prof K.B.Ramesh

* ADC [Analog to Digital Converter] :

→ The Lpc-2148 microcontroller has two 10-bit ADC's
  1) ADC0 : Six channels
  2) ADC1 : Eight channels
  } Totally 14 channels.

→ **ADC channels :**
  1) ADC-0 : Has 6 input channels
  2) ADC-1 : Has Eight(8) input channels

NOTE :

→ The Lpc2148 has Single 10-bit DAC

→ The Lpc2148 has two ADC modules. → ADC0, ADC1

ADC0		ADC1	
Input channels	Lpc 2148 pins	I/p channels	Lpc 2148 pins
AD0.1	P0.28 PIN-13	AD1.0	P0.6 PIN 30
AD0.2	P0.29 PIN-14	AD1.1	P0.8 PIN 33
AD0.3	P0.30 PIN-15	AD1.2	P0.10 35
AD0.4	P0.25 PIN-9	AD1.3	P0.12 38
AD0.6	P0.4 PIN-27	AD1.4	P0.13 39
AD0.7	P0.5 PIN-29	AD1.5	P0.15 45
		AD1.6	P0.21 1
		AD1.7	P0.22 2

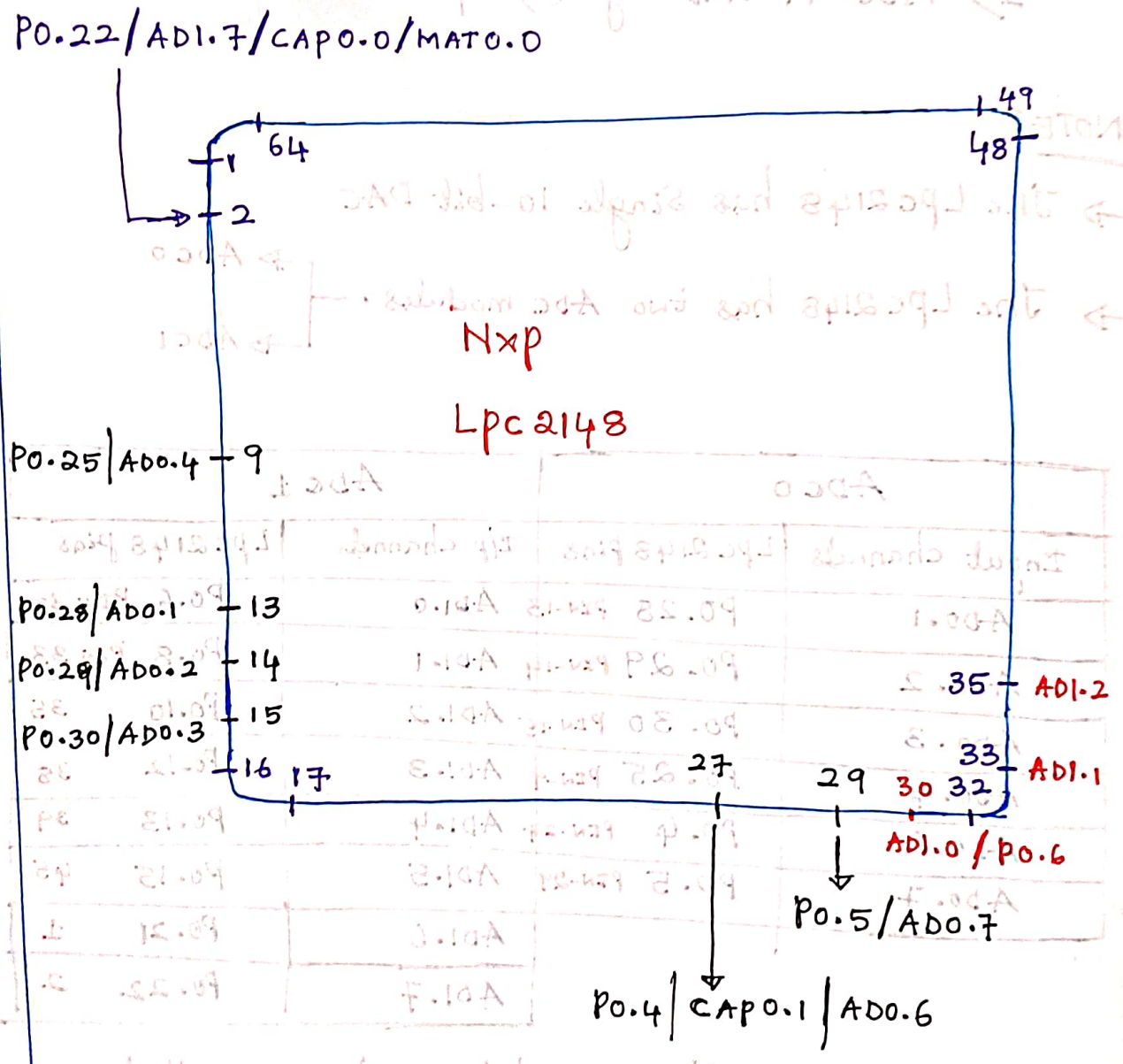
NOTE : ADC modules offering 10-bit resolution, means that, Convert analog voltages into 10-bit digital values.

NOTE :

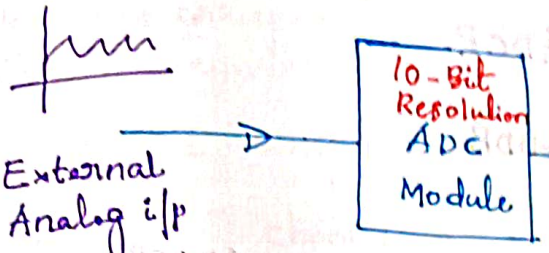
→ We can connect six distinct types of input analog input signals to ADC0. And, eight distinct types of analog input signals to ADC1.

→ Successive approximation Technique has been implemented to convert analog signal into digital form.

* Pins of ADC in ARM Lpc 2148



NOTE:



We can Connect 14 Analog i/p's: → 14 channels

NOTE: In 8051 µc, there is no built-in ADC. Hence, we used external ADC's which leads to more errors. whereas, In LPC2148 µc has got built-in ADC's which are having more advantages.

→ Analog inputs ranges from 0V to Vref (3.3V).

→ Conversion time of ADC's is 2.44 µs.

NOTE: ADC 0.0 — AD0.0
ADC 0.5 — AD0.5
} These pins are not available

* SFRs Involved : [Special Function Register]

1) ADC control Register: ADCR

2) ADC Data Register: ADDR

* ADC control Register – ADCR → ADOCR, ADICR

* ADC Data Register – ADDR → ADODR, ADIDR } specific channels

ADODR → ADIDR

For all channels

ADOGDR } ADO Global Data Register

ADIGDR } ADI Global data Register

* ADC control Register: ADCR : 32-bit Register

31 – 28	27	26	24 23	22 21	20	19 – 17	16	15 – 8	7 – 0
Reserved	Edge	START	RESERVED	PDN	~~20~~	CLKS	BURST	CLKDIV	SEL

- Edge: Rising (0) / Falling (1); operational = 1
- START: start of conversion; It has to be 001
- PDN: Power down Bit; 1 = ON
- ~~20~~: Reserved
- CLKS: For setting the resolution of ADC
- BURST: Burst pin must be 1 for Continuous Conversion.
- CLKDIV: clock Divider; Speed of conversion
- SEL: channel select

* Select Bits: SEL[7-0] → channel select bits.

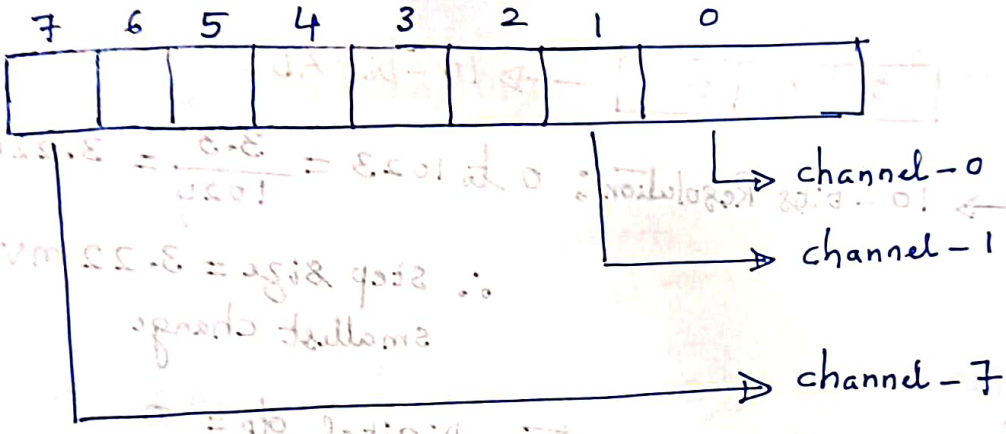
→ Select bits corresponds to 8 different channels available on either A/D converter.

→ Bit set to 1 corresponds to the particular channel.

Example: channel-7: 10000000 → AD1.7

channel-5: 00100000 → AD1.5

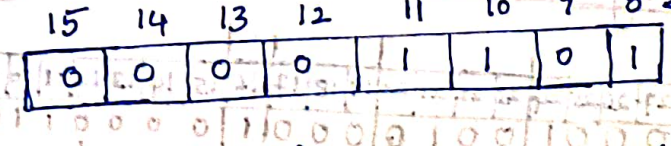
Both channel 7 & 5: 10100000 → Both AD1.7 & AD1.5.



* CLKDIV: The A/D converters are able to run at a maximum speed of 4.5 MHz.

→ P_{CLK} = peripheral clock = 60 MHz

→ Example, To get 4.5 MHz speed of conversion, then 8-bits CLKDIV will be, $CLKDIV = \frac{60\ MHz}{4.5\ MHz} = 14 - 1 = 13$



* BURST: this bit must be 1 for continuous conversion.

* CLKS: used to set Resolution of ADC

19	18	17	Bit position
0	0	0	→ 10-bits Resolution
0	0	1	→ 9-bits Resolution
0	1	0	→ 8-bits Resolution
0	1	1	→ 7-bits Resolution
1	0	0	→ 6-bits Resolution
1	0	1	→ 5-bits Resolution
1	1	0	→ 4-bits Resolution
1	1	1	→ 3-bits Resolution

0	0	0	→ 10-bit ADC

→ 000 → 10-bits Resolution: 0 to 1023 $= \frac{3.3}{1024} = 3.22\,mV$

∴ step size = 3.22 mV
smallest change

Example: Vin → 0V then Digital o/p = 0

Vin → 3.22 mV then Digital o/p = 1

* Start of Conversion:

↳ Bits [26:24] = 001 for start of Conversion.

Example: Configuration of ADC1 control Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0
0				1				2				1				0				D				8				0			

∴ ADICR = 0x01210D80

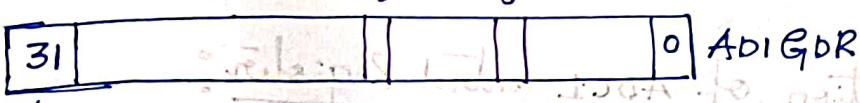
A - 10
B - 11
C - 12
D - 13

* ADIGDR — ADC1 Global Data Register:

→ This Register is used for two purposes.

1) To get Digital o/p. → Bits [15:6]

2) To check EOC. → Bit 31 → Result



↳ Done bit = 1 → Conversion is over.

= 0 → ADC is busy.

→ After the conversion, Digital data is available at bits [15:6]

* How to check End of Conversion?

↳ while (ADIGDR & (1<<31)==0); → decimal value → 1 left shift by 31 times

If this result is zero, then keep wait

or while (ADIGDR & (0x01<<31)==0); → Hexa value → Done bit → EOC

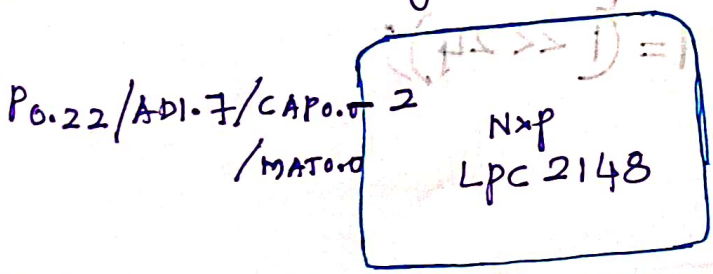
↳ If the above condition is failed, then fetch the ADC data.

* Write a function to acquire ADC value:

ADC = ADC-1 → channel no 7; AD1.7

Algorithm:

1) PIN SELECT Register:



PINSELECT (22, 1);

PINSELECT → Function

22 → P0.22

1 → First special function

↳ PINSELECT(22,1); This function call configures P0.22 Pin as AD1.7 [7th channel of ADC-1]

2) Reset ADC control Register:

ADICR = 0x00000000; Initially clear the register

3) Configuration of ADC1 control Register:

1) Setting the channel number

ADICR = (ADICR) | (1 << channel number);

ADICR = (ADICR) | (1 << 7);

or ADICR |= (1 << 7);

2) Setting the clock Divider (CLKDIV) value:

ADICR |= (13 << 8);

↳ Decimal value

or ADICR |= (0x0D << 8);

↳ Hexa value

3) Setting the Burst pin for Continuous conversion.

ADICR |= (1 << 16);

4) Setting the power down bit = ON (PDN Bit)

ADICR |= (1 << 21);

5) Setting the start of conversion: it has to be 001

ADICR |= (1 << 24);

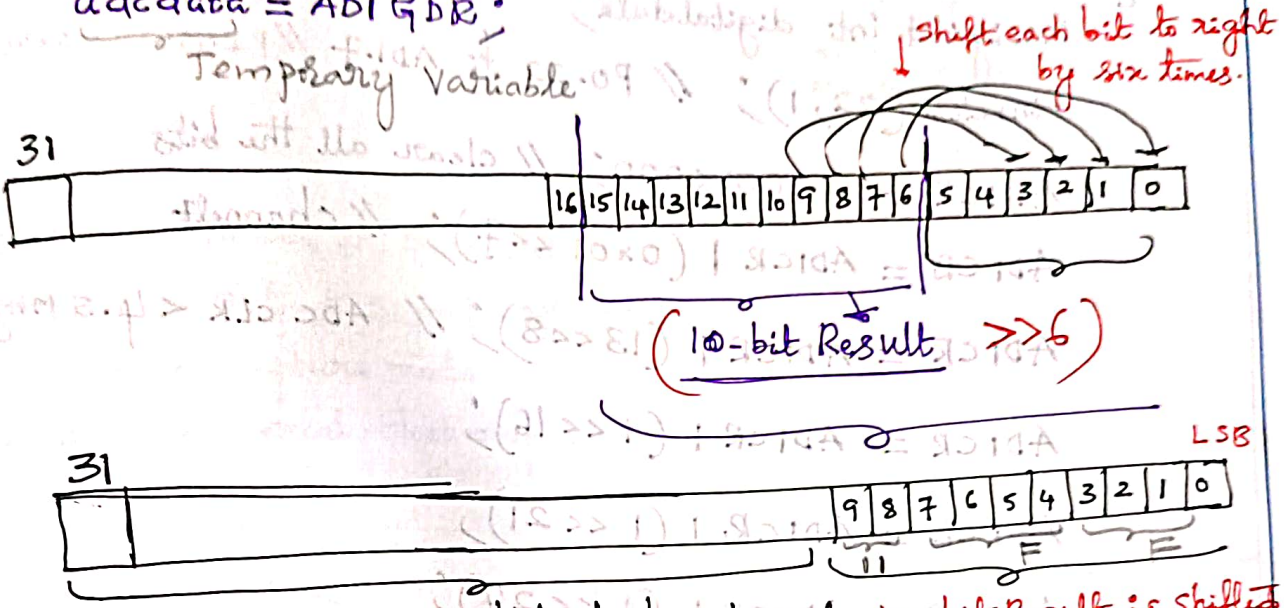
4) check for EOC

while(ADIGDR & (0x01 << 31) == 0);

If the above condition is failed, then read the digital value

5) adcdata = ADIGDR;

Temporary Variable



6) adcdata = (adcdata >>6) & 0x000003FF;

7) return adcdata;

NOTE: The Resolution of ADC = $\frac{3.3}{2^{10}} = \frac{3.3}{1024} = 0.00322 = 3.2\,mV$

2^{10} = 0 to 1023

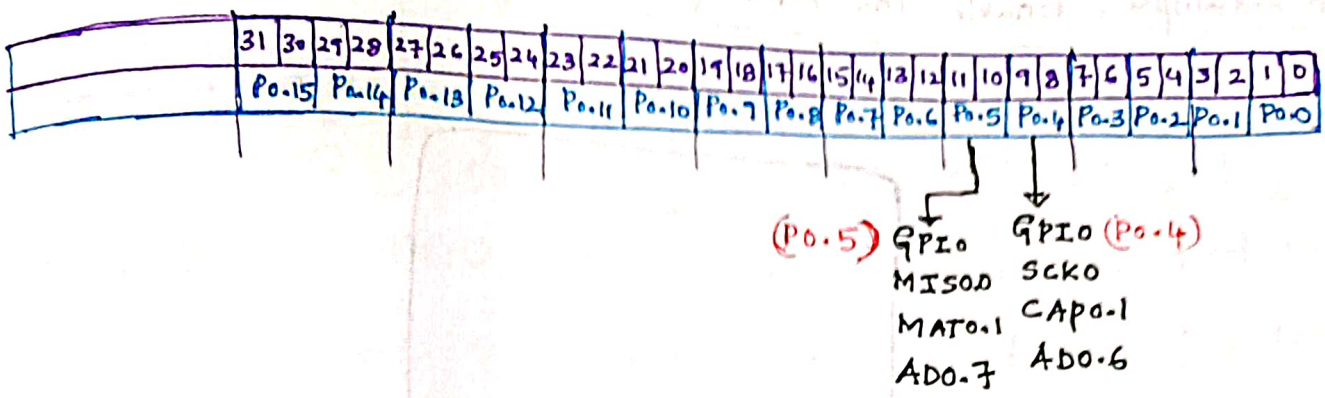
= 1024

* ADC function snippet :

```
unsigned int getadcdata()
{
    unsigned int digitaldata;
    Pinselect(22,1); // P0.22 to AD1.7 //PINSEL1 = 0x00001000
    AD1CR = 0x00000000; // clear all the bits
    AD1CR = AD1CR | (0x01<<7); // channel7
    AD1CR = AD1CR | (13<<8); // ADC CLK < 4.5 MHz
    AD1CR = AD1CR | (1<<16);
    AD1CR = AD1CR | (1<<21);
    AD1CR = AD1CR | (1<<24);
     // check for EOC //
    while ((AD1GDR & (1<<31))==0);
    digitaldata = AD1GDR;
    digitaldata = digitaldata >>6; // ADC result is available from 15:6
    digitaldata = digitaldata & 0x3FF;
    return digitaldata;
}
```

LBR

PINSEL0



MISO0 → Master In, Slave out

MAT0 → output Signals of Timer0

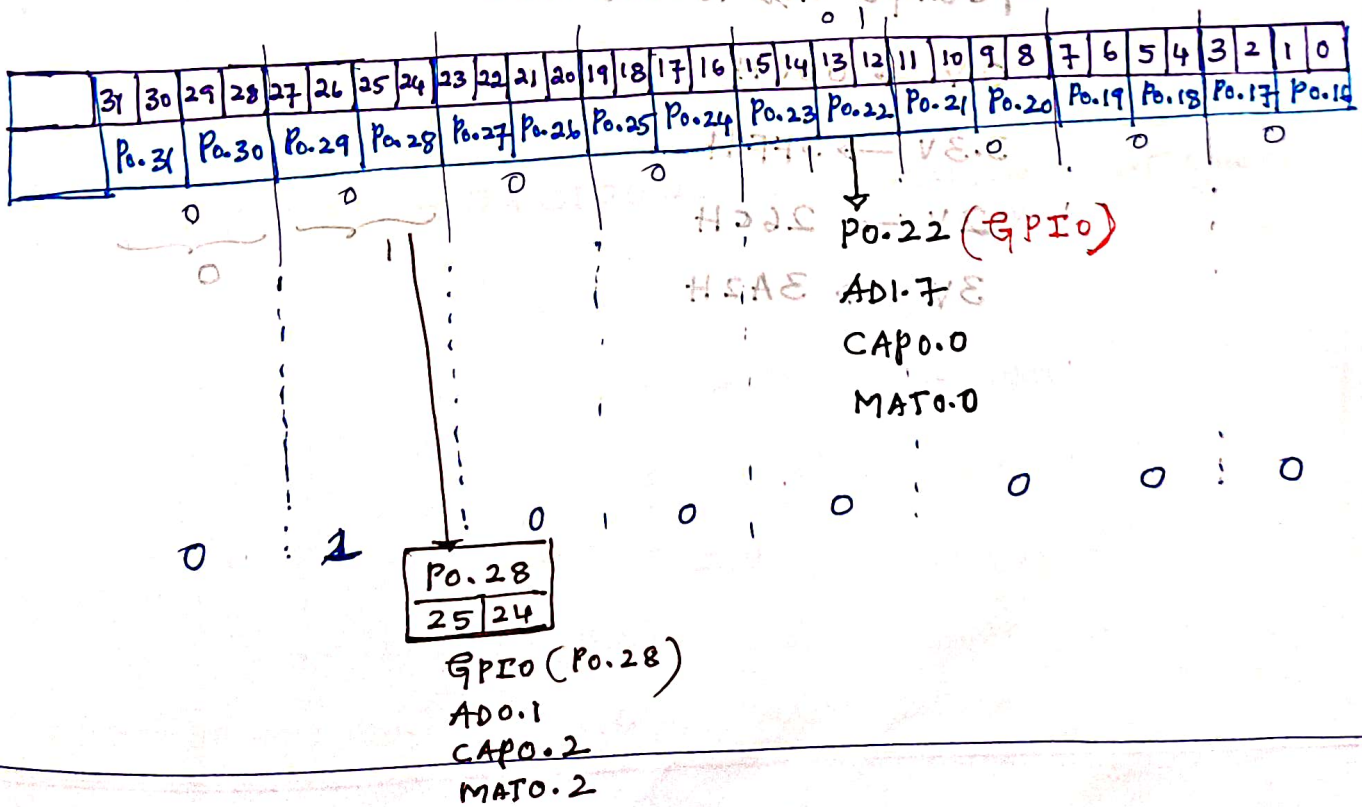
CAP0 → Capture mode

SCK0 → Serial clock output

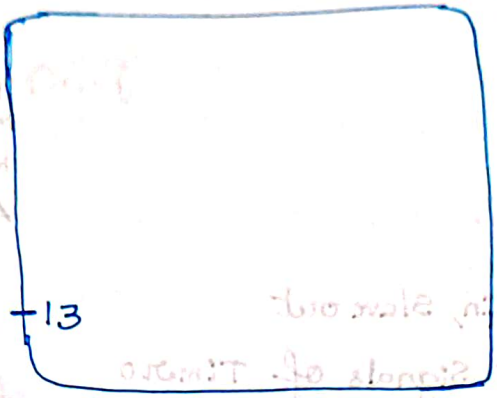
Ex: If we want to configure P0.4 as AD0.6 then Pins 8 & 9 will be both 1.

If we want to configure P0.5 as AD0.7 then Pins 10 & 11 will be both 1.

PINSEL1



* Example: Enable the ADC functionality of LPC-2148 at P0.28



P0.28/AD0.1/CAP | MAT +13

PINSEL1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P0.31	P0.30	P0.29	P0.28	P0.27	P0.26	P0.25	P0.24	P0.23	P0.22	P0.21	P0.20	P0.19	P0.18	P0.17	P0.16	P0.15	P0.14	P0.13	P0.12	P0.11	P0.10	P0.9	P0.8	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
				0	0	0	1																								
0	1	0	0	0	0	0	0																								

∴ PINSEL1 = 0x01000000;

NOTE: V_{ref} = 3.3V

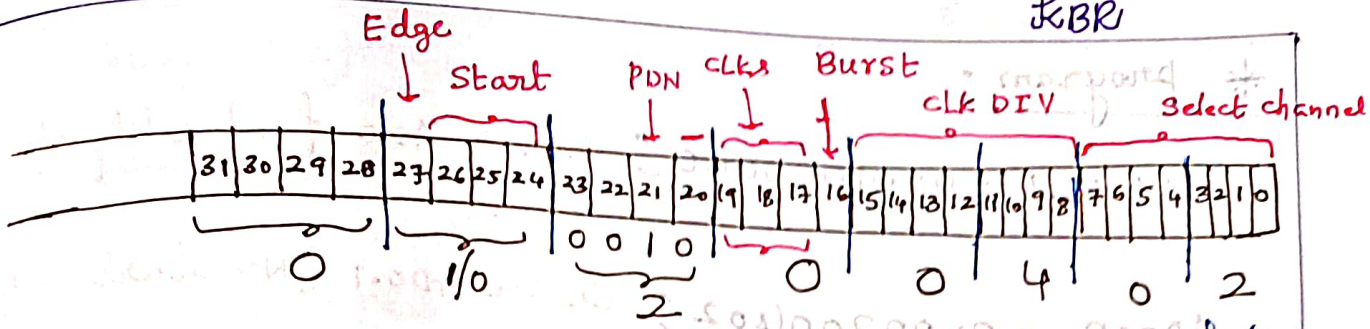
LPC2148 has 10-bit ADC

0V → 000H

3.3V → 3FFH

2V → 26CH

3V → 3A2H



Select channel: 0000 0010 = 0x02 → Select channel-1 of ADC0

CLK DIV: 0000 0100 = 0x04 → Divide by 5; 5-1=4

Burst: 0 = 0x0 → Disable the BURST conversion mode

CLKS: 000 → 10-bits Resolution mode

PDN: 1 → The A/D converter is operational

Start: 000 → No Start A/D conversion

001 → Start A/D Conversion

Edge: 0 → Don't care

∴ ADOCR = 0x00200402 ; Enable AD0.1 (No Start)

PINSEL1 = 0x01000000 ; Enabling AD0.1 channel: P0.28

* program :

```
PINSEL1 = 0x01000000;   Enable ADC functionality at P0.28
ADOCR = 0x00200402;     Enable AD0.1 (No start mode)
ADOCR = ADOCR | (1<<24);  Start AD0.1 Conversion
while(!(ADODR1 & 0x80000000));  Monitor bit DONE in ADODR1
(ADODR1>>6);
digital_data = (ADODR1>>6);   Read Equivalent digital value.
digital_data = digital_data & 0x000003FF;
```

* Embedded 'C' program :

```
#include <LPC214x.h>
int main (void)
{
   unsigned int digital_result;
   PINSEL1 = 0x01000000;
   ADOCR = 0x00200402;
   while (1)
      {
         ADOCR = ADOCR | (1<<24);
         while (!(ADODR1 & 0x80000000));
         digital_result = ADODR1;
         digital_result = (digital_result >> 6);
         digital_result = (digital_result & 0x000003FF);
      }
}
```

→ Debug (click)

→ peripherals

- GPIO slow Interface
- GPIO Fast Interface
- UART
- I^2C Interface
- SPI Interface
- Timer
- PWM
- A/D converter
- D/A Converter

A/D converter ——— A/D converter 0
A/D converter 1
A/D Global Start

A/D channel data

ADODR0 [ ]

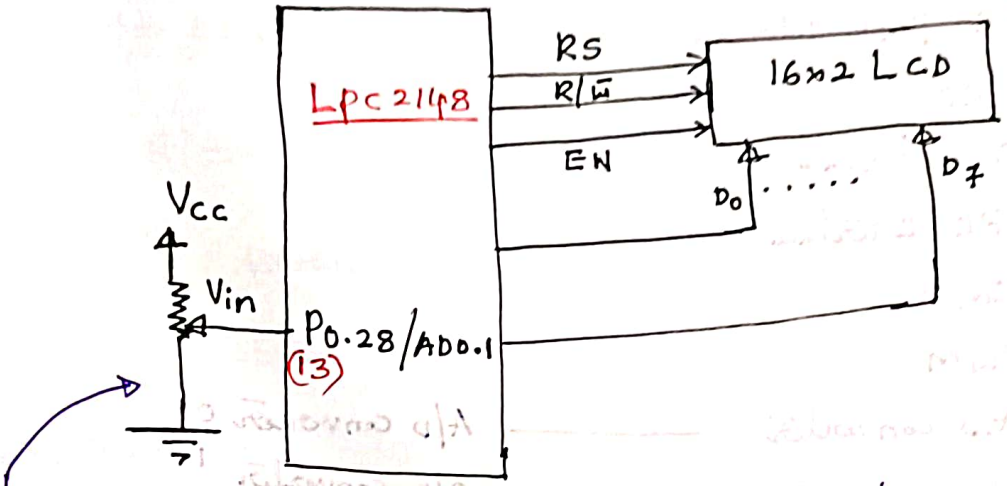
RESULT0 [ ]

RESULT1 [0x03A2]

Analog inputs:

AD01 [3.00]

* Interfacing diagram of Sensor and LCD Interfacing to LPC2148

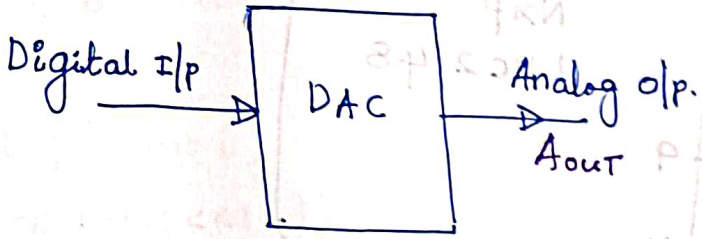


Analog Sensor / voltage divider circuit / potentiometer

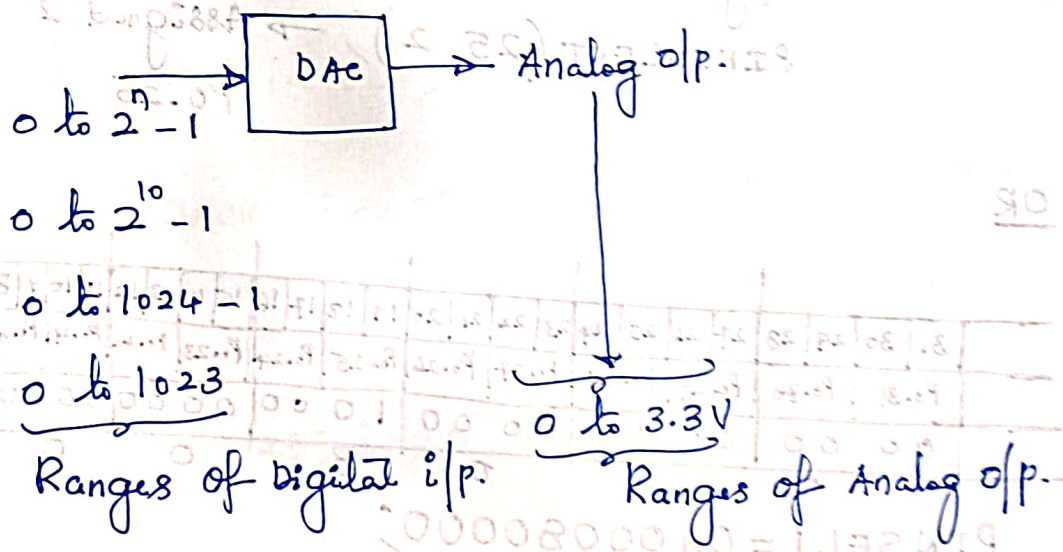
→ Converted digital value is displayed on LCD.

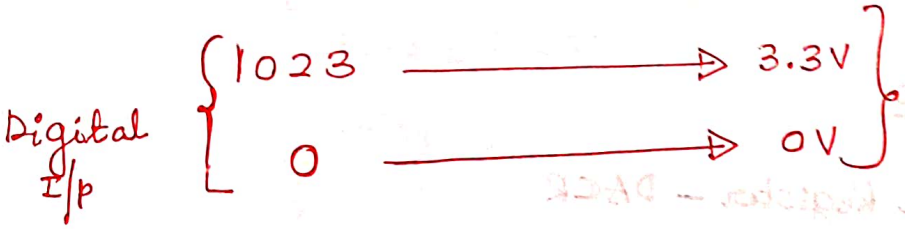
Prof K.B.Ramesh

* Digital to Analog Converter [DAC] LPC 2148 Microcontroller:



→ 10 bit resolution DAC.





Ex: For 650, what will be the Analog o/p voltage?

1023 → 3.3V

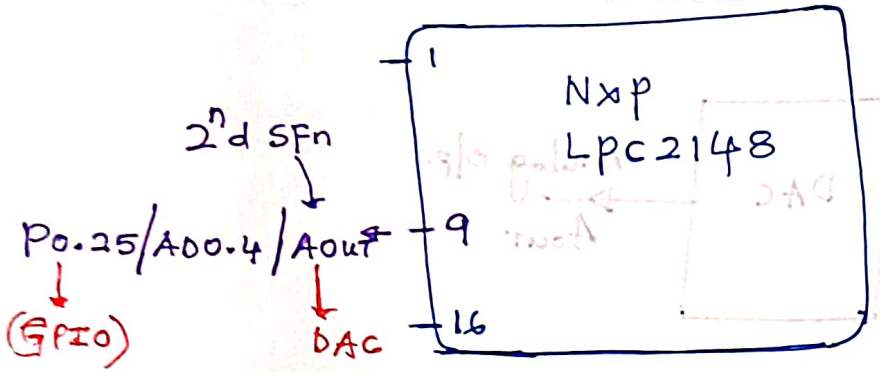
650 ? (Value)

$$\frac{650 \times 3.3}{1023} = 2V$$

→ Resistor string architecture.

NOTE: $A_{OUT} = \frac{Value}{1024} \times 3.3V$

or $Value = \frac{A_{out} \times 1024}{3.3V}$



→ How to configure pin9 as DAC?

PINSELECT(25, 2); → Assigned 2nd SFn to P0.25

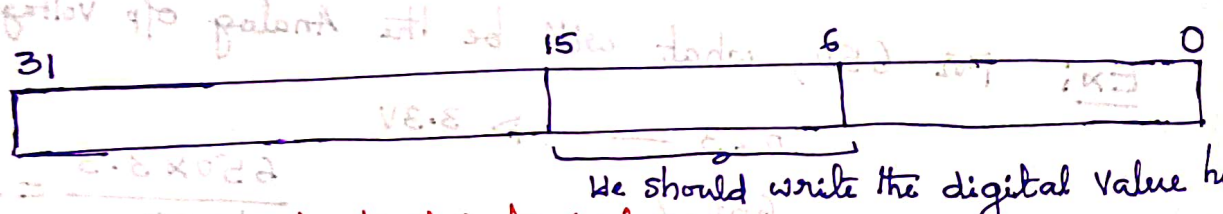
OR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P0.31		P0.30		P0.29		P0.28		P0.27		P0.26		P0.25		P0.24		P0.23		P0.22		P0.21		P0.20		P0.19		P0.18		P0.17		P0.16	
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0		0		0		0		0		0		8		0		0		0		0		0		0		0		0		0	

PINSEL1 = 0x00080000;

* SFR of DAC

↳ DAC Control Register – DACR



* Function to convert digital to Analog: digital_data

```
Void analogvalue(unsigned int dacoutput)   ↳ 0 to 1023
{
  Pinselect(25,2); //Convert P0.25/Pin9 to 2nd SFn DAC
  DACR = dacoutput<<6;      DACR = [digital_data]000000
         digital_data
}
```

* program to generate the Traingular waveform :

```
#include <Lpc214x.h>
int main (void)  ---> void delay (void);
{
   int i, j;  int i;
   PINSEL1 = (1<<19);
   DACR = 0;  // DACR = 0x00000000 // Initial Digital value is 0
   while (1)
    {
      for (i=0; i<1024; i++)
        {
          DACR = i<<6;
          delay ();
        }
      for (i=1023; i>=0; i--)
        {
          DACR = i<<6;
          delay ();
        }
    }
}
void delay ()
{
   int i;
   for (i=0; i<2000; i++);
}
```

For Sawtooth Waveform

```
while (1)
{
  for (i=0; i<1024; i++)
   {
     DACR = i<<6;
     delay ();
   }
  i=0;
  DACR = i<<6;
}
```

o/p waveform

check the result in Logic Analyzer window.

* program to generate the square waveform using DAC

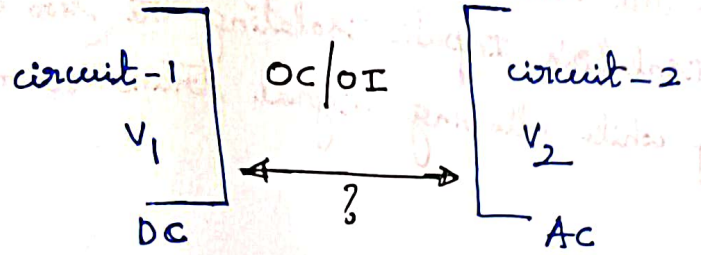
```
#include<Lpc214x.h>
Void delay (Void);
int main (Void)
{
    int i;
    PINSEL1 = (1 <<19);
    DACR = 0;
    while (1)
    {
        i=0;
        DACR = i<<6;
        delay ();
        i = 1023;
        DACR = i<<6;
        delay();
    }
}
void delay (Void)
{
    int i;
    for(i=0; i<100; i++);
}
```

LBR

* program to generate the Sine Wave from using DAC

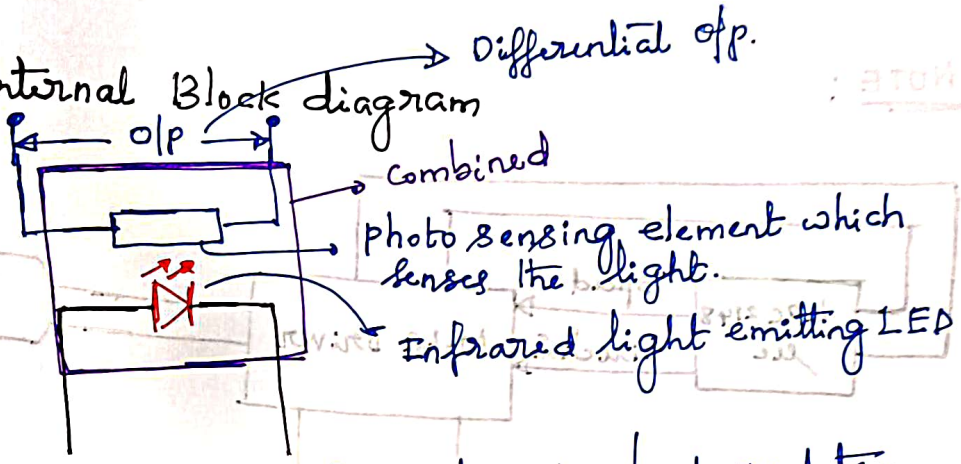
```
#include <Lpc214x.h>
#include <stio.h>
#include <math.h>
#define rad 0.0175   // 1° = 0.0174533 radians
int teta, dac-in;
float Sine_Val;
int main(void)
{
  PINSEL1 = 0x00080000;
    while(1)
      {
        for( teta=0; teta<360; teta++)
          {
            Sine_Val = 1.65 * Sin(teta * rad);
            dac_in = (((Sine_Val * 1024 / 3.3) + 512);
            DACR = (dac_in <<6);
          }
      }
}
```

* optocoupler / optoisolator :



opto-Isolators also known as optocouplers use light to transfer signals between two electrically isolated circuits.

Internal Block diagram

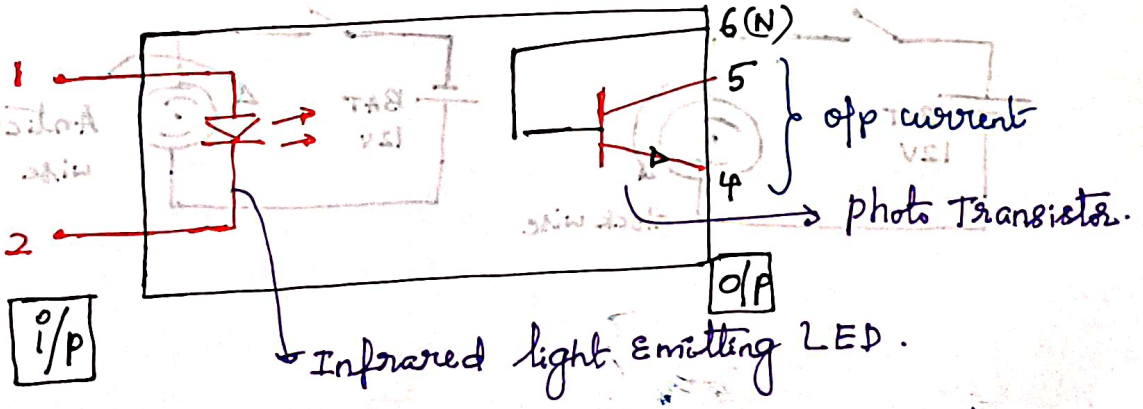


→ There are different types of optocouplers / opto Isolators based on their Internal structures.

→ Types: 1) photo Transistor
2) photo Darlington Transistor
3) photo - TRIAC
4) photo - SCR

NOTE: Darlington Transistor

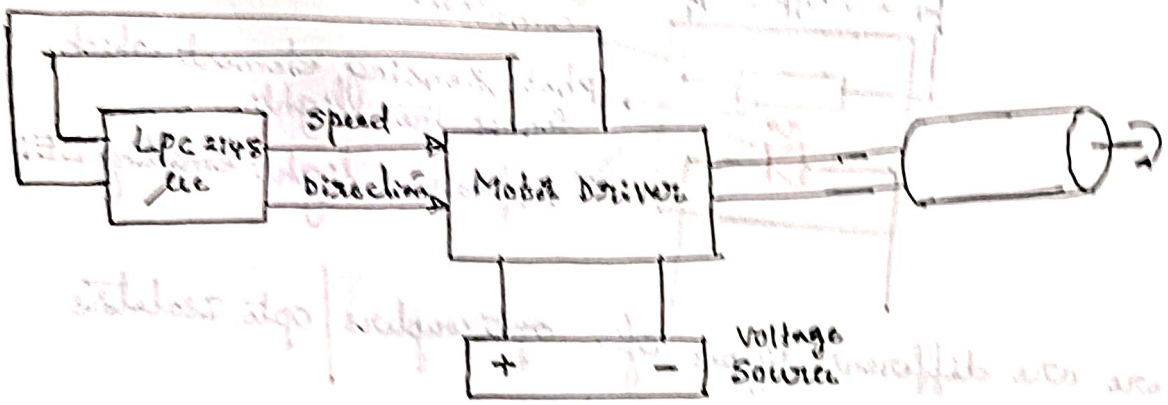
→ Construction of photo Transistor:



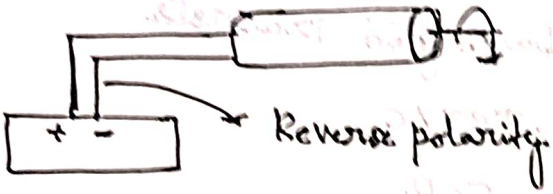
→ The base of the photo Transistor is controlled by light which is emitted by LED. Base is connected optically.

NOTE : Connect the opto isolator's output to a microcontroller's Gpio pin. The microcontroller's o/p can then control the opto isolator's input, isolating the two circuits electrically while allowing signal transmission.

NOTE :



Simple Connection :



DC Motor specifications

1) Motor voltage
2) Average current
3) Stall current

* In proteous Software :