```
#include <16F877A.h>
#device *=16 ADC=10
#use delay(clock=20000000)
#use rs232(baud=9600,xmit=PIN_D0,stream=A0)
#use rs232(baud=9600,xmit=PIN_D1,stream=A1)
#use rs232(baud=9600,xmit=PIN_D2,stream=A2)
```

First, we have to include the library of the micro-controller we used, which is PIC16F877A. The next line is used to set the ADC resolution to 10 bits. We can set it to 16 bits as well but 10 bits is enough in this exercise. The clock frequency is set to 20 MHz and 3 serial communication ports (RS232) with a baud rate of 9600 is set up as well. Each port is assigned to a specific pin: D0, D1, and D2 respectively. The streams A0, A1, and A2 are associated with each port for communication.

```
int t=0;
float adc_value=0;    //10bit ADC result
char str_value[10];   //ADC result in string
```

The “t” variable is used to contain the value of the timer 0. The “adc_value” variable is used to hold the result of ADC read from AN0, AN1 and AN2 pins. The “str_value” array is just the text value of “adc_value” because in order to display the result to the virtual terminals, the numeric value must be turned to text.

```
#byte ADCON0=0x1F
#byte ADCON1=0x9F
#byte ADRESH=0x1E
#bit ADRESH0=ADRESH.0
#bit ADRESH1=ADRESH.1
#byte ADRESL=0x9E
#bit ADCON02=ADCON0.2   //finish conversion flag
```

```
#bit ADCON03=ADCON0.3
#bit ADCON04=ADCON0.4
#bit ADCON05=ADCON0.5
#byte TMRO=0x01
#byte OPREG=0x81
#byte INTCON=0x0b
```

The following lines declare the registers used as well as their location in the memory. They are: ADCON0, ADCON1, ADRESH, ADRESL, TMRO, OPREG, and INTCON.

The ADCON0 and ADCON1 are used to configure and control the ADC, the ADRESH and ADRESL registers are used to contain the ADC value after conversion is done. The TMRO, OPREG and INTCON are used to hold the initial value of timer 0, setup timer 0 and control the interrupt, especially for timer 0, respectively. Some bits that required changing are also declared for easier usage such as the GO/DONE bit, Analog channel select bits, etc.

```
#int_timer0
void ngat()
{
   t=t+1;
   TMRO=60;
}
```

This is the interrupt for timer 0, every time the timer 0 is overflow, the “t” variable increases by 1 unit and the initial value for timer 0 is also set to 60.

```
void main()
{
   OPREG=0b00000111;
   TMRO=60;
```

```
INTCON=0b11100000;
ADCON0=0b10000001;
ADCON1=0b10001001;
ADRESL=0;
ADRESH=0;
int counter=1,done_an1=0,done_an2=0,done_an3=0;
```

This is the setup step for the registers. For the OPREG register, the function of the bits can be seen below.

REGISTER 5-1: OPTION_REG REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$**

bit 6 **INTEDG**

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

In short, T0CS is set to internal instruction cycle clock, prescaler is assigned to timer 0 and is set to 1:256. The initial value of timer 0 is 60, which can be found through the following calculation: (255-60)*256*1/(F osc /4) = 0.01 (s). This value is used because the the “t” value acts as a timer, for every 100 overflows it increases by 1, which means it counts up every 0.1*100 = 1 second.

For the INTCON register, the function of the bits can be seen below. In short, global interrupt, peripheral interrupt, timer 0 interrupt are enabled while the rest are not used.

REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE:** Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **TMR0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE:** RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF:** RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF:** RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software).
0 = None of the RB7:RB4 pins have changed state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

For the ADCON0 and ADCON1, the function of each bit in the registers are listed below.

REGISTER 11-1: ADCON0 REGISTER (ADDRESS 1Fh)

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 PIC16F873A/876A devices only implement A/D channels 0 through 4; the unimplemented selections are reserved. Do not select any unimplemented channels with these devices.

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

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 11-2: ADCON1 REGISTER (ADDRESS 9Fh)

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 shaded area and 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

ADCON0=0b10000001 means that AD conversion clock is set to Fosc/32 (for 20MHz crystal), Analog select channel channel 0, Go/done=0 for A/D conversion not in progress and A/D on bit is set to power up the ADC. ADCON1=0b10001001 means that the result format is right justified, AD channel 0 to channel 5 is on and channel 6,7 is used for digital I/O purpose. The result registers ADRESL and ADRESH are initially set to 0, “counter” variable acts as 1 second timer and “done_anx” variables are used to ensure the conversion is done 1 time per second only.

```
while(true)
  {
     if(t==100)
     {
        counter+=1;
        t=0;
        done_an1=0;
        done_an2=0;
        done_an3=0;
        if(counter==7) counter=1;     //reset
     }
     if(done_an1==0)
     {
     ADCON03=0;
     ADCON04=0;
     ADCON05=0;
     delay_ms(1); //wait the required acquisition time
     ADCON02=1;
     while(ADCON02==1);
     adc_value=ADRESL+256*ADRESH0+512*ADRESH1;
     sprintf(str_value,"%f",adc_value);
     fputs(str_value,A0);
     done_an1=1;
     }
     if(counter%2==0)
     {
        if(done_an2==0)
```

```
        {
           ADCON03=1;
           ADCON04=0;
           ADCON05=0;
           delay_ms(1); //wait the required acquisition time
           ADCON02=1;
           while(ADCON02==1);
           adc_value=ADRESL+256*ADRESH0+512*ADRESH1;
           sprintf(str_value,"%f",adc_value);
           fputs(str_value,A1);
           done_an2=1;
        }
     }
     if(counter%3==0)
     {
        if(done_an3==0)
        {
           ADCON03=0;
           ADCON04=1;
           ADCON05=0;
           delay_ms(1); //wait the required acquisition time
           ADCON02=1;
           while(ADCON02==1);
           adc_value=ADRESL+256*ADRESH0+512*ADRESH1;
           sprintf(str_value,"%f",adc_value);
           fputs(str_value,A2);
           done_an3=1;
```

```
         }
      }
   }
}
```

t=100 means that 1 second has passed, so it will increase the counter variable.

```
if(t==100)
      {
         counter+=1;
         t=0;
         done_an1=0;
         done_an2=0;
         done_an3=0;
         if(counter==7) counter=1;     //reset
      }
```

Because the lowest common multiple of 1, 2 and 3 seconds is 6, so after every 6 seconds, the loop works the same as the beginning at 1 second. Therefore, in order to reduce the value of counter variable, the program just needs to work 6 times in the while(true) loop.

```
 if(done_an1==0)
      {
      ADCON03=0;
      ADCON04=0;
      ADCON05=0;
      delay_ms(1); //wait the required acquisition time
      ADCON02=1;
      while(ADCON02==1);
```

```
adc_value=ADRESL+256*ADRESH0+512*ADRESH1;
sprintf(str_value,"%f",adc_value);
fputs(str_value,A0);
done_an1=1;
}
```

The “done_anx” variables are used to check if the conversion is done, if it is done, it is set to 1 whereas if it isn’t, it is set to 0. ADCON03, ADCON04 and ADCON05 are used to select the ADC channel. Value 0,0,0 means channel 0; 1,0,0 means channel 1 and 0,1,0 means channel 2 respectively. Next, a 1 second delay is performed as in the PIC16F87XA datasheet, it is required to wait for acquisition time. This is compulsory.

The conversion is then started with the GO/DONE bit (ADCON0 bit 2) set as 1. When the conversion is done, the 10 bit result can be extracted with adc_value variable equals 8 bit in ADRESL register and 2 least significant bits in ADRESH as the result is right justified as we declared above. The result is then changed to text form in order to be put onto the virtual terminal with fputs() function. “Done_anx” is then set to 1 to indicate that the conversion is finished.

The same principle applies for the AN1 channel and AN2 channel. The difference is that there are other conditions that need to be met which are whenever the counter variable is divisible to 2, it will read the AN1 channel and whenever the counter variable is divisible to 3, it will read the AN2 channel to ensure AN1 channel is read every 2 seconds and AN2 channel is read every 3 seconds.