

se desea información detallada sobre qué hace exactamente cada orden, enseguida se hará una descripción punto por punto de qué es lo que hace cada comando, y cómo afectan a las distintas banderas de estado. Cabe aclarar que en la notación de Microchip, cuando nos referimos a una localidad de memoria específica se le identifica como “F”, así que se denominará de esa forma de ahora en adelante.

No.	Comando	Descripción	Banderas
<i>Operaciones relacionadas con bytes en registros y localidades de memoria</i>			
1	<b>ADDWF</b>	Suma el contenido de W y una localidad de memoria F	C, DC, Z
2	<b>ANDWF</b>	Operación AND entre W y F	Z
3	<b>CLRF</b>	Borra el contenido de F	Z
4	<b>CLRW</b>	Borra el contenido de W	Z
5	<b>COMF</b>	Operación complemento a F	Z
6	<b>DECF</b>	Decrementa el contenido de F	Z
7	<b>DECFSZ</b>	Decrementa F, salta una posición si llega a cero	
8	<b>INCF</b>	Incrementa el contenido de F	Z
9	<b>INCFSZ</b>	Incrementa F, salta una posición si llega a cero	
10	<b>IORWF</b>	Operación OR entre W y F	Z
11	<b>MOVF</b>	Mueve F	Z
12	<b>MOVWF</b>	Mueve el contenido de W a F	
13	<b>NOP</b>	No operación	
14	<b>RLF</b>	Rotar F a la izquierda a través de carry	C
15	<b>RRF</b>	Rotar F a la derecha a través de carry	C
16	<b>SUBWF</b>	Resta entre W y F	C, DC, Z
17	<b>SWAPF</b>	Intercambia los nibbles de F	
18	<b>XORWF</b>	Operación XOR entre W y F	Z
<i>Operaciones relacionadas con bits dentro de registros o localidades de memoria</i>			
19	<b>BCF</b>	Borra un bit dentro de F	
20	<b>BSF</b>	Activa un bit dentro de F	
21	<b>BTFSC</b>	Prueba un bit dentro de F, salta una posición si es cero	
22	<b>BTFSS</b>	Prueba un bit dentro de F, salta una posición si es uno	
<i>Operaciones con literales y de control</i>			

23	<b>ADDLW</b>	Suma el contenido de W con una literal	C, DC, Z
24	<b>ANDLW</b>	Operación AND entre W y una literal	Z
25	<b>CALL</b>	Llama a una subrutina	TO PD
26	<b>CLRWDT</b>	Borra el temporizador watchdog	
27	<b>GOTO</b>	Salto incondicional a una dirección	
28	<b>IORLW</b>	Operación OR entre W y una literal	Z
29	<b>MOVLW</b>	Mueve una literal a W	
30	<b>RETFIE</b>	Regreso de una interrupción	
31	<b>RETLW</b>	Regresa de una subrutina con una literal en W	
32	<b>RETURN</b>	Regreso de una subrutina	
33	<b>SLEEP</b>	Entrar en modo de espera y bajo consumo	TO, PD
34	<b>SUBLW</b>	Subtrae W de una literal	C, DC, Z
35	<b>XORLW</b>	Operación XOR entre W y una literal	Z

Estas son todas las órdenes que se pueden indicar a un microcontrolador PIC de la familia 12-16 (8 bits); ahora se detallará qué significa cada una y cómo se lleva a cabo. Se hará esta descripción en estricto orden alfabético, para localizar fácilmente la explicación de cualquier comando.

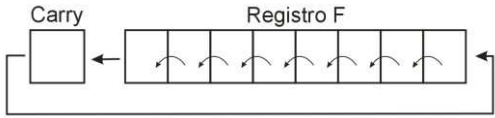
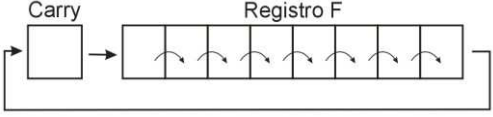
<p><b>ADDLW k</b></p> <p>Se suma el valor de W con una literal k, donde k = de 0 a 255, el resultado se guarda en W.</p> <p>Ejemplo: <b>ADDLW 0x15</b></p> <p>Si W era igual a 0x10, después de la instrucción, W = 0x25</p> <p>Si el resultado rebasa los 8 bits, se activa el bit CARRY.</p>	<p><b>ADDWF</b></p> <p>Se suma el valor de W con el contenido en la localidad de memoria F; el resultado se guarda en F si el bit de opción es 1, y en W si es 0.</p> <p>Ejemplo: <b>ADDWF REG1, 1</b></p> <p>En este caso, se suman W y REG1, y el resultado se guarda en REG1.</p> <p>Si el resultado rebasa los 8 bits, se activa el bit CARRY.</p>
--	--

<p><b>ANDLW</b></p> <p>Operación AND entre W y una literal k, donde k = de 0 a 255, el resultado se guarda en W.</p> <p>Ejemplo: <b>ANDLW 0x5F</b></p> <p>Si W era igual a 0xA3, después de la instrucción, W = 0x03</p>	<p><b>ANDWF</b></p> <p>Operación AND entre W y el contenido de una localidad de memoria F; el resultado se guarda en F si el bit de opción es 1, y en W si es 0.</p> <p>Ejemplo: <b>ANDWF REG1, 0</b></p> <p>En este caso, se hace una AND entre W y REG1, y el resultado se guarda en W.</p>
<p><b>BCF</b></p> <p>Borra un bit dentro de una localidad de memoria F.</p> <p>Ejemplo: <b>BCF REG1, 7</b></p> <p>Se borra el bit 7 de REG1; si REG1 era 0xF1, después de la orden será 0x71</p>	<p><b>BSF</b></p> <p>Activa un bit dentro de una localidad de memoria F.</p> <p>Ejemplo: <b>BSF REG1, 3</b></p> <p>Se activa el bit 3 de REG1, si REG1 era 0x51, después de la orden será 0x55</p>
<p><b>BTFSC</b></p> <p>Prueba un bit dentro de una localidad F, y el PC salta una unidad si es cero.</p> <p>Ejemplo: <b>BTFSC REG1, 5</b></p> <p>Si el bit 5 de REG1 es 1, el PC sigue con su cuenta normal, pero si es 0, salta una unidad.</p>	<p><b>BTFSS</b></p> <p>Prueba un bit dentro de una localidad F, y el PC salta una unidad si es uno.</p> <p>Ejemplo: <b>BTFSS REG1, 5</b></p> <p>Si el bit 5 de REG1 es 0, el PC sigue con su cuenta normal, pero si es 1, salta una unidad.</p>

<p><b>CALL</b></p> <p>Se llama al contenido de una subrutina.</p> <p>Ejemplo: <b>CALL 1SEG</b></p> <p>El contenido del PC se guarda en el Stack, y cambia a la dirección de la subrutina. Cuando se regresa de ella, se recupera el número del Stack hacia el PC, y sigue con su cuenta normal.</p>	<p><b>CLRF</b></p> <p>Borra el contenido de una localidad de memoria F.</p> <p>Ejemplo: <b>CLRF REG1</b></p> <p>Sin importar qué número estuviera guardado en REG1, después de la orden se tendrá un 0x00</p>
---	---

<p><b>CLRW</b></p> <p>Borra el contenido de una localidad de memoria F. Ejemplo: <b>CLRW</b></p> <p>Sin importar qué número estuviera guardado en W, después de la orden se tendrá 0x00</p>	<p><b>CLRWDT</b></p> <p>Borra el temporizador Watchdog de vigilancia.</p> <p>Ejemplo: <b>CLRWDT</b></p> <p>Se reinicializa el contador interno del WDT.</p>
<p><b>COMF</b></p> <p>Operación complemento en el contenido de la localidad de memoria F. Si el bit de opción es 0, el resultado se guarda en W, y en REG1 si el bit es 1.</p> <p>Ejemplo: <b>COMF REG1, 0</b></p> <p>Si REG1 era 0x13, luego de la operación REG1 = 0x13 y W = 0xEC</p>	<p><b>DECF</b></p> <p>Se decrementa en una unidad el contenido de F. Si el bit de opción es 0, el resultado se guarda en W, y en REG1 si el bit es 1.</p> <p>Ejemplo: <b>DECF REG1, 1</b></p> <p>Si REG1 era 0x24, después de la instrucción REG1 = 0x23</p>
<p><b>DECFSZ</b></p> <p>Se decrementa en una unidad el valor</p>	<p><b>GOTO</b></p> <p>Salto incondicional hacia cualquier</p>
<p>de F, y cuando llega a cero, el PC hace un salto de una unidad en su cuenta.</p> <p>Ejemplo: <b>DECFSZ REG1</b></p> <p>Si REG1 no es cero, el PC sigue con su cuenta normal, pero si llega a cero, el PC hace un salto de una unidad.</p>	<p>dirección válida de memoria</p> <p>Ejemplo: <b>GOTO RUTINAX</b></p> <p>Sin importar dónde estuviera el PC, luego de la orden irá a la dirección de la rutina-X</p>

<p><b>INCF</b></p> <p>Se incrementa en una unidad el valor dentro de F. Si el bit de opción es 1, el valor se guarda dentro de F, y si es 0, se guarda en W.</p> <p>Ejemplo: <b>INCF REG1, 1</b></p> <p>Si REG1 era 0x24, después de la orden REG1 = 0x25</p>	<p><b>INCFSZ</b></p> <p>Se incrementa el valor de F, y si llega a cero, el PC hace un salto de una unidad. Si el bit de opción es 1, el valor se guarda dentro de F, y si es 0, se guarda en W.</p> <p>Ejemplo: <b>INCFSZ REG1, 1</b></p> <p>Si REG1 tiene cualquier valor distinto a cero, el PC sigue su cuenta normal, pero si llega a cero, el PC hace un salto de una unidad.</p>
<p><b>IORLW</b></p> <p>Se hace una operación OR entre W y una literal k, donde k = de 0 a 255; el resultado se guarda en W.</p> <p>Ejemplo: <b>IORLW 0x35</b></p> <p>Si W era 0x9A, luego de la instrucción W = 0xBF</p>	<p><b>IORWF</b></p> <p>Se hace una operación OR entre W y el contenido de F; si el bit de opción es 1, el valor se guarda dentro de F, y si es 0, se guarda en W.</p> <p>Ejemplo: <b>IORWF REG1, 0</b></p> <p>Si W = 0x91 y REG1 = 0x13, luego de la instrucción W = 0x93</p>
<p><b>MOVLW</b></p> <p>Carga una literal k en el registro W, donde k = de 0 a 255.</p> <p>Ejemplo: <b>MOVLW 0x3A</b></p> <p>Sin importar qué hubiera en W anteriormente, después de la orden W = 0x3A</p>	<p><b>MOVF</b></p> <p>El contenido de la dirección de memoria F se mueve a W si el bit de opción es 0, y se queda en F si el bit de opción es 1.</p> <p>Ejemplo: <b>MOVF REG1, 0</b></p> <p>El valor de REG1 se pasará a W después de la orden.</p>
<p><b>MOVWF</b></p> <p>El contenido del registro W se graba en la localidad de memoria F.</p> <p>Ejemplo: <b>MOVWF REG1</b></p> <p>Si W = 0xA1, después de la orden este valor se cargará en REG1.</p>	<p><b>NOP</b></p> <p>Ninguna operación; se utiliza para introducir un retardo de un ciclo de reloj, cuando se quiere llevar un tiempo muy preciso.</p> <p>Ejemplo: <b>NOP</b></p>

<p><b>RETFIE</b></p> <p>Regresa desde una interrupción: cuando se recibe una interrupción, el PC hace un salto incondicional a una cierta localidad de memoria, pero guarda su valor anterior en el Stack; cuando se termina de ejecutar la rutina que se invoca por la interrupción, una orden RETFIE regresa al PC a la localidad de memoria donde estaba anteriormente.</p> <p>Ejemplo: <b>RETFIE</b></p>	<p><b>RETLW</b></p> <p>Regresa de una subrutina con una literal cargada en W. Cuando termina una subrutina y se desea regresar al programa normal, pero con un dato específico en W, esta orden permite hacerlo en un solo comando.</p> <p>Ejemplo: <b>RETLW, 0xA3</b></p> <p>Al regresar de la subrutina, el registro W tendrá en su interior un valor de 0xA3.</p>
<p><b>RETURN</b></p> <p>Regreso desde una subrutina. Cuando se termine de ejecutar una subrutina y se desee regresar al programa normal, una orden RETURN hace que el PC recupere el valor guardado en el Stack, y retome su cuenta normal.</p> <p>Ejemplo: <b>RETURN</b></p>	<p><b>RLF</b></p> <p>Rota los bits del registro F hacia la izquierda, pasando por el bit CARRY</p>  <p>Ejemplo: <b>RLF REG1</b></p>
<p><b>RRF</b></p> <p>Rota los bits del registro F hacia la derecha, pasando por el bit CARRY.</p>  <p>Ejemplo: <b>RRF REG1</b></p>	<p><b>SLEEP</b></p> <p>Esta orden coloca al micro en un modo de mínima energía, mientras espera nuevas órdenes. Ejemplo: <b>SLEEP</b></p>
<p><b>SUBLW</b></p>	<p><b>SUBWF</b></p>

<p>Subtrae el valor de W de una literal k, donde k = de 0 a 255. El resultado se guarda en W.</p> <p>Ejemplo: <b>SUBLW 0x28</b></p> <p>Si W = 0x03, luego de la orden, W = 0x25. Dependiendo si W es mayor, menor o igual a la literal, se activan los bits C, Z y DC</p>	<p>Subtrae el valor de W del contenido de F. El resultado se guarda en W si el bit de opción es 0, y en F si es 1.</p> <p>Ejemplo: <b>SUBWF REG1, 1</b></p> <p>Dependiendo si W es mayor, menor o igual a la literal, se activan los bits C, Z y DC</p>
<p><b>SWAPF</b></p> <p>Intercambia los nibbles (grupos de 4 bits) del contenido de F. El resultado se guarda en W si el bit de opción es 0, y en F si es 1.</p> <p>Ejemplo: <b>SWAPF REG1, 1</b></p> <p>Si REG1 = 0x9A, después de la orden REG1 = 0xA9</p>	<p><b>XORLW</b></p> <p>Operación XOR entre el contenido de W y una literal k, donde k = de 0 a 255. El resultado se guarda en W.</p> <p>Ejemplo: <b>XORLW 0xAF</b></p> <p>Si W era 0xB5, después de la orden W = 0x1A</p>
<p><b>XORWF</b></p> <p>Operación XOR entre W y el contenido de F. El resultado se guarda en W si el bit de opción es 0, y en F si es 1.</p> <p>Ejemplo: <b>XORWF REG1, 0</b></p> <p>Si W = 0xB5 y REG1 = 0xAF, luego de la orden W = 0x1A</p>	

De este modo, se puede tener una idea más clara de para qué sirve cada una de las órdenes incluidas en el set de instrucciones de los microcontroladores PIC serie 12-16. Una gran ventaja que tienen estos dispositivos es que su set de comandos casi no ha cambiado desde que comenzaron a fabricarlos, así que un programa que se diseñe en este momento puede funcionar prácticamente sin modificaciones en otros micros más antiguos o en dispositivos que aún están en la mesa de diseño; de ahí la conveniencia de realizar los programas en forma de bloques funcionales, para que cuando llegue algún proyecto con una etapa que ya se había diseñado para un circuito anterior, se pueda retomar haciéndole sólo modificaciones mínimas.

Antes de pasar a las prácticas de programación, se debe describir un aspecto importante de la operación de este microcontrolador: los registros especiales donde se activan las banderas de estado; y esto implica echar un vistazo al mapa de memoria del dispositivo.

A continuación se encuentra el mapa de memoria de un microcontrolador PIC 16F628A; se puede observar que varias de las localidades de memoria que posee el micro ya están ocupadas por una gran cantidad de registros, cada uno con una función específica dentro de la estructura del micro. Se describirán los más importantes, y se dejarán algunos para cuando se estudien ciertos temas específicos sobre este controlador.

Se iniciará por orden numérico: en la localidad 01h se encuentra el registro correspondiente al temporizador No. 0; por el momento no se indicará para qué sirve, ya que este bloque sólo se utiliza para ciertas funciones avanzadas del micro.

El registro PCL corresponde a los 8 bits inferiores del registro PC o contador de programa; dado que el 16F628A posee 2kB de memoria de programa, pero con 8 bits sólo se pueden direccionar 256 bits, esto implica que necesita por lo menos 3 bits adicionales para direccionar a toda su memoria, entonces, en PCL se guardan los 8 bits inferiores, y en PCLATH los adicionales.

El registro STATE o de estado guarda algunos de los bits más usuales durante la programación del dispositivo, como las banderas C (carry), Z (cero) y DC (decrement carry), además del IRP, RP0 y RP1 (elección del banco de memoria usado), TO (Time out) y PD (Power down). En programas sencillos sólo se usarán los bits C, Z y DC, pero en programas más complejos se llegan a utilizar todos los demás, especialmente los tres primeros, ya que se activan o desactivan dependiendo de en qué bloque de memoria se alojen las instrucciones del programa.

El registro OPTION\_REG controla aspectos de las interrupciones y los temporizadores, pero hay un bit especial RBPU, que habilita o desactiva unas resistencias de pull-up en las terminales de salida del micro. Si se activan esas resistencias (poniendo este bit en "0"), se podrá conectar un LED directamente a las terminales del chip, sin necesidad de resistencia limitadora, lo que reduce