CHAPTER (5) الباب الخامس

محول القيمة التناظرية إلى قيمة رقمية

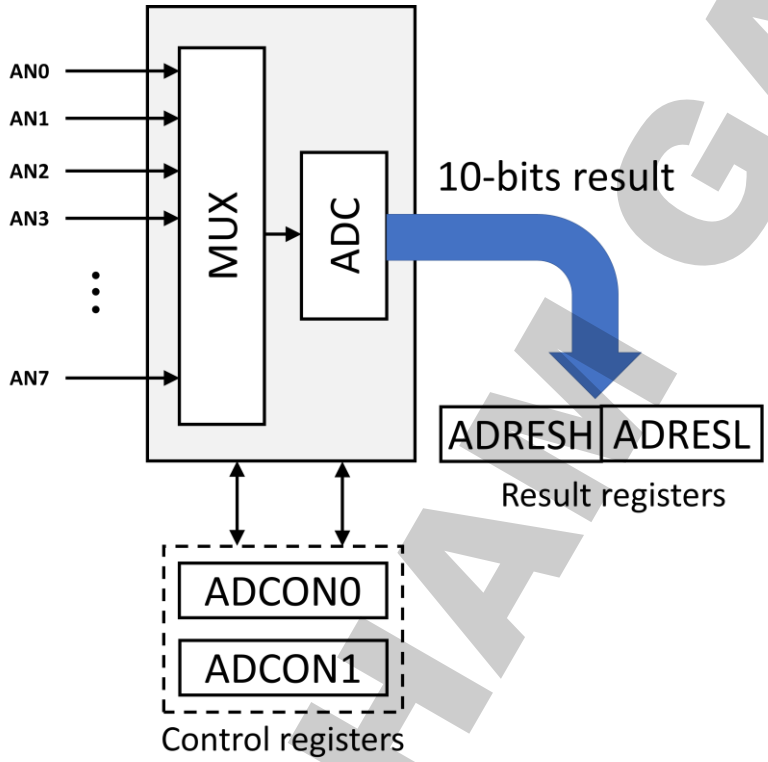
Analog-to-Digital Converter

يعتبر الـ Analog-to-digital converter من أهم الأشياء التى تستخدم فى الـ Embedded system وأنظمة الميكاترونيك بشكل عام . وذلك بسبب أن معظم أنظمة التحكم تحتوى على حساسات Sensors تعمل على إخراج جهود Voltages أو تيارات Currents تتناسب مع القيمة التى تقيسها هذه الحساسات أى أنها تسمى Analog sensors. وبما أن الـ Microcontrollers بشكل عام تعتبر من المكونات الـ Digital فإن يجب وجود أداة تعمل على تحويل القيم الـ Analog إلى قيم Digital يمكن إستخدامها فى برنامج الميكروكونترلر أو أى Embedded system كما هو موضح فى شكل 5-1.

شكل (5-1): وظيفة الـ Analog to digital converter.

بالنسبة للشريحة PIC18F452 فإنها تحتوى على ADC بدقة 10 bits أى أنه يستطيع إخراج 1023 level أى أنها تعمل على إخراج قيمة رقمية تتراوح بين 0 إلى 1023 حسب الجهد الداخل للمحول. كما أنه يستطيع قياس 8 جهود مختلفة عن طريق الأطراف AN0:AN7 والتى تسمى بإسم Channels. ولكن يجب أن تعلم أنه لا يمكن قياس كل هذه الجهود فى نفس الوقت لأن شريحة الـ PIC18F452 لا تحتوى إلى على ADC واحد فقط. لذلك يتم إستخدام Multiplexer للأختيار بين هذه الـ Channels كما هو موضح فى شكل (5-2). ويتم الأختيار بين هذه الـ Channels عن طريق الـ CHS2:CHS0 bits والموجودة داخل الـ ADCON0 register والذى سوف يتم شرحة لاحقاً. ويتم التحكم فى الـ ADC الذى بداخل هذا المايكروكونترلر عن طريق الـ ADCON0 register والـ ADCON1 register. ونظراً لأنه ناتج عملية التحويل أكبر من 8-bits، فإنه يتم تخزين ناتج

التحويل يكون مقسماً فى Two registers وهم ADRESL و ADRESH كما هو موضح فى نفس الشكل.



شكل (2-5): تركيب مُبسط للـ ADC الموجود داخل الشريحة PIC18F452.

وبسبب أن الأطراف AN0:AN7 موجودين مع نفس أطراف الـ PORTA والـ PORTE، فإن المستخدم يجب أن يكون لديه القدرة على الإختيار بين إمكانية تشغيل هذه الأطراف إما للـ PORT أو للـ ADC ويتم ذلك عن طريق الـ Port configuration bits أو (PCFG0:PCFG3) والموجودة بداخل الـ ADCON1 Register. حيث أنه إذا كانت هذه الـ Bits تساوى 0000 فإن كل الأطراف الموجودة على PORTA و PORTE سوف تصبح Analog أما إذا كانت قيم هذه الـ bits تساوى 0111، فإن هذه الأطرف سوف تصبح Digital أى أنها تابعة للـ PORTA والـ PORTE.

ويجب أن تعرف أن بالرغم من دخل الأطراف AN7:AN0 عبارة عن جهود تتراوح من 0v إلى 5v إلا أن الخرج النهائى للـ ADC بعد عملية التحويل عبارة عن قيمة 10 bits تتراوح بين 0 إلى 1023. ولكى يتم تحويل هذه القيمة الناتجة إلى قيمتها الأصلية فى البرنامج، فإنه يتم إستخدام هذه المعادلة:

$$\text{Measured voltage} = \text{Output from ADC} \times \frac{5v}{1023} \qquad (2.1)$$

فعلى سبيل المثال إذا كان خرج الـ ADC بعد إتمام عملية التحويل يساوى 256 فإن الجهد الذى تم قياسة يساوى:

$$\text{Measured voltage} = 256 \times \frac{5v}{1023} = 1.251\text{Volt}$$

وكما ذكرنا سابقاً يتم التحكم بالـ ADC converter عن طريق الـ ADCON0 register والـ ADCON1 register. كما يتم تخزين ناتج التحويل والمكون من 10 bits فى ADREH register والـ ADRESL register.

ويستخدم الـ **ADCON0 register** للأتى:

1. تشغيل وإطفاء والـ ADC.
2. إختيار الطرف الذى نريد تحويل الجهد منه من Analog إلى Digital.
3. إختيار سرعة التحويل Conversion speed.

أما الـ **ADCON1 register** فيستخدم للأتى:

1. إختيار حالة الأطراف الموجودة على PORTA و الـ PORTE من حيث كونها أطراف Analog أو Digital.
2. إختيار طريقة تخزين القيمة الخارجة من الـ ADC.

شرح الـ ASDCON0 Register

ADON bit:

يستخدم هذا الـ Bit لتشغيل وإطفاء الـ ADC converter فعندما تساوى الصفر فإن ADC سوف يكون OFF وعندما تكون 1 فإن الـ ADC سوف يكون ON. لذلك يعتبر هذا ADON bit من أهم الـ Bits الخاصة بالـ ADC converter.

GO/DONE bit:

يستخدم هذا الـ Bit لبدء عملية التحويل من Analog إلى Digital عندما يتم جعل قيمتها تساوى 1 من البرنامج الذى يكتبة المستخدم. وعندما تنتهى عملية التحويل (أى تخزين الناتج فى ADRESL و ADRESH)، فإن هذا الـ Bit يتحول بشكل آلى إلى الصفر لكى يتم يعرف المستخدم أن عملية التحويل قد أنتهت.

CHS2:CHS0 bits: (Channel select bits)

يستخدم هذه الـ Bits لإختيار الـ Channel الذى نريد قياسه أو تحويله من Analog إلى Digital كما هو موضح، فإذا كانت قيم هذه الـ Bits تساوى 000 فإنه سوف يتم توصيل المدخل AN0 إلى الـ ADC وفإذا كانت قيم هذه الـ Bits تساوى 111 فإنه سوف يتم توصيل الطرف AN7 بالـ ADC.

ADCS1:ADCS0 bits: (A/D conversion clock select bits)

تستخدم هذه الـ Bits لإختيار مصدر الـ Clock الخاص بالـ ADC. حيث أن مصدر الـ Clock يحدد سرعة التحويل Conversion speed. وتستخدم هذه الـ Bits مع ADCS2 bit والموجودة فى ADCON1 register. فعلى سبيل المثال إذا كانت قيمة Crystal oscillator الموصلة بالـ Microcontroller تساوى 4MHz وكانت قيمة هذه الـ Bits نساوى 001 فإن سرعة التحويل سوف تساوى:

$$\text{Conversion speed} = \frac{F_{osc}}{8} = \\ = \frac{4\times10^{6}}{8} = 500000 \ \text{sample/sec}$$

أى أن الـ ADC فى هذه الحالة يمكنه إجراء 500000 تحويله فى الثانية الواحدة.

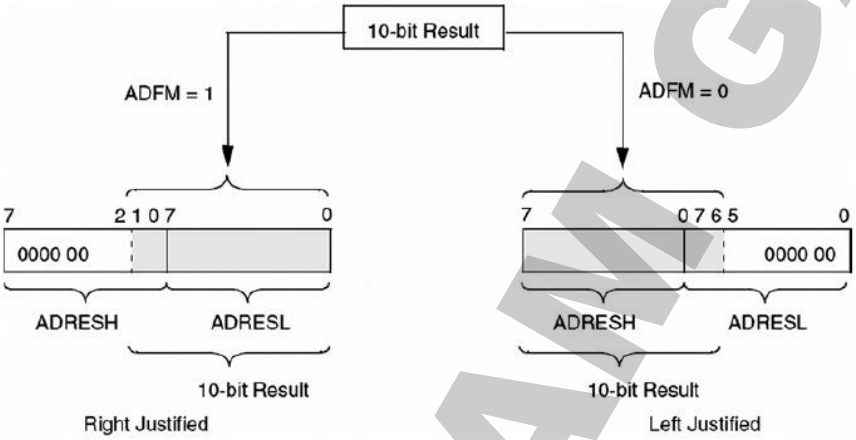
ADCON1 Register

PCFG3:PCFG0 bits:

تستخدم هذه الـ Bits لإختيار نوع الأطرف من حيث كونها Analog او Digital كما تم شرحه سابقاً. ويمكن الحوصول على تركيبات مختلفة، أى أنه يمكن جعل نصف الأطراف Analog والباقى Digital كما هو موضح فى الجدول المبين أسفل الـ ADCON1 register.

ADFM bit:

يستخدم هذا الـ Bit لإختيار طريقة تخزين الـ 10 bits الذين تم إخراجهم من الـ ADC فى الـ ADRESL و ADRESH حيث أنه يمكن تخزين هذه الـ Bits فى الجهه اليمنى Right justified أو اليسرى Left justified فى هذه الـ Registers كما هو موضح بشكل 3 . ويجب أن تعرف أن باقى الـ Bits الـ ADRESL والـ ADSRESH سوف تكون عبارة عن أصفار.



شكل (3-5): طريقة تخزين الناتج النهائى فى الـ ADRESL والـ ADRESH.

ADCON0 Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

bit 7-6 **ADCS1:ADCS0:** A/D Conversion Clock Select bits (ADCON0 bits in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	**00**	FOSC/2
0	**01**	FOSC/8
0	**10**	FOSC/32
0	**11**	FRC (clock derived from the internal A/D RC oscillator)
1	**00**	FOSC/4
1	**01**	FOSC/16
1	**10**	FOSC/64
1	**11**	FRC (clock derived from the internal A/D RC oscillator)

bit 5-3 **CHS2:CHS0:** Analog Channel Select bits
000 = channel 0, (AN0)
001 = channel 1, (AN1)
010 = channel 2, (AN2)
011 = channel 3, (AN3)
100 = channel 4, (AN4)
101 = channel 5, (AN5)
110 = channel 6, (AN6)
111 = channel 7, (AN7)

Note: The PIC18F2X2 devices do not implement the full 8 A/D channels; the unimplemented selections are reserved. Do not select any unimplemented channel.

bit 2 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
When ADON = 1:
1 = A/D conversion in progress (setting this bit starts the A/D conversion which is automatically cleared by hardware when the A/D conversion is complete)
0 = A/D conversion not in progress

bit 1 **Unimplemented:** Read as '0'

bit 0 **ADON:** A/D On bit
1 = A/D converter module is powered up
0 = A/D converter module is shut-off and consumes no operating current

ADCON1 Register

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified. Six (6) Most Significant bits of ADRESH are read as '0'.
0 = Left justified. Six (6) Least Significant bits of ADRESL are read as '0'.

bit 6 **ADCS2:** A/D Conversion Clock Select bit (ADCON1 bits in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

bit 5-4 **Unimplemented:** Read as '0'

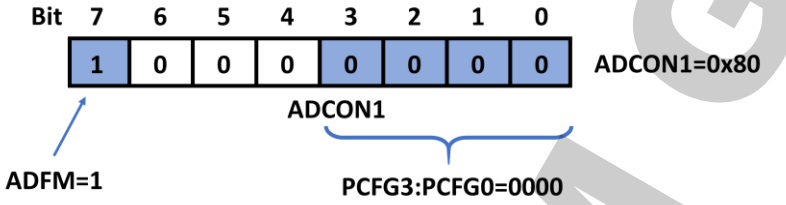
bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C / R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8 / 0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7 / 1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5 / 0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4 / 1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3 / 0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2 / 1
011x	D	D	D	D	D	D	D	D	—	—	0 / 0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6 / 2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6 / 0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5 / 1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4 / 2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3 / 2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2 / 2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1 / 0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1 / 2

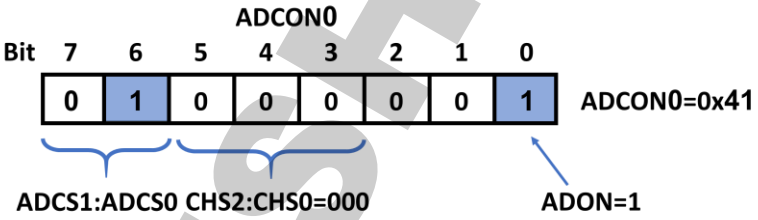
A = Analog input D = Digital I/O
C/R = # of analog input channels / # of A/D voltage references

خطوات إستخدام الـ ADC converter:

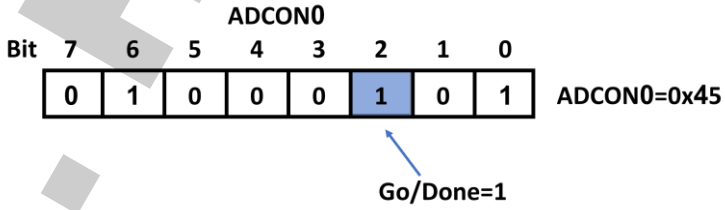
1. يتم ضبط الأطراف الموجودة على PORTA و PORTE بحيث تكون Analog عن طريق PCFG3:PCFG0 والموجودين بداخل **ADCON1 register** مع جعل ADFM=1 لكى يتم تخزين الناتج فى الجانب الأيمن Right justified .



2. يتم إختيار المدخل المطلوب قياسه عن طريق الـ CHS2:CHS0 bits وليكن المدخل AN0.
3. يتم إختيار سرعة التحويل المطلوبة ولتكن Fosc/32 عن طريق ADCS2:ADCS0 والموجودين بداخل ADCON0, ADCON1 registers ، كما يتم جعل ADON=1 ولذلك لتشغيل ADC والموجود ADCON0 register.



4. يتم الإنتظار **لمدة 1 ميللى ثانية** قبل بدأ عملية التحويل وذلك لتمهيد الـ ADC.
5. يتم بدأ عملية التحويل من Analog إلى Digital عن طريق جعل GO/DONE=1 والموجود بالـ ADCON0 Register.



6. يتم التحقق بشكل مستمر الـ GO/DONE bit إلى أن تصبح قيمتها تساوى 0 وهذا يعنى إنتهاء عملية التحويل.

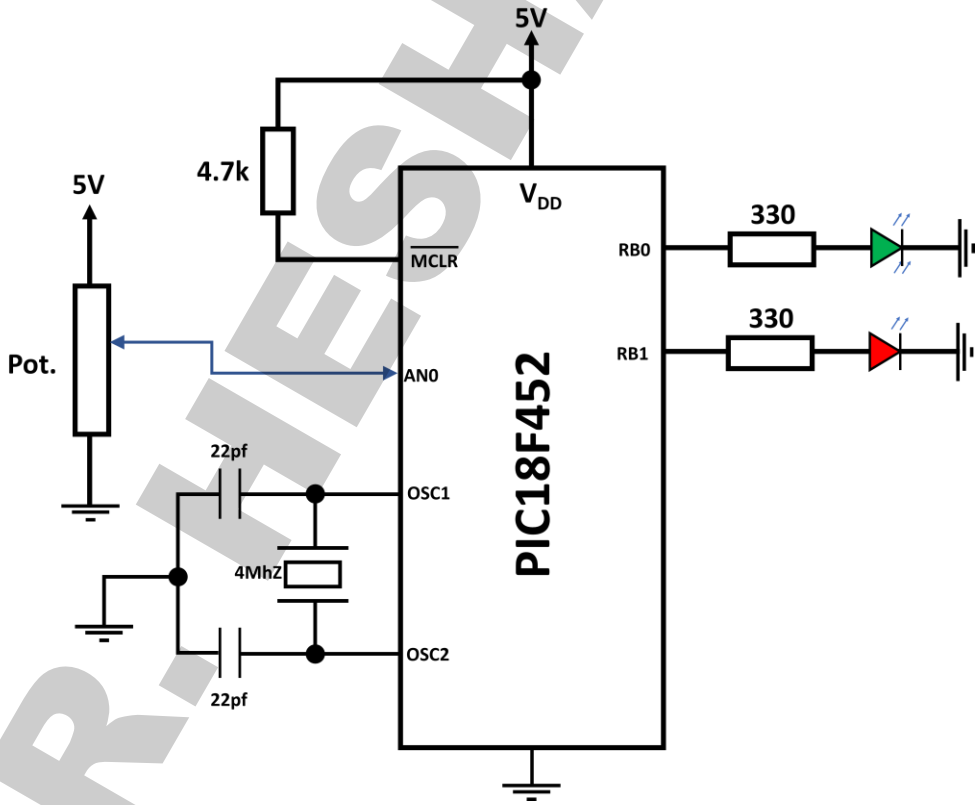
7. يتم قراءة القيمة المخزنة على ADRESH, ADRESH وتحويلها إلى جهد عن طريق المعادلة (2.1) لكى يتم إستخدامها فى البرنامج للأغراض المختلفة.

Example (1)

Draw a circuit and write a microcontroller program by using PIC18F452 to read the voltage on the pin AN0 from potentiometer. If the voltage is higher than 2.5V then turn on red LED connected to pin RB0 and turn ON a green LED if the voltage is lower than 2.5V.

Solution:

فى هذا المثال يريد أن يشغل الـ LEDs تبعاً للجهد المقاس. فإذا كان الجهد المقاس أعلى من 2.5v يتم تشغيل الـ LED الموصل بالطرف RB0 وإذا كان الجهد المقاس أقل من 2.5v يتم تشغيل الـ LED الموصل بالطرف RB1.



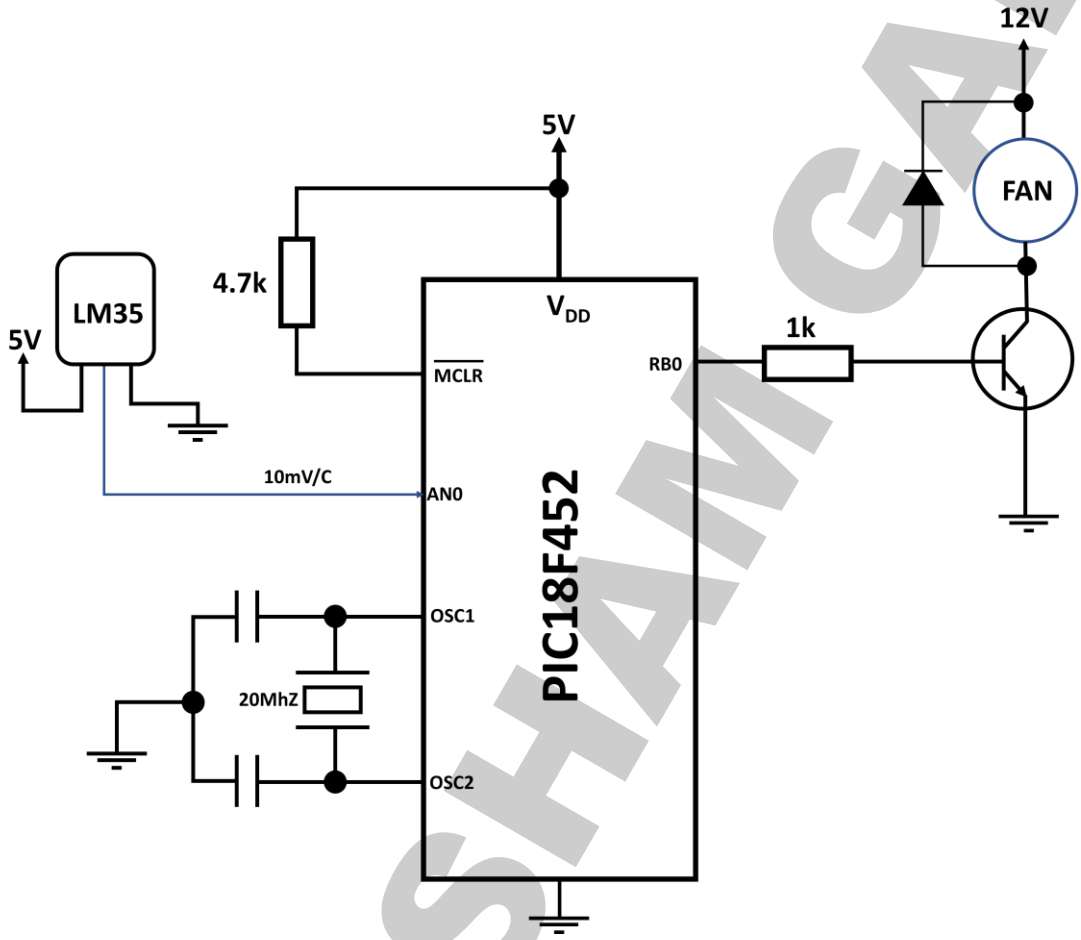
Program:

```
void main ()
{
 float volt, temp;
 TRISB=0x00;                          //جعل البورت خرج
 PORTB=0;                             //تصفير الخرج فى البداية
 ADCON1=0x80;                         // إعداد المحول
 ADCON0=0x41;
while (1)
        {
          delay_ms(1);                //أنتظر لمدة 1 ميللى ثاينة قبل بدء عملية التحويل
          ADCON0=0x45;                      // أبدأ عملية التحويل
          while (ADCON0.b2==1);             // أنتظر حتى تنتهى عملية التحويل
                                            //خزن ناتج عملية التحويل فى مخزن
          temp=(float)ADRESH*256.0+(float)ADRESL;
          volt=temp*(5.0/1023.0);           //استخدم المعادلة
          if(volt>=2.5)                     //تحقق من الجهد المقاس
                PORTB=0x01;                 //شغل الليد الأول
          else
                 PORTB=0x02;                //شغل الليد الثانى
         }
}
```

Example (2)

Draw a complete circuit and write a microcontroller program by using PIC18F452 to measure the temperature by using the sensor LM35DZ, which is connected to the pin AN0. The output from the sensor is (10mV/C^{o}). If the temperature is higher than 50 ^{o}C then turn ON a 12V fan, which is connected through a transistor to RB0 pin, otherwise turn it OFF.

Solution:



فى هذا المثال يريد التحكم بمروحة Fan عن طريق درجة الحرارة المقاسة عن طريق الحساس LM35، بحيث تعمل المروحة عندما تكون درجة الحرارة أعلى من 50 درجة. فى هذا المثال أن العلاقة بين درجة الحرارة المُقاسة وخرج الحساس هى 10mV لكل درجة مئوية. أى أنه لمعرفة درجة الحرارة يجب قياس الجهد الخارج من الـ Sensor بالـميللى فولت ثم يتم قسمة هذا الجهد على 10 للحصول على درجة الحرارة المُقاسة.

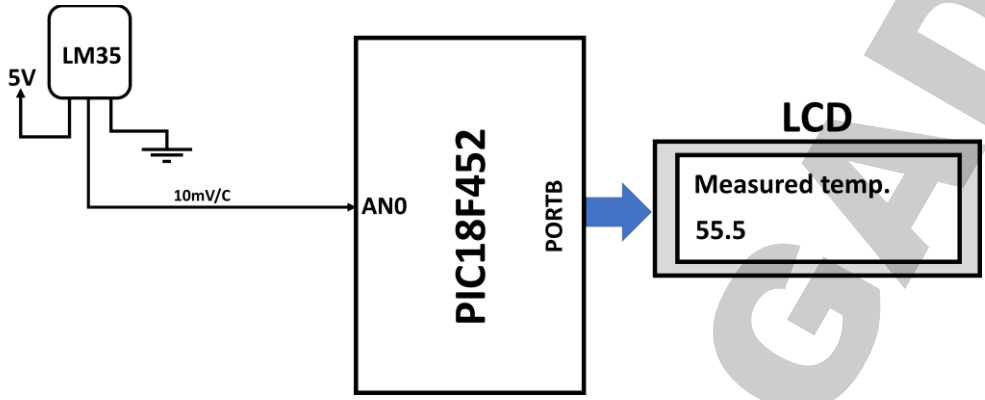
Program:

```
void main ()
{
 float volt, temp, temperature;
TRISB=0x00;                          //جعل البورت خرج
```

```
PORTB=0;                          تصفير الخرج فى البداية//
ADCON1=0x80;                      // إعداد المحول
ADCON0=0x41;


while (1)
        {
         delay_ms(1);             أنتظر لمدة 1 ميللى ثاينة قبل بدء عملية التحويل//
         ADCON0=0x45;             // أبدأ عملية التحويل
         while (ADCON0.b2==1);    // أنتظر حتى تنتهى عملية التحويل
                                  خزن ناتج عملية التحويل فى مخزن//
         temp=(float)ADRESH*256.0+(float)ADRESL;
         volt=temp*(5.0/1023.0);  //استخدم المعادلة
         volt=volt*1000.0;        //إضرب الجهد الناتج فى 1000 لتحويله إلى ميللى فولت
         temperature=volt/10.0;   // يتم القسمة على 10 للحصول على درجة الحرارة
         if(temperature>=50.0)    // يتم التحقق من درجة الحرارة
               PORTB=0x01;
         else
               PORTB=0x00;
        }
}
```

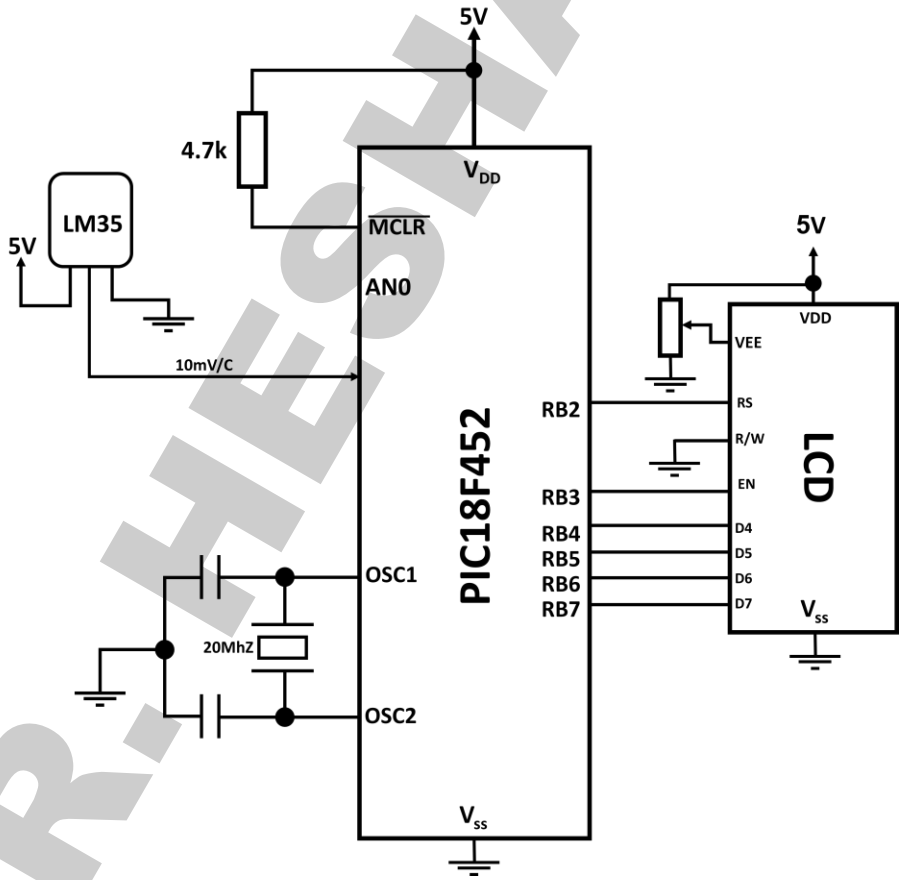
Example (3)

Draw a complete circuit and write a microcontroller program by using PIC18F452 to measure the temperature by using the sensor LM35DZ, which is connected to the pin AN0. Display the temperature on the LCD screen as shown in Fig. 4. The LCD is connected to PORTB and the measured value should be refreshed every 0.5 second.



Solution:

فى هذا البرنامج مطلوب قياس درجة الحرارة عن طريق الحساس LM35 الموصل بالطرف AN0 وعرض هذه القيمة على شاشة الـ LCD الموصل بالـ PORTB كما هو موضح فى شكل 4.



Program:

```
sbit LCD_RS at RB2_bit;
sbit LCD_EN at RB3_bit;
sbit LCD_D4 at RB4_bit;
sbit LCD_D5 at RB5_bit;
sbit LCD_D6 at RB6_bit;
sbit LCD_D7 at RB7_bit;
```

توصيلات الشاشة إلى أطراف الـ PORTB

```
sbit LCD_RS_Direction at TRISB2_bit;
sbit LCD_EN_Direction at TRISB3_bit;
sbit LCD_D4_Direction at TRISB4_bit;
sbit LCD_D5_Direction at TRISB5_bit;
sbit LCD_D6_Direction at TRISB6_bit;
sbit LCD_D7_Direction at TRISB7_bit;
```

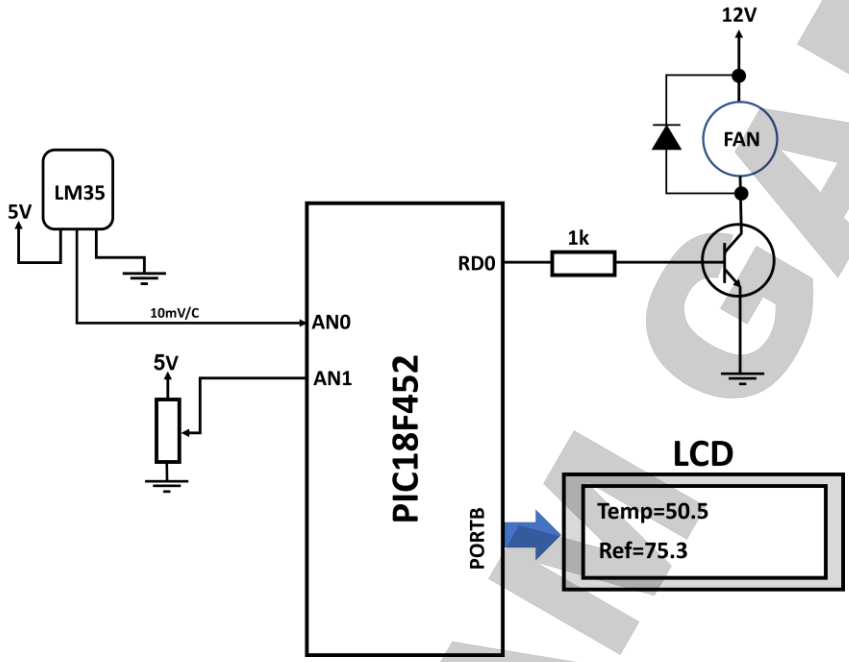
تحديد إتجاه الأطراف الخاصه بتوصيلات الشاشة

```
void main()
{
float  res, temperature;
char txt [15];                  // مخزن لتخزين الرسالة التى سوف تعرض على الشاشة
ADCON1=0x80;                    //  Right justified إعداد الـ ADC لكى تكون الأطراف كلها Analog مع عمل الـ

lcd_Init();                     // إعداد الشاشة
lcd_cmd(_lcd_clear);            //مسح الشاشة
lcd_cmd(_lcd_cursor_off);       //إطفاء المؤشر الذى على الشاشة

while (1)
    {
      ADCON0=0x41;              //  تشغيل الـ ADC مع إختيار الطرف AN0 وسرعة التحويل
      delay_ms(1);              // إنتظر 1 ميللى ثانية
      ADCON0=0x45;              //إبدأ عملية التحويل
      while (ADCON0.b2==1);     //انتظر حتى تنتهى عملية التحويل

       res=(float)ADRESH*256.0+(float)ADRESL;     //خزن ناتج التحويل فى مخزن
       res=res*(5.0/1023.0)*1000.0;               //حول ناتج التحويل إلى ميللى فولت
       temperature=res/10.0;                      // إحسب درجة الحرارة
```

```
        lcd_cmd(_lcd_clear);                أمسح الشاشة//
        lcd_Out(1,1,"Measured temp");       أعرض رسالة درجة الحرارة على السطر الأول//
        FloatToStr(temperature, txt);       // حول درجة الحرارة إلى حروف
        lcd_out (2,1,txt);                  أعرض درجة الحرارة على السطر الثانى//

        delay_ms(500);                      إنتظر لمدة نصف ثانية قبل القراءة التالية//
        }
}
```

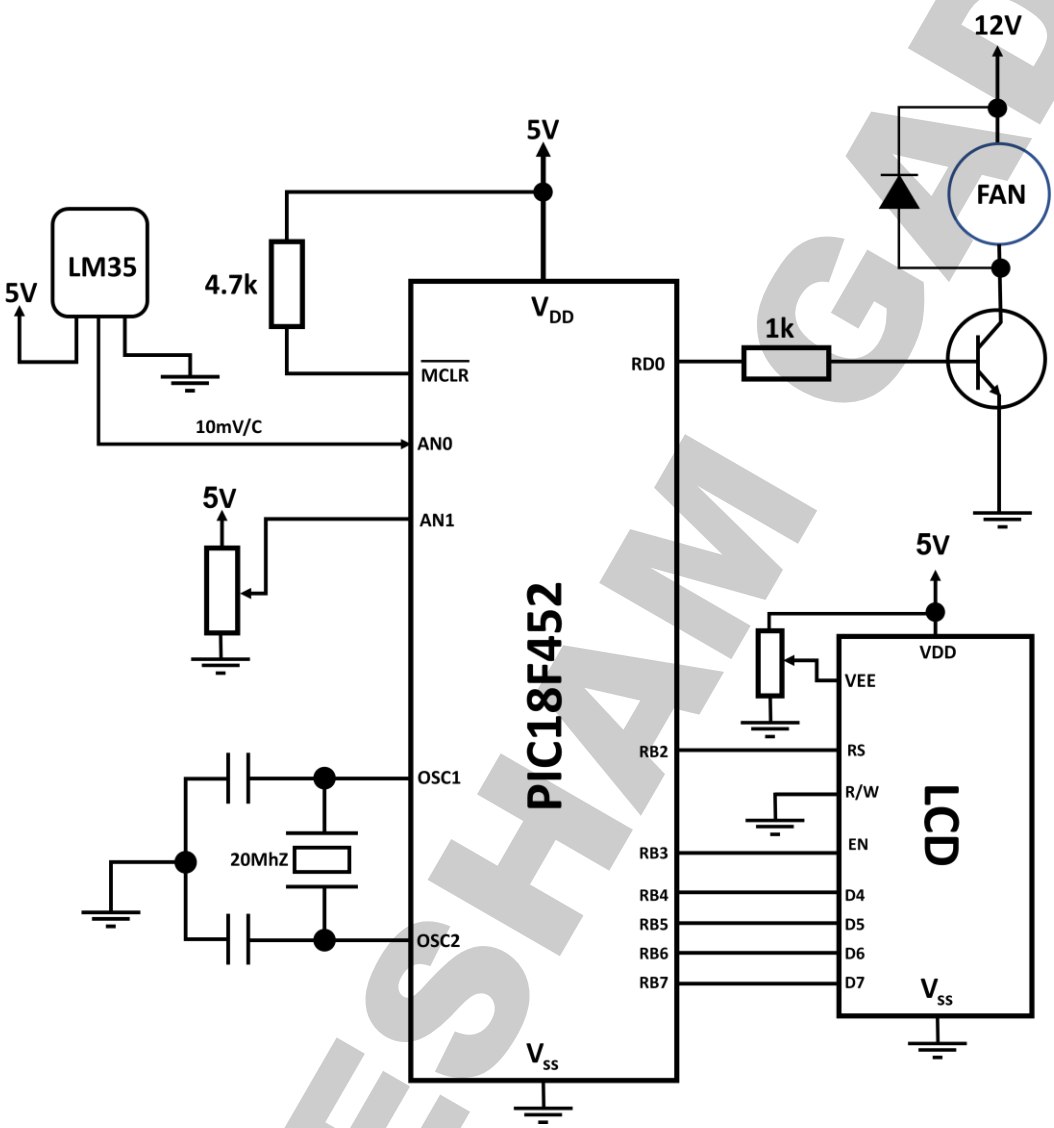
Example (3)

Draw a complete circuit and write a microcontroller program by using PIC18F452 to measure the temperature by using the sensor LM35DZ, which is connected to the pin AN0. The reference temperature has been adjusted through a potentiometer, which is connected to the pin AN1. The purpose of the potentiometer is to choose a value from 0 ºC to 100 ºC. If the measured temperature is higher than reference value, then the fan is ON, which is connected through a transistor to RD0 pin, otherwise turn it OFF. Also, display the temperature on the first line of the LCD and the reference value on the second line. The LCD is connected to PORTB. The program is refreshed every 0.5 second.



Solution:

فى هذا البرنامج مطلوب قياس درجة الحرارة عن طريق الحساس LM35 الموصل بالطرف AN0 ومقارنة هذه الدرجة بالـ Potentiometer (الموصل بالطرف AN1). فإذا كانت درجة الحرارة المقاسة أعلى من قيمة الـ Ref القادمة من الـ Potentiometer، فإن الميكروكونترلر سوف يعمل على تشغيل المروحة الموصلة بالطرف RD0. مع العلم أنه سوف يتم معايرة الجهد الخارج من الـ Potentiometer من 0 إلى 100 بدلاً من 0 فولت إلى 5 فولت.



Program:

```
sbit LCD_RS at RB2_bit;
sbit LCD_EN at RB3_bit;
sbit LCD_D4 at RB4_bit;
sbit LCD_D5 at RB5_bit;
sbit LCD_D6 at RB6_bit;
sbit LCD_D7 at RB7_bit;

sbit LCD_RS_Direction at TRISB2_bit;
sbit LCD_EN_Direction at TRISB3_bit;
sbit LCD_D4_Direction at TRISB4_bit;
sbit LCD_D5_Direction at TRISB5_bit;
sbit LCD_D6_Direction at TRISB6_bit;
sbit LCD_D7_Direction at TRISB7_bit;
```

```
void main()
{
float  res, ref, temperature;
char txt1[15];
char txt2[15];

TRISD=0x00;
PORTD=0;
ADCON1=0x80;                    //  إعداد الـ ADC لكى تكون الأطراف كلها Analog مع عمل الـ Right justified

Lcd_Init();
lcd_cmd(_lcd_clear);
lcd_cmd(_lcd_cursor_off);

while (1)
{
     ADCON0=0x41;
     delay_ms(1);
     ADCON0=0x45;
     while (ADCON0.b2==1);

     res=(float)ADRESH*256.0+(float)ADRESL;
     res=res*(5.0/1023.0)*1000.0;
     temperature=res/10.0;

     ADCON0=0x49;
     delay_ms(1);
     ADCON0=0x4D;
     while (ADCON0.b2==1);
```

تشغيل الـ ADC مع إختيار AN0
والأنتظار لمدة 1ms ثم بدأ التحويل
ثن الإنتظار حتى ينتهى التحويل

تخزين ناتج التحويل فى مخزن الـ res
ثم تحويل هذا الناتج إلى ميللى فولت
وأخيراً حساب درجة الحرارة

تشغيل الـ ADC مرةً أخرى مع إختيار
الـ AN1 هذه المرة ثم الإنتظار 1ms
ثم بدأ التحويل ثم الإنتظار حتى ينتهى

```
    res=(float)ADRESH*256.0+(float)ADRESL;
    res=res*(5.0/1023.0);
    ref=res*(100.0/5.0);
```

تخزين ناتج التحويل فى مخزن الـ res ثم تحويل هذا الناتج إلى فولت وأخيراً حساب قيمة الـ ref مع معايرته بحيث تكون أقصى قيمة له تساوى 100 عن الـ 5v

```
    lcd_cmd(_lcd_clear);
    lcd_Out(1,1,"Temp=");
    FloatToStr(temperature, txt1);
    lcd_out_cp(txt1);
```

1. مسح الشاشة
2. عرض رسالة Temp= على السطر الأول
3. تحويل درجة الحرارة إلى characters
4. عرض درجة الحرارة على الشاشة فى نفس السطر

```
    lcd_Out(2,1,"Ref=");
    FloatToStr(ref, txt2);
    lcd_out_cp(txt2);
```

1. عرض رسالة Ref= على السطر الثانى
2. تحويل قيمة الـ ref إلى characters

```
    if(temperature>=ref)
            PORTD=0x01;
    else
            PORTD=0x00;
```

إذا كانت درجة الحرارة أكبر من أو تساوى قيمة الـ ref يتم تشغيل المروحة، وإذا لم يكن يتم إطفاء المروحة.

```
     delay_ms(500);              // أنتظر لمدة نصف ثانية
  }

}
```