作業五：　LCM 控制實驗 – 文字走馬燈

資工三乙 406262199 陳奕帆
資工三乙 406262216 劉品萱

繳交日期：_____

(1) 問題

Q1 : 請說明 Sitronix ST7066U 晶片之功能？詳細解釋輸入腳位 RS、E、R/W 與 DB0 ~ DB7 之意義(作用)為何?

Sitronix ST7066U 屬於液晶顯示器，控制 LSI 控制器和驅動器，可以由 4 位或 8 位點陣的微處理器配置

DB0~DB6 ➔ 用來存放讀進來共 7 個位元的 Address Counter 的內容

DB7 ➔ 存放讀進的 Busy Flag

E ➔ Starts data read/write

RS ➔ 0 為選擇 Instruction register，1 為選擇 Data register

R/W ➔ 0 用來 write　1 用來 read

Q2：同上，使用 ADP-WT58F2C9 實驗板上哪個 GPIO port 對應 RS、E、R/W 與 DB0 ~ DB7 腳位？此 GPIO port 必須規劃為輸入或輸出? 必須寫出各輸入腳位對應到此 port 之位置(如 bit_0, bit_1, ...)。

Port E 對應 RS、E、R/W 與 DB0 ~ DB7 腳位

若 R/W 為 read 的話規劃為 input，為 write 的話規劃為 output

RS ➔ Port_E_10 ；DB0 ~ DB7 ➔ Port_E_0~Port_E_7

R/W ➔ Port_E_8 ；E ➔ Port_E_9

Q3：請仔細說明 Sitronix ST7066U 晶片之 4 種指令分類。提示：與腳位 RS、與 R/W 有關。

RS	R/W	
0	0	選擇 Instruction Register 進行 write (例如 display, clear...)
0	1	選擇 Instruction Register 進行 read busy flag (DB7) 和 address counter (DB0 to DB6)
1	0	選擇 Data Register 進行 write (DDRAM / CGRAM)
1	1	選擇 Data Register 進行 read (DDRAM / CGRAM)

Q4：請解釋 AC(Address Counter) content 與 cursor (游標)之關係。

Address Counter 用來讀寫特定的記憶體，即讀取的位置存放於 Address Counter 裡面，也就是 cursor 的位置。

Q5：請說明 DDRAM 與 CGRAM 之用途為何？

2-line display mode 時，display position (顯示位置) 與 DDRAM address (位址)之關係為何?

Display shift (位移)時，display position (顯示位置) 與 DDRAM address (位址)之關係會如何改變?

DDRAM ➔ 用來儲存 8-bit 字元的顯示資料

CGRAM ➔ 提供使用者自行造字之功能，重新定義字元形式

2-line display mode 上排為 display position/下排為 DDRAM address(10 進位)

1	2	3	4	5	. . .	39	40

00	01	02	03	04	. . .	26	27
40	41	42	43	44	. . .	66	67
SHIFT LEFT							
01	02	03	04	05		27	00
41	42	43	44	45		67	40
SHIFT RIGHT							
27	00	01	02	03		25	26
67	40	41	42	43		65	66

Q6:請詳細解釋 Entry Mode Set 指令之作用,包含各個 control flag 說明。

Entry Mode Set 用來設置 cursor 的位置及指定 display 的移位。

其包含的 Control flag 為

Control flag I/D ➔ cursor 每一次的位移，I/D = 1 時表示遞增，因此視覺上看起來 cursor 會往右邊移動,而 I/D = 0 時為遞減,因此視覺上看起來 cursor 會往左邊移動。

Control flag S ➔ 設定是否要伴隨 diplay 的移位。S = 1 時表示伴隨移位，S = 0 時則不伴隨移位

Q7：請詳細解釋 Function Set 指令之作用，包含各個 control flag 說明。

Function Set 設置 DB5 = 1。其中包括多個 Control flage 如下，

DL ➔ 用來設置資料長度(Data Length)，D = 1 時為 8bit，D = 0 時為 4bit

N ➔ 可選擇是一行或兩行模式，N = 1 時為兩行模式，N = 0 為一行模式。

F ➔ 可以設置點字元的字體，F = 1 時為 5*10 dot character font。F = 0

時為 5*8 dot character font

Q8:請詳細解釋 Display on/off Control 指令之作用,包含各個 control flag 說明。

Display on/off Control 指令能夠對 LCM 模組做字顯示、cursor 顯示、blink 顯示的設定。

Control flag D ➔ 設定 LCM 上的字要不要顯示。D=1 時 LCM 上不顯示字，D=0 時 LCM 可以顯示字。

Control flag C ➔ 設定 cursor 要不要顯示。C=1 的時候 cursor 不顯示，C=0 的時候 cursor 顯示。

Control flag B ➔ 設定 cursor 的位置有無閃爍效果。B=1 時有閃爍效果，B=0 時則無閃爍效果。

Q9：請詳細解釋 Display or Cursor Shift 指令之作用，包含各個 control flag 說明。

Display or Cursor Shift 的指令作用為在不更改 DDRAM 內容的情況下移動光標並切換顯示，其包含的 Control Flag 為：

Control Flag S/C ➔ 用來選擇要偏移 display 或移動 cursor。S/C = 1 時為偏移 display，而 S/C = 0 時則選擇移動 cursor。

Control Flag R/L ➔ 用來決定 shift 的方向，R/L = 1 時往右邊 shift，R/L = 0 時往左邊 shift。

Q10：請詳細解釋 Set DDRAM Address 指令之作用。

DDRAM 用來寫入要顯示在 LCM 模組上的字，也可以將 LCM 模組上的字讀入

Set DDRAM Address 會將 DDRAM 的二進位地址設置進 address counter 中。接著會將數據寫入至 DDRAM 的 MPU 中或從其中讀取數據。

Q12：請詳細解釋 Read Busy Flag& Address 指令之作用，包含 BF (Busy Flag)說明。

Busy Flag 可以用來判斷前一個指令是否還處於運作過程當中，若前一個還沒指令還沒執行好，Read Busy Flag 就會讀到 1，反之若前一個指令若已經

完成，Read Busy Flag 就會讀到 0，變成 0 之後才能寫下一個指令。

(2) 印出 LCM.h& C 程式碼

因為有利用呼叫 function 的寫法，所以也把 function 附上。

```
void SetCursor(unsigned short Line, unsigned short Position)
{
    unsigned short Address = Line * 0x0040 + Position * 0x0001;
    WriteIns(0x80 | Address);
}

void Display_First_Line()
{
  //char ABCD[]={'A','N','D','E','S'};
  char Line1[]="AbCdEfGhIjKlMnOp";
  char Line2[]="aBcDeFgHiJKimNo";
  char i;
  WriteIns(0x38);   //FUNCTION SET
  WriteIns(0x0E);   //DISPLAY CONTROL
  WriteIns(0x06);   //SET INPUT MODE

  SetCursor(0,0);   //1-LINE DD RAM SET Address
  for(i=0;i<16;i++)
  {
      WriteData(Line1[i]);
      delay1(200);
  }
  SetCursor(1,0);
  for(i=0;i<15;i++)
  {
      WriteData(Line2[i]);
      delay1(200);
  }

```

```
}

void Display_Second_Line()
{

    char Line1[]="123456789101112!";
      char Line2[]="211101987654321";
      char i;
      WriteIns(0x38);   //FUNCTION SET
      WriteIns(0x0E);   //DISPLAY CONTROL
      WriteIns(0x06);   //SET INPUT MODE

      SetCursor(0,0);   //1-LINE DD RAM SET Address
      for(i=0;i<16;i++)
      {
          WriteData(Line1[i]);
          delay1(200);
      }
      SetCursor(1,0);
      for(i=0;i<15;i++)
      {
          WriteData(Line2[i]);
          delay1(200);
      }
}

void Close_Cursor()
{
    if(!Close_Cursor_Flag)      //關閉游標
        WriteIns(0x0C);
    else                        //顯示游標
        WriteIns(0x0E);
    Close_Cursor_Flag = ~(Close_Cursor_Flag);
```

```
63  }
64
65  void Clear_Display()
66  {
67      WriteIns(0x01);
68  }
69
70
71  int main()
72  {
73
74      unsigned int First_Display_Flag = 0;
75      unsigned int Second_Display_Flag = 0;
76
77      InitialLCD();
78
79      unsigned int tmp = 0;
80      unsigned int Button_One_Click = 0;
81
82
83      OS_PowerOnDriverInitial();
84
85
86      DRV_Printf("==================================
    =========\r\n", 0);
87      DRV_Printf("     ADP-WT58F2C9 Key Matrix demo program
    \r\n", 0);
88      DRV_Printf("==================================
    =========\r\n", 0);
89
90
91      DRV_Printf("Key Matrix testing...\r\n", 0);
92      DRV_Printf("Press SW17 then SW16 to EXIT.\r\n", 0);
```

```
93
94          GPIO_PTA_FS = 0x0000;
95          GPIO_PTA_PADINSEL = 0x0000;
96
97          // Setting for 7LED select
98          GPIO_PTA_DIR = 0x0000;
99          GPIO_PTA_CFG = 0x0000;
100         GPIO_PTA_GPIO = Digit_8;
101         // Setting for 7LED number
102         GPIO_PTD_DIR = 0x0000;
103         GPIO_PTD_CFG = 0x0000;
104         GPIO_PTD_GPIO = Number_8 | Number_Dot;
105
106         unsigned int col;
107         unsigned int check = 0;
108         unsigned int key;
109         unsigned int index_7LED_NUM[17] = {Number_0, Number_1,
    Number_2, Number_3, Number_4, Number_5, Number_6, Number_7,
110                 Number_8, Number_9, Number_A, Number_b,
    Number_C, Number_d, Number_E, Number_F, Number_Dot, };
111         unsigned int ready_exit = 0;
112
113
114         unsigned int clickShiftButton = 0;
115         unsigned int LeftShift = 0;
116         unsigned int RightShift = 0;
117         while(1)
118         {
119
120             key = 0xFF;
121             GPIO_PTA_DIR = 0x0FF0;
122             GPIO_PTA_CFG = 0x0000;
123             for (col=0; col<4; col++)
```

```
124             {
125                 GPIO_PTA_BS = 0x000F;
126                 GPIO_PTA_BR = 0x0000 | (1 << col);
127                 tmp = ((~GPIO_PTA_PADIN) & 0xFF0) >> 4;
128                 if (tmp > 0)
129                 {
130                     if (tmp & 0x1)
131                         key = 0*4 + col;
132                     else if (tmp & 0x2)
133                         key = 1*4 + col;
134                     else if (tmp & 0x4)
135                         key = 2*4 + col;
136                     else if (tmp & 0x8)
137                         key = 3*4 + col;
138                     else if (tmp & 0x80)
139                         GPIO_PTA_GPIO = Digit_8;
140                     else if (tmp & 0x40)
141                         GPIO_PTA_GPIO = Digit_7;
142                     else if (tmp & 0x20)
143                         GPIO_PTA_GPIO = Digit_6;
144                     else if (tmp & 0x10)
145                         GPIO_PTA_GPIO = Digit_5;
146                     break;
147                 }
148             }
149             if (key != 0xFF)
150             {
151                 if(!check)
152                 {
153                     GPIO_PTD_GPIO = index_7LED_NUM[key+1];
154 
155                     if(key == 0)
156                     {
```

```
157                         //顯示第一組文字
158                         First_Display_Flag = 1;
159                         Second_Display_Flag = 0;
160                         Display_First_Line();
161                         clickShiftButton = 0;
162                     }
163                     if(key == 1)
164                     {
165                         //顯示第二組文字
166                         First_Display_Flag = 0;
167                         Second_Display_Flag = 1;
168                         Display_Second_Line();
169                         clickShiftButton = 0;
170                     }
171                     if(key == 2)
172                     {
173                         //文字左移
174                         WriteIns(0x18);
175                         delay1(3000);
176                         clickShiftButton = 1;
177                         LeftShift = 1;
178                         RightShift = 0;
179 
180                     }
181                     if(key == 3)
182                     {
183                         //文字右移
184                         WriteIns(0x1C);
185                         delay1(3000);
186                         clickShiftButton = 1;
187                         LeftShift = 0;
188                         RightShift = 1;
189                     }
```

```
190                     if(key == 4)
191                     {
192                         //Clesr display
193                         Clear_Display();
194                         clickShiftButton = 0;
195                     }
196                     if(key == 5)
197                     {
198                         //Return home
199                         WriteIns(0x02);
200                         delay1(300000);
201                         clickShiftButton = 0;
202                     }
203                     if(key == 6)
204                     {
205                         //Cursor on/off
206                         Close_Cursor();
207                         clickShiftButton = 0;
208                     }
209                 }
210                 check = 1;
211             }
212             else
213             {
214                 //當按了 shift 鍵(2 或 3)  但還沒按下一個鍵
215                 //仍持續做 shift 的動作
216                 if(clickShiftButton == 1)
217                 {
218                     //判斷持續做左旋還是右旋
219                     if(LeftShift == 1 && RightShift == 0)
220                     {
221                         WriteIns(0x18);
222                         delay1(300000);
```

```
223                         clickShiftButton = 1;
224                     }
225                     if(LeftShift == 0 && RightShift == 1)
226                     {
227                         WriteIns(0x1C);
228                         delay1(300000);
229                         clickShiftButton = 1;
230                     }
231                 }
232                 check = 0;
233             }
234 
235         }
236 
237         GPIO_PTA_GPIO = Digit_1;
238         GPIO_PTD_GPIO = Number_8 | Number_Dot;
239 
240         DRV_Printf("==================================
    =========\r\n", 0);
241         DRV_Printf("==================================
    ===\r\n", 0);
242 
243 
244         return 0;
245 }
```