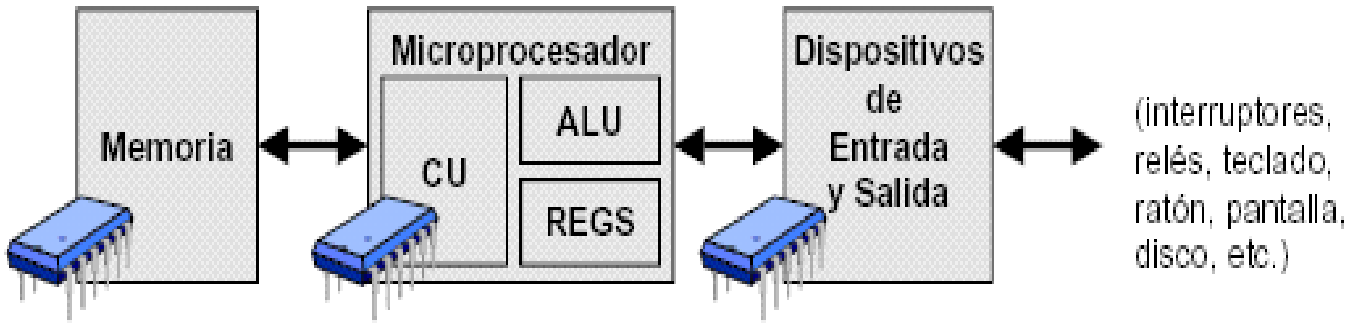
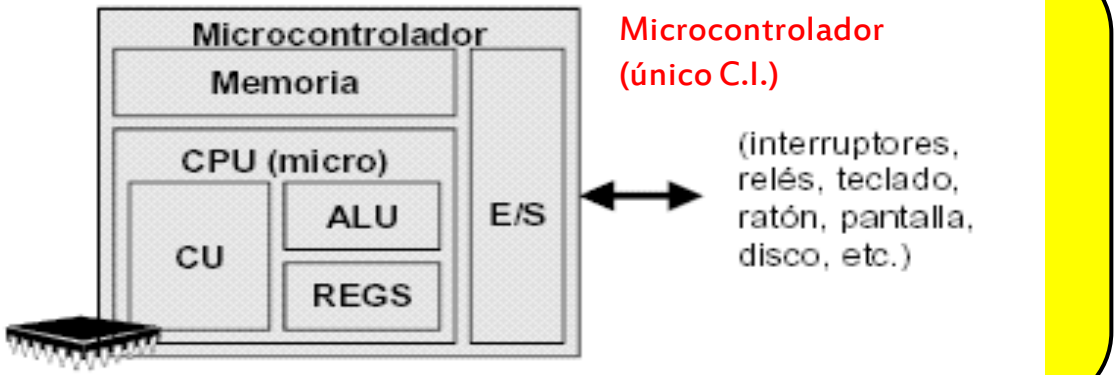
MICROCONTROLADOR PIC DE MICROCHIP

Sistema Microprocesador (varios C.I. en una PCB)





Microcontrolador (único C.I.)

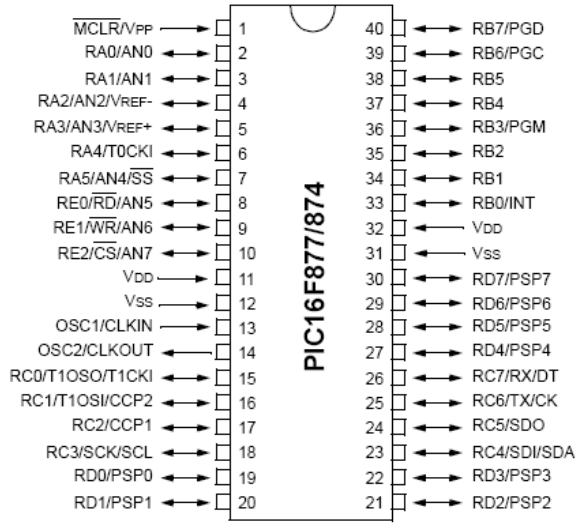
MICROCONTROLADOR PIC DE MICROCHIP

PIC16F877 - 28-PIN 8-BIT CMOS FLASH MICROCONTROLLER

Microcontroller Core Features:

- 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 the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
  Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

PDIP



Peripheral Features:

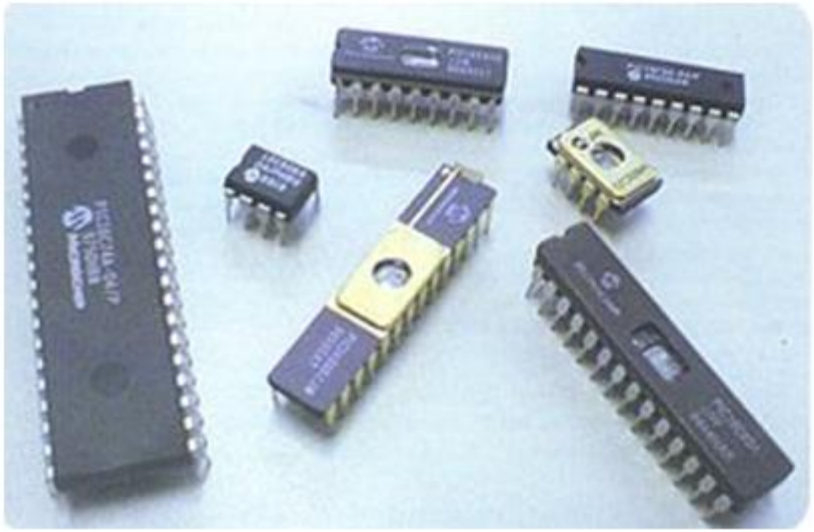
- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max.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

PIC16CXXX/PIC16FXXX Family: 14-bit program word

With the introduction of new PIC16CXXX/PIC16FXXX family members, Microchip now provides the industry's highest performance Analog-to-Digital Converter capability at 12-bits for an MCU. The PIC16CXXX/PIC16FXXX family offers a wide-range of options, from 18- to 68-pin packages as well as low to highest levels of peripheral integration. This family has a 14-bit wide instruction set, interrupt handling capability and a deep, 8-level hardware stack. The PIC16CXXX/PIC16FXXX family provides the performance and versatility to meet the more demanding requirements of today's cost-sensitive marketplace for mid-range applications.

Manuel Rico Secades
Marzo 2003

• Familia PIC10F20x	4 Dispositivos
• Familia PIC12CXXX/12FXXX (12/14 bits)	8 Dispositivos
• Familia PIC16C5X (12 bits)	9 Dispositivos
• Familia PIC16CXXX/16FXXX (14 bits)	74 Dispositivos
• Familia PIC17CXXX (16 bits)	7 Dispositivos
• Familia PIC18CXXX/18FXXX (16 bits)	82 Dispositivos



TOTAL: ¡ 177 Dispositivos !

Versiones de Memoria de Programa:
OTP, EPROM, EEPROM y FLASH

PIC16 Microcontroller Family

PIC16 Microcontroller Family

	Data RAM	ADC	Serial COM	Comparators	Timers	Low Voltage Device
PIC16C54C	25	-	-	-	1+WDT	PIC16LC54C
PIC16CR54C	25	-	-	-	1+WDT	PIC16LCR54C
PIC16C55A	24	-	-	-	1+WDT	PIC16LC55A
PIC16C56A	25	-	-	-	1+WDT	PIC16LC56A
PIC16CR56A	25	-	-	-	1+WDT	PIC16LCR56A
PIC16C57C	72	-	-	-	1+WDT	PIC16LC57C
PIC16CR57C	72	-	-	-	1+WDT	PIC16LCR57C
PIC16C58B	73	-	-	-	1+WDT	PIC16LC58B
PIC16CR58B	73	-	-	-	1+WDT	PIC16LCR58B
PIC16C505	72	-	-	-	1+WDT	PIC16LC505
PIC16HV540	25	-	-	-	1+WDT	-
PIC16C554	80	-	-	-	1+WDT	PIC16LC554
PIC16C558	128	-	-	-	1+WDT	PIC16LC558
PIC16C62B	128	-	I²C, SPI	-	3+WDT	PIC16LC62B
PIC16C63A	192	-	USART, I²C, SPI	-	3+WDT	PIC16LC63A
PIC16CR63	192	-	USART, I²C, SPI	-	3+WDT	PIC16CR63
PIC16C65B	192	-	USART, I²C, SPI	-	3+WDT	PIC16LC65B
PIC16CR65	192	-	USART, I²C, SPI	-	3+WDT	PIC16LCR65
PIC16C66	368	-	USART, I²C, SPI	-	3+WDT	PIC16LC66
PIC16C67	368	-	USART, I²C, SPI	-	3+WDT	PIC16LC67
PIC16C432	128	-	-	2	1+WDT	PIC16LC432
PIC16C433	128	4/8-bit	-	-	1+WDT	PIC16LC433
PIC16C620A	96	-	-	2	1+WDT	PIC16LC620A
PIC16CR620A	96	-	-	2	1+ WDT	-
PIC16C621A	96	-	-	2	1+WDT	PIC16LC621A
PIC16C622A	128	-	-	2	1+WDT	PIC16LC622A
PIC16CE623	96	-	-	2	1+WDT	PIC16LCE623
PIC16CE624	96	-	-	2	1+WDT	PIC16LCE624
PIC16CE625	128	-	-	2	1+WDT	PIC16LCE625
PIC16C642	176	-	-	2	1+WDT	PIC16LC642
PIC16C662	176	-	-	2	1+WDT	PIC16LC662
PIC16C710	36	4/8-bit	-	-	1+WDT	PIC16LC710
PIC16C711	68	4/8-bit	-	-	1+WDT	PIC16LC711
PIC16C712	128	4/8-bit	-	-	3+WDT	PIC16LC712
PIC16C715	128	4/8-bit	-	-	1+WDT	PIC16LC715
PIC16C716	128	4/8-bit	-	-	3+WDT	PIC16LC716
PIC16C717	256	6/10-bit	I²C, SPI	-	3+WDT	PIC16LC717
PIC16C72A	128	5/8-bit	I²C, SPI	-	3+WDT	PIC16LC72A
PIC16CR72	128	5/8-bit	I²C, SPI	-	3+WDT	PIC16LCR72
PIC16C73B	192	5/8-bit	USART, I²C, SPI	-	3+WDT	PIC16LC73B
PIC16C74B	192	8/8-bit	USART, I²C, SPI	-	3+WDT	PIC16LC74B
PIC16C76	368	5/8-bit	USART, I²C, SPI	-	3+WDT	PIC16LC76
PIC16C77	368	8/8-bit	USART, I²C, SPI	-	3+WDT	PIC16LC77
PIC16C770	256	6/12-bit	I²C, SPI	-	3+WDT	PIC16LC770
PIC16C771	256	6/12-bit	I²C, SPI	-	3+WDT	PIC16LC771
PIC16C773	256	6/12-bit	USART, I²C, SPI	-	3+WDT	PIC16LC773
PIC16C774	256	10/12-bit	USART, I²C, SPI	-	3+WDT	PIC16LC774
PIC16C745	256	5/8-bit	USB, USART	-	3+WDT	-
PIC16C765	256	8/8-bit	USB, USART	-	3+WDT	-
PIC16C781	128	8/8-bit	-	2	2+WDT	PIC16LC781
PIC16C782	128	8/8-bit	-	2	2+WDT	PIC16LC782
PIC16C923	176	-	I²C, SPI	-	3+WDT	PIC16LC923
PIC16C924	176	5/8-bit	I²C, SPI	-	3+WDT	PIC16LC924
PIC16C925	176	5/10-bit	SPI, I²C	-	3+WDT	PIC16LC925
PIC16C926	336	5/10-bit	SPI, I²C	-	3+WDT	PIC16LC926
PIC16F627	224	-	USART	2	3 + WDT	PIC16LF627
PIC16F628	224	-	USART	2	3 + WDT	PIC16LF628
PIC16F73	192	5/8-bit	I²C, SPI, USART	-	3+WDT	PIC16LF73
PIC16F74	192	8/8-bit	I²C, SPI, USART	-	3+WDT	PIC16LF74
PIC16F76	368	5/8-bit	I²C, SPI, USART	-	3+WDT	PIC16LF76
PIC16F77	368	8/8-bit	I²C, SPI, USART	-	3+WDT	PIC16LF77
PIC16F84A	68	-	-	-	1+WDT	PIC16LF84A
PIC16F870	128	5/10-bit	USART	-	3+WDT	PIC16LF870
PIC16F871	128	8/10-bit	USART	-	3+WDT	PIC16LF871
PIC16F872	128	5/10-bit	I²C, SPI	-	3+WDT	PIC16LF872
PIC16F873	192	5/10-bit	USART, I²C, SPI	-	3+WDT	PIC16LF873
PIC16F873A	192	5/10-bit	USART, I²C, SPI	2	3+WDT	PIC16LF873A
PIC16F874	192	8/10-bit	USART, I²C, SPI	-	3+WDT	PIC16LF874
PIC16F874A	192	8/10-bit	USART, I²C, SPI	2	3+WDT	PIC16LF874A
PIC16F876	368	5/10-bit	USART, I²C, SPI	-	3+WDT	PIC16LF876
PIC16F876A	368	5/10-bit	USART, I²C, SPI	2	3+WDT	PIC16LF876A
PIC16F877	368	8/10-bit	USART, I²C, SPI	-	3+WDT	PIC16LF877
PIC16F877A	368	8/10-bit	USART, I²C, SPI	2	3+WDT	PIC16LF877A
PIC16C54	25	-	-	-	1+WDT	PIC16C54-LP
PIC16CR54A	25	-	-	-	1+WDT	PIC16LCR54A
PIC16C54A	25	-	-	-	1+WDT	PIC16LC54A, PIC16LV54A
PIC16C55	24	-	-	-	1+WDT	PIC16C55-LP
PIC16C56	25	-	-	-	1+WDT	PIC16LC56
PIC16C57	72	-	-	-	1+WDT	PIC16C57-LP
PIC16C62A	128	-	I²C, SPI	-	3+WDT	PIC16LC62A
PIC16C63	192	-	USART, I²C, SPI	-	3+WDT	PIC16LC63
PIC16C64A	128	-	I²C, SPI	-	3+WDT	PIC16LC64A
PIC16C65A	192	-	USART, I²C, SPI	-	3+WDT	PIC16LC65A
PIC16C620	80	-	-	2	1+WDT	PIC16LC620
PIC16C621	80	-	-	2	1+WDT	PIC16LC621
PIC16C622	128	-	-	2	1+WDT	PIC16C622
PIC16C71	36	4/8-bit	-	-	1+WDT	PIC16LC71
PIC16C72	128	5/8-bit	I²C, SPI	-	3+WDT	PIC16LC72
PIC16C73A	192	5/8-bit	USART, I²C, SPI	-	3+WDT	PIC16LC73A
PIC16C74A	192	8/8-bit	USART, I²C, SPI	-	3+WDT	PIC16LC74A
PIC16F83	36	-	-	-	1+WDT	PIC16LF83
PIC16CR83	36	-	-	-	1+WDT	PIC16LCR83
PIC16F84	68	-	-	-	1+WDT	PIC16LF84
PIC16CR84	68	-	-	-	1+WDT	PIC16LCR84
PIC16F87	368	-	AUSART	2	3+WDT	PIC16LF87
PIC16F88	192	4/8-bit	AUSART	2	3+WDT	PIC16LF88
PIC16F818	128	5/10-bit	I²C, SPI	-	3+WDT	PIC16LF818
PIC16F819	256	5/10-bit	I²C, SPI	-	3+WDT	PIC16LF819

PIC16C5X gama baja 33 instrucciones de 12 bits

PIC16C6X gama media 35 instrucciones de 14 bits

PIC16F876:
PRINCIPALES CARACTERÍSTICAS

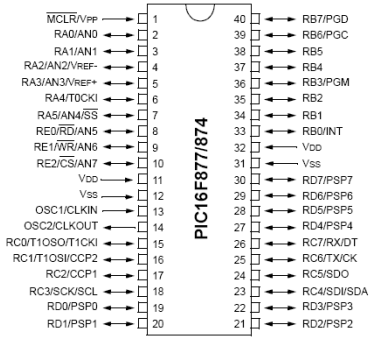
Key Features PICmicro™ Mid-Range Reference Manual (DS33023)	PIC16F873	PIC16F874	PIC16F876	PIC16F877
Operating Frequency	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz
RESETS (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory	128	128	256	256
Interrupts	13	14	13	14
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
Instruction Set	35 instructions	35 instructions	35 instructions	35 instructions

PIC16F877: ESTRUCTURA INTERNA

Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes

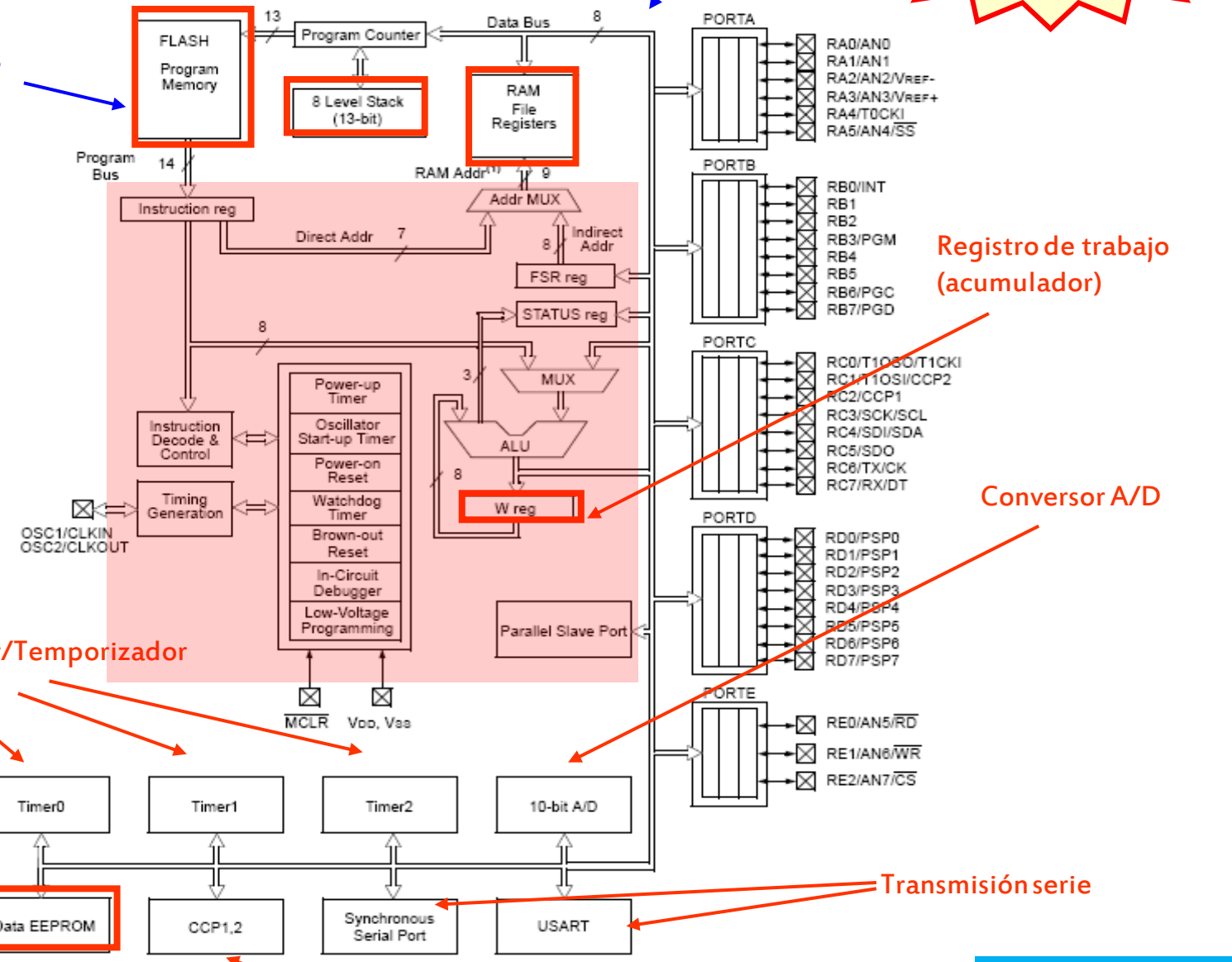
¡Tiene de todo!

PDIP



Bus de Datos

Bus de Datos (programa)



Registro de trabajo (acumulador)

Conversor A/D

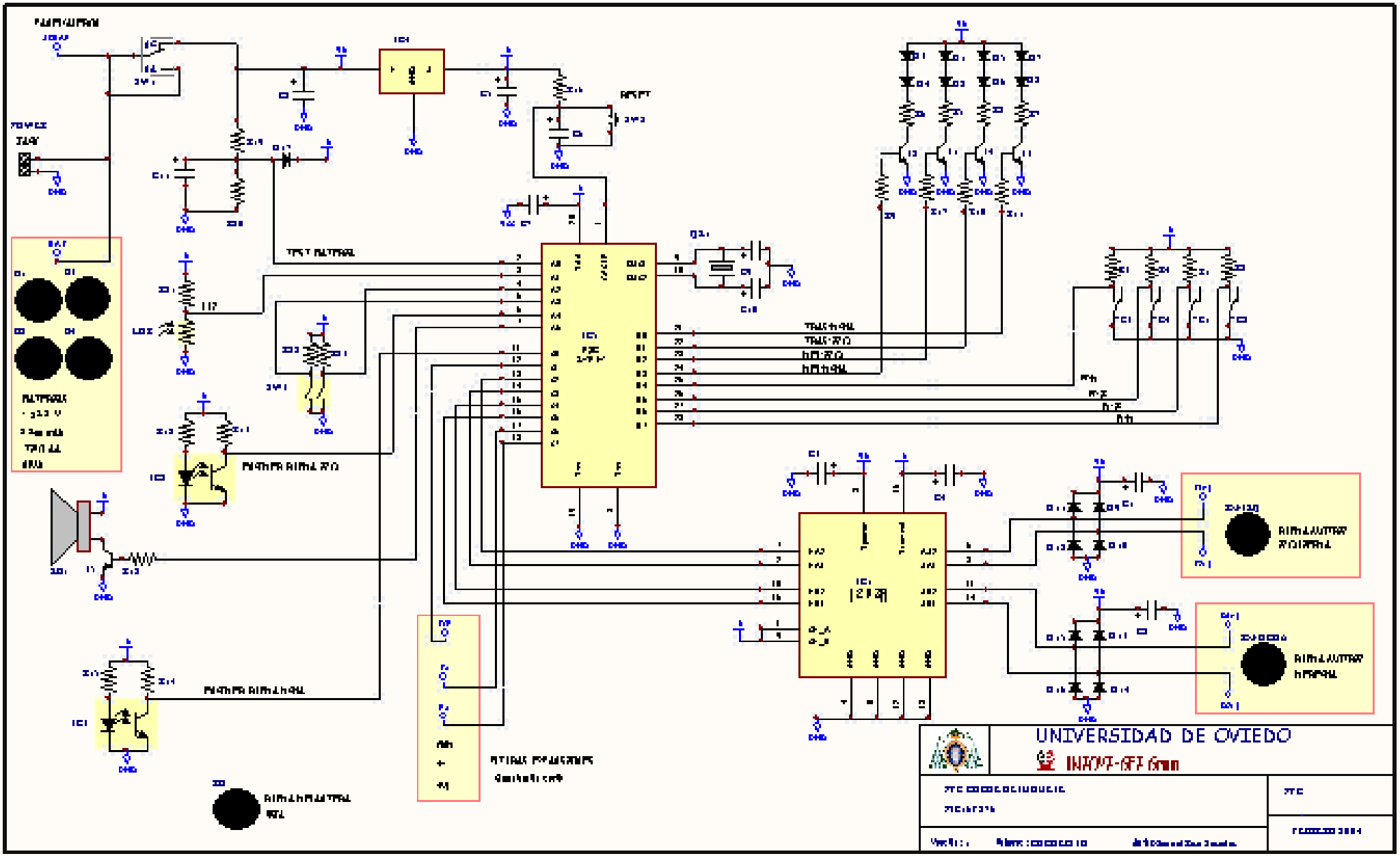
Contador/Temporizador

Memoria para datos no volátiles

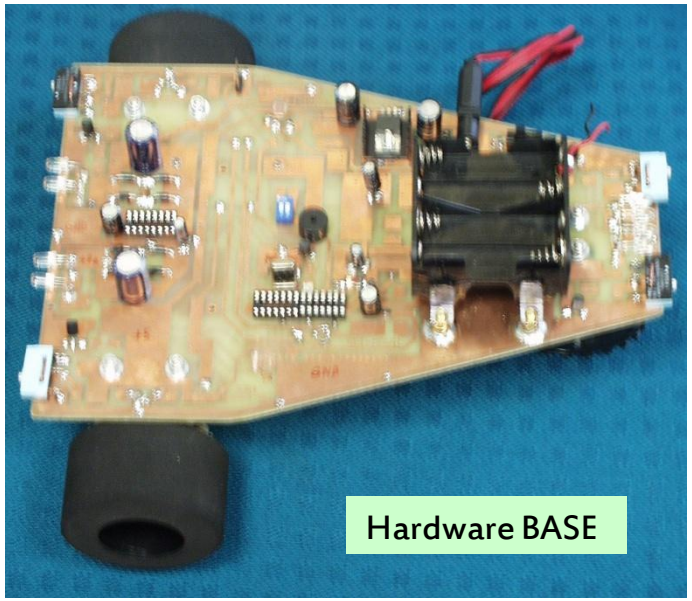
Transmisión serie

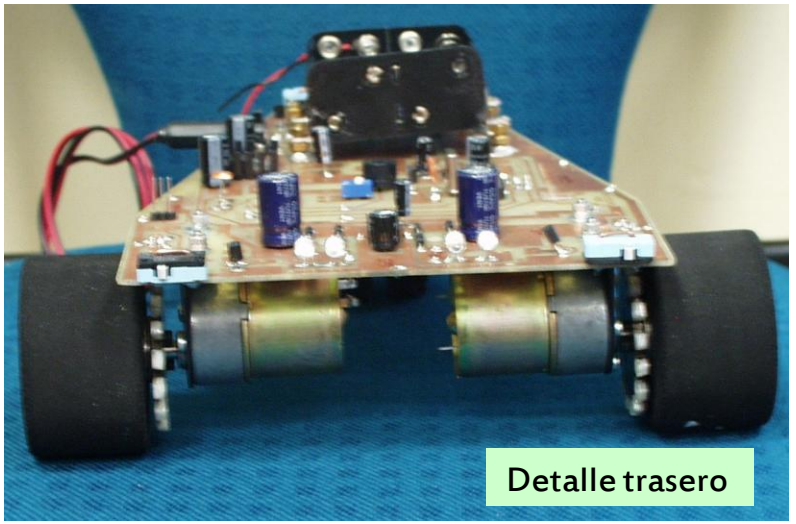
Compare/Capture/PWM

UN PEQUEÑO EJEMPLO: UN COCHE DE JUGUETE

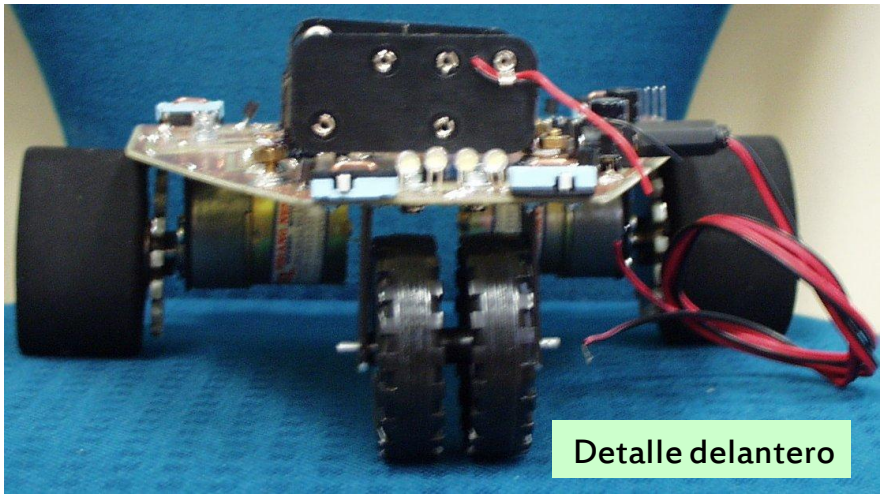


SELECCIÓN DE FOTOS


Hardware BASE


Detalle trasero


Con ALERÓN SOLAR


Detalle delantero

PIC16F877:
FUNCIÓN DE CADA PIN DEL CHIP

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS(4)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP	1	2	18	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	4	20	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	5	21	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	6	22	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	7	23	I/O	ST	RA4 can also be the clock input to the Timer0 timer/ counter. Output is open drain type.
RA5/$\overline{SS}$/AN4	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
						PORTB is a bi-directional I/O port. PORTB can be soft-ware programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	37	41	14	I/O	TTL	Interrupt-on-change pin.
RB5	38	42	15	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	39	43	16	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	44	17	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.

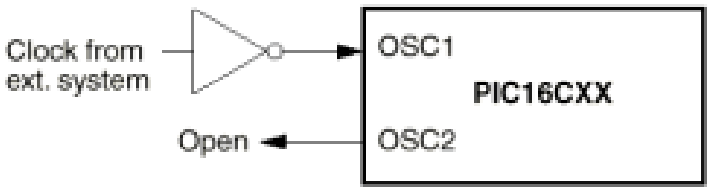
Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

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.

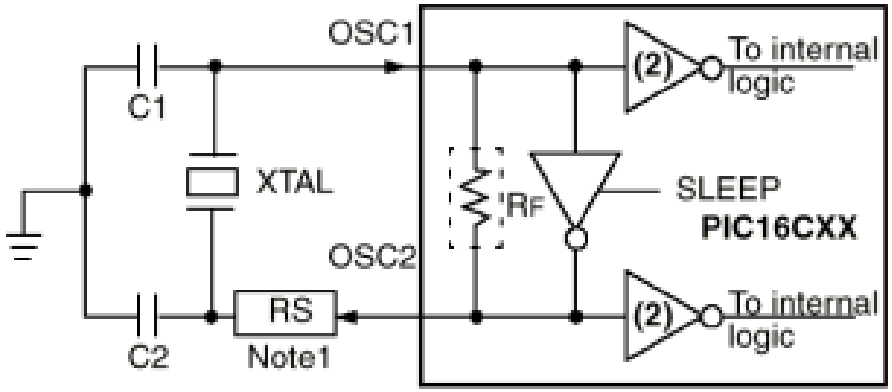
PIC16F877:
FUNCIÓN DE CADA PIN DEL CHIP

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
						PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	RC0 can also be the Timer1 oscillator output or a Timer1 clock input.
RC1/T1OSI/CCP2	16	18	35	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	17	19	36	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18	20	37	I/O	ST	RC3 can also be the synchronous serial clock input/ output for both SPI and I^2C modes.
RC4/SDI/SDA	23	25	42	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	24	26	43	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	25	27	44	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	26	29	1	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
						PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38	I/O	ST/TTL(3)	
RD1/PSP1	20	22	39	I/O	ST/TTL(3)	
RD2/PSP2	21	23	40	I/O	ST/TTL(3)	
RD3/PSP3	22	24	41	I/O	ST/TTL(3)	
RD4/PSP4	27	30	2	I/O	ST/TTL(3)	
RD5/PSP5	28	31	3	I/O	ST/TTL(3)	
RD6/PSP6	29	32	4	I/O	ST/TTL(3)	
RD7/PSP7	30	33	5	I/O	ST/TTL(3)	
						PORTE is a bi-directional I/O port.
RE0/$\overline{RD}$/AN5	8	9	25	I/O	ST/TTL(3)	RE0 can also be read control for the parallel slave port, or analog input5.
RE1/$\overline{WR}$/AN6	9	10	26	I/O	ST/TTL(3)	RE1 can also be write control for the parallel slave port, or analog input6.
RE2/$\overline{CS}$/AN7	10	11	27	I/O	ST/TTL(3)	RE2 can also be select control for the parallel slave port, or analog input7.
Vss	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
VDD	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17,28, 40	12,13, 33,34		—	These pins are not internally connected. These pins should be left unconnected.

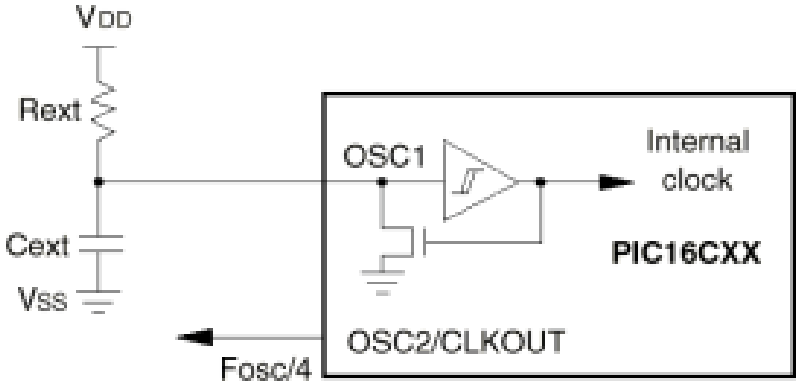
OPCIONES PARA EL RELOJ



Reloj externo

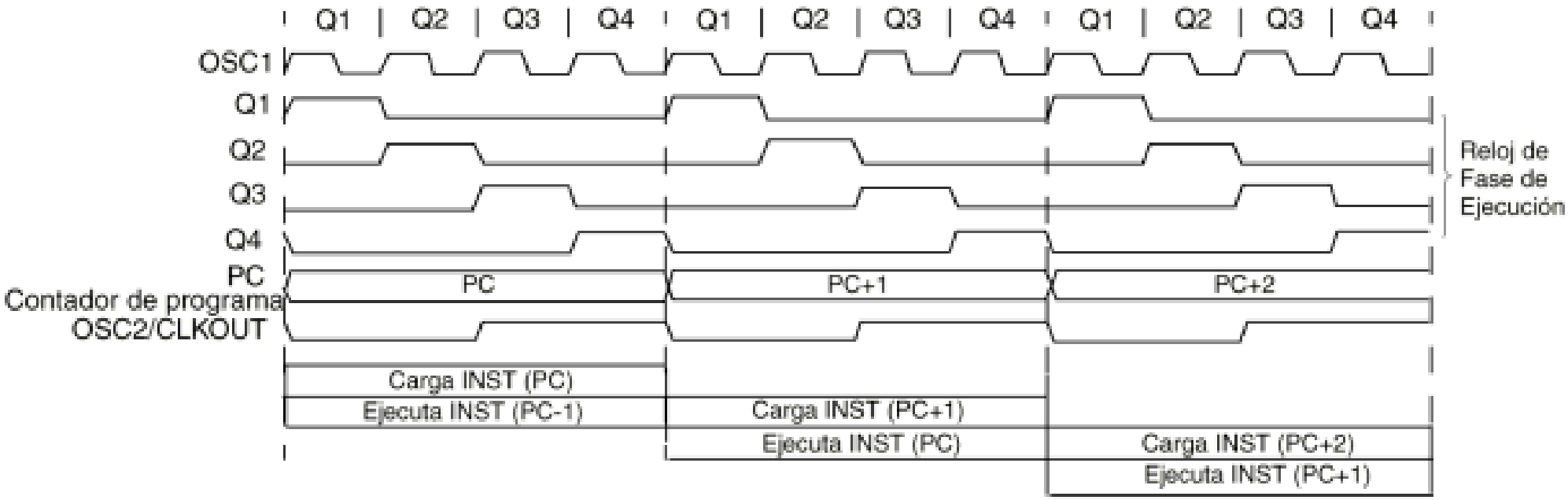


Oscilador a cristal



Red RC

MICROCONTROLADORES PIC: EJECUCIÓN SEGMENTADA DE INSTRUCCIONES (Arquitectura Harvard)



"Todas las instrucciones se ejecutan en 4 ciclos de reloj, excepto los saltos que necesitan 8"

PIC16F877: MEMORIA DE PROGRAMA

TIENE 8K x 14 CAPACIDAD DE MEMORIA FLASH CMOS DE ALTA VELOCIDAD ("HIGH SPEED CMOS FLASH").

DISPONE DE UNA PILA DE 8 NIVELES.

LA PILA OCUPA UN ESPACIO APARTE DENTRO DEL CHIP Y NO SE PUEDE LEER NI ESCRIBIR DATOS EN ELLA.

LA PILA TIENE FUNCIONAMIENTO CIRCULAR, ES DECIR UNA VEZ LLENADO EL NIVEL 8 COMIENZA POR EL 1 OTRA VEZ.

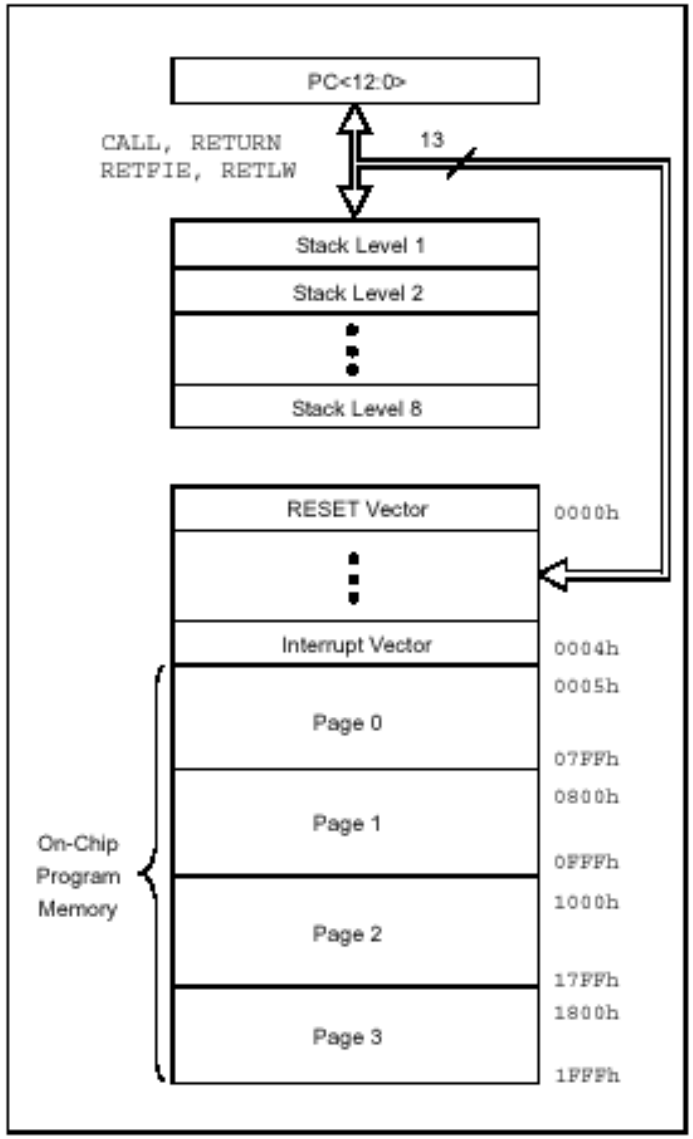
¡CUIDADO!. NO TENEMOS NINGÚN FLAG PARA INDICAR QUE LA PILA ESTA LLENA.

NOTA:

La posición 2007 (fuera del área de programación) están los bits de configuración (CONFIGURATION WORD) que ocupan 14 bits como una instrucción de programa.

Muy importante configurarlo adecuadamente.
(Ver siguiente lámina y parte de características especiales)

FIGURE 2-1: PIC16F877/876 PROGRAM MEMORY MAP AND STACK



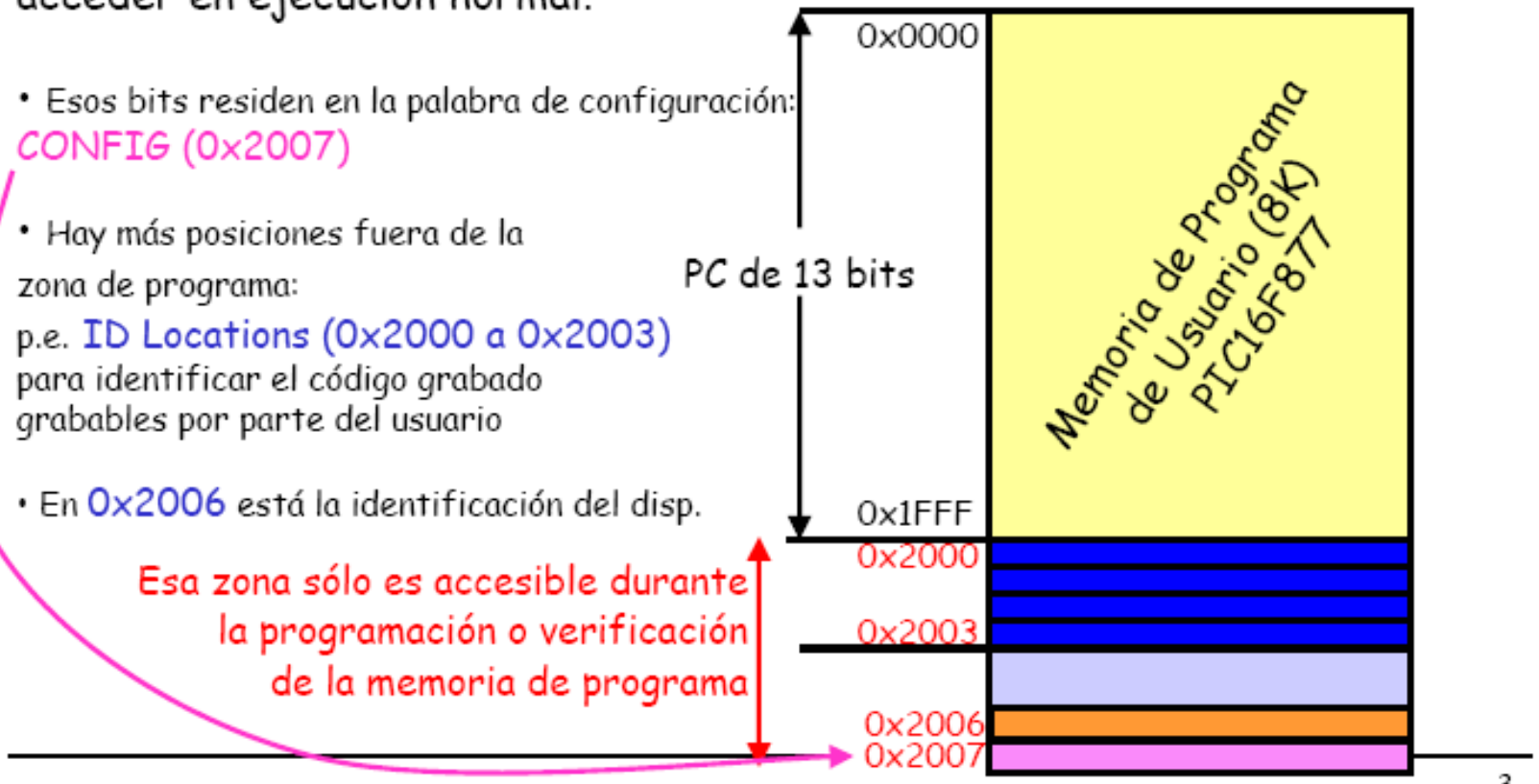
Configuración de las Características Especiales

• La mayor parte de las características especiales se establecen en unos bits que residen en una palabra de iguales características que la memoria de programa pero está fuera del mapa de memoria de código a la que se puede acceder en ejecución normal.

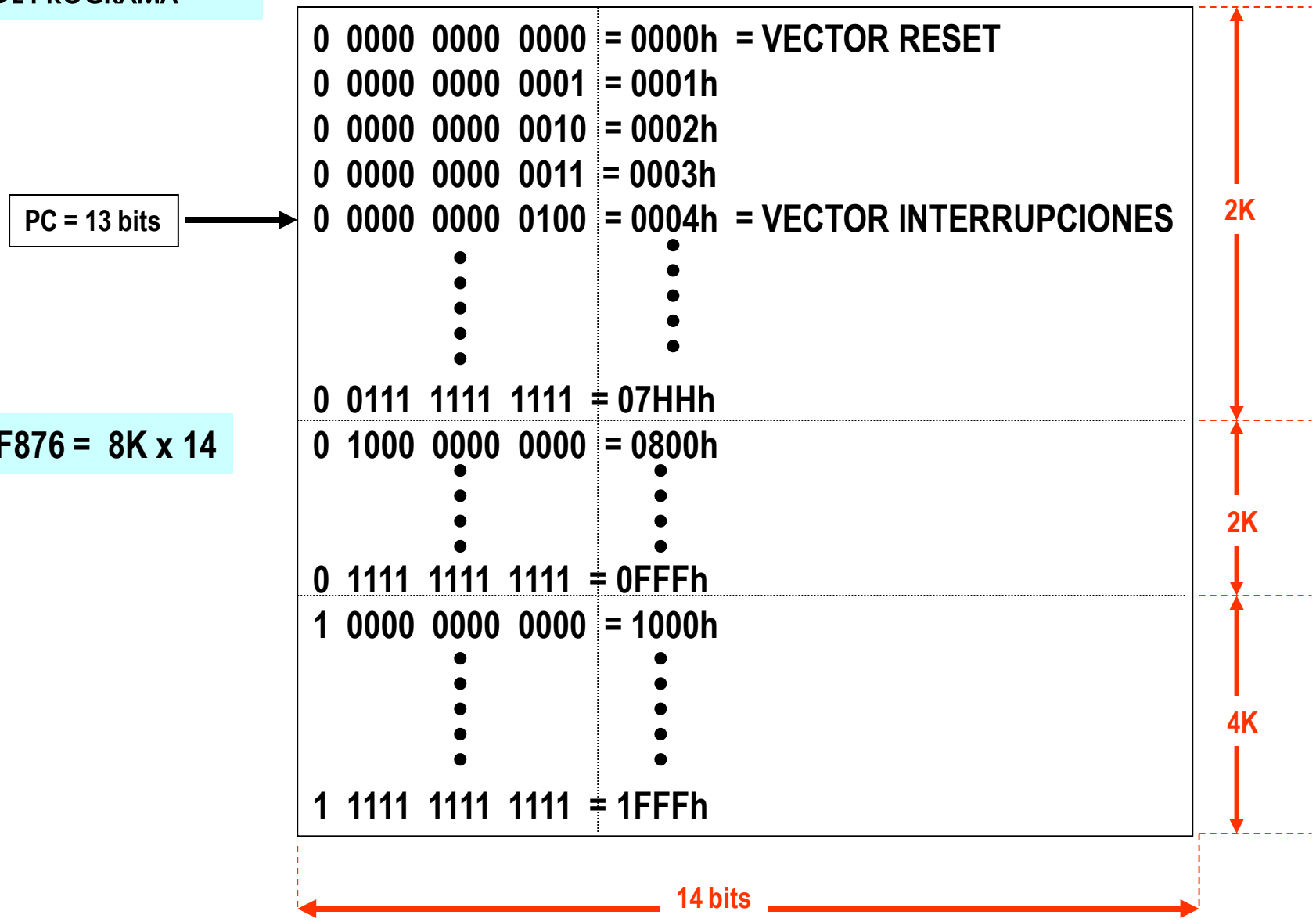
• Esos bits residen en la palabra de configuración: CONFIG (0x2007)

• Hay más posiciones fuera de la zona de programa:
p.e. ID Locations (0x2000 a 0x2003) para identificar el código grabado grabables por parte del usuario

• En 0x2006 está la identificación del disp.

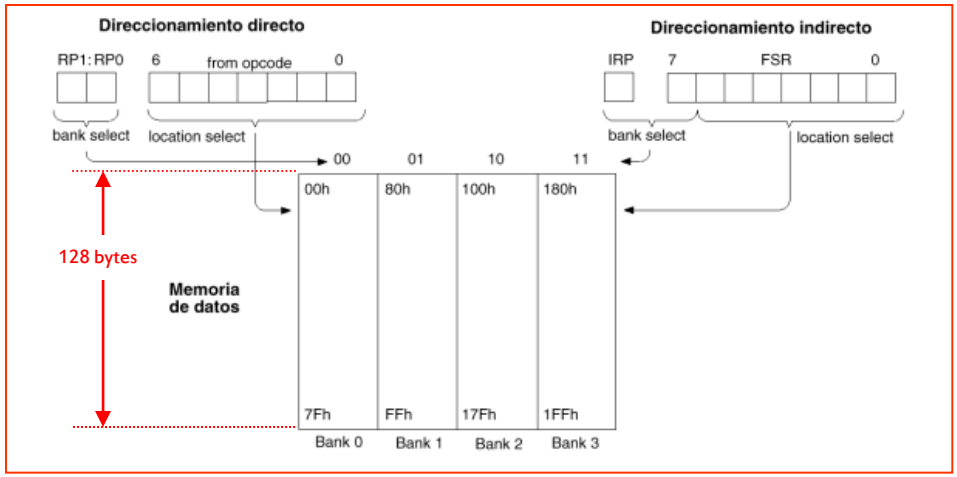


PIC16F876: MEMORIA DE PROGRAMA



PIC16F876 = 8K x 14

PIC16F876: MEMORIA DE DATOS

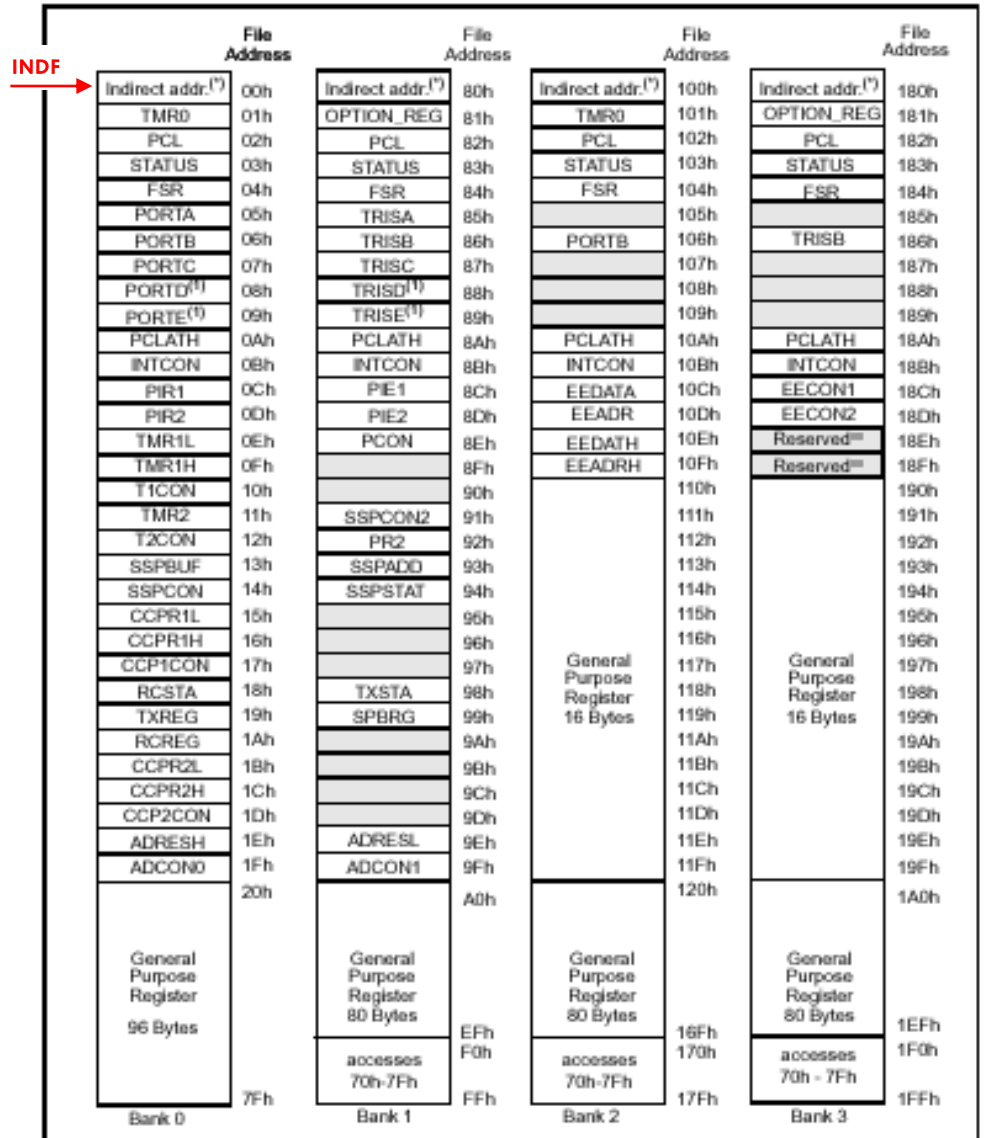


RP1:RP0	Bank
00	0
01	1
10	2
11	3

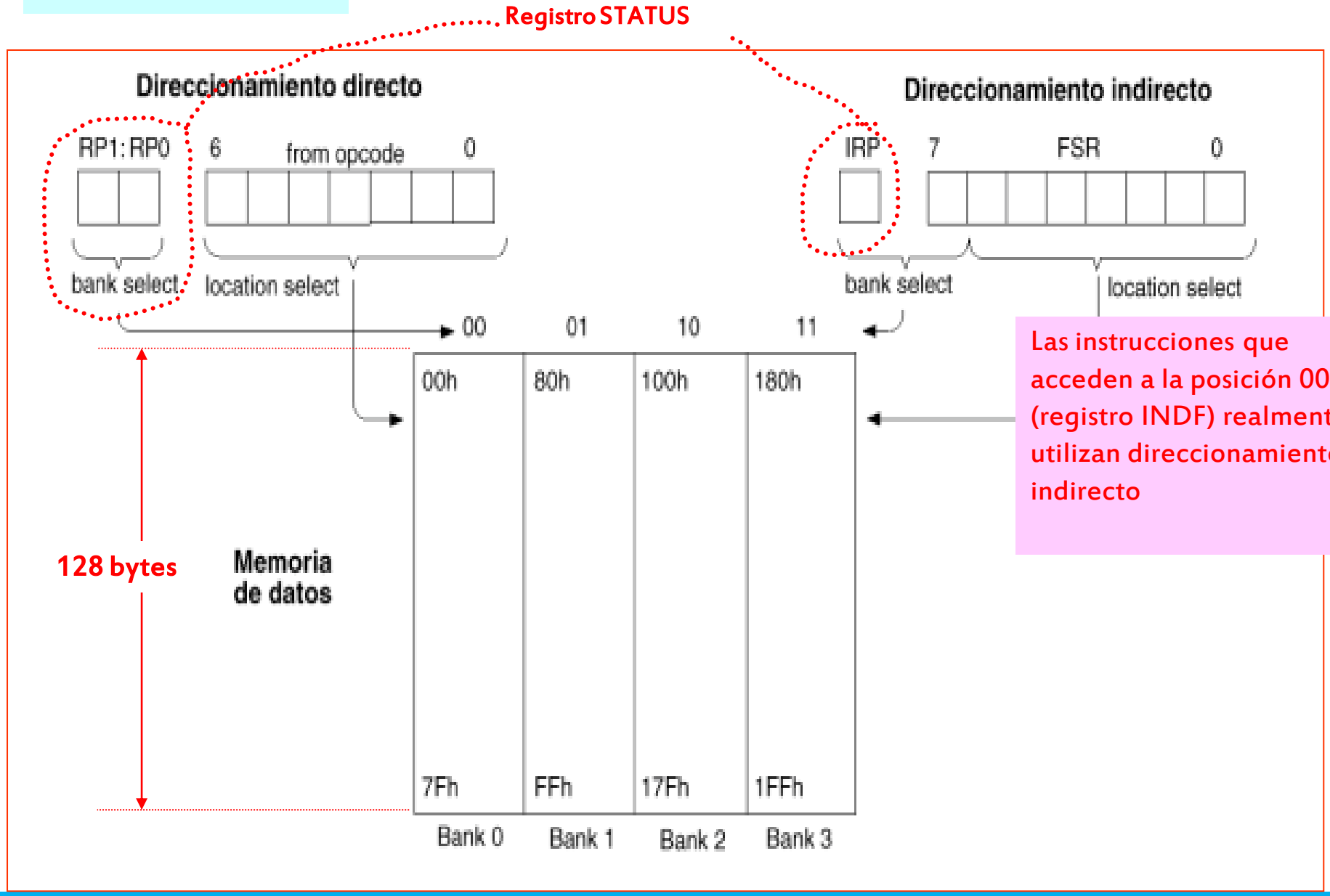
PIC16F876.....368 x 8 de RAM

Banco 0: 96 bytes
Banco 1: 80 bytes
Banco 2: 16 bytes + 80 bytes
Banco 3: 16 bytes + 80 bytes

FIGURE 2-3: PIC16F877/876 REGISTER FILE MAP



PIC16F876: MEMORIA DE DATOS



Reflexión:

No confundir el sitio donde están los datos (dirección) con los datos propiamente.

Para el programa:

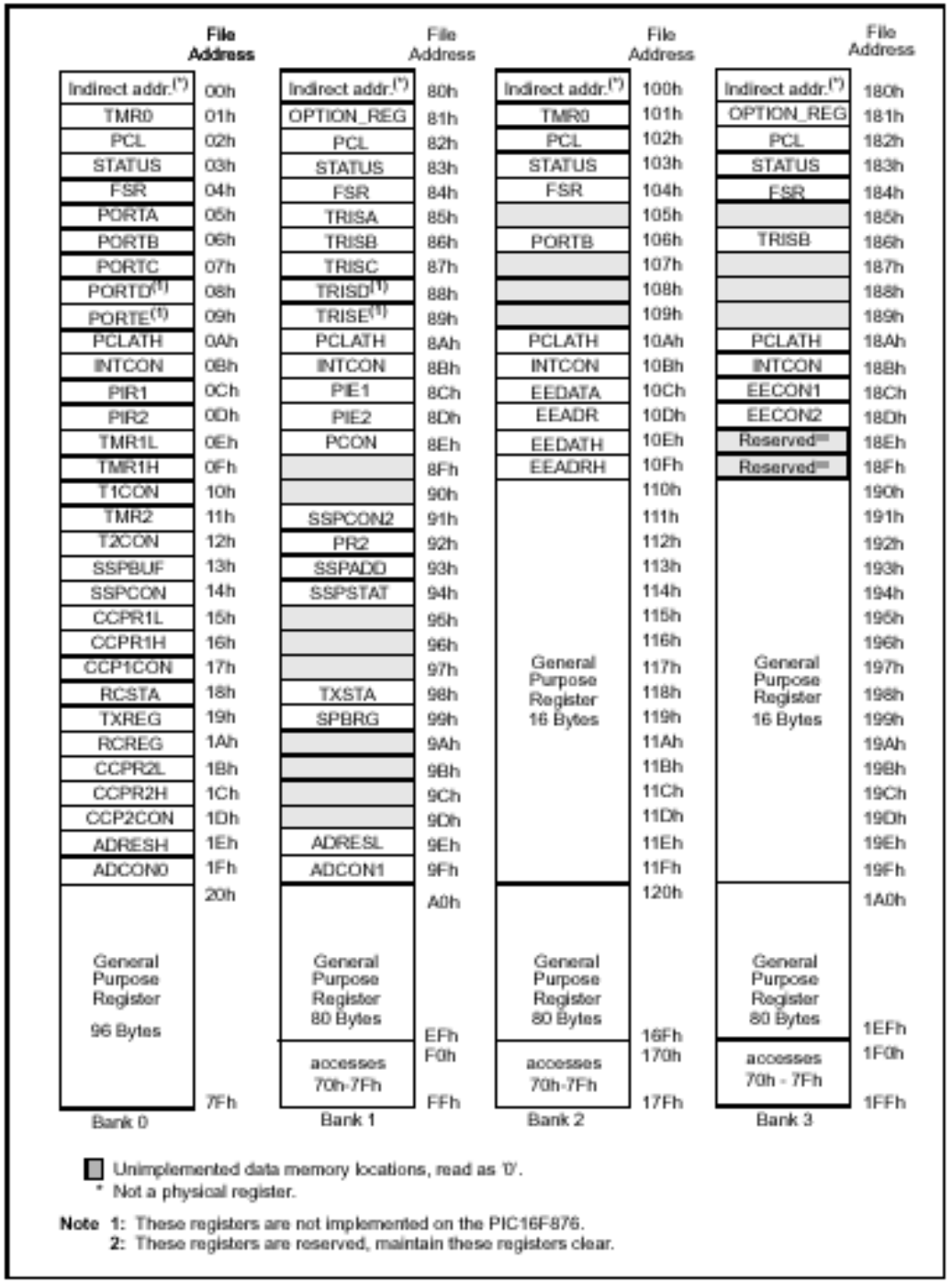
- dirección (13 bits = 8K)
- datos (14 bits = instrucción)

Para la memoria RAM/REGISTROS

- dirección (9 bits = 512K o bien
  2bits (4 bancos) + 7 bits (posición dentro del banco) = 4 bancos de 128)
- datos (8 bits = datos propiamente)

PIC16F876:
REGISTROS ESPECIALES DENTRO DE LA MEMORIA DE DATOS

FIGURE 2-3: PIC16F877/876 REGISTER FILE MAP



Unimplemented data memory locations, read as '0'.
* Not a physical register.

Note 1: These registers are not implemented on the PIC16F876.
2: These registers are reserved, maintain these registers clear.

PIC16F876: REGISTROS ESPECIALES DENTRO DE LA MEMORIA DE DATOS

En el banco 0

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	27
01h	TMR0	Timer0 Module Register								xxxx xxxx	47
02h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	26
03h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
04h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	29
06h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	31
07h	PORTC	PORTC Data Latch when written: PORTC pins when read								xxxx xxxx	33
08h(4)	PORTD	PORTD Data Latch when written: PORTD pins when read								xxxx xxxx	35
09h(4)	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	36
0Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
0Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
0Ch	PIR1	PSPIF(3)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	22
0Dh	PIR2	—	(5)	—	EEIF	BCLIF	—	—	CCP2IF	-r-0 0--0	24
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	52
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	52
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	51
11h	TMR2	Timer2 Module Register								0000 0000	55
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	55
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	70, 73
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	67
15h	CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx	57
16h	CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx	57
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	58
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	96
19h	TXREG	USART Transmit Data Register								0000 0000	99
1Ah	RCREG	USART Receive Data Register								0000 0000	101
1Bh	CCPR2L	Capture/Compare/PWM Register2 (LSB)								xxxx xxxx	57
1Ch	CCPR2H	Capture/Compare/PWM Register2 (MSB)								xxxx xxxx	57
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	58
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	116
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	$\overline{GO/DONE}$	—	ADON	0000 00-0	111

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 PIC16F873/876 devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD, and TRISE are not physically implemented on PIC16F873/876 devices; read as '0'.

5: PIR2<6> and PIE2<6> are reserved on these devices; always maintain these bits clear.

PIC16F876: REGISTROS ESPECIALES DENTRO DE LA MEMORIA DE DATOS

En el banco 1

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	27
81h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	19
82h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	26
83h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
84h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	29
86h	TRISB	PORTB Data Direction Register								1111 1111	31
87h	TRISC	PORTC Data Direction Register								1111 1111	33
88h(4)	TRISD	PORTD Data Direction Register								1111 1111	35
89h(4)	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	37
8Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
8Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
8Ch	PIE1	PSPIE(2)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	21
8Dh	PIE2	—	(5)	—	EEIE	BCLIE	—	—	CCP2IE	-r-0 0--0	23
8Eh	PCON	—	—	—	—	—	—	$\overline{POR}$	$\overline{BOR}$	---- --qq	25
8Fh	—	Unimplemented								—	—
90h	—	Unimplemented								—	—
91h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	68
92h	PR2	Timer2 Period Register								1111 1111	55
93h	SSPADD	Synchronous Serial Port (I^2C mode) Address Register								0000 0000	73, 74
94h	SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	66
95h	—	Unimplemented								—	—
96h	—	Unimplemented								—	—
97h	—	Unimplemented								—	—
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	95
99h	SPBRG	Baud Rate Generator Register								0000 0000	97
9Ah	—	Unimplemented								—	—
9Bh	—	Unimplemented								—	—
9Ch	—	Unimplemented								—	—
9Dh	—	Unimplemented								—	—
9Eh	ADRESL	A/D Result Register Low Byte								xxxx xxxx	116
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	0--- 0000	112

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 PIC16F873/876 devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD, and TRISE are not physically implemented on PIC16F873/876 devices; read as '0'.

5: PIR2<6> and PIE2<6> are reserved on these devices; always maintain these bits clear.

PIC16F876: REGISTROS ESPECIALES DENTRO DE LA MEMORIA DE DATOS

En el banco 2

En el banco 3

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	27
101h	TMR0	Timer0 Module Register								xxxx xxxx	47
102h(3)	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	26
103h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
104h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
105h	—	Unimplemented								—	—
106h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	31
107h	—	Unimplemented								—	—
108h	—	Unimplemented								—	—
109h	—	Unimplemented								—	—
10Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
10Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
10Ch	EEDATA	EEPROM Data Register Low Byte								xxxx xxxx	41
10Dh	EEADR	EEPROM Address Register Low Byte								xxxx xxxx	41
10Eh	EEDATH	—	—	EEPROM Data Register High Byte						xxxx xxxx	41
10Fh	EEADRH	—	—	—	EEPROM Address Register High Byte					xxxx xxxx	41
Bank 3											
180h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	19
182h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	26
183h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
184h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
185h	—	Unimplemented								—	—
186h	TRISB	PORTB Data Direction Register								1111 1111	31
187h	—	Unimplemented								—	—
188h	—	Unimplemented								—	—
189h	—	Unimplemented								—	—
18Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
18Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
18Ch	EECON1	EEPGD	—	—	—	WRERR	WREN	WR	RD	x--- x000	41, 42
18Dh	EECON2	EEPROM Control Register2 (not a physical register)								---- ----	41
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 PIC16F873/876 devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD, and TRISE are not physically implemented on PIC16F873/876 devices; read as '0'.

5: PIR2<6> and PIE2<6> are reserved on these devices; always maintain these bits clear.

Registro de estado: STATUS

STATUS = 03

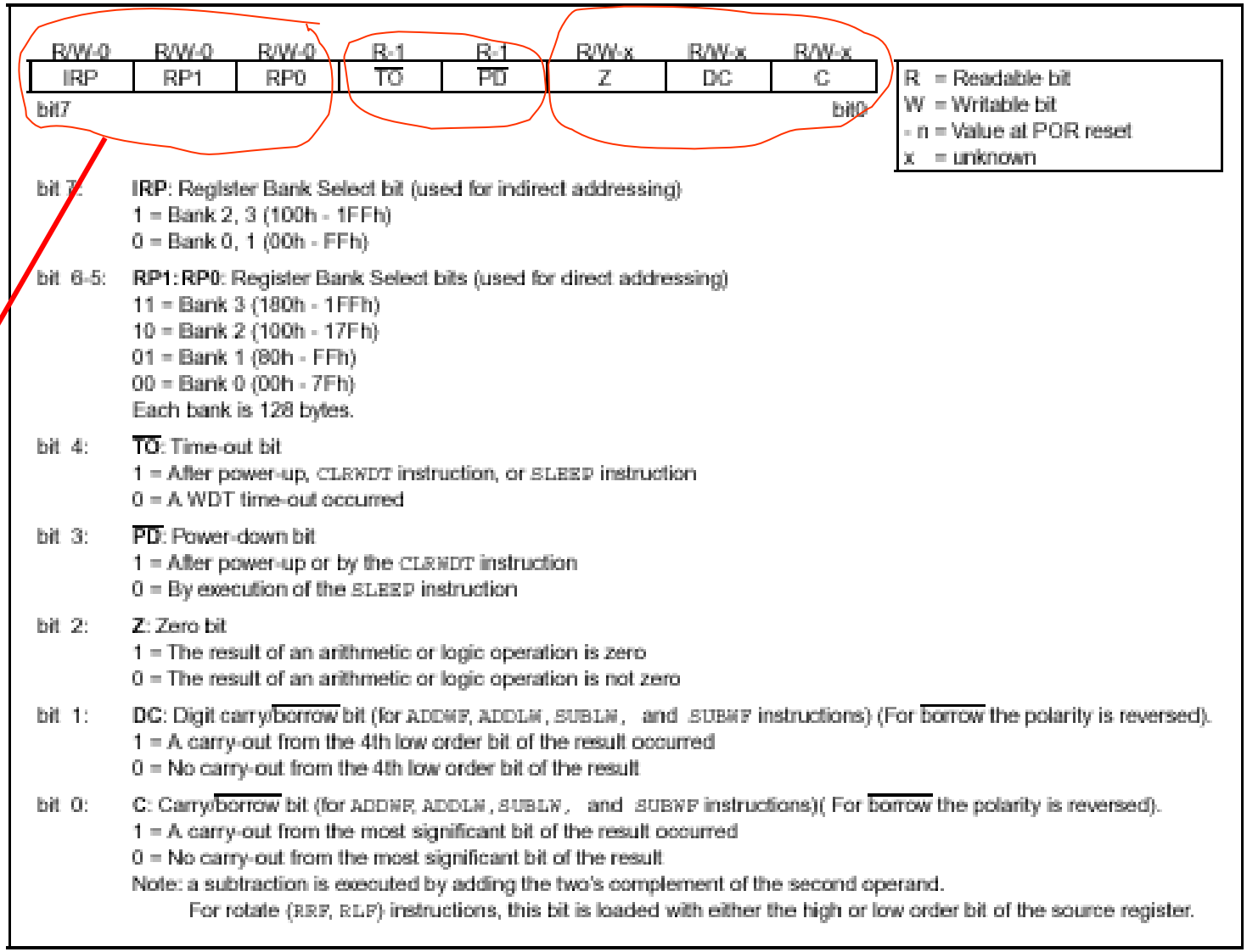
3 bits indican estado de la ALU.

2 bits relacionados con WATCHDOG y SLEEP

3 bits para seleccionar bancos de memoria RAM

Recordar el uso de FSR para direccionamiento indirecto.

FSR = 04

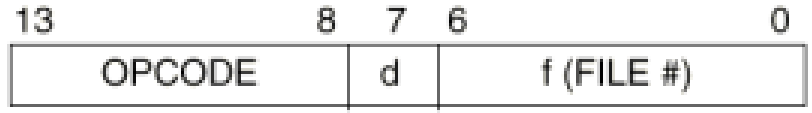


JUEGO DE INSTRUCCIONES

Cada instrucción se codifica en 14 bits.
(CÓDIGO MÁQUINA)

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1 .
PC	Program Counter
TO	Time-out bit
PD	Power-down bit

Byte-oriented file register operations



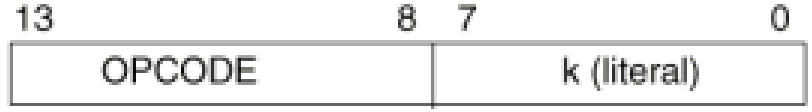
d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address

Bit-oriented file register operations

b = 3-bit bit address
f = 7-bit file register address

Literal and control operations

General



k = 8-bit immediate value

CALL and GOTO instructions only

k = 11-bit immediate value

INSTRUCCIONES

- 35 instrucciones

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	

sintaxis para los datos

Type	Syntax	Example
Decimal	D'<digits>'	D'100'
	.<digits>	.100
Hexadecimal	H'<hex_digits>'	H'9f'
	0x<hex_digits>	0x9f
Octal	O'<octal_digits>'	O'777'
Binary	B'<binary_digits>'	B'00111001'
ASCII	A'<character>'	A'C'
	'<character>'	'C'

Instrucciones para mover datos

MOVLW Move Literal to W

Syntax:	[label] MOVLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (W)
Status Affected:	None
Encoding:	`11 00xx kkkk kkkk`
Description:	The eight bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.
Words:	1
Cycles:	1
Example	`MOVLW 0x5A` After Instruction W = 0x5A

MOVF Move f

Syntax:	[label] MOVF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) → (destination)
Status Affected:	Z
Encoding:	`00 1000 dfff ffff`
Description:	The contents of register f is moved to a destination dependant upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file regis-ter since status flag Z is affected.
Words:	1
Cycles:	1
Example	`MOVF FSR, 0` After Instruction W = value in FSR register Z = 1

MOVWF Move W to f

Syntax:	[label] MOVWF f
Operands:	0 ≤ f ≤ 127
Operation:	(W) → (f)
Status Affected:	None
Encoding:	`00 0000 1fff ffff`
Description:	Move data from W register to register 'f'.
Words:	1
Cycles:	1
Example	`MOVWF OPTION_REG` Before Instruction OPTION = 0xFF W = 0x4F After Instruction OPTION = 0x4F W = 0x4F

Instrucciones para sumar

ADDLW	Add Literal and W
Syntax:	[label] ADDLW k
Operands:	$0 \le k \le 255$
Operation:	$(W) + k \rightarrow (W)$
Status Affected:	C, DC, Z
Encoding:	`11 111x kkkk kkkk`
Description:	The contents of the W register are added to the eight bit literal 'k' and the result is placed in the W register.
Words:	1
Cycles:	1
Example:	`ADDLW 0x15` Before Instruction W = 0x10 After Instruction **W = 0x25**

ADDWF	Add W and f
Syntax:	[label] ADDWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	$(W) + (f) \rightarrow$ (destination)
Status Affected:	C, DC, Z
Encoding:	`00 0111 dfff ffff`
Description:	Add the contents of the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1
Example	`ADDWF FSR, 0` Before Instruction W = 0x17 FSR = 0xC2 After Instruction W = 0xD9 **FSR = 0xC2**

Instrucciones para restar

SUBLW — Subtract W from Literal

Syntax: [label] SUBLW k
Operands: 0 ≤ k ≤ 255
Operation: k - (W) → (W)
Status Affected: C, DC, Z
Encoding: `11 110x kkkk kkkk`
Description: The W register is subtracted (2's complement method) from the eight bit literal 'k'. The result is placed in the W register.
Words: 1
Cycles: 1

Example 1: `SUBLW 0x02`

Before Instruction
W =1
C =?
Z =?
After Instruction
W =1
C = 1; result is positive
Z =0

Example 2: Before Instruction
W =2
C =?
Z =?
After Instruction
W =0
C = 1; result is zero
Z =1

Example 3: Before Instruction
W =3
C =?
Z =?
After Instruction
W = 0xFF
C = 0; result is negative
Z =0

SUBWF — Subtract W from f

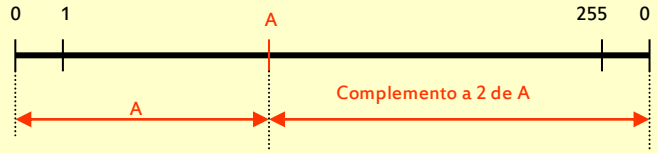
Syntax: [label] SUBWF f,d
Operands: 0 ≤ f ≤ 127
d ∈ [0,1]
Operation: (f) - (W) → (destination)
Status Affected: C, DC, Z
Encoding: `00 0010 dfff ffff`
Description: Subtract (2's complement method) W register from register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words: 1
Cycles: 1

Example 1: `SUBWF REG1,1`
Before Instruction
REG1 = 3
W =2
C =?
Z =?
After Instruction
REG1 = 1
W =2
C = 1; result is positive
Z =0

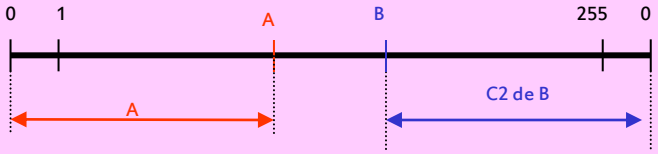
Example 2: Before Instruction
REG1 = 2
W =2
C =?
Z =?
After Instruction
REG1 = 0
W =2
C = 1; result is zero
Z =1

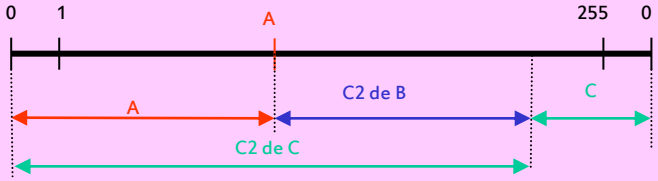
Example 3: Before Instruction
REG1 = 1
W =2
C =?
Z =?
After Instruction
REG1 = 0xFF
W =2 and C = 0; result is negative
Z =0

INTERPRETACIÓN GEOMÉTRICA DE LA RESTA EN COMPLEMENTO A 2 EN 8 BITS (Lopera 2003)



$C = A - B = A + (-B)$





"NOTAR: Si Carry=0 la resta es negativa
B > A"

Instrucciones para incrementar

INCF	Increment f
Syntax:	[label] INCF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) + 1 → (destination)
Status Affected:	Z
Encoding:	`00 1010 dfff ffff`
Description:	The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
Words:	1
Cycles:	1
Example	`INCF CNT, 1` **Before Instruction** CNT = 0xFF Z =0 **After Instruction** CNT = 0x00 Z =1

INCFSZ	Increment f, Skip if 0
Syntax:	[label] INCFSZ f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) + 1 → (destination), skip if result = 0
Status Affected:	None
Encoding:	`00 1111 dfff ffff`
Description:	The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'. If the result is 1, the next instruction is executed. If the result is 0, a NOP is executed instead making it a 2TCY instruction.
Words:	1
Cycles:	1(2)

Example	`HERE  INCFSZ CNT, 1`
	`      GOTO  LOOP`
	`CONT  *`
	`      *`
	`      *`
	Before Instruction
	PC = address `HERE`
	After Instruction
	CNT = CNT + 1
	if CNT= 0,
	PC = address `CONTINUE`
	if CNT≠ 0,
	PC = address `HERE +1`

Instrucciones para decrementar

DECF	Decrement f
Syntax:	[label] DECF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) - 1 → (destination)
Status Affected:	Z
Encoding:	`00 0011 dfff ffff`
Description:	Decrement register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1
Example	`DECF CNT, 1` **Before Instruction** CNT = 0x01 Z =0 **After Instruction** CNT = 0x00 Z =1

DECFSZ	Decrement f, Skip if 0
Syntax:	[label] DECFSZ f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) - 1 → (destination); skip if result = 0
Status Affected:	None
Encoding:	`00 1011 dfff ffff`
Description:	The contents of register 'f' are decre-mented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'. If the result is 1, the next instruction, is executed. If the result is 0, then a NOP is executed instead making it a 2TCY instruc-tion.
Words:	1
Cycles:	1(2)

Example	`LOOP  DECFSZ CNT,1`
	`  GOTO  LOOP`
	`CONTIN  *`
	`  *`
	`  *`
	Before Instruction
	PC = address `LOOP`
	After Instruction
	CNT = CNT - 1
	if CNT = 0,
	PC = address `CONTINUE`
	if CNT ≠ 0,
	PC = address `LOOP+1`

Instrucciones especiales

NOP	No Operation
Syntax:	[label] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Encoding:	00 0000 0xx0 0000
Description:	No operation.
Words:	1
Cycles:	1
Example	NOP

CLRWDT	Clear Watchdog Timer
Syntax:	[label] CLRWDT
Operands:	None
Operation:	00h → WDT 0 → WDT prescaler, 1 → TO 1 → PD
Status Affected:	TO, PD
Encoding:	00 0000 0110 0100
Description:	CLRWDT instruction resets the Watch-dog Timer. It also resets the prescaler of the WDT. Status bits TO and PD are set.
Words:	1
Cycles:	1
Example	CLRWDT **Before Instruction** WDT counter = ? **After Instruction** WDT counter = 0x00 WDT prescaler= 0 **TO** =1 **PD** =1

SLEEP

Syntax: [label] SLEEP

Operands: None

Operation: 00h → WDT,
0 → WDT prescaler,
1 → TO,
0 → PD

Status Affected: TO, PD

Encoding: 00 0000 0110 0011

Description: The power-down status bit, PD is cleared. Time-out status bit, TO is set. Watchdog Timer and its pres-caler are cleared.
The processor is put into SLEEP mode with the oscillator stopped. See Section 13.8 for more details.

Words: 1

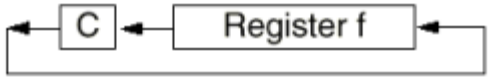
Cycles: 1

Q Cycle Activity: Q1 Q2 Q3 Q4
Decode No-Operation No-Operation Go to Sleep

Example: SLEEP

Instrucciones para rotar datos

RLF	Rotate Left f through Carry
Syntax:	[label] RLF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	See description below
Status Affected:	C
Encoding:	00 1101 dfff ffff
Description:	The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1

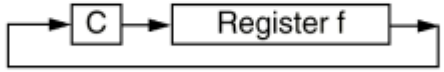


Example

```
RLF REG1,0
```

Before Instruction
REG1 = 1110 0110
C =0
After Instruction
REG1 = 1110 0110
W = 1100 1100
C =1
Register f C

RRF	Rotate Right f through Carry
Syntax:	[label] RRF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	See description below
Status Affected:	C
Encoding:	00 1100 dfff ffff
Description:	The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
Words:	1
Cycles:	1



Example

```
RRF REG1,0
```

Before Instruction
REG1 = 1110 0110
C =0
After Instruction
REG1 = 1110 0110
W = 0111 0011
C =0
Register f C

Instrucciones para rotar datos (intercambiar nibbles)

SWAPF	Swap Nibbles in f
Syntax:	[label] SWAPF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f<3:0>) → (destination<7:4>), (f<7:4>) → (destination<3:0>)
Status Affected:	None
Encoding:	`00 1110 dfff ffff`
Description:	The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0 the result is placed in W register. If 'd' is 1 the result is placed in register 'f'.
Words:	1
Cycles:	1
Example	`SWAPF REG, 0` Before Instruction REG1 = 0xA5 After Instruction REG1 = 0xA5 W = 0x5A

Curiosidad:

SWAPF REGISTRO,1
SWAPF REGISTRO,0

Recupera el registro en cuestión en el W sin afectar a los flag (es decir si tocar el STATUS).

Es útil para los retorno de interrupción donde se salva el STATUS y el W (mirar el template F876TEMP.ASM).

Instrucciones lógicas (AND)

ANDLW	AND Literal with W
Syntax:	[label] ANDLW k
Operands:	$0 \le k \le 255$
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Encoding:	`11 1001 kkkk kkkk`
Description:	The contents of W register are AND'ed with the eight bit literal 'k'. The result is placed in the W register.
Words:	1
Cycles:	1
Example	`ANDLW 0x5F` Before Instruction W = 0xA3 After Instruction **W = 0x03**

ANDWF	AND W with f
Syntax:	[label] ANDWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W) .AND. (f) → (destination)
Status Affected:	Z
Encoding:	`00 0101 dfff ffff`
Description:	AND the W register with register 'f'. If 'd' is 0 the result is stored in the W regis-ter. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1
Example	`ANDWF FSR, 1` Before Instruction W = 0x17 FSR = 0xC2 After Instruction W = 0x17 FSR = 0x02

Instrucciones lógicas (OR)

IORLW	Inclusive OR Literal with W
Syntax:	[label] IORLW k
Operands:	$0 \leq k \leq 255$
Operation:	(W) .OR. k → (W)
Status Affected:	Z
Encoding:	`11 1000 kkkk kkkk`
Description:	The contents of the W register is OR'ed with the eight bit literal 'k'. The result is placed in the W register.
Words:	1
Cycles:	1
Example	`IORLW 0x35` **Before Instruction** W = 0x9A **After Instruction** W = 0xBF Z =1

IORWF	Inclusive OR W with f
Syntax:	[label] IORWF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	(W) .OR. (f) → (destination)
Status Affected:	Z
Encoding:	`00 0100 dfff ffff`
Description:	Inclusive OR the W register with regis-ter 'f'. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
Words:	1
Cycles:	1
Example	`IORWF RESULT, 0` **Before Instruction** RESULT = 0x13 W = 0x91 **After Instruction** RESULT = 0x13 W = 0x93 Z =1

XORLW	Exclusive OR Literal with W
Syntax:	[label] XORLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .XOR. k → (W)
Status Affected:	Z
Encoding:	11 1010 kkkk kkkk
Description:	The contents of the W register are XOR'ed with the eight bit literal 'k'. The result is placed in the W regis-ter.
Words:	1
Cycles:	1
Example:	XORLW 0xAF Before Instruction W = 0xB5 After Instruction W = 0x1A

XORWF	Exclusive OR W with f
Syntax:	[label] XORWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .XOR. (f) → (destination)
Status Affected:	Z
Encoding:	00 0110 dfff ffff
Description:	Exclusive OR the contents of the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1
Example	XORWF REG 1 Before Instruction REG = 0xAF W = 0xB5 After Instruction REG = 0x1A W = 0xB5

Instrucciones lógicas (complementar)

COMF	Complement f
Syntax:	[label] COMF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	(f) → (destination)
Status Affected:	Z
Encoding:	`00 1001 dfff ffff`
Description:	The contents of register 'f' are complemented. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1
Example	`COMF REG1,0` **Before Instruction** REG1 = 0x13 **After Instruction** REG1 = 0x13 W = 0xEC

Instrucciones para borrar (poner a cero)

CLRW	Clear W
Syntax:	[label] CLRW
Operands:	None
Operation:	00h → (W) 1 → Z
Status Affected:	Z
Encoding:	`00 0001 0xxx xxxx`
Description:	W register is cleared. Zero bit (Z) is set.
Words:	1
Cycles:	1
Example	`CLRW` **Before Instruction** W = 0x5A **After Instruction** W = 0x00 Z =1

CLRF	Clear f
Syntax:	[label] CLRF f
Operands:	0 ≤ f ≤ 127
Operation:	00h → (f) 1 → Z
Status Affected:	Z
Encoding:	`00 0001 1fff ffff`
Description:	The contents of register 'f' are cleared and the Z bit is set.
Words:	1
Cycles:	1
Example	`CLRF FLAG_REG` **Before Instruction** FLAG_REG = 0x5A **After Instruction** FLAG_REG = 0x00 Z =1

Instrucciones manipular bits (poner a cero y a uno)

BCF	Bit Clear f
Syntax:	[label] BCF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	0 → (f<b>)
Status Affected:	None
Encoding:	01 00bb bfff ffff
Description:	Bit 'b' in register 'f' is cleared.
Words:	1
Cycles:	1
Example	BCF FLAG_REG, 7 Before Instruction FLAG_REG = 0xC7 After Instruction FLAG_REG = 0x47

BSF	Bit Set f
Syntax:	[label] BSF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	1 → (f<b>)
Status Affected:	None
Encoding:	01 01bb bfff ffff
Description:	Bit 'b' in register 'f' is set.
Words:	1
Cycles:	1
Example	BSF FLAG_REG, 7 Before Instruction FLAG_REG = 0x0A After Instruction FLAG_REG = 0x8A

Instrucciones para saltar

GOTO	Unconditional Branch
Syntax:	[label] GOTO k
Operands:	$0 \le k \le 2047$
Operation:	$k \rightarrow PC<10:0>$ $PCLATH<4:3> \rightarrow PC<12:11>$
Status Affected:	None
Encoding:	10 1kkk kkkk kkkk
Description:	GOTO is an unconditional branch. The eleven bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a two cycle instruction.
Words:	1
Cycles:	2
Example	GOTO THERE After Instruction PC = Address THERE

PC = 13 bits (8 K)

En este salto solo ponemos 11 bits (2K)

¡ Cuidado con los saltos de mas de 2K !

Debemos cargar el PCLATH con el valor adecuado.

Instrucciones para saltar (condicionado al estado de un bit)

BTFSC — **Bit Test, Skip if Clear**

Syntax: [label] BTFSC f,b

Operands: $0 \le f \le 127$
$0 \le b \le 7$

Operation: skip if (f<b>) = 0

Status Affected: None

Encoding: `01 10bb bfff ffff`

Description: If bit 'b' in register 'f' is '1' then the next instruction is executed.
If bit 'b', in register 'f', is '0' then the next instruction is discarded, and a NOP is executed instead, making this a 2TCY instruction.

Words: 1

Cycles: 1(2)

Example

```
HERE    BTFSC   FLAG,1
FALSE   GOTO    PROCESS_CODE
TRUE    *
        *
        *
```

Before Instruction
PC = address HERE
After Instruction
if FLAG<1> = 0,
PC = address TRUE
if FLAG<1>=1,
PC = address FALSE

BTFSS — **Bit Test f, Skip if Set**

Syntax: [label] BTFSS f,b

Operands: $0 \le f \le 127$
$0 \le b < 7$

Operation: skip if (f<b>) = 1

Status Affected: None

Encoding: `01 11bb bfff ffff`

Description: If bit 'b' in register 'f' is '0' then the next instruction is executed.
If bit 'b' is '1', then the next instruction is discarded and a NOP is executed instead, making this a 2TCY instruction.

Words: 1

Cycles: 1(2)

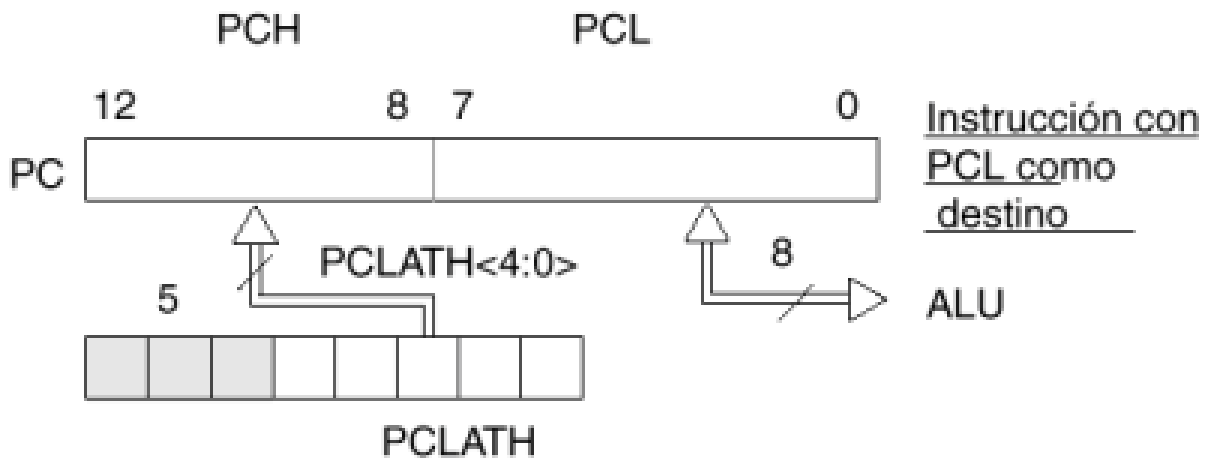
Example

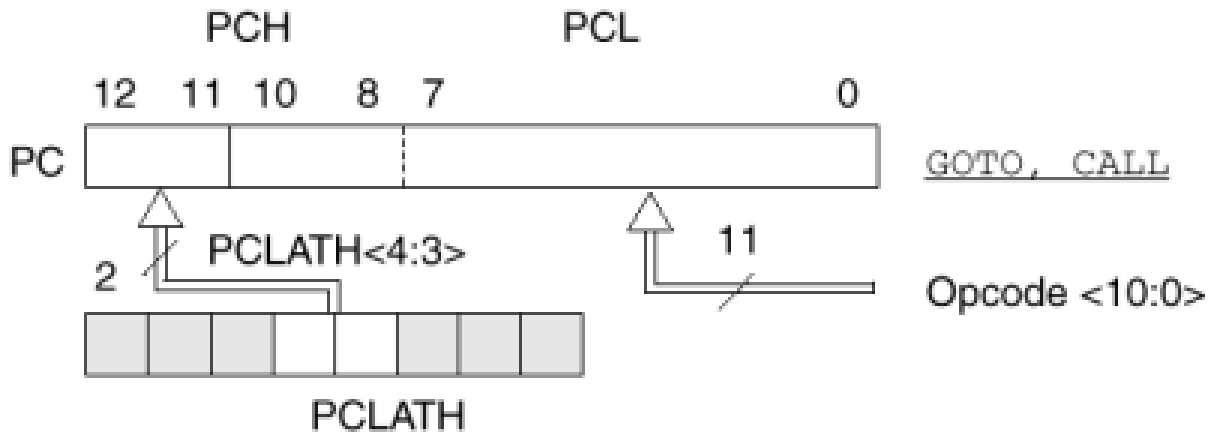
```
HERE    BTFSC   FLAG,1
FALSE   GOTO    PROCESS_CODE
TRUE    *
        *
        *
```

Before Instruction
PC = address HERE
After Instruction
if FLAG<1> = 0,
PC = address FALSE
if FLAG<1> = 1,
PC = address TRUE

Registros de control del contador de programa: PCL y PCLATH

PCLATH = 0A
PCL = 02





"Escribir en PCLATH no tiene efecto inmediato sobre el PC.

El valor de PCLATH solo se transfiere al modificar el registro PCL".

Ver manejo de tablas AN556

"Las instrucciones de salto GOTO y CALL reservan 11 bits dentro del código para el salto"

Instrucciones para hacer subprogramas (salto)

CALL	Call Subroutine
Syntax:	[label] CALL k
Operands:	$0 \leq k \leq 2047$
Operation:	(PC)+ 1→ TOS, k → PC<10:0>, (PCLATH<4:3>) → PC<12:11>
Status Affected:	None
Encoding:	10 0kkk kkkk kkkk
Description:	Call Subroutine. First, return address (PC+1) is pushed onto the stack. The eleven bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a two cycle instruction.
Words:	1
Cycles:	2
Example	HERE CALL THERE **Before Instruction** PC = Address HERE **After Instruction** PC = Address THERE TOS = Address HERE+1

PC = 13 bits (8 K)

En este salto solo ponemos 11 bits (2K)

¡ Cuidado con los saltos de mas de 2K !

Debemos cargar el PCLATH con el valor adecuado.

Igual que en la instrucción GOTO

Instrucciones para hacer subprogramas (retornos)

RETURN	Return from Subroutine
Syntax:	[label] RETURN
Operands:	None
Operation:	TOS → PC
Status Affected:	None
Encoding:	00 0000 0000 1000
Description:	Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two cycle instruction.
Words:	1
Cycles:	2
Example	RETURN After Interrupt PC = TOS

Comentario

RETLW es útil para manejar tablas de datos grabadas en la memoria de programa.

Ver nota de aplicación AN556

RETLW	Return with Literal in W
Syntax:	[label] RETLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (W); TOS → PC
Status Affected:	None
Encoding:	11 01xx kkkk kkkk
Description:	The W register is loaded with the eight bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two cycle instruction.
Words:	1
Cycles:	2

Example

```
        CALL TABLE ;W contains table
                   ;offset value
                   ;W now has table value
        *
        *
TABLE   ADDWF PC ;W = offset
        RETLW k1 ;Begin table
        RETLW k2 ;
        *
        *
        *
        RETLW kn ; End of table
```

Before Instruction
W = 0x07
After Instruction
W = value of k8

Instrucciones para retornar de subprogramas de interrupciones

RETFIE	Return from Interrupt
Syntax:	[label] RETFIE
Operands:	None
Operation:	TOS → PC, 1 → GIE
Status Affected:	None
Encoding:	00 0000 0000 1001
Description:	Return from Interrupt. Stack is POPed and Top of Stack (TOS) is loaded in the PC. Interrupts are enabled by setting Global Interrupt Enable bit, GIE (INTCON<7>). This is a two cycle instruction.
Words:	1
Cycles:	2
Example	RETFIE After Interrupt PC = TOS GIE = 1

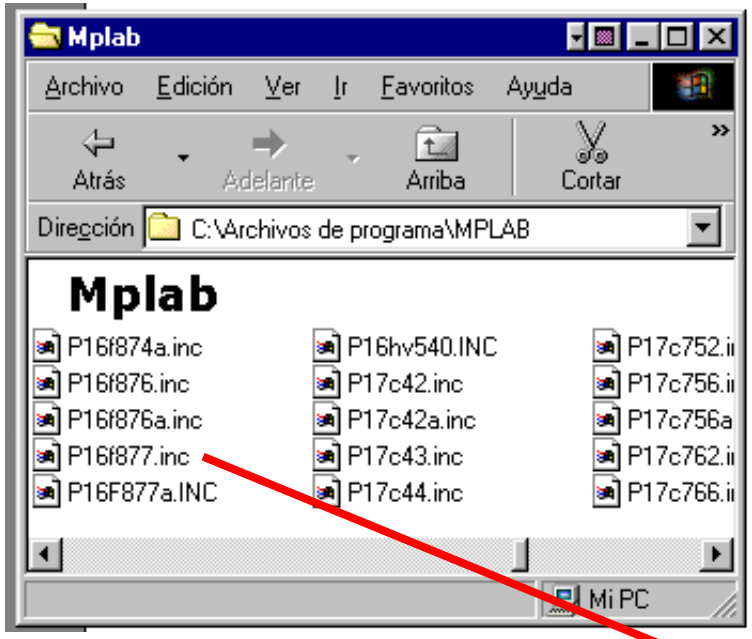
Podemos considerar una interrupción como una llamada a un subprograma generada por Hardware.

Tiene el matiz, que no sabemos el sitio donde se va a producir. Normalmente necesitaremos salvar el registro W y el STATUS y recuperarlos antes de retornar.

COMENTARIO

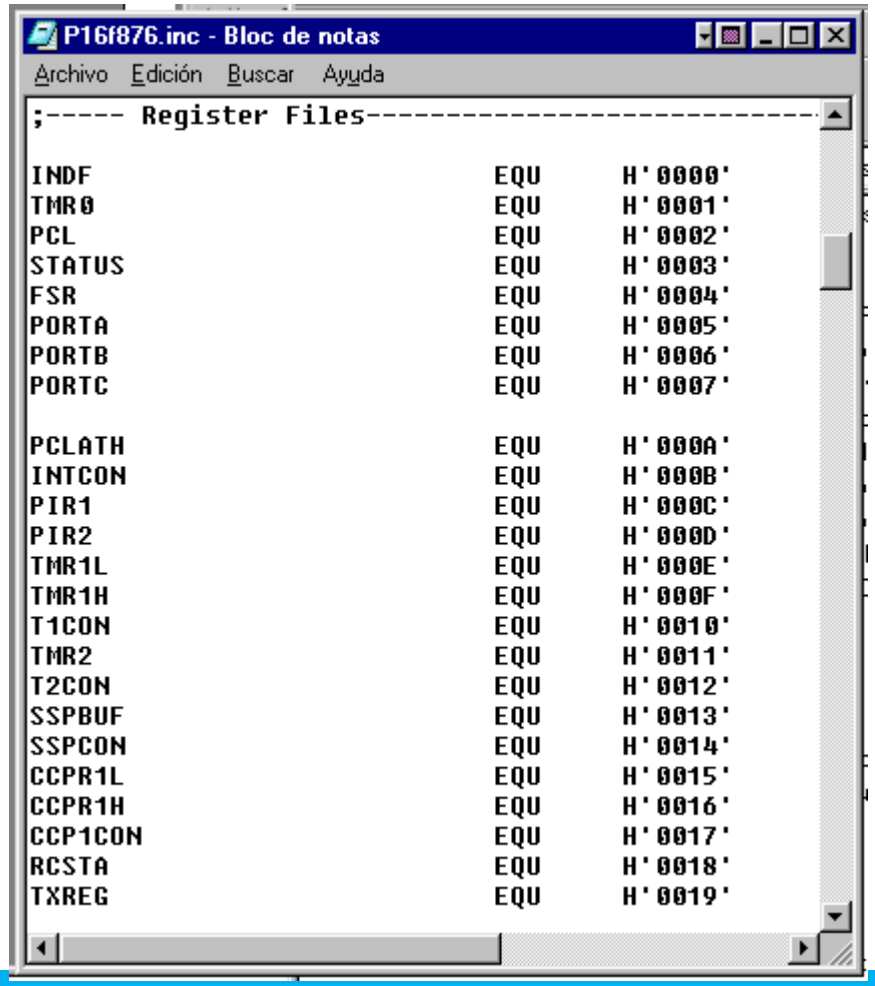
Como norma general, antes de retornar de una interrupción debemos borrar el bit que la ha generado.

Ver la parte de la presentación dedicada a las interrupciones.



Fichero P16F877.INC en el directorio MPLAB

Tiene definidas todas la etiquetas de registros para realizar el programa de forma cómoda.



NOTA:

El fichero F877TEMP.ASM es un fichero patrón (template) que sirve de ayuda para realizar programas en ensamblador. Esta en el directorio MPLAB también.