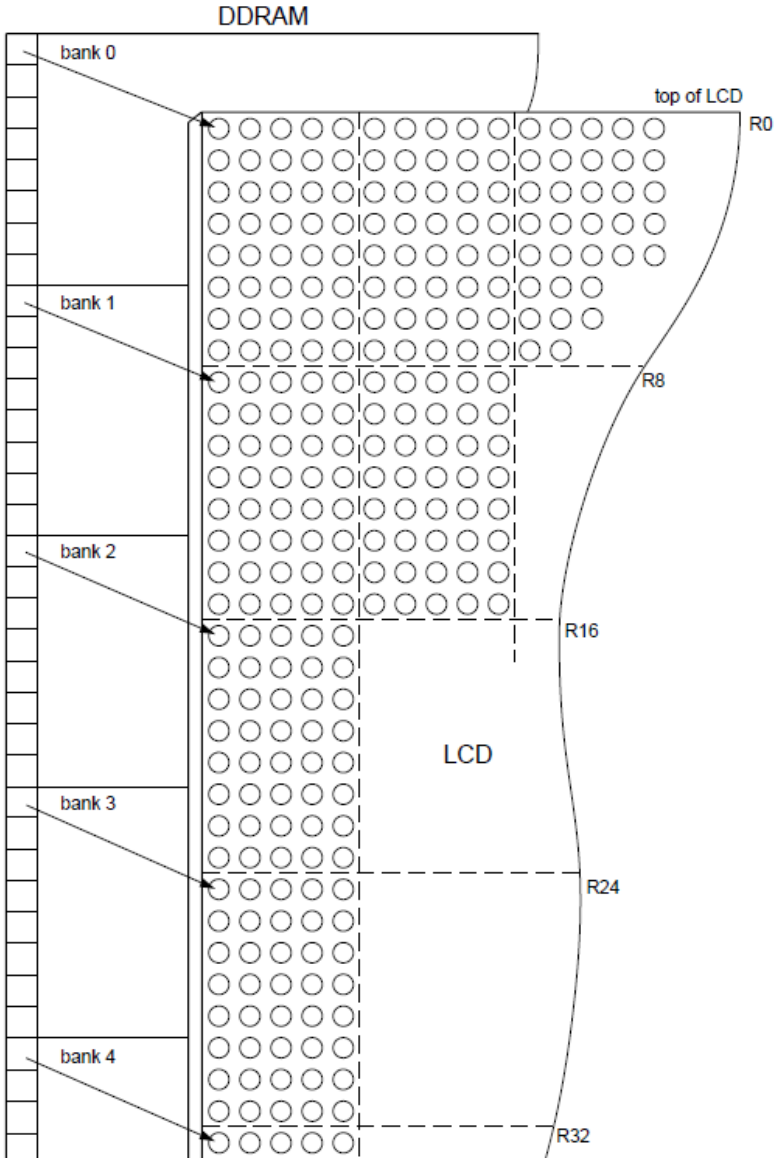
- LCD screen coordinates



Programming the LCD



Programming the LCD



- Each bank contains a horizontal band of pixels which stretches from the left to right side of the display.
- The pixels are arranged in vertical blocks of 8, and each block of 8 pixels is packed into a byte.
- Every time we write data to the LCD display we replace a complete block of 8 pixels

Programming the LCD

- To write an 8-bit block of pixel data **(pixel_block)** at screen coordinates **(px,py)**

- Get the cursor position:

```
x = px;
y = py / 8;
```

- Move LCD internal cursor:

```
LCD_CMD(lcd_set_function, lcd_instr_basic | lcd_addr_horizontal);
LCD_CMD(lcd_set_x_addr, x);
LCD_CMD(lcd_set_y_addr, y);
```

- Write the byte value:

```
LCD_DATA(pixel_block);
```

Teensy
Let's write some code

- Draw characters
  - TeensyBlank.c
- Draw a line
  - TeensyLines.c
- Changing contrast
  - ContrastDemo.c
  - Additional examples are provide on blackboard under Topic 8.

Summary

- Key learning topics:
  - Basic working principle of the LCD
  - cab202_teensy library
- Example programs