

Displays

Networks and Embedded Software

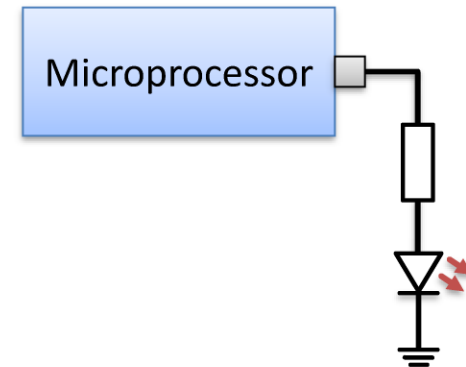
Module 4.3.8

by Wolfgang Neff

Indicator Lamps (1)

- Active High Lamps

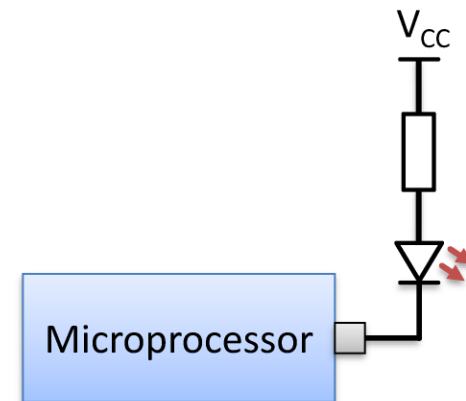
```
- void led_init(void)
- {
-     LED_PORT.OUT  &= ~LED_PIN_bm; // Turn LED off
-     LED_PORT.DIR |= LED_PIN_bm;  // LED pin is output
- }
- void led_on(void)
- {
-     LED_PORT.OUT |= LED_PIN_bm;
- }
- void led_off(void)
- {
-     LED_PORT.OUT  &= ~LED_PIN_bm;
- }
```



Indicator Lamps (2)

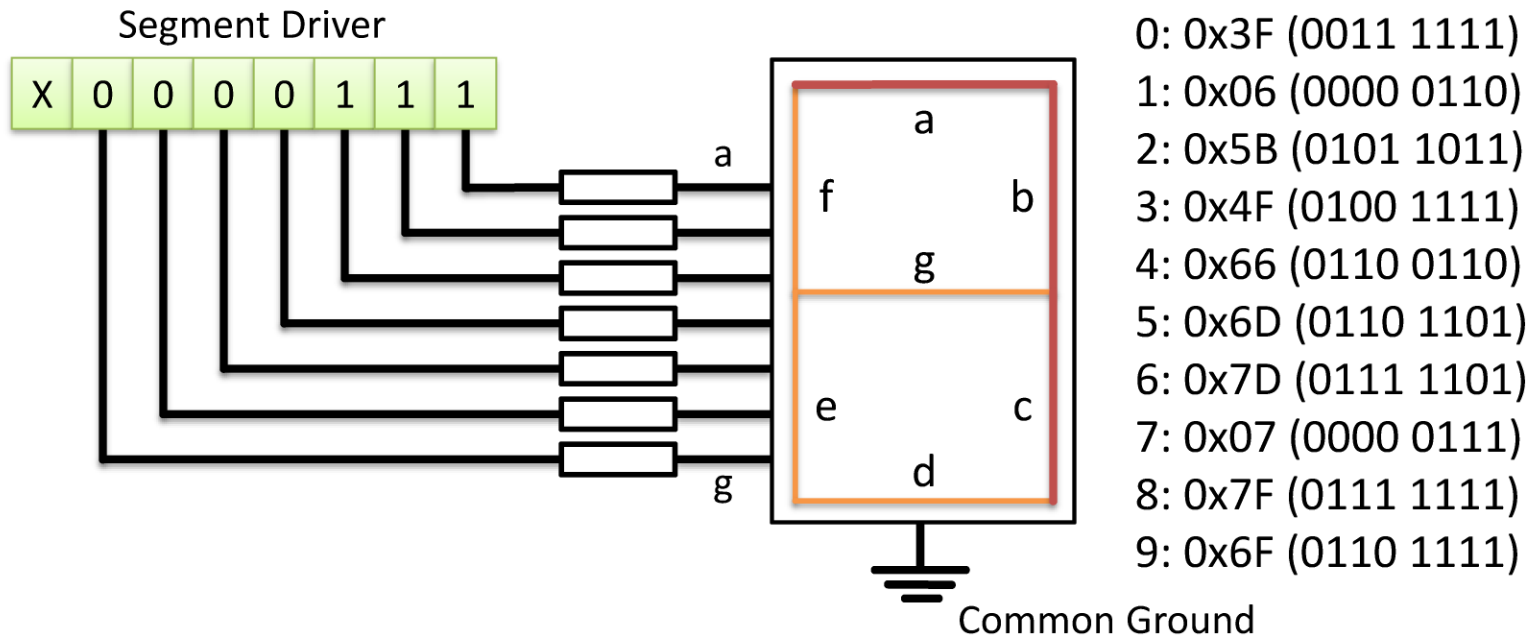
- Active Low Lamps

```
- void led_init(void)
- {
-     LED_PORT.OUT |= LED_PIN_bm; // Turn LED off
-     LED_PORT.DIR |= LED_PIN_bm; // LED pin is output
- }
- void led_on(void)
- {
-     LED_PORT.OUT &|= ~ LED_PIN_bm;
- }
- void led_off(void)
- {
-     LED_PORT.OUT |= LED_PIN_bm;
- }
```



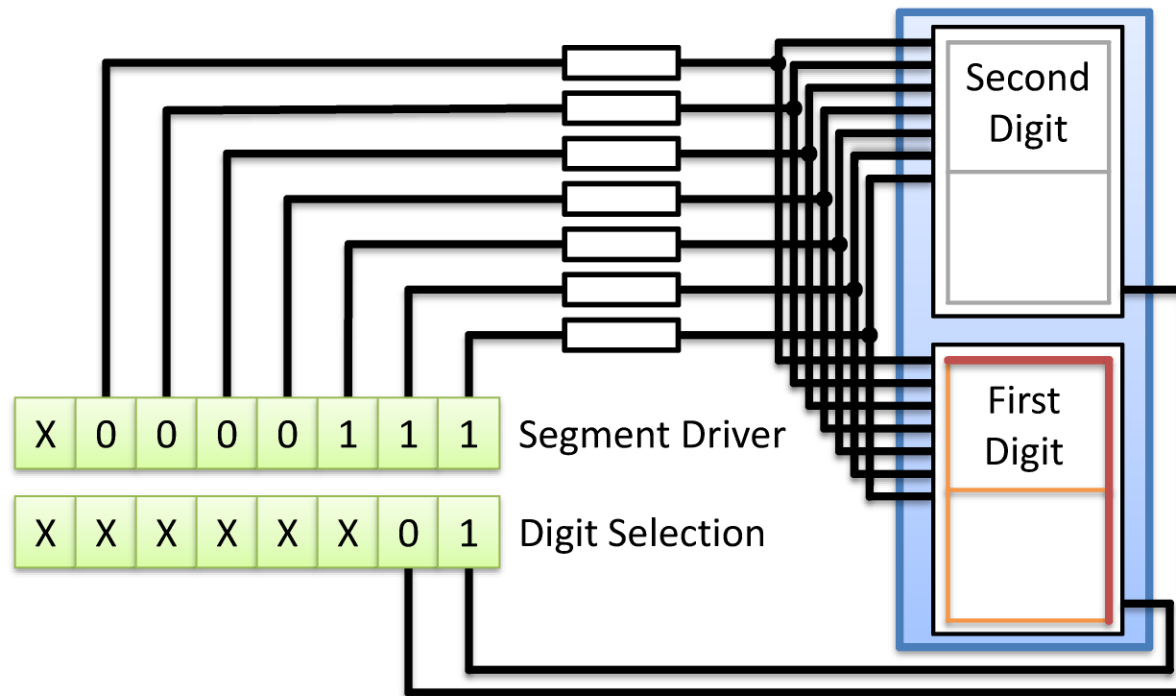
Seven-Segment Displays (1)

- Single Digit Display (Common Cathode)



Seven-Segment Displays (2)

- Multi Digit Display (Common Cathode)



Seven-Segment Displays (3)

- Programming Example (multi digit display)

```
- #define DIGITS_gm 0x03
- #define DIG1_PIN_bm 0x01
- #define DIG2_PIN_bm 0x02
- uint8_t seg[10] = { 0x3F, 0x06, ...};
- void display_off(void)
- {
-     DIG_PORT.OUT |= DIGITS_gm; // Turn active low digits off
- }
- void display_show_number(uint8_t n)
- {
-     SEG_PORT.OUT = seg[n];     // Write segment pattern
- }
```

Seven-Segment Displays (4)

- Programming Example (continued)

```
- void display_select_new_digit(uint8_t n)
- {
-     // Display must be turned off before
-     switch (n) {
-         case 1:
-             DIG_PORT.OUT &= ~DIG1_PIN_bm;
-             break;
-         case 2:
-             DIG_PORT.OUT &= ~DIG2_PIN_bm;
-             break;
-     }
- }
```

Seven-Segment Displays (5)

- Programming Example (finished)

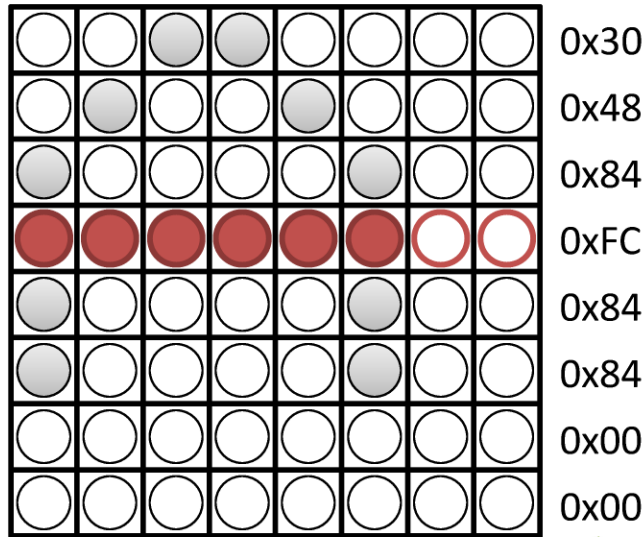
```
- uint8_t digits[2] = { 2, 1 };  
- void display_drive(void)  
- {  
-     static n = 0;  
-     display_off();  
-     display_show_number(digit[n]);  
-     display_select_new_digit(n);  
-     n = n+1;  
-     if (n==2) n = 0;  
- }
```


Dot-Matrix Displays (1)

- Operation Mode (row anode) • • •

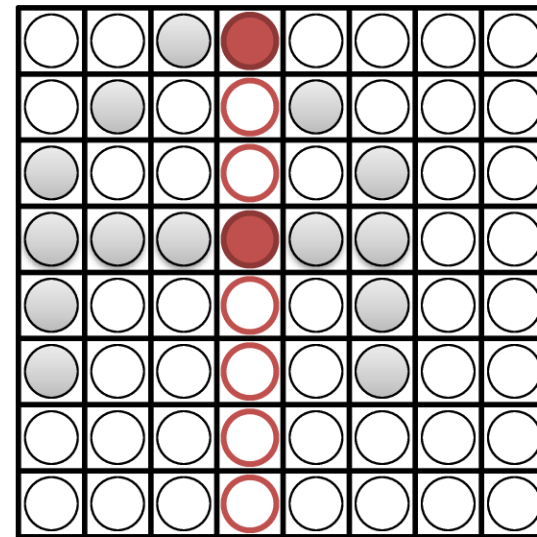
Rows are Active High
Cols are Active Low

Row by Row



Row Patterns

Column by Column

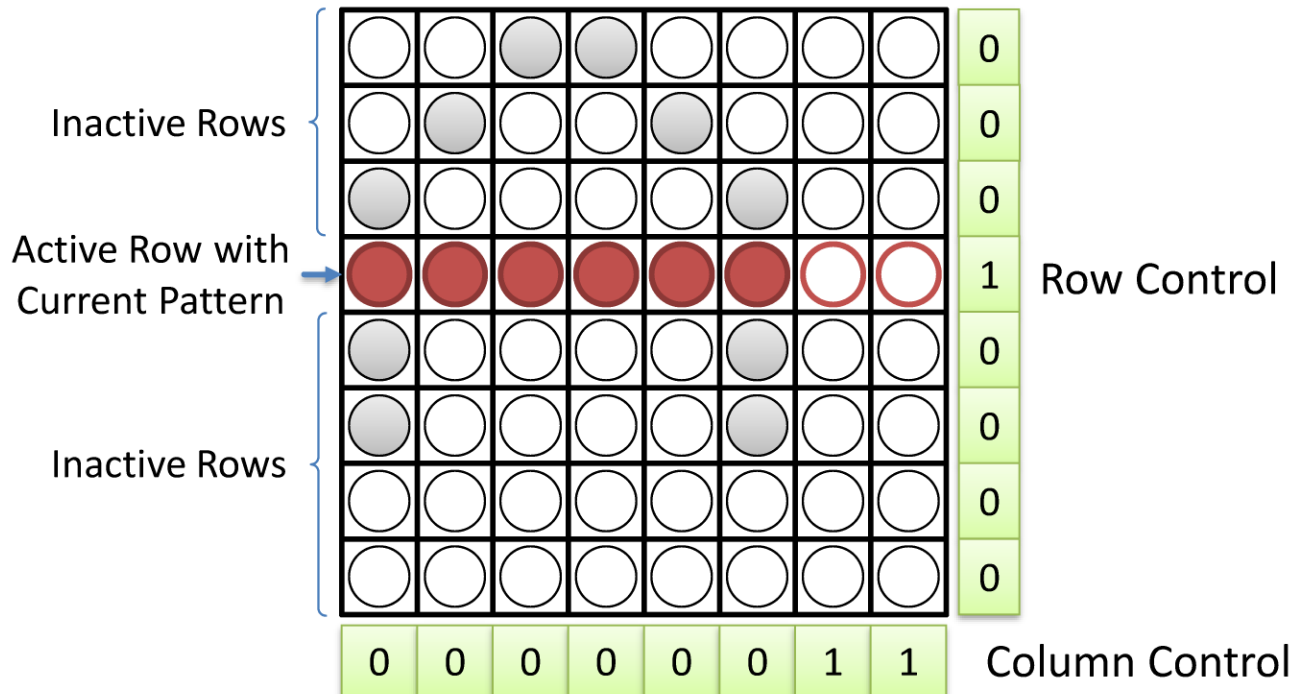


3C 50 90 90 50 3C 00 00

Column Patterns

Dot-Matrix Displays (2)

- Physical Setup (row anode, row by row)



Dot-Matrix Displays (3)

- Programming Example (row anode, row by row)


```
- #define COLS 8
- #define ROWS 8
- uint8_t row[ROWS] = { 0x30, 0x48, ...}; // Row patterns
- void display_rows_off(void)
- {
-     ROW_PORT.OUT = 0x00; // Turn all rows off
- }
- void display_new_pattern(uint8_t n)
- {
-     COL_PORT.OUT = ~row[n]; // Display pattern active low
- }
```

Dot-Matrix Displays (4)

- Programming Example (finished)

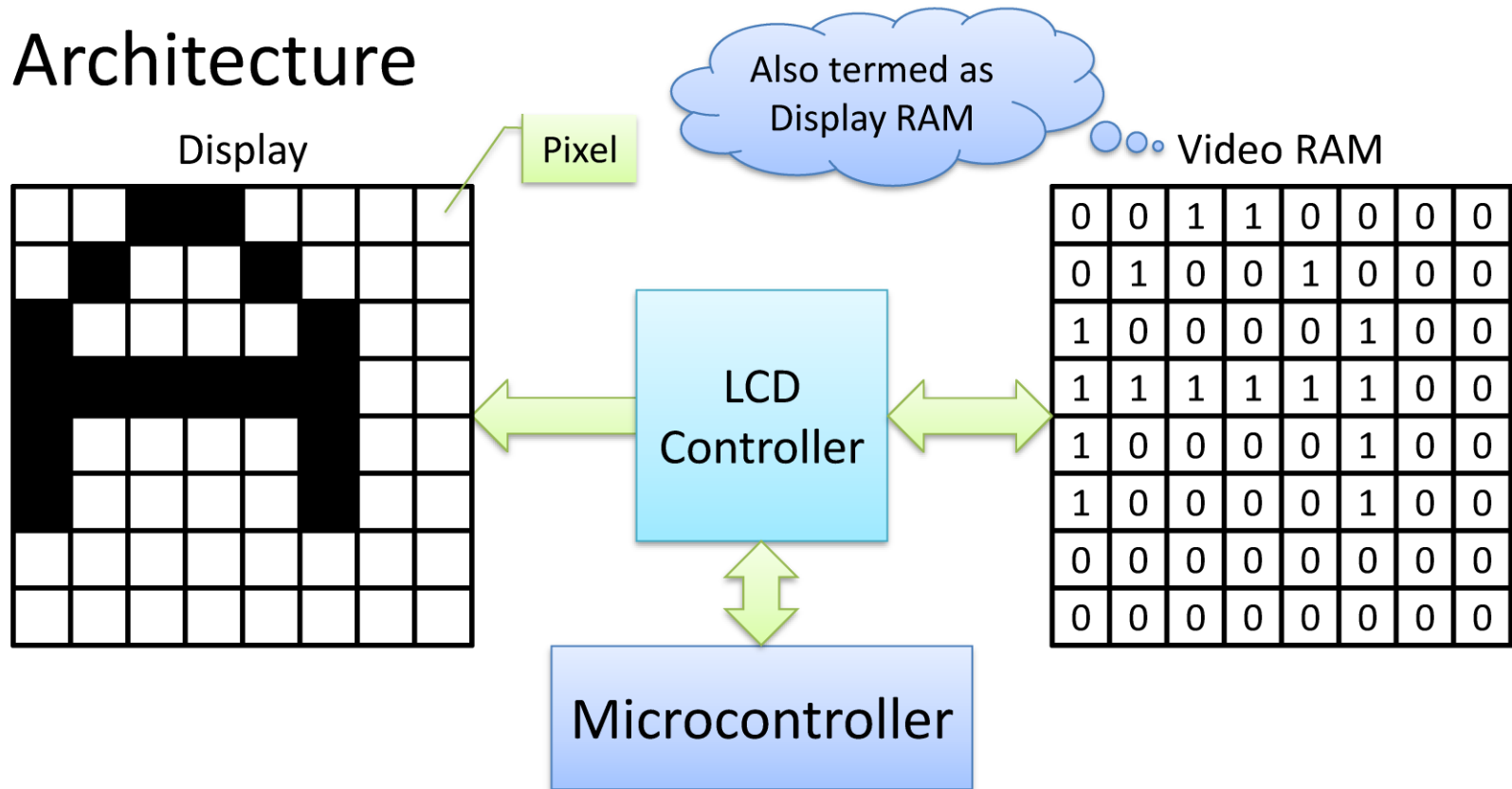
```
- void display_select_row(uint8_t n)
- {
-     ROW_PORT.OUT = (1 << n); // Select n-th row
- }
- void display_drive(void)
- {
-     static n = 0;
-     display_rows_off();
-     display_new_pattern(n);
-     display_select_row(n);
-     n = n+1;
-     if (n==ROWS) n = 0;
- }
```

Dot-Matrix Displays (5)

- Pulse Mode
 - Duty cycle: 1/8
 - Series resistor
 - $I_F = 20 \text{ mA} \rightarrow I_{\text{Pulse}} = 160 \text{ mA}$ (check data sheet for limits)
 - $U_F = 5 \text{ V} \rightarrow R = 5 \text{ V} / 160 \text{ mA} = 33 \Omega$
 - Attention 
 - Series resistor is for pulse mode operation
 - Continuous operation will destroy the display
 - Series resistor is only 33Ω instead of 270Ω

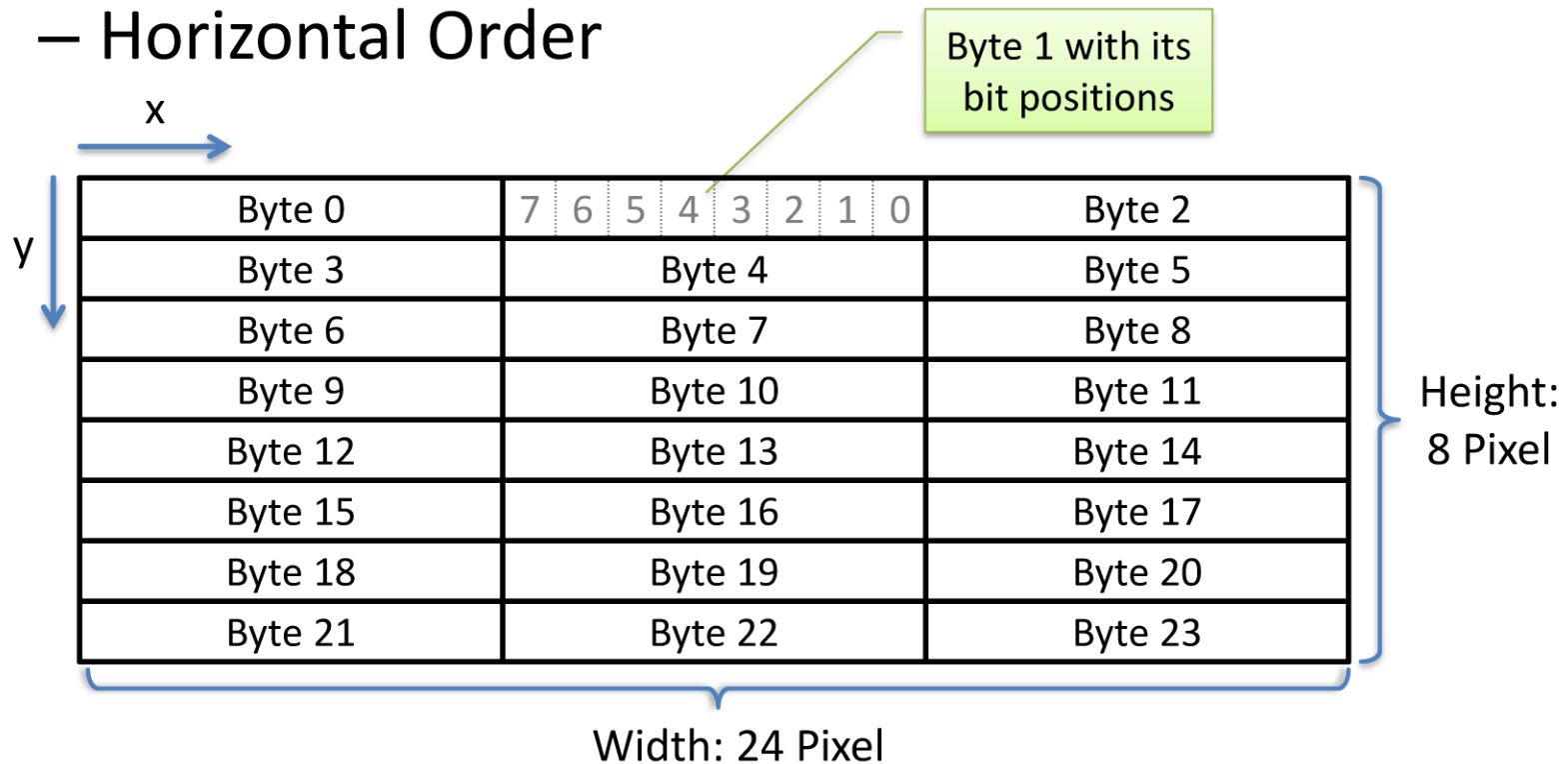
Monochrome Display (1)

- Architecture



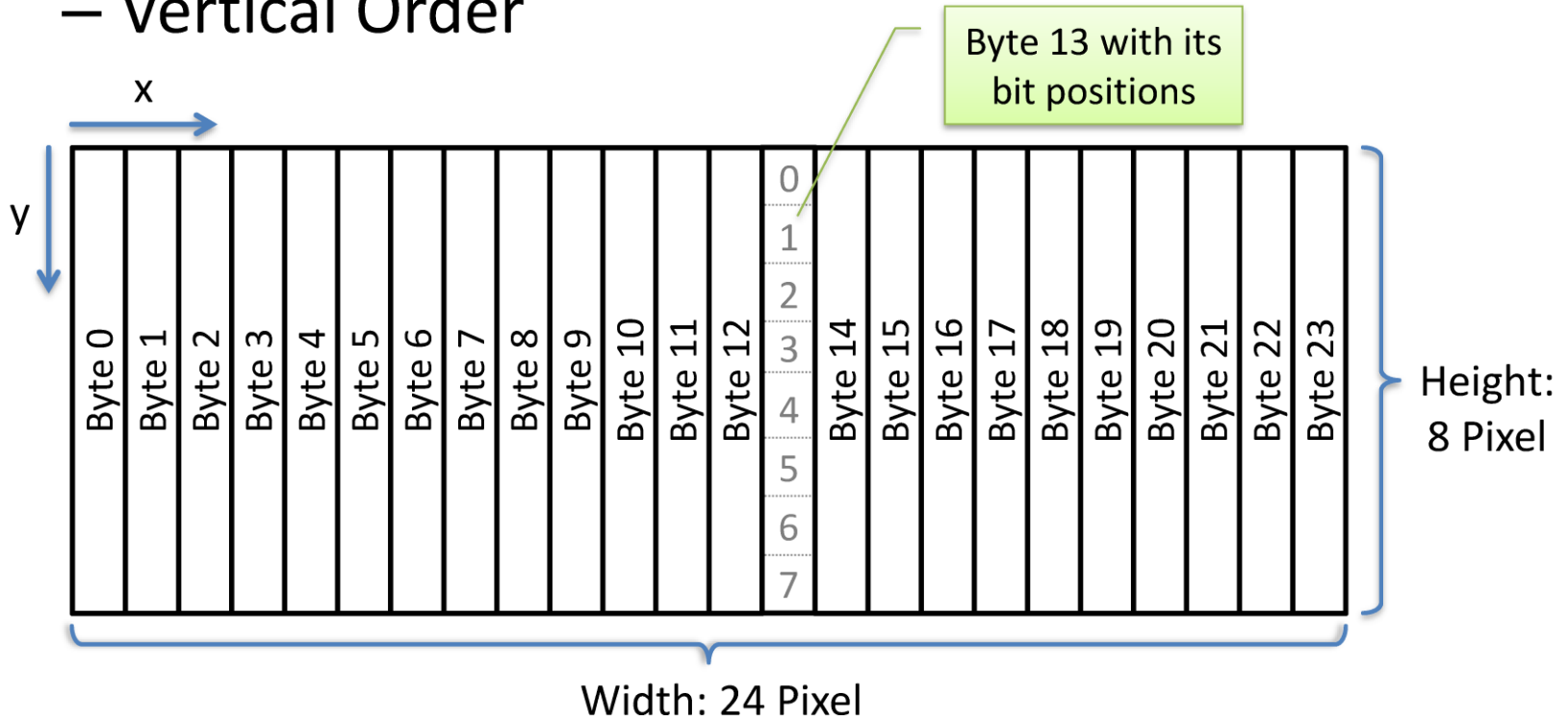
Monochrome Display (2)

- Byte Orientation
 - Horizontal Order



Monochrome Display (3)

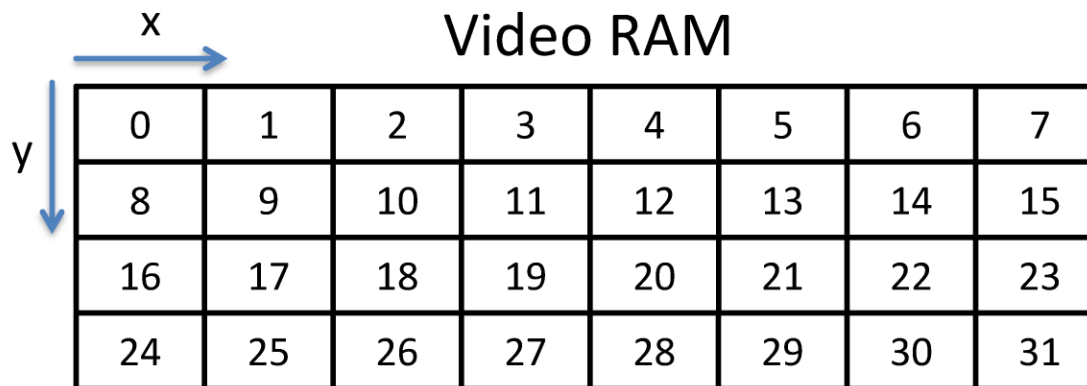
- Byte Orientation
 - Vertical Order



Monochrome Display (4)

- Addressing Modes

- Linear Horizontal Addressing (row-major)



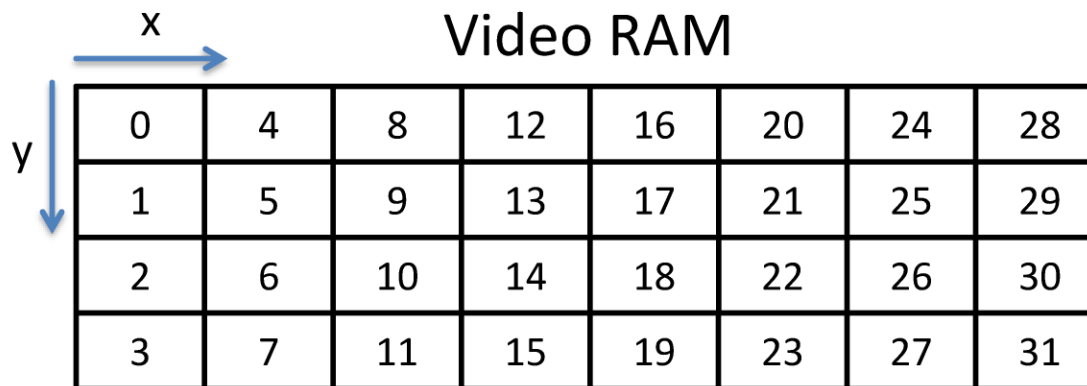
- `uint8_t pixel[HEIGHT*WIDTH]; // Video RAM`
- `pixel[WIDTH*y+x] = 1; // Access individual pixel`

Monochrome Display (5)

- Addressing Modes

- Linear Vertical Addressing (column-major)

Video RAM



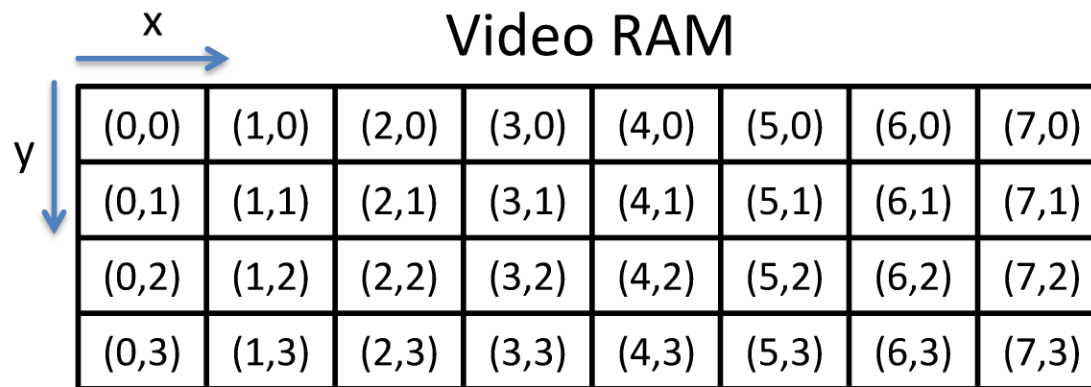
	x →							
y ↓	0	4	8	12	16	20	24	28
	1	5	9	13	17	21	25	29
	2	6	10	14	18	22	26	30
	3	7	11	15	19	23	27	31

- `uint8_t pixel[HEIGHT*WIDTH]; // Video RAM`
- `pixel[HEIGHT*x+y] = 1; // Access individual pixel`

Monochrome Display (6)

- Addressing Modes

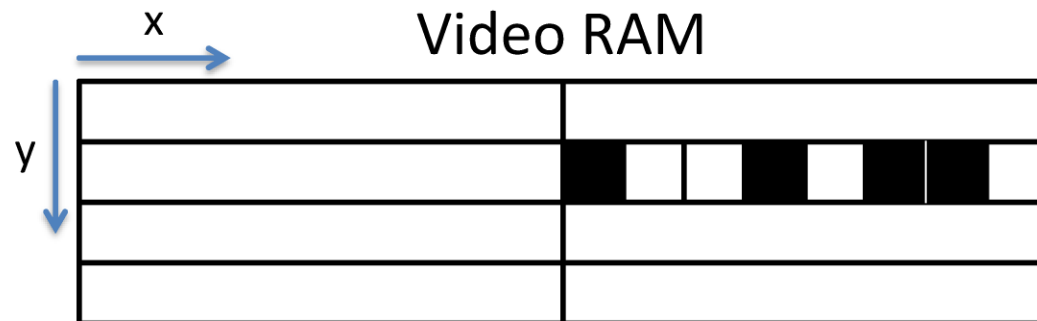
- Two-Dimensional Addressing



- `uint8_t pixel[HEIGHT][WIDTH];`
 - `pixel[y][x] = 1;`
- } If video RAM is row-major
- `uint8_t pixel[WIDTH][HEIGHT];`
 - `pixel[x][y] = 1;`
- } If video RAM is column-major

Monochrome Display (7)

- Programming Example
 - 4×16-bit display
 - Two-dimensional addressing mode
 - Horizontal byte orientation
 - Set pixel (1,8), (1,11), (1,13), (1,14).



Monochrome Display (8)

- Programming Example (continued)

```
- #define WITDH 16
- #define HEIGHT 4
- #define CMD_SET_XADDR 0x40 // Command to set x position
- #define CMD_SET_YADDR 0x80 // Command to set y position
- void display_show_byte(uint8_t x, uint8_t y, uint8_t byte)
- {
-     display_send_command(CMD_SET_XADDR | x);
-     display_send_command(CMD_SET_YADDR | y);
-     display_send_data(byte);
- }
- display_show_byte(1, 1, 0x96);
```

Usually in LCD Controller Header

In Display Module

In Main Program