Universidad de Valladolid

Curso de Microcontroladores PIC

Dr. D. Jesús M. Hernández Mangas
Departamento de Electricidad y Electrónica

Microcontroladores PIC

Introducción

1. **Microcontrolador versus microprocesador**
2. **Arquitectura Hardvard versus arquitectura Von Neumann**
3. **RISC, CISC, SISC**
4. **Aplicaciones de los microcontroladores:** Control en tiempo real. Sistemas en tiempo real. Etapas de diseño.
5. **Consideraciones prácticas:** Alimentación. Reset.

Introducción : *Microcontrolador versus Microprocesador*

Microprocesador

Dispositivo programable de propósito general que consta de una unidad central de proceso y de unos registros.

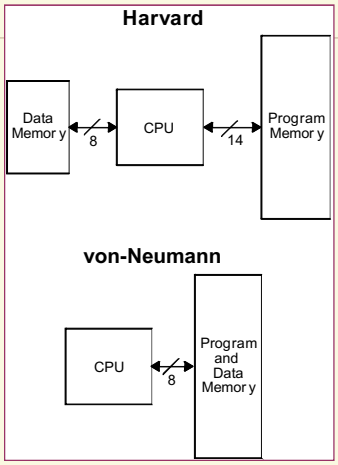
Se utilizan en computadoras. Gran capacidad de gestión de memoria y de proceso de datos. Trabajo *off-line.*

Microcontrolador

Dispositivo programable orientado a control. Tiempo de respuesta inmediato: tiempo real. Consta de CPU y registros, además de disponer de periféricos integrados (E/S, temporizadores/contadores, puertos de comunicaciones, etc.).

Se emplean en control de procesos *on-line*. Gestión de tiempos incorporada, atención de eventos. Periféricos integrados adaptados a la tarea específica.

Introducción : *Arquitectura Hardvard versus Von Neumann*



Arquitectura Von Neumann

- Memoria común para instrucciones y datos.
- Ahorro en el coste y en la complejidad del diseño.

Arquitectura Hardvard

- Memorias diferentes para datos e instrucciones.
- Direccionamiento con buses distintos.
- Más rápido.
- Lee instrucción y operando a la vez.
- Existen variaciones como la *Hardvard modificada* (la memoria de datos puede guardar programas) y la *Hardvard mejorada* (varias memorias de datos que pueden guardar programas).

Introducción : *RISC, CISC, SISC*

CISC (Computación con un Conjunto Complejo de Instrucciones)

- Conjunto de instrucciones amplio.
- Facilita la programación.

RISC (Computación con un Conjunto Reducido de Instrucciones)

- Conjunto reducido de instrucciones.
- Menor tamaño, menos pines, menor consumo.
- Más complejo de programar.

SISC (Computación con un Conjunto Específico de Instrucciones)

- Instrucciones para agilizar la E/S.
- Idem para manejo de interrupciones.
- Idem para manipulación de bits.

Introducción : *Aplicaciones de los microcontroladores*

Proporcionan inteligencia a los sistemas electrónicos que nos rodean

Instrumentos portátiles

•Polímetro, medidor ultrasónico de distancias, balanza electrónica.

Subfunciones de instrumentos

•Pantallas táctiles, teclado, ratón, display LCD, On Screen Display (OSD).

Dispositivos periféricos

•Impresora láser o de tinta, modems, plotters.

Dispositivos autónomos

•Fotocopiadora, vídeo doméstico, teléfonos.

Aplicaciones en automoción

•Inyección electrónica, frenos ABS, tarificación de Taxis, cuadro de instrumentos, GPS.

Introducción : *Aplicaciones de los microcontroladores*

Control en tiempo real. ¿Qué es respuesta en tiempo real?

Las acciones de salida deben estar disponibles en un intervalo de tiempo reducido y acotado desde que se tienen las entradas.

Sistemas en tiempo real: lavadora automática, control de presencia, instrumentación electrónica, robótica, satélites artificiales.

Sistemas no en tiempo real: control de nóminas, elaboración de estadísticas, simulaciones *off-line.*

No todos los sistemas en tiempo real son críticos:

Críticos: el incumplimiento del plazo temporal de respuesta provoca un fallo catastrófico en el sistema. Ej: satélite artificial, brazo de un robot, misil anti-misil.

No críticos: el incumplimiento de plazos no ocasiona, necesariamente, un fallo en el sistema. Ej: cafetera, semáforo, instrumento de medida del laboratorio.

Por supuesto, todo es relativo.

Introducción : *Aplicaciones de los microcontroladores*

Sistemas en tiempo real. Necesidades.

Estructura temporal

- Generadores de base de tiempos, contadores/temporizadores.

Entradas/salidas al exterior

- Analógicas y digitales; alta corriente. Reprogramables.

Capacidad de atención al entorno

- Interrupciones hardware o software.

Capacidad de comunicaciones

- Serie, paralelo; bucle de corriente, módem. Que permitan la interoperabilidad con otros equipos y que implementen los protocolos necesarios.

Introducción : *Aplicaciones de los microcontroladores*

Etapas de diseño

1. Conocimiento de los dispositivos aplicables y sus recursos: microcontroladores, microprocesadores, memorias, teclados, displays, sensores, actuadores, etc.

2. Conocimiento de los algoritmos típicos: máquinas de estado, teoría de autómatas, redes de Petri, sistemas operativos (en tiempo real).

3. Diseño del sistema digital: hardware y software. Proceso iterativo y recurrente.

En resumen, el sistema digital debe ser:

Lo más **compacto** posible, lo más **barato** posible y con un **tiempo de desarrollo** lo más **corto** posible.

Si cabe en un solo chip, y es más barato, y ya estaba hecho antes: ***¡MEJOR!***

Introducción : *Consideraciones prácticas*

En el diseño de un sistema en tiempo real : Alimentación

Alimentación

- Convencional 5V
- Baja tensión 3.3 V Equipos portátiles.

Prestaciones de la alimentación

- *Prevención de fallos catastróficos o pérdida de datos.*
- Detección de anomalías en el suministro: caídas de tensión, cortes esporádicos.

Sistemas con baterías

- *Aumentar autonomía; prevención de fallos en el sistema.*
- Bajo consumo. Autodesconexión en caso de inactividad.
- Detección de batería baja.

Introducción : *Consideraciones prácticas*

En el diseño de un sistema en tiempo real : Alimentación

Criterio de diseño

- ***Si el sistema puede fallar, fallará ...***

en el momento más inoportuno y causando el mayor daño posible.

Precauciones elementales

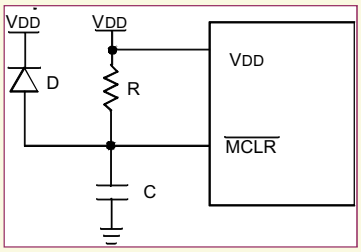
- Medios alternativos de alimentación. Detección de los fallos y actuar rápidamente.
- Los datos trascendentales del sistema (configuración, seguridad) deben almacenarse en memoria no volátil.
- El arranque del sistema debe iniciar correctamente las E/S.
- Que el programa no alcance nunca un estado indeterminado: ***Watch Dog*** (perro guardián).

Introducción : *Consideraciones prácticas*

En el diseño de un sistema en tiempo real : Reset

Reset. Los circuitos RC no proporcionan un reset fiable.

- Ideal un reset temporizado predecible.
- Supervisión continua del circuito de alimentación.
- Control adecuado del flujo de programa.
- Memoria estática que sea no volátil



La solución son los circuitos de supervisión de los microcontroladores: MC33164P, TL7705, MAX690, MAX691, etc..

Microcontroladores PIC

Características de los microcontroladores

1. **Memoria.**
2. **Boot Loader.**
3. **Protección.**
4. **Programación In System.**
5. **Periféricos:** Temporizadores. E/S paralela. E/S serie *síncrona* (I2C, SPI) o *asíncrona*. Conversores A/D. Conversores D/A. Conversores PWM. Controlador DMA.
6. **Watch-dog.**
7. **Reducción del consumo eléctrico.**
8. **Interrupciones.**

Características de los uC: *Memoria*

Memorias de los uC

Incorporan memorias, volátiles o no, de **programa** y de **datos.**

Memoria de datos:

- Debe ser de lectura/escritura.
- Volátil o no.
- No incorpora grandes cantidades ya que los registros están mapeados en ella.

Memoria de programa:

- Almacena las instrucciones del programa de control.
- Solo lectura.
- Puede ser interna o externa (sólo en algunos uC, necesita buses).

Características de los uC: *Memoria*

Memoria de datos

RAM (Random Access Memory). *Memoria de acceso aleatorio.*

- Almacena los datos del programa.
- Lectura/escritura.
- En algunos casos puede almacenar el programa (Arq. Von Neumann).
- Puede ser *dinámica* (externa, necesita refresco), o *estática* (típicamente interna, más rápida).
- No suele ser muy grande (hasta 2kb).

Bancos de registros.

- Mapeados en la memoria de datos.
- Algunos tienen funcionalidad específica (SFRs: *Specific Function Registers*).
- Otros de propósito general (GPRs: *General Purpose Registers*).
- Número de registros grande (en comparación con los uP)

Características de los uC: *Memoria*

Memoria de programa

ROM (Read Only Memory). *Memoria de solo lectura.*

- Almacena el programa de control. Se graba durante su fabricación.
- Coste de diseño elevado. Coste de producción muy barato.
- Tiempo de desarrollo elevado (hasta 44 semanas).

EPROM (Erasable Programmable ROM). *Memoria ROM borrable por UV.*

- Puede grabarse eléctricamente muchas veces.
- Se borra con rayos ultravioleta (tarda varios minutos).
- Interesante en fase de diseño y fabricación.

OTP (One Time Programmable). *Programable una vez.*

- Solo se graba una vez.
- Muy barata.
- Para producción en cantidades pequeñas.

Características de los uC: *Memoria*

Memoria de programa

EEPROM (Electrically Erasable Programmable ROM).
Memoria programable y borrable eléctricamente.

- Tiempo de borrado corto, aunque lento comparado con RAM.
- Hasta 1.000.000 de veces.
- Necesita una tensión de programación mayor que la de alimentación.

FLASH™. *Memoria EEPROM borrable eléctricamente por bloques.*

- Borra bloques enteros. Es menos compleja
- Su borrado es más rápido.
- Ideales para el diseño.

Características de los uC: *Boot Loader*

El **boot loader** es una prestación hardware de algunos microcontroladores que permite durante el *reset*, al activarse unos pines, que se cargue el programa de arranque del microcontrolador.

La carga puede ser:

- En paralelo, procedente de una memoria externa.
- A través del puerto serie.

Es interesante para poder cambiar el programa sin extraer el uC, Ideal cuando desarrollamos aplicaciones.

Características de los uC: *Programación In System*

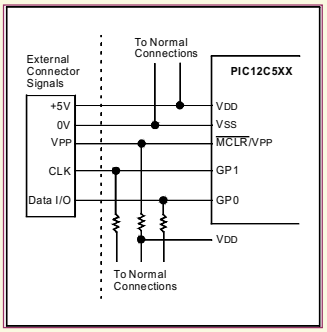
Los microcontroladores pueden grabarse:

- Con un protocolo paralelo.
- Con un protocolo serie.

Para reducir el tiempo de desarrollo, algunos fabricantes facilitan el que se pueda programar el uC si extraerlo del sistema final: programación **In System** o **In Circuit.**

Típicamente emplean una grabación serie:

- Reloj (transferencia serie síncrona).
- Datos.
- Alimentación, tierra y tensión de programación.



Características de los uC: *Protección software*

Protección del código

- La información grabada en algunos microcontroladores puede **protegerse**, **por hardware**, para evitar su lectura.
- Protege la *propiedad intelectual* y evita el *espionaje industrial*.
- Mediante unos bits de configuración se activa la protección de la información (al leerlo salen valores sin sentido, o todos los valores iguales).
- Si no se ha activado la protección, se puede leer el contenido del uC para verificarlo.
- Aún así, hay quien emplea la *ingeniería inversa.*

Características de los uC: *Periféricos*

TEMPORIZADORES

Función:

- Temporizan o cuentan eventos.

Características típicas:

- Número de bits (8, 16, etc.).
- Selección de reloj interno/externo.
- Selección de nivel/flanco (externo).
- Pre-escala programable.
- Generan interrupción al finalizar la cuenta.
- Se pueden modificar/consultar en cualquier momento.
- Generan salidas regulares para dispositivos externos.

Características de los uC: *Periféricos*

E/S PARALELA

Función:

- Acceso digital en paralelo a entradas y salidas.

Características típicas:

- Sentido: entrada, salida, bidireccional.
- Posibilidad de cambio de sentido en tiempo de ejecución.
- Capacidad para alimentar LEDs (mA que es capaz de generar o recibir por pin y en conjunto).
- Capacidad de generación de interrupciones al cambiar su estado.
- Colector/drenador abierto.
- Pull-up entrada programable.
- Entrada con disparador Schmitt.
- Multifuncionalidad: pines compartidos con otros periféricos internos.

Características de los uC: *Periféricos*

E/S SERIE

Función:

- Acceso digital en serie. Típicamente para comunicación con otros periféricos externos u otros uC.

Características típicas:

- Hardware específico para facilitar la transferencia serie o no.
- Comunicación serie síncrona: bus I2C, bus SPI, bus MicroWire, etc.
- Comunicación serie asíncrona RS-232C, USB, etc.
- Velocidad de transferencia.
- Comunicación punto a punto o bus.

Características de los uC: *Periféricos*

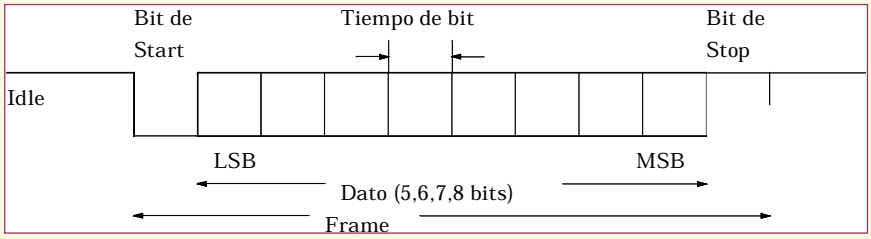
E/S SERIE ASíNCRONA

- Consiste en la **transmisión serie** de palabras de datos *multi-bit* en intervalos indeterminados donde cada palabra tiene un **número fijo de bits** y un **tiempo de bit constante**.
- Tanto emisor como receptor disponen de un reloj para medir el **tiempo de bit** (conocido por ambos), pero pueden no coincidir exactamente o estar desfasados.
- Es necesaria una **resincronización periódica**.
- El primer bit de cada dato será un **bit de sincronización**.
- Para finalizar la transferencia de un dato se señaliza con un **bit de stop**. Puede haber más de un bit de stop (1,1½,2)
- Se puede añadir un **bit de paridad** para comprobar si la transmisión fue correcta:

1. Paridad par (nº de unos transmitidos es par, incluida la paridad)
2. Paridad impar (nº de unos transmitidos es impar).

Características de los uC: *Periféricos*

E/S SERIE ASíNCRONA: Formato de la trama



La comunicación es **bidireccional** (dos líneas: TxD y RxD, y GROUND).

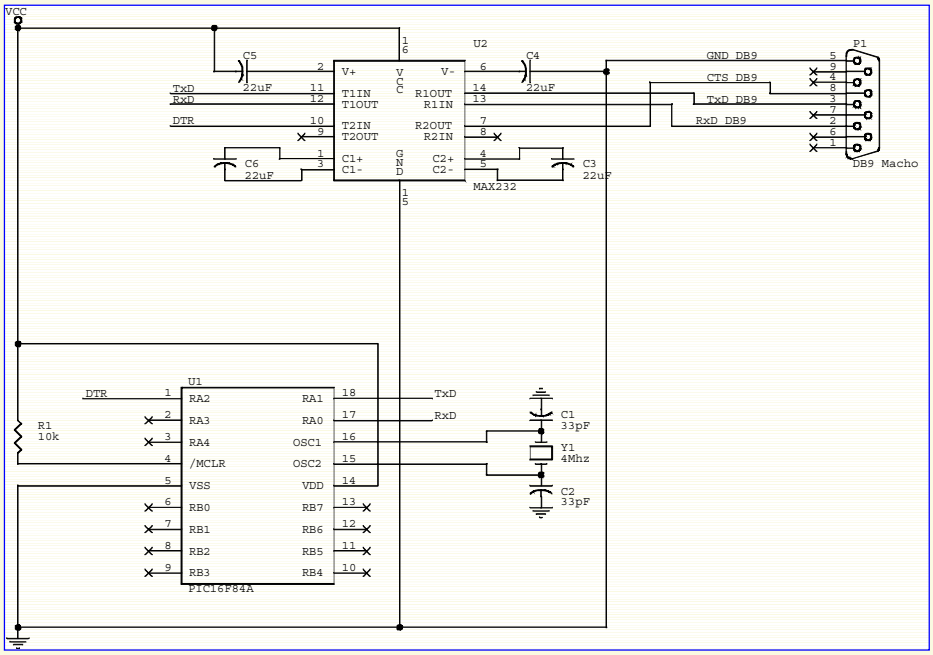
- **Simultanea** (*full duplex*).Control de flujo hardware (2 líneas extra) o software (Xon/Xoff).
- **No simultanea** (*half duplex*). Control de flujo hardware para decidir quien transmite, o configuración *maestro-esclavo.*

RS-232C

- Espacio, "0" lógico: +3 .. +25 V
- Marca, "1" lógico: -3.. -25 V

Características de los uC: *Periféricos*

E/S SERIE ASíNCRONA: Montaje típico

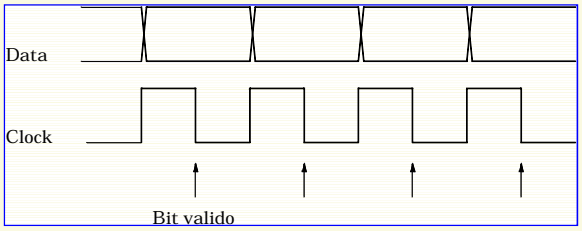


Características de los uC: *Periféricos*

E/S SERIE SíNCRONA

- Consiste en la **transmisión serie** de datos *multi-bit,* donde cada bit transmitido está asociado con un pulso de reloj transmitido por una línea de datos separada.
- Puede tener o no bits de marcaje de datos y pueden o no estar asociados al reloj.

Transmisión de bits:



Transmisión de datos:

Los bytes se reconocen contando bits (bus SPI, **longitud fija**) o pueden necesitar **bits de marcaje** especiales (bus I2C).

Características de los uC: *Periféricos*

E/S SERIE SíNCRONA

Utilidad:

- Más rápido que el método asíncrono (¿?), menor número de líneas, implementan buses.

Protocolo de bus:

- Hay un dispositivo *maestro* (uC) y los demás son *esclavos* (perif. o uCs)
- El *maestro* inicia y finaliza la comunicación (controla el bus).
- El *maestro* direcciona a los *esclavos* y genera la señal de reloj.
- Algunos protocolos permiten cambiar quién es el *maestro* (arbitraje del bus).

Periféricos soportados:

- Memorias serie EEPROM, RAM, etc.
- ADC y DAC.
- Displays, RTC (*Real Time Clocks*).

Características de los uC: *Periféricos*

E/S SERIE SíNCRONA

Estándares de comunicación serie síncrona:

•**Bus I2C** (Inter Integrated Circuit, Phillips)

Dos líneas (reloj:SCL y datos:SDA). Además de Vdd y GND. *Half duplex*. Direccionamiento software de los esclavos.

•**Bus SPI** (Serial Peripheral Interface, Motorola)

Tres líneas (entrada: MOSI, salida: MISO, reloj: SCK). Además de Vdd, GND y líneas de selección del esclavo. *Full duplex*. Longitud fija de datos.

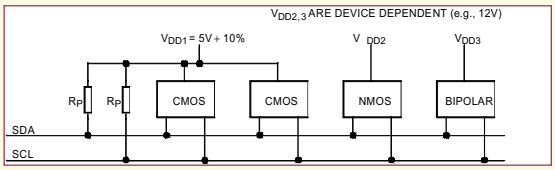
•**Bus MicroWire** (National Semiconductor)

Tres líneas y la selección del esclavo además de Vdd y GND. *Full duplex.* Longitud fija de datos.

Características de los uC: *Periféricos*

E/S SERIE SíNCRONA: Bus I2C

Es un protocolo de bus **multi-maestro**. Tiene un mecanismo de detección de colisiones.



•La velocidad de transferencia máxima es de **100 kbit/seg.** (400 kbit/seg.)

•El bus tiene unas resistencias de *pull-up* (10k a 100kHz, 2k a 400 kHz).

•Los transistores de salida conectados al bus deben ser de **drenador** o **colector abierto** (conexión AND).

•La capacidad máxima del bus es **400 pF** (limitará el nº de dispositivos a conectar).

•El **control de flujo** se realiza enviando bits de reconocimiento **ACK** (SDA=0) por parte del *maestro* o del *esclavo*.

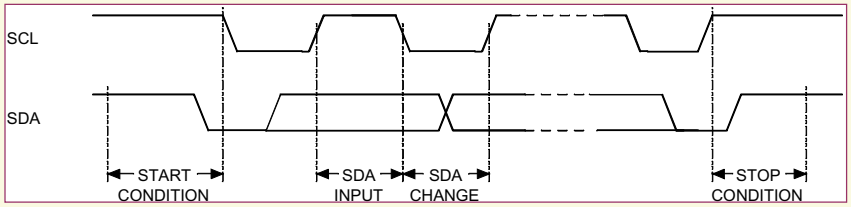
•La **tensión** del bus no es fija. Los periféricos ponen un "0" o *sueltan* el bus que va a "1" gracias a las resistencias de *pull-up*.

Características de los uC: *Periféricos*

E/S SERIE SíNCRONA: Bus I2C

Condición de inicio y de parada.

- El bus se considera **ocupado** después de la **condición de inicio** y se considerará **libre** un cierto tiempo después de la **condición de parada**.
- Condición de inicio: SCL en alta y el dato cambia de 1 a 0.
- Condición de parada: SCL en alta y el dato cambia de 0 a 1.



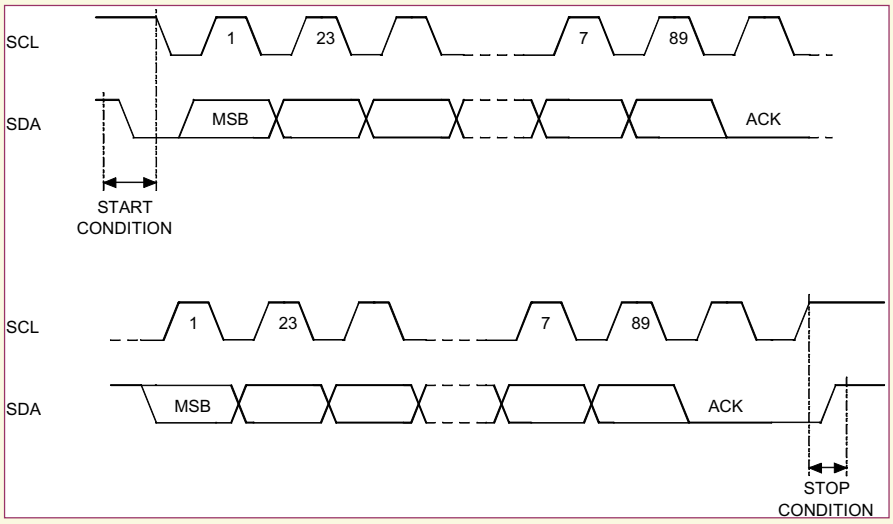
Transferencia de un bit.

- Cuando SCL está en alta el dato es válido.
- Cuando SCL está en baja el dato puede cambiar.

Características de los uC: *Periféricos*

E/S SERIE SíNCRONA: Bus I2C

Transferencia de datos. Los bytes son de 8 bits y tienen una señal ACK que genera el receptor.



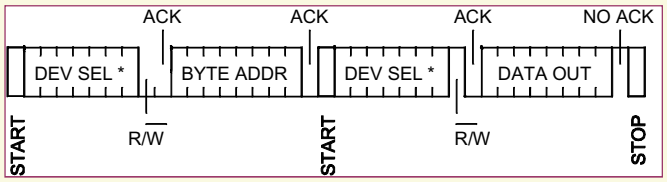
Características de los uC: *Periféricos*

E/S SERIE SíNCRONA: Bus I2C

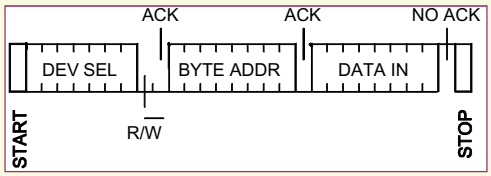
Formato:

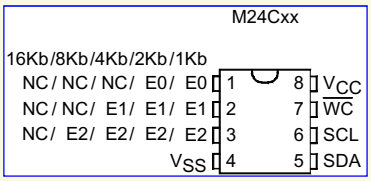
- Al enviar el maestro la dirección, todos los esclavos la leen y la comparan con la suya própia y responden con ACK si coincide.

Escritura aleatoria en la EEPROM 24Cxx



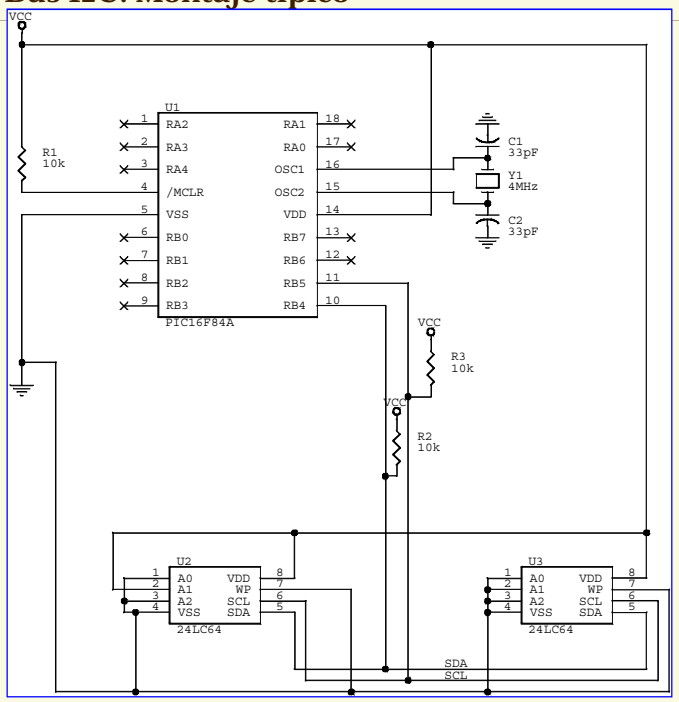
Lectura aleatoria en la EEPROM 24Cxx





Características de los uC: *Periféricos*

E/S SERIE SíNCRONA: Bus I2C. Montaje típico



Características de los uC: *Periféricos*

CONVERSOR A/D

Función:

- Acceso digital a señales analógicas. Para aplicaciones de instrumentación y captura de datos.

Características típicas:

- Hardware específico interno o no.
- Se convierten tensiones.
- Tiempo de conversión.
- Resolución (8, 10, 12, 14, 16 bits)
- Varias entradas multiplexadas.
- Métodos:

1.- Conversión por aproximaciones sucesivas.

2.- Contando el tiempo de carga de un condensador.

Características de los uC: *Periféricos*

CONVERSOR D/A

Función:

- Salida analógica.

Características típicas:

- Hardware específico interno o no.
- Tiempo de conversión.
- Resolución (8, 10, 12, 14, 16 bits).
- No es común que dispongan de este tipo de conversores.

Características de los uC: *Periféricos*

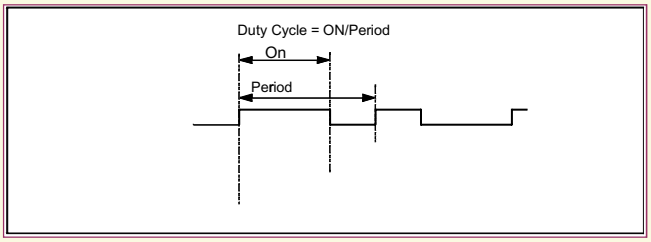
CONVERSOR PWM (Pulse Width Modulation).
Modulación por anchura de pulsos.

Función:

- Salida digital periódica. Es una forma de conversor D/A.

Características típicas:

- Hardware específico interno o no.
- Generación de frecuencia
- El ciclo de trabajo *(duty cycle)* se selecciona para que la señal una vez filtrada (pasa-baja) tenga como tensión efectiva la que se desea.



Características de los uC: *Periféricos*

Controlador DMA (Direct Access Memory). *Acceso directo a memoria.*

Función:

- Movimiento de datos de memoria a memoria, E/S a memoria, memoria a E/S y E/S a E/S sin intervención de la CPU del uC.

Características típicas:

- Espacio direccionable.
- Número de canales.
- No es muy común. Sólo cuando se puede poner memoria externa.

Características de los uC: *Watch dog timer*

Temporizador perro guardián

Características:

- El **Watch dog** o perro guardián es un temporizador interno del uC que dispone de su propio oscilador (RC) interno.
- Cuando se activa comienza una cuenta, predefinida por el programador, de forma que cuando finaliza se produce un *reset* en el uC.
- El programa debe refrescar el contaje del perro guardián para evitar el *reset*.
- Es un mecanismo pensado para evitar que el uC se encuentre en un estado indeterminado como consecuencia de un error de programación o de un fallo hardware (alimentación, etc.).
- Al rearrancar el sistema se vuelve a un estado conocido y seguro.

Características de los uC: *Reducción del consumo eléctrico*

Modos de bajo consumo

Los uC, típicamente, funcionan en sistemas autónomos o alimentados por baterías.

Es muy interesante alargar la duración de las baterías.

Disponen de **modos de funcionamiento** de **bajo consumo**:

- El sistema **reduce** su **frecuencia** de funcionamiento.
- El sistema entra en un **estado de reposo** (SLEEP) pero manteniendo las salidas. Congela el reloj (arquitectura estática).
- El sistema entra en un estado de **muy bajo consumo** sin mantener las salidas (HALT).

Al sistema se le puede despertar por interrupciones externas.

Características de los uC: *Interrupciones*

Las **interrupciones** son fundamentales en los sistemas en tiempo real para atender sucesos urgentes (externos o internos).

- **Tipos de interrupciones:**

Por software. Polling (Sondeo).

- Cuando no hay mecanismo de interrupción hardware.
- *Pregunta constantemente* a los periféricos.
- Es *lento.*
- Impide realizar otras tareas.

Por hardware.

- Los periféricos interrumpen al uC.
- El uC identifica el periférico y actúa.
- Los uC tienen, al menos, una patilla de petición de interrupción externa. Esta es sensible al *flanco* (insensible a la duración, pero sensible a rebotes y ruidos) o al *nivel* (debe mantenerse constante un cierto tiempo, no le afectan los ruidos).

Características de los uC: *Interrupciones*

Tipos de interrupciones. Hardware

Interrupciones enmascarables.

- Pueden ser desactivadas.
- Interesante cuando el uC está realizando tareas críticas que no pueden ser interrumpidas.
- Suelen tener un bit de activación global (GIE, *Global Interrupt Enable*).

Interrupciones no enmascarables.

Se atienden siempre y no pueden ser desactivadas.

Para tareas críticas del uC.

Características de los uC: *Interrupciones*

Tipos de interrupciones:

Interrupciones simples.

- El contador de programa (PC) se carga con una dirección fija.
- La subrutina de atención a la interrupción (ISR) identifica el periférico.
- La respuesta es *lenta*.
- Las prioridades las define el programador.

Interrupciones vectorizadas.

- El hardware automáticamente salta a una dirección u otra dependiendo del origen de la interrupción.
- La respuesta es *rápida*.
- El programador no tiene control sobre las prioridades.

Microcontroladores PIC

» Introducción

» Características generales de los microcontroladores

» **Estudio a fondo del PIC 16F84**

» Otros microcontroladores PIC

Estudio a fondo del PIC 16F84

1. **Generalidades**
2. **Memoria. Registros**
3. **Puertos E/S**
4. **Temporizadores**
5. **EEPROM**
6. **Características especiales:** Oscilador. Reset. Power On Reset. Startup Timer (PWRTE). Oscillator Startup Timer (OST). Interrupciones. Perro guardián (WDT). Modo de bajo consumo (SLEEP). Identificación.
7. **Instrucciones**

Estudio a fondo del PIC 16F84 : *Generalidades*

Características generales

- 35 instrucciones. Códigos de instrucción de 14 bits. Todas las instrucciones ocupan una palabra
- Todas las instrucciones duran un ciclo excepto las de salto que duran dos.
- Velocidad de funcionamiento 20MHz máximo (instr. 200 ns). Típicamente a 4MHz (instr. 1us).
- 1024 palabras (14 bits) de memoria de programa FLASH.
- 68 bytes de RAM de datos.
- 64 bytes de EEPROM de datos.
- 15 registros de función específica.
- Pila hardware de 8 niveles.
- Modos de direccionamiento directo, indirecto y relativo.
- Cuatro fuentes de interrupción.
- 13 pines de E/S con control individual de sentido.

Estudio a fondo del PIC 16F84 : *Generalidades*

Características generales

Pines de alta corriente (fuente/sumidero) capaces de controlar directamente un LED.(25 mA máximo por pin).

Temporizador/contador TMR0 de 8 bits con pre-escala programable.

Memoria de programa FLASH borrable/escribible 1.000 veces.

Memoria de datos EEPROM borrable/escribible 1.000.000 de veces

Retención de datos en EEPROM > 40 años.

Programación serie *In Circuit* via 2 pines.

Power-on Reset (POR), *Power-up Timer* (PWRT), *Oscillator Start-up Timer* (OST).

Watch dog Timer (WDT) con su propio oscilador RC integrado.

Protección del código.

Modo de ahorro de energía (SLEEP).

Opciones de oscilador seleccionables.

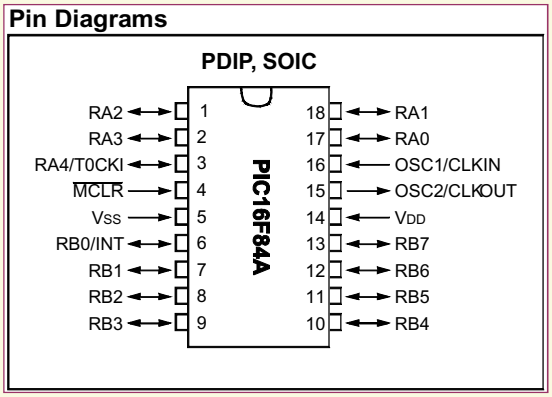
Estudio a fondo del PIC 16F84 : *Generalidades*

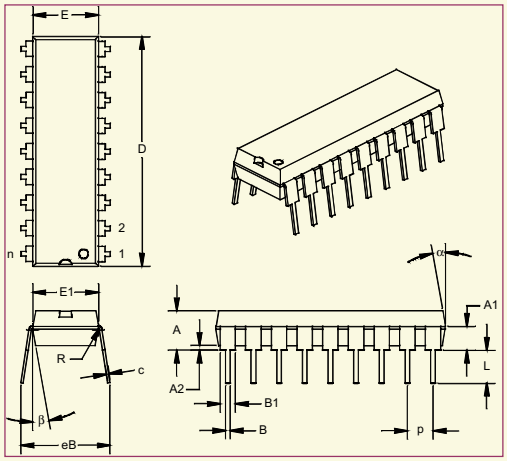
Patillaje

Consumo típico < 2 mA @ 5V, 4MHz

Tensión de alimentación de 2.0 a 5.5 V

Pin Diagrams





Estudio a fondo del PIC 16F84 : *Generalidades*

Patillaje: descripción

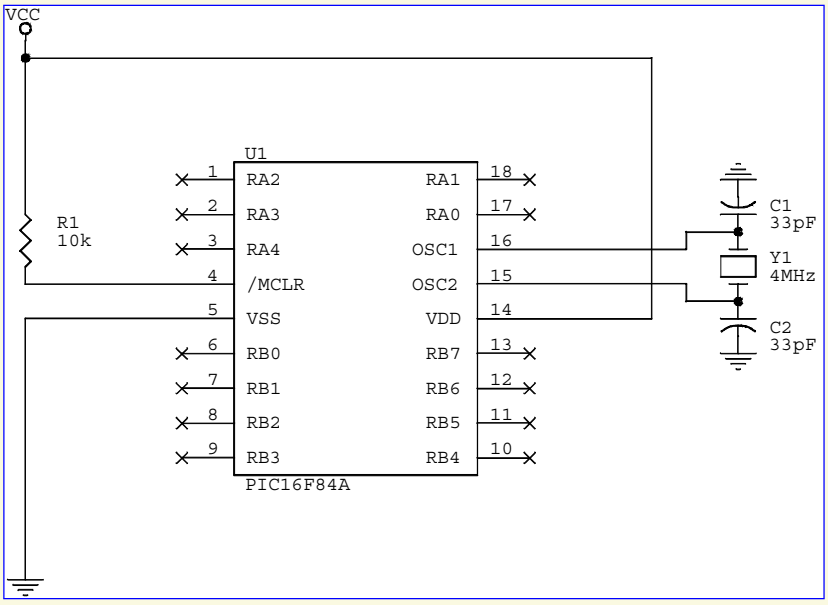
Pin Name	DIP No.	SOIC No.	SSOP No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	16	18	I	ST/CMOS (3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	15	19	O	-	Oscillator crystal output. Connects to cr crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{MCLR}$	4	4	4	I/P	ST	Master clear (reset) input/programming voltage input. This pin is an active low reset to the device.
						PORTA is a bi-directional I/O port.
RA0	17	17	19	I/O	TTL	
RA1	18	18	20	I/O	TTL	
RA2	1	1	1	I/O	TTL	
RA3	2	2	2	I/O	TTL	
RA4/T0CKI	3	3	3	I/O	ST	Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type.
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	6	6	7	I/O	TTL/ST (1)	RB0/INT can also be selected as an e xternal interrupt pin.
RB1	7	7	8	I/O	TTL	
RB2	8	8	9	I/O	TTL	
RB3	9	9	10	I/O	TTL	
RB4	10	10	11	I/O	TTL	Interrupt on change pin.
RB5	11	11	12	I/O	TTL	Interrupt on change pin.
RB6	12	12	13	I/O	TTL/ST (2)	Interrupt on change pin. Serial programming clock.
RB7	13	13	14	I/O	TTL/ST (2)	Interrupt on change pin. Serial programming data.
VSS	5	5	5,6	P	-	Ground reference for logic and I/O pins.
VDD	14	14	15,16	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

Estudio a fondo del PIC 16F84 : *Generalidades*

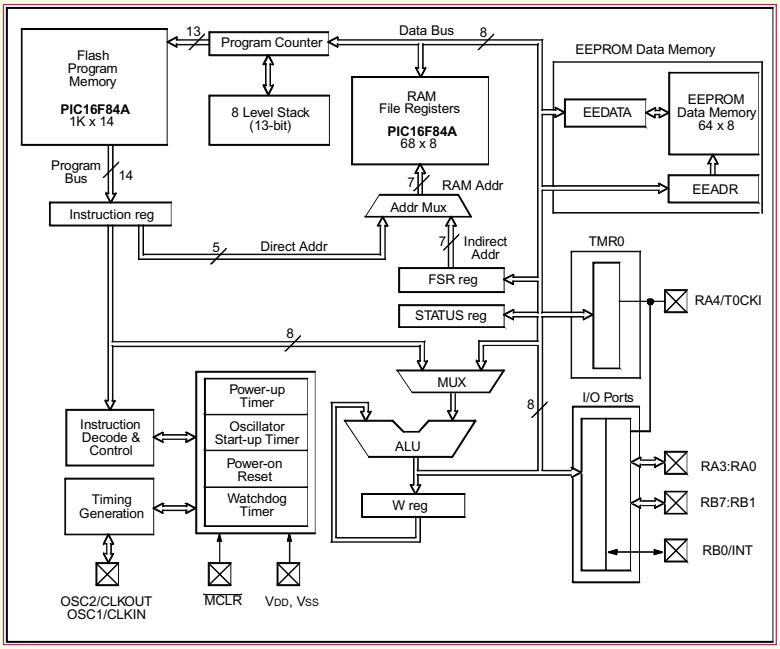
Montaje básico



Estudio a fondo del PIC 16F84 : *Generalidades*

Diagrama de bloques

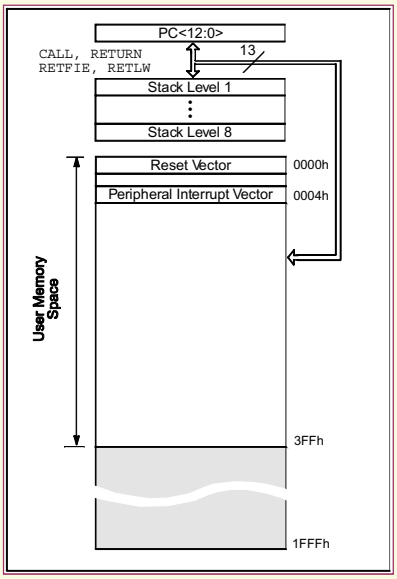
- 1024 palabras de memoria de programa (14 bit).
- Memoria de datos RAM de 68 bytes.
- Memoria de datos no volátil EEPROM de 64 bytes.
- Tiene 13 pines E/S configurables uno a uno. Algunos tienen funcionalidades múltiples.



Estudio a fondo del PIC 16F84 : *Memoria*

Organización de la memoria de programa

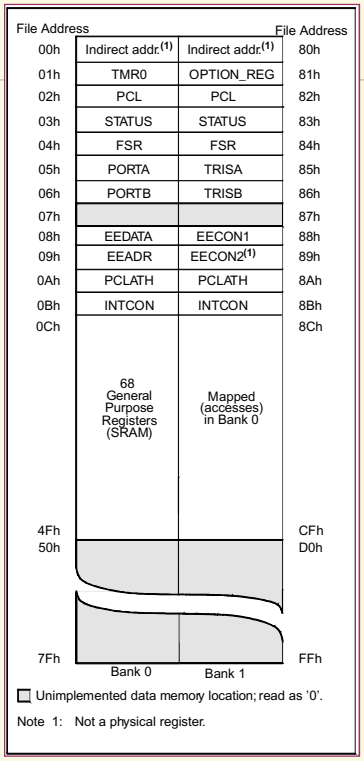
- El **contador de programa** (PC) es de 13 bits
- Puede direccionar 2^13 = 8k palabras de 14 bit desde *0000h-1FFFh*.
- En el PIC 16F84A sólo está implementada **1k** (*0000h-03FFh*).
- El **vector de reset** está en *0000h.*
- El **vector de interrupción** está en *0004h.*
- Dispone de una **pila hardware** (llamadas a subrutinas o interrupciones) de **8 niveles.**



Estudio a fondo del PIC 16F84 : *Memoria*

Organización de la memoria de datos

- Memoria RAM: Datos.
- Dividida en: *Registros de propósito general* (GPR) y *Registros de función específica* (SFR).
- La memoria de datos RAM está repartida en bancos (hasta 4 bancos). Se seleccionan a través de unos bits de control situados en el registro STATUS. En PIC 16F84A, hay 2 bancos.
- Las instrucciones MOVWF y MOVFW copian datos desde el registro W (Working Register) a la RAM (Register (F)ile) o viceversa.



Estudio a fondo del PIC 16F84 : *Memoria*

Registros de función específica (SFR)

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets (Note3)
Bank 0											
00h	INDF	Uses contents of FSR to address data memory (not a physical register)								---- ----	---- ----
01h	TMR0	8-bit real-time clock/counter								xxxx xxxx	uuuu uuuu
02h	PCL	Low order 8 bits of the Program Counter (PC)								0000 0000	0000 0000
03h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	ZD	C	C	0001 1xxx	000q quuu
04h	FSR	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
05h	PORTA (4)	-	-	-	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
06h	PORTB (5)	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
07h		Unimplemented location, read as '0'								---- ----	---- ----
08h	EEDATA	EEPROM data register								xxxx xxxx	uuuu uuuu
09h	EEADR	EEPROM address register								xxxx xxxx	uuuu uuuu
0Ah	PCLATH	-	-	-	Write buffer for upper 5 bits of the PC(1)					---0 0000	---0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
Bank 1											
80h	INDF	Uses contents of FSR to address data memory (not a physical register)								---- ----	---- ----
81h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	0000 0000
83h	STATUS (2) IRP		RP1	RP0	$\overline{TO}$	$\overline{PD}$	ZD	C	C	0001 1xxx	000q quuu
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
85h	TRISA	-	-	-	PORTA data direction register					---1 1111	---1 1111
86h	TRISB	PORTB data direction register								1111 1111	1111 1111
87h		Unimplemented location, read as '0'								---- ----	---- ----
88h	EECON1	-	-	-	EEIF	WRERR	WREN	WR	RD	---0 x000	---0 q000
89h	EECON2	EEPROM control register 2 (not a physical register)								---- ----	---- ----
0Ah	PCLATH	-	-	-	Write buffer for upper 5 bits of the PC(1)					---0 0000	---0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u

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

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> is never transferred to PCLATH.

2: The $\overline{TO}$ and $\overline{PD}$ status bits in the STATUS register are not affected by a $\overline{MCLR}$ reset.

3: Other (non power-up) resets include: external reset through $\overline{MCLR}$ and the Watchdog Timer Reset.

4: On any device reset, these pins are configured as inputs

5: This is the value that will be in the port output latch.

Estudio a fondo del PIC 16F84 : *Memoria*

Registro de estado (STATUS)

Contiene:

- El estado aritmético de la ALU.
- El estado del RESET.
- Los bit de selección de los bancos de memoria.

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
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7: **IRP**: Register Bank Select bit (used for indirect addressing)
The IRP bit is not used by the PIC16F84A. IRP should be maintained clear.

bit 6-5: **RP1:RP0**: Register Bank Select bits (used for direct addressing)
00 = Bank 0 (00h - 7Fh)
01 = Bank 1 (80h - FFh)
Each bank is 128 bytes. Only bit RP0 is used by the PIC16F84A. RP1 should be maintained clear.

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 (for `ADDWF` and `ADDLW` 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 (for `ADDWF` and `ADDLW` 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.

Estudio a fondo del PIC 16F84 : *Memoria*

Registro de opciones (OPTION_REG)

Configura:

- La pre-escala del TMR0/WDT.
- La interrupción externa INT.
- TMR0.
- Las resistencias de pull-up débil de PORTB.

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7: **$\overline{RBPU}$**: 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 (CLKOUT)

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 assigned to the WDT
0 = Prescaler assigned to TMR0

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

Estudio a fondo del PIC 16F84 : *Memoria*

Registro control de interrupciones (INTCON)

- Habilita todas las fuentes de interrupción

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	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7: **GIE:** Global Interrupt Enable bit
1 = Enables all un-masked interrupts
0 = Disables all interrupts

Note: For the operation of the interrupt structure, please refer to Section *.

bit 6: **EEIE**: EE Write Complete Interrupt Enable bit
1 = Enables the EE write complete interrupt
0 = Disables the EE write complete interrupt

bit 5: **T0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4: **INTE**: RB0/INT Interrupt Enable bit
1 = Enables the RB0/INT interrupt
0 = Disables the RB0/INT 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: **T0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 has overflowed (must be cleared in software)
0 = TMR0 did not overflow

bit 1: **INTF**: RB0/INT Interrupt Flag bit
1 = The RB0/INT interrupt occurred
0 = The RB0/INT interrupt did not occur

bit 0: **RBIF**: RB Port Change Interrupt Flag bit
1 = When at least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Estudio a fondo del PIC 16F84 : *Memoria*

PCL y PCLATH

- El contador de programa PC es de 13 bits.
- Su parte baja es el registro PCL (8 bits).
- Su parte alta es PCH (12..8) y no es accesible directamente. Se accede a él a través de PCLATH.

Pila

- Se pueden anidar hasta 8 llamadas a subrutinas o interrupciones.
- Después de una instrucción CALL o una interrupción, el PC se mete en la pila.
- Al retornar (RETURN, RETLW, RETFIE) se recupera de la pila.

Estudio a fondo del PIC 16F84 : *Memoria*

Direccionamiento indirecto: INDF y FSR

- El registro INDF no es un registro físico.
- Al direccionar INDF, direccionamos el registro apuntado por FSR.

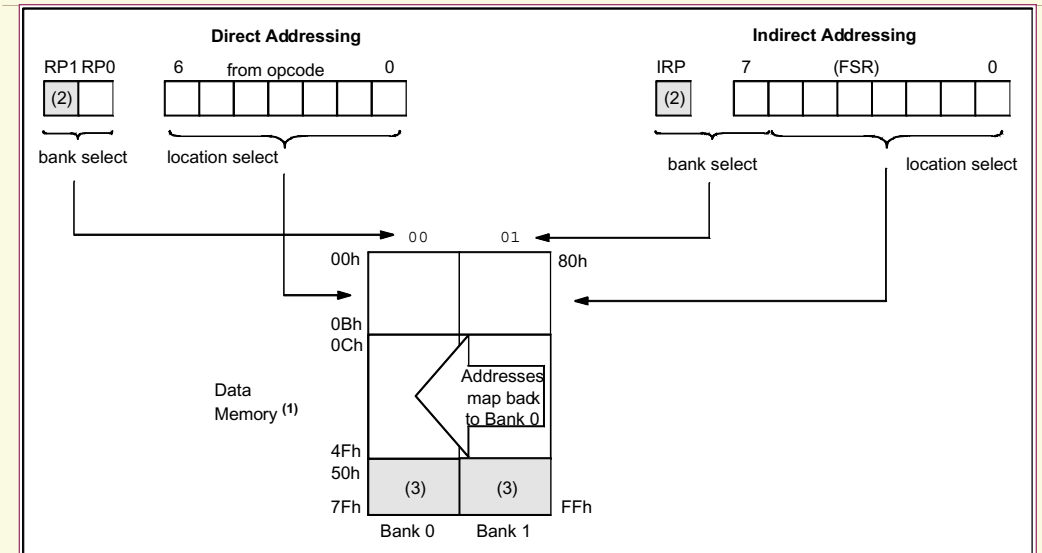
Ejemplo

- El registro 05 contiene 10h
- El registro 06 contiene 0Ah
- Cargamos 05 en FSR
- Al leer INDF nos devuelve 10h
- Incremento FSR (=>06)
- Al leer INDF nos devuelve 0Ah

```
; Como borrar un bloque de RAM
; con direccionamiento indirecto
        movlw   0x20
        movwf   FSR
Sig     clrf    INDF
        incf    FSR
        btfss   FSR,4
        goto    Sig
```


Estudio a fondo del PIC 16F84 : *Memoria*

Direccionamiento indirecto: INDF y FSR



Note 1: For memory map detail see Figure 2-1.
2: Maintain as clear for upward compatiblity with future products.
3: Not implemented.

Estudio a fondo del PIC 16F84 : *Puertos E/S*

Registros PORTA y TRISA

- PORTA es un puerto bidireccional de 5 bit.
- TRISA es quien controla la dirección de cada bit: (=1) entrada, (=0) salida.
- Al arrancar (POR) todos están configurados como entradas y se leen como ceros.
- Las operaciones de escritura son del tipo lee-modifica-escribe.
- La corriente máxima que puede recibir (*sunk*) en conjunto es de 80 mA.
- La corriente máxima que puede proporcionar (*sourced*) es de 50 mA.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets
05h	PORTA	-	-	-	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
85h	TRISA	-	-	-	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111

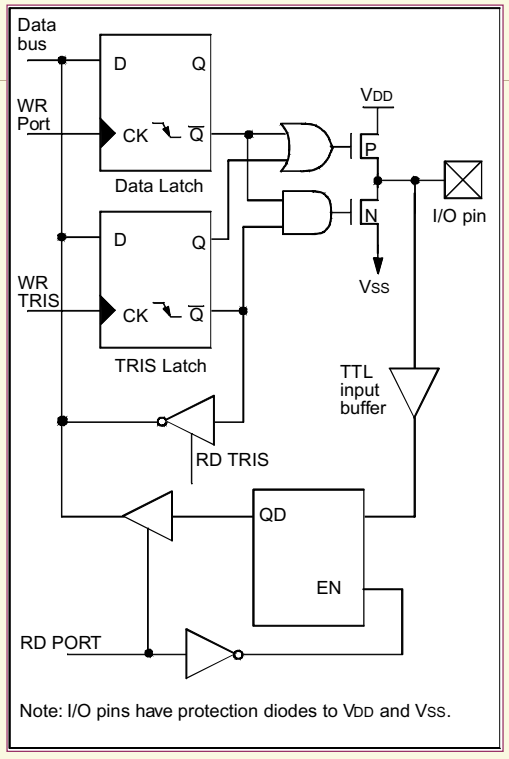
Legend: x = unknown, u = unchanged, - = unimplemented read as '0'.Shaded cells are unimplemented, read as '0'

Estudio a fondo del PIC 16F84 : *Puertos E/S*

RA3:RA0. Diagrama de bloques

Características:

- **Entrada** tipo **TTL**.
- **Salida** tipo **TTL.**
- Proporciona máximo 20 mA y es capaz de recibir máximo 25 mA.

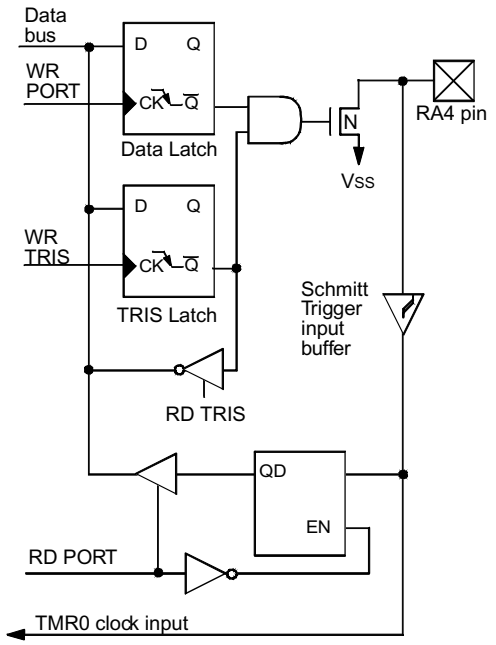


Estudio a fondo del PIC 16F84 : *Puertos E/S*

RA4 : Diagrama de bloques

Caracterísiticas:

- Multiplexada con el módulo del temporizador: **RA4/T0CKI.**
- **Entrada** con disparador **Schmitt.**
- Salida con el **drenador abierto**. Necesita una resistencia de *pull-up* para poner un "1" en la salida.



Estudio a fondo del PIC 16F84 : *Puertos E/S*

Registros PORTB y TRISB

- PORTB es un puerto bidireccional de 8 bit.
- TRISB es quien controla la dirección de cada bit: (=1) entrada, (=0) salida.
- Tienen una resistencia débil de *pull-up* programable (OPTION_REG.RBPU=0).
- Al arrancar (POR) todos están configurados como entradas y se leen como "0". También se desactivan las resistencias débiles de *pull-up* interno.
- Las operaciones de escritura son del tipo lee-modifica-escribe.
- La corriente máxima que puede recibir (*sumk*) en conjunto es de 80 mA.
- La corriente máxima que puede proporcionar (*sourced*) es de 50 mA.

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111
81h	OPTION_REG	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

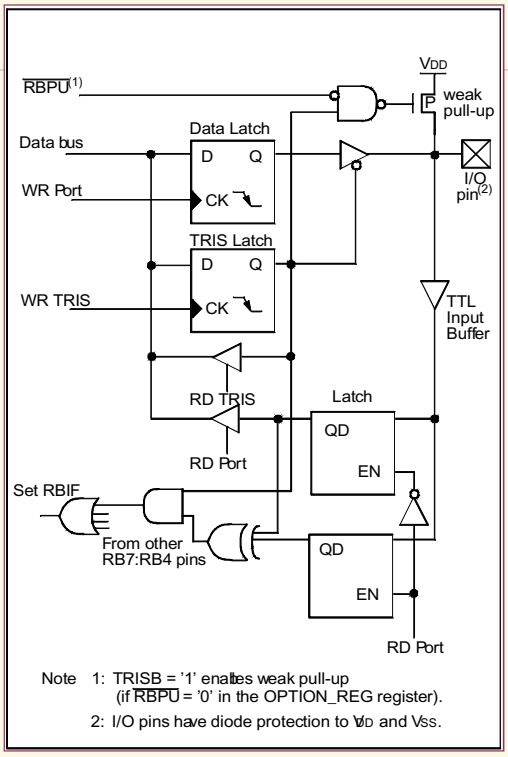
Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

Estudio a fondo del PIC 16F84 : *Puertos E/S*

RB7:RB4. Diagrama de bloques.

Características:

- **Salida TTL**.
- **Entrada** con resistencia de ***pull-up*** **programable** (OPTION_REG.RBPU=0).
- Pueden producir una **interrupción** cuando cambia su valor de **entrada**. Despierta del modo SLEEP. Recomendado para atender teclados.
- RB6 es la patilla de programación serie (*In Circuit*): **Clock**.
- RB7 es la patilla de programación serie (*In Circuit*): **Data**.

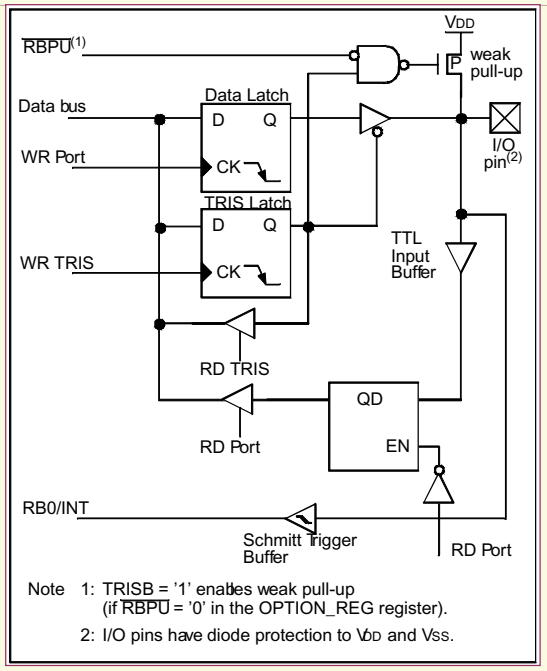


Estudio a fondo del PIC 16F84 : *Puertos E/S*

RB3:RB0. Diagrama de bloques.

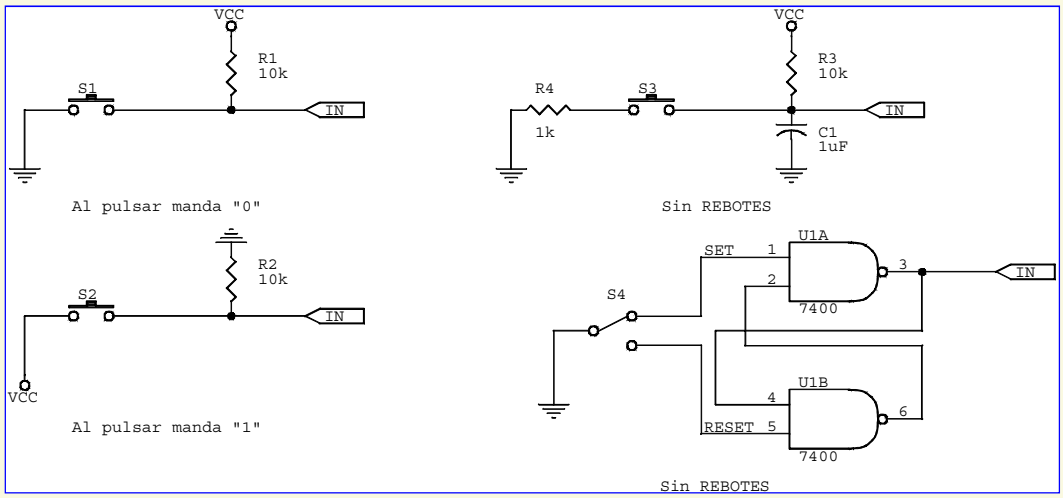
Características:

- **Salida TTL.**
- **Entrada** con resistencia de ***pull-up*** **programable** (OPTION.RBPU=0).
- **RB0/INT** permite atender **interrupciones externas** gracias a un disparador Schmitt.



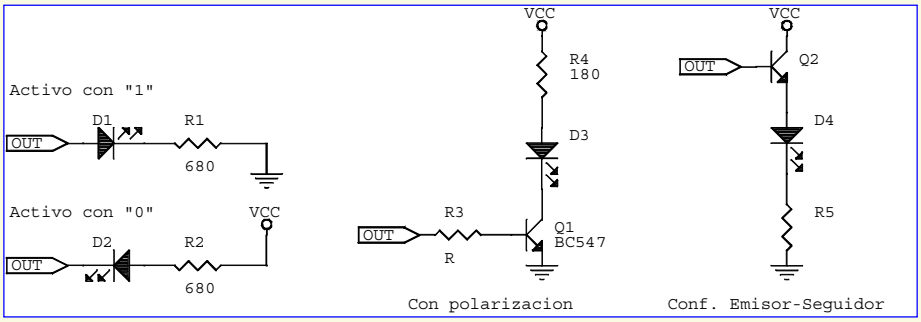
Estudio a fondo del PIC 16F84 : *Puertos E/S*

Hardware: Conexión de pulsadores

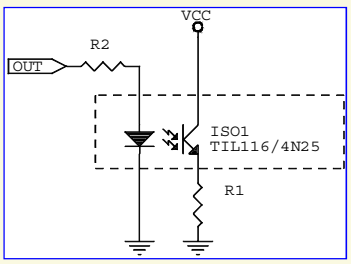


Estudio a fondo del PIC 16F84 : *Puertos E/S*

Hardware: Conexión de LEDs

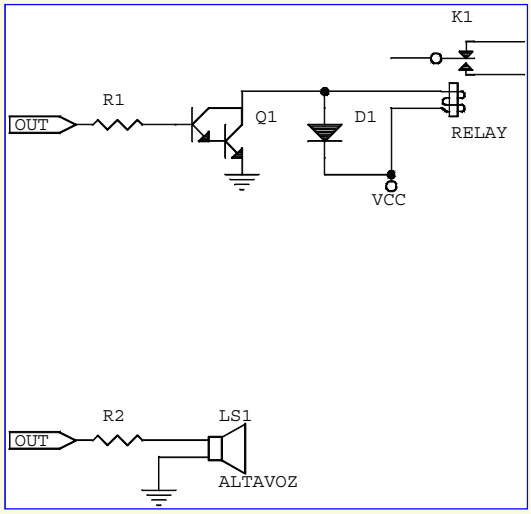


Hardware: Salida opto-acoplada



Estudio a fondo del PIC 16F84 : *Puertos E/S*

Hardware: Conexión de Relay y altavoz.



Estudio a fondo del PIC 16F84 : *Puertos E/S*

Ejemplo

```
01      LIST p=16F84A
02      INCLUDE "P16F84A.INC"
03      RADIX HEX
04      ERRORLEVEL -302
05
06      ORG 0
07
08      BSF     STATUS,RP0      ; Selecciona Banco 1
09      MOVLW   B'11111111'     ; W = 0FFh (Todo entradas)
10      MOVFW   TRISA           ; Configuro PORTA
11      MOVLW   B'00000000'     ; W = 00h  (Todo salidas)
12      MOVWF   TRISB           ; Configuro PORTB
13      BCF     STATUS,RP0      ; Selecciono Banco 0
14
15  INICIO
16      MOVF    PORTA,W         ; Leo W = PORTA
17      ADDLW   0X02            ; W = W + 2
18      MOVWF   PORTB           ; PORTB = W
19
20      GOTO    INICIO          ; Repito ...
21
22      END
23
```

Estudio a fondo del PIC 16F84 : *Temporizador*

Timer0

Características:

- Temporizador/contador de 8 bits.
- Se puede leer y escribir.
- Selección de reloj interno o externo.
- Selección de flanco (de subida o bajada) en el reloj externo.
- Pre-escala programable de 8 bit.
- Interrupción cuando pasa de FFh a 00h.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other resets
01h	TMR0	Timer0 module's register								xxxx xxxx	uuuu uuuu
0Bh,8Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
81h	OPTION_REG	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
85h	TRISA	-	-	PORTA Data Direction Register						--11 1111	--11 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'.Shaded cells are not used by Timer0.

Estudio a fondo del PIC 16F84 : *Temporizador*

Timer0: Funcionamiento

Modo temporizador

- Se selecciona con OPTION_REG.T0CS = 0.
- Se incrementa con cada ciclo de instrucción si no usamos la pre-escala (4 ciclos de reloj).
- Cuando se **escribe** en TMR0 el incremento **se inhibe** durante **2 ciclos** de instrucción.

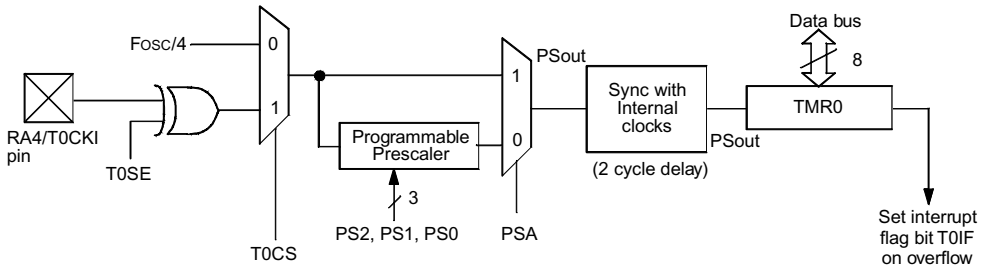
Modo contador

- Se selecciona con OPTION_REG.T0CS = 1.
- Se incrementará con cada flanco (de subida o bajada) del pin **RA4/T0CKI**.
- Con OPTION_REG.T0SE = 0 detecta el flanco de **subida**.
- Con OPTION_REG.T0SE = 1 detecta el flanco de **bajada.**

Estudio a fondo del PIC 16F84 : *Temporizador*

Timer0: Diagrama de bloques

El temporizador dispone de un modulo de pre-escala, que funcionará como post-escala si lo usa el temporizador del perro guardián.



Note 1: T0CS, T0SE, PSA, PS2:PS0 (OPTION_REG<5:0>).
2: The prescaler is shared with Watchdog Timer (refer to Figure 4-2 for detailed block diagram).

Estudio a fondo del PIC 16F84 : *Temporizador*

Timer0: Escalado

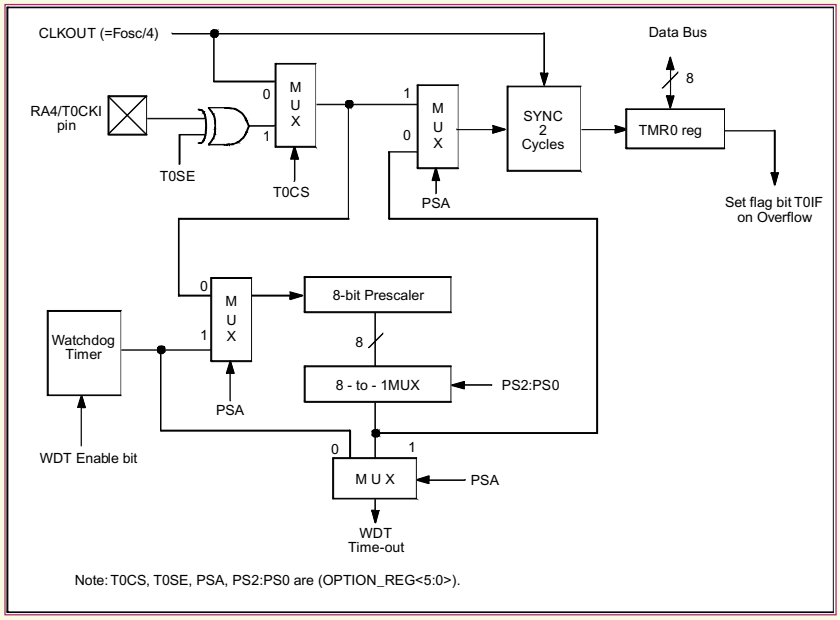
El temporizador dispone de un modulo de pre-escala, que funcionará como post-escala si lo usa el temporizador del perro guardián.

Características:

- No se puede leer o escribir. Cualquier modificación de TMR0 borra la pre-escala. Idem CLRWDT (borra el temporizador del perro guardián) cuando asignado al WDT.
- OPTION_REG.PSA = 0 asigna la *pre-escala* al **temporizador.** OPTION_REG.PS2:PS0 definen la pre-escala: 1:2, 1:4, ..., 1:256.
- OPTION_REG.PSA=1 asigna al **Watch dog** una *post-escala*. OPTION_REG.PS2:PS0 definen la post-escala: 1:1, 1:2, ...,1:128.

Estudio a fondo del PIC 16F84 : *Temporizador*

Timer0: Escalado



Estudio a fondo del PIC 16F84 : *Temporizador*

Timer0: Interrupción

- El TMR0 genera una interrupción cuando pasa de FFh a 00h.
- Cuando esto ocurre, se indica automáticamente con INTCON.T0IF = 1.
- El bit INTCON.T0IF debe ser puesto de nuevo a cero por software una vez atendida la interrupción.
- La interrupción se puede enmascarar con INTCON.T0IE = 1.
- La interrupción TMR0 no puede despertar al uC del estado SLEEP de bajo consumo.

Estudio a fondo del PIC 16F84 : *Temporizador*

Timer0: Ejemplo

```
01  ; Queremos que un LED conectado a RB7 parpadee cada 8.2 ms
02  ; El PIC funciona a 1 MHz
03  ;
04  ; Temp = 4 * Tosc * TMR0 * Divisor
05  ; Temp = 4 * 1 usec * 16 * 128  + 2 * 4 * 1 usec =  0.0082
06
07      LIST P=16F84A
08      INCLUDE "P16F84A.INC"
09      ERRORLEVEL -302
10
11      ORG 0h
12
13      bsf     STATUS,RP0      ; Pongo Banco 1
14      movlw   B'11010110'     ; W=11010110b
15      movwf   OPTION_REG      ; /RBPU  = 1    Pull-up disabled
16                              ; INTEDG = 1    RB0/INT flanco subida
17                              ; T0CS   = 0    Timer
18                              ; T0SE   = 1    RA4/T0CKI flanco bajada
19                              ; PSA    = 0    Prescaler to Timer
20                              ; PS2:PS0= 110  1:128
21      movlw   0x00            ; W=0
22      movwf   TRISB           ; PORTB todo salidas
23      bcf     STATUS,RP0      ; Pongo Banco 0
24      clrf    PORTB           ; PORTB = 0
25
26  PARPAD
27      bsf     PORTB,7         ; Enciende LED RB7
28      call    RETARDO
29      bcf     PORTB,7         ; Apaga LED RB7
30      call    RETARDO
31      goto    PARPAD
32
33  RETARDO
34      clrf    TMR0            ; TMR0 = 0
35  REP btfss   TMR0,4          ; TMR0 >= 16
36      goto    REP
37      return
38
39      END
40
41  ; ¿Qué habría que tocar para que funcionara en un PIC a 4 Mhz?
42  ;   Temp = 4 * 0.25 usec * ( 64 * 128 + 2)  = 8,194 ms
43  ; ¿Cual es el mayor valor de temporización que podemos alcanzar?
44  ;   Temp = 4 * 0.25 usec * ( 255 * 255 + 2 )= 65.027 ms
45  ;   Deberíamos comprobar INTCON.T0IF que indica fuera de cuenta
46
```


Estudio a fondo del PIC 16F84 : *Memoria EEPROM*

Memoria EEPROM

Características:

- La memoria de datos EEPROM es de **lectura/escritura.**
- Es una memoria **no volátil**.
- No se accede a ella directamente (necesita un **protocolo**).
- Se accede a través de los registros:
  - **EEDATA** contiene el dato de 8 bit para leer o escribir.
  - **EEADR** contiene la dirección accedida (00h..3Fh).
  - **EECON1** y **EECON2** (físicamente no existe).
- La escritura de un byte es un proceso *borra-escribe*. El tiempo necesario para la escritura está controlado por un temporizador interno y *varía con* el *voltaje* y la *temperatura* (típicamente **4 ms**).

Estudio a fondo del PIC 16F84 : *Memoria EEPROM*

Registro EECON1

U	U	U	R/W-0	R/W-x	R/W-0	R/S-0	R/S-x
-	-	-	EEIF	WRERR	WREN	WR	RD
bit7							bit0

R = Readable bit
W = Writable bit
S = Settable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7:5 **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 **WRERR**: EEPROM Error Flag bit
1 = A write operation is prematurely terminated (any $\overline{\text{MCLR}}$ reset or any WDT reset during normal operation)
0 = The write operation completed

bit 2 **WREN**: EEPROM Write Enable bit
1 = Allows write cycles
0 = Inhibits write to the data EEPROM

bit 1 **WR**: Write Control bit
1 = initiates a write cycle. (The bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.
0 = Write cycle to the data EEPROM is complete

bit 0 **RD**: Read Control bit
1 = Initiates an EEPROM read (read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software).
0 = Does not initiate an EEPROM read

Estudio a fondo del PIC 16F84 : *Memoria EEPROM*

Memoria EEPROM

Lectura

- Se escribe la dirección en EEADR.
- Se activa EECON1.RD=1.
- En el siguiente ciclo se dispone del dato leído en EEDATA.

Escritura

- Se escribe la dirección en EEADR.
- Se escribe el dato en EEDATA.
- Se habilita la escritura EECON1.WREN=1.
- Se escribe 55h en EECON2. Se escribe AAh en EECON2.
- Se activa la escritura con EECON1.WR=1.
- Cuando se termina la escritura, automáticamente EECON1.WR=0 y INTCON.EEIF=1 para señalizarlo. Se producirá una interrupción si no está enmascarada.

Estudio a fondo del PIC 16F84 : *Memoria EEPROM*

Ejemplo

Lectura

```
BCF     STATUS, RP0  ; Bank 0
MOVLW   CONFIG_ADDR  ;
MOVWF   EEADR        ; Address to read
BSF     STATUS, RP0  ; Bank 1
BSF     EECON1, RD   ; EE Read
BCF     STATUS, RP0  ; Bank 0
MOVF    EEDATA, W    ; W = EEDATA
```

Escritura

```
BSF     STATUS, RP0  ; Bank 1
BCF     INTCON, GIE  ; Disable INTs.
BSF     EECON1, WREN ; Enable Write
MOVLW   55h          ;
Required Sequence
MOVWF   EECON2       ; Write 55h
MOVLW   AAh          ;
MOVWF   EECON2       ; Write AAh
BSF     EECON1,WR    ; Set WR bit
                     ;   begin write
BSF     INTCON, GIE  ; Enable INTs.
```

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets
08h	EEDATA	EEPROM data register								xxxx xxxx	uuuu uuuu
09h	EEADR	EEPROM address register								xxxx xxxx	uuuu uuuu
88h	EECON1	-	-	-	EEIF	WRERR	WREN	WR	RD	---0 x000	---0 q000
89h	EECON2	EEPROM control register 2								---- ----	---- ----

Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q = value depends upon condition. Shaded cells are not used by data EEPROM.

Estudio a fondo del PIC 16F84 : *Características especiales*

Bits de Configuración

Existen unos *bits de configuración* que están en el espacio de memoria especial para test y configuración (**2000h-3FFFh**), que sólo es accesible durante la programación.

La palabra de configuración está en la dirección **2007h**.

R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u
CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	$\overline{\text{PWRTE}}$	WDTE	FOSC1	FOSC0
bit13													bit0

R = Readable bit
P = Programmable bit
- n = Value at POR reset
u = unchanged

bit 13:4 **CP**: Code Protection bit
1 = Code protection off
0 = All memory is code protected

bit 3 **$\overline{\text{PWRTE}}$**: Power-up Timer Enable bit
1 = Power-up timer is disabled
0 = Power-up timer is enabled

bit 2 **WDTE**: Watchdog Timer Enable bit
1 = WDT enabled
0 = WDT disabled

bit 1:0 **FOSC1:FOSC0**: Oscillator Selection bits
11 = RC oscillator
10 = HS oscillator
01 = XT oscillator
00 = LP oscillator

Estudio a fondo del PIC 16F84 : *Características especiales*

Bits de configuración

La forma de modificar los bits de configuración se puede hacer a través del programa:

```
INCLUDE "P16F84A.INC"

__CONFIG _WDT_OFF & _CP_OFF & _XT_OSC & _PWRTE_ON
```

cuyo significado es

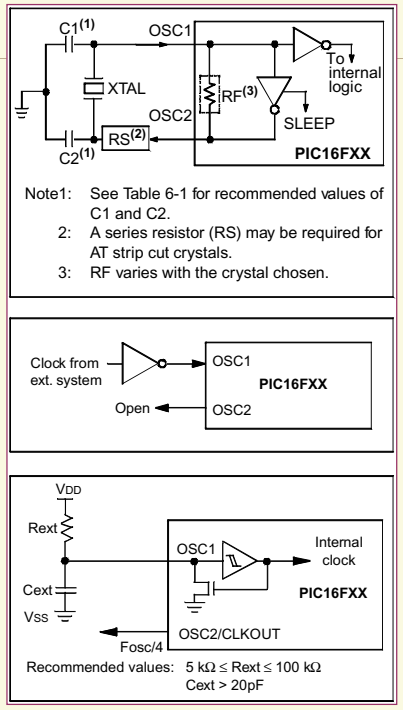
- Perro guardián (WDT) desactivado.
- Protección del código (CP) desactivada.
- Oscilador de cristal seleccionado. Otras opciones son: **_HS_OSC, _LP_OSC, _RC_OSC**.
- Temporizador de arranque (PWRT) desactivado (notar que es activo en baja).

Estudio a fondo del PIC 16F84 : *Características especiales*

Configuración del oscilador

Tiene cuatro modos de funcionamiento:

- LP. *Low power crystal.* Con un cristal de bajo consumo y baja frecuencia (hasta 200 kHz).
- XT. *Crystal/resonator.* Con un cristal o un resonador de frecuencias medias (hasta 4 MHz).
- HS. *High speed crystal/resonator*. Con un cristal/resonador de alta velocidad (hasta 20 MHz).
- RC. *Resistor/capacitor*. Con una red RC. Es una opción barata cuando la aplicación no depende del tiempo. La frecuencia de oscilación dependerá de *Rext*, *Cext*, y de la *tensión* de alimentación y de la *temperatura* de funcionamiento.



Estudio a fondo del PIC 16F84 : *Características especiales*

Reset

Características:

Se diferencia entre varios tipos de **reset**.

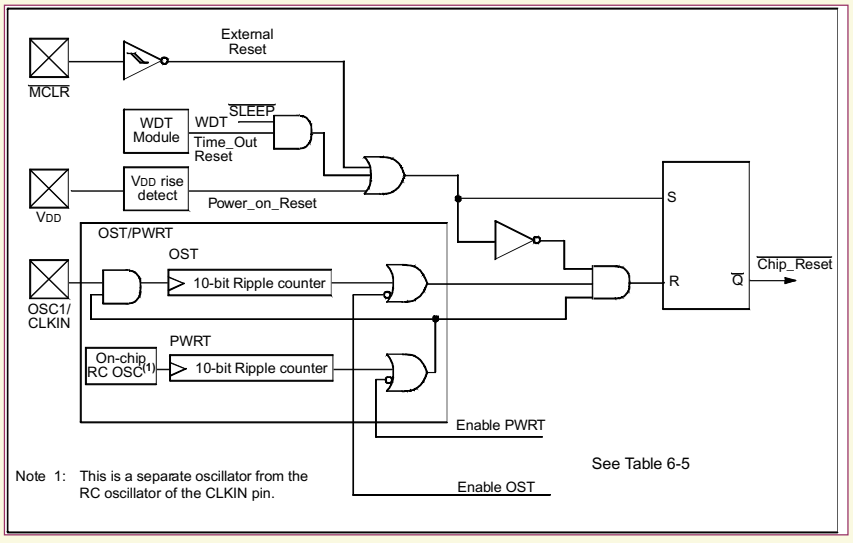
- Power-on Reset (POR).
- /MCLR en modo de operación normal.
- /MCLR en modo de operación SLEEP.
- WDT reset en modo de operación normal.
- WDT despertador en modo de operación SLEEP.

Condition	Program Counter	STATUS Register
Power-on Reset	000h	0001 1xxx
$\overline{MCLR}$ Reset during normal operation	000h	000u uuuu
$\overline{MCLR}$ Reset during SLEEP	000h	0001 0uuu
WDT Reset (during normal operation)	000h	0000 1uuu
WDT Wake-up	PC + 1	uuu0 0uuu
Interrupt wake-up from SLEEP	PC + 1 (1)	uuu1 0uuu

Legend: u = unchanged, x = unknown.
Note 1: When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).

Estudio a fondo del PIC 16F84 : *Características especiales*

Reset: Diagrama de bloques.



Estudio a fondo del PIC 16F84 : *Características especiales*

Power-on Reset (POR)

- Es un *reset* generado *on-chip* cuando la tensión de alimentación está entre 1.2 y 1.7 V.
- Esto elimina las redes RC necesarias para que se inicie el uC correctamente.

Power-up Timer (PWRT)

- Cuando está activado proporciona una señal *reset* durante **72 ms** (temporización fija gracias a una red RC integrada).
- El PWRT permite que Vdd alcance un nivel aceptable.

Oscillator Start-up Timer (OST)

- Proporciona un tiempo de *reset* extra (después de PWRT) de 1024 ciclos de reloj.
- Se asegura de que el oscilador de cristal o el resonador han comenzado a oscilar y se han estabilizado.
- Sólo funciona si el oscilador está en modo XT, LP o HS.

Estudio a fondo del PIC 16F84 : *Características especiales*

Resumen reset

Oscillator Configuration	Power-up		Wake-up from SLEEP
	PWRT Enabled	PWRT Disabled	
XT, HS, LP	72 ms + 1024Tosc	1024Tosc	1024Tosc
RC	72 ms	-	-

$\overline{TO}$	$\overline{PD}$	Condition
11		Power-on Reset
0x		Illegal, $\overline{TO}$ is set on $\overline{POR}$
x0		Illegal, $\overline{PD}$ is set on $\overline{POR}$
01		WDT Reset (during normal operation)
00		WDT Wake-up
11		$\overline{MCLR}$ Reset during normal operation
10		$\overline{MCLR}$ Reset during SLEEP or interrupt wake-up from SLEEP

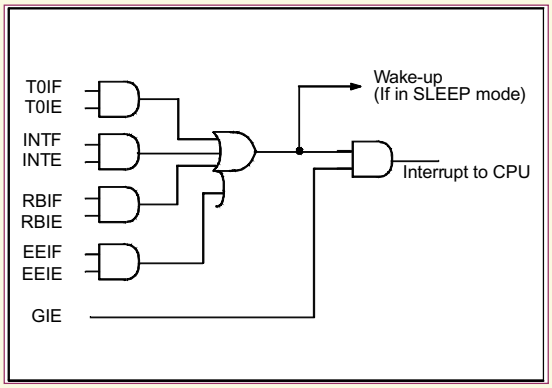
- Al arrancar, el uC permite conocer qué tipo de reset se ha producido a través de los bits /TO y /PD del registro STATUS.

Estudio a fondo del PIC 16F84 : *Características especiales*

Interrupciones

Dispone de 4 fuentes de interrupción:

- Interrupción externa a través del pin **RB0/INT**.
- Interrupción con el desbordamiento de **TMR0**.
- Interrupción cuando cambia alguna entrada en **RB4:RB7**.
- Interrupción cuando se completa una **escritura** de datos en la **EEPROM**.



Estudio a fondo del PIC 16F84 : *Características especiales*

Interrupciones

- **Activación/desactivación global** de las interrupciones con INTCON.GIE.
- Se desactivan con el *reset*.
- La instrucción de retorno de interrupción (**RETFIE**) activa las interrupciones.
- Cuando se atiende una interrupción, se desactivan globalmente las interrupciones para impedir, por defecto, las interrupciones anidadas. A continuación se salta a la posición **0004h** de la memoria de programa.
- En **interrupciones externas** la atención se **retrasa 3 o 4 ciclos** de instrucción.
- El origen de la interrupción se conoce sondeando los **flags de interrupción** (T0IF, INTF, RBIF y EEIF del registro INTCON) .
- Estos flags deben ser borrados por el programador después de atender el evento.

Estudio a fondo del PIC 16F84 : *Características especiales*

Resumen interrupciones

Interrupción INT

- RB0/INT se dispara por flanco.
- OPTION_REG.INTEDG selecciona el flanco (1, subida; 0, bajada).
- Cuando se produce INTCON.INTF se activa.
- Se puede enmascarar con INTCON.INTE=1.
- Esta interrupción despierta al uC del modo SLEEP.

Interrupción TMR0

- Se produce cuando se desborda TMR0 (FFh a 00h).
- Se señaliza con INTCON.T0IF=1.
- Se puede enmascarar con INTCON.T0IE=1.
- No despierta al uC del modo SLEEP.

Estudio a fondo del PIC 16F84 : *Características especiales*

Resumen interrupciones

Interrupción PORTB

- Un cambio en RB7:RB4 produce la interrupción.
- Se señaliza con INTCON.RBIF=1.
- Se enmascara con INTCON.RBIE=1.

Interrupción EEPROM escritura completada

- Se produce cuando completa una escritura en la EEPROM.
- Se señaliza con INTCON.EEIF=1.
- Se enmascara con INTCON.EEIE=1.

Estudio a fondo del PIC 16F84 : *Características especiales*

Interrupciones: salvar el contexto

Cuando se produce una interrupción solamente el PC se salva en la pila hardware.

Hay que guardar el contexto de ejecución (W, STATUS).

```
PUSH    MOVWF   W_TEMP          ; W_TEMP=W
        SWAPF   STATUS,W        ; W=swap(STATUS)
        MOVWF   STATUS_TEMP     ; STATUS_TEMP=W
        ...
POP     SWAPF   STATUS_TEMP,W   ; W=swap(STATUS_TEMP)
        MOVWF   STATUS          ; STATUS=W
        SWAPF   W_TEMP,F        ; W_TEMP=swap(W_TEMP)
        SWAPF   W_TEMP,W        ; W = swap(W_TEMP)
        RETFIE
```


Estudio a fondo del PIC 16F84 : *Características especiales*

Ejemplo

```
01  ; Hacer que parpadee un LED conectado a RB7 cada segundo.
02  ; Además dos entradas en RA0, RA1 deben ser copiadas en
03  ;  dos salidas RB0, RB1
04  ; El uC funciona a 4 MHz
05  ;
06  ;   Temp = 4 * 250 nseg * (244 * 256 + 2) = 62.466 ms
07  ;
08  ; Si contamos 16 pasadas => 16*62.466 = 999.456 ms
09
10      LIST    P=16F84A
11      RADIX   DEC
12      INCLUDE "P16F84A.INC"
13      ERRORLEVEL -302
14      __CONFIG _WDT_OFF & _CP_OFF & _XT_OSC & _PWRTE_ON
15
16  MiContador  EQU     12      ; Primer registro de RAM
17
18      ORG     0
19      GOTO    Inicio
20
21  ; Interrupciones
22      ORG     4
23      decfsz  MiContador,F    ; Dec MiContador y salta instr si cero
24      goto    Seguir
25  Conta_0
26      movlw   16
27      movwf   MiContador      ; Recago MiContador
28
29      btfsc   PORTB,7         ; Conmuta el valor de RB7
30      goto    rB7_1
31      bsf     PORTB,7
32      goto    Seguir
33  rB7_1
34      bcf     PORTB,7
35
36  Seguir
37      movlw   b'10100000'
38      movwf   INTCON
39      movlw   256-244
40      movwf   TMR0
41      retfie
42
```

Estudio a fondo del PIC 16F84 : *Características especiales*

Ejemplo

```
43  ; Programa principal
44  Inicio
45      bsf STATUS,RP0          ; Sel. Banco 1
46      clrf    TRISB           ; PORTB todo salidas
47      movlw   b'00000011'
48      movwf   TRISA           ; PORTA[0,1] entradas
49      movlw   b'00000111'
50      movwf   OPTION_REG      ; Define:
51                              ; TMR0 con Prescaler 1:256
52      bcf     STATUS,RP0      ; Sel. Banco 0
53      movlw   b'10100000'     ;
54      movwf   INTCON          ; GIE=1 y T0IE=1
55      movlw   16
56      movwf   MiContador      ; Guardo el valor máximo de repeticiones
57      movlw   256-244
58      movwf   TMR0            ; Inicia la cuenta
59
60  Bucle
61      btfsc   PORTA,0         ; Copia RB0 = RA0
62      goto    rA0_1
63      bcf     PORTB,0
64      goto    ralx
65  rA0_1
66      bsf     PORTB,0
67
68  ralx
69      btfsc   PORTA,1         ; Copia RB1 = RA1
70      goto    rA1_1
71      bcf     PORTB,1
72      goto    Bucle
73  rA1_1
74      bsf     PORTB,1
75      goto    Bucle
76
77      END
78
```


Estudio a fondo del PIC 16F84 : *Características especiales*

Watch dog Timer

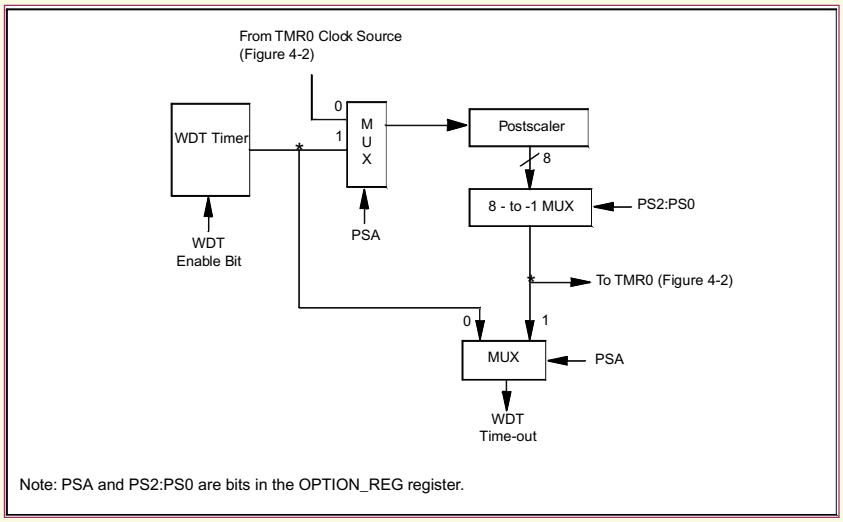
Características:

- Es un temporizador con un **oscilador RC interno**.
- Si el uC está en **modo normal**, un desbordamiento del WDT provoca un ***reset***.
- Si está en **modo SLEEP**, un desbordamiento del WDT **despierta** al uC.
- Se puede **desactivar permanentemente**.
- El periodo nominal de desbordamiento es de **18 ms**, aunque varía con la *temperatura* y el *voltaje* de alimentación.
- Se puede aumentar ese tiempo utilizando el módulo escalador (hasta 1:128, lo que da **2.3 segundos**)
- Las instrucciones CLRWDT y SLEEP borran el contaje del WDT y de la post-escala.
- El bit STATUS.TO vale 0 después de un desbordamiento del WDT.

Estudio a fondo del PIC 16F84 : *Características especiales*

Watch dog Timer: diagrama de bloques

Se utiliza para **evitar** que el uC entre en un **estado indeterminado**, así que el programador está obligado a **refrescar** el temporizador (ponerlo a cero con CLRWDT cada cierto número de instrucciones).



Estudio a fondo del PIC 16F84 : *Características especiales*

Power-down Mode (SLEEP)

Características:

- Es un modo de **bajo consumo** caracterizado por la congelación del oscilador externo.
- Para entrar en este modo debe ejecutarse la **instrucción** SLEEP.
- El WDT es refrescado (puesto a cero), pero no congelado.
- Se señaliza el modo SLEEP con **STATUS.PD=0** y **STATUS.TO=0.**
- Los pines de E/S mantienen su estado.
- Para **despertar** al uC:

1.- Reset externo en /MCLR.

2.- Desbordamiento WDT.

3.- Interrupción de RB0/INT, cambio de RB o escritura completa en EEPROM.

Estudio a fondo del PIC 16F84 : *Características especiales*

Identificación

- Existen 4 posiciones de memoria en el espacio de configuración y test que permiten almacenar **información de identificación**.
- Abarcan las posiciones **2000h-2003h** y sólo se pueden utilizar los *4 bits menos significativos* de cada palabra (14 bit).
- Se suele almacenar en ellos el *checksum* del programa grabado o alguna información para identificar la **versión** del mismo.

Estudio a fondo del PIC 16F84 : *Instrucciones*

Resumen de las 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	

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.

Estudio a fondo del PIC 16F84 : *Instrucciones*

Revisión de las instrucciones

ADDLW	Add Literal and W
Syntax:	[label] ADDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) + k → (W)
Status Affected:	C, DC, Z
Description:	The contents of the W register are added to the eight bit literal 'k' and the result is placed in the W register.

ANDLW	AND Literal with W
Syntax:	[label] ANDLW k
erands:	0 ≤ k ≤ 255
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Description:	The contents of W register are AND'ed with the eight bit literal 'k'. The result is placed in the W register.

ADDWF	Add W and f
Syntax:	[label] ADDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) → (destination)
Status Affected:	C, DC, Z
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'.

ANDWF	AND W with f
Syntax:	[label] ANDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .AND. (f) → (destination)
Status Affected:	Z
Description:	AND 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'.

Estudio a fondo del PIC 16F84 : *Instrucciones*

Revisión de las instrucciones

BCF	Bit Clear f
Syntax:	[label] BCF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	0 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is cleared.

BTFSS	Bit Test f, Skip if Set
Syntax:	[label] BTFSS f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b < 7
Operation:	skip if (f<b>) = 1
Status Affected:	None
Description:	If bit 'b' in register 'f' is '0', 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.

BSF	Bit Set f
Syntax:	[label] BSF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	1 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is set.

BTFSC	Bit Test, Skip if Clear
Syntax:	[label] BTFSC f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	skip if (f<b>) = 0
Status Affected:	None
Description:	If bit 'b' in register 'f' is '1', the next instruction is executed. If bit 'b', in register 'f', is '0', the next instruction is discarded, and a NOP is executed instead, making this a 2TCY instruction.

Estudio a fondo del PIC 16F84 : *Instrucciones*

Revisión de las instrucciones

CALL	Call Subroutine
Syntax:	[label] CALL k
Operands:	0 ≤ k ≤ 2047
Operation:	(PC)+ 1→ TOS, k → PC<10:0>, (PCLATH<4:3>) → PC<12:11>
Status Affected:	None
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.

CLRF	Clear f
Syntax:	[label] CLRF f
Operands:	0 ≤ f ≤ 127
Operation:	00h → (f) 1 → Z
Status Affected:	Z
Description:	The contents of register 'f' are cleared and the Z bit is set.

CLRW	Clear W
Syntax:	[label] CLRW
Operands:	None
Operation:	00h → (W) 1 → Z
Status Affected:	Z
Description:	W register is cleared. Zero bit (Z) is set.

CLRWDT	Clear Watchdog Timer
Syntax:	[label] CLRWDT
Operands:	None
Operation:	00h → WDT 0 → WDT prescaler, 1 → $\overline{TO}$ 1 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Description:	CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.

Estudio a fondo del PIC 16F84 : *Instrucciones*

Revisión de las instrucciones

COMF	Complement f
Syntax:	[label] COMF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	($\bar{f}$) → (destination)
Status Affected:	Z
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'.

DECF	Decrement f
Syntax:	[label] DECF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) - 1 → (destination)
Status Affected:	Z
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'.

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
Description:	The contents of register 'f' are decremented. 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 instruction.

GOTO	Unconditional Branch
Syntax:	[label] GOTO k
Operands:	0 ≤ k ≤ 2047
Operation:	k → PC<10:0> PCLATH<4:3> → PC<12:11>
Status Affected:	None
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.

Estudio a fondo del PIC 16F84 : *Instrucciones*

Revisión de las instrucciones

INCF	**Increment f**
Syntax:	[label] INCF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) + 1 → (destination)
Status Affected:	Z
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'.

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
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.

IORLW	**Inclusive OR Literal with W**
Syntax:	[label] IORLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .OR. k → (W)
Status Affected:	Z
Description:	The contents of the W register are OR'ed with the eight bit literal 'k'. The result is placed in the W register.

IORWF	**Inclusive OR W with f**
Syntax:	[label] IORWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .OR. (f) → (destination)
Status Affected:	Z
Description:	Inclusive OR the W register with register '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'.

Estudio a fondo del PIC 16F84 : *Instrucciones*

Revisión de las instrucciones

MOVF	**Move f**
Syntax:	[label] MOVF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) → (destination)
Status Affected:	Z
Description:	The contents of register f are 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 register since status flag Z is affected.

MOVLW	**Move Literal to W**
Syntax:	[label] MOVLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (W)
Status Affected:	None
Description:	The eight bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.

MOVWF	**Move W to f**
Syntax:	[label] MOVWF f
Operands:	0 ≤ f ≤ 127
Operation:	(W) → (f)
Status Affected:	None
Description:	Move data from W register to register 'f'.

NOP	**No Operation**
Syntax:	[label] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Description:	No operation.

Estudio a fondo del PIC 16F84 : *Instrucciones*

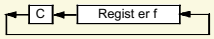
Revisión de las instrucciones

RETFIE	**Return from Interrupt**
Syntax:	[label] RETFIE
Operands:	None
Operation:	TOS → PC, 1 → GIE
Status Affected:	None

RETLW	**Return with Literal in W**
Syntax:	[label] RETLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (W); TOS → PC
Status Affected:	None
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.

RETURN	**Return from Subroutine**
Syntax:	[label] RETURN
Operands:	None
Operation:	TOS → PC
Status Affected:	None
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.

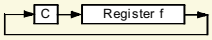
RLF	**Rotate Left f through Carry**
Syntax:	[label] RLF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	See description below
Status Affected:	C
Description:	The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd'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



Estudio a fondo del PIC 16F84 : *Instrucciones*

Revisión de las instrucciones

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
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'.



SLEEP	
Syntax:	[label] SLEEP
Operands:	None
Operation:	00h → WDT, 0 → WDT prescaler, 1 → $\overline{TO}$, 0 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Description:	The power-down status bit, $\overline{PD}$ is cleared. Time-out status bit, $\overline{TO}$ is set. Watchdog Timer and its prescaler are cleared. The processor is put into SLEEP mode with the oscillator stopped.

SUBLW	**Subtract W from Literal**
Syntax:	[label] SUBLW k
Operands:	0 ≤ k ≤ 255
Operation:	k - (W) → (W)
Status Affected:	C, DC, Z
Description:	The W register is subtracted (2's complement method) from the eight bit literal 'k'. The result is placed in the W register.

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
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'.

Estudio a fondo del PIC 16F84 : *Instrucciones*

Revisión de las instrucciones

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
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'.

XORLW	**Exclusive OR Literal with W**
Syntax:	[label] XORLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .XOR. k → (W)
Status Affected:	Z
Description:	The contents of the W register are XOR'ed with the eight bit literal 'k'. The result is placed in the W register.

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
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'.

Estudio a fondo del PIC 16F84 : *Instrucciones*

Ejemplos: *Dado electrónico*

```
01  ; Dado electronico. Cada vez que se presiona un pulsador se genera
02  ; un número aleatorio entre el 1 y el 6, que se visualiza a través
03  ; de un display de 7 segmentos directamente.
04      LIST P=16F84A
05      RADIX DEC
06      INCLUDE "P16F84A.INC"
07      ERRORLEVEL -302
08      __CONFIG _CP_OFF & _WDT_OFF & _XT_OSC & _PWRTE_ON
09
10  Resultado EQU 12
11
12      ORG 0
13  Inicio
14      BSF     STATUS,RP0  ; Sel. Banco 1
15      MOVLW   B'00000001'
16      MOVWF   TRISA       ; RA0 entrada
17      CLRF    TRISB       ; PORTB todo salidas
18      BCF     STATUS,RP0  ; Sel. Banco 0
19      MOVLW   1
20      MOVWF   Resultado
21
22  Bucle
23      CALL    Aleatorio   ; Calcula un número aleatorio (en W)
24      CALL    Dec7Seg     ; Traduce a 7 seg (de W a W)
25      MOVWF   PORTB       ; Muestra el resultado
26      GOTO    Bucle
```

Estudio a fondo del PIC 16F84 : *Instrucciones*

Ejemplos: *Dado electrónico*

```
27
28  Dec7Seg
29      ADDWF   PCL,F
30      RETLW   0X79        ; "0"
31      RETLW   0X06
32      RETLW   0X5B
33      RETLW   0X4F
34      RETLW   0X66
35      RETLW   0X6D
36      RETLW   0X7D        ; "6"
37
38  Aleatorio
39      BTFSC   PORTA,0     ; Usa el pulsador para contar
40                          ; más o menos y dar un valor
41                          ; aleatorio
42      GOTO    Volver
43      MOVLW   6
44      SUBWF   Resultado,W ; Comparo con 6
45      BTFSC   STATUS,Z
46      CLRF    Resultado
47      INCF    Resultado,F
48      GOTO    Aleatorio
49  Volver
50      MOVWF   Resultado
51      RETURN
52      END
53
```


Estudio a fondo del PIC 16F84 : *Instrucciones*

Ejemplos: *Alarma*

```
01  ; Se trata de controlar una alarma. En RB5- RB7 hay cuatro sensores a las
02  ; cuatro puertas de un coche. Cuando una puerta es abierta, el sensor se
03  ; pone a uno y debe activar la alarma realizada mediante dos zumbadores
04  ; conectados a RA0 y RA1.
05  ; Para detener la alarma, el usuario debe mandar una señal infrarroja a un sensor
06  ; que cuando la detecta activa la patilla RB0/INT y provoca una interrupción
07  ; parando los zumbadores y activando deee nuevo el sistema.
08      LIST P=16F84A
09      RADIX   DEC
10      INCLUDE "P16F84A.INC"
11      ERRORLEVEL -302
12      __CONFIG _WDT_OFF & _XT_OSC & _PWRTE_ON & _CP_OFF
13
14      ORG     0
15      GOTO    Inicio
16      ORG     4
17      GOTO    Interrupcion
18
19  Inicio
20      BSF     STATUS,RP0      ; Sel. Banco 1
21      MOVLW   B'00000000'
22      MOVWF   TRISA           ; PORTA todo salidas
23      MOVLW   B'11111111'
24      MOVWF   TRISB           ; PORTB todo entradas
25      BCF     STATUS,RP0      ; Sel. Banco 0
26      CLRF    PORTA
27      CLRF    PORTB
28      MOVLW   B'10011000'
29      MOVWF   INTCON          ; Activo GIE, RBIE, INTE
30  Bucle
31      GOTO    Bucle           ; Bucle sin fin
```

Estudio a fondo del PIC 16F84 : *Instrucciones*

Ejemplos: *Alarma*

```
32
33  Interrupcion
34      BTFSS   INTCON,RBIF     ; Comprueba int de RB4-RB7
35      GOTO    Parar
36  Alarma
37      CLRF    PORTB
38      MOVLW   B'10011000'
39      MOVWF   INTCON          ; Borro RBIF
40  Buzzer
41      BSF     PORTA,0
42      NOP
43      BCF     PORTA,0
44      BSF     PORTA,1
45      NOP
46      BCF     PORTA,1
47      GOTO    Buzzer          ; Hago sonar los zumbadores
48  Parar
49      CLRF    PORTA
50      BCF     PORTB,0         ; Cambio el estado de la puerta
51      MOVLW   B'10011000'     ; Borro INTF
52      MOVWF   INTCON
53      GOTO    Bucle           ; Notar que no retorno de interrupción
54      END
55
```


Microcontroladores PIC

Otros microcontroladores PIC

1. Familia de gama baja:	**12x**
2. Familia de gama media:	**16x**
3. Familia de gama alta:	**17x**

Otros microcontroladores PIC: *Familia 12xxx*

Características

- Tiene un tamaño reducido: 8 pines.
- 33/35 instrucciones de 12/14 bits.
- Pila hardware de 2 niveles.
- Alimentación 2.5 a 5.5 V. Consumo < 2 mA a 5V y 4 MHz.
- Disponen de oscilador RC interno. Pueden usar hasta 6 pines como E/S.

MODELO	MEMORIA DE PROGRAMA	MEMORIA DE DATOS	FREC	ADC 8 BIT	OTROS
PIC 12C508A	512x12	25X8	4 MHZ		TMR0+EDT
PIC 12C509A	1024x12	41X8	4 MHZ		TMR0+WDT
PIC 12C670	512x14	80X8	4 MHZ		TMR0+WDT
PIC 12C671	1024x14	128X8	4 MHZ	2	TMR0+WDT
PIC 12C672	2048x14	128X8	4 MHZ	4	TMR0+WDT
PIC 12CE673	1024x12 Flash	128X8, 16X8 EEPROM	4 MHZ	4	TMR0+WDT
PIC 12CE674	2048x14 Flash	128X8, 16X8 EEPROM	4 MHZ	4	TMR0+WDT

Otros microcontroladores PIC: *Familia 16xxx*

Características

- Encapsulados desde 18 a 68 pines.
- 35 instrucciones de 14 bits, compatibles con la gama baja.
- Interrupciones y pila hardware de 8 niveles.

Clasificación:

- Gama media estándar: PIC16C55x.
- Gama media con comparador analógico: PIC16C62x/64x/66x.
- Idem con módulos de captura (CCP), PWM y puerto serie: PIC16C6x.
- Idem con ADC de 8 bits: PIC16C7x.
- Idem con ADC de precisión: PIC14000.
- Gama media con Flash y EEPROM: PIC16x8x. (Incluye el 16F84).
- Gama media con *driver* LCD: PIC16C92x.

Otros microcontroladores PIC: *Familia 16xxx*

Modelos

MODELO	MEM. PROGRAMA	EEPROM	RAM	E/S	Pines	ADC	PWM	Timers/WDT	SERIAL
PIC 16F84A	1024X14 Flash	64	68	13	18/20			1x8,1	
PIC 16F870	2048x14 Flash	64	128	22	28	5x10 bit	1	1x16,2x8,1	USART
PIC 16F871	2048x14 Flash	64	128	33	40/44	8x10 bit	1	1x16,2x8,1	USART
PIC 16F872	2048x14 Flash	64	128	22	28	5x10 bit	1	1x16,2x8,1	MI2C/SPI
PIC 16F873	4096x14 Flash	128	192	22	28	5x10 bit	2	1x16,2x8,1	USART,MI2C/SPI
PIC 16F874	4096x14 Flash	128	192	33	40/44	8x10 bit	2	1x16,2x8,1	ESART,MI2C/SPI
PIC 16F876	8192x14 Flash	256	368	22	28	5x10 bit	2	1x16,2x8,1	USART,MI2C/SPI
PIC 16F877	8192x14 Flash	256	368	33	40/44	8x10 bit	2	1x16,2x8,1	USART,MI2C/SPI

Dependiendo de la aplicación:

- Se debe seleccionar el uC que **mejor** se **adapte** a la misma, sin sobredimensionarlo.
- Que a la vez sea el más **barato.**

Otros microcontroladores PIC: *Familia 17xxx*

Características

- 58 instrucciones de 16 bits.
- Interrupciones vectorizadas.
- Multiplicador hardware.
- Arquitectura abierta: memoria externa, periféricos externos, etc.

MODELO	MEMORIA DE PROGRAMA	RAM	E/S	Pines ADC		PWM	Timers WDT	Serie
PIC 17C42A	2048x16	232	33	40/44		2	2x16,2x8	USART
PIC 17C43	4096x16	454	33	40/44		2	2x16,2x8	USART
PIC 17C44	8192x16	454	33	40/44		2	2x16,2x8	USART
PIC 17C752	8192x16	678	50	64/68	12x10 bit	3	2x16,2x8	USARTx2,MI2C,SPI
PIC 17C756A	16384x16	902	50	64/68	12x10 bit	3	2x16,2x8	USARTx2,MI2C,SPI

Bibliografía

Predko. *Programming and customizing the PIC microcontroller*.
Tab Electronics, 1997. ISBN 007913646X.

Benson. *PIC'n up the pace.*
Square One Electronics, 1999. ISBN 0965416216.

Stevens, *Serial PIC'n.*
Square One Electronics, 1999. ISBN 0965416224.

Peatman. *Design with PIC microcontrollers.*
Prentice-Hall, 1997. ISBN 0137592590.

Tavernier. *Microcontroladores PIC.*
Paraninfo, 1997. ISBN 8428323739.

Martín Cuenca, Angulo Usategui, Angulo Martínez. *Microcontroladores PIC, la solución en un chip.*
Paraninfo, 1998. ISBN 8428323712.

Angulo Usategui, Angulo Martínez.
Microcontroladores PIC, diseño práctico de aplicaciones.
McGraw-Hill, 1999. ISBN 8448124960.

Angulo Usategui, Romero Yesa, Angulo Martínez.
Microcontroladores PIC, diseño práctico de aplicaciones: PIC 16F877.
McGraw-Hill, 2000. ISBN 8484128583.

Bibliografía

PIC 16F84A Datasheet.

PIC 16F877 Datasheet.

MPLAB Datasheet.

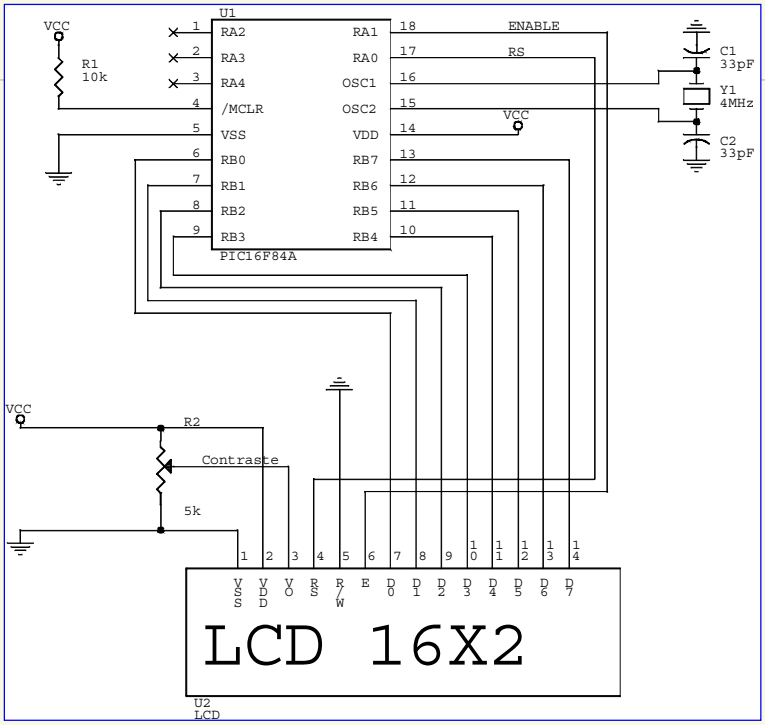
Anexo 1

Directivas del ensamblador

Directiva	Descripción	Sintaxis
_ _BADRAM	Especifica las posiciones del RAM inválidas.	_badram <expr>
BANKISEL	Genera el código que selecciona el banco de memoria de memoria RAM para direccionamiento indirecto.	bankisel <label>
BANKSEL	Genera el código que selecciona el código de memoria RAM.	banksel <label>
CBLOCK	Define un bloque de constantes.	cblock [<expr>]
CODE	Empieza la sección del código ejecutable.	[<name>] code [<address>]
_ _CONFIG	Especifica los bits de configuración.	_ _config <expr>
CONSTANT	Declara los simbolos de las constantes.	constant <label>[=<expr>,...,<label>[=<expr>]]
DATA	Crea datos numéricos y de texto.	[<label>] data <expr>,[,<expr>,...,<expr>]
		[<label>] data "<text_string>"[,"<text_string>",...]
DB	Declara datos de un byte.	[<label>] db <expr>[,<expr>,...,<expr>]
		[<label>] db "<text_string>"[,"<text_string>",...]
DE	Define los datos en la EEPROM.	[<label>] de <expr>[,<expr>,...,<expr>]
		[<label>] de "<text_string>"[,"<text_string>",...]
#DEFINE	Define una etiqueta de substitución de texto.	define <name> [<value>]
		define <name> [<arg>,...,<arg>] <value>
DT	Define tabla	[<label>] dt <expr>[,<expr>,...,<expr>]
		[<label>] dt "<text_string>"[,"<text_string>",...]
DW	Declara datos word	[<label>] dw <expr>[,<expr>,...,<expr>]
		[<label>] dw "<text_string>"[,"<text_string>",...]
ELSE	Empieza el bloque alternativo de un IF.	Else
END	Fin de bloque de programa.	End
ENDC	Acaba un bloque constante automático.	Endc
ENDIF	Fin del bloque de condiciones ensambladas.	Endif
ENDM	Fin de la definición de una macro.	Endm
ENDW	Fin de un bucle de while.	Endw
EQU	Define una constante para el ensamblador.	<label> equ <expr>
ERROR	Manda un mensaje de error.	error "<text_string>"
ERRORLEVEL	Activa/desactiva un error. Establece el nivel de errores a ver.	errorlevel 0 | 1 | 2 | <+ | ->message number>
EXITM	Salida de una macro.	Exitm
EXPAND	Expande una lista de macro.	expand
EXTERN	Declara una etiqueta externa	extern <label>[,<label>]
FILL	Llena la memoria.	[<label>] fill <expr>, <count>
GLOBAL	Exporta una etiqueta definida.	global <label>[,<label>]
IDATA	Comienza una sección de identificadores (ID).	[<name>] idata [<address>]
_ _IDLOCS	Especifica donde están colocados los identificadores (ID).	_ _idlocs <expr>
IF	Empieza un bloque de código condicional.	if <expr>
IFDEF	Ensambla si el simbolo ha sido definido.	ifdef <label>
IFNDEF	Ensambla si el simbolo no ha sido definido	ifndef <label>
#INCLUDE	Incluye ficheros fuentes adicionales.	include <<include_file>> |"<include_file>"
LIST	Opciones de listado.	list [<list_option>,...,<list_option>]
LOCAL	Declara una macro variable como local.	local <label>[,<label>]
MACRO	Declara la definición del macro.	<label> macro [<arg>,...,<arg>]
_ _MAXRAM	Especifica la dirección de RAM máxima.	_ _maxram <expr>
MESSG	Emite mensajes definidos por el usuario.	messg "<message_text>"
NOEXPAND	Termina la expansión del macro.	Noexpand
NOLIST	Termina el listado.	Nolist
ORG	Pone el origen del programa.	<label> org <expr>
PAGE	Inserta el número de página del listado.	Page
PAGESEL	Genere el código de selección de la página de ROM .	pagesel <label>
PROCESSOR	Define el tipo de procesador.	processor <processsor_type>
RADIX	Especifica el formato por defecto (radix) de los números.	radix <default_radix>
RES	Reserva memoria	[<label>] res <mem_units>
SET	Define variable de ensamblador.	<label> set <expr>
SPACE	Inserta lineas en blanco.	space <expr>
SUBTITLE	Especifica el subtitulo del programa.	subtitle "<sub_text>"
TITLE	Especifica el titulo del programa.	title "<title_text>"
UDATA	Empieza la sección de datos no inicializados.	[<name>] udata [<address>]
UDATA_OVR	Empieza la sección de datos no inicializados superpuestos.	[<name>] udata_ovr [<address>]
UDATA_SHR	Empieza la sección de datos no inicializados compartidos.	[<name>] udata_shr [<address>]
#UNDEFINE	Anula una etiqueta de la substitución.	#undefine <label>
VARIABLE	Declara un simbolo como variable.	variable <label>[=<expr>,...,<label>[=<expr>]]
WHILE	Realiza el bucle mientras la condición es verdadera.	while <expr>

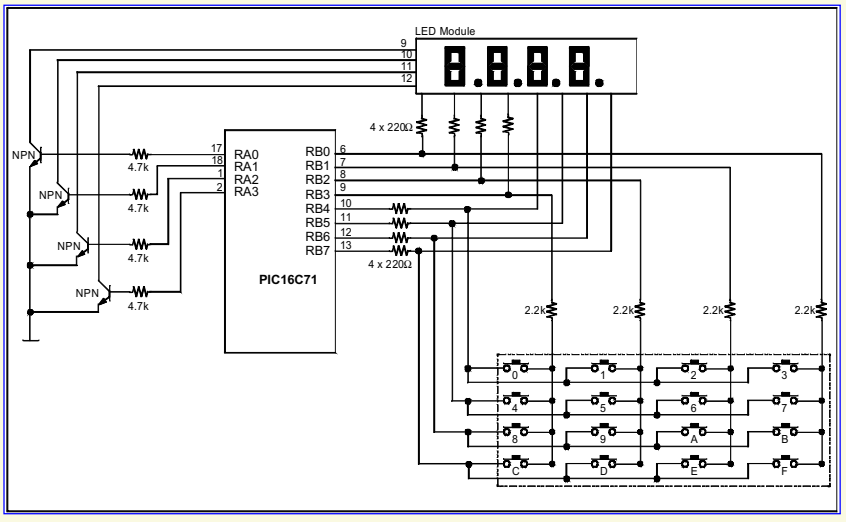
Anexo 2

Esquema PIC-LCD



Anexo 3

Esquema PIC-Teclado



Anexo 4

Esquema Grabador PIC

Programador JDM de muy bajo coste (16F84)