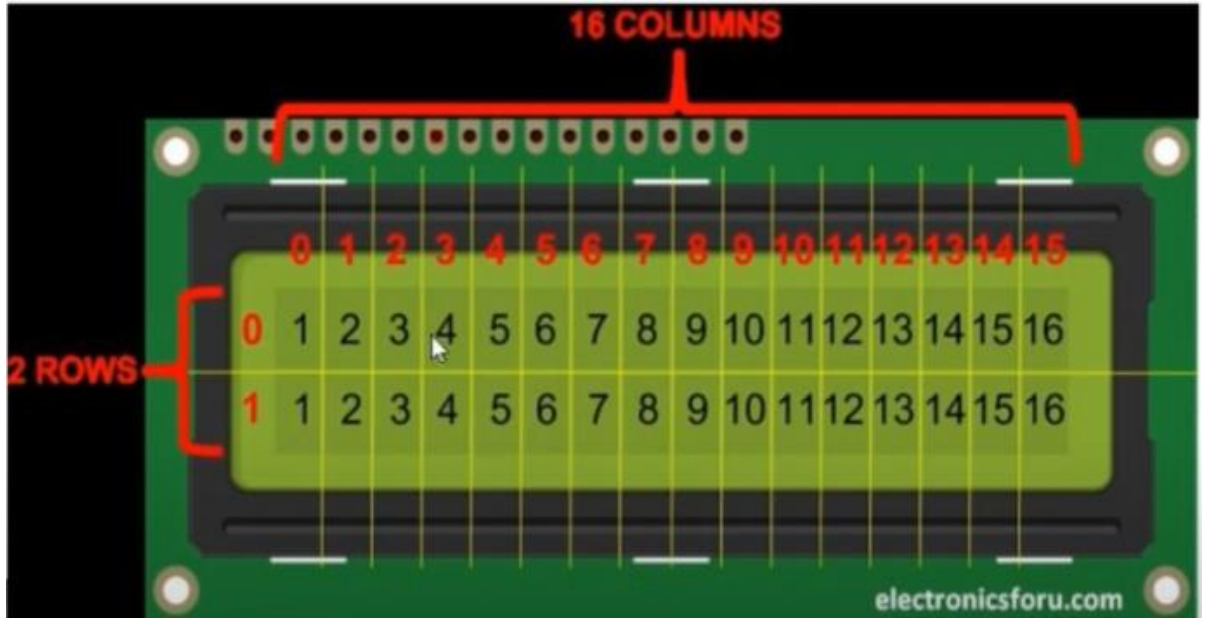
24-02-2025

LCD Programming

LCDs (Liquid Crystal Display)

- 16X2 LCD: capable of displaying 2 lines, each having 16 Characters
  - ➔ 16×2 LCD is a 16 pin device
  - Out of 16 Pins,
    - 8 pins are data pins
    - 2 pins for power supply
    - 1 for contrast setting of the display
    - 1 is Register Select
    - 1 is Read/Write select
    - 1 Enable
    - 2 for the background lightning LED

Control lines/bits



- It has two types of input lines: Data line and Control line
  - 8 data lines (8 bits)
  - 3 control lines: Register select, Read/Write and Enable.

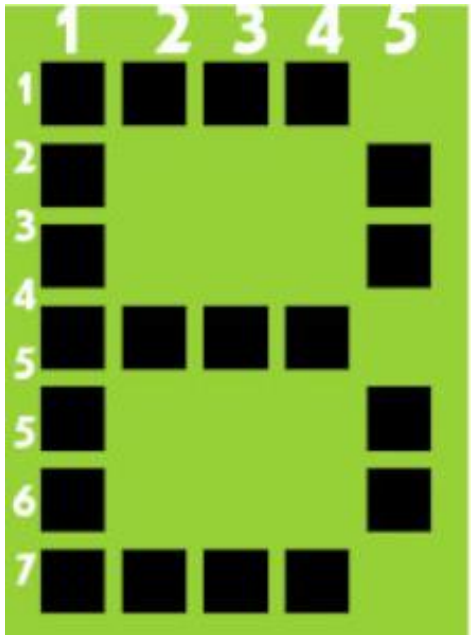
- **Data Bus:** 8-bit data bus ➔ D0-D7
- Can send the data/cmd to LCD in bytes. (data & cmds sent in same line)
- Can also send the data/cmd in chunks of 4-bit
- So, there are two modes of operation:
  - 8 bit mode and 4 bit mode
- The kit that we are using it is in 4 bit mode. i.e., D7-D4 are used

Register Select(RS):

- Two registers ➔ Data register and Command register.
- Any data that needs to be displayed on the LCD has to be written to the data register of LCD.
- Command can be issued to LCD by writing it to Command register of LCD.
- RS=0➔ LCD interprets the 8-bit info (in the D0-D7) as **Command** and writes the info into the **Command register** and performs the action as per the command.
- RS=1➔ LCD interprets the 8-bit info (in the D0-D7) as **data** and writes the info into the **data register**.

CXR

- Any character on LCD is displayed with a matrix of 5×8 or 5×7.
  - 5 ➔ No of columns
  - 7/8 ➔ No of rows
- Normally 5×7 matrix used to display a character & 8th is used to for cursor. (kinda like a blinking underscore)



Read/Write(RW):

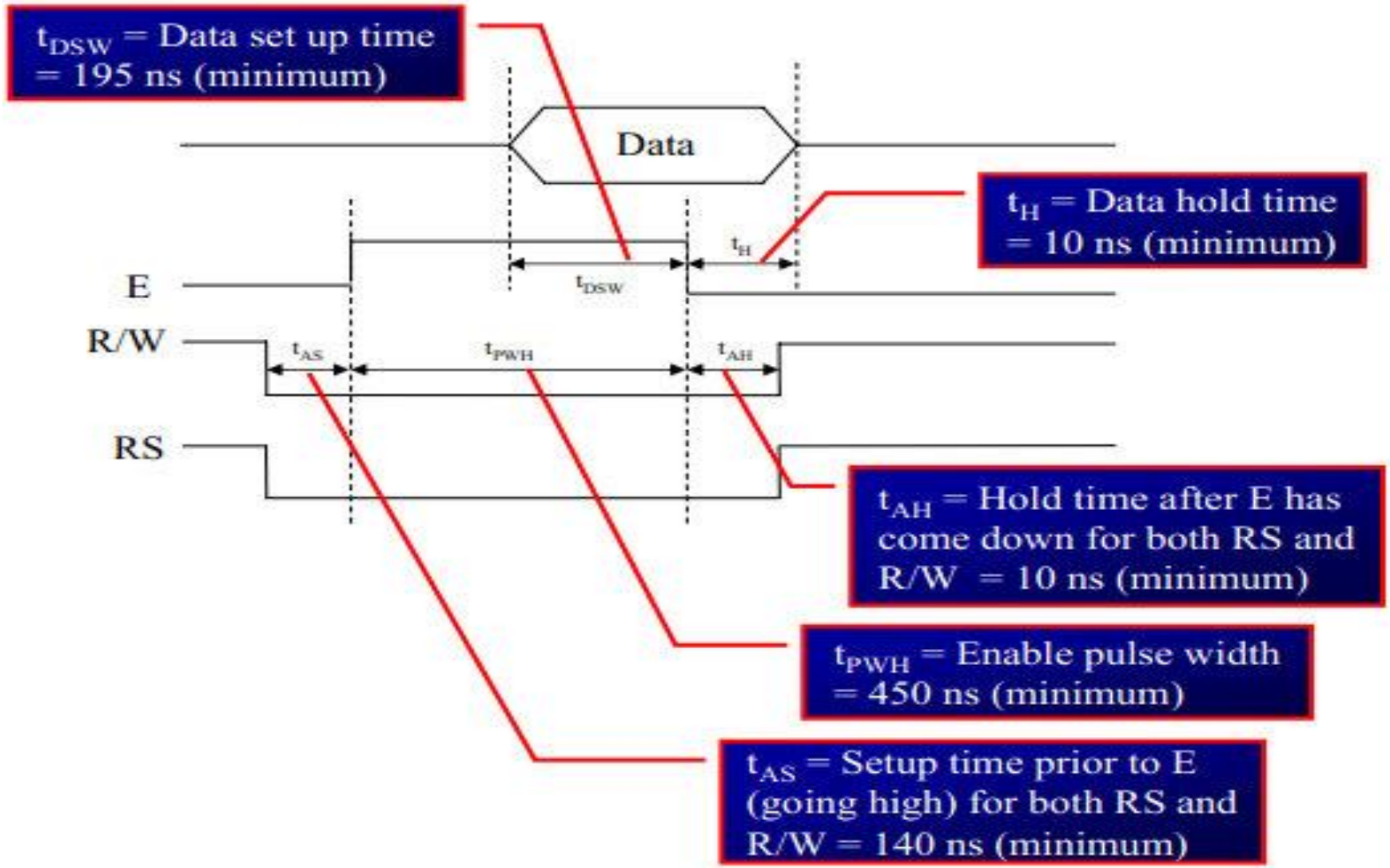
- This signal is used to write the data/cmd to LCD and read the busy flag of LCD.
- RW=1➔ Read operation is being performed
- RW=0➔ Write operation is being performed
- In our kit always RW=0, because this pin is grounded➔ Only writing into the LCD can take place
- from the registr the data is displayed

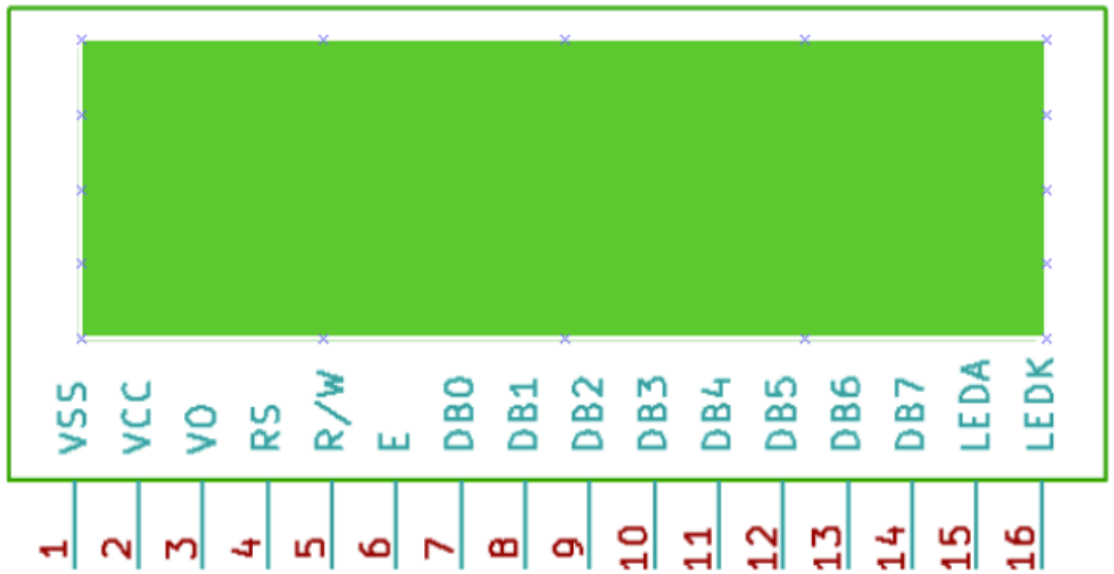
Enable(EN):

- Used to send the enable trigger to LCD.
- After sending the data/cmd, a HIGH-to-LOW (1➔0) (negative edge triggered) pulse needs to be sent on this to latch the info into the LCD register and triggers the LCD to act accordingly.

strobing signal

25-02-25


t_{DSW} = Data set up time = 195 ns (minimum)
Data
t_H = Data hold time = 10 ns (minimum)
E
R/W
RS
t_{DSW}
t_H
t_{AS}
t_{PWH}
t_{AH}
t_{AH} = Hold time after E has come down for both RS and R/W = 10 ns (minimum)
t_{PWH} = Enable pulse width = 450 ns (minimum)
t_{AS} = Setup time prior to E (going high) for both RS and R/W = 140 ns (minimum)



Pin No	Symbol	Pin Function
1	VSS	Ground
2	VCC	+5v
3	VEE	Contrast adjustment (VO)
4	RS	Register Select. 0:Command, 1: Data
5	R/W	Read/Write, R/W=0: Write & R/W=1: Read
6	EN	Enable. Falling edge triggered
7	D0	Data Bit 0 (Not used in 4-bit operation)
8	D1	Data Bit 1 (Not used in 4-bit operation)
9	D2	Data Bit 2 (Not used in 4-bit operation)
10	D3	Data Bit 3 (Not used in 4-bit operation)
11	D4	Data Bit 4
12	D5	Data Bit 5
13	D6	Data Bit 6
14	D7	Data Bit 7/Busy Flag
15	A/LED+	Back-light Anode(+)
16	K/LED-	Back-Light Cathode(-)

4bit Mode

(MS half first, LS half second)

LCD display takes a time of 39-43µS to place a character or execute a command. Except for clearing display and to seek cursor to home position it takes 1.53ms to 1.64ms. Any attempt to send any data before this interval may lead to failure to read data or execution of the current data. So we have to give appropriate delay after each command.

Control and display commands

* Will be given

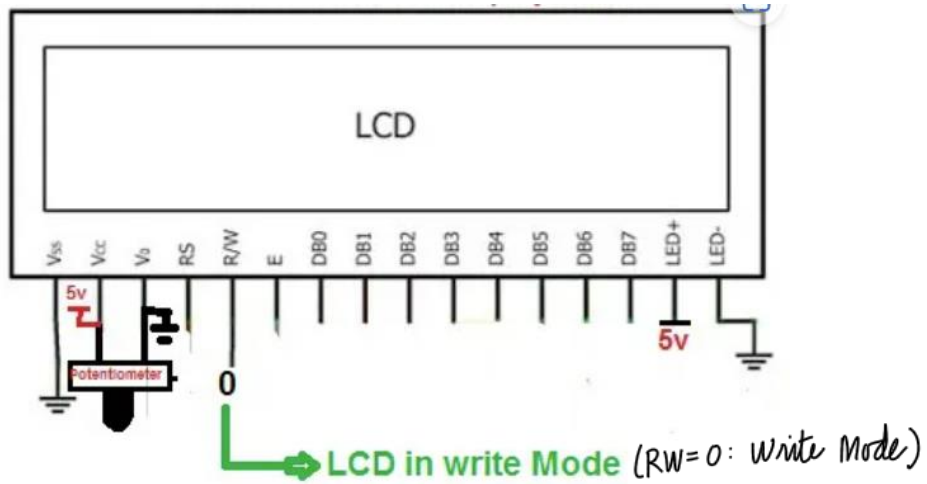
Sr.No.	Hex Code	Command to LCD instruction Register
1	01	Clear display screen
2	02	Return home (cursor goes back to the start)
3	04	Decrement cursor (shift cursor to left)
4	06	Increment cursor (shift cursor to right)
5	05	Shift display right
6	07	Shift display left

Sr.No.	Hex Code	Command to LCD instruction Register
7	08	Display off, cursor off
8	0A	Display off, cursor on
9	0C	Display on, cursor off
10	0E	Display on, cursor not blinking
11	0F	Display on, cursor blinking
12	10	Shift cursor position to left

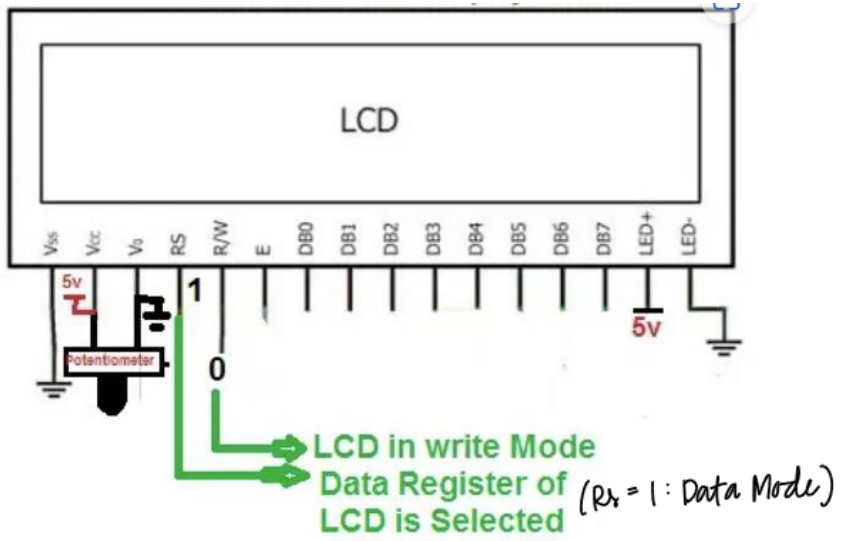
Sr.No.	Hex Code	Command to LCD instruction Register
13	14	Shift cursor position to right
14	18	Shift the entire display to the left
15	1C	Shift the entire display to the right
16	80	Force cursor to beginning (1st line)
17	C0	Force cursor to beginning (2nd line)

Lets write 8-bit data with 8-bit mode

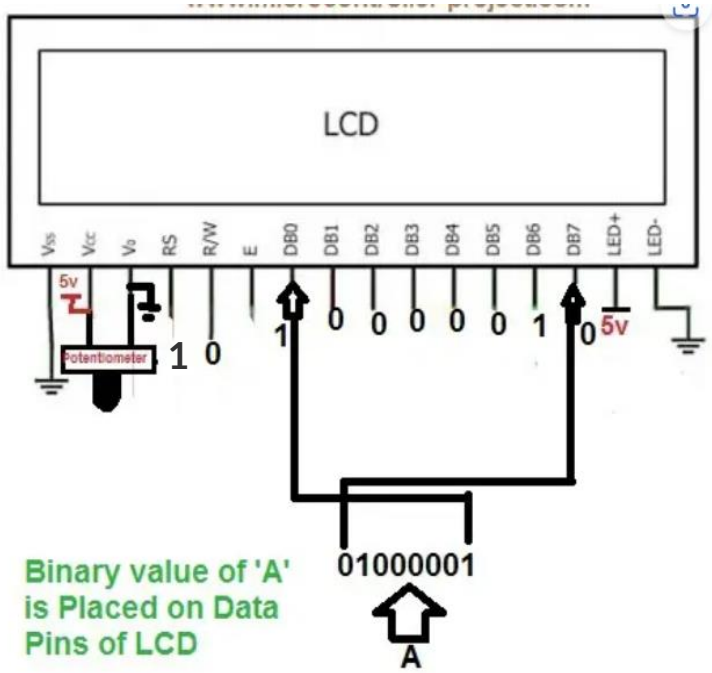
Step:1



Step:2

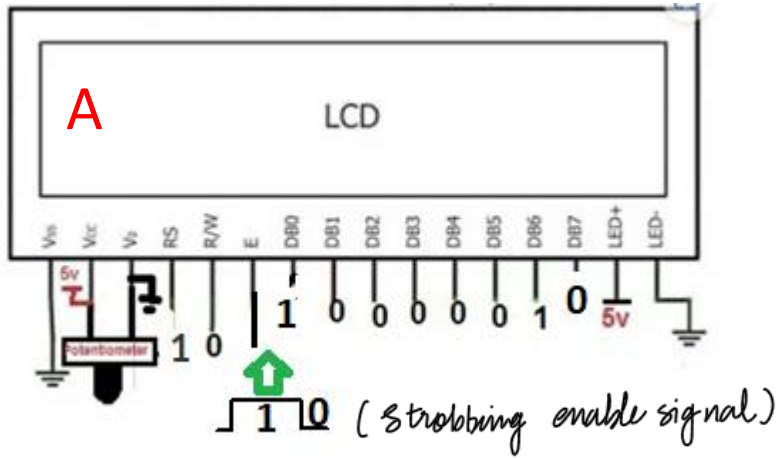


Step:3



ASCII value of '**A**' is **65**, **01000001**(Binary), **0x41**

Step:4

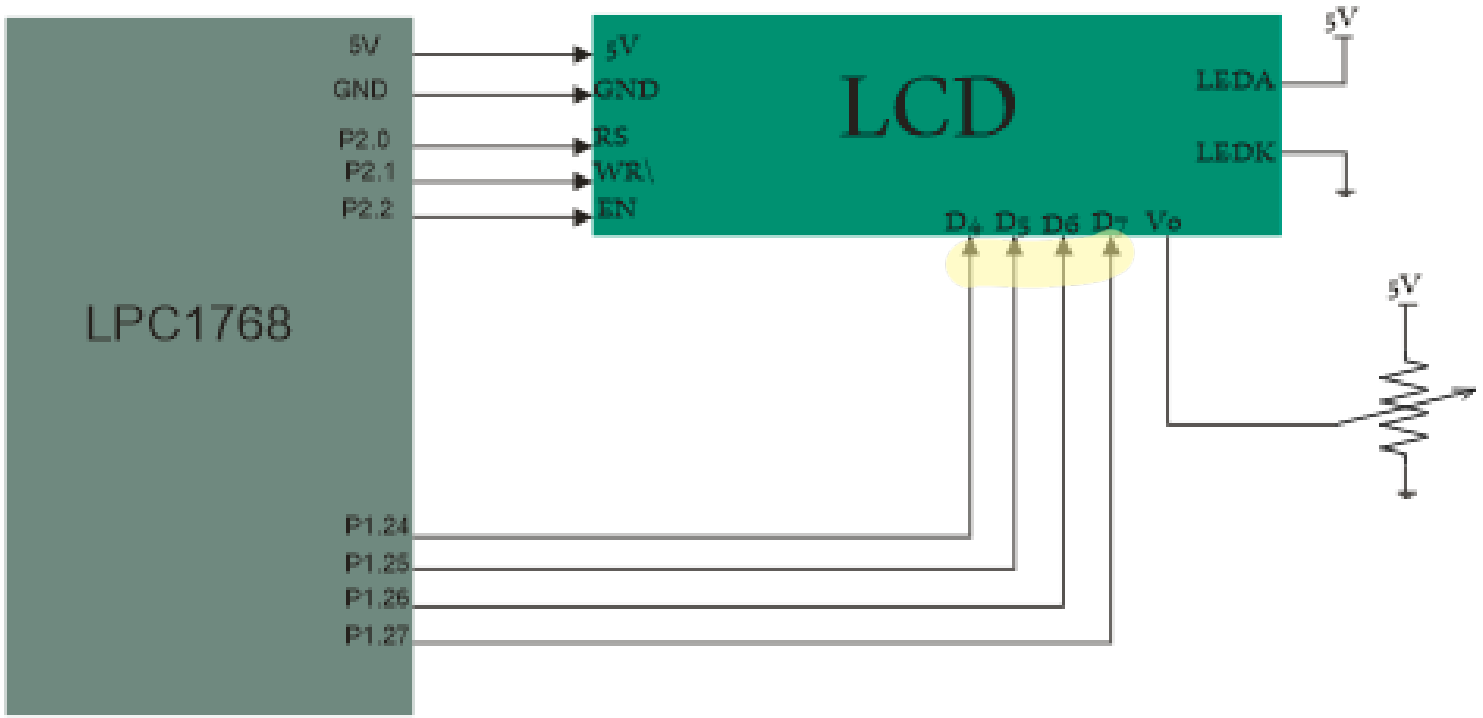


Make **en=1** and after some microseconds again make **en=0**.

- Our kit works in 4 bit mode.
- To send data in 4bit mode;
  - First put control signals (R/W=0; RS=0/1) in the control bus
  - Then put upper 4bit in the 4bit data bus connected to 4MSB pins of LCD display,
  - Then pulse the En pin once.
  - Next put the lower 4 bit in the data bus
  - And pulse the En pin again.

Schematic

Below schematic shows the minimum connection required for interfacing the LCD with the microcontroller. As we are interfacing the LCD in 4-bit mode, only the higher 4 data lines are used as data bus.



Initialization for 4-bit operation VIVA Q

The module powers up in 8-bit mode. The initial start-up instructions are sent in 8-bit mode, with the lower four bits (which are not connected) of each instruction as don't cares.

<POWER ON>

<Wait at least 15ms>

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

<Wait at least 4.1ms>

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

} 33

<Wait at least 100us>

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

<Wait 4.1ms>

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

} 32

After the fourth instruction shown above, which switches the module to 4-bit operation, the control bytes are sent on consecutive enable cycles (no delay is required between nibbles). The most significant nibble is sent first, followed immediately by the least significant nibble. See the next page for details.

33, 32 need to be sent as a command initially ⇒ switches to 4-bit op. mode

LCD Initialization

Before displaying characters on the LCD display, it must be configured first. To configure an LCD display, four command words must be sent to LCD. The commands are:

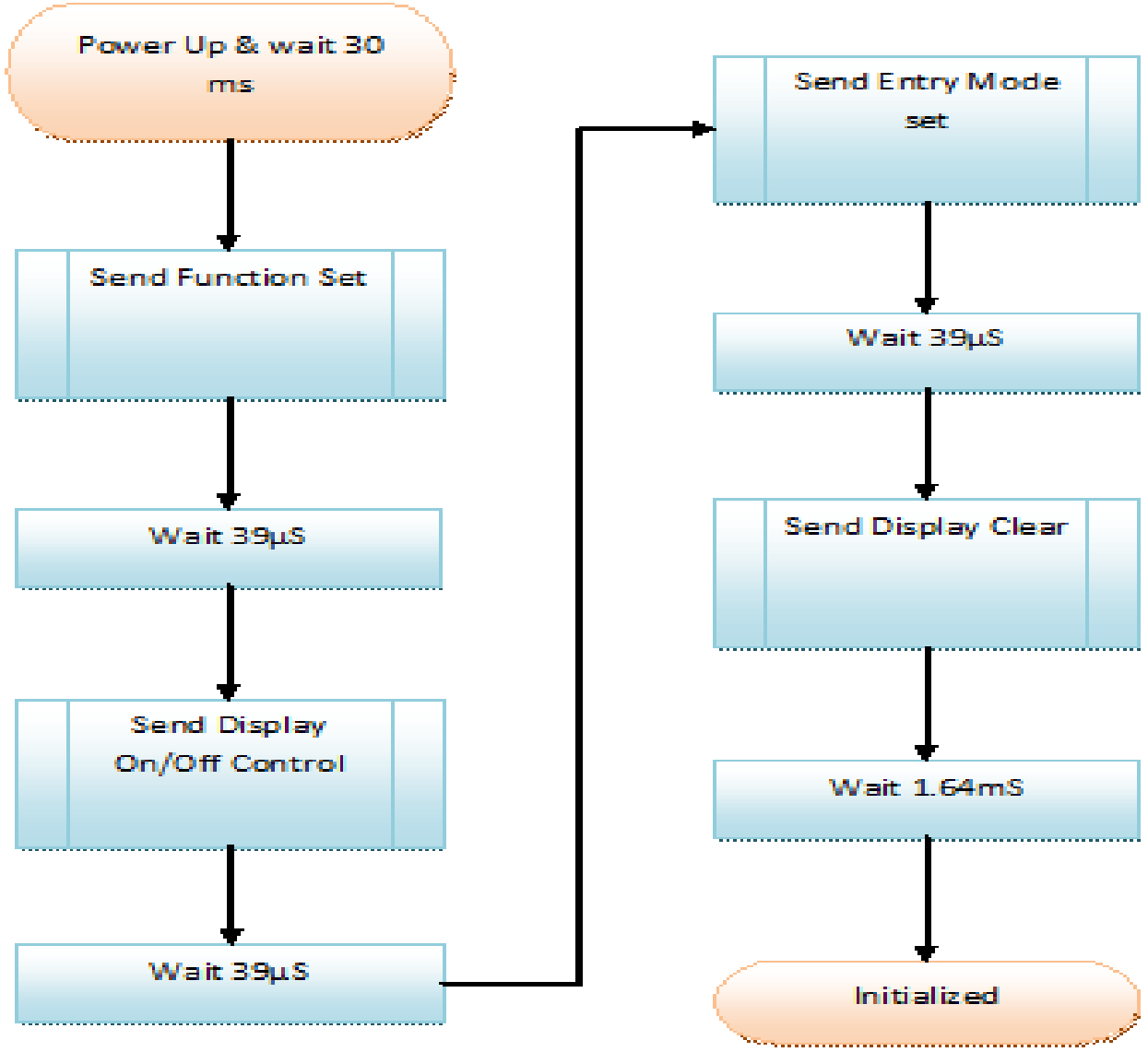
1. Function set
2. Display On/Off control
3. Entry mode set
4. Display Clear

By default set DL = 0

Instruction	Code										Description	Execution time**
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Function set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL), number of display line (N) and character font(F).	40uS
Display On/Off control	0	0	0	0	0	0	1	D	C	B	Sets On/Off of all display (D), cursor On/Off (C) and blink of cursor position character (B).	40uS
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction (I/D), specifies to shift the display (S). These operations are performed during data read/write.	40uS
Clear display	0	0	0	0	0	0	0	0	0	1	Clears display and returns cursor to the home position (address 0).	1.64mS

Bit names

Bit	Settings	
I/D	0 = Decrement cursor position	1 = Increment cursor position
S	0 = No display shift	1 = Display shift
D	0 = Display off	1 = Display on
C	0 = Cursor off	1 = Cursor on
B	0 = Cursor blink off	1 = Cursor blink on
S/C	0 = Move cursor	1 = Shift display
R/L	0 = Shift left	1 = Shift right
DL	0 = 4-bit interface	1 = 8-bit interface
N	0 = 1/8 or 1/11 Duty (1 line)	1 = 1/16 Duty (2 lines)
F	0 = 5x7 dots	1 = 5x10 dots
BF	0 = Can accept instruction	1 = Internal operation in progress



LCD Initialization Flow Chart

Before function set, we need to configure in 4 bit mode

(33,32)

LCD Interfacing (Video)
https://www.youtube.com/watch?v=WocACC9xZbI

Websites for LCD:
https://mil.ufl.edu/3744/docs/lcdmanual/commands.html
https://msoe.us/taylor/tutorial/ce2800/lcddisplay
https://www.exploreembedded.com/wiki/LPC1768:_Lcd_4bit
https://www.elprocus.com/lcd-16x2-pin-configuration-and-its-working/
https://openlabpro.com/guide/16x2-character-lcd-interfacing-using-lpc1768/

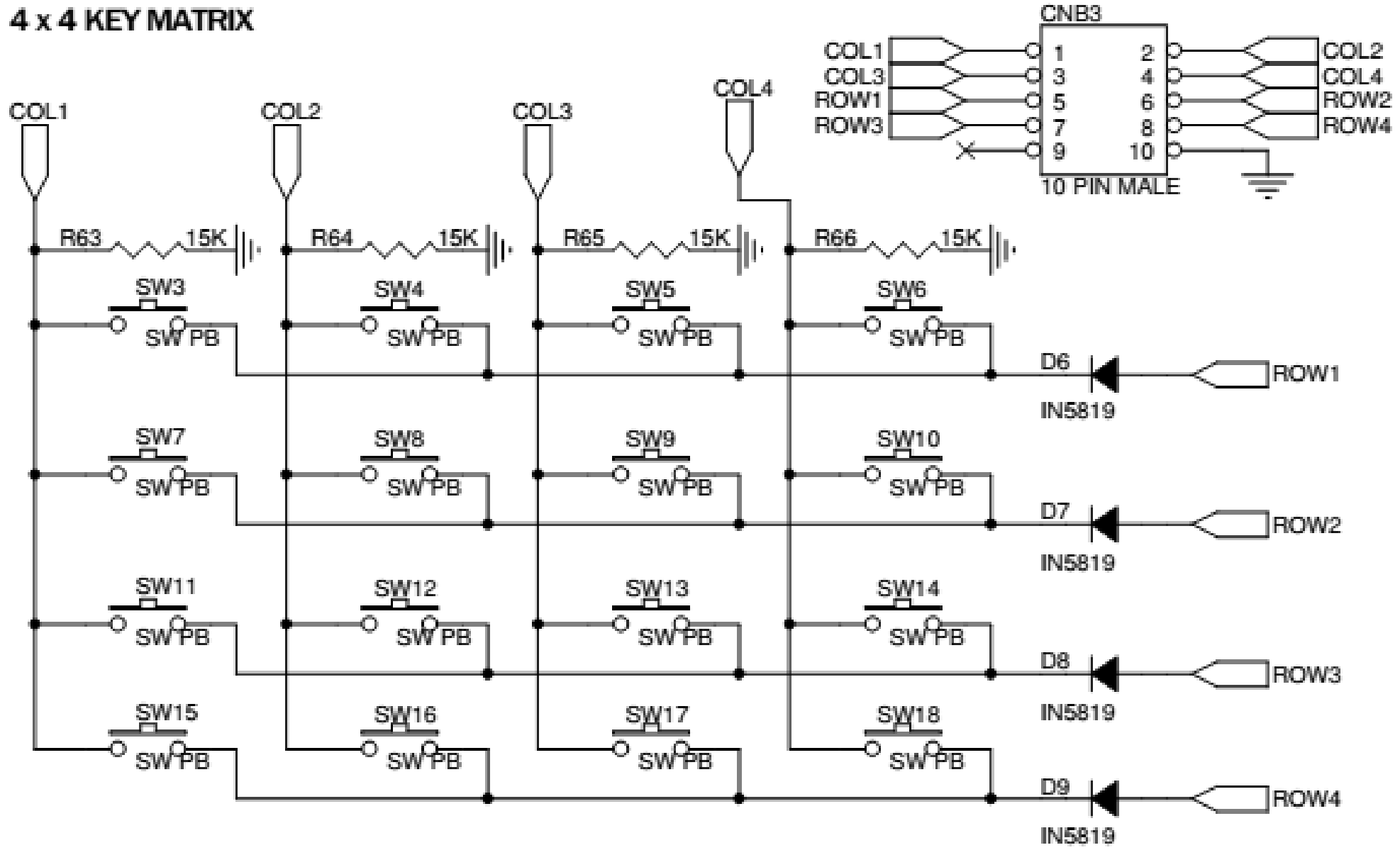
```
char msg3[11] = {"MIT"};
char msg4[16] = {"Dept. of CSE:"};
int temp1=0, temp2=0;
int main(void){

    LPC_GPIO0->FIODIR |= 0x0F<<23 | 1<<27 | 1<<28;
    clear_ports();                          (23:26 → D0:D7, 27:RS,
    delay_lcd(3200); //Random Value          ↳ GND          28:EN)

    lcd_init();
    lcd_com(0x80);          // First line First char
    delay_lcd(800);
    lcd_ptr(&msg3[0]);

    lcd_com(0xC0);          // Second Line First char
    delay_lcd(800);          } insane
    lcd_ptr(&msg4[0]);
```

```
void clear_ports{
        LPC_GPIO->FIOCLR = 0x0F<<23;    // Clear data line
        LPC_GPIO->FIOCLR = 1<<27;       // Clear RS
        LPC_GPIO->FIOCLR = 1<<28;       // Clear EN
}
void lcd_init(){

        lcd_comm(0x33);     //4-bit mode
        delay_lcd(800);

        lcd_comm(0x32);
        delay_lcd(800);

        lcd_comm(0x28);     // Function set
        delay_lcd(800);

        lcd_comm(0x0C);     // disp on cursor off
        delay_lcd(800);

        lc_comm(0x06);      //entry mode set increment
        delay_lcd(800);       cursor right

        lcd_comm(0x01);     // display clear
        delay_lcd(10000);
}
```

```
void lcd_comm(int temp1){

    temp2 = temp1 & 0xF0;        //MSB
    temp2 = temp2 << 19;  //Shift bp4 to 23-26
    write_cmd();
    delay_lcd(30000);
    temp2 = temp1 &0x0F;
    temp2 = temp2 << 23;         //same w bottom half
    write_cmd()
    delay_lcd(30000);
    return;
}

void write_cmd(){
        LPC_GPIO->FIOPIN = temp2;
        LPC_GPIO->FIOCLR = 1<<27;
        LPC_GPIO->FIOSET = 1<<28;   // EN=1
        delay_lcd(25);
        LPC_GPIO->FIOCLR = 1<<28;     //EN=0
        return;
}

void lcd_puts(unsigned char *buf1){
    int i=0;
    while(buf1[i] != '\0');
        temp1 = buf1[i];
        lcd_data();
        delay_lcd(800);
        i++;
```

```
if (i==16){
        lcd_com (0xc0);
        delay_lcd (800);
}
}
```

4 x 4 KEY MATRIX

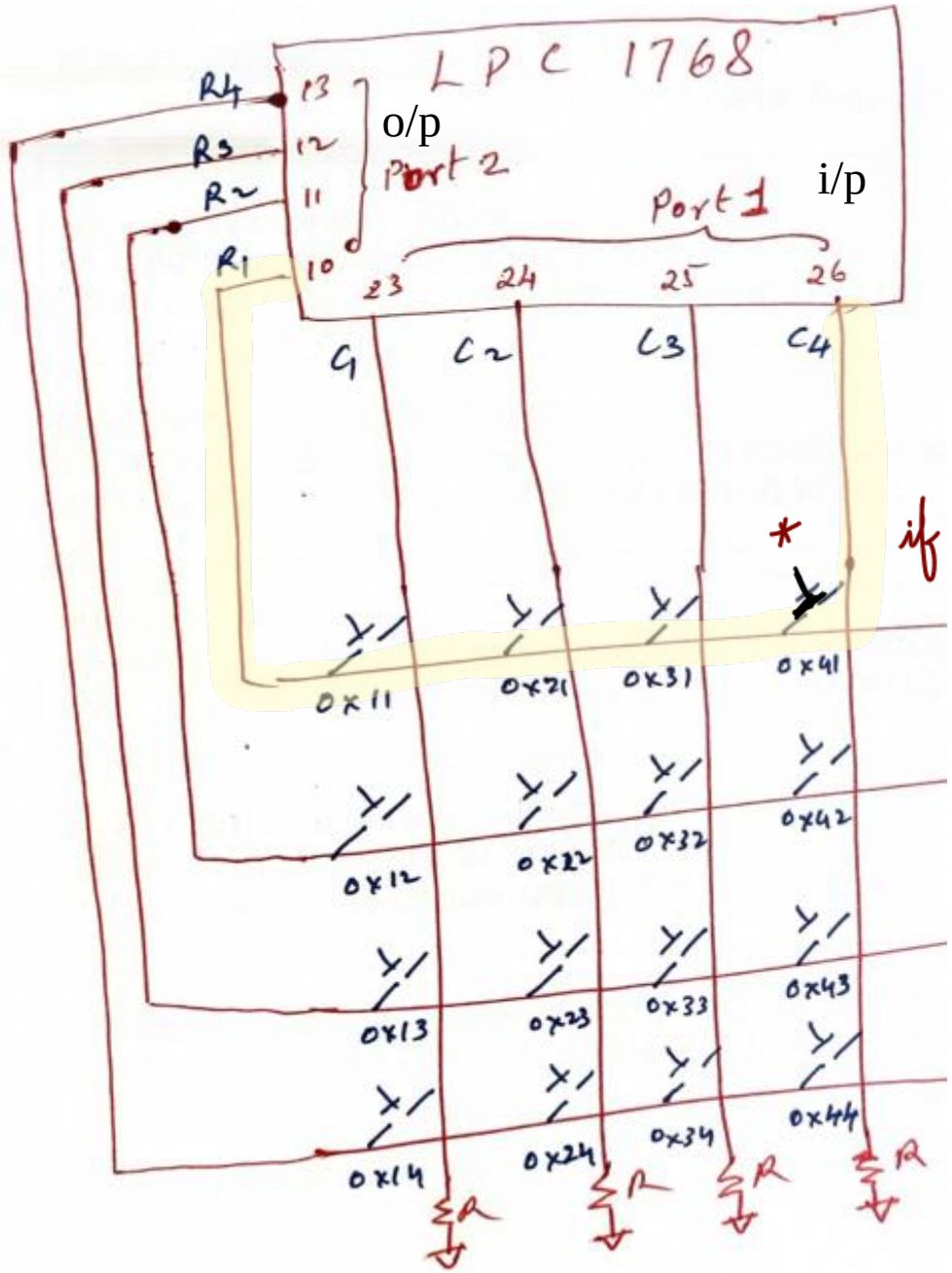


C1 to C4 -> P1.23 to P1.26. R1 to R4 -> P2.10 to P2.13

The switches SW3 to SW18 are organized as 4 rows X 4 columns matrix. One end of all the switches are configured as columns. The other end of the matrix configured as rows.

- A row line will be always an output from the controller.
- Column lines are pulled to ground.
- A high level sent from the row will appear at column end if the switch is pressed.

o 1 row is made high at a time.
(otherwise won't be able to determine which row)



Col, Row

* if pressed

3 grounded

```
unsigned char ASCII_CODE[16] = {'0', '1', '2', ... 'F'};

unsigned char SCAN_CODE[16] = {0x11, 0x21, 0x41, 0x81, .... 0x88};

int main(void){

    LPC_GPIO2->FIODIR |= 0x00003C00;     o/p rows  P2.10 - P2.13
    LPC_GPIO1->FIODIR &= 0xF87FFFFF;     i/p cols  P1.23 to P1.26

    lcd_init();
    lcd_comdata(0x80, 0);
    delay_lcd(800);
    lcd_puts(&Msg1[0]);     // display

    while(1){
        while(1){
            for(row=1; row<5; row++){
                if(row==1)
                    var1 = 0x00000400;  // P2.10 =1
                else if (row==2)

                else if (row==3)

                else if (row==4)

                LPC_GPIO2->FIOCLR = 0x00003C00;
                LPC_GPIO2->FIOSET = var1;
                flag = 0;
                scan();
                if (flag==1) break;
```

```
                }
            }

            if (key == SCAN_CODE[i])
            {
                key = ASCII_CODE[i]
                break;
            }
        }
        lcd_comdata(0xC0,0);
        delay_lcd(800);
        lcd_putr(key);
}
void scan(void){
        temp3 = LPC_GPIO1->FIOPIN
        temp3 &= 0x07800000;    extract P1.23 — P1.26 values
        if(temp3 != 0x00000000){
            flag = 1;
            temp3 >>= 19;       // shifted to upper nibble
            var1 >>= 10;        // shift to lower nibble
            key = temp3 | var1;     // get scan code
        }
}
```

12-14th April : LAB FINAL