Microprocessors & Interfacing

Input/Output Devices (I)

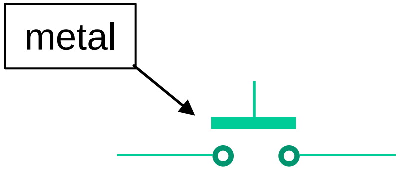
Lecturer : Annie Guo

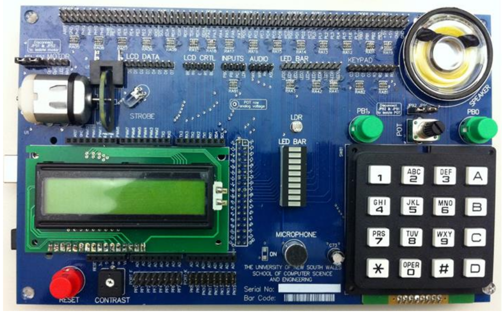
Lecture Overview

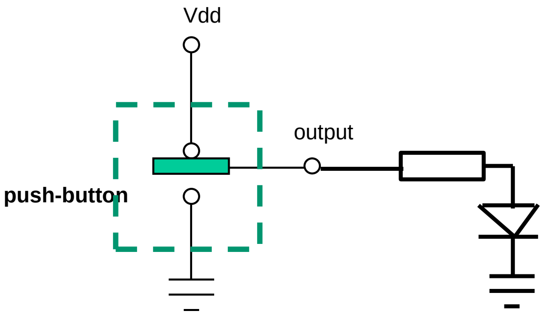
- Input devices
  - Push Button
  - Input switch
  - Keypad

Push Button

- A small mechanical device that can control the connection of two electric nodes (wires).
  - When it is pushed, the small metal inside the button connects two wires.
- Can be used as a 1-bit input device, as used in our lab board
  - Not pushed: 1
  - Pushed down: 0



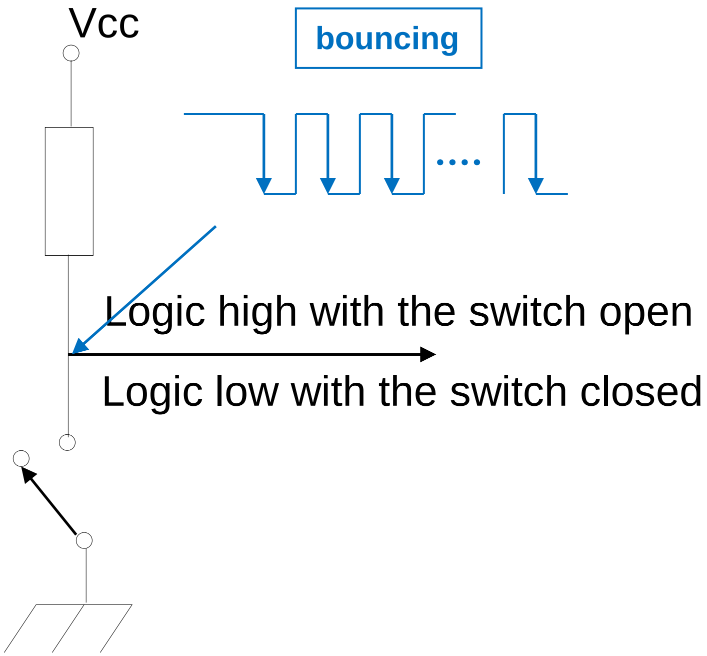




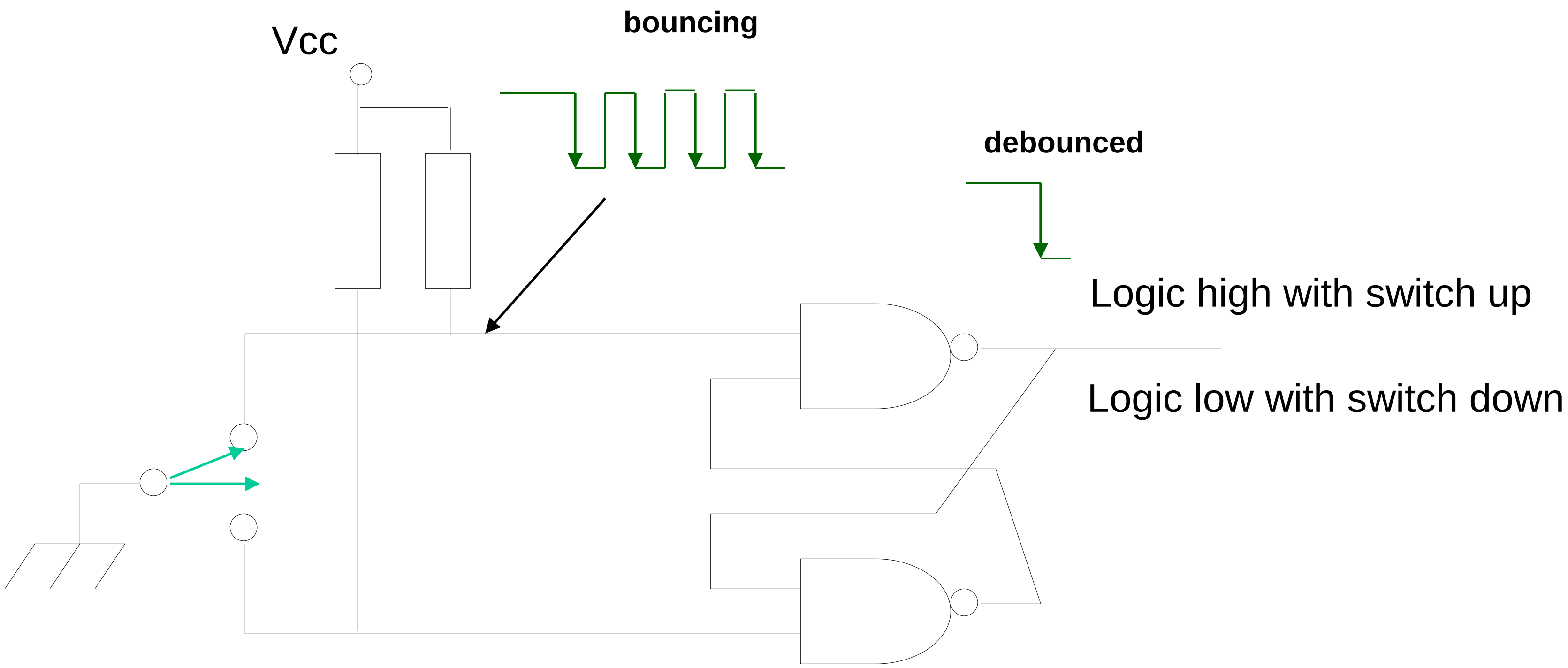
Input Switch

- Like the push button, a switch provides two different values, depending on the switch position.
- Pull-up resistor/circuit may be needed for the switch to provide a high logic level when the switch is open.
- Problem with switch (also push button):
  - Switch bouncing弹跳
    - When a switch makes contact, its mechanical 弹性springiness will cause the contact to bounce, namely contact and break, for a few milliseconds (typically 5 to 10 ms).

Switch Bouncing Example


Vcc
bouncing
….
Logic high with the switch open
Logic low with the switch closed

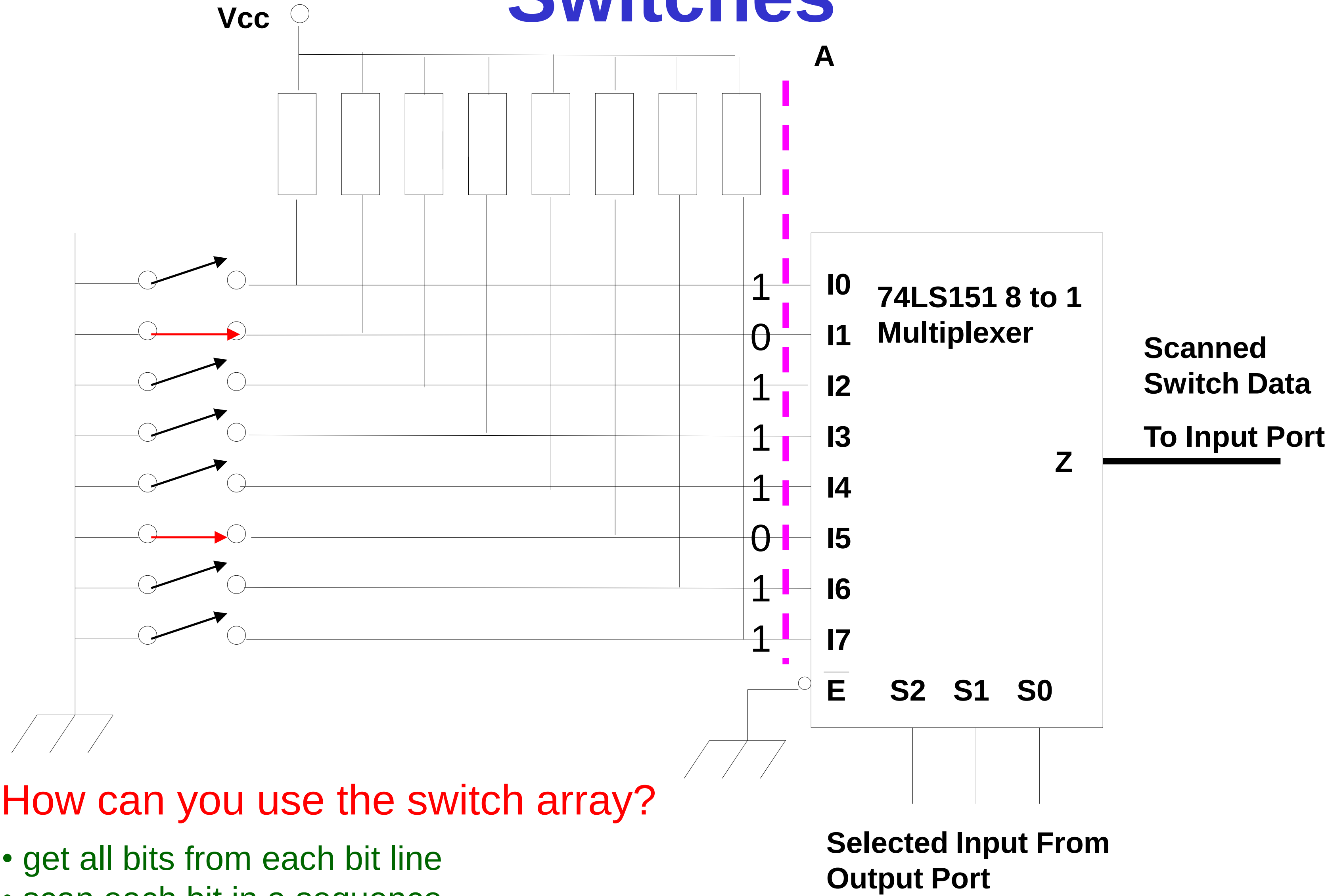
NAND Latch Debouncer*



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 tests if the switch is still low.
  - Counter-based approach:
    - Initialize a counter to 10.
    - 查询 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 output has been high for at least 10 ms.

One-Dimensional Array of Switches



How can you use the switch array?

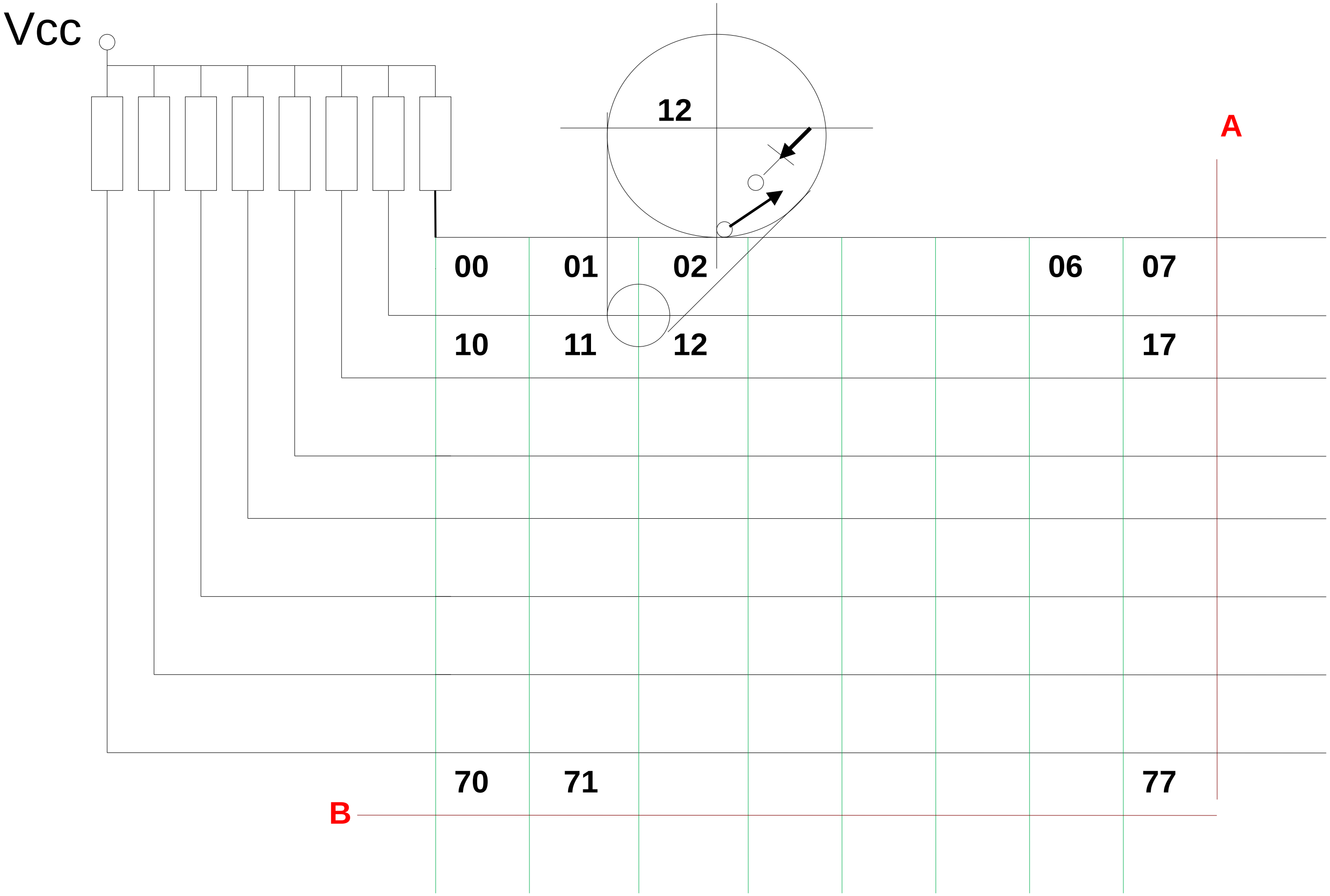
- get all bits from each bit line
- scan each bit in a sequence

One-Dimensional Array of Switches (cont.)

- Switch bouncing problem must be solved
  - Either using software or hardware
- The output of switch array can be interfaced directly to an eight-bit port at point A.
- The array of switches can also be scanned by the software to find out which switches are closed or open.
  - The software outputs a 3-bit sequence from 000 to 111 and the multiplexer selects each of the switch inputs.

Keypad

- Internal circuit diagram



Keypad (cont.)

- A keypad is a set of switches arranged in a two-dimensional matrix, consisting of two layers

  水平的

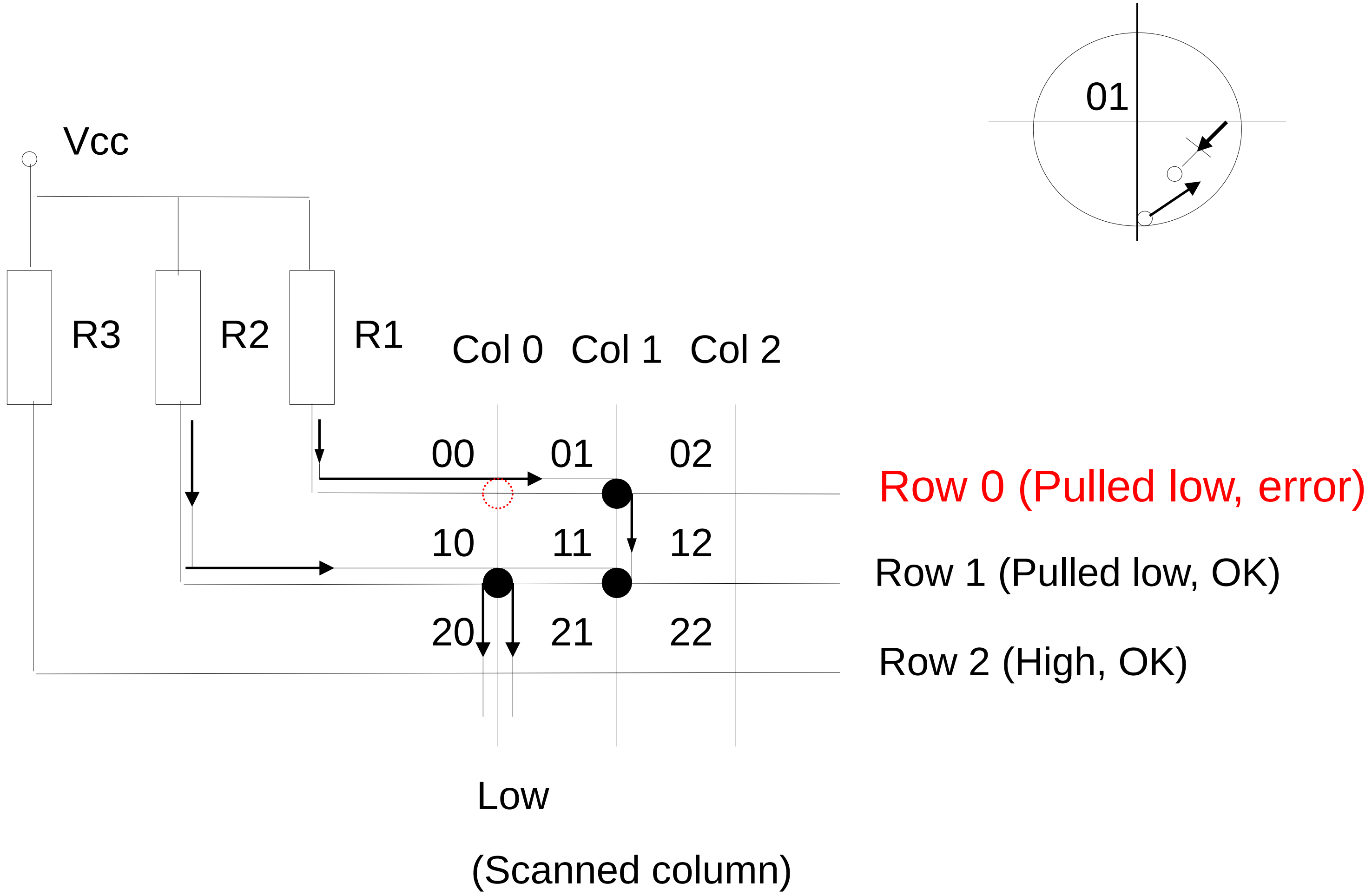
  - A layer of the horizontal lines
    - connected to the power supply via resistors
  - A layer of the vertical lines
    - normally disconnected to the horizontal layer
- Each intersection of the vertical and horizontal lines forms a switch
  - The switch can be operated by a key button
  - When the key is pressed, the switch connects both two lines.

Keypad (cont.)

- The 8*8 keypad can be interfaced directly to 8-bit output and input ports
  - at point *A (to input port)* and point *B (to output port)*
- The output from each horizontal line
  - Normally is a logic high (1)
  - Becomes logic low (0) when a key is pressed and the related vertical line is set/connected to logic low (0)
- The diode prevents a problem called ghosting.

What happens to the output for a key press if the related vertical line is not connected to logic low?

Ghosting*



Row 0 (Pulled low, error)

Row 1 (Pulled low, OK)

Row 2 (High, OK)

Low

(Scanned column)

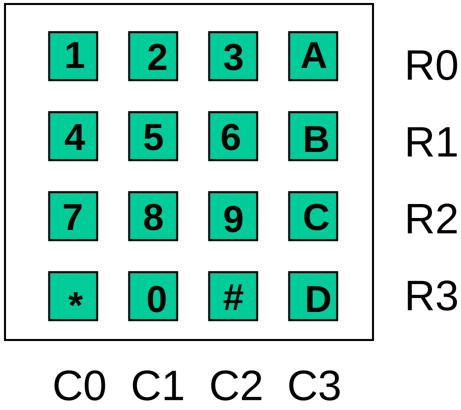
Ghosting (cont.)*

- Ghosting occurs when several keys are pushed at once.
- Consider the case shown in the figure in the previous slide, 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.

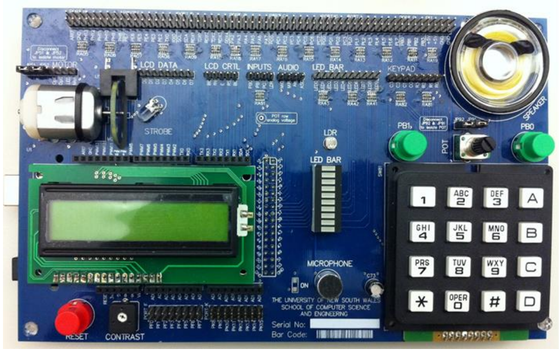
  As the rows are scanned, a low is sensed on Row 1, which is true because switch 10 is closed. But a low is also seen on Row 0, indicating switch 00 is closed, which is NOT true.
- The diodes in the switches eliminate this problem by preventing current flow from R1 through switches 01 and 11. Thus Row 0 will not be low when it is scanned.

Example

- Get an input from 4x4 keypad used in our lab board.



C0	C1	C2	C3	
1	2	3	A	R0
4	5	6	B	R1
7	8	9	C	R2
*	0	#	D	R3



Example (solution)

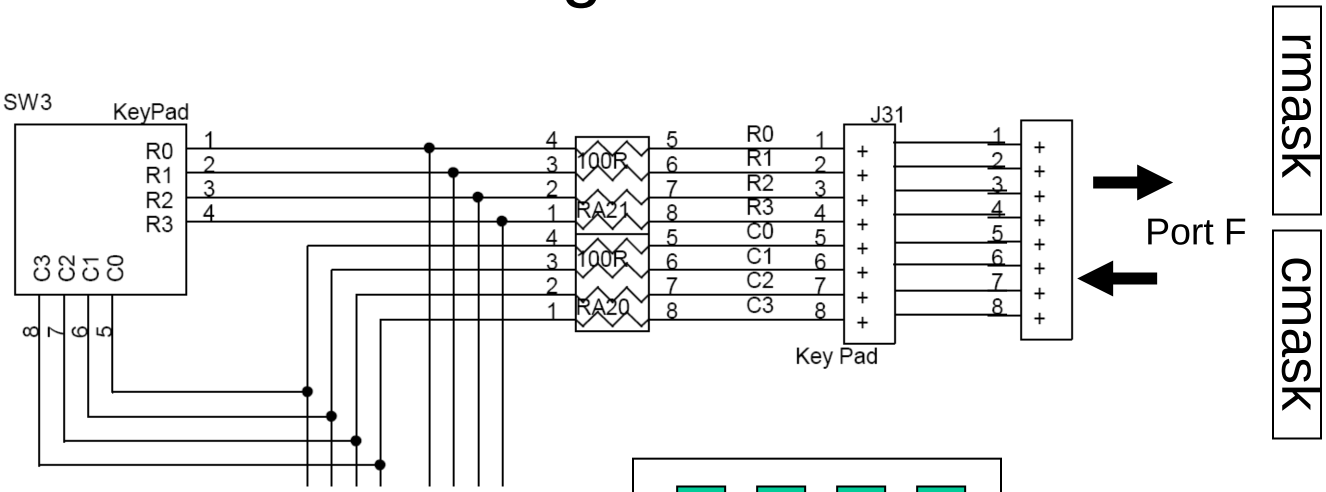
- Algorithm

```
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
```

  - To select a column, set the related Cx value to 0
  - A mask is used to read one row at a time.

Example (solution)

- Hardware Interfacing



1	2	3	A	R0
4	5	6	B	R1
7	8	9	C	R2
*	0	#	D	R3
C0	C1	C2	C3	

Code Implementation

```
; 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 PORTFDIR = 0xF0               ; PF7-4: output, PF3-0, input
.equ ROWMASK  =0x0F                ; for obtaining input from Port F
.equ INITCOLMASK = 0xEF            ; scan from the leftmost column,
.equ INITROWMASK = 0x01            ; scan from the top row
```

Code Implementation

```
RESET:

        ldi     temp1, PORTFDIR              ; PF7:4/PF3:0, out/in
        out     DDRF, temp1
        ser     temp1                        ; PORTC is set output
        out     DDRC, temp1                  ; to display ASCII of pressed.
        out     PORTC, temp1                 ; Initially LEDs are turned on

main:
        ldi     cmask, INITCOLMASK           ; initial column mask
        clr     col                          ; initial column
```

Code Implementation

```
colloop:
        cpi     col, 4
        breq    main                          ; if all keys are scanned, repeat.
        out     PORTF, cmask                  ; otherwise, scan a column

        ldi     temp1, 0xFF                   ; slow down the scan operation.
delay:  dec  -1 temp1
        brne    delay      不相等转移

        in      temp1, PINF                   ; read PORTF
        andi    temp1, ROWMASK   与立即数     ; get the keypad output value
        cpi     temp1, 0xF                    ; check if any row is low    low:pressed, 0
        breq    nextcol
                                              ; if yes, find which row is low
        ldi     rmask, INITROWMASK            ; initialize for row check
        clr     row                           ;
```

Code Implementation

```
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   右移  rmask
        rjmp        rowloop

nextcol:                                       ; if row scan is over
        lsl cmask
        inc col                                ; increase column value
        rjmp colloop                           ; go to the next column
```

Code Implementation

```
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’
        rjmp    convert_end
```

Code Implementation

```
letters:
            ldi temp1, 'A'
            add temp1, row                      ; Get the ASCII value for the key
            rjmp convert_end

symbols:
            cpi col, 0                          ; Check if we have a star
            breq star
            cpi col, 1                          ; or if we have zero
            breq zero
            ldi temp1, '#'                      ; if not we have hash
            rjmp convert_end
star:
            ldi temp1, '*'                      ; Set to star
            rjmp convert_end
zero:
            ldi temp1, '0'                      ; Set to zero

convert_end:
            out PORTC, temp1                    ; Write value to PORTC
            ;delay
            rjmp main                           ; Restart main loop
```

Reading Material

- Chapter 9: Computer Buses and Parallel Input and Output. Microcontrollers and Microcomputers by Fredrick M. Cady.
  - Simple I/O Devices

Homework

1. Refer to the AVR Instruction Set manual, study the following instructions:
    - Arithmetic and logic instructions
        - lsr, ror
        - lsl, rol
    - Data transfer instructions
        - sts, lds
    - Bit
        - clc
        - sec

Homework

2. White an AVR assembly program to map the number-keys on the keypad to the individual LEDs on the LED bar. For example, when key 0 is pressed, LED 0 is turned on. After all number keys are pressed, all LEDs are on.