Lecture Overview

- Input devices
  - Input switches
    - Basics of switches
  - Keypads
- Output devices
  - LCD
Input Switches

- Most basic binary input devices
- The switch output is high or low, depending on 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 contact and break, for a few milliseconds (typically 5 to 10 ms).
Input Switches (cont.)

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

(b) Multiple pole switch.

R Typically 1K Ohm

Logic high with switch open
Logic low with switch closed

Data Bus

\(1/2\) 74LS244 Octal Buffer
NAND Latch Debouncer

Logic high with switch up

Logic low with switch down
Software Debouncing

● Basic idea: wait until the switch is stable
● For example:
  ● Wait and see:
    ● If the software detects a low logic level, indicating that
      switch has closed, it simply waits for some time, 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, decrease the counter;
      otherwise, increment the counter.
    ● If the counter is 0, we know that switch output has been low
      (closed) for at least 10 ms. If, on the other hand, the
      counter reaches 20, we know that the switch has been
      open for at least 10 ms.
One-Dimensional Array of Switches

Selected Input From Output Port

I0 74LS151 8 to 1 Multiplexer
I1
I2
I3
I4
I5
I6
I7

One-Dimensional Array of Switches

Scanned Switch Data To Input Port

Vcc

A
One-Dimensional Array of Switches

- Switch bouncing problem must be solved
  - Either using software or hardware
- 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-bit sequence from 000 to 111, the multiplexer selects each of the switch inputs.
  - The output of switch array could be interfaced directly to an eight-bit port at point A.
Keyboard Matrix of Switches

74LS151 8-to-1 Input Multiplexer

74LS138 3-of-8 Decoder

Scan Input From Output Port

Select Input From Output Port

Scanned Switch Data To Input Port

Vcc

A

I0 I1 I2 I3 I4 I5 I6 I7

B

00 01 02 06 07

10 11 12 17

06 07

E

S0 S1 S2

Vcc

0

Z

O0 O1 O2 O3 O4 O5 O6 O7

E3 E2 E1 A2 A1 A0

Scan Input From Output Port

3-of-8

74LS138 Decodor

To Input Port
Keyboard Matrix of Switches (cont.)

- 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.
Keyboard Matrix of Switches (cont.)

- Software can scan the keyboard by outputting a three-bit code to the decoder and then scanning the multiplexer to find the closed switch or switches.
  - The combination of the two 3-bit scan codes (A2A1A0 and S2S1S0) identifies which switch is closed. For example, the code 00000000 scan switch 00 in the upper left-hand corner.
- The diode prevents a problem called ghosting.
Ghosting

Row 0 (Pulled low, error)
Row 1 (Pulled low, OK)
Row 2 (High, OK)

Low
(Scanned column)
Ghosting (cont.)

- 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.
Example

- Get the input from 4*4 keypad

```
<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>B</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>C</td>
</tr>
<tr>
<td>*</td>
<td>0</td>
<td>#</td>
<td>D</td>
</tr>
</tbody>
</table>
```

C0  C1  C2  C3
R0  R1  R2  R3
Example (solution)

- Algorithm

Scan columns from left to right
  for each column, scan rows from top to bottom
  for each key being scanned
    if it is pressed
      display
      wait
    endif
  endfor
endfor
Repeat the scan process

- A column is selected, its related Cx value is set to 0.
- A mask is used to read one row at a time.
The program gets input from keypad and displays its ascii value on the LED bar

.include "m2560def.inc"

def row = r16 ; current row number
def col = r17 ; current column number
def rmask = r18 ; mask for current row during scan
def cmask = r19 ; mask for current column during scan
def temp1 = r20
def temp2 = r21

equ PORTADIR = 0xF0 ; PD7-4: output, PD3-0, input
equ INITCOLMASK = 0xEF ; scan from the rightmost column,
equ INITROWMASK = 0x01 ; scan from the top row
equ ROWMASK = 0x0F ; for obtaining input from Port D
RESET:

```assembly
ldi    temp1, low(RAMEND)    ; initialize the stack
out    SPL, temp1
ldi    temp1, high(RAMEND)
out    SPH, temp1

ldi    temp1, PORTADIR       ; PA7:4/PA3:0, out/in
out    DDRA, temp1
ser    temp1                  ; PORTC is output
out    DDRC, temp1
out    PORTC, temp1

main:

ldi    cmask, INITCOLMASK    ; initial column mask
clr    col                    ; initial column
```
colloop:
cpi col, 4
breq main ; If all keys are scanned, repeat.
out PORTA, cmask ; Otherwise, scan a column.

ldi temp1, 0xFF ; Slow down the scan operation.
delay:
dec temp1
brne delay

in temp1, PINA ; Read PORTA
andi temp1, ROWMASK ; Get the keypad output value
cpi temp1, 0xF ; Check if any row is low
breq nextcol

ldi rmask, INITROWMASK ; If yes, find which row is low
clr row ; Initialize for row check

breq nextcol
Code Implementation

code:

rowloop:
cpi row, 4
breq nextcol ; the row scan is over.
mov temp2, temp1
and temp2, rmask ; check un-masked bit
breq convert ; if bit is clear, the key is pressed
inc row ; else move to the next row
lsl rmask
jmp rowloop

nextcol:
lsl cmask ; if row scan is over
inc col ; increase column value
jmp colloop ; go to the next column
convert:
  cpi  col, 3 ; If the pressed key is in col.3
  breq letters ; we have a letter

  ; If the key is not in col.3 and
  cpi  row, 3 ; If the key is in row3,
  breq symbols ; we have a symbol or 0

  mov  temp1, row ; Otherwise we have a number in 1-9
  lsl  temp1
  add  temp1, row
  add  temp1, col ; temp1 = row*3 + col
  subi temp1, -'1' ; Add the value of character ‘1’
  jmp  convert_end
letters:
  ldi temp1, 'A'
  add temp1, row
  jmp convert_end ; Get the ASCII value for the key

symbols:
  cpi col, 0
  breq star
  cpi col, 1
  breq zero
  ldi temp1, '#'
  jmp convert_end ; if not we have hash

star:
  ldi temp1, '*'
  jmp convert_end ; Set to star

zero:
  ldi temp1, '0'
  jmp convert_end ; Set to zero

convert_end:
  out PORTC, temp1 ; Write value to PORTC
  jmp main ; Restart main loop
LCD

- Liquid Crystal Display
- Programmable output device
Dot Matrix LCD

- Characters are displayed using a dot matrix.
  - 5x7, 5x8, and 5x11
- A controller is used for communication between the LCD and other devices, e.g. MPU
- The controller has an internal character generator ROM. All display functions are controllable by instructions.
## Pin Assignments

<table>
<thead>
<tr>
<th>Pin Number</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$V_{ss}$</td>
</tr>
<tr>
<td>2</td>
<td>$V_{cc}$</td>
</tr>
<tr>
<td>3</td>
<td>$V_{ee}$</td>
</tr>
<tr>
<td>4</td>
<td>RS</td>
</tr>
<tr>
<td>5</td>
<td>R/W</td>
</tr>
<tr>
<td>6</td>
<td>E</td>
</tr>
<tr>
<td>7</td>
<td>DB0</td>
</tr>
<tr>
<td>8</td>
<td>DB1</td>
</tr>
<tr>
<td>9</td>
<td>DB2</td>
</tr>
<tr>
<td>10</td>
<td>DB3</td>
</tr>
<tr>
<td>11</td>
<td>DB4</td>
</tr>
<tr>
<td>12</td>
<td>DB5</td>
</tr>
<tr>
<td>13</td>
<td>DB6</td>
</tr>
<tr>
<td>14</td>
<td>DB7</td>
</tr>
</tbody>
</table>
## Pin Descriptions

<table>
<thead>
<tr>
<th>Signal name</th>
<th>No. of Lines</th>
<th>Input/Output</th>
<th>Connected to</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB4 ~ DB7</td>
<td>4</td>
<td>Input/Output</td>
<td>MPU</td>
<td>4 lines of high order data bus. Bi-directional transfer of data between MPU and module is done through these lines. Also DB7 can be used as a busy flag. These lines are used as data in 4 bit operation.</td>
</tr>
<tr>
<td>DB0 ~ DB3</td>
<td>4</td>
<td>Input/Output</td>
<td>MPU</td>
<td>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.</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>Input</td>
<td>MPU</td>
<td>Enable - Operation start signal for data read/write.</td>
</tr>
<tr>
<td>R/W</td>
<td>1</td>
<td>Input</td>
<td>MPU</td>
<td>Signal to select Read or Write</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>“0”: Write</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>“1”: Read</td>
</tr>
<tr>
<td>RS</td>
<td>1</td>
<td>Input</td>
<td>MPU</td>
<td>Register Select</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>“0”: Instruction register (Write)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>: Busy flag; Address counter (Read)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>“1”: Data register (Write, Read)</td>
</tr>
<tr>
<td>Vee</td>
<td>1</td>
<td></td>
<td>Power Supply</td>
<td>Terminal for LCD drive power source.</td>
</tr>
<tr>
<td>Vcc</td>
<td>1</td>
<td></td>
<td>Power Supply</td>
<td>+5V</td>
</tr>
<tr>
<td>Vss</td>
<td>1</td>
<td></td>
<td>Power Supply</td>
<td>0V (GND)</td>
</tr>
</tbody>
</table>
Dot Matrix LCD Diagram

Controller & Driver IC

RS

RW

E

DB_9 ~ DB_7

Common signals

Dot matrix LCD panel

Segment Driver

Segment Driver

Segment Driver

Serial data

Segment

Timing

40

40

40

40
Operations

- MPU communicates with LCD through two registers
  - Instruction Register (IR)
    - To store instruction codes like Display clear or Cursor Shift as well as addresses for the Display Data RAM (DD RAM) or the Character Generator RAM (CG RAM)
  - Data Register (DR)
    - To temporarily store data to be read/written to/from the DD RAM of the display controller.
The register select (RS) signal determines which of these two registers is selected.

<table>
<thead>
<tr>
<th>RS</th>
<th>R/W</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>IR write, internal operation (Display Clear etc.)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Busy flag (DB7) and Address Counter (DB0 ~ DB6) read</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>DR Write, Internal Operation (DR ~ DD RAM or CG RAM)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>DR Read, Internal Operation (DD RAM or CG RAM)</td>
</tr>
</tbody>
</table>
Operations (cont.)

- When the busy flag is high or ‘1’, the LCD module is busy with internal operation.
- The next instruction must not be written until the busy flag is low or ‘0’.
- For details, refer to the LCD USER’S MANUAL.
LCD Instructions

- A list of binary instructions are available for LCD operations
- Some typical ones are explained in the next slides.
Instructions

- **Clear Display**

<table>
<thead>
<tr>
<th>RS</th>
<th>R/W</th>
<th>DB7</th>
<th>DB6</th>
<th>DB5</th>
<th>BD4</th>
<th>DB3</th>
<th>DB2</th>
<th>DB1</th>
<th>DB0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- 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.
Instructions

- Return Home

<table>
<thead>
<tr>
<th>RS</th>
<th>R/W</th>
<th>DB7</th>
<th>DB6</th>
<th>DB5</th>
<th>BD4</th>
<th>DB3</th>
<th>DB2</th>
<th>DB1</th>
<th>DB0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- The cursor or the blink moves to the upper left edge of the display. Text on the display remains unchanged.
Instructions

- Entry Mode Set

<table>
<thead>
<tr>
<th>Code</th>
<th>RS</th>
<th>R/W</th>
<th>DB7</th>
<th>DB6</th>
<th>DB5</th>
<th>BD4</th>
<th>DB3</th>
<th>DB2</th>
<th>DB1</th>
<th>DB0</th>
<th>I/D</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>I/D</td>
<td>S</td>
</tr>
</tbody>
</table>

- Sets the Increment/Decrement and Shift modes to the desired settings.
  - I/D: Increments (I/D = 1) or decrements (ID = 0) the DD RAM address by 1 when a character code is written into or read from the DD RAM.
  - The cursor or blink moves to the right when incremented by +1.
  - The same applies to writing and reading the CG RAM.
  - S: Shifts the entire display either to the right or to the left when S = 1; shift to the left when I/D = 1 and to the right when I/D = 0.
Instructions

Display ON/OFF Control

- Controls the display ON/OFF status, Cursor ON/OFF and Cursor Blink function.
  - D: The display is ON when D = 1 and OFF when D = 0.
  - C: The cursor displays when C = 1 and does not display when C = 0.
  - B: The character indicated by the cursor blinks when B = 1.

<table>
<thead>
<tr>
<th>Code</th>
<th>RS R/W</th>
<th>DB7</th>
<th>DB6</th>
<th>DB5</th>
<th>BD4</th>
<th>DB3</th>
<th>DB2</th>
<th>DB1</th>
<th>DB0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Instructions

• Cursor or Display Shift

<table>
<thead>
<tr>
<th>RS</th>
<th>R/W</th>
<th>DB7</th>
<th>DB6</th>
<th>DB5</th>
<th>BD4</th>
<th>DB3</th>
<th>DB2</th>
<th>DB1</th>
<th>DB0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>S/C</td>
<td>R/L</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Shifts the cursor position or display to the right or left without writing or reading display data.

<table>
<thead>
<tr>
<th>S/C</th>
<th>R/L</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Instructions

• Function Set

<table>
<thead>
<tr>
<th>RS</th>
<th>R/W</th>
<th>DB7</th>
<th>DB6</th>
<th>DB5</th>
<th>BD4</th>
<th>DB3</th>
<th>DB2</th>
<th>DB1</th>
<th>DB0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>x</td>
</tr>
</tbody>
</table>

- Sets the interface data length, the number of lines, and character font.
  - DL = 1, 8 –bits; otherwise 4 bits
  - N: Sets the number of lines
    - N = 0 : 1 line display
    - N = 1 : 2 line display
  - F: Sets character font.
    - F = 1 : 5 x 10 dots
    - F = 0 : 5 x 7 dots
Instructions

- Read Busy Flag and Address

Reads the busy flag (BF) and value of the address counter (AC). BF = 1 indicates that an internal operation is in progress and the next instruction will not be accepted until BF is set to ‘0’. If the display is written while BF = 1, abnormal operation will occur.
Instructions

- **Write Data to CG or DD RAM**

<table>
<thead>
<tr>
<th>RS</th>
<th>R/W</th>
<th>DB7</th>
<th>DB6</th>
<th>DB5</th>
<th>BD4</th>
<th>DB3</th>
<th>DB2</th>
<th>DB1</th>
<th>DB0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
</tbody>
</table>

- Writes binary 8-bit data DDDDDDDDDDDDDDD 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.
Timing Characteristics

- For write operation
Timing Characteristics

- For read operation
Examples

- Send a command to LCD

; Register data stores value to be written to the LCD
; Port D is output and connects to LCD; Port A controls the LCD.
; Assume all other labels are pre-defined.

.macro lcd_write_com
    out PORTD, data     ; set the data port's value up
    clr temp
    out PORTA, temp     ; RS = 0, RW = 0 for a command write
    nop                 ; delay to meet timing (Set up time)
    sbi PORTA, LCD_E    ; turn on the enable pin
    nop
    nop
    nop
    cbi PORTA, LCD_E    ; turn off the enable pin
    nop
    nop
    nop
.endmacro
Examples

- Send data to display

; comments are same as in previous slide.

.macro lcd_write_data
  out PORTD, data
  ldi temp, 1 << LCD_RS
  out PORTA, temp
  nop
  sbi PORTA, LCD_E
  nop
  nop
  nop
  cbi PORTA, LCD_E
  nop
  nop
  nop
.endmacro
Examples

- Check LCD and wait until LCD is not busy

```assembly
; comments are same as in the previous slide
.macro lcd_wait_busy
    clr temp
    out DDRD, temp ; Make PORTD be an input port for now
    out PORTD, temp
    ldi temp, 1 << LCD_RW
    out PORTA, temp ; RS = 0, RW = 1 for a command port read
busy_loop:
    nop ; delay to meet set-up time)
    sbi PORTA, LCD_E ; turn on the enable pin
    nop ; delay to meet timing (Data delay time)
    nop
    nop
    in temp, PIND ; read value from LCD
    cbi PORTA, LCD_E ; turn off the enable pin
    sbrc temp, LCD_BF ; if the busy flag is set
    rjmp busy_loop ; repeat command read
    clr temp ; else
    out PORTA, temp ; turn off read mode,
    ser temp ;
    out DDRD, temp ; make PORTD an output port again
.endmacro
```
LCD Initialization

- LCD should be initialized before use
- Internal Reset Circuit can be used, but it is related to power supply loading, may not work properly.
- Therefore, software initialization is recommended.
Software Initialization

8-Bit Initialization:

Power ON

Wait more than 15ms after Vcc = 4.5V

No data should be transferred to or from the display during this time.

RS R/W DB₇ DB₆ DB₅ DB₄ DB₃ DB₂ DB₁ DB₀
0 0 0 0 0 1 1 x x x x

Function Set Command: (8-Bit interface)
BF cannot be checked before this command.

Wait more than 4.1ms

No data should be transferred to or from the display during this time.

RS R/W DB₇ DB₆ DB₅ DB₄ DB₃ DB₂ DB₁ DB₀
0 0 0 0 0 1 1 x x x x

Function Set Command: (8-Bit interface)
BF cannot be checked before this command.

Wait more than 100μs

No data should be transferred to or from the display during this time.
Software Initialization

Wait more than 100μs

No data should be transferred to or from the display during this time.

Function Set Command: (8-Bit interface)
After this command is written, BF can be checked.

Function Set (Interface = 8 bits, Set No. of lines and display font)
Display OFF
Clear Display
Entry Mode Set:
Display ON  (Set C and B for cursor/Blink options.)

Note: BF should be checked before each of the instructions starting with Display OFF.
Example of Initialization Code

.include "m2560def.inc"

; The del_hi:del_lo register pair store the loop counts
; each loop generates about 1 us delay
.macro delay
loop:  subi del_lo, 1
       sbci del_hi, 0
       nop
       nop
       nop
       nop
       brne loop ; taken branch takes two cycles.
       ; one loop time is 8 cycles = ~1.08us
.endmacro

; continued
Example of Initialization Code

ldi del_lo, low(15000) ; delay (>15ms)
ldi del_hi, high(15000)
delay

; Function set command with N = 1 and F = 0
; for 2 line display and 5*7 font. The 1st command
ldi data, LCD_FUNC_SET | (1 << LCD_N)
lcd_write_com

ldi del_lo, low(4100) ; delay (>4.1 ms)
ldi del_hi, high(4100)
delay

lcd_write_com ; 2nd Function set command

; continued
Example of Initialization Code

```assembly
ldi del_lo, low(100)          ; delay (>100 ns)
ldi del_hi, high(100)
delay

lcd_write_com                 ; 3rd Function set command
lcd_write_com                 ; Final Function set command

lcd_wait_busy                 ; Wait until the LCD is ready
ldi data, LCD_DISP_OFF
lcd_write_com                 ; Turn Display off

lcd_wait_busy                 ; Wait until the LCD is ready
ldi data, LCD_DISP_CLR
lcd_write_com                 ; Clear Display

; continued
```
Example of Initialization Code

```
lcd_wait_busy ; Wait until the LCD is ready
; Entry set command with I/D = 1 and S = 0
; Set Entry mode: Increment = yes and Shift = no
ldi data, LCD_ENTRY_SET | (1 << LCD_ID)
lcd_write_com

lcd_wait_busy ; Wait until the LCD is ready
; Display On command with C = 1 and B = 0
ldi data, LCD_DISP_ON | (1 << LCD_C)
lcd_write_com
```
Reading Material

- Chapter 7: Computer Buses and Parallel Input and Output. Microcontrollers and Microcomputers by Fredrick M. Cady.
  - Simple I/O Devices
- DOT Matrix LCD User’s Manual
  - Available on the course website.
Homework

1. The circuit shown in the next slide is a switch array input circuit. Is there any switch bounce issue with this circuit? Can the CLK frequency have any impact on this problem? How to solve it in hardware?
Selected Input From Output Port

Vcc

74LS151 8 to 1 Multiplexer

Selected Input From Output Port

D

CLK
Homework

2. Write an assembly program to initialize LCD panel to display characters in one line with 5x7 font.