Using the ADC (Analog to Digital Converter) of PIC18F4520

Many electrical signals around us are Analog in nature. That means a quantity varies directly with some other quantity. The first quantity is mostly voltage while that second quantity can be anything like temperature, pressure, light, force or acceleration. For example in **LM35 temperature sensor** the output voltage varies according to the temperature, so if we could measure voltage, we can measure temperature.

But most of our computer (or Microcontrollers) are digital in nature. They can only differentiate between HIGH or LOW level on input pins. For example if input is more than 2.5v it will be read as 1 and if it is below 2.5 then it will be read as 0 (in case of 5v systems). So we cannot measure voltage directly from MCUs. To solve this problem most modern MCUs have an ADC unit. ADC stands for analog to digital converter. It will convert a voltage to a number so that it can be processed by a digital systems like MCU.

This enables us to easily interface all sort of analog devices with MCUs. Some really helpful example of analog devices are

1. Light Sensors.
2. Temperature Sensors.
3. Accelerometers.
4. Touch Screens.
5. Microphone for Audio Recording.

And possibly many more.

In this tutorials we will learn to use the internal ADC of PIC18 devices (Example is for PIC18F4520 which is a 40 PIN device).

Specifications of ADCs

Most important specification of ADCs is the resolution. This specifies how accurately the ADC measures the analog input signals. Common ADCs are 8 bit, 10 bit and 12 bit. For example if the reference voltage(explained latter) of ADC is 0 to 5v then a 8 bit ADC will break it in 256 divisions so it can measure it accurately up to 5/256 v= 19mV approx. While the 10 bit ADC will break the range in 5/1024 = 4.8mV approx. So you can see that the 8 bit ADC can’t tell the difference between 1mV and 18mV. The ADC in PIC18 are 10 bit.

Other specification include (but not limited to) the sampling rate, that means how fast the ADC can take readings. Microchip claims that pic18f4520's ADC can go as high as 100K samples per second.

ADC Terminology

Reference Voltage: The reference voltage specifies the minimum and maximum voltage range of analog input. In PIC 18 there are two reference voltage, one is the Vref- and one is Vref+. The Vref- specifies the minimum input voltage of analog input while the Vref+ specifies the maximum. For example if the input signal Vref- is applied to analog input channel then the result of conversion will be 0 and if voltage equal to Vref+ is applied to the input channel the result will be 1023 (max value for 10bit ADC).

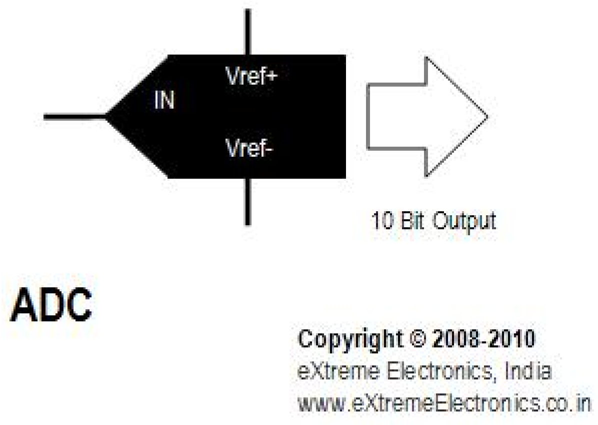


Fig.: ADC Reference Voltage.

The Vref+ and Vref- pins are available in PIN5 and PIN4 of the PIC18F4520 chip. So you can connect the reference voltage here. For a simple design the Vref- is GND and Vref+ is Vcc. As this is such a common configuration that the ADC can be set up to use these reference internally. Therefore you do not need to connect these on the external Vref pins, so you can use them for other purpose.

ADC Channels: The ADC module is connected to several channels via a multiplexer. The multiplexer can connect the input of the ADC to any of the available channels. This allows you to connect many analog signals to the MCU (say 3 temperature sensors). In PIC18F4520 there are 13 analog input channels, they are named AN0, AN1 etc. You can have a look at the pin configuation in the pic18f4520's datasheet to locate their position.

Acquisition Time: When an specific channel is selected the voltage from that input channel is stored in an internal holding capacitor. It takes some time for the capacitor to get fully charged and become equal to the applied voltage. This time is called acquisition time. The PIC18F4520's ADC provides a programmable acquisition time, so you can setup the acquisition time. Once acquisition time is over the input channel is disconnected from the source and the conversion begin. The acquisition times depends on several factor like the source impedance, Vdd of the system and temperature. You can refer to the page 227 and 228 in the datasheet for details on its calculation. A safe value is **2.45uS**, so acquisition time must be set to any value more than this.

ADC Clock: ADC Requires a clock source to do its conversion, this is called ADC Clock. The time period of the ADC Clock is called TAD. It is also the time required to generate 1 bit of conversion. The ADC requires 11 TADto do a 10 bit conversion. It can be derived from the CPU clock (called TOSC) by dividing it by a suitable division factor. There are Seven possible option.

- 2 x TOSC
- 4 x TOSC
- 8 x TOSC
- 16 x TOSC
- 32 x TOSC
- 64 x TOSC
- Internal RC

For Correct A/D Conversion, the A/D conversion clock (TAD) must be as short as possible but greater than the minimum TAD . See table 26-25 in PIC18F4520 datasheet (or table 28-29 in PIC18F4550/PIC18F2550 datasheet). It is 0.7uS for PIC18FXXXX device and 1.4uS for PIC18LFXXXX device.

We are running at 20MHz in our **PIC Development board** so we set prescaler of 32 TOSC.

```
Our FOSC = 20MHz
```

```
Therefore our Fosc = 1/20MHz
                  = 50nS


32 Tosc = 32 x 50 nS
        = 1600nS
        = 1.6uS


1.6 uS is more than the minimum requirement.
```

You can calculate the value for division factor using the above example in case you are using crystal of other frequency. Also now we have the T_{AD} we can calculate the division factor for acquisition time. Acquisition time can be specified in terms of T_{AD}. It can be set to one of the following values.

- 20 x T_{AD}
- 16 x T_{AD}
- 12 x T_{AD}
- 8 x T_{AD}
- 6 x T_{AD}
- 4 x T_{AD}
- 2 x T_{AD}
- 0 x T_{AD}

As we saw in above paragraph that the safe acquisition time is 2.45uS, so we select 2 x T_{AD} as acquisition time.

```
TACQ=2 x TAD
    =2 x 1.6uS (Replacing TAD= 1.6uS)
    =3.2uS
```

3.2uS is more than required 2.45uS so its ok.

Programming ADC in HI-TECH C for MPLAB

ADC is connect to the PIC CPU by 3 control register and 2 data register. The control registers are used to setup and give commands to the ADC. They also provides the status of ADC. The two data registers holds the 10 bit of converted data. Since each resister in PIC18 is of 8 bits therefore 2 registers are required to hold the 10bit data.

We will develop two functions to support ADC in our projects. One will help initialize the module and other will help us select any of the 13 channels and start the conversion. After the conversion is done it will return us the results.

I am not giving here the description of the control and data registers as they are very clearly explained in PIC18F4520's datasheet on page 223 to 225. I request you to download the datasheet and read the description so that you will have an Idea of what every bit in the registers do. As I told before, ADC is connected to the CPU via three control register and two data registers. The three control registers are :-

- **ADCON0** – Used to select analog input channel,start the conversion, check if the conversion is done and to switch on/off the module.(We use this in ADCRead() function.)
- **ADCON1** – Used to Select Voltage reference, and to configure ports as Analog of digital. (We leave these to defaults)
- **ADCON2** – Used to select ADC data format, Set acquisition time, ADC clock setup (We setup these in ADCInit() function)

First we configure the ADC to our needs in the **ADCInit()** function.

```
        //Function to Initialise the ADC Modulevoid ADCInit()
{
   //We use default value for +/- Vref


   //VCFG0=0,VCFG1=0

   //That means +Vref = Vdd (5v) and -Vref=GEN


   //Port Configuration

   //We also use default value here too

   //All ANx channels are Analog


   /*     ADCON2

      *ADC Result Right Justified.     *Acquisition Time = 2TAD     *Conversion Clock = 32 Tos
c   */


   ADCON2=0b10001010;
}
```

You can see that we only set up the ADCON2 register.We setup the ADC as follows

- ADC Result format as Right Justified(Explained latter).
- Acquisition time = 2T$_{AD}$(As Calculated Above)
- Conversion Clock as 32 T$_{OSC}$(As Calculated Above)

We also leave ADCON1 to defaults, which implies the following

- +VREF is 5v (Our Vcc)
- -VREF is GND
- All ANx channels are Analog. If you need some of them to do digital I/O then setup them accordingly.

Now we have our ADC Module setup, when ever you want to do the ADC Conversion in any channel, simply call ADCRead(). For example to do ADC Conversion on **channel 0** and store the result in variable **val** call the function in the following way.

```
val=ADCRead(0);
```

That’s it ! the analog value present on AN0 will be converted to a digital value and stored in variable **val**.

How the ADCRead() function works?

```
        //Function to Read given ADC channel (0-13)unsigned int ADCRead(unsigned char ch)
{
   if(ch>13) return 0;  //Invalid Channel

   ADCON0=0x00;

   ADCON0=(ch<<2);   //Select ADC Channel

   ADON=1;  //switch on the adc module

   GODONE=1;  //Start conversion

   while(GODONE); //wait for the conversion to finish

   ADON=0;  //switch off adc

   return ADRES;
}
```

The first line checks if the input channel provided by the user is valid or not. Then we select ADC channel. After that we switch on the module by setting **ADON** bit. Then conversion is started by setting the **GODONE**bit. As soon as the **GODONE** bit is set to 1 the module starts the conversion process. As long as the module is busy the **GODONE** bit is HIGH, and when the conversion is complete it is cleared by the module. So we wait in the while loop as long as GODONE is high. Remember that the while loop is empty (a semi colon just after it), so as long as **GODONE** is high the CPU will do nothing. As soon as **GODONE** is cleared the while loop breaks and we switch of the module by writing 0 to the **ADON** bit. Finally the result of conversion is returned, **ADRES** register holds the converted value.

Demo program to test PIC ADC Code

We will write a very simple program that will demonstrate ADC usage. The program will read ADC channel 0 (AN0 PIN) and display its value on **LCD Screen**. Before attempting the experiment please read the following tutorials. As you will need the **LCD** Support in addition to the ADC Interface code.

- LCD Interfacing with PIC MCU
- Making the LCD Expansion board

The program is intended to be compiled using the HI-TECH C for PIC18 using the MPLAB IDE. So if you are not familiar with the build steps please see the following tutorials.

- Downloading, Configuring and Using Basic Development Tools for Microchip PIC Series of MCUs.

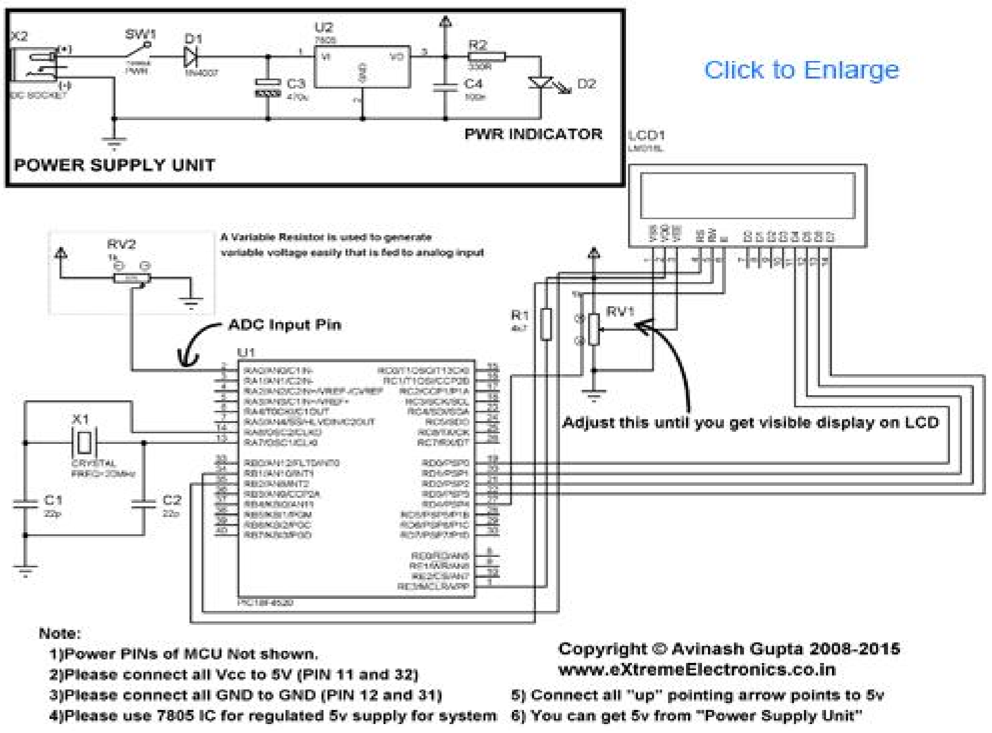
```
/*********************************************************************

ANALOG TO DIGITAL CONVERTOR INTERFACING TEST PROGRAM

---------------------------------------------------------Simple Program to connect with the
internal ADC of PIC MCUs.The program reads and display the analog input value at AN0.Requires
the PIC18 lcd library.

MCU: PIC18FXXXX Series from Microchip.Compiler: HI-TECH C Compiler for PIC18 MCUs (http://www.
htsoft.com/)

Copyrights 2008-2010 Avinash GuptaeXtreme Electronics, India

For More Info visithttp://www.eXtremeElectronics.co.in

Mail: me@avinashgupta.com

*********************************************************************/#include <htc.h>

#include "lcd.h"

//Chip Settings

__CONFIG(1,0x0200);

__CONFIG(2,0X1E1F);

__CONFIG(3,0X8100);

__CONFIG(4,0X00C1);

__CONFIG(5,0XC00F);


//Simple Delay Routinevoid Wait(unsigned int delay)

{

   for(;delay;delay--)

      __delay_us(100);

}

//Function to Initialise the ADC Modulevoid ADCInit()

{

   //We use default value for +/- Vref


   //VCFG0=0,VCFG1=0

   //That means +Vref = Vdd (5v) and -Vref=GEN


   //Port Configuration

   //We also use default value here too

   //All ANx channels are Analog


   /*     ADCON2

      *ADC Result Right Justified.     *Acquisition Time = 2TAD     *Conversion Clock = 32 Tos
c   */


   ADCON2=0b10001010;

}
```

```
//Function to Read given ADC channel (0-13)unsigned int ADCRead(unsigned char ch)
{
   if(ch>13) return 0;  //Invalid Channel

   ADCON0=0x00;

   ADCON0=(ch<<2);   //Select ADC Channel

   ADON=1;  //switch on the adc module

   GODONE=1;//Start conversion

   while(GODONE); //wait for the conversion to finish

   ADON=0;  //switch off adc

   return ADRES;
}void main()
{
   //Let the LCD Module start up
   Wait(100);

   //Initialize the LCD Module
   LCDInit(LS_BLINK);

   //Initialize the ADC Module

   ADCInit();

   //Clear the Module
   LCDClear();

   //Write a string at current cursor pos
   LCDWriteString("ADC Test");

   while(1)
   {
      unsigned int val; //ADC Value

      val=ADCRead(0);   //Read Channel 0
      LCDWriteIntXY(0,1,val,4);
```

```
    Wait(1000);
  }






}
```

Basic PIC Hardware to test the ADC Code.

The following image shows the basic hardware required to run the above code. Note that the crystal frequency is 20MHz, so please use this value only, if you want quick and error free operation. Or if you are experienced enough you can sort out errors if any.



Note:
1)Power PINs of MCU Not shown.
2)Please connect all Vcc to 5V (PIN 11 and 32)
3)Please connect all GND to GND (PIN 12 and 31)
4)Please use 7805 IC for regulated 5v supply for system
5) Connect all "up" pointing arrow points to 5v
6) You can get 5v from "Power Supply Unit"

Copyright © Avinash Gupta 2008-2015
www.eXtremeElectronics.co.in

Fig.: Schematic for PIC ADC Test. (Click To Enlarge/Print)