

# ESCUELA EDUCACIÓN SECUNDARIA TÉCNICA N° 8 - MORÓN

## Aplicaciones Electronica Digital III

### Capítulo 1

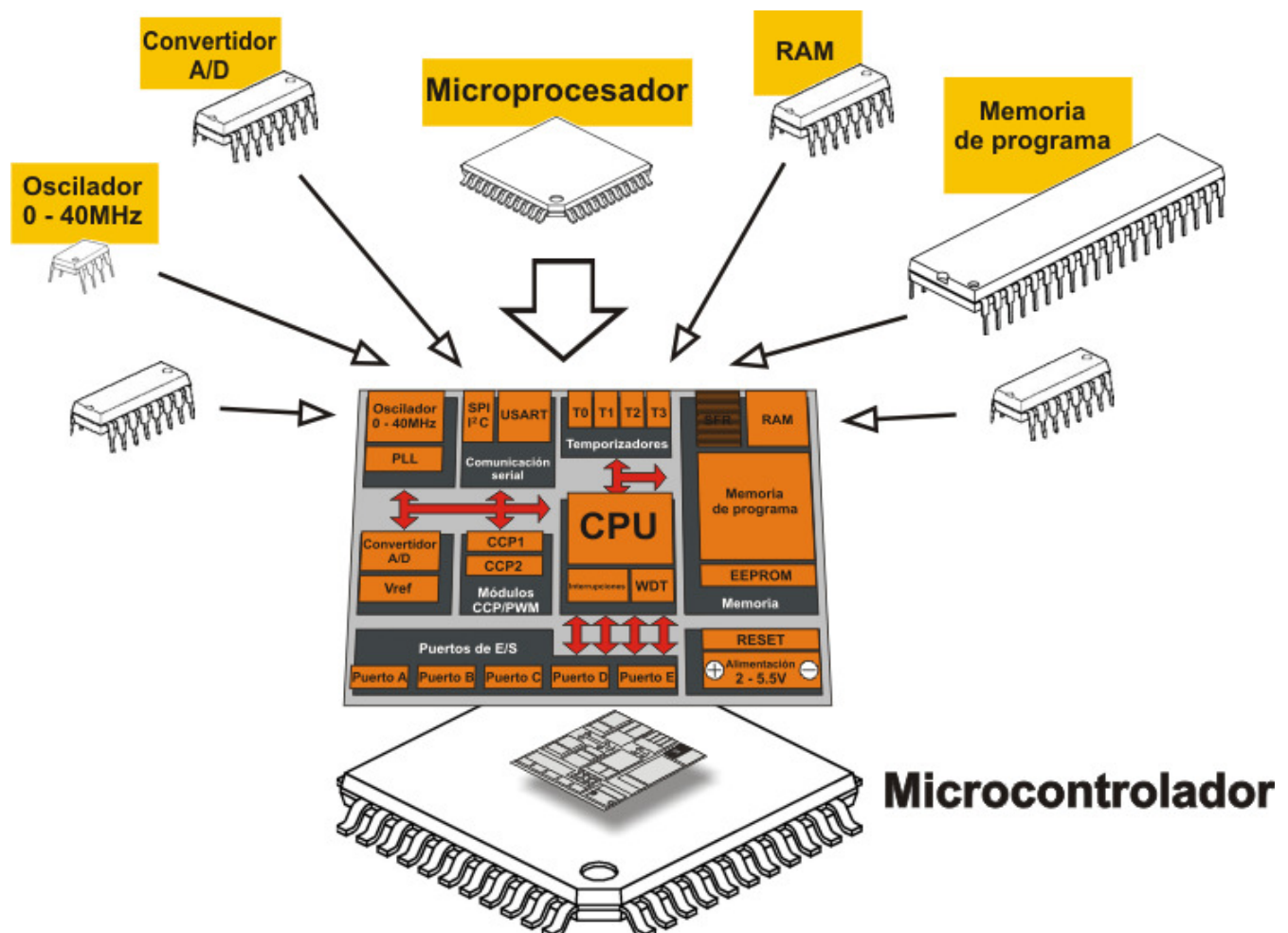
#### Capítulo 1: El mundo de los microcontroladores

La situación actual en el campo de los microcontroladores se ha producido gracias al desarrollo de la tecnología de fabricación de los circuitos integrados. Este desarrollo ha permitido construir las centenas de miles de transistores en un chip. Esto fue una condición previa para la fabricación de un microprocesador. Las primeras microcomputadoras se fabricaron al añadirles periféricos externos, tales como memoria, líneas de entrada/salida, temporizadores u otros. El incremento posterior de la densidad de integración permitió crear un circuito integrado que contenía tanto al procesador como periféricos. Así es cómo fue desarrollada la primera microcomputadora en un solo chip, denominada más tarde microcontrolador.

- 1.1 Introducción
- 1.2 NÚMEROS, NÚMEROS, NÚMEROS...
- 1.3 DETALLES IMPORTANTES
- 1.4 MICROCONTROLADORES PIC

#### 1.1 INTRODUCCIÓN

Los principiantes en electrónica creen que un microcontrolador es igual a un microprocesador. Esto no es cierto. Difieren uno del otro en muchos sentidos. La primera y la más importante diferencia es su funcionalidad. Para utilizar al microprocesador en una aplicación real, se debe de conectar con componentes tales como memoria o componentes buses de transmisión de datos. Aunque el microprocesador se considera una máquina de computación poderosa, no está preparado para la comunicación con los dispositivos periféricos que se le conectan. Para que el microprocesador se comunique con algún periférico, se deben utilizar los circuitos especiales. Así era en el principio y esta práctica sigue vigente en la actualidad.



Por otro lado, al microcontrolador se le diseña de tal manera que tenga todas las componentes integradas en el mismo chip. No necesita de otros componentes especializados para su aplicación, porque todos los circuitos necesarios, que de otra manera correspondan a los periféricos, ya se encuentran incorporados. Así se ahorra tiempo y espacio necesario para construir un dispositivo.

## ¿QUE PUEDEN HACER LOS MICROCONTROLADORES?

Para entender con más facilidad las razones del éxito tan grande de los microcontroladores, vamos a prestar atención al siguiente ejemplo. Hace unos 10 años, diseñar un dispositivo electrónico de control de un ascensor de un edificio de varios pisos era muy difícil, incluso para un equipo de expertos. ¿Ha pensado alguna vez en qué requisitos debe cumplir un simple ascensor? ¿Cómo lidiar con la situación cuando dos o más personas llaman al ascensor al mismo tiempo? ¿Cuál llamada tiene la prioridad? ¿Cómo solucionar las cuestiones de seguridad, de pérdida de electricidad, de fallos, de uso indebido? Lo que sucede después de resolver estos problemas básicos es un proceso meticuloso de diseñar los dispositivos adecuados utilizando un gran número de los chips especializados. Este proceso puede tardar semanas o meses, dependiendo de la complejidad del dispositivo. Cuando haya terminado el proceso, llega la hora de diseñar una placa de circuito impreso y de montar el dispositivo. ¡Un dispositivo enorme! Es otro trabajo difícil y tardado. Por último, cuando todo está terminado y probado adecuadamente, pasamos al momento crucial y es cuando uno se concentra, respira profundamente y enciende la fuente de alimentación.

Esto suele ser el punto en el que la fiesta se convierte en un verdadero trabajo puesto que los dispositivos electrónicos casi nunca funcionan apropiadamente desde el inicio. Prepárese para muchas noches sin dormir, correcciones, mejoras... y no se olvide de que todavía estamos hablando de cómo poner en marcha un simple ascensor.

Cuando el dispositivo finalmente empiece a funcionar perfectamente y todo el mundo esté satisfecho, y le paguen por el trabajo que ha hecho, muchas compañías de desarrollo estarán interesadas en su trabajo. Por supuesto, si tiene suerte, cada día le traerá una oferta de trabajo de un nuevo inversionista. Sin embargo, si lo requieren para trabajar en el control de los elevadores de un nuevo edificio que tiene cuatro pisos más de los que ya maneja su sistema de control. ¿Sabe cómo proceder? ¿Cree acaso que se pueden controlar las demandas de sus clientes? Pensamos que usted va a construir un dispositivo universal que se puede utilizar en los edificios de 4 a 40 pisos, una obra maestra de electrónica. Bueno, incluso si usted consigue construir una joya electrónica, su inversionista le esperará delante de la puerta pidiendo una cámara en el ascensor o una música relajante en caso de fallo de ascensor. O un ascensor con dos puertas.

De todos modos, la ley de Murphy es inexorable y sin duda usted no podrá tomar ventaja a pesar de todos los esfuerzos que ha hecho. Por desgracia, todo lo que se ha dicho hasta ahora sucede en la realidad. Esto es lo que "dedicarse a la ingeniería electrónica" realmente significa. Es así como se hacían las cosas hasta aparición de los microcontroladores diseñados - pequeños, potentes y baratos. Desde ese momento su programación dejó de ser una ciencia, y todo tomó otra dirección ...

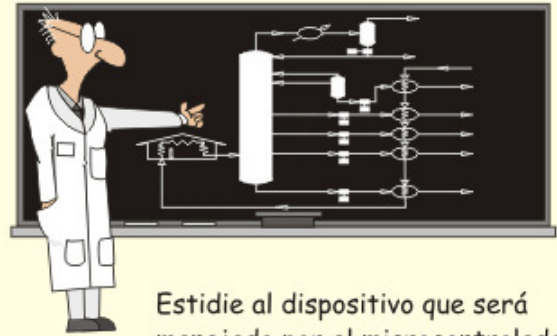
El dispositivo electrónico capaz de controlar un pequeño submarino, una grúa o un ascensor como el anteriormente mencionado, ahora está incorporado en un sólo chip. Los microcontroladores ofrecen una amplia gama de aplicaciones y sólo algunas se exploran normalmente. Le toca a usted decidir qué quiere que haga el microcontrolador y cargar un programa en él con las instrucciones apropiadas. Antes de encender el dispositivo es recomendable verificar su funcionamiento con ayuda de un simulador. Si todo funciona como es debido, incorpore el microcontrolador en el sistema. Si alguna vez necesita cambiar, mejorar o actualizar el programa, hágalo. ¿Hasta cuándo? Hasta quedar satisfecho. Eso puede realizarse sin ningún problema.

## Programación para los principiantes



Diga adiós a su familia y a sus parientes por un par de días...

...Provea a su mascota de comida y prepare suficientes bocadillos

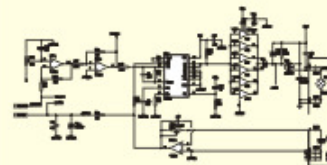


Estudie al dispositivo que será manejado por el microcontrolador.



Examine todos los microcontroladores disponibles y sus características (número de entradas/salidas, temporizadores, convertidores A/D etc.)

Elija el que pueda cumplir con los requisitos del funcionamiento del sistema deseado.



Diseñe y si es posible construya un prototipo que será controlado por el microcontrolador incluyendo a los dispositivos periféricos que se utilizarán en la aplicación real.



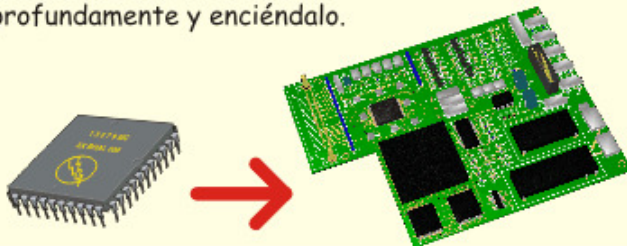
Utilice una PC y alguno de los lenguajes de programación de alto nivel disponibles para escribir un programa que ejecutará el microcontrolador. También utilice un programa para la simulación del entorno real en el que se desempeñará el sistema. ¡Qué estupendo se van desarrollando las cosas!



Al pulsar sobre de presión apropiado el programa entero se convierte en código máquina comprensible para el microcontrolador. Utilice un programador para grabar este código en la memoria del microcontrolador.



Es hora de que el microcontrolador empiece su vida productiva. Coloque al chip programado del programador al dispositivo destino (recién creado), respire profundamente y enciéndalo.



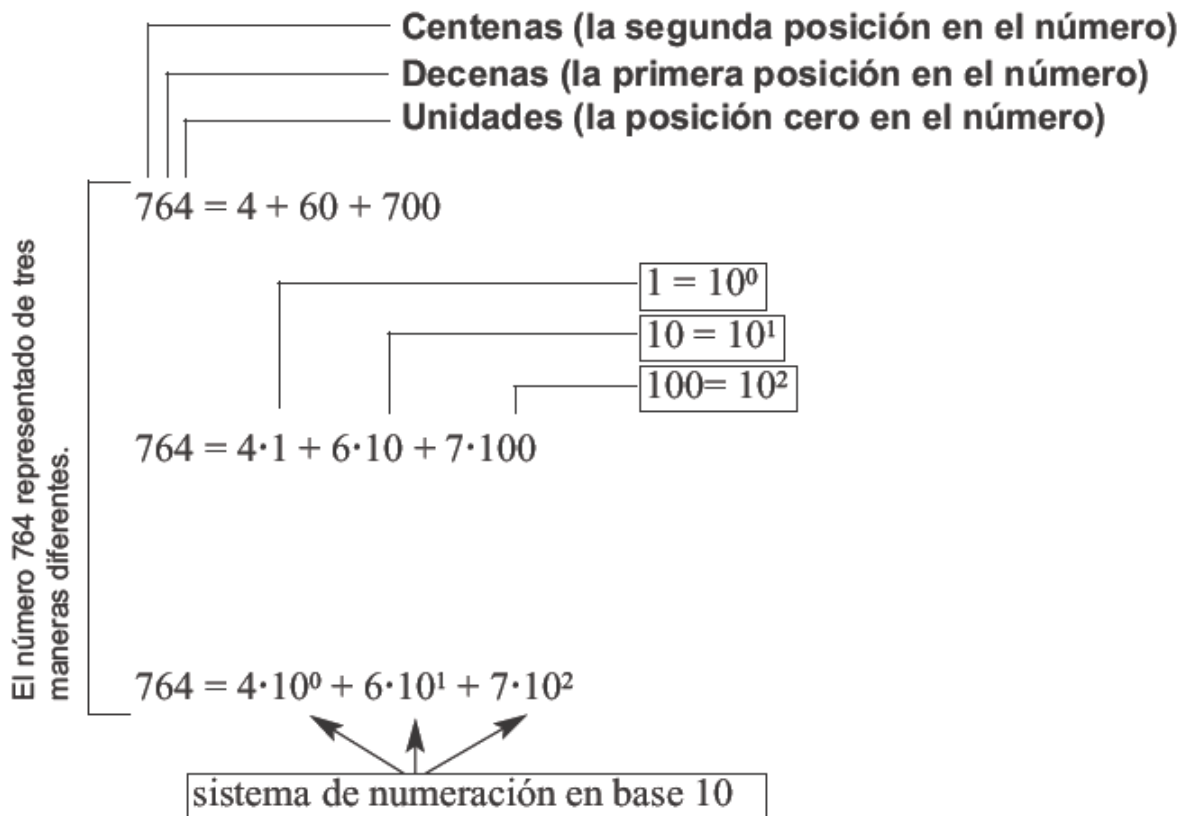
¡Esto es todo! Disfrute del éxito y empiece a pensar en los nuevos proyectos...



Si desea cambiar su estado, lea el siguiente texto que describe brevemente algunos de los conceptos básicos utilizados más tarde en este libro (sólo para estar seguro de que estamos hablando en los mismos términos).

## 1.2 NÚMEROS, NÚMEROS, NÚMEROS...

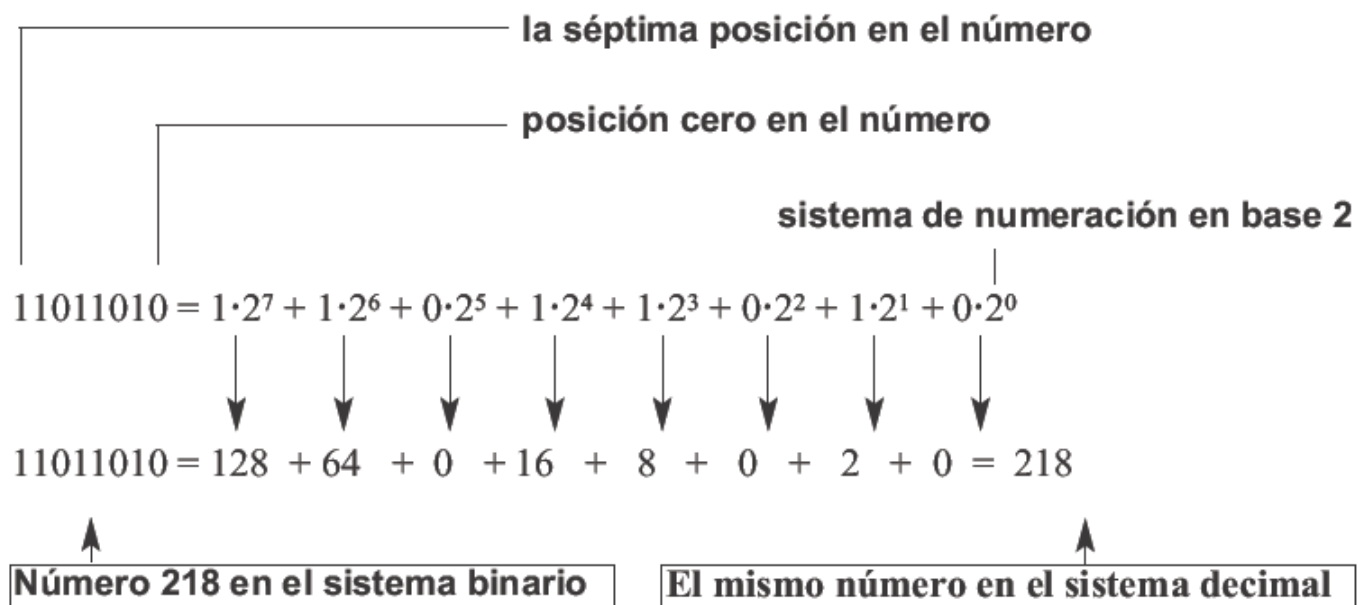
¡La matemática es una gran ciencia! Todo es tan lógico y simple... El universo de los números se puede describir con sólo diez dígitos. No obstante, ¿realmente tiene que ser así? ¿Necesitamos exactamente esos 10 dígitos? Por supuesto que no, es sólo cuestión del hábito. Acuérdesse de las lecciones de la escuela. Por ejemplo, ¿qué significa el número 764? Cuatro unidades, seis decenas y siete centenas. ¡Muy simple! ¿Se podría expresar de una forma más desarrollada? Por supuesto que sí:  $4 + 60 + 700$ . ¿Aún más desarrollado? Sí:  $4 \cdot 1 + 6 \cdot 10 + 7 \cdot 100$ . ¿Podría este número parecer un poco más “científico”? La respuesta es sí otra vez:  $4 \cdot 100 + 6 \cdot 101 + 7 \cdot 102$ . ¿Qué significa esto realmente? ¿Por qué utilizamos exactamente estos números 100, 101 y 102? ¿Por qué es siempre el número 10? Es porque utilizamos 10 dígitos diferentes (0, 1, 2...8, 9). En otras palabras, es porque utilizamos el sistema de numeración en base 10, es decir el sistema de numeración decimal.



## SISTEMA DE NUMERACIÓN BINARIO

¿Qué pasaría si utilizáramos sólo dos números 0 y 1? Si sólo pudiéramos afirmar (1) o negar (0) que algo existe. La respuesta es “nada especial”, seguiríamos utilizando los mismos números de la misma manera que utilizamos hoy en día, no obstante ellos parecerían un poco diferentes. Por ejemplo: 11011010. ¿Cuántas son realmente 11011010 páginas de un libro? Para entenderlo, siga la misma lógica como en el ejemplo anterior, pero en el orden invertido. Tenga en cuenta que se trata de aritmética con sólo dos dígitos 0 y 1, es decir, del sistema de numeración en base 2 (sistema de numeración binario).





Evidentemente, se trata del mismo número representado en dos sistemas de numeración diferentes. La única diferencia entre estas dos representaciones yace en el número de dígitos necesarios para escribir un número. Un dígito (2) se utiliza para escribir el número 2 en el sistema decimal, mientras que dos dígitos (1 y 0) se utilizan para escribir aquel número en el sistema binario. ¿Ahora está de acuerdo que hay 10 grupos de gente? ¡Bienvenido al mundo de la aritmética binaria! ¿Tiene alguna idea de dónde se utiliza?

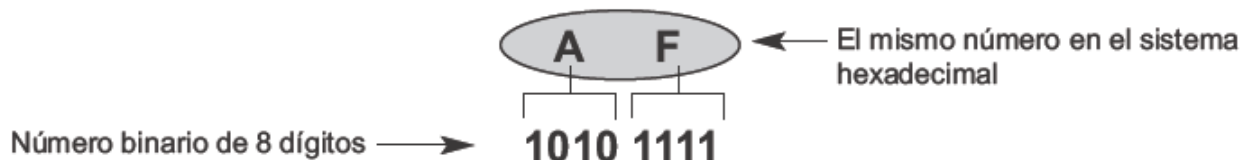
Excepto en las condiciones de laboratorio estrictamente controladas, los circuitos electrónicos más complicados no pueden especificar con exactitud la diferencia entre dos magnitudes (dos valores de voltaje, por ejemplo), si son demasiado pequeños (más pequeños que unos pocos voltios). La razón son los ruidos eléctricos y fenómenos que se presentan dentro de lo que llamamos "entorno de trabajo real" (algunos ejemplos de estos fenómenos son los cambios imprevisibles de la tensión de alimentación, cambios de temperatura, tolerancia a los valores de los componentes etc...). Imagínese una computadora que opera sobre números decimales al tratarlos de la siguiente manera: 0=0V, 1=5V, 2=10V, 3=15V, 4=20V... 9=45V!?

¿Alguien dijo baterías?

Una solución mucho más fácil es una lógica binaria donde 0 indica la ausencia de voltaje, mientras que 1 indica la presencia de voltaje. Simplemente, es fácil de escribir 0 o 1 en vez de "no hay voltaje" o "hay voltaje". Mediante el cero lógico (0) y uno lógico (1) la electrónica se enfrenta perfectamente y realiza con facilidad todas las operaciones aritméticas. Evidentemente, se trata de electrónica que en realidad aplica aritmética en la que todos los números son representados con sólo dos dígitos y donde sólo es importante saber si hay voltaje o no. Por supuesto, estamos hablando de electrónica digital.

## SISTEMA DE NUMERACIÓN HEXADECIMAL

En el principio del desarrollo de las computadoras era evidente que a la gente le costaba mucho trabajar con números binarios. Por eso, se estableció un nuevo sistema de numeración, que utilizaba 16 símbolos diferentes. Es llamado el sistema de numeración hexadecimal. Este sistema está compuesto de 10 dígitos a los que estamos acostumbrados (0, 1, 2, 3,... 9) y de seis letras del alfabeto A, B, C, D, E y F. ¿Cuál es el propósito de esta combinación aparentemente extraña? Basta con mirar cómo todo en la historia de los números binarios encaja perfectamente para lograr una mejor comprensión del tema.



El mayor número que puede ser representado con 4 dígitos binarios es el número 1111. Corresponde al número 15 en el sistema decimal. En el sistema hexadecimal ese número se representa con sólo un dígito F. Es el mayor número de un dígito en el sistema hexadecimal. ¿Se da cuenta de la gran utilidad de estas equivalencias? El mayor número escrito con ocho dígitos binarios es a la vez el mayor número de dos dígitos en el sistema hexadecimal. Tenga en cuenta que una computadora utiliza números binarios de 8 dígitos. ¿Acaso se trata de una casualidad?

## CÓDIGO BCD

El código BCD (*Binary-Coded Decimal* - Código binario decimal) es un código binario utilizado para representar a los números decimales. Se utiliza para que los circuitos electrónicos puedan comunicarse con los periféricos utilizando el sistema de numeración decimal o bien utilizando

el sistema binario dentro de “su propio mundo”. Consiste en números binarios de 4 dígitos que representan los primeros diez dígitos (0, 1, 2, 3...8, 9). Aunque cuatro dígitos pueden hacer 16 combinaciones posibles en total, el código BCD normalmente utiliza a las primeras diez.

## CONVERSIÓN DE SISTEMAS DE NÚMERACIÓN

El sistema de numeración binario es el que utilizan los microcontroladores, el sistema decimal es el que nos resulta más comprensible, mientras que el sistema hexadecimal presenta un balance entre los dos. Por eso, es muy importante aprender cómo convertir los números de un sistema de numeración a otro, por ejemplo, cómo convertir una serie de ceros y unos a una forma de representación comprensible para nosotros.

### CONVERSIÓN DE NÚMEROS BINARIOS A DECIMALES

Los dígitos en un número binario tienen ponderaciones diferentes lo que depende de sus posiciones dentro del número que están representando. Además, cada dígito puede ser 1 o 0, y su ponderación se puede determinar con facilidad al contar su posición empezando por la derecha. Para hacer una conversión de un número binario a decimal es necesario multiplicar las ponderaciones con los dígitos correspondientes (0 o 1) y sumar todos los resultados. La magia de la conversión de un número binario a decimal funciona de maravilla...

¿Tiene duda? Veamos el siguiente ejemplo:

Número binario	El mismo número en el sistema decimal
110	$1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 6$

Cabe destacar que es necesario utilizar sólo dos dígitos binarios para representar a todos los números decimales de 0 a 3. Por consiguiente, para representar los números de 0 a 7 es necesario utilizar tres dígitos binarios, para representar los números de 0 a 15 - cuatro dígitos etc. Dicho de manera sencilla, el mayor número binario que se puede representar utilizando n dígitos se obtiene al elevar la base 2 a la potencia n. Luego, al resultado se le resta 1. Por ejemplo, si n=4:

$$2^4 - 1 = 16 - 1 = 15$$

Por consiguiente, al utilizar 4 dígitos binarios, es posible representar los números decimales de 0 a 15, que son 16 valores diferentes en total.

### CONVERSIÓN DE NÚMEROS HEXADECIMALES A DECIMALES

Para realizar una conversión de un número hexadecimal a decimal, cada dígito hexadecimal debe ser multiplicado con el número 16 elevado al valor de su posición. Por ejemplo:

A37E (número hexadecimal)

14	$\cdot 16^0 = 14 \cdot 1$	= 14
7	$\cdot 16^1 = 7 \cdot 16$	= 112
3	$\cdot 16^2 = 3 \cdot 256$	= 768
10	$\cdot 16^3 = 10 \cdot 4096$	= 40960

41854 (el mismo número en el sistema decimal)

### CONVERSIÓN DE NÚMEROS HEXADECIMALES A BINARIOS

No es necesario realizar ningún cálculo para convertir un número hexadecimal a binario. Los dígitos hexadecimales se reemplazan simplemente por los cuatro dígitos binarios apropiados. Ya que el dígito hexadecimal máximo es equivalente al número decimal 15, es necesario utilizar cuatro dígitos binarios para representar un dígito hexadecimal. Por ejemplo:

$$E4 = 11100100$$

E						4	

## MARCAR LOS NÚMEROS

El sistema de numeración hexadecimal, junto con los sistemas binario y decimal, se consideran los más importantes para nosotros. Es fácil realizar una conversión de cualquier número hexadecimal a binario, además es fácil de recordarlo. Sin obstante, estas conversiones pueden provocar una confusión. Por ejemplo, ¿qué significa en realidad la sentencia: “Es necesario contar 110 productos en una cadena de

montaje"? Dependiendo del sistema en cuestión (binario, decimal o hexadecimal), el resultado podría ser 6, 110 o 272 productos, respectivamente. Por consiguiente, para evitar equivocaciones, diferentes prefijos y sufijos se añaden directamente a los números. El prefijo \$ o 0x así como el sufijo h marca los números en el sistema hexadecimal. Por ejemplo, el número hexadecimal 10AF se puede escribir así: \$10AF, 0x10AF o 10AFh. De manera similar, los números binarios normalmente obtienen el sufijo % o 0B. Si un número no tiene ni sufijo ni prefijo se considera decimal. Desafortunadamente, esta forma de marcar los números no es estandarizada, por consiguiente depende de la aplicación concreta.

La siguiente es tabla comparativa que contiene los valores de números 0-255 representados en tres sistemas de numeración diferentes. Esto es probablemente la manera más fácil de entender lógica común aplicada a todos los sistemas de numeración.

DEC.	BINARIO								HEX.
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	1
2	0	0	0	0	0	0	1	0	2
3	0	0	0	0	0	0	1	1	3
4	0	0	0	0	0	1	0	0	4
5	0	0	0	0	0	1	0	1	5
6	0	0	0	0	0	1	1	0	6
7	0	0	0	0	0	1	1	1	7
8	0	0	0	0	1	0	0	0	8
9	0	0	0	0	1	0	0	1	9
10	0	0	0	0	1	0	1	0	A
11	0	0	0	0	1	0	1	1	B
12	0	0	0	0	1	1	0	0	C
13	0	0	0	0	1	1	0	1	D
14	0	0	0	0	1	1	1	0	E
15	0	0	0	0	1	1	1	1	F
16	0	0	0	1	0	0	0	0	10
17	0	0	0	1	0	0	0	1	11
.....									
.....									
.....									
253	1	1	1	1	1	1	0	1	FD
254	1	1	1	1	1	1	1	0	FE
255	1	1	1	1	1	1	1	1	FF

## NÚMEROS NEGATIVOS

Como ya hemos visto, para escribir un número negativo en matemáticas, basta con añadirle el prefijo "-" (signo menos). Sin embargo, en la programación, al acabarse el proceso de la compilación, se quedan sólo los números binarios, volviéndose las cosas más complicadas. Se utilizan sólo dos dígitos - 0 y 1, mientras que todos los demás números, símbolos y signos se forman por medio de las combinaciones de estos dos dígitos. En el caso de los números negativos, la solución es la siguiente: En los números negativos, el bit más significativo (el bit del extremo izquierdo) representa el signo del número (donde 0 será positivo y 1 negativo). En el caso de un número de 8 bits, para representar un valor numérico sólo quedan 7 bits. De acuerdo a este tipo de codificación el número +127 es el mayor número positivo con signo que se puede representar con 8 bits. Asimismo, hay cero (0) positivo y negativo (refiérase a la tabla de la derecha). La siguiente pregunta sería: ¿Cómo es posible saber de qué número se trata? Por ejemplo, si ponemos el número 10000001, ¿es -1 o 129? No se preocupe, de eso se encarga el compilador. Ésta es la razón por la que se declaran variables al escribir el programa. Bueno, de eso vamos a hablar a continuación.

Binario	con signo	sin signo
00000000	+0	0
00000001	1	1
...	...	...
01111111	127	127
10000000	-0	128
10000001	-1	129
...	...	...
11111111	-127	255

## BIT

La teoría dice que un bit es la unidad básica de información...Vamos a olvidarlo por un momento y demostrar qué es eso en la práctica. La respuesta es - nada especial- un bit es un sólo dígito binario. Similar a un sistema de numeración decimal en el que los dígitos de un número no tienen la misma ponderación (por ejemplo, los dígitos en el número 444 son los mismos pero tienen los valores diferentes), el “significado” de un bit depende de la posición que tiene en número binario. En este caso no tiene sentido hablar de unidades, centenas etc. en los números binarios, sus dígitos se denominan el bit cero (el primer bit a la derecha), el primer bit (el segundo bit a la derecha) etc. Además, ya que el sistema binario utiliza solamente dos dígitos (0 y 1), el valor de un bit puede ser 0 o 1.

No se confunda si se encuentra con un bit que tiene el valor 4, 16 o 64. Son los valores representados en el sistema decimal. Simplemente, nos hemos acostumbrado tanto a utilizar los números decimales que estas expresiones llegaron a ser comunes. Sería correcto decir por ejemplo, “el valor del sexto bit en cualquier número binario equivale al número decimal 64”. Pero todos somos humanos y los viejos hábitos mueren difícilmente. Además, ¿cómo le suena “número uno-uno-cero-uno-cero...”?

## BYTE

Un byte consiste en 8 bits colocados uno junto al otro. Si un bit es un dígito, es lógico que los bytes representen los números. Todas las operaciones matemáticas se pueden realizar por medio de ellos, como por medio de los números decimales comunes. Similar a los dígitos de cualquier número, los dígitos de un byte no tienen el mismo significado. El bit del extremo izquierdo tiene la mayor ponderación, por eso es denominado el bit más significativo (MSB). El bit del extremo derecho tiene la menor ponderación, por eso es denominado el bit menos significativo (LSB). Puesto que los 8 dígitos de un byte se pueden combinar de 256 maneras diferentes, el mayor número decimal que se puede representar con un byte es 255 (una combinación representa un cero).

Un nibble o un cuarteto representa una mitad de byte. Dependiendo de la mitad del número en cuestión (izquierda o derecha), se les denomina nibbles “altos” o “bajos”, respectivamente.



*Usted seguramente ha pensado alguna vez en cómo es la electrónica dentro de un circuito integrado digital, un microcontrolador o un microprocesador. ¿Cómo son los circuitos que realizan las operaciones matemáticas complicadas y toman decisiones? ¿Sabía que sus esquemas, aparentemente complicados consisten en sólo unos pocos elementos diferentes, denominados circuitos lógicos o compuertas lógicas?*

## 1.3 DETALLES IMPORTANTES

El funcionamiento de estos elementos es basado en los principios establecidos por el matemático británico *George Boole* en la mitad del siglo 19 - es decir, ¡antes de la invención de la primera bombilla! En breve, la idea principal era de expresar las formas lógicas por medio de las funciones algebraicas. Tal idea pronto se transformó en un producto práctico que se convirtió más tarde en lo que hoy en día conocemos como circuitos lógicos Y (AND), O (OR) o NO (NOT). El principio de su funcionamiento es conocido como álgebra de Boole.

## CIRCUITOS LÓGICOS

Algunas instrucciones de programa utilizadas por un microcontrolador funcionan de la misma manera que las compuertas lógicas, pero en forma de comandos. A continuación vamos a explicar el principio de su funcionamiento.

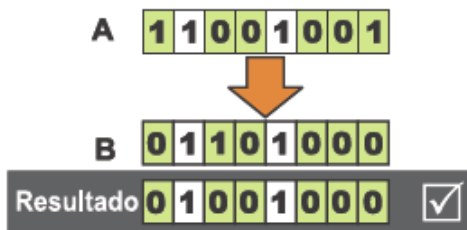


## COMPUERTA Y (AND)

Una compuerta lógica "Y" dispone de dos o más entradas y de una salida. En este caso la compuerta utilizada dispone de sólo dos entradas. Un uno lógico (1) aparecerá en su salida sólo en caso de que ambas entradas (A Y B) sean llevadas a alto (1). La tabla a la derecha es la tabla de verdad que muestra la relación entre las entradas y salidas de la compuerta. El principio de funcionamiento es el mismo cuando la compuerta disponga de más de dos entradas: la salida proporciona un uno lógico (1) sólo si todas las entradas son llevadas a alto (1).



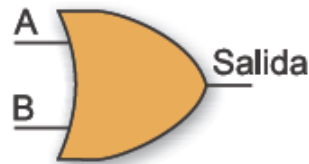
A	B	Salida
0	0	0
0	1	0
1	0	0
1	1	1



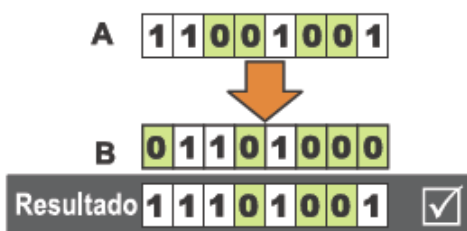
Cualquier otra combinación de voltajes de entrada proporcionará un cero lógico (0) en su salida. Utilizada en el programa, la operación Y lógico es realizada por una instrucción de programa, de la que vamos a hablar más tarde. Por ahora basta con conocer que Y lógico en un programa se refiere a la realización de este tipo de operación sobre los bits correspondientes de dos registros diferentes.

## COMPUERTA O (OR)

De manera similar, la compuerta O también dispone de dos o más entradas y de una salida. Si la compuerta dispone de sólo dos entradas, es aplicable lo siguiente: la salida proporciona un uno lógico (1) si una u otra entrada (A o B) es llevada a alto (1). En caso de que la compuerta O disponga de más de dos entradas, es aplicable lo siguiente: La salida proporciona un uno lógico (1) si por lo menos una entrada es llevada a alto (1). Si todas las entradas están a cero lógico (0), la salida estará a cero lógico (0) también.



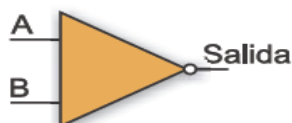
A	B	Salida
0	0	0
0	1	1
1	0	1
1	1	1



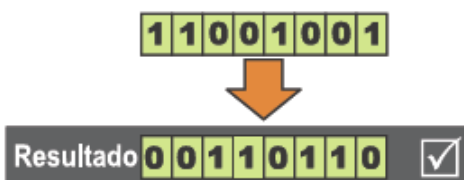
En un programa, la operación O lógico se realiza de la misma manera que la operación Y.

## COMPUERTA NO (NOT)

La compuerta lógica NO dispone de una sola entrada y una sola salida, por lo que funciona muy simplemente. Cuando un cero lógico (0) aparezca en su entrada, la salida proporciona un uno lógico (1) y viceversa. Esto significa que esta compuerta invierte las señales por sí mismas y por eso es denominada inversor.



A	Salida
0	1
1	0

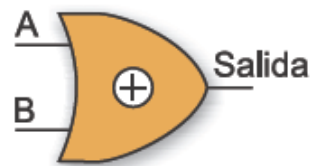


En el programa la operación lógica NO se realiza sobre un byte. El resultado es un byte con los bits invertidos. Si los bits de un byte se consideran número, el valor invertido es un complemento a ese número. El complemento de un número es el valor que se añade al número hasta llegar al mayor número binario de 8 dígitos. En otras palabras, la suma de un dígito de 8 números y de su complemento es siempre 255.

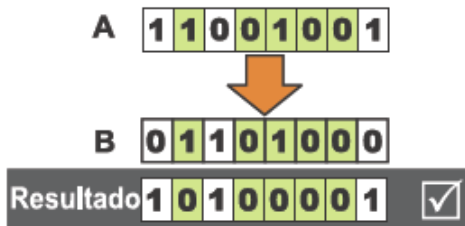
## COMPUERTA XOR (O EXCLUSIVA)

La compuerta XOR (O EXCLUSIVA) es un poco complicada en comparación con las demás.

Representa una combinación de todas las compuertas anteriormente descritas. La salida proporciona un uno lógico (1) sólo si sus entradas están en estados lógicos diferentes.



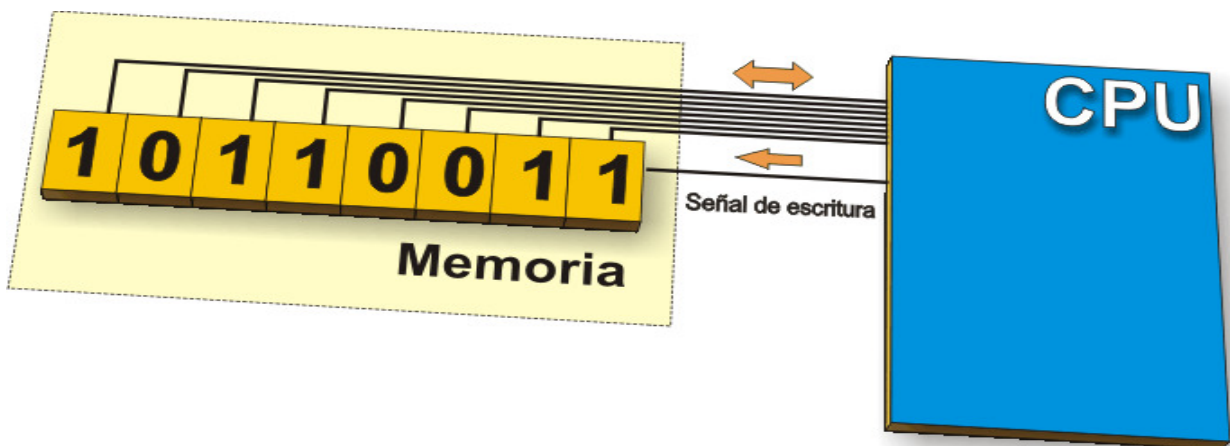
A	B	Salida
0	0	0
0	1	1
1	0	1
1	1	0



En el programa, esta operación se utiliza con frecuencia para comparar dos bytes. La resta se puede utilizar con el mismo propósito (si el resultado es 0, los bytes son iguales). A diferencia de la resta, la ventaja de esta operación lógica es que no es posible obtener los resultados negativos.

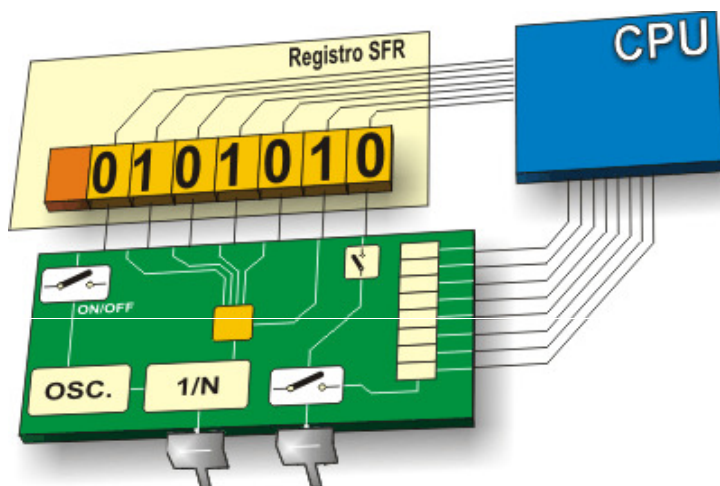
## REGISTROS

Un registro o una celda de memoria es un circuito electrónico que puede memorizar el estado de un byte.



## REGISTROS SFR

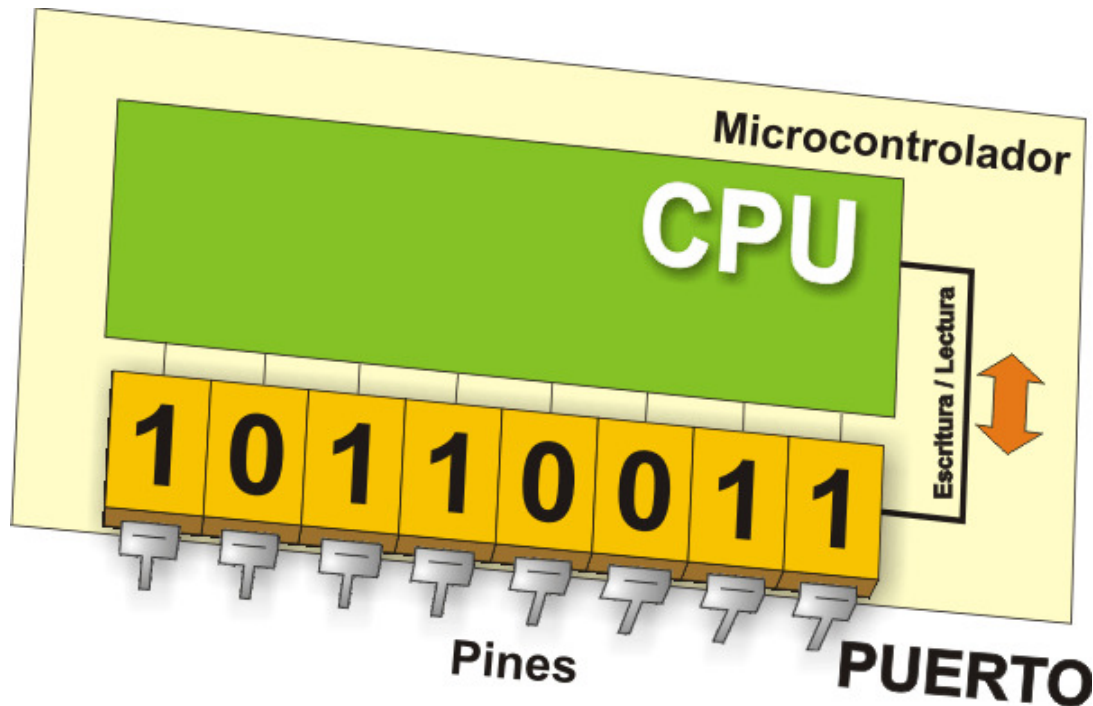
A diferencia de los registros que no tienen ninguna función especial y predeterminada, cada microcontrolador dispone de un número de registros de funciones especiales (SFR), con la función predeterminada por el fabricante. Sus bits están conectados a los circuitos internos del microcontrolador tales como temporizadores, convertidores A/D, osciladores entre otros, lo que significa que directamente manejan el funcionamiento de estos circuitos, o sea del microcontrolador. Imagínese ocho interruptores que manejan el funcionamiento de un circuito pequeño dentro del microcontrolador. Los registros SFR hacen exactamente lo mismo.



En otras palabras, el estado de los bits de registros se fija dentro de programa, los registros SFR dirigen los pequeños circuitos dentro del microcontrolador, estos circuitos se conectan por los pines del microcontrolador a un dispositivo periférico utilizado para... Bueno, depende de usted.

## PUERTOS DE ENTRADA/SALIDA (E/S)

Para hacer útil un microcontrolador, hay que conectarlo a un dispositivo externo, o sea, a un periférico. Cada microcontrolador tiene uno o más registros (denominados puertos) conectados a los pines en el microcontrolador. ¿Por qué se denominan como puertos de entrada/salida? Porque usted puede cambiar la función de cada pin como quiera. Por ejemplo, usted desea que su dispositivo encienda y apague los tres señales LEDs y que simultáneamente monitoree el estado lógico de 5 sensores o botones de presión. Uno de los puertos debe estar configurado de tal manera que haya tres salidas (conectadas a los LEDs) y cinco entradas (conectadas a los sensores). Eso se realiza simplemente por medio de software, lo que significa que la función de algún pin puede ser cambiada durante el funcionamiento.



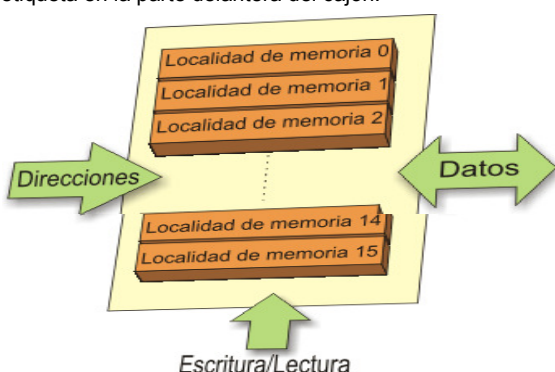
Una de las características más importantes de los pines de entrada/salida (E/S) es la corriente máxima que pueden entregar/recibir. En la mayoría de los microcontroladores la corriente obtenida de un pin es suficiente para activar un LED u otro dispositivo de baja corriente (10-20mA). Mientras más pines de E/S haya, más baja es la corriente máxima de un pin. En otras palabras, todos los puertos de E/S comparten la corriente máxima declarada en la hoja de especificación técnica del microprocesador.

Otra característica importante de los pines es que pueden disponer de los resistores pull-up. Estos resistores conectan los pines al polo positivo del voltaje de la fuente de alimentación y su efecto se puede ver al configurar el pin como una entrada conectada a un interruptor mecánico o a un botón de presión. Las últimas versiones de los microcontroladores tienen las resistencias pull-up configurables por software.

Cada puerto de E/S normalmente está bajo el control de un registro SFR especializado, lo que significa que cada bit de ese registro determina el estado del pin correspondiente en el microcontrolador. Por ejemplo, al escribir un uno lógico (1) a un bit del registro de control (SFR), el pin apropiado del puerto se configura automáticamente como entrada. Eso significa que el voltaje llevado a ese pin se puede leer como 0 o 1 lógico. En caso contrario, al escribir 0 al registro SFR, el pin apropiado del puerto se configura como salida. Su voltaje (0V o 5V) corresponde al estado del bit apropiado del registro del puerto.

## UNIDAD DE MEMORIA

La unidad de memoria es una parte del microcontrolador utilizada para almacenar los datos. La manera más fácil de explicarlo es compararlo con un armario grande con muchos cajones. Si marcamos los cajones claramente, será fácil acceder a cualquiera de sus contenidos al leer la etiqueta en la parte delantera del cajón.



De manera similar, cada dirección de memoria corresponde a una localidad de memoria. El contenido de cualquier localidad se puede leer y se le puede acceder al direccionarla. La memoria se puede escribir en la localidad o leer.

Hay varios tipos de memoria dentro del microcontrolador:

## MEMORIA ROM (READ ONLY MEMORY) - MEMORIA DE SÓLO LECTURA

La memoria ROM se utiliza para guardar permanentemente el programa que se está ejecutando. El tamaño de programa que se puede escribir depende del tamaño de esta memoria. Los microcontroladores actuales normalmente utilizan el direccionamiento de 16 bits, que significa que son capaces de direccionar hasta 64 Kb de memoria, o sea 65535 localidades. Por ejemplo, si usted es principiante, su programa excederá pocas veces el límite de varios cientos de instrucciones. Hay varios tipos de memoria ROM.

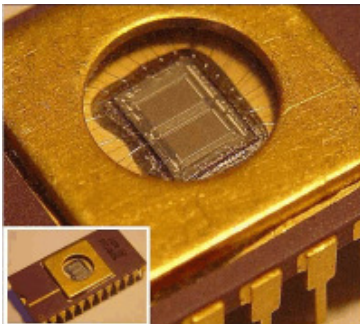
### Rom de máscara (enmascarada) - MROM

La ROM enmascarada es un tipo de ROM cuyo contenido es programado por el fabricante. El término “de máscara” viene del proceso de fabricación, donde las partes del chip se plasman en las máscaras utilizadas durante el proceso de fotolitografía. En caso de fabricación de grandes series, el precio es muy bajo. Olvide la idea de modificarla...

### OTP ROM (One Time Programmable ROM) - ROM programable una sola vez

La memoria programable una sola vez permite descargar un programa en el chip, pero como dice su nombre, una sola vez. Si se detecta un error después de descargarlo, lo único que se puede hacer es descargar el programa correcto en otro chip.

### UV EPROM (UV Erasable Programmable ROM) - ROM programable borrrable por rayos ultravioleta



El encapsulado de este microcontrolador tiene una “ventana” reconocible en la parte alta. Eso permite exponer la superficie del chip de silicio a la luz de ultravioleta y borrar el programa completamente en varios minutos. Luego es posible descargar un nuevo programa en él.

La instalación de esta ventana es complicada, lo que por supuesto afecta al precio. Desde nuestro punto de vista, desgraciadamente, de manera negativa...

### Memoria Flash

Este tipo de memoria se inventó en los años 80 en los laboratorios de la compañía INTEL, como forma desarrollada de la memoria UV EPROM. Ya que es posible escribir y borrar el contenido de esta memoria prácticamente un número ilimitado de veces, los microcontroladores con memoria Flash son perfectos para estudiar, experimentar y para la fabricación en pequeña escala. Por la gran popularidad de esta memoria, la mayoría de los microcontroladores se fabrican con tecnología *flash* hoy en día. Si usted va a comprar un microcontrolador, ¡éste es en definitiva la opción perfecta!

## MEMORIA RAM (RANDOM ACCESS MEMORY) - MEMORIA DE ACCESO ALEATORIO

Al apagar la fuente de alimentación, se pierde el contenido de la memoria RAM. Se utiliza para almacenar temporalmente los datos y los resultados inmediatos creados y utilizados durante el funcionamiento del microcontrolador. Por ejemplo, si el programa ejecuta la adición (de cualquier cosa) es necesario tener un registro que representa lo que se llama “suma” en vida cotidiana. Con tal propósito, uno de los registros de la RAM es denominado “suma” y se utiliza para almacenar los resultados de la adición.

## MEMORIA EEPROM (ELECTRICALLY ERASABLE PROGRAMMABLE ROM) - ROM PROGRAMABLE Y BORRABLE ELÉCTRICAMENTE

El contenido de la EEPROM se puede cambiar durante el funcionamiento (similar a la RAM), pero se queda permanentemente guardado después de la pérdida de la fuente de alimentación (similar a la ROM). Por lo tanto, la EEPROM se utiliza con frecuencia para almacenar los valores creados durante el funcionamiento, que tienen que estar permanentemente guardados. Por ejemplo, si usted ha diseñado una llave electrónica o un alarma, sería estupendo permitir al usuario crear e introducir una contraseña por su cuenta. Por supuesto, la nueva contraseña tiene que estar guardada al apagar la fuente de alimentación. En tal caso una solución perfecta es el microcontrolador con una EEPROM embebida.

## INTERRUPCIÓN

La mayoría de programas utilizan interrupciones durante ejecución de programa regular. El propósito del microcontrolador generalmente consiste en reaccionar a los cambios en su entorno. En otras palabras, cuando ocurre algo, el microcontrolador reacciona de alguna manera... Por ejemplo, al apretar el botón del mando a distancia, el microcontrolador lo registra y responde al comando cambiando de canal, subiendo o bajando el volumen etc. Si el microcontrolador pasará la mayoría del tiempo comprobando varios botones sin parar - las horas, los

días, esto no sería nada práctico.

Por lo tanto, el microcontrolador “aprendió un truco” durante su evolución. En vez de seguir comprobando algún pin o bit, el microcontrolador deja su “trabajo de esperar” a un “experto” que reaccionará sólo en caso de que suceda algo digno de atención.

La señal que informa al procesador central acerca de tal acontecimiento se denomina **INTERRUPCIÓN**.

## UNIDAD CENTRAL DE PROCESAMIENTO (CENTRAL PROCESSOR UNIT - CPU)

Como indica su nombre, esto es una unidad que controla todos los procesos dentro del microcontrolador. Consiste en varias unidades más pequeñas, de las que las más importantes son:

- **Decodificador de instrucciones** es la parte que descodifica las instrucciones del programa y acciona otros circuitos basándose en esto. El “conjunto de instrucciones” que es diferente para cada familia de microcontrolador expresa las capacidades de este circuito;
- **Unidad lógica aritmética (Arithmetical Logical Unit - ALU)** realiza todas las operaciones matemáticas y lógicas sobre datos; y
- **Acumulador** o registro de trabajo. Es un registro SFR estrechamente relacionado con el funcionamiento de la ALU. Es utilizado para almacenar todos los datos sobre los que se debe realizar alguna operación (sumar, mover). También almacena los resultados preparados para el procesamiento futuro. Uno de los registros SFR, denominado Registro Status (PSW), está estrechamente relacionado con el acumulador. Muestra el “estado” de un número almacenado en el acumulador (el número es mayor o menor que cero etc.) en cualquier instante dado.



## BUS

El bus está formado por 8, 16 o más cables. Hay dos tipos de buses: el bus de direcciones y el bus de datos. El bus de direcciones consiste en tantas líneas como sean necesarias para direccionar la memoria. Se utiliza para transmitir la dirección de la CPU a la memoria. El bus de datos es tan ancho como los datos, en este caso es de 8 bits o líneas de ancho. Se utiliza para conectar todos los circuitos dentro del microcontrolador.

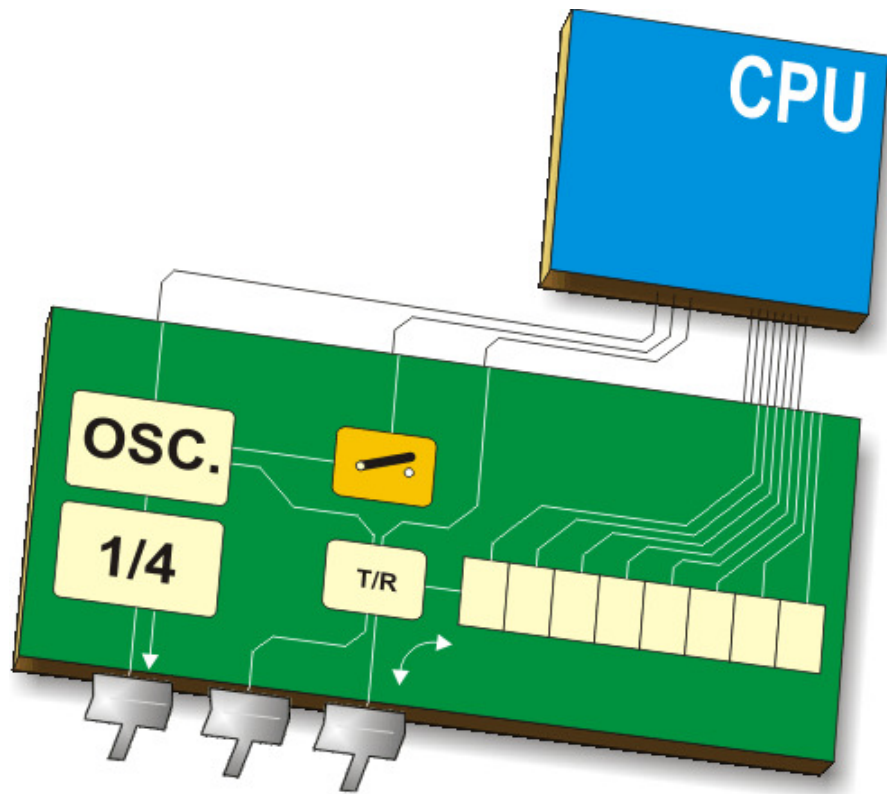
## COMUNICACIÓN EN SERIE

La conexión paralela entre el microcontrolador y los periféricos a través de los puertos de entrada/salida es una solución perfecta para las distancias cortas - hasta varios metros. No obstante, en otros casos cuando es necesario establecer comunicación entre dos dispositivos a largas distancias no es posible utilizar la conexión paralela. En vez de eso, se utiliza la conexión en serie.

Hoy en día, la mayoría de los microcontroladores llevan incorporados varios sistemas diferentes para la comunicación en serie, como un equipo estándar. Cuál de estos sistemas se utilizará en un caso concreto, depende de muchos factores, de los que más importantes son:

- ¿Con cuántos dispositivos el microcontrolador tiene que intercambiar los datos?
- ¿Cuál es la velocidad del intercambio de datos obligatoria?
- ¿Cuál es la distancia entre los dispositivos?
- ¿Es necesario transmitir y recibir los datos simultáneamente?





Una de las cosas más importantes en cuanto a la comunicación en serie es el Protocolo que debe ser estrictamente observado. Es un conjunto de reglas que se aplican obligatoriamente para que los dispositivos puedan interpretar correctamente los datos que intercambian mutuamente. Afortunadamente, los microcontroladores se encargan de eso automáticamente, así que el trabajo de programador/usuario es reducido a la escritura y lectura de datos.

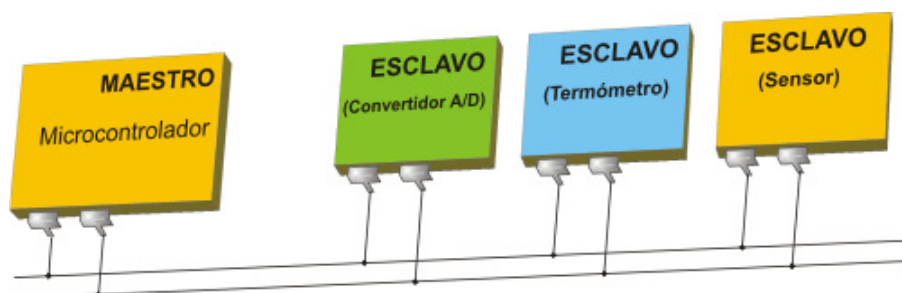
### VELOCIDAD DE TRANSMISIÓN SERIAL

La velocidad de transmisión serial (*baud rate*) es el término utilizado para denotar el número de bits transmitidos por segundo [bps]. ¡Fíjese que este término se refiere a bits, y no a bytes! El protocolo normalmente requiere que cada byte se transmita junto con varios bits de control. Eso quiere decir que un byte en un flujo de datos serial puede consistir en 11 bits. Por ejemplo, si velocidad de transmisión serial es 300 bps un máximo de 37 y un mínimo de 27 bytes se pueden transmitir por segundo.

Los sistemas de comunicación serial más utilizados son:

### I<sup>2</sup>C (INTER INTEGRATED CIRCUIT) - CIRCUITO INTER-INTEGRADO

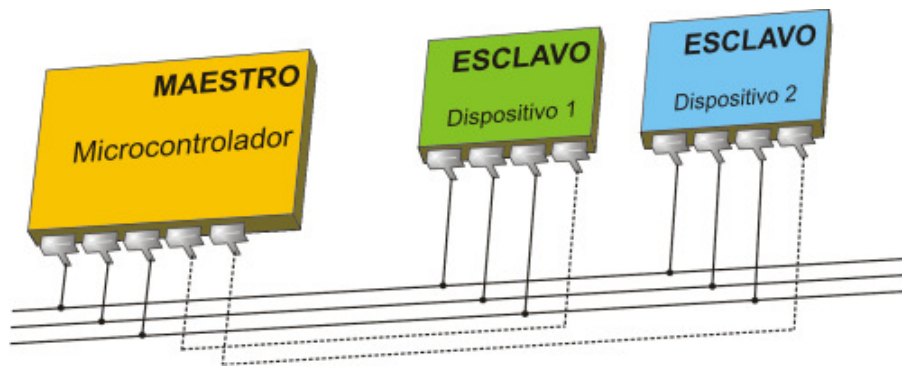
Circuito inter-integrado es un sistema para el intercambio de datos serial entre los microcontroladores y los circuitos integrados especializados de generación. Se utiliza cuando la distancia entre ellos es corta (el receptor y el transmisor están normalmente en la misma placa de circuito impreso). La conexión se establece por medio de dos líneas - una se utiliza para transmitir los datos, mientras que la otra se utiliza para la sincronización (la señal de reloj). Como se muestra en la figura, un dispositivo es siempre el principal (master - maestro), el que realiza el direccionamiento de un chip subordinado (slave - esclavo) antes de que se inicie la comunicación. De esta manera un microcontrolador puede comunicarse con 112 dispositivos diferentes. La velocidad de transmisión serial es normalmente 100 Kb/seg (el modo estándar) o 10 Kb/seg (modo de velocidad de transmisión baja). Recientemente han aparecido los sistemas con la velocidad de transmisión serial 3.4 Mb/sec. La distancia entre los dispositivos que se comunican por el bus I<sup>2</sup>C está limitada a unos metros.



### SPI (SERIAL PERIPHERAL INTERFACE BUS) - BUS SERIAL DE INTERFAZ DE PERIFÉRICOS

Un bus serial de interfaz de periféricos es un sistema para la comunicación serial que utiliza hasta cuatro líneas (normalmente solo son necesarias tres) - para recibir los datos, para transmitir los datos, para sincronizar y (opcional) para seleccionar el dispositivo con el que se comunica. Esto es la conexión full duplex, lo que significa que los datos se envían y se reciben simultáneamente.

La velocidad de transmisión máxima es mayor que en el sistema de conexión I2C.

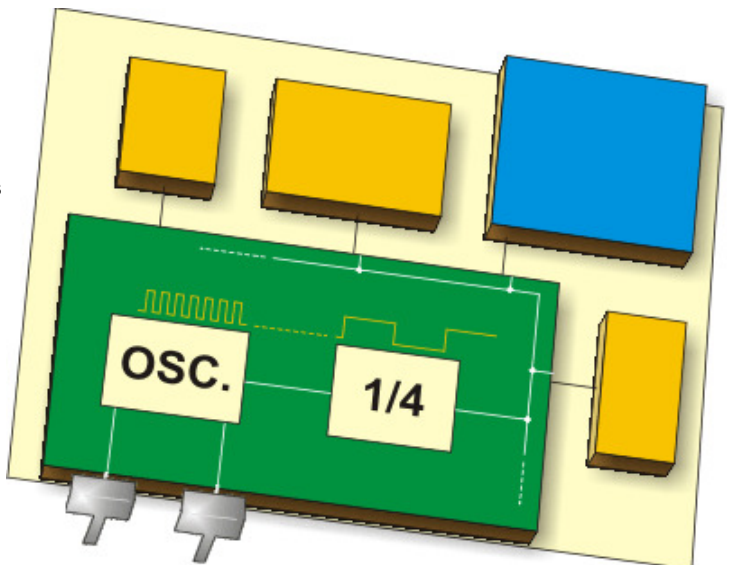


### UART (UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER) - TRANSMISOR-RECEPTOR ASÍNCRONO UNIVERSAL

Este tipo de conexión es asíncrona, lo que significa que no se utiliza una línea especial para transmitir la señal de reloj. En algunas aplicaciones este rasgo es crucial (por ejemplo, en mandar datos a distancia por RF o por luz infrarroja). Puesto que se utiliza sólo una línea de comunicación, tanto el receptor como el transmisor reciben y envían los datos a velocidad misma que ha sido predefinida para mantener la sincronización necesaria. Esto es una manera simple de transmitir datos puesto que básicamente representa una conversión de datos de 8 bits de paralelo a serial. La velocidad de transmisión no es alta, es hasta 1 Mbit/sec.

### OSCILADOR

Los pulsos uniformes generados por el oscilador permiten el funcionamiento armónico y síncrono de todos los circuitos del microcontrolador. El oscilador se configura normalmente de tal manera que utilice un cristal de cuarzo o resonador cerámico para estabilización de frecuencia. Además, puede funcionar como un circuito autónomo (como oscilador RC). Es importante decir que las instrucciones del programa no se ejecutan a la velocidad impuesta por el mismo oscilador sino varias veces más despacio. Eso ocurre porque cada instrucción se ejecuta en varios ciclos del oscilador. En algunos microcontroladores se necesita el mismo número de ciclos para ejecutar todas las instrucciones, mientras que en otros el tiempo de ejecución no es el mismo para todas las instrucciones. Por consiguiente, si el sistema utiliza el cristal de cuarzo con una frecuencia de 20 MHz, el tiempo de ejecución de una instrucción de programa no es 50 nS, sino 200, 400 o 800 nS dependiendo del tipo del microcontrolador.



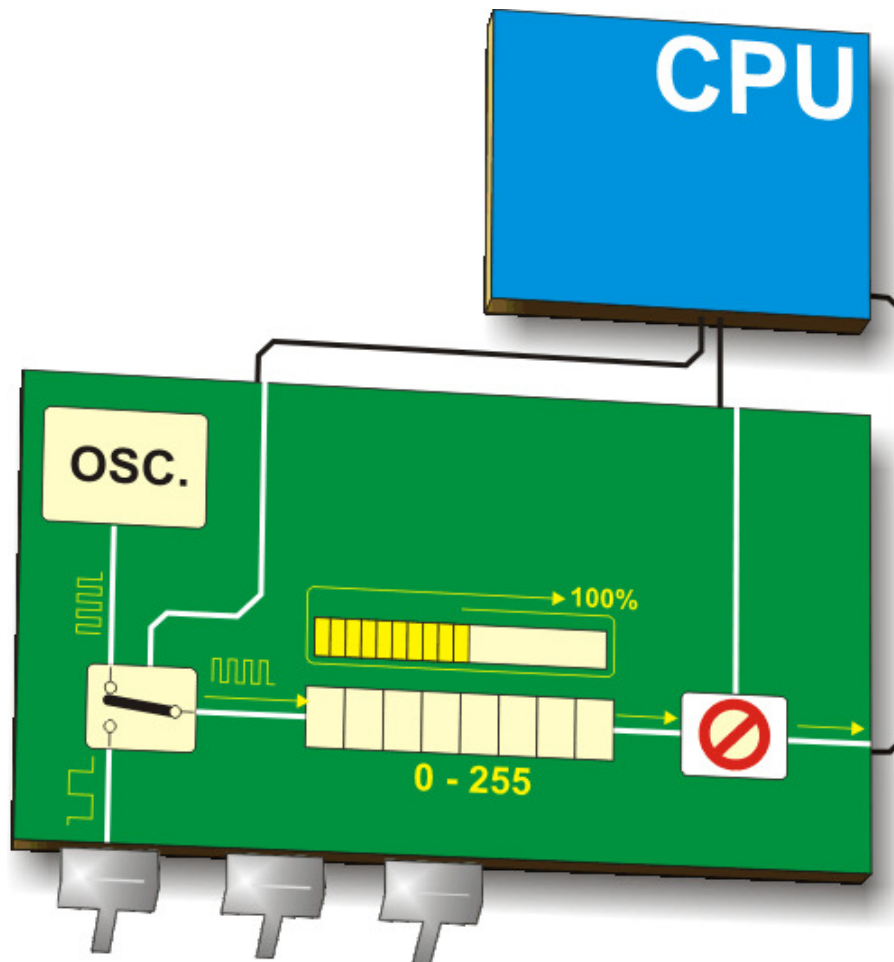
### CIRCUITO DE ALIMENTACIÓN

Hay que mencionar dos cosas dignas de atención con relación al circuito de la fuente de alimentación de microcontroladores:

- **Brown out** es un estado potencialmente peligroso que ocurre al apagar el microcontrolador o en caso de que el voltaje de la fuente de alimentación salga de unos márgenes debido al ruido eléctrico. Como el microcontrolador dispone de varios circuitos que funcionan a niveles de voltaje diferentes, ese estado puede causar un comportamiento descontrolado. Para evitarlo, el microcontrolador normalmente tiene un circuito incorporado para el brown out reset. El circuito reinicia inmediatamente el microcontrolador si el voltaje de alimentación cae por debajo del límite.
- **El pin de reset** (reinicio), marcado frecuentemente con MCLR (Master Clear Reset), sirve para el reinicio externo del microcontrolador al aplicar un cero (0) o un uno (1) lógico dependiendo del tipo del microcontrolador. En caso de que el circuito brown out no esté incorporado, un simple circuito externo para el brown out reset se puede conectar al pin MCLR.

### TEMPORIZADORES/CONTADORES

El oscilador del microcontrolador utiliza cristal de cuarzo para su funcionamiento. Aunque no se trata de la solución más simple, hay muchas razones para utilizarlo. La frecuencia del oscilador es definida con precisión y muy estable, así que siempre genera los pulsos del mismo ancho, lo que los hace perfectos para medición de tiempo. Tales osciladores se utilizan en los relojes de cuarzo. Si es necesario medir el tiempo transcurrido entre dos eventos, basta con contar los pulsos generados por este oscilador. Esto es exactamente lo que hace el temporizador.



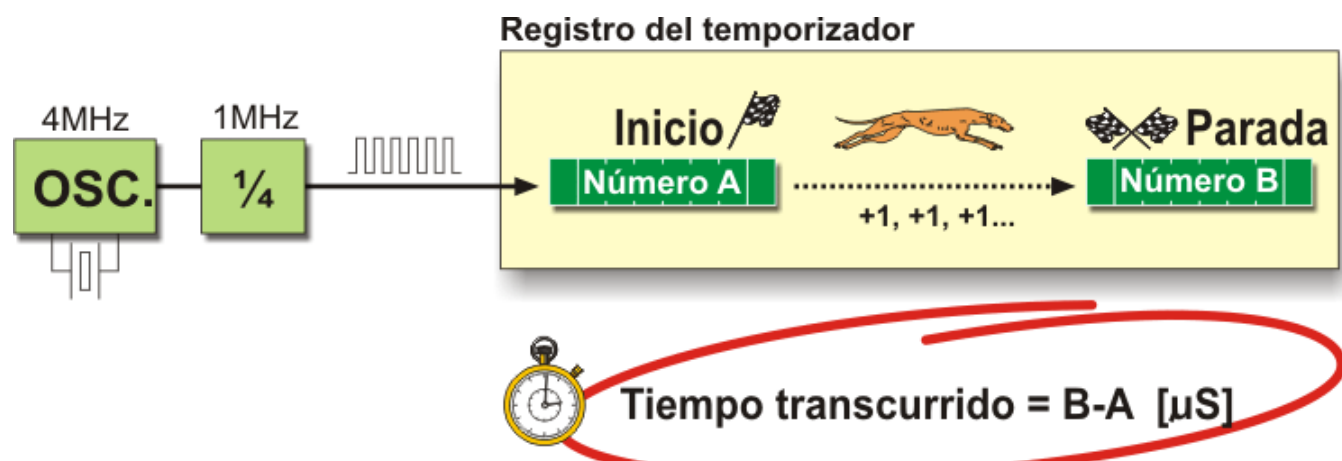
La mayoría de los programas utiliza estos cronómetros electrónicos en miniatura. Generalmente son registros SFR de 8 o 16 bits cuyo contenido se aumenta automáticamente con cada pulso. ¡Una vez que se llena el registro, se genera una interrupción!

Si el temporizador utiliza el oscilador de cuarzo interno para su funcionamiento, es posible medir el tiempo entre dos eventos (el valor de registro en el momento de iniciar la medición es T1, en el momento de finalizar la medición es T2, el tiempo transcurrido es igual al resultado de la resta  $T2 - T1$ ). Si los registros se aumentan con los pulsos que vienen de la fuente externa, tal temporizador se convierte en un contador.

Esto es una explicación simple de su funcionamiento. Es un poco más complicado en práctica.

### ¿CÓMO FUNCIONAN LOS TEMPORIZADORES?

En práctica, los pulsos generados por el oscilador de cuarzo son llevados al circuito una vez por cada ciclo de máquina directamente o por el pre-escalador, lo que aumenta el número en el registro del temporizador. Si una instrucción (un ciclo de máquina) dura cuatro períodos del oscilador de cuarzo, este número será cambiado un millón de veces por segundo (cada microsegundo) al incorporar al cuarzo que oscila con una frecuencia de 4 MHz.

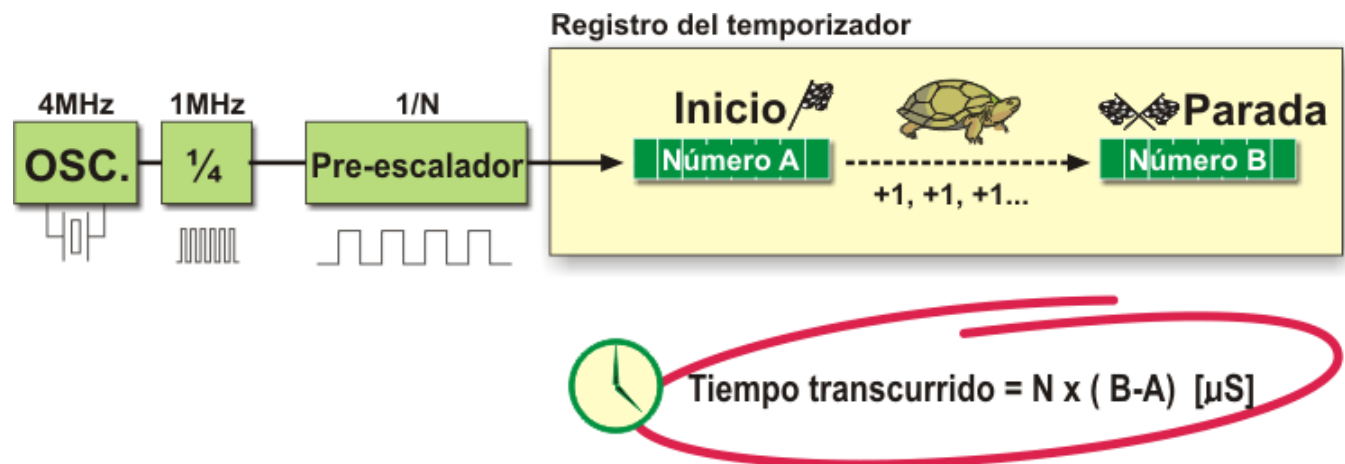


Es fácil de medir los intervalos de tiempo cortos de la manera descrita anteriormente (hasta 256 microsegundos porque es el mayor número que un registro puede contener). Esta obvia desventaja se puede superar de varias maneras: al utilizar el oscilador más lento, por medio de

registros con más bits, del pre-escalador o de la interrupción. Las primeras dos soluciones tienen algunas debilidades así que se recomienda utilizar el pre-escalador y/o la interrupción.

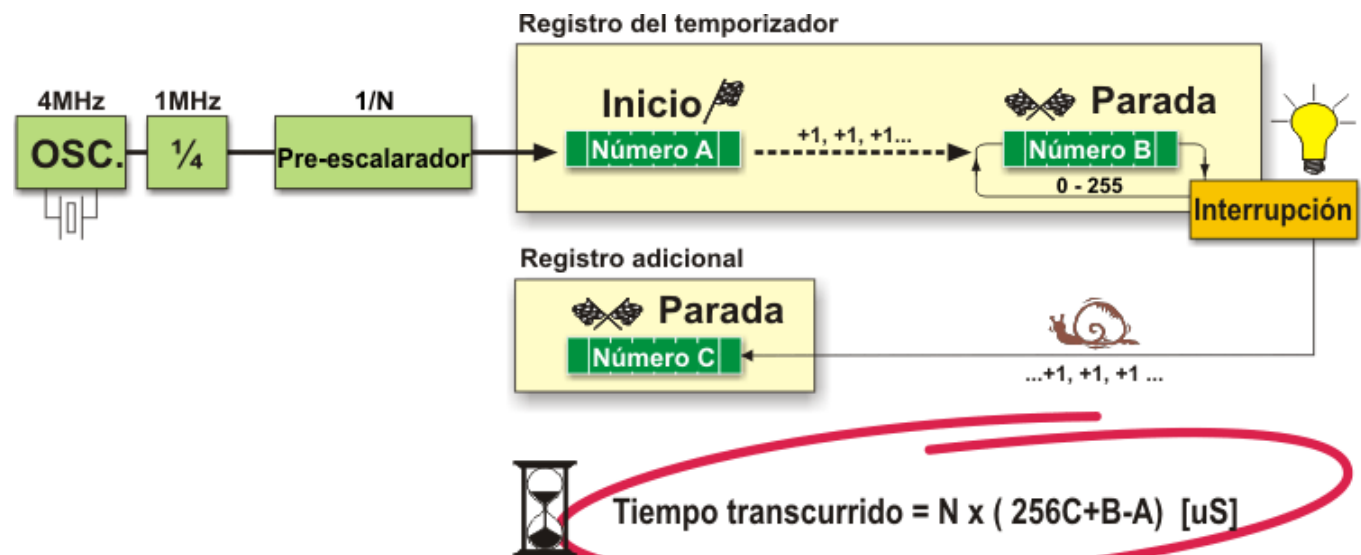
### UTILIZAR UN PREESCALADOR EN EL FUNCIONAMIENTO DEL TEMPORIZADOR

Un pre-escalador es un dispositivo electrónico utilizado para dividir la frecuencia por un factor predeterminado. Esto quiere decir que se necesita llevar 1, 2, 4 o más pulsos a su entrada para generar un pulso a la salida. La mayoría de los microcontroladores disponen de uno o más pre-escaladores incorporados y su tasa de división puede ser cambiada dentro del programa. El pre-escalador se utiliza cuando es necesario medir los períodos de tiempo más largos. Si el temporizador y el temporizador perro guardián comparten un pre-escalador, éste no se puede utilizar por los dos simultáneamente.



### UTILIZAR UNA INTERRUPCIÓN EN EL FUNCIONAMIENTO DEL TEMPORIZADOR

Si el registro del temporizador es de 8 bits, el mayor número que se puede escribir en él es 255 (en los registros de 16 bits es el número 65.535). Si se excede este número, el temporizador se reinicia automáticamente y el conteo comienza de nuevo en cero. Esto es denominado desbordamiento o sobreflujo (*overflow*). Permitido por el programa, el desbordamiento puede provocar una interrupción, lo que abre completamente nuevas posibilidades. Por ejemplo, el estado de registros utilizados para contar segundos, minutos o días puede ser implementado en una rutina de interrupción. El proceso entero (excepto la rutina de interrupción) se lleva a cabo internamente, lo que permite que los circuitos principales del microcontrolador funcionen regularmente.



La figura anterior describe el uso de una interrupción en el funcionamiento del temporizador. Al asignarle un pre-escalador al temporizador, se producen retrasos de duración arbitraria con mínima interferencia en la ejecución del programa principal.

### CONTADORES

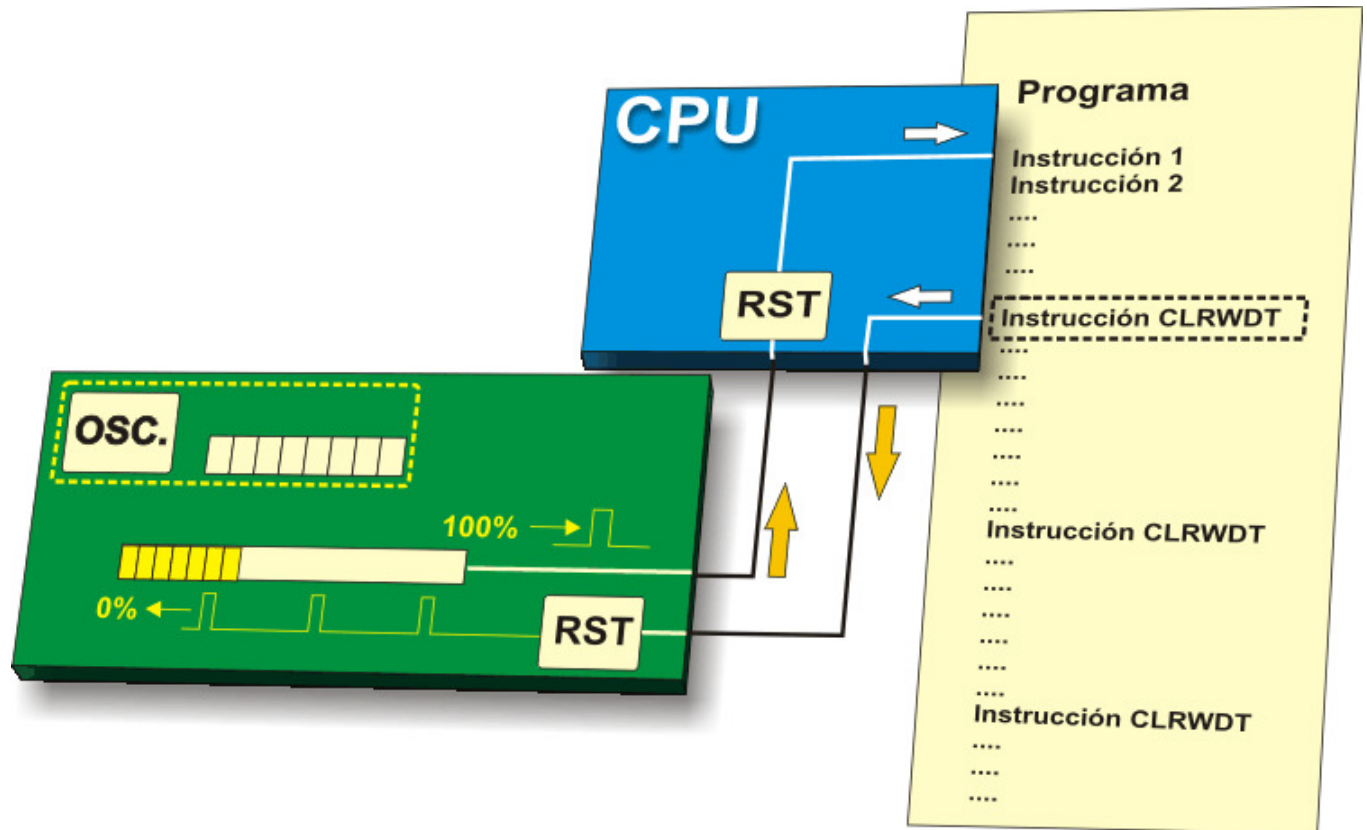
Si un temporizador se suministra por los pulsos ingresados por el pin de entrada en el microcontrolador, se produce un contador. Evidentemente, es el mismo circuito electrónico. La única diferencia es que los pulsos para contar se ingresan por el pin de entrada y que su duración (anchura) no es definida. Por eso, no se pueden utilizar para medición de tiempo, sino que se utilizan para otros propósitos, por ejemplo: contar los productos en la cadena de montaje, número de rotaciones del eje de un motor, pasajeros etc. (dependiendo del sensor utilizado).

## TEMPORIZADOR PERRO GUARDIÁN (WATCHDOG)

El perro guardián es un temporizador conectado a un oscilador RC completamente independiente dentro del microcontrolador.

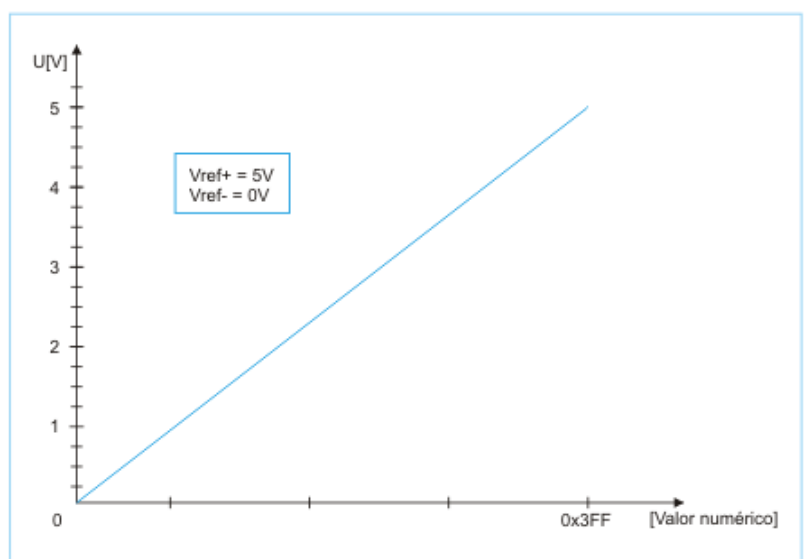
Si el perro guardián está habilitado, cada vez que cuenta hasta el máximo valor en el que ocurre el desbordamiento del registro se genera una señal de reinicio del microcontrolador y la ejecución de programa inicia en la primera instrucción. El punto es evitar que eso ocurra al utilizar el comando adecuado.

La idea se basa en el hecho de que cada programa se ejecuta en varios bucles, más largos o cortos. Si las instrucciones que reinician el temporizador perro guardián se colocan en lugares estratégicos del programa, aparte los comandos que se ejecutan regularmente, el funcionamiento del perro guardián no afectará a la ejecución del programa. Si por cualquier razón (ruidos eléctricos frecuentes en la industria) el contador de programa “se queda atrapado” dentro de un bucle infinito, el valor del registro continuará aumentado por el temporizador perro guardián alcanzará el máximo valor, el registro se desbordará y, ¡aleluya! ¡Ocurre el reinicio!



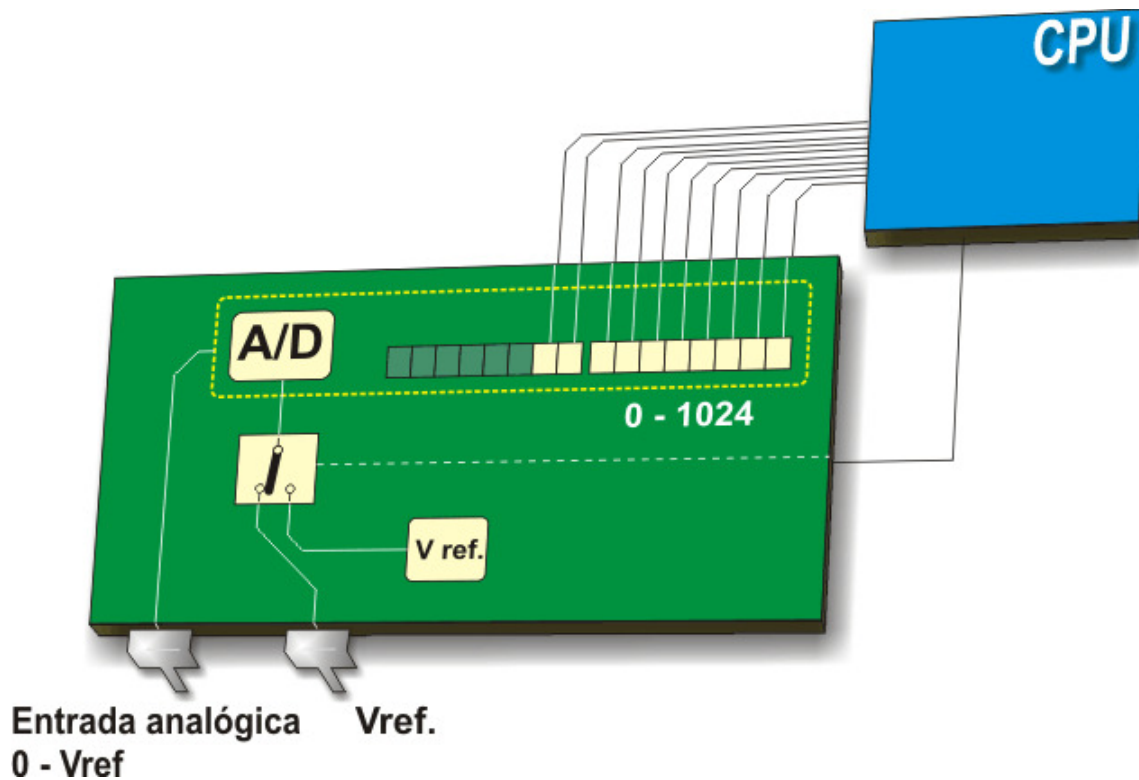
## CONVERTIDOR A/D

Las señales del mundo real son muy diferentes de las que “entiende” el microcontrolador (ceros y unos), así que deben ser convertidas para que el microcontrolador pueda entenderlas. Un convertidor analógico-digital es un circuito electrónico encargado de convertir las señales continuas en números digitales discretos. En otras palabras, este circuito convierte un número real en un número binario y se lo envía a la CPU para ser procesado. Este módulo se utiliza para medir el voltaje en el pin de entrada.



El resultado de esta medición es un número (el valor digital) utilizado y procesado más tarde en el programa.





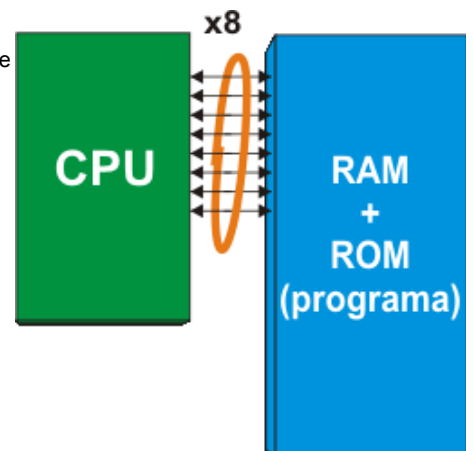
## ARQUITECTURA INTERNA

Todos los microcontroladores actuales utilizan uno de dos modelos básicos de arquitectura denominados *Harvard* y *von-Neumann*.

Son dos maneras diferentes del intercambio de datos entre la CPU y la memoria.

### Arquitectura de von-Neumann

Los microcontroladores que utilizan la arquitectura von-Neumann disponen de un solo bloque de memoria y de un bus de datos de 8 bits. Como todos los datos se intercambian por medio de estas 8 líneas, este bus está sobrecargado, y la comunicación por si misma es muy lenta e ineficaz. La CPU puede leer una instrucción o leer/escribir datos de/en la memoria. Los dos procesos no pueden ocurrir a la vez puesto que las instrucciones y los datos utilizan el mismo bus. Por ejemplo, si alguna línea de programa dice que el registro de la memoria RAM llamado "SUM" debe ser aumentado por uno (instrucción: `incf SUMA`), el microcontrolador hará lo siguiente:

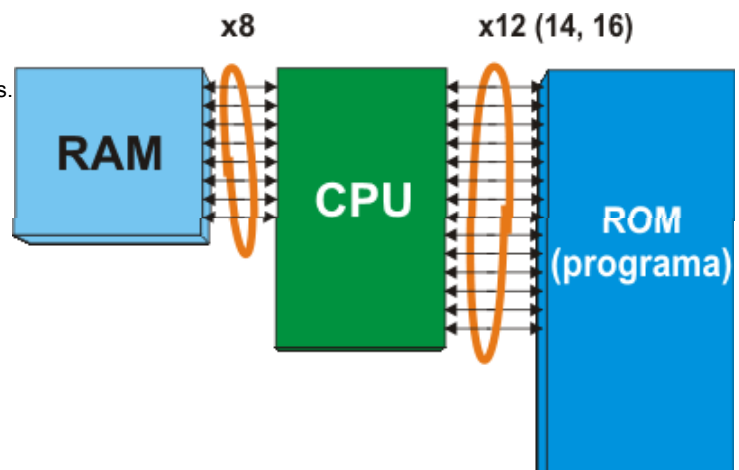


1. Leer la parte de la instrucción de programa que especifica QUÉ es lo que debe realizar (en este caso es la instrucción para incrementar "`incf`")
2. Seguir leyendo la misma instrucción que especifica sobre CUÁL dato lo debe realizar (en este caso es el contenido del registro "SUMA")
3. Después de haber sido incrementado, el contenido de este registro se debe escribir en el registro del que fue leído (dirección del registro "SUMA")

El mismo bus de datos se utiliza para todas estas operaciones intermedias.

### ARQUITECTURA DE HARVARD

Los microcontroladores que utilizan esta arquitectura disponen de dos buses de datos diferentes. Uno es de 8 bits de ancho y conecta la CPU con la memoria RAM. El otro consiste en varias líneas (12, 14 o 16) y conecta a la CPU y la memoria ROM. Por consiguiente, la CPU puede leer las instrucciones y realizar el acceso a la memoria de datos a la vez. Puesto que todos los



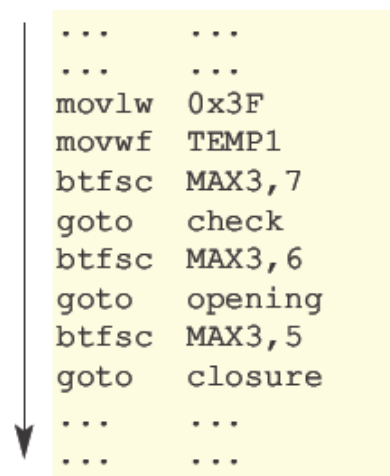
registros de la memoria RAM son de 8 bits de ancho, todos los datos dentro del microcontrolador que se intercambian son de la misma anchura. Durante el proceso de la escritura de programa, sólo se manejan los datos de 8 bits. En otras palabras, todo lo que usted podrá cambiar en el programa y a lo que podrá afectar será de 8 bits de ancho. Todos los programas escritos para estos microcontroladores serán almacenados en la memoria ROM interna del microcontrolador después de haber sido compilados a código máquina. No obstante, estas localidades de memoria ROM no tienen 8, sino 12, 14 o 16 bits. 4, 6 o 8 bits adicionales representan una instrucción que especifica a la CPU qué hacer con los datos de 8 bits.

Las ventajas de este diseño son las siguientes:

- Todos los datos en el programa son de un byte (8 bits) de ancho. Como un bus de datos utilizado para lectura de programa tiene unas líneas más (12, 14 o 16), tanto la instrucción como el dato se pueden leer simultáneamente al utilizar estos bits adicionales. Por eso, todas las instrucciones se ejecutan en un ciclo salvo las instrucciones de salto que son de dos ciclos.
- El hecho de que un programa (la ROM) y los datos temporales (la RAM) estén separados, permite a la CPU poder ejecutar dos instrucciones simultáneamente. Dicho de manera sencilla, mientras que se realiza la lectura o escritura de la RAM (que marca el fin de una instrucción), la siguiente instrucción se lee por medio de otro bus.
- En los microcontroladores que utilizan la arquitectura de *von-Neumann*, nunca se sabe cuánta memoria ocupará algún programa. Generalmente, la mayoría de las instrucciones de programa ocupan dos localidades de memoria (una contiene información sobre QUÉ se debe realizar, mientras que la otra contiene información sobre CUÁL dato se debe realizar). Sin embargo, esto no es una fórmula rígida, sino el caso más frecuente. En los microcontroladores que utilizan una arquitectura Harvard, el bus de la palabra de programa es más ancho que un byte, lo que permite que cada palabra de programa esté compuesto por una instrucción y un dato. En otras palabras, una localidad de memoria - una instrucción de programa.

## JUEGO DE INSTRUCCIONES

El nombre colectivo de todas las instrucciones que puede entender el microcontrolador es llamado *Juego de Instrucciones*. Cuando se escribe un programa en ensamblador, en realidad se especifican instrucciones en el orden en el que deben ser ejecutadas. La restricción principal es el número de instrucciones disponibles. Los fabricantes aceptan cualquiera de los dos enfoques descritos a continuación:



```
...    ...  
...    ...  
movlw  0x3F  
movwf  TEMP1  
btfsc  MAX3,7  
goto   check  
btfsc  MAX3,6  
goto   opening  
btfsc  MAX3,5  
goto   closure  
...    ...  
...    ...
```

### RISC (Reduced Instruction Set Computer) - Computadora con Juego de Instrucciones Reducidas

En este caso la idea es que el microcontrolador reconoce y ejecuta sólo operaciones básicas (sumar, restar, copiar etc...) Las operaciones más complicadas se realizan al combinar éstas (por ejemplo, multiplicación se lleva a cabo al realizar adición sucesiva). Es como intentar explicarle a alguien con pocas palabras cómo llegar al aeropuerto en una nueva ciudad. Sin embargo, no todo es tan oscuro. Además, el microcontrolador es muy rápido así que no es posible ver todas las "acrobacias" aritméticas que realiza. El usuario sólo puede ver el resultado final de todas las operaciones. Por último, no es tan difícil explicar dónde está el aeropuerto si se utilizan las palabras adecuadas tales como: a la derecha, a la izquierda, el kilómetro etc.

### CISC (Complex Instruction Set Computer) - Computadoras con un juego de instrucciones complejo

¡CISC es opuesto a RISC! Los microcontroladores diseñados para reconocer más de 200 instrucciones diferentes realmente pueden realizar muchas cosas a alta velocidad. No obstante, uno debe saber cómo utilizar todas las posibilidades que ofrece un lenguaje tan rico, lo que no es siempre tan fácil...

## ¿CÓMO ELEGIR UN MICROCONTROLADOR?

Bueno, si usted es principiante, y ha tomado decisión de trabajar con los microcontroladores. ¡Felicitaciones por la elección! No obstante, a

primera vista, no es fácil la elección del microcontrolador más adecuado como parece a la primera vista. ¡El problema no es el pequeño rango de dispositivos a elegir, sino todo lo contrario!

Antes de empezar a diseñar un dispositivo basado en un microcontrolador, tome en cuenta lo siguiente: cuántas entradas/líneas son necesarias para su funcionamiento, realizaría el dispositivo otras operaciones además encender/apagar un relé, necesita algún modulo especializado tal como el de comunicación en serie, convertidor A/D etc. Cuando usted tiene una clara imagen de lo que quiere, el rango de selección se reduce considerablemente, y le queda pensar en el precio. ¿Va a tener varios dispositivos? ¿Varios cientos? ¿Un millón? De todos modos ahora es más claro.

Si está pensando en todas estas cosas por primera vez, todo le parecerá un poco confuso. Por esa razón, vaya paso a paso. Antes que nada, seleccione al fabricante, es decir, la familia de microcontroladores que ofrece. Luego, aprenda a trabajar con un modelo particular. Sólo aprenda lo que necesite aprender, no entre demasiado en detalles. Resuelva el problema específico y le pasará una cosa increíble - será capaz de manejar cualquier modelo del mismo fabricante...

Más o menos, todo se parece a montar en bicicleta: después de varias caídas inevitables en el principio, será capaz de mantener el equilibrio y montar en cualquier otra bicicleta. ¡Por supuesto, nunca se olvida tanto de montar en bicicleta, como de la destreza de programación!

## 1.4 MICROCONTROLADORES PIC

Los microcontroladores PIC desarrollados por *Microchip Technology* son probablemente la mejor opción si es principiante. Hay varias razones por lo que esto es verdadero...

El nombre verdadero de este microcontrolador es PICmicro (*Peripheral Interface Controller*), conocido bajo el nombre PIC. Su primer antecesor fue creado en 1975 por la compañía *General Instruments*. Este chip denominado PIC1650 fue diseñado para propósitos completamente diferentes. Diez años más tarde, al añadir una memoria EEPROM, este circuito se convirtió en un verdadero microcontrolador PIC. Hace unos pocos años la compañía *Microchip Technology* fabricó la 5 billonésima muestra. Si está interesado en aprender más sobre eso, siga leyendo.

Familia	ROM [Kbytes]	RAM [bytes]	Pines	Frecuencia de reloj. [MHz]	Entradas A/D	Resolución del convertidor A/D	Comparadores	Temporizadores de 8/16 bits	Comunicación
Arquitectura de la gama baja de 8 bits, palabra de instrucción de 12 bits									
PIC10FXXX	0.375 - 0.75	16 - 24	6 - 8	4 - 8	0 - 2	8	0 - 1	1 x 8	
PIC12FXXX	0.75 - 1.5	25 - 38	8	4 - 8	0 - 3	8	0 - 1	1 x 8	
PIC16FXXX	0.75 - 3	25 - 134	14 - 44	20	0 - 3	8	0 - 2	1 x 8	
PIC16HVXXX	1.5	25	18 - 20	20	-	-	-	1 x 8	
Arquitectura de la gama media de 8 bits, palabra de instrucción de 14 bits									
PIC12FXXX	1.75 - 3.5	64 - 128	8	20	0 - 4	10	1	1 - 2 x 8 1 x 16	
PIC12HVXXX	1.75	64	8	20	0 - 4	10	1	1 - 2 x 8 1 x 16	
PIC16FXXX	1.75 - 14	64 - 368	14 - 64	20	0 - 13	8 or 10	0 - 2	1 - 2 x 8 1 x 16	USART
PIC16HVXXX	1.75 - 3.5	64 - 128	14 - 20	20	0 - 12	10	2	2 x 8 1 x 16	USART I2C SPI
Arquitectura de la gama alta de 8 bits, palabra de instrucción de 16 bits									
PIC18FXXX	4 - 128	256 - 3936	18 - 80	32 - 48	4 - 16	10 or 12	0 - 3	0 - 2 x 8 2 - 3 x 16	CAN
PIC18FXXJXX	8 - 128	1024 - 3936	28 - 100	40 - 48	10 - 16	10	2	0 - 2 x 8 2 - 3 x 16	USB2.0 USART I2C SPI

PIC18FXXKXX	8 - 64	768 - 3936	28 - 44	64	10 - 13	10	2	1 x 8 3 x 16	USART I2C S
-------------	--------	------------	---------	----	---------	----	---	--------------	-------------

Todos los microcontroladores PIC utilizan una arquitectura Harvard, lo que quiere decir que su memoria de programa está conectada a la CPU por más de 8 líneas. Hay microcontroladores de 12, 14 y 16 bits, dependiendo de la anchura del bus. La tabla anterior muestra las características principales de estas tres categorías.

Como se puede ver en la tabla de la página anterior, salvo “los monstruos de 16 bits” PIC 24FXXX y PIC 24HXXX - todos los microcontroladores tienen la arquitectura Harvard de 8 bits y pertenecen a una de las tres grandes grupos. Por eso, dependiendo del tamaño de palabra de programa existen la primera, la segunda y la tercera categoría de microcontroladores, es decir microcontroladores de 12, 14 o 16 bits. Puesto que disponen del núcleo similar de 8 bits, todos utilizan el mismo juego de instrucciones y el “esqueleto” básico de hardware conectado a más o menos unidades periféricas.

Los microcontroladores PIC con palabras de programa de 14 bits parecen ser la mejor opción para los principiantes. Aquí está el porqué...

## JUEGO DE INSTRUCCIONES

El juego de instrucciones para los microcontroladores 16F8XX incluye 35 instrucciones en total. La razón para un número tan reducido de instrucciones yace en la arquitectura RISC. Esto quiere decir que las instrucciones son bien optimizadas desde el aspecto de la velocidad operativa, la sencillez de la arquitectura y la compacidad del código. Lo malo de la arquitectura RISC es que se espera del programador que haga frente a estas instrucciones. Por supuesto, esto es relevante sólo si se utiliza el lenguaje ensamblador para la programación. Este libro se refiere a la programación en el lenguaje de alto nivel C, lo que significa que la mayor parte del trabajo ya fue hecho por alguien más. Así, sólo se tienen que utilizar instrucciones relativamente simples.

## TIEMPO DE EJECUCIÓN DE INSTRUCCIONES

Todas las instrucciones se ejecutan en un ciclo. La únicas excepciones pueden ser las instrucciones de ramificación condicional o las instrucciones que cambian el contenido del contador de programa. En ambos casos, dos ciclos de reloj son necesarios para la ejecución de la instrucción, mientras que el segundo ciclo se ejecuta como un NOP (*No operation*). Las instrucciones de un ciclo consisten en cuatro ciclos de reloj. Si se utiliza un oscilador de 4 MHz, el tiempo nominal para la ejecución de la instrucción es 1µS. En cuanto a las instrucciones de ramificación, el tiempo de ejecución de la instrucción es 2µS.

Juego de instrucciones de los microcontroladores PIC de 14 bits:

INSTRUCCIÓN	DESCRIPCIÓN	OPERACIÓN	BANDERA	CLK	*
<b>Instrucciones para la transmisión de datos</b>					
MOVLW k	Mover literal a W	k -> w		1	
MOVWF f	Mover el contenido de W a f	W -> f		1	
MOVF f,d	Mover el contenido de f a d	f -> d	Z	1	1, 2
CLRW	Borrar el contenido de W	0 -> W	Z	1	
CLRF f	Borrar el contenido de f	0 -> f	Z	1	2
SWAPF f,d	Intercambiar de nibbles en f	f(7:4),(3:0) -> f(3:0),(7:4)		1	1, 2
<b>Instrucciones aritmético - lógicas</b>					
ADDLW k	Sumar literal a W	W+k -> W	C, DC, Z	1	
ADDWF f,d	Sumar el contenido de W y f	W+f -> d	C, DC ,Z	1	1, 2
SUBLW k	Restar W de literal	k-W -> W	C, DC, Z	1	
SUBWF f,d	Restar W de f	f-W -> d	C, DC, Z	1	1, 2
ANDLW k	AND W con literal	W AND k -> W	Z	1	
ANDWF f,d	AND W con f	W AND f -> d	Z	1	1, 2
IORLW k	OR inclusivo de W con literal	W OR k -> W	Z	1	
IORWF f,d	OR inclusivo de W con f	W OR f -> d	Z	1	1, 2
XORWF f,d	OR exclusivo de W con literal	W XOR k -> W	Z	1	1, 2
XORLW k	OR exclusivo de W con f	W XOR f -> d	Z	1	
INCF f,d	Sumar 1 a f	f+1 -> f	Z	1	1, 2
DECF f,d	Restar 1 a f	f-1 -> f	Z	1	1, 2
RLF f,d	Rotar F a la izquierda a través del bit de Acarreo		C	1	1, 2

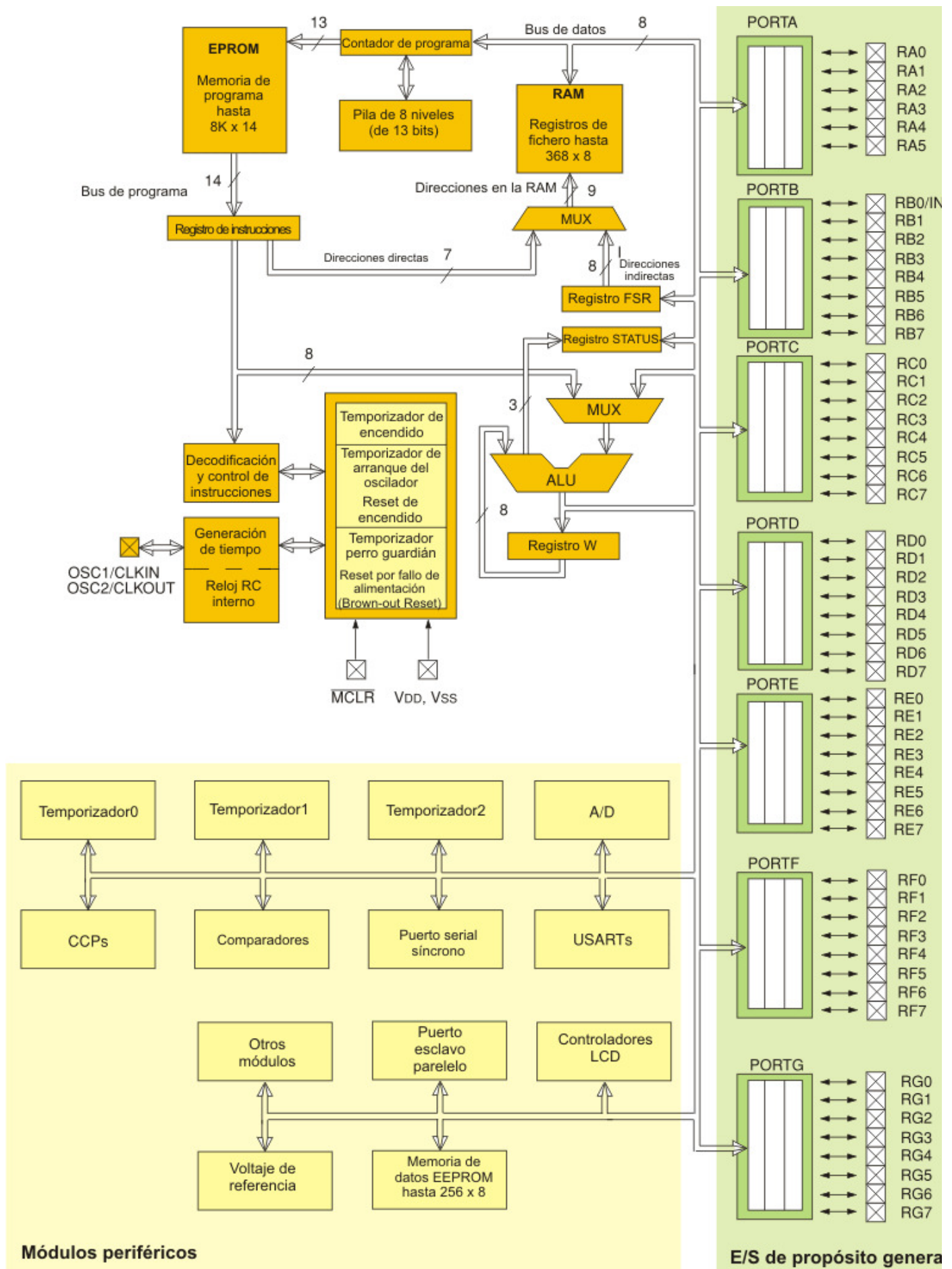
RRF f,d	Rotar F a la derecha a través del bit de Acarreo		C	1	1, 2
COMF f,d	Complementar f	f -> d	Z	1	1, 2
<b>Instrucciones orientadas a bit</b>					
BCF f,b	Poner a 0 el bit b del registro f	0 -> f(b)		1	1, 2
BSF f,b	Poner a 1 el bit b del registro f	1 -> f(b)		1	1, 2
<b>Instrucciones de control de programa</b>					
BTFSK f,b	Saltar si bit b de registro f es 0	Skip if f(b) = 0		1 (2)	3
BTFSK f,b	Saltar si bit b de reg. f es 1	Skip if f(b) = 1		1 (2)	3
DECFSZ f,d	Disminuir f en 1. Saltar si el resultado es 0	f-1 -> d skip if Z = 1		1 (2)	1, 2, 3
INCFSZ f,d	Incrementar f en 1. Saltar si el resultado es 1	f+1 -> d skip if Z = 0		1 (2)	1, 2, 3
GOTO k	Saltar a una dirección	k -> PC		2	
CALL k	Llamar a una subrutina	PC -> TOS, k -> PC		2	
RETURN	Retornar de una subrutina	TOS -> PC		2	
RETLW k	Retornar con literal en W	k -> W, TOS -> PC		2	
RETFIE	Retornar de una interrupción	TOS -> PC, 1 -> GIE		2	
<b>Otras instrucciones</b>					
NOP	No operación	TOS -> PC, 1 -> GIE		1	
CLRWDT	Reiniciar el temporizador perro guardián	0 -> WDT, 1 -> TO, 1 -> PD	TO, PD	1	
SLEEP	Poner en estado de reposo	0 -> WDT, 1 -> TO, 0 -> PD	TO, PD	1	

\*1 Si un registro de E/S está modificado, el valor utilizado será el valor presentado en los pines del microcontrolador.

\*2 Si la instrucción se ejecuta en el registro TMR y si d=1, el pre-escalador será borrado.

\*3 Si la instrucción se ejecuta en el registro TMR y si d=1, el pre-escalador será borrado.





Arquitectura de los microcontroladores PIC de 8 bits. Cuáles de estos módulos pertenecerán al microcontrolador, dependerá del tipo de microcontrolador.