DC Motor:

PWM module - extension of timer module
↳ can be used as timer

Single edge → control only ON time (falling edge)
double edge → both rising & falling edge

* Pulse period/rate
* Pulse width

PClk * MR0 value = time period

100%
1000

MR1 = 500, duty cycle = PClk × MR1/MR0 = 50%

* shadow registers for MR1 to MR6

Latch enable register PWMLER

↓ here user wants to change speed but a cycle is stored in shadow register

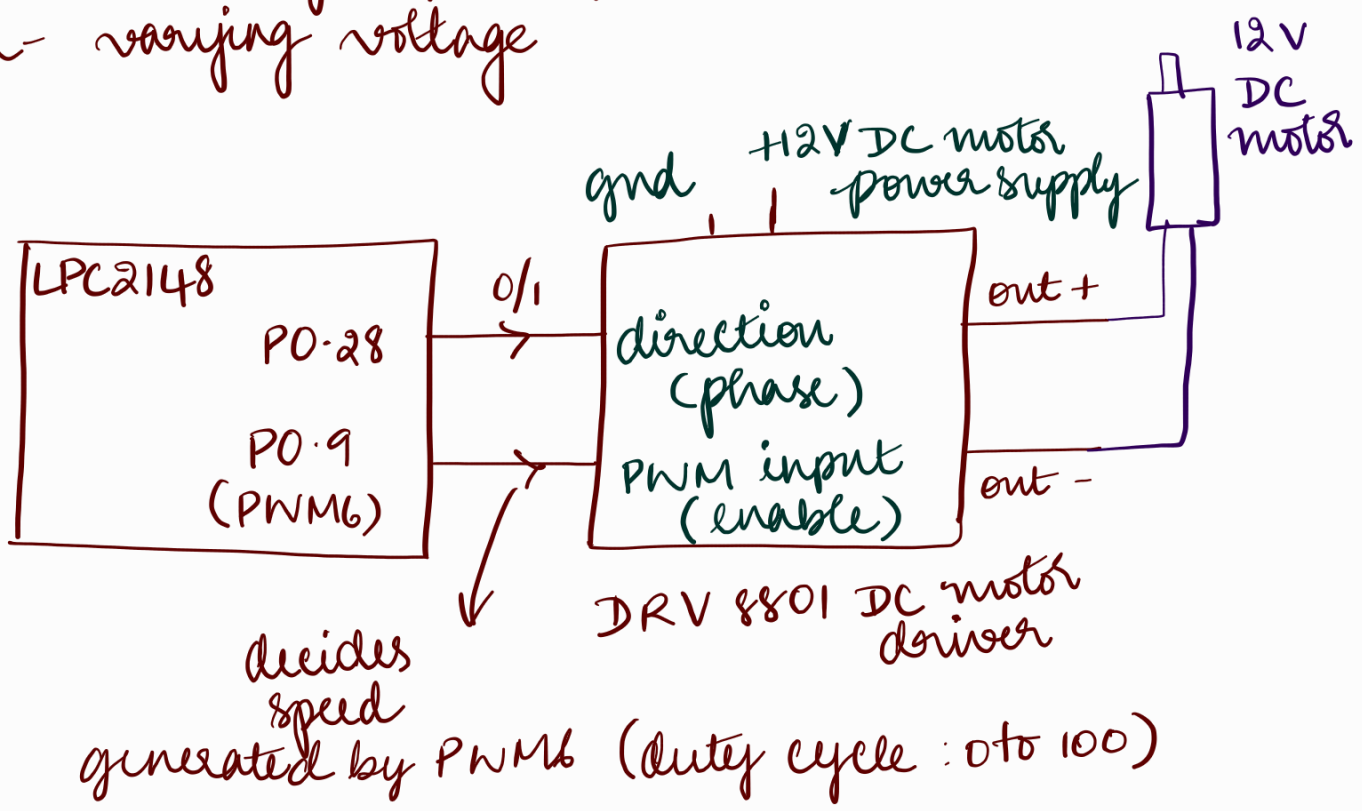
say ⎍ the cycle has to be completed

how does the module know that shadow register is changed? - PWMLER

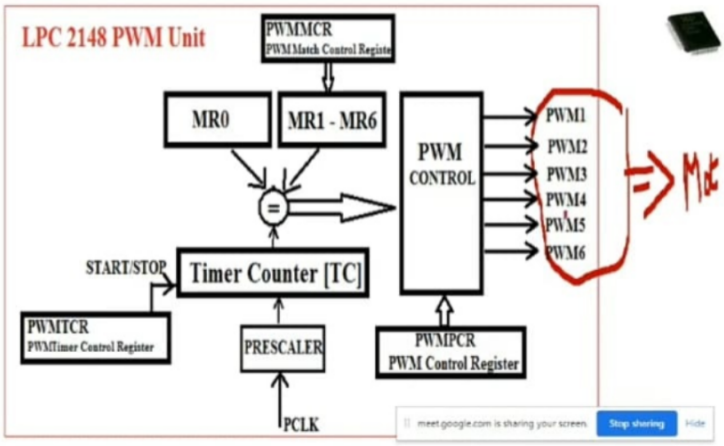
DC motor
↑
0 to 1 ≡ 0 to 12V
from
H bridge motor driver
doesn't really convert only simulates

DC motor speed control:

direction - change polarity
speed - varying voltage



PWM Block Diagram ...



Generate the PWM waveform of 25% duty cycle at the PWM channel 3

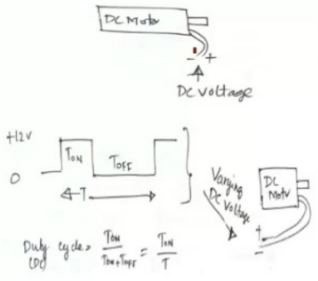
```
#include <LPC214x.h>
int main()
{
PWM_Init();
while(1)
 {
  PWMMR3  =  500; // value which decides pulse ON time, ¼ of 2000
  PWMLER  = 0X08;
 }
}
```



```
void PWM_Init(void)
 {
  //P0.1 pin has second alternate function as PWM3 channel, so using PINSEL0 register
  // Select  P0.1 as PWM output ,  bits D2 & D3 are for P0.1
  PINSEL0 |=  0x00000008;

  //load the value to MR0 to fix the pulse rate
  PWMMR0 = 2000; // any other value, I could have taken

  //Configure PWM channel 3  as single edge type and enable the channel
  //bit D3 to select single edge(make it  0), bit D11 is for enabling PWM3 channel(make it 1)
  PWMPCR = 0x00000800;

  // enable PWM unit of LPC2148 and start the timer
  PWMTCR = 0x00000009;   // bit D3 = 1 (enable  PWM), bit D0=1 (start the timer)
 }
```

Calculation of duty cycle in the above program:
if PCLK is 15MHz, then T= 0.067µsec,
Pulse rate/Period, T = ON+OFF = 2000 * T
Puse width, Ton = 500 * T
Duty cycle = Ton/T = 500*T/2000*T = ¼= 0.25 x 100 = 25 %

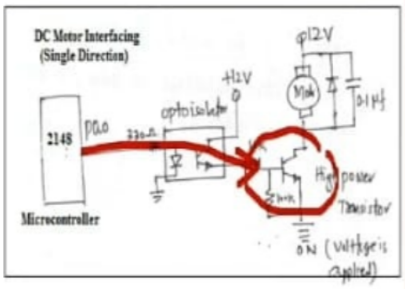
DC Motor Speed Control

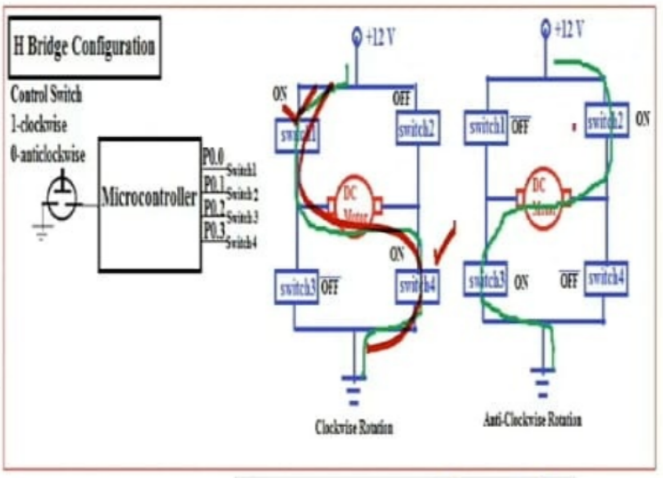


Effective DC Voltage Applied to DC Motor
= (Ton/T) x Supply voltage (say 12V)

If Ton = T, Full Power i.e 12V
If Ton = T/2, 50 % DC (1/2 Power)

H Bridge arrangement To Drive DC Motor



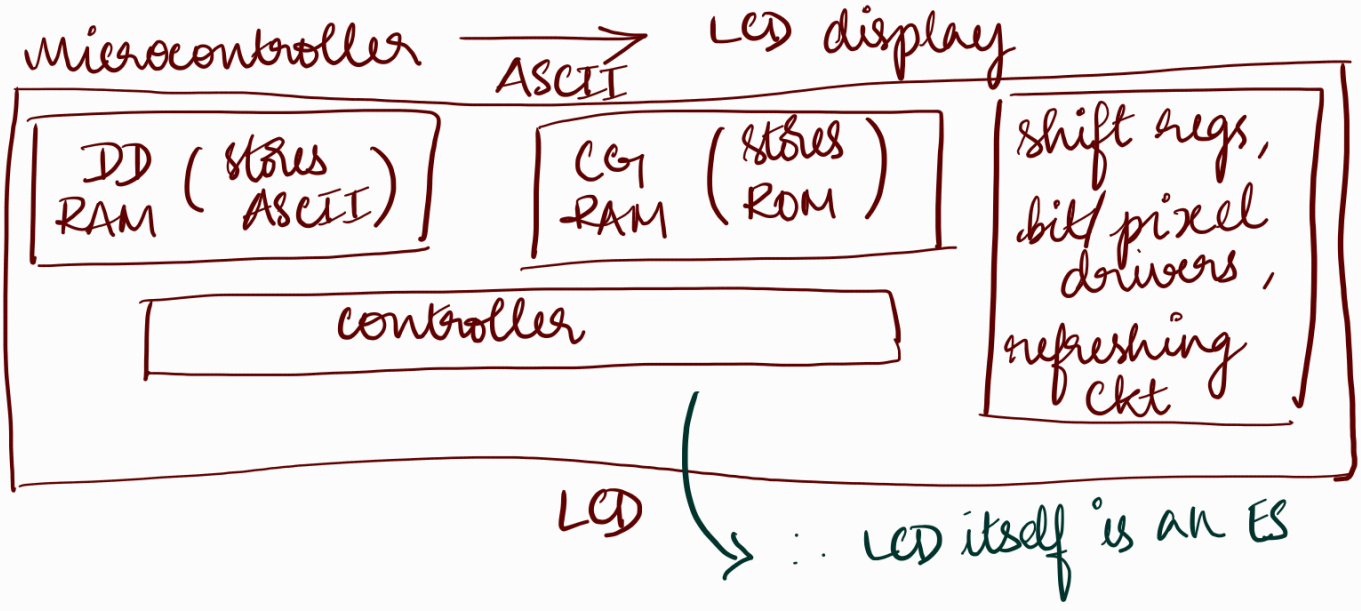


```
#include <lpc214x.h>
void runDCMotor(int direction, int dutycycle) ;

int main()
{
   IO0DIR |= 1U << 28;    //set P0.28 as output pin
   PINSEL0 |= 2 << 18;   //select P0.9 as PWM6 (option 2)
   runDCMotor(2,10);   // run at 10% duty cycle
   while(1);  // do other jobs
}
```



```
void runDCMotor(int direction, int dutycycle)
{
   if (direction == 1)
      IO0SET = 1 << 28;  //set to 1, to choose anti-clockwise direction
   else
      IO0CLR = 1 << 28;  //set to 0, to choose clockwise direction
   PWMMR0 = 1000;      // set PULSE rate to value suitable for DC Motor operation
   PWMMR6 = (1000U*dutycycle)/100;  // set PULSE period
   PWMPCR = (1 << 14);  // enable PWM6 channel
   PWMTCR = 0x00000009;  // bit D3 = 1 (enable  PWM), bit D0=1 (start the timer)
   PWMLER = 0X70;  // load the new values to PWMMR0 and PWMMR6 registers
}
```

LCD:



∴ LCD itself is an ES

lot of settings available: /configure
out of 2 lines - use only one
cursor appearance
each char - 5X7 & 5X10 dots
in DDROM - 40 bytes can be stored/line
even though 16 appear ∴ can give rotating affect

configurations } commands
behave instructions }

what to be displayed? } data

LCD module 16X2 — 16 pins

- 5 → power supply - 5V, gnd (5 pins) — 2 Vcc, 1
- 3 → handshaking - RS, R/W, E
  - E → ⏚
- 8 → data - 7 to 14

3 - VO - contrast adjustment
15, 16 - backlight pins

LCD - llc in LCD - low speed
the (llc) we use (connected to LCD) - v. high speed
∴ LCD using its llc reads its status register to check if its busy or not to display next char from (llc).

LCD is ↳ writing
LCD is reading

handshaking pin
∴ r/w pin

∴ LCD is not just o/p, it's i/p - o/p

the problem can also be overcome by introducing delay after every character is written.

5
r/w = 0 → only write
1 ↳ read

4
RS = 0 → write command byte to command reg of LCD
1 → write data into data reg of LCD

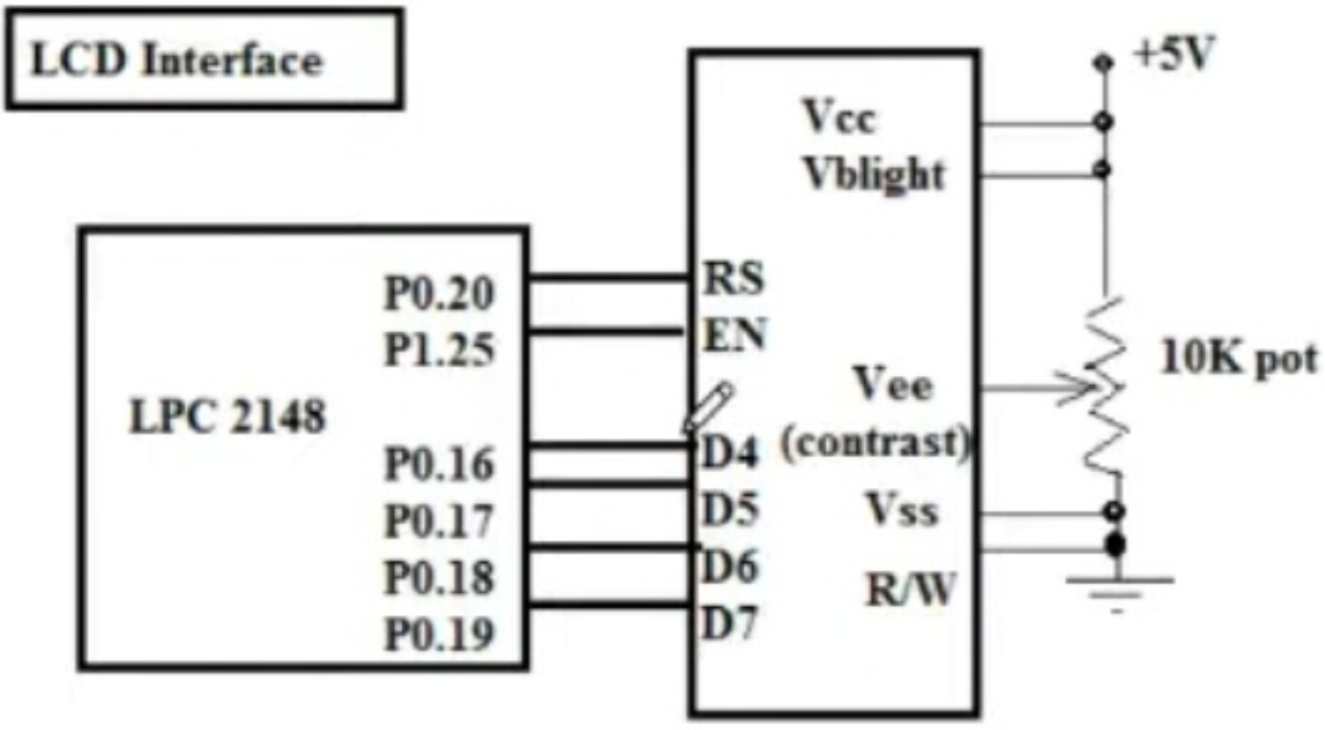
8 bit bus (7 to 14 pin)

say u place A after sometime send another A & so on... how does llc know its the same A?
after sending a char, send a high to low pulse to E (enable) pin. LCD will copy to its reg

6

↓ byte mode

↓ Nibble mode
4 pins needed
to save (llc) pins → upper (1st)
↳ lower

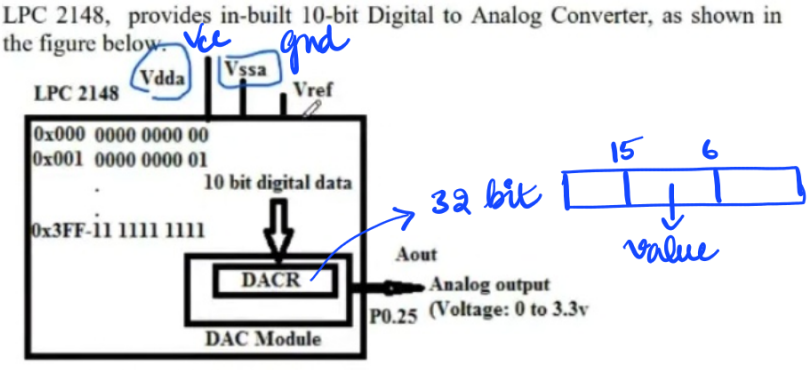


3 steps to write data -

→ setting RS
→ writing data
→ enable

DAC -

LPC 2148, provides in-built 10-bit Digital to Analog Converter, as shown in the figure below.



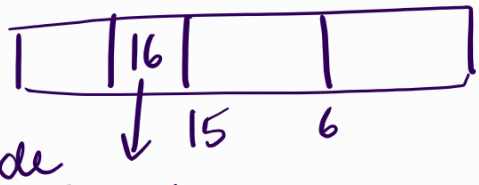
- DAC module of LPC 2148 is a 10 bit Digital to Analog converter used to convert 10 bit Digital data to corresponding Analog voltage.
- Digital I/P : **000 to 3FF (0 to 1023)**, corresponding Analog O/P : **0V to 3.3V**
- Resolution = (3.3/1024) ≈ **3.2mili volts**

10 bit value → 1024

3·3V ∴ resolution = $\frac{3.3}{1024}$ V = 3·2 mV

i/p = 100

100 × 3·2mV = 320mV



DACR

BIAS

mode

0 → fast mode
→ high drive
→ high power

1 → slow mode
→ low drive
→ low power
→ more accuracy

ADC $\leq 4MHz$

Before ADC were there outside
in LPC2148, ADC is present inside

V_{DDA} } power supply
V_{SSA} }

ADC - 2 blocks

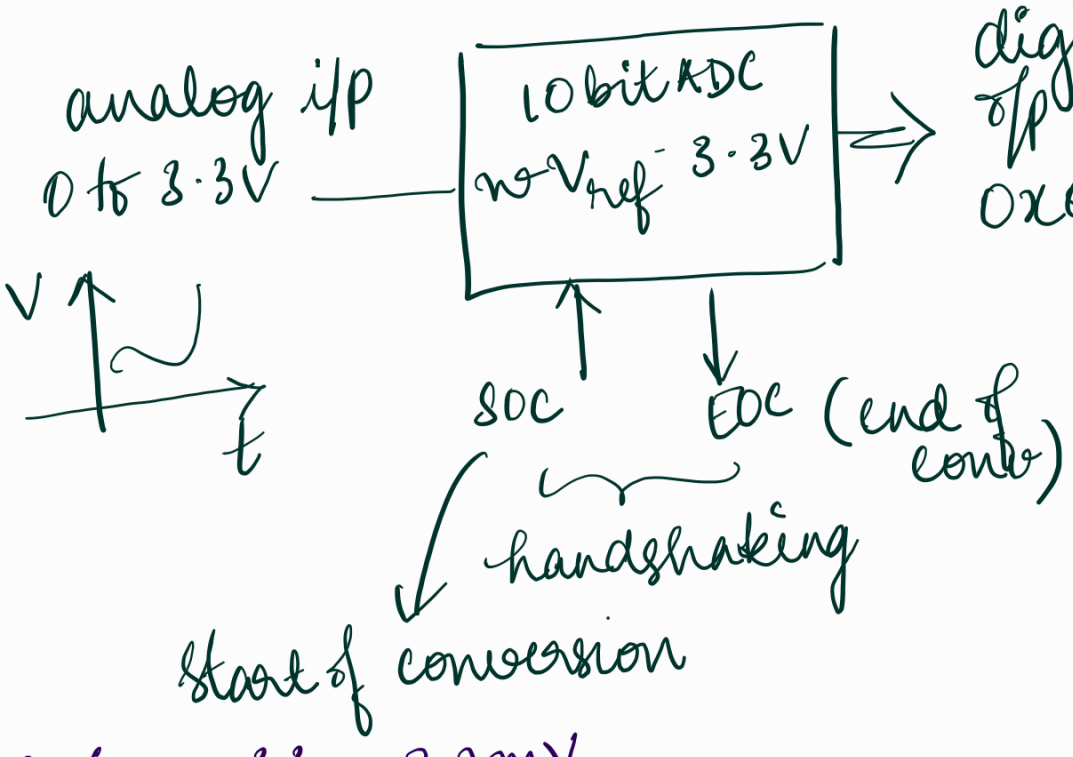
many inputs, but conversion one at a time

ADO(7:6) & (4:1) → 6 } 14 analog i/ps
AD1 (7:0) → 8 }

2 conversion at a time
ADO & 1

Techniques -
* dual slope conversion
* successive approximation

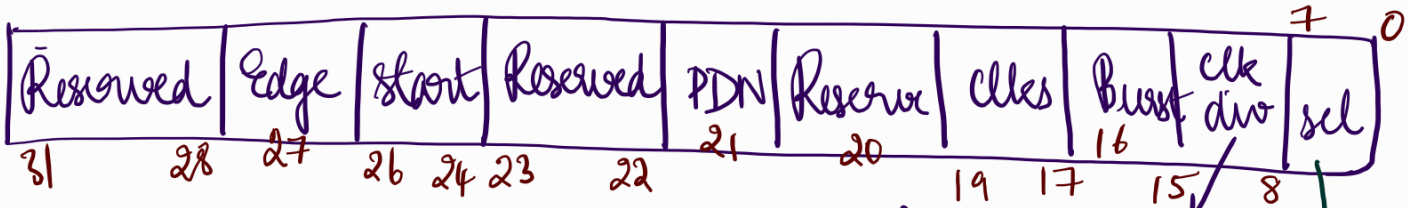
V_{ref} can be fixed



Resolution = $\frac{3.3}{2^{10}}$ = 3.2mV

1V = $\frac{10^3}{3.2}$ = 310 = 136 (hex)

0x136 = 01 0011 0110



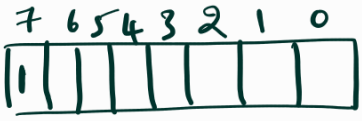
15 to 8 = x

ADC control reg 32bit

x

$$\frac{15\text{MHz (PClk)}}{x+1} \geq \text{ADC clk} \leq 4.5\text{MHz}$$

PDN- Power Down Bit (21)
- 1 - ADC starts working
- 0 - PDN mode

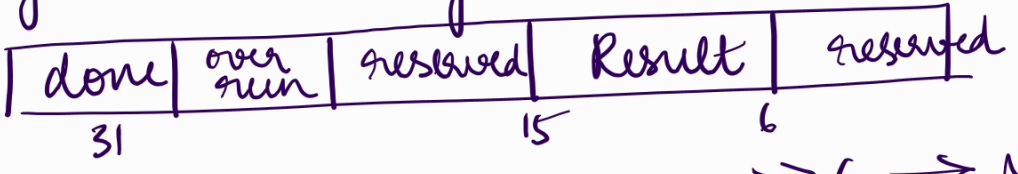


to select ADI.7

Note: ADO, ADI have their own ADC control reg

Start conversion : 001 (26 25 24)

ADC global data reg (32bit) → common o/p reg

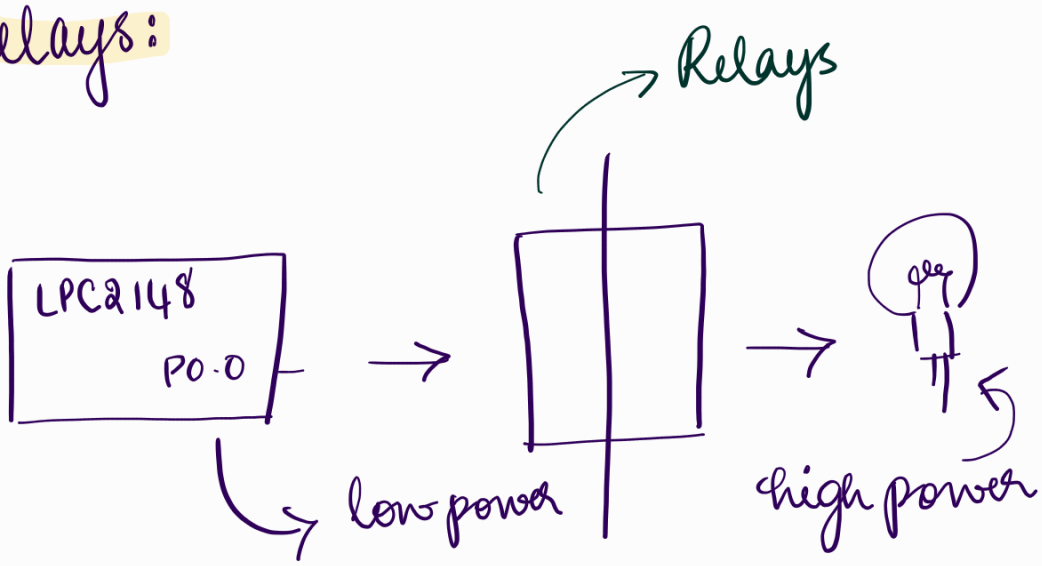


o/p variable → 6

In burst mode - each i/p has an o/p data reg

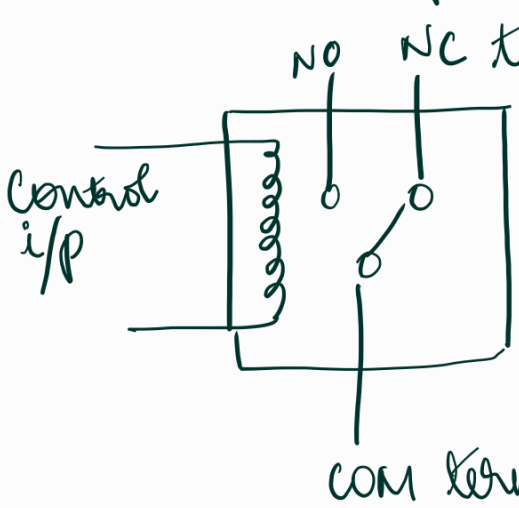
ADOGDR
ADIGDR

Relays:



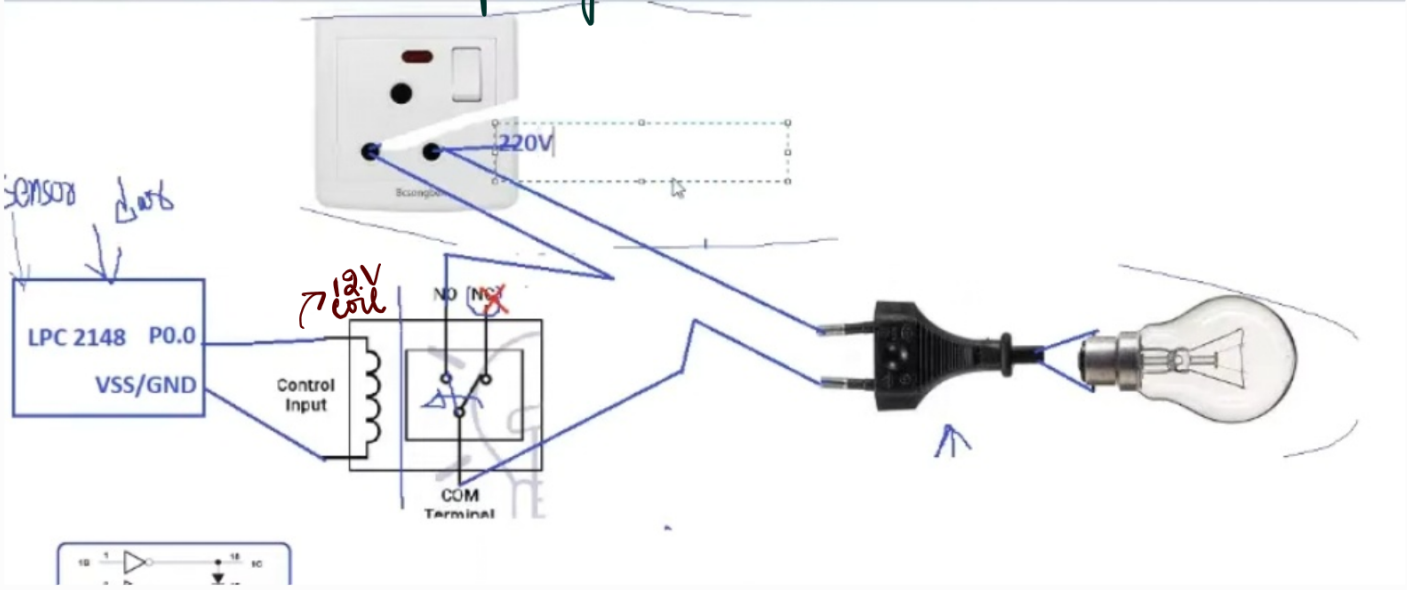
* transistor
8V, 12V ✓
220V, 48V X
(high power)

* Electromagnet relay – mech: relay
* Solid state - mosfets are used

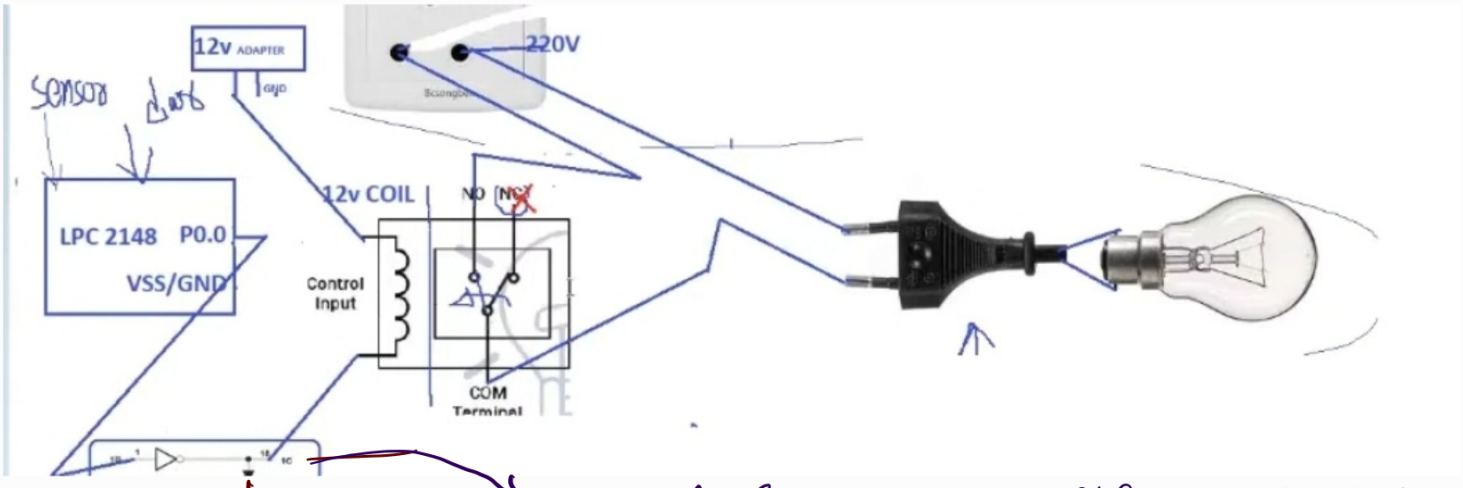


NC - normal close
NO - open

coil → current → EM → free COM term to NO
ON
OFF → spring action —"— to NC



so coil can't be directly connected to uc.

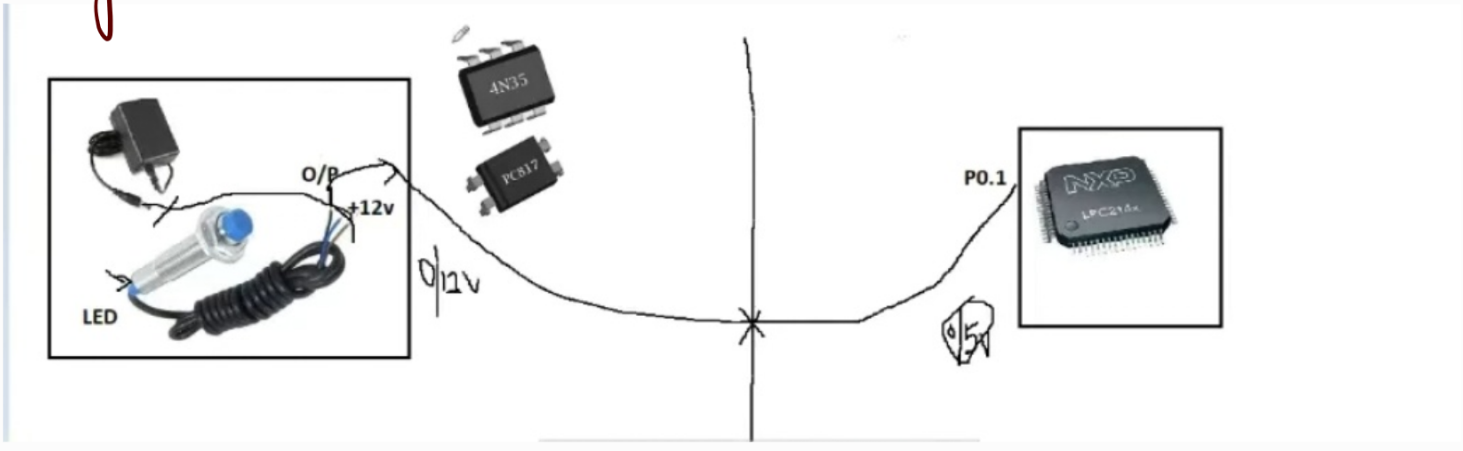


frigling diodes

GND

transistor - can amplify upto 12V
also shud use a
12v adapter.

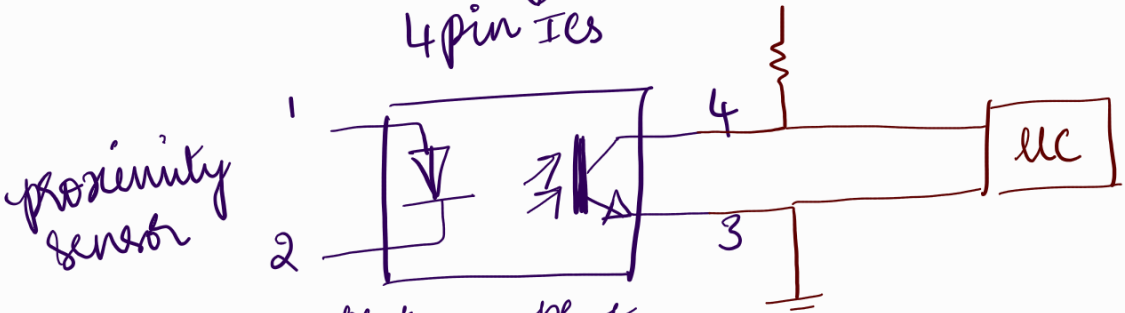
Digital sensor



to isolate - opto-isolator/ coupler

↓

4pin ICs



sensor ON = when transistor o/p = 1 uc reads 0

OFF = o/p = 0 uc reads 1