Liquid Crystal Display & Touch Sensor

Microcontroller Application and Development 2565

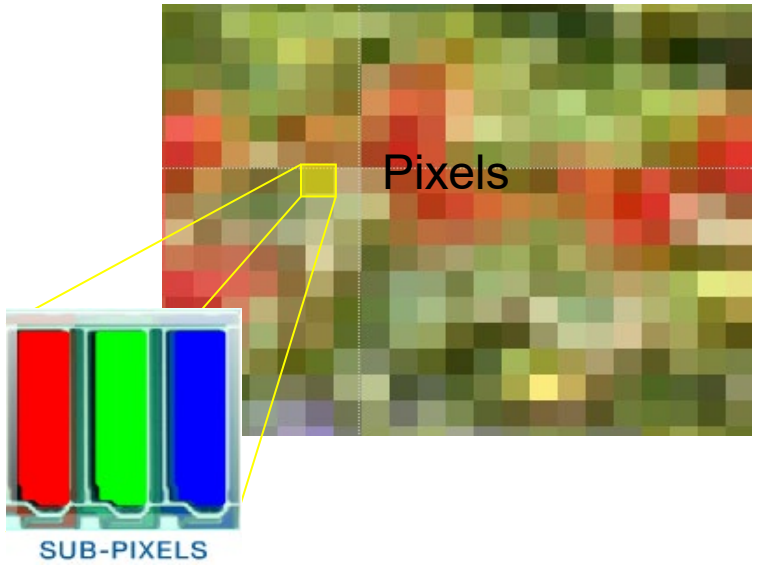
Sorayut Glomglome

Outline

1. Digital Image
2. LCD Technology
3. Touch Sensor
4. Lib

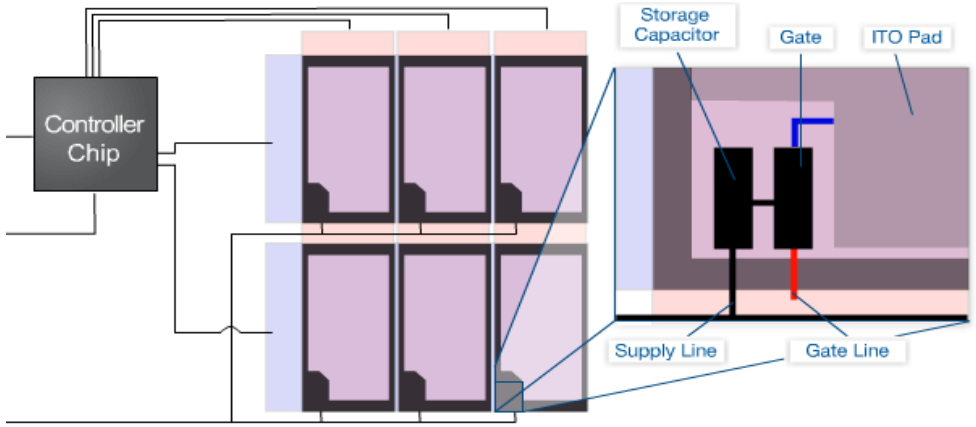
Digital Images and Pixels

- A digital image is a binary (digital) representation of a two-dimensional pictorial data.
- Digital images may have a **raster** or **vector** representation.
- Raster Images defined over a 2D grid of picture elements, called pixels.
- A pixel is the basic items of a raster image and include intensity or color value.



Active Matrix Display

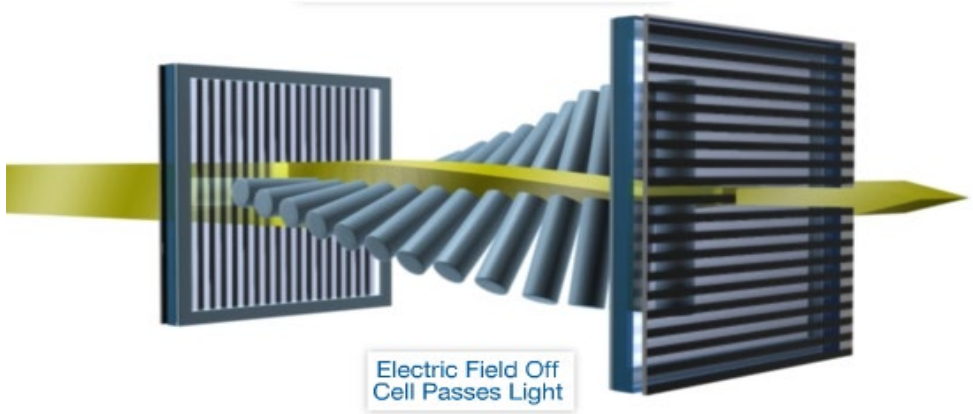
- Allow very high resolution
- Each sub-pixel is individually controlled by an isolated thin-film transistor (TFT).
- It allows the electrical signal for each sub-pixel to avoid influencing adjacent elements.
- The TFT is patterned into the glass layer



A display with
1024x768 resolution
Include 1024x768x3
= 2,359,296 sub-pixels

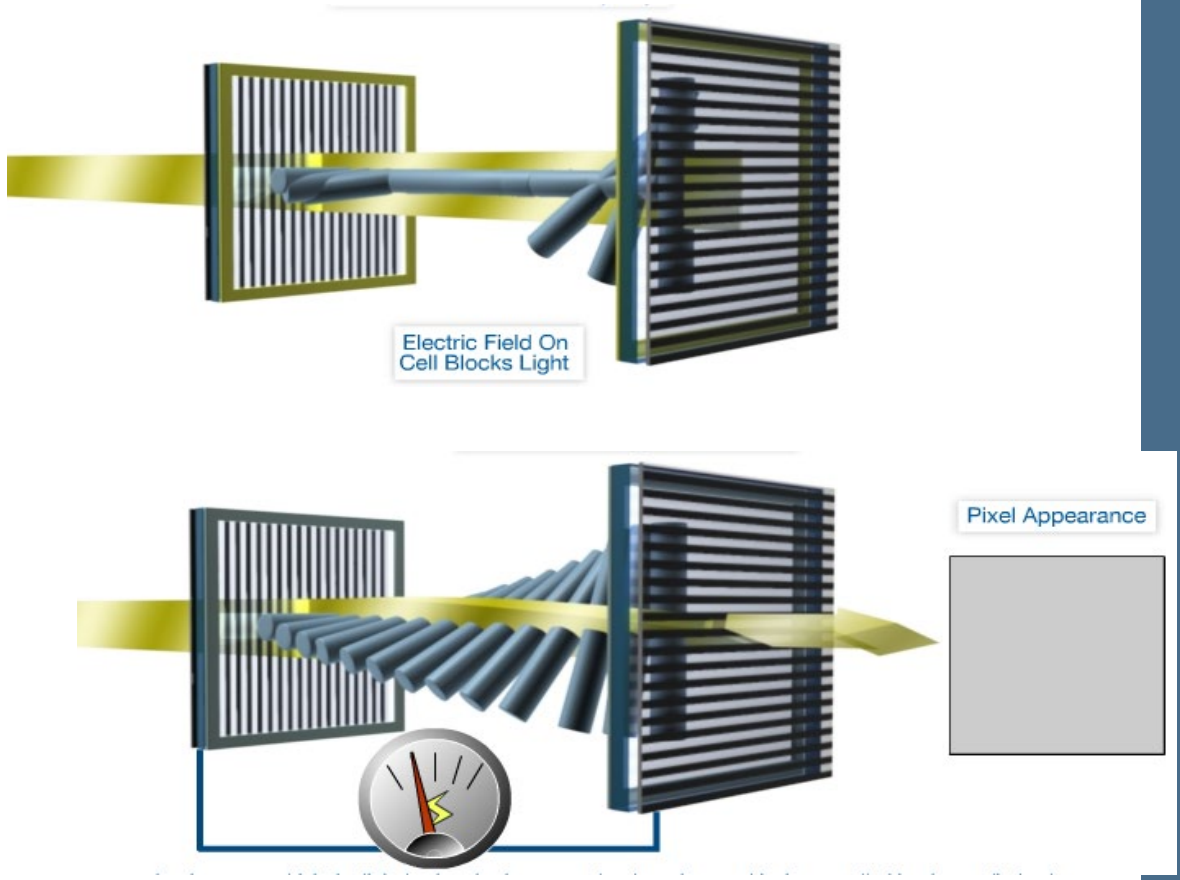
Twisted Nematic (TN) Display

- Is the most common LCD Display.
- The two alignments layer for the liquid crystal material are orthogonal.
- The light entering the polarize panel rotates by the twist in the liquid crystal and allowing it to pass through the second polarize

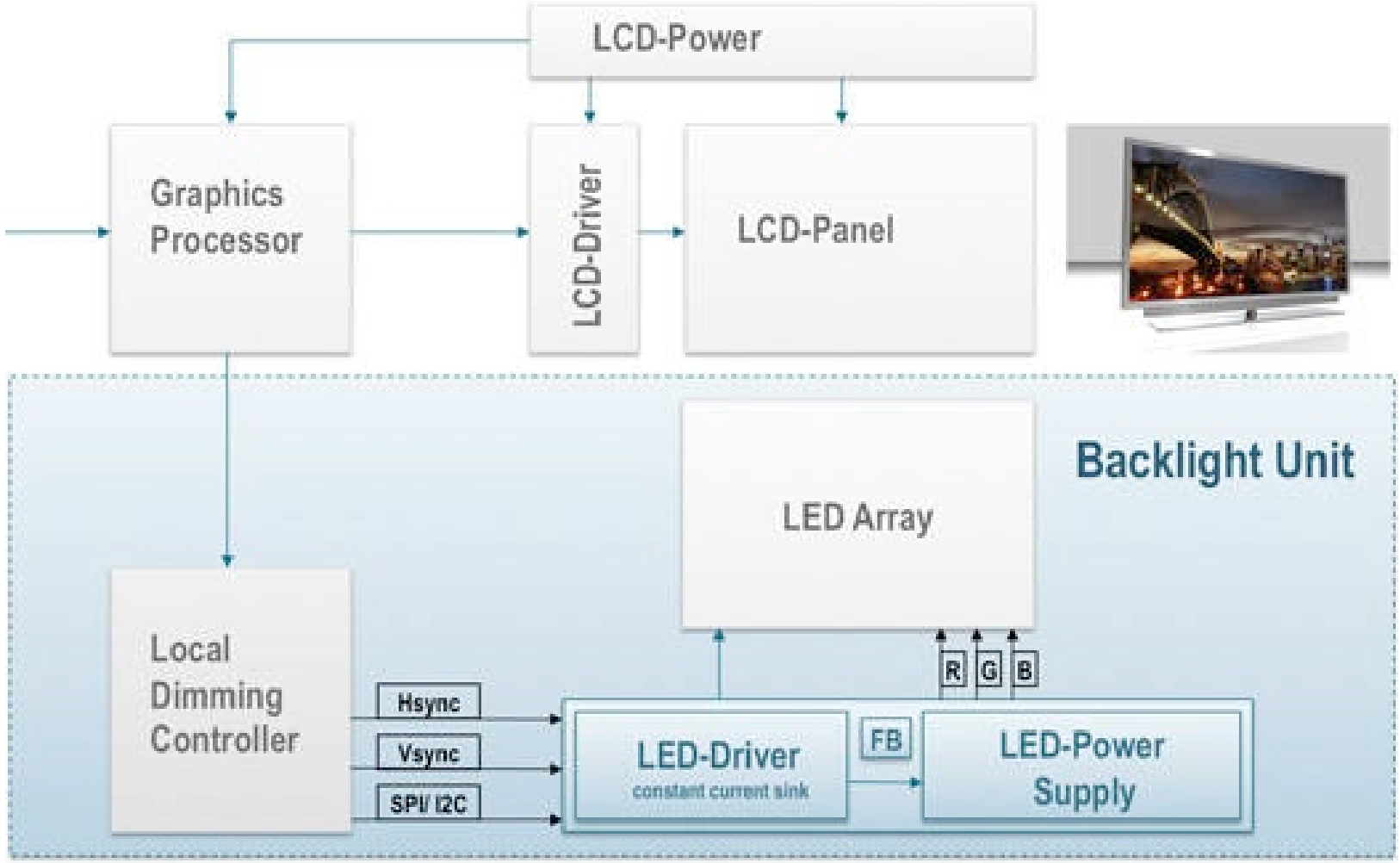


Twisted Nematic (TN) Display

- The electric field is applied
  - The liquid crystal loses its twist.
  - A light to the electric field.
  - Prevents the rotation of the polarized light
  - The second polarizer absorbs the light.
- The applied voltage control the absorbed and transmitted light

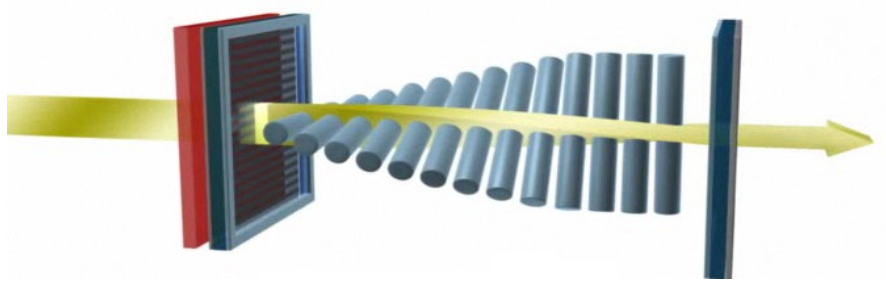


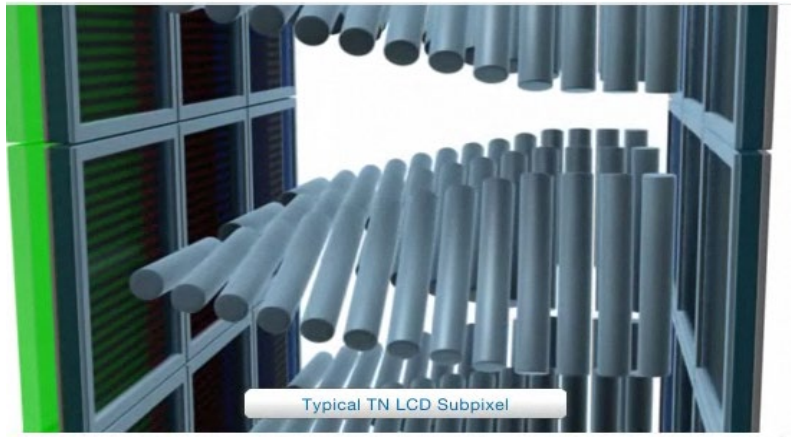
LCD Display Structure



LCD (Liquid Crystal Display)

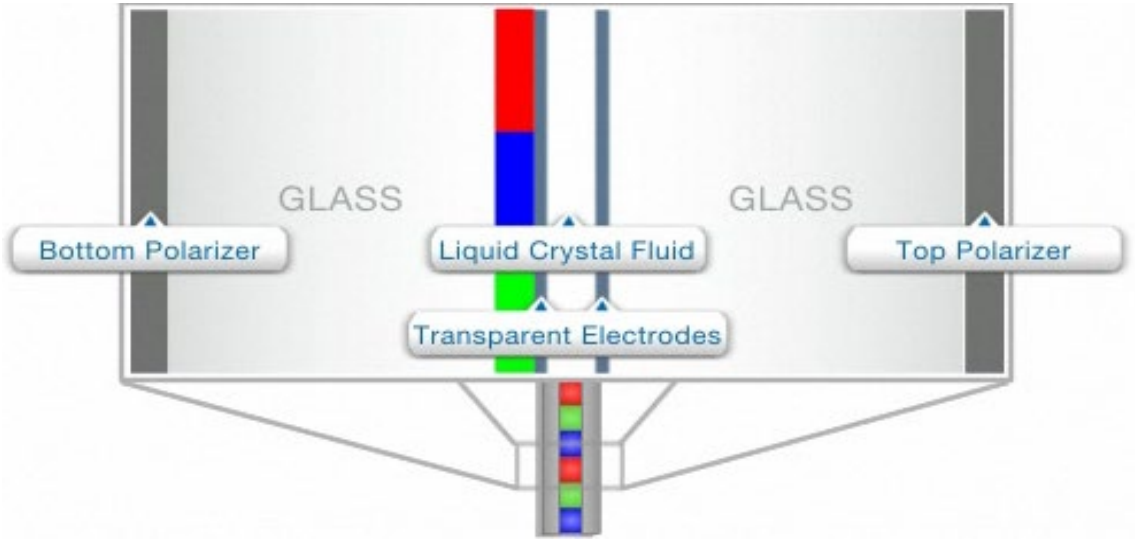
- LCD Panel is based on
  - A light valve for each pixel that turn the light on, off, or an intermediate level.
- Grid of such light valve for the LCD display panel.
- A back light and display enhancement films create the illumination.

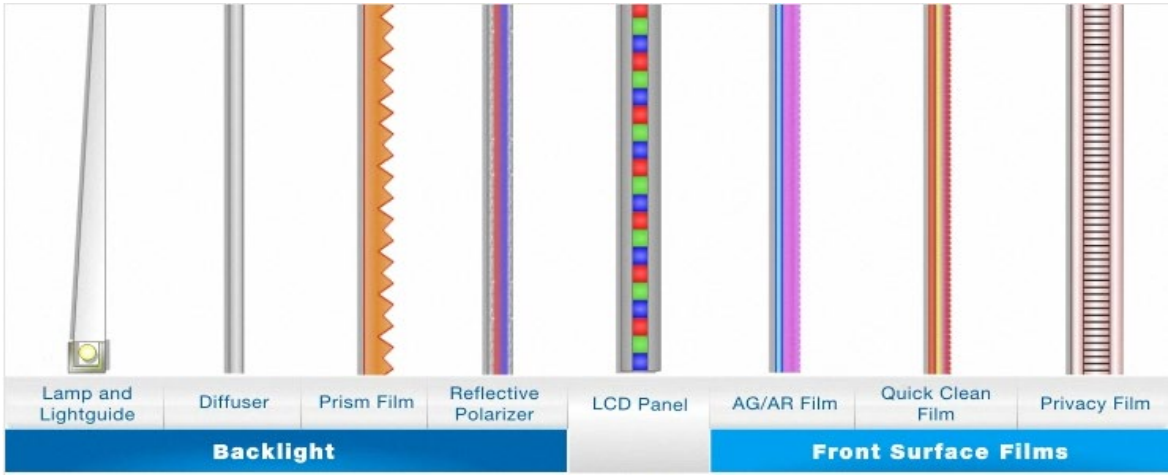




LCD-Display

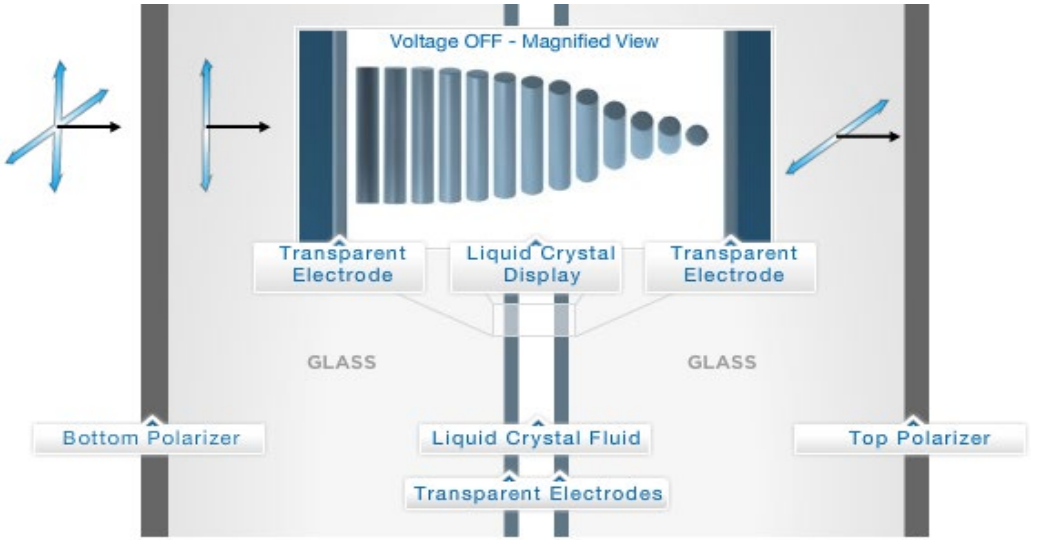
- Applying voltage to the electrodes changes the level of illumination in each sub-pixel
- The panel is sandwiched between
  - Front surface films to enhance display property
  - Backlight

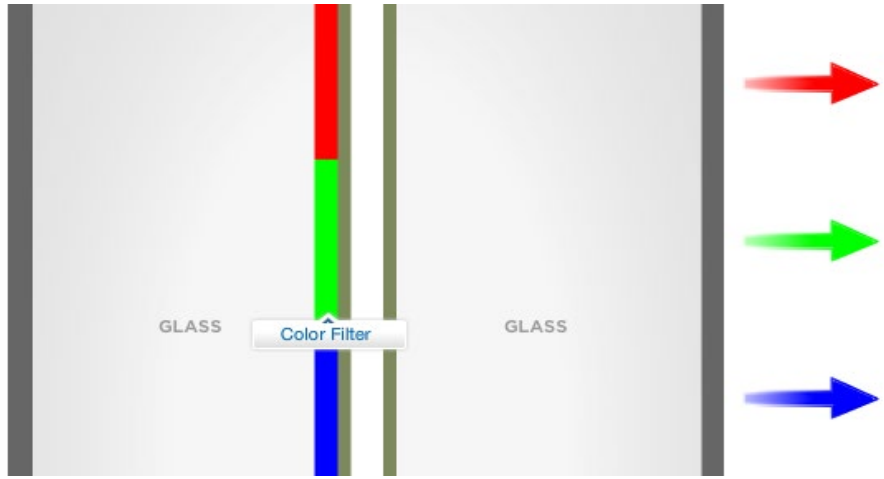




Light Path

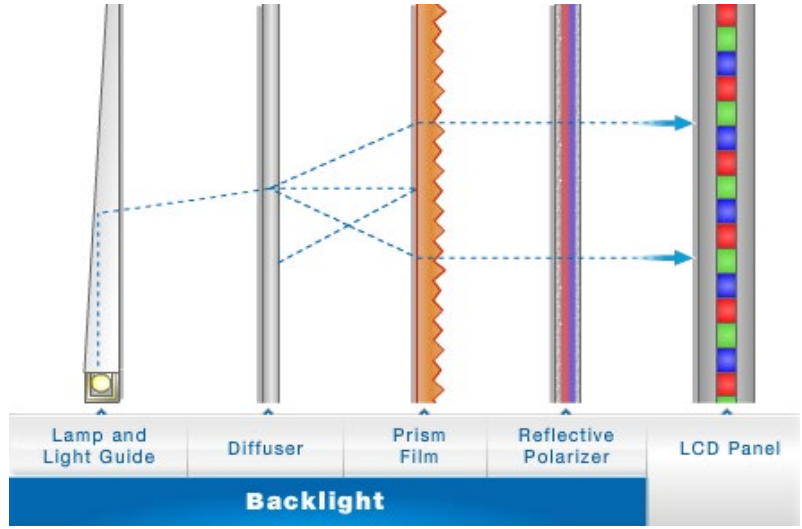
- The light passes through the polarizer.
- The voltage applied to the electrodes controls the liquid crystal orientation
- The liquid crystal orientation controls the rotation of the incoming polarized light.
- Color filters are used in color LCD, where each color sub-pixel is controlled individually

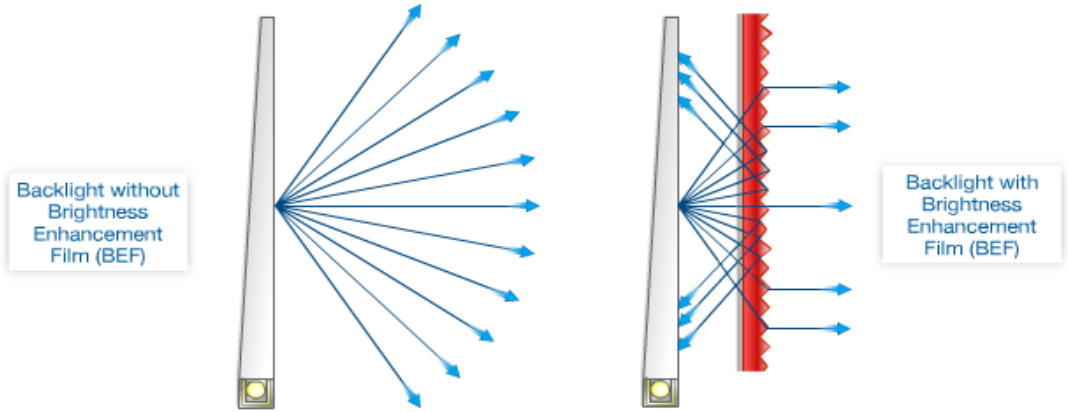




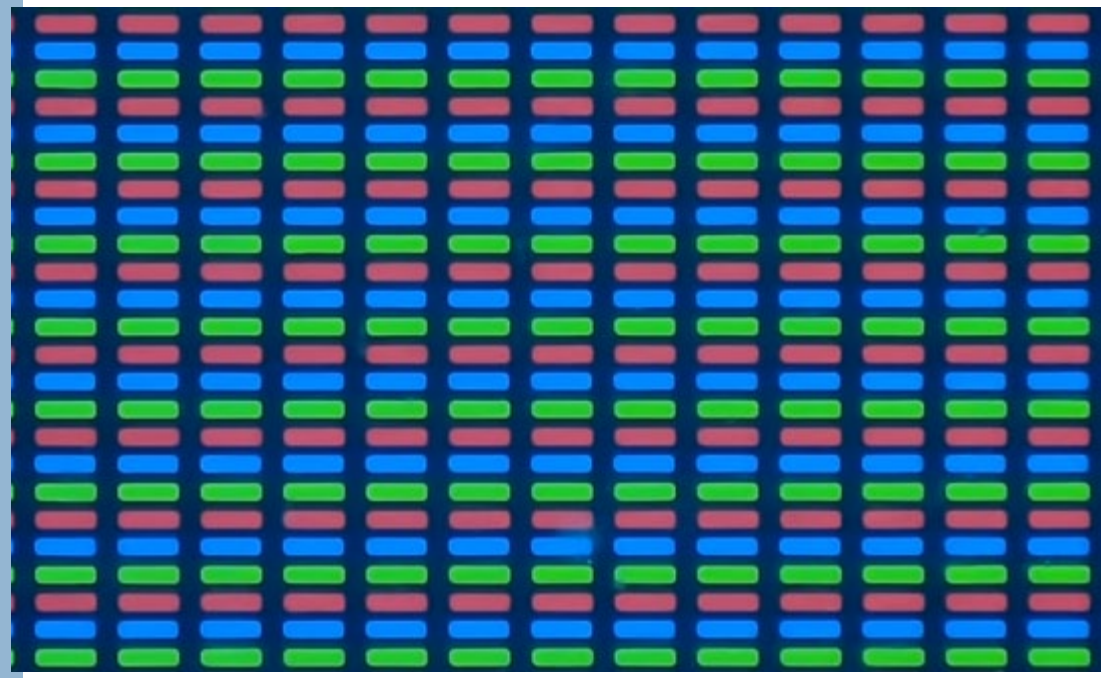
Back-Light

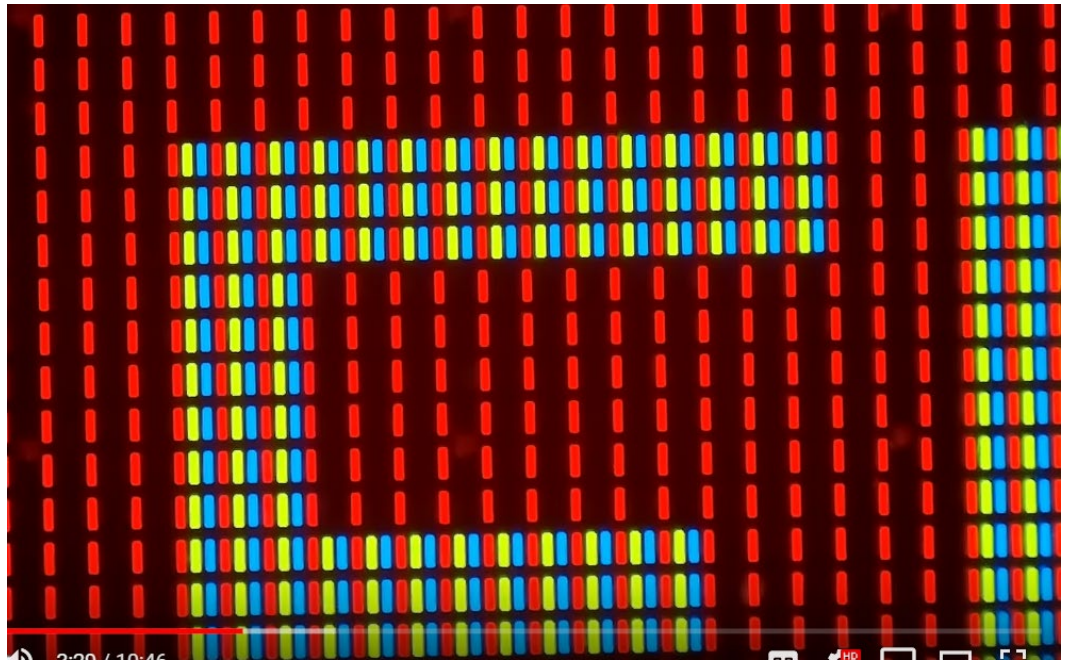
- The light generated by the backlight.
- The light is evenly distributed the light evenly over the LCD panel.
- Display enhancement films are placed between the light diffuser and the LCD panel. They aim to maximize the light reaching the observer.





LCD under Microscopic



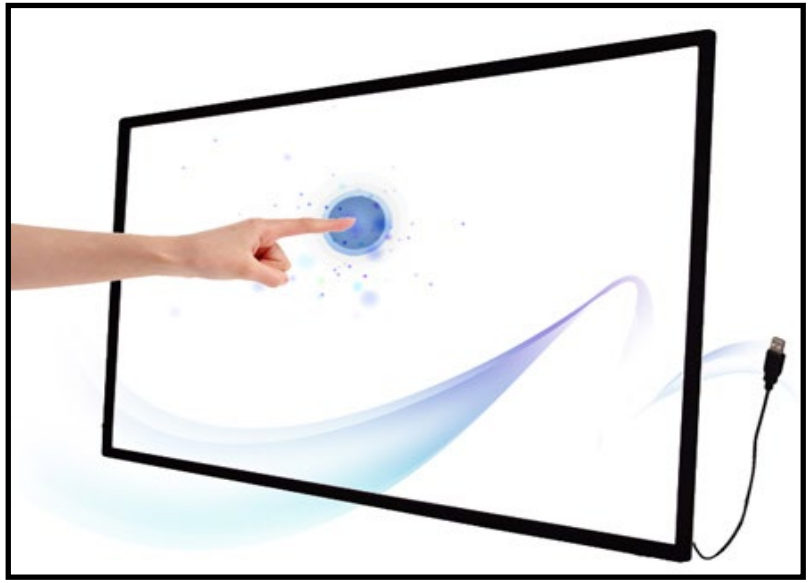


https://www.youtube.com/watch?v=Slo_Gv7K7Fo

Touch Screen Sensor

What is a touch screen?

- An electronic visual display that locates the coordinates of a users touch within display area
- Works independently of what is being displayed on screen

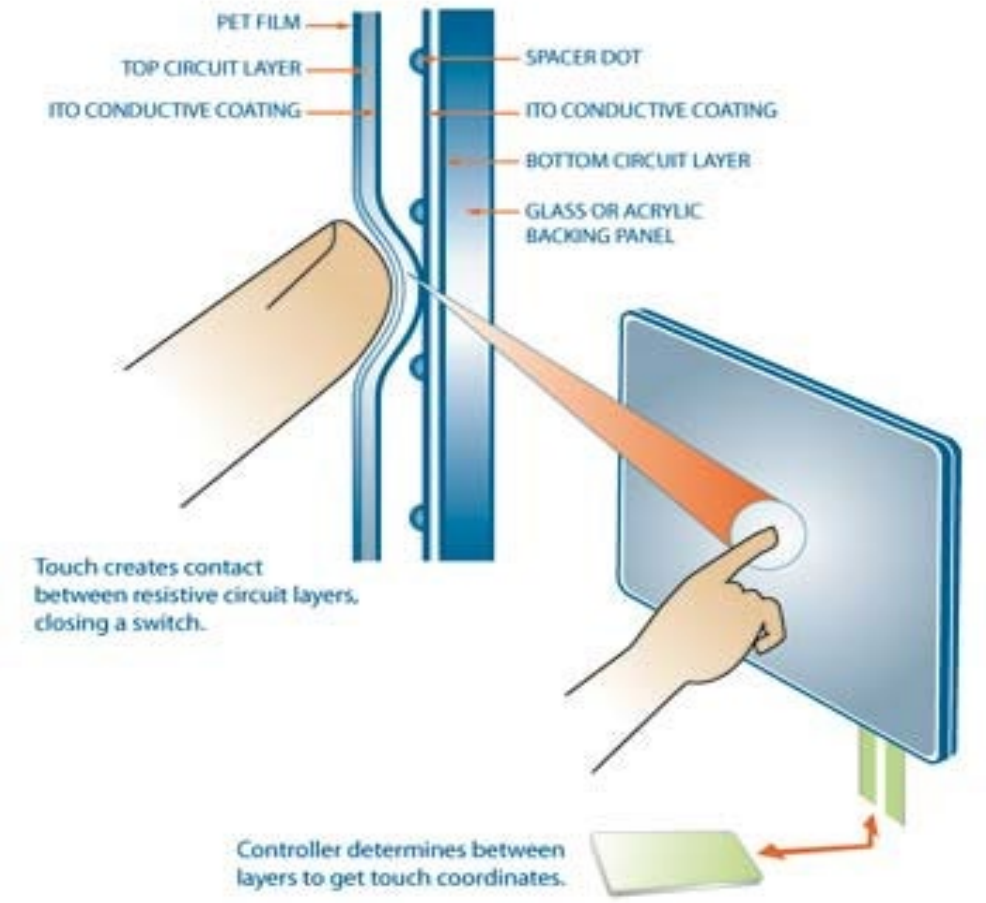


Touchscreen Technologies

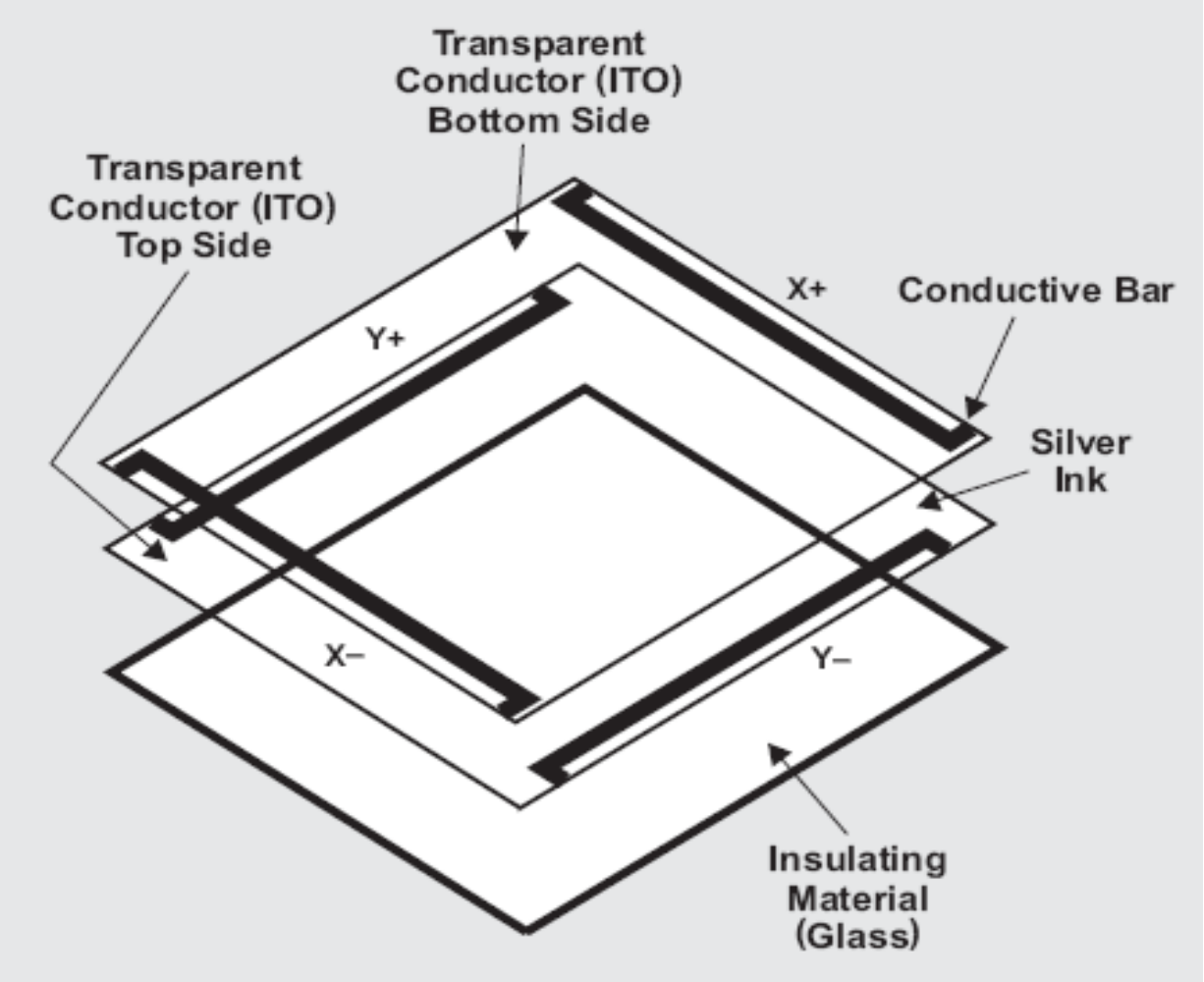
- Resistive touchscreen
- Capacitive touchscreen
- Infrared touchscreen

Resistive touchscreen Structure

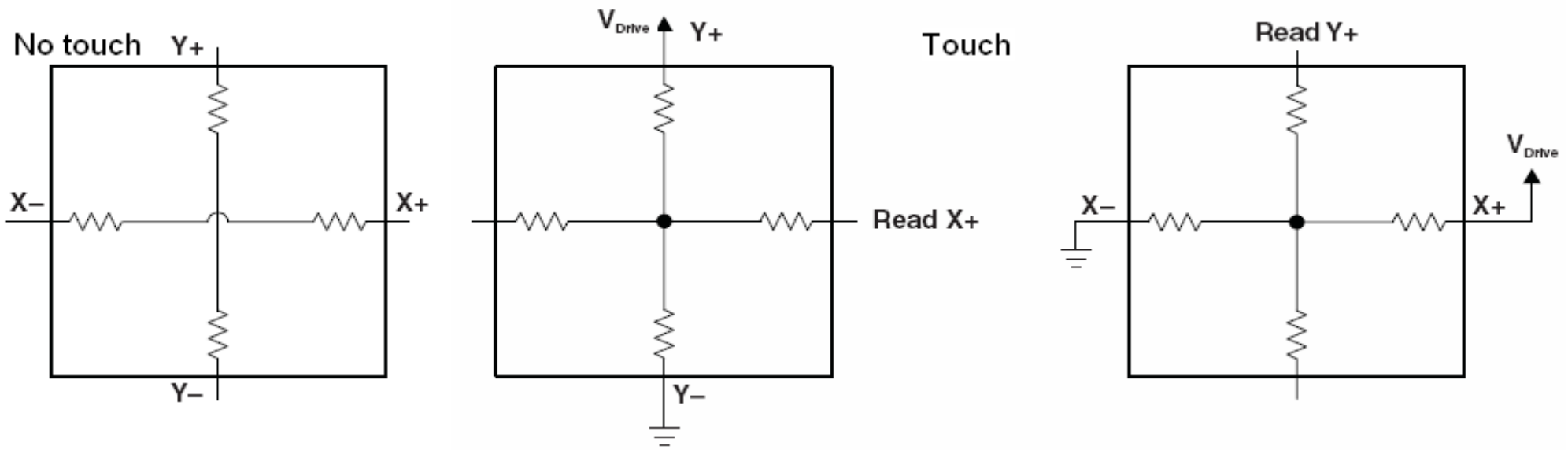
Resistive touch screens consist of a glass or acrylic panel that is coated with electrically conductive and resistive layers made with indium tin oxide (ITO) .The thin layers are separated by invisible spacers.



4-wire resistive touchscreen



4-wire resistive touchscreen



$$y = \frac{V_{X+}}{V_{Drive}} \times height_{screen}$$

$$x = \frac{V_{Y+}}{V_{Drive}} \times width_{screen}$$

Resistive touchscreen Characteristics

1. Cost effective solutions
2. Activated by a stylus, a finger or gloved hand
3. Not affected by dirt, dust, water, or light
4. 75%~85% clarify
5. resistive layers can be damaged by a very sharp object

Capacitive Touch Technology

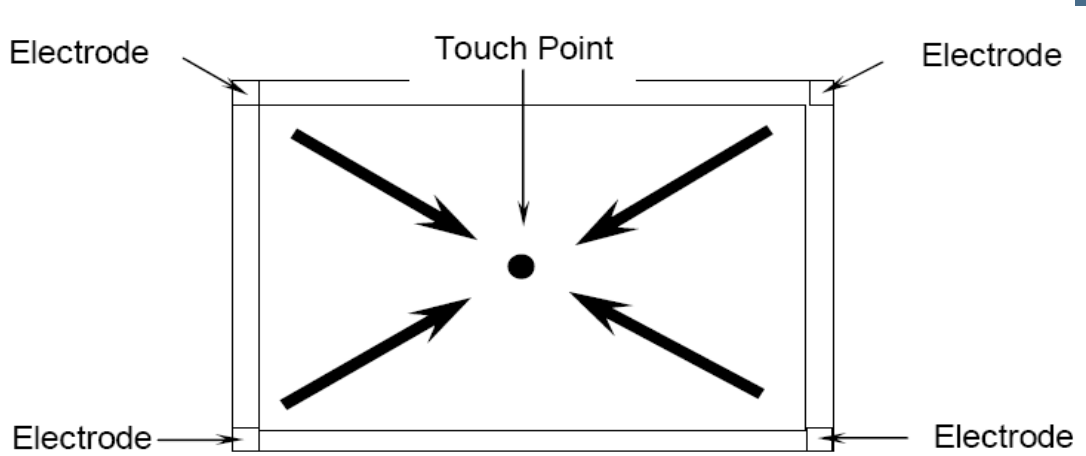
- Consists of:
  - Insulator (glass or Air)
  - Conductive coating (ITO)
- Two types:
  - Surface
  - Projected

Why Capacitive?

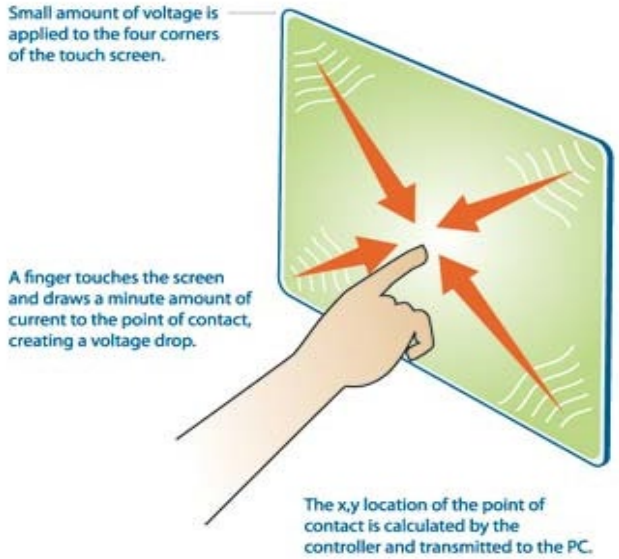
- Advantages
  - Durable surface material
  - High endurance (~255 million touches)
  - Very accurate
  - Good optical quality
- Disadvantages
  - Triggered only by bare finger or active stylus

Surface-capacitive touchscreen Structure

- Surface capacitive technology consists of a uniform conductive coating on a glass panel.
- Electrodes around the panel’ s edge evenly distribute a low voltage across the conductive layer, creating a uniform electric field.

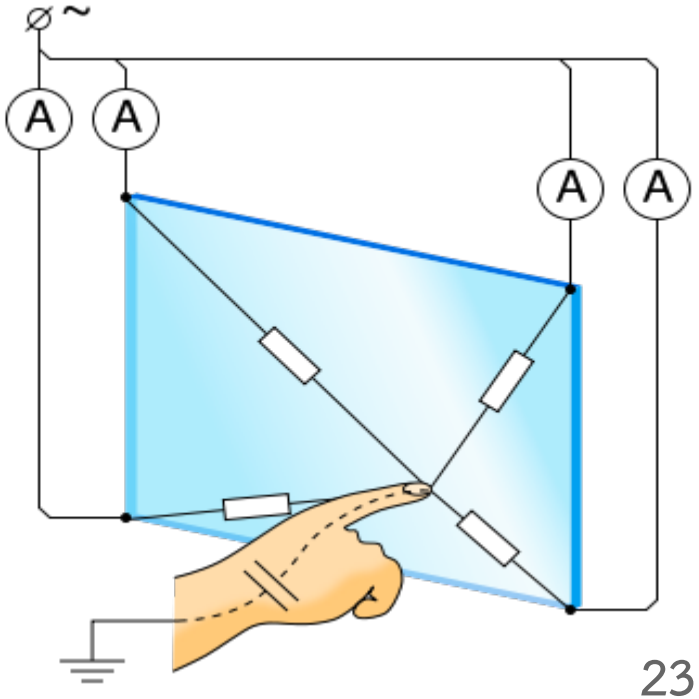


Surface-capacitive touchscreen



Working principle

- A human body is an electric conductor, so when you touch the screen with a finger, a slight amount of current is drawn, creating a voltage drop.
- The current respectively drifts to the electrodes on the four corners.
- Theoretically, the amount of current that drifts through the four electrodes should be proportional to the distance from the touch point to the four corners.
- The controller precisely calculates the proportion of the current passed through the four electrodes and figures out the X/Y coordinate of a touch point.

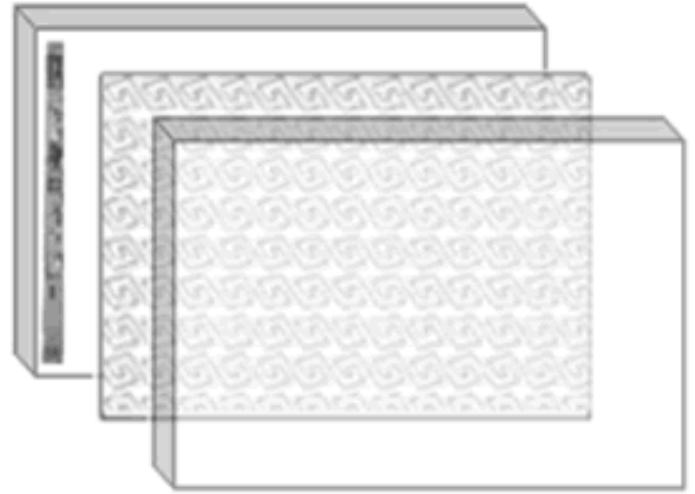


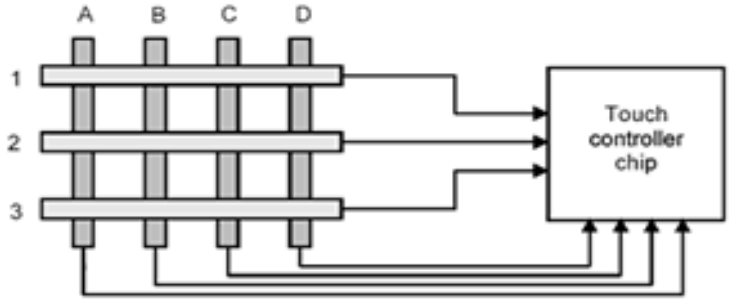
Projected-capacitive touchscreen

- Structure

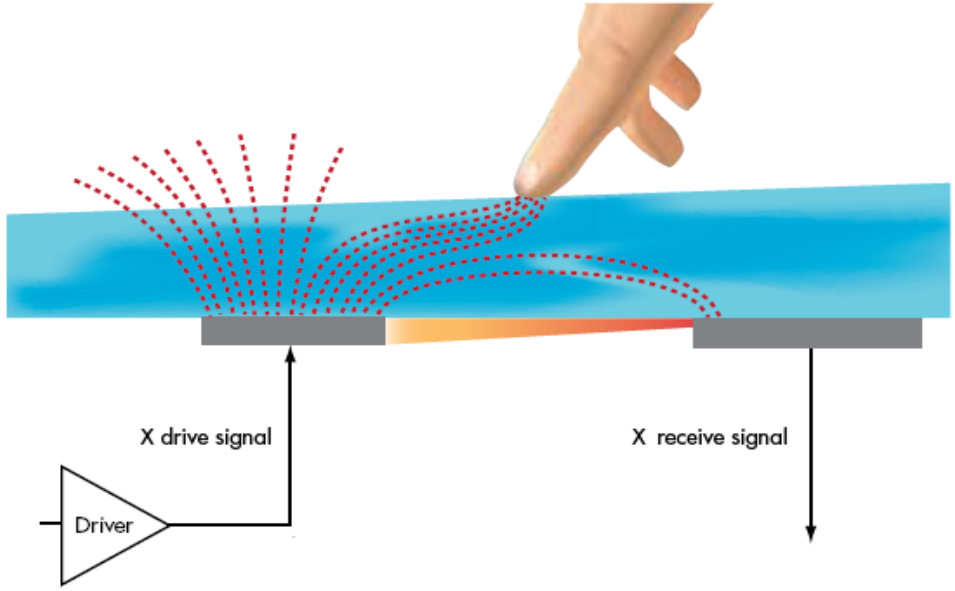
Projected capacitive touchscreens have front and back protective glass providing optical and strength enhancement options.

Its middle layer consists of a laminated sensor grid of micro-fine wires, and optical enhancement options.



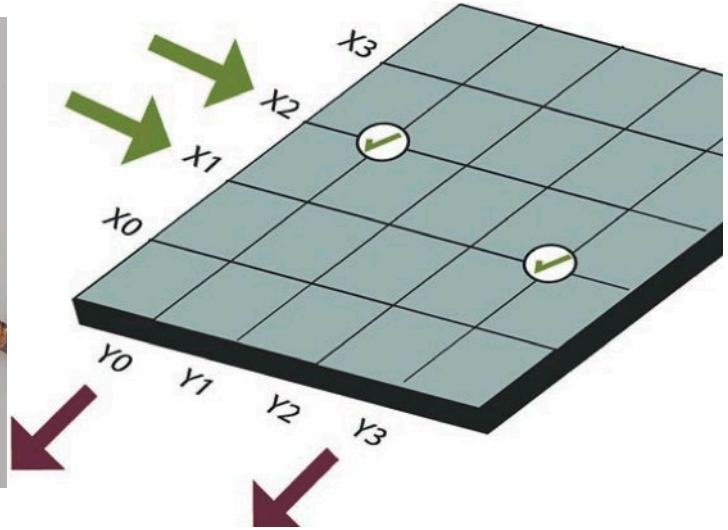


Projected-capacitive touchscreen



- During a touch, capacitance forms between the finger and the sensor grid.
- The embedded serial controller in the touchscreen calculates touch location coordinates and transmits them to the computer for processing.

Projected-capacitive Multi-touch

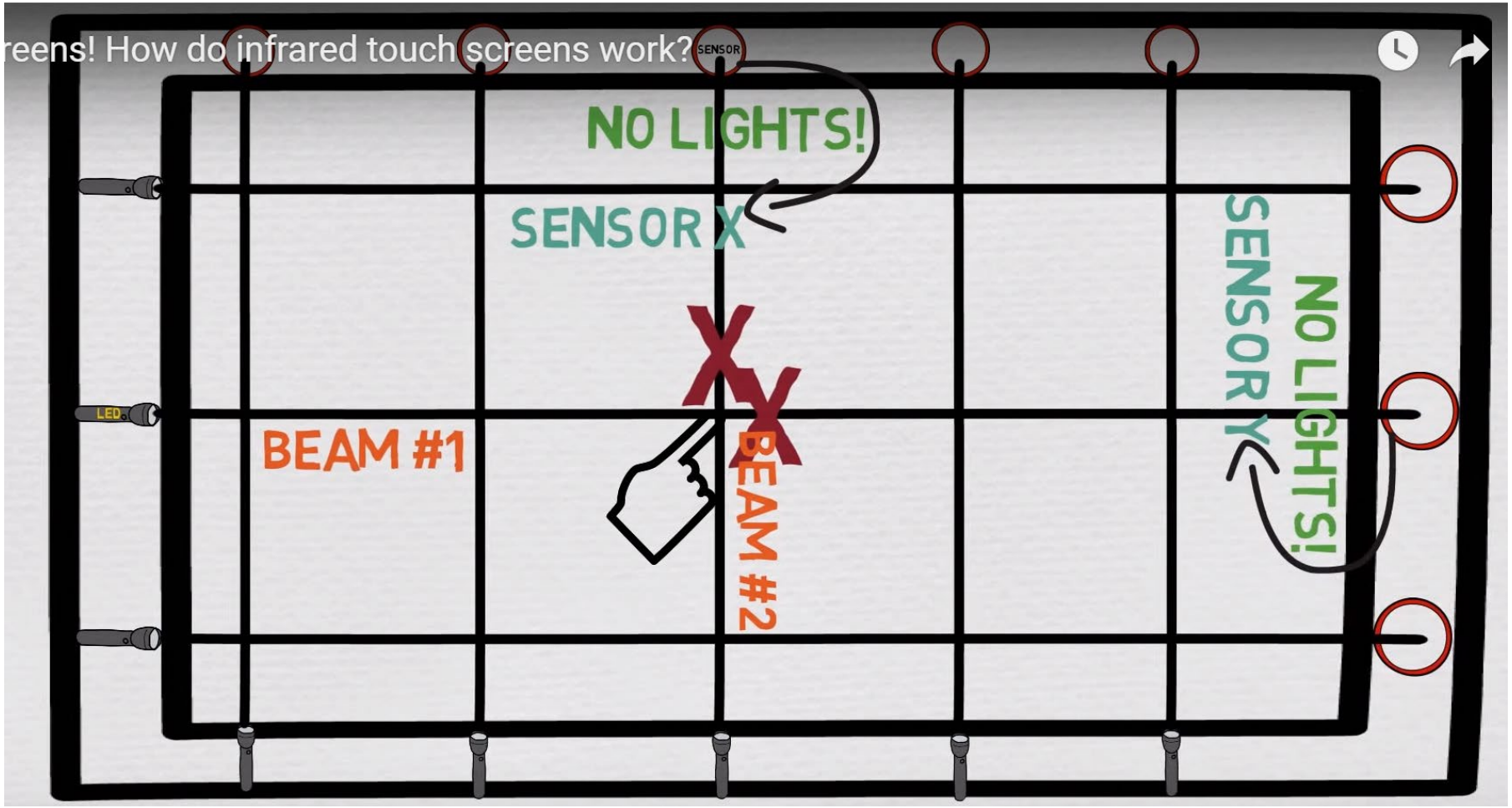


Surface vs. Projected

- Limited resolution
- Single touch
- Operation with direct contact

- High resolution
- Multi touch
- Operation with indirect contact

Infrared Touchscreen



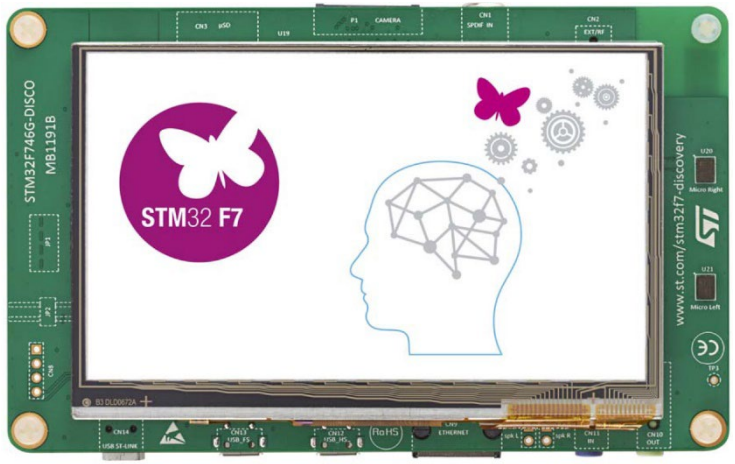
https://www.youtube.com/watch?v=52YYCEkC-6Q

Infrared Touchscreen

- Infrared (IR) technology relies on **the interruption of an IR light grid in front of the display screen**.
- The touch frame contains a row of IR-light emitting diode (LEDs) and photo transistors, each mounted on two opposite sides to create a grid of invisible infrared light.
- The IR controller sequentially pulses the LEDs to create a grid of IR light beams.
- When a stylus, such as a finger, enters the grid, it obstructs the beams. One or more photo transistors from each axis detect the absence of light and transmit signals that identifies the x and y coordinates.

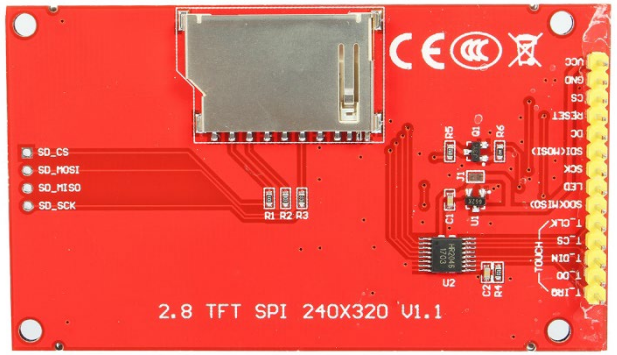
STM32F746 DISCO LCD

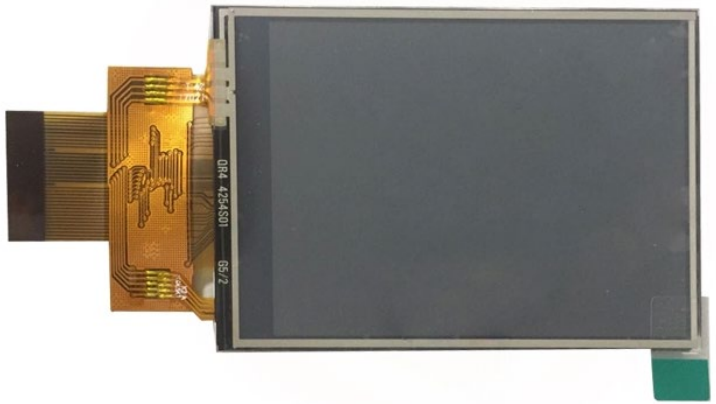
- 4.3-inch 480x272 color LCD-TFT with
- capacitive touch screen
- Rocktech RK043FN48H
- FT5336



LCD Module

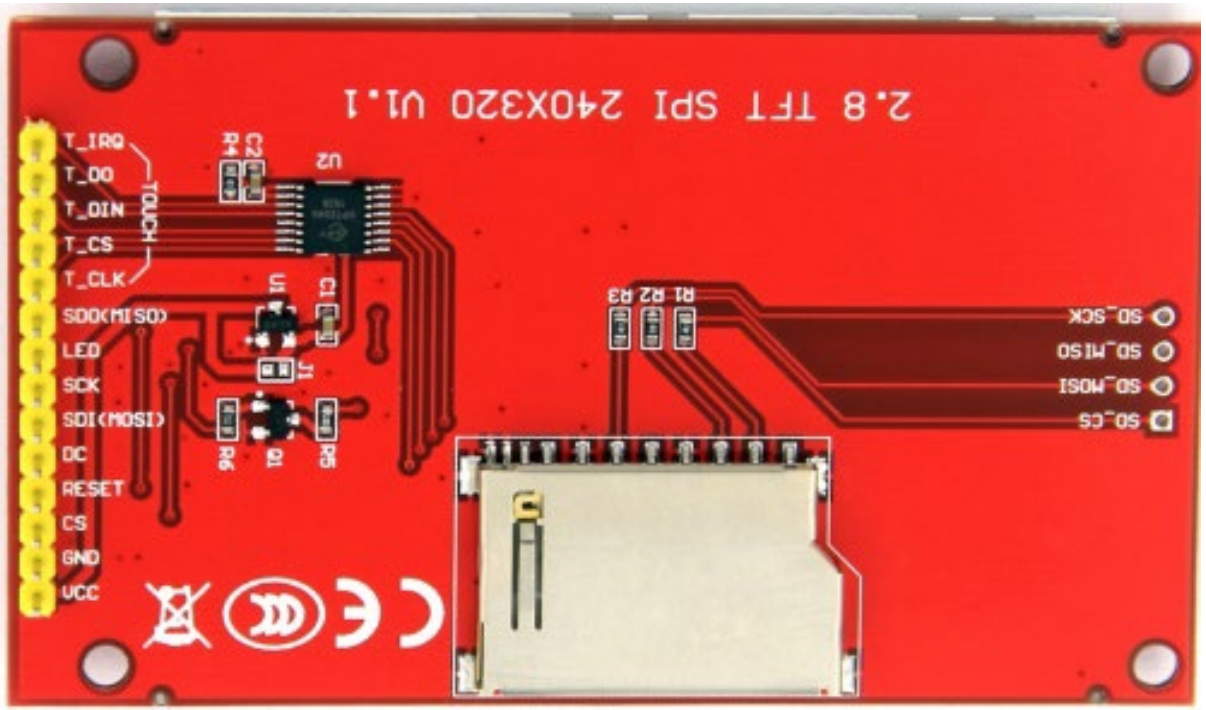
- 2.8-inch color screen
- 65K color display
- 320X240 resolution
- ILI9341
- 3.3v – 5v
- Touch function
- SPI serial bus
  - LCD
  - Touch sensor
  - SD Card





Interface

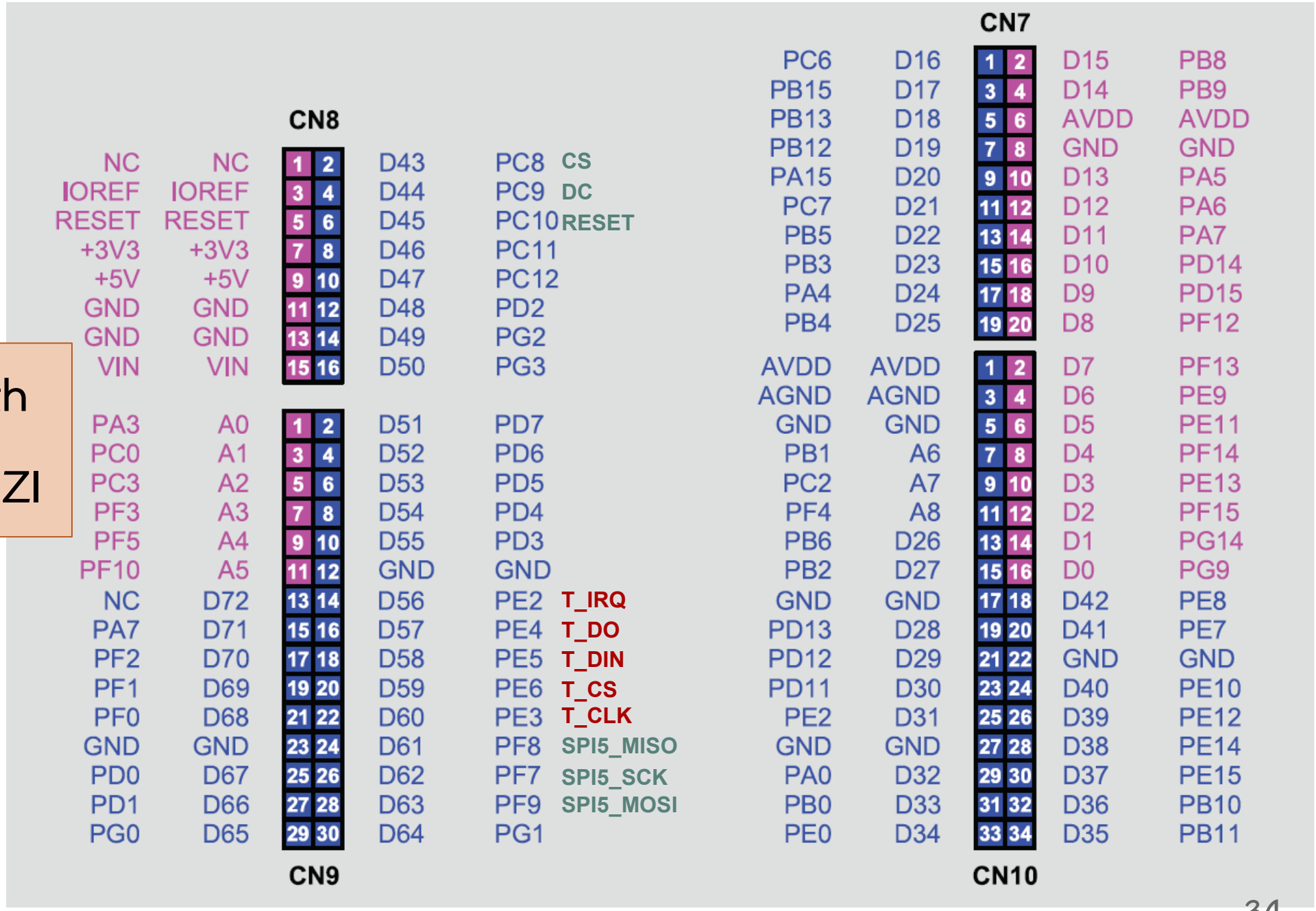
Number	Pin Label	Description
1	VCC	5V/3.3V power input
2	GND	Ground
3	CS	LCD chip select signal, low level enable
4	RESET	LCD reset signal, low level reset
5	DC/RS	LCD register / data selection signal, high level: register, low level: data
6	SDI(MOSI)	SPI bus write data signal
7	SCK	SPI bus clock signal
8	LED	Backlight control, high level lighting, if not controlled, connect 3.3V always bright
9	SDO(MISO)	SPI bus read data signal, if you do not need to the read function, you can not connect it
(The following is the touch screen signal line wiring, if you do not need to touch function or the module itself does not have touch function, you can not connect them)		
10	T_CLK	Touch SPI bus clock signal
11	T_CS	Touch screen chip select signal, low level enable
12	T_DIN	Touch SPI bus input
13	T_DO	Touch SPI bus output
14	T_IRQ	Touch screen interrupt signal, low level when touch is detected



Interface with Nucleo-F767ZI

STM32F767ZI	Function	LCD	Note	
PE2	GPIO_Input	T_IRQ		Bit Banging
PE4	GPIO_Input	T_DO	MISO	Bit Banging
PE5	GPIO_Output	T_DIN	MOSI	Bit Banging
PE6	GPIO_Output	T_CS		Bit Banging
PE3	GPIO_Ouput	T_CLK		Bit Banging
PF8	SPI5_MISO	SDO	MISO	SPI5
3.3V		LED		
PF7	SPI_SCK	SCK		SPI5
PF9	SPI_MOSI	SDI	MOSI	SPI5
PC9	GPIO_Output	DC		
PC10	GPIO_Output	RESET		
PC8	GPIO_Output	CS		
GND		GND		
3.3V		Vcc		

Interface with Nucleo-F767ZI



Setting STM32CubeMX : GPIO

- RCC: HSE@200 MHz

GPIO Mode and Configuration

Configuration

Group By Peripherals

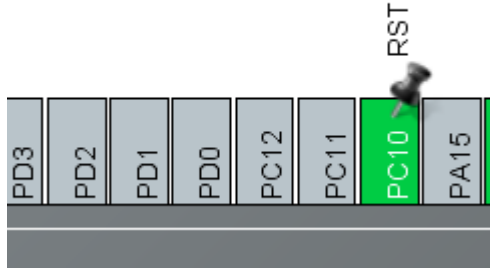
GPIO | Single Mapped Signals | RCC | SPI5 | SYS | USART3

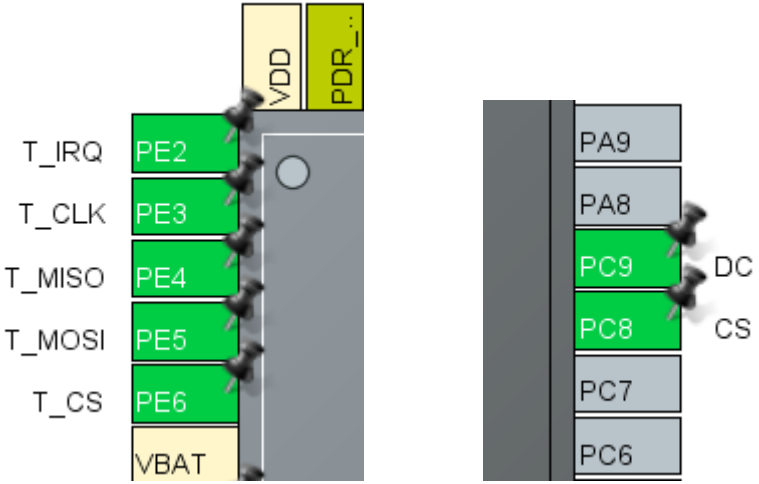
Search Signals

Search (Crt/+F)

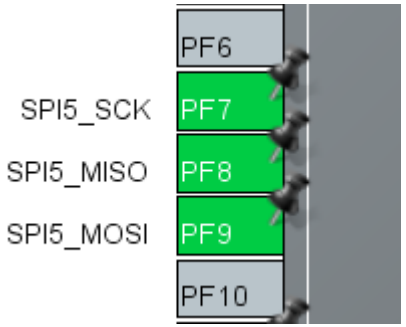
Show only Modified Pins

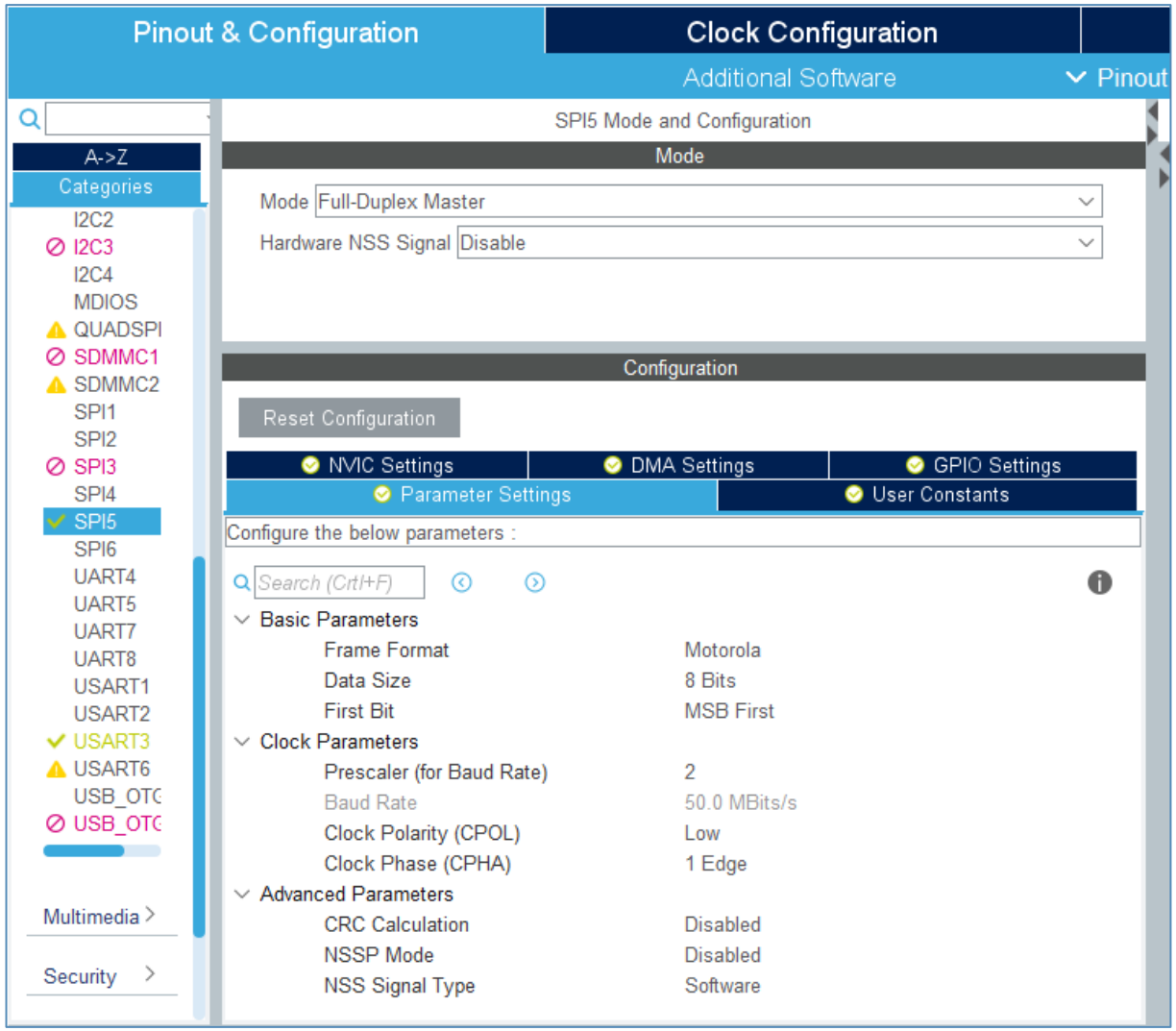
Pin Name	Signal on Pin	GPIO output...	GPIO mode	GPIO Pull-u...	Maximum o...	Fast Mode	User Label	Modified
PB7	n/a	Low	Output Pus...	No pull-up a...	Low	Disable	LD2 [Blue]	☑
PB14	n/a	Low	Output Pus...	No pull-up a...	Low	n/a	LD3 [Red]	☑
PC8	n/a	Low	Output Pus...	No pull-up a...	Very High	n/a	CS	☑
PC9	n/a	Low	Output Pus...	No pull-up a...	Very High	n/a	DC	☑
PC10	n/a	Low	Output Pus...	No pull-up a...	Very High	n/a	RST	☑
PC13	n/a	n/a	Input mode	No pull-up a...	n/a	n/a	User Blue B...	☑
PE2	n/a	n/a	Input mode	No pull-up a...	n/a	n/a	T_IRQ	☑
PE3	n/a	Low	Output Pus...	No pull-up a...	Very High	n/a	T_CLK	☑
PE4	n/a	n/a	Input mode	No pull-up a...	n/a	n/a	T_MISO	☑
PE5	n/a	Low	Output Pus...	No pull-up a...	Very High	n/a	T_MOSI	☑
PE6	n/a	Low	Output Pus...	No pull-up a...	Very High	n/a	T_CS	☑
PG6	n/a	Low	Output Pus...	No pull-up a...	Low	n/a	USB_Power...	☑
PG7	n/a	n/a	Input mode	No pull-up a...	n/a	n/a	USB_OverC...	☑



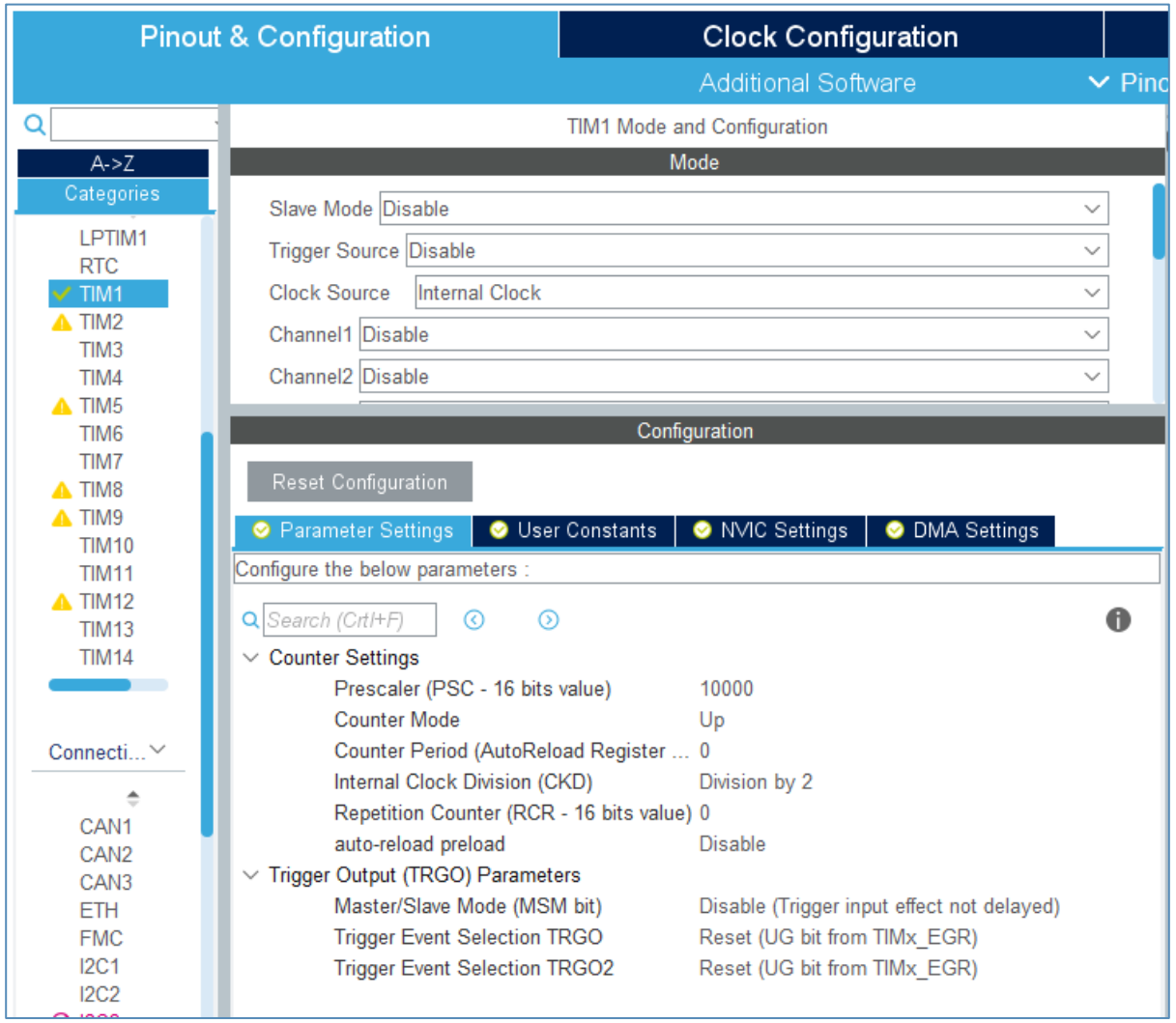


Setting STM32CubeMX : SPI5

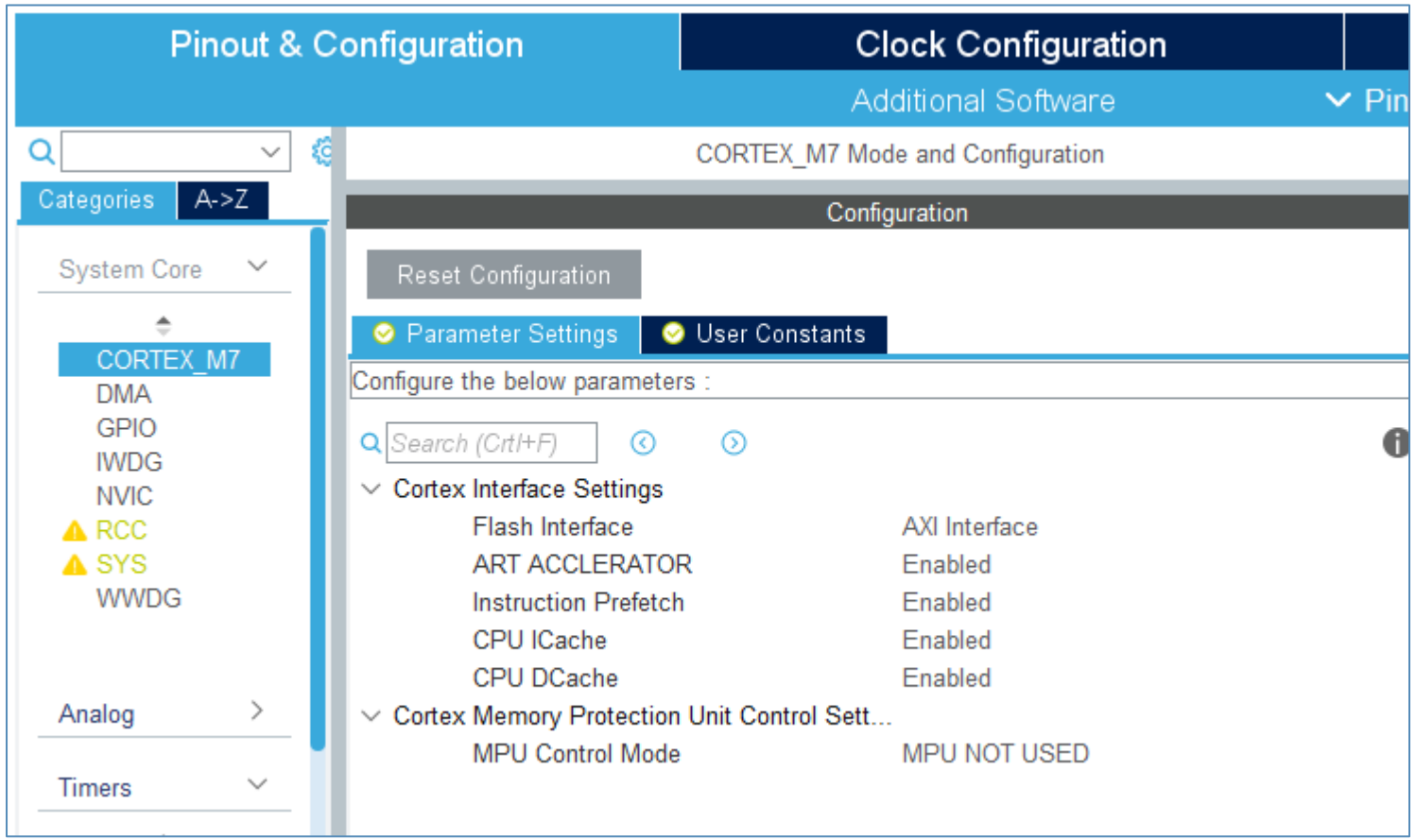




Setting STM32CubeMX : TIM1



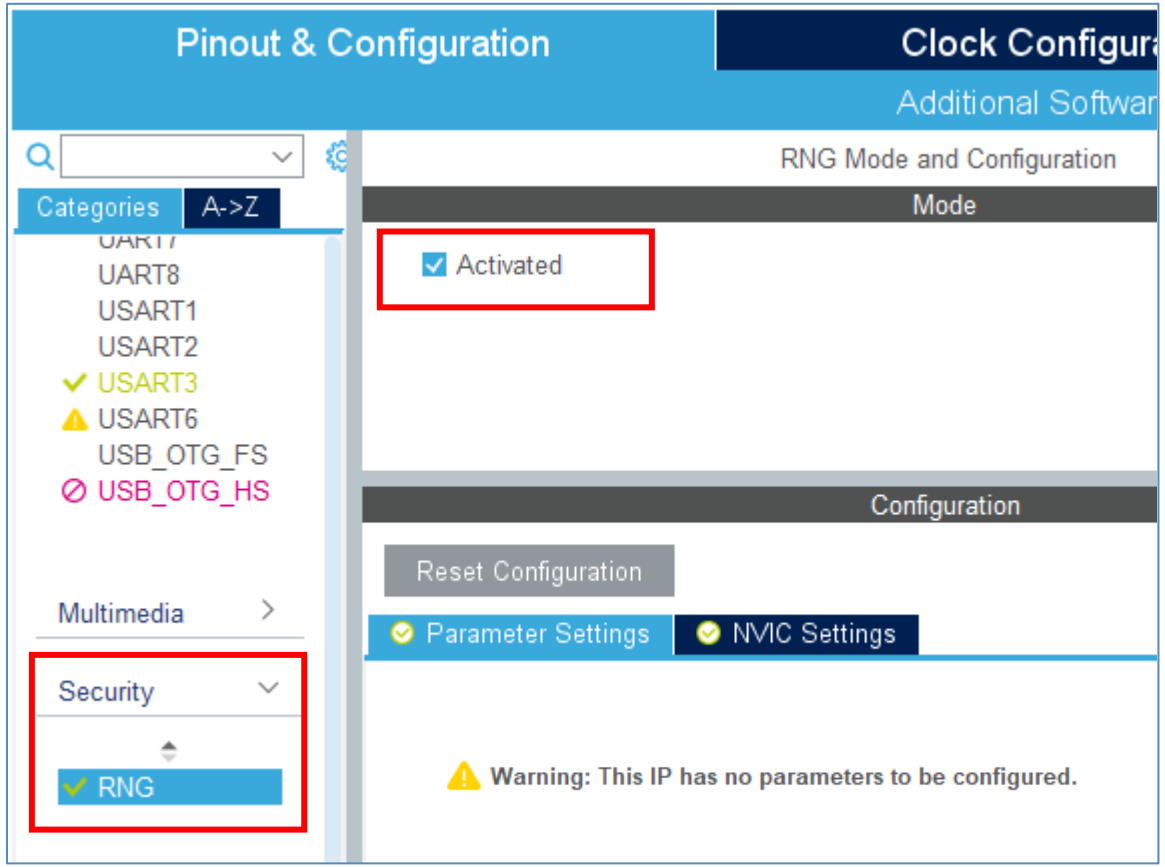
Setting STM32CubeMX : Cortex_M7




```
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* Enable I-Cache---------
  SCB_EnableICache();

  /* Enable D-Cache---------
  SCB_EnableDCache();
```

Setting STM32CubeMX :

```
uint32_t random_num = HAL_RNG_GetRandomNumber(&hrng);
```



The RNG processor is a random number generator, based on a continuous analog noise, that provides a random 32-bit value to the host when read.

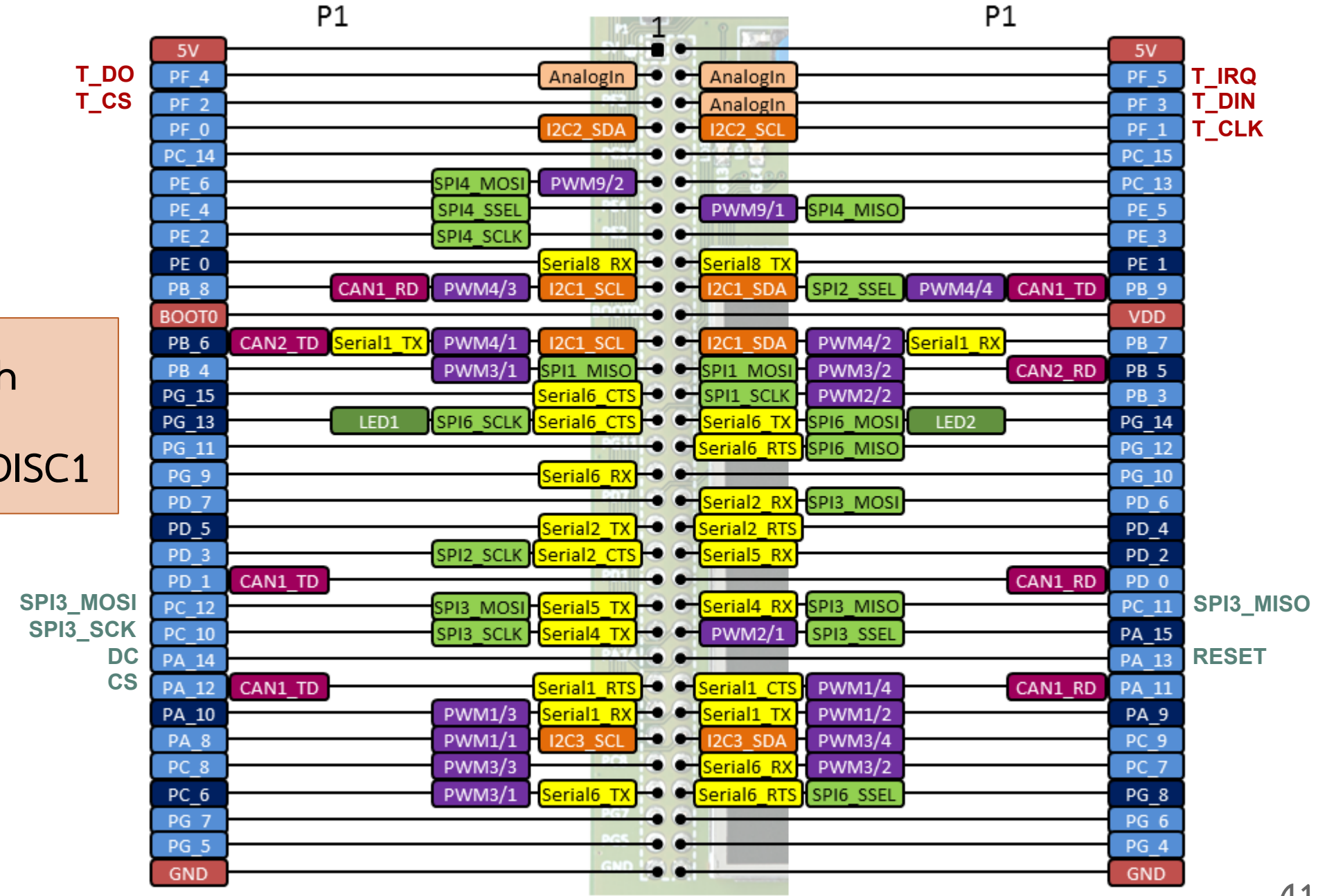
- It delivers **32-bit random numbers**, produced by an analog generator
- **40 periods** of the RNG_CLK clock signal between two consecutive random numbers

Interface with STM32F429-DISC1

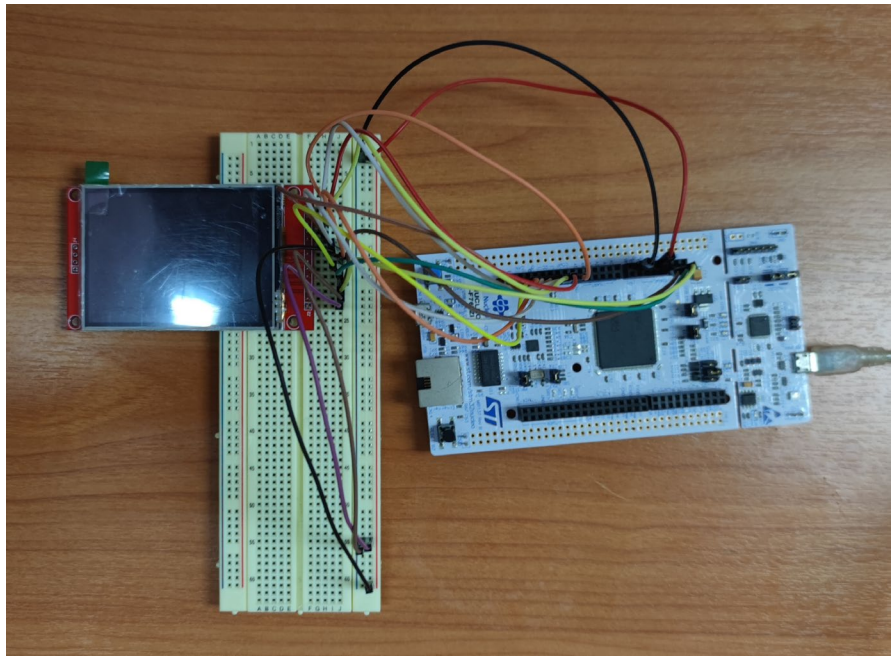
STM32F767ZI	Function	LCD	Note	
PF5	GPIO_Input	T_IRQ		Bit Banging
PF4	GPIO_Input	T_DO	MISO	Bit Banging
PF3	GPIO_Output	T_DIN	MOSI	Bit Banging
PF2	GPIO_Output	T_CS		Bit Banging
PF1	GPIO_Ouput	T_CLK		Bit Banging
PC11	SPI5_MISO	SDO	MISO	SPI3
3.3V		LED		
PC10	SPI_SCK	SCK		SPI3
PC12	SPI_MOSI	SDI	MOSI	SPI3
PA14	GPIO_Output	DC		
PA13	GPIO_Output	RESET		
PA12	GPIO_Output	CS		
GND		GND		
3.3V		Vcc		

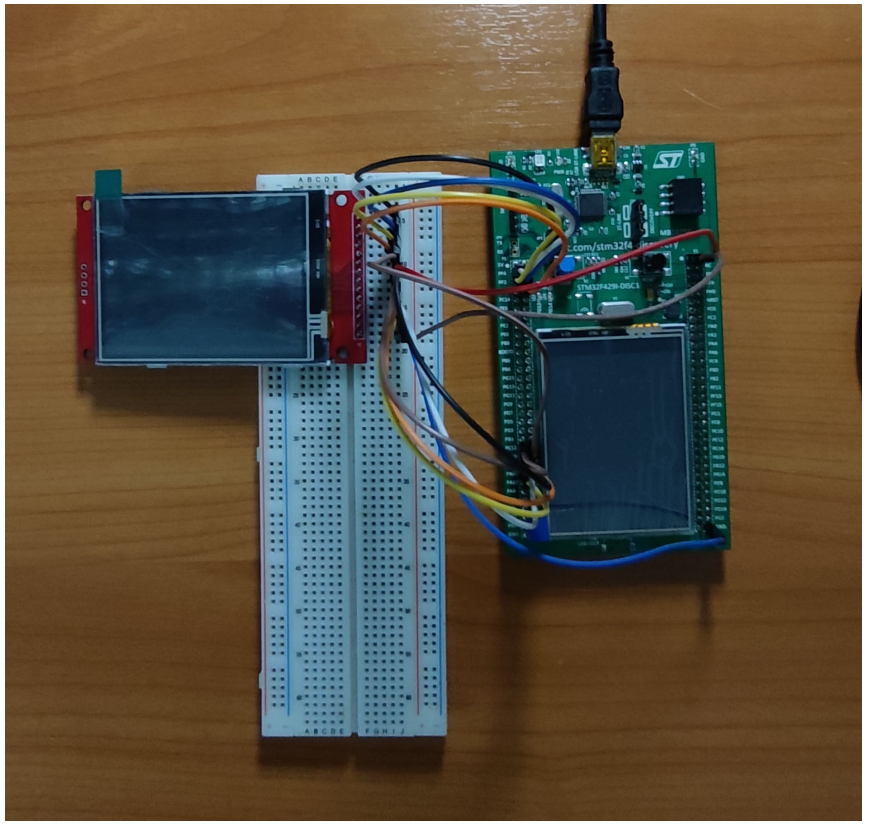
Interface with

STM32F429-DISC1



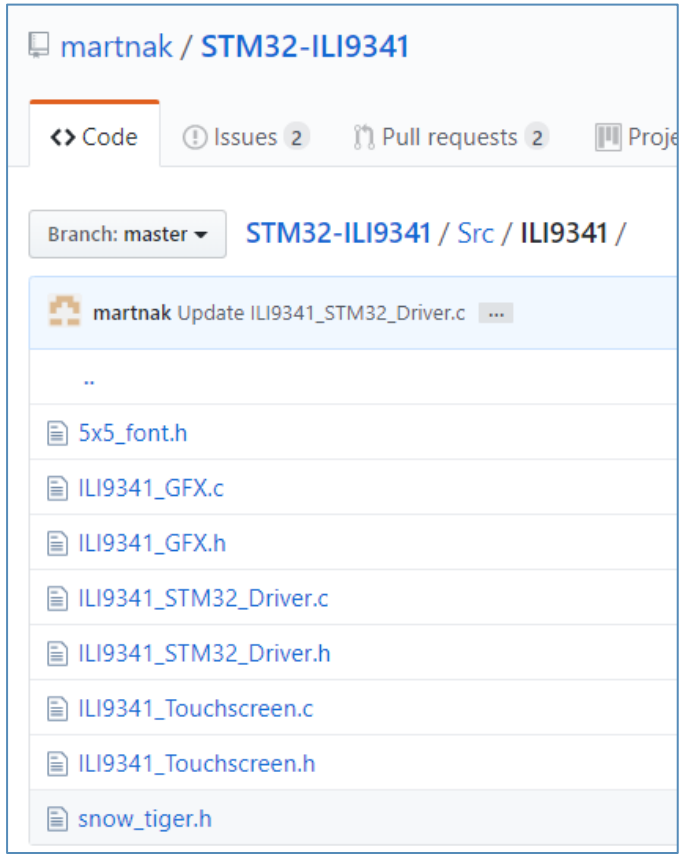
Wiring





LCD Lib

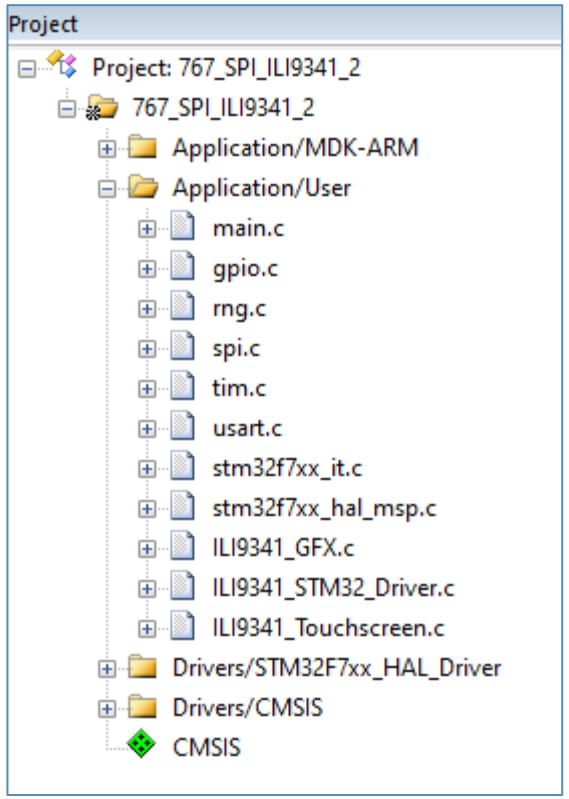
- https://github.com/martnak/STM32-ILI9341/tree/master/Src/ILI9341



Graphics

LCD Driver

SPI



Graphics Lib : ILI9341_GFX.h

```
#define HORIZONTAL_IMAGE  0
#define VERTICAL_IMAGE    1

void ILI9341_Draw_Hollow_Circle(uint16_t X, uint16_t Y, uint16_t Radius, uint16_t Colour);
void ILI9341_Draw_Filled_Circle(uint16_t X, uint16_t Y, uint16_t Radius, uint16_t Colour);

void ILI9341_Draw_Hollow_Rectangle_Coord(uint16_t X0, uint16_t Y0, uint16_t X1, uint16_t Y1, uint16_t Colour);
void ILI9341_Draw_Filled_Rectangle_Coord(uint16_t X0, uint16_t Y0, uint16_t X1, uint16_t Y1, uint16_t Colour);

void ILI9341_Draw_Char(char Character, uint8_t X, uint8_t Y, uint16_t Colour, uint16_t Size, uint16_t Background_Colour);
void ILI9341_Draw_Text(const char* Text, uint8_t X, uint8_t Y, uint16_t Colour, uint16_t Size, uint16_t Background_Colour);

void ILI9341_Draw_Filled_Rectangle_Size_Text(uint16_t X0, uint16_t Y0, uint16_t Size_X, uint16_t Size_Y, uint16_t Colour);

//USING CONVERTER: http://www.digole.com/tools/PicturetoC_Hex_converter.php
//65K colour (2Bytes / Pixel)
void ILI9341_Draw_Image(const char* Image_Array, uint8_t Orientation);
```

https://sourceforge.net/projects/lcd-image-converter/

Graphics Lib : ILI9341_STM32_Driver.h

```
void ILI9341_SPI_Init(void);
void ILI9341_SPI_Send(unsigned char SPI_Data);
void ILI9341_Write_Command(uint8_t Command);
void ILI9341_Write_Data(uint8_t Data);
void ILI9341_Set_Address(uint16_t X1, uint16_t Y1, uint16_t X2, uint16_t Y2);
void ILI9341_Reset(void);

void ILI9341_Set_Rotation(uint8_t Rotation);

void ILI9341_Enable(void);
void ILI9341_Init(void);

void ILI9341_Fill_Screen(uint16_t Colour);
void ILI9341_Draw_Colour(uint16_t Colour);
void ILI9341_Draw_Pixel(uint16_t X,uint16_t Y,uint16_t Colour);
void ILI9341_Draw_Colour_Burst(uint16_t Colour, uint32_t Size);


void ILI9341_Draw_Rectangle(uint16_t X, uint16_t Y, uint16_t Width, uint16_t Height, uint16_t Colour);
void ILI9341_Draw_Horizontal_Line(uint16_t X, uint16_t Y, uint16_t Width, uint16_t Colour);
void ILI9341_Draw_Vertical_Line(uint16_t X, uint16_t Y, uint16_t Height, uint16_t Colour);
```

LCD Screen Rotation

```
#define SCREEN_VERTICAL_1      0
#define SCREEN_HORIZONTAL_1    1
#define SCREEN_VERTICAL_2      2
#define SCREEN_HORIZONTAL_2    3
```

```
void ILI9341_Set_Rotation(uint8_t Rotation);
```

Colour

- Colour 16 bit
- Format R-G-B: 5-6-5 bit
- 65536 Colours

```
#define BLACK       0x0000
#define NAVY        0x000F
#define DARKGREEN   0x03E0
#define DARKCYAN    0x03EF
#define MAROON      0x7800
#define PURPLE      0x780F
#define OLIVE       0x7BE0
#define LIGHTGREY   0xC618
#define DARKGREY    0x7BEF
#define BLUE        0x001F
#define GREEN       0x07E0
#define CYAN        0x07FF
#define RED         0xF800
#define MAGENTA     0xF81F
#define YELLOW      0xFFE0
#define WHITE       0xFFFF
#define ORANGE      0xFD20
#define GREENYELLOW 0xAFE5
#define PINK        0xF81F
```

Touch Sensor Lib : ILI9341_Touchscreen.h

```
//Internal Touchpad command, do not call directly
uint16_t TP_Read(void);

//Internal Touchpad command, do not call directly
void TP_Write(uint8_t value);

//Read coordinates of touchscreen press. Position[0] = X, Position[1] = Y
uint8_t TP_Read_Coordinates(uint16_t Coordinates[2]);

//Check if Touchpad was pressed. Returns TOUCHPAD_PRESSED (1) or TOUCHPAD_NOT_PRESSED (0)
uint8_t TP_Touchpad_Pressed(void);
```

```
if(TP_Touchpad_Pressed())
  {
    uint16_t x_pos = 0;
    uint16_t y_pos = 0;

    uint16_t position_array[2];
    if(TP_Read_Coordinates(position_array) == TOUCHPAD_DATA_OK)
    {
    x_pos = position_array[0];
    y_pos = position_array[1];
    }
  }
```

Touch & Draw Circle

```
while(1)
{
  HAL_Delay(20);

  if(TP_Touchpad_Pressed())
    {

      uint16_t x_pos = 0;
      uint16_t y_pos = 0;


      HAL_GPIO_WritePin(GPIOB, LD3_Pin|LD2_Pin, GPIO_PIN_SET);

      uint16_t position_array[2];

      if(TP_Read_Coordinates(position_array) == TOUCHPAD_DATA_OK)
      {
      x_pos = position_array[0];
      y_pos = position_array[1];
      ILI9341_Draw_Filled_Circle(x_pos, y_pos, 2, BLACK);

      ILI9341_Set_Rotation(SCREEN_HORIZONTAL_1);
      char counter_buff[30];
      sprintf(counter_buff, "POS X: %.3d", x_pos);
      ILI9341_Draw_Text(counter_buff, 10, 80, BLACK, 2, WHITE);
      sprintf(counter_buff, "POS Y: %.3d", y_pos);
      ILI9341_Draw_Text(counter_buff, 10, 120, BLACK, 2, WHITE);
      ILI9341_Set_Rotation(SCREEN_VERTICAL_1);
      }

      ILI9341_Draw_Pixel(x_pos, y_pos, BLACK);

    }
  else
  {
    HAL_GPIO_WritePin(GPIOB, LD3_Pin|LD2_Pin, GPIO_PIN_RESET);
  }

}
```