ECE 271Microcomputer Architecture and Applications
Lab 2: Liquid Crystal Display (LCD) Driver in C
Instructor: Prof. Yifeng Zhu
Spring 2015

Goals
1. Understand alternative function of GPIO pins
2. Understand basic concepts of a LCD driver, particularly *Bias* and *Duty Ratio*
3. Understand concepts of double buffer memory to ensure the coherency of the displayed information
4. Understand clock configurations of GPIO pins and LCD drivers

Pre-lab Assignment:
1. Read Chapter 17 of Textbook
2. Complete the pin configuration tables included in this handout

In-Lab Assignment:
1. Complete LCD_PIN_Init( ) and LCD_Configure ( )
2. Complete LCD_Display_Name( ) to display the first five letters of your last name.
3. Complete LCD_DisplayString() to display a short string (letters and numbers only, length ≤ 6)
4. Something cool. This following gives a few examples.
    a. LCD cool animations
    b. LCD scrolling to display a long string
    c. Set LCD contrast min-->max-->min by pressing user button

Introduction
PIN configuration: A total of 28 GPIO pins from Port A, B, and C are used to drive the LCD segment, as shown below. The duty ratio of this LCD is 4 and therefore there are four common terminals (COM0-COM3), which are connected to four GPIO pins. The other 24 GPIO pins are mapped to pixel bits stored in the internal LCD RAM. The mapping between GPIO pins and LCD RAM are given in Textbook. Each pin should be configured as Alternative Function 11 (LCD Driver).

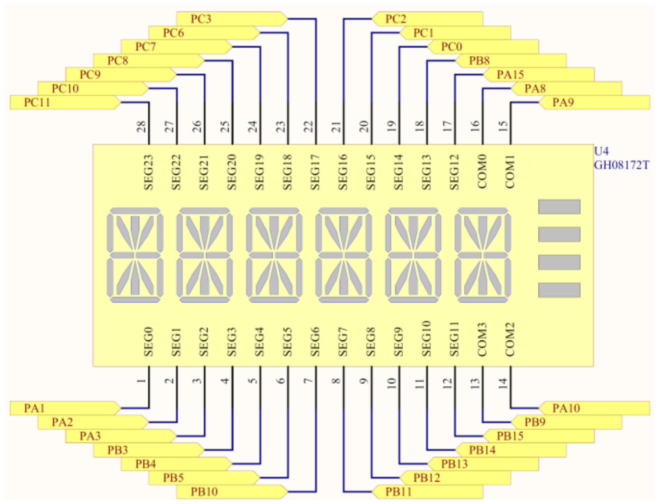


Figure 1. PIN connection to six 14-segment digits and 4 bars.

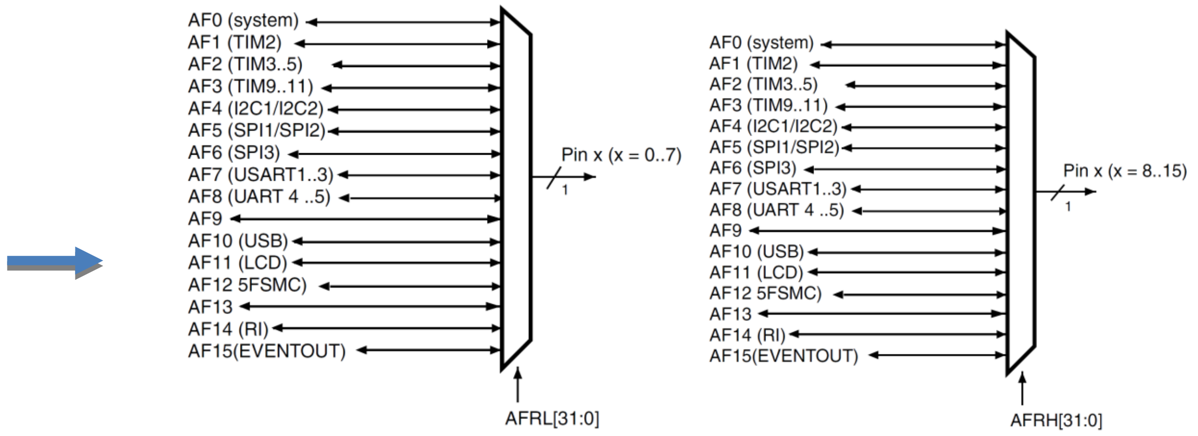
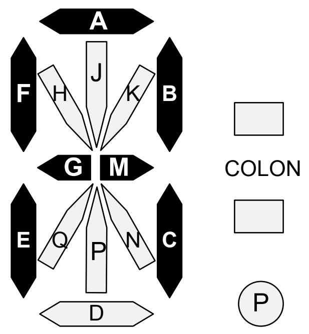


Figure 2. Selecting an alternate function. GPIOx_AFRL[31:00] defines the alternate function for pins 0 to 7, and GPIO_AFHL for pins 8 to 15. To drive LCD, the Alternative Function of all pins used by the LCD driver has to be set as Alternative Function 11 (LCD).

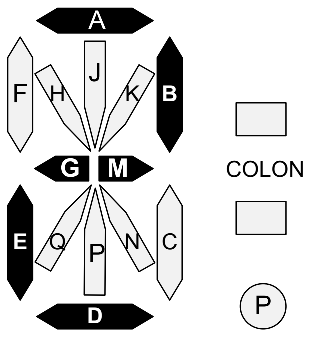
Character Encoding. Each digit consists of 16 display segments, including 14 segments for the digit, 1 segment of the colon, and 1 segment for the floating point. We use a 16-bit binary value to encode an alphabetic letter and a digit number. When a segment becomes visible, the corresponding bit in its 16-bit code is set. As shown in the following two tables, character "A" and "2" are encoded as `0xFE00` and `0xF500`, respectively.

Refer to Textbook for a complete description of encoding.



G	B	M	E	
1	1	1	1	0xF
F	A	C	D	
1	1	1	0	0xE
Q	K	Q	P	
0	0	0	0	0x0
H	J	DP	N	
0	0	0	0	0x0

Encoding "A" as `0xFE00`



G	B	M	E	
1	1	1	1	0xF
F	A	C	D	
0	1	0	1	0x5
Q	K	Q	P	
0	0	0	0	0x0
H	J	DP	N	
0	0	0	0	0x0

Encoding "2" as `0xF500`

LCD Driver. The hardware driver for LCD is built within the processor. The input of the hardware is the LCD RAM, which should be set up by software. The output of the hardware driver is voltage signals for 28 GPIO pins, which are connected to the LCD.

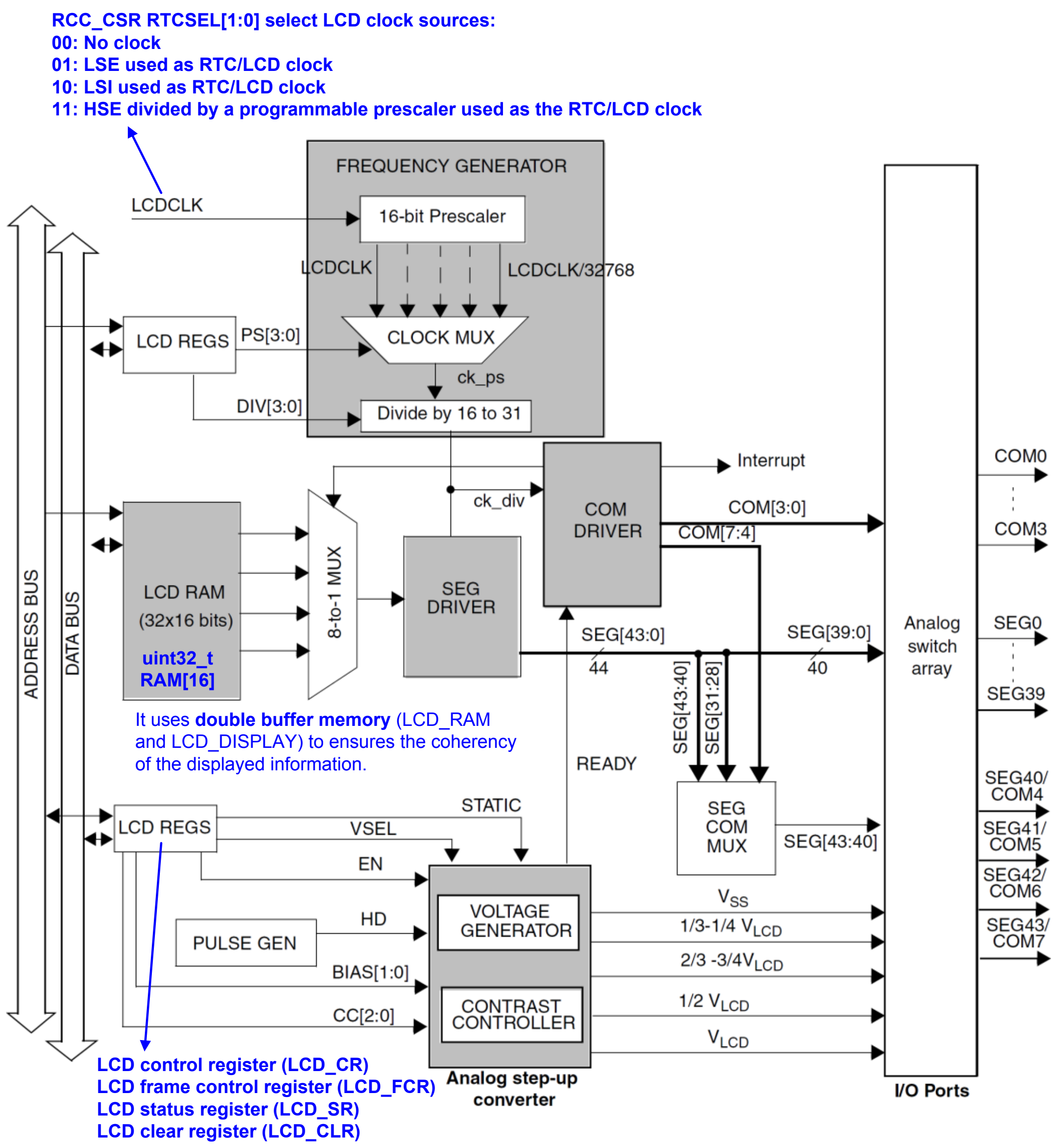


Figure 3. The built-in LCD driver. The signals of common terminals and the segment lines are automatically built according to the corresponding pixel bits stored in LCD RAM.

The controller uses double buffer memory to ensure the coherency of the displayed information without having to use interrupt the control of LCD_RAM modification.

Application writes the pixel bits into LCD_RAM by using the APB address bus and APB data bus. After the completion of modification to LCD_RAM, the software driver sets the ***Update Display Request*** (UDR) flag in the LCD Status Register (LCD_SR). The UDR requests the controller to copy the pixel bits in LCD_RAM to the second buffer (LCD_DISPLAY). The controller then generates the signals of common terminals (COM0-COM3) and segment lines (SEG0-SEG43) to drive the external LCD.

Before writing bit pixels into the LCD_RAM memory, the application should wait until the UDR flag is cleared. After writing into LCD_RAM, the application set up the UDR flag to transfer the updated data to the second level buffer LCD_DISPLAY. The UDR bit stays set until the end of the update and during this time the LCD_RAM is write-protected.

After setting the UDR flag, the application should wait until the Update Display Done (UDD) flag in the LCD Status Register (LCD_SR) is set.

The update, i.e. UDR = 1 and UDD = 0, will not be served until the display is enabled. Setting LCDEN of the LCD Control Register (LCD_CR) can enable the display.

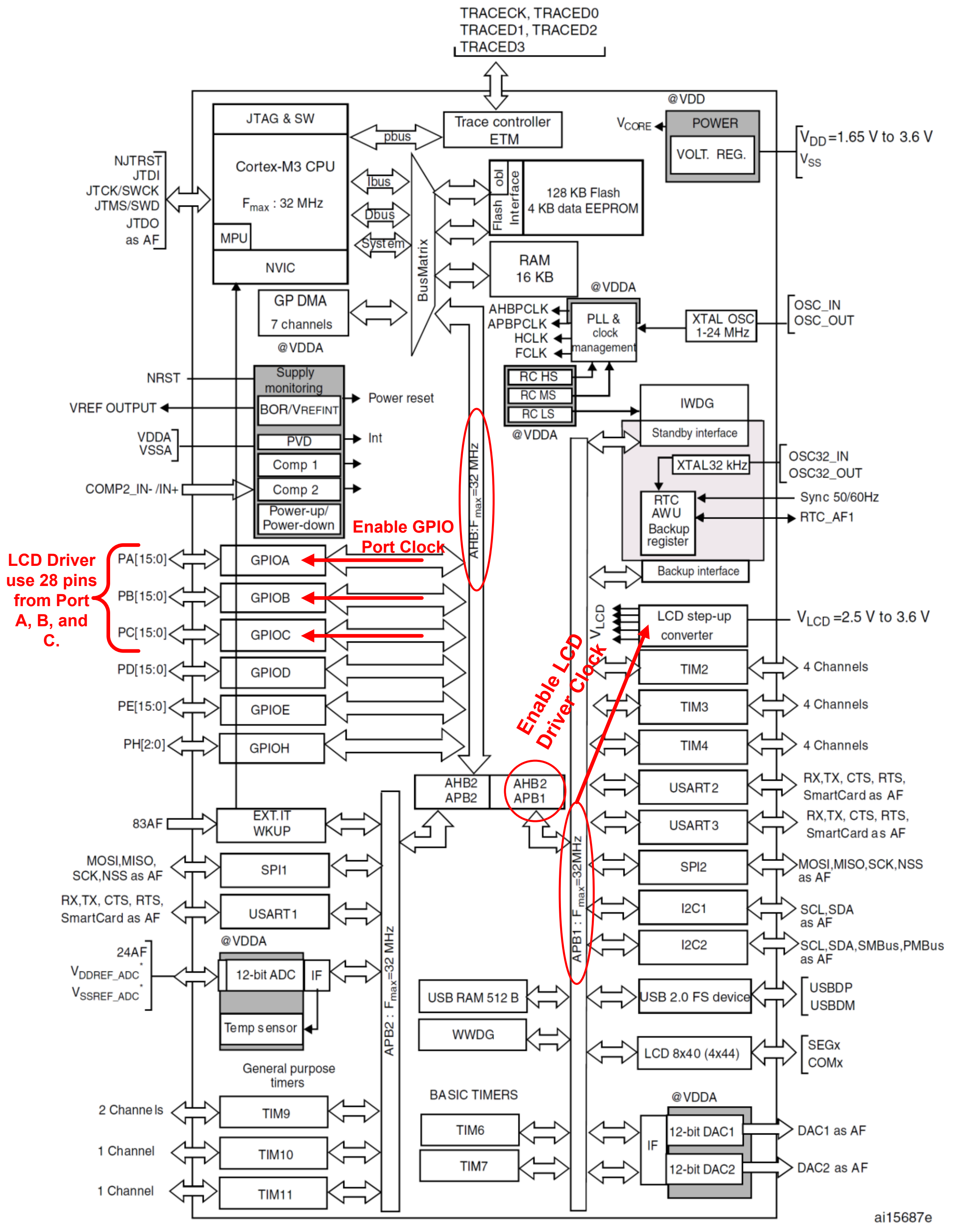


Figure 4. Enable the clock of LSE, Port A, B, C, and LCD/RTC

Lab 2:Pre-Lab Assignment

Student Name: ______________________
TA: ______________________
Time & Date: ______________________

1. Configure Port A: Pin 1, 2, 3, 8, 9, 10, and 15 as Alternative Function Mode

GPIO Mode: Digital Input (00, reset), Digital Output(01), Alternative Function(10), Analog(11)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOA MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
MASK																																
VALUE																																

GPIOA Mode Register MASK Value = 0x______________________ (in HEX)
GPIOA Mode Register Value = 0x______________________ (in HEX)

Configure Port A: Pin 1, 2, 3, 8, 9, 10, and 15 as Alternative Function 11 (0x0B)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOA AFR[0]	AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]				AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
MASK																																
VALUE																																
GPIOA AFR[1]	AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]				AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[3:0]			
MASK																																
VALUE																																

GPIOA Alternative Function Register [0] MASK = 0x______________________ (in HEX)
GPIOA Alternative Function Register [0] = 0x______________________ (in HEX)
GPIOA Alternative Function Register [1] MASK = 0x______________________ (in HEX)
GPIOA Alternative Function Register [1] = 0x______________________ (in HEX)

2. Configure Port B: Pin 3, 4, 5, 8, 9, 10, 11, 12, 13, 14, and 15 as Alternative Function Mode

GPIO Mode: Digital Input (00, reset), Digital Output(01), Alternative Function(10), Analog(11)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOB MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
MASK																																
VALUE																																

GPIOB Mode Register MASK Value = 0x________________________ (in HEX)
GPIOB Mode Register Value = 0x________________________ (in HEX)

Configure Port B: Pin 3, 4, 5, 8, 9, 10, 11, 12, 13, 14, and 15 as Alternative Function 11 (0x0B)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOB AFR[0]	AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]				AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
MASK																																
VALUE																																
GPIOB AFR[1]	AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]				AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[3:0]			
MASK																																
VALUE																																

GPIOB Alternative Function Register [0] MASK = 0x________________________ (in HEX)
GPIOB Alternative Function Register [0] = 0x________________________ (in HEX)
GPIOB Alternative Function Register [1] MASK = 0x________________________ (in HEX)
GPIOB Alternative Function Register [1] = 0x________________________ (in HEX)

3. Configure Port C: Pin 0, 1, 2, 3, 6, 7, 8, 9, 10, 11 as Alternative Function Mode

GPIO Mode: Digital Input (00, reset), Digital Output(01), Alternative Function(10), Analog(11)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOC MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
MASK																																
VALUE																																

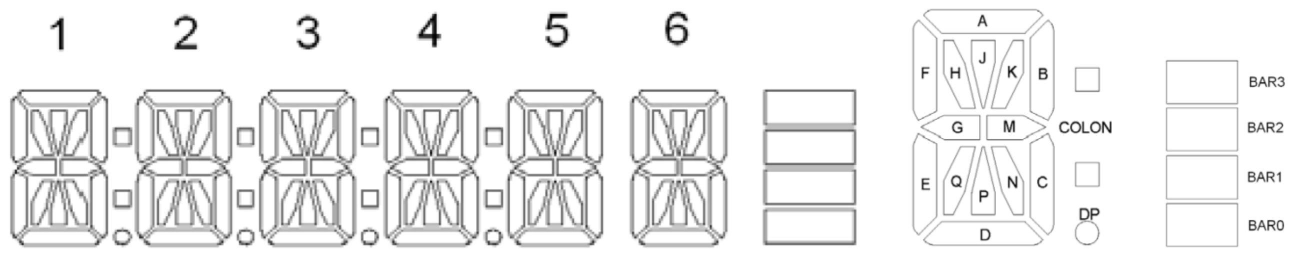
GPIOC Mode RegisterMASK Value = 0x_________________________ (in HEX)
GPIOC Mode Register Value = 0x_________________________ (in HEX)

Configure Port C: Pin 0, 1, 2, 3, 6, 7, 8, 9, 10, 11 as Alternative Function 11 (0x0B)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOC AFR[0]	AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]				AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
MASK																																
VALUE																																
GPIOC AFR[1]	AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]				AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[3:0]			
MASK																																
VALUE																																

GPIOC Alternative Function Register [0] MASK = 0x_________________________ (in HEX)
GPIOC Alternative Function Register [0] = 0x____________________________ (in HEX)
GPIOC Alternative Function Register [1] MASK = 0x_________________________ (in HEX)
GPIOC Alternative Function Register [1] = 0x____________________________ (in HEX)

Write down your last name, and complete the following table.



Your Last Name: ______________________________ (First Six Characters)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
			1G	1B	2G	2B	3G	3B			4G	4B	5G	5B	6B	6G	6M	6E	5M	5E	4M	4E	3M	3E	2M					2E	1M	1E
LCD_RAM (COM0)	0	0							0	0																0	0	0	0			
	Reserved																				S43	S42	S41	S40	S39	S38	S37	S36	S35	S34	S33	S32
																					0	0	0	0	0	0	0	0	0	0	0	0
	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
			1F	1A	2F	2A	3F	3A			4F	4A	5F	5A	6A	6F	6C	6D	5C	5D	4C	4D	3C	3D	2C					2D	1C	1D
LCD_RAM (COM1)	0	0							0	0																0	0	0	0			
	Reserved																				S43	S42	S41	S40	S39	S38	S37	S36	S35	S34	S33	S32
																					0	0	0	0	0	0	0	0	0	0	0	0
	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
			1Q	1K	2Q	2K	3Q	3K			4Q	4K	5Q	5K	6K	6Q	Bar 1	6P	Bar 3	5P	4Col	4P	3Col	3P	2Col					2P	1Col	1P
LCD_RAM (COM2)	0	0							0	0																0	0	0	0			
	Reserved																				S43	S42	S41	S40	S39	S38	S37	S36	S35	S34	S33	S32
																					0	0	0	0	0	0	0	0	0	0	0	0
	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
			1H	1J	2H	2J	3H	3J			4H	4J	5H	5J	6J	6H	Bar 0	6N	Bar 2	5N	4DP	4N	3DP	3N	2DP					2N	1DP	1N
LCD_RAM (COM3)	0	0							0	0																0	0	0	0			
	Reserved																				S43	S42	S41	S40	S39	S38	S37	S36	S35	S34	S33	S32
																					0	0	0	0	0	0	0	0	0	0	0	0

LCD_RAM is an array of 32-bit unsigned integers.

LCD_RAM[0] = 0x________________________ (in Hex) LCD_RAM[4] = 0x________________________ (in Hex)

LCD_RAM[2] = 0x________________________ (in Hex) LCD_RAM[6] = 0x________________________ (in Hex)

Lab 2: In-Lab Assignment

Book Errata:
On Page 365, in the flow chart (Figure 17-7), the LCD configuration function ***LCD_Configure*** needs to set up the pulse on duration to eliminate the disappearance of some segments.

```
// Set Pulse ON duration
// Use high drive internal booster to provide large drive current
// Set the duration that the low-resister voltage divider is used
LCD->FCR |= 0x7 << 4;     // PON[2:0] = 0x111
```

The basic requirement of this lab is to display your last name on the LCD. Refer to Textbook for the flow charts.

Notes:

1. The code uses the ***stm32l1xx.h*** head file provided in the STM library. It includes many useful macro definitions and data structures. Using them makes your code easier to understand and debug.

2. The program code to initialize the LCD clock is provided to you. The function is ***LCD_Clock_Init***();
   a. The LCD clock is the same clock as the Real-time clock (RTC). RTC clock domain is protected by default. To configure the LCD clock source, the RTC domain needs to be unlocked first by writing "0xCA" and "0x53" to the RTC->WPR register.

3. You are required to implement four functions:
   a. ***LCD_PIN_Init***() that enables GPIO clocks and configures GPIO pins as the alternative function 11 (LCD)
   b. ***LCD_Configure***() that performs the LCD configuration in the flow chart
   c. ***LCD_Display_Name***() that display the first six letters of your last name
   d. ***LCD_Display_String***() that sets up the LCD_RAM and displays the input string on LCD.
   e. ***LCD_Clear***() that clear the LCD screen.

4. Examples of something cool
   a. LCD cool animations
   b. LCD scrolling to display a long string
   c. Set LCD contrast min → max → min by pressing user button
   d. Something really cool

Lab Demo Questions:

1. In the LCD_WriteChar( ) function, why we have to have the following while statement
   while ((LCD->SR & LCD_SR_UDD) == 0);
2. What clock is used to drive the LCD? How can you find out? (Hint: check RCC register value in debug environment)
3. Explain to TA why double-buffering can ensure the coherency of the displayed information

Lab 2: Post-Lab Assignment

Answer the following questions in the file Readme.md and submit it with your lab code to the gitlab server.

1. Suppose the duty ratio of a LCD display is ¼ and it has a total of 120 display segments (pixels). How many pins are required to drive this LCD?

2. Figure 2 shows the alternative function selection of a GPIO pin. Can a GPIO pin perform all functions listed in Figure 2?

3. Figure 3 shows the basic diagram of LCD driver. Is this driver built in within the processor chip? What is the function of the COM driver and SEG driver?

4. How many pixels can the STM32L processor LCD driver drive? How large is the LCD_RAM in terms of bits? (Read STM32L Reference Manual)

5. How many pixels on the LCD installed on the STM32L discovery kit? Explain why many LCD_RAM registers (such as LCD_RAM[1], LCD_RAM[3], LCD_RAM[5]) are not used for STM32L discovery kit?