

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

$\text{Pclk} \times \frac{1000}{\text{MRO value}} = \text{time period}$

$\text{MRI} = 500$, duty cycle = $\text{Pclk} \times \frac{\text{MRI}}{\text{MRO}} = 50\%$

* shadow registers for MRI to MR6

latch enable register PWMLER

↓ here user wants to change speed b/w a cycle
say  stored in shadow register

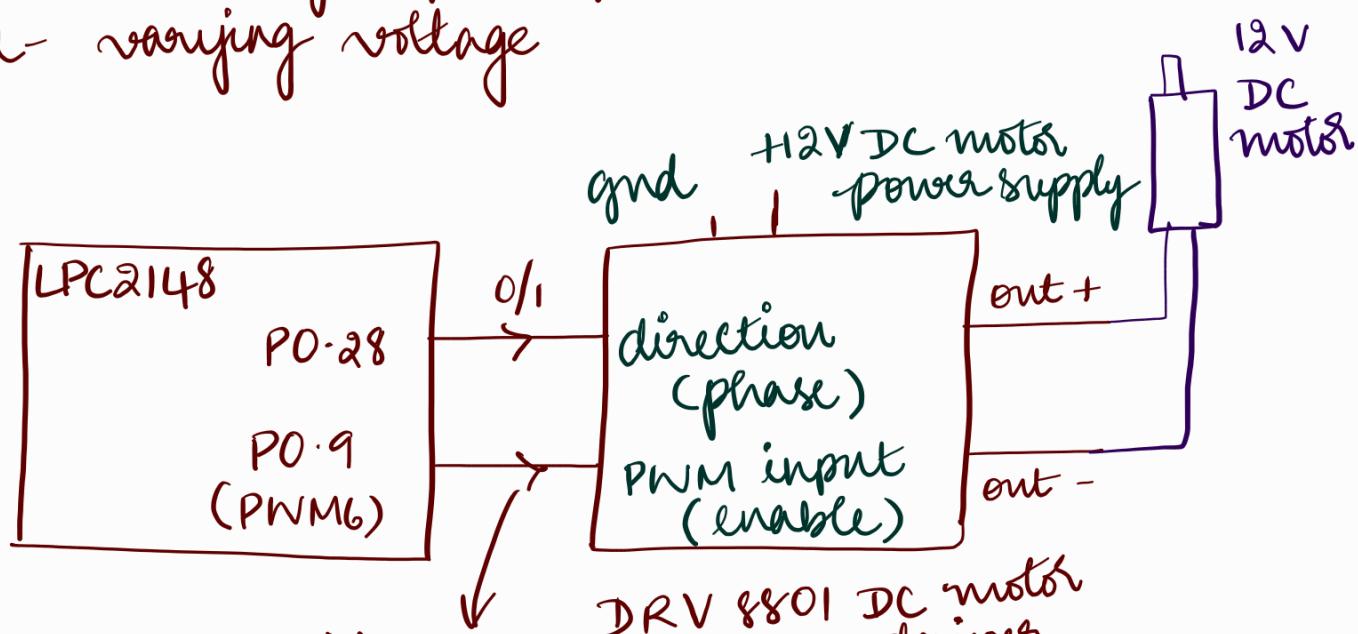
the cycle has to be completed

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

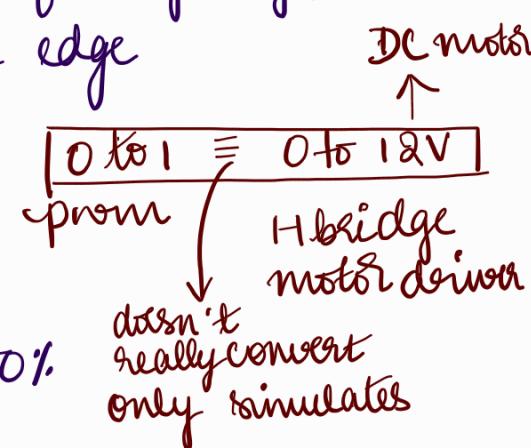
DC motor speed control:

direction - change polarity

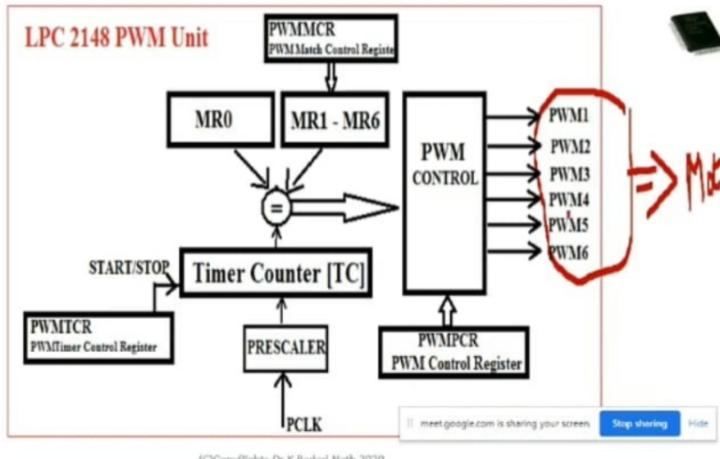
speed - varying voltage



decides speed
generated by PWML6 (duty cycle : 0 to 100)



PWM Block Diagram ...



(C)CopyRights,Dr K Badari Nath 2020

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, % of 2000
        PWMLER = 0X08;
    }
}
```

(C)CopyRights,Dr K Badari Nath 2020

```
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 MRO 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;

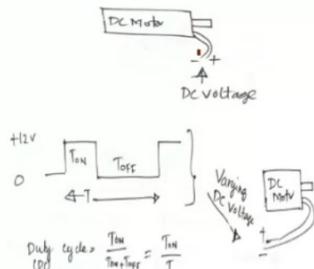
    // enable PWM unit of LPC2148 and start the timer
    PWMTCR = 0x00000009; // bit D3 = 1 (enable PWM), bit D0=1 (start the timer)
}
```

(C)CopyRights,Dr K Badari Nath 2020

Calculation of duty cycle in the above program:if PCLK is 15MHz, then T= 0.067μsec,Pulse rate/Period, T = ON+OFF = 2000 * TPuse width, Ton = 500 * TDuty cycle = Ton/T = 500*T/2000*T = 1/4 = 0.25 x 100 = 25 %

meet.google.com is sharing your screen Stop sharing Hide

(C)CopyRights,Dr K Badari Nath 2020

DC Motor Speed Control

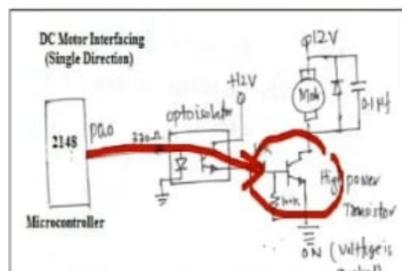
Effective DC Voltage Applied to DC Motor
 $= (Ton/T) \times \text{Supply voltage (say } 12V)$

If $T_{on} = T$, Full Power i.e $12V$

If $T_{on} = T/2$, 50 % DC (1/2 Power)

(C)CopyRights,Dr K Badari Nath 2020

H Bridge arrangement To Drive DC Motor

**H Bridge Configuration**

Control Switch

1-clockwise

0-anticlockwise

0-Switch1

P0.0

Switch2

P0.1

Switch3

P0.2

Switch4

P0.3

Switch5

05

SWITCH1

OFF

switch2

SWITCH3

OFF

switch4

ON

switch5

ON

SWITCH6

OFF

switch7

ON

switch8

OFF

switch9

ON

switch10

OFF

switch11

ON

switch12

OFF

switch13

ON

switch14

OFF

switch15

ON

switch16

OFF

switch17

ON

switch18

OFF

switch19

ON

switch20

OFF

switch21

ON

switch22

OFF

switch23

ON

switch24

OFF

switch25

ON

switch26

OFF

switch27

ON

switch28

OFF

switch29

ON

switch30

OFF

switch31

ON

switch32

OFF

switch33

ON

switch34

OFF

switch35

ON

switch36

OFF

switch37

ON

switch38

OFF

switch39

ON

switch40

OFF

switch41

ON

switch42

OFF

switch43

ON

switch44

OFF

switch45

ON

switch46

OFF

switch47

ON

switch48

OFF

switch49

ON

switch50

OFF

switch51

ON

switch52

OFF

switch53

ON

switch54

OFF

switch55

ON

switch56

OFF

switch57

ON

switch58

OFF

switch59

ON

switch60

OFF

switch61

ON

switch62

OFF

switch63

ON

switch64

OFF

switch65

ON

switch66

OFF

switch67

ON

switch68

OFF

switch69

ON

switch70

OFF

switch71

ON

switch72

OFF

switch73

ON

switch74

OFF

switch75

ON

switch76

OFF

switch77

ON

switch78

OFF

switch79

ON

switch80

OFF

switch81

ON

switch82

OFF

switch83

ON

Clockwise Rotation

Anti-Clockwise Rotation

meet.google.com is sharing your screen Stop sharing Hide

(C)CopyRights,Dr K Badari Nath 2020

```
#include <lp214x.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
}
```

(C)CopyRights,Dr K Badari Nath 2020

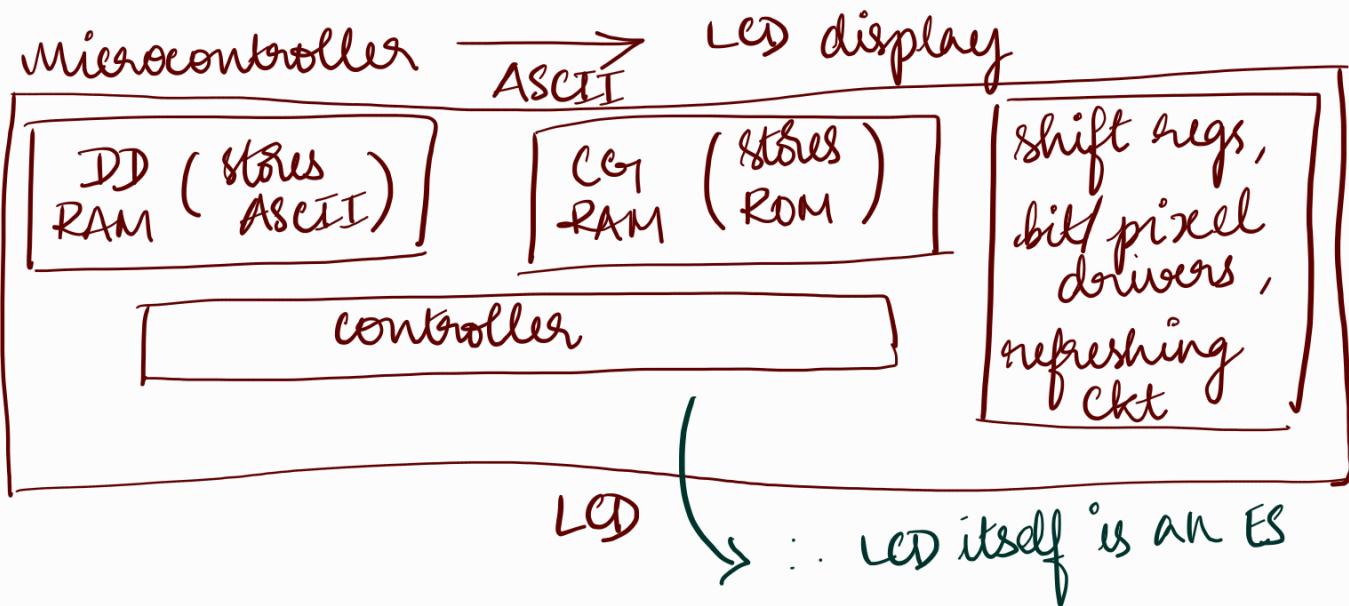
 meet.google.com is sharing your screen. Stop sharing Hide

```
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
}
```

(C)CopyRights,Dr K Badari Nath 2020

 meet.google.com is sharing your screen. Stop sharing Hide

LCD:



lot of settings available: /configure

out of 2 lines - we only one

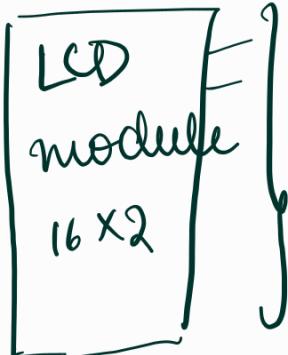
cursor appearance

each char - 5x7 or 5x10 dots

in DDRAM - 40 bytes can be stored /line

even though 16 appear ∵ can give rotating effect

configurations } commands
behaviour instructions }
what to be } data
displayed?



- 16 pins
- 5 → power supply - V_{cc} , 1 , $5V$, gnd (5 pins)
 - 3 → handshaking - RS , R/W , E
 - 8 → data - 7 to 14

3 - V_0 - contrast adjustment 3

15, 16 - backlight pins

LCD - I_C in LCD - low speed
 the I_C we use (connected to LCD) - v. high speed
 LCD using its I_C reads its status register to
 check if it's busy & not to display next char from
I_C.
 LCD is writing handshaking
 LCD is reading : pin
 - R/W pin

LCD is not just off, it's I/P-O/P
 the problem can also be overcome by introducing
 delay after every character is written.

5 $R/S = 0 \rightarrow$ only write
 6 $R/S = 1 \rightarrow$ read

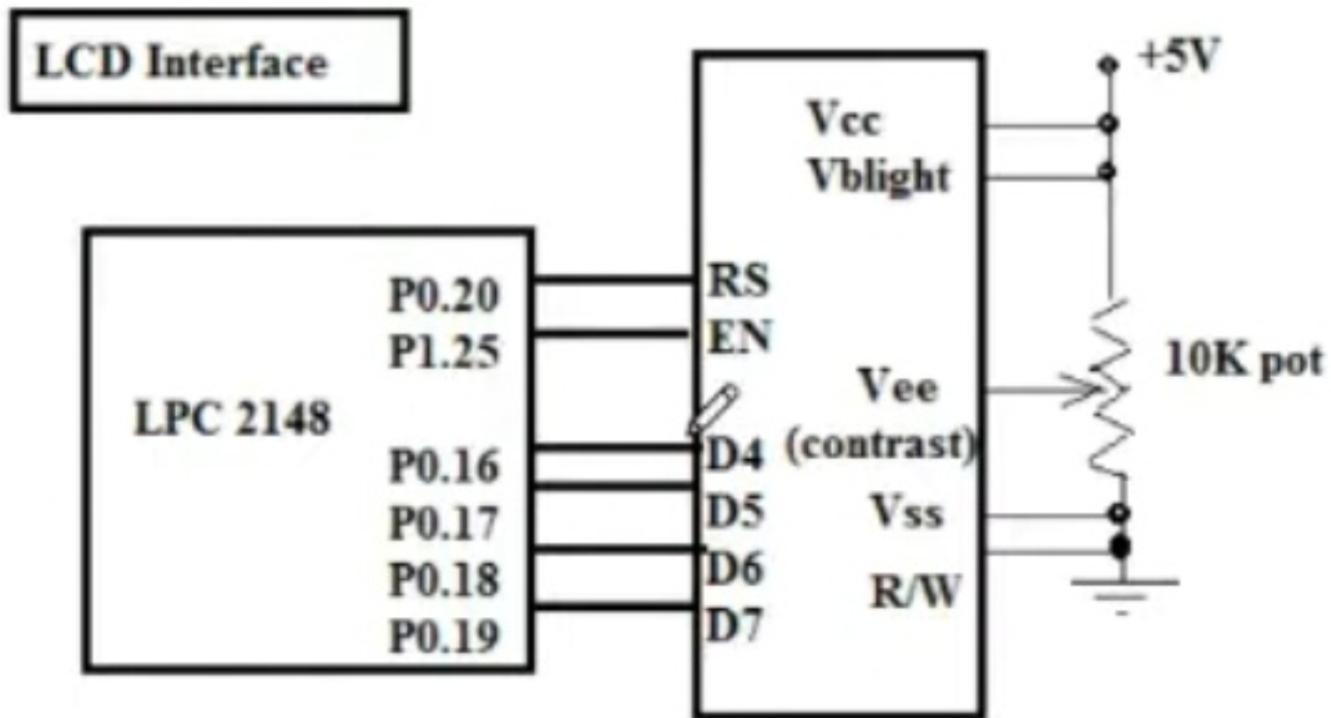
4
 RS = 0 \rightarrow write command byte to command reg of LCD
 1 \rightarrow write data into data reg of LCD

8 bit bus (\pm to 14 pin)

say u place A after sometime send another A
 & so on... how does I_C know its the same?
 after sending a char, send a high to low pulse
 to E(enable) ⁶ pin. LCD will copy to its reg


 byte mode nibble mode

4 pins needed
 to save I_C pins upper (1st)
 lower



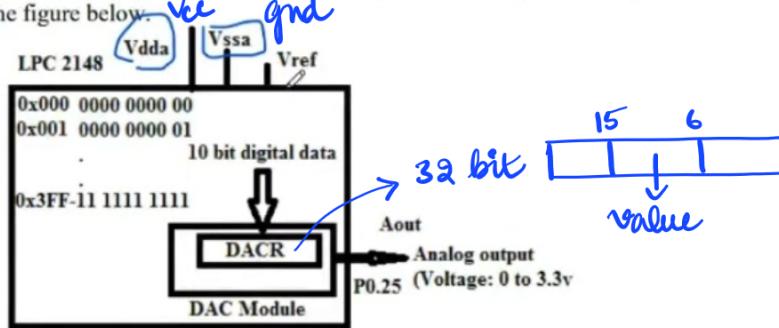
3 steps to write data -

- setting RS
- writing data
- enable

DAC -



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



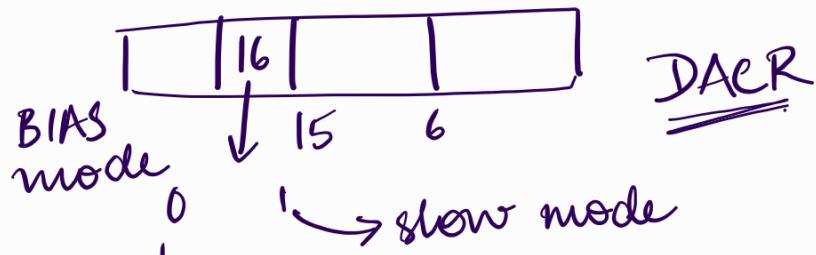
- DAC module of LPC 2148 is a 10 bit Digital to Analog converter used to convert 10 bit Digital data to corresponding Analog voltage.
- Digital I/P : 000 to 3FF (0 to 1023), corresponding Analog O/P : 0V to 3.3V
- Resolution = $(3.3/1024)$ = 3.2 mili volts

$$10 \text{ bit value} \rightarrow 1024$$

$$3.3V \therefore \text{resolution} = \frac{3.3}{1024} V = \underline{\underline{3.2 \text{ mV}}}$$

$$i/f = 100$$

$$100 \times 3.2 \text{ mV} = \underline{\underline{320 \text{ mV}}}$$



- fast mode
→ high drive
→ high power
- slow mode
→ low drive
→ low power
→ more accuracy

ADC $\leq 4\text{MHz}$

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

V_{DDA} } power supply
 V_{SSA}

ADC - 2 blocks
↓ ↓
0 1

many inputs, but conversion one at a time

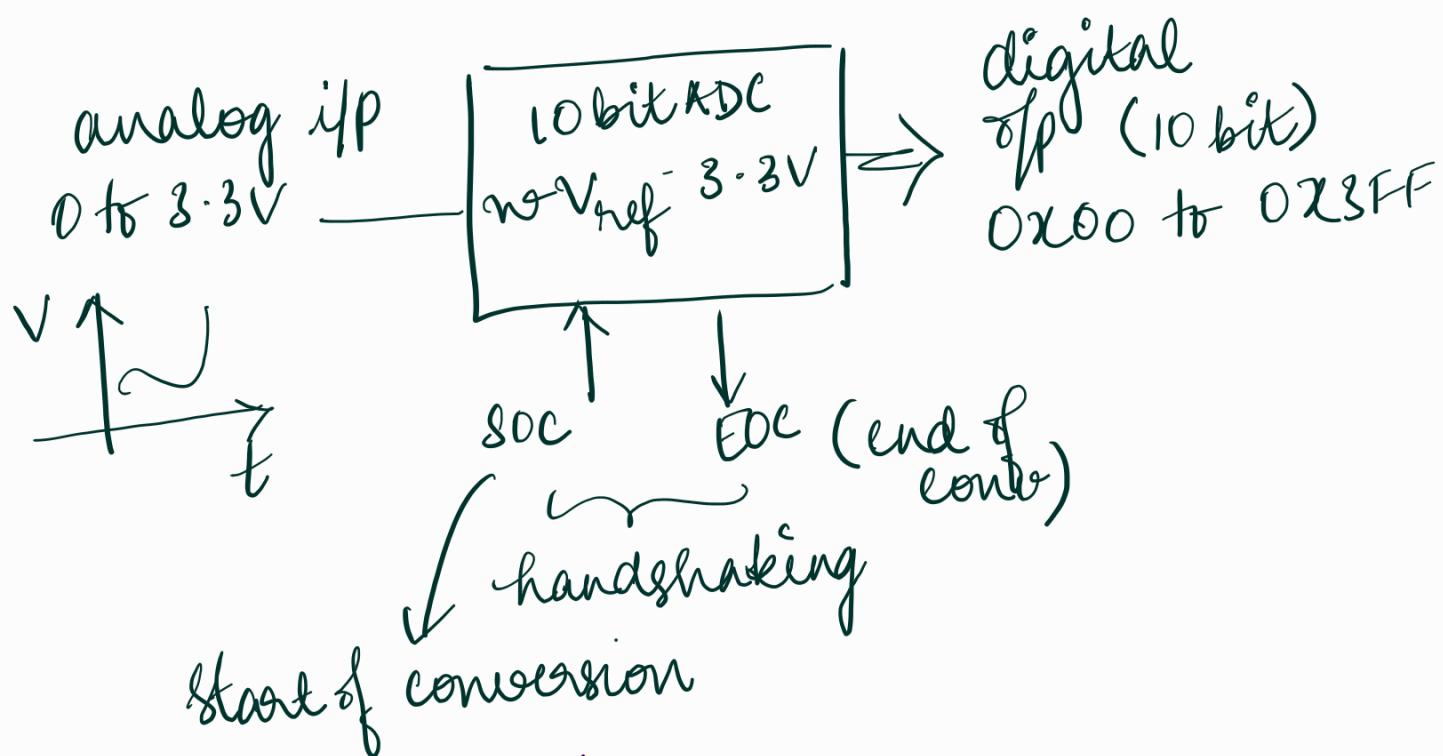
ADOC (7:6) $\Sigma 1$ (4:1) $\rightarrow 6$ } 14 analog i/p's
ADI (7:0) $\rightarrow 8$

2 conversion at a time
ADO $\Sigma 1$

Techniques -

* dual slope conversion
* successive approximation

V_{ref} can be
tuned



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

$$IV = \frac{10^3}{3.2} = 310 = 136(\text{hex})$$

$$0x136 = 01\ 0011\ 0110$$

Reserved	Edge	Start	Reserved	PDN	Reserved	Ckes	Burst	Clk	7	0					
31	28	27	26	24	23	22	21	20	19	17	16	15	8	7	0

$$15 \times 8 = x$$

ADC control reg
32bit

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

PDN - Power Down Bit
(21)

1 - ADC starts working
0 - PDN mode

7	6	5	4	3	2	1	0
1	1						

to select ADI-7

Note: AD0 { have their own
ADI } ADC control reg

start conversion : 001^{26 25 24}

ADC global data reg (32bit) → common I/P reg

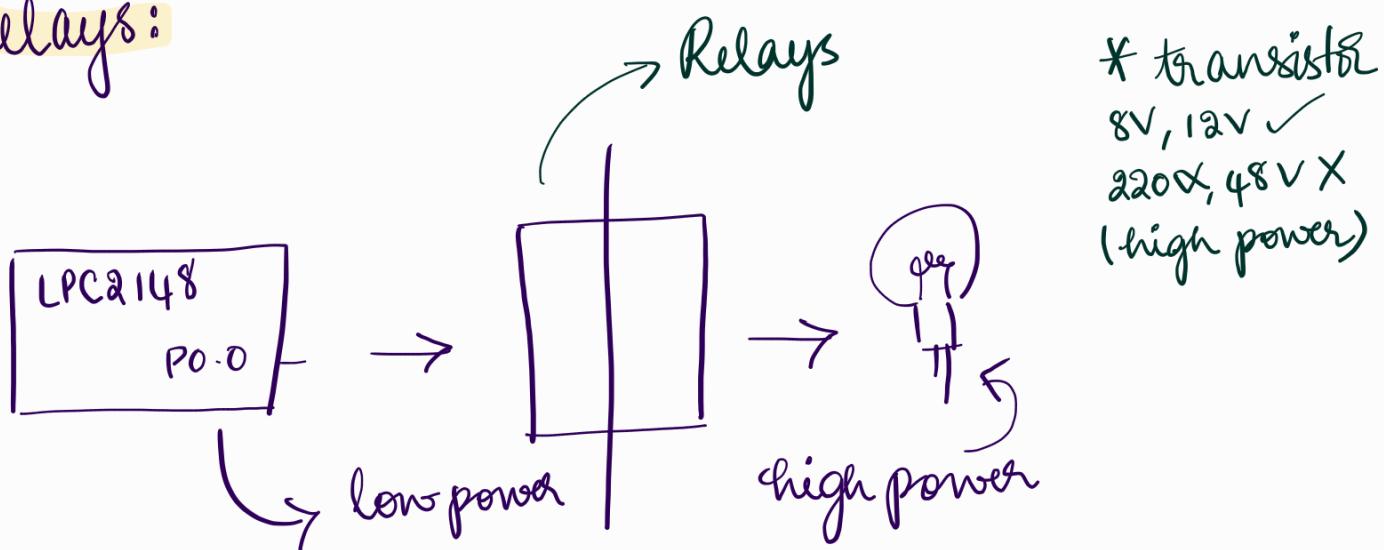
done	over run	reserved	Result	reserved	I/P
31	15	6	6	6	variable

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

AD0GDR

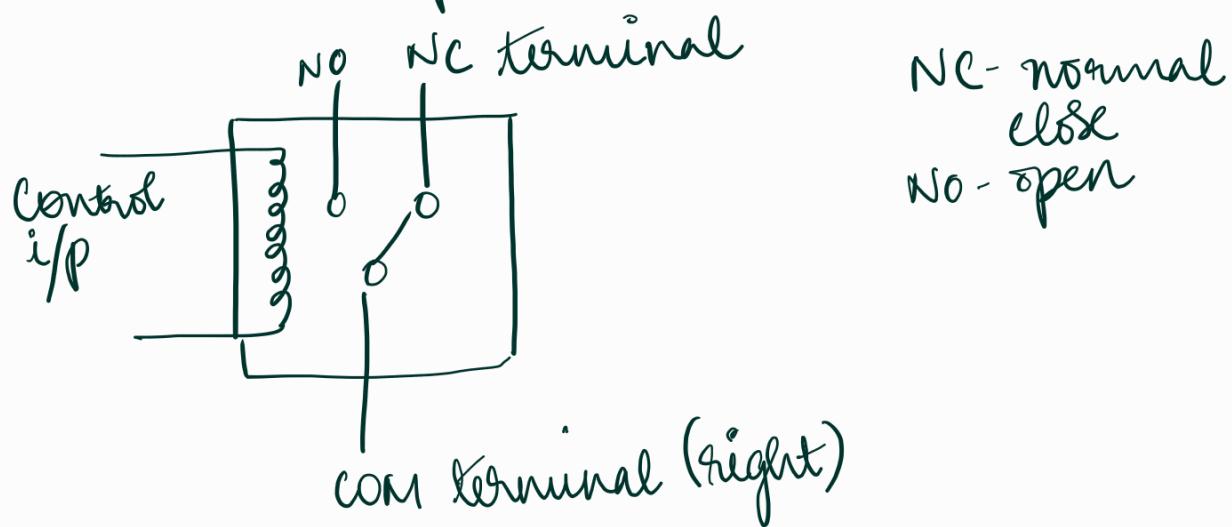
AD1GDR

Relays:

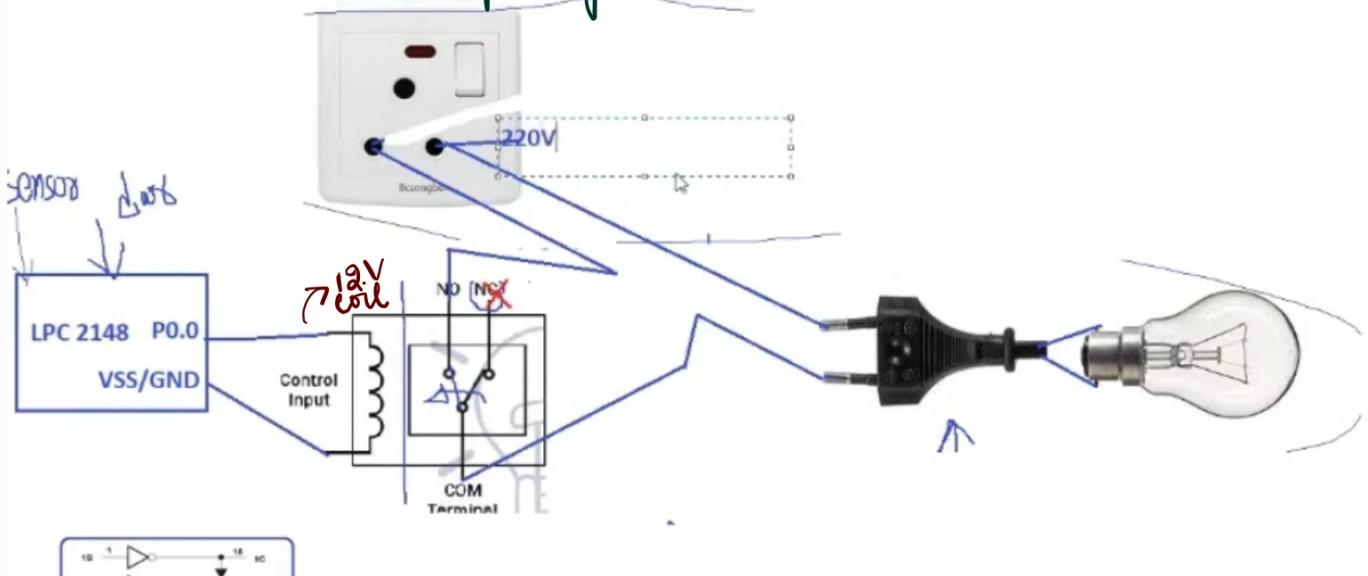


- * transistor
- 8V, 12V ✓
- 220V, 48V X
- (high power)

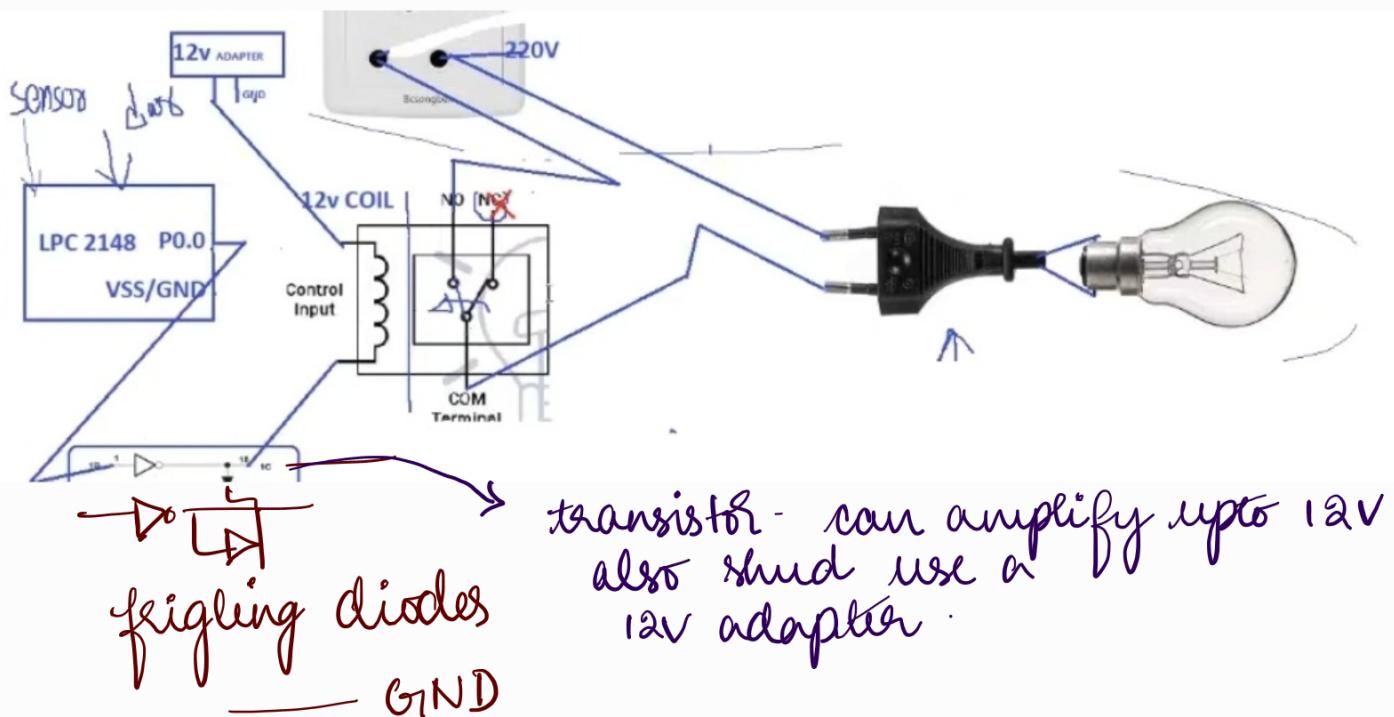
- * electromagnet relay - mech. relay
- * solid state - mosfets are used



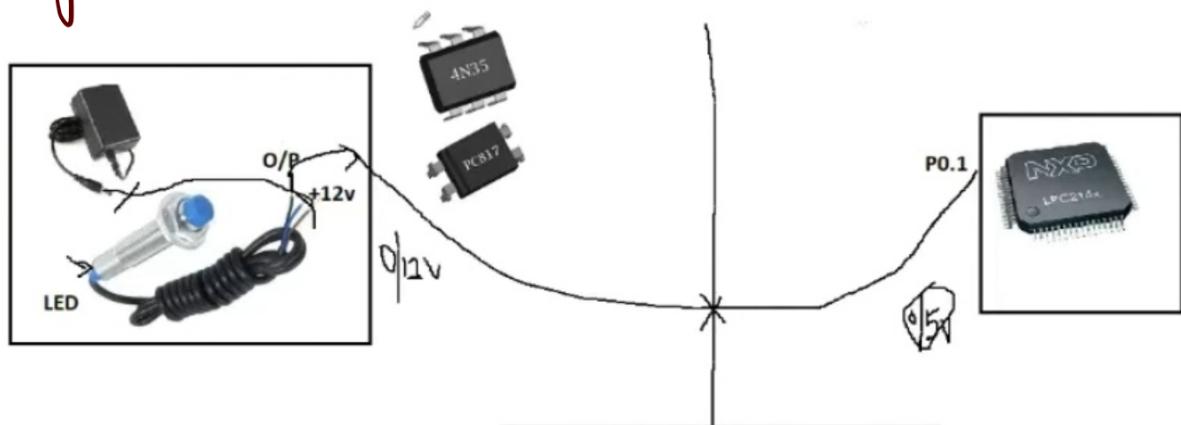
coil \rightarrow current \rightarrow EM \rightarrow free com term to NO
 ON \rightarrow spring action \rightarrow to NC
 OFF \rightarrow



coil can't be directly connected to EEC.



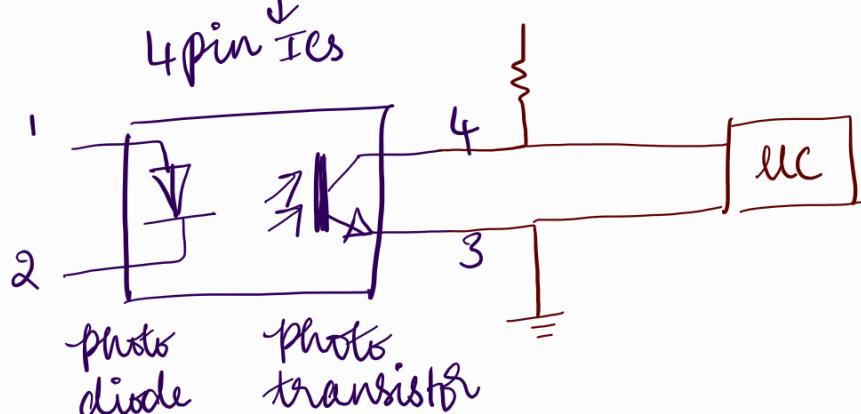
Digital sensor



to isolate - opto-isolator/coupler

4pin ICs

proximity
sensor



sensor ON = when transistor $Q_P = 1$
OFF = $Q_P = 0$

EEC reads 0
EEC reads 1