

Rep_Lab_05_B0829002

Problem Description:

在這個 Lab 裡，我們要在 basic 實作在按下按鈕的時候 LCD 螢幕上的指針會顯示出預先設定好的字元。

而 bonus 設一個捲動 LCD 螢幕的方法以及讓指針如果在有字元的地方可以將欲加入的字元插入指針所在的位置。

Code and explanations:

Basic:

Part 1: Port, page, in- and output, delay 設定

在這部分基本和上個 lab 一樣，將基本的輸入、輸出、看門狗計數器、頁面和延遲設定好。

```
1  #include "C8051F040.h"
2  #include "LCD.h"
3  #include <stdio.h>
4  #include <string.h>
5
6  char LCD_status;
7
8  void
9  LCD_PortConfig ()
10 {
11     //initialize SFR setup page
12     SFRPAGE = CONFIG_PAGE;           // Switch to configuration page
13
14     //setup the cross-bar and configure the I/O ports
15     XBR2 = 0xc0;
16     P3MDOUT = 0x3f;
17     P1MDIN = 0xff;
18
19     //set to normal mode
20     SFRPAGE = LEGACY_PAGE;
21 } //end of function LCD_PortConfig ()
22
23 /*****
24  The delay_lcd seem to be not long enough.
25  Try to change the default value as 10000...
26  *****/
27 unsigned int delay_lcd=1000;
28
29 void
30 LCD_Delay ()
31 {
32     int i;
33     for (i=0;i<delay_lcd;i++); // wait for a long enough time...
34 }
```

```

1 void
2 Shutup_WatchDog ()
3 {
4     WDTCN = 0xde;
5     WDTCN = 0xad;
6 } //end of function Shutup_WatchDog

```

Part 2: 設定 LCD Status 時送 command 的 function

為了在 LCD 上設定輸出字元、移動指針位置以及初始化設定，我們必須送一些訊號到 LCD 使其在接收訊號的時候可以做出我們預期的對應反應，這裡需要注意的是在設定字元、設定 Enable LCD 和清除螢幕時所需要的原狀態和特定數值做 arithmetic 運算。

```

1 void
2 LCD_Init ()
3 {
4     LCD_SendCommand(0x02); // Initialize as 4-bit mode
5     LCD_SendCommand (0x28); //Display function: 2 rows for 4-bit data, small
6     LCD_SendCommand (0x0e); //display and cursor ON, cursor blink off
7     LCD_SendCommand (0x01); //clear display, cursor to home
8     //LCD_SendCommand (0x10); //cursor shift left
9     //LCD_SendCommand (0x06); //cursor increment, shift off
10 }
11
12 void
13 LCD_Status_SetRS ()
14 {
15     LCD_status = LCD_status | 1;
16 }
17
18 void
19 LCD_Status_ClearRS ()
20 {
21     LCD_status = LCD_status & 0xfe;
22 }
23
24 void
25 LCD_Status_SetWord (char word)
26 {
27     word = word & 0x0f;
28     LCD_status = LCD_status & 0x03;
29     LCD_status = LCD_status | (word<<2);
30 }
31
32 void
33 LCD_Status_SetEnable ()
34 {
35     LCD_status = LCD_status | 0x02;
36 }
37
38
39 void
40 LCD_Status_ClearEnable ()
41 {
42     LCD_status = LCD_status & 0xfd;
43 }

```

```
1 void
2 LCD_SendCommand (char cmd)
3 {
4     LCD_Status_ClearRS ();
5
6     ///send the higher half
7     LCD_Status_SetWord ((cmd>>4) & 0x0f);
8     LCD_Status_SetEnable ();
9     P3 = LCD_status;
10    LCD_Delay ();
11    LCD_Status_ClearEnable ();
12    P3 = LCD_status;
13    LCD_Delay ();
14
15    ///send the lower half
16    LCD_Status_SetWord (cmd&0x0f);
17    LCD_Status_SetEnable ();
18    P3 = LCD_status;
19    LCD_Delay ();
20    LCD_Status_ClearEnable ();
21    P3 = LCD_status;
22    LCD_Delay ();
23 }
24
25 void
26 LCD_SendData (char dat)
27 {
28     LCD_Status_SetRS ();
29
30     ///send the higher half
31     LCD_Status_SetWord ((dat>>4) & 0x0f);
32     LCD_Status_SetEnable ();
33     P3 = LCD_status;
34     LCD_Delay ();
35     LCD_Status_ClearEnable ();
36     P3 = LCD_status;
37     LCD_Delay ();
38
39     ///send the lower half
40     LCD_Status_SetWord (dat&0x0f);
41     LCD_Status_SetEnable ();
42     P3 = LCD_status;
43     LCD_Delay ();
44     LCD_Status_ClearEnable ();
45     P3 = LCD_status;
46     LCD_Delay ();
47 }
```

Part 3: 針對送 data 到 LCD 螢幕的設定

這部分因為螢幕上半部和下半部需要對資料做不一樣的 arithmetic 運算，所以需要如前一張圖的後面在 `setWord()` 的時候一個是 origin data, whereas another have to shift 4 bit。

```
1  void
2  LCD_ClearScreen ()
3  {
4      LCD_SendCommand (0x01);
5  }
6
7  void
8  LCD_CursorLeft()
9  {
10     LCD_SendCommand (0x10);
11
12 }
13
14 void
15 LCD_CursorRight()
16 {
17     LCD_SendCommand (0x14);
18
19 }
20
21 void
22 LCD_CursorTop()
23 {
24     LCD_SendCommand (0x02);
25 }
26
27 void
28 LCD_CursorDown()
29 {
30     LCD_SendCommand (0xC0);
31 }
```


Part 4: button_detect

這部分和上次實驗差不多，主要是用來偵測按鈕數入 port 上的按鈕並做 debounce 的動作，當按鈕被按下以後 count 會 count 到 100 然後才會再次偵測按鈕是否被按下，然後回傳 button state 給 main function。

```
1  int
2  button_detect ()
3  {
4      int key_hold;
5      int key_release;
6      int count;
7      int button_state;
8      do {
9          ,      key_hold = P1;
10             //button_state = key_hold;
11             button_state = P1 ;
12         } while (!key_hold);
13
14         //Stage 2: wait for key released
15         key_release = 0;
16         count = 100;
17         while (!key_release) {
18             key_hold = P1;
19             if (key_hold) {
20                 count = 10;
21             }
22             else {
23                 count--;
24                 if (count==0) {
25                     key_release = 1;
26                 }
27             }
28         } //Stage 2: wait for key released
29         return button_state;
30     } //end of function button_detect ()
```

Part 5: main function

基本上做一些必要的初始化設定。



```
1  int i;      // for loop
2  int button_push;
3  int line = 1; // 1 and 2
4  int CursorPos = 0;
5
6  char str[16] = "";
7  char str2[16] = "";
8  char strTmp[16] = "";
9
10 Shutup_WatchDog ();
11 LCD_PortConfig ();
12 LCD_Init ();
13 P1 = 0;
14 LCD_ClearScreen ();
15
16 for (i=0 ; i<16 ; i++){
17     str[i] = ' ';
18     str2[i] = ' ';
19 }
```

Part 6: 顯示字元和插入字元(basic and Bonus 2nd)

這裡拿其中一個按鈕的 `handleOnClick()` 解釋，因為其他兩個大同小異，指有 `button_push` 和輸出字元的不同。在這個 `handleOnClick()` 裡，首先會看按鈕按下的位置是不是我們在輸入在所要求的且他會不會超過螢幕範圍，若有其一不符就不做。其次在按下對應按鈕後，會去看是在哪行並將對應的字元輸入對應位置，若在兩個字元之間則並不會影響字元的差入，因為是將後面整排 `char array` 做線性移動。在對 `char array` 輸入好後，將要輸入到 LCD 的 `command and data` 呼叫對應函數來讓螢幕顯示每個 `digit` 所需要顯示的字元。最後再將指針一到下一個對應的位置。

```
1  if (button_push == 1 && CursorPos < 16){
2      if (line == 1){          // judge which line
3          for (i = 15 ; i>= CursorPos ; i--){
4              str[i] = str[i-1];
5          }
6          str[CursorPos] = 'A';
7      }
8      else{
9          for (i = 15 ; i>= CursorPos ; i--){
10             str2[i] = str2[i-1];
11         }
12         str2[CursorPos] = 'A';
13     }
14     LCD_SendData ('A');
15     CursorPos += 1;
16     for (i = CursorPos ; i<16 ; i++){
17         if (line == 1)
18             LCD_SendData(str[i]);
19         else
20             LCD_SendData(str2[i]);
21     }
22     for (i = 15 ; i >= CursorPos ; i--){
23         LCD_CursorLeft();
24     }
25 }
```

Part 6: 捲動螢幕和換行 (Bonus 1st)

這邊判斷按鈕按下位置後並判斷在哪行，如果是在第一行就讓他做基本的指針下移，若在第一行就先將第二個 char array 的值 map 到第一個 char array 並將第二個 char array 清空。接著清空 LCD 螢幕並對對應的 digit 做設定 map 後的字元陣列，最後設定指針到第二行的初始位置。

```
1  else if (button_push == 8){
2      /*
3          Implement the 'new-line key
4          Change to the next line if a new-line button is pressed at Line 1
5          Scroll the screen if a new-line button is pressed at Line 2
6      */
7      if(line == 1){
8          LCD_CursorDown();
9          for (i = 0 ; i < CursorPos ; i++){
10             LCD_CursorRight();
11         }
12         line = 2;
13         LCD_SendCommand (0xC0);
14     }
15     else{
16         // line 2 scroll down, replace the first line with the second line and clear the second line
17         for (i = 0 ; i < 16 ; i++){
18             str[i] = str2[i];
19             str2[i] = ' ';
20         }
21         LCD_ClearScreen();
22         CursorPos = 0;
23         LCD_CursorTop();
24         for (i = 0 ; i < 16 ; i++){
25             LCD_SendData(str[i]);
26         }
27         LCD_CursorDown();
28         for (i = 0 ; i < 16 ; i++){
29             LCD_SendData(str2[i]);
30         }
31         LCD_SendCommand (0xC0);
32     }
33 }
```


Part 6: 指針移動

基本的指針在螢幕上移動，左右移動對 `CursorPos` 做 LCD 設定後將其做加減法移動在陣列上的位置，上下移動則是要送訊號到設定在哪一行並移動到對應的行數並且移動指針到原本行數的位置上。

```
1  else if (button_push == 16){
2      if (CursorPos<16){
3          LCD_CursorRight();
4          CursorPos += 1;
5      }
6  }
7  else if (button_push == 32){
8      if (CursorPos>-1){
9          LCD_CursorLeft();
10         CursorPos -= 1;
11     }
12 }
13 else if (button_push == 64){
14     if(line == 2){
15         LCD_CursorTop();
16         for (i = 0 ; i < CursorPos ; i++){
17             LCD_CursorRight();
18         }
19         line = 1;
20     }
21 }
22 else if (button_push == 128){
23     if(line == 1){
24         LCD_CursorDown();
25         for (i = 0 ; i < CursorPos ; i++){
26             LCD_CursorRight();
27         }
28         line = 2;
29     }
30 }
```

Difficulties you've encountered and your solution:

這部分遇到的最大問題就是如何針對 LCD 所需要做的 **operation** 送出對應的訊號，以及在捲動螢幕的時後要怎麼讓 **map** 後的字元映射到對應的 **digit** 上。而針對第一個問題我做的方式就是先了解老師給的 **data sheet** 上 LCD 的指令以及對應會做的事。第二個問題則是利用清空並將 **map** 後的字元一個一個輸入上去，在移動指針位置來達到我們的需求。

Discussion:

在這個實驗內我學到了如何在 **8051 CPU** 做 LCD 螢幕的顯示和訊號控制，並且運用了特定的想法實作了一些很酷的功能，例如：轉動螢幕和在字元之間差入其他字元...等。