COMP2121: Microprocessors and Interfacing

I/O Devices (II)

http://www.cse.unsw.edu.au/~cs2121

Lecturer: Hui Wu

Term 2, 2019

1

Overview

- Keyboard
- LCD (Liquid Crystal Display)

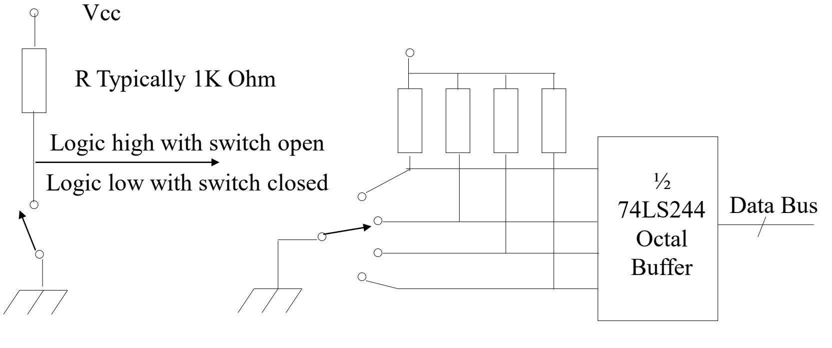
2

2

Input Switches (1/2)

- Most basic of all binary input devices.
- The switch output is high or low depends on the the switch position.
- Pull-up resistors are necessary in each switch to provide a high logic level when the switch is open.
- Problem with switches:
  - ❑ Switch bounce.
    - ❖ When a switch makes contact, its mechanical springiness will cause the contact to bounce, or make and break, for a few millisecond (typically 5 to 10 ms).

Input Switches (2/2)



(a) Single-pole, single-throw (SPST) logic switch

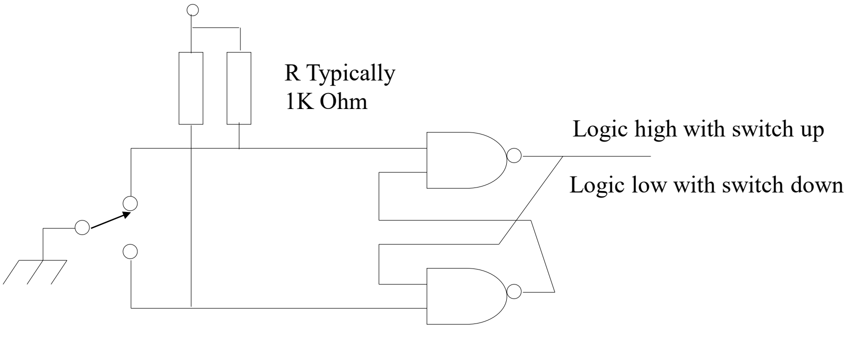
(b) Multiple pole switch.

Software Debouncing

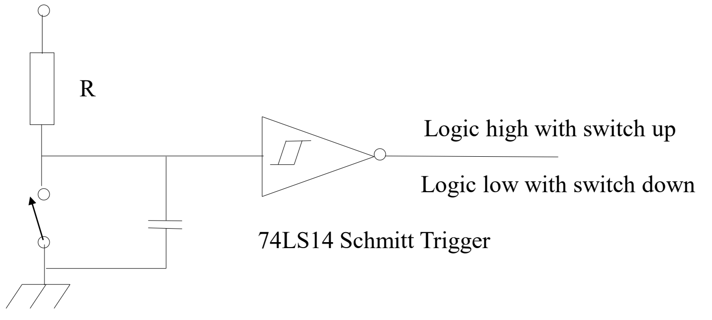
Two software debouncing approach:

- Wait and see:
  - ❑ If the software detects a low logic level, indicating that switch has closed, it simply waits for a short period, say 20 to 100ms, and then test if the switch is still low.
- Counter-based approach:
  - ❑ Initialize a counter to 10.
  - ❑ Poll the switch every millisecond until the counter is either 0 or 20. If the switch output is low, decrement the counter; otherwise, increment the counter.
  - ❑ If the counter is 0, we know that switch output has been low for at least 10 ms. If, on the other hand, the counter reaches 20, we know that the switch has been closed for at least 10 ms.

NAND Latch Debouncer



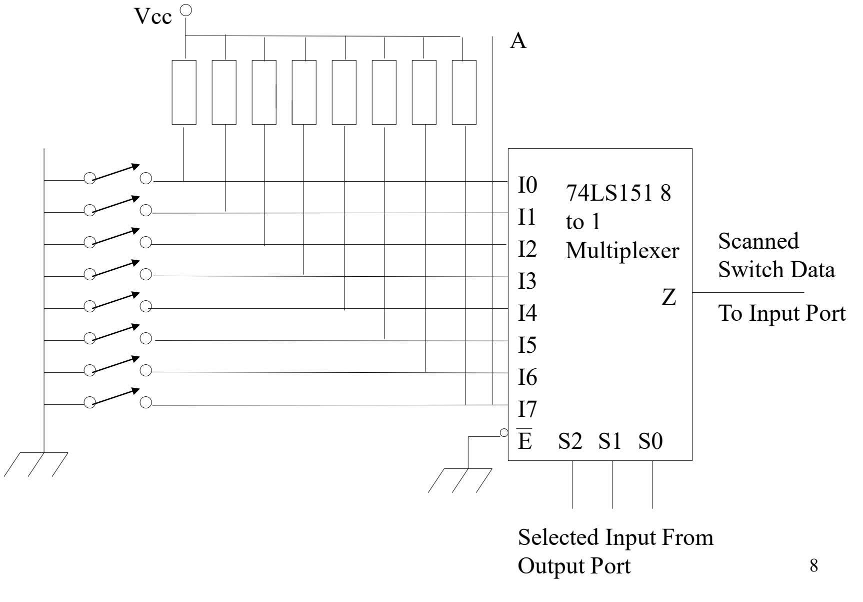
Integrating Debouncer with Schmitt Trigger



7

7

One-Dimensional Array of Switches



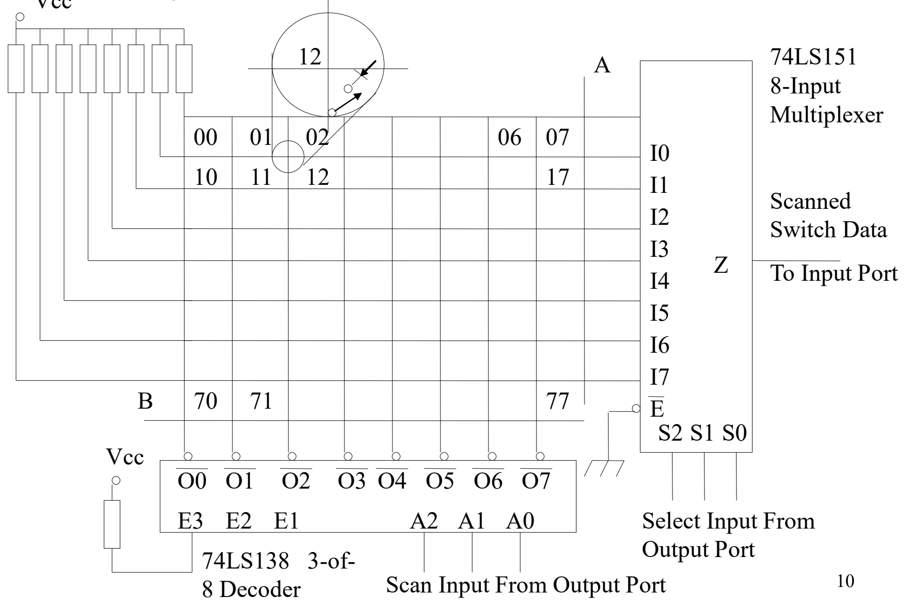
8

8

One-Dimensional Array of Switches

- Switch bounce problem must be solved.
- The array of switches must be scanned to find out which switches are closed or open.
  - ❑ Software is required to scan the array. As the software outputs a 3-bits sequence from 000 to 111, the multiplexer selects each of the switch inputs. The software scanner then read one bit at an input port.
- The output of switch array could be interfaced directly to an eight-bit port at point A.
- To save I/O lines, a 74LS1518 Input Multiplexer can be used.

Keyboard Matrix of Switches (1/3)



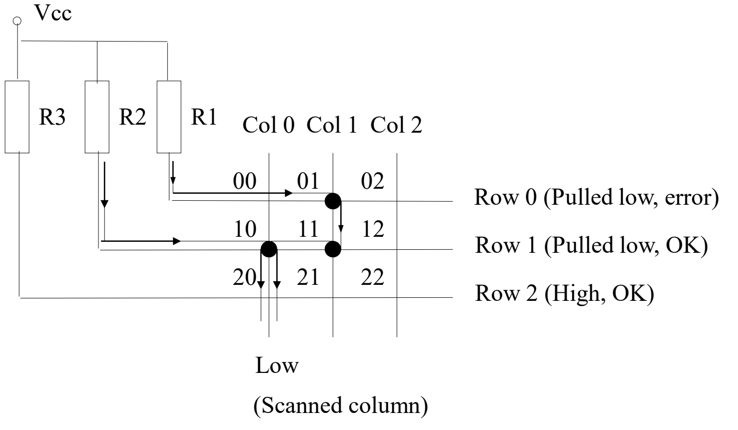
Keyboard Matrix of Switches (2/3)

- A keyboard is an array of switches arranged in a two-dimensional matrix.
- A switch is connected at each intersection of vertical and horizontal lines.
- Closing the switch connects the horizontal line to the vertical line.
- 8*8 keyboard can be interfaced directly into 8-bit output and input ports at point A and B.
- Some input and output lines can be saved by using a 74LS138 3-of-8 decoder and a 74LS1518 Input Multiplexer.

Keyboard Matrix of Switches (3/3)

- Software can scan the key board by outputting a three-bit code to 74LS138 and then scanning the 74LS151 multiplexer to find the closed switch.
  - The combination of the two 3-bit scan codes identifies which switch is closed. For example, the code 000000 scan switch 00 in the upper left-hand corner.
- The diode prevents a problem called ghosting.

Ghosting (1/2)



Ghosting (2/2)

- Ghosting occurs when several keys are pushed at once.
- Consider the case shown in the figure where three switches 01, 10 and 11 are all closed. Column 0 is selected with a logic low and assume that the circuit does not contain the diodes. As the rows are scanned, a low is sensed on Row 1, which is acceptable because switch 10 is closed. In addition, Row 0 is seen to be low, indicating switch 00 is closed, which is NOT true. The diodes in the switches eliminate this problem by preventing current flow from R1 through switches 01 and 11. Thus Row 0 will not be low when it is scanned.

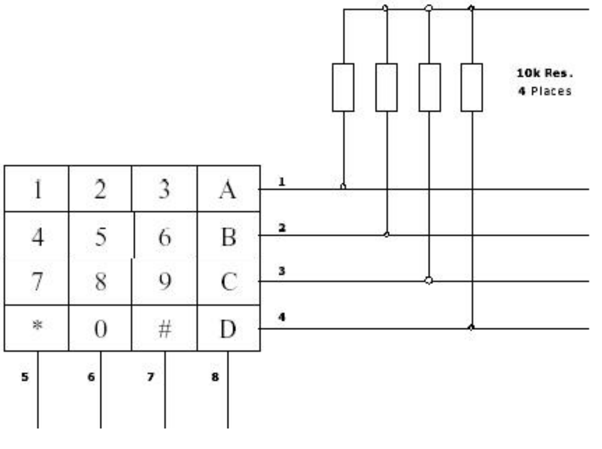
N-Key Rollover

- The problem with a typist hitting more than one key at once, or rapidly rolling the finger from one key to another, is called n-key rollover.
- Solutions:
  - ❑ Store the rapidly pressed keys in first in , first out (FIFO) buffer for later readout. Or
  - ❑ Use n-key lockout, where only the first or last of the sequence of keys pressed is recorded.

15

15

Key Scanning (1/3)

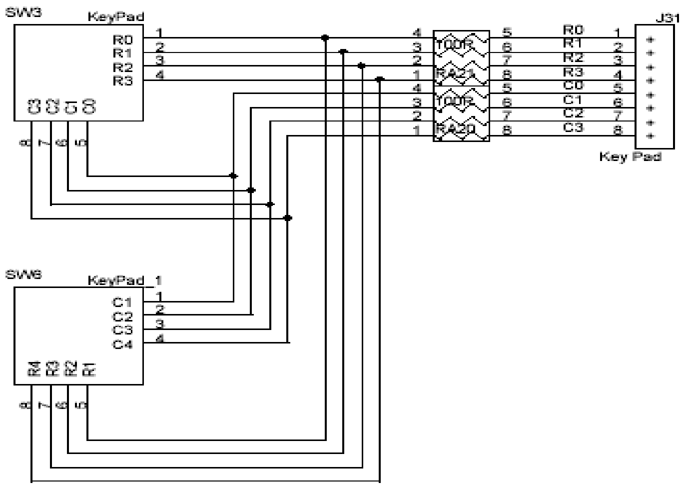


Keypad Layout

16

16

Key Scanning (2/3)



Keypad Schematic

17

17

Key Scanning (3/3)

The following procedure describes how the software knows which key was pressed:

1. Connect the columns to output pins, and the rows to input pins.
2. Sequentially drive each column to a low voltage (write zero to the corresponding output pin) and sample each row (read from the corresponding input pin) at the same instance.
   - ❑ Since the rows are all pulled high with internal pull-up resistors, all four inputs will normally be high. If a key is pressed in a column which is at a low level, that low level will be conducted to the input pin through the closed key and the corresponding row will be sensed as a low.

18

18

Sample Code for Key Scanning (1/10)

IMPORTANT NOTICE:

- The labels on PORTL are reversed, i.e., PLi is actually PL7-i (i=0, 1, …, 7).

Board settings:

- Connect the four columns C0~C3 of the keypad to PL3~PL0 of PORTL and the four rows R0~R3 to PL7~PL4 of PORTL.
- Connect LED0~LED7 of LEDs to PC0~PC7 of PORTC.

Output of the sample code:

- Each key has a code of one byte.
  - ❑ For a number key (0~9, A, B, C, D), its code is itself. For example, the code of A is 0xA (10 in decimal).
  - ❑ The code of * is 0xE and the code of # is 0xF.
- When a key is pressed, the key scanning program stores its code in the register r16 and displays its binary value on LEDs.

Sample Code for Key Scanning (2/10)

The pseudo code of the key scanning procedure:

```
make PL0~PL3 of PORTL all outputs;
make bits PL4~PL7all inputs;
make PC0~PC7 all outputs;
Turn all LEDs on;
 for (col=0; col<=3; col++)
   { write 0 to PLi where i=col;
      for (row=0; row<=3; row++)
        { read PL4~PL7;
          if ( the value of PL4~PL7 != 0xF ) /* one key was pressed */
              convert() ; /* convert() computes the code for the key pressed and
write the binary value of the code on LEDs */ }
   }
```

Sample Code for Key Scanning (3/10)

```
.include "m2560def.inc"
.def temp =r16
.def row =r17
.def col =r18
.def mask =r19
.def temp2 =r20
.equ PORTLDIR = 0xF0
.equ INITCOLMASK = 0xEF
.equ INITROWMASK = 0x01
.equ ROWMASK = 0x0F
```

21

21

Sample Code for Key Scanning (4/10)

```
.cseg
jmp RESET
RESET:
ldi temp, low(RAMEND)
out SPL, temp
ldi temp, high(RAMEND)
out SPH, temp
ldi temp, PORTLDIR        ; columns are outputs, rows are inputs
sts DDRL, temp
ser temp
out DDRC, temp            ; Make PORTC all outputs
out PORTC, temp           ; Turn on all the LEDs
```

22

22

Sample Code for Key Scanning (5/10)

```
main:   ; main keeps scanning the keypad to find which key is pressed.
ldi mask, INITCOLMASK    ; initial column mask
clr col ; initial column
colloop:
sts PORTL, mask      ; set column to mask value (sets column 0 off)
ldi temp, 0xFF          ; implement a delay so the hardware can stabilize
delay:
dec temp
brne delay
lds temp, PINL          ; read PORTL
```


23

Sample Code for Key Scanning (6/10)

```
andi temp, ROWMASK      ; read only the row bits
cpi temp, 0xF                  ; check if any rows are grounded
breq nextcol                   ; if not go to the next column
ldi mask, INITROWMASK   ; initialise row check
clr row                        ; initial row
rowloop:
mov temp2, temp
and temp2, mask               ; check masked bit
brne skipconv     ; if the result is non-zero, we need to look at next row
rcall convert                 ; if bit is clear, convert the bitcode
jmp main                      ; and start again
```


24

Sample Code for Key Scanning (7/10)

```
skipconv:
inc row          ; else move to the next row
lsl mask         ; shift the mask to the next bit
jmp rowloop
nextcol:
cpi col, 3       ; check if we're on the last column
breq main        ; if so, no buttons were pushed,  so start again.
sec              ; else shift the column mask:
rol mask         ; We must set the carry bit and then rotate left by a bit,
                 ; shifting the carry into bit zero.
```

25

25

Sample Code for Key Scanning (8/10)

```
inc col          ; increment column value
jmp colloop      ; check the next column.
convert:         ; convert function converts the row and column given to a
                 ; binary number and also outputs the value to PORTC.
                 ; inputs come from registers row and col and output is in  temp.
cpi col, 3       ; if column is 3 we have a letter
breq letters
cpi row, 3       ; if row is 3 we have a symbol or 0
breq symbols
mov temp, row    ; otherwise we have a number (1-9)
```

26

26

Sample Code for Key Scanning (9/10)

```
lsl temp            ; temp = row * 2
add temp, row       ; temp = row * 3
add temp, col       ; add the column address to get the offset from 1
inc temp            ; add 1. The value of switch is   row*3 + col + 1.
jmp convert_end
letters:  ldi temp, 0xA
add temp, row       ; increment from 0xA by the row value
jmp convert_end
symbols:  cpi col, 0   ; check if we have a star
breq star
cpi col, 1          ; or if we have zero
```

27

27

Sample Code for Key Scanning (10/10)

```
breq zero
ldi temp, 0xF       ; we'll output 0xF for hash
jmp convert_end
star:
ldi temp, 0xE       ; we'll output 0xE for star
rjmp convert_end;
zero: clr temp      ; we have zero
convert_end:
out PORTC, temp     ; write value to PORTC
ret                 ; return to caller
```

28

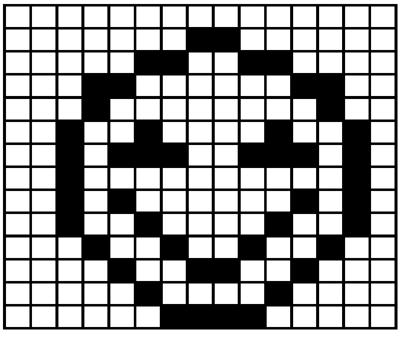
28

Dot Matrix Character LCD (Liquid Crystal Display)

- Equipped with an internal character generator ROM, RAM and RAM for display data.
  - ❑ Characters are displayed using a dot matrix.
- Has its own instruction set.
- All display functions are controllable by instructions.

Principle of Dot Matrix LCD

- Display units (dots) are arranged in rows and columns to form a character, a number, a symbol or graphics.



Pin Assignments

Pin Number	Symbol
1	V_{ss}
2	V_{cc}
3	V_{ee}
4	RS
5	R/W
6	E
7	DB0
8	DB1
9	DB2
10	DB3
11	DB4
12	DB5
13	DB6
14	DB7

31

31

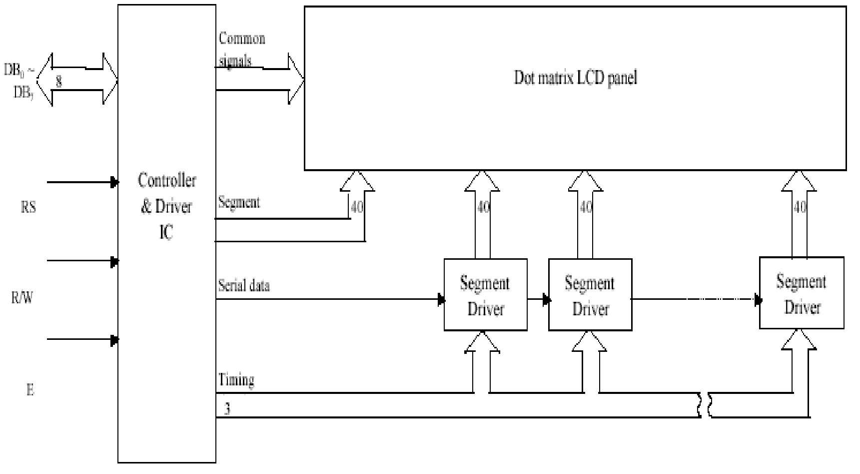
Pin Descriptions

Signal name	No. of Lines	Input/Output	Connected to	Function
DB4 ~ DB7	4	Input/Output	MPU	4 lines of high order data bus. Bi-directional transfer of data between MPU and module is done through these lines. Also DB_7 can be used as a busy flag. These lines are used as data in 4 bit operation.
DB0 ~ DB3	4	Input/Output	MPU	4 lines of low order data bus. Bi-directional transfer of data between MPU and module is done through these lines. In 4 bit operation, these are not used and should be grounded.
E	1	Input	MPU	Enable - Operation start signal for data read/write.
R/W	1	Input	MPU	Signal to select Read or Write "0": Write "1": Read
RS	1	Input	MPU	Register Select "0": Instruction register (Write) : Busy flag; Address counter (Read) "1": Data register (Write, Read)
Vee	1		Power Supply	Terminal for LCD drive power source.
Vcc	1		Power Supply	+5V
Vss	1		Power Supply	0V (GND)

32

32

LCD Block Diagram



33

33

LCD Registers (1/2)

Two internal 8-bit registers:

- Data Register (DR)
  - ❑ The DR is a read/write register used for temporarily storing data to be read/written to/from the DD RAM or CG RAM.
  - ❑ Data written into the DR is automatically written into DD RAM or CG RAM by an internal operation of the display controller.
  - ❑ The DR is also used to store data when reading out data from DD RAM or CG RAM. When address information is written into IR, data is read out from DD RAM or CG RAM to DR by an internal operation. Data transfer is then completed by reading the DR.
  - ❑ After performing a read from the DR, data in the DD RAM or CG RAM at the next address is sent to the DR for the next read cycle.

34

34

LCD Registers (2/2)

- Instruction Register (IR)
  - ❑ The IR is a write-only register storing LCD instructions and addresses for the Display Data RAM (DD RAM) or the Character Generator RAM (CG RAM).
- The register select (RS) signal determines which of these two registers is selected.

RS	R/W	Operation
0	0	IR write, internal operation (Display Clear etc.)
0	1	Busy flag (DB_7) and Address Counter (DB_0 ~ DB_6) read
1	0	DR Write, Internal Operation (DR ~ DD RAM or CG RAM)
1	1	DR Read, Internal Operation (DD RAM or CG RAM)

35

35

Busy Flag

- When the busy flag is high or “1” the module is performing an internal operation and the next instruction will not be accepted.
- The busy flag outputs to DB7 when RS=0 and a read operation is performed. The next instruction must not be written until ensuring that the busy flag is low or “0”.

36

36

LCD Instructions (1/6)

- Clear Display
  - ❑ Writes the space code "20" (hexadecimal) into all addresses of DD RAM. Returns display to its original position if it was shifted. In other words the display clears and the cursor or blink moves to the upper left edge of the display. The execution of clear display instruction sets entry mode to increment mode.

	RS	R/W	DB_7	DB_6	. . .				DB_1	DB_0
Code	0	0	0	0	0	0	0	0	0	1

37

37

LCD Instructions (2/6)

- Return Home
  - ❑ Return the display to its original position if it was shifted. DD RAM contents do not change.
  - ❑ The cursor or the blink moves to the upper left edge of the display. Text on the display remains unchanged.

	RS	R/W	DB_7	DB_6	. . .				DB_1	DB_0
Code	0	0	0	0	0	0	0	0	1	x

38

38

LCD Instructions (3/6)

- Function Set
  - ❑ Sets the interface data length, the number of lines, and character font.
  - ❑ DL: Sets interface data length. Data is sent or received in 8-bit length (DB7 ~ DB0) when DL = “1”, and in 4-bit length (DB7 ~ DB4) when DL = 0. When the 4-bit length is selected, data must be sent or received twice.

	RS	R/W	DB_7	DB_6	. . .				DB_1	DB_0
Code	0	0	0	0	1	DL	N	F	x	x

Note: x = Don’t Care

LCD Instructions (4/6)

- Function Set
  - ❑ N: Sets the number of lines
    - ❖ N = “0” : 1 line display (1/8 duty)
    - ❖ N = “1” : 2 line display (1/16 duty)
  - ❑ F: Sets character font.
    - ❖ F = “1” : 5 x 10 dots
    - ❖ F = “0” : 5 x 7 dots
  - ❑ Note: Perform the function at the head of the program before executing all instructions (except Busy flag/address read). From this point, the function set instruction cannot be executed other than to change interface length.

LCD Instructions (5/6)

- Read busy flag and address
  - ❑ Reads the busy flag (BF) and value of the address counter (AC). BF = 1 indicates that on internal operation is in progress and the next instruction will not be accepted until BF is set to “0”.
  - ❑ The BF status should be checked before each write operation.
  - ❑ At the same time the value of the address counter expressed in binary AAAAAAA is read out. The address counter is used by both CG and DD RAM and its value is determined by the previous instruction.

	RS	R/W	DB_7	DB_6	. . .				DB_1	DB_0
Code	0	1	BF	A	A	A	A	A	A	A

← Higher order bits   Lower order bits →

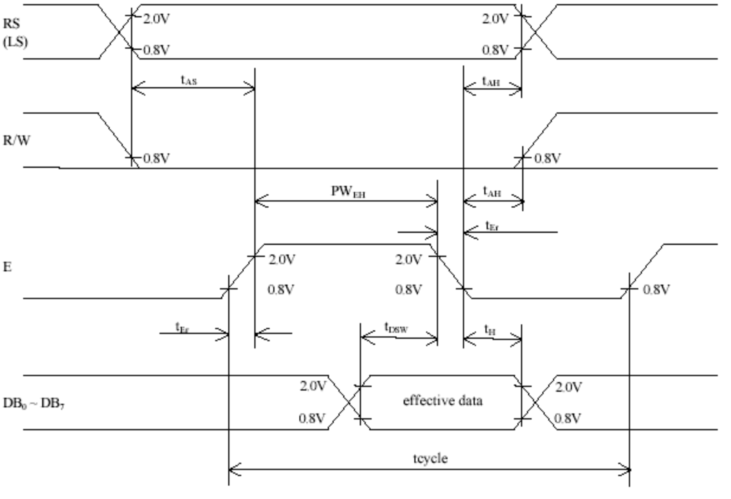
LCD Instructions (6/6)

- Write data to CG or DD RAM
  - ❑ Writes binary 8-bit data DDDDDDDD to the CG or DD RAM.
  - ❑ The previous designation determines whether the CG or DD RAM is to be written (CG RAM address set or DD RAM address set). After a write the entry mode will automatically increase or decrease the address by 1. Display shift will also follow the entry mode.

	RS	R/W	DB_7	DB_6	. . .				DB_1	DB_0
Code	1	0	D	D	D	D	D	D	D	D

← Higher order bits   Lower order bits →

Timing Characteristics of Write (1/2)



43

Timing Characteristics of Write (2/2)

Item		Symbol	Specs. Value		Unit
			Min.	Max.	
Enable cycle time		tcycle	1000	-	ns
Enable pulse width	"High" level	PW_{EH}	450	-	ns
Enable rising, falling time		t_{Er}, t_{Ef}	-	25	ns
Set up time	RS, R/W-E	t_{AS}	140	-	ns
Address hold time		t_{AH}	10	-	ns
Data set up time		t_{DSW}	195	-	ns
Data hold time		t_H	10	-	ns

44

AVR Code for Writing Commands

```
; assume   LCD D0-D7 -> PF0-PF7 and LCD BE-RS -> PA4-PA7
out PORTF, data        ; send the command stored in the register data to PORTF
clr temp               ; temp is a register defined in main
out PORTA, temp        ; RS = 0, RW = 0 for a command write
nop                    ; delay to meet timing (Set up time)
sbi PORTA, 6           ; turn on the enable pin
nop                    ; delay to meet timing (Enable pulse width)
nop
nop
cbi PORTA, 6           ; turn off the enable pin
nop                    ; delay to meet timing (Enable cycle time)
nop
nop
```

45

45

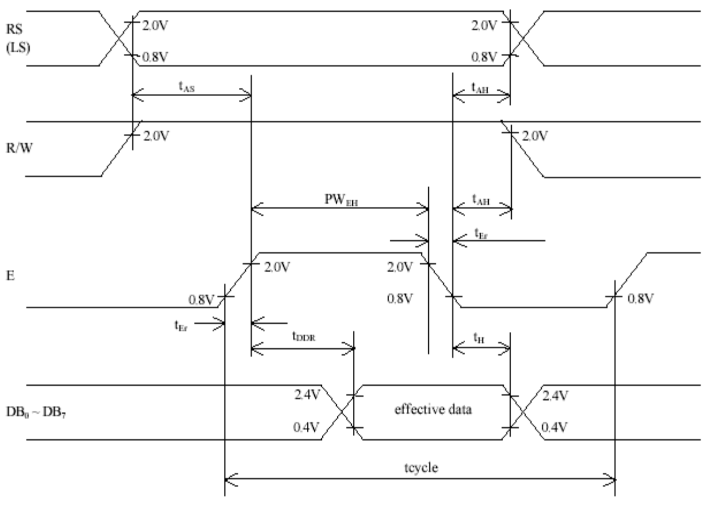
AVR Code for Writing Data

```
; assume   LCD D0-D7 -> PF0-PF7 and LCD BE-RS -> PA4-PA7
out PORTF, data      ; send the command stored in the register data to PORTF
ldi temp, 0b10000000
out PORTA, temp      ; RS = 1, RW = 0 for a data write
nop                  ; delay to meet timing (Set up time)
sbi PORTA, 6         ; turn on the enable pin
nop                  ; delay to meet timing (Enable pulse width)
nop
nop
cbi PORTA, 6         ; turn off the enable pin
nop                  ; delay to meet timing (Enable cycle time)
nop
nop
```

46

46

Timing Characteristics of Read (1/2)



47

47

Timing Characteristics of Read (2/2)

Item		Symbol	Specs. Value		Unit
			Min.	Max.	
Enable cycle time		tcycle	1000	-	ns
Enable pulse width	"High" level	PW_{EH}	450	-	ns
Enable rise, fall time		t_{Er}, t_{Ef}	-	25	ns
Set up time	RS, R/W-E	t_{AS}	140	-	ns
Data delay time		t_{DDR}	-	320	ns
Data hold time		t_H	20	-	ns

48

48

AVR Code for Reading Busy Flag (1/2)

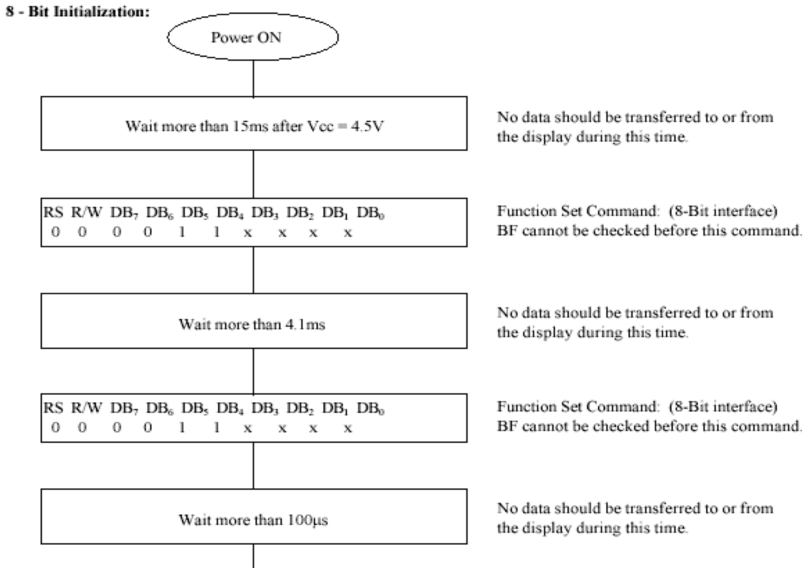
```
; assume   LCD D0-D7 -> PF0-PF7 and LCD BE-RS -> PA4-PA7
; the following code keeps checking the busy flag until the busy flag is 0.

clr temp
out DDRF, temp        ; make PORTF be an input port for now
out PORTF, temp
ldi temp, 0b00100000
out PORTA, temp       ; RS = 0, RW = 1 for a command port read
busy_loop:
nop                   ; delay to meet timing (Set up time / Enable cycle time)
sbi PORTA, 6          ; turn on the enable pin
```


AVR Code for Reading Busy Flag (2/2)

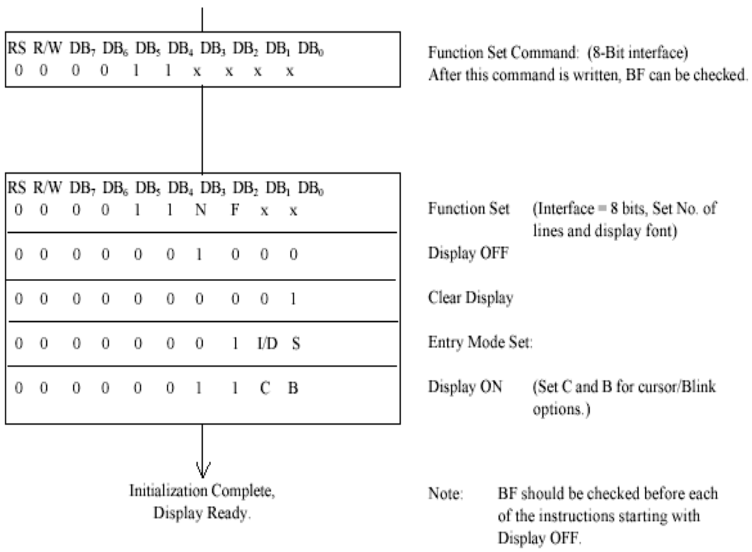
```
nop                   ; delay to meet timing (Data delay time)
nop
nop
in temp, PINF         ; read value from LCD
cbi PORTA, 6          ; turn off the enable pin
sbrc temp, 7          ; if the busy flag is set
rjmp busy_loop        ; repeat command read
clr temp              ; else
out PORTA, temp       ; turn off read mode,
ser temp
sts DDRF, temp        ; make PORTD an output port again
```

LCD Initialization (1/2)



51

LCD Initialization (2/2)



52

52

AVR Code for LCD Initialization (1/3)

```
ser temp        ; temp is a register defined in main
sts DDRD, temp        ; PORTF,  the data port is usually all outputs
out DDRA, temp        ; PORTA, the control port is always all outputs
ldi del_lo, low(15000)
ldi del_hi, high(15000)
rcall delay                   ; delay for > 15ms
                              ; function set command with N = 1 and F = 0
ldi data, 0b00111000
rcall lcd_write_com  ; 1st Function set command with 2 lines and 5*7 font
ldi del_lo, low(4100)  ; del_high:del_low is the input of the delay subroutine
ldi del_hi, high(4100) ; which is a loop defined elsewhere
```

53

53

AVR Code for LCD Initialization (2/3)

```
rcall delay                   ; delay for > 4.1ms
rcall lcd_write_com    ; 2nd Function set command with 2 lines and 5*7 font
ldi del_lo, low(100)
ldi del_hi, high(100)
rcall delay                   ; delay for > 100us
rcall lcd_write_com   ; 3rd Function set command with 2 lines and 5*7 font
rcall lcd_write_com   ; final Function set command with 2 lines and 5*7 font
rcall lcd_wait_busy    ; wait until the LCD is ready
ldi data, 0b00001000
rcall lcd_write_com   ; turn Display off
rcall lcd_wait_busy   ; wait until the LCD is ready
```

54

54

AVR Code for LCD Initialization (3/3)

```
ldi data, 1
rcall lcd_write_com    ; clear display
rcall lcd_wait_busy    ; wait until the LCD is ready
                       ; entry set command with I/D = 1 and S = 0
ldi data, 0b00000110
rcall lcd_write_com    ; set Entry mode: Increment = yes and Shift = no
rcall lcd_wait_busy    ; wait until the LCD is ready
                       ; display on command with C = 0 and B = 1
ldi data, 0b000001110
rcall lcd_write_com    ; turn display on with a cursor that doesn't blink
```

55

55

Reading

1. Chapter 7: Computer Buses and Parallel Input and Output. Microcontrollers and Microcomputers by Fredrick M. Cady.
2. Dot Matrix Character LCD User's Manual (http://www.cse.unsw.edu.au/~cs2121/LCD_Manual.pdf).

56

56