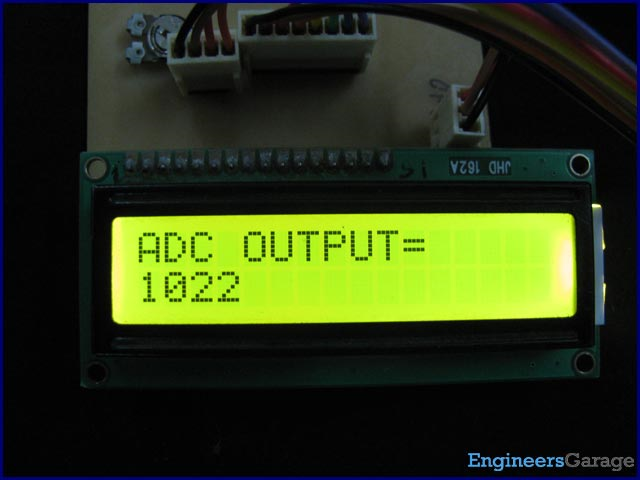
**[How to work with inbuilt ADC Module of PIC Microcontroller (PIC18F4550)](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/adc-circuit" \o "How to work with inbuilt ADC Module of PIC Microcontroller (PIC18F4550))**



A [microcontroller](http://www.engineersgarage.com/microcontroller), a digital device, can read, execute and transmit only digital signals. On the contrary, the outputs of the most of the transducers are analog in nature. Thus it is hard to interface these transducers directly with controllers. Analog-to-digital convertor (ADC) ICs are one way to make the analog input compatible with the microcontroller.

Using an external ADC adds complexity to the circuit. To avoid this complexity, [PIC Microcontrollers](http://www.engineersgarage.com/articles/pic-microcontroller-tutorial) have in-built ADC module which reduces the cost and connections of the circuit. This article explains the in-built ADC of [PIC18F4550](http://www.engineersgarage.com/electronic-components/pic18f4550-microcontroller) controller.

As mentioned in the summary, a [PIC microcontroller](http://www.engineersgarage.com/articles/pic-microcontroller-tutorial) has inbuilt ADC for A/D conversion. The ADC module of [PIC18F4550](http://www.engineersgarage.com/electronic-components/pic18f4550-microcontroller) controller has following specifications:

·         10-bit resolution output which means that an analog input gets converted into a corresponding 10-bit digital output.

·         13 channels which means that a total of 13 analog signals can be converted simultaneously into digital.

·         Vref+ (RA3) and Vref- (RA2) pins for external reference voltage.

·         8 selectable clock options.

·         ADC can be in auto-triggering mode for continuous A/D conversion.

**ADC Registers:**

To work with the inbuilt ADC of this PIC microcontroller, the certain registers are required to be configured. Each of these ADC registers has been explained below.

1. **ADCON0 (A/D CONTROL REGISTER 0)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| — | — | CHS3 | CHS2 | CHS1 | CHS0 | GO/DONE | ADON |

**ADON:** This bit is used to enable/disable the ADC peripheral of the PIC.

1 = A/D converter module is enabled

0 = A/D converter module is disabled

**GO/DONE:** This is A/D conversion status bit. For ADON=1,

1 = A/D conversion in progress

0 = A/D Idle

**CHS3: CHS0:** These bits are used to select a particular analog channel from 13 available channels (0-12) which are multiplexed with digital I/O pins. The following table shows the bit configuration to select these analog channels:

|  |  |  |
| --- | --- | --- |
| **CHS3:CHS0** | **Analog Channel** | **Pin** |
| 0000 | Channel 0 | RA0/AN0 |
| 0001 | Channel 1 | RA1/AN1 |
| 0010 | Channel 2 | RA2/AN2 |
| 0011 | Channel 3 | RA3/AN3 |
| 0100 | Channel 4 | RA5/AN4 |
| 0101 | Channel 5 | RE0/AN5 |
| 0110 | Channel 6 | RE1/AN6 |
| 0111 | Channel 7 | RE2/AN7 |
| 1000 | Channel 8 | RB2/AN8 |
| 1001 | Channel 9 | RB3/AN9 |
| 1010 | Channel 10 | RB1/AN10 |
| 1011 | Channel 11 | RB4/AN11 |
| 1100 | Channel 12 | RB0/AN12 |

2. **ADCON1 (A/D CONTROL REGISTER 1)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| — | — | VCFG1 | VCFG0 | PCFG3 | PCFG2 | PCFG1 | PCFG0 |

**PCFG0:PCFG3:** As mentioned earlier, there are 13 analog channels in PIC18F4550 which are multiplexed with digital I/O pins. This means that such a (multiplexed) pin can act as either a digital I/O pin or an analog input pin. Either of these configurations is selected by these bits. The following table shows the bit configuration to make a pin D (Digital I/O) or A (Analog input):

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **PCFG3:**  **PCFG0** | **AN12** | **AN11** | **AN10** | **AN9** | **AN8** | **AN7** | **AN6** | **AN5** | **AN4** | **AN3** | **AN2** | **AN1** | **AN0** |
| 0000 | A | A | A | A | A | A | A | A | A | A | A | A | A |
| 0001 | A | A | A | A | A | A | A | A | A | A | A | A | A |
| 0010 | A | A | A | A | A | A | A | A | A | A | A | A | A |
| 0011 | D | A | A | A | A | A | A | A | A | A | A | A | A |
| 0100 | D | D | A | A | A | A | A | A | A | A | A | A | A |
| 0101 | D | D | D | A | A | A | A | A | A | A | A | A | A |
| 0110 | D | D | D | D | A | A | A | A | A | A | A | A | A |
| 0111 | D | D | D | D | D | A | A | A | A | A | A | A | A |
| 1000 | D | D | D | D | D | D | A | A | A | A | A | A | A |
| 1001 | D | D | D | D | D | D | D | A | A | A | A | A | A |
| 1010 | D | D | D | D | D | D | D | D | A | A | A | A | A |
| 1011 | D | D | D | D | D | D | D | D | D | A | A | A | A |
| 1100 | D | D | D | D | D | D | D | D | D | D | A | A | A |
| 1101 | D | D | D | D | D | D | D | D | D | D | D | A | A |
| 1110 | D | D | D | D | D | D | D | D | D | D | D | D | A |
| 1111 | D | D | D | D | D | D | D | D | D | D | D | D | D |

**VCGF1-VCGF0:** These are voltage configuration bits to select the reference voltage (Vref) for ADC module.

|  |  |  |
| --- | --- | --- |
| **Bit** | **= 1** | **= 0** |
| **VCGF0** | Vref+ (RA3) | Vcc |
| **VCGF1** | Vref- (RA2) | Vss (GND) |

3. **ADCON2 (A/D CONTROL REGISTER 2)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| ADFM | — | ACQT2 | ACQT1 | ACQT0 | ADCS2 | ADCS1 | ADCS0 |

**ADCS2:ADCS0:**These bits are used to select the clock option for ADC peripheral. The following table shows the bit configuration to select from different clock options:

|  |  |
| --- | --- |
| **ADCS2:ADCS0** | **Clock Option** |
| 000 | FRC (clock derived from A/D RC oscillator) |
| 001 | FOSC/64 |
| 010 | FOSC/16 |
| 011 | FOSC/4 |
| 100 | FRC (clock derived from A/D RC oscillator) |
| 101 | FOSC/32 |
| 110 | FOSC/8 |
| 111 | FOSC/2 |

**ACQT2:ACQT0:**  These bits are used to set the acquisition time of the ADC. The acquisition time is the time required to charge and discharge the holding capacitor of the ADC.

**ADFM:** This bit is used to select the format of digital output.

1 = Right justified (LSB to MSB)

0 = Left justified (MSB to LSB)

4. **ADRESL & ADRESH:**

Since the ADC module of PIC provides 10-bit digital output after A/D conversion, this output is stored in two 8-bit registers, namely, ADRESL & ADRESH. The lower byte is stored in ADRESL (A/D Result High register) while the higher byte is stored in ADRESH (A/D Result Low register).

**Working with ADC**

**Objective:** To select an analog channel of PIC18F4550’s in-built ADC and provide an analog input (0 to 5 volt) to it using a variable resistor or [preset](http://www.engineersgarage.com/electronic-components/presets-variable-resistors) (20 khttp://www.engineersgarage.com/sites/default/files/Ohm%20grey%203_0.jpg). (See circuit diagram) The main objective is to read the analog signal and display the corresponding digital value on [LCD](http://www.engineersgarage.com/electronic-components/16x2-lcd-module-datasheet). (Also see [interfacing LCD with PIC](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/interface-lcd-circuit))

**Programming Steps:**

1. Set number of analog inputs by setting PCFG3:PCFG0 (ADCON1).

2. Select the analog channel by setting CHS3:CHS0 bits (ADCON0).

3. Select the clock option, acquisition time output format by configuring the ADCON2 register.

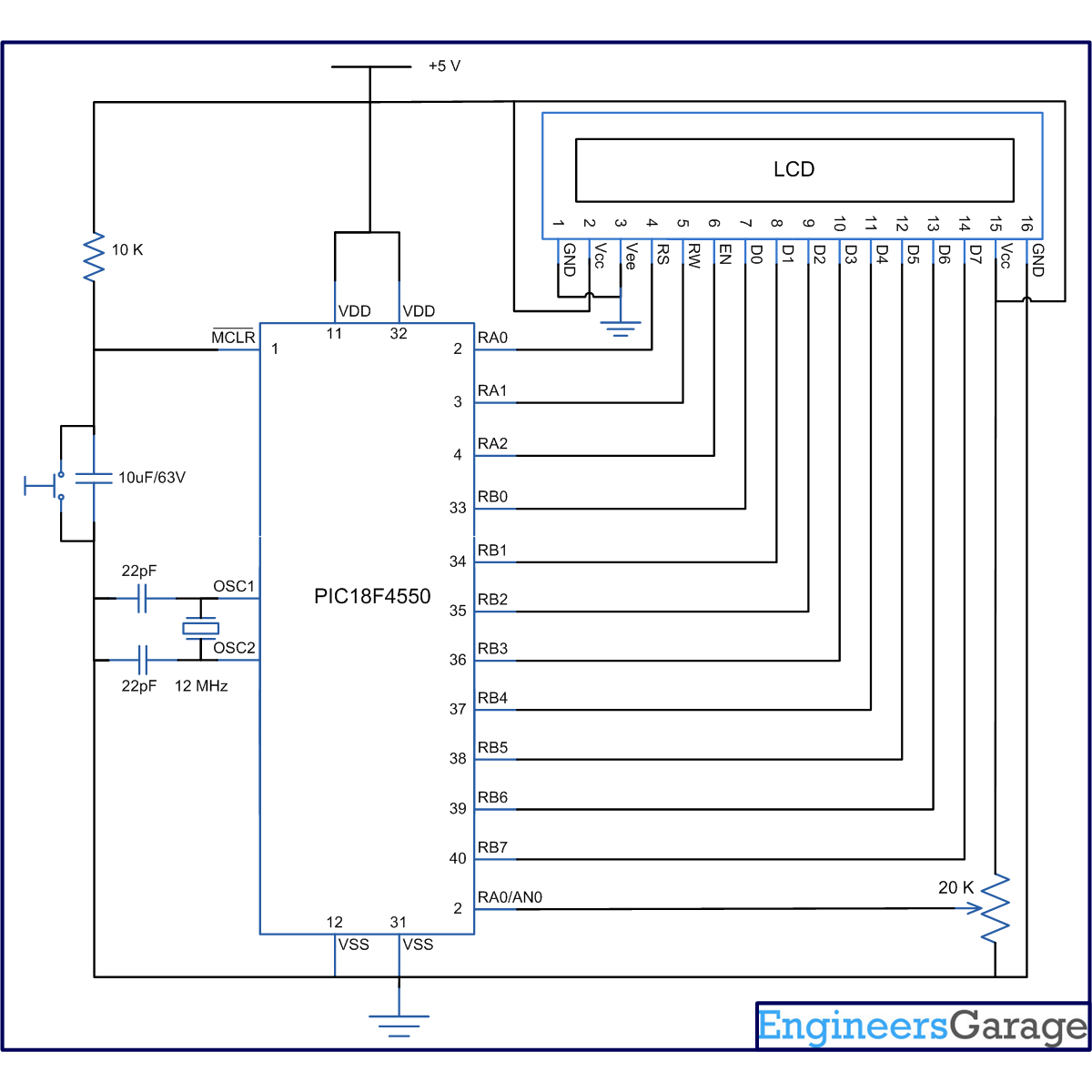
4. Enable the ADC by making ADON bit (ADCON0) high.

5. Start the conversion by setting GO/DONE bit (ADCON0).

6. Wait until GO/DONE bits becomes low. This indicates that the A/D conversion is over.

7. Store A/D conversion result from ADRESH:ADRESL into a variable.

8. Covert the resultant value to its corresponding ASCII value and display on LCD.



**// Program to depict working with inbuilt ADC of PIC18F4550 Microcontroller**  
// This code uses Channel0 (zero) of PIC's ADC Module  
  
// Configuration bits  
/\* \_CPUDIV\_OSC1\_PLL2\_1L,  // Divide clock by 2  
   \_FOSC\_HS\_1H,           // Select High Speed (HS) oscillator  
   \_WDT\_OFF\_2H,           // Watchdog Timer off  
   MCLRE\_ON\_3H            // Master Clear on  
\*/  
  
#define rs LATA.F0  
#define rw LATA.F1  
#define en LATA.F2  
#define lcdport LATB  
  
void lcd\_ini();  
void lcdcmd(unsigned char);  
void lcddata(unsigned char);  
void adc\_con(unsigned int);  
void adc\_init();  
  
unsigned char data[20]="ADC OUTPUT=";  
unsigned int digital\_out[10],avg\_output=0,temp;  
unsigned int i=0;  
  
  
void main()  
{  
    TRISA=0x01;        // Configure RA0 as input pin  
    LATA=0;  
    TRISB=0;        // Configure Port B as output port  
    LATB=0;  
    TRISD=0;  
    LATD=0;  
    lcd\_ini();        // LCD initialization  
    while(data[i]!='\0')  
    {  
        lcddata(data[i]);      // Call lcddata function to send character one by from 'data' array  
        i++;  
    }  
  
    adc\_init();        //ADC Initialization  
  
    while(1)  
    {  
        temp=0;  
        for(i=0;i<10;i++)  
        {  
            ADCON0|=(1<<GO);                              // Start A/D conversion  
            while(!(ADCON0 & (1<<GO)));                   // Wait until conversion gets over  
            digital\_out[i]=((ADRESL)|(ADRESH<<8));        // Store 10-bit output into a 16-bit variable  
            Delay\_ms(20);  
            temp=temp+digital\_out[i];  
        }  
        avg\_output=temp/10;                           // Take average of ten digital values for stablity  
        adc\_con(avg\_output);                          // Function to convert the decimal vaule to its corresponding ASCII  
  
    }  
}  
  
  
void adc\_init()  
{  
    ADCON1=0x0E;                            // Make RA0/AN0 pin as analog pin (Other pins remain to be digital I/O)  
    ADCON0=0x00;                            // Select Channel0 & ADC off  
    ADCON2=0x8A;                            // Left justified, 2TAD acquiciation time, Fosc/32 clock option  
    ADCON0.ADON=1;                          // Enable ADC  
}  
  
  
void lcd\_ini()  
{  
    lcdcmd(0x38);        // Configure the LCD in 8-bit mode, 2 line and 5x7 font  
    lcdcmd(0x0C);        // Display On and Cursor Off  
    lcdcmd(0x01);        // Clear display screen  
    lcdcmd(0x06);        // Increment cursor  
    lcdcmd(0x80);        // Set cursor position to 1st line, 1st column  
}  
  
  
void adc\_con(unsigned int adc\_out)  
{  
    unsigned int adc\_out1;  
    int i=0;  
    char position=0xC3;  
  
    for(i=0;i<=3;i++)  
    {  
        adc\_out1=adc\_out%10;                     // To exract the unit position digit  
        adc\_out=adc\_out/10;  
        lcdcmd(position);  
        lcddata(48+adc\_out1);                    // Convert into its corresponding ASCII  
        position--;  
  
    }  
}  
  
  
void lcdcmd(unsigned char cmdout)  
{  
    lcdport=cmdout;        //Send command to lcdport=PORTB  
    rs=0;                          
    rw=0;  
    en=1;  
    Delay\_ms(10);  
    en=0;  
}  
  
  
void lcddata(unsigned char dataout)  
{  
    lcdport=dataout;    //Send data to lcdport=PORTB  
    rs=1;  
    rw=0;  
    en=1;  
    Delay\_ms(10);  
    en=0;  
}