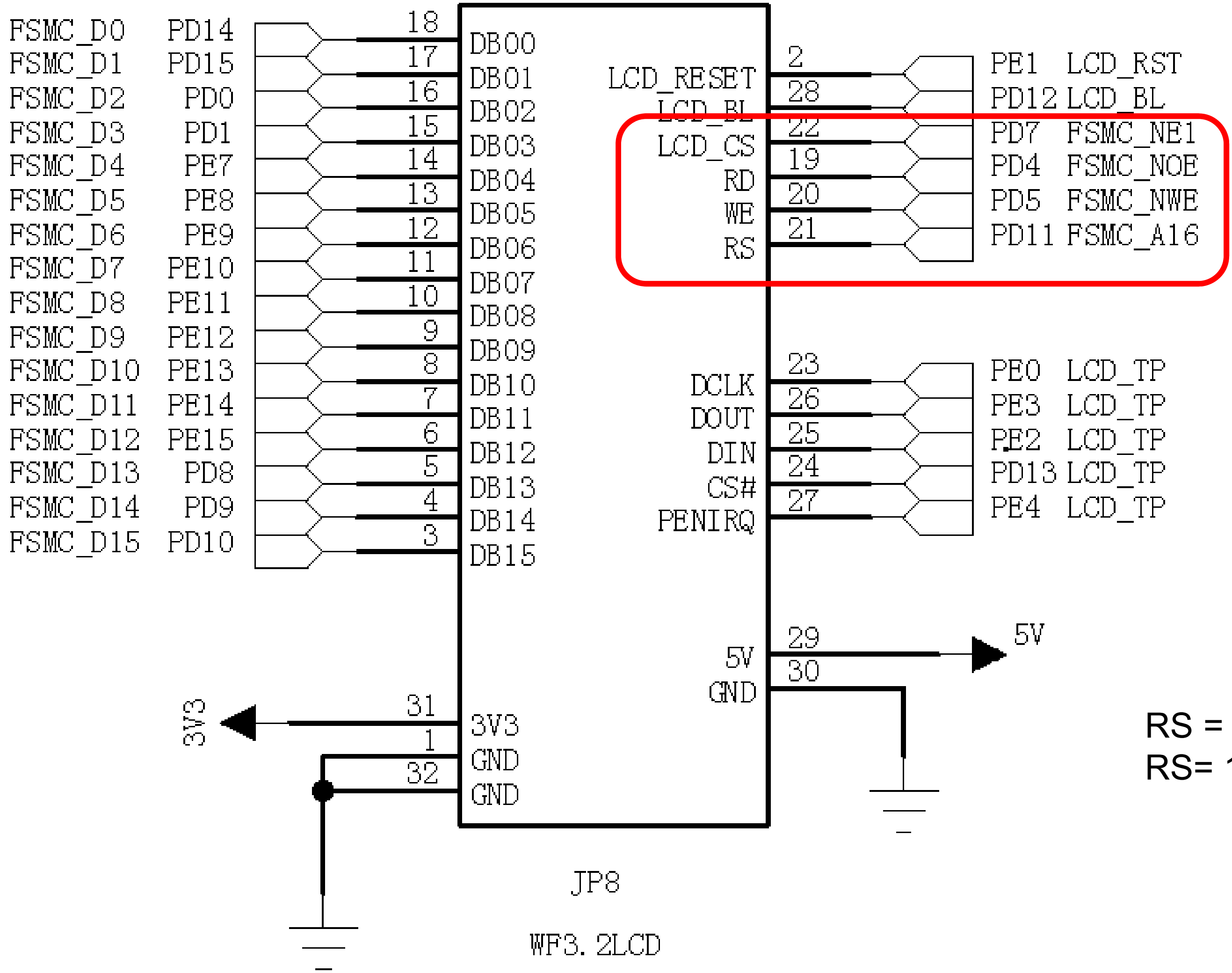
ELEC 3300 – Tutorial for LAB3

Department of Electronic and Computer Engineering
HKUST
by WU Chi Hang

LCD (Liquid Crystal Display)

- In MINI-V3 Development Board, it has a build in Color Graphic LCD module which is 240x320 dots. The driver of this LCD is ILI9341. Please check the datasheet of the LCD from the course webpage.
- Traditionally, we need to control the LCD by connect all the lines to the I/O port to do the control. However, in the MINI-V3 Development Board, it considered the LCD as a memory and using the FSMC function to control the LCD. Refer to the connection next page.

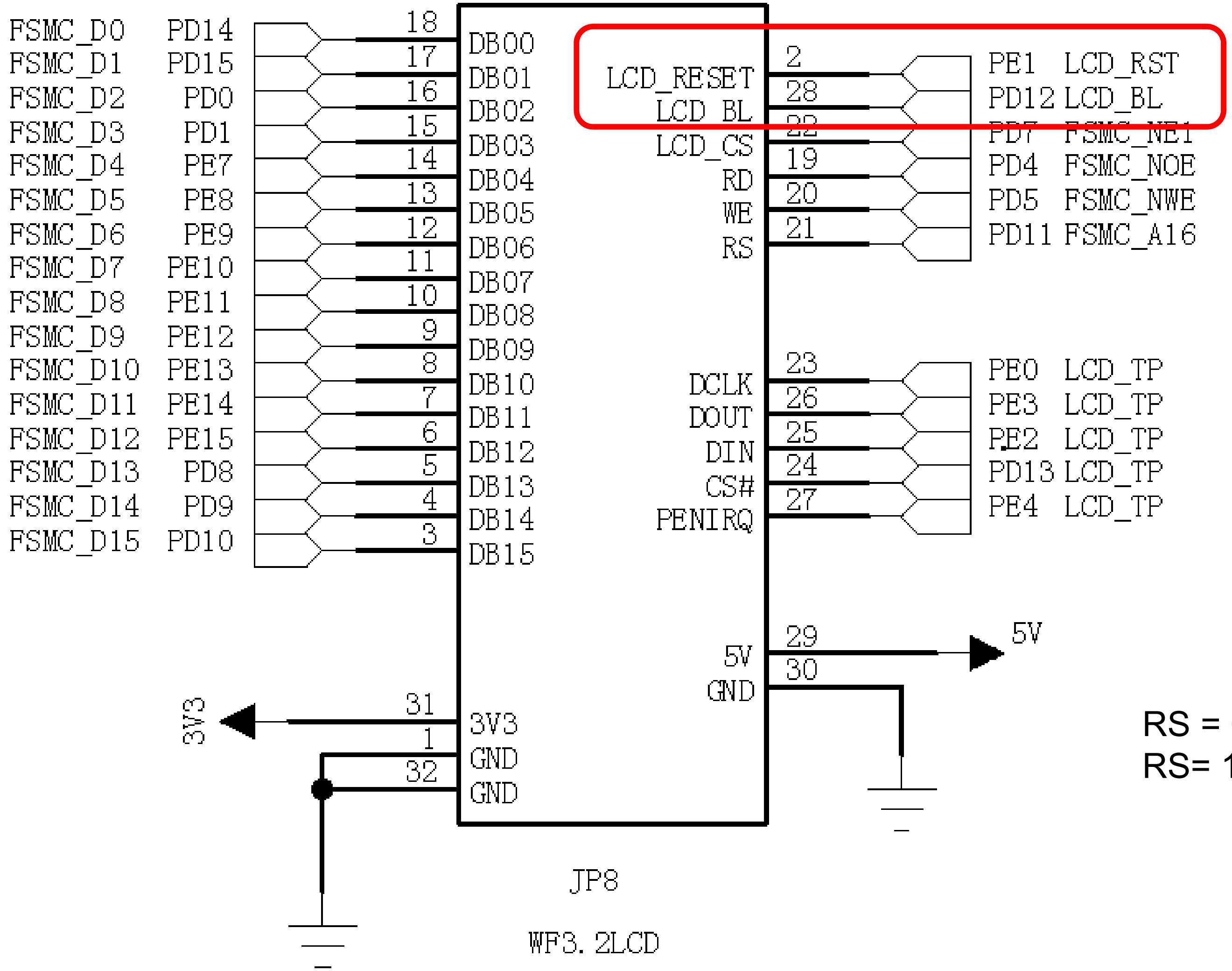
LCD CONNECTOR



LCD_CS → FSMC_NE1
(Neg. Enable 1)
RD → FSMC_NOE
(Neg. Output Enable)
WE → FSMC_NWE
(Neg. Write Enable)

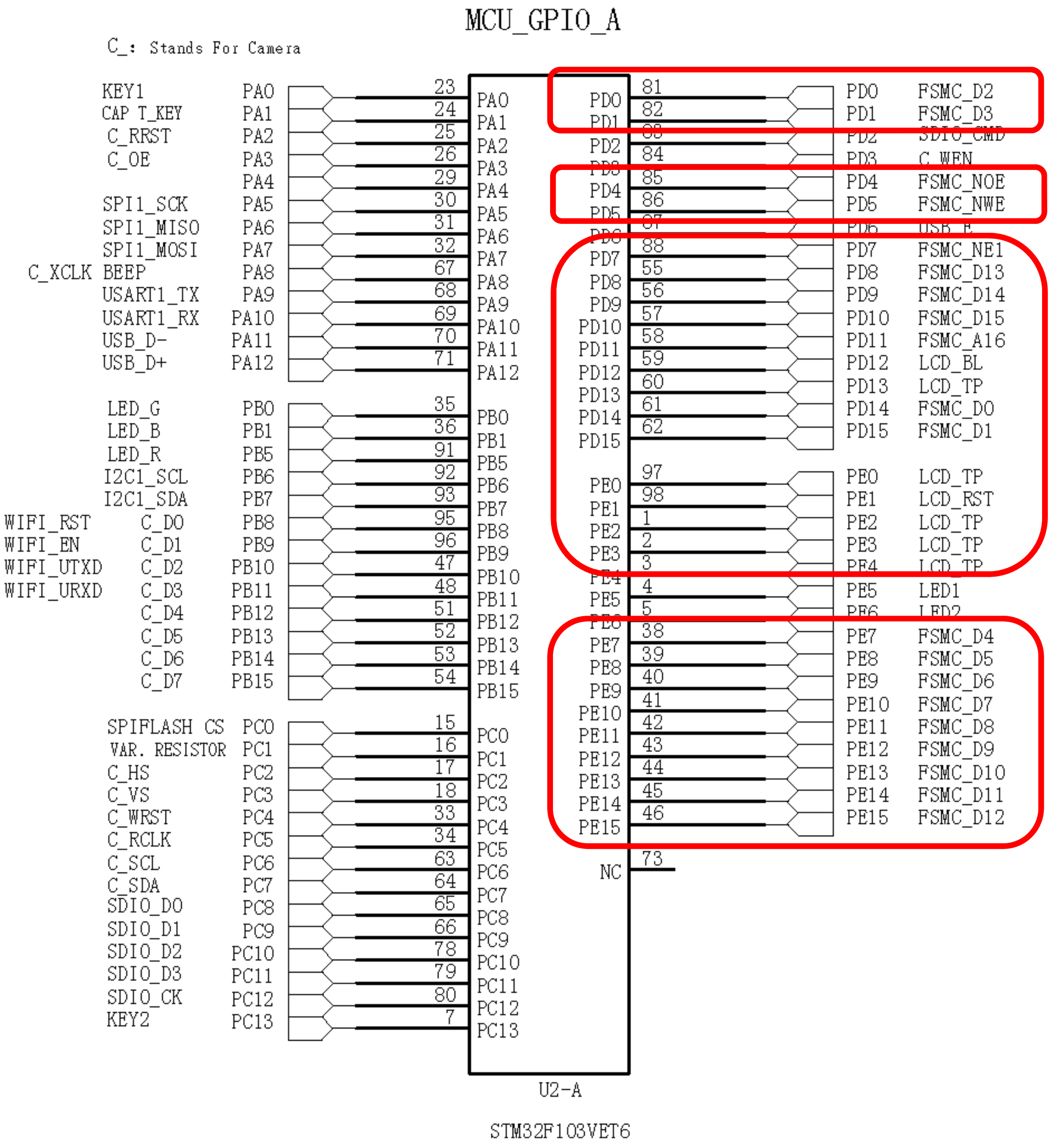
RS = 0 → Command
RS= 1 → Data

LCD CONNECTOR



LCD_RST
LCD__BL
These 2 pins are not in FSMC, they are GPIO

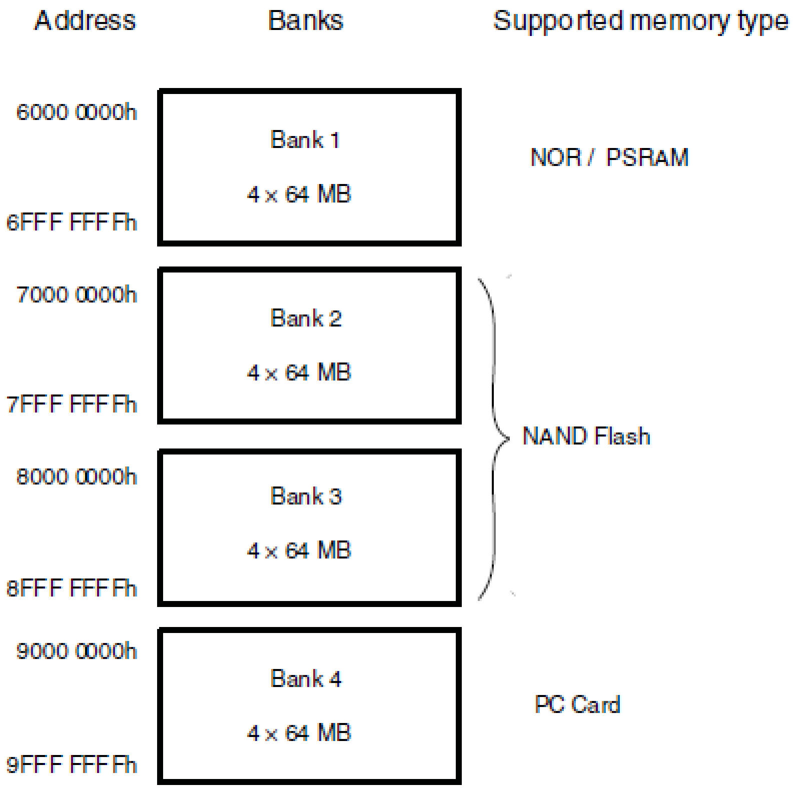
RS = 0 → Command
RS= 1 → Data



These pins are used for LCD

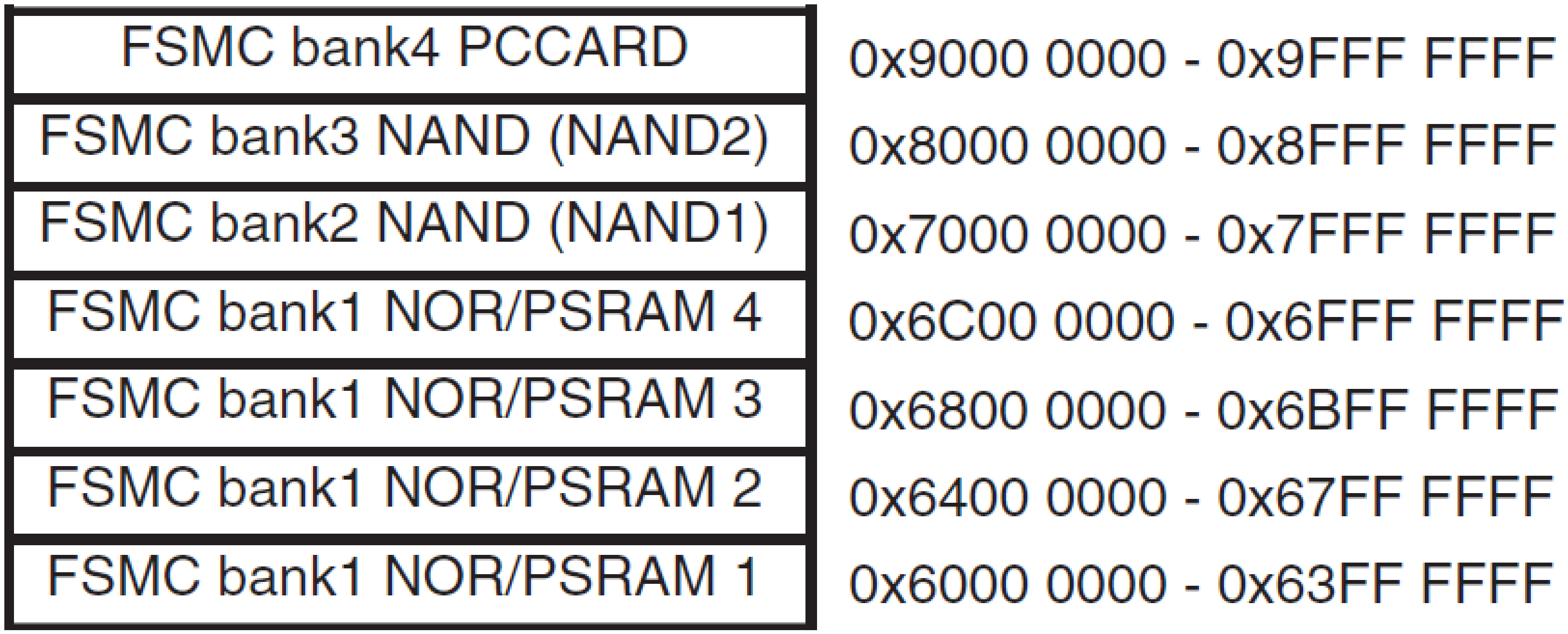
FSMC (Flexible Static Memory Controller)

- Basically, the FSMC block of STM32 is able to interface with synchronous and asynchronous memories or PC cards.
- In STM32, the FSMC is divided into 4 fixed-size banks of 256Mbytes each.



FSMC (Flexible Static Memory Controller)

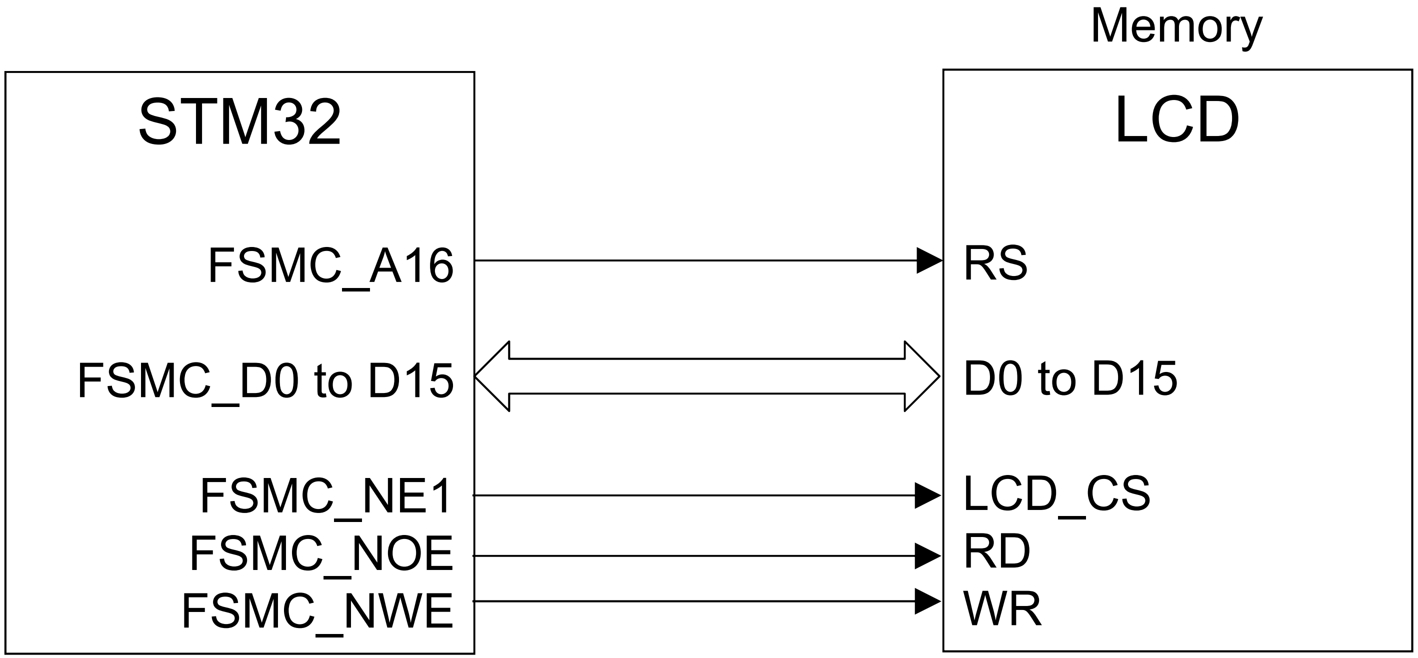
- Specifically, the address range is as follows



FSMC bank4 PCCARD	0x9000 0000 - 0x9FFF FFFF
FSMC bank3 NAND (NAND2)	0x8000 0000 - 0x8FFF FFFF
FSMC bank2 NAND (NAND1)	0x7000 0000 - 0x7FFF FFFF
FSMC bank1 NOR/PSRAM 4	0x6C00 0000 - 0x6FFF FFFF
FSMC bank1 NOR/PSRAM 3	0x6800 0000 - 0x6BFF FFFF
FSMC bank1 NOR/PSRAM 2	0x6400 0000 - 0x67FF FFFF
FSMC bank1 NOR/PSRAM 1	0x6000 0000 - 0x63FF FFFF

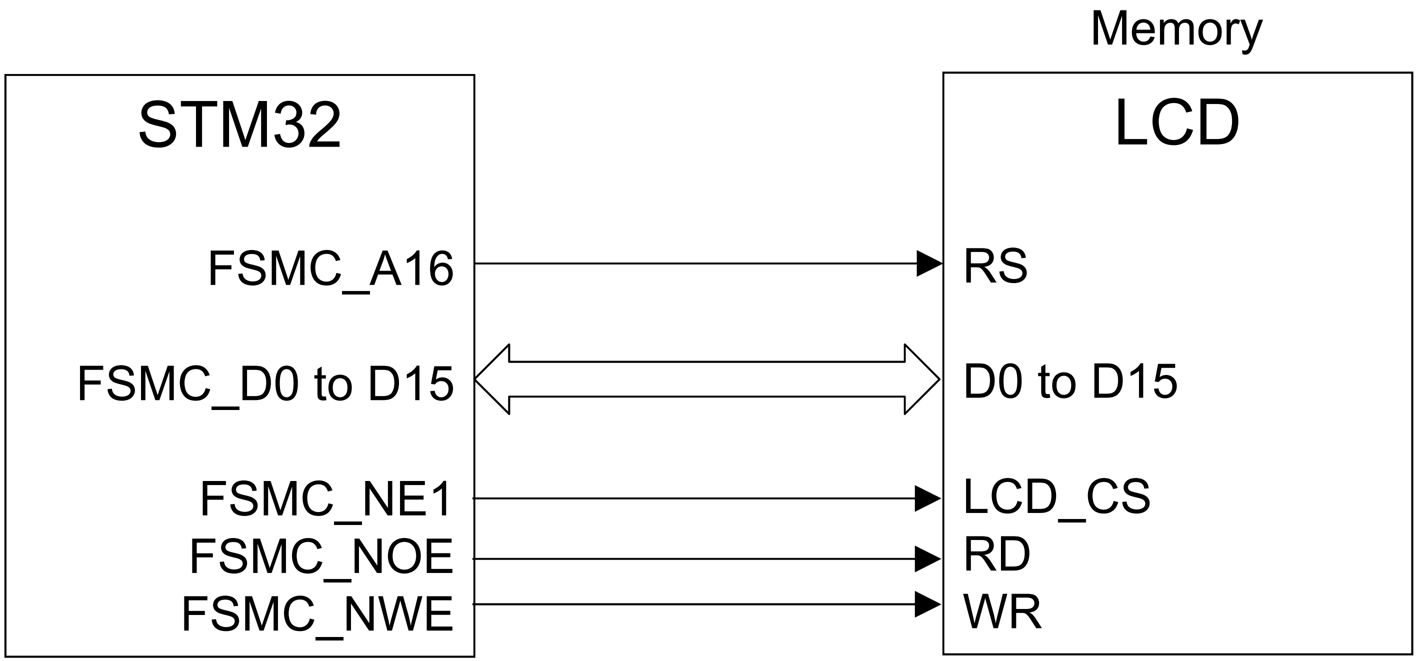
LCD FSMC Communication

- As said, the STM32 will treat the LCD as a memory, so the few pins are important. Note the arrows indicate the data transfer.
- In this LAB, the LCD is mapped into Bank1 NOR/PSRAM1 (as NE1 is connected to the Chip Select of the LCD)
- Refer to last page, what should be the addresses it mapped to ?



LCD FSMC Communication

- Do you think STM32 perform a Read or Write to LCD for most of the time ?
- What information should STM32 give out in order to achieve the function ?



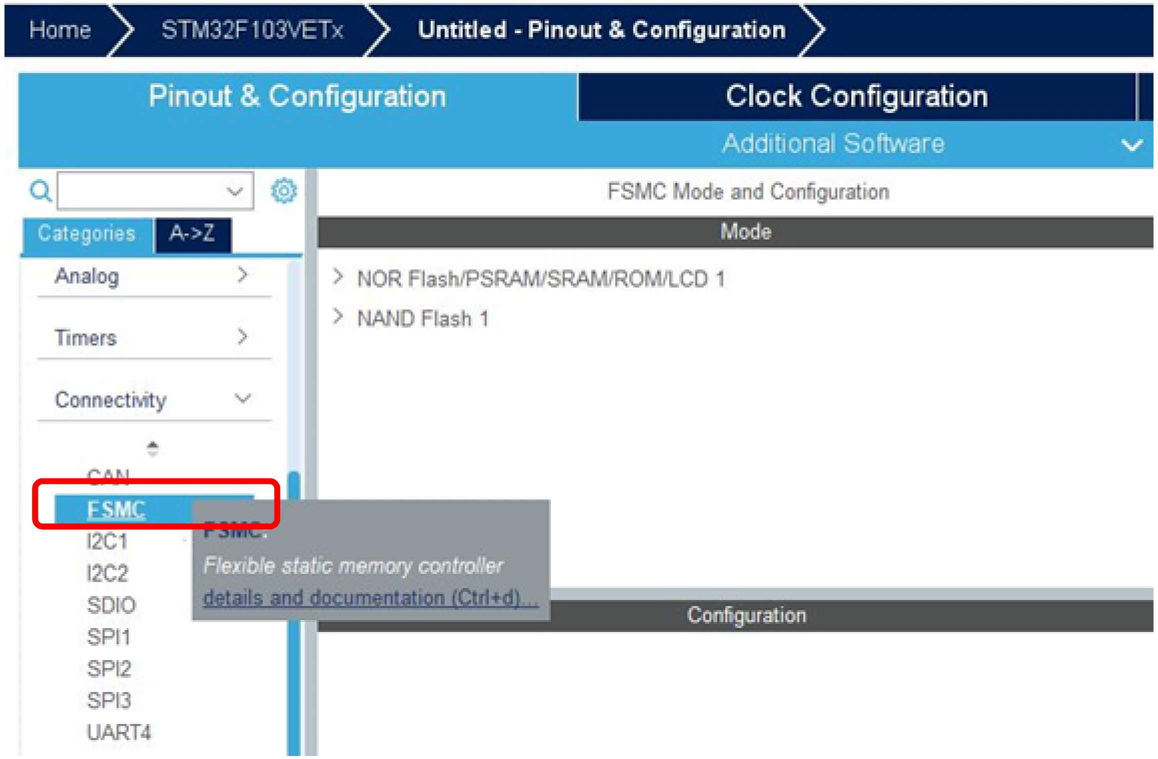
FSMC (Flexible Static Memory Controller)

- In CubeIDE, you can initialize the FSMC to be an LCD Interface

- Below I listed how you can setup and FSMC Interface in CubeIDE.
- Please note that you are required to follow the CubeIDE tutorial to setup the clock and also the debugging interface

CubeIDE Setting

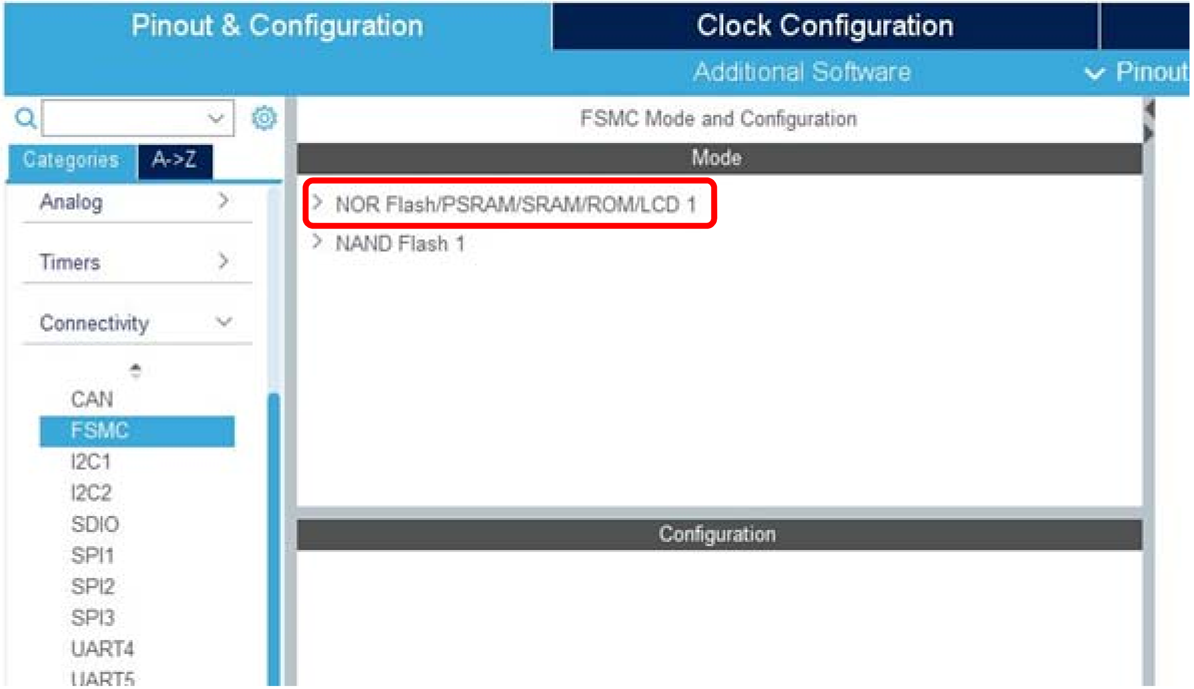
Choose FSMC



FSMC:
Flexible static memory controller
details and documentation (Ctrl+d)...

CubeIDE Setting

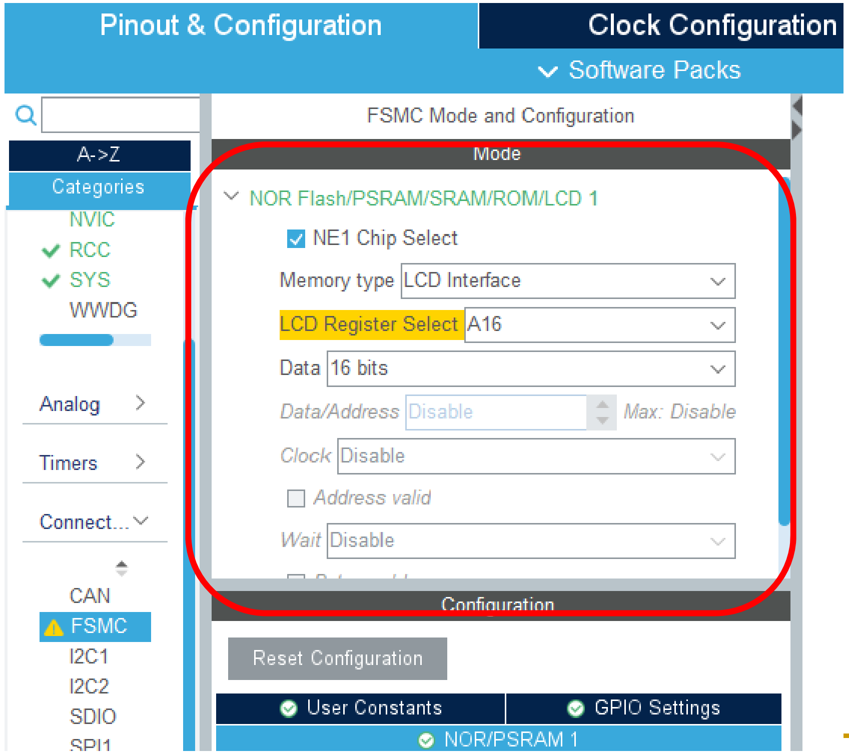
Choose NOR Flash/PSRAM/SRAM/ROM/LCD 1



CubeIDE Setting

Setup the LCD Interface.

** IMPORTANT ** Please refer to page 8, understand why these settings applies. Ignore the Yellow Warning



FSMC Mode and Configuration

Mode

NOR Flash/PSRAM/SRAM/ROM/LCD 1

☑ NE1 Chip Select

Memory type LCD Interface

LCD Register Select A16

Data 16 bits

Data/Address Disable Max: Disable

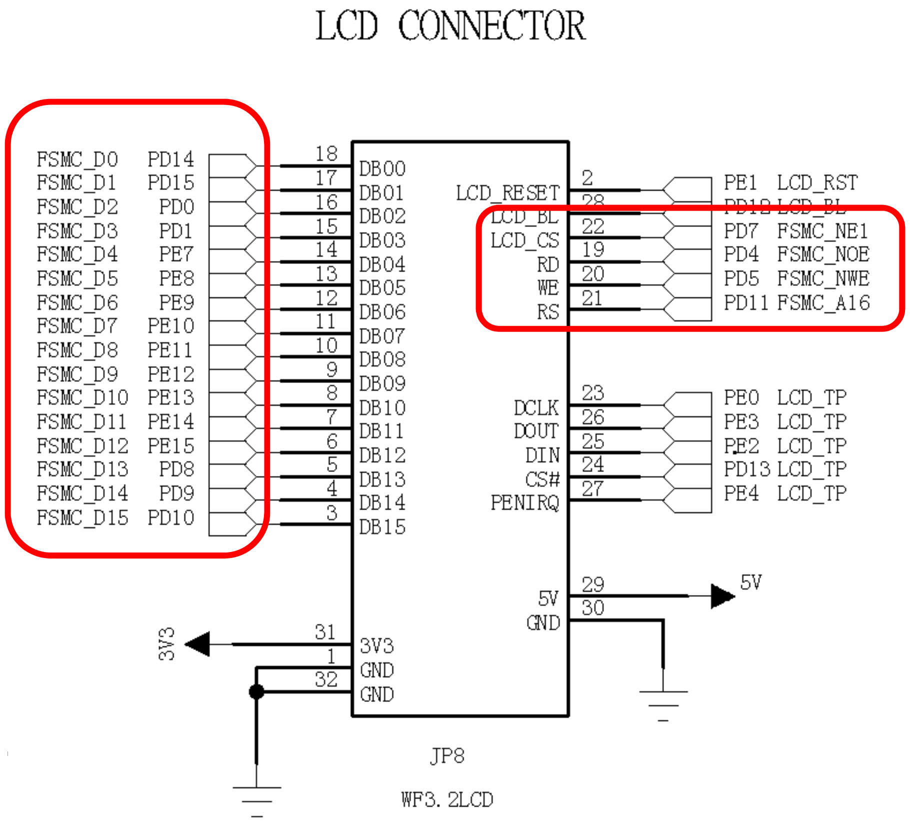
Clock Disable

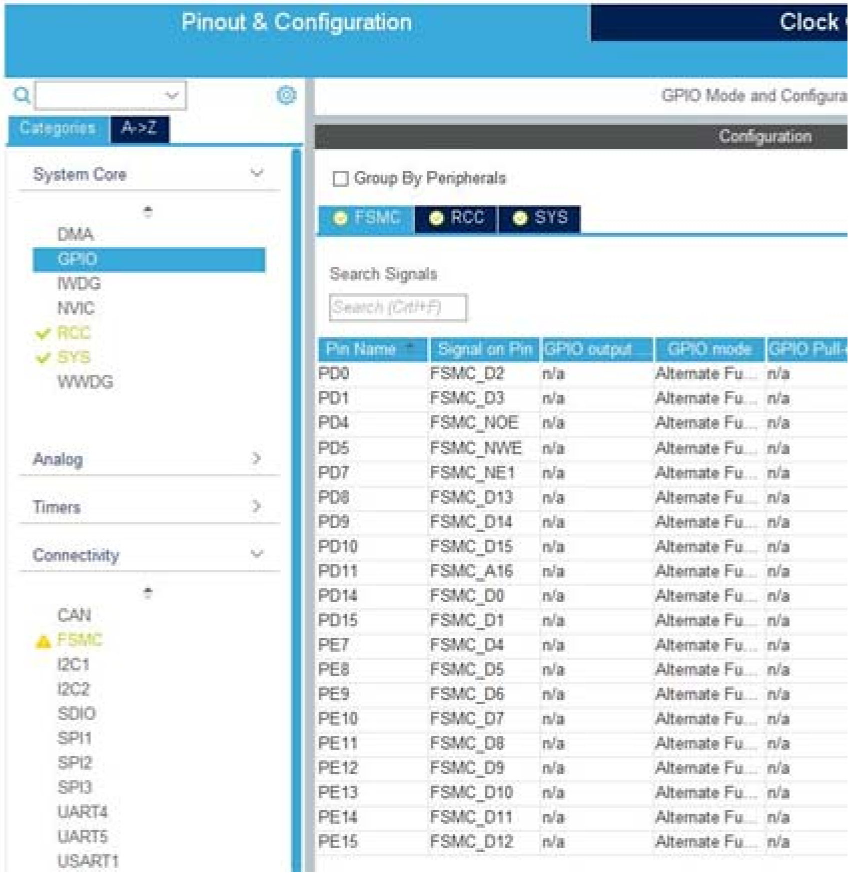
□ Address valid

Wait Disable

CubeIDE Setting

After above procedure, double check all the pins labelled FSMC appear in the



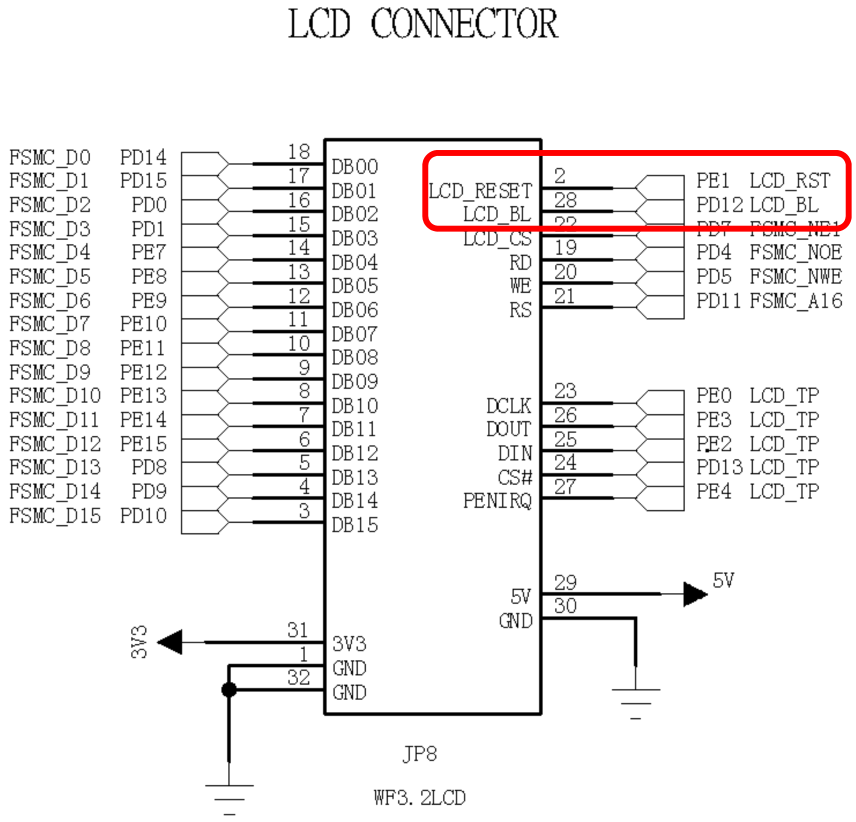


Pin Name	Signal on Pin	GPIO output ...	GPIO mode	GPIO Pull-
PD0	FSMC_D2	n/a	Alternate Fu...	n/a
PD1	FSMC_D3	n/a	Alternate Fu...	n/a
PD4	FSMC_NOE	n/a	Alternate Fu...	n/a
PD5	FSMC_NWE	n/a	Alternate Fu...	n/a
PD7	FSMC_NE1	n/a	Alternate Fu...	n/a
PD8	FSMC_D13	n/a	Alternate Fu...	n/a
PD9	FSMC_D14	n/a	Alternate Fu...	n/a
PD10	FSMC_D15	n/a	Alternate Fu...	n/a
PD11	FSMC_A16	n/a	Alternate Fu...	n/a
PD14	FSMC_D0	n/a	Alternate Fu...	n/a
PD15	FSMC_D1	n/a	Alternate Fu...	n/a
PE7	FSMC_D4	n/a	Alternate Fu...	n/a
PE8	FSMC_D5	n/a	Alternate Fu...	n/a
PE9	FSMC_D6	n/a	Alternate Fu...	n/a
PE10	FSMC_D7	n/a	Alternate Fu...	n/a
PE11	FSMC_D8	n/a	Alternate Fu...	n/a
PE12	FSMC_D9	n/a	Alternate Fu...	n/a
PE13	FSMC_D10	n/a	Alternate Fu...	n/a
PE14	FSMC_D11	n/a	Alternate Fu...	n/a
PE15	FSMC_D12	n/a	Alternate Fu...	n/a

CubeIDE Setting

Please note that the LCD_RST (LCD Reset) and LCD_BL (LCD Backlight) is not in FSMC function, we need to set it as GPIO.

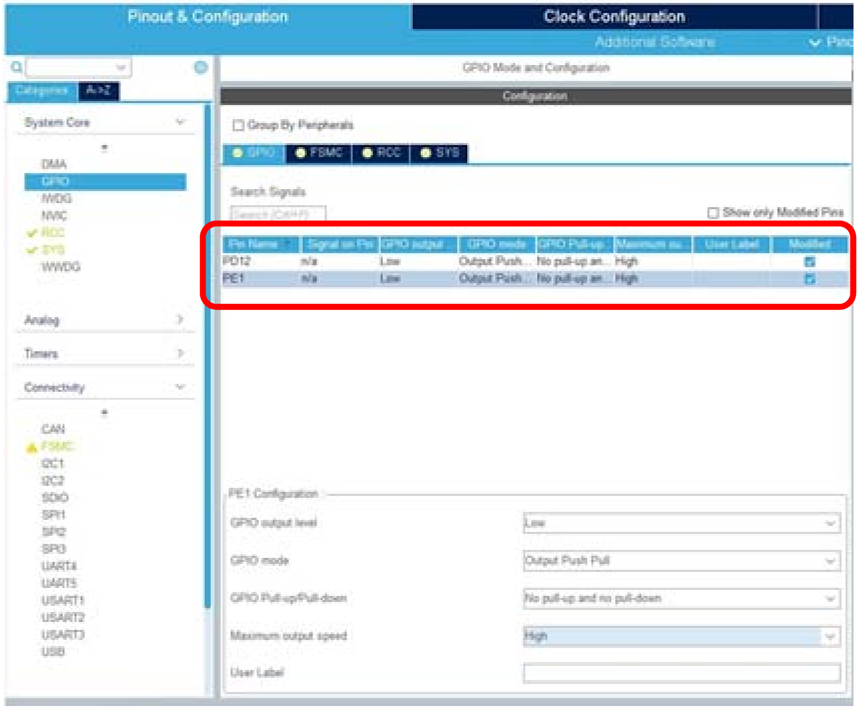
You can check from the previous page.



LCD_RST
LCD_BL
These 2 pins are not in FSMC, they are GPIO

CubeIDE Setting

Use your knowledge from LAB2, set the 2 pins to GPIO, set the output speed to HIGH. Then you can Generate the Code



Pin Name	Signal on Pin	GPIO output ...	GPIO mode	GPIO Pull-up...	Maximum ou...	User Label	Modified
PD12	n/a	Low	Output Push...	No pull-up an...	High		☑
PE1	n/a	Low	Output Push...	No pull-up an...	High		☑

PE1 Configuration

GPIO output level	Low
GPIO mode	Output Push Pull
GPIO Pull-up/Pull-down	No pull-up and no pull-down
Maximum output speed	High
User Label	

Adding LCD library

On Canvas, there is a lcd.zip, which is a folder contains the further initialization of the LCD. You need to unzip it and add them into your Project.

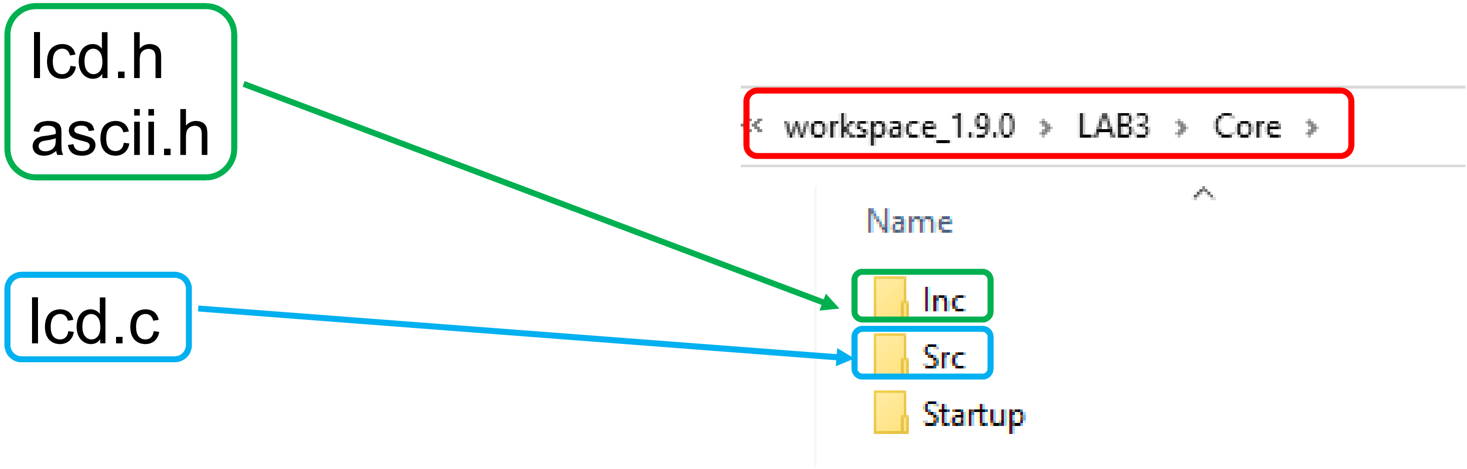
Unzip the lcd.zip, it will create a folder lcd with 3 files inside

- lcd.c
- lcd.h
- ascii.h

Adding LCD library

Go to your workspace folder, under your LAB3 Project > Core

Put the corresponding files in the folders

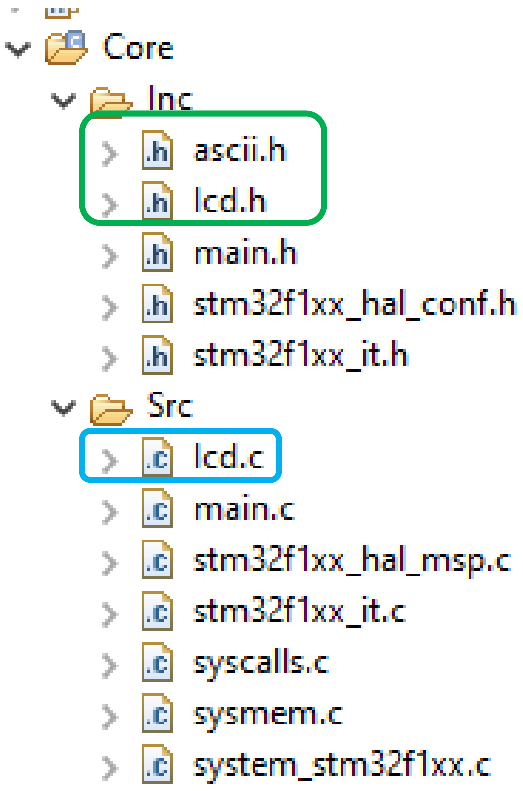


lcd.h
ascii.h

lcd.c

CubeIDE Setting

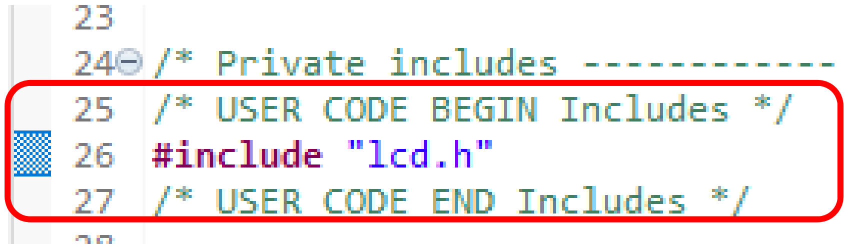
Try to Build, you should be able to see the files in under Core



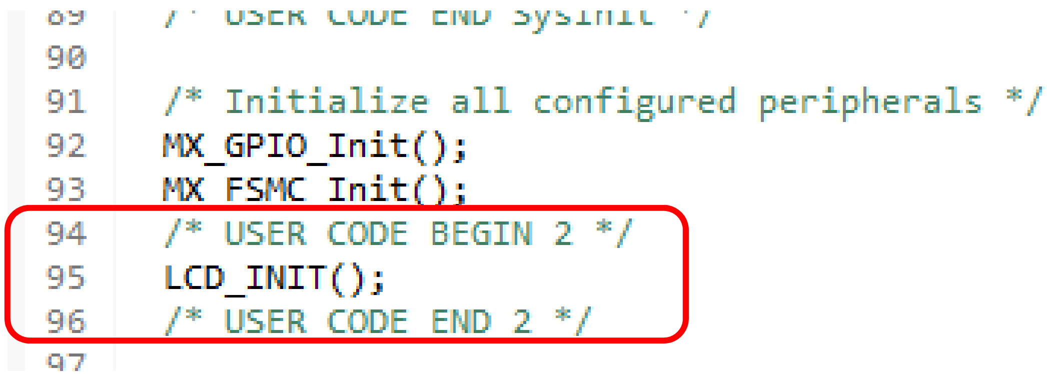
Core
Inc
ascii.h
lcd.h
main.h
stm32f1xx_hal_conf.h
stm32f1xx_it.h
Src
lcd.c
main.c
stm32f1xx_hal_msp.c
stm32f1xx_it.c
syscalls.c
sysmem.c
system_stm32f1xx.c

Edit main.c

In main.c add the following, remember they are between BEGIN and END



```
/* Private includes ------------
/* USER CODE BEGIN Includes */
#include "lcd.h"
/* USER CODE END Includes */
```



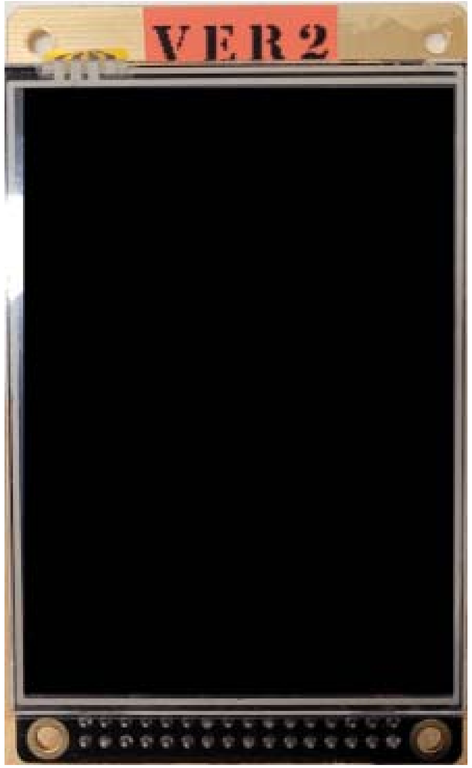
```
  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_FSMC_Init();
  /* USER CODE BEGIN 2 */
  LCD_INIT();
  /* USER CODE END 2 */
```

2 Versions of LCD

There are 2 Versions of LCD. If your LCD has LABEL VER2 on top, it means it is Version 2, otherwise, it is Version 1.

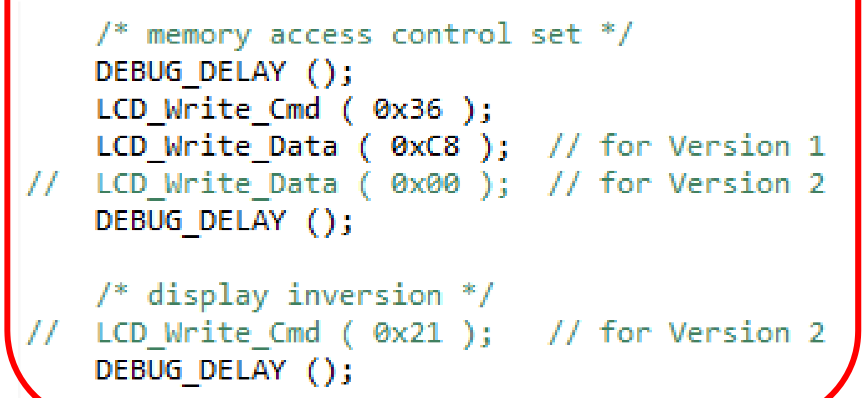
The right side shows the Version 2 LCD.



Edit lcd.c

Open lcd.c, based on your LCD version, use the correct line of code.

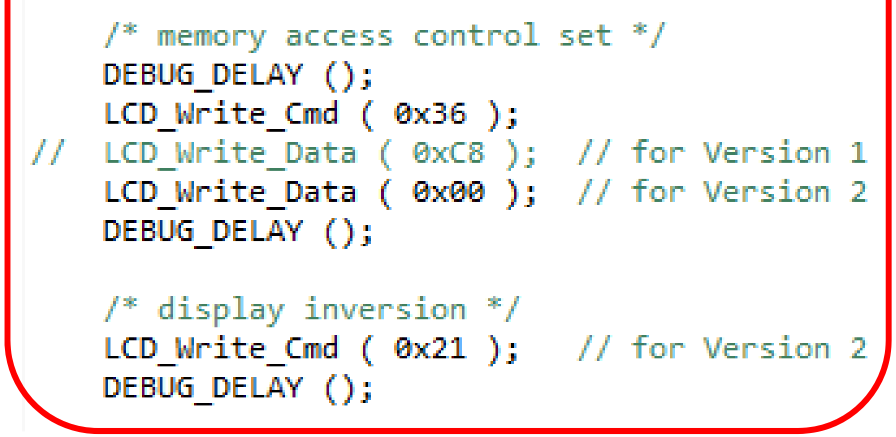
Version 1



```
    /* memory access control set */
    DEBUG_DELAY ();
    LCD_Write_Cmd ( 0x36 );
    LCD_Write_Data ( 0xC8 );  // for Version 1
//  LCD_Write_Data ( 0x00 );  // for Version 2
    DEBUG_DELAY ();

    /* display inversion */
//  LCD_Write_Cmd ( 0x21 );   // for Version 2
    DEBUG_DELAY ();
```

Version 2



```
    /* memory access control set */
    DEBUG_DELAY ();
    LCD_Write_Cmd ( 0x36 );
//  LCD_Write_Data ( 0xC8 );  // for Version 1
    LCD_Write_Data ( 0x00 );  // for Version 2
    DEBUG_DELAY ();

    /* display inversion */
    LCD_Write_Cmd ( 0x21 );   // for Version 2
    DEBUG_DELAY ();
```

Actually, these line sets some setting for LCD, if you want to learn, you may try to swap the settings and see what happens. 😬 However, remember to use the correct version after that.

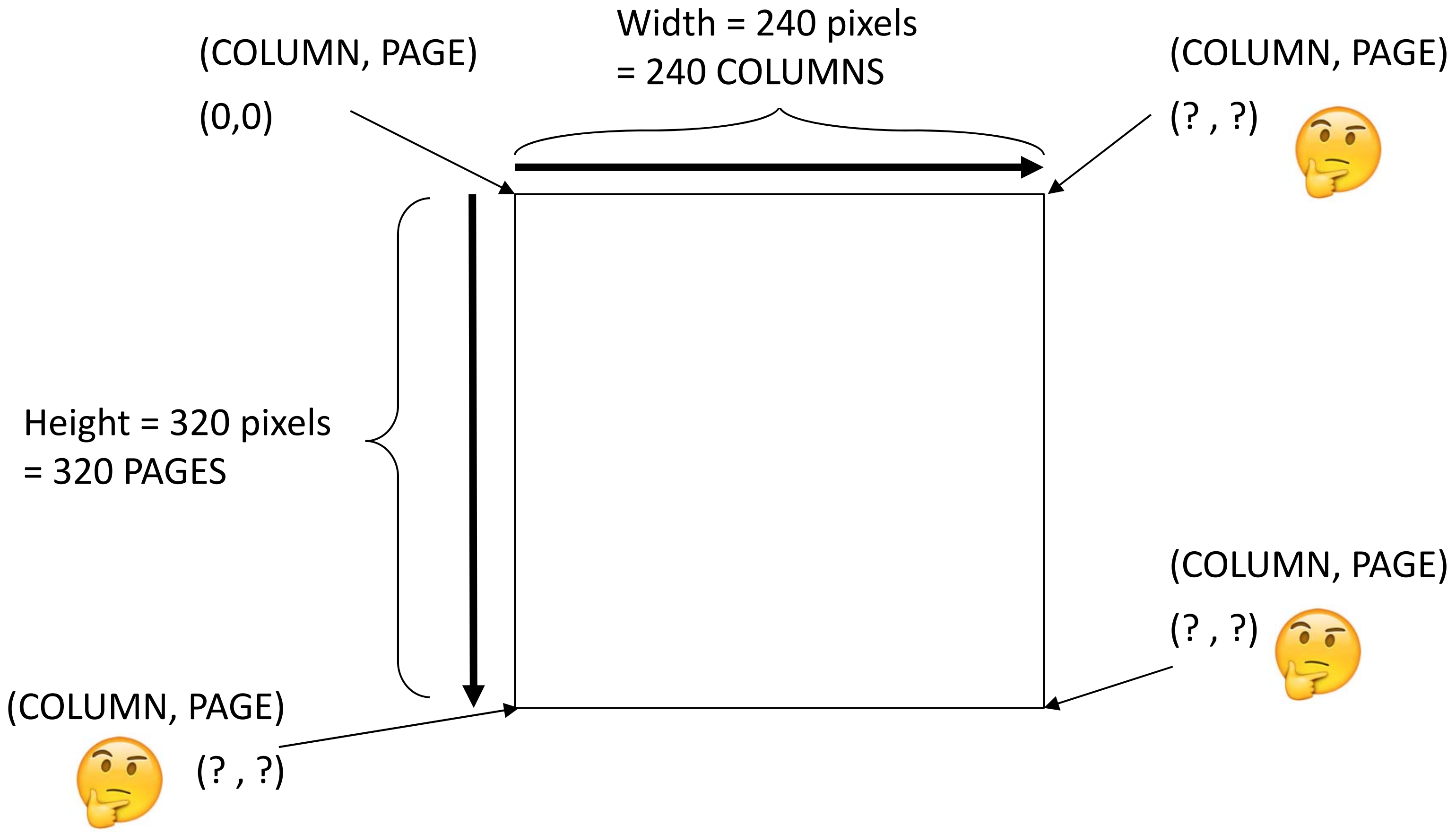
Test your code

- Now you should be able to compile your code.
- You can try to run your code and then see if your LCD shows a plain White screen.

- **NOTE : PLEASE NOTE THAT ALL THE EXTRA SETTINGS ADDED AFTER CODE GENERATION MAY BE DELETED AFTER RE-GENERATION OF CODE BY CubeIDE.**
- **PLEASE DOUBLE CHECK IF ALL THE SETTINGS ARE STILL THERE AFTER CODE GENERATION**

LCD Layout

- The LCD is 240x320 as follows :



Width = 240 pixels
= 240 COLUMNS

Height = 320 pixels
= 320 PAGES

(COLUMN, PAGE)
(0,0)

(COLUMN, PAGE)
(? , ?) 🤔

(COLUMN, PAGE)
(? , ?) 🤔

(COLUMN, PAGE)
🤔 (? , ?)

LCD functions available to you

```
void LCD_DrawChar(uint16_t usC, uint16_t usP, const char cChar);
void LCD_DrawString(uint16_t usC, uint16_t usP, const char * pStr);
```

- In these functions, usC and usP are the corresponding COLUMN pixel number and Page pixel number.
- Each character : Width is 8 pixels, Height is 16 pixels
- If I write

  `LCD_DrawChar(0, 0, ’F’);` in main.c
- It will display a character F with the Upper left corner at (0,0)
- Same idea for LCD_DrawString
- Your first task is to displaying your name by using the LCD_DrawChar or LCD_DrawString functions.

Task 1: Write your Name on LCD

```
void LCD_DrawChar(uint16_t usC, uint16_t usP, const char cChar);
void LCD_DrawString(uint16_t usC, uint16_t usP, const char * pStr);
```

- You need to display your English name shown on your Student ID by using the LCD_DrawChar or LCD_DrawString functions.
- You can choose any location and any color you like

LCD Cmd/Data

- With the FSMC Interface is setup and with the help of the library given, STM32 can send a 16-bit value to the LCD by two functions, depending on the meaning of the 16-bit value to the LCD

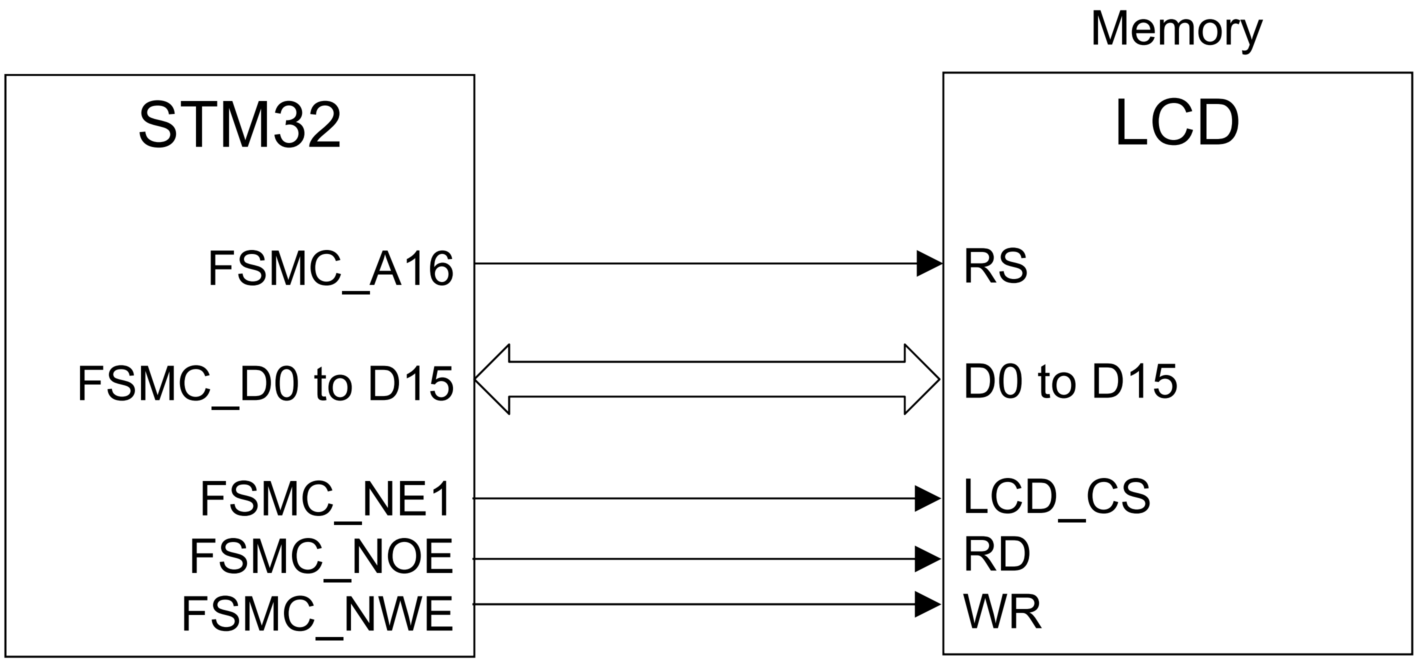
  LCD_Write_Cmd (uint16_t usCmd) ;

  LCD_Write_Data (uint16_t usData) ;

- What Cmd / Data means ?
  - A Cmd to LCD is a value to control the function of the LCD
  - A Data to LCD is an information (normally color) to be displayed.
- How do the LCD knows the 16-bit value it received is a Cmd or Data ?

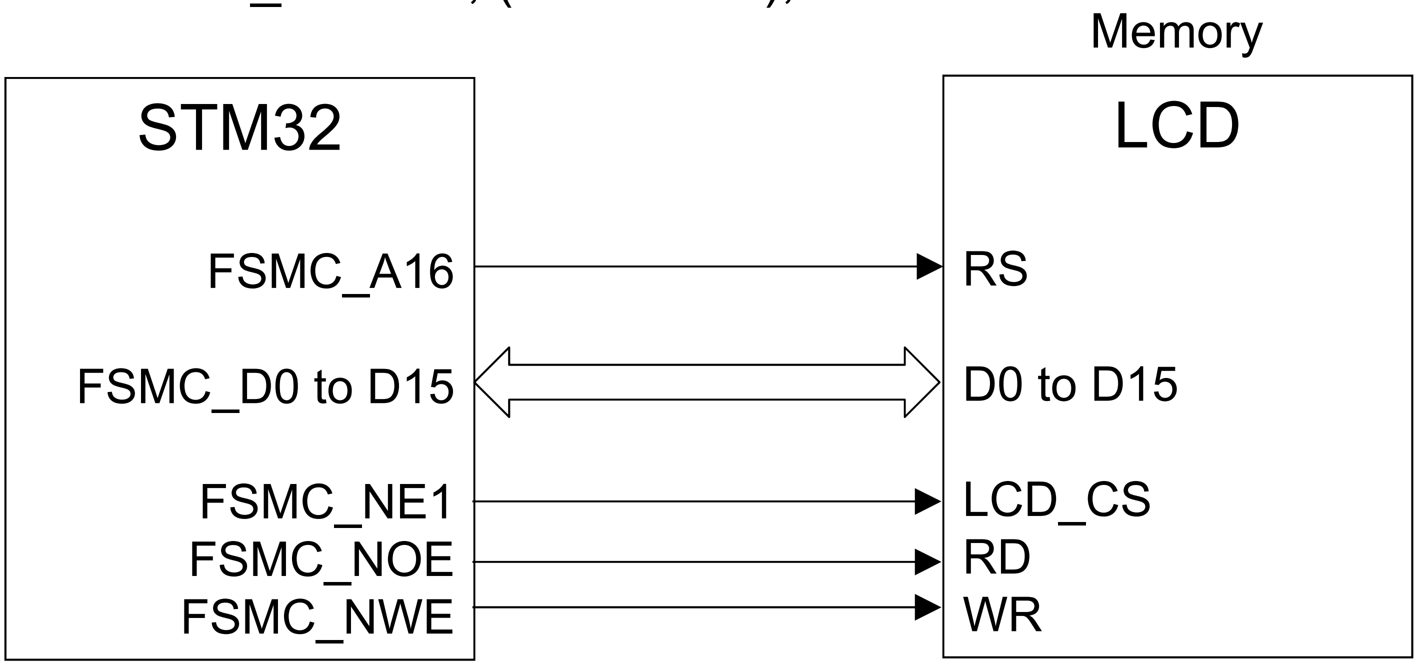
LCD Cmd/Data

- Cmd / Data is indicated by the status of RS
  - RS = 0, the 16-bit value means Cmd,
  - RS = 1, the 16-bit value means Data,



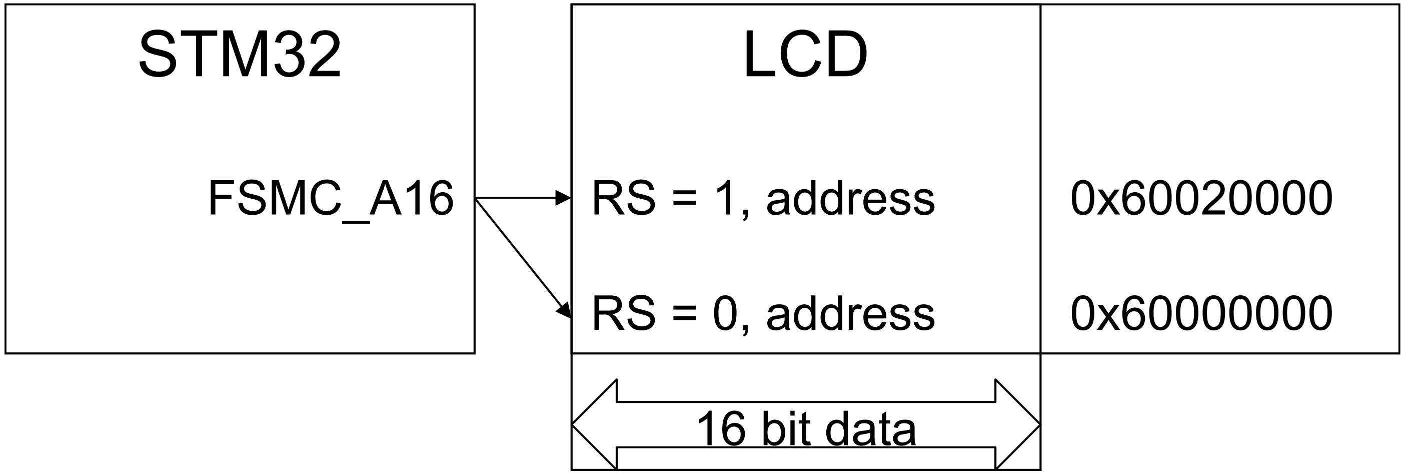
LCD Cmd/Data

- Bank 1 address starts at 0x60000000 at STM32, and 😬
- STM32 uses FSMC_A16 to turn RS to 1 or 0
  - When FSMC_A16 = 0, (i.e. RS = 0), what is the address at STM32?
  - When FSMC_A16 = 1, (i.e. RS = 1), what is the address at STM32?



LCD Cmd/Data

- However, as each data in LCD is 16-bit and it is a word address. So, corresponding byte address should be (i.e. x 2).
- Refer to lcd.c
  - LCD_Write_Cmd address is at 0x60000000 😬
  - LCD_Write_Data address is at 0x60020000



STM32

FSMC_A16

LCD

RS = 1, address

0x60020000

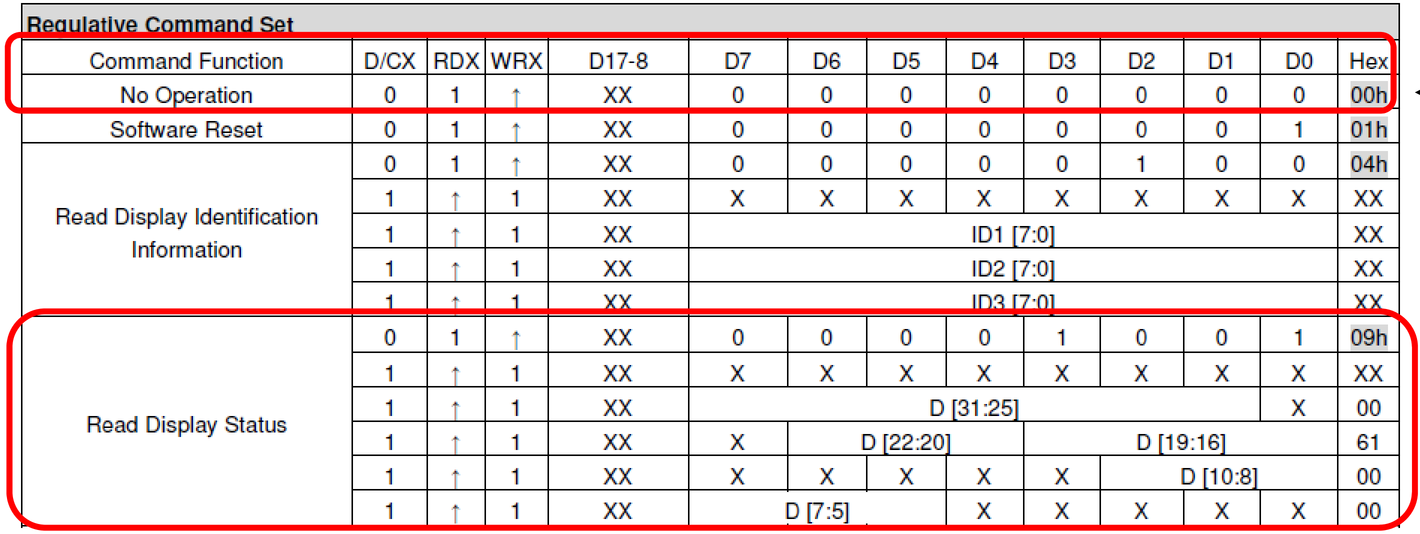
RS = 0, address

0x60000000

16 bit data

LCD Command List

- Refer to Section 8 of the LCD datasheet, it shows all the command that needed to control the function of the LCD
- Note that some commands with data to follow



Regulative Command Set

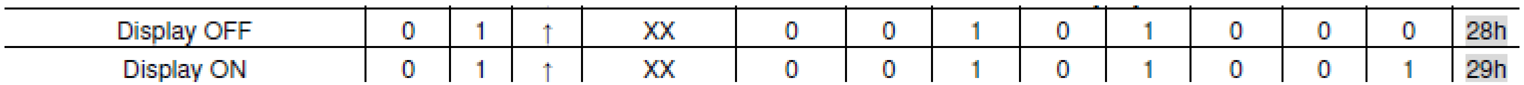
Command Function	D/CX	RDX	WRX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	Hex
No Operation	0	1	↑	XX	0	0	0	0	0	0	0	0	00h
Software Reset	0	1	↑	XX	0	0	0	0	0	0	0	1	01h
Read Display Identification Information	0	1	↑	XX	0	0	0	0	0	1	0	0	04h
	1	↑	1	XX	X	X	X	X	X	X	X	X	XX
	1	↑	1	XX	ID1 [7:0]								XX
	1	↑	1	XX	ID2 [7:0]								XX
	1	↑	1	XX	ID3 [7:0]								XX
Read Display Status	0	1	↑	XX	0	0	0	0	1	0	0	1	09h
	1	↑	1	XX	X	X	X	X	X	X	X	X	XX
	1	↑	1	XX	D [31:25]							X	00
	1	↑	1	XX	X	D [22:20]			D [19:16]				61
	1	↑	1	XX	X	X	X	X	X	D [10:8]			00
	1	↑	1	XX	D [7:5]			X	X	X	X	X	00

Command only.
No data to follow

Command follow
by 5 data

LCD Command Example

- Refer to the Command List



Display OFF	0	1	↑	XX	0	0	1	0	1	0	0	0	28h
Display ON	0	1	↑	XX	0	0	1	0	1	0	0	1	29h

- If want to turn the Display ON, I need to send a command 0x29 to the LCD, it will be achieved by

  LCD_Write_Cmd (0x29) ;

- If I want to make the LCD enter sleep mode, what should I do ?

Task 2: Implement the DrawDot

```
void LCD_DrawDot(uint16_t usC, uint16_t usP, uint16_t usColor);
```

- You need to implement a function that will draw a particular dot in the LCD.
- Where usC and usP are the corresponding Pixel’s Column and Page in the LCD.
- How do draw a dot ?

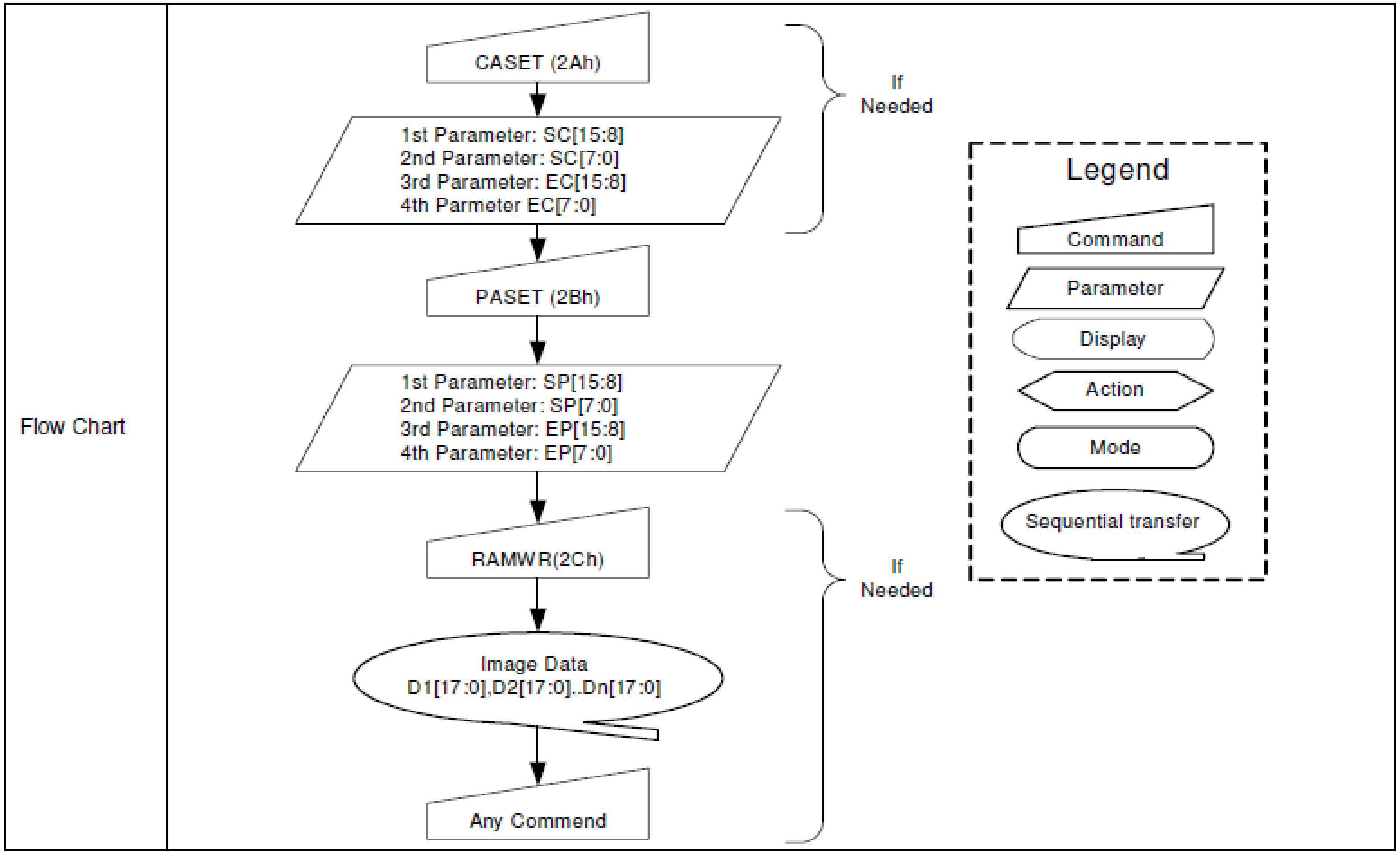
How to Display data to the LCD ?

- The flow of Displaying a data to the LCD is the following

1. Set the starting and the ending columns of the LCD
2. Set the starting and the ending pages of the LCD
3. Write the pixels of the LCD

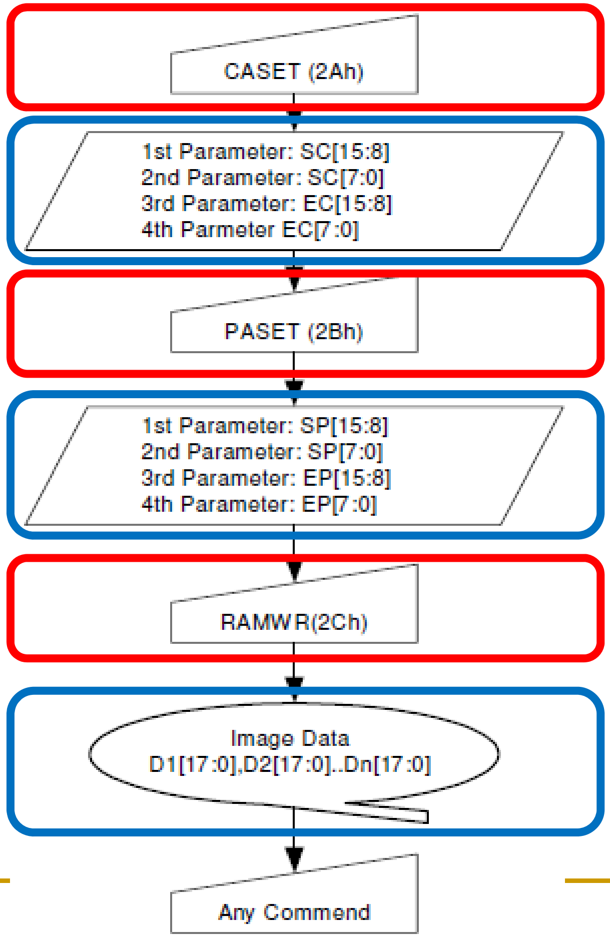
How to Display data to the LCD ?

- Refer to page 111 of the LCD datasheet.



How to Display data to the LCD ?

- Refer to page 111 of the LCD datasheet.



CASET (2Ah)

1st Parameter: SC[15:8]
2nd Parameter: SC[7:0]
3rd Parameter: EC[15:8]
4th Parmeter EC[7:0]

PASET (2Bh)

1st Parameter: SP[15:8]
2nd Parameter: SP[7:0]
3rd Parameter: EP[15:8]
4th Parameter: EP[7:0]

RAMWR(2Ch)

Image Data
D1[17:0],D2[17:0]..Dn[17:0]

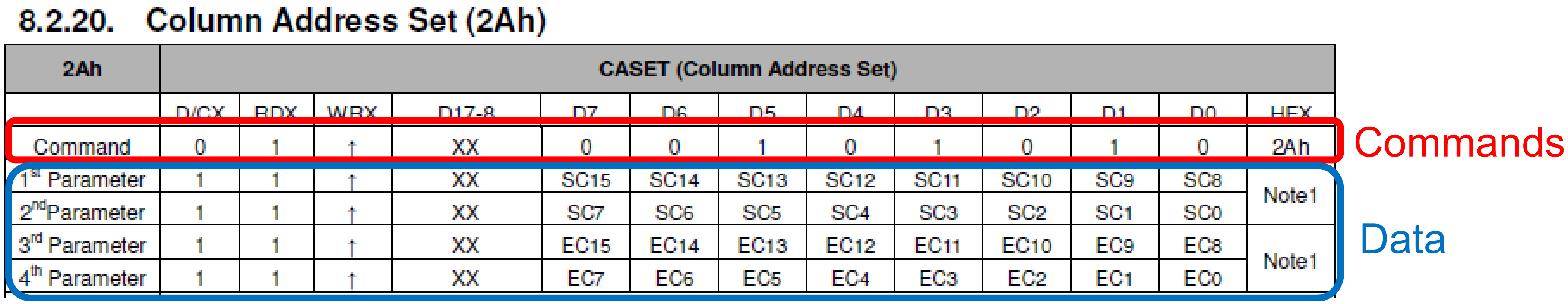
Any Commend

🟥 Red are Commands

🟦 Blue are Data

Set the Column Address (Cmd 0x2A)

- Refer to page 110 of the datasheet



8.2.20. Column Address Set (2Ah)

2Ah	CASET (Column Address Set)												
	D/CX	RDX	WRX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	HEX
Command	0	1	↑	XX	0	0	1	0	1	0	1	0	2Ah
1st Parameter	1	1	↑	XX	SC15	SC14	SC13	SC12	SC11	SC10	SC9	SC8	Note1
2ndParameter	1	1	↑	XX	SC7	SC6	SC5	SC4	SC3	SC2	SC1	SC0	
3rd Parameter	1	1	↑	XX	EC15	EC14	EC13	EC12	EC11	EC10	EC9	EC8	Note1
4th Parameter	1	1	↑	XX	EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0	

Commands

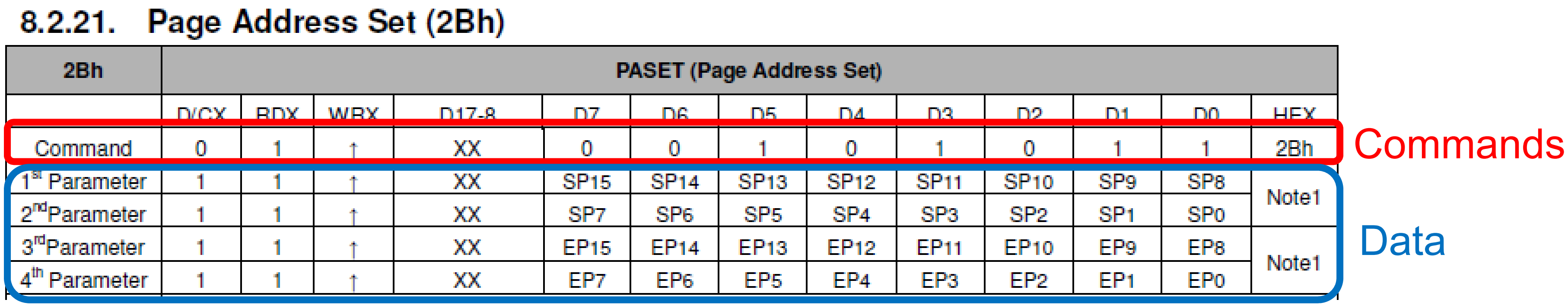
Data

- Say, if we want to set the start column to be 10 and end column to be 110 we need to

```
LCD_Write_Cmd (0x2A) ;                    // Command
LCD_Write_Data (              ) ;         // 1st Parameter
LCD_Write_Data (              ) ;         // 2nd Parameter
LCD_Write_Data (              ) ;         // 3rd Parameter
LCD_Write_Data (              ) ;         // 4th Parameter
```

Set the Page Address (Cmd 0x2B)

- Refer to page 112 of the datasheet



8.2.21. Page Address Set (2Bh)

2Bh	PASET (Page Address Set)												
	D/CX	RDX	WRX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	HEX
Command	0	1	↑	XX	0	0	1	0	1	0	1	1	2Bh
1st Parameter	1	1	↑	XX	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	Note1
2nd Parameter	1	1	↑	XX	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	
3rd Parameter	1	1	↑	XX	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	Note1
4th Parameter	1	1	↑	XX	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0	

Commands

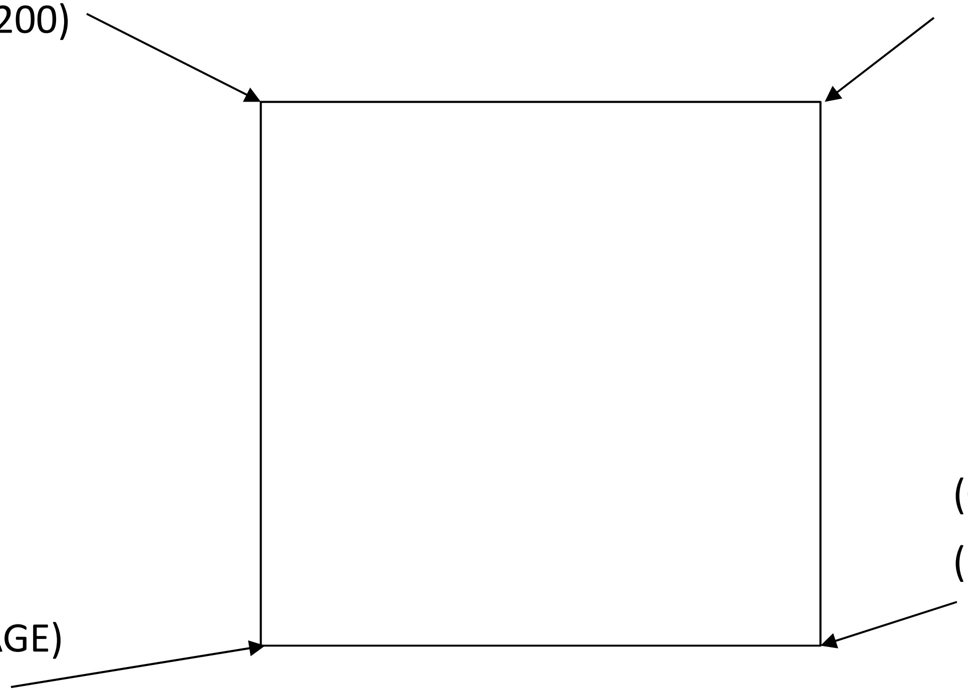
Data

- Say, if we want to set the start page to be 200 and end page to be 300 we need to

```
LCD_Write_Cmd (0x2B) ;                  // Command
LCD_Write_Data (            ) ;         // 1st Parameter
LCD_Write_Data (            ) ;         // 2nd Parameter
LCD_Write_Data (            ) ;         // 3rd Parameter
LCD_Write_Data (            ) ;         // 4th Parameter
```

Memory Write (Cmd 0x2C)

- From the above command, we created a memory window



(COLUMN, PAGE)
(10,200)

(COLUMN, PAGE)
(110 , 200)

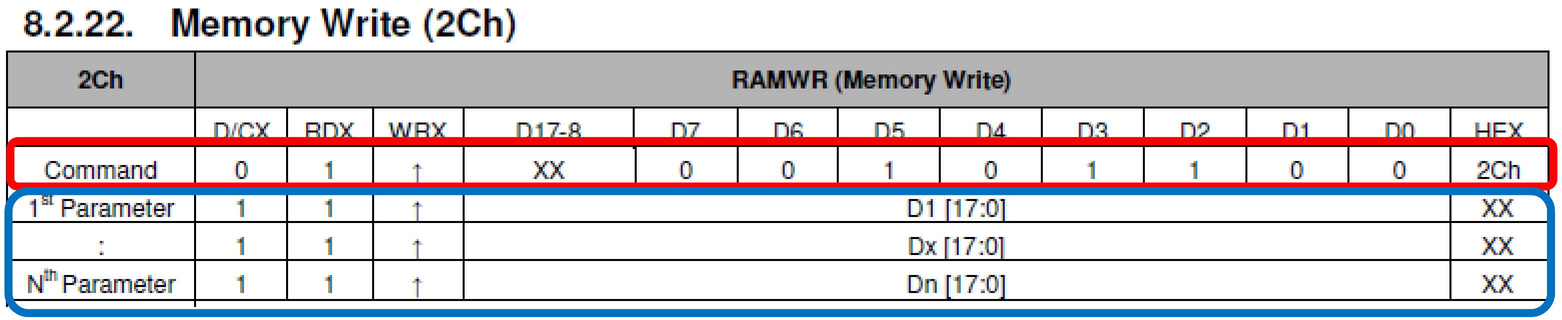
(COLUMN, PAGE)
(110 , 300)

(COLUMN, PAGE)
(10,300)

How many pixels in total in this window ? 🤔

Memory Write (Cmd 0x2C)

- After we create the window, we can write the colors by Memory Write Command. Refer to page 114 of the datasheet



8.2.22. **Memory Write (2Ch)**

2Ch	RAMWR (Memory Write)											
	D/CX	RDX	WRX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	HEX
Command	0	1	↑	XX	0	0	1	0	1	1	0	0	2Ch
1st Parameter	1	1	↑	D1 [17:0]									XX
:	1	1	↑	Dx [17:0]									XX
Nth Parameter	1	1	↑	Dn [17:0]									XX

Commands

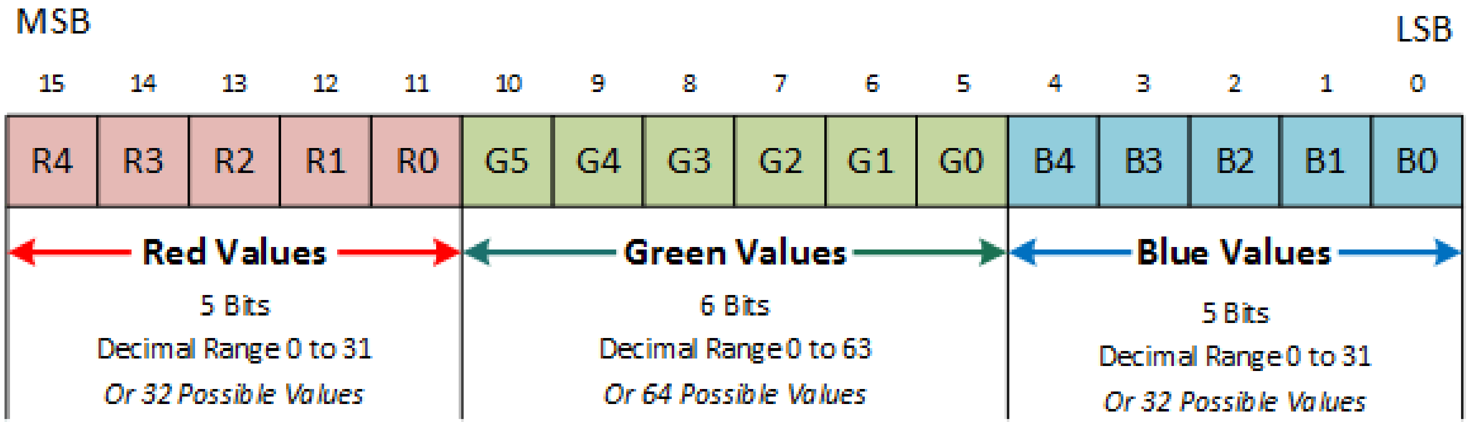
Data

- You can then send in sequence

```
LCD_Write_Cmd (0x2C) ;                  // Command
LCD_Write_Data (              ) ;       // 1st Parameter
LCD_Write_Data (              ) ;       // 2nd Parameter
..
LCD_Write_Data (              ) ;       // Nth Parameter
```

Color of Pixel

- The Color of each pixel is represented by a 16-bit number RGB565 format



MSB

LSB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

Red Values

5 Bits

Decimal Range 0 to 31

Or 32 Possible Values

Green Values

6 Bits

Decimal Range 0 to 63

Or 64 Possible Values

Blue Values

5 Bits

Decimal Range 0 to 31

Or 32 Possible Values

How to DrawDot ?

- Use the above idea to implement your DrawDot
- You can build your own function OR by the assistance of the two functions below.

```
void LCD_OpenWindow(uint16_t usCOLUMN, uint16_t usPAGE, uint16_t
     usWidth, uint16_t usHeight);
void LCD_FillColor(uint32_t alAmout_Point, uint16_t usColor);
```

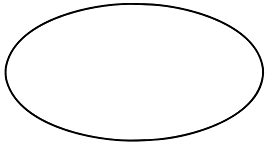
- If you want to use the above functions, make sure you understand what they do.

Task 3: LCD_DrawEllipse

- Use your DrawDot to implement a DrawEllipse

```
void LCD_DrawEllipse(uint16_t usC, uint16_t usP, uint16_t SR, uint16_t LR,
     uint16_t usColor);
```

  - where (usC, usP) represents the center point of the Ellipse
  - SR is the short radius of the Ellipse
  - LR is the long radius of the Ellipse
  - usColor represents the Ellipse color.



- The Ellipse should be a Hollow Ellipse with thickness of 1 pixel.

- In your main.c, verify your LCD_DrawEllipse function by

```
 LCD_DrawEllipse(120, 160, 25, 75, BLACK);
```

Task 4: Combining with LAB2

- Combining your knowledge with LAB2 and your knowledge for Task 1, modify your main.c such that
- After K2 is pressed, anything showing on the screen will be cleared, and after that, it will display your Last Character of Chinese Name shown on your Student ID Card.
  - Example 陳大文 , Display 文
  - If you do NOT have a Chinese Name shown on your Student ID Card, you need to display one character depending your student ID as shown on next page

Task 4: Chinese Character Table

Character	六國論
00 to 09	六國破滅　非兵不利　戰不
10 to 19	善　弊在賂秦　賂秦而力虧
20 to 29	破滅之道也　或曰　六國互
30 to 39	喪　率賂秦耶　曰　不賂者以
40 to 49	賂者喪　蓋失強援　不能獨
50 to 59	完　故曰　弊在賂秦也　秦以
60 to 69	攻取之外　小則獲邑　大則
70 to 79	得城　較秦之所得與戰勝
80 to 80	而得者　其實百倍　諸侯之
90 to 99	所亡與戰敗而亡者　其實

- ID : 20123456
- Character to display
- 56 →秦

Task 4: Things to note

- Each Chinese Character should be at least 24 pixels x 24 pixels.
- You can use your own way to implement the Chinese Character, below are the questions you may need to think
  - Can you hardcode it by using multiple calls of LCD_DrawDot ?
    - LCD_DrawDot(10,20,BLACK);
    - LCD_DrawDot(10,21,BLACK);
    - LCD_DrawDot(10,22,BLACK);
    - …
  - There is a LCD_DrawLine available, can you implement using LCD_DrawLine ?
  - Can I implement it using an array ?
  - What is the array size ?
  - What is the data type for each element in the array ?
  - How many colors you want to display in the Chinese Character ? How does it affect your array size and datatype ?
  - Where should the array being stored ?

Task 4: Things to note

- TA will ask you how you implemented your Chinese Character.
- You can refer to the DrawChar on how the English Characters are implemented.
- You can use any Chinese Character fonts you like as long as the character is clearly readable
- If there is a need for you to declare an array, please remember to declare it outside main() as a global variable.
- There is a tool you can use if you want. http://dotmatrixtool.com/ However, if you are going to use this tool, please make sure you understand before use. The TA will ask about your understanding of the data generated by the tool.
- Please note that I use K2, NOT K1 in this LAB, please refer to the MINI V3 schematic for the connections of K2

END