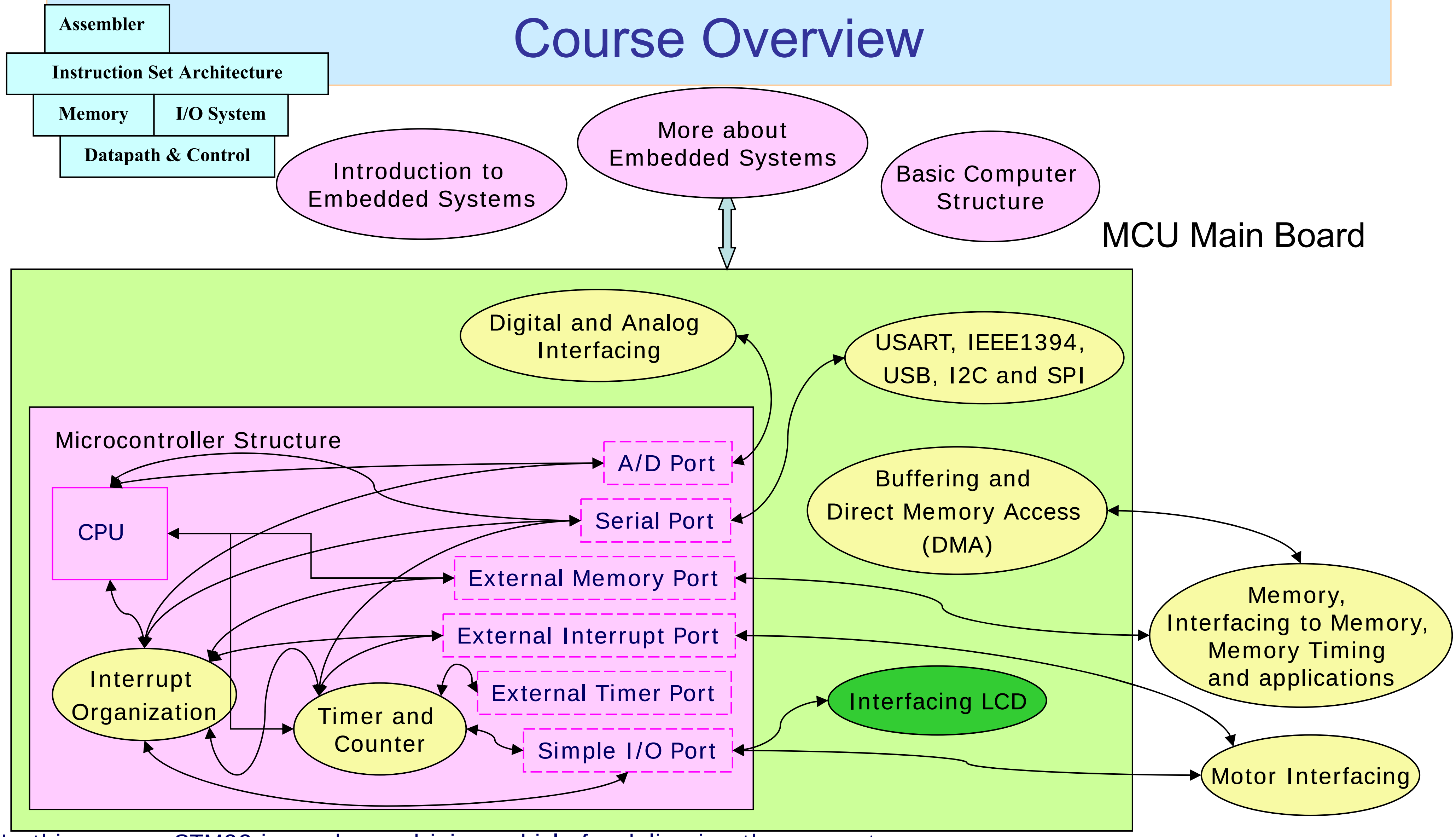
ELEC 3300
Introduction to Embedded Systems

Topic 5

Interfacing LCD

Prof. Tim Woo

Course Overview



Assembler

Instruction Set Architecture

Memory

I/O System

Datapath & Control

Introduction to Embedded Systems

More about Embedded Systems

Basic Computer Structure

MCU Main Board

Digital and Analog Interfacing

USART, IEEE1394, USB, I2C and SPI

Buffering and Direct Memory Access (DMA)

Memory, Interfacing to Memory, Memory Timing and applications

Interfacing LCD

Motor Interfacing

Microcontroller Structure

CPU

A/D Port

Serial Port

External Memory Port

External Interrupt Port

External Timer Port

Simple I/O Port

Interrupt Organization

Timer and Counter

In this course, STM32 is used as a driving vehicle for delivering the concepts.

To be covered	In progress	Done

Expected Outcomes

- On successful completion of this topic, you will be able to
  - Introduce several types of LCD
  - Understand the drivers of graphic type LCD module (in Lab 3)
  - Interface the character type LCD module with a ARM microprocessor including both initialization and data communication modes

LED v/s LCD

- Both LEDs and LCDs use liquid crystals to help create an image.
- The difference between the two is the placement and type of backlight used to illuminate the pixels.
- LEDs use light emitting diodes while LCDs use fluorescent lights for backlights.
- LED also uses liquid crystals, so an “LED monitor” should be technically called “LED LCD monitor.”
- All LED monitors are LCD monitors. But not all LCD monitors are LEDs.
- LEDs are slimmer than LCDs and provide a better quality, clearer picture with high definition output.
- OLED:

(Organic LED) has a film of an organic compound that emits light in response to electricity (emits their own light without backlights).

Common types of LCDs

Character Type

Graphic Type

Flexible OLED

Alphanumeric Type

TFT (Thin film Transistor)

Low power, cheaper, Low image quality and longer response time (than TFT)

CSTN (Color Super-Twisted Nematic)

Graphic type LCD

A typical graphic type LCD has

- 8-bit data bus : D0-D7

- 3-bit control bus: E, RS, R/W

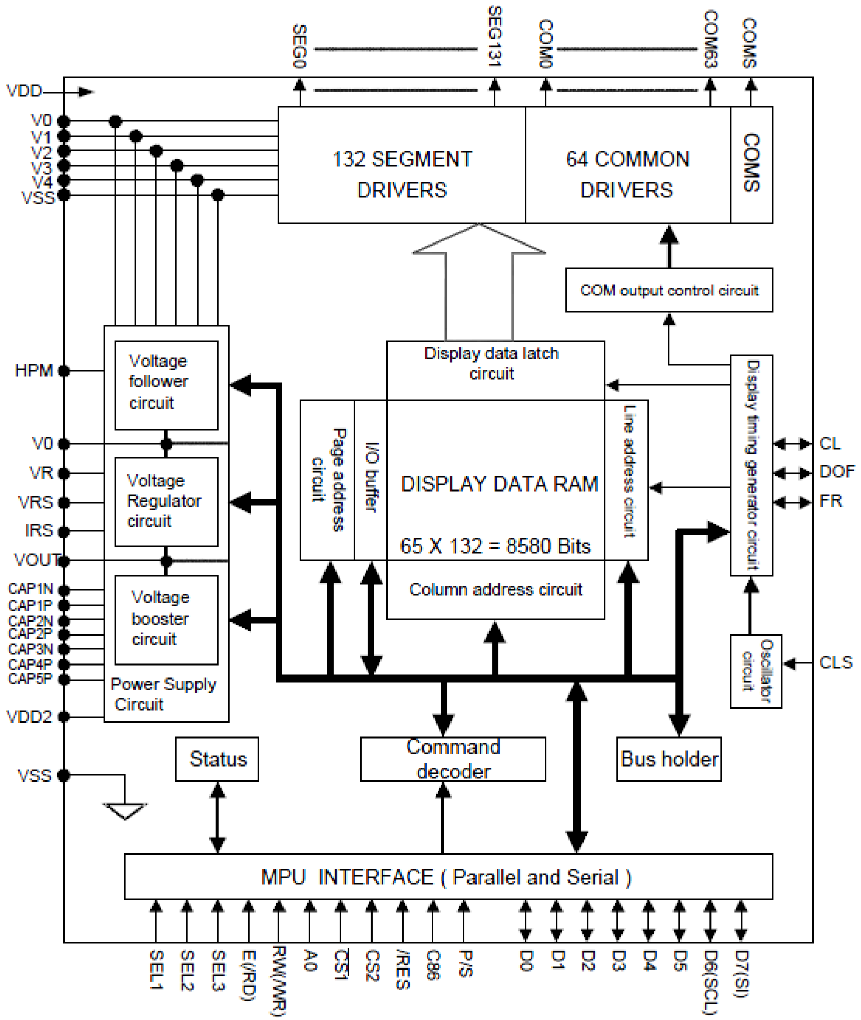
The controller uses RS and RW lines along with E to operate the LCD.

Register Select (RS): Determines weather a command (RS = 0) is sent to set up the display or actual data (RS=1) is sent.

Read/Write RW=0; writes to the LCD. RW=1; Reads from the LCD.

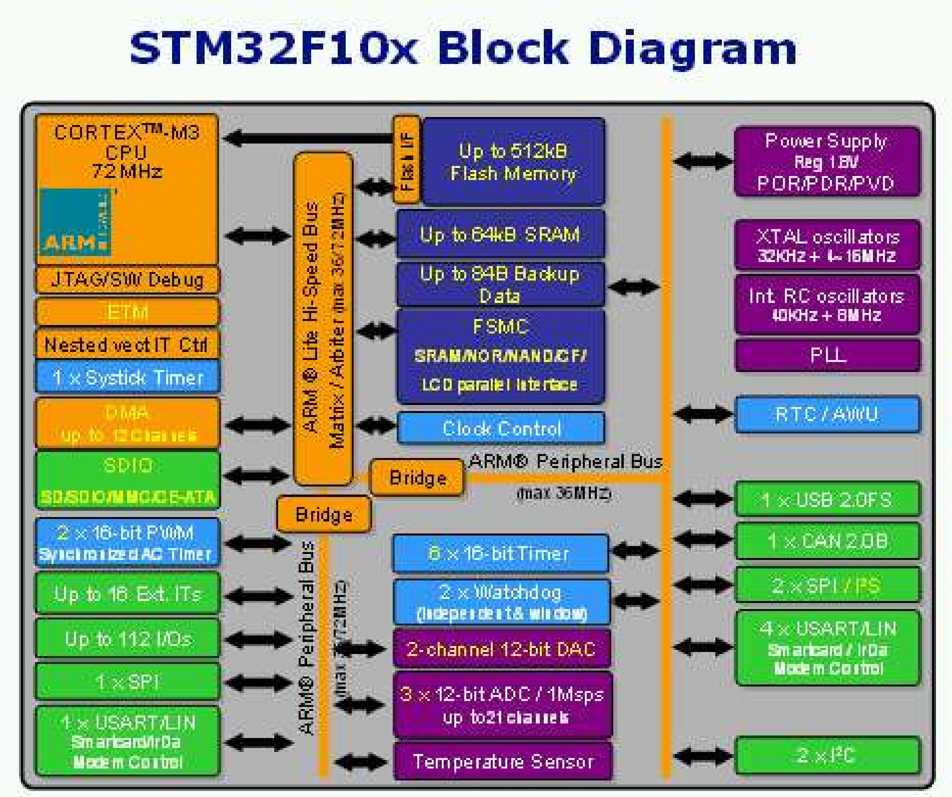
Enable (E) signal is used to Latch the information on data bus.

- X power lines: adjust the power level (for adjusting the brightness level)



Interface between the STM32 and LCD controller

- There are two interfacing techniques
  - GPIO
  - FSMC (Flexible Static Memory Controller)
- In lab session, FSMC is applied – It provides library functions.

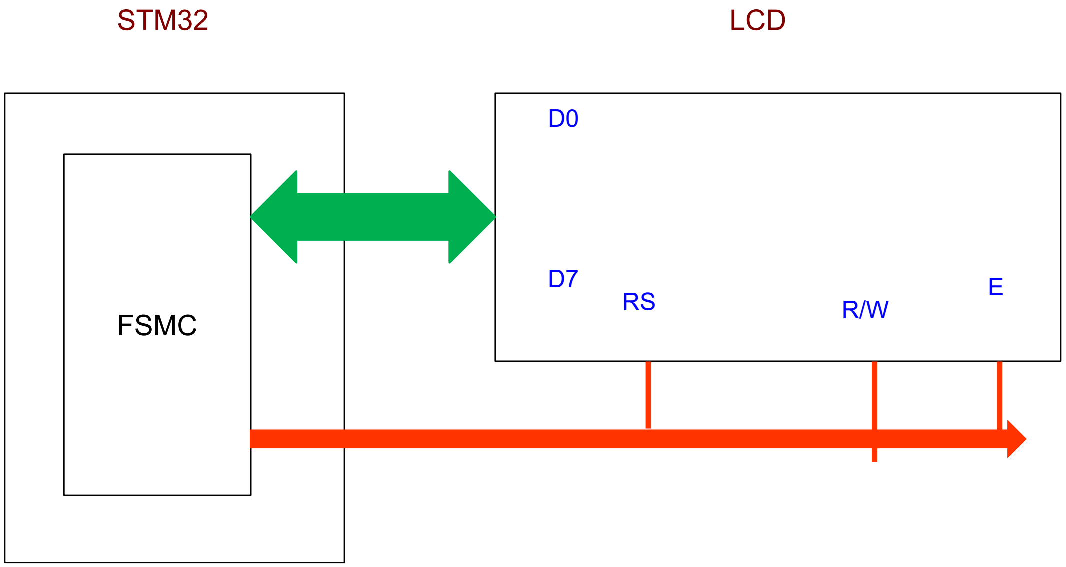

STM32F10x Block Diagram
CORTEX™-M3 CPU 72 MHz
ARM
JTAG/SW Debug
ETM
Nested vect IT Ctrl
1 x Systick Timer
DMA up to 12 Channels
SDIO SD/SDIO/MMC/CE-ATA
2 x 16-bit PWM Synchronized AC Timer
Up to 16 Ext. ITs
Up to 112 I/Os
1 x SPI
1 x USART/LIN Smartcard/IrDa Modem Control
ARM ® Lite Hi-Speed Bus Matrix / Arbiter (max 36/72MHz)
Flash I/F
Up to 512kB Flash Memory
Up to 64kB SRAM
Up to 84B Backup Data
FSMC SRAM/NOR/NAND/CF/ LCD parallel interface
Clock Control
Bridge
ARM® Peripheral Bus (max 36MHz)
Bridge
ARM® Peripheral Bus (max 72MHz)
6 x 16-bit Timer
2 x Watchdog (independent & window)
2-channel 12-bit DAC
3 x 12-bit ADC / 1Msps up to21 channels
Temperature Sensor
Power Supply Reg 1.8V POR/PDR/PVD
XTAL oscillators 32KHz + 4~16MHz
Int. RC oscillators 40KHz + 8MHz
PLL
RTC / AWU
1 x USB 2.0FS
1 x CAN 2.0B
2 x SPI / I²S
4 x USART/LIN Smartcard / IrDa Modem Control
2 x I²C

Examples of LCD interface

Description	Choices in this course
Abstract idea of project (Define the functionality of the system)	Clear the LCD Display a dot (value2) in specified location (location)
Data format / representation	8 bits
Programming Language	C-language
Communication Protocol	FSMC
Physical connection (Pins assignment)	Pins for FSMC
Hardware devices (Microcontroller, Peripherals)	Microcontroller: STM32 ARM Platform Peripherals: ST7565R:55 x 132 Dot Matrix LCD Controller / Driver

Examples of LCD interface

- Tasks:
  - Clear the LCD
  - Display a dot (value2) in specified location (location)

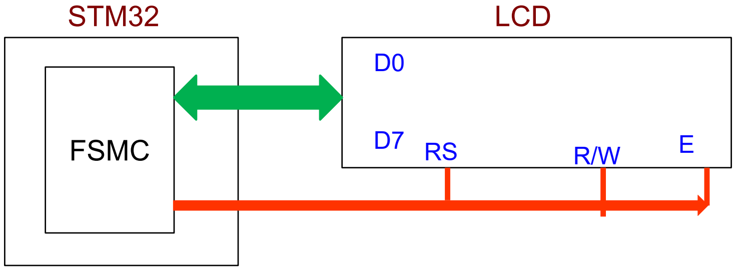


Configuration setting can be done using STM32 CubeMX

•ST7565R:55 x 132 Dot Matrix LCD Controller / Driver

Examples of LCD interface

- Tasks:
  - Clear the LCD
  - Display a dot (value2) in specified location (location)



Initialization

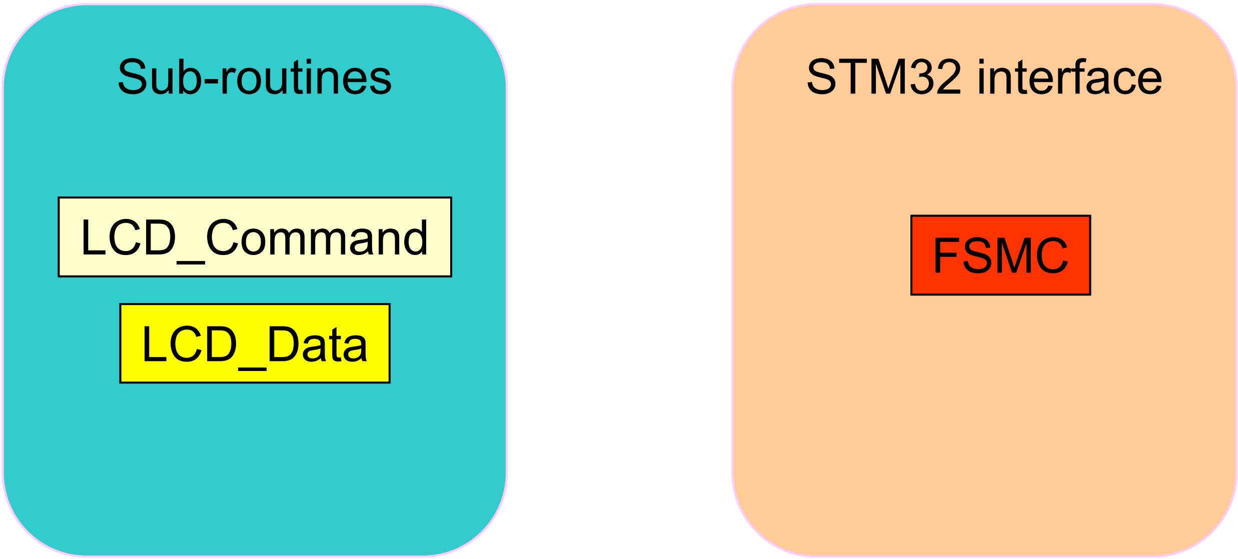
```
Void Main{
    Initialization of FSMC
    LCD_Command = <value0>   ; LCD size / dimension
    LCD_Command = <value1>   ; Clear LCD
….


    LCD_Command = <location> ; Specify the location
    LCD_Data = <value2>      ; Display data at the specified location
}
```

implementation

Note:
LCD_Command and LCD_Data are assignment statements that controlled by FSMC.

LCD Interface

- In the laboratory experiment, we have



- What can we do if neither sub-routines nor FSMC are provided?

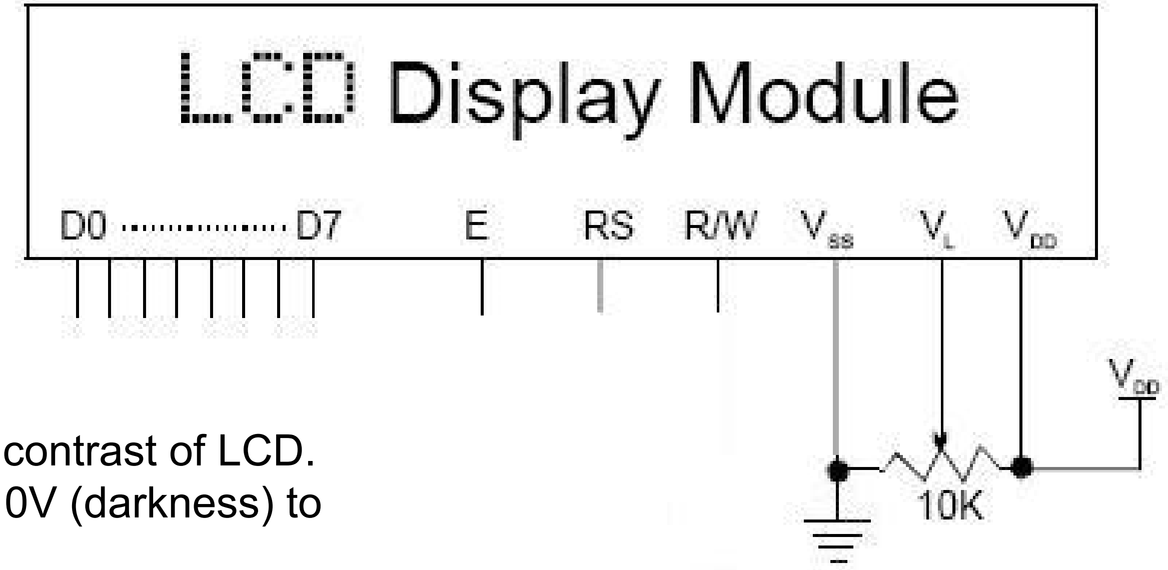
Remember: STM32 can support two configurations: General devices (without Ethernet access) and Connectivity line (with Ethernet access) – latter has no FSMC.

Can we use GPIO?

Understand hardware component: Character type LCD

- A typical character type LCD has
  - 8-bit data bus : D0-D7
  - 3-bit control bus: E, RS, R/W
  - 3 power lines: V_{SS}, V_{DD}, V_L

Example: TM162A: 16 characters / line x 2 lines
Module number: TM162AAAU6

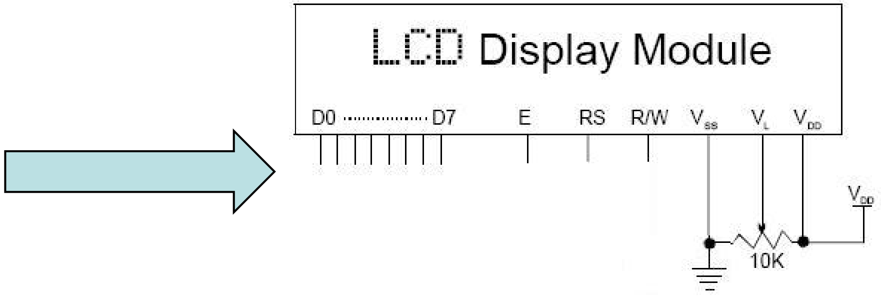


Note:
V_L is used to fine tune the contrast of LCD.
Typical value ranges from 0V (darkness) to 0.3V (brightness)
Attention: LCD may burn if V_L > 1.6V.

Design architecture

Description	Choices in this course
Abstract idea of project (Define the functionality of the system)	Clear the LCD Display a dot (value2) in specified location (location)
Data format / representation	8 bits
Programming Language	C-language
Communication Protocol	GPIO
Physical connection (Pins assignment)	Pins for GPIO
Hardware devices (Microcontroller, Peripherals)	Microcontroller: STM32 ARM Platform Peripherals: TM162A: 16 characters / line x 2 lines

Understand hardware component: Character type LCD Interfacing with GPIO

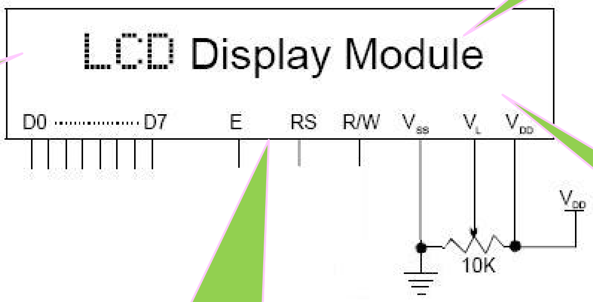


Description
Abstract idea of project (Define the functionality of the system)
Data format / representation
Programming Language
Communication Protocol
Physical connection (Pins assignment)
Hardware devices (Microcontroller, Peripherals)

Physical Devices	Pin Assignment	Signal Type	Initialization (Configuration)	Signals at Physical connection
LCD Display Module	General Purpose Input & Output	~~Input~~ / Output	General Purpose IO setting For D0-D7, E, RS, R/W	Particular Binary sequence (provided by the manufacturer)

Understand hardware component: Character type LCD

- **Questions:**
  - Does the LCD have read and write modes?



How to display these texts?

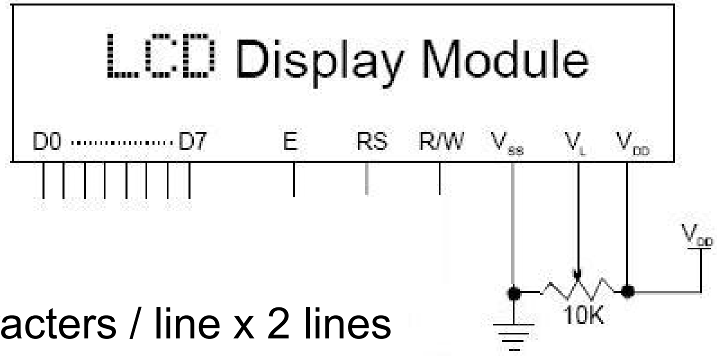
How to clear all the texts?

How to write a sub-routine LCD_Command ?

How to write a sub-routine LCD_Data ?

Understand hardware component: Character type LCD

- Pay attention on three control signal lines and the instruction code:
  - Register-Select (RS): an input line to steer the use of command register or data register
  - Read/Write (R/W): an input line to control read or write
  - Enable (E): Used to latch information presented to its data buses
  - Instruction code: the library for activating the LCD

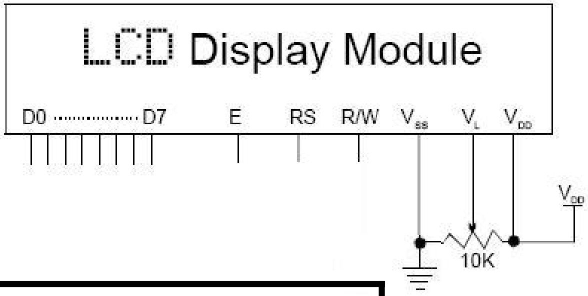


Example: TM162A: 16 characters / line x 2 lines
Module number: TM162AAAU6

From datasheets: Signaling of character type LCD

- D0-D7: 8-bit data bus

  What are the signal type of D0-D7 when R/W=1?

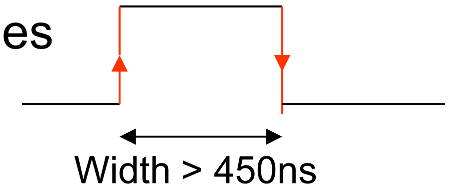


- 4 different operations:

	RS = 0	RS = 1
R/W = 0	Write command to LCD module	Write data to LCD module
R/W =1	Read the status of LCD module	Read data from LCD module

  – The busy flag (D7) is used to check whether the LCD is ready to receive information
    - when D7=1, LCD is busy and will not accept any new information
    - when D7=0, LCD is ready to receive and new information

- The operation is activated by applying an edge-detection pulse in Enable pin.
  – Is used by LCD to latch information presented to its data buses



From datasheet: **Write mode** of character type LCD

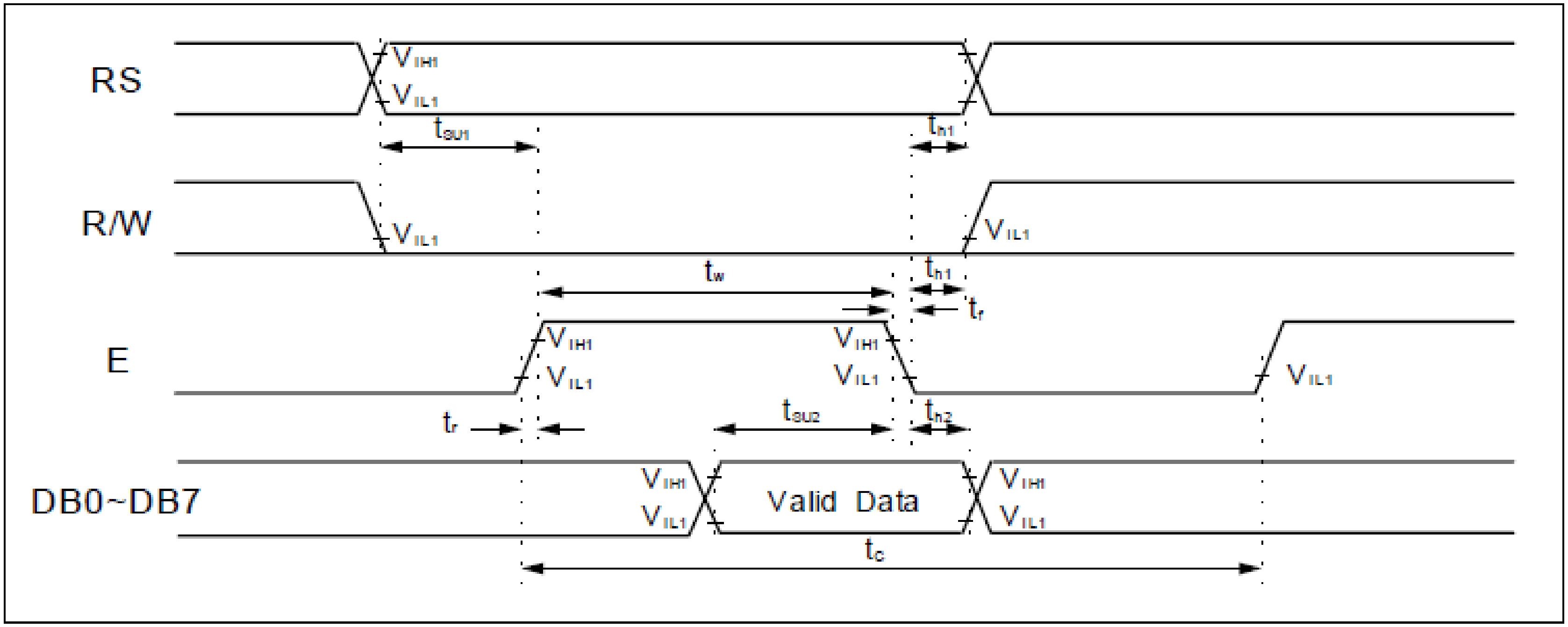


Figure 6 . Write Mode Timing Diagram

RS=0/1 → Command/Data

R/W=0/1 → Write/Read

E=0/1 → Unlatch/Latch

Mode	Characteristic	Symbol	Min.	Typ.	Max.	Unit
Write Mode (Refer to Fig-6)	E Cycle Time	tc	500	-	-	ns
	E Rise / Fall Time	t_R,t_F	-	-	20	
	E Pulse Width (High, Low)	tw	230	-	-	
	R/W and RS Setup Time	tsu1	40	-	-	
	R/W and RS Hold Time	t_{H1}	10	-	-	
	Data Setup Time	tsu2	80	-	-	
	Data Hold Time	t_{H2}	10	-	-	

From datasheet: **Read mode** of character type LCD

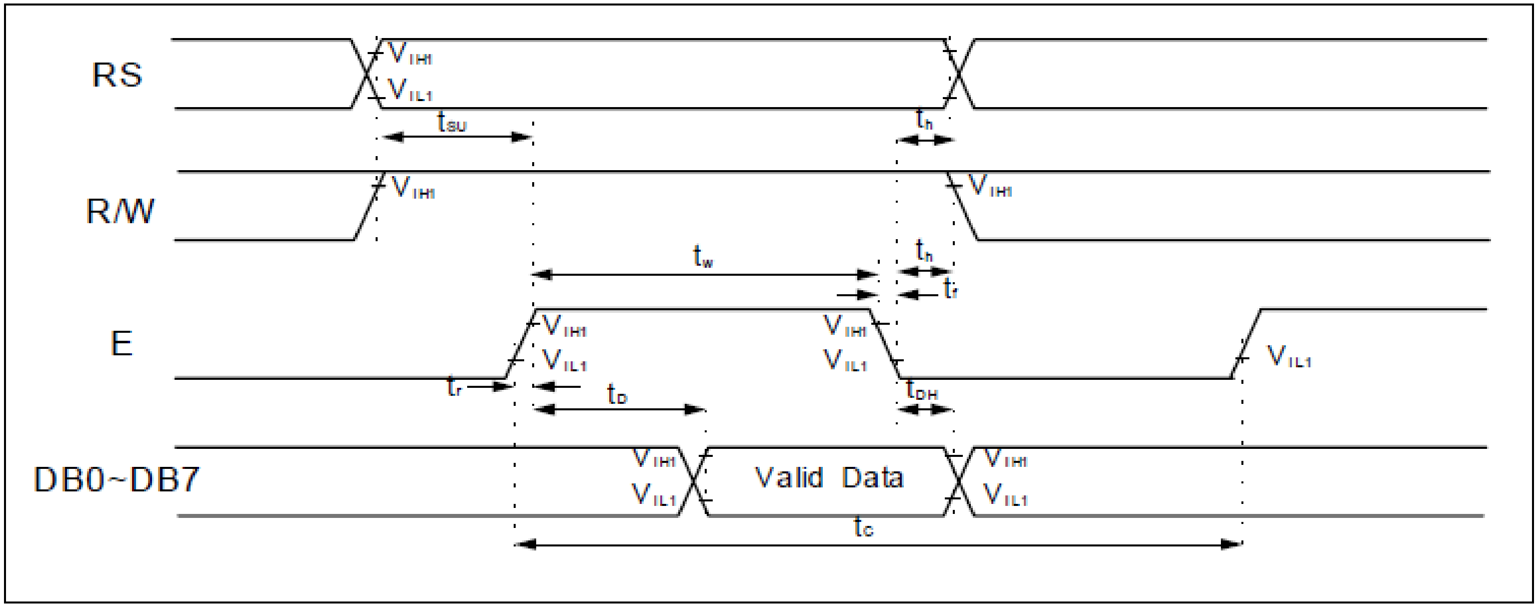


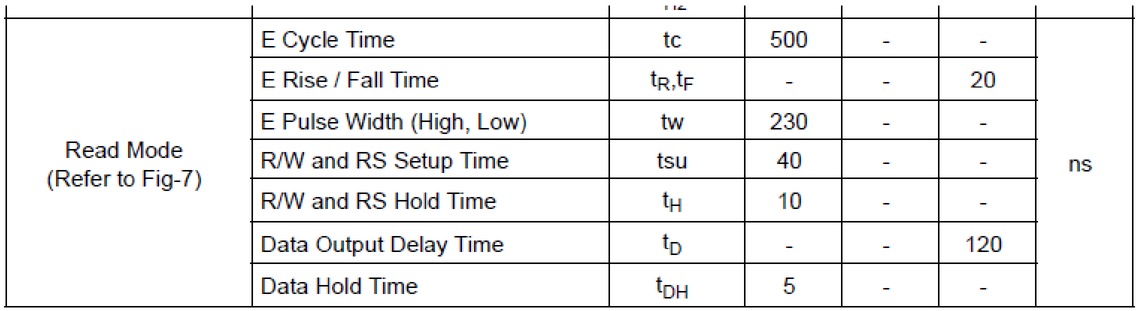
Figure 7 . Read Mode Timing Diagram

RS=0/1 → Command/Data

R/W=0/1 → Write/Read

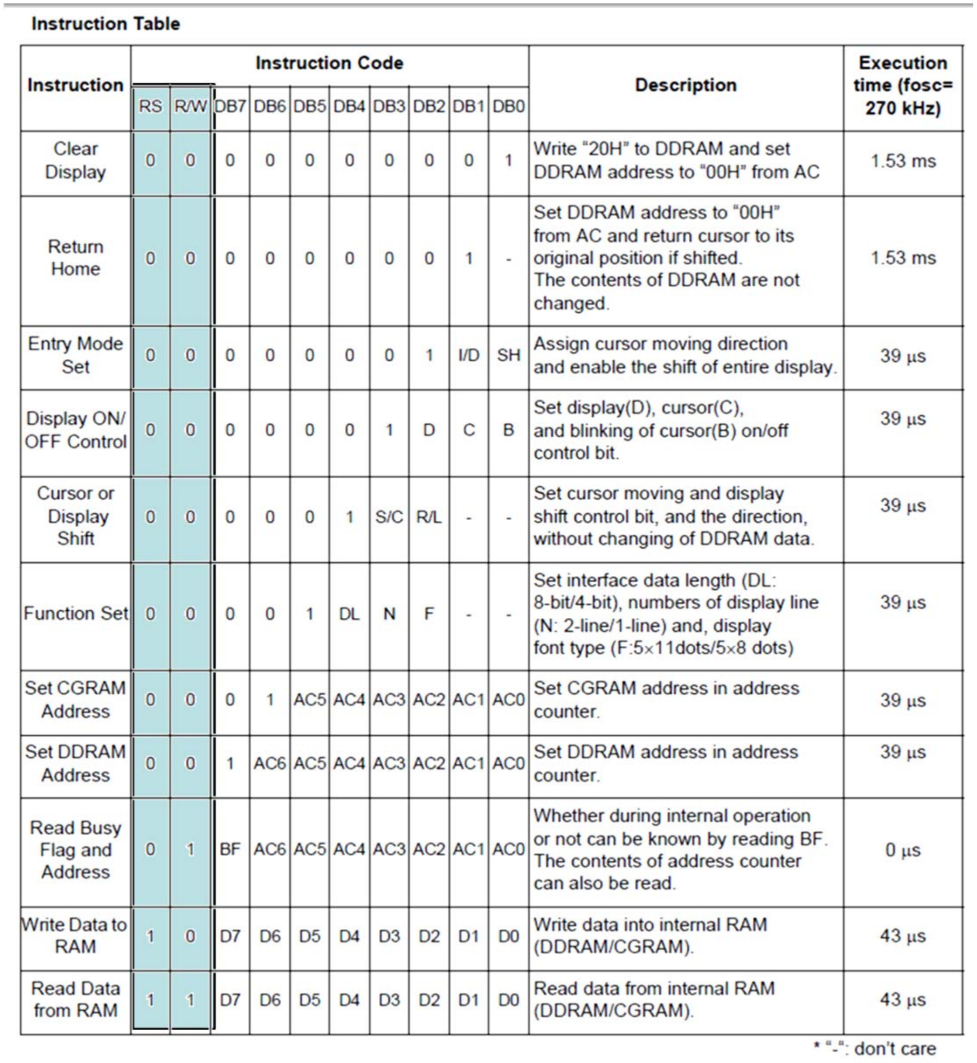
E=0/1 → Unlatch/Latch

- Note: Data (DB0~DB7) should be ready after triggering the write operation (E)



Read Mode (Refer to Fig-7)	E Cycle Time	tc	500	-	-	ns
	E Rise / Fall Time	t_R,t_F	-	-	20	
	E Pulse Width (High, Low)	tw	230	-	-	
	R/W and RS Setup Time	tsu	40	-	-	
	R/W and RS Hold Time	t_H	10	-	-	
	Data Output Delay Time	t_D	-	-	120	
	Data Hold Time	t_{DH}	5	-	-	

From datasheet: LCD Command Codes



Instruction Table

Instruction	Instruction Code										Description	Execution time (fosc= 270 kHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRAM and set DDRAM address to "00H" from AC	1.53 ms
Return Home	0	0	0	0	0	0	0	0	1	-	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.	1.53 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	SH	Assign cursor moving direction and enable the shift of entire display.	39 µs
Display ON/ OFF Control	0	0	0	0	0	0	1	D	C	B	Set display(D), cursor(C), and blinking of cursor(B) on/off control bit.	39 µs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	-	-	Set cursor moving and display shift control bit, and the direction, without changing of DDRAM data.	39 µs
Function Set	0	0	0	0	1	DL	N	F	-	-	Set interface data length (DL: 8-bit/4-bit), numbers of display line (N: 2-line/1-line) and, display font type (F:5×11dots/5×8 dots)	39 µs
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter.	39 µs
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter.	39 µs
Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.	0 µs
Write Data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM).	43 µs
Read Data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM).	43 µs

* "-": don't care

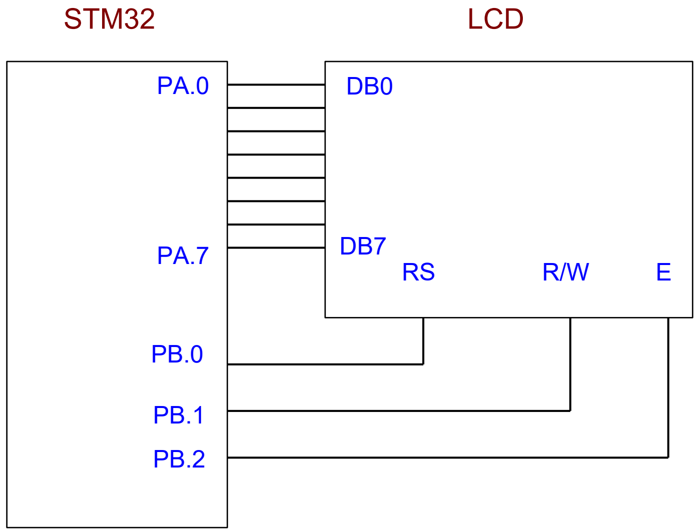
Example:

Command (hex) D7~D0	Description
01	Clear the display
02 (or 03)	Return home
06	Entry Mode Set (address increment)
0A	Display on, cursor off
14	Shift cursor to right
38	***Function set:*** ***8-bit data length*** ***2-line display*** ***Font type : 5 x 8 dots***

Provided by manufacturer

Interfacing LCD module using GPIO

- Let's start



Description
Abstract idea of project (Define the functionality of the system)
Data format / representation
Programming Language
Communication Protocol
Physical connection (Pins assignment) STM32 ←→ LCD Commend signals: Data signals: Control signals:
Hardware devices (Microcontroller, Peripherals) STM32 TM162A: 16 characters / line x 2 lines

ARM codes for interfacing LCD module using GPIO

- A procedure of writing a command onto LCD, LCD_Command
- Notes: RS = 0 (Command), R/W = 0 (Write)

Which port do they are connected to?

void LCD_Command(int number)
{
    GPIOB→ODR = 0x00;
    Delay(> t_F+t_{su1});
    GPIOB→ODR = 0x04;
    Delay(> t_r +t_w-t_{su2});
    GPIOA→ODR = <number>;
    Delay(> t_{su2});
    GPIOB→ODR = 0x00;
    Delay(> t_R);
}

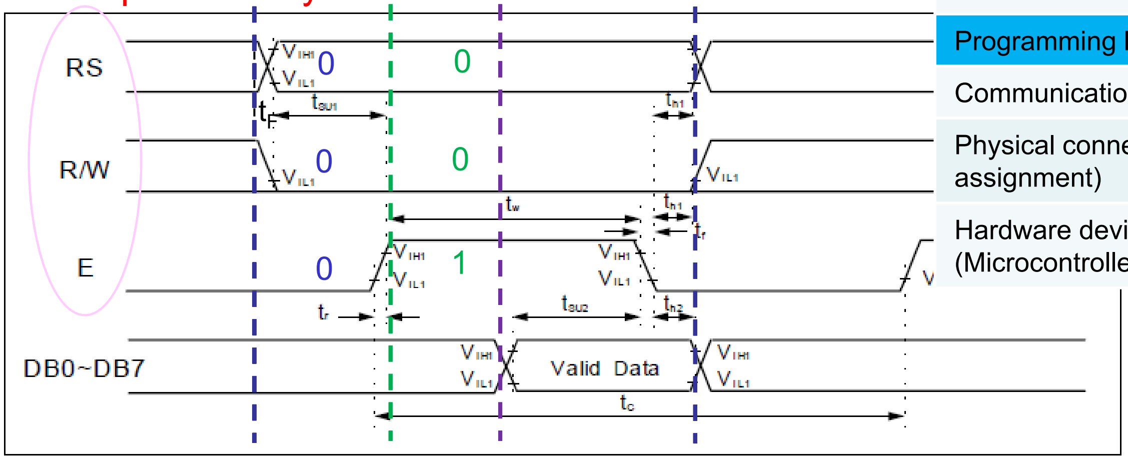
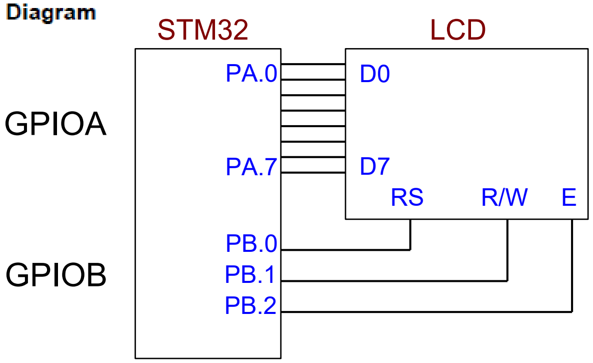


Figure 6 . Write Mode Timing Diagram

Mode	Characteristic	Symbol	Min.	Typ.	Max.	Unit
Write Mode (Refer to Fig-6)	E Cycle Time	tc	500	-	-	ns
	E Rise / Fall Time	t_R,t_F	-	-	20	
	E Pulse Width (High, Low)	tw	230	-	-	
	R/W and RS Setup Time	tsu1	40	-	-	
	R/W and RS Hold Time	t_{H1}	10	-	-	
	Data Setup Time	tsu2	80	-	-	
	Data Hold Time	t_{H2}	10	-	-	



ODR (Output Data Register) holds all outputs of the Pins of a GPIO port to the stated value (example: 0x00)

Description
- Abstract idea of project (Define the functionality of the system)
- Data format / representation
- Programming Language
- Communication Protocol
- Physical connection (Pins assignment)
- Hardware devices (Microcontroller, Peripherals)

ARM codes for interfacing LCD module using GPIO

- A procedure of writing a data onto LCD, LCD_Data
- Notes: RS = 1 (Data), R/W = 0 (Write)

Description

Abstract idea of project (Define the functionality of the system)

Data format / representation

Programming Language

Communication Protocol

Physical connection (Pins assignment)

Hardware devices (Microcontroller, Peripherals)

```
void LCD_Data(int number)
{
    GPIOB→ODR = 0x01;
    Delay(> tF+tsu1);
    GPIOB→ODR = 0x05;
    Delay(> tR +tw-tsu2);
    GPIOA→ODR = <number>;
    Delay(> tsu2);
    GPIOB→ODR = 0x01;
    Delay(> tR);
}
```

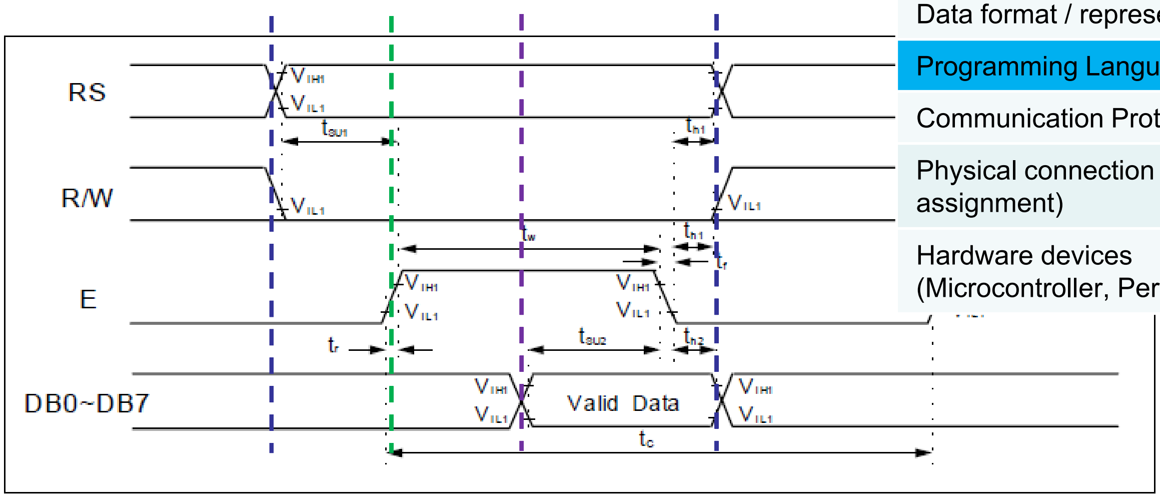
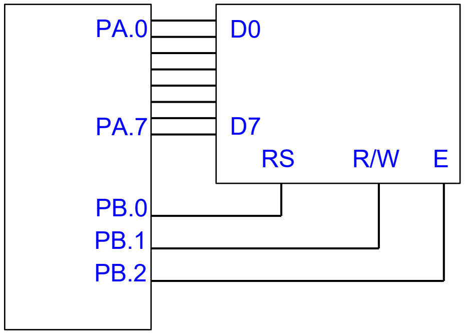


Figure 6 . Write Mode Timing Diagram

Mode	Characteristic	Symbol	Min.	Typ.	Max.	Unit
Write Mode (Refer to Fig-6)	E Cycle Time	tc	500	-	-	ns
	E Rise / Fall Time	t_R,t_F	-	-	20	
	E Pulse Width (High, Low)	tw	230	-	-	
	R/W and RS Setup Time	tsu1	40	-	-	
	R/W and RS Hold Time	t_{H1}	10	-	-	
	Data Setup Time	tsu2	80	-	-	
	Data Hold Time	t_{H2}	10	-	-	



Examples of LCD interface with GPIO

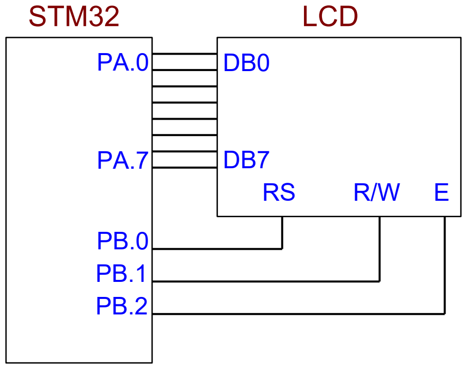
- Tasks:
  - Clear the LCD
  - Display a blinking cursor at beginning of 1st line
  - Display string “N1” onto the LCD
  - Display a blinking cursor followed by the string

Initialization

```
void LCD_Command(int number)
void LCD_Data(int number)
Void Main{
        LCD_Command(0x38)      ; LCD : 2 lines, 16 Character, font size
        LCD_Command (0x01)    ; Clear LCD
        LCD_Command (0x06)    ; Entry mode set
        LCD_Command(0x0F)     ; Cursor blinking

        LCD_Data(‘N’)             ; Display a character “N”
        LCD_Data(0x31)            ; Display a character “1” (hex code)
}
```

implementation

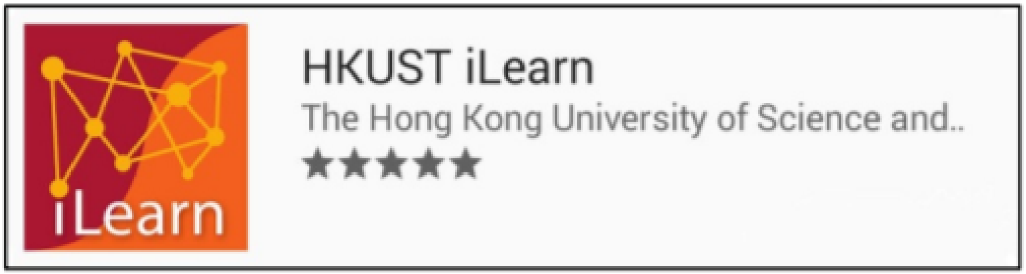
0x31 is the ASCII code of ‘1’.



Command (hex) DB7~DB0	**Description**
01	Clear the display
02 (or 03)	Return home
0A	Display on, cursor off
14	Shift cursor to right
06	Entry Mode Set (address increment)
38	***Function set:*** ***8-bit data length*** ***2-line display*** ***Font type : 5 x 8 dots***

In-class activities (Topic 5 Questions 1, 2)

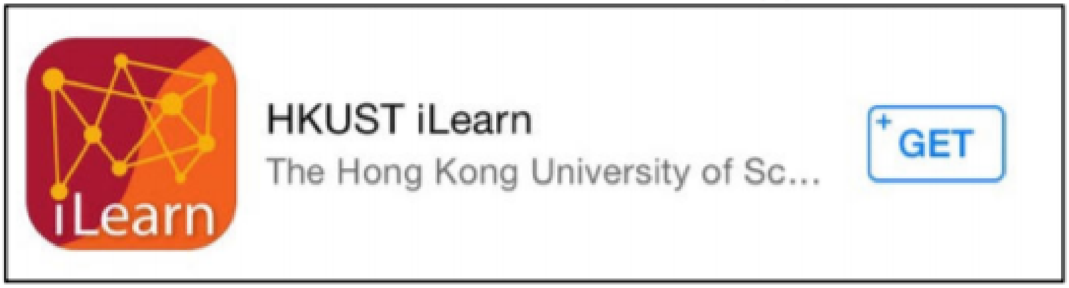
For Android devices, search **HKUST iLearn** at Play Store.



HKUST iLearn

The Hong Kong University of Science and..

For iOS devices, search **HKUST iLearn** at App Store.



HKUST iLearn

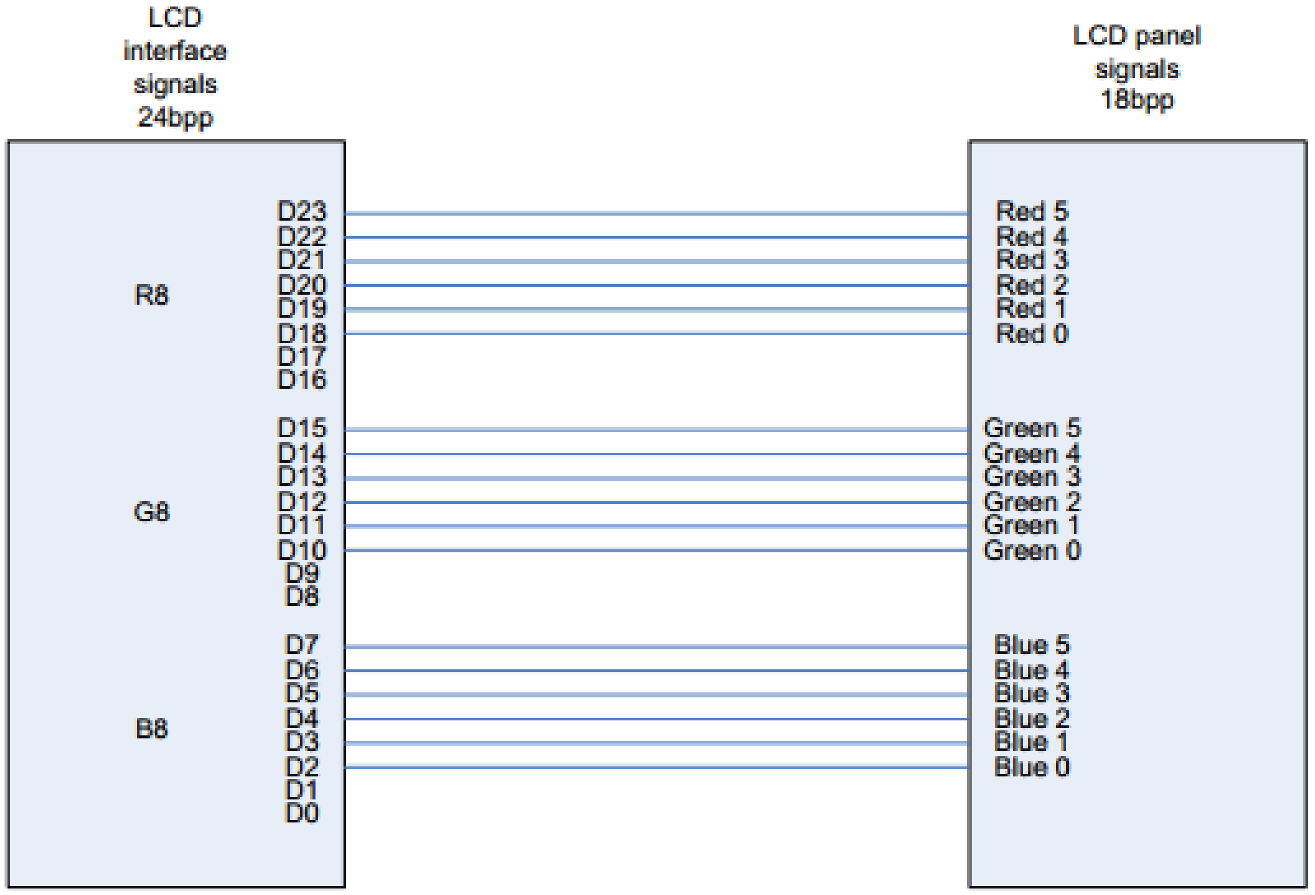
The Hong Kong University of Sc...

Color patterns

- Digital systems usually stored data in RGB (red-green-blue) colorspace format
  - Red, green, and blue components are bitfields of a pixel's color value
    - Usually referred to as bits per pixel (bpp)
  - RGB332 → 8 bpp (Red 3, Green 3, Blue 2)
    - Organized as a byte in memory as (RRR GGG BB)
  - RGB555 → 16 bpp (Red 5, Green 5, Blue 5)
    - Organized as a half-word in memory (U RRRRR GGGGG BBBBB)
  - RGB565 → 16 bpp (Red 5, Green 6, Blue 5)
    - Organized as a half-word in memory (RRRRR GGGGGG BBBBB)
  - RGB888 → 24 bpp (Red 8, Green 8, Blue 8)
    - Organized as a word (32-bit) in memory
    - (UUUUUUUU RRRRRRRR GGGGGGGG BBBBBBBB) (U = unused)

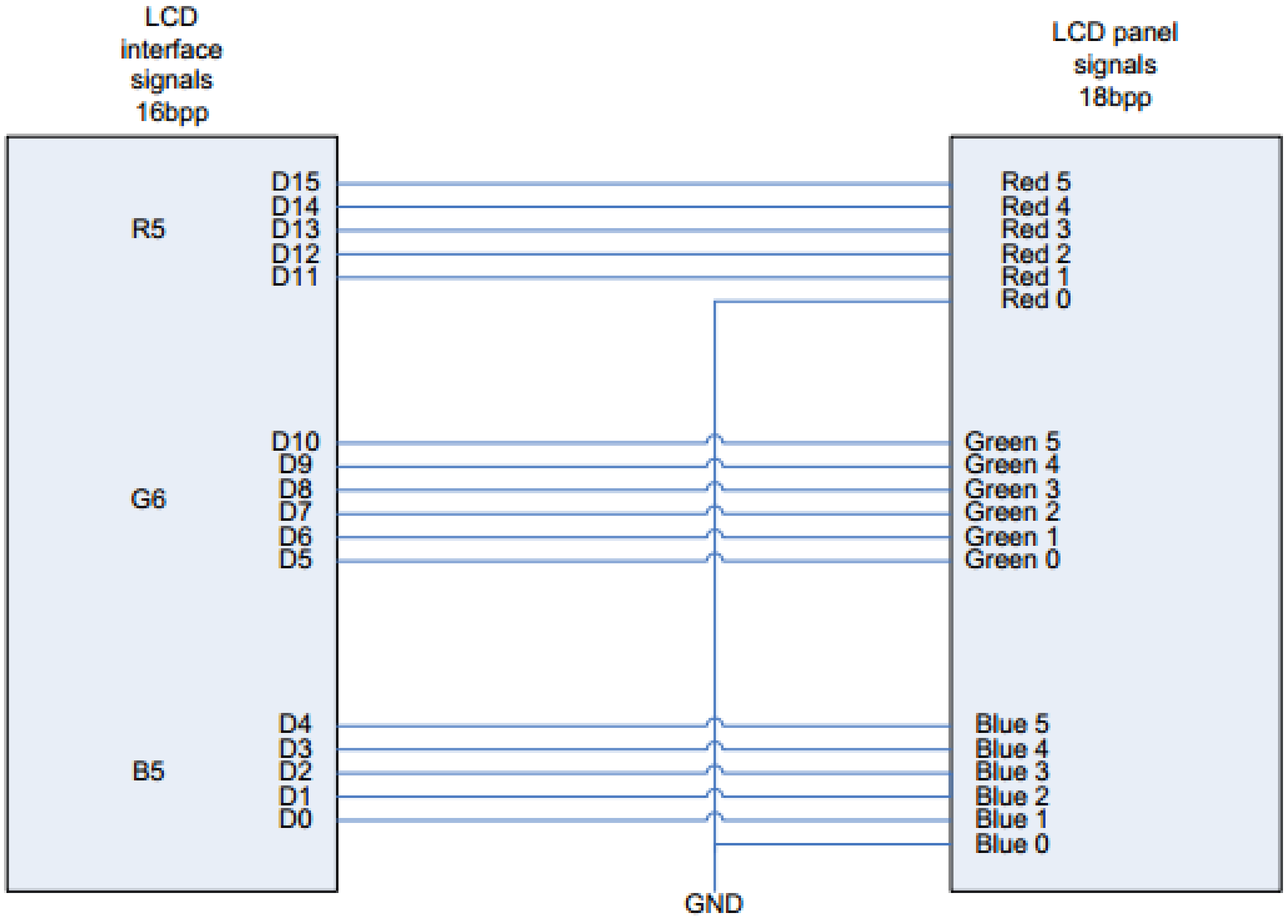
24-bit RGB888 interface to 18bpp LCD panel TFT example

Can keep LCD interface signals unused for interfaces with more signals than the LCD



16-bit RGB565 interface to 18bpp LCD panel TFT example

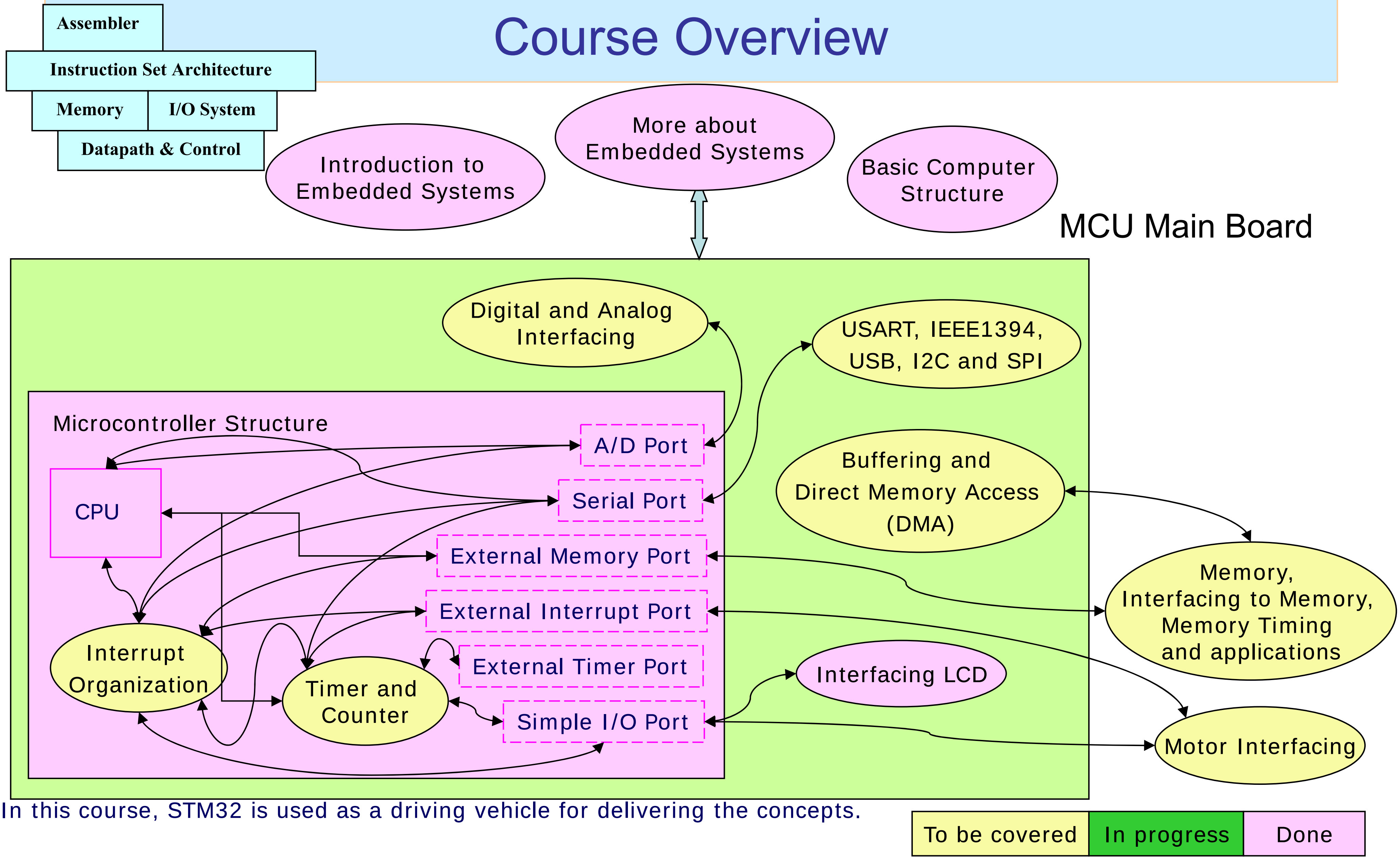
Can ground unused LCD signals (usually the lower weighted bits) for interfaces with less signals than the LCD



Reflection (Self-evaluation)

- Do you ….
  - Describe several types of LCD ?
  - Construct the hardware interfacing of the character type LCD module with a ARM microprocessor ?
  - Write the sub-routines of displaying cursor and characters onto the graphic type LCD ?
  - List the steps for displaying graphics onto a graphic type LCD ?
  - Write a software routine in mapping the color space from 24-bit RGB888 to 16-bit RGB565 ?

Course Overview



Assembler

Instruction Set Architecture

Memory

I/O System

Datapath & Control

Introduction to Embedded Systems

More about Embedded Systems

Basic Computer Structure

MCU Main Board

Digital and Analog Interfacing

USART, IEEE1394, USB, I2C and SPI

Buffering and Direct Memory Access (DMA)

Memory, Interfacing to Memory, Memory Timing and applications

Interfacing LCD

Motor Interfacing

Microcontroller Structure

CPU

A/D Port

Serial Port

External Memory Port

External Interrupt Port

External Timer Port

Simple I/O Port

Interrupt Organization

Timer and Counter

To be covered	In progress	Done

In this course, STM32 is used as a driving vehicle for delivering the concepts.