Se recomienda consultar el siguiente enlace y el datasheet del PIC18F2550.
http://picfernalia.blogspot.com.es/2012/07/conversor-adc.html

ADCON0:

- **ADON**: Habilita el módulo convertidor A/D.
- **GO/DONE**: Con un 1 se lanza la adquisición y conversión A/D. Se pone a 0 cuando está disponible el dato.
- **CHS3:CHS1**: Selecciona el canal que se va a leer.

REGISTER 21-1: ADCON0: A/D CONTROL REGISTER 0

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-2 **CHS3:CHS0:** Analog Channel Select bits

0000 = Channel 0 (AN0)
0001 = Channel 1 (AN1)
0010 = Channel 2 (AN2)
0011 = Channel 3 (AN3)
0100 = Channel 4 (AN4)
0101 = Channel 5 (AN5)(1,2)
0110 = Channel 6 (AN6)(1,2)
0111 = Channel 7 (AN7)(1,2)
1000 = Channel 8 (AN8)
1001 = Channel 9 (AN9)
1010 = Channel 10 (AN10)
1011 = Channel 11 (AN11)
1100 = Channel 12 (AN12)
1101 = Unimplemented(2)
1110 = Unimplemented(2)
1111 = Unimplemented(2)

bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit

When ADON = 1:
1 = A/D conversion in progress
0 = A/D Idle

bit 0 **ADON:** A/D On bit

1 = A/D converter module is enabled
0 = A/D converter module is disabled

Note 1: These channels are not implemented on 28-pin devices.
2: Performing a conversion on unimplemented channels will return a floating input measurement.

ADCON1:

- **PCFG3:PCFG0**: Configura los puertos que van a ser analógicos y digitales.
- **VCFG0**: Voltaje de referencia máximo (Vref+) en AN3. Si vale cero, se toma Vdd (Vcc).
- **VCFG1**: Voltaje de referencia mínimo (Vref-) en AN2. Si vale cero, se toma Vss (GND).

REGISTER 21-2: ADCON1: A/D CONTROL REGISTER 1

U-0	U-0	R/W-0	R/W-0	R/W-0(1)	R/W(1)	R/W(1)	R/W(1)
—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **VCFG1:** Voltage Reference Configuration bit (VREF- source)
1 = VREF- (AN2)
0 = VSS

bit 4 **VCFG0:** Voltage Reference Configuration bit (VREF+ source)
1 = VREF+ (AN3)
0 = VDD

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

PCFG3: PCFG0	AN12	AN11	AN10	AN9	AN8	AN7(2)	AN6(2)	AN5(2)	AN4	AN3	AN2	AN1	AN0
0000(1)	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(1)	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

A = Analog input D = Digital I/O

ADCON2:

- **ADCS2:ADCS0**: Configura la velocidad a la que se va a muestrear cada bit, múltiplo de la frecuencia del PIC. El inverso de este valor es TAD: tiempo de conversión por bit.
- **ACQT2:ACQT0**: Configura el tiempo de adquisición. Desde que se lanza GO hasta que empieza la conversión hay que dejar un tiempo que se cargue el condensador interno.
- **ADFM**: El resultado es un número de 10 bits que se almacena en dos registros de 8 bits ADRESH:ADRESL. Con ADFM configuramos si lo queremos justificado a la derecha o izquierda.

REGISTER 21-3: ADCON2: A/D CONTROL REGISTER 2

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7							bit 0

bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified
0 = Left justified

bit 6 **Unimplemented:** Read as '0'

bit 5-3 **ACQT2:ACQT0:** A/D Acquisition Time Select bits
111 = 20 TAD
110 = 16 TAD
101 = 12 TAD
100 = 8 TAD
011 = 6 TAD
010 = 4 TAD
001 = 2 TAD
000 = 0 TAD(1)

bit 2-0 **ADCS2:ADCS0:** A/D Conversion Clock Select bits
111 = FRC (clock derived from A/D RC oscillator)(1)
110 = FOSC/64
101 = FOSC/16
100 = FOSC/4
011 = FRC (clock derived from A/D RC oscillator)(1)
010 = FOSC/32
001 = FOSC/8
000 = FOSC/2

Note 1: If the A/D FRC clock source is selected, a delay of one TCY (instruction cycle) is added before the A/D clock starts. This allows the SLEEP instruction to be executed before starting a conversion.

PASOS A SEGUIR:

1. ***Configuramos los pines que serán analógicos y digitales, y las tensiones de referencia (ADCON1):***

Normalmente usaremos como tensiones de referencia Vdd (Vcc) y Vss (GND). Por lo tanto:

VCFG1:VCFG0 = 00.

Y si configurásemos AN0 como analógico y resto como digitales, según la tabla sería:

PCFG3:PCFG0 = 1110.

Por lo tanto la instrucción quedaría:

ADCON1 = 0b00001110;

NOTA: No olvidar que los puertos por defecto están configurados analógicos. Así que si todos van digitales, a la hora de configurar puertos (TRIS), emplear esta instrucción:

ADCON1 = 0x0F; // 0b00001111

2. ***Configuramos la frecuencia de muestreo por bit (tiempo de conversión) y el tiempo de adquisición (ADCON2).***

Tiempo de conversión:

La frecuencia de muestreo se intenta que sea lo más alta posible, múltiplo de la frecuencia del PIC, para que muestree lo más rápido posible. Pero el período de esa frecuencia tiene que superar un tiempo mínimo que marca el datasheet. Si no superase ese tiempo, el resultado de la conversión sería incorrecto. Ese tiempo viene marcado por la siguiente tabla, donde se asume que el tiempo mínimo de conversión analógico digital por bit es de 0.8us:

TABLE 21-1: TAD vs. DEVICE OPERATING FREQUENCIES

AD Clock Source (TAD)		Assumes TAD Min. = 0.8 µs
Operation	**ADCS2:ADCS0**	**Maximum Fosc**
2 Tosc	000	2.50 MHz
4 Tosc	100	5.00 MHz
8 Tosc	001	10.00 MHz
16 Tosc	101	20.00 MHz
32 Tosc	010	40.00 MHz
64 Tosc	110	48.00 MHz
RC(2)	x11	1.00 MHz(1)

Note 1: The RC source has a typical TAD time of 2.5 µs.

2: For device frequencies above 1 MHz, the device must be in Sleep for the entire conversion or a Fosc divider should be used instead. Otherwise, the A/D accuracy may be out of specification.

Como estamos usando 48MHz, la única opción para superar los 0.8us es escoger 64 TOSC:

TAD = 64/48MHz = 1.33us > 0.8us.

ADCS2:ADCS0 = 110.

Tiempo de adquisición:

A la entrada del conversor hay un condensador que se tiene que cargar. Una vez se carga, se desconecta con el exterior, y esa tensión a la que se carga es la que se usa para la conversión. Es por lo que se recomienda que no haya una resistencia de carga exterior superior a 2.5kohm. Los cálculos que realiza el PIC que necesita para esa adquisición se detallan a continuación. Pero lo que realmente nos interesa saber es que hay que superar los 2.45us. En nuestro caso:

4TAD = 5.32us > 2.45us.

Pero vamos a coger 6TAD, por si acaso la resistencia es algo mayor que 2.5kohm

6TAD:

ACQT2:ACQT0 = 011.

Y si justificamos a la derecha, máxima resolución (1023), nuestra instrucción quedaría:

ADCON2 = 0b10011101;

EQUATION 21-1: ACQUISITION TIME

TACQ	=	Amplifier Settling Time + Holding Capacitor Charging Time + Temperature Coefficient
	=	TAMP + TC + TCOFF

EQUATION 21-2: A/D MINIMUM CHARGING TIME

VHOLD	=	$(V_{REF} - (V_{REF}/2048)) \bullet (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))})$
or		
TC	=	$-(C_{HOLD})(R_{IC} + R_{SS} + R_S) \ln(1/2048)$

EQUATION 21-3: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME

TACQ	=	TAMP + TC + TCOFF
TAMP	=	0.2 µs
TCOFF	=	(Temp – 25°C)(0.02 µs/°C) (85°C – 25°C)(0.02 µs/°C) 1.2 µs
Temperature coefficient is only required for temperatures > 25°C. Below 25°C, TCOFF = 0 µs.		
TC	=	-(CHOLD)(RIC + RSS + RS) ln(1/2048) µs -(25 pF) (1 kΩ + 2 kΩ + 2.5 kΩ) ln(0.0004883) µs 1.05 µs
TACQ	=	0.2 µs + 1.05 µs + 1.2 µs 2.45 µs

3. ***Habilitamos el módulo convertidor (ADCON0).***
   ADCON0bits.ADON = 1;

4. ***Seleccionamos el canal de lectura (ADCON0).***

En nuestro caso como tenemos sólo un pin configurado como analógico, siempre leeremos AN0:

CHS3:CHS0 = 0000.

El paso 3 se podría haber incluido en este:

ADCON0 = 0b00000001;

5. ***Lanzamos la adquisición y conversión (ADCON0).***
   ADCON0bits.GO_DONE = 1;

6. ***Esperamos a que finalice el proceso (ADCON0).***

Se queda el programa enganchado en un bucle hasta que finalice la conversión, es decir, cuando GO_DONE valga 0.

while (ADCON0bits.GO_DONE == 1);

7. ***Leemos los registros donde se almacena el resultado.***

Aquí tenemos dos opciones:

- *Mínima resolución*, es decir, me quedo con los 8 bits más significativos y sólo leo el registro ADRESH, porque anteriormente los he justificado a la izquierda. En este caso el valor máximo es 255, que equivaldría a 5V aproximadamente.

```
int adc = 0;
adc =  ADRESH;
```

- *Máxima resolución*, es decir, me quedo con los 10 bits, por tanto, tengo que leer los dos registros. En este caso 1023 equivaldría con los 5V.

```
adc = ADRESH;
adc = adc << 8;
adc = adc + ADRESL;
```

A continuación, si queremos pasar este resultado entero al valor real de la tensión en decimal:
- *Mínima resolución*:
  value = (float) (5.0*adc) / 255;
- *Máxima resolución*:
  value = (float) (5.0*adc) / 1023;

8. ***Deshabilitamos el módulo de conversión para reducir consumo (ADCON0).***
   ADCON0bits.ADON = 1;

```
void main() {
  int adc = 0;

  TRISB = 0b10000000;
  TRISAbits.TRISA0 = 1;      // Potenciómetro está en PIN_A0

  ADCON1bits.PCFG = 0b1110;  // Todos los puertos digitales menos el PIN_A0 analógico
  ADCON1bits.VCFG = 0b00;    // Tensiones de referencia 0 y 5V

  ADCON2bits.ADFM = 0;       // Justificación a la izquierda, resolución de 0 a 255
  ADCON2bits.ADCS = 0b110;   // Tiempo de conversión, TAD = 64/48MHz = 1.33us > 0.8us(mínimo necesario)
  ADCON2bits.ACQT = 0b011;   // Tiempo de adquisición, 6TAD = 7.98us > 2.45us(mínimo para una carga no superior a 2.5Kohm)

  while(1){
    ADCON0bits.ADON = 1;             // Habilita el conversor analógico digital
    ADCON0bits.CHS = 0b0000;         // Selecciona el AN0 (PIN_A0)
    ADCON0bits.GO_DONE = 1;          // Comienza la conversión

    while (ADCON0bits.GO_DONE == 1);   // Espera a que GO_DONE se ponga a 0
    adc =  ADRESH;                   // Lee los 8 bits más significativos
    vumetro(adc);
  }
}
```