EEL 4742 – Embedded Systems
Module 5 –Low Power Mode and LCD

Hadi M Kamali

Department of Electrical and Computer Engineering (**ECE**)
University of Central Florida

Office Location/phone: HEC435 – (407) 823-0764
webpage: https://www.ece.ucf.edu/~kamali/
e-mail: kamali@ucf.edu

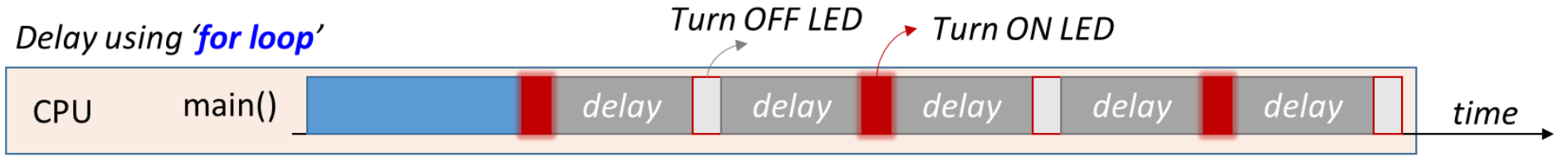
HAVEN Research Group
https://haven.ece.ucf.edu/

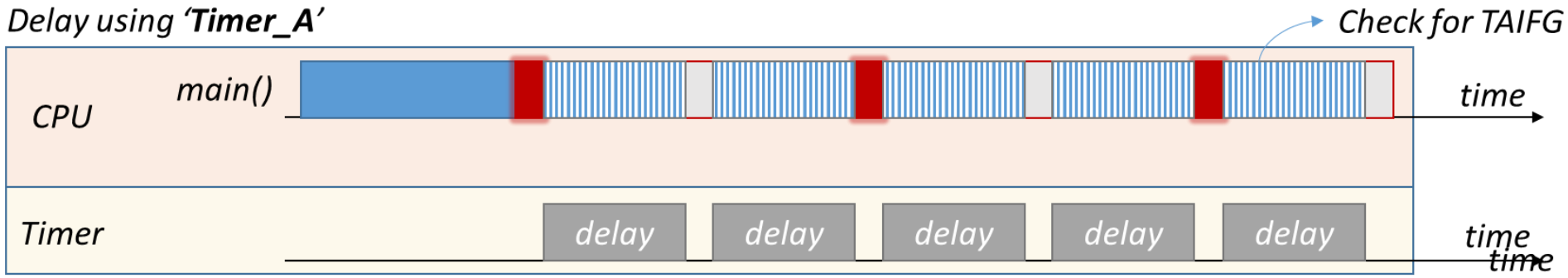
Partially adopted from previous UCF EEL4742C by Zakhia Abichar & Zhishan Guo

A Quick Recap: Why Low Power Mode

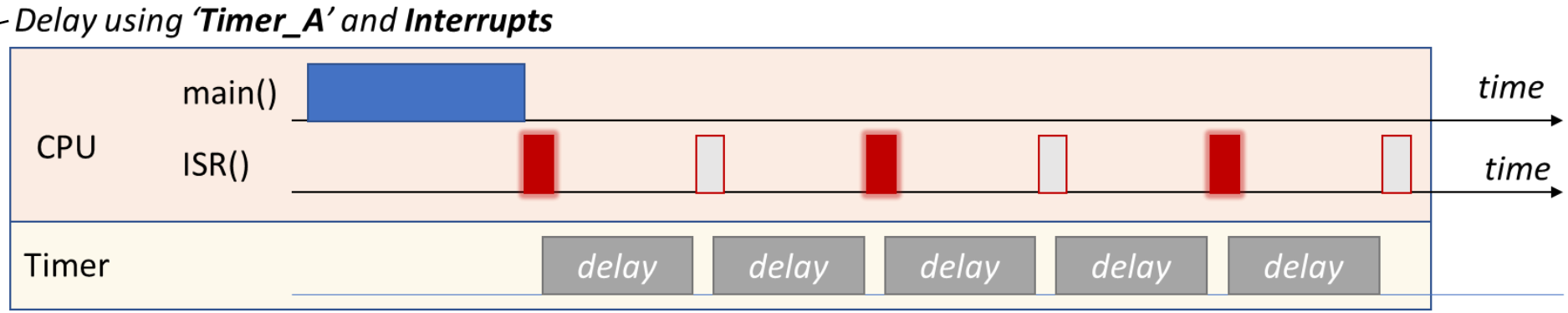
- **We need more automation, leaving the CPU less busy for other important operations!**

*The main() function is in a forever loop. At the hardware level, the MCU executes "no operation" **(NOP)** instructions.*





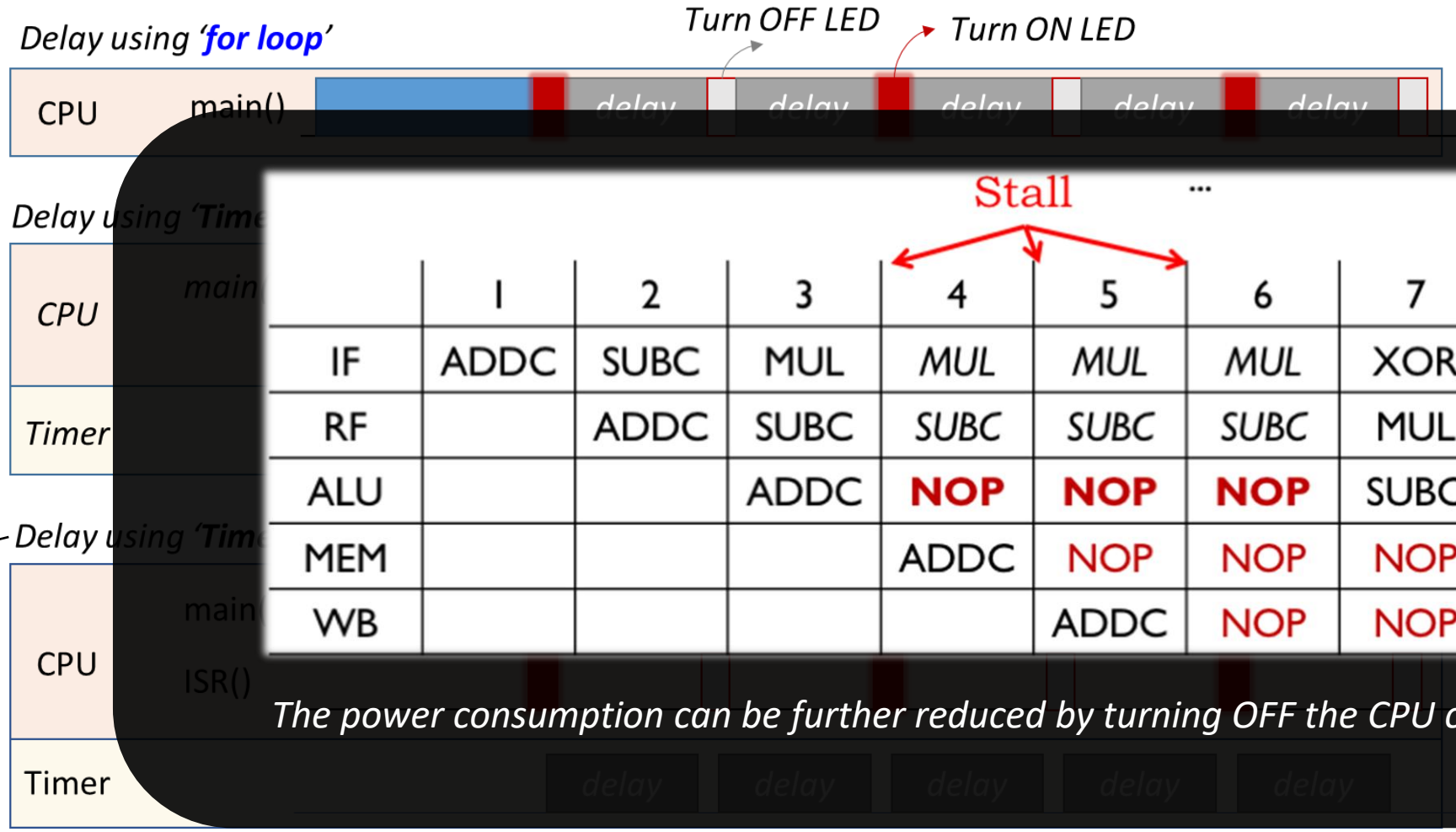
Ideal scenario keeping CPU less busy as we can!



A Quick Recap: Why Low Power Mode

- **We need more automation, leaving the CPU less busy for other important operations!**

*The main() function is in a forever loop. At the hardware level, the MCU executes "no operation" **(NOP)** instructions.*



Stall

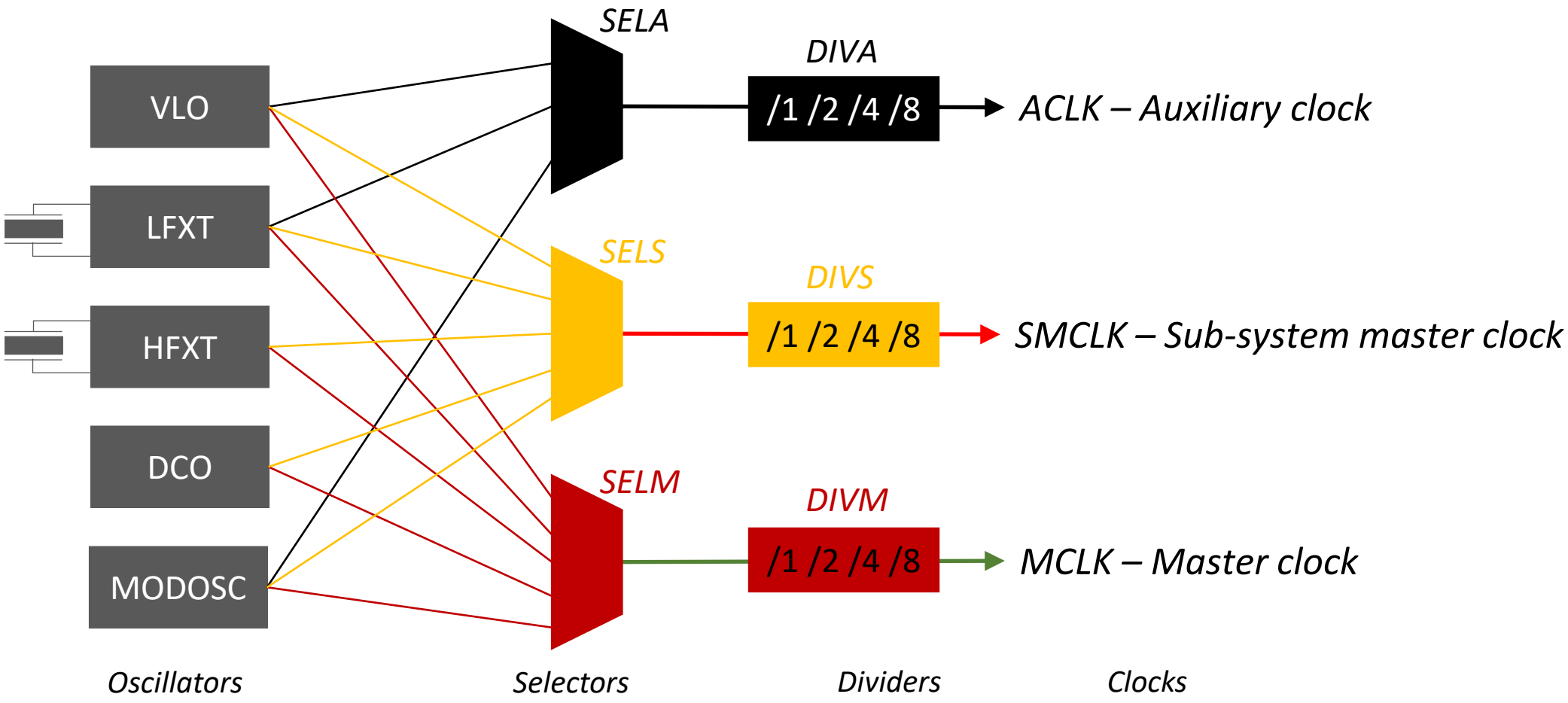
	1	2	3	4	5	6	7	8
IF	ADDC	SUBC	MUL	*MUL*	*MUL*	*MUL*	XOR	
RF		ADDC	SUBC	*SUBC*	*SUBC*	*SUBC*	MUL	XOR
ALU			ADDC	**NOP**	**NOP**	**NOP**	SUBC	MUL
MEM				ADDC	NOP	NOP	NOP	SUBC
WB					ADDC	NOP	NOP	NOP

The power consumption can be further reduced by turning OFF the CPU clock.

Ideal scenario keeping CPU less busy as we can!

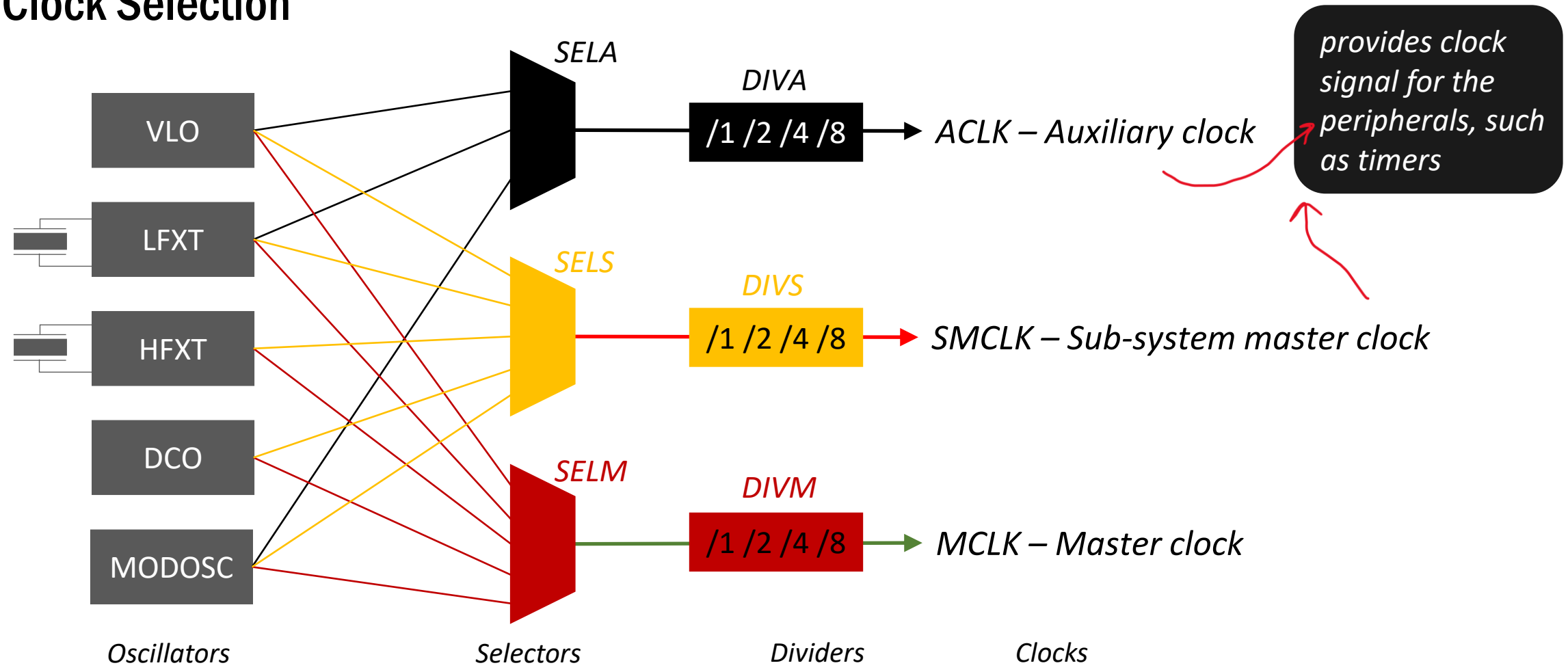
Clock Sources Running Continuously

- **Clock Selection**



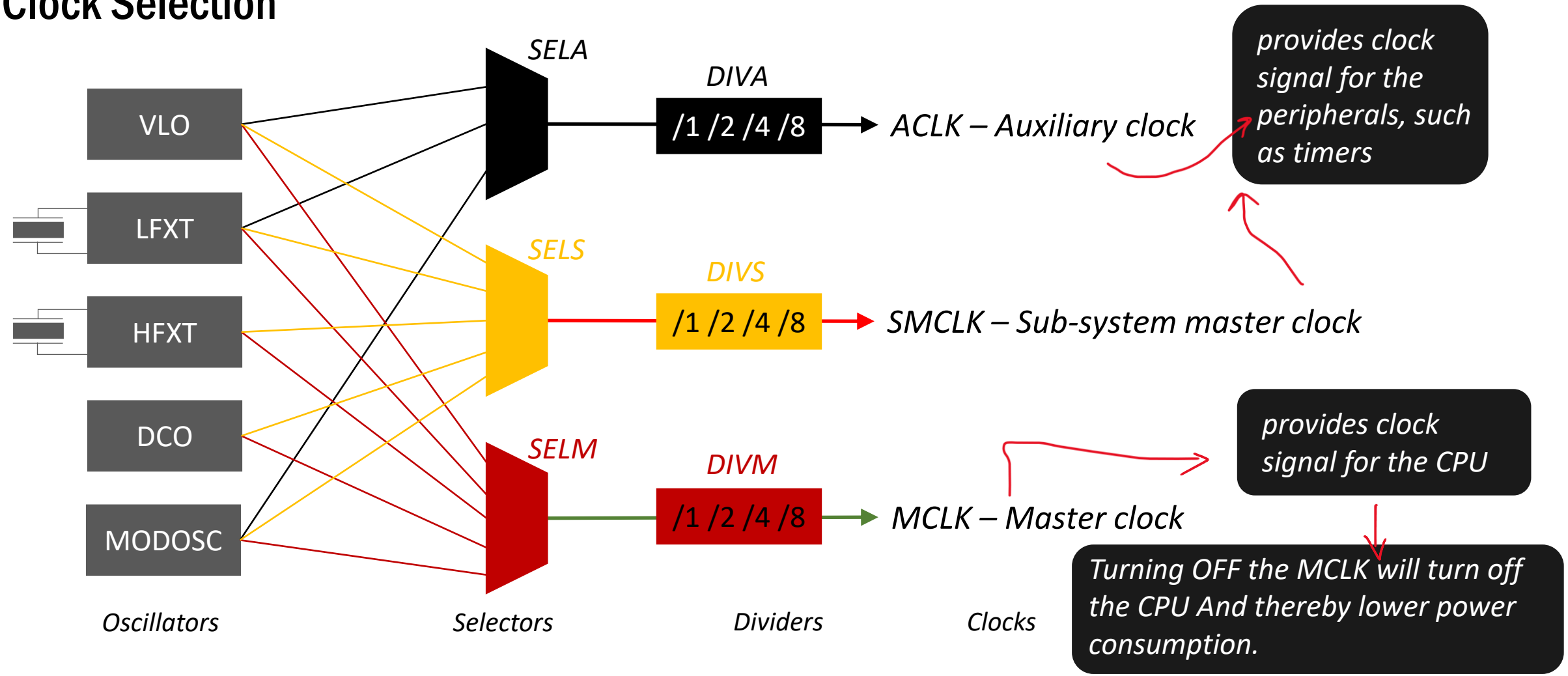
Clock Sources Running Continuously

- **Clock Selection**



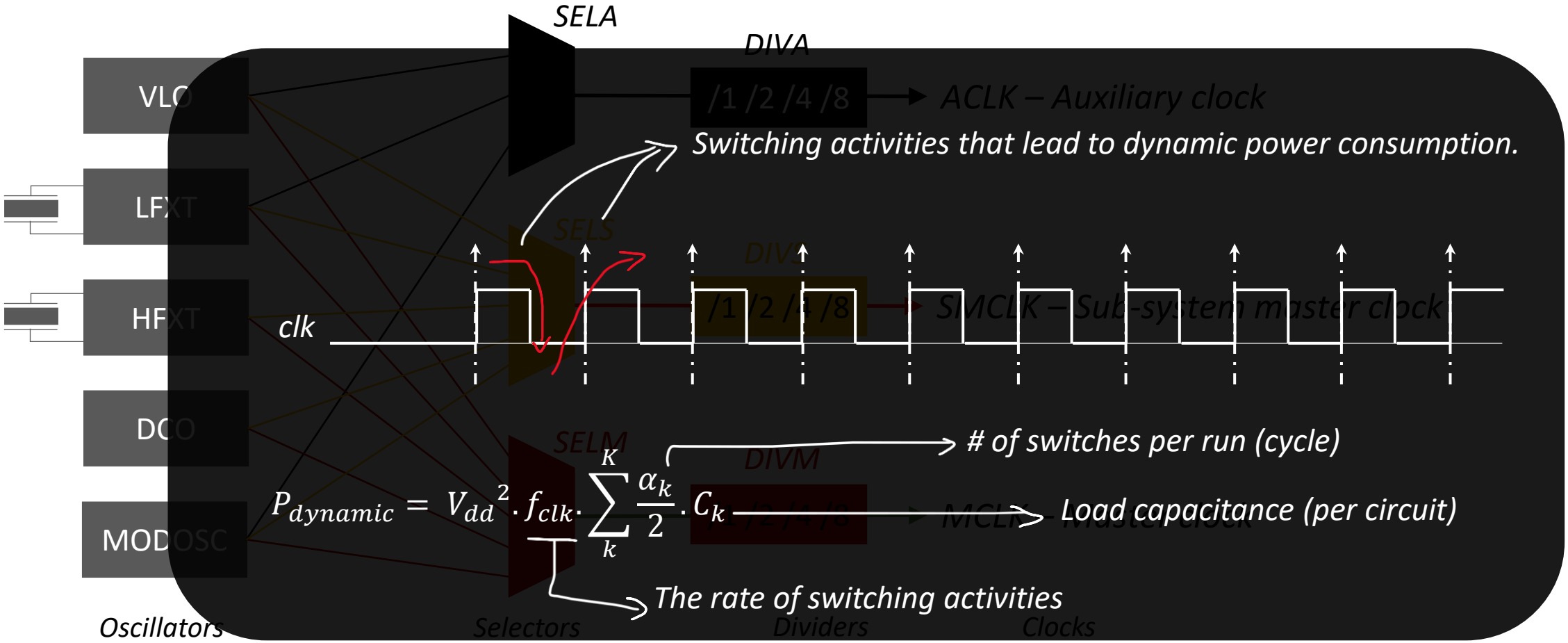
Clock Sources Running Continuously

- **Clock Selection**



Clock Sources Running Continuously

- **Clock Selection**



Low Power Modes in MSP430

- Commonly used LPMs in MSP430

Mode	MCLK	SMCLK	ACLK	SCG1	SCG0	OSCOFF	CPUOFF
Active	*ON*	*ON*	*ON*	*0*	*0*	*0*	*0*
LPM0	*OFF*	*ON*	*ON*	*0*	*0*	*0*	*1*
LPM3	*OFF*	*OFF*	*ON*	*1*	*1*	*0*	*1*
LPM4	*OFF*	*OFF*	*OFF*	*1*	*1*	*1*	*1*

Low Power Modes in MSP430

- **Commonly used LPMs in MSP430**

Mode	MCLK	SMCLK	ACLK	SCG1	SCG0	OSCOFF	CPUOFF
Active	*ON*	*ON*	*ON*	*0*	*0*	*0*	*0*
LPM0	*OFF*	*ON*	*ON*	*0*	*0*	*0*	*1*
LPM3	*OFF*	*OFF*	*ON*	*1*	*1*	*0*	*1*
LPM4	*OFF*	*OFF*	*OFF*	*1*	*1*	*1*	*1*

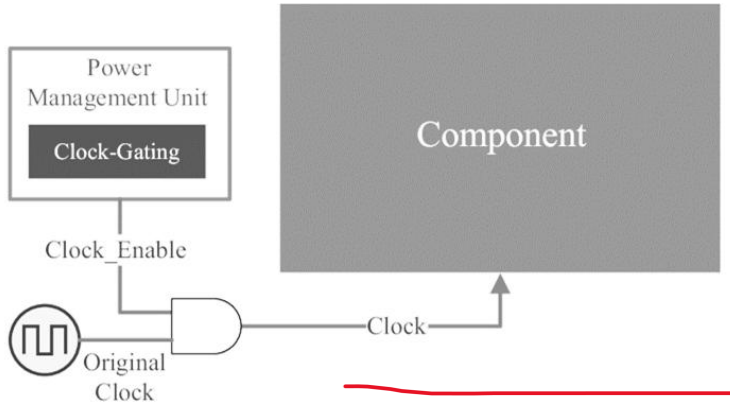
Configuration bits defined for controlling the clock sources (using power gating)

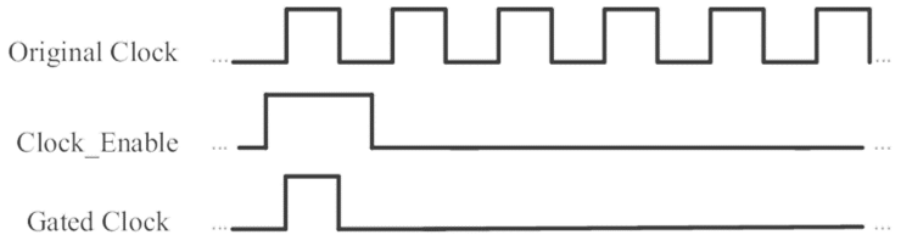
Low Power Modes in MSP430

- **Commonly used LPMs in MSP430**

Mode	MCLK	SMCLK	ACLK	SCG1	SCG0	OSCOFF	CPUOFF
Active	*ON*	*ON*	*ON*	*0*	*0*	*0*	*0*
LPM0	*OFF*	*ON*	*ON*	*0*	*0*	*0*	*1*
LPM3	*OFF*	*OFF*	*ON*	*1*	*1*	*0*	*1*
LPM4	*OFF*	*OFF*	*OFF*	*1*	*1*	*1*	*1*

Configuration bits defined for controlling the clock sources (using power gating)





Low Power Modes in MSP430

- **Commonly used LPMs in MSP430**

Mode	MCLK	SMCLK	ACLK	SCG1	SCG0	OSCOFF	CPUOFF
Active	*ON*	*ON*	*ON*	*0*	*0*	*0*	*0*
LPM0	*OFF*	*ON*	*ON*	*0*	*0*	*0*	*1*
LPM3	*OFF*	*OFF*	*ON*	*1*	*1*	*0*	*1*
LPM4	*OFF*	*OFF*	*OFF*	*1*	*1*	*1*	*1*

SR (R2)

rsvd.							V	SCG1	SCG0	OSCOFF	CPUOFF	GIE	N	Z	C
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Modes: LPM0, LPM1, LPM2, LPM3.5, LPM4.5 → Refer to the family guide for each!

Low Power Modes in MSP430

- Commonly used LPMs in MSP430

(Q) How can we set these registers?

Mode	MCLK	SMCLK	ACLK	SCG1	SCG0	OSCOFF	CPUOFF
Active	ON	ON	ON	0	0	0	0
LPM0	OFF	ON	ON	0	0	0	1
LPM3	OFF	OFF	ON	1	1	0	1
LPM4	OFF	OFF	OFF	1	1	1	1

SR (R2)

rsvd.							V	SCG1	SCG0	OSCOFF	CPUOFF	GIE	N	Z	C
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Modes: LPM0, LPM1, LPM2, LPM3.5, LPM4.5 → Refer to the family guide for each!

Low Power Modes in MSP430

- **Commonly used LPMs in MSP430**

Mode	MCLK	SMCLK	ACLK	SCG1	SCG0	OSCOFF	CPUOFF
Active	ON	ON	ON	0	0	0	0
LPM0	OFF	ON	ON	0	0	0	1
LPM3	OFF	OFF	ON	1	1	0	1
LPM4	OFF	OFF	OFF	1	1	1	1

(Q) How can we set these registers?

(A) Using Intrinsic function. Like _enable_interrupt() for GIE.

SR (R2)

rsvd.							V	SCG1	SCG0	OSCOFF	CPUOFF	GIE	N	Z	C
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Modes: LPM0, LPM1, LPM2, LPM3, LPM4 → Refer to the family guide for each!

Low Power Modes in MSP430

- **Commonly used LPMs in MSP430**

Mode	MCLK	SMCLK	ACLK	SCG1	SCG0	OSCOFF	CPUOFF
Active	ON	ON	ON	0	0	0	0
LPM0	OFF	ON	ON	0	0	0	1
LPM3	OFF	OFF	ON	1	1	0	1
LPM4	OFF	OFF	OFF	1	1	1	1

(Q) How can we set these registers?

(A) Using Intrinsic function. Like _enable_interrupt() for GIE.

SR (R2)

rsvd.							V	SCG1	SCG0	OSCOFF	CPUOFF	GIE	N	Z	C
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Modes: LPM0, LPM1, LPM2, LPM3, LPM4 → Refer to the family guide for each!
For Enabling Mode X of LPM:

`_low_power_modes_x();` *e.g., LPM0* → `_low_power_modes_0();`

External Interrupts with LPM

- Toggle Red LED when Push button 1 is pressed, toggle Green LED when Push button 2 is pressed. Reduce power consumption while idle using low power modes

```
// include header files
void main(void)
{
        // initial watchdog
        // Configure peripherals (LED), push buttons, timer, and interrupts

        _low_power_modes_0();      // LPM0 – Not needed to set GIE
        for(;;) {}                 // Infinite loop
}
```

```
#pragma vector = PORT1_VECTOR          // preprocessor directive??
__interrupt void Port1_ISR(void) {     // This ISR can be triggered by multiple events
        if (push button 1?) {          // P1IFG.1??
        // Clear pin0 flag
        // Toggle red LED

        if (push button 2?) {          // P1IFG.2??
        // Clear pin1 flag
        // Toggle green LED

        …
}
```

External Interrupts with LPM

- Toggle Red LED when Push button 1 is pressed, toggle Green LED when Push button 2 is pressed. Reduce power consumption while idle using low power modes

```
// include header files
void main(void)
{
        // initial watchdog
        // Configure peripherals (LED), push buttons, timer, and interrupts

        _low_power_modes_0();      // LPM0 – Not needed to set GIE
        for(;;) {}                 // Infinite loop
}
```

```
#pragma vector = PORT1_VECTOR          // preprocessor directive??
__interrupt void Port1_ISR(void) {     // This ISR can be triggered by multiple events
        if (push button 1?) {          // P1IFG.1??
        // Clear pin0 flag
        // Toggle red LED

        if (push button 2?) {          // P1IFG.2??
        // Clear pin1 flag
        // Toggle green LED

        …
}
```

(Q) Why

External Interrupts with LPM

- **Toggle Red LED when Push button 1 is pressed, toggle Green LED when Push button 2 is pressed. Reduce power consumption while idle using low power modes**

```
// include header files
void main(void)
{
        // initial watchdog
        // Configure peripherals (LED), push buttons, timer, and interrupts

        _low_power_modes_0();      // LPM0 – Not needed to set GIE
        for(;;) {}                 // Infinite loop
}
```

```
#pragma vector = PORT1_VECTOR          // preprocessor directive??
__interrupt void Port1_ISR(void) {     // This ISR can be triggered by multiple events
        if (push button 1?) {          // P1IFG.1??
        // Clear pin0 flag
        // Toggle red LED

        if (push button 2?) {          // P1IFG.2??
        // Clear pin1 flag
        // Toggle green LED

        …
}
```

(Q) Why

*(A) Interrupts are the only way to "**wake up**" the CPU/MCLK.*
Therefore, the interrupts are automatically enabled when entering low power modes. It is not necessary to set GIE, if low power modes are used.

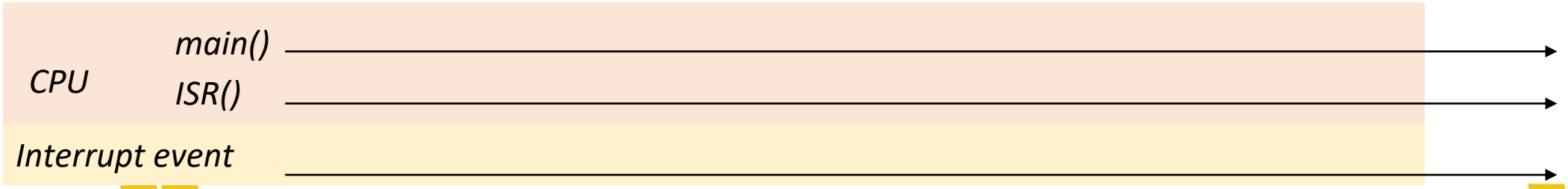
When the timer register TA0R counts from 0xFFFF to 0x0000, TAIFG flag is set.

External Interrupts with LPM

- Toggle Red LED when Push button 1 is pressed, toggle Green LED when Push button 2 is pressed. Reduce power consumption while idle using low power modes

```
// include header files
void main(void)
{
        // initial watchdog
        // Configure peripherals (LED), push buttons, timer, and interrupts

        _low_power_modes_0();      // LPM0 – Not needed to set GIE
        for(;;) {}                 // Infinite loop
}
```

```
#pragma vector = PORT1_VECTOR          // preprocessor directive??
__interrupt void Port1_ISR(void) {     // This ISR can be triggered by multiple events
        if (push button 1?) {          // P1IFG.1??
        // Clear pin0 flag
        // Toggle red LED

        if (push button 2?) {          // P1IFG.2??
        // Clear pin1 flag
        // Toggle green LED

        …
}
```

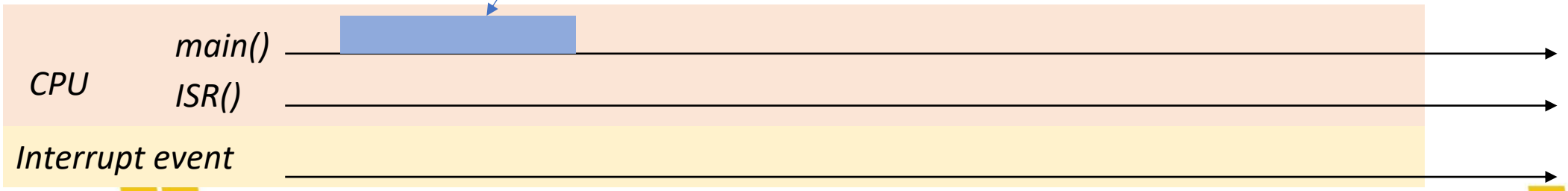


External Interrupts with LPM

- Toggle Red LED when Push button 1 is pressed, toggle Green LED when Push button 2 is pressed. Reduce power consumption while idle using low power modes

```
// include header files
void main(void)
{
    // initial watchdog
    // Configure peripherals (LED), push buttons, timer, and interrupts

    _low_power_modes_0();      // LPM0 – Not needed to set GIE
    for(;;) {}                 // Infinite loop
}
```

```
#pragma vector = PORT1_VECTOR          // preprocessor directive??
__interrupt void Port1_ISR(void) {     // This ISR can be triggered by multiple events
        if (push button 1?) {          // P1IFG.1??
        // Clear pin0 flag
        // Toggle red LED

        if (push button 2?) {          // P1IFG.2??
        // Clear pin1 flag
        // Toggle green LED

        …
}
```

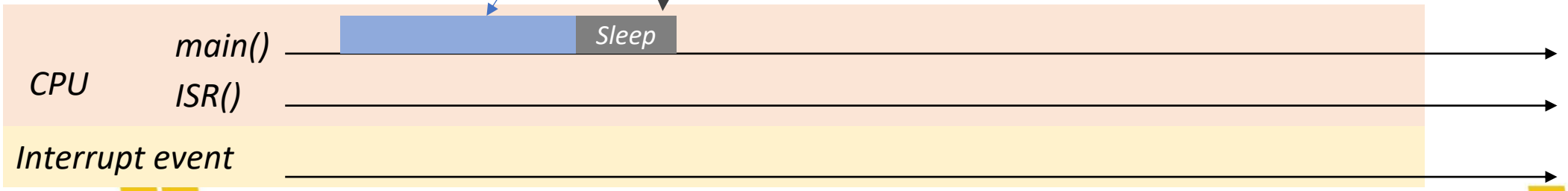


External Interrupts with LPM

- Toggle Red LED when Push button 1 is pressed, toggle Green LED when Push button 2 is pressed. Reduce power consumption while idle using low power modes

```
// include header files
void main(void)
{
    // initial watchdog
    // Configure peripherals (LED), push buttons, timer, and interrupts

    _low_power_modes_0();      // LPM0 – Not needed to set GIE
    for(;;) {}                 // Infinite loop
}
```

```
#pragma vector = PORT1_VECTOR          // preprocessor directive??
__interrupt void Port1_ISR(void) {     // This ISR can be triggered by multiple events
        if (push button 1?) {          // P1IFG.1??
        // Clear pin0 flag
        // Toggle red LED

        if (push button 2?) {          // P1IFG.2??
        // Clear pin1 flag
        // Toggle green LED

        …
}
```

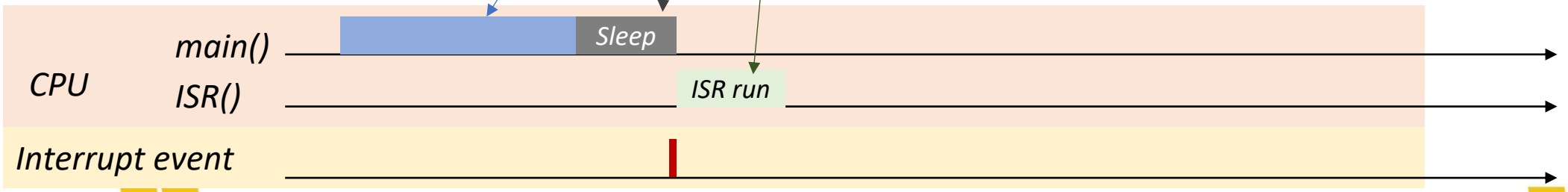


External Interrupts with LPM

- Toggle Red LED when Push button 1 is pressed, toggle Green LED when Push button 2 is pressed. Reduce power consumption while idle using low power modes

```
// include header files
void main(void)
{
    // initial watchdog
    // Configure peripherals (LED), push buttons, timer, and interrupts

    _low_power_modes_0();      // LPM0 – Not needed to set GIE
    for(;;) {}                 // Infinite loop
}
```

```
#pragma vector = PORT1_VECTOR          // preprocessor directive??
__interrupt void Port1_ISR(void) {     // This ISR can be triggered by multiple events
                if (push button 1?) {  // P1IFG.1??
                // Clear pin0 flag
                 // Toggle red LED

                 if (push button 2?) { // P1IFG.2??
                // Clear pin1 flag
                 // Toggle green LED

                ...
}
```

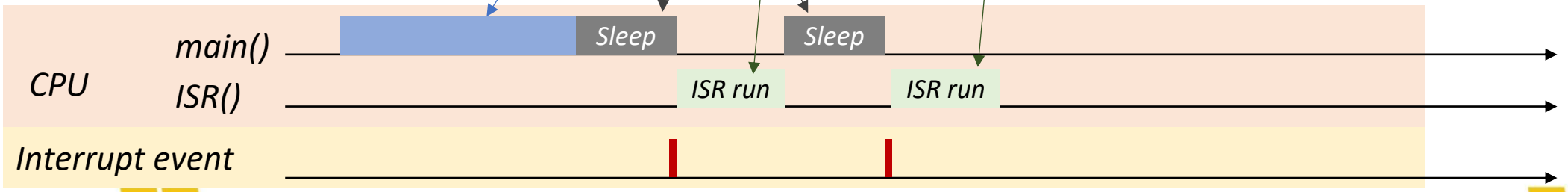


External Interrupts with LPM

- Toggle Red LED when Push button 1 is pressed, toggle Green LED when Push button 2 is pressed. Reduce power consumption while idle using low power modes

```
// include header files
void main(void)
{
    // initial watchdog
    // Configure peripherals (LED), push buttons, timer, and interrupts

    _low_power_modes_0();      // LPM0 – Not needed to set GIE
    for(;;) {}                 // Infinite loop
}
```

```
#pragma vector = PORT1_VECTOR          // preprocessor directive??
__interrupt void Port1_ISR(void) {     // This ISR can be triggered by multiple events
                if (push button 1?) {  // P1IFG.1??
                // Clear pin0 flag
                 // Toggle red LED

                 if (push button 2?) { // P1IFG.2??
                // Clear pin1 flag
                 // Toggle green LED

                ...
}
```

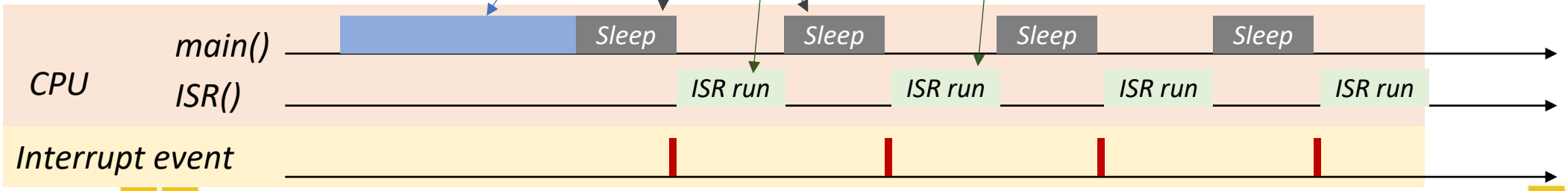


External Interrupts with LPM

- Toggle Red LED when Push button 1 is pressed, toggle Green LED when Push button 2 is pressed. Reduce power consumption while idle using low power modes

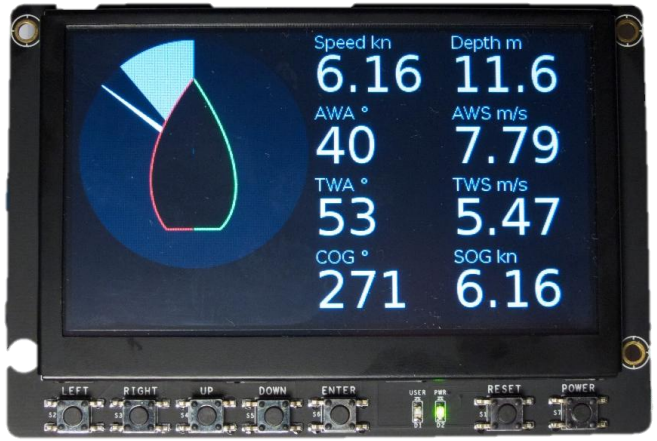
```
// include header files
void main(void)
{
    // initial watchdog
    // Configure peripherals (LED), push buttons, timer, and interrupts

    _low_power_modes_0();      // LPM0 – Not needed to set GIE
    for(;;) {}                 // Infinite loop
}
```

```
#pragma vector = PORT1_VECTOR          // preprocessor directive??
__interrupt void Port1_ISR(void) {     // This ISR can be triggered by multiple events
        if (push button 1?) {          // P1IFG.1??
        // Clear pin0 flag
        // Toggle red LED

        if (push button 2?) {          // P1IFG.2??
        // Clear pin1 flag
        // Toggle green LED

        …
}
```

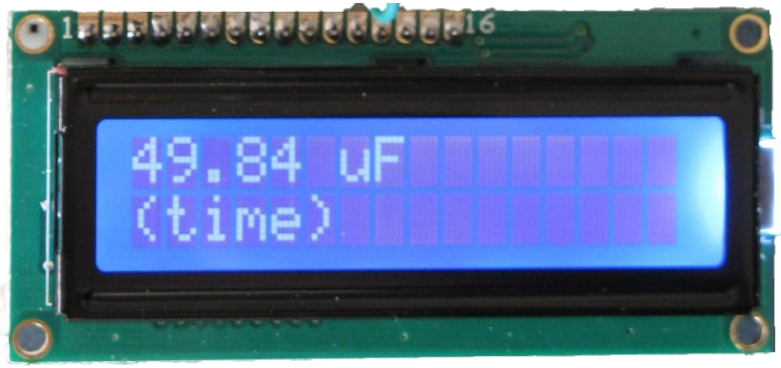


LCD – Liquid Crystal Display

- LCD is made of pixels and each pixel can be controlled independently

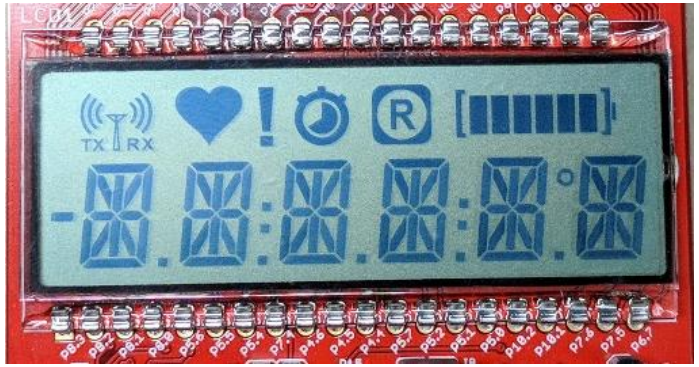


Graphic LCD



16x2 Character LCD

7-segment LCD

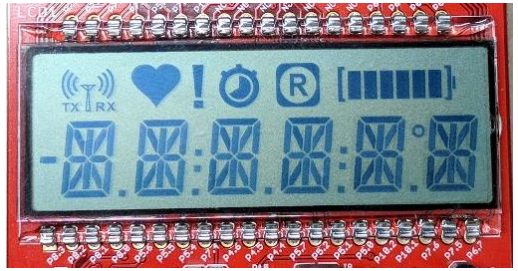


14-segment LCD

LCD – Liquid Crystal Display

- LCD is made of pixels and each pixel can be controlled independently
- For MSP430
  - LCD display consists of several individual segments that can be individually turned ON/OFF.

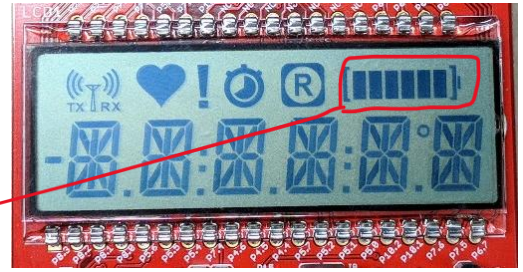
14-segment LCD



LCD – Liquid Crystal Display

- LCD is made of pixels and each pixel can be controlled independently
- For MSP430
  - LCD display consists of several individual segments that can be individually turned ON/OFF.

14-segment LCD



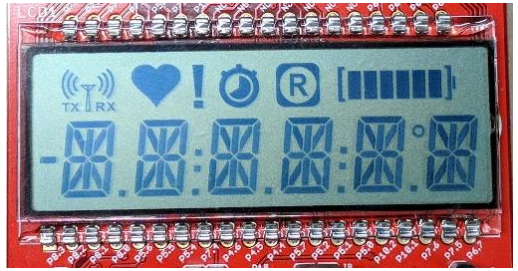
One segment of battery level

Several LCD segments combined to show battery level

LCD – Liquid Crystal Display

- LCD is made of pixels and each pixel can be controlled independently
- For MSP430
  - LCD display consists of several individual segments that can be individually turned ON/OFF.

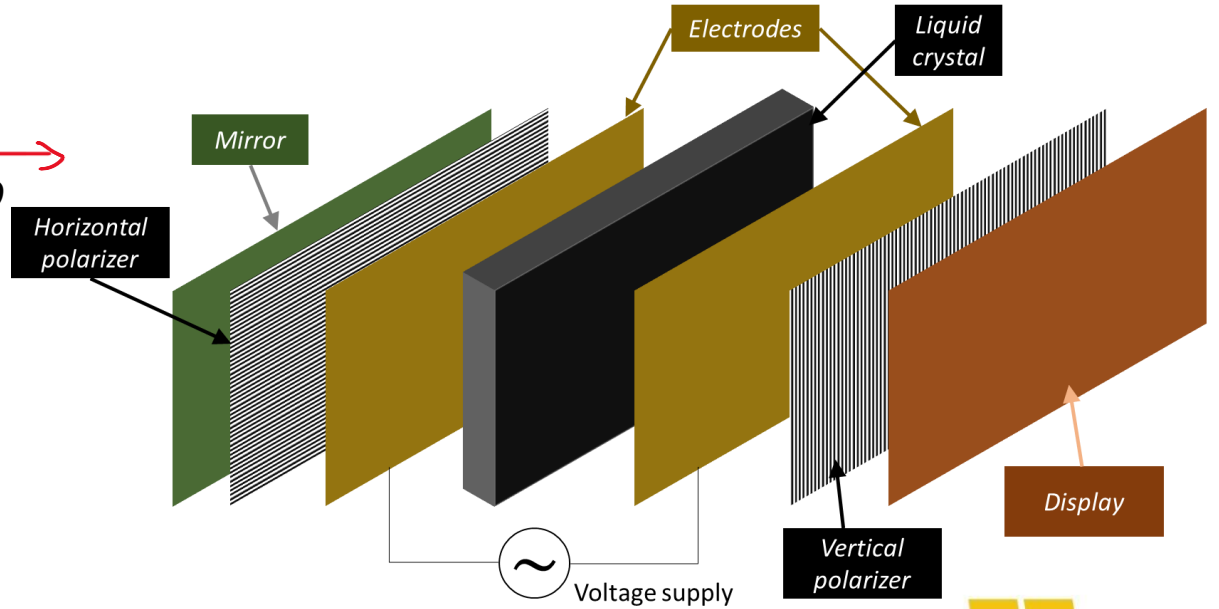
14-segment LCD



One segment of battery level

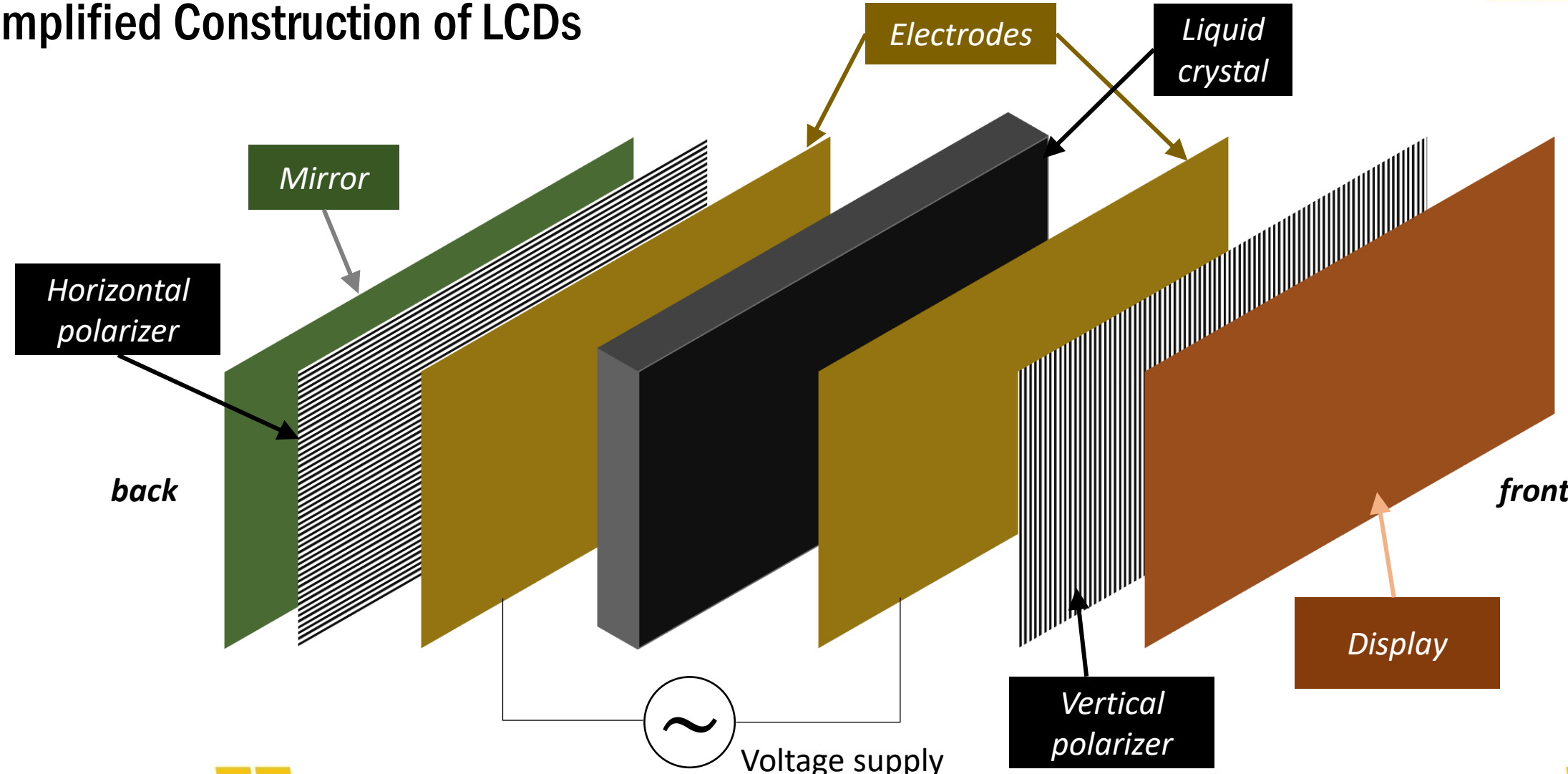
Several LCD segments combined to show battery level

Simplified version of layers in LCD



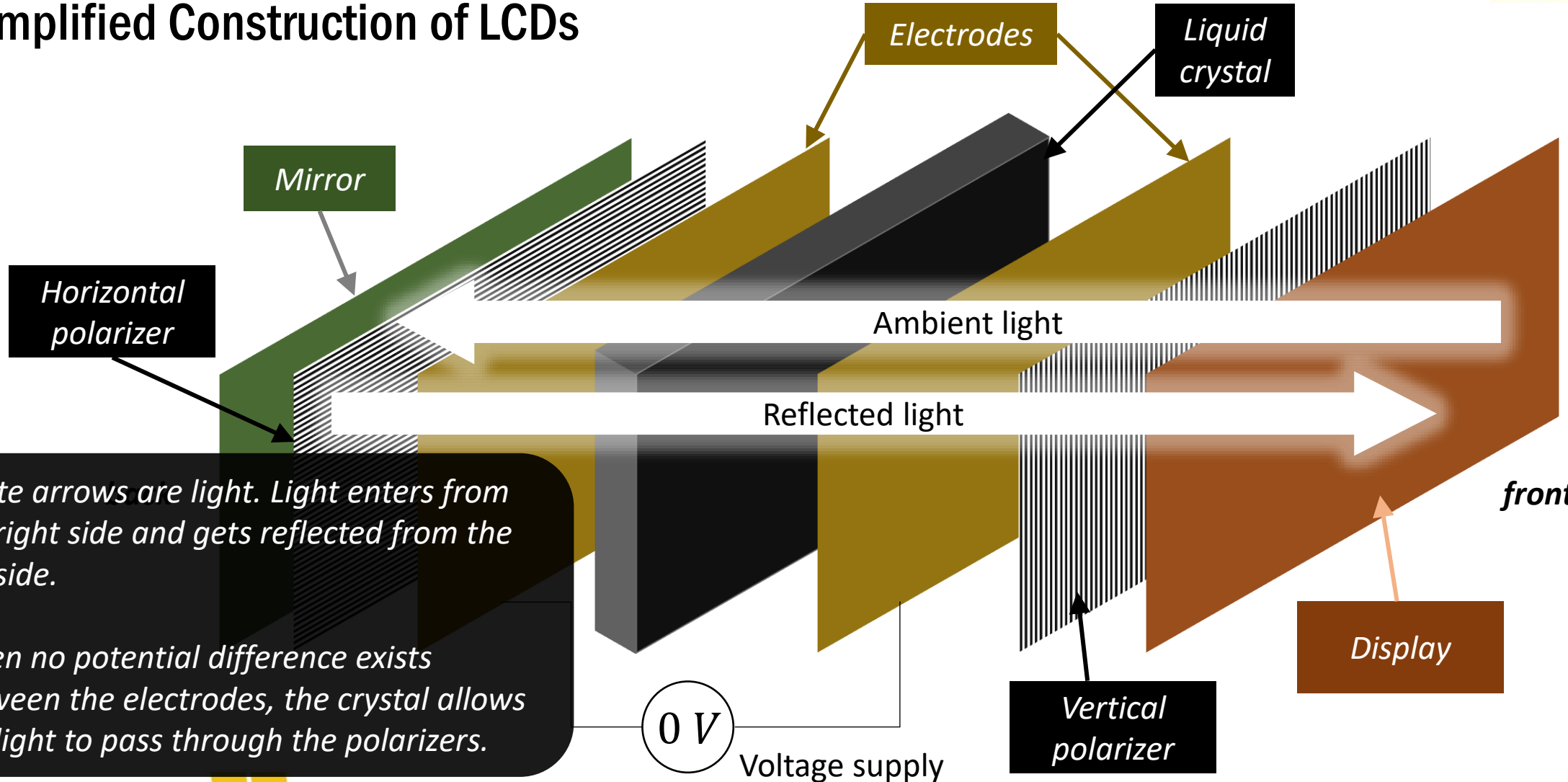
LCD – Liquid Crystal Display

- Simplified Construction of LCDs



LCD – Liquid Crystal Display

- **Simplified Construction of LCDs**

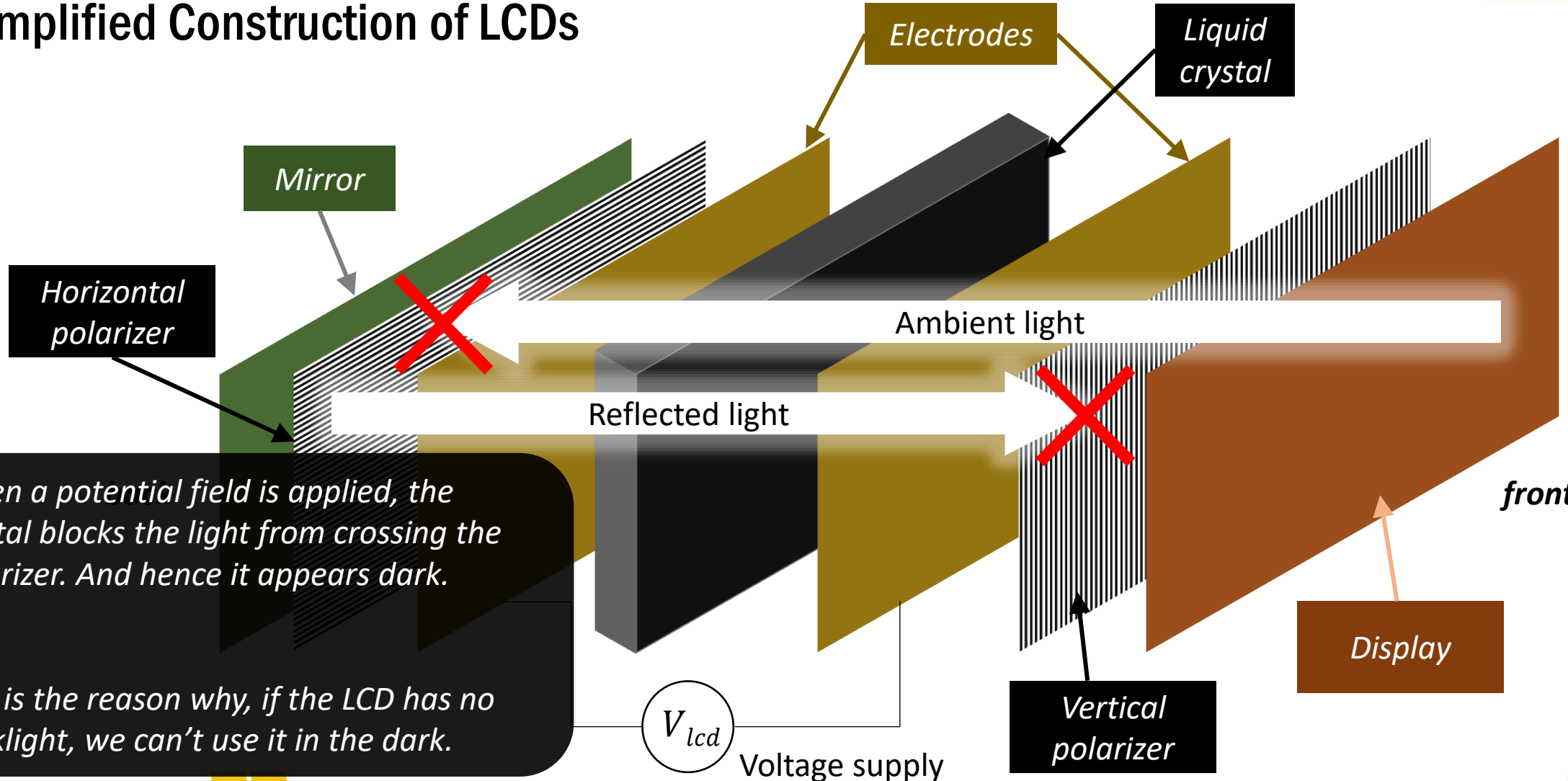


White arrows are light. Light enters from the right side and gets reflected from the left side.

When no potential difference exists between the electrodes, the crystal allows the light to pass through the polarizers.

LCD – Liquid Crystal Display

- **Simplified Construction of LCDs**

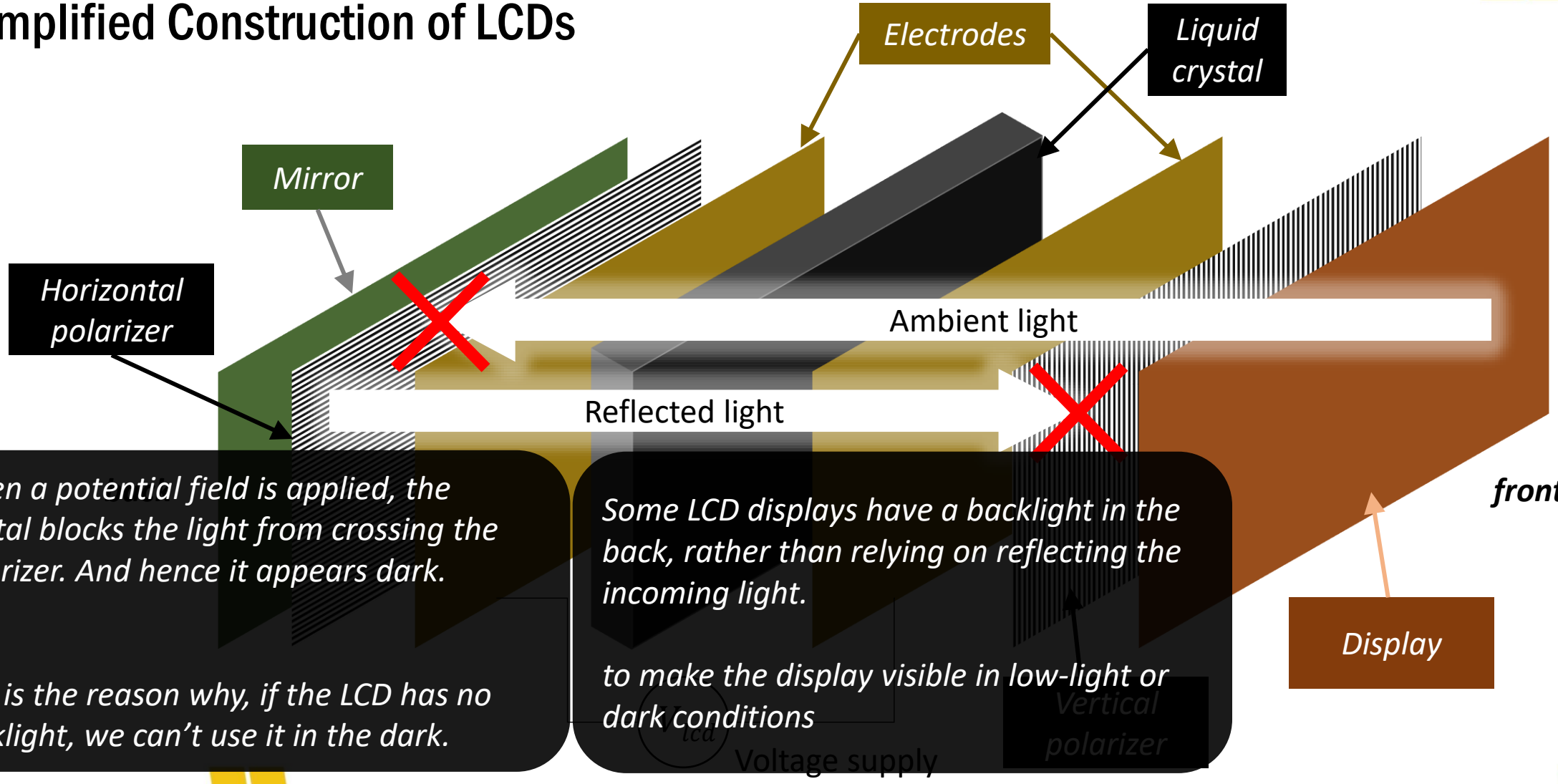


When a potential field is applied, the crystal blocks the light from crossing the polarizer. And hence it appears dark.

This is the reason why, if the LCD has no backlight, we can't use it in the dark.

LCD – Liquid Crystal Display

- **Simplified Construction of LCDs**



When a potential field is applied, the crystal blocks the light from crossing the polarizer. And hence it appears dark.

This is the reason why, if the LCD has no backlight, we can't use it in the dark.

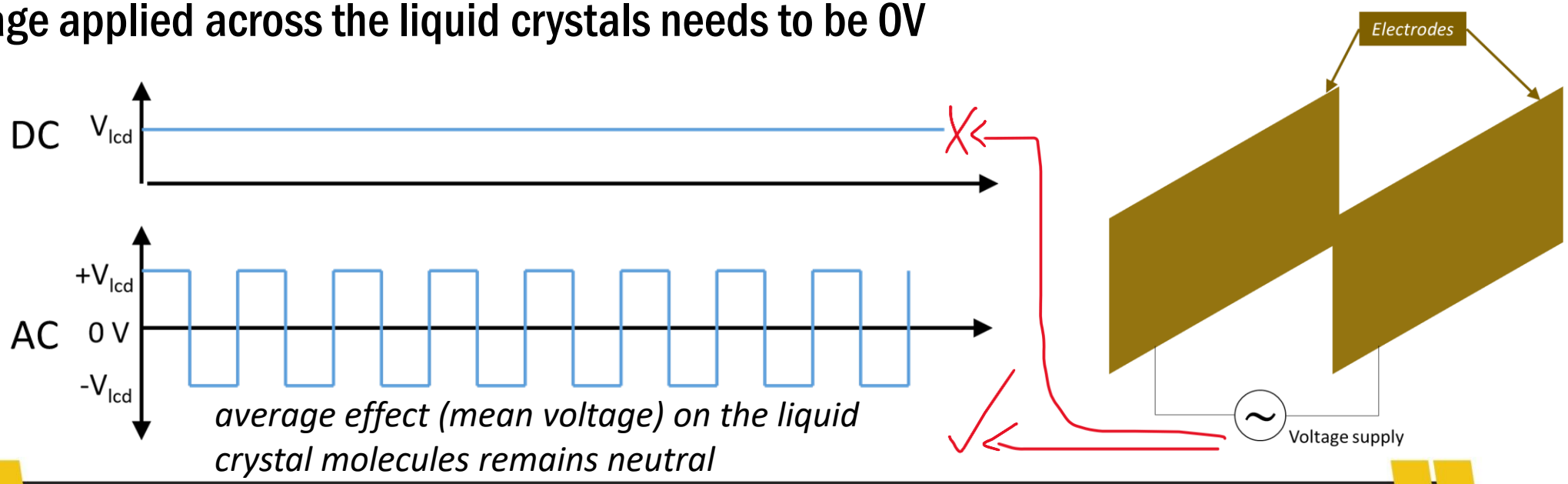
Some LCD displays have a backlight in the back, rather than relying on reflecting the incoming light.

to make the display visible in low-light or dark conditions

AC against Electrochemical Degradation

- Voltage supply is not constant DC voltage
  - can permanently alter the properties of the liquid crystals
  - Leading to degradation of the display over time (known as electrochemical degradation)

- An AC voltage is applied to the electrodes instead of a DC voltage
  - the mean voltage applied across the liquid crystals needs to be 0V

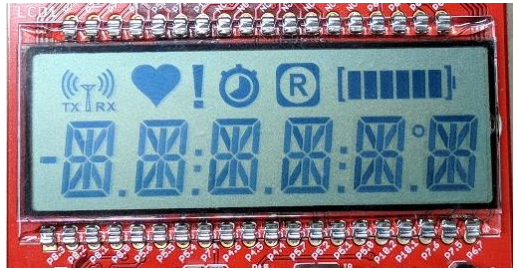


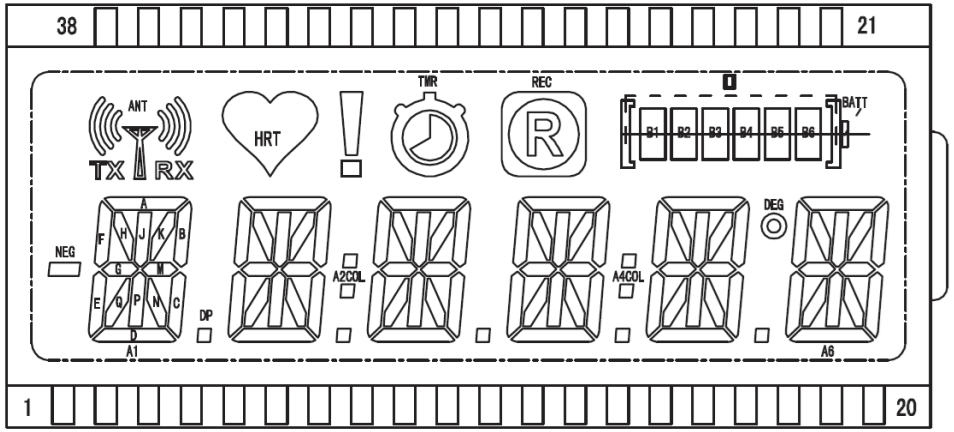
average effect (mean voltage) on the liquid crystal molecules remains neutral

LCD in MSP430

- LCD display consists of several individual segments that can be individually turned ON/OFF.

14-segment LCD

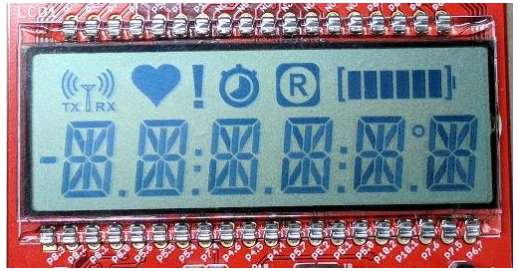




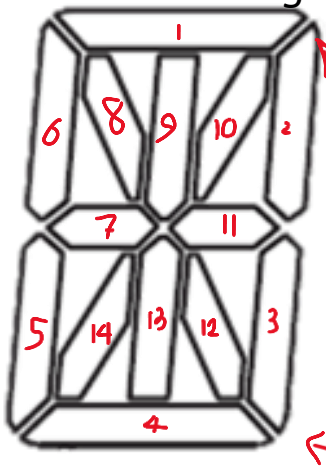
LCD in MSP430

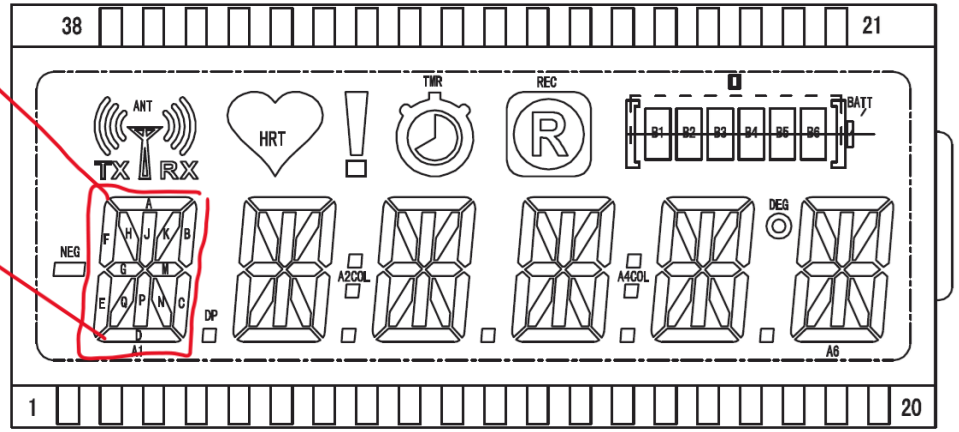
- LCD display consists of several individual segments that can be individually turned ON/OFF.

14-segment LCD



Individual segments

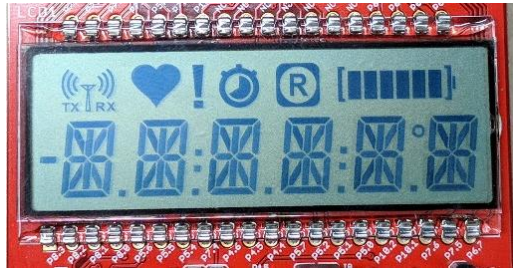




LCD in MSP430

- **LCD display consists of several individual segments that can be individually turned ON/OFF.**

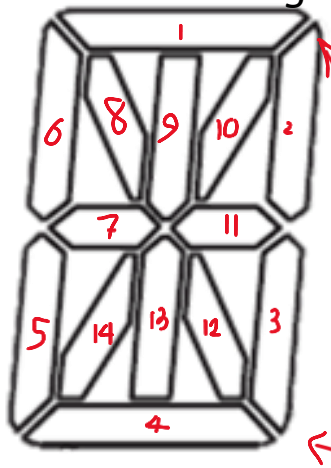
14-segment LCD

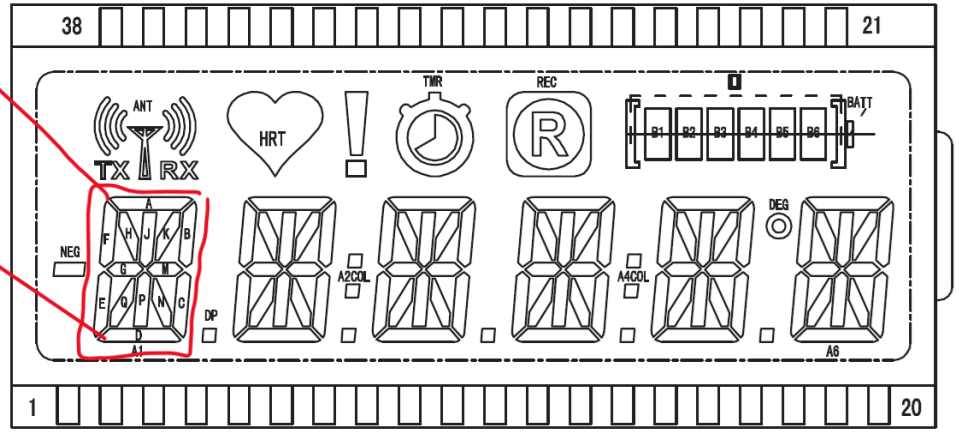


*There are **108 segments** in the LCD display. Each segment needs to be controlled independently and therefore each segment needs **2 pins for applying the AC voltage**.*

How many pins do we need?

Individual segments



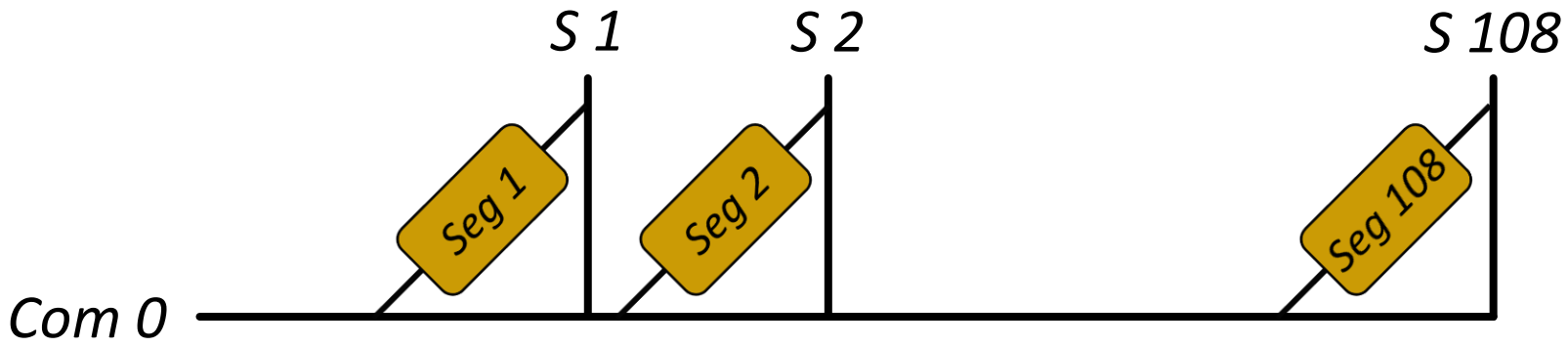


- **Worst-case: 108x2 = 216 pins for LCD**

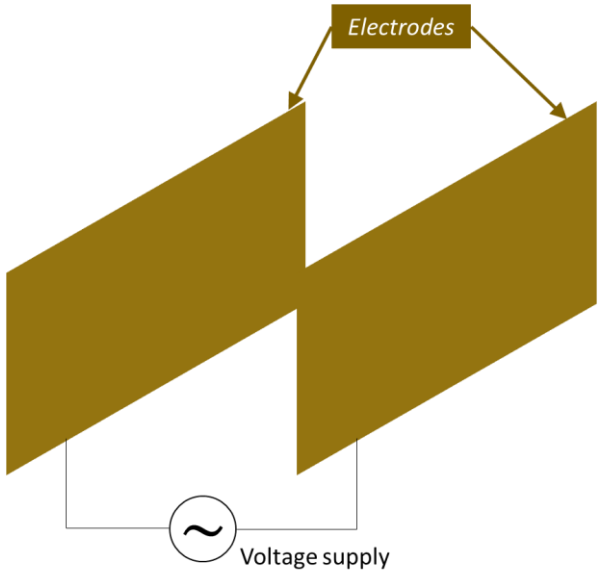
*methods like **multiplexing** or **common electrodes** are used to reduce the number of pins significantly.*

Multiplexing in LCD

- Static mode
  - Each segment needs two pins.
  - Let's make one of the pins common to every segment and change the voltage on the other pin.
    - 108 + 1 = 109 pins needed.

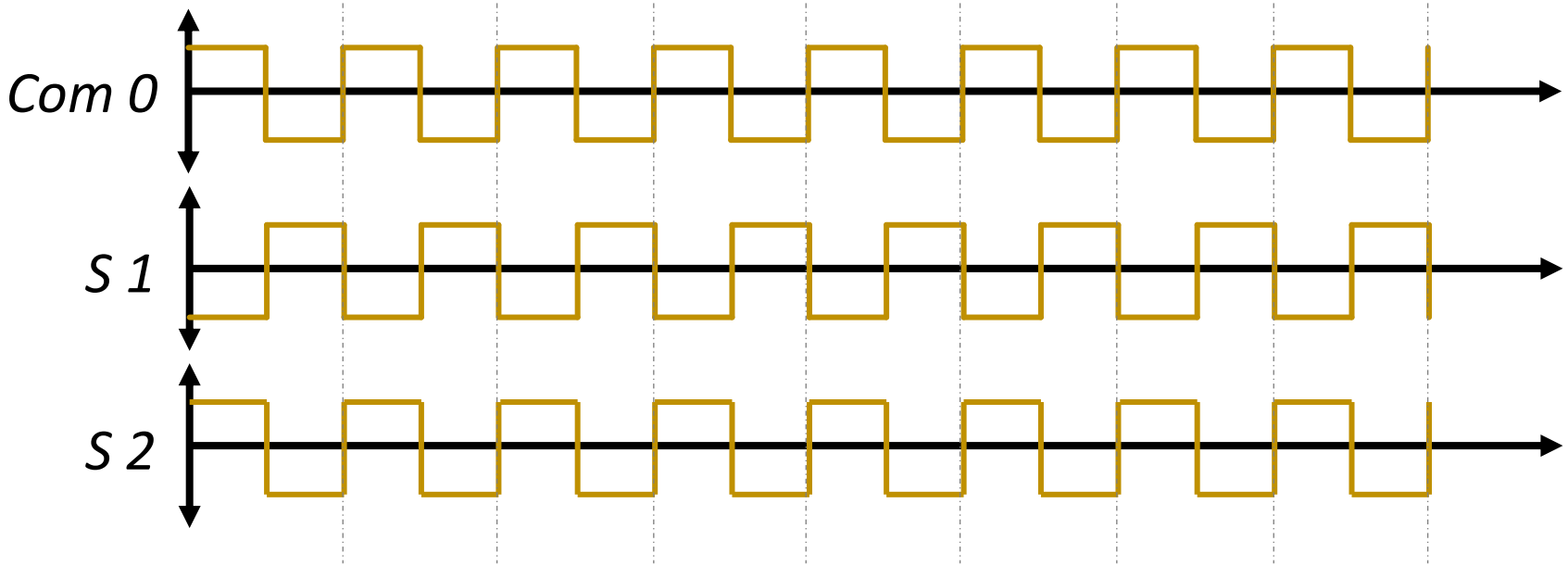


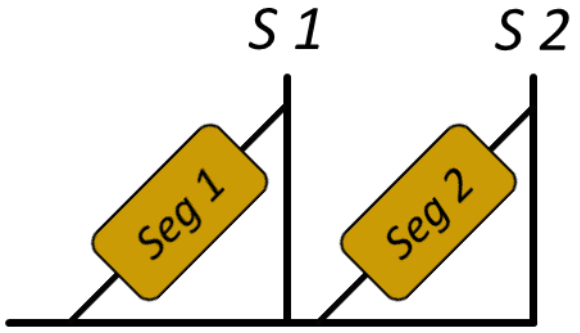
One pin per electrode



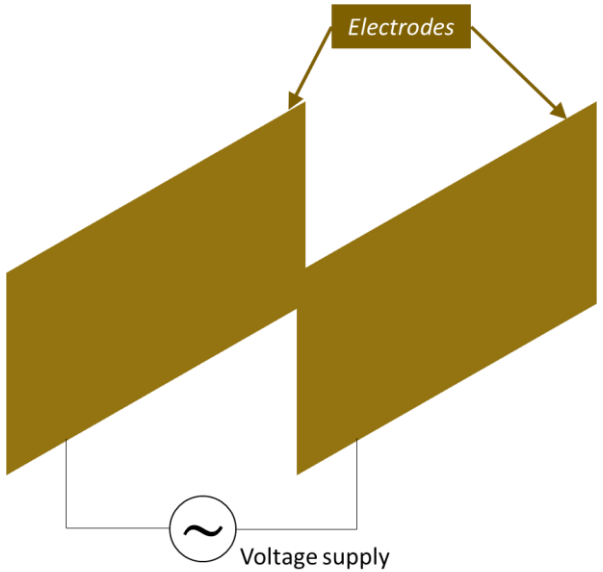
Multiplexing in LCD

- The voltage of both end (differential)



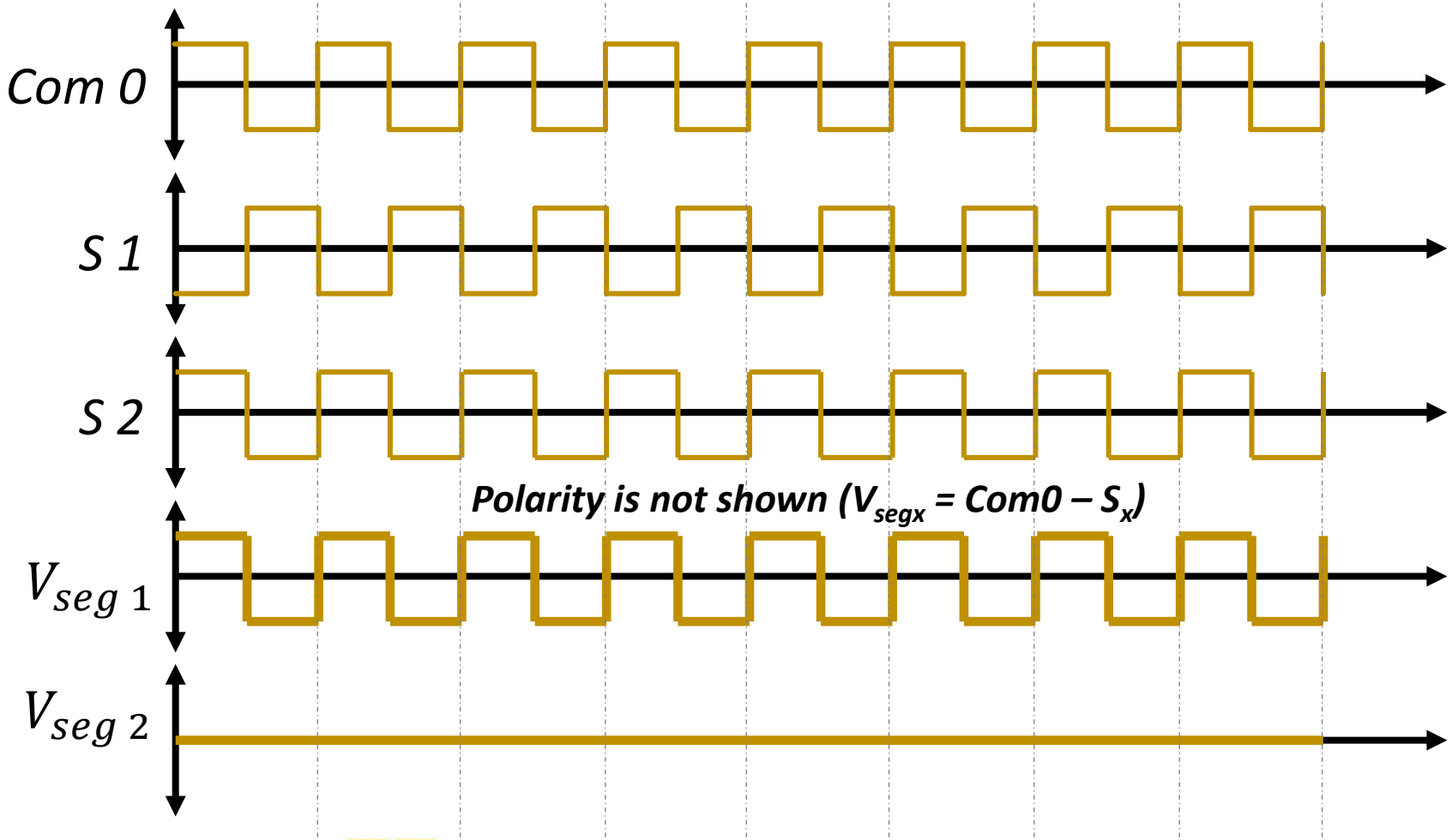


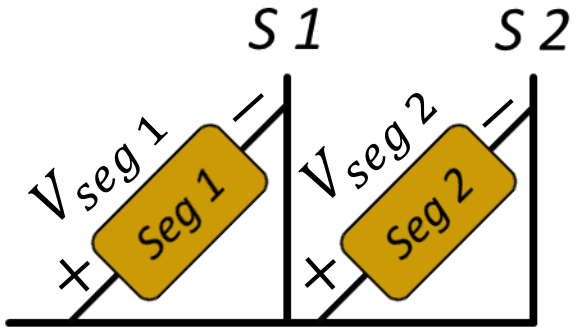
One pin per electrode



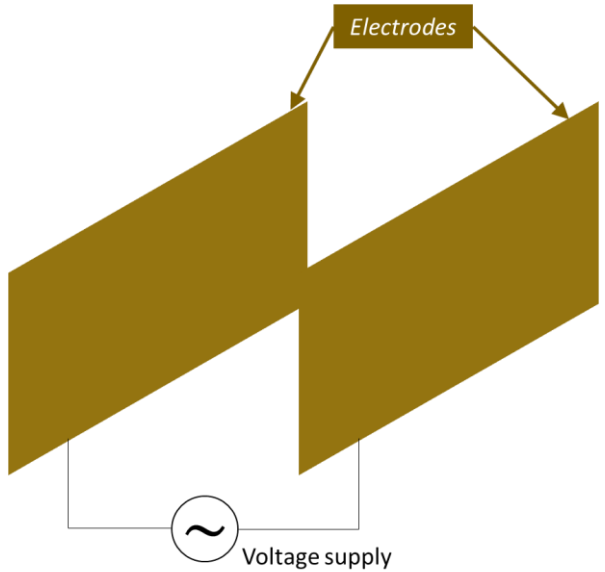
Multiplexing in LCD

- The voltage of both end (differential)



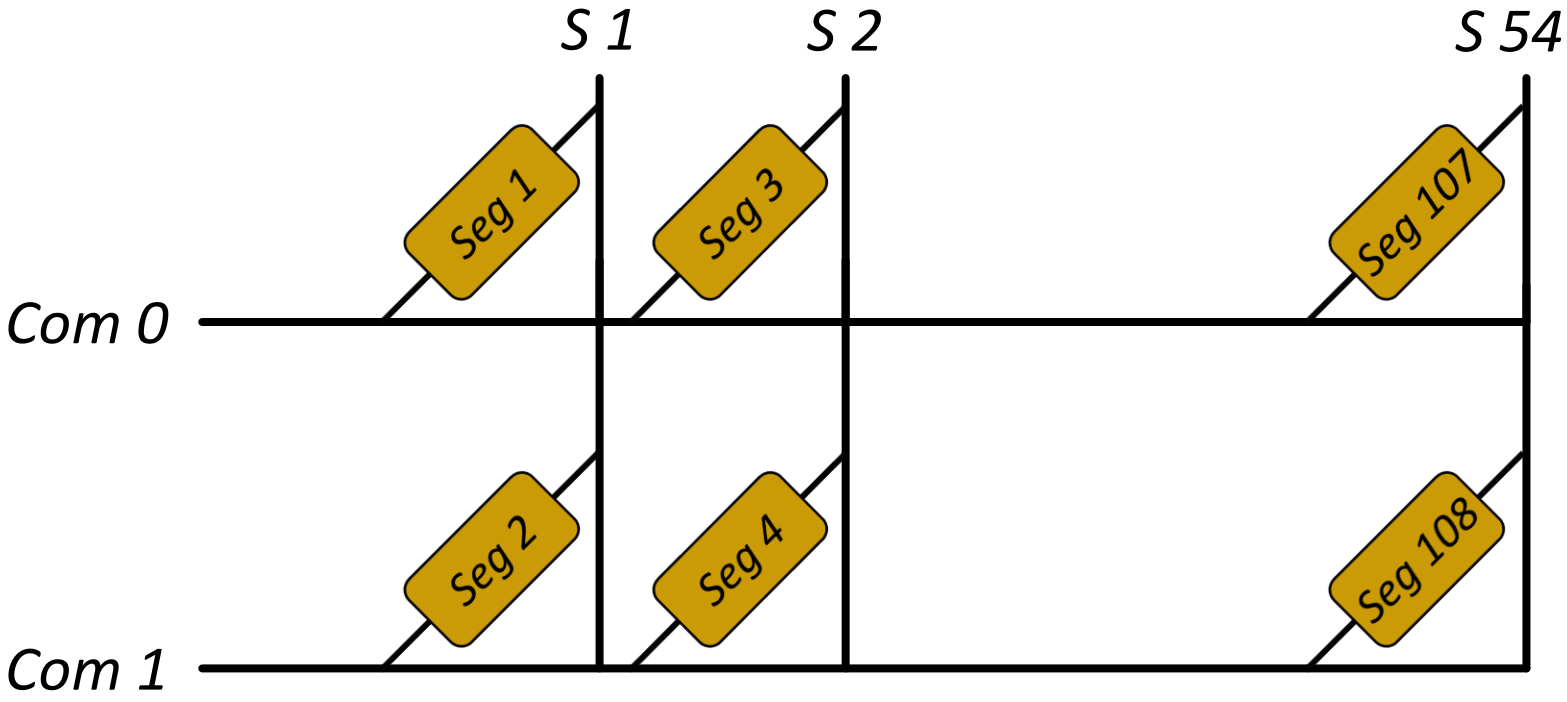


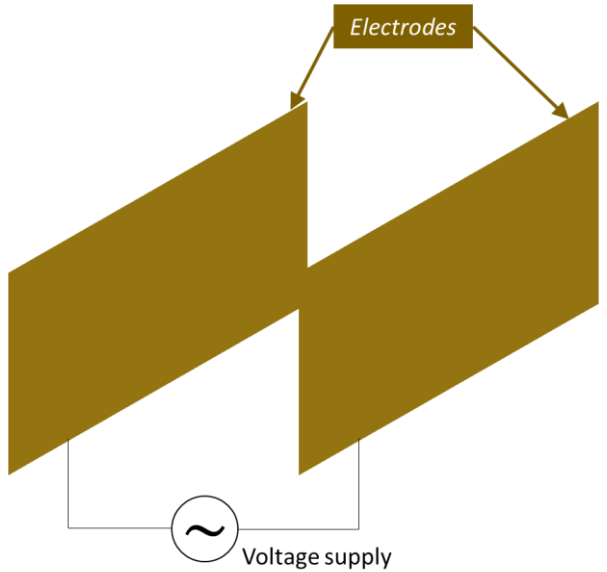
One pin per electrode



Multiplexing in LCD

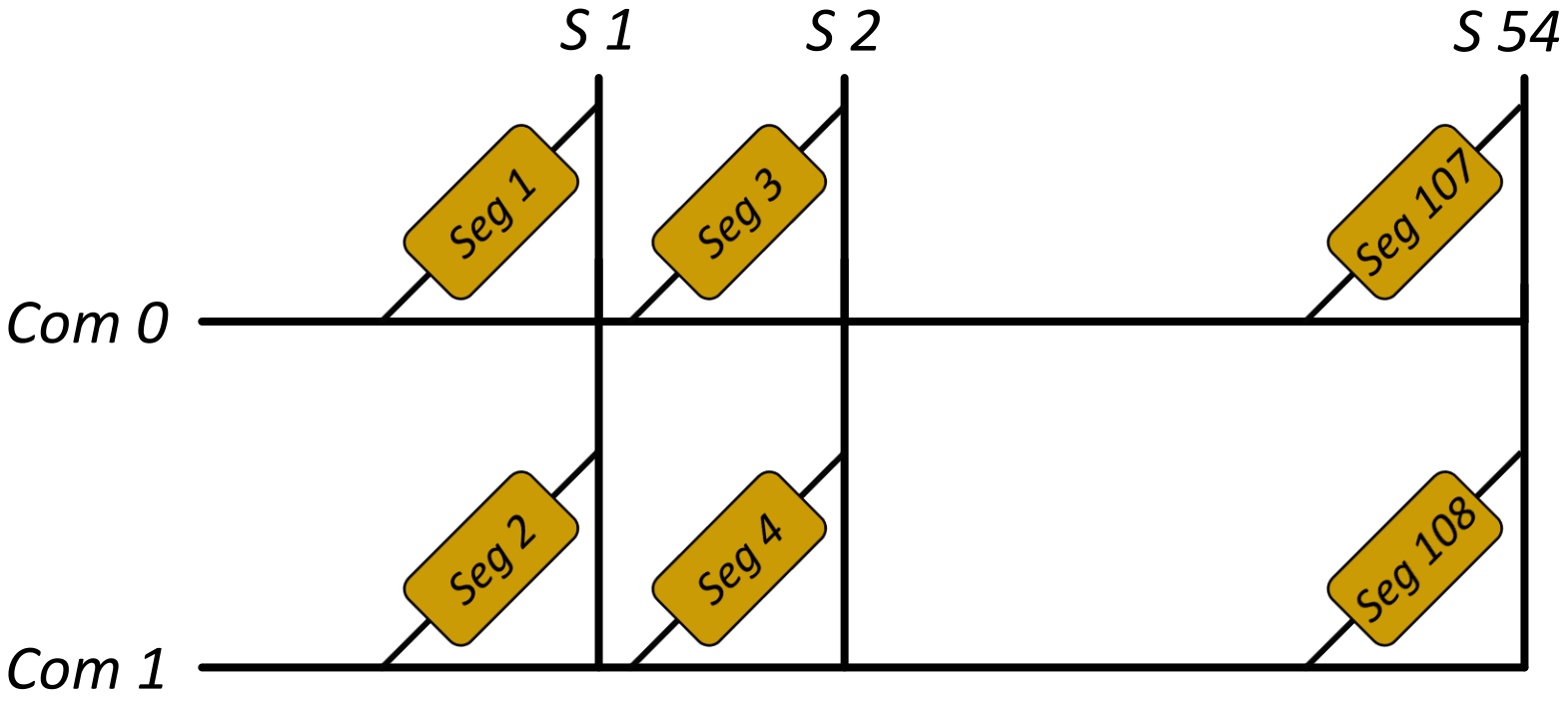
- 2-mux mode
  - Let's divide all the pins into 2 groups, therefore 2 common pins.
  - Let's display group 1 for half the period, group 2 for half the period.
  - 108/2 + 2 = 54 + 2 = 56 pins needed.



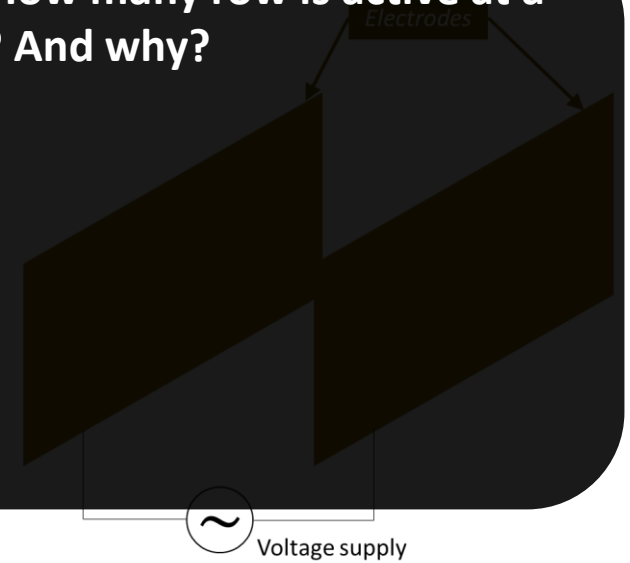


Multiplexing in LCD

- 2-mux mode
  - Let's divide all the pins into 2 groups, therefore 2 common pins.
  - Let's display group 1 for half the period, group 2 for half the period.
  - 108/2 + 2 = 54 + 2 = 56 pins needed.

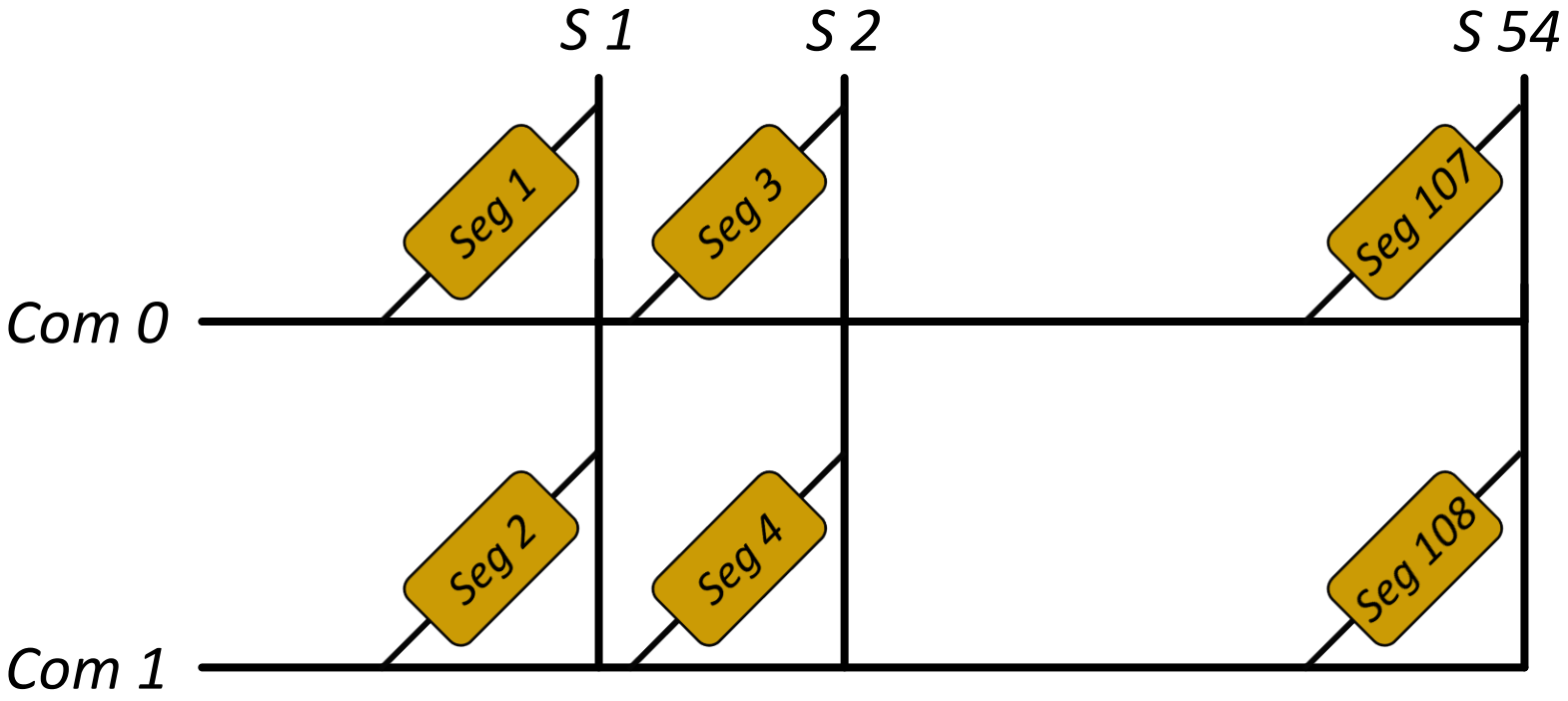


(Q1) How many row is active at a time? And why?



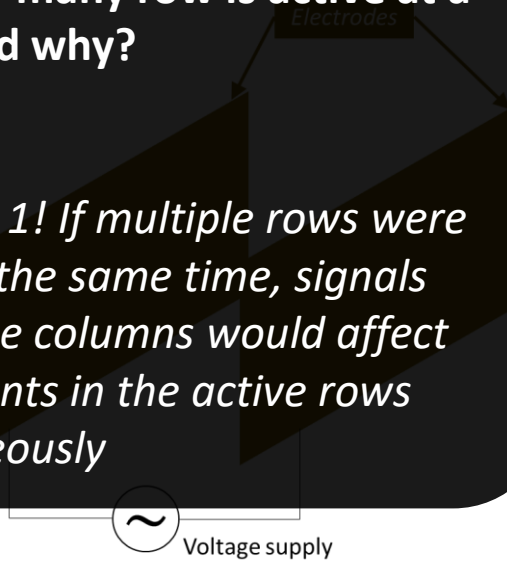
Multiplexing in LCD

- 2-mux mode
  - Let's divide all the pins into 2 groups, therefore 2 common pins.
  - Let's display group 1 for half the period, group 2 for half the period.
  - 108/2 + 2 = 54 + 2 = 56 pins needed.



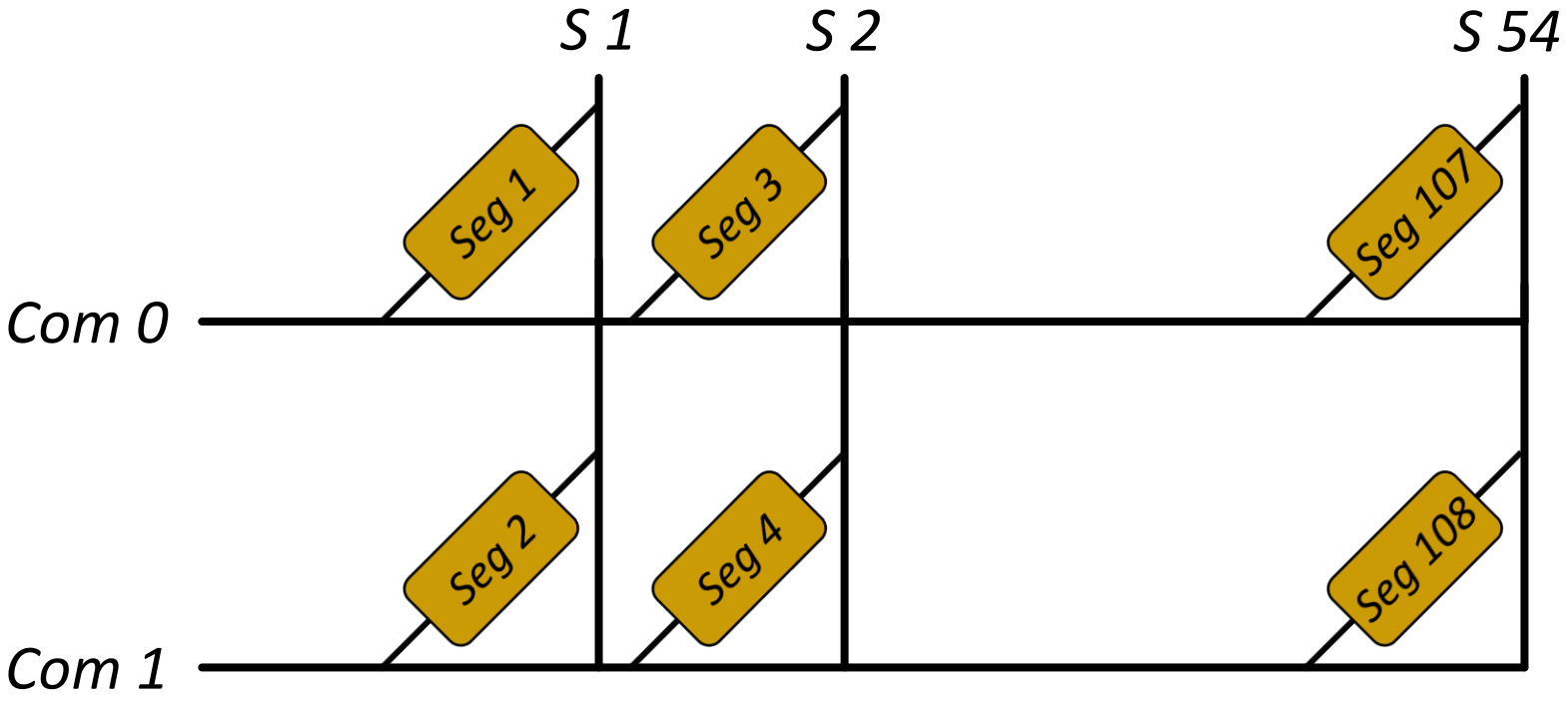
(Q1) How many row is active at a time? And why?

(A1) Only 1! If multiple rows were active at the same time, signals sent to the columns would affect all segments in the active rows simultaneously

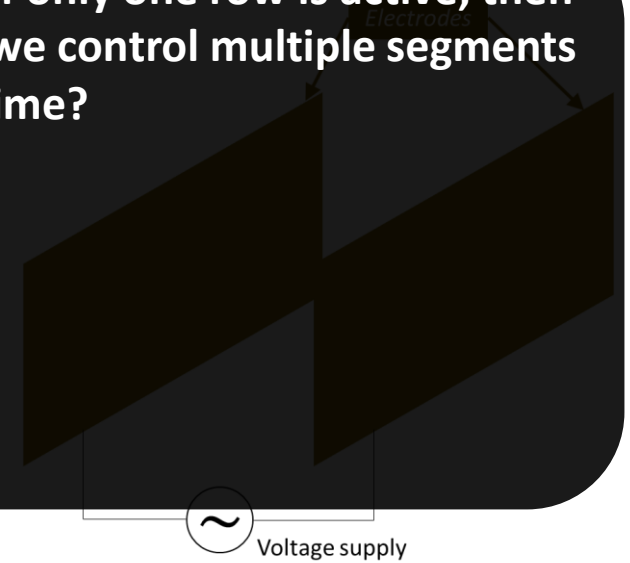


Multiplexing in LCD

- 2-mux mode
  - Let's divide all the pins into 2 groups, therefore 2 common pins.
  - Let's display group 1 for half the period, group 2 for half the period.
  - 108/2 + 2 = 54 + 2 = 56 pins needed.

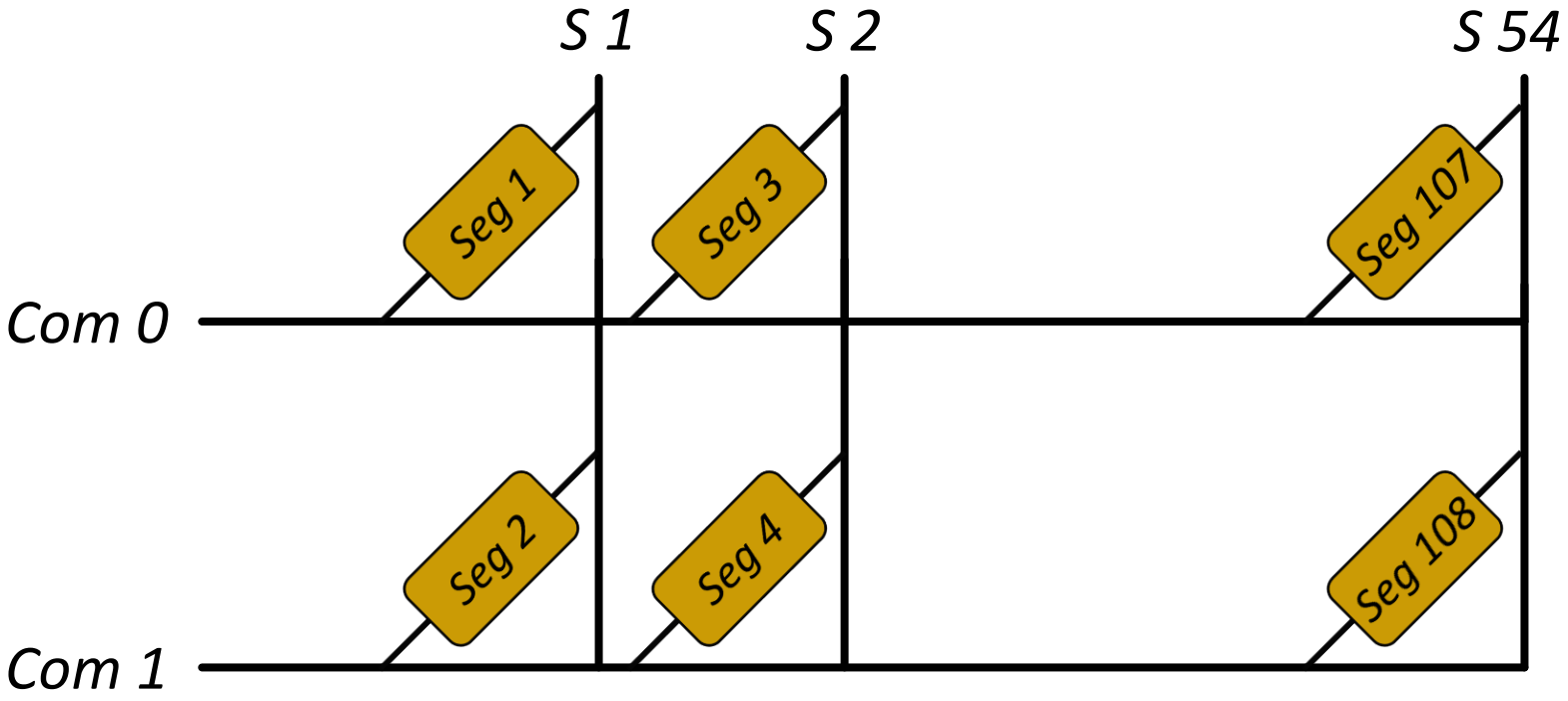


(Q2) If only one row is active, then how we control multiple segments at a time?



Multiplexing in LCD

- 2-mux mode
  - Let's divide all the pins into 2 groups, therefore 2 common pins.
  - Let's display group 1 for half the period, group 2 for half the period.
  - 108/2 + 2 = 54 + 2 = 56 pins needed.

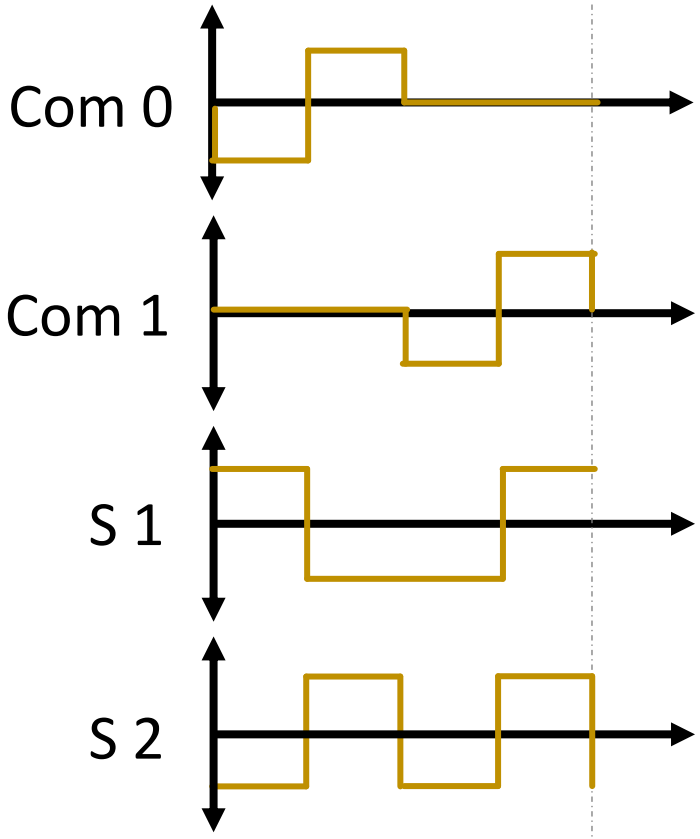


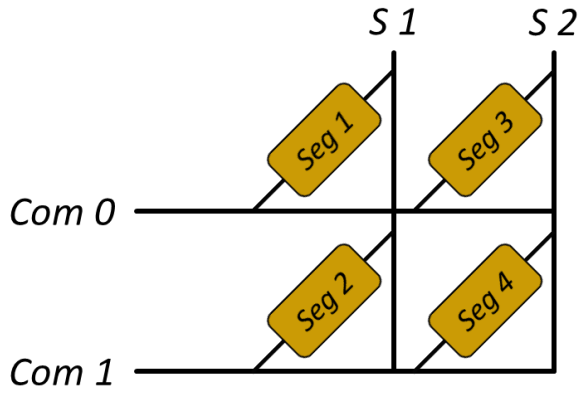
(Q2) If only one row is active, then how we control multiple segments at a time?

(A2) time-division multiplexing with short period of time. Human eyes cannot perceive display faster than 60-90 Hz. Even using 32KHz and TDMA is fine.

Multiplexing in LCD

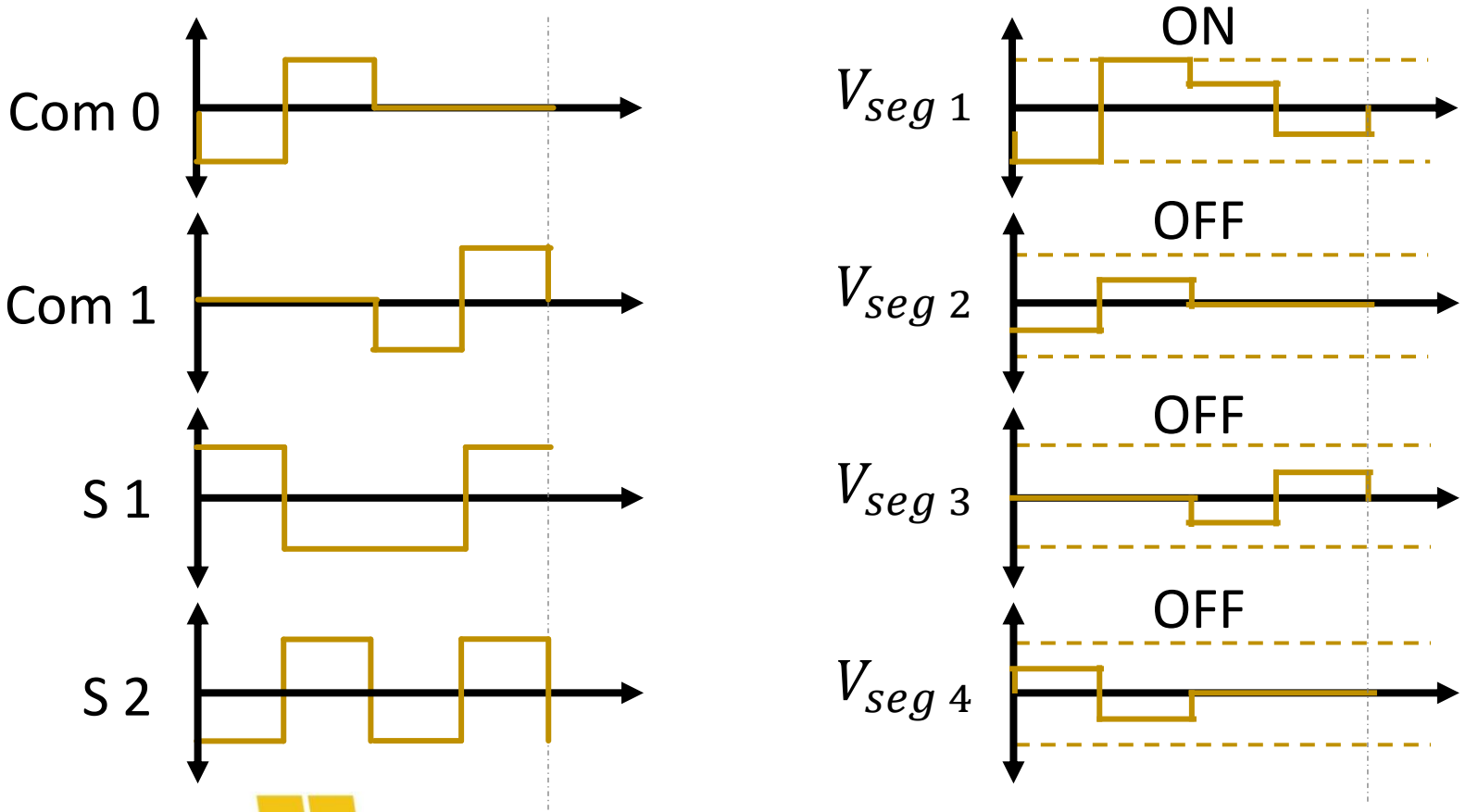
- 2-mux mode
  - Turn on segment 1, turn off segments 2,3,4.

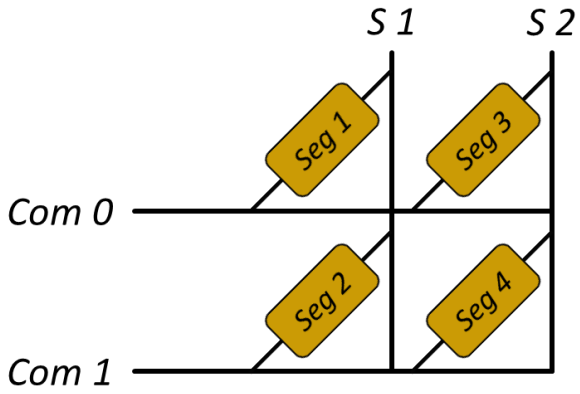




Multiplexing in LCD

- 2-mux mode
  - Turn on segment 1, turn off segments 2,3,4.



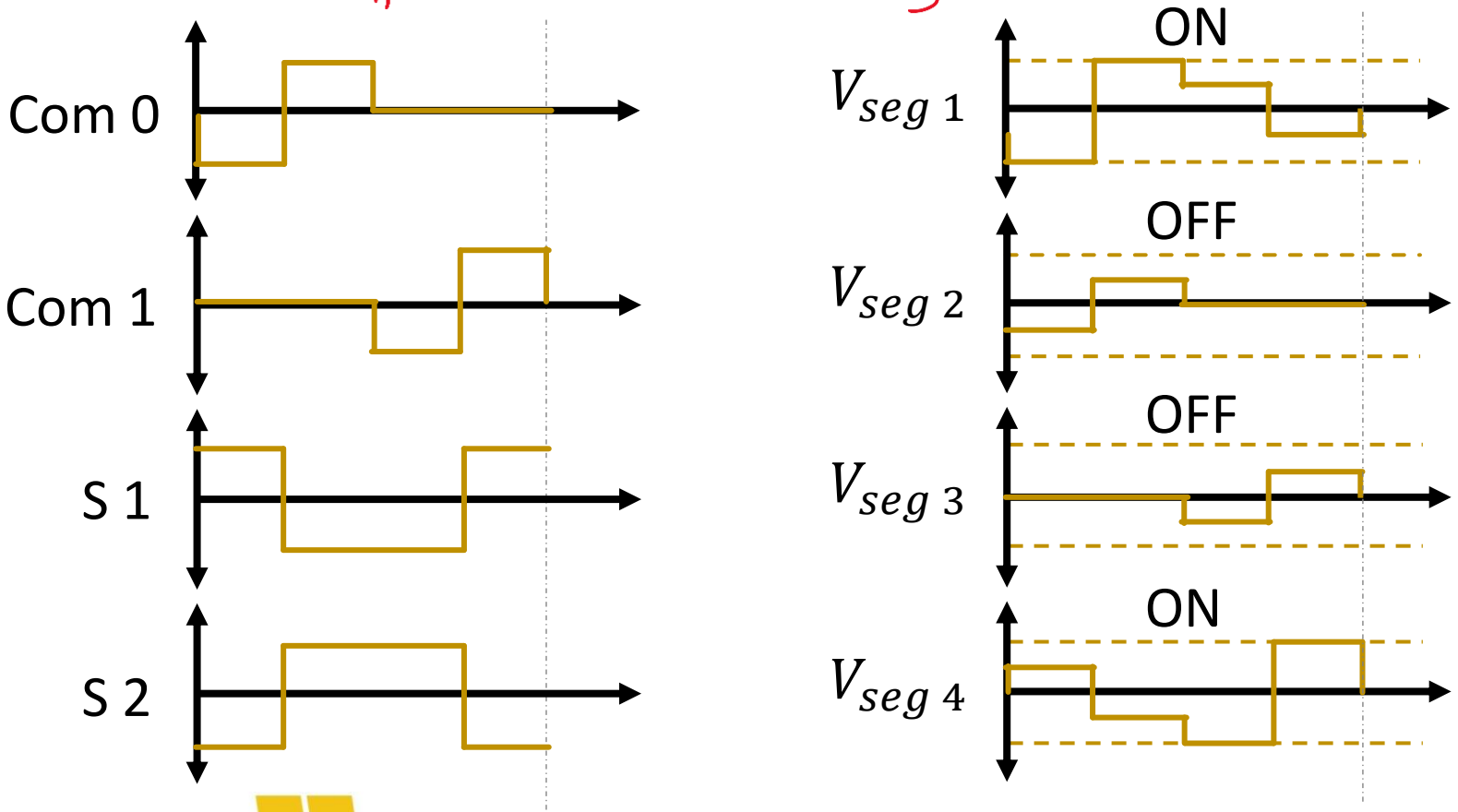


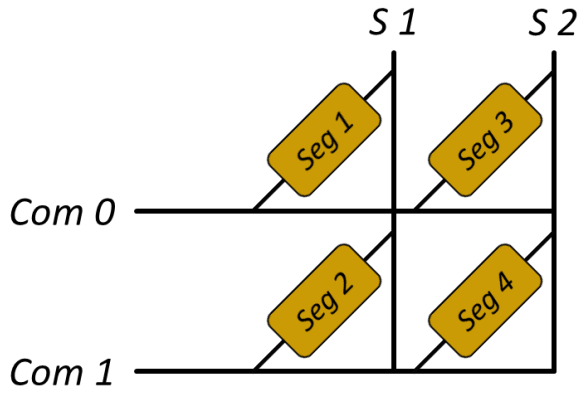
Multiplexing in LCD

- 2-mux mode
  - Turn on segment ~~1~~, turn off segments ~~2,3,4~~.

1,4

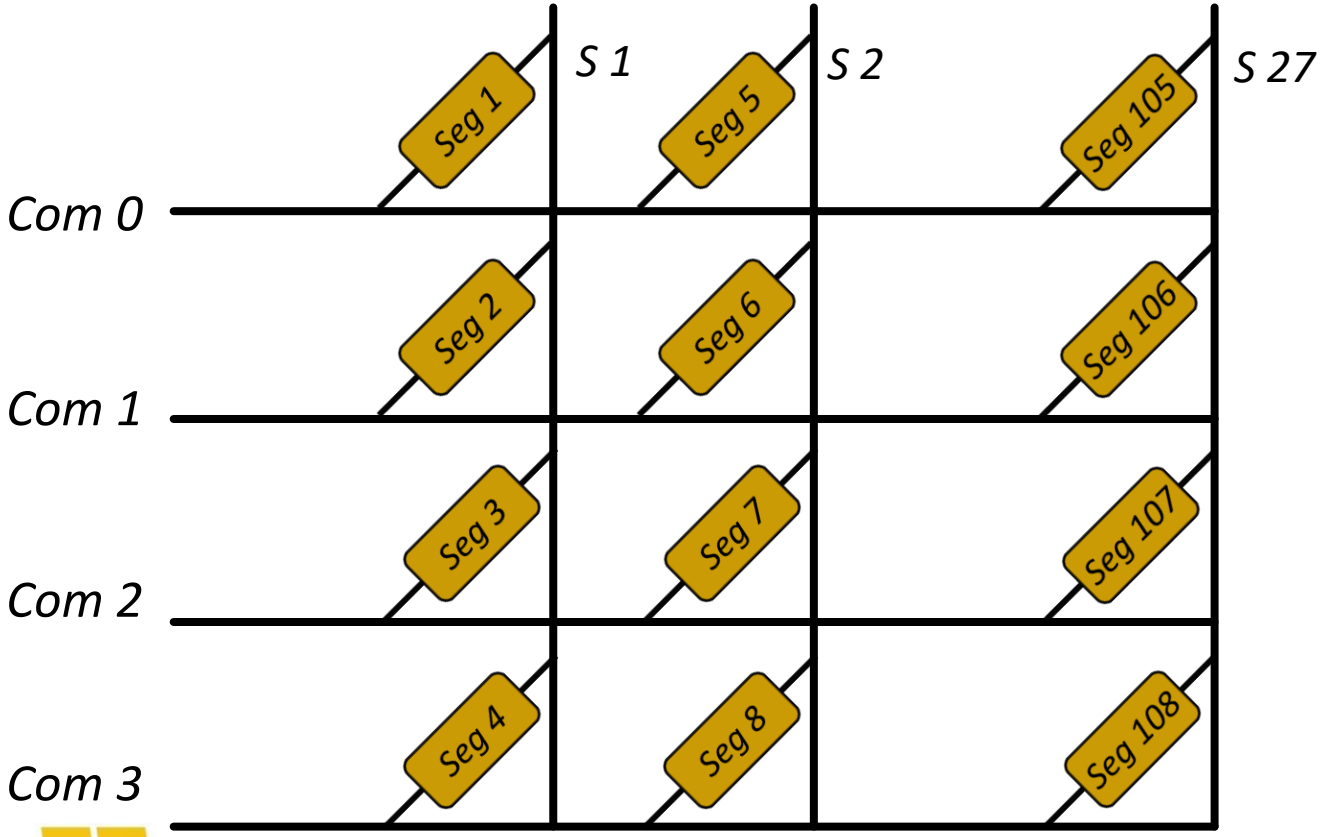
2.3

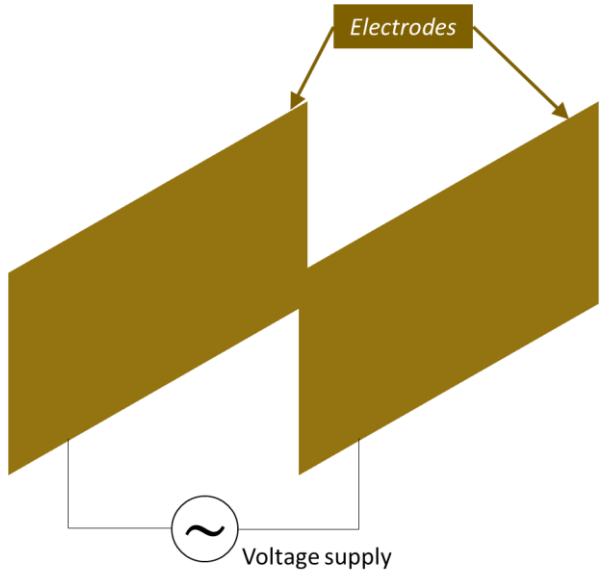




Multiplexing in LCD

- 4-mux mode
  - Let's divide all the pins into 4 groups, therefore 4 common pins.
  - 108/4 + 4 = 27 + 4 = 31 pins needed.

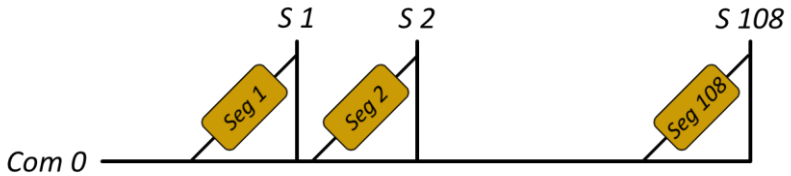


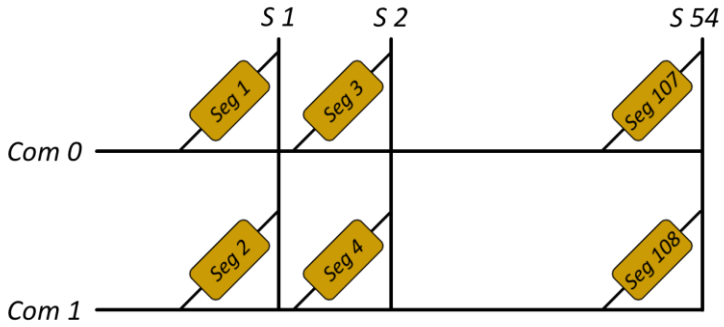


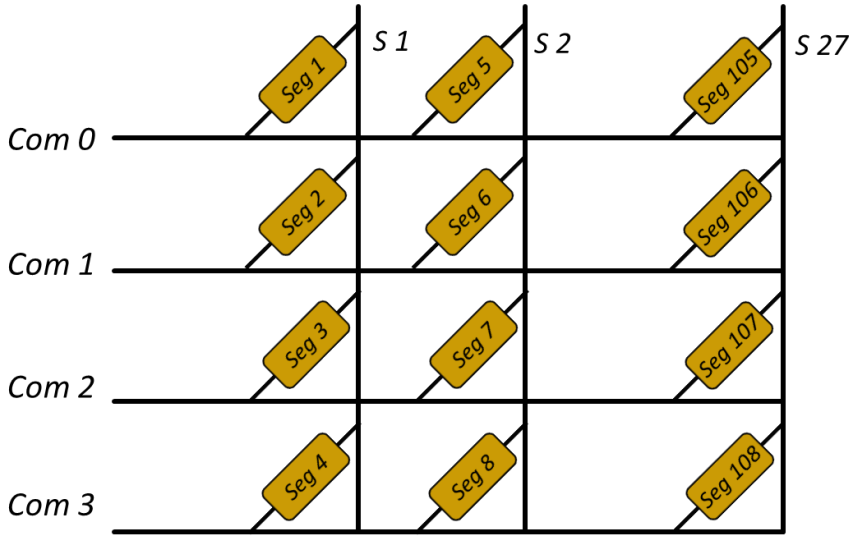
Multiplexing in LCD

- Modes

Mode	Common lines (Com i)	Signal lines (S j)	Total pins	$n = 108$
Static mode	*-*	*n*	*n*	*108*
2-mux mode	*2*	*n/2*	*n/2 + 2*	*56*
4-mux mode	*4*	*n/4*	*n/4 + 4*	*31*





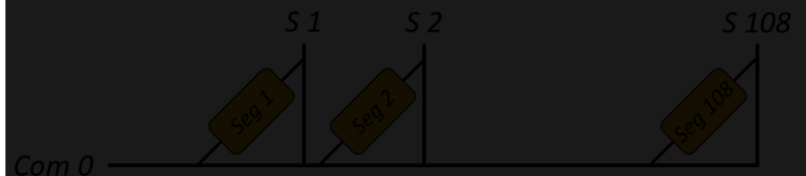


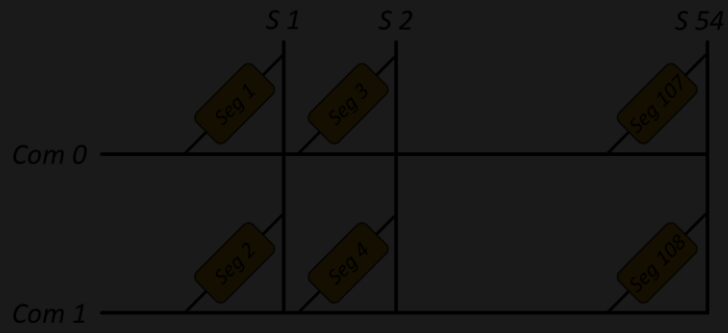
Multiplexing in LCD

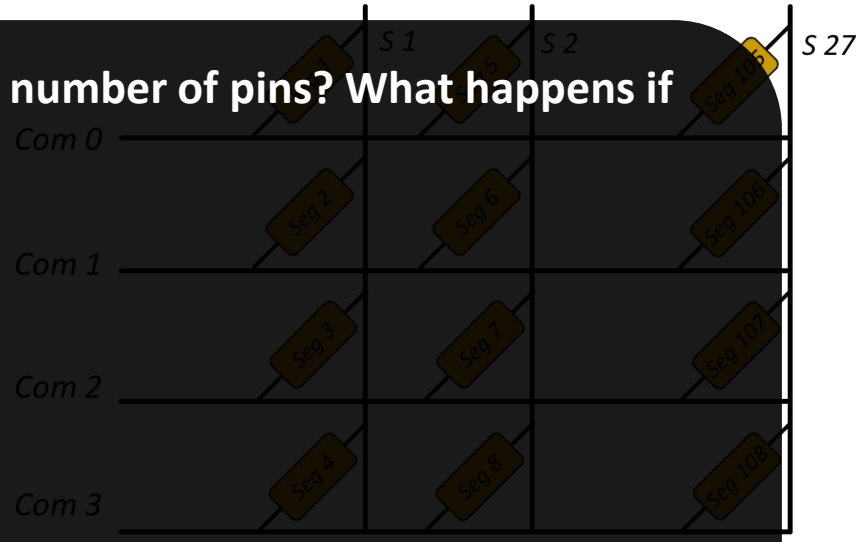
- Modes

Mode	Common lines (Com i)	Signal lines (S j)	Total pins	$n = 108$
Static mode	-	*n*	*n*	*108*
2-mux mode	*2*	*n/2*	*n/2 + 2*	*56*
4-mux mode	*4*	*n/4*	*n/4 + 4*	*31*

(Q) Can we increase the number of common lines indefinitely to reduce the total number of pins? What happens if we keep increasing the number of common lines?





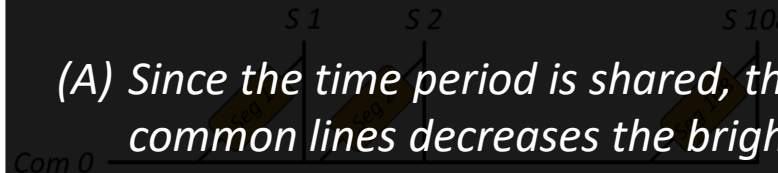


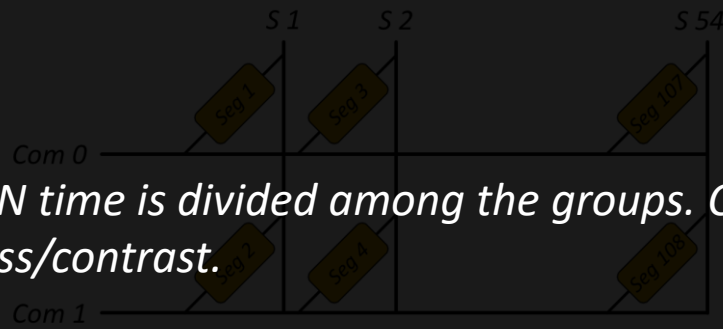
Multiplexing in LCD

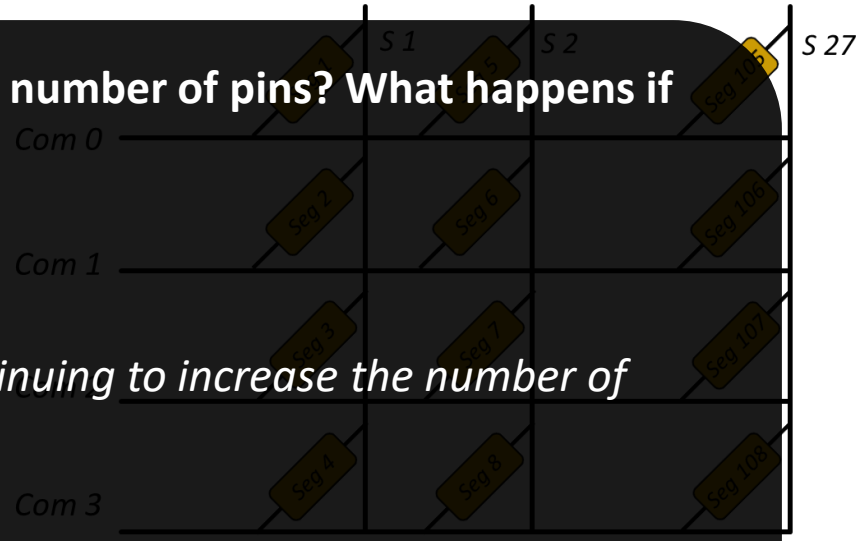
- Modes

Mode	Common lines (Com i)	Signal lines (S j)	Total pins	$n = 108$
Static mode	*-*	*n*	*n*	*108*
2-mux mode	*2*	*n/2*	*n/2 + 2*	*56*
4-mux mode	*4*	*n/4*	*n/4 + 4*	*31*

(Q) Can we increase the number of common lines indefinitely to reduce the total number of pins? What happens if we keep increasing the number of common lines?







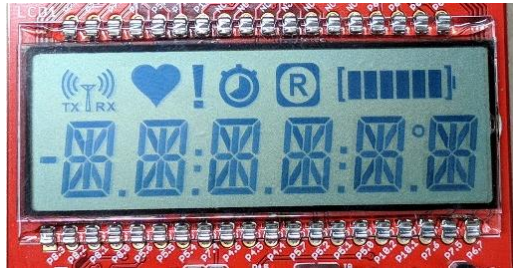
(A) Since the time period is shared, the ON time is divided among the groups. Continuing to increase the number of common lines decreases the brightness/contrast.

MSP430 supports up to 8-mux.

LCD in MSP430

- **LCD display consists of several individual segments that can be individually turned ON/OFF.**

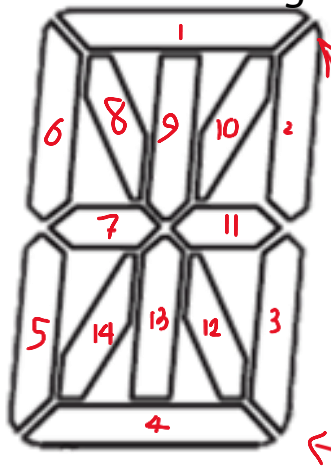
14-segment LCD

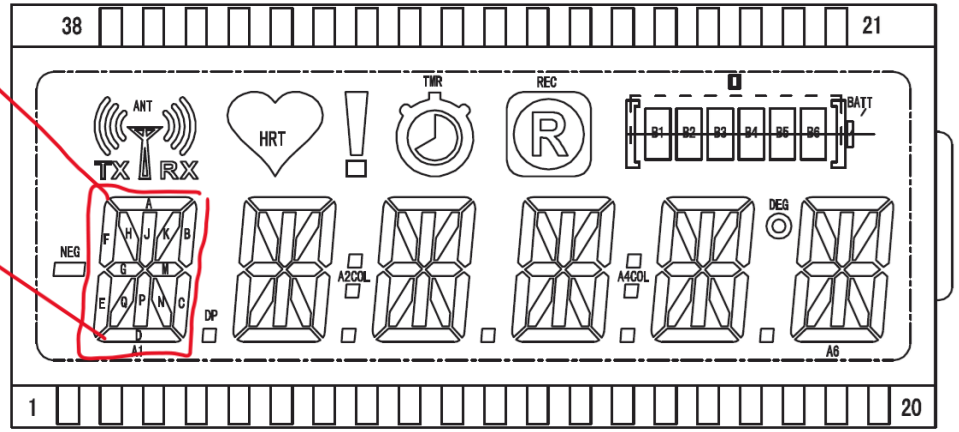


*There are **108 segments** in the LCD display. Each segment needs to be controlled independently and therefore each segment needs **2 pins for applying the AC voltage**.*

How many pins do we need?

Individual segments





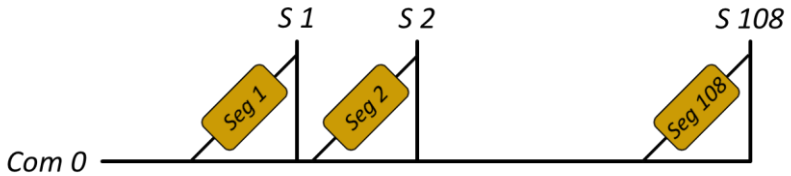
- **Worst-case: 108x2 = 216 pins for LCD**

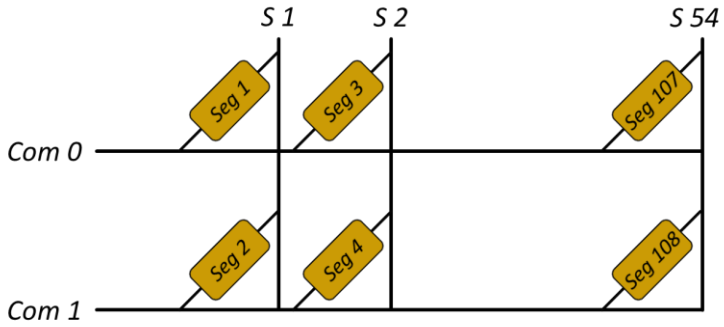
*methods like **multiplexing** or **common electrodes** are used to reduce the number of pins significantly.*

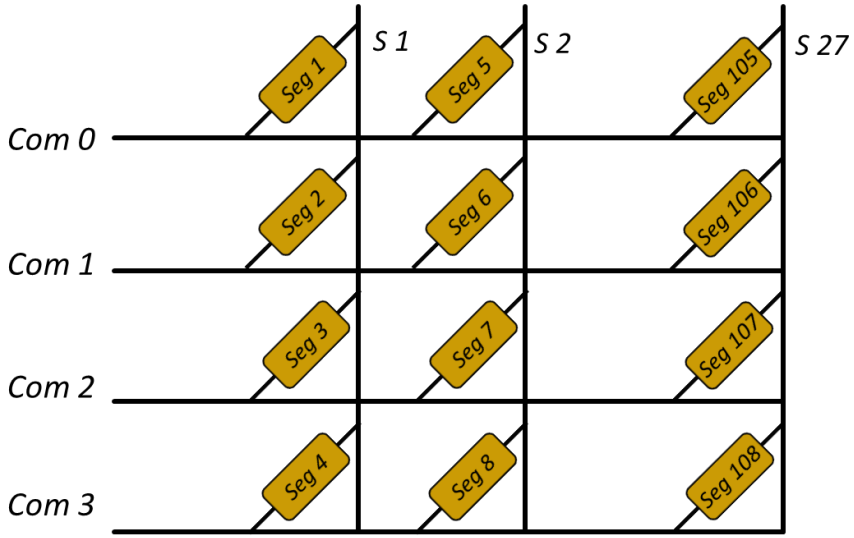
Multiplexing in LCD

- Modes

Mode	Common lines (Com i)	Signal lines (S j)	Total pins	$n = 108$
Static mode	*-*	*n*	*n*	*108*
2-mux mode	*2*	*n/2*	*n/2 + 2*	*56*
4-mux mode	*4*	*n/4*	*n/4 + 4*	*31*





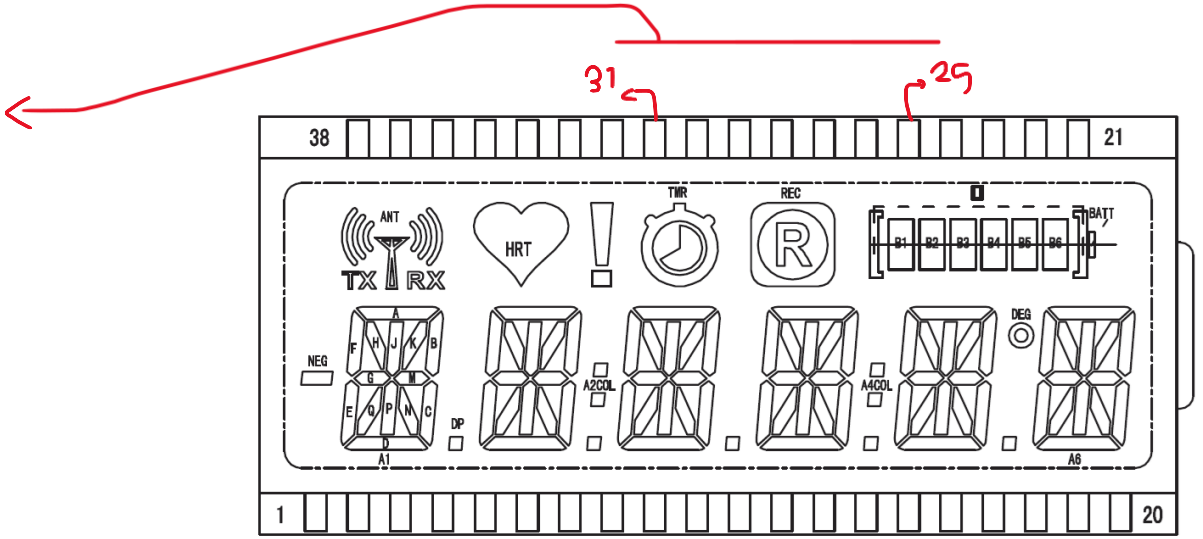


Pins related to LCD

- Modes

Mode	Common lines (Com i)	Signal lines (S j)	Total pins	$n = 108$
Static mode	-	*n*	*n*	*108*
2-mux mode	*2*	*n/2*	*n/2 + 2*	*56*
4-mux mode	*4*	*n/4*	*n/4 + 4*	*31*

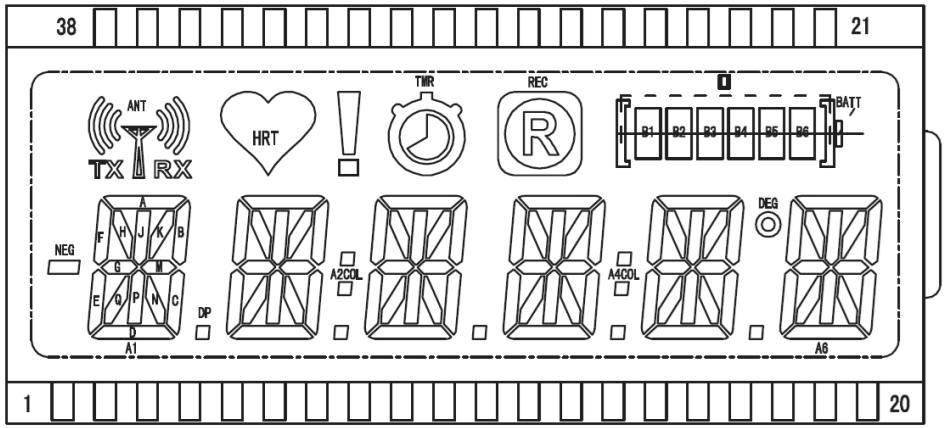
- The LCD is MSP430EXPFR6989 evaluation board uses 4-way multiplexing.
  - 31 pins required.
  - There are 38 pins for it (7 pins are unused).



Pins related to LCD

- Pins

PIN	COM3	COM2	COM1	COM0
1	A1E	A1F	A1G	A1M
2	A1A	A1B	A1C	A1D
3	A1Q	NEG	A1N	A1DP
4	A1H	A1J	A1K	A1P
5	A2E	A2F	A2G	A2M
6	A2A	A2B	A2C	A2D
7	A2Q	A2COL	A2N	A2DP
8	A2H	A2J	A2K	A2P
9	A3R	A3F	A3G	A3M
10	A3A	A3B	A3C	A3D
11	A3Q	ANT	A3N	A3DP
12	A3H	A3J	A3K	A3P
13	A4R	A4F	A4G	A4M
14	A4A	A4B	A4C	A4D
15	A4Q	A4COL	A4N	A4DP
16	A4H	A4J	A4K	A4P
17	A5E	A5F	A5G	A5M
18	A5A	A5B	A5C	A5D
19	A5Q	DEG	A5N	A5DP
20	A5H	A5J	A5K	A5P

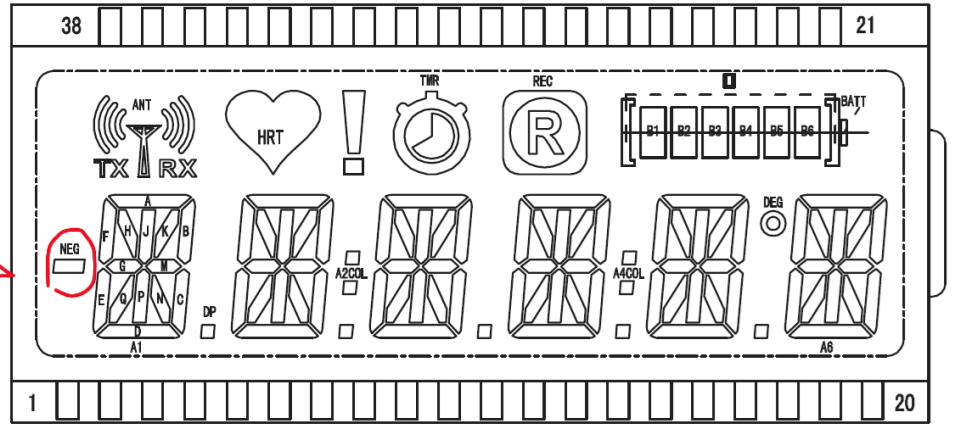


Multiplexing in LCD

- Pins

PIN	COM3	COM2	COM1	COM0
1	A1E	A1F	A1G	A1M
2	A1A	A1B	A1C	A1D
3	A1Q	NEG	A1N	A1DP
4	A1H	A1J	A1K	A1P
5	A2E	A2F	A2G	A2M
6	A2A	A2B	A2C	A2D
7	A2Q	A2COL	A2N	A2DP
8	A2H	A2J	A2K	A2P
9	A3R	A3F	A3G	A3M
10	A3A	A3B	A3C	A3D
11	A3Q	ANT	A3N	A3DP
12	A3H	A3J	A3K	A3P
13	A4R	A4F	A4G	A4M
14	A4A	A4B	A4C	A4D
15	A4Q	A4COL	A4N	A4DP
16	A4H	A4J	A4K	A4P
17	A5E	A5F	A5G	A5M
18	A5A	A5B	A5C	A5D
19	A5Q	DEG	A5N	A5DP
20	A5H	A5J	A5K	A5P

for negative numbers

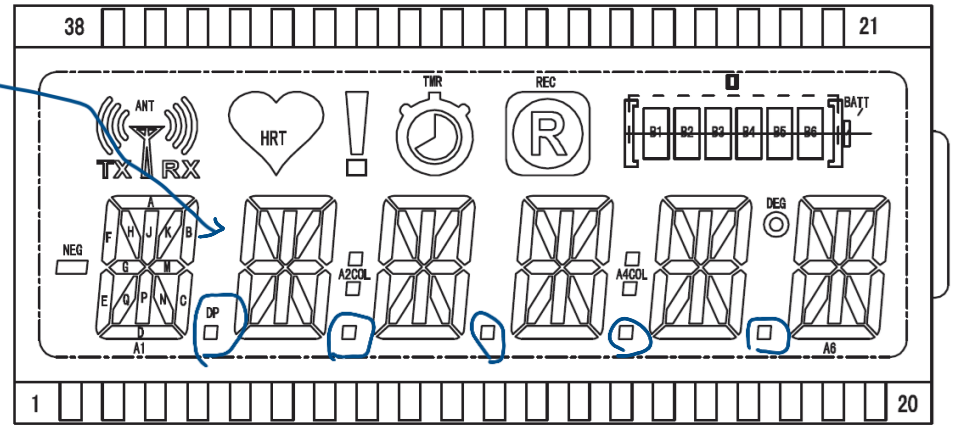


Multiplexing in LCD

- Pins

PIN	COM3	COM2	COM1	COM0
1	A1E	A1F	A1G	A1M
2	A1A	A1B	A1C	A1D
3	A1Q	NEG	A1N	A1DP
4	A1H	A1J	A1K	A1P
5	A2E	A2F	A2G	A2M
6	A2A	A2B	A2C	A2D
7	A2Q	A2COL	A2N	A2DP
8	A2H	A2J	A2K	A2P
9	A3R	A3F	A3G	A3M
10	A3A	A3B	A3C	A3D
11	A3Q	ANT	A3N	A3DP
12	A3H	A3J	A3K	A3P
13	A4R	A4F	A4G	A4M
14	A4A	A4B	A4C	A4D
15	A4Q	A4COL	A4N	A4DP
16	A4H	A4J	A4K	A4P
17	A5E	A5F	A5G	A5M
18	A5A	A5B	A5C	A5D
19	A5Q	DEG	A5N	A5DP
20	A5H	A5J	A5K	A5P

For floating numbers

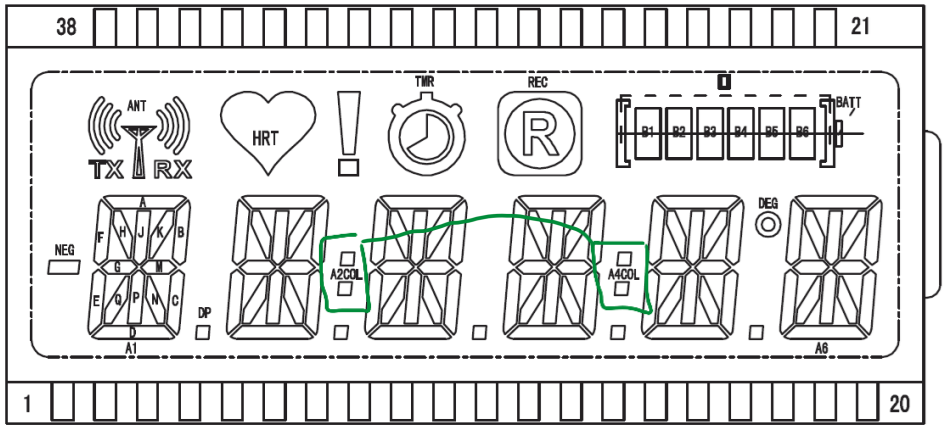


Multiplexing in LCD

- Pins

PIN	COM3	COM2	COM1	COM0
1	A1E	A1F	A1G	A1M
2	A1A	A1B	A1C	A1D
3	A1Q	NEG	A1N	A1DP
4	A1H	A1J	A1K	A1P
5	A2E	A2F	A2G	A2M
6	A2A	A2B	A2C	A2D
7	A2Q	A2COL	A2N	A2DP
8	A2H	A2J	A2K	A2P
9	A3R	A3F	A3G	A3M
10	A3A	A3B	A3C	A3D
11	A3Q	ANT	A3N	A3DP
12	A3H	A3J	A3K	A3P
13	A4R	A4F	A4G	A4M
14	A4A	A4B	A4C	A4D
15	A4Q	A4COL	A4N	A4DP
16	A4H	A4J	A4K	A4P
17	A5E	A5F	A5G	A5M
18	A5A	A5B	A5C	A5D
19	A5Q	DEG	A5N	A5DP
20	A5H	A5J	A5K	A5P

For timer (RT)



LCD to MSP Mapping

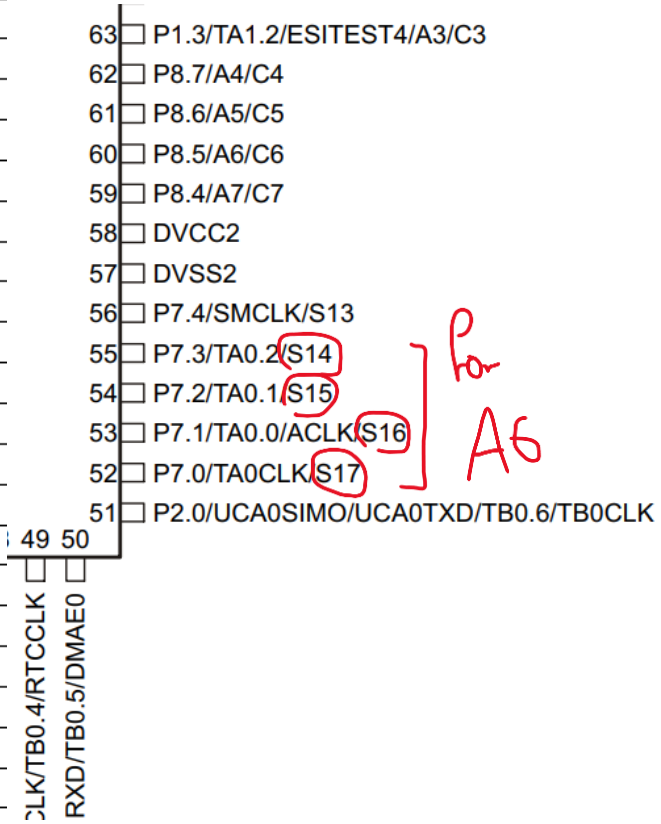
- Pins

LCDMEM	Port Pin	FR6989 Pin	LCD Pin	COM3	COM2	COM1	COM0	Port Pin	FR6989 Pin	LCD Pin	COM3	COM2	COM1	COM0
LCDM22	P2.4	S43						P2.5	S42					
LCDM21	P2.6	S41						P2.7	S40					
LCDM20	P10.2	S39	16	A4H	A4J	A4K	A4P	P5.0	S38	15	A4Q	A4COL	A4N	A4DP
LCDM19	P5.1	S37	14	A4A	A4B	A4C	A4D	P5.2	S36	13	A4R	A4F	A4G	A4M
LCDM18	P5.3	S35	34	B5	B3	B1	[]	P3.0	S34					
LCDM17	P3.1	S33						P3.2	S32					
LCDM16	P6.7	S31	20	A5H	A5J	A5K	A5P	P7.5	S30	19	A5Q	DEG	A5N	A5DP
LCDM15	P7.6	S29	18	A5A	A5B	A5C	A5D	P10.1	S28	17	A5E	A5F	A5G	A5M
LCDM14	P7.7	S27	33	B6	B4	B2	BATT	P3.3	S26					
LCDM13	P3.4	S25						P3.5	S24					
LCDM12	P3.6	S23						P3.7	S22					
LCDM11	P8.0	S21	4	A1H	A1J	A1K	A1P	P8.1	S20	3	A1Q	NEG	A1N	A1DP
LCDM10	P8.2	S19	2	A1A	A1B	A1C	A1D	P8.3	S18	1	A1E	A1F	A1G	A1M
LCDM9	P7.0	S17	38	A6H	A6J	A6K	A6P	P7.1	S16	37	A6Q	TX	A6N	RX
LCDM8	P7.2	S15	36	A6A	A6B	A6C	A6D	P7.3	S14	35	A6E	A6F	A6G	A6M
LCDM7	P7.4	S13	8	A2H	A2J	A2K	A2P	P5.4	S12	7	A2Q	A2COL	A2N	A2DP
LCDM6	P5.5	S11	6	A2A	A2B	A2C	A2D	P5.6	S10	5	A2E	A2F	A2G	A2M
LCDM5	P5.7	S9	12	A3H	A3J	A3K	A3P	P4.4	S8	11	A3Q	ANT	A3N	A3DP
LCDM4	P4.5	S7	10	A3A	A3B	A3C	A3D	P4.6	S6	9	A3R	A3F	A3G	A3M
LCDM3	P4.7	S5						P10.0	S4	32	TMR	HRT	REC	!
LCDM2	P4.0	S3						P4.1	S2					
LCDM1	P1.4	S1						P1.5	S0					

LCD to MSP Mapping

- Pins

LCDMEM	Port Pin	FR6989 Pin	LCD Pin	COM3	COM2	COM1	COM0	Port Pin	FR6989 Pin	LCD Pin	COM3	COM2	COM1	COM0
LCDM22	P2.4	S43						P2.5	S42					
LCDM21	P2.6	S41						P2.7	S40					
LCDM20	P10.2	S39	16	A4H	A4J	A4K	A4P	P5.0	S38	15	A4Q	A4COL	A4N	A4DP
LCDM19	P5.1	S37	14	A4A	A4B	A4C	A4D	P5.2	S36	13	A4R	A4F	A4G	A4M
LCDM18	P5.3	S35	34	B5	B3	B1	[]	P3.0	S34					
LCDM17	P3.1	S33						P3.2	S32					
LCDM16	P6.7	S31	20	A5H	A5J	A5K	A5P	P7.5	S30	19	A5Q	DEG	A5N	A5DP
LCDM15	P7.6	S29	18	A5A	A5B	A5C	A5D	P10.1	S28	17	A5E	A5F	A5G	A5M
LCDM14	P7.7	S27	33	B6	B4	B2	BATT	P3.3	S26					
LCDM13	P3.4	S25						P3.5	S24					
LCDM12	P3.6	S23						P3.7	S22					
LCDM11	P8.0	S21	4	A1H	A1J	A1K	A1P	P8.1	S20	3	A1Q	NEG	A1N	A1DP
LCDM10	P8.2	S19	2	A1A	A1B	A1C	A1D	P8.3	S18	1	A1E	A1F	A1G	A1M
LCDM9	P7.0	S17	38	A6H	A6J	A6K	A6P	P7.1	S16	37	A6Q	TX	A6N	RX
LCDM8	P7.2	S15	36	A6A	A6B	A6C	A6D	P7.3	S14	35	A6E	A6F	A6G	A6M
LCDM7	P7.4	S13	8	A2H	A2J	A2K	A2P	P5.4	S12	7	A2Q	A2COL	A2N	A2DP
LCDM6	P5.5	S11	6	A2A	A2B	A2C	A2D	P5.6	S10	5	A2E	A2F	A2G	A2M
LCDM5	P5.7	S9	12	A3H	A3J	A3K	A3P	P4.4	S8	11	A3Q	ANT	A3N	A3DP
LCDM4	P4.5	S7	10	A3A	A3B	A3C	A3D	P4.6	S6	9	A3R	A3F	A3G	A3M
LCDM3	P4.7	S5						P10.0	S4	32	TMR	HRT	REC	!
LCDM2	P4.0	S3						P4.1	S2					
LCDM1	P1.4	S1						P1.5	S0					



LCD_C: LCD Controller - Registers

- LCDCPCTL0 to LCDCPCTLx – LCD_C port control registers
  - x could be 1 or 3 on MSP430
- The required signals (Sx) must be enabled, so that they generate the required LCD signals.
- LCDCPCTLx is a 16-bit register

for x=1 → 32 bits of configuration

for x=3 → 64 bits of configuration

  - They are used to enable the LCD signals.

- LCDCPCTL0 – S0 to S15
- LCDCPCTL1 – S16 to S31
- LCDCPCTL2 – S32 to S47
- LCDCPCTL3 – S48 to S53

LCDDIVx					*LCDPREx*			*LCDSEL*	*rsvd.*	*LCDMXx*			*LCDSON*	*LCDLP*	*LCDON*
15	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

```
LCDCPCTL0 = 0xFFD0;
LCDCPCTL1 = 0xF83F;
LCDCPCTL2 = 0x05F8;
LCDCPCTL3 = 0x0000;
```

LCD_C: LCD Controller - Registers

- LCDCPCTL0 – LCD Controller Register 0

LCDDIVx					LCDPREx			LCDSEL	rsvd.	LCDMXx			LCDSON	LCDLP	LCDON
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

  - LCDDIVx – LCD frequency divider
  - LCDPREx – LCD frequency pre-scaler
  - LCDSEL – LCD clock source select
  - LCDMXx – LCD mux mode
  - LCDSON – Turn on/off LCD segments
  - LCDLP – LCD low power mode
  - LCDON – Turn on/off LCD module

LCD_C: LCD Controller - Registers

- LCDCPCTL0 – LCD Controller Register 0

LCDDIVx					LCDPREx			LCDSEL	rsvd.	LCDMXx			LCDSON	LCDLP	LCDON
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- LCDDIVx – LCD frequency divider
- LCDPREx – LCD frequency pre-scaler
- LCDSEL – LCD clock source select
- LCDMXx – LCD mux mode
- LCDSON – Turn on/off LCD segments
- LCDLP – LCD low power mode
- LCDON – Turn on/off LCD module

LCDCCTL0 sets the LCD frequency >> $f_{LCD} = \frac{f_{clk}}{2^{LCDPREx}(LCDDIVx+1)}$

LCD_C: LCD Controller - Registers

- LCDCPCTL0 – LCD Controller Register 0

LCDDIVx					LCDPREx			LCDSEL	rsvd.	LCDMXx			LCDSON	LCDLP	LCDON
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- LCDDIVx – LCD frequency divider
- LCDPREx – LCD frequency pre-scaler
- LCDSEL – LCD clock source select
- LCDMXx – LCD mux mode
- LCDSON – Turn on/off LCD segments
- LCDLP – LCD low power mode
- LCDON – Turn on/off LCD module

LCDCCTL0 sets the LCD frequency >> $f_{LCD} = \frac{f_{clk}}{2^{LCDPREx}(LCDDIVx+1)}$

We have two frequency definition here in LCD

Frame frequency: *Number of times per second that the LCD refreshes its entire display.*

LCD frequency: *Number of times per second that the individual segments (or pixels) of the LCD are driven.*

f_{LCD} will be calculated from f_{frame}!

LCD_C: LCD Controller - Registers

- LCDCPCTL0 – LCD Controller Register 0

LCDDIVx					LCDPREx			LCDSEL	rsvd.	LCDMXx			LCDSON	LCDLP	LCDON
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- LCDDIVx – LCD frequency divider
- LCDPREx – LCD frequency pre-scaler
- LCDSEL – LCD clock source select
- LCDMXx – LCD mux mode
- LCDSON – Turn on/off LCD segments
- LCDLP – LCD low power mode
- LCDON – Turn on/off LCD module

LCDCPCTL0 sets the LCD frequency >> $f_{LCD} = \frac{f_{clk}}{2^{LCDPREx}(LCDDIVx+1)}$

We have two frequency definition here in LCD

Frame frequency: *Number of times per second that the LCD refreshes its entire display.*

LCD frequency: *Number of times per second that the individual segments (or pixels) of the LCD are driven.*

1 KHz ← $f_{LCD} = 2 \cdot LCDMXx \cdot f_{frame}$ → 128 Hz

LCD_C: LCD Controller - Registers

- LCDCPCTL0 – LCD Controller Register 0

LCDDIVx					LCDPREx			LCDSEL	rsvd.	LCDMXx			LCDSON	LCDLP	LCDON
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- LCDDIVx – LCD frequency divider
- LCDPREx – LCD frequency pre-scaler
- LCDSEL – LCD clock source select
- LCDMXx – LCD mux mode
- LCDSON – Turn on/off LCD segments
- LCDLP – LCD low power mode
- LCDON – Turn on/off LCD module

LCDCCTL0 sets the LCD frequency >> $f_{LCD} = \frac{f_{clk}}{2^{LCDPREx}(LCDDIVx+1)}$

(Q) The f_{clk} is 32768 Hz. Select the appropriate LCDPREx and LCDDIVx, to have f_{LCD} = 1KHz.

We have two frequency definition here in LCD

Frame frequency: *Number of times per second that the LCD refreshes its entire display.*

LCD frequency: *Number of times per second that the individual segments (or pixels) of the LCD are driven.*

1 KHz ← $f_{LCD} = 2 \cdot LCDMXx \cdot f_{frame}$ → 128 Hz

LCD_C: LCD Controller - Registers

- **LCDCPCTL0 – LCD Controller Register 0**

LCDDIVx					LCDPREx			LCDSEL	rsvd.	LCDMXx			LCDSON	LCDLP	LCDON
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- **LCDDIVx – LCD frequency divider**
- **LCDPREx – LCD frequency pre-scaler**
- **LCDSEL – LCD clock source select**
- **LCDMXx – LCD mux mode**
- **LCDSON – Turn on/off LCD segments**
- **LCDLP – LCD low power mode**
- **LCDON – Turn on/off LCD module**

LCDCCTL0 sets the LCD frequency >> $f_{LCD} = \frac{f_{clk}}{2^{LCDPREx}(LCDDIVx+1)}$

(Q) The f_{clk} is 32768 Hz. Select the appropriate LCDPREx and LCDDIVx, to have f_{LCD} = 1KHz.

(A) $2^{LCDPREx}.(LCDDIVx+1) = f_{clk}/f_{LCD} \rightarrow (PREx,DIVx) = (0, 31)$

We have two frequency definition here in LCD

Frame frequency: *Number of times per second that the LCD hardware refreshes the entire display.*

LCD frequency: *Number of times per second that the individual segments (or pixels) of the LCD are driven.*

1 KHz ← $f_{LCD} = 2 \cdot LCDMXx \cdot f_{frame}$ → 128 Hz

LCD_C: LCD Controller - Registers

- LCDCPCTL0 – LCD Controller Register 0

LCDDIVx					LCDPREx			LCDSEL	rsvd.	LCDMXx			LCDSON	LCDLP	LCDON
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- LCDDIVx – LCD frequency divider
- LCDPREx – LCD frequency pre-scaler
- LCDSEL – LCD clock source select
- LCDMXx – LCD mux mode
- LCDSON – Turn on/off LCD segments
- LCDLP – LCD low power mode
- LCDON – Turn on/off LCD module

LCDCCTL0 sets the LCD frequency >> $f_{LCD} = \frac{f_{clk}}{2^{LCDPREx}(LCDDIVx+1)}$

(Q) The f_{clk} is 32768 Hz. Select the appropriate LCDPREx and LCDDIVx, to have f_{LCD} = 1KHz. Is it the only valid answer for it?

We have two frequency definition here in LCD

Frame frequency: *Number of times per second that the LCD refreshes its entire display.*

LCD frequency: *Number of times per second that the individual segments (or pixels) of the LCD are driven.*

1 KHz ← $f_{LCD} = 2 \cdot LCDMXx \cdot f_{frame}$ → 128 Hz

LCD_C: LCD Controller - Registers

- **LCDCPCTL0 – LCD Controller Register 0**

LCDDIVx					LCDPREx			LCDSEL	rsvd.	LCDMXx			LCDSON	LCDLP	LCDON
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- LCDDIVx – LCD frequency divider
- LCDPREx – LCD frequency pre-scaler
- LCDSEL – LCD clock source select
- LCDMXx – LCD mux mode
- LCDSON – Turn on/off LCD segments
- LCDLP – LCD low power mode
- LCDON – Turn on/off LCD module

LCDCCTL0 sets the LCD frequency >> $f_{LCD} = \frac{f_{clk}}{2^{LCDPREx}(LCDDIVx+1)}$

(Q) The f_{clk} is 32768 Hz. Select the appropriate LCDPREx and LCDDIVx, to have f_{LCD} = 1KHz. Is it the only valid answer for it?

(A) $2^{LCDPREx}.(LCDDIVx+1) = f_{clk}/f_{LCD}$ → (PREx,DIVx) = (1, 15) or (2,7) or (3, 3) or (4, 1) or (5, 0)

We have two frequency definition here in LCD

Frame frequency: *Number of times per second that the LCD refreshes the display.*

LCD frequency: *Number of times per second that the individual segments (or pixels) of the LCD are driven.*

1 KHz ← $f_{LCD} = 2 \cdot LCDMXx \cdot f_{frame}$ → 128 Hz

LCD to MSP Mapping

- LCDMx – LCD_C memory registers
  - is designed to drive LCD displays.

memory locations that store the segment data for the LCD display.

Whatever is written in these registers is immediately displayed on the LCD.

The voltage levels and signal generation is taken care of by the LCD_C controller based on the control register we saw earlier.

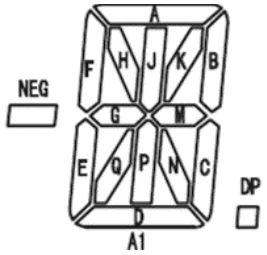
LCDMEM	Port Pin	FR6989 Pin	LCD Pin	COM3	COM2	COM1	COM0	Port Pin	FR6989 Pin	LCD Pin	COM3	COM2	COM1	COM0
LCDM22	P2.4	S43						P2.5	S42					
LCDM21	P2.6	S41						P2.7	S40					
LCDM20	P10.2	S39	16	A4H	A4J	A4K	A4P	P5.0	S38	15	A4Q	A4COL	A4N	A4DP
LCDM19	P5.1	S37	14	A4A	A4B	A4C	A4D	P5.2	S36	13	A4R	A4F	A4G	A4M
LCDM18	P5.3	S35	34	B5	B3	B1	[]	P3.0	S34					
LCDM17	P3.1	S33						P3.2	S32					
LCDM16	P6.7	S31	20	A5H	A5J	A5K	A5P	P7.5	S30	19	A5Q	DEG	A5N	A5DP
LCDM15	P7.6	S29	18	A5A	A5B	A5C	A5D	P10.1	S28	17	A5E	A5F	A5G	A5M
LCDM14	P7.7	S27	33	B6	B4	B2	BATT	P3.3	S26					
LCDM13	P3.4	S25						P3.5	S24					
LCDM12	P3.6	S23						P3.7	S22					
LCDM11	P8.0	S21	4	A1H	A1J	A1K	A1P	P8.1	S20	3	A1Q	NEG	A1N	A1DP
LCDM10	P8.2	S19	2	A1A	A1B	A1C	A1D	P8.3	S18	1	A1E	A1F	A1G	A1M
LCDM9	P7.0	S17	38	A6H	A6J	A6K	A6P	P7.1	S16	37	A6Q	TX	A6N	RX
LCDM8	P7.2	S15	36	A6A	A6B	A6C	A6D	P7.3	S14	35	A6E	A6F	A6G	A6M
LCDM7	P7.4	S13	8	A2H	A2J	A2K	A2P	P5.4	S12	7	A2Q	A2COL	A2N	A2DP
LCDM6	P5.5	S11	6	A2A	A2B	A2C	A2D	P5.6	S10	5	A2E	A2F	A2G	A2M
LCDM5	P5.7	S9	12	A3H	A3J	A3K	A3P	P4.4	S8	11	A3Q	ANT	A3N	A3DP
LCDM4	P4.5	S7	10	A3A	A3B	A3C	A3D	P4.6	S6	9	A3R	A3F	A3G	A3M
LCDM3	P4.7	S5						P10.0	S4	32	TMR	HRT	REC	!
LCDM2	P4.0	S3						P4.1	S2					
LCDM1	P1.4	S1						P1.5	S0					

LCD to MSP Mapping

- LCDMx – LCD_C memory registers
  - is designed to drive LCD displays.

Example1. we want to write "8" on A1.

LCDMEM	Port Pin	FR6989 Pin	LCD Pin	COM3	COM2	COM1	COM0	Port Pin	FR6989 Pin	LCD Pin	COM3	COM2	COM1	COM0
LCDM22	P2.4	S43						P2.5	S42					
LCDM21	P2.6	S41						P2.7	S40					
LCDM20	P10.2	S39	16	A4H	A4J	A4K	A4P	P5.0	S38	15	A4Q	A4COL	A4N	A4DP
LCDM19	P5.1	S37	14	A4A	A4B	A4C	A4D	P5.2	S36	13	A4R	A4F	A4G	A4M
LCDM18	P5.3	S35	34	B5	B3	B1	[]	P3.0	S34					
LCDM17	P3.1	S33						P3.2	S32					
LCDM16	P6.7	S31	20	A5H	A5J	A5K	A5P	P7.5	S30	19	A5Q	DEG	A5N	A5DP
LCDM15	P7.6	S29	18	A5A	A5B	A5C	A5D	P10.1	S28	17	A5E	A5F	A5G	A5M
LCDM14	P7.7	S27	33	B6	B4	B2	BATT	P3.3	S26					
LCDM13	P3.4	S25						P3.5	S24					
LCDM12	P3.6	S23						P3.7	S22					
LCDM11	P8.0	S21	4	A1H	A1J	A1K	A1P	P8.1	S20	3	A1Q	NEG	A1N	A1DP
LCDM10	P8.2	S19	2	A1A	A1B	A1C	A1D	P8.3	S18	1	A1E	A1F	A1G	A1M
LCDM9	P7.0	S17	38	A6H	A6J	A6K	A6P	P7.1	S16	37	A6Q	TX	A6N	RX
LCDM8	P7.2	S15	36	A6A	A6B	A6C	A6D	P7.3	S14	35	A6E	A6F	A6G	A6M
LCDM7	P7.4	S13	8	A2H	A2J	A2K	A2P	P5.4	S12	7	A2Q	A2COL	A2N	A2DP
LCDM6	P5.5	S11	6	A2A	A2B	A2C	A2D	P5.6	S10	5	A2E	A2F	A2G	A2M
LCDM5	P5.7	S9	12	A3H	A3J	A3K	A3P	P4.4	S8	11	A3Q	ANT	A3N	A3DP
LCDM4	P4.5	S7	10	A3A	A3B	A3C	A3D	P4.6	S6	9	A3R	A3F	A3G	A3M
LCDM3	P4.7	S5						P10.0	S4	32	TMR	HRT	REC	!
LCDM2	P4.0	S3						P4.1	S2					
LCDM1	P1.4	S1						P1.5	S0					



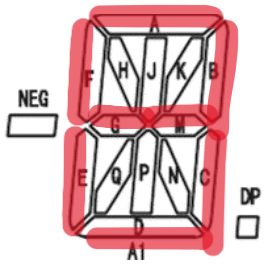
LCD to MSP Mapping

- LCDMx – LCD_C memory registers
  - is designed to drive LCD displays.

Example1. we want to write "8" on A1.

For "8", the following segments must be ON:
A,B,C,D,E,F,G,M

LCDMEM	Port Pin	FR6989 Pin	LCD Pin	COM3	COM2	COM1	COM0	Port Pin	FR6989 Pin	LCD Pin	COM3	COM2	COM1	COM0
LCDM22	P2.4	S43						P2.5	S42					
LCDM21	P2.6	S41						P2.7	S40					
LCDM20	P10.2	S39	16	A4H	A4J	A4K	A4P	P5.0	S38	15	A4Q	A4COL	A4N	A4DP
LCDM19	P5.1	S37	14	A4A	A4B	A4C	A4D	P5.2	S36	13	A4R	A4F	A4G	A4M
LCDM18	P5.3	S35	34	B5	B3	B1	[]	P3.0	S34					
LCDM17	P3.1	S33						P3.2	S32					
LCDM16	P6.7	S31	20	A5H	A5J	A5K	A5P	P7.5	S30	19	A5Q	DEG	A5N	A5DP
LCDM15	P7.6	S29	18	A5A	A5B	A5C	A5D	P10.1	S28	17	A5E	A5F	A5G	A5M
LCDM14	P7.7	S27	33	B6	B4	B2	BATT	P3.3	S26					
LCDM13	P3.4	S25						P3.5	S24					
LCDM12	P3.6	S23						P3.7	S22					
LCDM11	P8.0	S21	4	A1H	A1J	A1K	A1P	P8.1	S20	3	A1Q	NEG	A1N	A1DP
LCDM10	P8.2	S19	2	A1A	A1B	A1C	A1D	P8.3	S18	1	A1E	A1F	A1G	A1M
LCDM9	P7.0	S17	38	A6H	A6J	A6K	A6P	P7.1	S16	37	A6Q	TX	A6N	RX
LCDM8	P7.2	S15	36	A6A	A6B	A6C	A6D	P7.3	S14	35	A6E	A6F	A6G	A6M
LCDM7	P7.4	S13	8	A2H	A2J	A2K	A2P	P5.4	S12	7	A2Q	A2COL	A2N	A2DP
LCDM6	P5.5	S11	6	A2A	A2B	A2C	A2D	P5.6	S10	5	A2E	A2F	A2G	A2M
LCDM5	P5.7	S9	12	A3H	A3J	A3K	A3P	P4.4	S8	11	A3Q	ANT	A3N	A3DP
LCDM4	P4.5	S7	10	A3A	A3B	A3C	A3D	P4.6	S6	9	A3R	A3F	A3G	A3M
LCDM3	P4.7	S5						P10.0	S4	32	TMR	HRT	REC	!
LCDM2	P4.0	S3						P4.1	S2					
LCDM1	P1.4	S1						P1.5	S0					



LCD to MSP Mapping

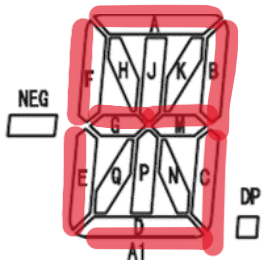
- LCDMx – LCD_C memory registers
  - is designed to drive LCD displays.

Example1. we want to write "8" on A1.

For "8", the following segments must be ON:
A,B,C,D,E,F,G,M

LCDM10 = 0xFF;
LCDM11 = 0x00;

LCDMEM	Port Pin	FR6989 Pin	LCD Pin	COM3	COM2	COM1	COM0	Port Pin	FR6989 Pin	LCD Pin	COM3	COM2	COM1	COM0
LCDM22	P2.4	S43						P2.5	S42					
LCDM21	P2.6	S41						P2.7	S40					
LCDM20	P10.2	S39	16	A4H	A4J	A4K	A4P	P5.0	S38	15	A4Q	A4COL	A4N	A4DP
LCDM19	P5.1	S37	14	A4A	A4B	A4C	A4D	P5.2	S36	13	A4R	A4F	A4G	A4M
LCDM18	P5.3	S35	34	B5	B3	B1	[]	P3.0	S34					
LCDM17	P3.1	S33						P3.2	S32					
LCDM16	P6.7	S31	20	A5H	A5J	A5K	A5P	P7.5	S30	19	A5Q	DEG	A5N	A5DP
LCDM15	P7.6	S29	18	A5A	A5B	A5C	A5D	P10.1	S28	17	A5E	A5F	A5G	A5M
LCDM14	P7.7	S27	33	B6	B4	B2	BATT	P3.3	S26					
LCDM13	P3.4	S25						P3.5	S24					
LCDM12	P3.6	S23						P3.7	S22					
LCDM11	P8.0	S21	4	A1H	A1J	A1K	A1P	P8.1	S20	3	A1Q	NEG	A1N	A1DP
LCDM10	P8.2	S19	2	A1A	A1B	A1C	A1D	P8.3	S18	1	A1E	A1F	A1G	A1M
LCDM9	P7.0	S17	38	A6H	A6J	A6K	A6P	P7.1	S16	37	A6Q	TX	A6N	RX
LCDM8	P7.2	S15	36	A6A	A6B	A6C	A6D	P7.3	S14	35	A6E	A6F	A6G	A6M
LCDM7	P7.4	S13	8	A2H	A2J	A2K	A2P	P5.4	S12	7	A2Q	A2COL	A2N	A2DP
LCDM6	P5.5	S11	6	A2A	A2B	A2C	A2D	P5.6	S10	5	A2E	A2F	A2G	A2M
LCDM5	P5.7	S9	12	A3H	A3J	A3K	A3P	P4.4	S8	11	A3Q	ANT	A3N	A3DP
LCDM4	P4.5	S7	10	A3A	A3B	A3C	A3D	P4.6	S6	9	A3R	A3F	A3G	A3M
LCDM3	P4.7	S5						P10.0	S4	32	TMR	HRT	REC	!
LCDM2	P4.0	S3						P4.1	S2					
LCDM1	P1.4	S1						P1.5	S0					

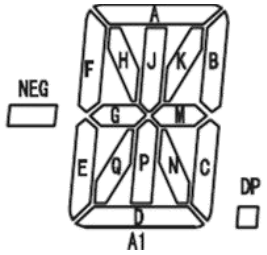


LCD to MSP Mapping

- LCDMx – LCD_C memory registers
  - is designed to drive LCD displays.

Example1. we want to write "X" on A6.

LCDMEM	Port Pin	FR6989 Pin	LCD Pin	COM3	COM2	COM1	COM0	Port Pin	FR6989 Pin	LCD Pin	COM3	COM2	COM1	COM0
LCDM22	P2.4	S43						P2.5	S42					
LCDM21	P2.6	S41						P2.7	S40					
LCDM20	P10.2	S39	16	A4H	A4J	A4K	A4P	P5.0	S38	15	A4Q	A4COL	A4N	A4DP
LCDM19	P5.1	S37	14	A4A	A4B	A4C	A4D	P5.2	S36	13	A4R	A4F	A4G	A4M
LCDM18	P5.3	S35	34	B5	B3	B1	[]	P3.0	S34					
LCDM17	P3.1	S33						P3.2	S32					
LCDM16	P6.7	S31	20	A5H	A5J	A5K	A5P	P7.5	S30	19	A5Q	DEG	A5N	A5DP
LCDM15	P7.6	S29	18	A5A	A5B	A5C	A5D	P10.1	S28	17	A5E	A5F	A5G	A5M
LCDM14	P7.7	S27	33	B6	B4	B2	BATT	P3.3	S26					
LCDM13	P3.4	S25						P3.5	S24					
LCDM12	P3.6	S23						P3.7	S22					
LCDM11	P8.0	S21	4	A1H	A1J	A1K	A1P	P8.1	S20	3	A1Q	NEG	A1N	A1DP
LCDM10	P8.2	S19	2	A1A	A1B	A1C	A1D	P8.3	S18	1	A1E	A1F	A1G	A1M
LCDM9	P7.0	S17	38	A6H	A6J	A6K	A6P	P7.1	S16	37	A6Q	TX	A6N	RX
LCDM8	P7.2	S15	36	A6A	A6B	A6C	A6D	P7.3	S14	35	A6E	A6F	A6G	A6M
LCDM7	P7.4	S13	8	A2H	A2J	A2K	A2P	P5.4	S12	7	A2Q	A2COL	A2N	A2DP
LCDM6	P5.5	S11	6	A2A	A2B	A2C	A2D	P5.6	S10	5	A2E	A2F	A2G	A2M
LCDM5	P5.7	S9	12	A3H	A3J	A3K	A3P	P4.4	S8	11	A3Q	ANT	A3N	A3DP
LCDM4	P4.5	S7	10	A3A	A3B	A3C	A3D	P4.6	S6	9	A3R	A3F	A3G	A3M
LCDM3	P4.7	S5						P10.0	S4	32	TMR	HRT	REC	!
LCDM2	P4.0	S3						P4.1	S2					
LCDM1	P1.4	S1						P1.5	S0					



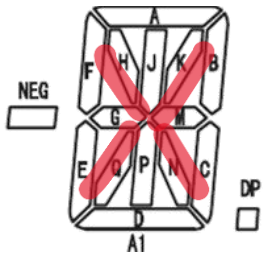
LCD to MSP Mapping

- LCDMx – LCD_C memory registers
  - is designed to drive LCD displays.

Example1. we want to write "X" on A6.

For "X", the following segments must be ON:
H,K,Q,N

LCDMEM	Port Pin	FR6989 Pin	LCD Pin	COM3	COM2	COM1	COM0	Port Pin	FR6989 Pin	LCD Pin	COM3	COM2	COM1	COM0
LCDM22	P2.4	S43						P2.5	S42					
LCDM21	P2.6	S41						P2.7	S40					
LCDM20	P10.2	S39	16	A4H	A4J	A4K	A4P	P5.0	S38	15	A4Q	A4COL	A4N	A4DP
LCDM19	P5.1	S37	14	A4A	A4B	A4C	A4D	P5.2	S36	13	A4R	A4F	A4G	A4M
LCDM18	P5.3	S35	34	B5	B3	B1	[]	P3.0	S34					
LCDM17	P3.1	S33						P3.2	S32					
LCDM16	P6.7	S31	20	A5H	A5J	A5K	A5P	P7.5	S30	19	A5Q	DEG	A5N	A5DP
LCDM15	P7.6	S29	18	A5A	A5B	A5C	A5D	P10.1	S28	17	A5E	A5F	A5G	A5M
LCDM14	P7.7	S27	33	B6	B4	B2	BATT	P3.3	S26					
LCDM13	P3.4	S25						P3.5	S24					
LCDM12	P3.6	S23						P3.7	S22					
LCDM11	P8.0	S21	4	A1H	A1J	A1K	A1P	P8.1	S20	3	A1Q	NEG	A1N	A1DP
LCDM10	P8.2	S19	2	A1A	A1B	A1C	A1D	P8.3	S18	1	A1E	A1F	A1G	A1M
LCDM9	P7.0	S17	38	A6H	A6J	A6K	A6P	P7.1	S16	37	A6Q	TX	A6N	RX
LCDM8	P7.2	S15	36	A6A	A6B	A6C	A6D	P7.3	S14	35	A6E	A6F	A6G	A6M
LCDM7	P7.4	S13	8	A2H	A2J	A2K	A2P	P5.4	S12	7	A2Q	A2COL	A2N	A2DP
LCDM6	P5.5	S11	6	A2A	A2B	A2C	A2D	P5.6	S10	5	A2E	A2F	A2G	A2M
LCDM5	P5.7	S9	12	A3H	A3J	A3K	A3P	P4.4	S8	11	A3Q	ANT	A3N	A3DP
LCDM4	P4.5	S7	10	A3A	A3B	A3C	A3D	P4.6	S6	9	A3R	A3F	A3G	A3M
LCDM3	P4.7	S5						P10.0	S4	32	TMR	HRT	REC	!
LCDM2	P4.0	S3						P4.1	S2					
LCDM1	P1.4	S1						P1.5	S0					



LCD to MSP Mapping

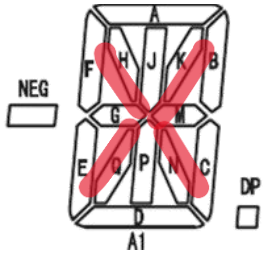
- LCDMx – LCD_C memory registers
  - is designed to drive LCD displays.

Example1. we want to write "X" on A6.

For "X", the following segments must be ON:
H,K,Q,N

LCDM9 = 0xAA;
LCDM11 = 0x00;

LCDMEM	Port Pin	FR6989 Pin	LCD Pin	COM3	COM2	COM1	COM0	Port Pin	FR6989 Pin	LCD Pin	COM3	COM2	COM1	COM0
LCDM22	P2.4	S43						P2.5	S42					
LCDM21	P2.6	S41						P2.7	S40					
LCDM20	P10.2	S39	16	A4H	A4J	A4K	A4P	P5.0	S38	15	A4Q	A4COL	A4N	A4DP
LCDM19	P5.1	S37	14	A4A	A4B	A4C	A4D	P5.2	S36	13	A4R	A4F	A4G	A4M
LCDM18	P5.3	S35	34	B5	B3	B1	[]	P3.0	S34					
LCDM17	P3.1	S33						P3.2	S32					
LCDM16	P6.7	S31	20	A5H	A5J	A5K	A5P	P7.5	S30	19	A5Q	DEG	A5N	A5DP
LCDM15	P7.6	S29	18	A5A	A5B	A5C	A5D	P10.1	S28	17	A5E	A5F	A5G	A5M
LCDM14	P7.7	S27	33	B6	B4	B2	BATT	P3.3	S26					
LCDM13	P3.4	S25						P3.5	S24					
LCDM12	P3.6	S23						P3.7	S22					
LCDM11	P8.0	S21	4	A1H	A1J	A1K	A1P	P8.1	S20	3	A1Q	NEG	A1N	A1DP
LCDM10	P8.2	S19	2	A1A	A1B	A1C	A1D	P8.3	S18	1	A1E	A1F	A1G	A1M
LCDM9	P7.0	S17	38	A6H	A6J	A6K	A6P	P7.1	S16	37	A6Q	TX	A6N	RX
LCDM8	P7.2	S15	36	A6A	A6B	A6C	A6D	P7.3	S14	35	A6E	A6F	A6G	A6M
LCDM7	P7.4	S13	8	A2H	A2J	A2K	A2P	P5.4	S12	7	A2Q	A2COL	A2N	A2DP
LCDM6	P5.5	S11	6	A2A	A2B	A2C	A2D	P5.6	S10	5	A2E	A2F	A2G	A2M
LCDM5	P5.7	S9	12	A3H	A3J	A3K	A3P	P4.4	S8	11	A3Q	ANT	A3N	A3DP
LCDM4	P4.5	S7	10	A3A	A3B	A3C	A3D	P4.6	S6	9	A3R	A3F	A3G	A3M
LCDM3	P4.7	S5						P10.0	S4	32	TMR	HRT	REC	!
LCDM2	P4.0	S3						P4.1	S2					
LCDM1	P1.4	S1						P1.5	S0					

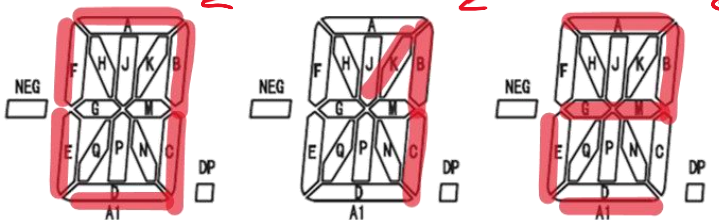


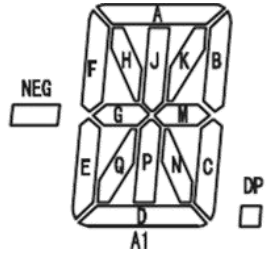
Mapping of Characters and Integers

- How we define these encoding values in code?

- A pre-defined array of numbers

```
unsigned char LCD_Shapes[10] = {0xFC, 0x60, 0xDB, ...};
```





- What about Characters?
  - Two 8-bit into LCDMx (e.g., LCDM10 and LCDM11 → A1)

A1H	A1J	A1K	A1P
A1A	A1B	A1C	A1D

A1Q	NEG	A1N	A1DP
A1E	A1F	A1G	A1M

LCD_C: LCD Controller - Registers

- LCDCMEMCTL – LCD_C memory control register

rsvd.														LCDCLRM	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- LCDCLRM = 1b
  - Clears all LCD memory registers LCDMx.

LCD_C Programming

- To initialize the LCD_C peripheral, configure the following registers:
  - LCDCVCTL LCD_C voltage control
  - LCDCCTL0 LCD_C control register 0
  - LCDCMEMCTL LCD_C memory control
  - LCDCPCTL0 to LCDCPCTL3 LCD_C port control registers

LCD_C Programming

- To initialize the LCD_C peripheral, configure the following registers:
  - LCDCVCTL LCD_C voltage control — *Controls the voltage levels of the signals for LCD!*
  - LCDCCTL0 LCD_C control register 0 — *Clock source, clock frequency and more!*
  - LCDCMEMCTL LCD_C memory control — *Configures the MSP430 pins with LCD pins!*
  - LCDCPCTL0 to LCDCPCTL3 LCD_C port control registers — *Clears the LCD memory!*

LCD_C Programming

- To initialize the LCD_C peripheral, configure the following registers:
  - LCDCVCTL LCD_C voltage control — *Controls the voltage levels of the signals for LCD!*
  - LCDCCTL0 LCD_C control register 0 — *Clock source, clock frequency and more!*
  - LCDCMEMCTL LCD_C memory control — *Configures the MSP430 pins with LCD pins!*
  - LCDCPCTL0 to LCDCPCTL3 LCD_C port control registers — *Clears the LCD memory!*
- Once the LCD_C peripheral is initialized,
  - optionally, clear the display using the LCDCMEMCTL LCD_C memory control register
  - Set the required values to LCDMx LCD_C memory registers
  - Changes made to LCDMx registers will be reflected immediately on the LCD display.
    - e.g. LCDM10 = 0xFC; // to display “0”

Example 1 – 430 on LCD

- **Demonstrate 430 on LCD. Clock is ACLK set to 32KHz**

```
// Sample code that prints 430 on the LCD monitor
#include <msp430fr6989.h>
#define redLED BIT0                         // Red at P1.0
#define greenLED BIT7                       // Green at P9.7
void Initialize_LCD();

const unsigned char LCD_Shapes[10] =
      {0xFC, 0x60, 0xDB, 0xF3, 0x67, …};     // The array with shapes of digits 0, 1, 2, …


int main(void) {
   volatile unsigned int n;
   WDTCTL = WDTPW | WDTHOLD;                // Stop WDT
   PM5CTL0 &= ~LOCKLPM5;                    // Enable GPIO pins
   P1DIR |= redLED;                         // Pins as output
   P1OUT |= redLED;                         // Red on
   Initialize_LCD();                        // Initializes the LCD_C module

   LCDCMEMCTL = LCDCLRM;                    // Clears all the segments

   LCDM19 = LCD_Shapes[4];                  // Display 4 on A4
   LCDM15 = LCD_Shapes[3];                  // Display 3 on A5
   LCDM8 = LCD_Shapes[0];                   // Display 0 on A6
   // Flash the red LED
   for(;;) {
                  for(n=0; n<=60000; n++) {} // Delay loop
                  P1OUT ^= redLED;
   }
   return 0;
}
```

```
// Initializes the LCD_C module
// *** Source: Function obtained from MSP430FR6989's Sample Code ***
void Initialize_LCD() {
   PJSEL0 = BIT4 | BIT5;                    // For LFXT
   LCDCPCTL0 = 0xFFD0;
   LCDCPCTL1 = 0xF83F;
   LCDCPCTL2 = 0x00F8;

   // Configure LFXT 32kHz crystal
   CSCTL0_H = CSKEY >> 8;                   // Unlock CS registers
   CSCTL4 &= ~LFXTOFF;                      // Enable LFXT
   do {
                  CSCTL5 &= ~LFXTOFFG;      // Clear LFXT fault flag
                  SFRIFG1 &= ~OFIFG;
    } while (SFRIFG1 & OFIFG);              // Test oscillator fault flag
   CSCTL0_H = 0;                            // Lock CS registers

   // ACLK, Divider = 1, Pre-divider = 16; 4-pin MUX
   LCDCCTL0 = LCDDIV__1 | LCDPRE__16 | LCD4MUX | LCDLP;
   // VLCD generated internally,
   // V2-V4 generated internally, v5 to ground
   // Set VLCD voltage to 2.60v
   // Enable charge pump and select internal reference for it
   LCDCVCTL = VLCD_1 | VLCDREF_0 | LCDCPEN;
   LCDCCPCTL = LCDCPCLKSYNC; // Clock synchronization enabled
   LCDCMEMCTL = LCDCLRM; // Clear LCD memory
   //Turn LCD on (do this at the end!)
   LCDCCTL0 |= LCDON;
   return;
}
```

Example 2

- **What is this code for?**

```
// Sample code that prints 430 on the LCD monitor
#include <msp430fr6989.h>
#define redLED BIT0                         // Red at P1.0
#define greenLED BIT7                       // Green at P9.7
void Initialize_LCD();

const unsigned char LCD_Shapes[10] =
      {0xFC, 0x60, 0xDB, 0xF3, 0x67, …};     // The array with shapes of digits 0, 1, 2, …


int main(void) {
   volatile unsigned int n, m=25212;
   WDTCTL = WDTPW | WDTHOLD;                // Stop WDT
   PM5CTL0 &= ~LOCKLPM5;                    // Enable GPIO pins
   P1DIR |= redLED;                         // Pins as output
   P1OUT |= redLED;                         // Red on
   Initialize_LCD();                        // Initializes the LCD_C module
   LCDCMEMCTL = LCDCLRM;                    // Clears all the segments
   LCDM6 = LCD_Shapes[(int)(m/10000)%10];
   LCDM4 = LCD_Shapes[(int)(m/1000)%10];
   LCDM19 = LCD_Shapes[(int)(m/100)%10];
   LCDM15 = LCD_Shapes[(int)(m/10)%10];
   LCDM8 = LCD_Shapes[(int)(m%10)];
   // Flash the red LED
   for(;;) {
                  for(n=0; n<=60000; n++) {} // Delay loop
                  P1OUT ^= redLED;
   }
   return 0;
}
```

```
// Initializes the LCD_C module
// *** Source: Function obtained from MSP430FR6989's Sample Code ***
void Initialize_LCD() {
   PJSEL0 = BIT4 | BIT5;                    // For LFXT
   LCDCPCTL0 = 0xFFD0;
   LCDCPCTL1 = 0xF83F;
   LCDCPCTL2 = 0x00F8;

   // Configure LFXT 32kHz crystal
   CSCTL0_H = CSKEY >> 8;                   // Unlock CS registers
   CSCTL4 &= ~LFXTOFF;                      // Enable LFXT
   do {
                  CSCTL5 &= ~LFXTOFFG;      // Clear LFXT fault flag
                  SFRIFG1 &= ~OFIFG;
    } while (SFRIFG1 & OFIFG);              // Test oscillator fault flag
   CSCTL0_H = 0;                            // Lock CS registers

   // ACLK, Divider = 1, Pre-divider = 16; 4-pin MUX
   LCDCCTL0 = LCDDIV__1 | LCDPRE__16 | LCD4MUX | LCDLP;
   // VLCD generated internally,
   // V2-V4 generated internally, v5 to ground
   // Set VLCD voltage to 2.60v
   // Enable charge pump and select internal reference for it
   LCDCVCTL = VLCD_1 | VLCDREF_0 | LCDCPEN;
   LCDCCPCTL = LCDCPCLKSYNC; // Clock synchronization enabled
   LCDCMEMCTL = LCDCLRM; // Clear LCD memory
   //Turn LCD on (do this at the end!)
   LCDCCTL0 |= LCDON;
   return;
}
```

Example 2 – Application?

- What is this code for?

(Q) Writing a function for lcd_write_uint16(unsigned int n)?

```
// Sample code that prints 430 on the LCD monitor
#include "msp430fr6989.h"
#define redLED BIT0                     // Red at P1.0
#define greenLED BIT7                   // Green at P9.7
void Initialize_LCD();

const unsigned char LCD_Shapes[10] =
      {0xFC, 0x60, 0xDB, 0xF3, 0x67, …};          // The array with shapes of digits 0, 1, 2, …


int main(void) {
    volatile unsigned int n, m=25212;
    WDTCTL = WDTPW | WDTHOLD;           // Stop WDT
    PM5CTL0 &= ~LOCKLPM5;               // Enable GPIO pins
    P1DIR |= redLED;                    // Pins as output
    P1OUT |= redLED;                    // Red on
    Initialize_LCD();                   // Initializes the LCD_C module
    LCDCMEMCTL = LCDCLRM;               // Clears all the segments
    LCDM6 = LCD_Shapes[(int)(m/10000)%10];
    LCDM4 = LCD_Shapes[(int)(m/1000)%10];
    LCDM19 = LCD_Shapes[(int)(m/100)%10];
    LCDM15 = LCD_Shapes[(int)(m/10)%10];
    LCDM8 = LCD_Shapes[(int)(m%10)];
    // Flash the red LED
    for(;;) {
                for(n=0; n<=60000; n++) {} // Delay loop
                P1OUT ^= redLED;
    }
    return 0;
}
```

```
// Initializes the LCD_C module
// *** Source: Modified from MSP430FR6989's Sample Code ***
void Initialize_LCD() {
  PJSEL0 = BIT4 | BIT5;                     // For LFXT
  LCDCPCTL0 = 0xFFD0;
  LCDCPCTL1 = 0xF83F;
  LCDCPCTL2 = 0x00F8;

  // Configure LFXT 32kHz crystal
  CSCTL0_H = CSKEY >> 8;                    // Unlock CS registers
  CSCTL4 &= ~LFXTOFF;                       // Enable LFXT
  do {
            CSCTL5 &= ~LFXTOFFG;            // Clear LFXT fault flag
            SFRIFG1 &= ~OFIFG;
  } while (SFRIFG1 & OFIFG);                // Test oscillator fault flag
  CSCTL0_H = 0;                             // Lock CS registers

  // ACLK, Divider = 1, Pre-divider = 16; 4-pin MUX
  LCDCCTL0 = LCDDIV__1 | LCDPRE__16 | LCD4MUX | LCDLP;
  // VLCD generated internally,
  // V2-V4 generated internally, v5 to ground
  // Set VLCD voltage to 2.60v
  // Enable charge pump and select internal reference for it
  LCDCVCTL = VLCD_1 | VLCDREF_0 | LCDCPEN;
  LCDCCPCTL = LCDCPCLKSYNC; // Clock synchronization enabled
  LCDCMEMCTL = LCDCLRM; // Clear LCD memory
  //Turn LCD on (do this at the end!)
  LCDCCTL0 |= LCDON;
  return;
}
```

Example 2 – Application?

- What is this code for?

(Q) Writing a function for lcd_write_uint16(unsigned int n)?

```
void lcd_write_uint16(unsigned int n) {
    unsigned char digits[5];          // To hold the individual digits of the number
    unsigned int i;                   // Extract individual digits from the number
    for (i = 0; i < 5; i++) {
        digits[i] = number % 10;      // Get the least significant digit
        number /= 10;                 // Remove the least significant digit
    }
```

```
// Sample code that prints 430 on the LCD monitor
#include <msp430.h>
#define redLED BIT0
#define greenLED BIT7
void Initialize_LCD();

const char LCD_Shapes[10] =
  {0xFC, 0x60, 0xDB, 0xF3, 0x67, ...};

int main(void) {
  volatile unsigned int n, m=25212;
  WDTCTL = WDTPW | WDTHOLD;
  PM5CTL0 &= ~LOCKLPM5;
  P1DIR |= redLED;
  P1OUT |= redLED;
  Initialize_LCD();
  LCDCMEMCTL = LCDCLRM;
  LCDM6 = LCD_Shapes[(int)(m/10000)%10];
  LCDM4 = LCD_Shapes[(int)(m/1000)%10];
  LCDM19 = LCD_Shapes[(int)(m/100)%10];
  LCDM15 = LCD_Shapes[(int)(m/10)%10];
  LCDM8 = LCD_Shapes[(int)(m%10)];
  // Flash the red LED
  for(;;) {
        for(n=0; n<=60000; n++) {} // Delay loop
        P1OUT ^= redLED;
  }
  return 0;
}
```

```
// Red at P1.0
// Green at P9.7

// The array with shapes of digits 0, 1, 2,

// Configure GPIO pins
// Pins as output
// Red on
// Initializes the LCD_C module
// Clears all the segments
```

```
// Initializes the LCD_C module
// *** Source: Modified from MSP430FR6989's Sample Code ***
void Initialize_LCD() {
  PJSEL0 = BIT4 | BIT5;                   // For LFXT
  LCDCPCTL0 = 0xFFD0;
  LCDCPCTL1 = 0xF83F;
  LCDCPCTL2 = 0x00F8;
  // Configure LFXT 32kHz crystal
  CSCTL0_H = CSKEY_H;                     // Unlock CS registers
  CSCTL4 &= ~LFXTOFF;                     // Enable LFXT
  do {
    CSCTL5 &= ~LFXTOFFG;                  // Clear LFXT fault flag
    SFRIFG1 &= ~OFIFG;
  } while (SFRIFG1 & OFIFG);              // Test oscillator fault flag
  CSCTL0_H = 0;                           // Lock CS registers

  // ACLK, Divider = 1, Pre-divider = 16; 4-pin MUX
  LCDCCTL0 = LCDDIV__1 | LCDPRE__16 | LCD4MUX | LCDLP;
  // VLCD generated internally,
  // V2-V4 generated internally, v5 to ground
  // Set VLCD voltage to 2.60v
  // Enable charge pump and select internal reference for it
  LCDCVCTL = VLCD_1 | VLCDREF_0 | LCDCPEN;
  LCDCCPCTL = LCDCPCLKSYNC; // Clock synchronization enabled
  LCDCMEMCTL = LCDCLRM; // Clear LCD memory
  //Turn LCD on (do this at the end!)
  LCDCCTL0 |= LCDON;
  return;
}
```

Thank You!

Questions?

Email: kamali@ucf.edu

UCF HEC 435 **(407) 823 – 0764**

https://www.ece.ucf.edu/~kamali/

HAVEN Research Group

https://haven.ece.ucf.edu/

UNIVERSITY OF CENTRAL FLORIDA