

Microcontrolador PIC 16f877a

Uso del convertidor ADC para implementar el proyecto de la Unidad de
Aprendizaje de Instrumentación

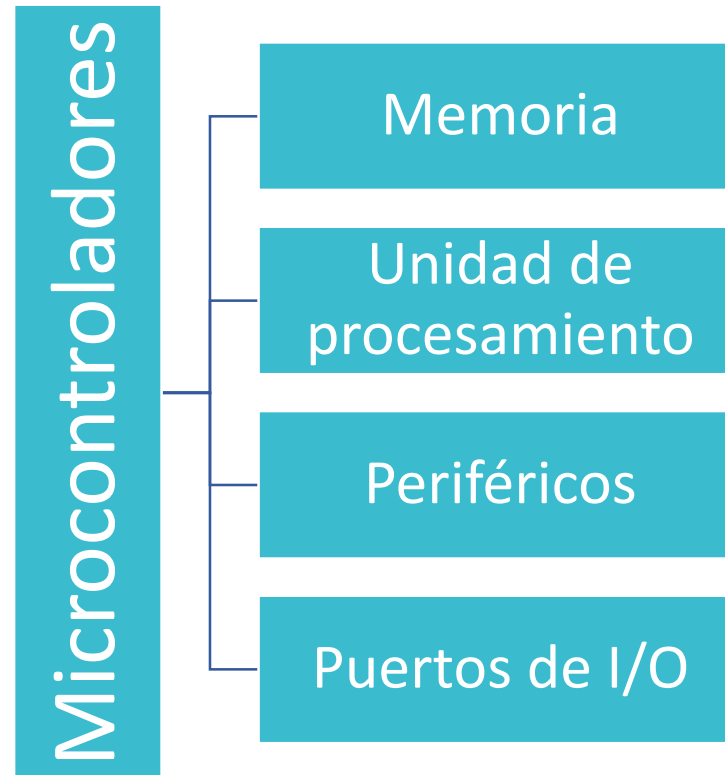
Introducción

Un microcontrolador es un dispositivo programable capaz de ejecutar diferentes instrucciones de forma secuencial con el fin de controlar o automatizar algún proceso.

Los microcontroladores se usan ampliamente como “computadoras” que se encuentran dentro de maquinaria, instrumentación, electrodomésticos, vehículos, juguetes y muchos otros aparatos. Debido a la naturaleza de las tareas para las que son diseñados, es común que los microcontroladores no posean una gran capacidad de cálculo, y en su lugar, incorporen facilidades para las comunicaciones, la conversión analógica-digital, la medición de tiempo, la cuenta de eventos y el manejo de “actuadores”, entre otras. También debido a la naturaleza de las aplicaciones, las personas que se dedican al desarrollo de proyectos con microcontroladores deben contar con conocimientos y habilidades en por lo menos dos disciplinas: electrónica y computación, situación que se ve reflejada para el desarrollo del proyecto final de la asignatura de Instrumentación.

Funcionalidades

- Está compuesto por cuatro unidades principales



Se encuentran las funciones que puede ejecutar, el código con las sentencias que ejecutará y todos los datos volátiles que ayudan en la ejecución de los cálculos


Se encarga de ejecutar las instrucciones programadas en la memoria

Que auxilian en la ejecución de las instrucciones como: temporizadores, convertidores ADC, puertos de comunicación y contadores.

Con éstos se leen señales desde el exterior, se programan señales para controlar elementos externos o bien permiten la comunicación con otros microcontroladores.

Procedimiento para grabar un microcontrolador:


Desarrollo.- De un código con las acciones que llevará acabo un microcontrolador por medio de una interfaz de desarrollo.



Decodificación.- Se pasa a un lenguaje máquina



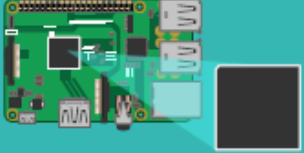


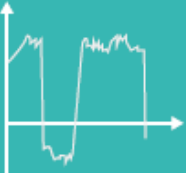
Almacena.- Una vez decodificado se almacena en el microcontrolador, con la ayuda de un grabador.



Ejecución.- Una vez que el uM cuenta con la lógica programable se ejecuta tantas veces sea necesario

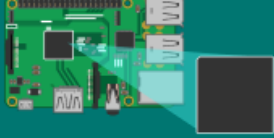
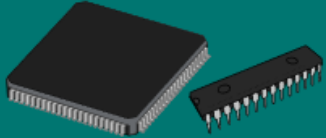
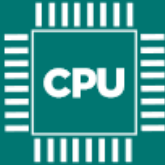

Diferencias entre microprocesadores y microcontroladores

Los microprocesadores y los microcontroladores utilizan los mismos componentes pero con distintas características, las cuales es necesario conocer para poderlos distinguir entre si.

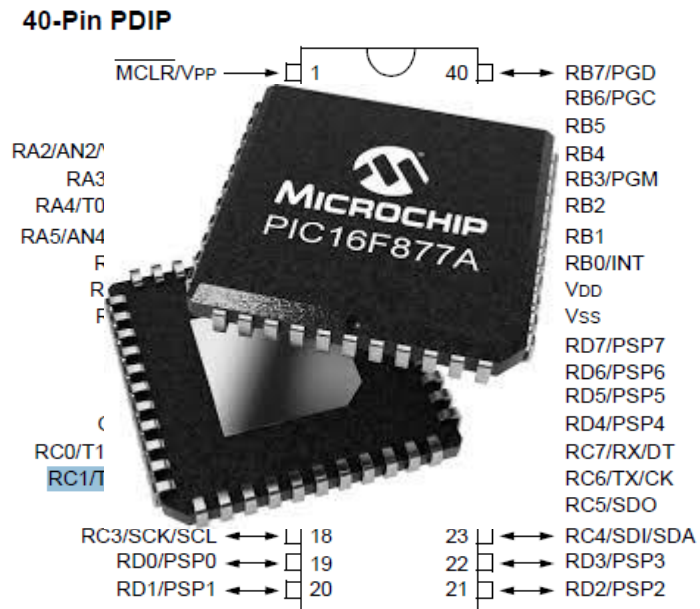
CARACTERÍSTICAS	MICROPROCESADORES	MICROCONTROLADORES
 VELOCIDAD DE OPERACIÓN	Su frecuencia de procesamiento es del orden de GHz.	Su frecuencia de procesamiento es del orden de MHz.
 ELEMENTOS NECESARIOS PARA SU FUNCIONAMIENTO	Normalmente el microprocesador opera en conjunto con una memoria RAM, una memoria ROM y un decodificador de direcciones dentro de lo que se conoce como un microordenador.	El microcontrolador incluye prácticamente todos los elementos necesarios para operar en un sólo circuito integrado por lo que en muchos casos únicamente es necesario energizarlos para funcionar.
 COSTO	Debido a que su arquitectura es mucho más compleja su costo es elevado y al requerir elementos adicionales para funcionar aumenta aún más.	Debido a que su arquitectura es de un bajo set de instrucciones y requiere de pocos elementos adicionales para operar, su costo es bajo.
 SUCEPTIBILIDAD AL RUIDO ELECTROMAGNÉTICO	Mayor susceptibilidad debido a su tamaño y al cableado externo necesario para interconectar los componentes.	Presenta menor susceptibilidad gracias a que todos los elementos se encuentran dentro del mismo circuito integrado.

Diferencias

Es importante analizar los requerimientos de la aplicación, para decidir la mejor opción .

COMPONENTES	 MICROPROCESADORES	 MICROCONTROLADORES
 CPU	El CPU tiene mayor potencia de cálculo por lo que es ideal para procesamiento de datos.	El CPU es una de sus partes principales, pero su capacidad de procesamiento es menor.
 MEMORIAS RAM Y ROM	A pesar de que requiere de estas memorias para operar, deben añadirse por separado con ayuda de una tarjeta madre.	En este caso las memorias son mucho más simples por lo que se fabrican dentro del mismo circuito integrado y ya no es necesario agregarlas de forma externa.
<p>Por lo anterior, para una aplicación en la que sea necesario monitorear el exterior por medio de sensores o controlar algún proceso por medio de motores, lo más conveniente es usar un microcontrolador; pero si lo que se busca es procesar un conjunto de datos, implementar una interfaz gráfica de usuario o transferir información por medio de internet, un microprocesador sería lo más conveniente, aunque también es posible integrar ambos de ser necesario.</p>		

Familia de los PIC



Familias de microcontroladores PIC Microchip produce diferentes dispositivos que según el número de terminales lo clasifica en:

Gama baja

- Microcontroladores de 8 bits, de bajo costo, de 6 pines y bajas prestaciones.

Algunos PIC's de las serie PIC10 se conocen como gama enana.

- PIC12: microcontroladores de 8 bits, de bajo costo, de 8 pines y bajas prestaciones.

Gama media

- PIC16: microcontroladores de 8 bits, con gran variedad de número de pines y prestaciones medias.

Gama alta

- PIC17 y PIC18: microcontroladores de 8 bits, con gran variedad de número de pines y prestaciones medias/altas.
- PIC24: microcontroladores de 16 bits.
- PIC32: microcontroladores de 32 bits.
- dsPIC's. Microcontroladores especializados para el procesamiento de señales digitales.

Las principales características de los microcontroladores de gama alta PIC18, se puede resumir en los siguientes:

- Arquitectura Harvard, RISC avanzada de 16 bits (instrucciones) con 8 bit de datos.
- 77 instrucciones a nivel de assembler.
- Desde 18 a 80 pines.
- Tecnología CMOS.
- Puertos de I/O con múltiples funciones.
- Módulos CCP, ECCP, PWM y comparadores incorporados.
- Conversores con múltiples canales multiplexados.
- Hasta 64K bytes de memoria de programa (hasta 2 Mbytes en ROMless).
- Multiplicador Hardware 8x8.
- Hasta 3968 bytes de RAM y 1KBytes de EEPROM.
- Frecuencia máxima de reloj 40Mhz. Hasta 10 MIPS.
- Pila de 32 niveles.
- Múltiples fuentes de interrupción.
- Resistencias pull-up programables en el puerto B.
- Temporizadores 8 y 16 bits.
- Comunicación serial síncrona y asíncrona.
- Periféricos de comunicación avanzados (CAN y USB).

Características del PIC16f877a



Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F876A
- PIC16F874A
- PIC16F877A

High-Performance RISC CPU:

- Only 35 single-word instructions to learn
- All single-cycle instructions except for program branches, which are two-cycle
- Operating speed: DC – 20 MHz clock input
DC – 200 ns instruction cycle
- Up to 8K x 14 words of Flash Program Memory,
Up to 368 x 8 bytes of Data Memory (RAM),
Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to other 28-pin or 40/44-pin
PIC16CXXX and PIC16FXXX microcontrollers

Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler,
can be incremented during Sleep via external
crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period
register, prescaler and postscaler
- Two Capture, Compare, PWM modules
 - Capture is 16-bit, max. resolution is 12.5 ns
 - Compare is 16-bit, max. resolution is 200 ns
 - PWM max. resolution is 10-bit
- Synchronous Serial Port (SSP) with SPI™
(Master mode) and I²C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver
Transmitter (USART/SCI) with 9-bit address
detection
- Parallel Slave Port (PSP) – 8 bits wide with
external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for
Brown-out Reset (BOR)

Analog Features:

- 10-bit, up to 8-channel Analog-to-Digital
Converter (A/D)
- Brown-out Reset (BOR)
- Analog Comparator module with:
 - Two analog comparators
 - Programmable on-chip voltage reference
(VREF) module
 - Programmable input multiplexing from device
inputs and internal voltage reference
 - Comparator outputs are externally accessible

Special Microcontroller Features:

- 100,000 erase/write cycle Enhanced Flash
program memory typical
- 1,000,000 erase/write cycle Data EEPROM
memory typical
- Data EEPROM Retention > 40 years
- Self-reprogrammable under software control
- In-Circuit Serial Programming™ (ICSP™)
via two pins
- Single-supply 5V In-Circuit Serial Programming
- Watchdog Timer (WDT) with its own on-chip RC
oscillator for reliable operation
- Programmable code protection
- Power saving Sleep mode
- Selectable oscillator options
- In-Circuit Debug (ICD) via two pins

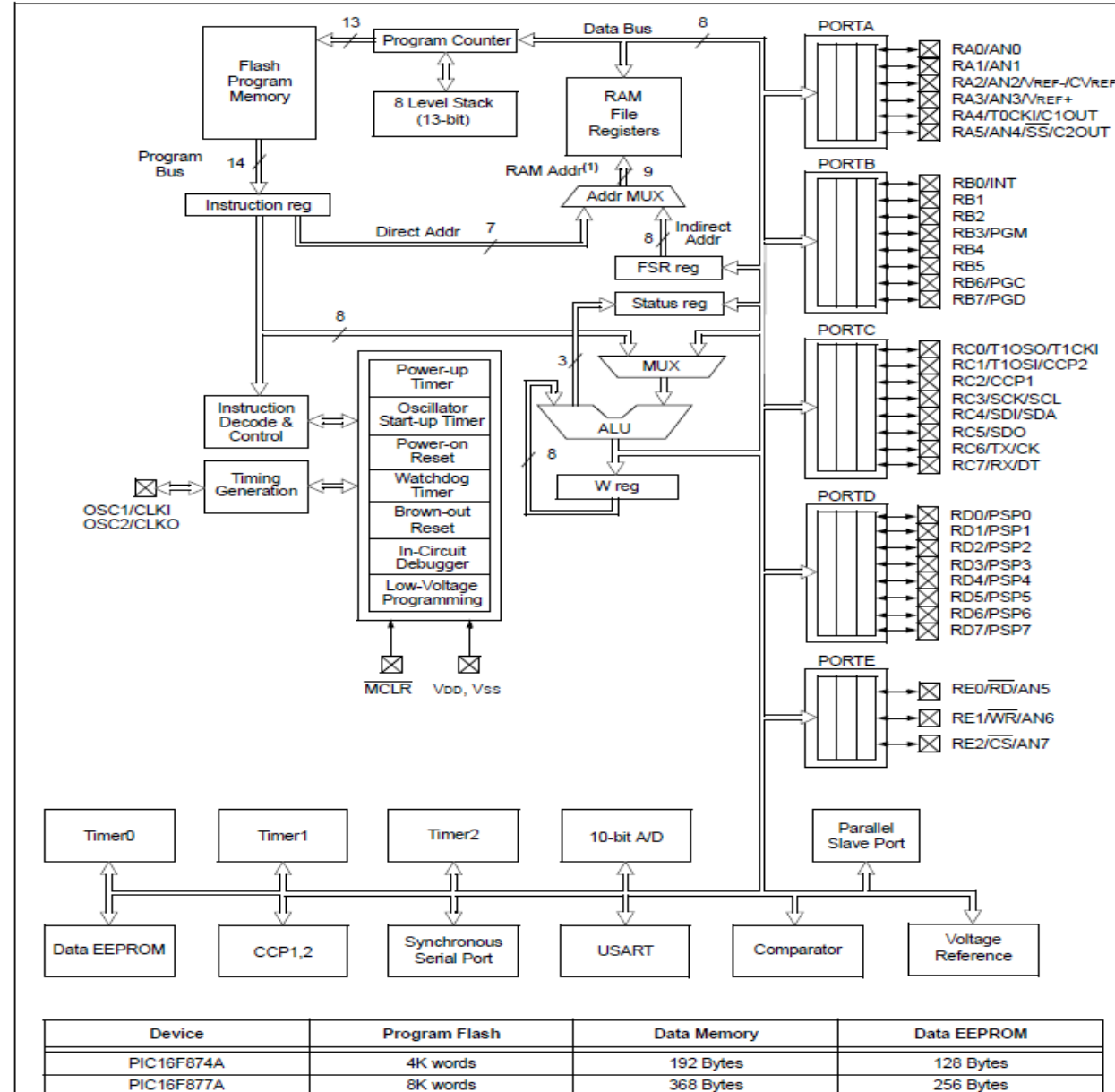
CMOS Technology:

- Low-power, high-speed Flash/EEPROM
technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Commercial and Industrial temperature ranges
- Low-power consumption

Device	Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
	Bytes	# Single Word Instructions						SPI	Master I²C			
PIC16F873A	7.2K	4096	192	128	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F874A	7.2K	4096	192	128	33	8	2	Yes	Yes	Yes	2/1	2
PIC16F876A	14.3K	8192	368	256	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F877A	14.3K	8192	368	256	33	8	2	Yes	Yes	Yes	2/1	2

PIC16F87XA

FIGURE 1-2: PIC16F874A/877A BLOCK DIAGRAM



Puertos del PIC16f877a

Cada cuadro de color representa los puertos que tiene el PIC16f877a. Es decir este PIC tiene 5 puertos (A,B,C,D,E)

PIC16F87XA

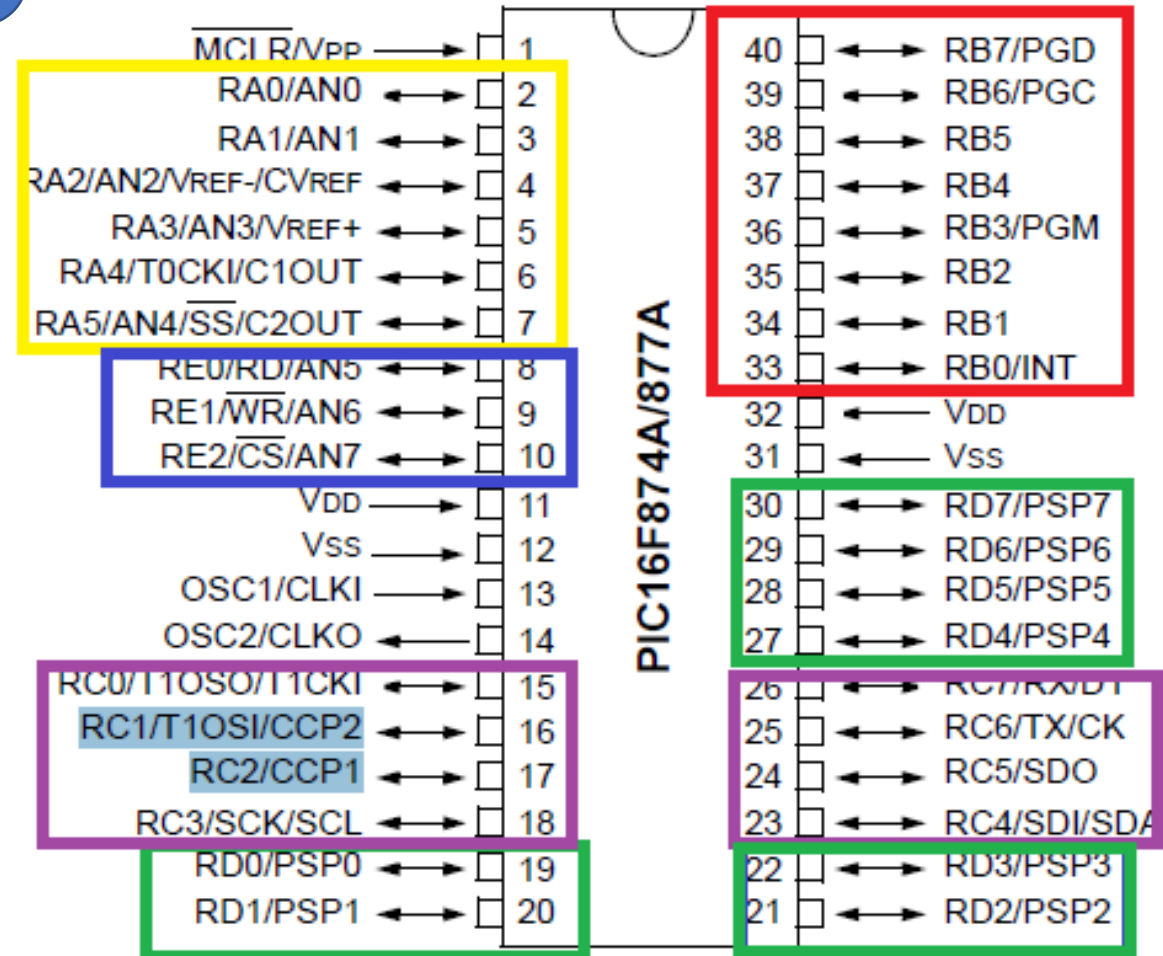
TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION

Pin Name	PDIP Pin#	PLCC Pin#	TQFP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKI OSC1	13	14	30	32	I	ST/CMOS(4)	Oscillator crystal or external clock input. Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; otherwise CMOS. External clock source input. Always associated with pin function OSC1 (see OSC1/CLKI, OSC2/CLKO pins).
OSC2/CLKO OSC2	14	15	31	33	O	—	Oscillator crystal or clock output. Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
MCLR/VPP MCLR VPP	1	2	18	18	I P	ST	Master Clear (input) or programming voltage (output). Master Clear (Reset) input. This pin is an active low Reset to the device. Programming voltage input.
RA0/AN0 RA0 AN0	2	3	19	19	I/O I	TTL	PORTA is a bidirectional I/O port. Digital I/O. Analog input 0.
RA1/AN1 RA1 AN1	3	4	20	20	I/O I	TTL	Digital I/O. Analog input 1.
RA2/AN2/VREF-/CVREF RA2 AN2 VREF- CVREF	4	5	21	21	I/O I I O	TTL	Digital I/O. Analog input 2. A/D reference voltage (Low) input. Comparator VREF output.
RA3/AN3/VREF+ RA3 AN3 VREF+	5	6	22	22	I/O I I	TTL	Digital I/O. Analog input 3. A/D reference voltage (High) input.
RA4/T0CKI/C1OUT RA4 T0CKI C1OUT	6	7	23	23	I/O I O	ST	Digital I/O – Open-drain when configured as output. Timer0 external clock input. Comparator 1 output.
RA5/AN4/SS/C2OUT RA5 AN4 SS C2OUT	7	8	24	24	I/O I I O	TTL	Digital I/O. Analog input 4. SPI slave select input. Comparator 2 output.

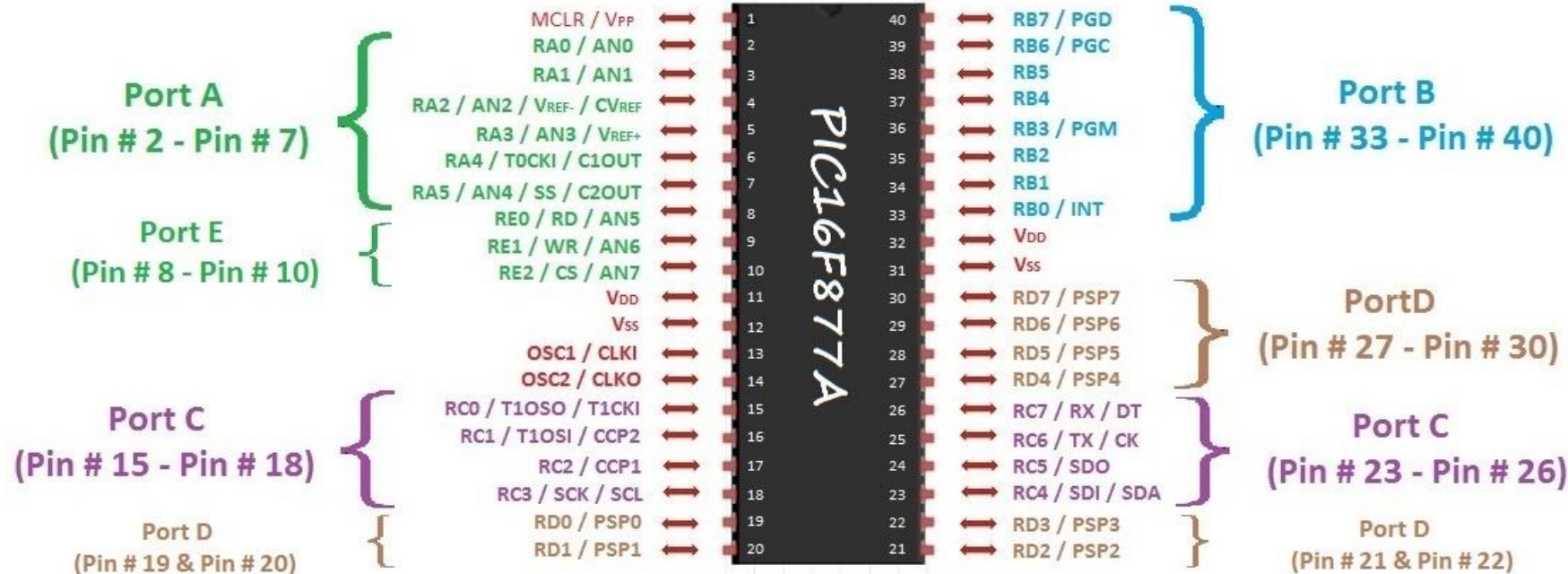
Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC Oscillator mode and a CMOS input otherwise.

40-Pin PDIP



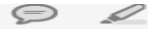
Distribución de los pines en los puertos del PIC16f877a



PIC16F877A Pin Diagram

Mapa de los registros del PIC16f877a

234



URE 2-3: PIC16F876A/877A REGISTER FILE MAP

File Address	File Address	File Address	File Address
Indirect addr. ^(*)	Indirect addr. ^(*)	Indirect addr. ^(*)	Indirect addr. ^(*)
TMR0 00h	OPTION_REG 80h	TMR0 100h	OPTION_REG 180h
PCL 01h	PCL 81h	PCL 101h	PCL 181h
STATUS 02h	STATUS 82h	STATUS 102h	STATUS 182h
FSR 03h	FSR 83h	FSR 103h	FSR 183h
PORTA 04h	TRISA 84h	PORTB 104h	TRISB 184h
PORTB 05h	TRISB 85h	PORTC 105h	TRISC 185h
PORTC 06h	TRISC 86h	PORTD 106h	TRISD 186h
PORTD ⁽¹⁾ 07h	TRISD ⁽¹⁾ 87h	PORTF 107h	TRISF 187h
PORTE ⁽¹⁾ 08h	TRISE ⁽¹⁾ 88h	PORTG 108h	TRISG 188h
PCLATH 09h	PCLATH 89h	PORTH 109h	TRISH 189h
INTCON 0Ah	INTCON 8Ah	PCLATH 10Ah	PCLATH 18Ah
PIR1 0Bh	PIE1 8Bh	INTCON 10Bh	INTCON 18Bh
PIR2 0Ch	PIE2 8Ch	EEDATA 10Ch	EECON1 18Ch
TMR1L 0Dh	PCON 8Dh	EEADR 10Dh	EECON2 18Dh
TMR1H 0Eh	SSPCON2 8Eh	EEDATH 10Eh	Reserved ⁽²⁾ 18Eh
T1CON 0Fh	PR2 8Fh	EEDATH 10Fh	Reserved ⁽²⁾ 18Fh
T2CON 10h	SSPSTAT 90h	General Purpose Register 16 Bytes 110h	General Purpose Register 16 Bytes 190h
SSPBUF 11h	CCPR1L 91h	General Purpose Register 16 Bytes 111h	General Purpose Register 16 Bytes 191h
SSPCON 12h	CCPR1H 92h	General Purpose Register 16 Bytes 112h	General Purpose Register 16 Bytes 192h
CCPR1L 13h	CCP1CON 93h	General Purpose Register 16 Bytes 113h	General Purpose Register 16 Bytes 193h
CCPR1H 14h	RCSTA 94h	General Purpose Register 16 Bytes 114h	General Purpose Register 16 Bytes 194h
CCP1CON 15h	TXREG 95h	General Purpose Register 16 Bytes 115h	General Purpose Register 16 Bytes 195h
RCSTA 16h	RCREG 96h	General Purpose Register 16 Bytes 116h	General Purpose Register 16 Bytes 196h
TXREG 17h	CCPR2L 97h	General Purpose Register 16 Bytes 117h	General Purpose Register 16 Bytes 197h
RCREG 18h	CCPR2H 98h	General Purpose Register 16 Bytes 118h	General Purpose Register 16 Bytes 198h
CCPR2L 19h	CCP2CON 99h	General Purpose Register 16 Bytes 119h	General Purpose Register 16 Bytes 199h
CCPR2H 1Ah	ADRESH 9Ah	General Purpose Register 16 Bytes 11Ah	General Purpose Register 16 Bytes 19Ah
CCP2CON 1Bh	ADCON0 9Bh	General Purpose Register 16 Bytes 11Bh	General Purpose Register 16 Bytes 19Bh
ADRESH 1Ch	ADCON1 9Ch	General Purpose Register 16 Bytes 11Ch	General Purpose Register 16 Bytes 19Ch
ADCON0 1Dh	General 9Dh	General Purpose Register 16 Bytes 11Dh	General Purpose Register 16 Bytes 19Dh
20h	General 9Eh	General Purpose Register 16 Bytes 11Eh	General Purpose Register 16 Bytes 19Eh
	General 9Fh	General Purpose Register 16 Bytes 11Fh	General Purpose Register 16 Bytes 19Fh
	General A0h	General 120h	General 1A0h

Programación de Microcontroladores PIC con lenguaje C

Tradicionalmente muchos programadores de microcontroladores PIC utilizan el lenguaje ensamblador para realizar sus proyectos, pero en la actualidad existen compiladores de lenguajes de alto nivel que permiten realizar las mismas tareas en un menor tiempo de desarrollo y con mucha mayor facilidad en la programación.

El compilador “**PIC C**” es una herramienta útil para programar microcontroladores PIC, en la cual están incluidas las librerías para manejar una pantalla LCD, el protocolo de comunicación serial, manejo de puertos, etc.

- Traduce el código C del archivo fuente (.c) a lenguaje de máquina para programar microcontroladores PIC (.HEX).
- Se incluye Drivers o librerías de código fuente para manejo de pantallas LCD, teclados, sensores, protocolos de comunicación, memorias, conversión analógico a digital, etc.
- Funciones para el manejo de interrupciones.

Programación de Microcontroladores PIC con lenguaje C

DIRECTIVAS EN EL CCS

#INCLUDE <NOMBRE_DEL_FICHERO>

Esta directiva hace que el compilador incluya en el fichero fuente el texto que contiene el archivo indicado.

Ejemplo: #include <16F877.H>

#FUSE

Esta directiva define qué fusibles deben activarse en el dispositivo cuando se programe. Esta directiva no afecta a la compilación; sin embargo, esta información se pone en el archivo de salida. Algunas de las opciones más usadas son:

- LP, XT, HS, RC (Tipo de oscilador)
- WDT, NOWDT (Activación del Watch Dog Timer)
- PROTECT, NOPROTECT (Protección del código)
- PUT, NOPUT (Temporizador de arranque)
- BROWNOUT, NOBROWNOUT (Detección de caídas de tensión de la fuente de alimentación)

Ejemplo #fuse HS, WDT.

#INT_XX

Estas directivas especifican que la función que le sigue es una función de interrupción. Las funciones de interrupción no pueden tener ningún parámetro. Como es natural, no todas las directivas pueden usarse con todos los dispositivos. Las directivas más comunes son las siguientes:

#INT_EXT: Interrupción externa

#INT_TRCC: Desbordamiento del TIMER0 (RTCC)

#INT_RB: Cambio en los pines B4, B5, B6, B7

#INT_AD: Conversor A/D

#INT_TIMER1: Desbordamiento del TIMER1.

#INT_TIMER2: Desbordamiento del TIMER2

#INT_CP1: Modo captura de datos por CCP1

#INT_CCP2: Modo captura por CCP2

Programación de Microcontroladores PIC con lenguaje C

#USE DELAY (Clock = Frecuencia):

Esta directiva indica al compilador la frecuencia del procesador, en ciclos por segundo, a la vez que habilita el uso de las funciones DELAY_MS() y DELAY_US().

Ejemplo: #USE DELAY (CLOCK = 4000000)

#USE STANDARD_io (Puerto)

Esta directiva afecta al código que el compilador generará para las instrucciones de entrada y salida. Este método rápido de hacer I/O ocasiona que el compilador realice I/O sin programar el registro de dirección. El puerto puede ser A-G.

Ejemplo: #USE STANDARD_io(B)

#USE RS232 (BAUD = baudios, XMIT = pin, RCV=pin)

Esta directiva le dice al compilador la velocidad en bits por segundo y los pines utilizados para la comunicación serie. Esta directiva tiene efecto hasta que se encuentra otra directiva RS232.

La directiva **#USE DELAY** debe aparecer antes de utilizar **#USE RS232**.

Esta directiva habilita el uso de funciones tales como GETCH, PUTCHAR y PRINTF.

Programación de Microcontroladores PIC con lenguaje C

FUNCIONES DISCRETAS DE I/O

Input(pin)

Devuelve el estado '0' o '1' de la patilla indicada en pin.

El método de acceso de I/O depende de la última directiva #USE *_IO utilizada. El valor de retorno es un entero corto.

Ejemplo : if (Input(Pin_B0)==1)

Output (Pin, Value)

Esta función saca el bit dado en value(0 o 1) por la patilla de I/O especificada en pin. El modo de establecer la dirección del registro, está determinada por la última directiva #USE *_IO.

Ejemplo : output_bit(PIN_B0, 0);

Output_high(pin)

Pone a 'uno' el pin indicado. El método de acceso de I/O depende de la última directiva #USE *_IO utilizada.

Ejemplo : Output_high(PIN_C0)

Output_low(pin)

Pone a 'cero' el pin indicado. El método de acceso de I/O depende de la última directiva #USE *_IO.

Ejemplo : Output_low(PIN_D0)

Programación de Microcontroladores PIC con lenguaje C

Set_tris_puerto(Valor)

Estas funciones permiten escribir directamente los registros tri-estado para la configuración de los puertos

(configurar pines de entrada y salida).

Esto debe usarse con FAST_IO(), cuando se accede a los puertos de I/O como si fueran memoria, igual que cuando se utiliza una directiva #BYTE. Cada bit de value representa una patilla. Un '1' indica que la patilla es de entrada y un '0' que es de salida.

Ejemplo : Set_tris_A(0xff); puerto A como entrada

FUNCIONES DE RETARDO

Delay_cicles(Valor)

Esta función realiza retardos según el número de ciclos de instrucción especificado en count; los valores posibles van desde 1 a 255. Un ciclo de instrucción es igual a cuatro periodos de reloj.

Ejemplo : Delay_cicles(100); Cuenta 100 ciclos

Delay_ms(Valor)

Esta función realiza retardos del valor especificado en time. Dicho valor de tiempo es en milisegundos y el rango es 0-65535.

Para obtener retardos más largos así como retardos 'variables' es preciso hacer llamadas a una función separada; véase el ejemplo siguiente.

Delay_us(Valor)

Esta función realiza retardos del valor especificado en time. Dicho valor es en microsegundos y el rango va desde 0 a 65535. Es necesario utilizar la directiva #use delay antes de la llamada a esta función para que el compilador sepa la frecuencia de reloj.

Programación de Microcontroladores PIC con lenguaje C

FUNCIONES PARA LA MANIPULACIÓN DE BITS

Bit_clear (Var, Bit)

Esta función simplemente borra (pone a '0') el dígito especificado en bit(0-7 ó 0-15) del byte o palabra aportado en var. El bit menos significativo es el 0.

Ejemplo : int x = 10;

Bit_clear(x,0) ;

Bit_set(Var, bit)

Esta función pone a '1' el dígito especificado en bit(0-7 o 0-15) del byte o palabra aportado en var.

MANEJO DEL PROTOCOLO RS232

GETC() , GETCH(), GETCHAR()

Estas funciones esperan un carácter por el PIN **RCV** del dispositivo RS232 y retorna el carácter recibido.

Es preciso utilizar la directiva #USE RS232 antes de la llamada a esta función para que el compilador pueda determinar la velocidad de transmisión y la patilla utilizada. La directiva #USE RS232 permanece efectiva hasta que se encuentre otra que anule la anterior.

Los procedimientos de I/O serie exigen incluir

#USE DELAY para ayudar a sincronizar de forma correcta la

velocidad de transmisión. Se debe tener en cuenta que es necesario adaptar los niveles de voltaje antes de conectar el PIC a un dispositivo RS-232.

Programación de Microcontroladores PIC con lenguaje C

PUT() , PUTCHAR()

Estas funciones envían un carácter a la patilla XMIT del dispositivo RS232. Es preciso utilizar la directiva **#USE RS232** antes de la llamada a esta función para que el compilador pueda determinar la velocidad de transmisión y la patilla utilizada.

La directiva **#USE RS232** permanece efectiva hasta que se encuentre otra que anule la anterior.

Printf ([funtion], string, [valor])

La función de impresión formateada PRINTF saca una cadena de caracteres al estándar serie RS-232 o a una función especificada.

Cuando se usan variables, string debe ser una constante.

El carácter % se pone dentro de string para indicar un valor variable, seguido de uno o más caracteres que dan formato al tipo de información a representar.

Gestión de Puertos en el Microcontrolador

En el Compilador PIC C se pueden gestionar los puertos de dos formas:

Definiendo la posición de la memoria RAM como una variable C para todos los registros TRISX y PORT X.

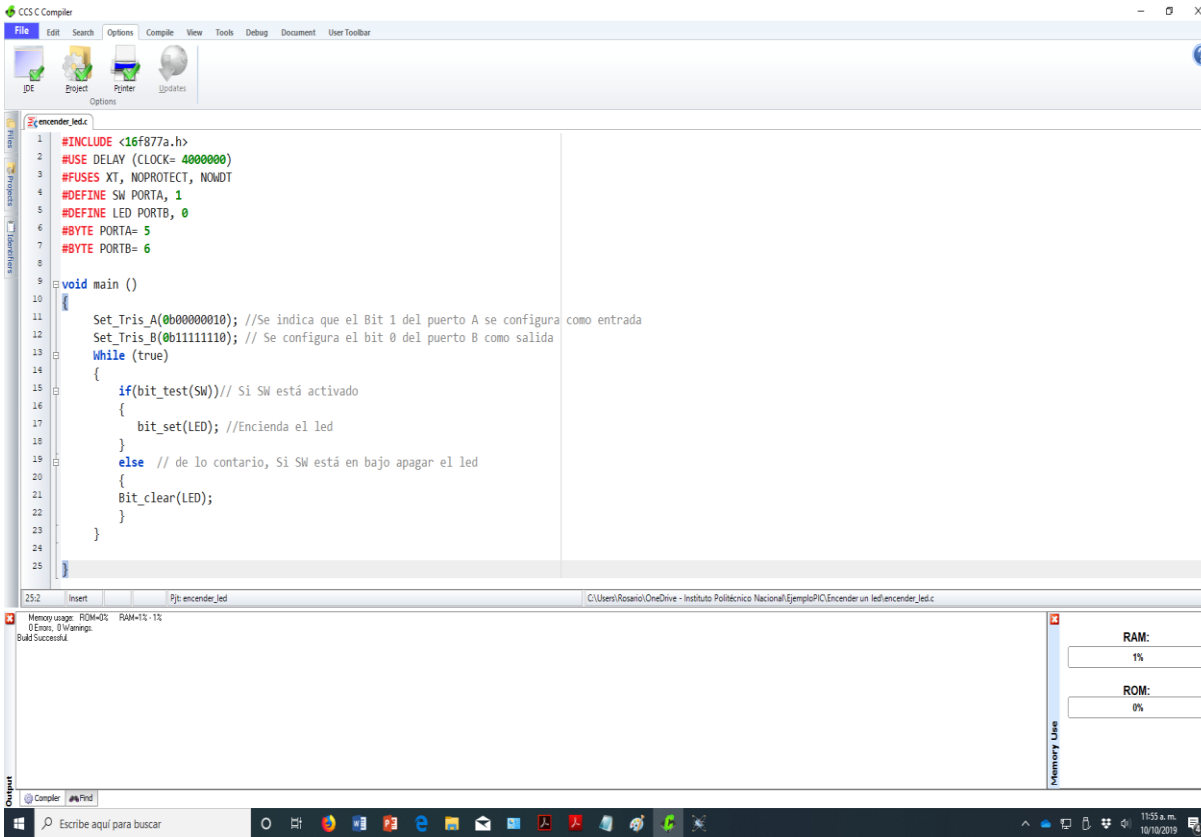
Usando las directivas específicas del compilador (#USE, #USE ESTÁNDAR_IO)

- Ejemplo:
- Usando la memoria RAM

```
#INCLUDE <16f877a.h>
#USE DELAY (CLOCK= 4000000)
#FUSES XT, NOPROTECT, NOWDT
#DEFINE SW PORTA, 1
#DEFINE LED PORTB, 0
#BYTE PORTA= 0x5 //Define el registro del PORTA en la localidad de memoria 0x5
#BYTE PORTB= 0x6 //Define el registro del PORTB en la localidad de memoria 0x6

void main ()
{
    Set_Tris_A(0b00000010); //Se indica que el Bit 1 del puerto A se configura
    Set_Tris_B(0b11111110); // Se configura el bit 0 del puerto B como salida
```

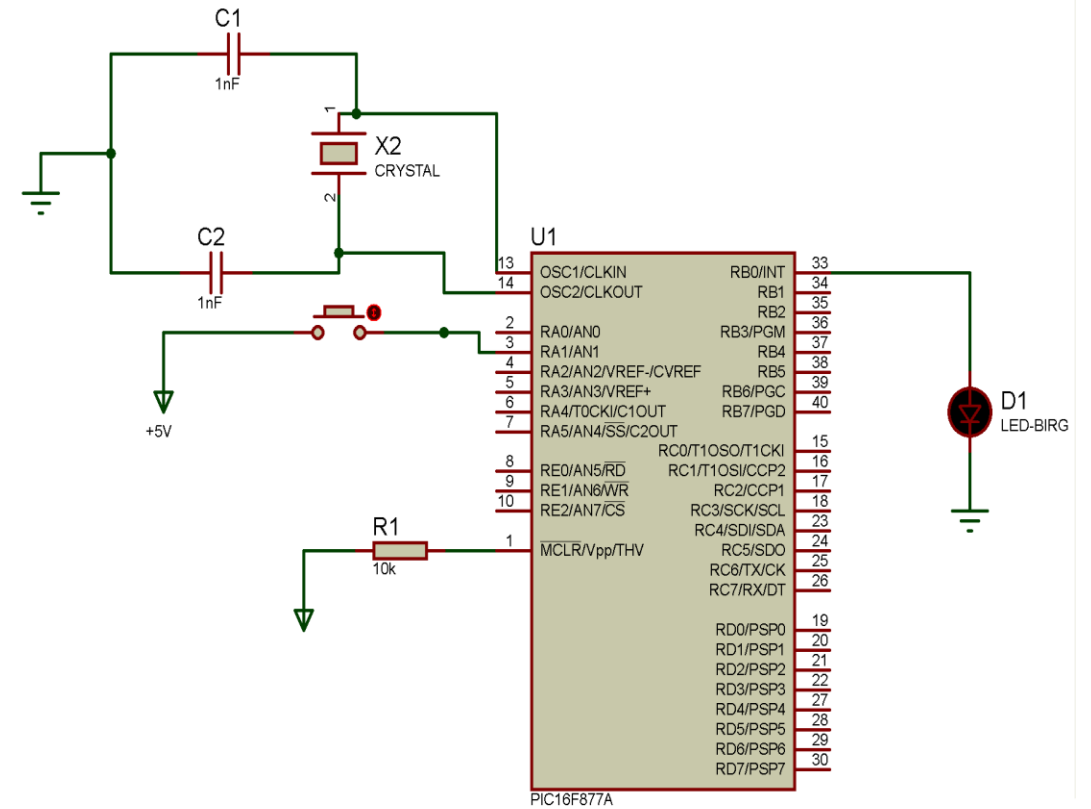
Encender un led



The screenshot shows the CCS C Compiler IDE with a project named "encender_led.c". The code is as follows:

```
1 #INCLUDE <16F877a.h>
2 #USE DELAY (CLOCK= 4000000)
3 #FUSES XT, NOPROTECT, NOWDT
4 #DEFINE SW PORTA, 1
5 #DEFINE LED PORTB, 0
6 #BYTE PORTA= 5
7 #BYTE PORTB= 6
8
9 void main ()
10 {
11     Set_Tris_A(0b00000010); //Se indica que el Bit 1 del puerto A se configura como entrada
12     Set_Tris_B(0b11111110); // Se configura el bit 0 del puerto B como salida
13     while (true)
14     {
15         if(bit_test(SW))// Si SW está activado
16         {
17             bit_set(LED); //Enciende el led
18         }
19         else // de lo contrario, Si SW está en bajo apagar el led
20         {
21             Bit_clear(LED);
22         }
23     }
24 }
```

The bottom status bar shows "Memory usage: ROM=0% RAM=1% 1%". The bottom right corner shows "RAM: 1% ROM: 0%".

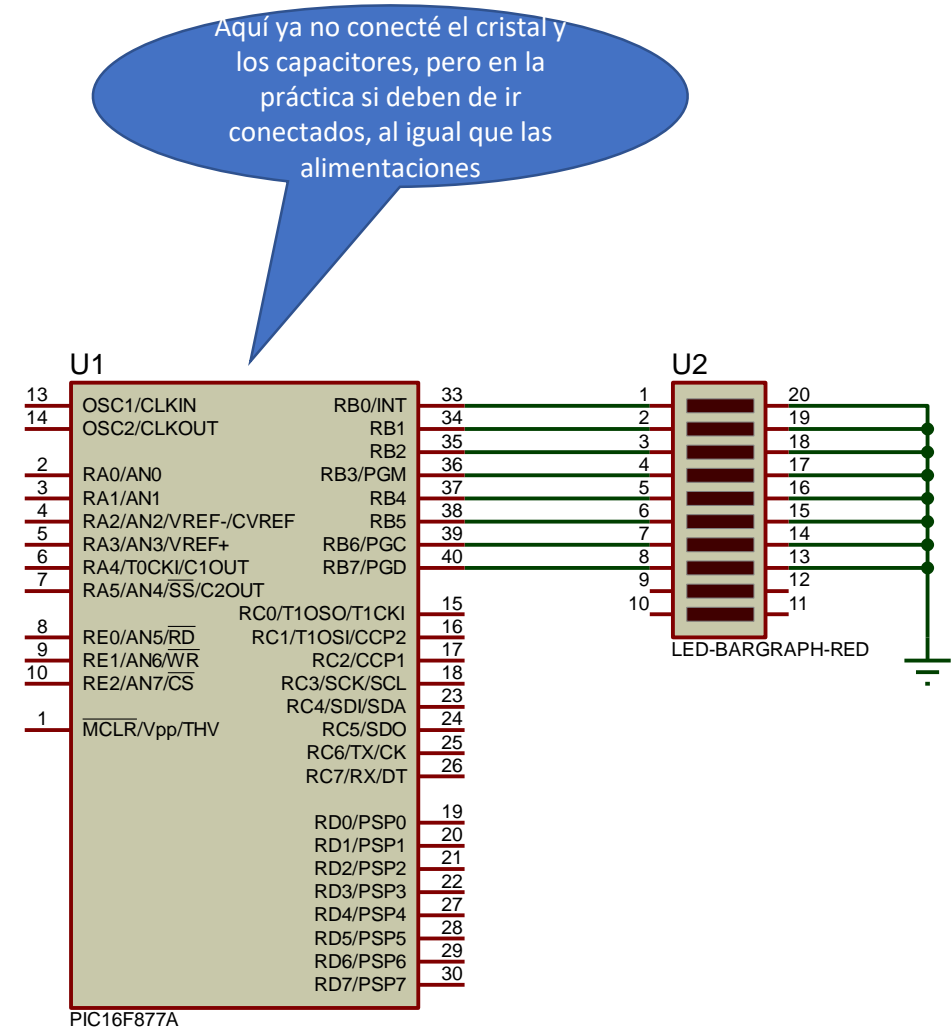


Corrimiento de leds

```
CCS C Compiler
File Edit Search Options Compile View Tools Debug Document User Toolbar
IDE Project Printer Updates
Options
corrimiento bits.c
1 #include<16f877a.h>
2 #fuses xt, nowdt
3 #use delay (clock=4M)
4
5 int8 i;
6 void main()
7 {
8
9     set_tris_b(0x00);
10
11
12
13 while(true)
14 {
15     for(i=0;i<9;i++)
16     {
17         output_b(0x01>>i);
18         delay_ms(500);
19     }
20 }
21
22
23 }
```

1:1 Insert Pjt: corrimiento bits

Memory usage: ROM=0% RAM=1% - 1%
0 Errors, 0 Warnings.
Build Successful.



Convertidor A/D del PIC16f877a

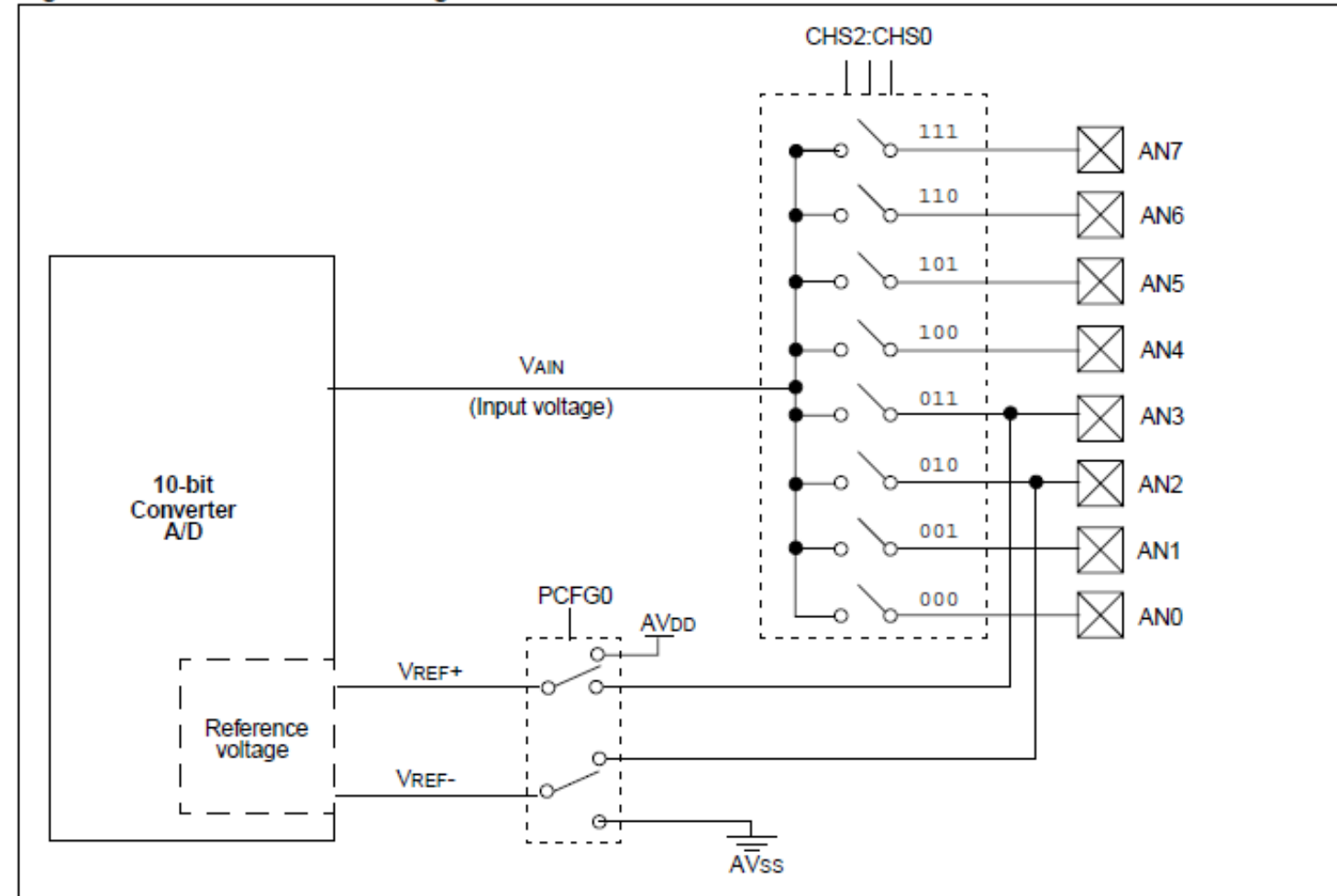
El convertidor A/D tiene 8 canales de entrada
Resolución de 10 bits.

Utiliza el método de aproximaciones sucesivas.

El módulo A / D tiene cuatro registros. Estos registros son:

- Registro alto de resultados A / D (ADRESH)
- Registro bajo de resultados A / D (ADRESL)
- Registro de control A / D0 (ADCON0)
- Registro de control A / D1 (ADCON1)

Figure 23-1: 10-bit A/D Block Diagram

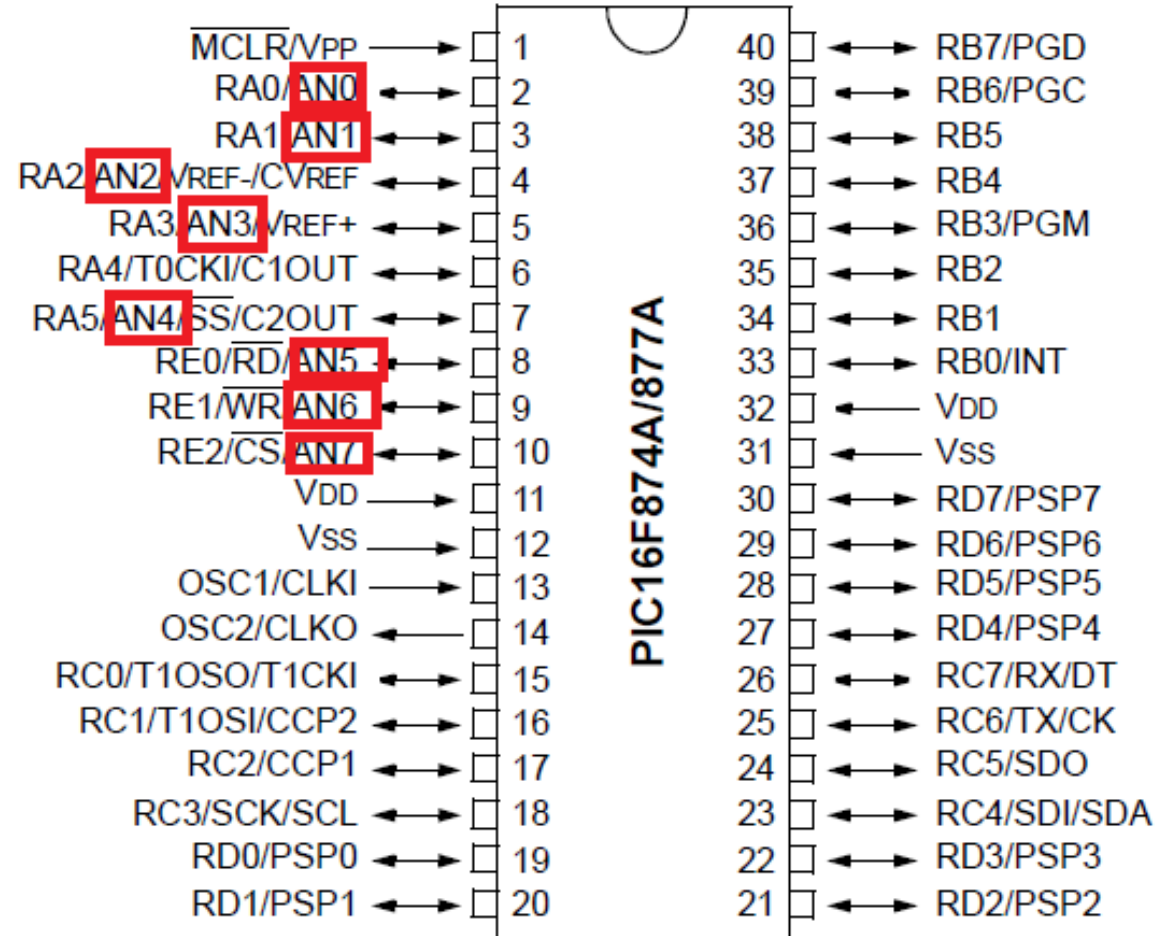


Entradas del ADC

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)

40-Pin PDIP



Configuración del ADC

SETUP_ADC_PORTS(VALOR)

Ejemplo:

SETUP_ADC_PORTS(AN0);

SETUP_ADC_(ALL_ANALOG);

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

Configuración del ADC

SETUP_ADC(MODO);

SETUP_ADC(MODO);

ADC_OFF

ADC_CLOCK_INTERNAL

ADC_CLOCK_DIV_2

ADC_CLOCK_DIV_8

ADC_CLOCK_DIV_16

ADC_CLOCK_DIV_32

ADC_CLOCK_DIV_64

Configuración del ADC

- SET_ADC_CHANNEL(canal);

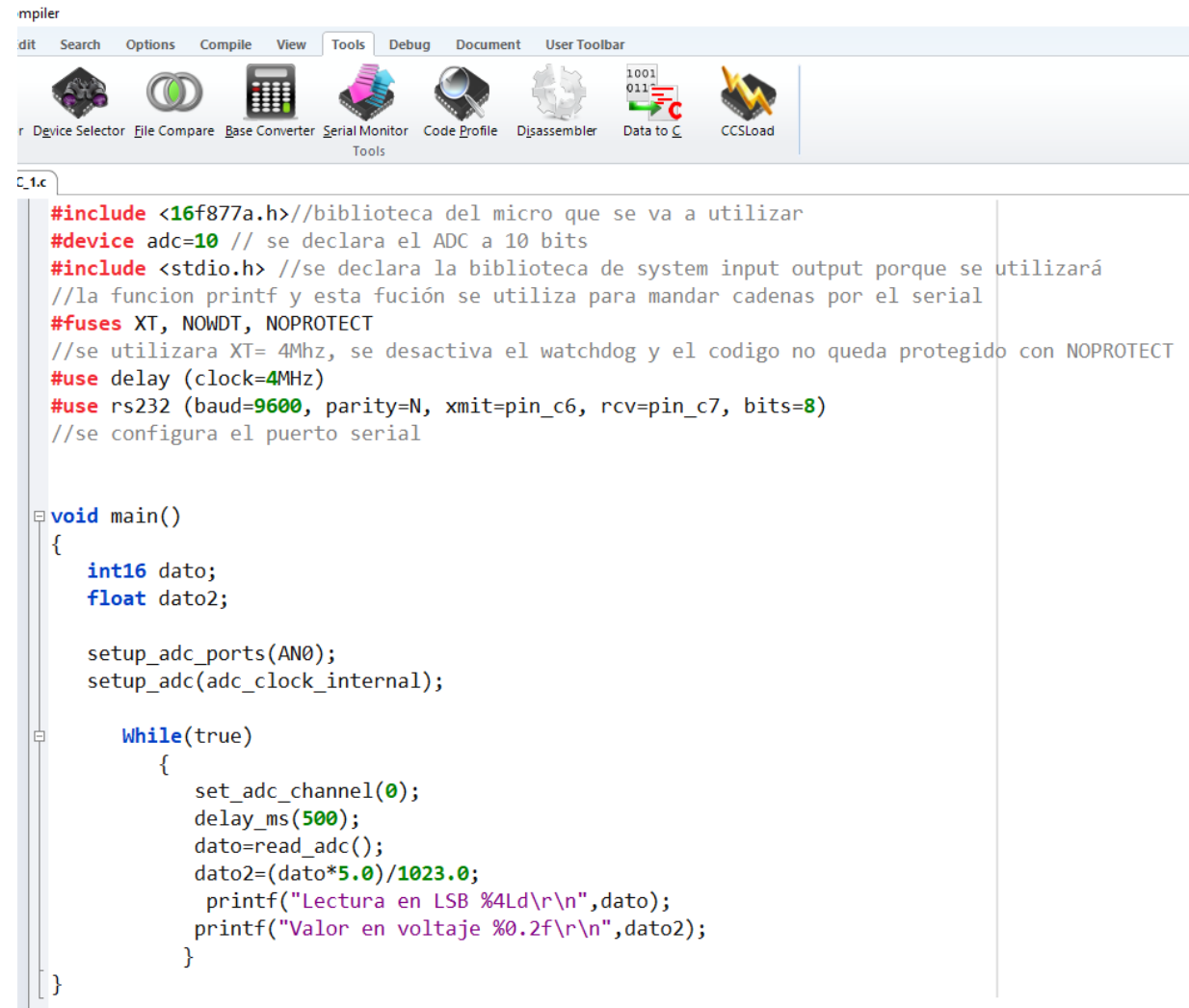
Ejemplo:

```
SET_ADC_CHANNEL(0);
```

```
SET_ADC_CHANNEL(1);
```

0 (AN0)	1 (AN1)	2 (AN2)	3 (AN3)
4 (AN5)	5 (AN6)	6 (AN7)	7 (AN8)

Configuración del ADC



The screenshot shows the MPLAB IDE interface. The menu bar includes 'mpiler', 'Edit', 'Search', 'Options', 'Compile', 'View', 'Tools', 'Debug', 'Document', and 'User Toolbar'. The 'Tools' menu is open, showing options like 'Device Selector', 'File Compare', 'Base Converter', 'Serial Monitor', 'Code Profile', 'Disassembler', 'Data to C', and 'CCSLoad'. The main window displays a C program for configuring an ADC. The code includes headers for the microcontroller, declares the ADC bits, includes the standard I/O library, and sets up the serial port. The main function initializes the ADC and enters a loop that reads the ADC value and prints it to the serial port.

```
#include <16f877a.h> // biblioteca del micro que se va a utilizar
#define adc=10 // se declara el ADC a 10 bits
#include <stdio.h> // se declara la biblioteca de system input output porque se utilizará
// la función printf y esta función se utiliza para mandar cadenas por el serial
#define XT, NOWDT, NOPROTECT
// se utilizará XT= 4Mhz, se desactiva el watchdog y el código no queda protegido con NOPROTECT
#define delay (clock=4Mhz)
#define rs232 (baud=9600, parity=N, xmit=pin_c6, rcv=pin_c7, bits=8)
// se configura el puerto serial

void main()
{
    int16 dato;
    float dato2;

    setup_adc_ports(AN0);
    setup_adc(adc_clock_internal);

    while(true)
    {
        set_adc_channel(0);
        delay_ms(500);
        dato=read_adc();
        dato2=(dato*5.0)/1023.0;
        printf("Lectura en LSB %4d\r\n",dato);
        printf("Valor en voltaje %.2f\r\n",dato2);
    }
}
```

INTERRUPCIONES

Una interrupción se define como un pedido de alta prioridad que un dispositivo exterior o un evento de programación solicita a la CPU para ejecutar otro programa.

El microcontrolador acepta dos tipos de interrupciones:

- Interrupciones por periféricos
- Interrupciones externas

- El microcontrolador pic 16f87x posee las siguientes fuentes de interrupción:

Interrupción externa por RB0/INT.

- Interrupción por cambio de nivel lógico en RB4 - RB7.
- Interrupción por desborde del timer 0.
- Interrupción del transmisor del modulo USART.
- Interrupción del receptor del modulo USART.
- Interrupción del modulo CPP.
- Interrupción del EEPROM.

Directivas habituales del compilador C

Las posibles directivas para la familia 16F87x son las siguientes:

#INT_AD ----- Conversión AD completa

#INT_BUSCOL ----- Colisión de bus

#INT_CPP1 ----- Unidad de captura 1, comparación y PWM

#INT_CPP2 ----- Unidad de captura 1, comparación y PWM

#INT_EEPROM ----- Escritura EEPROM finalizada

#INT_EXT ----- Interrupción externa RB0

#INT_RB ----- Cambio de estado en B4-B7

#INT_RDA ----- RS232 dato recibido