Experiment 1: General Propose I/Os (GPIOs)

In this experiment, student will learn how to configurate the GPIOs of the STM32F103C8T6 for both Input and Output modes. LEDs will be used to display the output signals while push switches will be used to capture user input. LCD module will be use to demonstrate the advanced output application on KU lab board.

1. LEDs display output

In this topic, the on-board LEDs will be used to demonstrate the output mode from GPIOs pins. There are 3 LEDs on the system, the first LED is connected to PC13 and on the Blue-pill board while two LEDs are on the KU Lab board and connected to PB1 (T3C4) and PB15 respectively as shown in Figure 1.

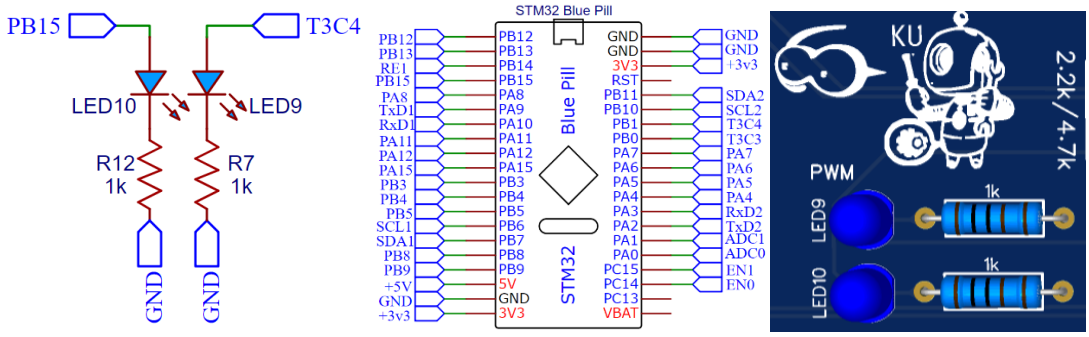


Figure 1. LEDs on KU lab Board

Each GPIO pin can be configured as 4 types; 1. Digital INPUT, 2. Digital OUTPUT, 3. Analog INPUT and 4. Alternate Function by modified the value inside the GPIO registers according to each pin via GPIOx_CRL and GPIOx_CRH. Operation mode of the GPIO pin is explained in Figure 2. Since the GPIO modules requires clock signal in order to work properly, thus, the programmer must assign clock to each GPIO module separately through RCC_APB2ENR. To configure the GPIO pins of STM32F103C8T6 to be the output pins, the bit CNF1, CNF0, MODE1 and MODE0 must be set to 0001 respectively for 50 MHz push pull output pin configuration. Thanks to the TrueSTUDIO IDE and SPL library, the programmer can use API for configuring the system clock and GPIO pin easily. RCC_APB2PeriphClockCmd( ) is the API function for enabling the GPIO clock. GPIO_Init( ) is the API function for setting the GPIO pin mode.

Configuration mode		CNF1	CNF0	MODE1	MODE0	PxODR register
General purpose output	Push-pull	0	0	01 10 11 see *Table 21*		0 or 1
	Open-drain		1			0 or 1
Alternate Function output	Push-pull	1	0			Don't care
	Open-drain		1			Don't care
Input	Analog	0	0	00		Don't care
	Input floating		1			Don't care
	Input pull-down	1	0			0
	Input pull-up					1

MODE[1:0]	Meaning
00	Reserved
01	Maximum output speed 10 MHz
10	Maximum output speed 2 MHz
11	Maximum output speed 50 MHz

Figure 2. GPIO pin configuration.

Experiment 1.1

1. First, new project according to experiment 0 and assign project name as Lab1.
2. Write down the **my_delay** function to the **main.c** file in the project before main( ) function as below:

```
1. void my_delay(unsigned int count)
2. {
3.    while(count>0)
4.    {   count--;  }
5. }
```

3. Write down the **GPIO_configuration** function to the **main.c** file in the project before main( ) function as below:

```
1. void GPIO_Configuration()
2. {
3.  // Define variable type GPIO
4.  GPIO_InitTypeDef GPIO_InitStruct;
5.  // Enable clock for GPIOC and GPIO B
6.  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB | RCC_APB2Periph_GPIOC , ENABLE);
7.  // Configure PC13 as output push pull 50MHz
8.  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_13;
9.  GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
10. GPIO_InitStruct.GPIO_Mode = GPIO_Mode_Out_PP;
11. GPIO_Init(GPIOC, &GPIO_InitStruct);
```

```
12. //  Configure PB1 and PB15 as output push pull 50MHz
13. GPIO_InitStruct.GPIO_Pin = GPIO_Pin_1 | GPIO_Pin_15;
14. GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
15. GPIO_InitStruct.GPIO_Mode = GPIO_Mode_Out_PP;
16. GPIO_Init(GPIOB, &GPIO_InitStruct);
17.}
```

4. Edit the **main( )** function in **main.c** as follow:

```
1. int main(void)
2. {
3.   int i = 0 , temp = 0;
4.   GPIO_Configuration();
5.   GPIO_WriteBit(GPIOC,GPIO_Pin_13,1);
6.   GPIO_WriteBit(GPIOB,GPIO_Pin_1,1);
7.   GPIO_WriteBit(GPIOB,GPIO_Pin_15,1);
8.   while(1)
9.   {
10.     temp = i % 2;
11.     GPIO_WriteBit(GPIOB,GPIO_Pin_1,temp);
12.     GPIO_WriteBit(GPIOB,GPIO_Pin_15,!temp);
13.     my_delay(400000);
14.     i++;
15.  }
16.}
```

5. Compile and program into the board. Write down the result from the experiment 1.1 code

_____

_____

_____

_____

_____

_____

Lab instructor / TA signature

2. Push-pull switch input

In this topic, the push-pull switches will be assigned as the GPIO input pins of the STM32F103C8T6. From the schematic in Figure 3, the push-pull switches are connected to PB3, PB4, PB8 and PB9. From the circuit, when the switch is pushed then it will be shorted to the ground and generate logic 0 to GPIO pin, but if the switch is not pushed then it will give +3.3v or logic 1 to the GPIO pin.

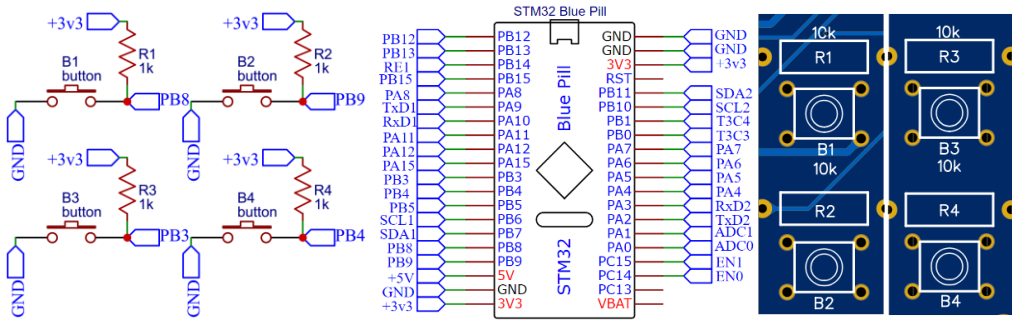


Figure 3. Push-pull switches schematic.

Experiment 1.2

1. Use the same project as experiment 1.1. Edit the **GPIO_configuration** function from experiment 1.1 in the **main.c** as below:

```
void GPIO_Configuration()
{
 GPIO_InitTypeDef GPIO_InitStruct;
 // Enable clock for GPIOC and GPIOB and AFIO and Disable JTAG Mode
 RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB | RCC_APB2Periph_GPIOC , ENABLE);
 RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
 GPIO_PinRemapConfig(GPIO_Remap_SWJ_JTAGDisable, ENABLE);
 // Configure PC13 as output push pull 50MHz
 GPIO_InitStruct.GPIO_Pin = GPIO_Pin_13;
 GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
 GPIO_InitStruct.GPIO_Mode = GPIO_Mode_Out_PP;
 GPIO_Init(GPIOC, &GPIO_InitStruct);
 //  Configure PB1 and PB15 as output push pull 50MHz
 GPIO_InitStruct.GPIO_Pin = GPIO_Pin_1 | GPIO_Pin_15;
 GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
 GPIO_InitStruct.GPIO_Mode = GPIO_Mode_Out_PP;
 GPIO_Init(GPIOB, &GPIO_InitStruct);
  //  Configure PB3,PB4 and PB8,PB9 as input floating
 GPIO_InitStruct.GPIO_Pin = GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_8 | GPIO_Pin_9;
 GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
 GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN_FLOATING;
 GPIO_Init(GPIOB, &GPIO_InitStruct);
}
```

2. Edit the **main( )** function in **main.c** as follow:

```
int main(void)
{
  int state = 0 , ReadValue = 0;
  GPIO_Configuration();
  GPIO_WriteBit(GPIOC,GPIO_Pin_13,1);
  GPIO_WriteBit(GPIOB,GPIO_Pin_1,1);
  GPIO_WriteBit(GPIOB,GPIO_Pin_15,1);
  while(1)
  {
     ReadValue = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_3);
     if (ReadValue==0)
         GPIO_WriteBit(GPIOB,GPIO_Pin_1,1);
     ReadValue = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_4);
     if (ReadValue==0)
         GPIO_WriteBit(GPIOB,GPIO_Pin_1,0);
     ReadValue = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_8);
     if (ReadValue==0)
         GPIO_WriteBit(GPIOB,GPIO_Pin_15,1);
     ReadValue = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_9);
     if (ReadValue==0)
         GPIO_WriteBit(GPIOB,GPIO_Pin_15,0);
     if (state == 0) {
         GPIO_WriteBit(GPIOC,GPIO_Pin_13,1);
         state = 1;
     }
     if (state == 1) {
         GPIO_WriteBit(GPIOC,GPIO_Pin_13,0);
         state = 0;
     }
     my_delay(400000);
  }
}
```

3. Compile and program into the board. Write down the result from the experiment 1.2 when push each switch.

_____

_____

_____

_____

_____

Lab instructor / TA signature

3. LCD module

In this topic, LCD module with HD44780 controller is explained and experimented. Liquid Crystal Display or LCD is the display that uses the polarization property of the liquid crystal that will blend light direction and create dark area. By using this technology, the display can create character or graphic on the LCD screen as shown in Figure 4. Since there is a complexity when blend the light using polarization, the LCD driver HD44780 is used. This driver is used to control the high voltage for generating light polarization effect to the LCD screen. HD44780 is also used to generate patterns that appear on the screen as shown in Figure 5. Therefore, in order to use LCD module, the communication between STM32F103C8T6 and HD44780 is required via a parallel communication through LCD module pins as shown in Table 1 using API command that explained in Figure 6.

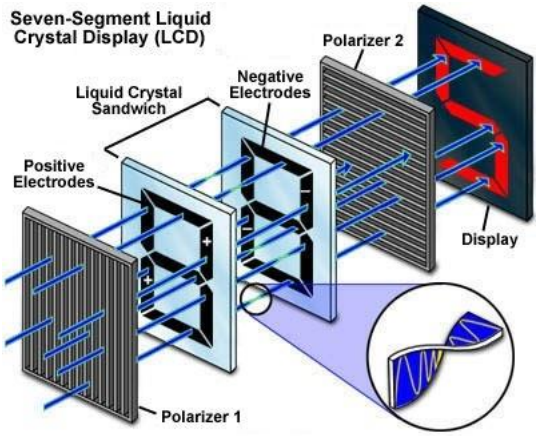


Figure 4. LCD module and Characters output.

Source: www.olympusmicro.com/primer/lightandcolor/polarization.html

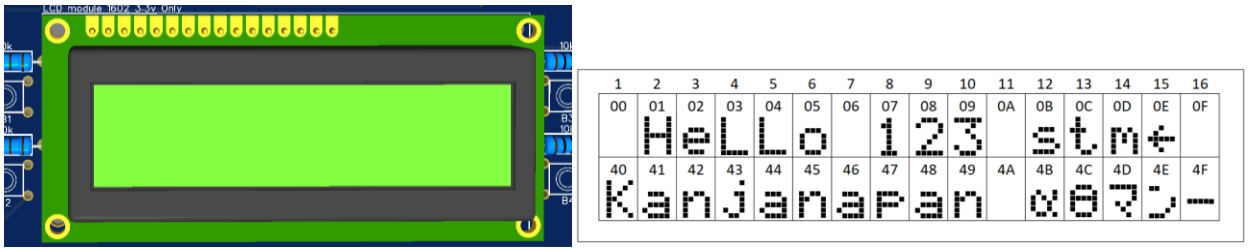


Figure 5. LCD module and Characters output.

Table 1. LCD module pins

Pin	Pin Name	Detail
1	V_{SS}	Ground (GND)
2	V_{DD}	Volt supply 3.3 – 5 volts
3	V_0	Voltage Reference for LCD Contrast
4	RS	Input pin for selecting HD44780 Register inside LCD. If RS=0, command register is selected and, If RS = 1, data register is selected
5	R/W	Input pin for commanding HD44780 Register inside LCD. If R/W = 0 means write data command to LCD and if R/W = 1 means read date from LCD
6	E	Input pin used for Enable LCD module active when E is a falling edge
7 - 10	DB0 – DB3	Low bit data (4bits)
11- 14	DB4 – DB7	High bit data (4bits)
15	LED+	Power source for Back Light 'A' (V_{DD})
16	LED-	Power source for Back Light 'K' (GND)

	Code										
Instruction	**RS**	**R/$\overline{W}$**	**DB7**	**DB6**	**DB5**	**DB4**	**DB3**	**DB2**	**DB1**	**DB0**	**Description**
Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address 0 in address counter.
Return home	0	0	0	0	0	0	0	0	1	—	Sets DDRAM address 0 in address counter. Also returns display from being shifted to original position. DDRAM contents remain unchanged.
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.
Display on/off control	0	0	0	0	0	0	1	D	C	B	Sets entire display (D) on/off cursor on/off (C), and blinking of cursor position character (B).
Cursor or display shift	0	0	0	0	0	1	S/C	R/L	—	—	Moves cursor and shifts display without changing DDRAM contents.
Function set	0	0	0	0	1	DL	N	F	—	—	Sets interface data length (DL), number of display lines (N), and character font (F).
Set CGRAM address	0	0	0	1	ACG	ACG	ACG	ACG	ACG	ACG	Sets CGRAM address. CGRAM data is sent and received after this setting.
Set DDRAM address	0	0	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD	Sets DDRAM address. DDRAM data is sent and received after this setting.
Read busy flag & address	0	1	BF	AC	AC	AC	AC	AC	AC	AC	Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.

Figure 6. LCD Module API

There are two methods to command LCD module which are 4bits data interface and 8bits data interface. Due to the limitation of GPIO pins of the Blue-pill board, the 4bits data interface is selected as the preferred method. The flow chart for initialize the LCD module using 4bits data interface is shown in Figure 7.

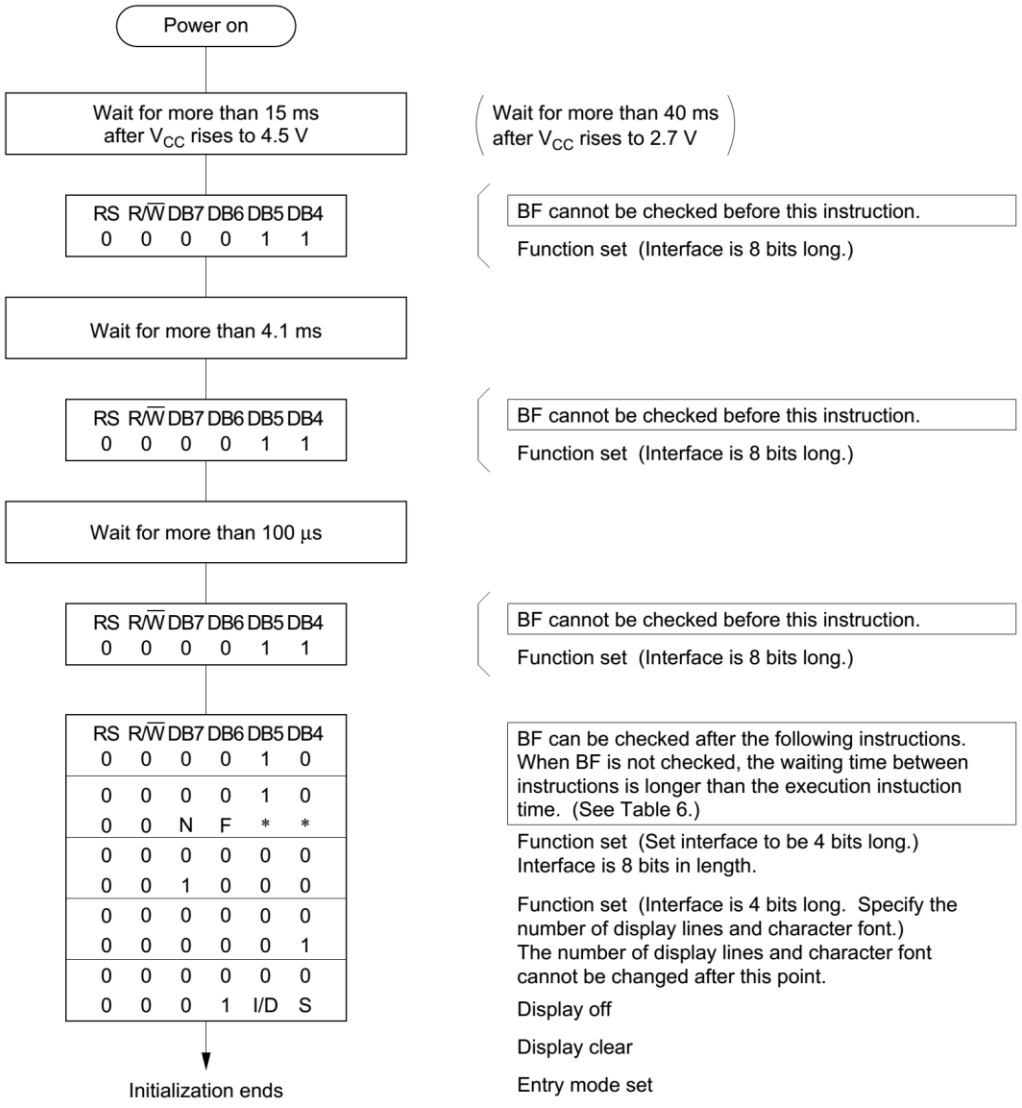


Figure 7. LCD Module 4bits initialization flow chart.

According to the KU Lab board schematic shown in Figure 8, PA4, PA5, PA6 and PA7 are used for 4bits data interface for DB4, DB5, DB6 and DB7 respectively. PB12 connects to E pin and PB13 connects to RS pin. The LCD_Rv is connected to a variable register for LCD screen contrast adjustment.

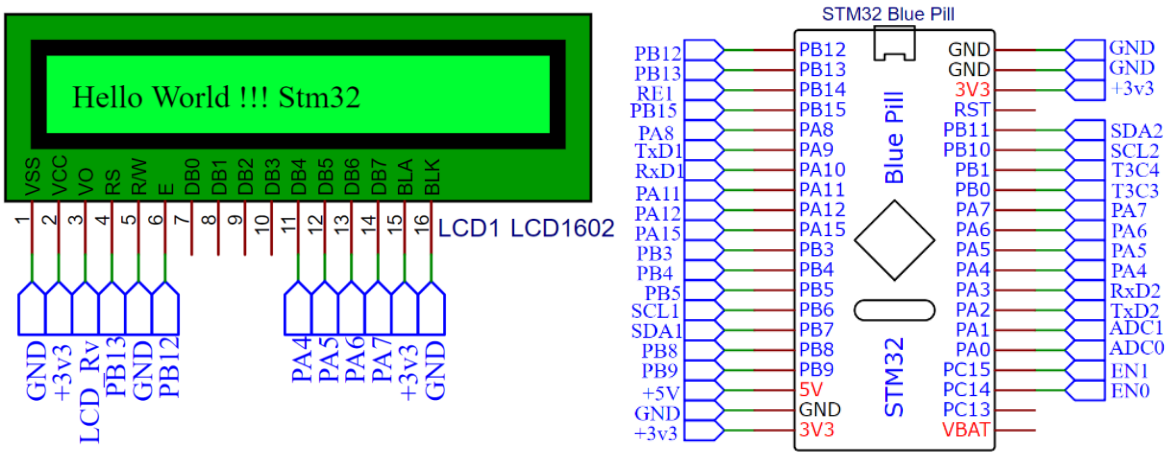


Figure 8. LCD schematic in KU lab board.

Experiment 1.3

1. Use the same project as experiment 1.1. Edit the **GPIO_configuration** function from experiment 1.1 in the **main.c** as below:

```
void GPIO_Configuration()
{
 // Define variable type GPIO
 GPIO_InitTypeDef GPIO_InitStruct;
 // Enable clock for GPIOA and GPIO B
 RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB , ENABLE);
 //  Configure PB12 and PB13 as output push pull 50MHz
 GPIO_InitStruct.GPIO_Pin = GPIO_Pin_12 | GPIO_Pin_13;
 GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
 GPIO_InitStruct.GPIO_Mode = GPIO_Mode_Out_PP;
 GPIO_Init(GPIOB, &GPIO_InitStruct);
 //  Configure PA3,PA4 and PA8,PA9 output push pull 50MHz
 GPIO_InitStruct.GPIO_Pin = GPIO_Pin_4 | GPIO_Pin_5 | GPIO_Pin_6 | GPIO_Pin_7;
 GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
 GPIO_InitStruct.GPIO_Mode = GPIO_Mode_Out_PP;
 GPIO_Init(GPIOA, &GPIO_InitStruct);
}
```

2. Add the new function **DelayMC( )** the **main.c** after include section as follow:

```
void DelayMC(unsigned int ms)
{
          volatile unsigned int nCount;
          nCount = (unsigned int)((72000000/4000000)*ms*1000 - 10);
          while(nCount--);
}
```

3. Add the new functions **Pulse_LCD( )**, **LCD_4bits_Init( )** and **LCD_send_data()** into **main.c** before main ( ) as follow:

```
void pulse_LCD()
{
   // E --> PB12
   DelayMC(2);
   GPIO_WriteBit(GPIOB,GPIO_Pin_12,1);
   DelayMC(2);
   GPIO_WriteBit(GPIOB,GPIO_Pin_12,0);
   DelayMC(2);
}
```

```
void LCD_4bits_Init()
{
    GPIO_WriteBit(GPIOB,GPIO_Pin_13,0);
    DelayMC(120);
    //send  0x30  1st time
    GPIO_WriteBit(GPIOA,GPIO_Pin_4,1);
    GPIO_WriteBit(GPIOA,GPIO_Pin_5,1);
    GPIO_WriteBit(GPIOA,GPIO_Pin_6,0);
    GPIO_WriteBit(GPIOA,GPIO_Pin_7,0);
    pulse_LCD();

    DelayMC(80);
    //send  0x30 2nd time
    GPIO_WriteBit(GPIOA,GPIO_Pin_4,1);
    GPIO_WriteBit(GPIOA,GPIO_Pin_5,1);
    GPIO_WriteBit(GPIOA,GPIO_Pin_6,0);
    GPIO_WriteBit(GPIOA,GPIO_Pin_7,0);
    pulse_LCD();

    DelayMC(50);
    //send  0x30  3rd time
    GPIO_WriteBit(GPIOA,GPIO_Pin_4,1);
    GPIO_WriteBit(GPIOA,GPIO_Pin_5,1);
    GPIO_WriteBit(GPIOA,GPIO_Pin_6,0);
    GPIO_WriteBit(GPIOA,GPIO_Pin_7,0);
    pulse_LCD();

    DelayMC(50);
   //send  0x20) to make LCD 4bits mode
    GPIO_WriteBit(GPIOA,GPIO_Pin_4,0);
    GPIO_WriteBit(GPIOA,GPIO_Pin_5,1);
    GPIO_WriteBit(GPIOA,GPIO_Pin_6,0);
    GPIO_WriteBit(GPIOA,GPIO_Pin_7,0);
    pulse_LCD();

    //-------------LCD_Config------------//


```

```
39.     DelayMC(50);
40.     //send function mode set
41.     GPIO_WriteBit(GPIOA,GPIO_Pin_4,0);
42.     GPIO_WriteBit(GPIOA,GPIO_Pin_5,1);
43.     GPIO_WriteBit(GPIOA,GPIO_Pin_6,0);
44.     GPIO_WriteBit(GPIOA,GPIO_Pin_7,0);
45.     pulse_LCD();
46.     GPIO_WriteBit(GPIOA,GPIO_Pin_4,0);
47.     GPIO_WriteBit(GPIOA,GPIO_Pin_5,0);
48.     GPIO_WriteBit(GPIOA,GPIO_Pin_6,0);
49.     GPIO_WriteBit(GPIOA,GPIO_Pin_7,1);
50.     pulse_LCD();
51.     //send LCD Cursor display set
52.     GPIO_WriteBit(GPIOA,GPIO_Pin_4,0);
53.     GPIO_WriteBit(GPIOA,GPIO_Pin_5,0);
54.     GPIO_WriteBit(GPIOA,GPIO_Pin_6,0);
55.     GPIO_WriteBit(GPIOA,GPIO_Pin_7,0);
56.     pulse_LCD();
57.     GPIO_WriteBit(GPIOA,GPIO_Pin_4,0);
58.     GPIO_WriteBit(GPIOA,GPIO_Pin_5,0);
59.     GPIO_WriteBit(GPIOA,GPIO_Pin_6,0);
60.     GPIO_WriteBit(GPIOA,GPIO_Pin_7,1);
61.     pulse_LCD();
62.     //send Entry Mode set
63.     GPIO_WriteBit(GPIOA,GPIO_Pin_4,0);
64.     GPIO_WriteBit(GPIOA,GPIO_Pin_5,0);
65.     GPIO_WriteBit(GPIOA,GPIO_Pin_6,0);
66.     GPIO_WriteBit(GPIOA,GPIO_Pin_7,0);
67.     pulse_LCD();
68.     GPIO_WriteBit(GPIOA,GPIO_Pin_4,0);
69.     GPIO_WriteBit(GPIOA,GPIO_Pin_5,1);
70.     GPIO_WriteBit(GPIOA,GPIO_Pin_6,1);
71.     GPIO_WriteBit(GPIOA,GPIO_Pin_7,0);
72.     pulse_LCD();
73.     //send clear LCD command
74.     GPIO_WriteBit(GPIOA,GPIO_Pin_4,0);
75.     GPIO_WriteBit(GPIOA,GPIO_Pin_5,0);
76.     GPIO_WriteBit(GPIOA,GPIO_Pin_6,0);
77.     GPIO_WriteBit(GPIOA,GPIO_Pin_7,0);
78.     pulse_LCD();
79.     GPIO_WriteBit(GPIOA,GPIO_Pin_4,1);
80.     GPIO_WriteBit(GPIOA,GPIO_Pin_5,0);
81.     GPIO_WriteBit(GPIOA,GPIO_Pin_6,0);
82.     GPIO_WriteBit(GPIOA,GPIO_Pin_7,0);
83.     pulse_LCD();
84.     //send LCD Cursor display set
85.     GPIO_WriteBit(GPIOA,GPIO_Pin_4,0);
86.     GPIO_WriteBit(GPIOA,GPIO_Pin_5,0);
87.     GPIO_WriteBit(GPIOA,GPIO_Pin_6,0);
88.     GPIO_WriteBit(GPIOA,GPIO_Pin_7,0);
89.     pulse_LCD();
90.     GPIO_WriteBit(GPIOA,GPIO_Pin_4,1);
91.     GPIO_WriteBit(GPIOA,GPIO_Pin_5,1);
92.     GPIO_WriteBit(GPIOA,GPIO_Pin_6,1);
93.     GPIO_WriteBit(GPIOA,GPIO_Pin_7,1);
94.     pulse_LCD();
95.
```

```
96.     //send clear LCD command
97.     GPIO_WriteBit(GPIOA,GPIO_Pin_4,0);
98.     GPIO_WriteBit(GPIOA,GPIO_Pin_5,0);
99.     GPIO_WriteBit(GPIOA,GPIO_Pin_6,0);
100.    GPIO_WriteBit(GPIOA,GPIO_Pin_7,0);
101.    pulse_LCD();
102.    GPIO_WriteBit(GPIOA,GPIO_Pin_4,1);
103.    GPIO_WriteBit(GPIOA,GPIO_Pin_5,0);
104.    GPIO_WriteBit(GPIOA,GPIO_Pin_6,0);
105.    GPIO_WriteBit(GPIOA,GPIO_Pin_7,0);
106.    pulse_LCD();
107.  }
```

```
1.  void LCD_send_data(unsigned int _data)
2.  {
3.
4.     unsigned char data_bit[4];
5.     unsigned char data_4bitlow, data_4bithigh;
6.     data_4bithigh=_data&0xf0;
7.     data_4bitlow=(_data<<4)&0xf0;
8.
9.     GPIO_WriteBit(GPIOB,GPIO_Pin_13,1);
10.
11.    data_bit[0] =      ( ((data_4bithigh>>4) & 0x01) ? 1 :0 ) ;
12.    data_bit[1] =      ( ((data_4bithigh>>5) & 0x01) ? 1 :0 ) ;
13.    data_bit[2] =      ( ((data_4bithigh>>6) & 0x01) ? 1 :0 ) ;
14.    data_bit[3] =      ( ((data_4bithigh>>7) & 0x01) ? 1 :0 ) ;
15.
16.    GPIO_WriteBit(GPIOA,GPIO_Pin_4,data_bit[0]);
17.    GPIO_WriteBit(GPIOA,GPIO_Pin_5,data_bit[1]);
18.    GPIO_WriteBit(GPIOA,GPIO_Pin_6,data_bit[2]);
19.    GPIO_WriteBit(GPIOA,GPIO_Pin_7,data_bit[3]);
20.    pulse_LCD();
21.
22.    data_bit[0] =      ( ((data_4bitlow>>4) & 0x01) ? 1 :0 ) ;
23.    data_bit[1] =      ( ((data_4bitlow>>5) & 0x01) ? 1 :0 ) ;
24.    data_bit[2] =      ( ((data_4bitlow>>6) & 0x01) ? 1 :0 ) ;
25.    data_bit[3] =      ( ((data_4bitlow>>7) & 0x01) ? 1 :0 ) ;
26.
27.    GPIO_WriteBit(GPIOA,GPIO_Pin_4,data_bit[0]);
28.    GPIO_WriteBit(GPIOA,GPIO_Pin_5,data_bit[1]);
29.    GPIO_WriteBit(GPIOA,GPIO_Pin_6,data_bit[2]);
30.    GPIO_WriteBit(GPIOA,GPIO_Pin_7,data_bit[3]);
31.    pulse_LCD();
32.
33.    GPIO_WriteBit(GPIOB,GPIO_Pin_13,0);
34. }
```

4. Edit the **main( )** function in **main.c** as follow:

```
int main(void)
{
  int i =0, state = 0;
  unsigned char text[]={84,104,73,115,32,105,83,32,65,32,76,97,66,'\0'};
  GPIO_Configuration();
  GPIO_WriteBit(GPIOB,GPIO_Pin_12,0);
  GPIO_WriteBit(GPIOB,GPIO_Pin_13,0);
  DelayMC(100);
  LCD_4bits_Init();
  DelayMC(100);
  while (text[i] != '\0')
  {
     LCD_send_data(text[i]);
     i++;
  }

  while(1);

}
```

5. Compile and program into the board. Write down the result from the experiment 1.3 in the picture below and explain what is the current LCD configuration according to Figure 6

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

_____

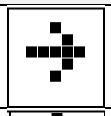
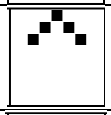
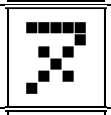
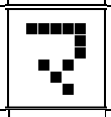
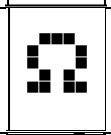
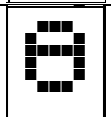
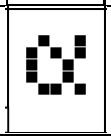
_____

_____

_____

Lab instructor / TA signature

6. Complete the values in the Table below if you have to show them on the LCD module

Character	Value
	
	
	
	
	
	
	

Lab instructor / TA signature

Experiment 1.4

1. Use the same project as experiment 1.3. Add function **LCD_send_command()** and **LCD_goto_position()** into the **main.c** as below:

```
void LCD_send_command(unsigned int _data)
{
   unsigned char data_bit[4];
   unsigned char data_4bitlow, data_4bithigh;
   data_4bithigh=_data&0xf0;
   data_4bitlow=(_data<<4)&0xf0;

   GPIO_WriteBit(GPIOB,GPIO_Pin_13,0);

   data_bit[0] =     ( ((data_4bithigh>>4) & 0x01) ? 1 :0 ) ;
   data_bit[1] =     ( ((data_4bithigh>>5) & 0x01) ? 1 :0 ) ;
   data_bit[2] =     ( ((data_4bithigh>>6) & 0x01) ? 1 :0 ) ;
   data_bit[3] =     ( ((data_4bithigh>>7) & 0x01) ? 1 :0 ) ;

   GPIO_WriteBit(GPIOA,GPIO_Pin_4,data_bit[0]);
   GPIO_WriteBit(GPIOA,GPIO_Pin_5,data_bit[1]);
   GPIO_WriteBit(GPIOA,GPIO_Pin_6,data_bit[2]);
   GPIO_WriteBit(GPIOA,GPIO_Pin_7,data_bit[3]);
   pulse_LCD();

   data_bit[0] =     ( ((data_4bitlow>>4) & 0x01) ? 1 :0 ) ;
   data_bit[1] =     ( ((data_4bitlow>>5) & 0x01) ? 1 :0 ) ;
   data_bit[2] =     ( ((data_4bitlow>>6) & 0x01) ? 1 :0 ) ;
   data_bit[3] =     ( ((data_4bitlow>>7) & 0x01) ? 1 :0 ) ;

   GPIO_WriteBit(GPIOA,GPIO_Pin_4,data_bit[0]);
   GPIO_WriteBit(GPIOA,GPIO_Pin_5,data_bit[1]);
   GPIO_WriteBit(GPIOA,GPIO_Pin_6,data_bit[2]);
   GPIO_WriteBit(GPIOA,GPIO_Pin_7,data_bit[3]);
   pulse_LCD();
}
```

```
void LCD_goto_position(unsigned char line,unsigned char position)
{
   unsigned char  cmd;
   if (position <= 15)
   {
         if(line==0)
               cmd= 0x80;
         if(line==1)
               cmd= 0xC0;
   cmd |= position;
   LCD_send_command(cmd);
   }
}
```

2. Edit the **main( )** function in **main.c** as follow:

```
int main(void)
{
  int i =0;
  unsigned char text2[]={117,80,114,111,32,76,97,98,'\0'};
  unsigned char text3[]={119,72,97,84 ,'\0'};
  unsigned char text4[]={94,95,95,95,94,97,'\0'};
  GPIO_Configuration();
  GPIO_WriteBit(GPIOB,GPIO_Pin_12,0);
  GPIO_WriteBit(GPIOB,GPIO_Pin_13,0);
  DelayMC(100);
  LCD_4bits_Init();
  DelayMC(100);
  LCD_goto_position(1,4);
  i=0;
  while (text2[i] != '\0')
  {
     LCD_send_data(text2[i]);
     i++;
  }
  LCD_goto_position(0,2);
  i=0;
  while (text3[i] != '\0')
  {
     LCD_send_data(text3[i]);
     i++;
  }
  LCD_goto_position(0,8);
  i=0;
  while (text4[i] != '\0')
  {
     LCD_send_data(text4[i]);
     i++;
  }
  while(1);
}
```

3. Compile and program into the board. Write down the result from the experiment 1.4 in the picture below and explain how function **LCD_goto_position()** works.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

_____

_____

_____

_____

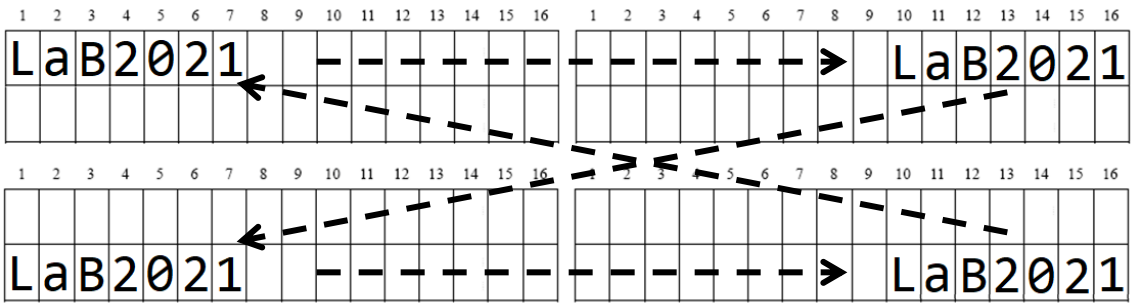
Lab instructor / TA signature

Homework

1. Write a program for displaying ‘STM32F103C8T6’ at the first line of LCD module and displaying the summation of student IDs in your group in the second line of LCD module

Lab instructor / TA signature

2. Write a program to 1. display “LaB2021” to show on the LCD screen at the first location of the first line. 2. This text will be moved from the left to the right of the first line until the last character ‘1’ arrive the last location of the first line. 3. This text will re-appear again on the first location of the second line and start to move again like the first line. 4. After the last character ‘1’ arrive the last location of the second line, then, this text will appear again in the first line and repeat the step 1. 2. 3. and 4 forever.



Lab instructor / TA signature