UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE TECNOLOGÍA

CARRERA DE ELECTRÓNICA Y TELECOMUNICACIONES

"DISEÑO DE UN SISTEMA ELECTRÓNICO DISPENSADOR DE COMIDA PARA MASCOTAS"

Trabajo de Aplicación - Examen de Grado presentado para obtener el Grado deLicenciatura

POR: WARA ABIGAIL HUANCA HIDALGO

LA PAZ – BOLIVIA

Noviembre, 2022

DEDICATORIA

Primeramente, a Dios por darme la oportunidad de lograr una meta importante en mi vida.

A mis queridos padres Jorge Reynaldo Huanca Cazas y Laura Luisa Hidalgo Cordero quienes me dieron la vida, amor, cariño y gracias a ellos he llegado a ser la persona que soy en la actualidad.

A mis hermanos Aleyda y Danilo que siempre me brindaron su amor y cariño.

AGRADECIMIENTO

Agradecido con Dios por la fortaleza, por la salud, por mi familia y por darme la oportunidad de lograr una meta importante en mi vida, a mis padres que me apoyaron y ayudaronen mi desarrollo, estudio y mi formación, a mis docentes que hicieron que miconocimiento crezca y se fortalezca hasta formar a la persona que soy hoy en día
Tengo que agradecer al apoyo de mi familia que me han ayudado en los momentos más difíciles y alentado a alcanzar mis objetivos.

INDICE

1. RESUMEN

El presente trabajo de aplicación titulado "DISEÑO DE UN SISTEMA ELECTRÓNICO DISPENSADOR DE COMIDA PARA MASCOTAS" para optar por el titulo de Licenciatura en Electrónica y Telecomunicaciones de la Universidad Mayor de San Andrés de la Facultad de Tecnología.

En el presente proyecto se realizará el diseño de un sistema electrónico dispensador de comida para mascotas parael hogar el cual sea capaz de funcionar con un sensor de proximidad que permita que en cuantos se aproxime la mascota a su plato de comida se dispense comida, además de poder controlarlo a través de Wifi y poder brindar el alimento a las mascotas.

Para lograr dicho proyecto se utilizará como base el microprocesador PIC16F877A como la central y encargada de tomar las decisiones, además sensor de proximidad que permitirá saber si la mascota se encuentra cerca y un servo motor que permitirá que se dispense la comida automáticamente.

El proyecto también contará con un módulo Wifi para poder controlarlo y se pueda llegar a dispensar la comida sin necesidad de que la mascota se encuentre cerca de su plato.

2. PLANTEAMIENTO DEL PROBLEMA

Bolivia es un país donde un gran porcentaje de sus habitantes tienen alguna mascota en casa, que en su mayoría con perros y gatos, además por cada 10 habitantes hay un perro o gato en Bolivia. (Correo del Sur, 21/07/2019)

En la actualidad, las familias tienen una mascota debido a que la misma llega a representar una parte de la compañía y cariño en el hogar. Sin embargo, este integrante de la familia representa una responsabilidad para todos los integrantes

de ella lo que implica que no se la debe descuidar y tratar de satisfacer todas sus necesidades básicas.

Para los integrantes de la familia llega a ser muy dificultoso el hecho de cumplir los horarios de alimentación debido a que en el mundo actual se tienen responsabilidades diarias que pueden llegar a causar que no se dispense la comida en su tiempo ni en cuanto la mascota lo requiera durante el día.

Asimismo, se llega a complicar la situación para los dueños de las mascotas debido por situaciones laborales se retrasan al retornar a casa o tiene que realizar viajes, por otro lado, muchas familias deciden tomarse vacaciones que muchas veces se imposibilita realizarla con una mascota por lo que se la deja sola en casa.

La mala nutrición de las mascotas llega a generarles enfermedades que afectan a su metabolismo y desarrollo de sus huesos y órganos a una temprana edad. Las principales afecciones que llega a tener una mascota a causa de la mala alimentación es la obesidad, deficiencia de desarrollo óseo, problemas cardiovasculares, respiratorios, sudoración, pérdida de peso, anemia, descalcificación ósea principalmente en la columna en los perros, alergias, diarrea, aumento en la orina, vómitos, gastritis, entre otros. (Hector Morlote ,2013)

Por lo que es importante garantizar la nutrición de las mascotas tomando en cuenta las siguientes condiciones: que sea administrada en proporciones adecuadas, que se provea el alimento que contenga los nutrientes necesarios para el correcto desarrollo animal.

3. JUSTIFICACIÓN DEL TRABAJO

En respuesta a la necesidad de poder alimentar a las mascotas cuando el dueño no se encuentra en casa surge el Sistema electrónico Dispensador de comida para

mascotas el cual es capaz de brindar la alimentación necesaria y requerida por la mascota.

El sistema electrónico de dispensador de comida para mascotas tiene la posibilidad de ser usado a requerimiento de la mascota o que uno dispense en los horarios determinados el alimento, mismas opciones que pueden ser utilizadas una a la vez.

Se tendrá un sensor para verificar el acercamiento de la mascota y también un motor que realizará la dispensación de la comida.

Además, el sistema dispensador de comida para mascotas podrá ser controlado por el módulo Wifi desde nuestros celulares.

Es por lo que podrá brindar el alimento a nuestra mascota de manera correcta, evitando enfermedades que podrían aquejarlo en un futuro y cuidando a un integrante importante de familia.

4. OBJETIVOS

4.1 GENERAL

Diseñar un Sistema electrónico dispensador de comida para mascotas, utilizando el microcontrolador PIC16F877A.

4.2 ESPECIFICO

- Desarrollar el hardware del Sistema electrónico Dispensador de comida para mascotas utilizandocomo base el microcontrolador PIC16F877A
- Desarrollar el Software para cumplir con las funciones más importantes de un dispensador de comida para mascotas.

- Desplegar el hardware para poder dispensar la comida de nuestra mascota.
- Realizar el software de dispensación de comida de nuestra mascota por un sensor de proximidad o en función a un módulo Wifi por la cual podemos brindar la comida según nuestra decisión.
- Utilizar sensor de proximidad para el sistema dispensador de comida.

5. FUNDAMENTACIÓN TEÓRICA

5.1 EL MICROCONTROLADOR

5.1.1 EL MICROCONTROLADOR

El microcontrolador fue inventado por Texas Instruments en la década de 1970, fue inventado casi al mismo tiempo que el primer microprocesador que estaba siendo inventado en Intel.

Los primeros microcontroladores eran simplemente microprocesadores con una función de memoria, como la memoria RAM y ROM. Más tarde, los microcontroladores se desarrollaron en una amplia gama de dispositivos diseñados para aplicaciones de sistemas integrados específicos en dispositivos tales como automóviles, teléfonos móviles y electrodomésticos.

En 1971, fue inventado el primer microcontrolador por dos ingenieros de Texas Instruments,de acuerdo con el Instituto Smithsoniano. Gary Boone y Michael Cochran crearon el TMS 1000, el cual era un microcontrolador de 4 bits con función de ROM y RAM. *(Hector Morlote, 2013)*

El mismo era utilizado internamente en Texas Instruments en sus productos de cálculo desde 1972 hasta 1974, y fue refinado con el paso de los años.

En 1974, TI puso a la venta el TMS 1000 para la industria de electrónicos. El TMS 1000 estuvo disponible en varios tamaños de RAM y ROM. A partir de

1983, cerca de un millón de TMS 1000 fueron vendidos. *(Steve Aycock, 2001-2018)*

5.1.2 Arquitectura del Microcontrolador

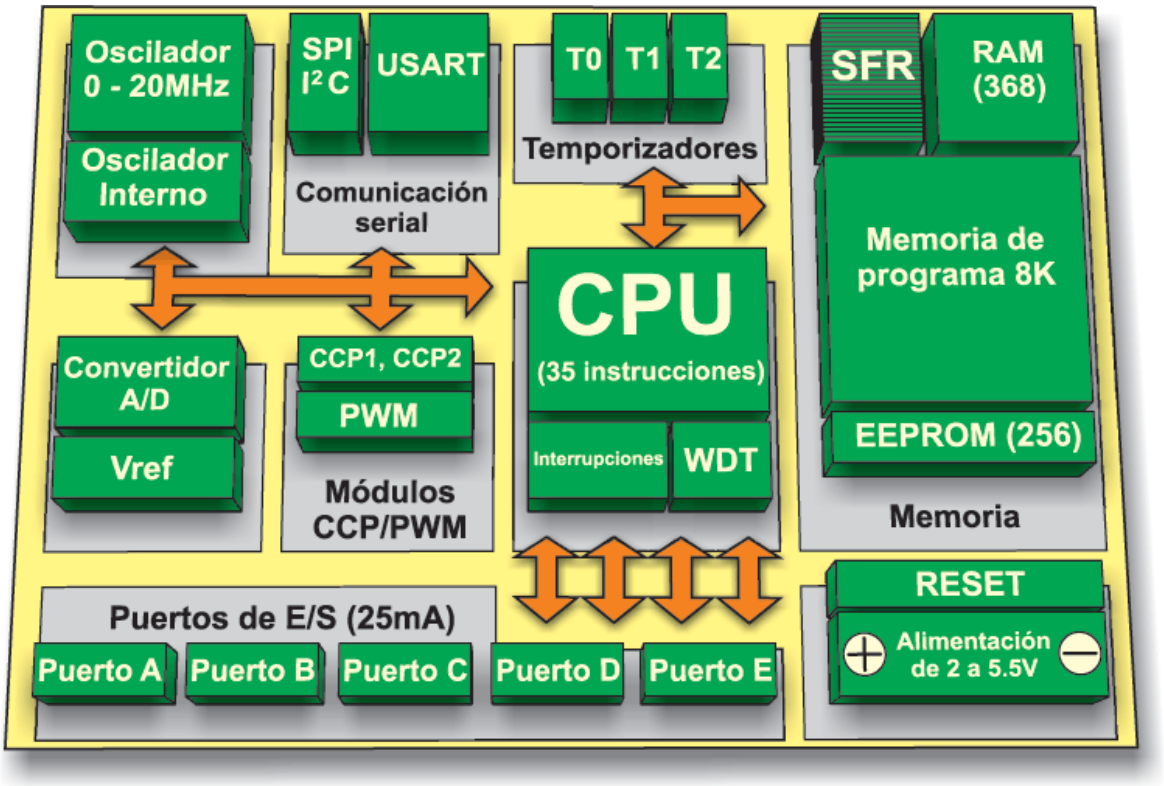


Figura 1. Arquitectura del Microcontrolador
Fuente:https://aprendiendoarduino.wordpress.com/2017/09/03/microcontroladores-arduino-a-fondo/

En la arquitectura del microcontrolador (ejemplo PIC16F877A) tenemos lo siguiente:

- ✓ Memoria de programa: FLASH de 8K de instrucciones de 14 bits
- ✓ Memorias de datos: SRAM de 512 bytes, EEPROM de 256 bytes
- ✓ Pines I/O (Input/Output) : 6 del puerto A, 8 del puerto B, 8 del puerto C, 8 del puertoD y 3 del puerto E, además de 8 entradas analógicas.

- ✓ Pila (Stack): 8 niveles (14 bits)
- ✓ Fuentes de interrupción: 14
- ✓ Instrucciones: 35
- ✓ Compatible modo SLEEP
- ✓ Frecuencia máxima del oscilador de 20MHz
- ✓ Conversor Analógico/Digital de 10 bits multicanal (8 canales de entrada)
- ✓ Corriente máxima absorbida/suministrada (sink/source) por pin: 25 mA
- ✓ Voltaje nominal: 3 a 5.5V DC (CMOS)
- ✓ Power On Reset
- ✓ Power Up Timer (PWRT)
- ✓ Oscilador Start Up Timer (OST)

5.1.3 Microcontrolador del PIC 16F877A

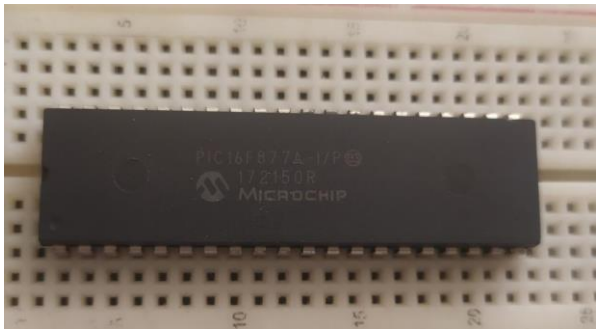


Figura 2. PIC 16F877A de Microchip

Fuente: Propia

Para el uso de un nuevo dispositivo debemos de descargar el datasheet u hoja de características, para adquirir una serie de conocimientos necesarios para su correcto uso.

El PIC 16F877A es un microcontrolador de Microchip Technology fabricado en tecnología CMOS, con un consumo de potencia muy bajo, además es completamente estático (lo que significa que el reloj puede detenerse y los datos

de la memoria no se pierden). Tiene una memoria de programa tipo FLASH, lo que permite reprogramarlo nuevamente sin tener queborrado lo anterior. *(Hector Morlote, 2013)*

5.1.3.1 Resumen de características principales del PIC 16F877

Las características principales del microcontrolador PIC16F877A son:

- ✓ Memoria de programa: FLASH de 8K de instrucciones de 14 bits
- ✓ Memorias de datos: SRAM de 512 bytes, EEPROM de 256 bytes
- ✓ Pines I/O (Input/Output) : 6 del puerto A, 8 del puerto B, 8 del puerto C, 8 del puertoD y 3 del puerto E, además de 8 entradas analógicas.
- ✓ Pila (Stack): 8 niveles (14 bits)
- ✓ Fuentes de interrupción: 14
- ✓ Instrucciones: 35
- ✓ Compatible modo SLEEP
- ✓ Frecuencia máxima del oscilador de 20MHz
- ✓ Conversor Analógico/Digital de 10 bits multicanal (8 canales de entrada)
- ✓ Corriente máxima absorbida/suministrada (sink/source) por pin: 25 mA
- ✓ Voltaje nominal: 3 a 5.5V DC (CMOS)
- ✓ Power On Reset
- ✓ Power Up Timer (PWRT)
- ✓ Oscilador Start Up Timer (OST)

El encapsulado que he utilizado es de tipo DIP (Dual In-Line Pin) de 40 pines, aunque posee otros encapsulados (SOIC, PLCC y QFP

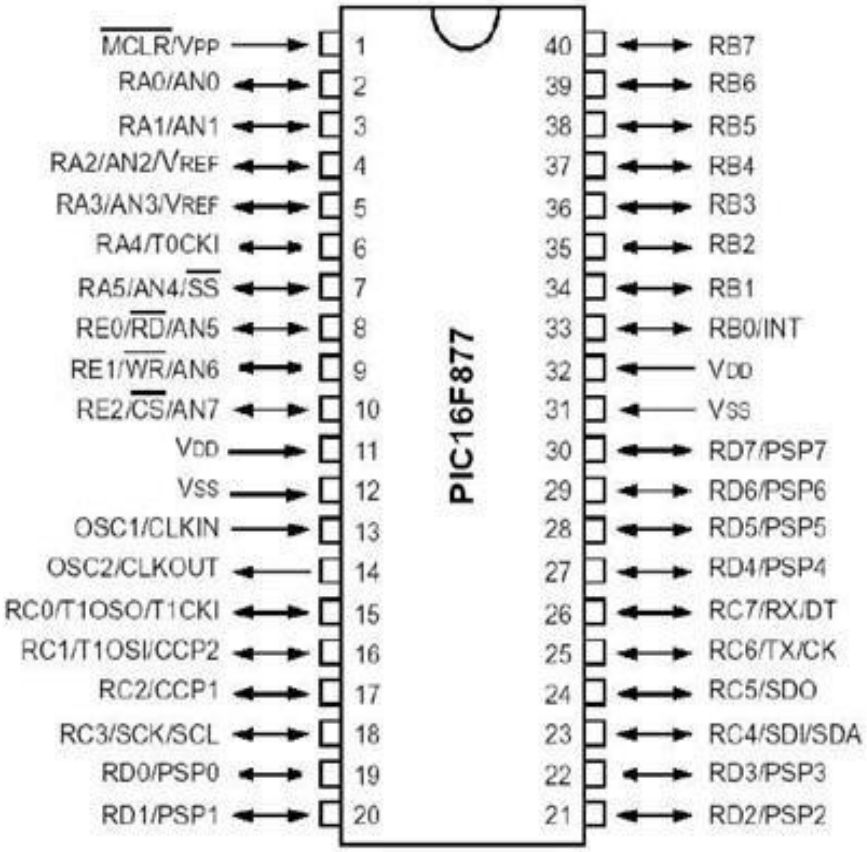


Figura 3. Encapsulado DIP del PIC 16F877 y la distribución de sus 40 pines.
Fuente: https://www.researchgate.net/figure/Figura-229-Distribucion-de-pines-del-microcontrolador-PIC16F877A_fig17_27557892

Sus pines I/O (Input/Output) están organizados en 5 puertos que son:

- ✓ Puerto A: 6 pines
- ✓ Puerto B: 8 pines
- ✓ Puerto C: 8 pines
- ✓ Puerto D: 8 pines
- ✓ Puerto E: 3 pines

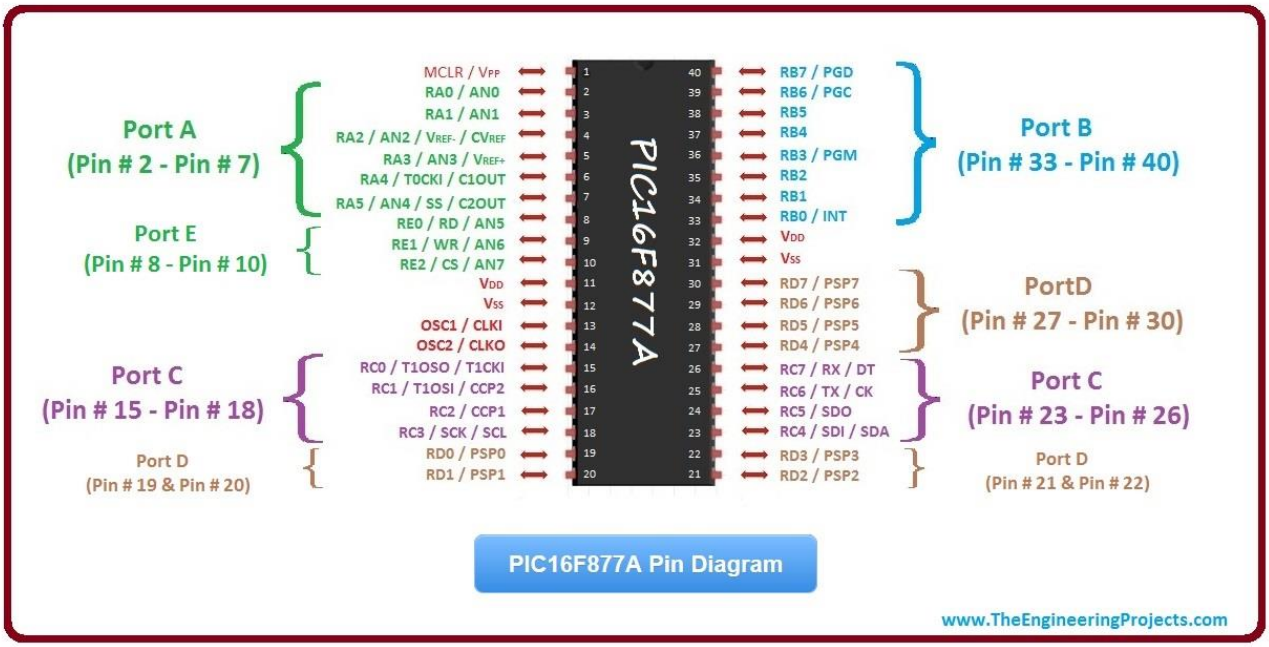


Figura 4. Puertos de del microcontrolador PIC16F877A.

Fuente: https://www.theengineeringprojects.com/2017/06/pic16f877a.html

Cada pin de esos puertos se puede configurar como entrada o como salida independiente programando un par de registros diseñados para tal fin.

En ese registro un bit en “0” configura el pin del puerto correspondiente como salida y un biten “1” lo configura como entrada.

Dichos pines del microcontrolador también pueden cumplir otras funciones especiales, siempre y cuando se configuren para ello. En la siguiente tabla se indican las funciones de todos los pines del PIC

Nombre	Número (DIP 40)	Función	Descripción
RA7/OSC1/CLKIN	13	RA7	E/S de próposito general en el puerto PORTA
		OSC1	Entrada del oscilador de cristal
		CLKIN	Entrada del reloj externo
RA6/OSC2/CLKOUT	14	OSC2	Salida del oscilador del cristal
		CLKO	Salida en la que se presenta la señal Fosc/4
		RA6	E/S de propósito general en el puerto PORTA
RC0/T1OSO/T1CKI	15	RC0	E/S de propósito general en el puerto PORTC
		T1OSO	Salida del oscilador del temporizador 1
		T1CKI	Entrada de reloj del temporizador 1
RC1/T1OSO/T1CKI	16	RC1	E/S de propósito general en el puerto PORTC
		T1OSI	Entrada del oscilador del temporizador 1
		CCP2	E/S de los módulos CCP1 y PWM1
RC2/P1A/CCP1	17	RC2	E/S de propósito general en el puerto PORTC
		P1A	Salida del módulo PWM
		CCP1	E/S de los módulos CCP1 y PWM1
RC3/SCK/SCL	18	RC3	E/S de propósito general en el puerto PORTC
		SCK	E/S de reloj del módulo MSSP en el modo SPI
		SCL	E/S de reloj del módulo MSSP en el modol^{2}C
RD0	19	RD0	E/S de propósito general en el puerto PORTD
RD1	20	RD1	E/S de propósito general en el puerto PORTD
RD2	21	RD2	E/S de propósito general en el puerto PORTD
RD3	22	RD3	E/S de propósito general en el puerto PORTD
RC4/SDI/SDA	23	RC4	E/S de propósito general en el puerto PORTC
		SDI	Entrada *Data* del módulo MSSP en el modo SPI
		SDA	E/S *Data* del módulo MSSP en el modo I^{2}C
RC5/SDO	24	RC5	E/S de propósito general en el puerto PORTC
		SDO	Salida *Data* del módulo MSSP en el modo SPI
RC6/TX/CK	25	RC6	E/S de propósito general en el puerto PORTC
		TX	Salida asíncrona del módulo USART
		CK	Reloj síncrono del módulo USART
RC7/RX/DT	26	RC7	E/S de propósito general en el puerto PORTC
		RX	Entrada asíncrona del módulo USART
		DT	Datos del módulo USART en modo síncrono

Tabla 1. Funciones de los pines del PIC16F877A

Fuente: https://www.mikroe.com/ebooks/microcontroladores-pic-programacion-en-c-con-ejemplos/caracteristicas-basicas-del-pic16f887

En la tabla anterior las “E/S” hacen referencia a “Entrada/Salida”, lo mismo que “I/O”(Input/Output).

TABLE 7-2: JUEGO DE INSTRUCCIONES DEL PIC16CXXX

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb	LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS							
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff		C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff		Z	1,2
CLRF	f	Clear f	1	00 0001 lfff ffff		Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx		Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff		Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff		Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1 (2)	00 1011 dfff ffff			1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff		Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1 (2)	00 1111 dfff ffff			1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff		Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff		Z	1,2
MOVWF	f	Move W to f	1	00 0000 lfff ffff			
NOP	-	No Operation	1	00 0000 0xx0 0000			
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff		C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff		C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff		C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff			1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff		Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS							
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff			1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff			1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff			3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff			3
LITERAL AND CONTROL OPERATIONS							
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk		C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk		Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk			
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100		$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk			
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk		Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk			
RETFIE	-	Return from interrupt	2	00 0000 0000 1001			
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk			
RETURN	-	Return from Subroutine	2	00 0000 0000 1000			
SLEEP	-	Go into standby mode	1	00 0000 0110 0011		$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk		C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk		Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 Module.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

Note: Additional information on the mid-range instruction set is available in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023).

Tabla 2. Juego de instrucciones del PIC16F877A

Fuente: https://www.diarioelectronicohoy.com/blog/imagenes/2010/12/juego-de-instrucciones-PIC.gif

En la tabla anterior las se observa el juego de instrucciones para poder programar en el lenguaje assembler o ensamblador.

5.1.3.2 Arquitectura interna

Este término se refiere a los bloques funcionales que componen en PIC internamente, comola memoria RAM, la memoria FLASH, la lógica de control, etc.

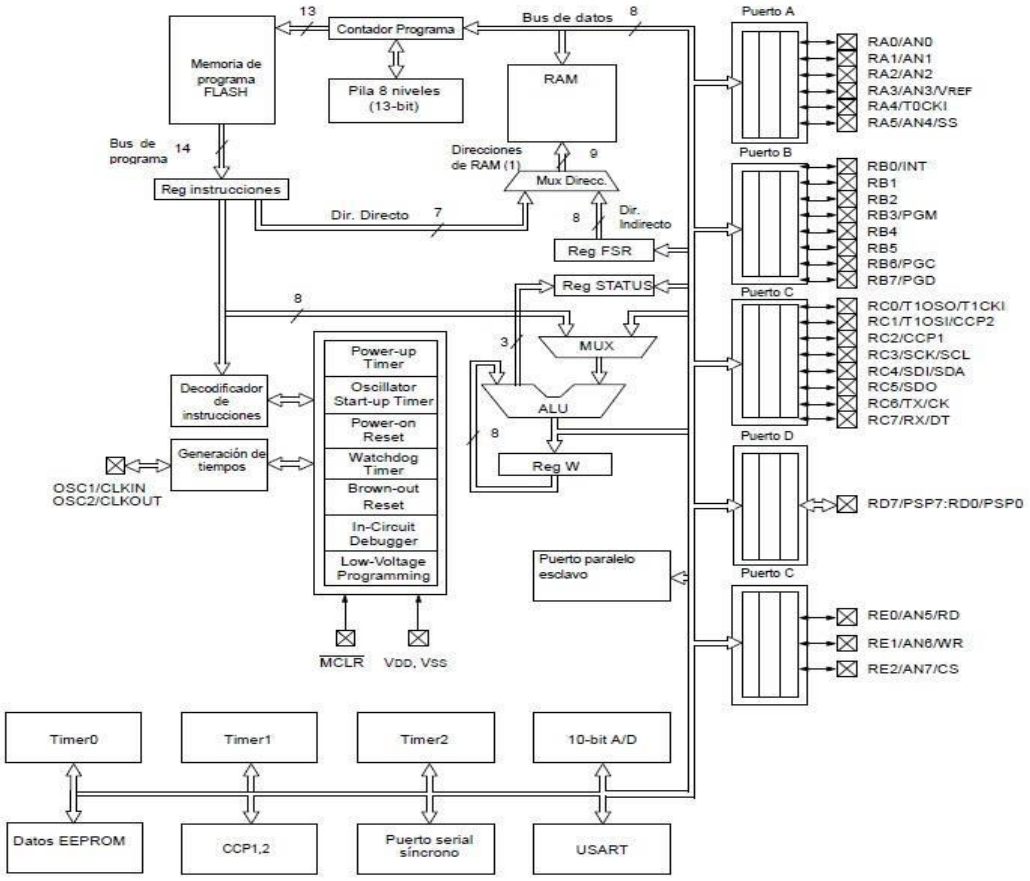


Figura 5. Estructura interna del PIC 16F877.

Fuente: https://cifpn1hectorm.wordpress.com/2013/04/10/estudio-de-la-estructura-interna-delpic-16f877/

El PIC 16F877 se basa en la arquitectura Harvard, en la cual el programa y los datos se pueden trabajar con buses (un bus es un conjunto de líneas que transportan información entre2 o más módulos) y memorias separadas, lo cual permite que las instrucciones y los datos tengan longitudes diferentes.

5.1.3.3 Memoria de programa (FLASH)

Es una memoria de 8K de capacidad con posiciones de 14 bits. En ella se graba o almacena el programa o códigos que el microcontrolador debe ejecutar.

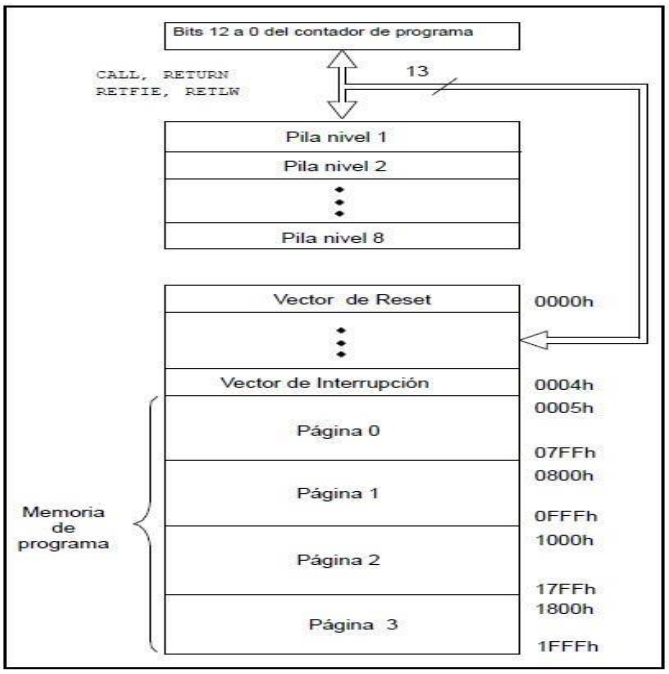


Figura 6. Mapa de memoria de programa (FLASH).
Fuente: https://cifpn1hectorm.wordpress.com/2013/04/10/estudio-de-la-estructura-interna-delpic-16f877/

En el mapa de la memoria FLASH tenemos lo siguiente:

- La memoria está dividida en cuatro páginas de 2K cada una. La Página 0 va de la posición de memoria 0005h a la 07FFh, la Página 1 de 0800h a 0FFFh, la Página 2 de 1000h a 17FFh y la Página 3 de 1800h a 1FFFh.

- El contador de programa (en este caso es de 13 bits) nos indica la dirección de la instrucción a ejecutar.

- Pila (Stack): son registros que no forman parte de ningún banco de memoria (los bancos de memoria los explico más abajo) y no permiten el acceso por parte del usuario. Se usan para guardar el valor del contador de programa cuandose hace un llamado a una subrutina o a una interrupción. Cuando el

micro vuelva a ejecutar su tarea normalmente, el contador de programa recupera su valor leyéndolo en la pila. Al tener una pila de 8 niveles, se pueden acumular 8 llamadas a subrutinas sin tener problemas.

- *Vector de RESET:* cuando se resetea el microcontrolador el contador de programa se pone a cero (0000h). Por esto, en la primera dirección del programa se debe escribir todo lo relacionado con la iniciación del mismo.

- *Vector de Interrupción:* cuando el microcontrolador recibe una llamada a una interrupción, el contador de programa apunta a la dirección 04H de la memoria de programa, por eso allí se debe escribir toda la información necesaria para atender dicha interrupción. *(Hector Morlote, 2013)*

5.1.3.4 Memorias de datos

El PIC16F877 tiene 2 memorias de datos:

5.1.3.4.1 La Memoria SRAM *(*Static Random Access Memory*)*

*E*s una memoria de tipo volátil (cuando deja de recibir alimentación se borran los datos que tenga almacenados) que está dividida en 4 bancos de 128 bytes cada uno. De esos 128 byteslos 32 primeros están dedicados a los SFR´s (Registros de Funciones Especiales, cumplen unpropósito general en el control y configuración del microcontrolador) y los 96 siguientes a los GPR´s (Registros de Propósitos Generales, se pueden usar para guardar los datostemporales de la tarea que se está ejecutando).

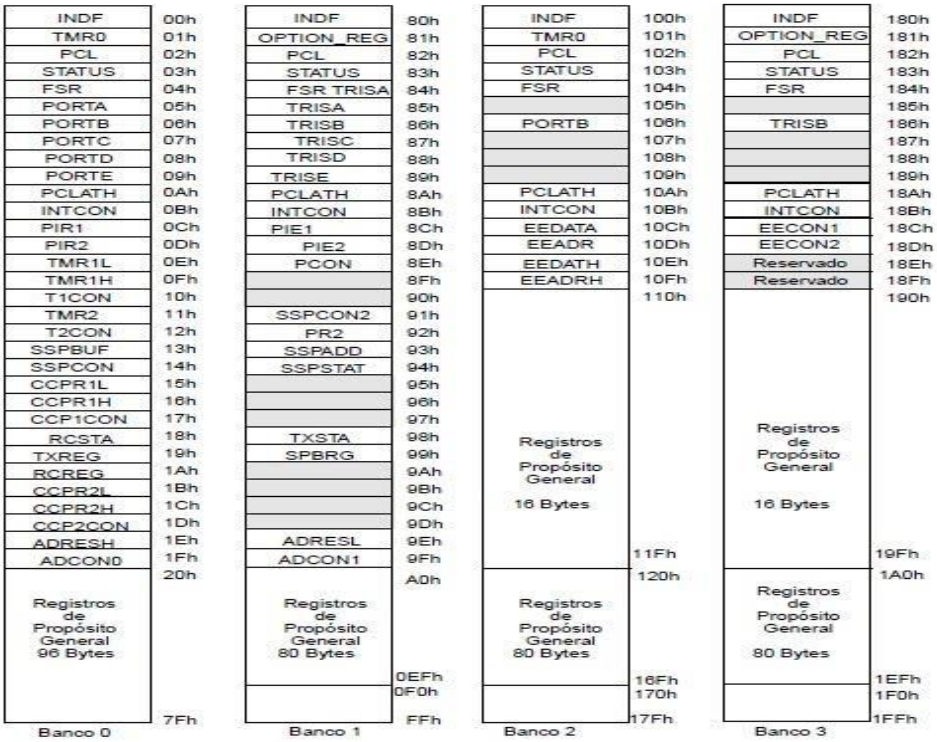


Tabla 3. Registros del PIC 16F877 y sus direcciones.

Fuente: https://cifpn1hectorm.wordpress.com/2013/04/10/estudio-de-la-estructurainterna-delpic-16f877/

5.1.3.4.2 La Memoria EEPROM

Es una memoria no volátil (guarda los datos, aunque le falte alimentación) con una capacidadde 256 bytes, que permite realizar operaciones de lectura y escritura sin interferir con el funcionamiento normal del microcontrolador.

5.1.3.5 Reloj u Oscilador

El pequeño circuito externo que los microcontroladores necesitan para que se les indique la velocidad de trabajo es conocido como reloj u oscilador. En función del montaje que se realice se puede conseguir más o menos precisión. En el momento de programar (o quemar los fusibles) el PIC se debe especificar el tipo de oscilador externo que se va a utilizar. El PIC 16F877A puede utilizar 4 tipos de oscilador diferentes:

- XT: Cristal genérico (de 1 a 4 MHz).

- *RC:* Oscilador con resistencia y condensador.
- *HS:* Cristal de alta frecuencia (de 10 a 20 MHz).
- *LP:* Cristal para baja frecuencia y bajo consumo.
- Las configuraciones más utilizadas son la XT y RC:

La configuración XT se suele utilizar con un cristal de 4 MHz, pues garantiza precisión y esbastante comercial. Internamente esta frecuencia es dividida entre 4, lo que hace que la frecuencia efectiva de trabajo sea de 1 MHz en este caso, por lo que cada instrucción se ejecuta en 1 μs (1 microsegundo):

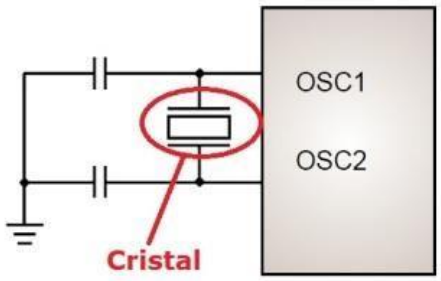


Figura 7. Oscilador XT

Fuente: https://cifpn1hectorm.wordpress.com/2013/04/10/estudio-de-la-estructura-interna-delpic-16f877/

El cristal debe ir acompañado de 2 condensadores.

La configuración RC se utiliza si no se precisa una gran precisión y se quiere economizar dinero:

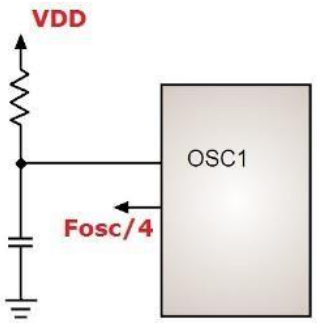


Figura 8. Oscilador RC

Fuente: https://cifpn1hectorm.wordpress.com/2013/04/10/estudio-de-la-estructura-interna-delpic-16f877/

Sólo se necesita una resistencia y un condensador. *(Hector Morlote, 2013)*

5.2. SENSOR DE DISTANCIA HC-SR04

Figura 9. Sensor HC-SR04

Fuente: PROPIA

El sensor HC-SR04 es un sensor de distancia de bajo costo que utiliza ultrasonido para determinar la distancia de un objeto en un rango de 2 a 450 cm. Destaca por su pequeño tamaño, bajo consumo energético, buena precisión y excelente precio. El sensor HC-SR04 es el más utilizado dentro de los sensores de tipo ultrasonido, principalmente por la cantidad de información y proyectos disponibles en la web. De igual forma es el más empleado en proyectos de robótica como robots laberinto o sumo, y en proyectos de automatización como sistemas de medición de nivel o distancia.

El sensor HC-SR04 posee dos transductores: un emisor y un receptor piezoeléctricos, además de la electrónica necesaria para su operación. El funcionamiento del sensor es el siguiente: el emisor piezoeléctrico emite 8 pulsos de ultrasonido(40KHz) luego de recibir la orden en el pin TRIG, las ondas de sonido viajan en el aire y rebotan al encontrar un objeto, el sonido de rebote es detectado por el receptor piezoeléctrico, luego el pin ECHO cambia a Alto (5V) por un tiempo igual al que demoró la onda desde que fue emitida hasta que fue detectada, el tiempo del pulso ECO es medido por el microcontrolador y asi se

puede calcular la distancia al objeto.

El funcionamiento del sensor no se ve afectado por la luz solar o material de color negro (aunque los materiales blandos acústicamente como tela o lana pueden llegar a ser difíciles de detectar).

La distancia se puede calcular utilizando la siguiente formula:
Distancia(m) = {(Tiempo del pulso ECO) * (Velocidad del sonido=340m/s)}/2
El sensor US-016 es similar al HC-SR04 pero con salida de tipo analógico, otro sensor ultrasonido es el sensor US-100 con salida de tipo uart/serial.

5.2.1 ESPECIFICACIÓN Y CARACTERÍSTICAS

- ✓ Tipo: Sensor Ultrasónico
- ✓ Modelo: HC-SR04
- ✓ Dimensiones: 45 mm x 20 mm x 15 mm
- ✓ Voltaje de funcionamiento: + 5V DC
- ✓ Corriente de alimentación: 15 mA
- ✓ Rango de medición: 2 cm a 400 cm (a 4 m considerar que puede haber poca sensibilidad del dispositivo)
- ✓ Frecuencia de trabajo: 40 KHz
- ✓ Ángulo de medición efectivo: < 15º
- ✓ Ángulo de medición: 30º
- ✓ Precisión: +- 3mm
- ✓ Duración mínima del pulso de disparo (nivel TTL): 10 μs
- ✓ Duración del pulso eco de salida (nivel TTL): 100-25000 μs
- ✓ Tiempo mínimo de espera entre una medida y el inicio de otra 20 ms
- ✓ Peso: 9 g
- ✓ Pines
- ✓ Vcc: 5 DC
- ✓ Trigger: Input, disparo del ultrasonido
- ✓ Echo: Output, repetición del ultrasonido o receptor
- ✓ GND

5.2.2 FUNCIONAMIENTO DEL SENSOR

El sensor mide el tiempo entre el envío y recepción de un pulso sonoro. Usando el Trigger para al menos 10 us de señal de alto nivel, enviado automáticamente ocho pulsos a 40 kHz y detecta si hay una señal de pulso de regreso. Si la señal regresa, a través de un nivel alto, el tiempo de salida del Trigger de alta salida es el tiempo desde el envío de ultrasonidos hasta el retorno y captado en Echo.

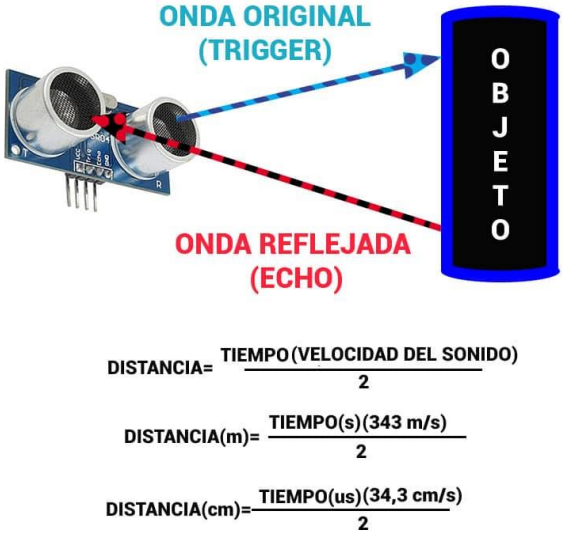


Figura 10. Funcionamiento del HC SR04 y cálculo de distancia

Fuente: https://uelectronics.com/producto/sensor-ultrasonico-hc-sr04/

5.3 MODULO WIFI ESP8266

El ESP8266 es un chip de bajo costo Wi-Fi con un stack TCP/IP completo y un microcontrolador, fabricado por Espressif, una empresa afincada en Shanghái, China1
El primer chip se hace conocido en los mercados alrededor de agosto de 2014 con el módulo ESP-01, desarrollado por la empresa AI-Thinker. Este pequeño módulo permite a otros microcontroladores conectarse a una red inalámbrica Wi-Fi y realizar conexiones simples con TCP/IP usando comandos al estilo Hayes.
El ESP8285 es como un ESP8266 pero con 1 MB de memoria flash interna, para permitir a dispositivos de un chip conexiones de Wi-Fi.

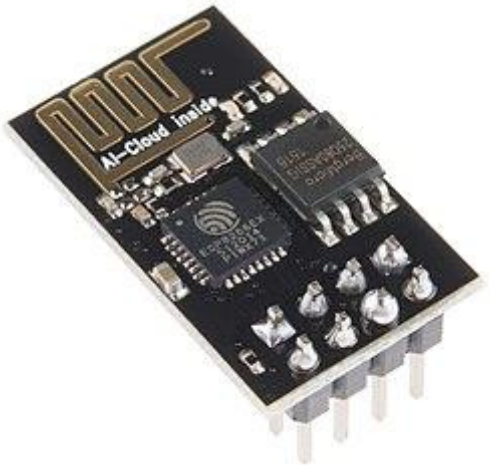


Figura 11. ESP8266

Fuente: https://es.wikipedia.org/wiki/ESP8266

5.3.1. CARACTERISTICAS

- ✓ CPU RISC de 32-bit: Tensilica Xtensa LX106 a un reloj de 80 MHza
- ✓ RAM de instrucción de 64 KB, RAM de datos de 96 KB
- ✓ Capacidad de memoria externa flash QSPI - 512 KB a 4 MB* (puede soportar hasta 16 MB)
- ✓ IEEE 802.11 b/g/n Wi-Fi
- ✓ Tiene integrados: TR switch, balun, LNA, amplificador de potencia de RF y una red de adaptación de impedancias
- ✓ Soporte de autenticación WEP y WPA/WPA2
- ✓ 16 pines GPIO (Entradas/Salidas de propósito general)
- ✓ SPI, I²C,
- ✓ Interfaz I²S con DMA (comparte pines con GPIO)
- ✓ Pines dedicados a UART, más una UART únicamente para transmisión que puede habilitarse a través del pin GPIO2
- ✓ 1 conversor ADC de 10-bit

5.4 MOTOR DC

El motor de corriente continua, denominado también motor de corriente directa, motor CC o motor DC (por las iniciales en inglés: direct current), es una máquina que convierte

energía eléctrica en energía mecánica, provocando un movimiento rotatorio, gracias a la acción de un campo magnético.

Los componentes de un motor de corriente continua se dividen en dos partes:

- Estator: parte que da soporte mecánico al aparato y contiene los polos de la máquina, que pueden ser devanados de hilo de cobre sobre un núcleo de hierro o imanes permanentes.
- Rotor: es un componente generalmente de forma cilíndrica, también devanado y con núcleo, alimentado con corriente continua a través del colector formado por delgas. Las delgas se fabrican generalmente de cobre y están en contacto alternante con las escobillas fijas.

El principal inconveniente que tienen estos motores es el mantenimiento costoso y laborioso, debido principalmente al desgaste que sufren las escobillas al entrar en contacto con las delgas. Las escobillas de los motores de baja potencia se fabrican de grafito. Por otro lado, los que requieren corrientes elevadas como los motores de arranque de los vehículos, se fabrican con una aleación de grafito y metal.

Algunas aplicaciones especiales de estos motores son: los motores lineales, servomotores, motores paso a paso o cuando ejercen tracción sobre un riel. Además, existen motores de CC sin escobillas (brushless en inglés) utilizados en el aeromodelismo por su bajo par motor y su gran velocidad.

Es posible controlar la velocidad y el par de estos motores utilizando técnicas de control de motores de corriente continua.

Figura 12.Motor DC

Fuente: https://es.wikipedia.org/wiki/Motor_de_corriente_continua

6. DESARROLLO DEL TRABAJO

6.1 HERRAMIENTAS SOFTWARE

6.1.1 COMPILADOR: PIC C COMPILER

PIC C Compiler es un inteligente y muy optimizado compilador C que contienen operadores estándar del lenguaje C y funciones incorporados en bibliotecas que son específicas a los registros de PIC, proporcionando a los desarrolladores una herramienta poderosa para el acceso al hardware las funciones del dispositivo desde el nivel de lenguaje C.

El compilador CCS contiene más de 307 funciones integradas que simplifiquen el acceso al hardware, mientras que la producción eficiente y altamente optimizado código. Se incluyen funciones de hardware del dispositivo de características tales como:

* Temporizadores y módulos PWM
* Convertidores A / D
* de datos on-chip EEPROM
* LCD controladores
* Memoria externa buses

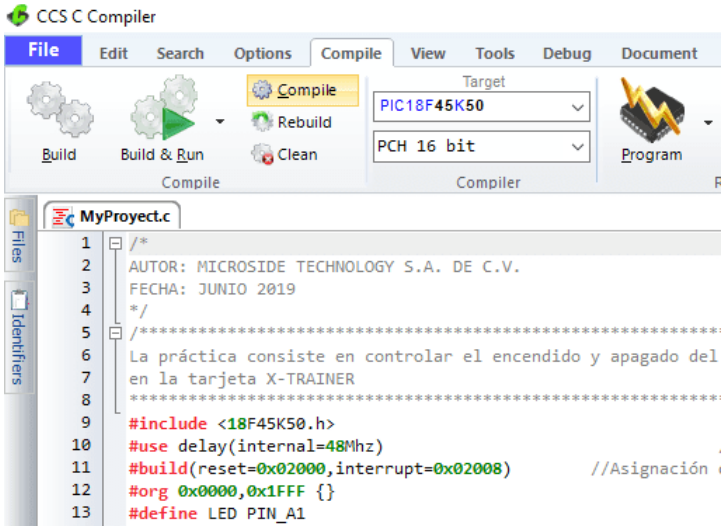


Figura 13. COMPILADOR PIC C

Fuente: https://microside.com/docs/ccs/manual-de-pic-c-compiler/

6.1.2 SOFTWARE WINPIC800

El software WinPic800 (Se lo puede bajar de www.winpic800.com), soporta los Pic nuevos, por ejemplo el Pic 18F2550, es freeware y lo podemos usar con este Programador de Pic y Memorias, (Solamente con los Pic).

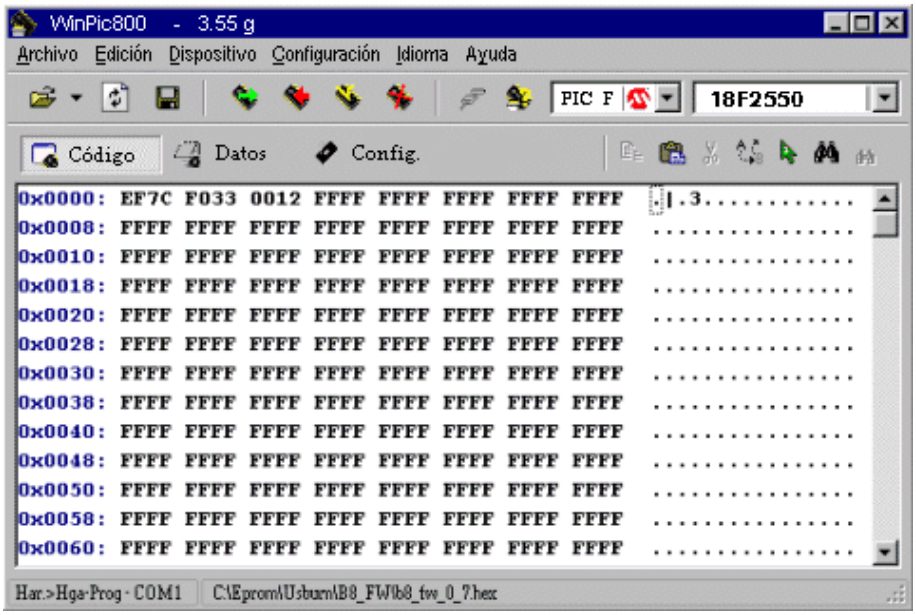


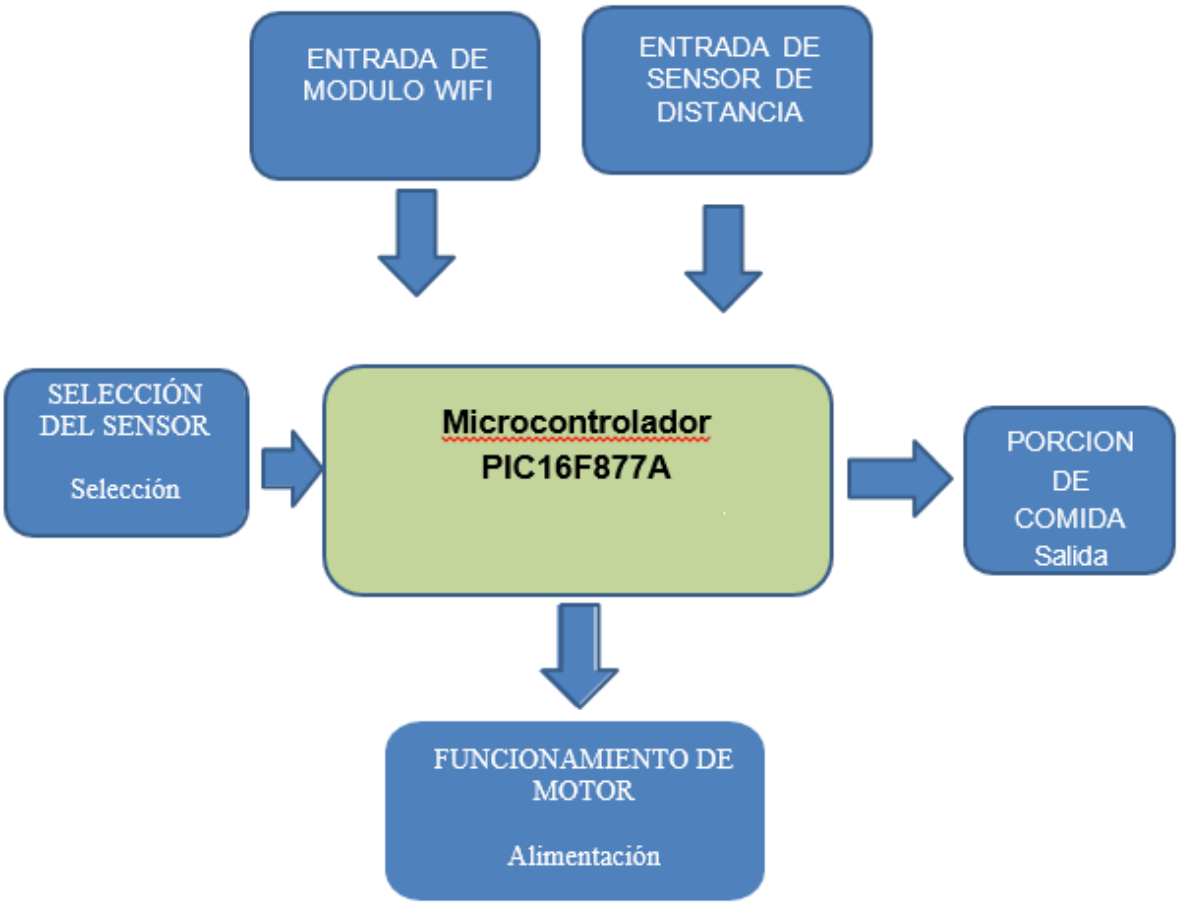
Figura 14. Grabación en PIC

Fuente: https://unicrom.com/software-winpic800-programador-de-pics-y-memorias/

6.1.2.1 GRABAR PIC CON WINPIC800

- Seleccionar el dispositivo a programar, elegimos Pic F, luego el PIC 16F877A por ejemplo.
- En el Programador seleccionar Pic, con el Sw 1.
- Abrir el archivo que contiene los datos a programar en el Pic. El programa trabaja con ficheros .HEX. En el menú Archivo seleccionamos Abrir archivo, en el cuadro de diálogo que nos aparece seleccionamos el fichero que deseamos grabar en el Pic. Pulsamos el botón Config para ajustar el tipo de oscilador y los bits de configuración.
- Para programar el dispositivo seleccionamos Programar todo del menú Dispositivo. Con ello comenzará la grabación.

6.2 DIAGRAMA DE BLOQUES



6.3 DESARROLLO DE SOFTWARE Y HARDWARE

Se realizó el desarrollo del código fuente en función a los requerimientos del proyecto, se tomó en cuenta lo siguiente:

- Porción que debe consumir al día mascota
- Distancia entre la mascota y el sensor
- Conexión vía Wifi para controlar la alimentación
- Funcionamiento del Servomotor para que se sirva la comida en el plato
- Regulador de voltaje 3.3 para el ES8266
- Colocación en parte externa del dispensador
- Alimentación mediante comida balanceada de tamaño intermedio.
- Dispensación de agua a la vez que el alimento balanceado

A continuación, se puede observar el diagrama de conexiones de los sensores, así

también el código que se realizó y se grabó en el PIC para el funcionamiento correcto del dispensador.

Figura 15. Prototipo final del proyecto

Fuente: Propia

Figura 15. Primer prototipo del proyecto

Fuente: Propia

CODIGO FUENTE EN LENGUAJE C

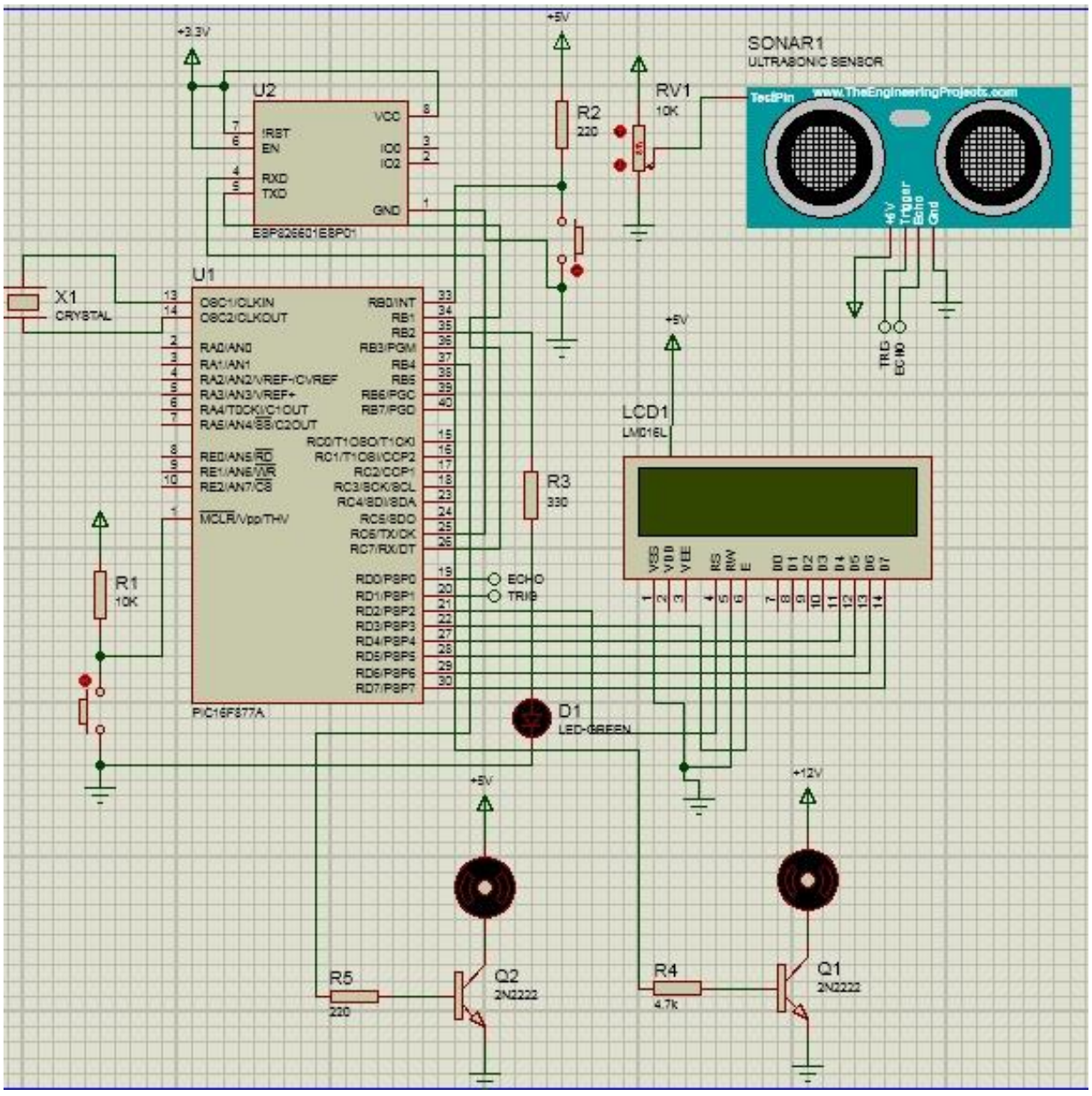
```
#include <16f877a.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP,NOPUT,BROWNOUT
#use delay(clock=20M)
#use standard_io(D)
#use rs232(baud=115200, xmit=PIN_C6,rcv=PIN_C7,parity=N, bits=8)

int32 fosc = 20000000;
#define P_ECHO PIN_D0
#define P_TRIG PIN_D1

#define LCD_DB4  PIN_D4
#define LCD_DB5  PIN_D5
#define LCD_DB6  PIN_D6
#define LCD_DB7  PIN_D7
#define LCD_RS   PIN_D2
#define LCD_E    PIN_D3
#include <LCD_16X2.c>
#include <HCSR04.c>
#define motor1 PIN_B1
#define motor2 PIN_B4
#define led_mode PIN_B2
volatile int1 __mode=1;
volatile int1 __espera=0;
float distancia = 0;// Variable para almacenar la distancia
const float dist_min = 30.0;//distancia de activacion
const int16 tmp_espera = 10; //tiempo de espera en segundos
const int16 tmp_activo=5; //tiempo de activacion de motor en segundos
char c='0';
#INT_EXT
void __interrup(){
  DISABLE_INTERRUPTS(INT_EXT);
```

```
  delay_ms(500);
  __mode=!__mode;
  __espera=0;
  c='0',
  ENABLE_INTERRUPTS(INT_EXT);
}


void espera(){
  int16 cont=0;
  while(__espera){
    delay_ms(1000);
    cont++;
    if(cont>=tmp_espera){
      __espera=0;
    }
  }
}
void main(){
  HCSR04_init();                          // Inicializa el sensor ultrasonico
  lcd_init();// Inicializa la LCD
  set_tris_b(0b11100001);
  output_low(motor1);
  output_low(motor2);
  output_low(led_mode);
  ENABLE_INTERRUPTS(GLOBAL);
  ENABLE_INTERRUPTS(INT_EXT);
  while(true){
    if(__mode){
      output_low(led_mode);
      distancia = HCSR04_get_distance();          // Obtiene la distancia y la
almacena
      lcd_clear();
      lcd_gotoxy(1,1);
```

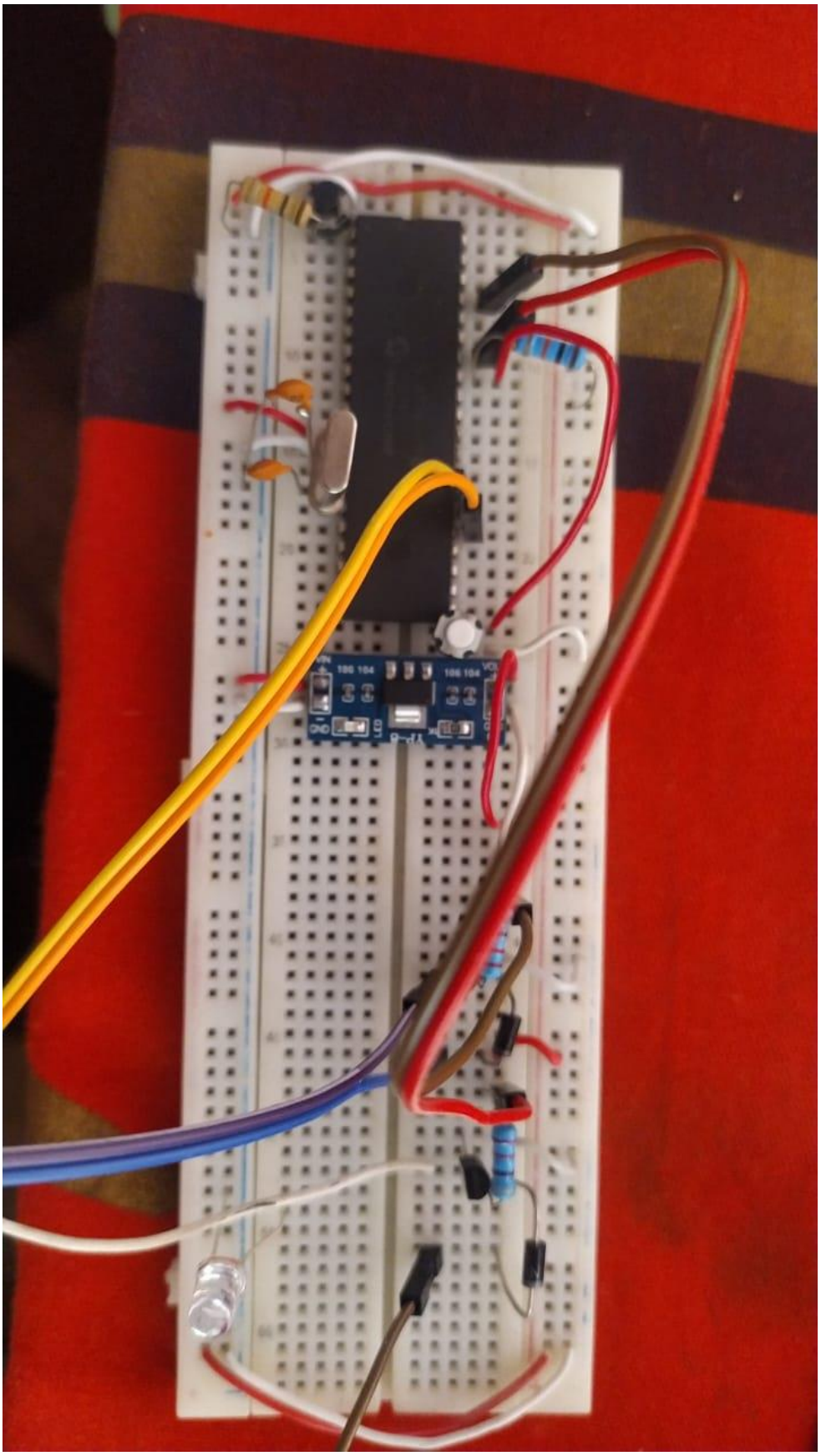
```
   lcd_putc("Sensor HC-SR04");
   lcd_gotoxy(1,2);
   printf(lcd_putc,"Dist: %0.2f cm", distancia);      // Imprime la distancia
   if(distancia<=dist_min){
      __espera = 1;
      output_high(motor1);
      output_high(motor2);
      delay_ms(tmp_activo*1000);
      output_low(motor1);
      output_low(motor2);
      espera();

   }

}else{

   putc('A');
   output_high(led_mode);
   lcd_clear();
   lcd_gotoxy(1,1);
   lcd_putc("Control por");
   lcd_gotoxy(1,2);
   lcd_putc("Internet");
   delay_ms(8000);
   if(kbhit()>0){
      c = getc();
    }
    if(c=='1'){
      __espera = 1;
      output_high(motor1);
      output_high(motor2);
      delay_ms(tmp_activo*1000);
      output_low(motor1);
      output_low(motor2);
```

```
          espera();
        }
        if(c=='0'){
          output_low(motor1);
          output_low(motor2);
        }

    }

    delay_ms(150);
  }
}
```

DIAGRAMA DEL CIRCUITO EN PROTEUS



CIRCUITO IMPLEMENTADO EN PROTOBOARD



7. CONCLUSIONES

- ✓Para realizar la dispensación de comida de la mascota se tuvo que realizar un tipo tornillo sin fin en el primer prototipo.
- ✓Para poder iniciar el funcionamiento se utilizó el PIC16F877A, siendo este el cerebro del circuito.
- ✓Se utilizó el compilador PIC C Compiler, para realizar el código fuente ygenerar el archivo. hex para el microcontrolador.
- ✓Se programó el microcontrolador PIC16F877A utilizando el WinPic800.
- ✓Para realizar la interfaz de usuario se utilizó aplicativo del ESP8266.
- ✓Se utilizó un pulsador para definir qué entrada se deseaba recibir para alimentar a la mascota.
- ✓Se regulo el voltaje para el uso de módulo Wifi ESP8266.
- ✓Se realizó el módulo de dispensación de agua en base a un motor DC.
- ✓Se utilizó un servidor web gratuito para el control remoto del wifi.

8. BIBLIOGRAFÍA

- Hector Morlote (2013) Estudio de la estructura interna del PIC16F877A. Recuperado de: https://cifpn1hectorm.wordpress.com/2013/04/10/estudiode-la- estructura-interna-del-pic-16f877/
- Curso Básico de PIC 16F877A: Autores: Raúl Peralta Meza y Carlos Quiñones Quispe
- Microcontroladores PIC 16F877A Diseño práctico de aplicaciones Autores: José M.Angulo Usategui, Susana Romero Yesa e Ignacio Angulo Martínez.
- PIC Microcontrollers - Programming in C Autores: Mikroc
- Cómo programar en lenguaje c los microcontroladores pic16f88, 16f628ay 16f877a Autor: JUAN RICARDO PENAGOS PLAZAS
- Basic para Microcontroldores PIC Autor: Crhistian Bodington Esteva

PAGINAS WEB CONSULTADAS

- https://aprendiendoarduino.wordpress.com/2017/09/03/microcontroladores-arduino-a-fondo/http://www.coolers\Extractores Eólicos - Caracteristicas, Rendimientos.htm
- https://www.diarioelectronicohoy.com/blog/imagenes/2010/12/juego-de-instrucciones-PIC
- https://cifpn1hectorm.wordpress.com
- *https://es.wikipedia.or*
- https://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf

9. ANEXOS

ESP8266EX

Datasheet

Version 6.9

Espressif Systems

2. Pin Definitions

Figure 2-1 shows the pin layout for 32-pin QFN package.

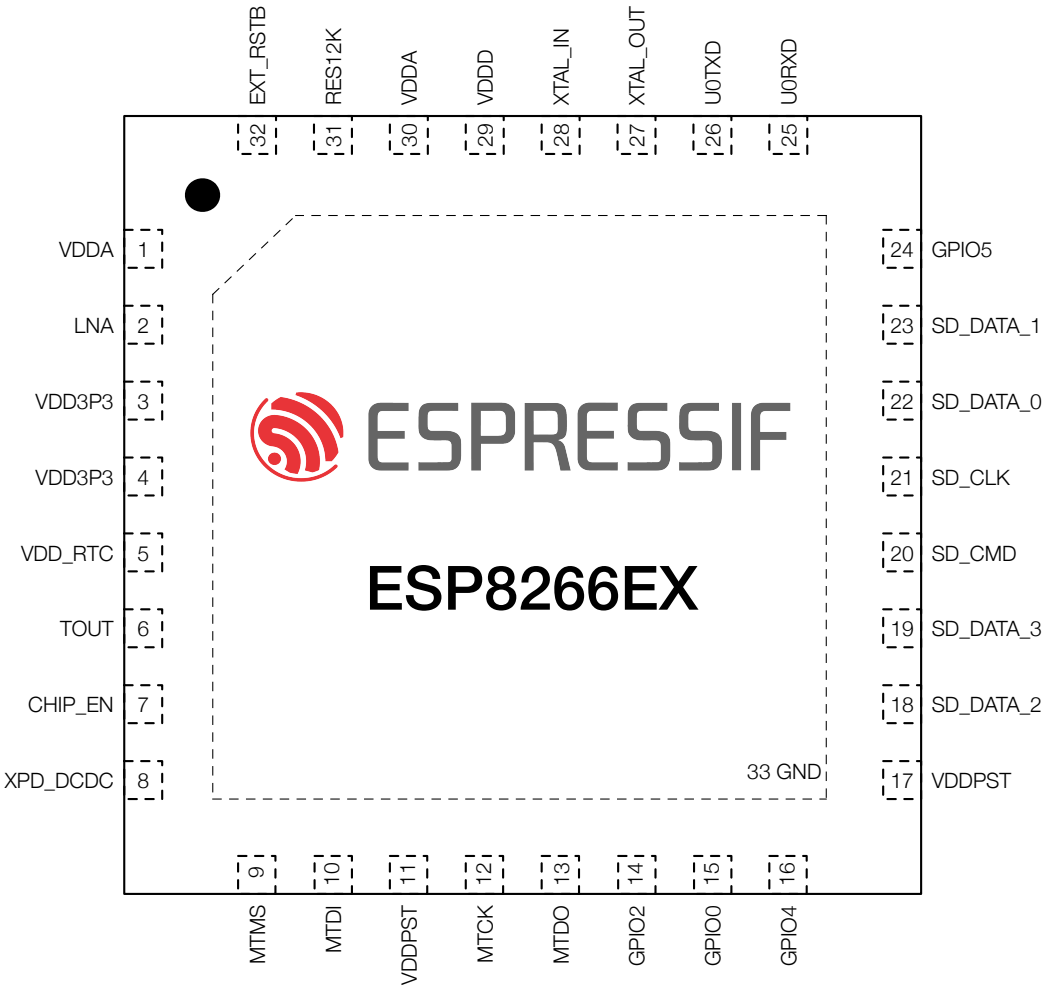


Figure 2-1. Pin Layout (Top View)

Table 2-1 lists the definitions and functions of each pin.

Table 2-1. ESP8266EX Pin Definitions

Pin	Name	Type	Function
1	VDDA	P	Analog Power 2.5 V ~ 3.6 V
2	LNA	I/O	RF antenna interface Chip output impedance = 39 + j6 Ω. It is suggested to retain the π-type matching network to match the antenna.
3	VDD3P3	P	Amplifier Power 2.5 V ~ 3.6 V

Pin	Name	Type	Function
4	VDD3P3	P	Amplifier Power 2.5 V ~ 3.6 V
5	VDD_RTC	P	NC (1.1 V)
6	TOUT	I	ADC pin. It can be used to test the power-supply voltage of VDD3P3 (Pin3 and Pin4) and the input power voltage of TOUT (Pin 6). However, these two functions cannot be used simultaneously.
7	CHIP_EN	I	Chip Enable High: On, chip works properly Low: Off, small current consumed
8	XPD_DCDC	I/O	Deep-sleep wakeup (need to be connected to EXT_RSTB); GPIO16
9	MTMS	I/O	GPIO 14; HSPI_CLK
10	MTDI	I/O	GPIO 12; HSPI_MISO
11	VDDPST	P	Digital/IO Power Supply (1.8 V ~ 3.6 V)
12	MTCK	I/O	GPIO 13; HSPI_MOSI; UART0_CTS
13	MTDO	I/O	GPIO 15; HSPI_CS; UART0_RTS
14	GPIO2	I/O	UART TX during flash programming; GPIO2
15	GPIO0	I/O	GPIO0; SPI_CS2
16	GPIO4	I/O	GPIO4
17	VDDPST	P	Digital/IO Power Supply (1.8 V ~ 3.6 V)
18	SDIO_DATA_2	I/O	Connect to SD_D2 (Series R: 20 Ω); SPIHD; HSPIHD; GPIO9
19	SDIO_DATA_3	I/O	Connect to SD_D3 (Series R: 200 Ω); SPIWP; HSPIWP; GPIO10
20	SDIO_CMD	I/O	Connect to SD_CMD (Series R: 200 Ω); SPI_CS0; GPIO11
21	SDIO_CLK	I/O	Connect to SD_CLK (Series R: 200 Ω); SPI_CLK; GPIO6
22	SDIO_DATA_0	I/O	Connect to SD_D0 (Series R: 200 Ω); SPI_MISO; GPIO7
23	SDIO_DATA_1	I/O	Connect to SD_D1 (Series R: 200 Ω); SPI_MOSI; GPIO8
24	GPIO5	I/O	GPIO5
25	U0RXD	I/O	UART Rx during flash programming; GPIO3
26	U0TXD	I/O	UART TX during flash programming; GPIO1; SPI_CS1
27	XTAL_OUT	I/O	Connect to crystal oscillator output, can be used to provide BT clock input
28	XTAL_IN	I/O	Connect to crystal oscillator input
29	VDDD	P	Analog Power 2.5 V ~ 3.6 V
30	VDDA	P	Analog Power 2.5 V ~ 3.6 V

Pin	Name	Type	Function
31	RES12K	I	Serial connection with a 12 kΩ resistor and connect to the ground
32	EXT_RSTB	I	External reset signal (Low voltage level: active)

📖 ***Note:***

1. *GPIO2, GPIO0, and MTDO are used to select booting mode and the SDIO mode;*
2. *U0TXD should not be pulled externally to a low logic level during the powering-up.*

PIC16F87XA
Data Sheet

28/40/44-Pin Enhanced Flash Microcontrollers

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

DNV Certification, Inc. USA — ANSI-RAB QMS ACCREDITED — DNV MSC The Netherlands Accredited by the RvA — DNV — ISO 9001 / QS-9000 REGISTERED FIRM

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999 and Mountain View, California in March 2002. The Company’s quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, non-volatile memory and analog products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001 certified.

PIC16F87XA

28/40/44-Pin Enhanced Flash Microcontrollers

Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- 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 I^2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) – 8 bits wide with external $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ 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 (V_{REF}) 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

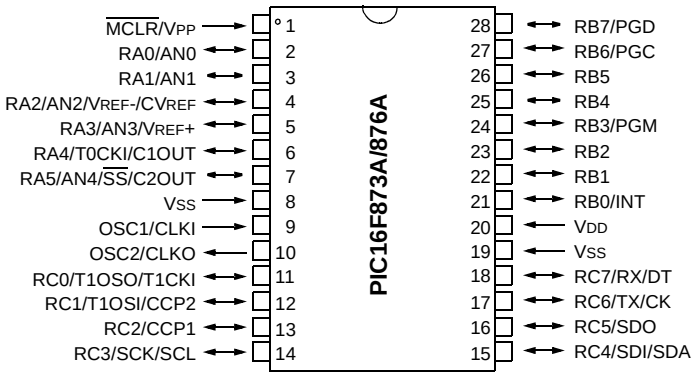
CMOS Technology:

- 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

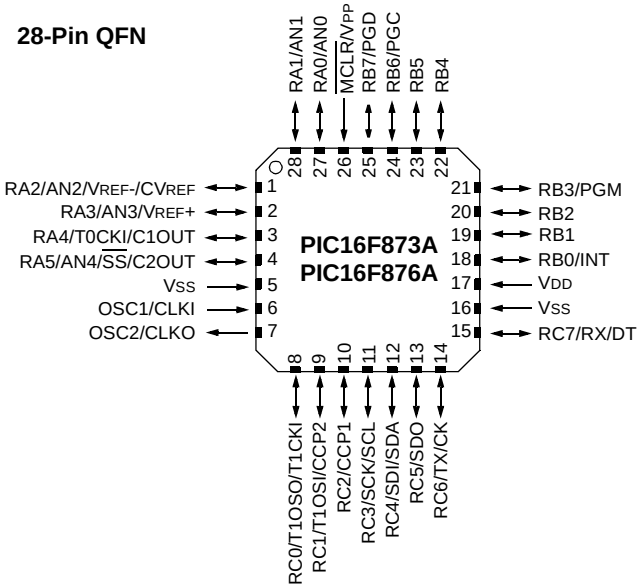
Device	Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
	Bytes	# Single Word Instructions						SPI	Master I^2C			
PIC16F873A	7.2K	4096	192	128	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F874A	7.2K	4096	192	128	33	8	2	Yes	Yes	Yes	2/1	2
PIC16F876A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

Pin Diagrams

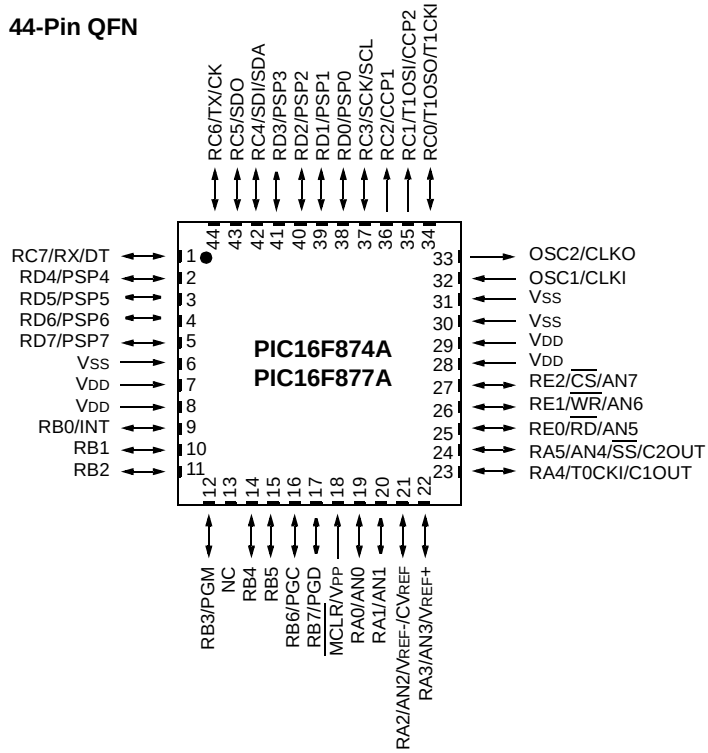
28-Pin PDIP, SOIC, SSOP



28-Pin QFN



44-Pin QFN



Pin Diagrams (Continued)

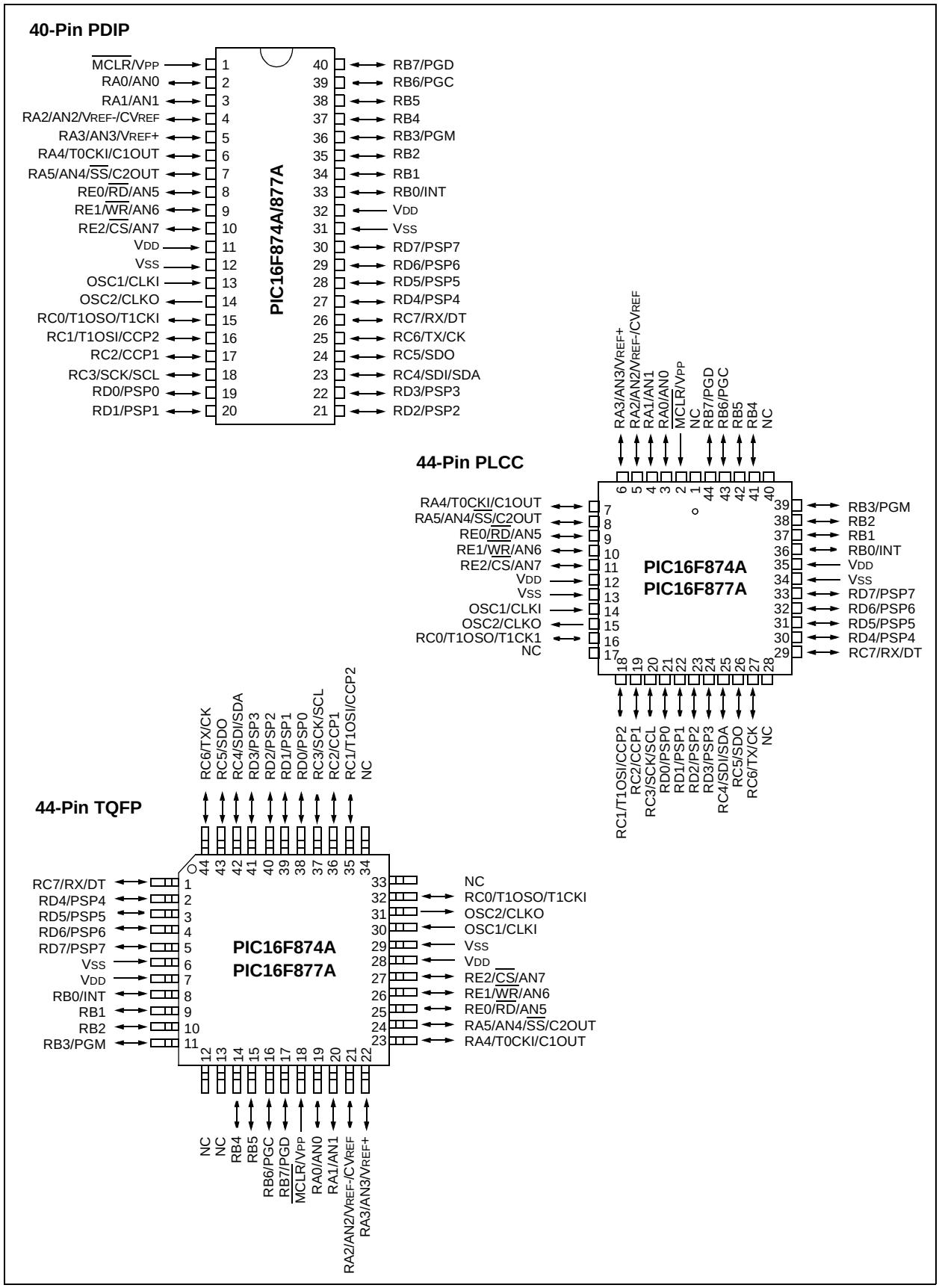


Table of Contents

TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@mail.microchip.com** or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000A is version A of document DS30000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)
- The Microchip Corporate Literature Center; U.S. FAX: (480) 792-7277

When contacting a sales office or the literature center, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Customer Notification System

Register on our Web site at **www.microchip.com/cn** to receive the most current information on all of our products.

1.0 DEVICE OVERVIEW

This document contains device specific information about the following devices:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

PIC16F873A/876A devices are available only in 28-pin packages, while PIC16F874A/877A devices are available in 40-pin and 44-pin packages. All devices in the PIC16F87XA family share common architecture with the following differences:

- The PIC16F873A and PIC16F874A have one-half of the total on-chip memory of the PIC16F876A and PIC16F877A
- The 28-pin devices have three I/O ports, while the 40/44-pin devices have five
- The 28-pin devices have fourteen interrupts, while the 40/44-pin devices have fifteen
- The 28-pin devices have five A/D input channels, while the 40/44-pin devices have eight
- The Parallel Slave Port is implemented only on the 40/44-pin devices

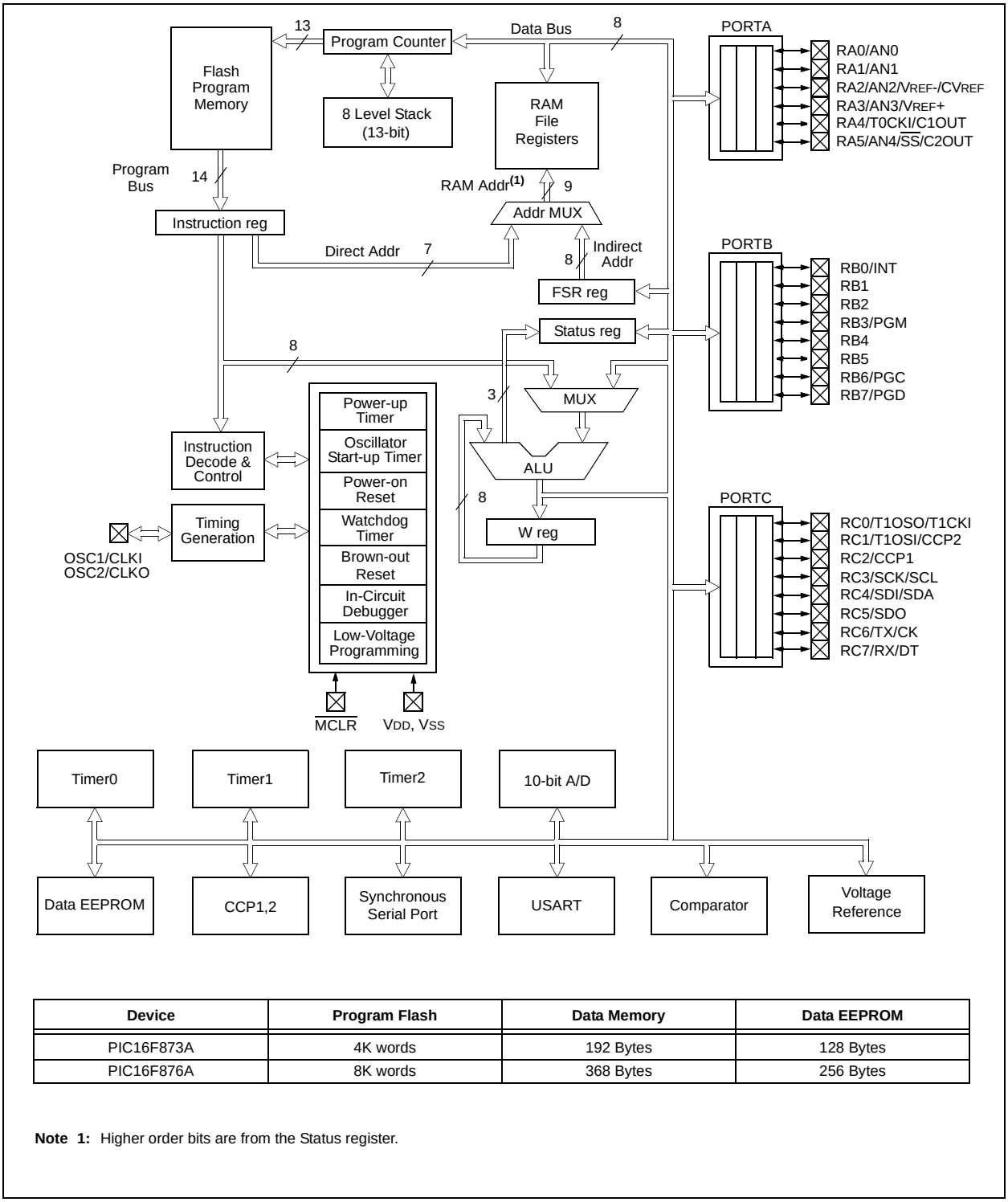
The available features are summarized in Table 1-1. Block diagrams of the PIC16F873A/876A and PIC16F874A/877A devices are provided in Figure 1-1 and Figure 1-2, respectively. The pinouts for these device families are listed in Table 1-2 and Table 1-3.

Additional information may be found in the PICmicro® Mid-Range Reference Manual (DS33023), which may be obtained from your local Microchip Sales Representative or downloaded from the Microchip web site. The Reference Manual should be considered a complementary document to this data sheet and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

TABLE 1-1: PIC16F87XA DEVICE FEATURES

Key Features	PIC16F873A	PIC16F874A	PIC16F876A	PIC16F877A
Operating Frequency	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz
Resets (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
Flash Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory (bytes)	128	128	256	256
Interrupts	14	15	14	15
I/O Ports	Ports A, B, C	Ports A, B, C, D, E	Ports A, B, C	Ports A, B, C, D, E
Timers	3	3	3	3
Capture/Compare/PWM modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Analog Comparators	2	2	2	2
Instruction Set	35 Instructions	35 Instructions	35 Instructions	35 Instructions
Packages	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN

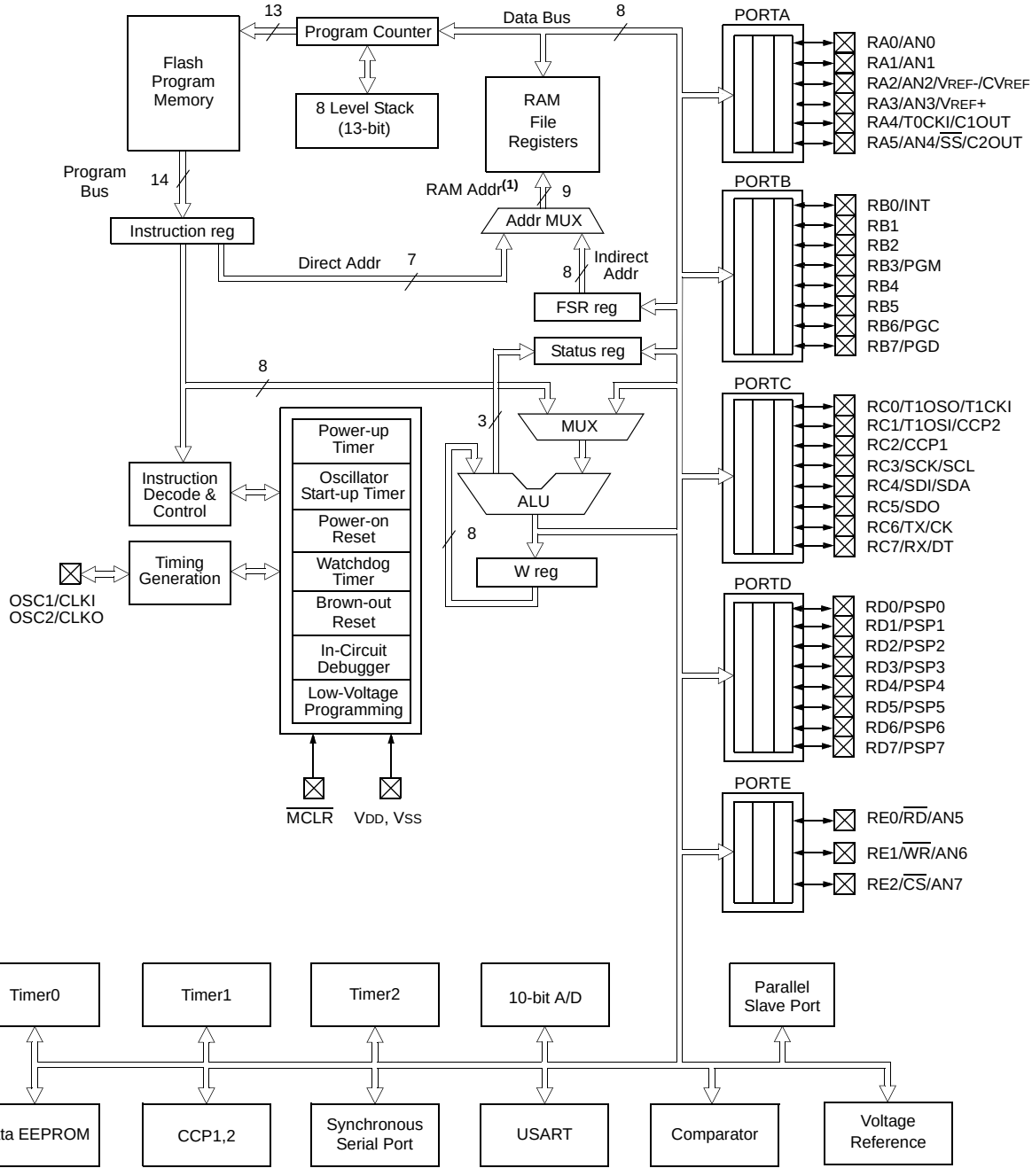
FIGURE 1-1: PIC16F873A/876A BLOCK DIAGRAM



Device	Program Flash	Data Memory	Data EEPROM
PIC16F873A	4K words	192 Bytes	128 Bytes
PIC16F876A	8K words	368 Bytes	256 Bytes

Note 1: Higher order bits are from the Status register.

FIGURE 1-2: PIC16F874A/877A BLOCK DIAGRAM



Device	Program Flash	Data Memory	Data EEPROM
PIC16F874A	4K words	192 Bytes	128 Bytes
PIC16F877A	8K words	368 Bytes	256 Bytes

Note 1: Higher order bits are from the Status register.

TABLE 1-2: PIC16F873A/876A PINOUT DESCRIPTION

Pin Name	PDIP, SOIC, SSOP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKI	9	6		ST/CMOS(3)	Oscillator crystal or external clock input.
OSC1			I		Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; otherwise CMOS.
CLKI			I		External clock source input. Always associated with pin function OSC1 (see OSC1/CLKI, OSC2/CLKO pins).
OSC2/CLKO	10	7		—	Oscillator crystal or clock output.
OSC2			O		Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO			O		In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP	1	26		ST	Master Clear (input) or programming voltage (output).
$\overline{MCLR}$			I		Master Clear (Reset) input. This pin is an active low Reset to the device.
VPP			P		Programming voltage input.
					PORTA is a bidirectional I/O port.
RA0/AN0	2	27		TTL	
RA0			I/O		Digital I/O.
AN0			I		Analog input 0.
RA1/AN1	3	28		TTL	
RA1			I/O		Digital I/O.
AN1			I		Analog input 1.
RA2/AN2/VREF-/	4	1		TTL	
CVREF			I/O		Digital I/O.
RA2			I		Analog input 2.
AN2			I		A/D reference voltage (Low) input.
VREF-			O		Comparator VREF output.
CVREF					
RA3/AN3/VREF+	5	2		TTL	
RA3			I/O		Digital I/O.
AN3			I		Analog input 3.
VREF+			I		A/D reference voltage (High) input.
RA4/T0CKI/C1OUT	6	3		ST	
RA4			I/O		Digital I/O – Open-drain when configured as output.
T0CKI			I		Timer0 external clock input.
C1OUT			O		Comparator 1 output.
RA5/AN4/$\overline{SS}$/C2OUT	7	4		TTL	
RA5			I/O		Digital I/O.
AN4			I		Analog input 4.
$\overline{SS}$			I		SPI slave select input.
C2OUT			O		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.

TABLE 1-2: PIC16F873A/876A PINOUT DESCRIPTION (CONTINUED)

Pin Name	PDIP, SOIC, SSOP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
					PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/INT	21	18		TTL/ST(1)	
RB0			I/O		Digital I/O.
INT			I		External interrupt.
RB1	22	19	I/O	TTL	Digital I/O.
RB2	23	20	I/O	TTL	Digital I/O.
RB3/PGM	24	21		TTL	
RB3			I/O		Digital I/O.
PGM			I		Low-voltage (single-supply) ICSP programming enable pin.
RB4	25	22	I/O	TTL	Digital I/O.
RB5	26	23	I/O	TTL	Digital I/O.
RB6/PGC	27	24		TTL/ST(2)	
RB6			I/O		Digital I/O.
PGC			I		In-circuit debugger and ICSP programming clock.
RB7/PGD	28	25		TTL/ST(2)	
RB7			I/O		Digital I/O.
PGD			I/O		In-circuit debugger and ICSP programming data.
					PORTC is a bidirectional I/O port.
RC0/T1OSO/T1CKI	11	8		ST	
RC0			I/O		Digital I/O.
T1OSO			O		Timer1 oscillator output.
T1CKI			I		Timer1 external clock input.
RC1/T1OSI/CCP2	12	9		ST	
RC1			I/O		Digital I/O.
T1OSI			I		Timer1 oscillator input.
CCP2			I/O		Capture2 input, Compare2 output, PWM2 output.
RC2/CCP1	13	10		ST	
RC2			I/O		Digital I/O.
CCP1			I/O		Capture1 input, Compare1 output, PWM1 output.
RC3/SCK/SCL	14	11		ST	
RC3			I/O		Digital I/O.
SCK			I/O		Synchronous serial clock input/output for SPI mode.
SCL			I/O		Synchronous serial clock input/output for I^2C mode.
RC4/SDI/SDA	15	12		ST	
RC4			I/O		Digital I/O.
SDI			I		SPI data in.
SDA			I/O		I^2C data I/O.
RC5/SDO	16	13		ST	
RC5			I/O		Digital I/O.
SDO			O		SPI data out.
RC6/TX/CK	17	14		ST	
RC6			I/O		Digital I/O.
TX			O		USART asynchronous transmit.
CK			I/O		USART1 synchronous clock.
RC7/RX/DT	18	15		ST	
RC7			I/O		Digital I/O.
RX			I		USART asynchronous receive.
DT			I/O		USART synchronous data.
VSS	8, 19	5, 6	P	—	Ground reference for logic and I/O pins.
VDD	20	17	P	—	Positive supply for logic and I/O pins.

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.

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

TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION (CONTINUED)

Pin Name	PDIP Pin#	PLCC Pin#	TQFP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
							PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	9		TTL/ST(1)	
RB0					I/O		Digital I/O.
INT					I		External interrupt.
RB1	34	37	9	10	I/O	TTL	Digital I/O.
RB2	35	38	10	11	I/O	TTL	Digital I/O.
RB3/PGM	36	39	11	12		TTL	
RB3					I/O		Digital I/O.
PGM					I		Low-voltage ICSP programming enable pin.
RB4	37	41	14	14	I/O	TTL	Digital I/O.
RB5	38	42	15	15	I/O	TTL	Digital I/O.
RB6/PGC	39	43	16	16		TTL/ST(2)	
RB6					I/O		Digital I/O.
PGC					I		In-circuit debugger and ICSP programming clock.
RB7/PGD	40	44	17	17		TTL/ST(2)	
RB7					I/O		Digital I/O.
PGD					I/O		In-circuit debugger and ICSP programming data.

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.

TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION (CONTINUED)

Pin Name	PDIP Pin#	PLCC Pin#	TQFP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
							PORTC is a bidirectional I/O port.
RC0/T1OSO/T1CKI	15	16	32	34		ST	
RC0					I/O		Digital I/O.
T1OSO					O		Timer1 oscillator output.
T1CKI					I		Timer1 external clock input.
RC1/T1OSI/CCP2	16	18	35	35		ST	
RC1					I/O		Digital I/O.
T1OSI					I		Timer1 oscillator input.
CCP2					I/O		Capture2 input, Compare2 output, PWM2 output.
RC2/CCP1	17	19	36	36		ST	
RC2					I/O		Digital I/O.
CCP1					I/O		Capture1 input, Compare1 output, PWM1 output.
RC3/SCK/SCL	18	20	37	37		ST	
RC3					I/O		Digital I/O.
SCK					I/O		Synchronous serial clock input/output for SPI mode.
SCL					I/O		Synchronous serial clock input/output for I^2C mode.
RC4/SDI/SDA	23	25	42	42		ST	
RC4					I/O		Digital I/O.
SDI					I		SPI data in.
SDA					I/O		I^2C data I/O.
RC5/SDO	24	26	43	43		ST	
RC5					I/O		Digital I/O.
SDO					O		SPI data out.
RC6/TX/CK	25	27	44	44		ST	
RC6					I/O		Digital I/O.
TX					O		USART asynchronous transmit.
CK					I/O		USART1 synchronous clock.
RC7/RX/DT	26	29	1	1		ST	
RC7					I/O		Digital I/O.
RX					I		USART asynchronous receive.
DT					I/O		USART synchronous data.

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.

TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION (CONTINUED)

Pin Name	PDIP Pin#	PLCC Pin#	TQFP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
							PORTD is a bidirectional I/O port or Parallel Slave Port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38	38		ST/TTL(3)	
RD0					I/O		Digital I/O.
PSP0					I/O		Parallel Slave Port data.
RD1/PSP1	20	22	39	39		ST/TTL(3)	
RD1					I/O		Digital I/O.
PSP1					I/O		Parallel Slave Port data.
RD2/PSP2	21	23	40	40		ST/TTL(3)	
RD2					I/O		Digital I/O.
PSP2					I/O		Parallel Slave Port data.
RD3/PSP3	22	24	41	41		ST/TTL(3)	
RD3					I/O		Digital I/O.
PSP3					I/O		Parallel Slave Port data.
RD4/PSP4	27	30	2	2		ST/TTL(3)	
RD4					I/O		Digital I/O.
PSP4					I/O		Parallel Slave Port data.
RD5/PSP5	28	31	3	3		ST/TTL(3)	
RD5					I/O		Digital I/O.
PSP5					I/O		Parallel Slave Port data.
RD6/PSP6	29	32	4	4		ST/TTL(3)	
RD6					I/O		Digital I/O.
PSP6					I/O		Parallel Slave Port data.
RD7/PSP7	30	33	5	5		ST/TTL(3)	
RD7					I/O		Digital I/O.
PSP7					I/O		Parallel Slave Port data.
							PORTE is a bidirectional I/O port.
RE0/$\overline{RD}$/AN5	8	9	25	25		ST/TTL(3)	
RE0					I/O		Digital I/O.
$\overline{RD}$					I		Read control for Parallel Slave Port.
AN5					I		Analog input 5.
RE1/$\overline{WR}$/AN6	9	10	26	26		ST/TTL(3)	
RE1					I/O		Digital I/O.
$\overline{WR}$					I		Write control for Parallel Slave Port.
AN6					I		Analog input 6.
RE2/$\overline{CS}$/AN7	10	11	27	27		ST/TTL(3)	
RE2					I/O		Digital I/O.
$\overline{CS}$					I		Chip select control for Parallel Slave Port.
AN7					I		Analog input 7.
VSS	12, 31	13, 34	6, 29	6, 30, 31	P	—	Ground reference for logic and I/O pins.
VDD	11, 32	12, 35	7, 28	7, 8, 28, 29	P	—	Positive supply for logic and I/O pins.
NC	—	1, 17, 28, 40	12,13, 33, 34	13	—	—	These pins are not internally connected. These pins should be left unconnected.

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.

NOTES:

2.0 MEMORY ORGANIZATION

There are three memory blocks in each of the PIC16F87XA devices. The program memory and data memory have separate buses so that concurrent access can occur and is detailed in this section. The EEPROM data memory block is detailed in **Section 3.0 "Data EEPROM and Flash Program Memory"**.

Additional information on device memory may be found in the PICmicro® Mid-Range MCU Family Reference Manual (DS33023).

2.1 Program Memory Organization

The PIC16F87XA devices have a 13-bit program counter capable of addressing an 8K word x 14 bit program memory space. The PIC16F876A/877A devices have 8K words x 14 bits of Flash program memory, while PIC16F873A/874A devices have 4K words x 14 bits. Accessing a location above the physically implemented address will cause a wraparound.

The Reset vector is at 0000h and the interrupt vector is at 0004h.

FIGURE 2-1: PIC16F876A/877A PROGRAM MEMORY MAP AND STACK

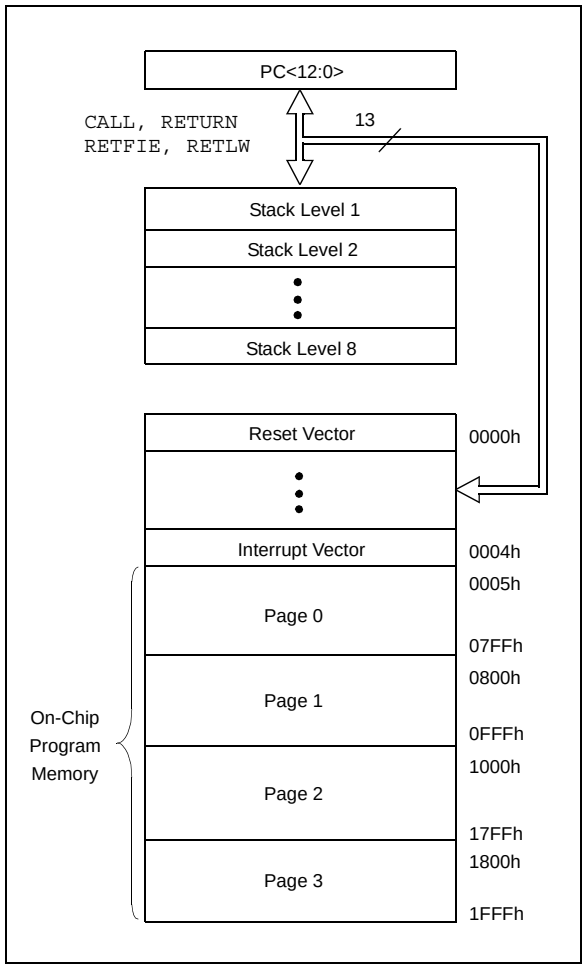
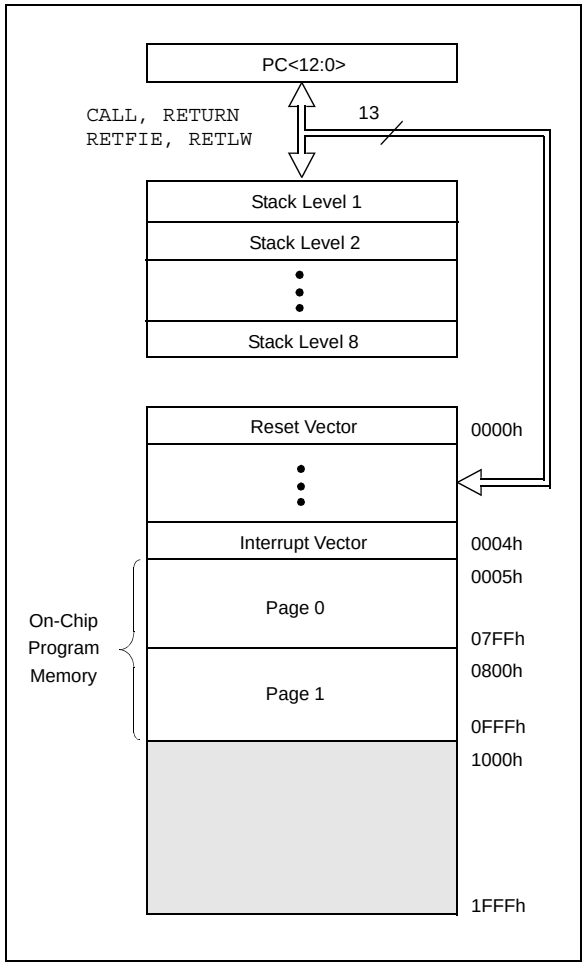


FIGURE 2-2: PIC16F873A/874A PROGRAM MEMORY MAP AND STACK



2.2 Data Memory Organization

The data memory is partitioned into multiple banks which contain the General Purpose Registers and the Special Function Registers. Bits RP1 (Status<6>) and RP0 (Status<5>) are the bank select bits.

RP1:RP0	Bank
00	0
01	1
10	2
11	3

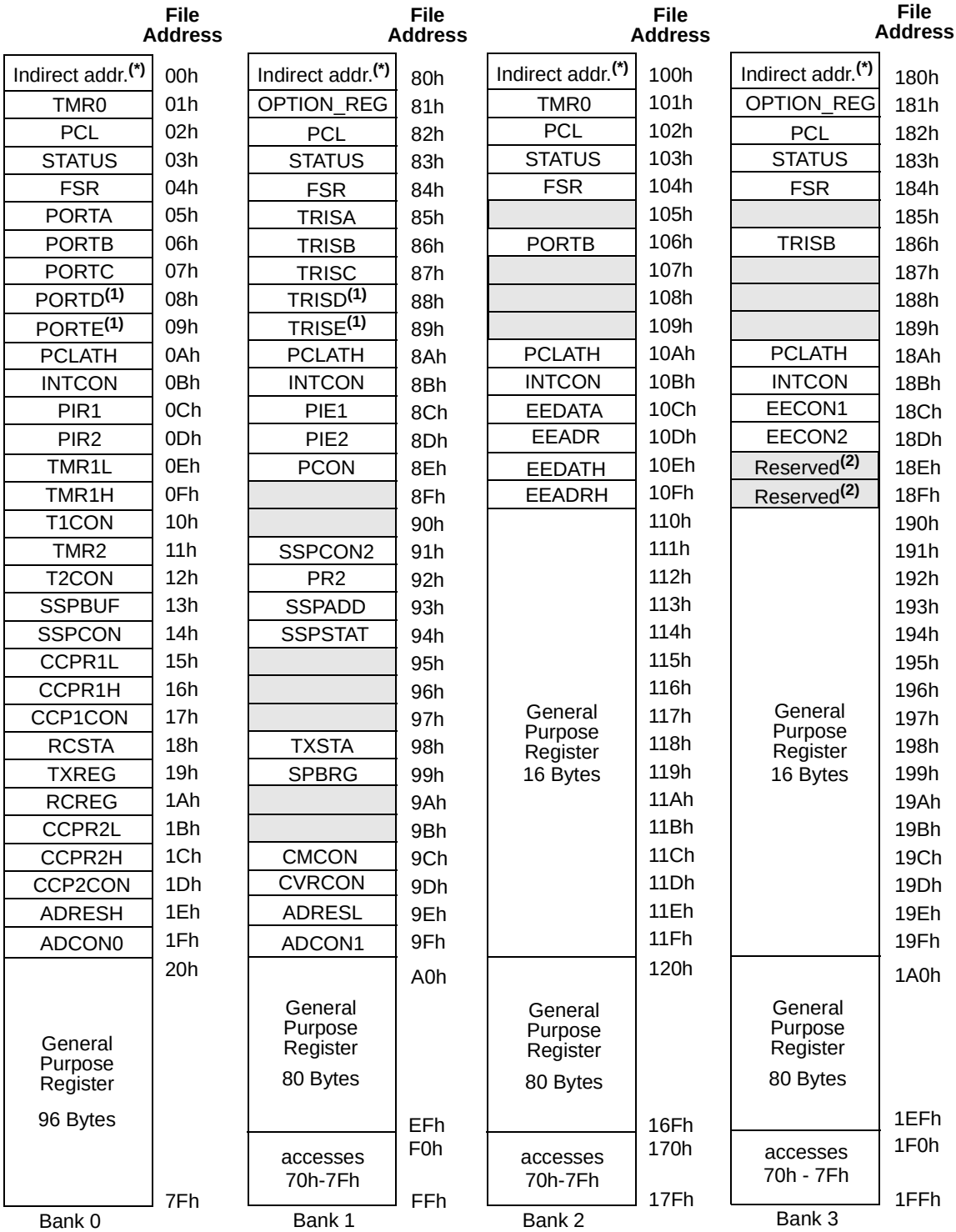
Each bank extends up to 7Fh (128 bytes). The lower locations of each bank are reserved for the Special Function Registers. Above the Special Function Registers are General Purpose Registers, implemented as static RAM. All implemented banks contain Special Function Registers. Some frequently used Special Function Registers from one bank may be mirrored in another bank for code reduction and quicker access.

Note:	The EEPROM data memory description can be found in **Section 3.0 "Data EEPROM and Flash Program Memory"** of this data sheet.

2.2.1 GENERAL PURPOSE REGISTER FILE

The register file can be accessed either directly, or indirectly, through the File Select Register (FSR).

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



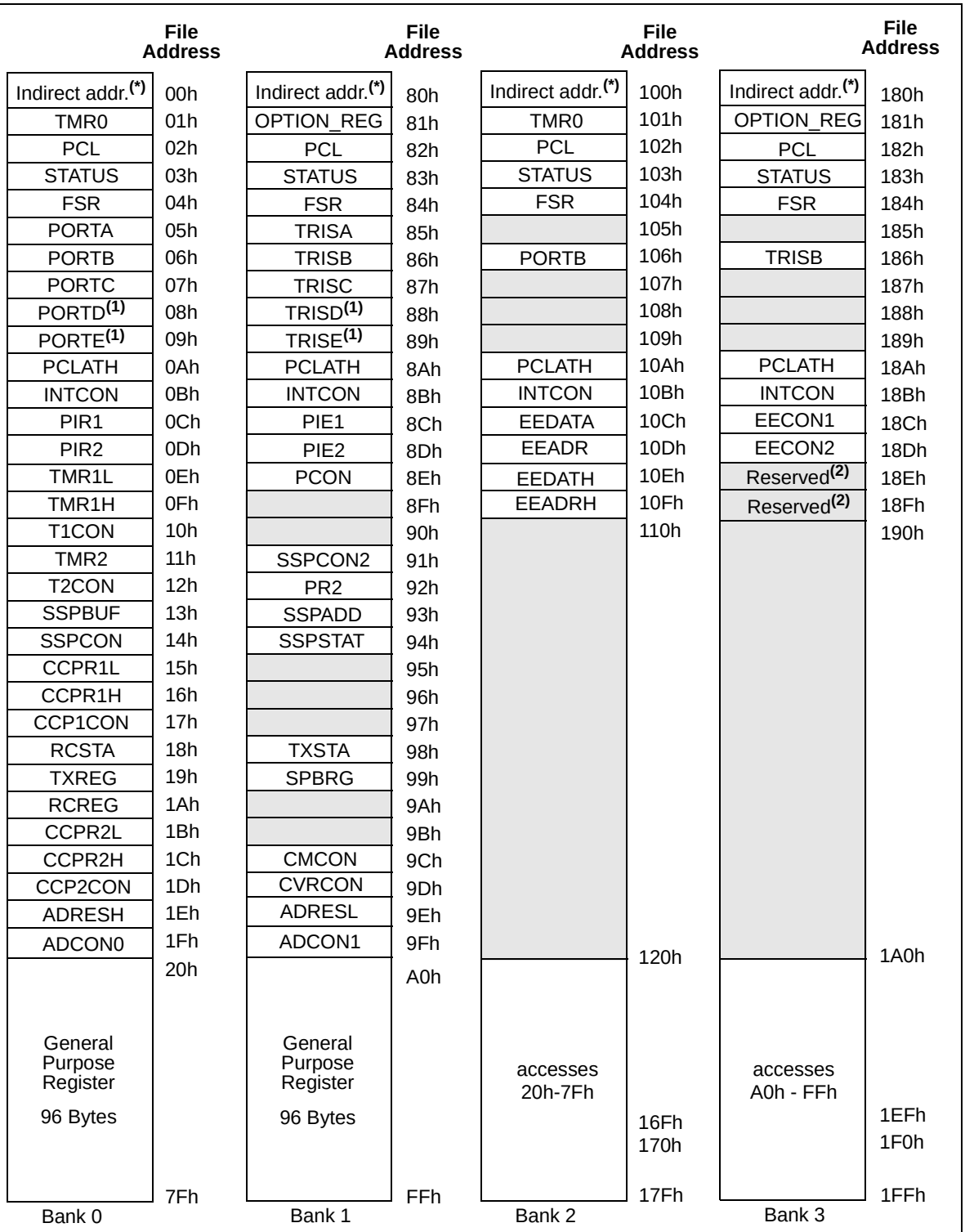
Unimplemented data memory locations, read as '0'.

* Not a physical register.

Note 1: These registers are not implemented on the PIC16F876A.

2: These registers are reserved; maintain these registers clear.

FIGURE 2-4: PIC16F873A/874A REGISTER FILE MAP



Unimplemented data memory locations, read as '0'.

* Not a physical register.

Note 1: These registers are not implemented on the PIC16F873A.

2: These registers are reserved; maintain these registers clear.

2.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 2-1.

The Special Function Registers can be classified into two sets: core (CPU) and peripheral. Those registers associated with the core functions are described in detail in this section. Those related to the operation of the peripheral features are described in detail in the peripheral features section.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 0											
00h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
01h	TMR0	Timer0 Module Register								xxxx xxxx	55, 150
02h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	30, 150
03h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22, 150
04h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	43, 150
06h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	45, 150
07h	PORTC	PORTC Data Latch when written: PORTC pins when read								xxxx xxxx	47, 150
08h(4)	PORTD	PORTD Data Latch when written: PORTD pins when read								xxxx xxxx	48, 150
09h(4)	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	49, 150
0Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
0Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
0Ch	PIR1	PSPIF(3)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	26, 150
0Dh	PIR2	—	CMIF	—	EEIF	BCLIF	—	—	CCP2IF	-0-0 0--0	28, 150
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	60, 150
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	60, 150
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	--00 0000	57, 150
11h	TMR2	Timer2 Module Register								0000 0000	62, 150
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	61, 150
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	79, 150
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	82, 82, 150
15h	CCPR1L	Capture/Compare/PWM Register 1 (LSB)								xxxx xxxx	63, 150
16h	CCPR1H	Capture/Compare/PWM Register 1 (MSB)								xxxx xxxx	63, 150
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	64, 150
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	112, 150
19h	TXREG	USART Transmit Data Register								0000 0000	118, 150
1Ah	RCREG	USART Receive Data Register								0000 0000	118, 150
1Bh	CCPR2L	Capture/Compare/PWM Register 2 (LSB)								xxxx xxxx	63, 150
1Ch	CCPR2H	Capture/Compare/PWM Register 2 (MSB)								xxxx xxxx	63, 150
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	64, 150
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	133, 150
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	—	ADON	0000 00-0	127, 150

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8>, whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873A/876A devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD and TRISE are not implemented on PIC16F873A/876A devices, read as '0'.

5: Bit 4 of EEADRH implemented only on the PIC16F876A/877A devices.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 1											
80h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
81h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	23, 150
82h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	30, 150
83h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22, 150
84h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	43, 150
86h	TRISB	PORTB Data Direction Register								1111 1111	45, 150
87h	TRISC	PORTC Data Direction Register								1111 1111	47, 150
88h(4)	TRISD	PORTD Data Direction Register								1111 1111	48, 151
89h(4)	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	50, 151
8Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
8Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
8Ch	PIE1	PSPIE(2)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	25, 151
8Dh	PIE2	—	CMIE	—	EEIE	BCLIE	—	—	CCP2IE	-0-0 0--0	27, 151
8Eh	PCON	—	—	—	—	—	—	$\overline{POR}$	$\overline{BOR}$	---- --qq	29, 151
8Fh	—	Unimplemented								—	—
90h	—	Unimplemented								—	—
91h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	83, 151
92h	PR2	Timer2 Period Register								1111 1111	62, 151
93h	SSPADD	Synchronous Serial Port (I^2C mode) Address Register								0000 0000	79, 151
94h	SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	79, 151
95h	—	Unimplemented								—	—
96h	—	Unimplemented								—	—
97h	—	Unimplemented								—	—
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	111, 151
99h	SPBRG	Baud Rate Generator Register								0000 0000	113, 151
9Ah	—	Unimplemented								—	—
9Bh	—	Unimplemented								—	—
9Ch	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111	135, 151
9Dh	CVRCON	CVREN	CVROE	CVRR	—	CVR3	CVR2	CVR1	CVR0	000- 0000	141, 151
9Eh	ADRESL	A/D Result Register Low Byte								xxxx xxxx	133, 151
9Fh	ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	128, 151

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8>, whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873A/876A devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD and TRISE are not implemented on PIC16F873A/876A devices, read as '0'.

5: Bit 4 of EEADRH implemented only on the PIC16F876A/877A devices.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 2											
100h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
101h	TMR0	Timer0 Module Register								xxxx xxxx	55, 150
102h(3)	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	30, 150
103h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22, 150
104h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
105h	—	Unimplemented								—	—
106h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	45, 150
107h	—	Unimplemented								—	—
108h	—	Unimplemented								—	—
109h	—	Unimplemented								—	—
10Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
10Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
10Ch	EEDATA	EEPROM Data Register Low Byte								xxxx xxxx	39, 151
10Dh	EEADR	EEPROM Address Register Low Byte								xxxx xxxx	39, 151
10Eh	EEDATH	—	—	EEPROM Data Register High Byte						--xx xxxx	39, 151
10Fh	EEADRH	—	—	—	—(5)	EEPROM Address Register High Byte				---- xxxx	39, 151
Bank 3											
180h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
181h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	23, 150
182h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	30, 150
183h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22, 150
184h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
185h	—	Unimplemented								—	—
186h	TRISB	PORTB Data Direction Register								1111 1111	45, 150
187h	—	Unimplemented								—	—
188h	—	Unimplemented								—	—
189h	—	Unimplemented								—	—
18Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
18Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
18Ch	EECON1	EEPGD	—	—	—	WRERR	WREN	WR	RD	x--- x000	34, 151
18Dh	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	39, 151
18Eh	—	Reserved; maintain clear								0000 0000	—
18Fh	—	Reserved; maintain clear								0000 0000	—

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8>, whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873A/876A devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD and TRISE are not implemented on PIC16F873A/876A devices, read as '0'.

5: Bit 4 of EEADRH implemented only on the PIC16F876A/877A devices.

2.2.2.1 Status Register

The Status register contains the arithmetic status of the ALU, the Reset status and the bank select bits for data memory.

The Status register can be the destination for any instruction, as with any other register. If the Status register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the $\overline{TO}$ and $\overline{PD}$ bits are not writable, therefore, the result of an instruction with the Status register as destination may be different than intended.

For example, `CLRF STATUS`, will clear the upper three bits and set the Z bit. This leaves the Status register as `000u u1uu` (where `u` = unchanged).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions are used to alter the Status register because these instructions do not affect the Z, C or DC bits from the Status register. For other instructions not affecting any status bits, see **Section 15.0 "Instruction Set Summary"**.

Note:	The C and DC bits operate as a $\overline{borrow}$ and digit $\overline{borrow}$ bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h, 103h, 183h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7 **IRP:** Register Bank Select bit (used for indirect addressing)
`1` = Bank 2, 3 (100h-1FFh)
`0` = Bank 0, 1 (00h-FFh)

bit 6-5 **RP1:RP0**: Register Bank Select bits (used for direct addressing)
`11` = Bank 3 (180h-1FFh)
`10` = Bank 2 (100h-17Fh)
`01` = Bank 1 (80h-FFh)
`00` = Bank 0 (00h-7Fh)
Each bank is 128 bytes.

bit 4 **$\overline{TO}$**: Time-out bit
`1` = After power-up, `CLRWDT` instruction or `SLEEP` instruction
`0` = A WDT time-out occurred

bit 3 **$\overline{PD}$**: Power-down bit
`1` = After power-up or by the `CLRWDT` instruction
`0` = By execution of the `SLEEP` instruction

bit 2 **Z**: Zero bit
`1` = The result of an arithmetic or logic operation is zero
`0` = The result of an arithmetic or logic operation is not zero

bit 1 **DC**: Digit carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)
(for $\overline{borrow}$, the polarity is reversed)
`1` = A carry-out from the 4th low order bit of the result occurred
`0` = No carry-out from the 4th low order bit of the result

bit 0 **C**: Carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)
`1` = A carry-out from the Most Significant bit of the result occurred
`0` = No carry-out from the Most Significant bit of the result occurred

Note: For $\overline{borrow}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high, or low order bit of the source register.

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

2.2.2.2 OPTION_REG Register

The OPTION_REG Register is a readable and writable register, which contains various control bits to configure the TMR0 prescaler/WDT postscaler (single assignable register known also as the prescaler), the external INT interrupt, TMR0 and the weak pull-ups on PORTB.

Note: To achieve a 1:1 prescaler assignment for the TMR0 register, assign the prescaler to the Watchdog Timer.

REGISTER 2-2: OPTION_REG REGISTER (ADDRESS 81h, 181h)

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 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/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

Note: When using Low-Voltage ICSP Programming (LVP) and the pull-ups on PORTB are enabled, bit 3 in the TRISB register must be cleared to disable the pull-up on RB3 and ensure the proper operation of the device

2.2.2.3 INTCON Register

The INTCON register is a readable and writable register, which contains various enable and flag bits for the TMR0 register overflow, RB port change and external RB0/INT pin interrupts.

Note:	Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

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

2.2.2.4 PIE1 Register

The PIE1 register contains the individual enable bits for the peripheral interrupts.

Note:	Bit PEIE (INTCON<6>) must be set to enable any peripheral interrupt.

REGISTER 2-4: PIE1 REGISTER (ADDRESS 8Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit(1)
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

Note 1: PSPIE is reserved on PIC16F873A/876A devices; always maintain this bit clear.

bit 6 **ADIE**: A/D Converter Interrupt Enable bit
1 = Enables the A/D converter interrupt
0 = Disables the A/D converter interrupt

bit 5 **RCIE**: USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4 **TXIE**: USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

bit 3 **SSPIE**: Synchronous Serial Port Interrupt Enable bit
1 = Enables the SSP interrupt
0 = Disables the SSP interrupt

bit 2 **CCP1IE**: CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE**: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

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

2.2.2.5 PIR1 Register

The PIR1 register contains the individual flag bits for the peripheral interrupts.

Note:	Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt bits are clear prior to enabling an interrupt.

REGISTER 2-5: PIR1 REGISTER (ADDRESS 0Ch)

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 7 **PSPIF:** Parallel Slave Port Read/Write Interrupt Flag bit(1)
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred

Note 1: PSPIF is reserved on PIC16F873A/876A devices; always maintain this bit clear.

bit 6 **ADIF**: A/D Converter Interrupt Flag bit
1 = An A/D conversion completed
0 = The A/D conversion is not complete

bit 5 **RCIF**: USART Receive Interrupt Flag bit
1 = The USART receive buffer is full
0 = The USART receive buffer is empty

bit 4 **TXIF**: USART Transmit Interrupt Flag bit
1 = The USART transmit buffer is empty
0 = The USART transmit buffer is full

bit 3 **SSPIF**: Synchronous Serial Port (SSP) Interrupt Flag bit
1 = The SSP interrupt condition has occurred and must be cleared in software before returning from the Interrupt Service Routine. The conditions that will set this bit are:
- SPI – A transmission/reception has taken place.
- I^2C Slave – A transmission/reception has taken place.
- I^2C Master
  - A transmission/reception has taken place.
  - The initiated Start condition was completed by the SSP module.
  - The initiated Stop condition was completed by the SSP module.
  - The initiated Restart condition was completed by the SSP module.
  - The initiated Acknowledge condition was completed by the SSP module.
  - A Start condition occurred while the SSP module was Idle (multi-master system).
  - A Stop condition occurred while the SSP module was Idle (multi-master system).

0 = No SSP interrupt condition has occurred

bit 2 **CCP1IF**: CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode.

bit 1 **TMR2IF**: TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF**: TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

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

2.2.2.6 PIE2 Register

The PIE2 register contains the individual enable bits for the CCP2 peripheral interrupt, the SSP bus collision interrupt, EEPROM write operation interrupt and the comparator interrupt.

Note:	Bit PEIE (INTCON<6>) must be set to enable any peripheral interrupt.

REGISTER 2-6: PIE2 REGISTER (ADDRESS 8Dh)

U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	CMIE	—	EEIE	BCLIE	—	—	CCP2IE
bit 7							bit 0

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

bit 6 **CMIE**: Comparator Interrupt Enable bit
1 = Enables the comparator interrupt
0 = Disable the comparator interrupt

bit 5 **Unimplemented:** Read as '0'

bit 4 **EEIE**: EEPROM Write Operation Interrupt Enable bit
1 = Enable EEPROM write interrupt
0 = Disable EEPROM write interrupt

bit 3 **BCLIE**: Bus Collision Interrupt Enable bit
1 = Enable bus collision interrupt
0 = Disable bus collision interrupt

bit 2-1 **Unimplemented:** Read as '0'

bit 0 **CCP2IE**: CCP2 Interrupt Enable bit
1 = Enables the CCP2 interrupt
0 = Disables the CCP2 interrupt

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

2.2.2.7 PIR2 Register

The PIR2 register contains the flag bits for the CCP2 interrupt, the SSP bus collision interrupt, EEPROM write operation interrupt and the comparator interrupt.

Note:	Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 2-7: PIR2 REGISTER (ADDRESS 0Dh)

U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	CMIF	—	EEIF	BCLIF	—	—	CCP2IF
bit 7							bit 0

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

bit 6 **CMIF**: Comparator Interrupt Flag bit
1 = The comparator input has changed (must be cleared in software)
0 = The comparator input has not changed

bit 5 **Unimplemented:** Read as '0'

bit 4 **EEIF**: EEPROM Write Operation Interrupt Flag bit
1 = The write operation completed (must be cleared in software)
0 = The write operation is not complete or has not been started

bit 3 **BCLIF**: Bus Collision Interrupt Flag bit
1 = A bus collision has occurred in the SSP when configured for I^2C Master mode
0 = No bus collision has occurred

bit 2-1 **Unimplemented:** Read as '0'

bit 0 **CCP2IF**: CCP2 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused.

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

2.2.2.8 PCON Register

The Power Control (PCON) register contains flag bits to allow differentiation between a Power-on Reset (POR), a Brown-out Reset (BOR), a Watchdog Reset (WDT) and an external MCLR Reset.

Note:	$\overline{\text{BOR}}$ is unknown on Power-on Reset. It must be set by the user and checked on subsequent Resets to see if BOR is clear, indicating a brown-out has occurred. The BOR status bit is a “don’t care” and is not predictable if the brown-out circuit is disabled (by clearing the BODEN bit in the configuration word).

REGISTER 2-8: PCON REGISTER (ADDRESS 8Eh)

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-1
—	—	—	—	—	—	$\overline{\text{POR}}$	$\overline{\text{BOR}}$
bit 7							bit 0

bit 7-2 **Unimplemented:** Read as ‘0’

bit 1 **$\overline{\text{POR}}$**: Power-on Reset Status bit
1 = No Power-on Reset occurred
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

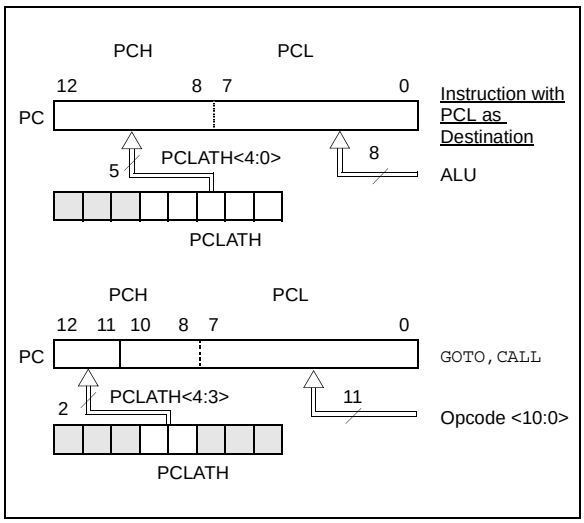
bit 0 **$\overline{\text{BOR}}$**: Brown-out Reset Status bit
1 = No Brown-out Reset occurred
0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)

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

2.3 PCL and PCLATH

The Program Counter (PC) is 13 bits wide. The low byte comes from the PCL register which is a readable and writable register. The upper bits (PC<12:8>) are not readable, but are indirectly writable through the PCLATH register. On any Reset, the upper bits of the PC will be cleared. Figure 2-5 shows the two situations for the loading of the PC. The upper example in the figure shows how the PC is loaded on a write to PCL (PCLATH<4:0> → PCH). The lower example in the figure shows how the PC is loaded during a CALL or GOTO instruction (PCLATH<4:3> → PCH).

FIGURE 2-5: LOADING OF PC IN DIFFERENT SITUATIONS



2.3.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL). When doing a table read using a computed GOTO method, care should be exercised if the table location crosses a PCL memory boundary (each 256-byte block). Refer to the application note, *AN556, "Implementing a Table Read"* (DS00556).

2.3.2 STACK

The PIC16F87XA family has an 8-level deep x 13-bit wide hardware stack. The stack space is not part of either program or data space and the stack pointer is not readable or writable. The PC is PUSHed onto the stack when a CALL instruction is executed, or an interrupt causes a branch. The stack is POP'ed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not affected by a PUSH or POP operation.

The stack operates as a circular buffer. This means that after the stack has been PUSHed eight times, the ninth push overwrites the value that was stored from the first push. The tenth push overwrites the second push (and so on).

Note 1: There are no status bits to indicate stack overflow or stack underflow conditions.

2: There are no instructions/mnemonics called PUSH or POP. These are actions that occur from the execution of the CALL, RETURN, RETLW and RETFIE instructions or the vectoring to an interrupt address.

2.4 Program Memory Paging

All PIC16F87XA devices are capable of addressing a continuous 8K word block of program memory. The CALL and GOTO instructions provide only 11 bits of address to allow branching within any 2K program memory page. When doing a CALL or GOTO instruction, the upper 2 bits of the address are provided by PCLATH<4:3>. When doing a CALL or GOTO instruction, the user must ensure that the page select bits are programmed so that the desired program memory page is addressed. If a return from a CALL instruction (or interrupt) is executed, the entire 13-bit PC is popped off the stack. Therefore, manipulation of the PCLATH<4:3> bits is not required for the RETURN instructions (which POPs the address from the stack).

Note: The contents of the PCLATH register are unchanged after a RETURN or RETFIE instruction is executed. The user must rewrite the contents of the PCLATH register for any subsequent subroutine calls or GOTO instructions.

Example 2-1 shows the calling of a subroutine in page 1 of the program memory. This example assumes that PCLATH is saved and restored by the Interrupt Service Routine (if interrupts are used).

EXAMPLE 2-1: CALL OF A SUBROUTINE IN PAGE 1 FROM PAGE 0

```
        ORG 0x500
        BCF PCLATH,4
        BSF PCLATH,3    ;Select page 1
                        ;(800h-FFFh)
        CALL SUB1_P1    ;Call subroutine in
        :               ;page 1 (800h-FFFh)
        :
        ORG 0x900       ;page 1 (800h-FFFh)
SUB1_P1
        :               ;called subroutine
                        ;page 1 (800h-FFFh)
        :
        RETURN          ;return to
                        ;Call subroutine
                        ;in page 0
                        ;(000h-7FFh)
```

2.5 Indirect Addressing, INDF and FSR Registers

The INDF register is not a physical register. Addressing the INDF register will cause indirect addressing.

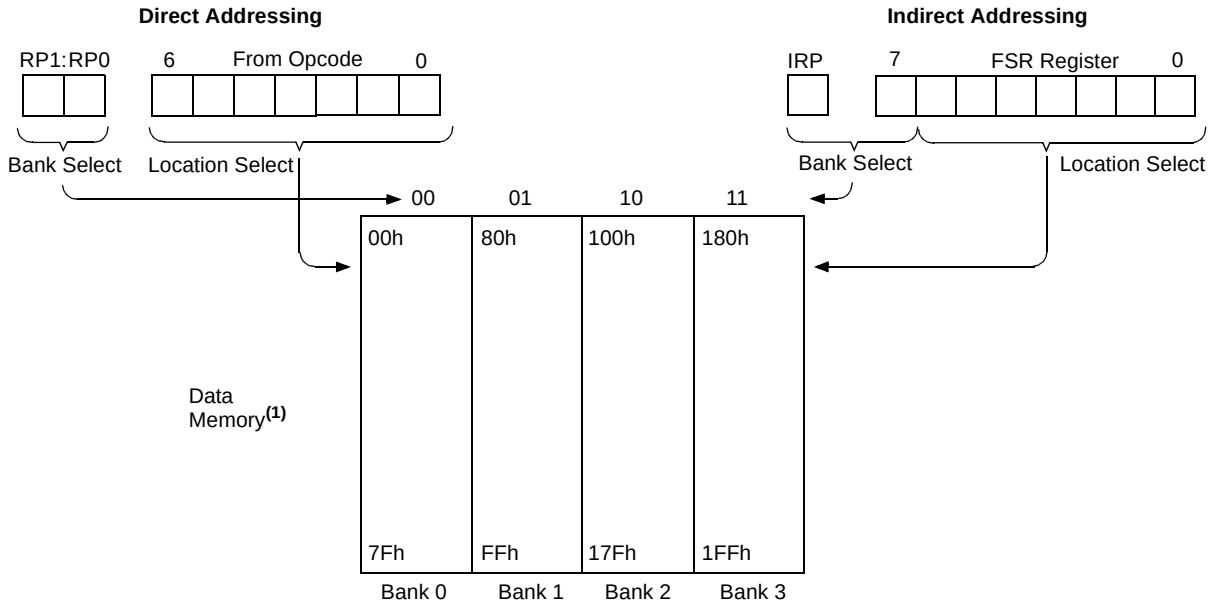
Indirect addressing is possible by using the INDF register. Any instruction using the INDF register actually accesses the register pointed to by the File Select Register, FSR. Reading the INDF register itself, indirectly (FSR = 0) will read 00h. Writing to the INDF register indirectly results in a no operation (although status bits may be affected). An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (Status<7>) as shown in Figure 2-6.

A simple program to clear RAM locations 20h-2Fh using indirect addressing is shown in Example 2-2.

EXAMPLE 2-2: INDIRECT ADDRESSING

```
        MOVLW  0x20    ;initialize pointer
        MOVWF  FSR     ;to RAM
NEXT    CLRF   INDF    ;clear INDF register
        INCF   FSR,F   ;inc pointer
        BTFSS  FSR,4   ;all done?
        GOTO   NEXT    ;no clear next
CONTINUE
        :              ;yes continue
```

FIGURE 2-6: DIRECT/INDIRECT ADDRESSING



Note 1: For register file map detail, see Figure 2-3.

Ultrasonic Ranging Module HC - SR04

Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
Test distance = (high level time×velocity of sound (340M/S) / 2,

Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

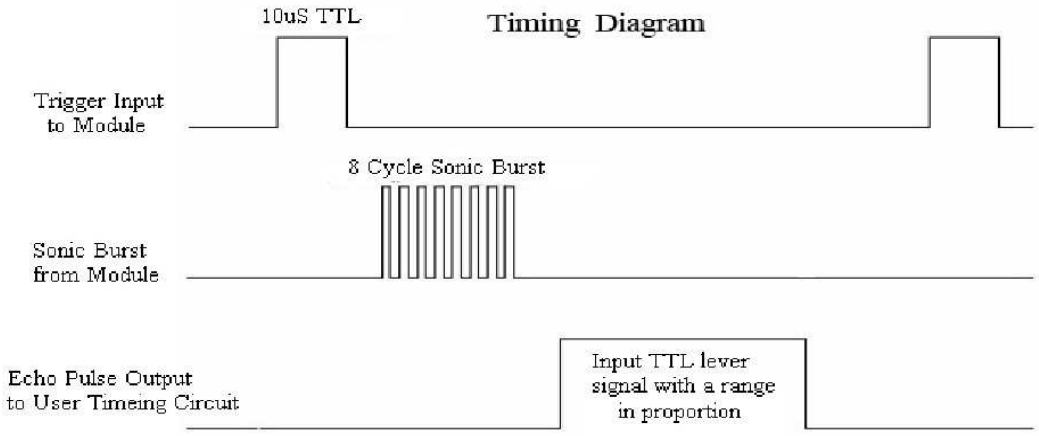
Electric Parameter

Working Voltage	**DC 5 V**
Working Current	**15mA**
Working Frequency	**40Hz**
Max Range	**4m**
Min Range	**2cm**
MeasuringAngle	**15 degree**
Trigger Input Signal	**10uS TTL pulse**
Echo Output Signal	**Input TTL lever signal and the range in proportion**
Dimension	**45*20*15mm**

Vcc Trig Echo GND

Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



Attention:

- The module is not suggested to connect directly to electric, if connected electric, the GND terminal should be connected the module first, otherwise, it will affect the normal work of the module.
- When tested objects, the range of area is not less than 0.5 square meters and the plane requests as smooth as possible, otherwise ,it will affect the results of measuring.

www.Elecfreaks.com