

CURSO DE MICROCONTROLADORES DE LA FAMILIA PIC16 DE MICROCHIP

CAPITULO 1

Electrónica Digital: Repaso de Conceptos

En este capítulo se realizará un repaso de los campos principales que conforman la electrónica digital como tal y que son las bases para el gran avance tecnológico de los últimos años

➤ **Lógica Combinatoria:**

Solo toma en cuenta el estado de las entradas. Trata desde compuertas digitales hasta decodificadores, sumadores, multiplexores, etc. Para el diseño se basa en tablas de verdad y mapas de Karnaugh.

➤ **Lógica Secuencial:**

Toma en cuenta no solo el estado actual de las entradas, si no también el estado pasado y el tiempo en el que son aplicadas. Trata desde Flip-Flop's hasta contadores, registros, etc. Para el diseño se basa en tablas de estado en el tiempo.

➤ **Lógica Programable o no cableada:**

Reúne los principios de la lógica combinatoria y secuencial, con el concepto de almacenamiento. Trata desde memorias digitales y PLD's, hasta los Microprocesadores y Microcontroladores. Para el diseño se basa en diagramas de flujo y sentencias de control que reemplazan conexiones y circuitos digitales.

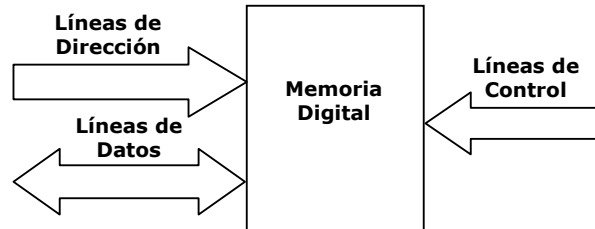
CAPITULO 2

Memorias Digitales y Dispositivos Programables

➤ **Definición de Memoria Digital:**

Las memorias digitales son dispositivos que pueden almacenar información en forma binaria dentro de casilleros denominados localidades de memoria, cada uno relacionado con una dirección específica.

Una memoria externamente consta de un conjunto de líneas de dirección, de datos y de control tal como se muestra a continuación.



➤ **Tipos de Memorias:**

RAM: Son memorias volátiles, es decir, al quitar la alimentación se borra la información en ellas, es por esto que nunca se sabe con que valor empiezan.

ROM: No son volátiles, y pueden guardar la información por varios miles de años, sin embargo, la información en ellas solo puede ser guardada una sola vez, después sirven solo como memorias de lectura.

EPROM: Similares a las ROM pero pueden ser borradas, para lo cual disponen de una ventana en la parte superior del chip, por donde se debe aplicar luz ultravioleta para borrar la información.

EEPROM y FLASH: Similares a las EPROM en el sentido de que pueden ser borradas, la diferencia radica en que estas no necesitan luz ultravioleta, son borradas eléctricamente. Las EEPROM generalmente requieren de altos voltajes en cambio que las FLASH requieren bajos voltajes, esto las hace muy útiles para ser utilizadas en dispositivos auto-programables disminuyendo el tiempo de reprogramación, de ahí el nombre de FLASH.

En general, las RAM son utilizadas para guardar información temporal (datos y variables), en cambio que las restantes son utilizadas para guardar información permanente (programas).

Las memorias ofrecen la posibilidad de cambiar el funcionamiento o la configuración de un circuito a voluntad del usuario, sin necesidad de cambiar el cableado del circuito externo.

➤ **Definición de Dispositivo Programable:**

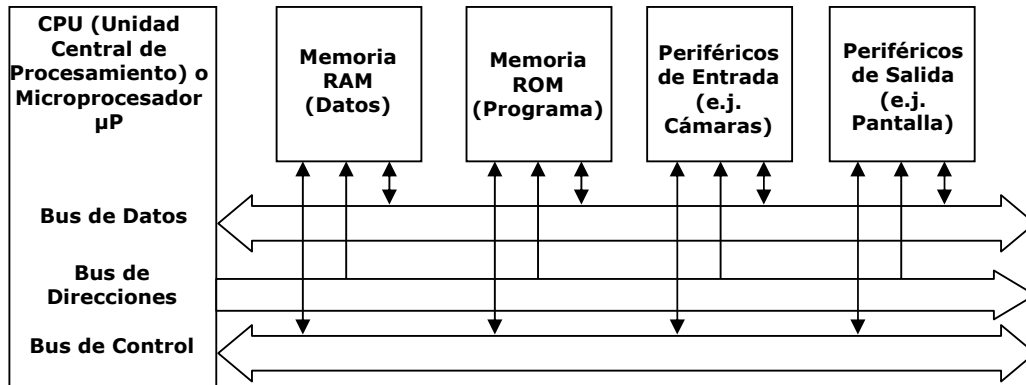
Esto da origen a los dispositivos programables, que son dispositivos capaces de realizar múltiples tareas utilizando los mismos dispositivos externos o hardware, lo único que necesitan es cambiar los datos internos, que reciben el nombre de programa o software.

CAPITULO 3

Computadora Digital, Microprocesador y Microcontrolador

➤ **Estructura y Operación Básica de una Computadora Digital:**

Como resultado de unir las memorias digitales con elementos combinatorios y secuenciales nace el computador digital mostrado a continuación y cuyos elementos son:



Microprocesador o uP: También conocido como CPU es el encargado de coordinar la operación de todo el computador, es decir, ejecución de los programas, el almacenamiento de información temporal y la comunicación con los dispositivos de entrada y salida.

Memoria: Puede ser de tipo temporal o permanente, en el primer caso se utiliza para guardar datos que son producidos por algún dispositivo o el mismo CPU (e.j. el texto de un documento de Word antes de ser grabado). El segundo tipo se utiliza para grabar el programa, que es el conjunto de operaciones que debe ejecutar el CPU.

Periféricos de Entrada y Salida o I/O (Input/Output): Son elementos que permiten al computador comunicarse con el mundo externo ya sea para enviar o para recibir información (e.j. teclados, ratones, cámaras, micrófonos, pantallas, impresoras, parlantes, etc.).

Buses: Los buses no son más que un conjunto de líneas que poseen funciones similares y que se utilizan para conectar y comunicar a los diferentes elementos de un computador. Existen básicamente 3 buses:

Bus de Direcciones: Es utilizado por el CPU para seleccionar una posición de memoria o un dispositivo específico del cual se va extraer o al cual se va a enviar información (datos). Este bus es en un solo sentido ya que solo el CPU direcciona los elementos del computador.

Bus de Datos: Se utiliza para transportar los datos que van o que vienen desde los diferentes elementos al CPU.

Bus de Control: Se utiliza para informar a los dispositivos cual es la acción a ser realizada, la cual puede ser: Encendido, apagado, reseteado, escritura, lectura, etc.

Básicamente el funcionamiento de la computadora digital se resume de la siguiente manera:

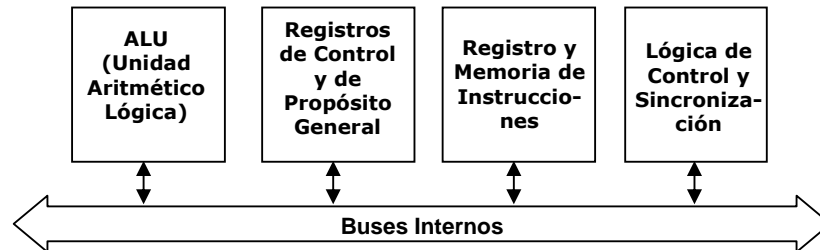
El uP constantemente lee los datos en la memoria de programa para lo cual utiliza los 3 buses: El de direcciones para indicarle a la memoria que casilla se va a leer, el de datos para sacar la información de la casilla y el de control para hacer válidas las dos operaciones anteriores, dichos datos actúan como órdenes que el uP

debe realizar. Una vez que el uP a leído e interpretado que es lo que tiene que hacer, entonces nuevamente hace uso de los buses para comunicarse con a memoria de datos o los dispositivos I/O cuando las órdenes especifiquen dispositivos externos al uP, si por otra parte la orden solo necesita elementos internos del uP entonces no son utilizados los buses.

Como se puede comprobar el CPU o Microprocesador es el cerebro del computador, ya que es el encargado de coordinar absolutamente todo, de tal manera que si el uP falla o es lento, también todo el computador fallará o será lento. En el siguiente punto se analiza la estructura interna de este dispositivo.

➤ **El Microprocesador:**

Independientemente del fabricante (Intel, AMD, Motorola, etc.), todos los uP's tiene una estructura mínima que es común entre todos y es mostrada a continuación junto con sus elementos principales:



ALU: Esta unidad se encarga de ejecutar operaciones aritméticas y lógicas tales como suma, multiplicación, AND, OR, NOT etc.

Registros de Control y de Propósito General: Los registros de control guardan la configuración de equipo como puede ser: velocidad de operación, capacidad de memoria, interrupciones, etc. En cambio los de propósito general se utilizan como destino o fuente de datos para la ALU más comúnmente.

Registro y Memoria de Instrucciones: Es una pequeña memoria ROM que contiene todas las instrucciones reconocibles por el uP. Cuando el uP realiza la lectura de instrucciones de la memoria de programa, estas vienen hasta este bloque, para que en base a ellas se generen señales de control para el resto de bloques.

Lógica de Control y Sincronización: Se encarga de generar las señales de control para dispositivos externos, es decir, señales de lectura, escritura, activación o desactivación, reset, sleep, etc. Además genera señales de reloj para los bloques internos.

➤ **El Microcontrolador:**

Al descubrir las posibilidades que ofrecía el computador, se dio una gran revolución en cuanto al desarrollo de aplicaciones comunes que por siempre habían sido realizadas manualmente y se dio origen a una nueva etapa de automatización, sin embargo, una desventaja que cada vez se hacía más intensa fue el hecho de que una computadora siempre debía disponer de una serie de dispositivos externos que muchas veces no eran necesarios. Un ejemplo de esto era la utilización de la pantalla de vídeo, cuando era suficiente con un LED o un Display, o el teclado y el ratón cuando era suficiente disponer de un único pulsante.

En base a las nuevas necesidades de las personas se originó el Microcontrolador (**µC**) que básicamente no es más que una computadora comprimida, es decir, tiene una CPU, memoria de datos y de programa, periféricos I/O y buses, pero todo esta incluido en un solo integrado o chip, sin embargo es necesario aclarar que no posee la misma capacidad que tiene un computador normal, a esto se hace referencia, a que posee muy poca memoria, son lentos, etc.,

COLEGIO TÉCNICO SALESIANO
Curso de Microcontroladores de la Familia PIC16 de MICROCHIP

relativamente hablando. Aún así, poseen características únicas que hacen de ellos, dispositivos muy populares hoy en día, una de ellas es su pequeño tamaño y que a pesar de no poseer las capacidades de las computadoras reales superan por mucho las exigencias del público.

CAPITULO 4

Microcontroladores MICROCHIP

➤ **Introducción:**

Existen varias empresas dedicadas a la fabricación de semiconductores, que entre una de sus varias líneas de producción se dedican al diseño y construcción de microcontroladores entre estas se encuentran INTEL, ATMEL, ANALOG DEVICES, MOTOROLA, INFIDEON, MICROCHIP, SIEMENS, etc. De estas nos concentraremos en la familia de microcontroladores de MICROCHIP, que en los últimos años se han posicionado como uno de los número uno en ventas a nivel internacional con su familia de microcontroladores PIC, siglas de **Peripheral Integrated Controller** que significa Controlador de Periféricos Integrados.

➤ **¿Por qué Microchip?:**

Antes de contestar esta pregunta, es necesario definir los parámetros bajo los cuales se debería optar por un uC de algún fabricante específico, estos parámetros se listan a continuación:

- *Aplicación a Desarrollarse:* Este es probablemente el más importante de todos ya que de alguna manera nos da una idea de las características mínimas que debería tener el uC a adquirir, tales como memoria, velocidad, número de puertos, periféricos de entrada y salida, consumo de energía, etc.
- *Presupuesto Disponible:* Aunque muchas veces pasado por alto, es un factor importante ya que nos ayudará a discriminar entre un gran número de uC's.
- *Conocimiento Previo:* Si el tiempo disponible para realizar el proyecto es muy corto, es preferible optar por una familia ya conocida y utilizada, antes que lanzarse a estudiar una nueva familia.
- *Disponibilidad en el Medio:* Como regla básica siempre se debería buscar dispositivos que puedan ser fácilmente adquiribles en el medio ya que se podría caer en gastos adicionales al traer componentes desde otros lugares, por no mencionar el tiempo de entrega.
- *Variiedad de Herramientas para la Programación:* Esto hace referencia al software y hardware necesario para implementar la aplicación, esto es de vital importancia, ya que algunas empresas ofrecen libremente software para la programación en cambio que otros solo demos o en el peor de los casos el software debe ser comprado, de igual manera con respecto a los programadores o grabadores.
- *Extensa Documentación:* Todos los fabricantes ofrecen libremente las hojas de datos (*Data Sheets*) de sus productos, pero pocos ofrecen documentación sobre aplicaciones ya realizadas con sus productos en donde se incluyan el software y hardware correspondiente. Esto puede resultar de gran ayuda para realizar rápidamente nuestra aplicación.
- *Variiedad de Dispositivos:* Finalmente se debe tomar en cuenta este parámetro con el fin de no desperdiciar recursos del uC y consecuentemente disminuir el costo del mismo, muchos fabricantes si bien poseen dispositivos muy potentes no tienen variedad.

En base a estos conceptos, podemos decir que MICROCHIP a pesar de no tener un microcontrolador tan poderoso como muchos otros, compensa en variedad tanto de dispositivos como de herramientas, disponibilidad, documentación y costo. Esta es la razón de su actual auge en nuestro medio.

➤ **Familia de Microcontroladores MICROCHIP:**

Los microcontroladores Microchip en general se clasifican en varias familias de PIC's, según las características que compartan:

COLEGIO TÉCNICO SALESIANO
Curso de Microcontroladores de la Familia PIC16 de MICROCHIP

PIC10XXX	uC de 6 pines
PIC12XXX	uC de 8 pines
PIC16XXX	uC de 18 a 80 pines
PIC18XXX	uC de 18 a 80 pines con hardware optimizado
rfPIC	uC de 18 pines con Transceptor de Radio Frecuencia
dsPIC	uC de 18 a 100 pines para procesamiento de señales digitales

Cada familia a su vez se puede dividir en sub-familias cada una con características específicas de memoria, velocidad, tamaño y periféricos. En lo que resta del curso se abarcará la familia PIC16F8XX y de esta se analizará el integrado PIC16F871.

CAPITULO 5

Microcontrolador PIC16F871

➤ **Características Generales:**

Entre las características más sobresalientes de este dispositivo, se tienen las siguientes:

- Viene en un empaque de 40 pines de los cuales 33 son utilizables para propósito general
- Puede trabajar a velocidades de hasta 20MHz.
- La memoria de Programa es de tipo FLASH y puede almacenar un programa de hasta 2K (2048) instrucciones.
- La memoria de Datos tienen capacidad de hasta 128 bytes.
- Posee una memoria de Datos adicional pero de tipo EEPROM de hasta 64 bytes.
- Puede trabajar con tensiones desde 2V hasta 5.5V.
- Cada una de sus pines puede proveer máximo hasta 25mA de corriente.
- Posee tres temporizadores o Timers programables.
- Posee un conversor analógico digital de 8 canales con resolución de 10bits
- Posee 1 módulos CCP (Comparar, Capturar y PWM)
- Sus pines son multifunción.
- Tiene un puerto de comunicaciones seriales de tipo USART.

➤ **Arquitectura Interna:**

A continuación se da una breve descripción de los bloques más importantes de este dispositivo:

Memoria de Programa y de Datos: Estos bloques son muy importantes ya que nos permiten almacenar el programa a ejecutarse y las variables temporales que el programa necesite.

Buses de Datos, Direcciones y Control: Son los encargados de transmitir las señales y la información para controlar a todos los distintos bloques del microcontrolador.

Contador de Programa: Es el encargado de direccionar una por una, en orden ascendente, las casillas de la memoria de programa, excepto cuando se trata de subrutinas, saltos o interrupciones. Este contador siempre empieza en cero, es decir, siempre lee la casilla cero de la memoria de programa después de un reset o del encendido del uC.

Pila (Stack): Cuando como resultado de una subrutina o interrupción el contador de programa tiene que dejar de contar en orden normal, para cargar una nueva dirección, la dirección actual se guarda en la pila, de esta manera cuando el uC sabe donde estaba antes de ir a la subrutina o a la interrupción.

Control y decodificador de Instrucciones: En este bloque se entiende la orden o instrucción guardada en la memoria de programa, y por ende es el encargado de controlar todo el resto del uC.

Generador de Tiempos: Aquí se recibe la señal del reloj externo (reloj significa un tren de pulsos), y se distribuye por todo el sistema de manera que todos trabajan a la misma velocidad.

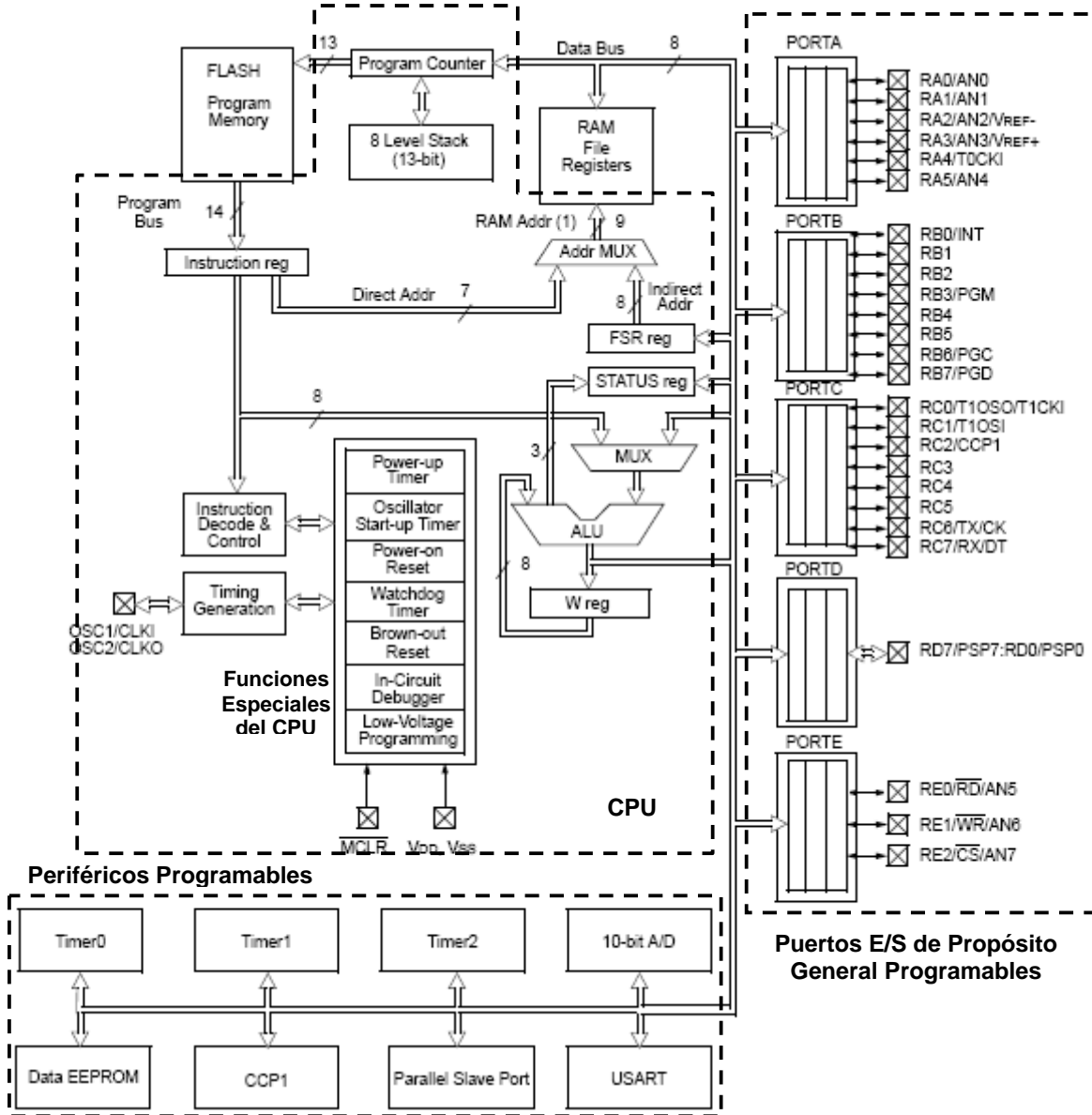
ALU (Unidad Lógica Aritmética): La ALU es la encargada de realizar las operaciones aritméticas y lógicas entre las que se incluyen suma, resta, AND, OR, XOR y NOT.

Registro Acumulador o W (working): Este registro es el más importante de todos, porque a través de él, se realizan un sin número de operaciones e instrucciones, en especial aquellas relacionadas con la ALU.

Registro de Estado (STATUS): Este se utiliza para indicar entre otras cosas, el resultado de una operación en la ALU. Por ejemplo si una suma dio como resultado un número mayor a 8 bits, un bit dentro de este registro

COLEGIO TÉCNICO SALESIANO
Curso de Microcontroladores de la Familia PIC16 de MICROCHIP

indica esta condición (bit de carry), por otra parte si el resultado de cualquier operación fue cero, otro bit indicará esta condición (bit "zero").



MCLR/Vpp/THV	1	40	RB7/PGD
RA0/AN0	2	39	RB6/PGC
RA1/AN1	3	38	RB5
RA2/AN2/VREF-	4	37	RB4
RA3/AN3/VREF+	5	36	RB3/PGM
RA4/T0CKI	6	35	RB2
RA5/AN4	7	34	RB1
RE0/RD/AN5	8	33	RB0/INT
RE1/WR/AN6	9	32	VDD
RE2/CS/AN7	10	31	VSS
VDD	11	30	RD7/PSP7
VSS	12	29	RD6/PSP6
OSC1/CLKI	13	28	RD5/PSP5
OSC2/CLKO	14	27	RD4/PSP4
RC0/T1OSO/T1CKI	15	26	RC7/RX/DT
RC1/T1OSI	16	25	RC6/TX/CK
RC2/CCP1	17	24	RC5
RC3	18	23	RC4
RD0/PSP0	19	22	RD3/PSP3
RD1/PSP1	20	21	RD2/PSP2

Puertos de Propósito General (PORT A, B, C, D, E): Los puertos no son más que los pines externos del uC, estos puertos pueden ser programados para actuar como entradas de datos (ej. Observar el estado de un pulsante, dipswitch, sensores, teclados, etc.) o como salida de datos (ej. Controlar el encendido de Led's, Displays, Motores, etc.).

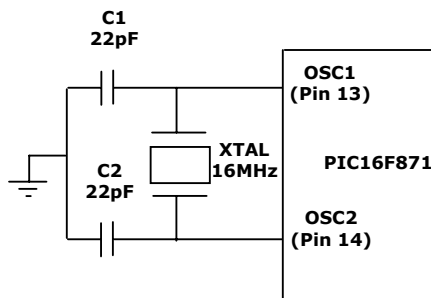
Periféricos Específicos: Son un conjunto de circuitos especializados que realizan funciones específicas (Temporizadores, Conversor Analógico Digital, etc.), comparten pines con los puertos, es decir, cuando los pines son utilizados por los periféricos específicos no se pueden utilizar los puertos como simples entrada y salidas.

Pin de Reset: Cuando se da un cero (tierra) en esta pin, el uC vuelve a cero, es decir el contador de programa regresa a la casilla cero y empieza de nuevo.

Funciones Especiales del CPU: Son un conjunto de bloques que permiten mejorar el rendimiento y la utilización de este dispositivo. Entre los más importantes están: detector de baja tensión, autoreset de inicialización, temporizador contra fallos, circuito para programación en línea, etc.

➤ **Oscilador Externo y Ciclo de Instrucción:**

El uC necesita externamente un circuito de reloj, es decir, un circuito que genere pulsos digitales de tal manera que se puedan ejecutar las ordenes de la memoria de programa. Existen varias formas de producir estos pulsos, pero la más usada es la que utiliza un cristal piezoeléctrico en conjunto con 2 condensadores, que son conectados a los pines del uC denominados OSC1 y OSC2. El esquema de esta conexión se muestra a continuación junto con el valor de los componentes y una tabla de referencia para futuros diseños:



Osc Type	Crystal Freq	Cap. Range C1	Cap. Range C2
LP	32 kHz	33 pF	33 pF
	200 kHz	15 pF	15 pF
XT	200 kHz	47-68 pF	47-68 pF
	1 MHz	15 pF	15 pF
	4 MHz	15 pF	15 pF
HS	4 MHz	15 pF	15 pF
	8 MHz	15-33 pF	15-33 pF
	20 MHz	15-33 pF	15-33 pF

Por medio del circuito de reloj se genera el ciclo de instrucción o también llamado "ciclo de máquina", que es el tiempo que se toma el uC para llevar a cabo una instrucción u orden de la memoria de programa, desde el momento en que lee la casilla de memoria hasta que la ejecuta en su totalidad y esta listo para leer la siguiente casilla.

Matemáticamente el ciclo de instrucción es igual a la siguiente expresión:

$$C. Inst. = 4 / F$$

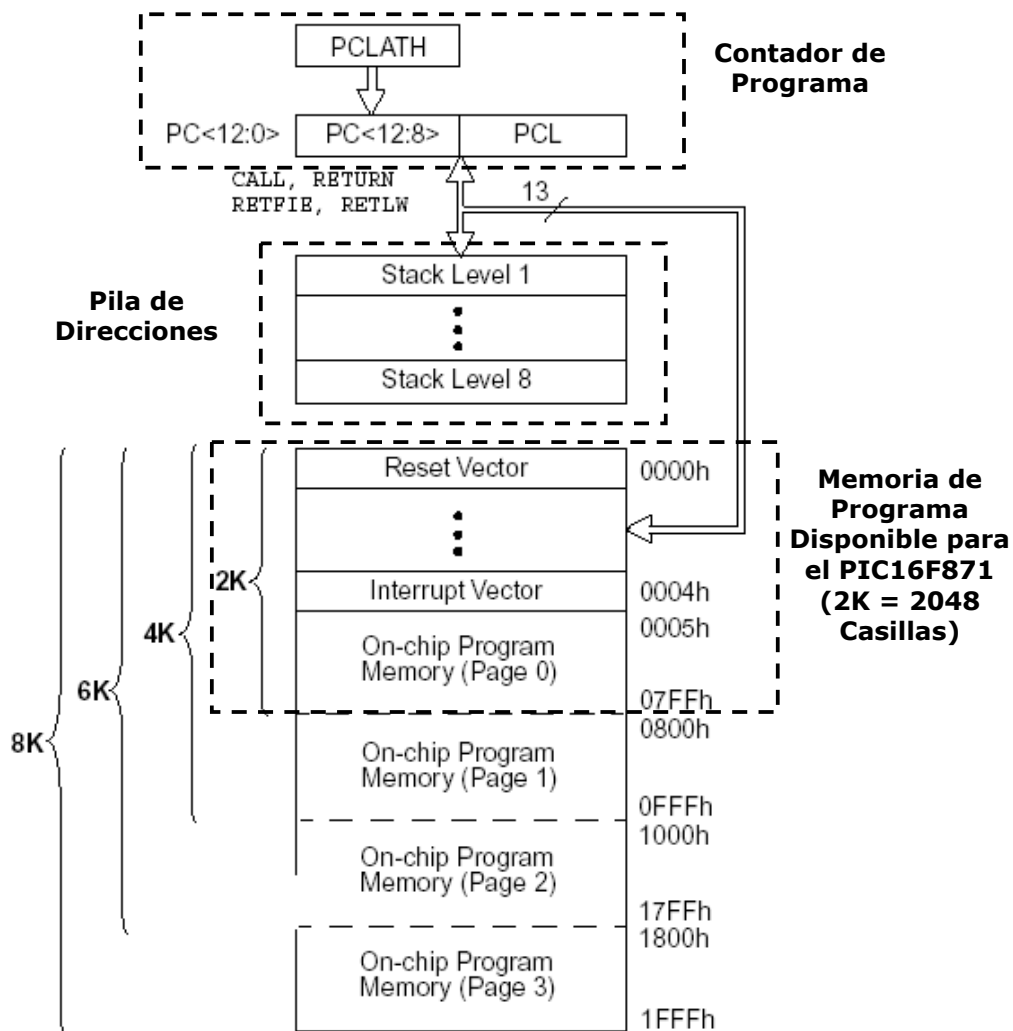
Donde F es la frecuencia del reloj del uC, en nuestro caso la frecuencia del oscilador externo. En la siguiente tabla se muestra el ciclo de instrucción para diferentes cristales:

Cristal (MHz):	C. Inst. (nseg):
4	1000 (ó 1useg)
10	400
12	333.33_
16	250
20	200

De la tabla se puede comprobar que mientras más alto sea la frecuencia del cristal en menor tiempo podrá realizar una instrucción, esto es importante cuando la aplicación que se desarrolla necesita ser ejecutada rápidamente, sin embargo se debe considerar que mientras más rápido trabaje el uC mayor será su consumo lo cual también es importante en aplicaciones portátiles como juguetes, walkman's, teléfonos celulares, etc.

➤ **Memoria de Programa:**

La familia de uC PIC16XXXX es capaz de soportar hasta 8K (8096 casillas) de memoria de programa ordenadas en páginas de 2K. Como ya se había mencionado, para leer cada casilla de memoria se hace uso del Contador de Programa el cual no es más que la unión de dos registros llamados PCL y PCLATH cada uno de 8 bits, juntos pueden leer todas las casillas de memoria.



COLEGIO TÉCNICO SALESIANO
Curso de Microcontroladores de la Familia PIC16 de MICROCHIP

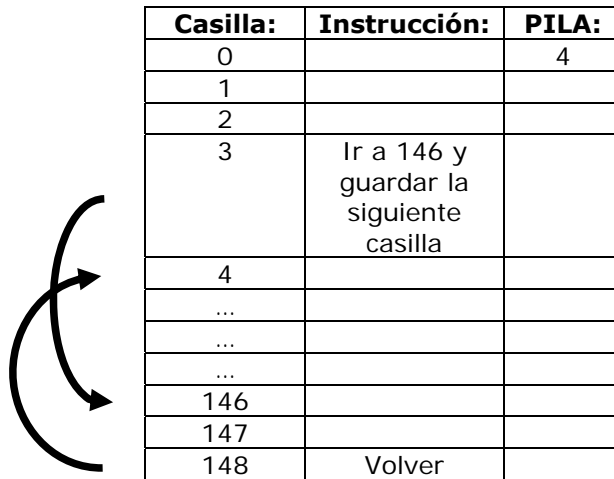
Normalmente estos dos registros siempre cuentan en forma normal, es decir, con cada ciclo de máquina aumentan uno a su conteo binario, tal como lo muestra la siguiente tabla:

PCLATH	PCL	Casilla Leída
0 0000	0000 0000	0
0 0000	0000 0001	1
0 0000	0000 0010	2
0 0000	0000 0011	3
...
0 0000	1111 1111	255
0 0001	0000 0000	256
...
0 0111	1111 1110	2046
0 0111	1111 1111	2047
...
1 1111	1111 1111	8191

Pero existen instrucciones u órdenes que hacen que estos registros cambien su conteo normal, y en vez de leer la siguiente casilla se dirijan a otras casillas de memoria, como en la siguiente figura:

Casilla:	Instrucción:
0	
1	
2	
3	
4	Ir a 146
...	
...	
...	
146	
147	
148	Ir a 1022
...	
...	
...	
1022	
1023	
1024	
1025	
1026	Ir a 1

Adicionalmente al Contador de Programa, se encuentra la pila de programa o pila de direcciones, que es un espacio de 8 casillas independiente de la memoria del dispositivo, que el contador de programa utiliza para guardar la dirección de la última casilla leída más uno, a la cual debe regresar siempre que se va a producir un salto del que se tiene que regresar (ej. Subrutinas e Interrupciones). En la siguiente figura se muestra este concepto.



➤ **Memoria de Datos tipo RAM:**

La memoria de datos tipo RAM al igual que la de programa también se encuentra dividida en bloques o bancos, específicamente se tienen 4 bancos de memoria de datos cada uno con 128 registros (00 hasta 7F). Cada banco se subdivide en dos grupos de registros denominados: Registros de Funciones Especiales y Registros de Propósito General.

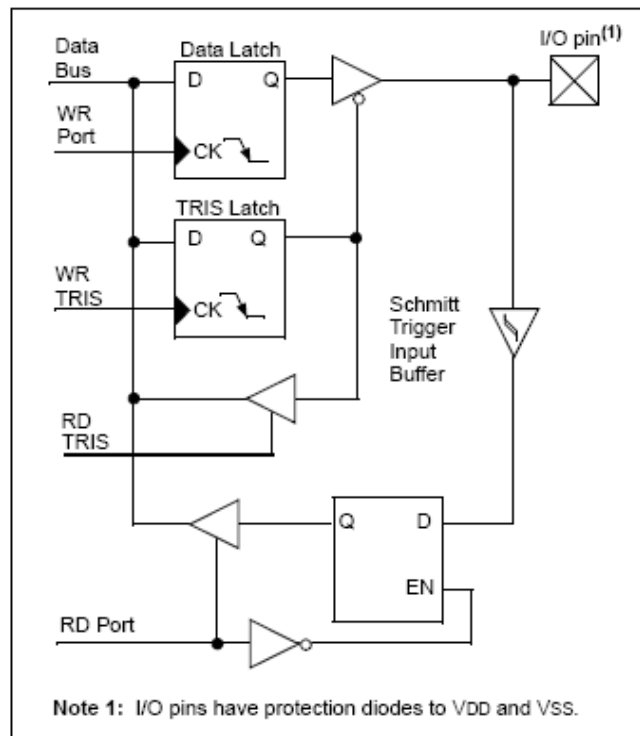
Estos se muestra a continuación en la página siguiente:

el programa original con la clave cambiada, otro ejemplo pueden ser sistemas de temporización de eventos para invernaderos, luces automáticas, ventiladores, etc., para los cuales puede ser necesario configurar periódicamente diferentes tiempos de activación y desactivación.

En general la EEPROM de datos que dispone el uC, se caracteriza por tener un sistema de grabado y borrado integrado, lo cual elimina la necesidad de tener una fuente adicional ya que funciona con la misma tensión que es alimentado el integrado, además los tiempos de programación son muy pequeños (5ms o menos). Sin embargo una desventaja es que solo se disponen de 64 bytes que pueden ser programados, aunque para la gran mayoría de aplicaciones esto es más que suficiente.

➤ **Puertos de Entrada y Salida:**

En el PIC16F871 se dispone de 6 puertos de entrada y salida de propósito general que multiplexan funciones con los periféricos internos. A continuación se muestra el esquema interno básico de cada línea de los puertos junto con una breve descripción de cada puerto disponible:



PORTA: Posee 6 líneas, este puerto junto con el PORTE, son las entradas del Módulo conversor A/D y del TIMER 0.

PORTB: Posee 8 líneas, este puerto es comúnmente utilizado para el manejo de teclados matriciales y en general cualquier tipo de pulsante o interruptor ya que posee resistencias internas de pull-up configurables e interrupciones programables por cambio de estado.

PORTC: Posee 8 líneas, este puerto es usado por el puerto de comunicaciones seriales (USART), el TIMER1 y el Módulo de CCP.

PORTD: Posee 8 líneas, este puerto es usado por el Módulo de Comunicaciones Paralelas como bus de Datos.

PORTE: Posee 3 líneas, este es utilizado para generar las señales de control para el Módulo de Comunicaciones Paralelas.

Cada línea de un puerto puede ser programada para operar como entrada o como salida independientemente de cómo estén programadas el resto de líneas del mismo puerto. Esto resulta de gran ayuda para evitar el desperdicio de líneas, considere como ejemplo de desperdicio el puerto paralelo de una computadora. Para permitir esta funcionalidad de los puertos, cada uno posee dos registros llamados TRISX y PORTX (X = A,B,C,D,E) cada uno de 8 bits (en el caso de puertos con menos de 8 líneas los bits sobrantes no se utilizan).

El registro TRIS se utiliza para configurar una línea como entrada (1) o como salida (0), en cambio que el registro PORT se utiliza para leer o escribir 1's o 0's en las líneas del puerto.

➤ **Periféricos Específicos:**

Existe una gran variedad de periféricos disponibles en el PIC16F871, lo que hace que este dispositivo sea ideal para el presente curso, ya que nos permite explorar muchas aplicaciones interesantes y de gran utilidad.

A continuación se presenta una breve descripción de cada uno de estos periféricos dejando para más adelante, cuando se empiecen a desarrollar las prácticas, una descripción más profunda y detallada:

TIMERS: Son los periféricos más sencillos pero aún así los más útiles y populares. Básicamente son contadores, cada vez que reciben un pulso digital de alguna fuente externa (puerto E/S, oscilador, etc.) o interna (pulsos provenientes del reloj del sistema), estos incrementa su conteo actual por una unidad, al completarse el número máximo de pulsos contables estos vuelven a cero y generan una señal que le indica al uC que se ha producido un **Desbordamiento**. Algunas aplicaciones incluyen: generador de demoras de tiempo o *delays* por hardware y contadores de eventos. Este uC posee 3 TIMERS, dos de 8 bits (máximo 256 pulsos) y uno de 16 bits (máximo 65536 pulsos).

CONVERSOR A/D: Este periférico también es otro de los más populares. Como su nombre lo indica su función es transformar una señal analógica en una representación digital similar, la fidelidad de esta representación dependerá de la resolución del conversor (número de bits que se utilizan para representar la señal analógica) y de la velocidad con la cual se tomen muestras de dicha señal (tiempo de muestreo). Entre las aplicaciones más importantes están: Multímetros Digitales en general, Osciloscopios Digitales, Sistemas de Control para industrias (medidores de calor, humedad, fuerza, presión, etc.). Este uC posee un conversor de 8 canales con una resolución de 10 bits (máximo 4096 valores de una señal).

PUERTO DE COMUNICACIÓN SERIAL USART: Este puerto encabeza la lista de puertos de comunicación seriales más populares, seguido del puerto USB, debido básicamente a su facilidad de implementación y manejo, y a la relativamente muy buena velocidad de transferencia de datos. En su forma más sencilla consiste de dos alambres uno para transmisión y otro para recepción que funcionan en forma independiente o asíncrona, es decir, no es necesario mandar una señal para recibir otra. Los voltajes en estas líneas se transmiten a altos voltajes con el fin de disminuir el efecto de atenuación por la distancia. Por esta misma razón en cada punto final se requiere un circuito adicional que transforme de altos voltajes a voltajes TTL y viceversa. Algunas aplicaciones incluyen: Sistemas de adquisición para la PC como osciloscopios, plantas industriales, invernaderos, etc., Sistemas de seguridad controlados por la PC, Sistemas de Comunicación inalámbricos con Radiofrecuencia o Infrarrojos para tele presencia, etc.

MÓDULO CCP: Este puede tener hasta 3 funciones configurables. La primera es como capturador, en esta se intenta hallar el tiempo requerido por un evento externo. Este modo es utilizado para el diseño de

frecuencímetros y tacómetros digitales. El segundo modo es como comparador, en este en cambio se busca generar un evento interno o externo cada cierto tiempo que ha sido previamente programado en este módulo. Una aplicación de este modo es como generador de ondas cuadradas de periodo y frecuencia variable para sintetizadores y generadores de tono. Finalmente el último modo es como generador de PWM, que es una señal de frecuencia constante pero de periodo ajustable. Probablemente este modo es el más utilizado de los tres anteriores ya que encuentra más aplicabilidad en aplicaciones industriales donde es necesario el control de velocidad de motores, temperatura de hornos, apertura de válvulas, iluminación de lámparas, etc.

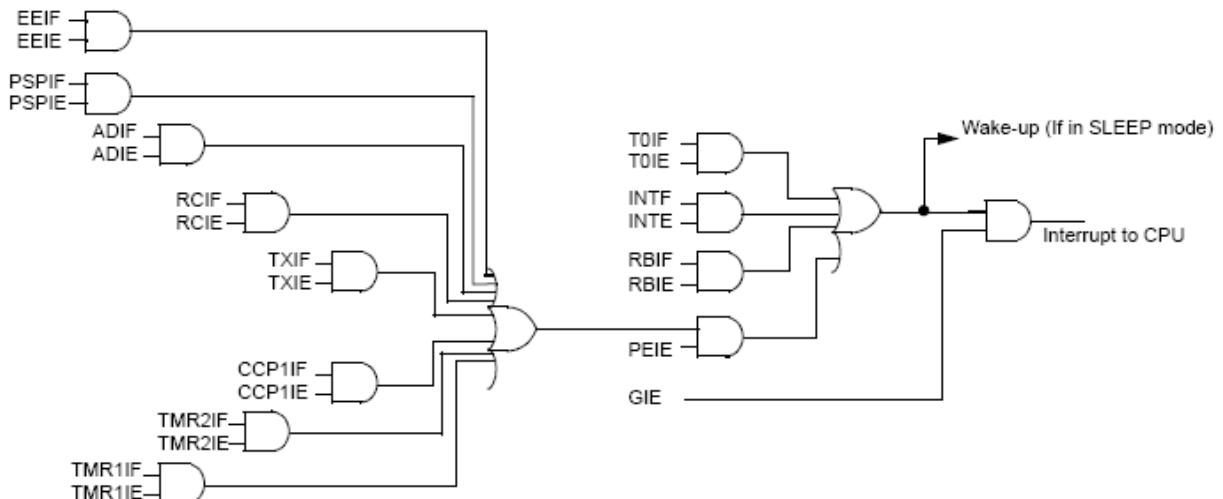
➤ **Interrupciones:**

El entender y dominar el tema de las interrupciones es uno de los objetivos más importantes del presente curso ya que representa una herramienta de gran utilidad tanto para microprocesadores como para microcontroladores, esto se debe entre otras a las siguientes razones: Permite el uso eficiente de todos los recursos del uC, optimiza el espacio utilizado de memoria de programa para la realización de una aplicación cualquiera, disminuye el consumo de energía del sistema al permitir que el uC pueda entrar en estado de hibernación o modo SLEEP durante la realización de un proceso específico, etc.

En forma sencilla se puede resumir el manejo de las interrupciones de la siguiente manera: En todo momento se puede tener un programa principal y uno o varios programas secundarios corriendo al mismo tiempo en el uC, sin embargo en algún momento específico un programa secundario puede requerir tomar control sobre el uC, por lo que este puede interrumpir al programa principal, realizar algunos procesos de breve duración y volver a su estado normal devolviendo el control sobre el uC al programa principal.

Como ejemplo de esto suponga que se tiene un juego de luces multi-secuencia. El programa principal en este caso se encarga de ejecutar la secuencia actual y un programa secundario se encarga de chequear los pulsantes que determinan la secuencia siguiente. Cuando se halla pulsado una tecla este programa interrumpirá al programa principal indicándole que tiene que cambiar la secuencia, una vez hecho esto, el programa secundario vuelve a su estado normal en cambio que el principal ahora ejecuta una secuencia diferente.

En el siguiente gráfico se puede observar la jerarquía en cuanto a las interrupciones que puede aceptar el uC y seguidamente se presenta una descripción de las mismas:



- EEIX:** Interrupción por escritura de la EEPROM de Datos.
- PSPIX:** Interrupción por escritura/lectura del puerto paralelo.
- ADIX:** Interrupción por finalización de conversión A/D.
- RCIX:** Interrupción por recepción de dato en el puerto USART.
- TXIX:** Interrupción por transmisión de dato en el puerto USART.
- CCP1IX:** Interrupción por evento en el módulo CCP1.
- TMR0IX:** Interrupción por desbordamiento del TIMER 0.
- TMR1IX:** Interrupción por desbordamiento del TIMER 1.
- TMR2IX:** Interrupción por desbordamiento del TIMER 2.
- INTX:** Interrupción por cambio de estado en el pin 0 del PORTB (RB0).
- RBIX:** Interrupción por cambio de estado en los pines 4 a 7 del PORTB.

Observe que para cada interrupción se tienen dos líneas una con terminación E y otra con terminación F. La primera se utiliza para habilitar (Enable) la interrupción y la otra para indicar que la interrupción a sucedido (Flag = bandera). Si no se activa la interrupción así el periférico requiera la atención del uC nada sucederá. Observe además la presencia de dos líneas adicionales llamadas PEIE y GIE. La primera se utiliza para habilitar a todas las interrupciones provenientes de periféricos y la otra se utiliza como línea de habilitación global de todas las interrupciones.

Estas líneas, tanto las de habilitación como las banderas de indicación, se encuentran en los registros INTCON, PIE1, PIR1, PIE2 y PIR2 (ver memoria RAM).

➤ **Palabra de Configuración Global:**

La palabra de configuración, no es más que un registro que se utiliza para indicarle al microcontrolador que funcione en un modo específico o que active o desactive ciertas funciones especiales del mismo. Sin embargo este registro no puede ser accedido durante el funcionamiento normal del uC ya que no se encuentra en la memoria datos ni en la de programa, solo puede ser accedido durante el proceso de grabación del chip.

A continuación se muestra este registro junto con una breve descripción de sus bits de configuración:

CP1	CP0	DEBUG	—	WRT	CPD	LVP	BOREN	CP1	CP0	PWRTE ⁿ	WDTEN	FOSC1	FOSC0
bit 13											bit 0		

- CP1,CP0,WRT:** Protección de Código de Memoria de Programa
- DEBUG:** Habilitación del Depurador en línea
- CPD:** Protección de Código de Memoria de Datos EEPROM
- LVP:** Programación en bajo voltaje
- BOREN:** Reset cuando se produzca caída de tensión
- PWRTEⁿ:** Reset inicial de 72ms después de conectar la alimentación
- WDTEN:** Habilitación del Perro Guardián
- FOSC1, FOSC0:** Modo de Operación del Reloj del UC: RC, XT, HS, LP (ver Oscilador Externo y Ciclo de Instrucción)

La habilitación de estas funciones se las puede realizar a través de una ventana especial disponible en el entorno gráfico donde se realice la programación del chip.

CAPITULO 6

Lenguaje de Programación MikroBasic

➤ Introducción

Como ya se había mencionado, una de las razones de porque los uC de Microchip son tan populares es la gran variedad de herramientas que se disponen para realizar aplicaciones con ellos. Entre estas herramientas se tiene el MikroBasic (desde ahora en adelante se utilizarán las siglas MB), que es un lenguaje de Programación basado en el popular lenguaje BASIC, pero que se encuentra orientado hacia los microcontroladores de MICROCHIP, es decir, en esencia programar MB es similar a programar en BASIC o en cualquier otro programa igualmente basado en BASIC tal como VisualBASIC.

De igual manera la pregunta obvia sería ¿por qué MB?. Si bien existen otros programas tales como PicBasic, ProPIC, HiTech, CPIC, etc., que también pueden ser utilizados para efectos de este curso, MB ofrece no solo un lenguaje amigable y fácil de utilizar, si no también una amplia variedad de librerías, que permiten controlar en forma extremadamente sencilla todos los periféricos de uC así como también periféricos externos tales como Pantallas LCD sencillas y gráficas. Adicionalmente MB ofrece un entorno gráfico de Programación con varias herramientas que facilitan la creación y prueba de aplicaciones de cualquier tipo, a todo esto se le suma el hecho de ser gratuito.

➤ Estructura de un Programa

Para empezar, es necesario conocer como esta estructurado un programa en MB, para tal propósito a continuación se muestra dicha estructura junta con una breve indicación de cada uno de sus bloques constituyentes:

```
program <Nombre_del_Programa>
include <Nombre_de_Librería>

'*****
'* Declaraciones Globales:
'*****

' Declaración de Constantes
const ...

' Declaración de Variables
dim ...

' Declaración de Procedimientos
sub procedure nombre_procedimiento(...)
<Declaraciones Locales>
...
end sub

' Declaración de Funciones
sub function nombre_función(...)
<Declaraciones Locales>
...
end sub

'*****
'* Programa Principal:
'*****
main:
' A partir de aquí se escribe el código
```

`end.`

En lo que sigue de este capítulo se explica cada uno de estos sectores así como también algunos de los elementos propios del lenguaje MB:

➤ **Bloque Principal**

El corazón de cualquier programa en MB, es el bloque principal, el cual se muestra a continuación:

```
program MiProyecto
' Bloque Principal
main:
' Aquí se escribe el código
end.
```

Como se observa el bloque principal incluye las declaraciones **program**, **main** y **end.**, este bloque es importante porque los comandos puestos aquí son los que realmente ejecutará el UC, no sucede lo mismo con las funciones y procedimientos, estos necesitan ser llamados por el bloque principal para funcionar.

➤ **Constantes y Variables: Tipos y Declaración**

Como en todo lenguaje de programación, las unidades básicas de datos son dos: constantes y variables. Las primeras son unidades cuyo valor se mantiene a lo largo del todo el programa, en general son útiles para definir dimensiones, direcciones, cantidades, etc., en cambio que las segundas son aquellas cuyo valor puede ser alterado mediante operaciones aritméticas o lógicas.

Dependiendo de la cantidad de memoria que utilicen en el UC (recuerde que la unidad mínima de memoria RAM es 1byte = 8bits), ambas pueden ser cualquiera de los siguientes tipos:

TIPO:	#BITS:	RANGO:
<code>byte</code>	8-bit	0 .. 255
<code>char*</code>	8-bit	0 .. 255
<code>word</code>	16-bit	0 .. 65535
<code>short</code>	8-bit	- 128 .. 127
<code>integer</code>	16-bit	-32768 .. 32767
<code>longint</code>	32-bit	-2147483648 .. 2147483647
<code>float</code>	32-bit	$\pm 1.17549435082 \cdot 10^{-38} \dots$ $\pm 6.80564774407 \cdot 10^{38}$

A continuación se muestra la manera de declarar una constante y una variable respectivamente, así como algunos ejemplos:

```

const nombre_constante as tipo = valor

const letra as char = "n"      ' constante tipo char valor "n"
const radio as short = -15    ' constante tipo short valor -15

dim nombre_variable as tipo

dim i, j, k as byte           ' variables de tipo byte
dim producto as float        ' variable de tipo float
  
```

Note que en el caso de variables se puede declarar más de una si todas son del mismo tipo, esto no es aplicable a constantes. Tanto para variables como constantes, se puede requerir en algún punto trabajar directamente con valores binarios o hexadecimales (manejo de puertos y displays de 7 segmentos), para esto solo es necesario utilizar la notación \$ para valores hexadecimales y % para valores binarios, como se muestra a continuación:

```

const P as word = $A56D      ' constante tipo word valor AF56D

dim CONTADOR as byte         ' variable de tipo byte
CONTADOR = %01001011         ' asigna valor a CONTADOR
  
```

Un caso especial de variables, son los registros de funciones específicas del uC (ver memoria RAM), los cuales no necesitan ser declarados ya que todos son de tipo byte y son reconocidos por el programa como variables globales:

```

TRISB = %11110000           ' Configura la mitad inferior del puerto
                             ' B como salidas.
PORTB = %11111111           ' Pone en 1 la mitad inferior del puerto
                             ' B.
  
```

➤ Operadores Aritméticos, Relacionales, Lógicos y Boléanos

Los operadores son elementos que permiten realizar algún tipo de operación aritmética o lógica sobre variables o constantes. Existen 4 tipos de operadores mismos que se explican a continuación:

- **Operadores Aritméticos:** Son los que realizan operaciones matemáticas y el resultado que devuelven es de tipo numérico y son:

OPERADOR:	OPERACIÓN:
+	Suma
-	Resta
*	Multiplicación
/	División Normal (con decimales)
div	División con Redondeo al entero más próximo

```

a = b * c
a = b + c
a = b div c
  
```

- **Operadores Relacionales:** Se utilizan en declaraciones condicionales (if, select, while, etc.), comprueban la igualdad o desigualdad de una expresión con respecto a la otra, el resultado entregado es TRUE o FALSE y son:

OPERADOR:	OPERACIÓN:
=	Igual que
<>	Diferente de

>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que

```
a = b           ' a es igual que b
a < b + c      ' a es menor que b+c
a >= b*c       ' a es mayor o igual que b*c
```

- **Operadores Lógicos:** Realizan operaciones lógicas y son más utilizados cuando el objetivo es modificar variables a nivel de sus bits y son:

OPERADOR:	OPERACIÓN:
and	Operación Lógica AND
or	Operación Lógica OR
xor	Operación Lógica XOR
not	Operación Lógica NOT
<<	Desplaza un número de bits hacia la izquierda
>>	Desplaza un número de bits hacia la izquierda

```
%1010 and %0011      ' el resultado es 0010
not %0011             ' el resultado es 1100
%1100 >> 2           ' el resultado es 0011
```

- **Operadores Boléanos:** Se utilizan para enlazar expresiones relacionales largas o multi-condicionales, el resultado que entregan es TRUE o FALSE y son:

OPERADOR:	OPERACIÓN:
and	<i>expresión 1 y expresión 2</i>
or	<i>expresión 1 o expresión 2</i>
not	Negar <i>expresión</i>

```
(a < b + c) and (b = c) ' a es mayor que b+c y b es igual que c
(a < b + c) or (b = c)  ' a es mayor que b+c o b es igual que c
```

➤ Estructuras de Control Condicionales e Iterativas

Las estructuras de control son aquellas que nos permiten tomar decisiones dentro de un programa o cambiar el avance del mismo. Básicamente existen dos tipos de estructuras: las condicionales y las iterativas, las cuales a su vez se subdividen en otros que se explican a continuación:

- **Estructura condicional "if":**

```
if expresión then
    declaraciones_1
else
    declaraciones_2
end if
```

Si la *expresión* o *expresiones_enlazadas* es verdadera entonces se ejecutan el conjunto de *declaraciones_1*, caso contrario se ejecuta el conjunto de *declaraciones_2*. No es obligatorio el bloque "**else**". Ejm:

```
if a > b then
    c = a - b
else
    c = b - a
end if
```

• **Estructura condicional "select":**

```
select case selector
  case valor_1
    declaraciones_1
    ...
  case valor_n
    declaraciones_n
  case else
    declaraciones_auxiliares
end select
```

El selector es una expresión que puede tomar al menos n valores diferentes, para cada uno existe un conjunto de declaraciones que se pueden ejecutar, y en el caso de no ser ninguno de estos n valores, se ejecutan las declaraciones auxiliares, aunque no son obligatorias. Ejm:

```
select case operación
  case "*"
    a = b * c
  case "/"
    a = b / c
  case "+"
    a = b + c
  case "-"
    a = b - c
  case else
    a = 0
end select
```

• **Estructura iterativa "for":**

```
for contador = valor_inicial to valor_final [step valor_paso]
  declaraciones
next contador
```

El conjunto de declaraciones encerradas dentro de un bucle "for" se repite continuamente hasta que la variable llamada contador alcanza el valor final, cada repetición incrementa un cierto número de veces al contador, este incremento está determinado por el parámetro "step". Si no se utiliza este parámetro el incremento es uno. Ejm:

```
s = 2
for i = 1 to 2
  s = s * 2 ' la operación resultante es 2 al cubo
next i
```

• **Estructura iterativa "while":**

```
while expresión
  declaraciones
wend
```

El conjunto de declaraciones encerradas dentro de un bucle "while" se repite continuamente siempre y cuando la expresión evaluada sea verdadera (TRUE). Ejm:

```
i = 1
s = 2
while i <= 2
```

```
s = s * 2    ' la operación resultante es 2 al cubo
i = i + 1
wend
```

➤ **Procedimientos y Funciones**

Generalmente conocidas como subrutinas, las funciones y procedimientos son conjuntos de instrucciones encerradas en un bloque, destinadas a realizar algún trabajo específico. En el caso de las funciones estas devuelven un valor, en cambio que los procedimientos no lo hacen. En la mayoría de casos estos bloques requieren parámetros de entrada que pueden ser incluidos al llamar a la función.

Las ventajas principales de utilizar funciones y procedimientos se mencionan a continuación junta con la forma en como se declara cada una:

- Evitan el desperdicio de memoria del microcontrolador, ya que para utilizar dicho conjunto de instrucciones solo se necesita llamar a la rutina en vez de copiar todo el código en cada punto que se requiera.
- Permiten obtener un programa con menos líneas, más ordenado y fácil de entender

• **Funciones:**

```
sub function nombre_función(lista_parametros) as tipo_resultado
    [ declaraciones_locales ]
    cuerpo_de_la_función
end sub
```

La lista de parámetros, define el número y el tipo de datos de entrada que se deberán esperar siempre que se llame a la función, las declaraciones locales son las variables internas que utiliza la función y que no son vistas por el programa principal. Siempre que se llama a una función, inmediatamente se crea una variable llamada "result" cuyo tipo depende del tipo especificado al declarar la función, es precisamente esta variable la que la función devolverá al final. Ejm:

```
sub function potencia(dim x, n as byte) as longint
    dim i as byte    ' declaración local de variable
    i = 0
    result = 1       ' "result" no necesita declaración
    if n > 0 then
        for i = 1 to n
            result = result*x
        next i
    end if
end sub
```

Una vez que la función ha sido declarada esta puede ser utilizada dentro del programa principal como en el siguiente ejemplo:

```
valor = potencia (2,3) ' calcula 2 al cubo
valor = potencia (3,5) ' calcula 3 a la quinta potencia
```

• **Procedimientos:**

```
sub procedure nombre_procedimiento(lista_parametros)
    [ declaraciones_locales ]
    cuerpo_de_la_función
end sub
```

COLEGIO TÉCNICO SALESIANO
Curso de Microcontroladores de la Familia PIC16 de MICROCHIP

El siguiente procedimiento se encarga de prender y apagar “n” número de veces a unos leds conectados en el puerto B del microcontrolador cada 2 segundo

```
sub procedure led_on_off(dim n as byte)
  dim i as byte
  for i = 1 to n
    PORTB = %11111111 ' Enciende LED's
    Delay_ms(1000)    ' Espera 1seg (1000ms)
    PORTB = %00000000 ' Apaga LED's
    Delay_ms(1000)    ' Espera 1seg (1000ms)
  next i
end sub
```

Una vez que el procedimiento ha sido declarado, este puede ser utilizada dentro del programa principal como en el siguiente ejemplo:

```
Led_on_off (10) ' Enciende y apaga los LED's 10 veces
```

Un tipo especial de procedimiento que requiere especial atención, son las interrupciones, en este caso se trata de un procedimiento que será llamado cada vez que se genere una interrupción por parte de algún periférico específico (Ver capítulo 5), todo procedimiento destinado a atender interrupciones debe llamarse “interrupt”. A continuación se muestra la forma de declara una interrupción y un ejemplo que atiende la interrupción por el TIMER 0 o el TIMER 1:

```
sub procedure interrupt
  [ declaraciones_locales ]
  cuerpo_de_la_función
end sub

sub procedure interrupt
  if TestBit(INTCON, TMR0IF) = 1 then ' Fue el TIMER 0?
    PORTB = %11111111 ' Enciende LED's del puerto B
    ClearBit(INTCON, TMR0F) ' Limpia Bandera antes de salir
  else
    if TestBit(PIR1, TMR1IF) = 1 then ' Fue el TIMER 1?
      PORTB = %00000000 ' Apaga LED's del puerto B
      ClearBit(PIR1, TMR1IF) ' Limpia Bandera antes de salir
    end if
  end if
end sub
```

Observe que parte esencial de una subrutina de interrupción, es el chequeo de banderas mediante la función “TestBit”, esto nos ayuda a saber cual fue el causante de la misma, esto solo es necesario cuando existe más de un dispositivo programado para generar una interrupción.

Recuerde que antes de salir de la interrupción se debe limpiar la bandera del dispositivo que generó la interrupción, caso contrario se volvería a entrar a la interrupción inmediatamente después de haber salido de ella.

➤ Ejemplos

Para finalizar este capítulo, se presenta un programa que reúne todos los puntos estudiados, además también da a conocer algunas funciones que más adelante serán estudiadas y utilizadas durante el desarrollo de las prácticas:

```
program DEMO
```

COLEGIO TÉCNICO SALESIANO
Curso de Microcontroladores de la Familia PIC16 de MICROCHIP

```
*****
'* Declaraciones Globales:
*****

' Declaración de Constantes
const evento as byte = 1

' Declaración de Variables
dim aa, bb, cc, i, n as byte

' Declaración de Procedimientos
sub procedure Led_on_off(dim n as byte)

    ' Declaraciones Locales
    dim i as byte
    ' Cuerpo del Procedimiento
    for i = 1 to n
        PORTB = %11111111      ' Enciende LED's
        Delay_ms(1000)      ' Espera 1seg (1000ms)
        PORTB = %00000000      ' Apaga LED's
        Delay_ms(1000)      ' Espera 1seg (1000ms)
    next i

end sub

' Declaración de Funciones
sub function sumar(dim x, y as byte) as byte

    result = x + y

end sub

*****
'* Programa Principal:
*****

main:

n = evento
aa = 10
bb = 5
cc = 0
TRISB = %00000000      ' Todo los pines son salidas

select case n

    case 1
        if aa > bb then
            cc = aa - bb
        else
            cc = bb - aa
        end if
    case 2
        for i = 1 to 2
            cc = cc + aa + bb
        next i
    case 3
        while aa > bb
            aa = aa - 1
        wend
    case 4
```

```
        cc = sumar(aa,bb)
case 5
        Led_on_off(10)
case else
        cc = 0

end select

end.
```

CAPITULO 9

Prácticas

- **Práctica N-1: "LEDS Titilantes"**
- **Práctica N-2: "Secuencias con LEDES v1"**
- **Práctica N-3: "Secuencias con LEDES v2"**
- **Práctica N-5: "Control ON / OFF de LEDES con pulsantes v1"**

- **Práctica N-6: "Control ON / OFF de LEDS con pulsantes v2"**
- **Práctica N-7: "Control ON / OFF de LEDS con TIMER"**
- **Práctica N-8: "Control de DISPLAYS de 7 segmentos v1"**
- **Práctica N-9: "Control de DISPLAYS de 7 segmentos v2"**
- **Práctica N-10: "Control de DISPLAYS de 7 segmentos v3"**
- **Práctica N-11: "Manejo del Teclado Matricial"**
- **Práctica N-12: "Manejo del LCD Inteligente v1"**
- **Práctica N-13: "Manejo del LCD Inteligente v2"**
- **Práctica N-14: "Manejo del Conversor A/D"**
- **Práctica N-15: "Manejo del Módulo PWM"**
- **Práctica N-16: "Manejo del Módulo UART"**