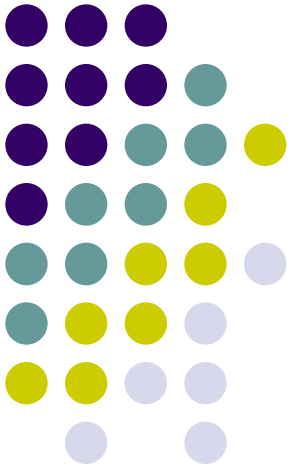
Chapter 6: Liquid Crystal Displays

EE2405

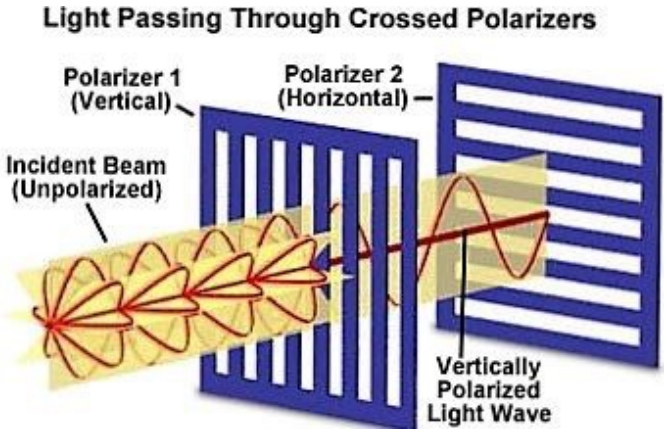
嵌入式系統與實驗

Embedded System Lab



Liquid Crystal Display Technology

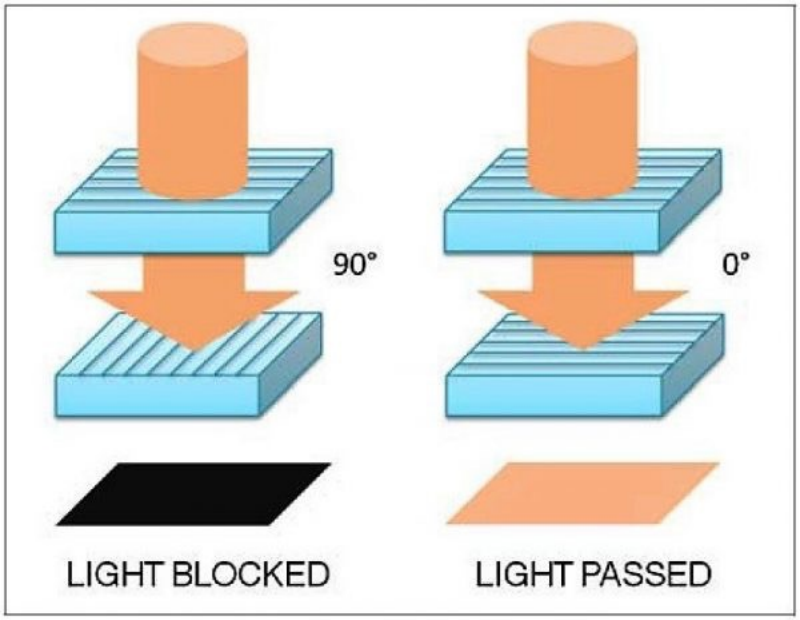
- LCD is the most common display technology.
- It uses **light modulating properties of liquid crystal** and **polarization of light**
  - Both electric and magnetic fields are vibrating perpendicular to the direction of propagation of light.
  - Most of the light sources are unpolarized.



https://electrosome.com/lcd-display-fundamentals/

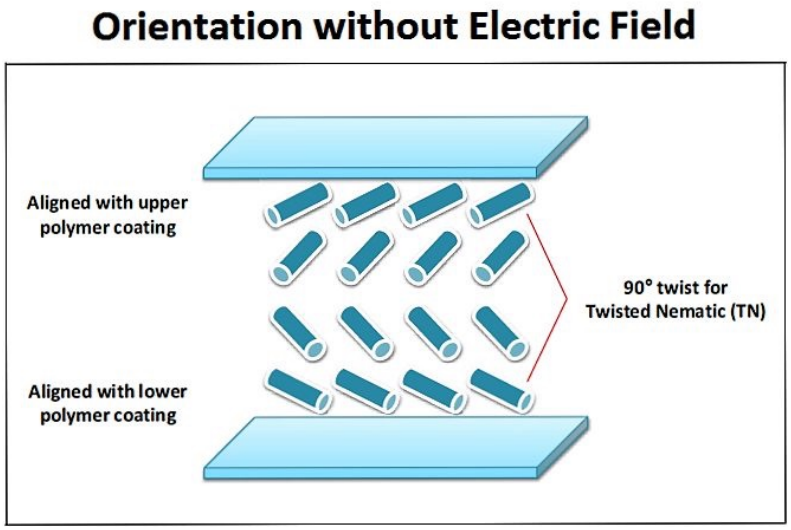
Polarization of Light

- **Polarization:** the process of converting unpolarized light to polarized light
  - Electric fields in polarized lights will vibrate only in a specific pattern
  - A vertical polarizer will pass only vertical components of the lights and horizontal components will be absorbed by it.

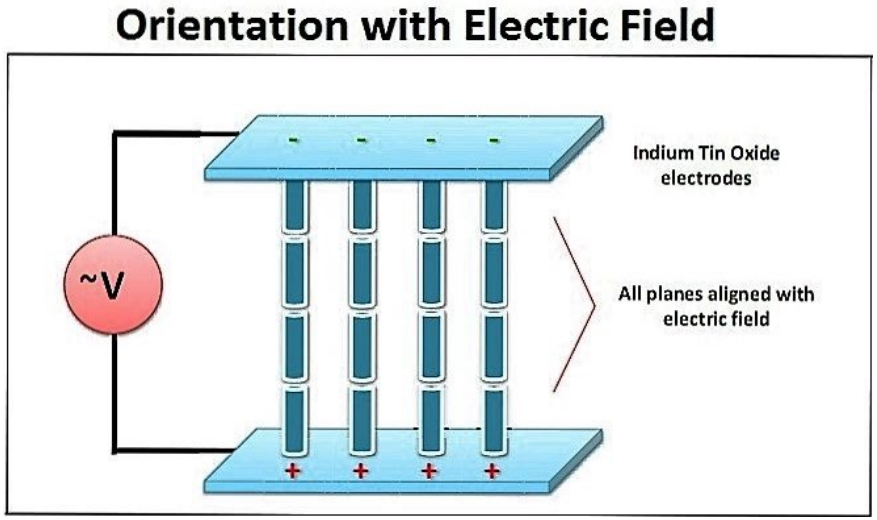


Liquid Crystal

- Exists in a state between crystalline (solid) and isotropic (liquid) state, which can be controlled by electric field.



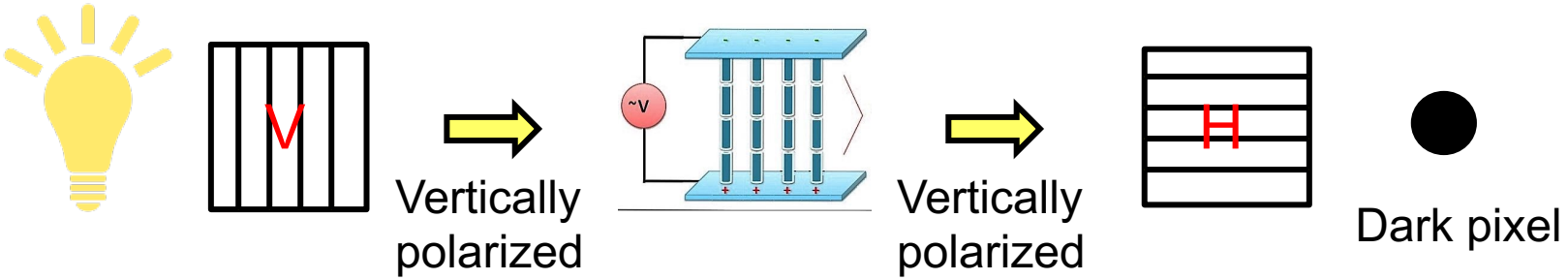
ON Pixel



OFF Pixel

How Pixels in LCD Works

Off pixel passes the light straight through keeping the same polarization.

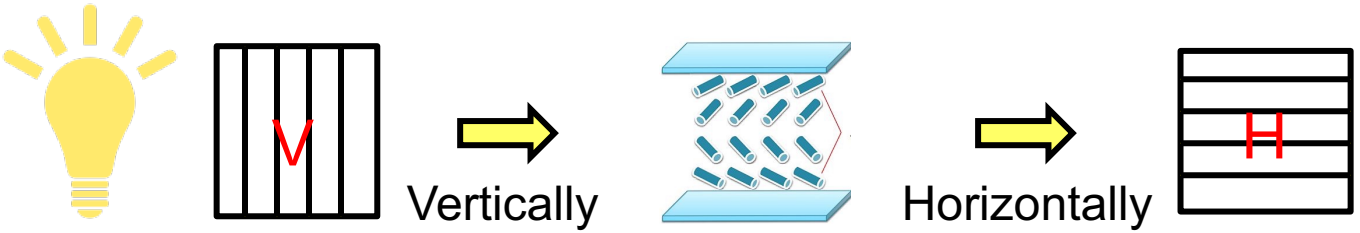


Vertically polarized

Vertically polarized

Dark pixel

On pixel twists the light 90° into horizontal polarization.



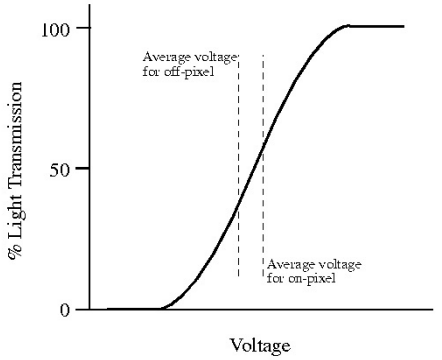
Vertically polarized

Horizontally polarized

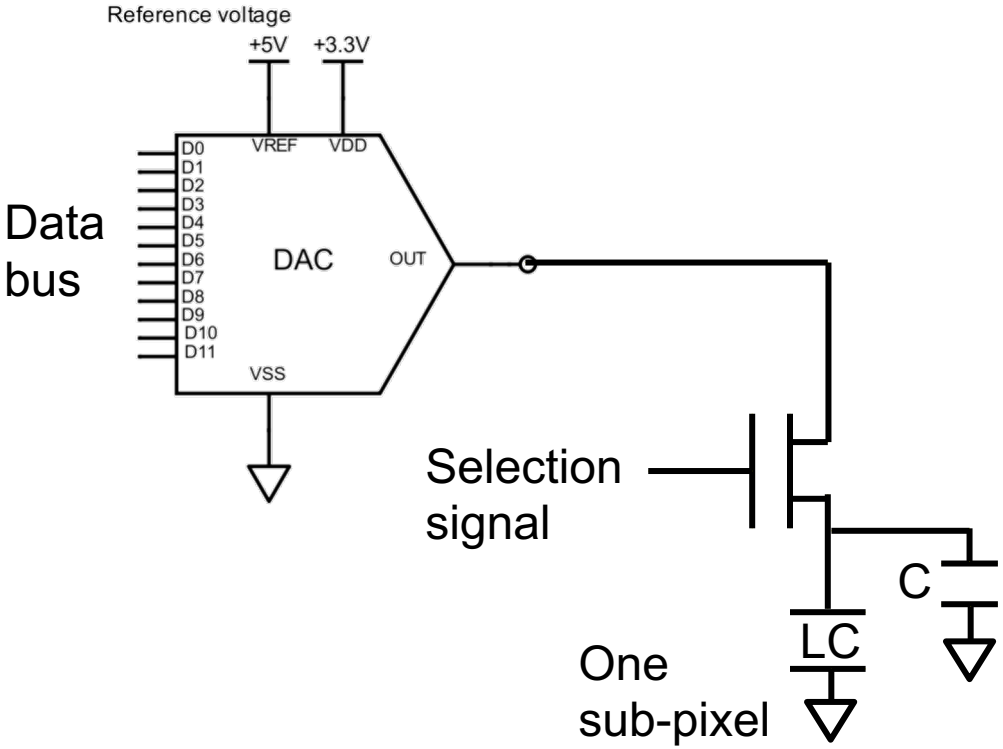
Bright pixel

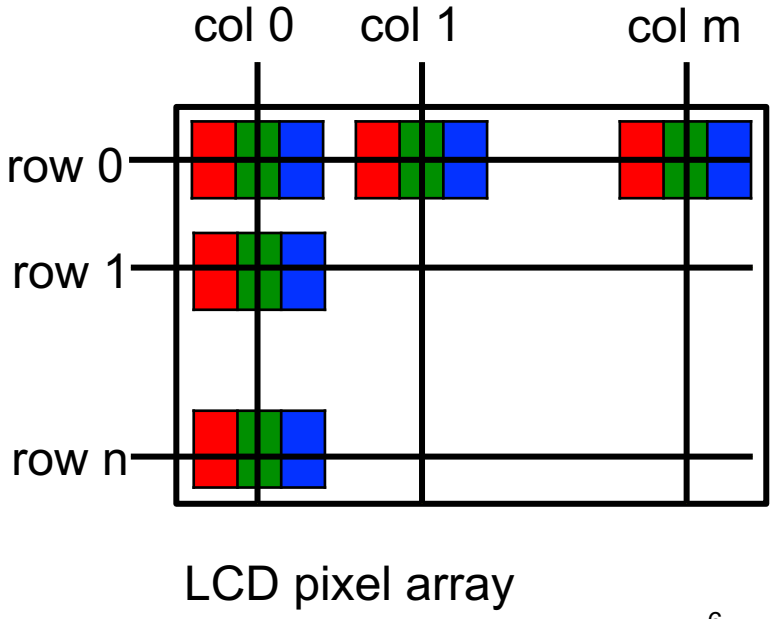
https://www.explainthatstuff.com/lcdtv.html

Driving LCD Pixels



- A pixel is the basic element of a display.
- Each pixel of a color LCD will have 3 sub pixels producing Red, Green and Blue colors.



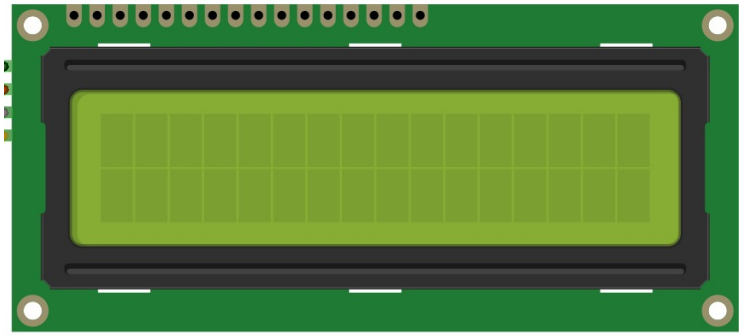


LCD pixel array

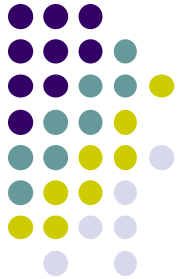
Liquid Crystal Character Displays

- A popular form of LCD is the character display.
  - These are widely available from one line of characters to four or more
  - Seen on photocopiers, burglar alarms or DVD players, etc.

- Driving this complex array of tiny LCD dots is far from simple, so such displays always contain a hidden microcontroller, customized to drive the display.
  - Most controllers are compatible to Hitachi HD44780.

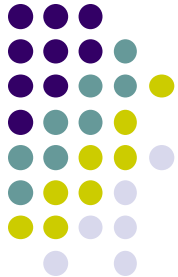


HD44780 Programming Interface



- The HD44780 contains an 80-byte RAM (Random Access Memory) to hold the display data, and a ROM (Read Only Memory) for generating the characters.
- It has a simple instruction set, including instructions for initialization, cursor control (moving, blanking, blinking), and clearing the display.
- Communication with the controller is made via an 8-bit data bus, 3 control lines, and an enable/strobe line (E).
- The HD44780 communication lines are shown below.

HD44780 Signals



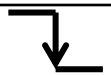
RS	Register select: 0 = instruction register, 1 = data register
$R/\bar{W}$	Read or write
E	Synchronize read or write operations
$DB4 - DB7$	MSB bits of data; $DB7$ is also a Busy flag
$DB0 - DB3$	LSB bits of data; not used in 4-bit mode

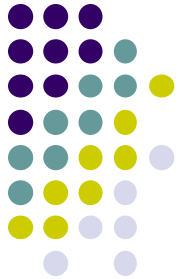
QC1602A LCD Display/Pins

Pin Number	Pin Name	Function
1	V_{SS}	Power supply (GND)
2	V_{DD}	Power supply (5V)
3	V_0	Contrast adjust
4	RS	Register select signal
5	R/IW	Data read / write
6	E	Enable signal
7	DB0	Data bus line bit 0
8	DB1	Data bus line bit 1
9	DB2	Data bus line bit 2
10	DB3	Data bus line bit 3
11	DB4	Data bus line bit 4
12	DB5	Data bus line bit 5
13	DB6	Data bus line bit 6
14	DB7	Data bus line bit 7
15	A	Power supply for LED back light (5V)
16	K	Power supply for LED back light (GND)

Compared with HD44780, there are a few additional pins, but control signals are the same.

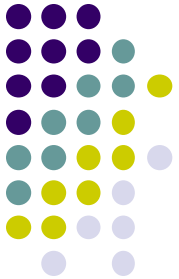
Write/read Registers

RS	$R/\overline{W}$	E	Action
0	0	Falling 	Write instruction register
0	1	Pulse 	Read Busy flag and address counter
1	0	Falling 	Write data register
1	1	Pulse 	Read data register



- Data written to the HD44780 controller is interpreted either as instruction or as display data, depending on the state of the RS (Register Select) line.
- The controller can be set up to operate in 8-bit or 4-bit mode.
- In 4-bit mode only the four most significant bits of the bus are used, and two write cycles are required to send a single byte.
- In both cases the most significant bit doubles as the Busy flag when a Read is undertaken.

QC1602A Programming



- Use the LCD in 4-bit mode.
  - Only the upper 4 MSB bits of the data bus (DB4-DB7) are connected.
  - LCD can be controlled with only 7 lines
    - Rather than the 11 lines for 8-bit mode.
  - The two halves of any byte are sent in turn on these lines.

- The display is initialized by sending control instructions to the configuration registers in the LCD.
  - Setting RS and $R/\overline{W}$ low and toggle E line from high to low.

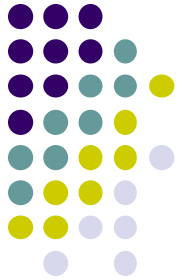
- Once the LCD has been initialized, display data can be sent
  - Setting RS high and $R/\overline{W}$ low and toggle E line from high to low.
  - Repeat above for every 4-bit words

Board to QC1602A Connections

Pin names	LCD pin number	LCD pin name	Power connection
GND	1	V_{SS}	0V
+5V	2	V_{DD}	5V
GND	3	V_0	0V
D2	4	RS	
GND	5	$R/\overline{W}$	0V (write only)
D3	6	E	
D4	11	$DB4$	
D5	12	$DB5$	
D6	13	$DB6$	
D7	14	$DB7$	
+5V	15	A	5V
GND	16	K	0V

If the $R/\overline{W}$ line is tied to ground, then a 1ms delay between data transfers is adequate to ensure that all internal processes can complete before the next transfer.

Code Structure for LCD Display

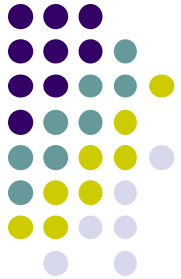


- LCD.cpp and LCD.h implements a library to work with LCD.
- The library has three functions:
  - toggle_enable( ): a function to toggle the enable bit E line from high to low.
  - LCD_init( ): a function to initialize the LCD
  - display_to_LCD( ): a function to display characters on the LCD

LCD.h

```
/* LCD.h header file */

#ifndef LCD_H
#define LCD_H
#include "mbed.h"

void toggle_enable(void); //function to toggle/pulse the enable bit
void LCD_init(void); //function to initialize the LCD
void display_to_LCD(char value); //function to display characters

#endif
```

Initializing the Display



- We first need to wait a short period (approximately 20 ms), then set the RS and E lines to zero (write instruction register) and then send a number of configuration messages to set up the LCD.

- Send configuration data to the instruction register for

  - Function Mode: 4-bit/8-bit, 1 line/2 line, 5x7/5x10 pixels
  - Display Mode: display on/off, cursor on/off, cursor blink/not
  - Clear display: clean all display buffers

Function mode

RS	$R/\bar{W}$
0	0

$DB7$	$DB6$	$DB5$	$DB4$	$DB3$	$DB2$	$DB1$	$DB0$
0	0	1	BW	N	F	X	X

BW=0 → 4-bit mode N=0 → 1-line mode F=0 → 5x7 pixels

BW=1 → 8-bit mode N=1 → 2-line mode F=1 → 5x10 pixels

X= don't care, 0 or 1

- For example, we send a binary value of 00101000 (0x28 hex) to the LCD data pins, this defines 4-bit mode, 2 line display and 5x7 dot characters.
- For 4-bit mode, we would therefore send the value 0x2, pulse E, then send 0x8, then pulse E again.

Display mode

RS	$R/\bar{W}$
0	0

$DB7$	$DB6$	$DB5$	$DB4$	$DB3$	$DB2$	$DB1$	$DB0$
0	0	0	0	1	P	C	B

P=0 → display off C=0 → cursor off B=0 → cursor no blink

P=1 → display on C=1 → cursor on B=1 → cursor blinking

Clear display

RS	$R/\bar{W}$
0	0

$DB7$	$DB6$	$DB5$	$DB4$	$DB3$	$DB2$	$DB1$	$DB0$
0	0	0	0	0	0	0	1

- Before data can be written to the display, the display must be cleared, and the cursor reset to the first character in the first row (or any other location that you wish to write data to).

Display characters

- Characters are encoded in ASCII codes
  - For example, if we send the data value 0x48 to the display, the character ‘H’ will be displayed.
- To send a 8-bit code
  - Set the $RS = 1$, send MSB 4 bits, and toggle E line from high to low.
  - Send LSB 4 bits, and toggle E line from high to low.

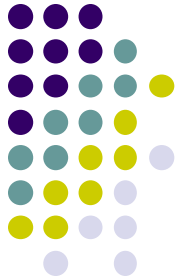
More significant bits (upper nibble)	Less significant bits (lower nibble)															
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF
0x0																
0x1																
0x2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0x3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0x4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0x5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0x6	`	a	b	C	d	e	f	g	h	i	j	k	l	m	n	o
0x7	p	q	r	S	t	u	v	w	x	y	z	{	|	}	~	

LCD.cpp --- LCD_init() 1/2

```
/* LCD function implementation*/
#include "LCD.h"
DigitalOut RS(D2);
DigitalOut E(D3);
BusOut data(D4, D5, D6, D7);

//initialize LCD function
void LCD_init(void)
{
    ThisThread::sleep_for(20ms); // pause for 20 ms
    RS = 0;      // set low to write control data
    E = 0;       // set low

    //function mode
    data = 0x2; // 4 bit mode (packet 1, DB4-DB7)
    toggle_enable();
    data = 0x8; // 2-line, 7 dot (packet 2, DB0-DB3)
    toggle_enable();
```

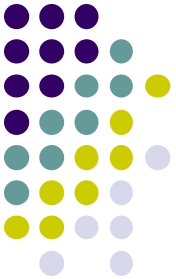

continue to next page

LCD.cpp --- LCD_init() 2/2



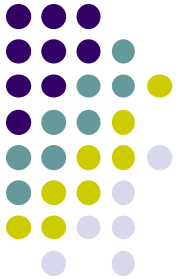
```
    //display mode
    data = 0x0; // 4 bit mode (packet 1, DB4-DB7)
    toggle_enable();
    data = 0xF; // display on, cursor on, blink on
    toggle_enable();

    //clear display
    data = 0x0;
    toggle_enable();
    data = 0x1; // clear
    toggle_enable();
}
```

LCD.cpp --- toggle_enable()



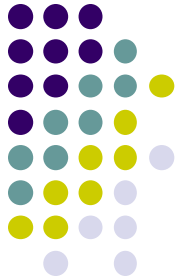
```
void toggle_enable(void)
{
    E = 1;
    ThisThread::sleep_for(1ms);
    E = 0;
    ThisThread::sleep_for(1ms);
}
```

LCD.cpp --- display_to_LCD()



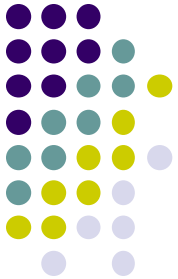
```
//display function
void display_to_LCD(char value)
{
    RS = 1;              // set high to write char data
    data = value >> 4; // shifted right 4 =upper nibble
    toggle_enable();
    data = value; //bitmask with 0x0F =low nibble
    toggle_enable();
}
```

LCD main.cpp



```
/* Utilizing LCD functions in the main.cpp file
*/
#include "LCD.h"
int main()
{
    LCD_init();                 // call the initialize function
    display_to_LCD(0x48);                // 'H'
    display_to_LCD(0x45);                // 'E'
    display_to_LCD(0x4C);                // 'L'
    display_to_LCD(0x4C);                // 'L'
    display_to_LCD(0x4F);                // 'O'
    for (char x = 0x30; x <= 0x39; x++)
    {                                    // display numbers 0-9
        display_to_LCD(x);
    }
}
```

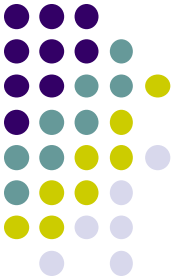
mbed TextLCD Library



```
// Hello World! for the TextLCD
#include "mbed.h"
#include "TextLCD.h”

TextLCD lcd(D2, D3, D4, D5, D6, D7);  // RS, E, DB4-
DB7

int main() {
    lcd.printf("HELLO");
    for (char x = 0x30; x <= 0x39; x++)
    { // display numbers 0-9
        lcd.printf(x);
    }_
}
```

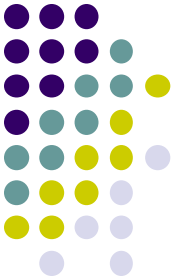
https://os.mbed.com/components/HD44780-Text-LCD/

Another mbed TextLCD Example



```
/* LCD Counter example with TextLCD library.
The example below displays a count variable on the LCD display.
The count variable increments every second.*/
#include "mbed.h"
#include "TextLCD.h"

TextLCD lcd(D2, D3, D4, D5, D6, D7); // RS, E, DB4-DB7
int main()
{
    int x = 0;
    lcd.printf("LCD Counter");
    while (1)
    {
        lcd.locate(5, 1);
        lcd.printf("%i", x);
        ThisThread::sleep_for(1s);
        x++;
    }
}
```

Example to Display ADC Values

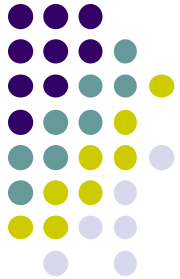


```
//Display ADC input data
#include "mbed.h"
#include "TextLCD.h"

TextLCD lcd(D2, D3, D4, D5, D6, D7);
AnalogIn Ain(A0);

int main(){
    float percentage;
    int D;
    while (1){
        percentage = Ain * 100;
        D = int(percentage);
        float B = percentage - D;
        int C = B * 1000000;

        lcd.printf("%d.", D);
        lcd.printf("%d", C);
        ThisThread::sleep_for(250ms);
        lcd.cls();
    }
}
```

Color LCD display --- 4D Systems uLCD-144-G2

- For color displays, each pixel is made up of three subpixels for red, green and blue.
  - Each pixel is a 24-bit value = (8-bit Red, 8-bit Green, 8-bit Blue)
  - Each subpixel can be set to 256 different shades of its color, so it is therefore possible for a single LCD pixel to display 256* 256 * 256 = 16.8 million different colors.

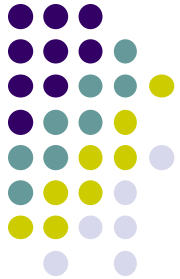
mbed Library for uLCD-144-G2



- The module uses UART serial port (to be discussed later)
  - uLCD_4DGL(Tx pin, Rx pin, rst pin);
- The library has six types of commands:
  - General : clear screen, reset screen, set background color…
  - Graphic : draw circle, line, triangle, set pen size…
  - Text: set font, set text color, locate, underline…
  - Media: display image, display video…
  - Screen data: get hardware information
  - Text data: get text information

https://os.mbed.com/components/4D-Systems-uLCD-144-G2/

uLCD Example: Countdown



```
#include "mbed.h"
#include "uLCD_4DGL.h"
uLCD_4DGL uLCD(D1, D0, D2); // UART4 tx, UART4 rx, reset pin;

int main() {
    uLCD.printf("\nHello uLCD World\n"); //Default Green on black text

    uLCD.text_width(4); //4X size text
    uLCD.text_height(4);
    uLCD.color(RED);
    for (int i = 10; i >= 0; --i) {
        uLCD.locate(1, 2);
        uLCD.printf("%2D", i);
        ThisThread::sleep_for(500ms);
    }
    uLCD.cls();
}
```