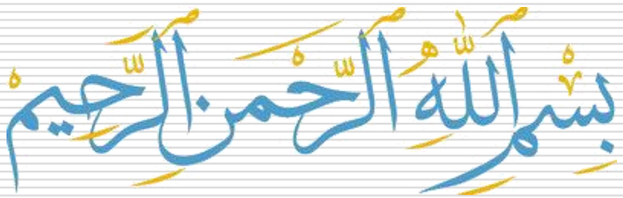


میکرو کنترلرهای AVR

LCD و KeyPad

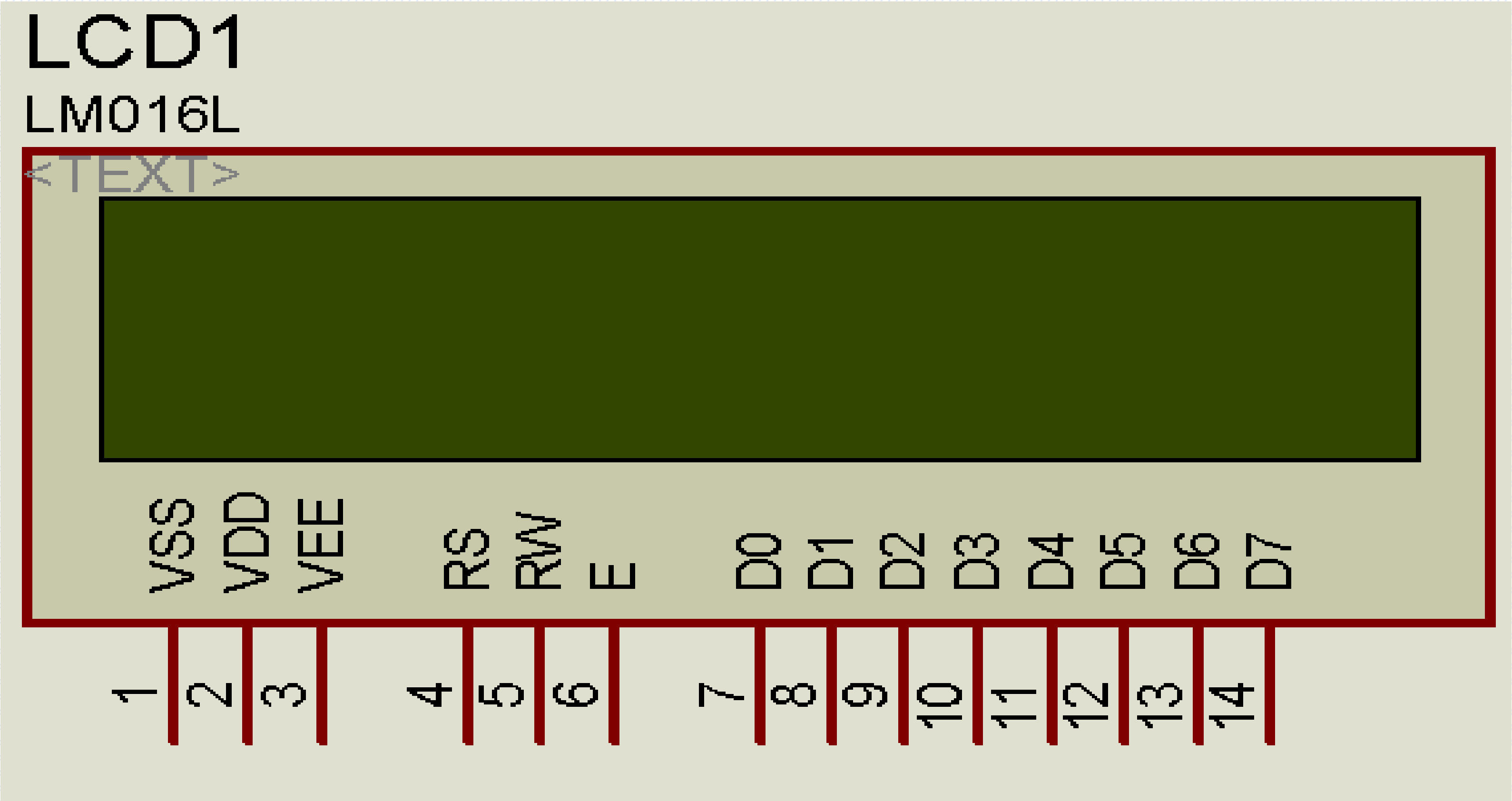
دانشکده برق و رباتیک
دانشگاه صنعتی شاهرود

حسین خسروی

LCD Interfacing

- LCD is finding widespread use replacing LEDs
  - The declining prices of LCD
  - The ability to display numbers, characters, and graphics
  - Incorporation of a refreshing controller into the LCD, thereby relieving the CPU of the task of refreshing the LCD
  - Ease of programming for characters and graphics

LCD Pin out



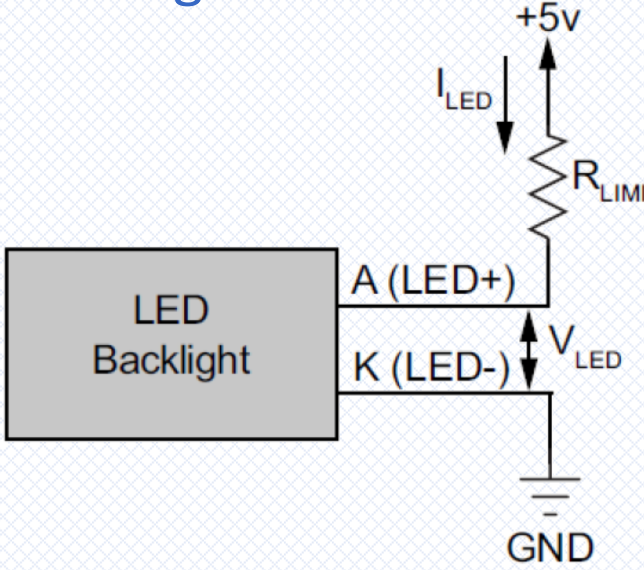
LCD1
LM016L
<TEXT>
VSS
VDD
VEE
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
1
2
3
4
5
6
7
8
9
10
11
12
13
14

LCD Pins Description

Pin	Symbol	Name	Description
1	VSS	Ground	0V (GND)
2	VDD	Power	Power supply for logic circuit and LCD (+4. 5V~+5. 5V)
3	V_{EE} or V_O	Contrast Supply	Bias voltage level to control contrast
4	RS	Register select	When RS= 1, data register is selected . When RS= 0, instruction register is selected .
5	RW	Read/Write	When RW= 1, read operation . When RW= 0, write operation .
6	E	Read Write enable	enable signal to read or write the data
7-10	DB0-DB3	Data bus 0-3	in 8-bit bus mode, used as low order bi-directional data bus. During 4-bit bus mode, open these pins
11-14	DB4-DB7	Data bus 4-7	in 8-bit bus mode, used as high order bi-directional data bus. In case of 4-bit bus mode, used as both high and low order. DB7 used for **Busy** Flag output

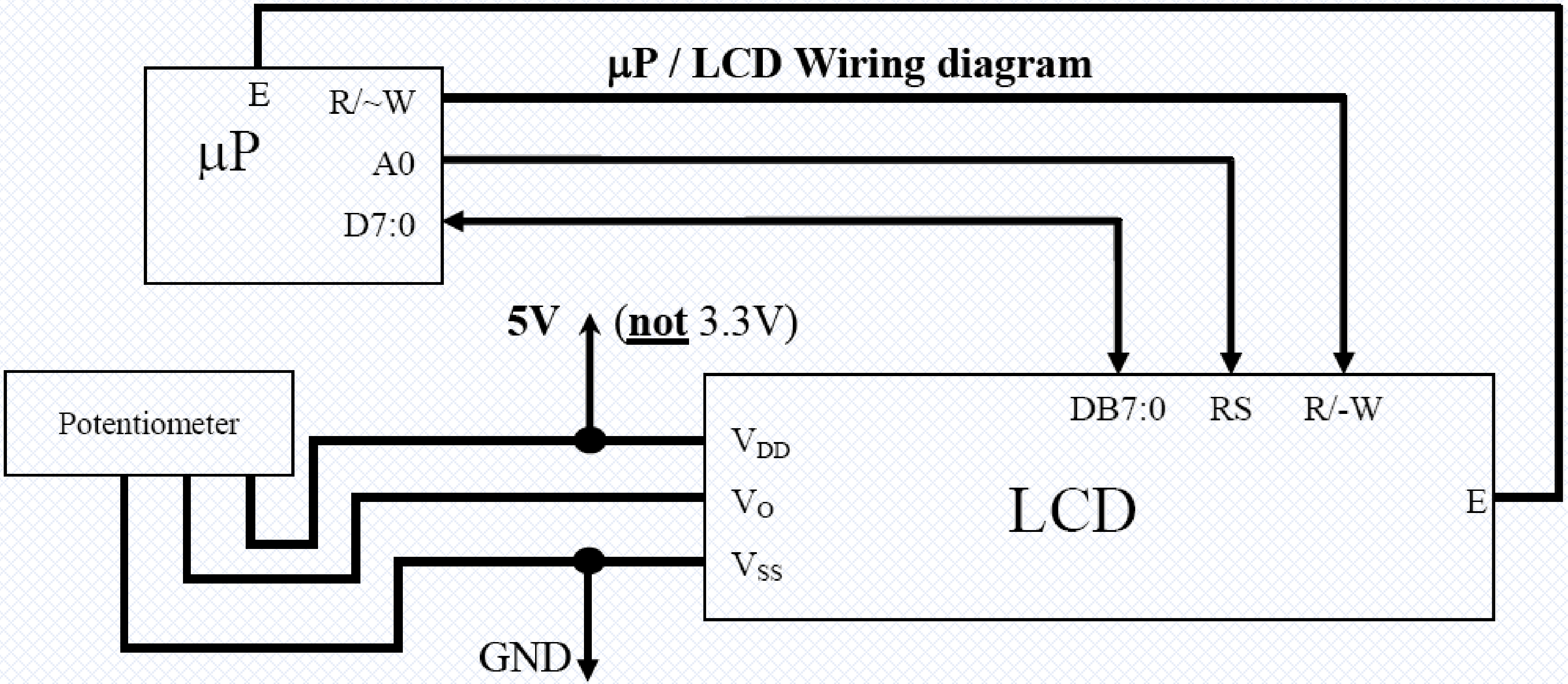
Contrast and Optional Backlight Information

- Optional Pins
  - 15 A (LED +) Optional: LED Backlight Anode
  - 16 K (LED -) Optional: LED Backlight Cathode



- The optimal contrast for the LCD (V_O) is 3.3 - 3.7V, but this may vary with viewing angle, ambient temperature and per-LCD.
- Setting the backlight up is optional, but may increase the readability of the LCD and is pretty cool. The backlight on your LCD is one large green LED

Typical connection



- You can verify that your LCD works properly before connecting your LCD data pins.
  - Give power to the device and twist the potentiometer one way or the other until you see black lines appear.

LCD Initialization

- The module powers up in 8-bit mode. Additional commands are required to put the module into 4-bit mode
- Now we are going to continue using it in 8-bit mode.

- <Wait 40us or till BF=0> BF = Busy Flag (D7 of data)
- **(Two lines LCD with 5×7 matrix) [DB=0x38]**
- **<Wait 40us or till BF=0>**
- **(Display on; cursor on; blink on) [DB=0x0F]**
- **<Wait 40us or till BF=0>**
- **(Clear screen; cursor home) [DB=0x01]**
- **<Wait 1.64ms or till BF=0>**

- Initialization Complete

Other useful Commands

- **[DB=0x06]**
  - Increment cursor to the right when writing; don’t shift screen
- <Wait 40us or till BF=0>
- **[DB=0x08]**
  - (Display off; cursor off; blink off)
- <Wait 40us or till BF=0>

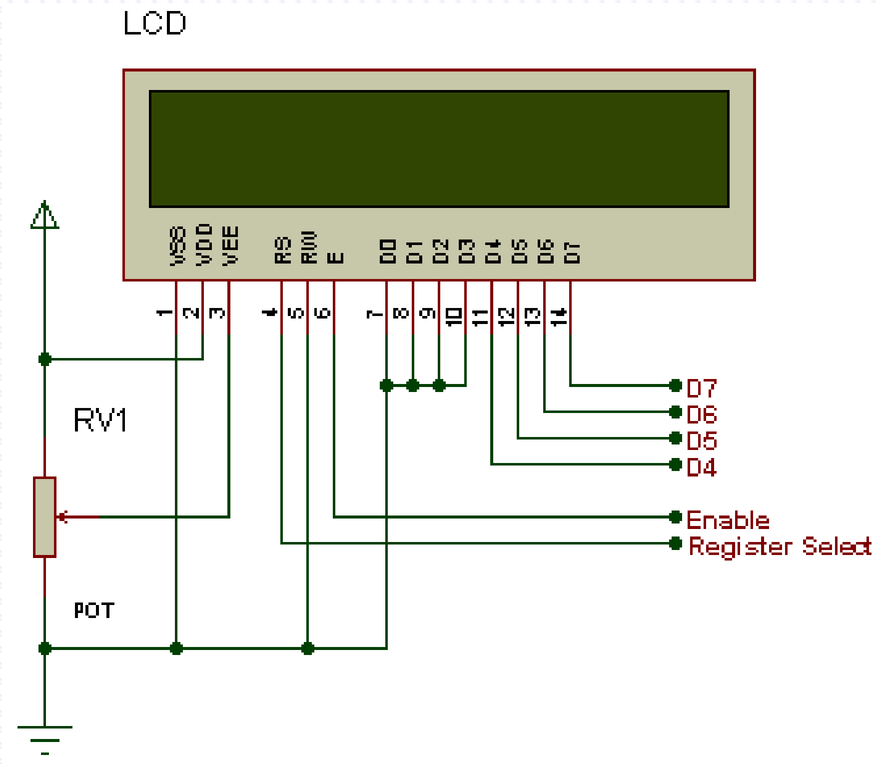
MORE LCD COMMANDS

Command	Code	Delay
Clear Display, Cursor to Home	**\$01**	1.65ms
Cursor to Home	**\$02**	1.65ms
Entry Mode:		
Cursor Decrement, Shift off	**\$04**	40μs
Cursor Decrement, Shift on	**\$05**	40μs
Cursor Increment, Shift off	**\$06**	40μs
Cursor Increment, Shift on	**\$07**	40μs
Display Control:		
Display, Cursor, and Cursor Blink off	**\$08**	40μs
Display on, Cursor and Cursor Blink off	**\$0C**	40μs
Display and Cursor on, Cursor Blink off	**\$0E**	40μs
Display, Cursor, and Cursor Blink on	**\$0F**	40μs
Cursor / Display Shift: (nondestructive move)		
Cursor shift left	**\$10**	40μs
Cursor shift right	**\$14**	40μs
Display shift left	**\$18**	40μs
Display shift right	**\$1C**	40μs
Display Function (2 rows for 4-bit data; big)	**\$2C**	40μs
Display Function (2 rows for 4-bit data; small))	**\$28**	40μs
Display Function (1 row for 4-bit data; big)	**\$24**	40μs
Display Function (1 row for 4-bit data; small)	**\$20**	40μs
Display Function (2 rows for 8-bit data; big)	**\$3C**	40μs
Display Function (2 rows for 8-bit data; small)	**\$38**	40μs
Display Function (1 row for 8-bit data; big)	**\$34**	40μs
Display Function (1 row for 8-bit data; small)	**\$30**	40μs
Move cursor to beginning of second row	**\$C0**	40μs
Character Generator RAM Address set	**\$40-\$7F**	40μs
Display RAM Address set	**\$80-\$FF**	40μs

For more detail see:
LCD_Notes_8-bit.pdf

Run sample program:
LCD_Functions_8bit

LCD 4-bit data mode



Using LCD in CodeVision

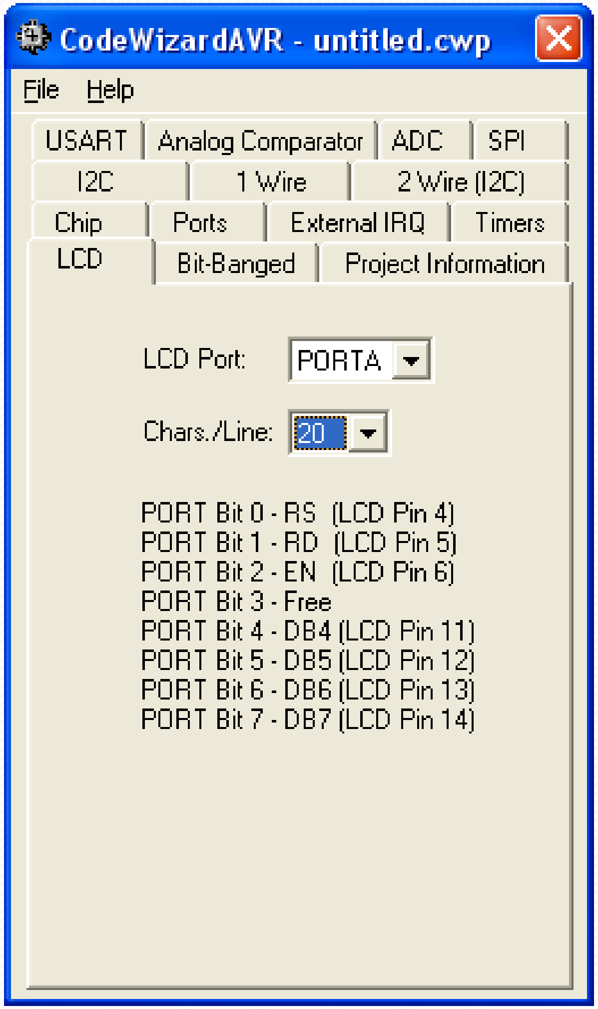
Initialization

```
/* the LCD module is connected to PORTC */
#asm
. equ __lcd_port=0x15
#endasm
/* now you can include the LCD Functions */
#include <lcd. h>
Lcd_init(16);
```

PORT ADDRESS

$1B ($3B)	PORTA
$1A ($3A)	DDRA
$19 ($39)	PINA
$18 ($38)	PORTB
$17 ($37)	DDRB
$16 ($36)	PINB
$15 ($35)	PORTC
$14 ($34)	DDRC
$13 ($33)	PINC
$12 ($32)	PORTD
$11 ($31)	DDRD
$10 ($30)	PIND

LCD Configuration With CodeWizard



lcd.h – High Level

```
unsigned char lcd_init(unsigned char lcd_columns)
void lcd_clear(void)
void lcd_gotoxy(unsigned char x, unsigned char y)
void lcd_putchar(char c)
void lcd_puts(char *str)
void lcd_putsf(char flash *str)
```

unsigned char lcd_init(unsigned char lcd_columns)

Example:

lcd_init(16)

```
void lcd_clear(void)
```

صفحه ال سی دی را پاک می کند

```
void lcd_gotoxy(unsigned char x, unsigned char y)
```

مکان نما را به سطر و ستون دلخواه می برد

```
lcd_gotoxy(4,2)
```

```
void lcd_putchar(char c)
  lcd_putchar('a');


void lcd_putsf(char flash *str)
  lcd_putsf("Hello World");
```

```
void lcd_puts(char *str)
    sprintf(buffer, “tempreture= %d”, temp);
    lcd_puts(buffer);
```

Example - 7

- در سطر اول ستون پنجم کاراکتر ‘a’ و در سطر دوم ستون اول عبارت “CodeVisionAVR” را بر روی یک LCD 2x16 نمایش دهید.

```
#include <mega16.h>

#asm
   .equ __lcd_port=0x1B ;PORTA
#endasm
#include <lcd.h>
#include <delay.h>
void main(void)
{

lcd_init(16);

while (1)
      {
lcd_clear();
lcd_gotoxy(5,0);
lcd_putchar('a');
lcd_gotoxy(0,1);
lcd_putsf("CodeVisionAVR");
delay_ms(200);
      };
}
```

Example-8

- تابعی بنویسید که بصورت روان کلمه CodeVision را بر روی LCD نمایش دهد.

```
#include <mega16.h>
// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x1B ;PORTA
#endasm
#include <lcd.h>
#include <delay.h>
void main(void)
{
int i;
lcd_init(16);

while (1)
      {
      for (i=0;i<7;i++)
      {
      lcd_clear();
      lcd_gotoxy(i,0);
      lcd_putsf("CodeVision");
      delay_ms(400);
      }
      };
}
```

Keypad

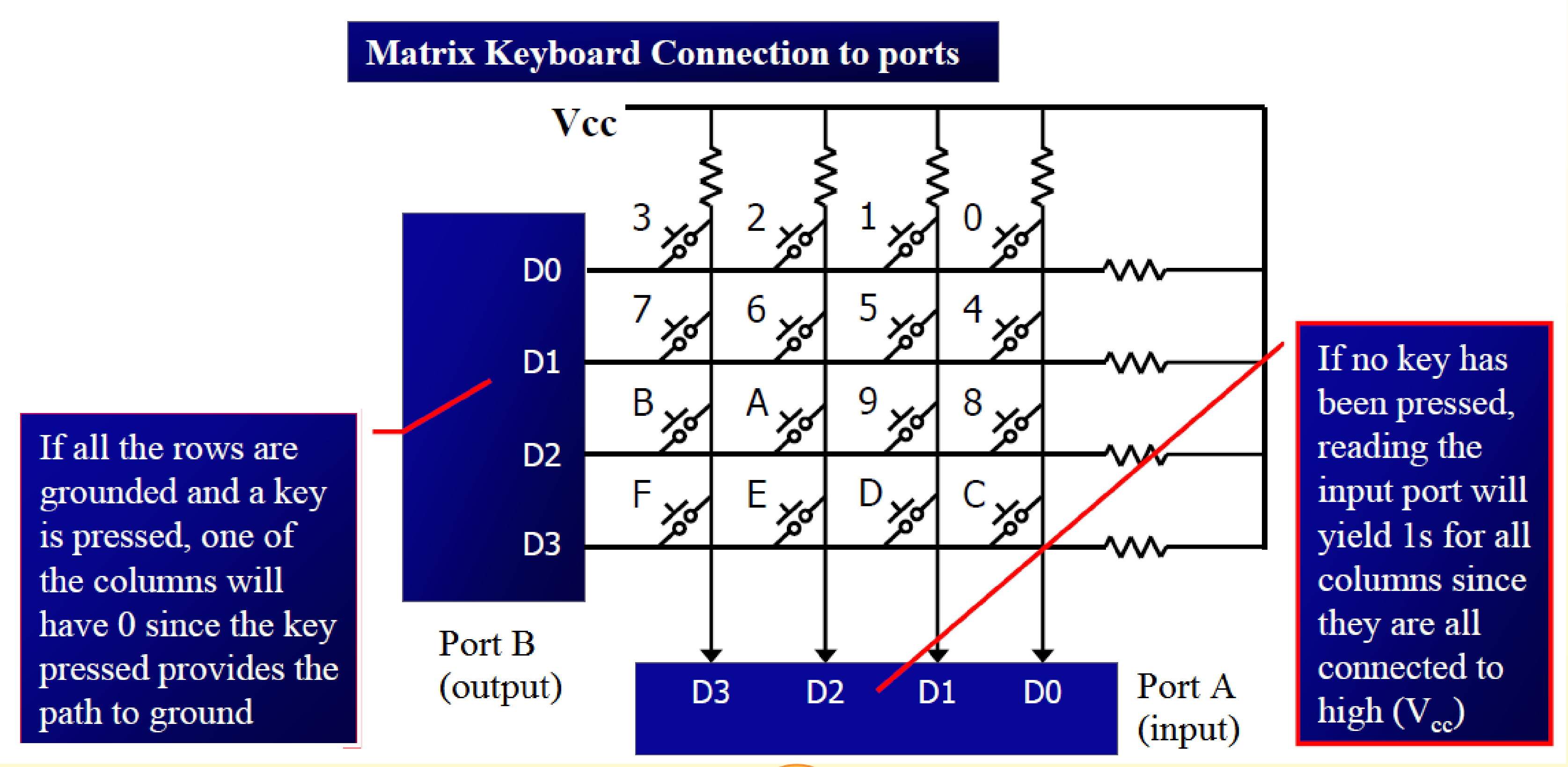
Keyboard Interfacing

- Keyboards are organized in a matrix of rows and columns
  - The CPU accesses both rows and columns through ports
    - Therefore, with two 8-bit ports, an 8 x 8 matrix of keys can be connected to a microprocessor
  - When a key is pressed, a row and a column make a contact
    - Otherwise, there is no connection between rows and columns

- In IBM PC keyboards, a single microcontroller takes care of hardware and software interfacing

Scanning and Identifying the Key

- A 4x4 matrix connected to two ports
  - The rows are connected to an output port and the columns are connected to an input port



Grounding Rows and Reading Columns

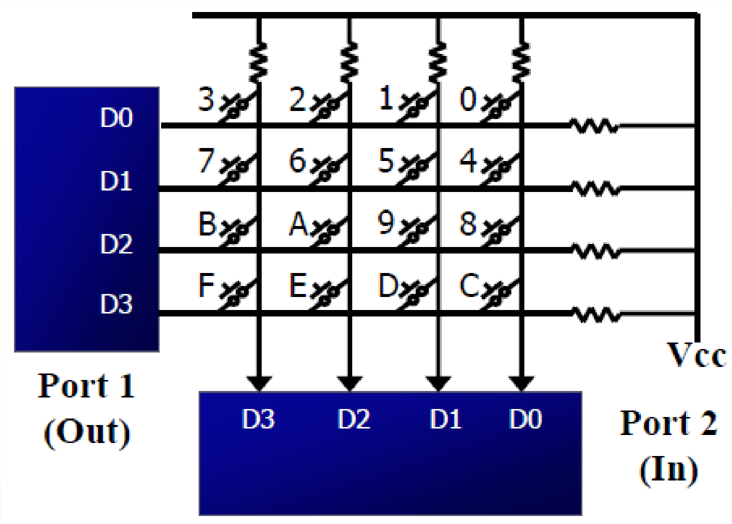
- It is the function of the microcontroller to scan the keyboard continuously to detect and identify the key pressed
- To detect a pressed key, the microcontroller grounds all rows by providing 0 to the output latch, then it reads the columns
  - If the data read from columns is D3 – D0 = 1111, no key has been pressed and the process continues till key press is detected
  - If one of the column bits has a zero, this means that a key press has occurred
    - For example, if D3 – D0 = 1101, this means that a key in the D1 column has been pressed
    - After detecting a key press, microcontroller will go through the process of identifying the key

Grounding Rows and Reading Columns (cnt'd)

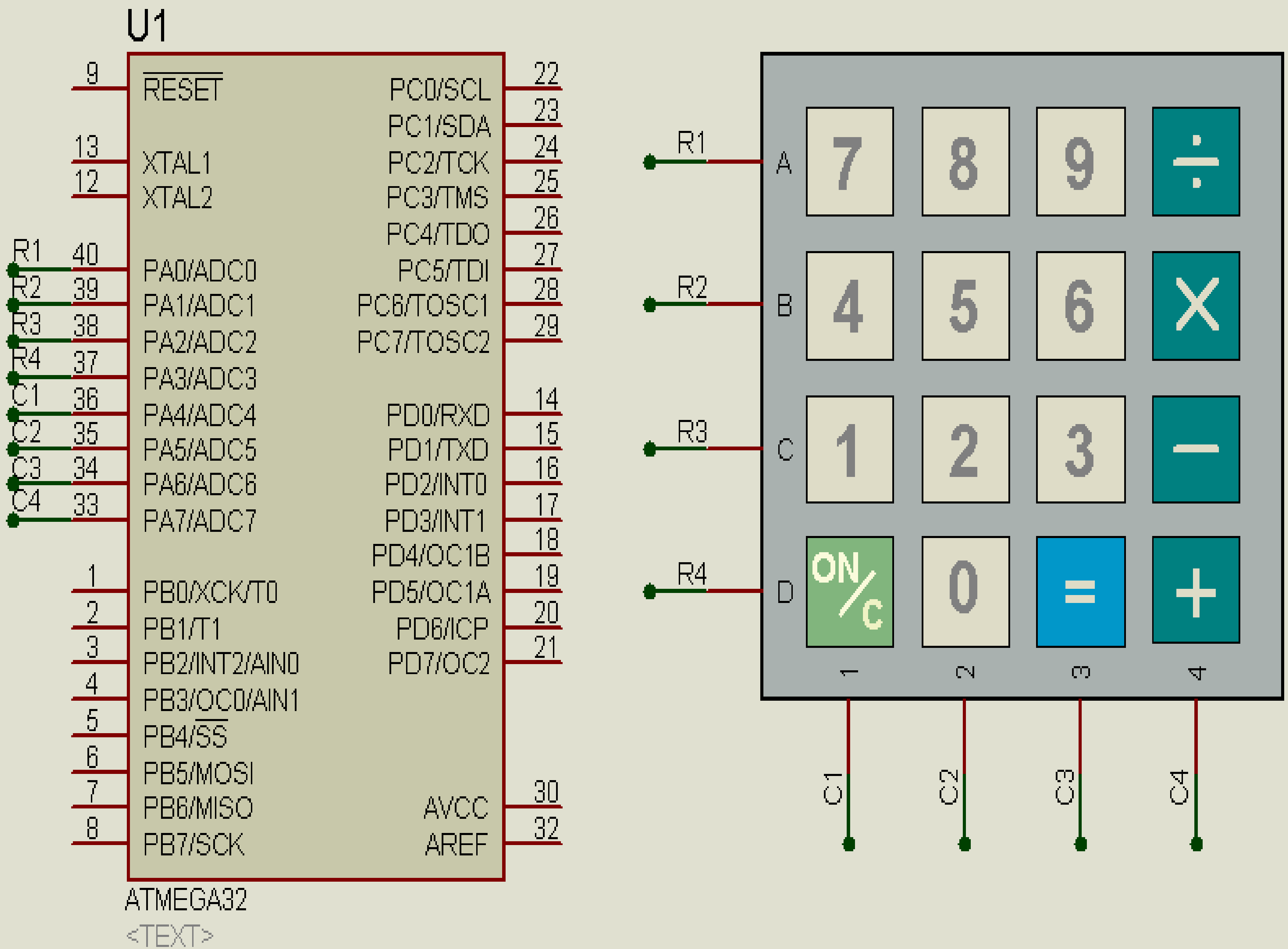
- Starting with the top row, the microcontroller grounds it by providing a low to row D0 only
  - It reads the columns, if the data read is all 1s, no key in that row is activated and the process is moved to the next row
- It grounds the next row, reads the columns, and checks for any zero
  - This process continues until the row is identified
- After identification of the row in which the key has been pressed
  - Find out which column the pressed key belongs to

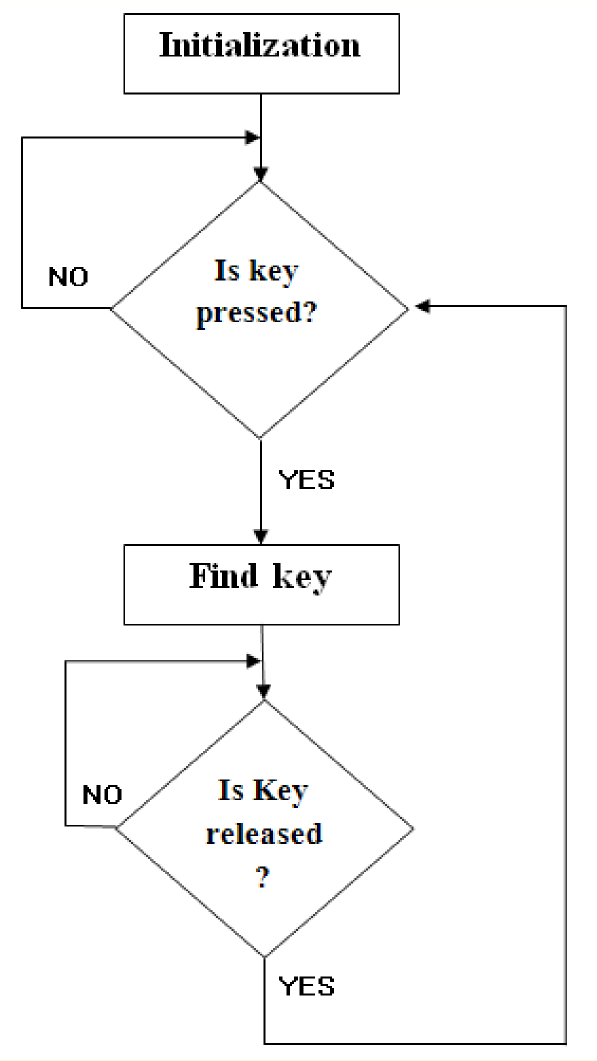
Grounding Rows and Reading Columns (cnt'd)

- **Example:** identify the row and column of the pressed key for each of the following.
- (a) D3 – D0 = 1110 for the row, D3 – D0 = 1011 for the column
- (b) D3 – D0 = 1101 for the row, D3 – D0 = 0111 for the column



- **Solution :**
- (a) The row belongs to D0 and the column belongs to D2; therefore, key number 2 was pressed.
- (b) The row belongs to D1 and the column belongs to D3; therefore, key number 7 was pressed.


U1
RESET
XTAL1
XTAL2
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PB0/XCK/T0
PB1/T1
PB2/INT2/AIN0
PB3/OC0/AIN1
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PC0/SCL
PC1/SDA
PC2/TCK
PC3/TMS
PC4/TDO
PC5/TDI
PC6/TOSC1
PC7/TOSC2
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP
PD7/OC2
AVCC
AREF
ATMEGA32
<TEXT>
R1
R2
R3
R4
C1
C2
C3
C4
A
B
C
D
7 8 9 ÷
4 5 6 X
1 2 3 −
ON/C 0 = +
1 2 3 4


Initialization
Is key pressed?
NO
YES
Find key
Is Key released ?
NO
YES

Run sample code (Keypad)