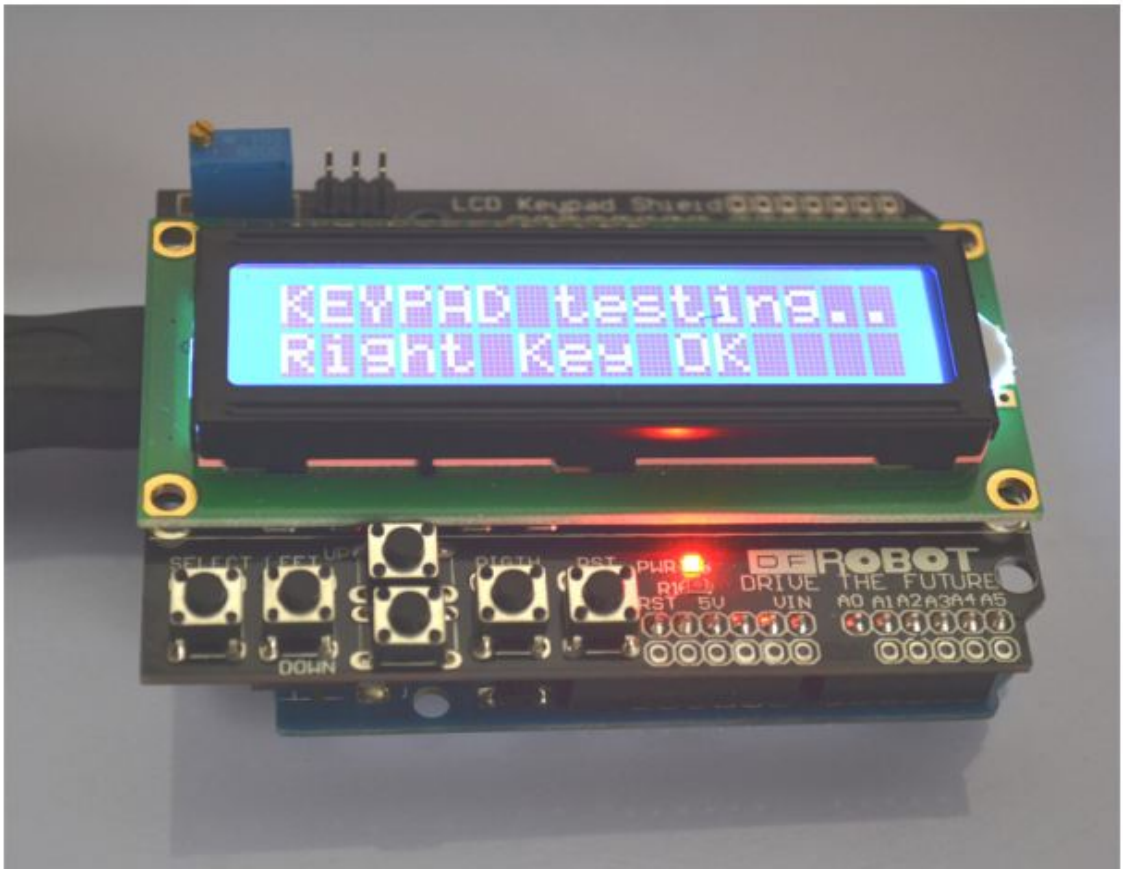
Arduino 1602 液晶顯示器



一. 引腳功能說明：

編號	符號	引腳說明	編號	符號	引腳說明
1	VSS	電源地	9	D2	數據
2	VDD	電源正極	10	D3	數據
3	VL	液晶顯示偏壓	11	D4	數據
4	RS	數據/命令選擇	12	D5	數據
5	R/W	讀/寫選擇	13	D6	數據
6	E	使能信號	14	D7	數據
7	D0	數據	15	BLA	背光源正極
8	D1	數據	16	BLK	背光源負極

編號符號引腳說明編號符號引腳說明
1. VSS 電源地9 D2 數據
2. VDD 電源正極10 D3 數據
3. VL 液晶顯示偏壓11 D4 數據
4. RS 數據/命令選擇12 D5 數據
5. R/W 讀/寫選擇13 D6 數據
6. E 使能信號14 D7 數據
7. D0 數據15 BLA 背光源正極
8. D1 數據16 BLK 背光源負極

第1 腳：VSS 爲地電源。
第2 腳：VDD 接5V 正電源。
第3 腳：VL 爲液晶顯示器對比度調整端，接正電源時對比度最弱，接地時對比度最高，對比度過高時會產生“鬼影”，使用時可以通過一個10K 的電位器調整對比度。
第4 腳：RS 爲寄存器選擇，高電平時選擇數據寄存器、低電平時選擇指令寄存器。
第5 腳：R/W 爲讀寫信號線，高電平時進行讀操作，低電平時進行寫操作。當RS和R/W 共同爲低電平時可以寫入指令或者顯示地址，當RS 爲低電平R/W 爲高電平時可以讀忙信號，當RS 爲高電平R/W 爲低電平時可以寫入數據。
第6 腳：E 端爲使能端，當E 端由高電平跳變成低電平時，液晶模塊執行命令。
第7～14 腳：D0～D7 爲8 位雙向數據線。
第15 腳：背光源正極。第16 腳：背光源負極。

二.1602LCD的指令說明：

1602 液晶模塊內部的控制器共有11 條控制指令，如表下表所示：

序號	指令	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
1	清屏	0	0	0	0	0	0	0	0	0	1
2	光標返回	0	0	0	0	0	0	0	0	1	*
3	置輸入模式	0	0	0	0	0	0	0	1	I/D	S
4	顯示開/關控制	0	0	0	0	0	0	1	D	C	B
5	光標或字符移位	0	0	0	0	0	1	S/C	R/L	*	*
6	置功能	0	0	0	0	1	DL	N	F	*	*
7	置字符發生儲存器地址	0	0	0	1	字符發生儲存器位置					
8	置數據儲存器地址	0	0	1	顯示數據儲存器位置						
9	讀忙標誌或地址	0	1	BF	計數器地址						
10	寫數到CGRAM或DDRAM	1	0	要寫的數據內容							
11	從CGRAM或DDRAM讀數	1	1	讀出的數據內容							

序號指令RS R/W D7 D6 D5 D4 D3 D2 D1 D0
1. 清屏0 0 0 0 0 0 0 0 0 1
2. 光標返回0 0 0 0 0 0 0 0 1 *
3. 置輸入模式0 0 0 0 0 0 0 1 I/DS
4. 顯示開/關控制0 0 0 0 0 0 1 DC B
5. 光標或字符移位0 0 0 0 0 1 S/CR/L * *

6. 置功能0 0 0 0 1 DL NF * *
7. 置字符發生存貯器地址0 0 0 1 字符發生存貯器地址
8. 置數據存貯器地址0 0 1 顯示數據存貯器地址
9. 讀忙標誌或地址0 1 BF 計數器地址
10. 寫數到CGRAM 或DDRAM）1 0 要寫的數據內容
11. 從CGRAM 或DDRAM 讀數 1 1 讀出的數據內容

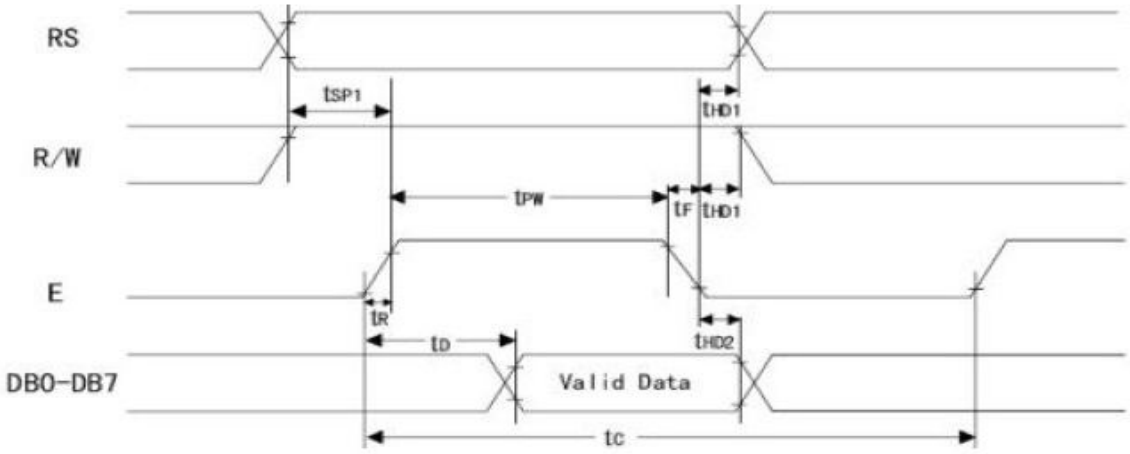
1602 液晶模塊的讀寫操作、屏幕和光標的操作都是通過指令編程來實現的。（說明：1 爲高電平、0 爲低電平）

指令1：清顯示，指令碼01H,光標復位到地址00H 位置。
指令2：光標復位，光標返回到地址00H。
指令3：光標和顯示模式設置。I/D：光標移動方向，高電平右移，低電平左移。S:屏幕上所有文字是否左移或者右移。高電平表示有效，低電平則無效。
指令4：顯示開關控制。 D：控制整體顯示的開與關，高電平表示開顯示，低電平表示關顯示。C：控制光標的開與關，高電平表示有光標，低電平表示無光標。B：控制光標是否閃爍，高電平閃爍，低電平不閃爍。
指令5：光標或顯示移位。S/C：高電平時移動顯示的文字，低電平時移動光標。
指令6：功能設置命令。DL：高電平時爲4 位總線，低電平時爲8 位總線。N：低電平時爲單行顯示，高電平時雙行顯示。F: 低電平時顯示5x7 的點陣字符，高電平時顯示5x10 的點陣字符。
指令7：字符發生器RAM 地址設置。
指令8：DDRAM 地址設置。
指令9：讀忙信號和光標地址。BF：爲忙標誌位，高電平表示忙，此時模塊不能接收命令或者數據，如果爲低電平表示不忙。
指令10：寫數據。
指令11：讀數據。

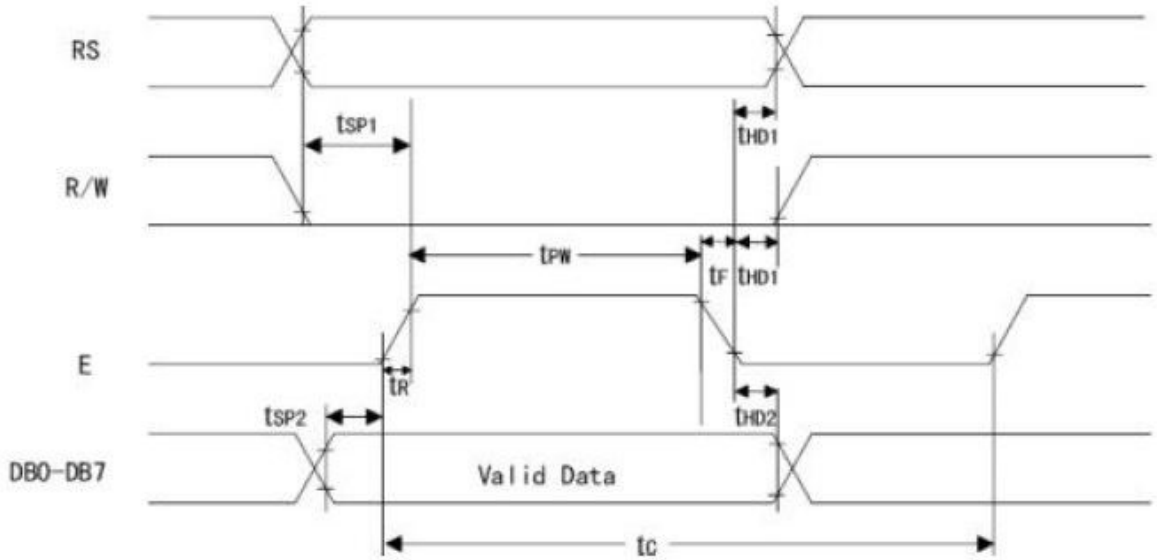
讀狀態	輸入	RS=L，R/W=H，E=H	輸出	D0—D7=狀態字
寫指令	輸入	RS=L，R/W=L，D0—D7=指令碼,高脈衝	輸出	無
讀數據	輸入	RS=H，R/W=H，E=H	輸出	D0—D7=數據
寫數據	輸入	RS=H，R/W=L，D0—D7=數據，E=高脈衝	輸出	無

與HD44780 相兼容的芯片時序表如下：
讀狀態輸入RS=L，R/W=H，E=H 輸出D0—D7=狀態字
寫指令輸入RS=L，R/W=L，D0—D7=指令碼，E=高脈衝輸出無
讀數據輸入RS=H，R/W=H，E=H 輸出D0—D7=數據
寫數據輸入RS=H，R/W=L，D0—D7=數據，E=高脈衝輸出無

讀操作時序圖



寫操作時序圖



1602LCD的一般初始化（復位）過程：

延時15mS
寫指令38H（不檢測忙信號）
延時5mS。寫指令38H（不檢測忙信號）
延時5mS。寫指令38H（不檢測忙信號）
以後每次寫指令、讀/寫數據操作均需要檢測忙信號
寫指令38H：顯示模式設置
寫指令08H：顯示關閉
寫指令01H：顯示清屏
寫指令06H：顯示光標移動設置
寫指令0CH：顯示開及光標設置
顯示KEYPAD testing... pressing

三.Arduino官方例程：

```
 //example use of LCD4Bit_mod library
 #include <LCD4Bit_mod.h>
 //create object to control an LCD.
 //number of lines in display=1
 LCD4Bit_mod lcd = LCD4Bit_mod(2);
//Key message
char msgs[5][15] = {"Right Key OK ",
                    "Up Key OK ",
                    "Down Key OK ",
                    "Left Key OK ",
                    "Select Key OK" };
int adc_key_val[5] ={30, 150, 360, 535, 760 };
int NUM_KEYS = 5;
int adc_key_in;
int key=-1;
int oldkey=-1;

void setup() {
  pinMode(13, OUTPUT); //we'll use the debug LED to output a heartbeat
  lcd.init();
  //optionally, now set up our application-specific display settings, overriding whatever the lcd did
in lcd.init()
  //lcd.commandWrite(0x0F);//cursor on, display on, blink on. (nasty!)
   lcd.clear();
  lcd.printIn("KEYPAD testing... pressing");
  }
  void loop() {
        adc_key_in = analogRead(0); // read the value from the sensor
   digitalWrite(13, HIGH);
key = get_key(adc_key_in); // convert into key press
        if (key != oldkey) // if keypress is detected
        {
  delay(50); // wait for debounce time
             adc_key_in = analogRead(0); // read the value from the sensor
  key = get_key(adc_key_in); // convert into key press
  if (key != oldkey)
  {
    oldkey = key;
    i f (key >=0){
      lcd.cursorTo(2, 0); //line=2, x=0
                      lcd.printIn(msgs[key]);
     }
    }
 }

 //delay(1000);
 digitalWrite(13, LOW);
}
```

```
// Convert ADC value to key number
int get_key(unsigned int input)
{
       int k;

       for (k = 0; k < NUM_KEYS; k++)
       {
            if (input < adc_key_val[k])
            {
     return k;
          }
          }
     if (k >= NUM_KEYS)
        k = -1; // No valid key pressed

return k;
}
```