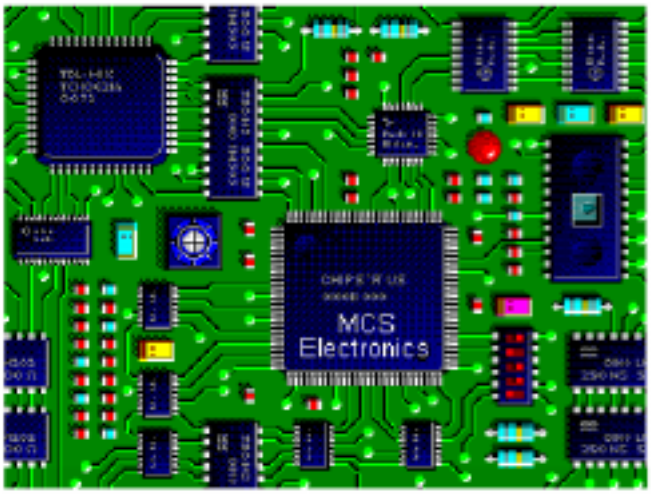
COMPILADOR DE BASIC

BASCOM-8051

J. Mengual - 2000

GUIA DE REFERENCIA DEL LENGUAJE

BASCOM-8051



El autor

Esta documentación ha sido traducida del idioma original en ingles al español por **Juan Mengual**, el único interés en la traducción ha sido el que este programa sea conocido por usuarios de habla hispana para sacar el máximo rendimiento a BASCOM-8051. Algunos textos están en el **idioma original** para una mejor comprensión del usuario técnico.
Los ejemplos en BASIC están en el idioma original salvo los comentarios.

Agradecer también a Giovanni Pedruzzi de la empresa Contrive que ha traducido la documentación original en italiano y que también me ha servido de referencia.

Mi deseo es facilitar también el camino a los usuarios que empiezan en el mundo de los microcontroladores, que de alguna manera también a ellos y sobre todo a los jóvenes, sea el mundo de los microcontroladores lo más ameno y divertido posible, si esto se consigue, mi trabajo no habrá caido en saco roto.

INICIO

GUIA DE REFERENCIA DEL LENGUAJE BASCOM-8051

Actualizado a la versión 1.20

INTRODUCCION

BASCOM 8051

BASCOM 8051 es un excelente compilador de BASIC para la familia de microprocesadores 8051.

En este manual encontrará el técnico programador toda la ayuda que necesita para desarrollar los proyectos basados en el micro 8051, todos los ejemplos son orientativos, cuando se adquiere el compilador, éste ya lleva muchos ejemplos que funcionan y que sirven de gran ayuda.

Es aconsejable y muy recomendable el leer las características del microprocesador que vaya a utilizar, a nivel de SFR y de hardware, es lo mejor para sacar el máximo rendimiento al microprocesador.

Antes de empezar un proyecto, debe de tener claro qué es lo que necesita y que microprocesador es el adecuado, un microprocesador con opciones sobrantes seguro que será mas caro que el necesario, por contra si le faltan puertos por ejemplo, no le servirá el proyecto.

Este manual tendrá errores como le ocurre a la mayoría de los manuales que se hacen por primera vez, no obstante, es mi deseo el corregirlo hasta que éstos queden resueltos.

A los que no les gusta leer, pueden buscar directamente las instrucciones como si de un manual de ayuda se tratara, de echo es igual que la ayuda del programa pero en formato manual, al final del manual hay un índice con las instrucciones y el número de página, también pueden buscar la página directamente a través del título.

Si alguien quiere enviarme algún comentario sobre este manual ó errores que encuentre, lo puede hacer en:

jmengual1@airtel.net

Julio-2000

FUNDAMENTOS DEL LENGUAJE

El juego de caracteres de **BASCOM 8051**

Mediante el juego de caracteres de BASCOM se forman las etiquetas, las palabras claves, variables y operadores.

Éstos combinan para formar instrucciones que constituyen un programa a su vez.

Este capítulo describe el juego de caracteres usado y la forma en que lo utiliza BASCOM para formar las líneas de los programas. En particular, lo siguiente:

- Los caracteres específicos en el juego de caracteres y los significados especiales de algunos caracteres.
- El formato de una línea en un programa de BASCOM.
- Las etiquetas de la línea.
- La longitud de línea de programa.

Juego de caracteres

El juego de caracteres BÁSICO de BASCOM consisten en caracteres alfabéticos, caracteres numéricos, y caracteres especiales.

Los caracteres alfabéticos en BASCOM son las letras mayúsculas (A-Z) y letras minúsculas (az) del alfabeto.

Los caracteres numéricos en BASCOM son dígitos del 0-9. Las letras pueden usarse como partes de números del formato hexadecimal. Los caracteres siguientes tienen los significados especiales en las instrucciones y expresiones de BASCOM:

Carácter	Nombre
ENTER	Termina una entrada de línea
	Blanco (o espacio)
'	Simple comilla (apóstrofe)
*	Asterisco (símbolo de multiplicación)
+	Signo de mas (suma)
,	Coma
-	Signo de menos (resta)
.	Periodo (punto decimal)
/	Slash (símbolo de división) se maneja como \
:	Colon (: dos puntos)
"	Doble comillas
;	Punto y coma
<	menor que
=	signo de igual (símbolo de asignación u operador relacional)
>	Mayor que
\	Backslash (integer/word símbolo de división)

FUNDAMENTOS DEL LENGUAJE

Líneas del programa BASCOM

Las líneas de programas en BASCOM tienen la sintaxis siguiente:

[[línea-identificador]] [[instrucción]] [[:instrucción]] ... [[comentario]]

Usando identificadores de línea

BASCOM soporta un tipo de identificador de línea; etiquetas alfanuméricas:

Una etiqueta de línea alfabética puede ser cualquier combinación de 1 a 32 letras y dígitos, iniciando con una letra y finalizando con dos puntos (**:**).

No se permiten las palabras claves (reservadas) de BASCOM. Lo siguiente son etiquetas de línea alfanuméricas válidas:

Alpha:
ScreenSUB:
Test3A:

El tipo de letra no es significante. Las etiquetas de línea siguientes son equivalentes:

alpha:
Alpha:
ALPHA:

Las etiquetas de la línea pueden empezar en cualquier columna, con tal de que ellas sean los primeros caracteres de la línea.
Los espacios en blanco no se permiten entre una etiqueta alfabética y los dos puntos que lo sigue. Una línea puede tener sólo una etiqueta.

Las instrucciones de BASCOM

Una instrucción de BASCOM es " ejecutable " o " no ejecutable ".
Una instrucción ejecutable adelanta el flujo de una lógica de los programas diciendo el programa qué tiene que hacer luego.

Las instrucciones no ejecutables realizan las tareas como asignar el almacenamiento para las variables, declarando y definiendo los tipos de variables.

Las siguientes instrucciones de BASCOM son ejemplo de instrucciones no ejecutables:

- REM or (inicio de un comentario)
- DIM

FUNDAMENTOS DEL LENGUAJE

El "comentario" es una instrucción no ejecutable usada para clarificar los programas como ejemplo; operaciones matemáticas y propósitos del programa. El comentario es introducido después de una instrucción REM ó el carácter (') que también es válido.
Las siguientes líneas son equivalentes:

```
PRINT " Hola Mundo"  : REM Print saludo inicial.
PRINT " Hola Mundo"   ' Print saludo inicial.
```

Más de una instrucción de BASCOM puede ponerse en una línea, pero los dos puntos (**:**) deben de separar las instrucciones, como el siguiente ejemplo:

```
FOR I = 1 TO 5 : PRINT " Primera parte."  : NEXT I
```

Longitud de la línea en BASCOM
Si usted usa para la elaboración de sus programas el editor de BASCOM, éste no se limita a cualquier longitud de la línea, aunque se aconsejaba acortar las líneas a 80 caracteres para una mejor claridad del programa.

Tipo de datos
Cada variable en BASCOM tiene un tipo de datos que determina lo que puede guardarse en la variable. La próxima sección resume los tipos de los datos elementales.

Tipos de datos elementales:

- **Bit (1/8 byte)**
- **Byte (1 byte).**

Bytes, son almacenados sin signo, son números de 8 bits con un rango entre 0 y 255.

- **Integer (dos bytes).**

Integers, se guardan con signo y tienen una longitud de 16 bits formado números con un valor entre -32,768 y +32,767.

- **Word (dos bytes).**

Words, se guardan sin signo y tienen una longitud de 16 bits formado números con un valor entre 0 y 65535.

- **Long (cuatro bytes).**

FUNDAMENTOS DEL LENGUAJE

Longs, se guardan con signo y tienen una longitud de 32 bits formando números con un valor entre -2147483648 y 2147483647.

· **Single**

Singles, se guardan con signo y tienen una longitud de 32 bits.

· String (hasta 254 bytes).

Strings (Cadenas), las cadenas son almacenadas en bytes y el último byte ha de ser siempre 0.
Una cadena dimensionada con una longitud de 10 bytes ocupa 11 bytes.

Pueden guardarse las variables internas (por defecto) o externas.

Variables
Una variable es un nombre que se refiere a un objeto, un número particular.

A una variable numérica, puede asignarse sólo un valor numérico (cualquier integer, byte, long, single o bit).

La lista siguiente muestra algunos ejemplos de asignaciones a variables:

· Valor constante:

A = 5
C = 1.1

· El valor de otra variable numérica:

abc = def
k = g

· El valor obtenido en combinación con otras variables, constantes y operadores:

Temp = a + 5
Temp = C + 5

Nombre de variables
En BASCOM el nombre de una variable puede contener hasta 32 caracteres.

Los caracteres permitidos en los nombres de variables pueden ser letras y números. El primer carácter en el nombre de la variable ha de ser una letra.

FUNDAMENTOS DEL LENGUAJE

El nombre de una variable no puede ser una palabra reservada, pero se permiten las palabras reservadas incluidas.

Por ejemplo, la instrucción siguiente es ilegal porque AND es una palabra reservada.

AND = 8

En cambio, la siguiente instrucción es legal:

ToAND = 8

Las palabras reservadas de BASCOM incluye todos los comandos, instrucciones, nombre de funciones, registros internos y nombres de operadores.
(ver en índice de palabras reservadas de BASCOM para una completa lista de palabras reservadas).

Si especifica un número hexadecimal o binario, tiene que anteponer el prefijo **&H** ó **&B**, por ejemplo:

a = &HA , a = &B1010 y a = 10 es lo mismo.

Antes de asignar una variable, esta debe de ser declarada mediante la instrucción DIM, sino el compilador producirá un error de declaración, ejemplo:

Dim b1 As Bit, I As Integer, k As Byte , s As String * 10

Usted también puede usar DEFINT, DEFBIT, DEFBYTE y/o DEFWORD.
Por ejemplo DEFINT **c** le dice al compilador que todas las variables que no son dimensionadas y que están empezando con el carácter **c** son del tipo del Entero.

Expresiones y Operadores
Este capítulo discute cómo combinar, modificar, comparar, o conseguir la información sobre las expresiones usando los operadores disponible en BASCOM.

Cuando usted hace un cálculo, está usando las expresiones y operadores.

Este capítulo describe cómo se forman las expresiones y concluye describiendo el tipo siguiente de operadores:

- Operadores **aritméticos**, realiza los cálculos.
- Operadores **relacional**, compara los valores numéricos.
- Operadores **lógicos**, prueba las condiciones o manipula los bits individuales.
- Operadores **funcionales**, complementa los operadores simples.

FUNDAMENTOS DEL LENGUAJE

Expresiones y Operadores
Una expresión puede ser una constante numérica, una variable, o un solo valor obtenido combinando constantes, variables, y otras expresiones con operadores.

Los operadores realizan funciones matemáticas ú operaciones lógicas en valores. Los operadores previstos en BASCOM se dividen en cuatro categorías, son las siguientes:

1	**Aritméticos**
2	**Relacional**
3	**Lógicos**
4	**Funcionales**

Aritméticos
Los operadores aritméticos son +, - , * y \.

- Integer (**entero**)

Integer la división se denota por el backslash (\).
Ejemplo: Z = X \ Y

- Módulo Aritmético

El modulo aritmético se denota por el operador del módulo MOD.
El módulo aritmético proporciona el resto, en lugar del cociente, de una división del entero.
Ejemplo: X = 10 \ 4 : el resto = 10 MOD 4

- Overflow and división by zero (Rebosamiento y división por cero)

División por cero, produces un error.

En este momento no hay ningún mensaje, tiene de asegurarse de que no le ocurran mensajes de este tipo.

Operadores Relacionales
Se usan los operadores relacionales para comparar dos valores como los mostrados en la tabla adjunta.

El resultado puede usarse para tomar una decisión con respecto al flujo del programa.

FUNDAMENTOS DEL LENGUAJE

Operador	Relación a comprobar	Expresión
=	Igual	X = Y
<>	Diferente	X <> Y
<	Menor que	X < Y
>	Mayor que	X > Y
<=	Menor o igual que	X <= Y
>=	Mayor o igual que	X >= Y

Operadores Lógicos
Los operadores lógicos realizan las pruebas en las relaciones, manipulaciones de bit, u operadores Booleanos.

Hay cuatro operadores en BASCOM, son:

Operador	Significado
NOT	Complemento lógico
AND	Conjunción
OR	Disyunción
XOR	Exclusiva or

Es posible usar los operadores lógicos para probar los bytes extrayendo el valor de un bit en particular.

Por ejemplo el operador **Y** puede usarse para enmascarar todos menos uno de los bits de un byte de estado, mientras **OR** puede usarse para unir dos bytes para crear un valor binario particular.

Ejemplo:
```
A = 63 And 19
PRINT A

A = 10 Or 9
PRINT A
```

Output
```
16
11
```

El punto flotante
Un nuevo tipo de dato es añadido a BASCOM : el **single**.
Números **Single** conforme al IEEE, punto flotante binario normal.

Soporta un exponente de 8 bits y mantisa de 24 bits.

FUNDAMENTOS DEL LENGUAJE

El formato de los cuatro bytes usado se muestra a continuación:

31 30_____23 22_____0
s exponente mantissa

El exponente es parcial por 128. Los exponentes por encima de 128 son positivos y debajo negativos. El bit de signo es 0 para los números positivos y 1 para el negativo. El mantisa se guarda en el momento oculto que normalizó el formato para que puedan obtenerse 24 bits de precisión.

Todas la operaciones matemáticas son soportadas por el formato **single**

Usted también puede convertir un **single** a un entero o palabra o viceversa:

```
Dim I As Integer, S As Single
S = 100.1              'asignamos a la variable S en formato single
I = S                  'Ahora convertimos el single an integer
```

Eche una mirada al ejemplo de single.bas para más información.

Arrays (Matrices)
Las matrices también son una nueva extensión a BASCOM.
Una matriz es un juego de elementos secuencialmente puestos en un índice que tienen el mismo tipo.

Cada elemento de una matriz tiene un único número del índice que lo identifica.
Los cambios realizados a un elemento de una matriz no afecta los otros elementos.

El índice es una constante numérica, un byte, un entero o palabra. Esto significa que una matriz puede contener 65535 elementos como máximo. El valor mínimo es 1 y no cero como en QB.

Las matrices, pueden usarse donde se espera una variable normal pero hay unas excepciones.

Estas excepciones se muestran en los temas de ayuda.
Note que no hay ninguna matriz en forma de BIT en BASCOM-8051.
Ejemplo:

```
Dim a(10) As Byte ' haga una serie nombrada a, con 10 elementos (1 a 10)
Dim c As Integer
For C = 1 To 10
  a(c) = c              ' asigne el elemento de la serie
  Print a(c)            ' imprime los valores de la serie
Next
End
```

FUNDAMENTOS DEL LENGUAJE

Strings (cadenas)

Las cadenas pueden disponer de 254 caracteres de longitud en BASCOM.
Para ahorrar la memoria usted debe de especificar cuanta longitud necesita en cada cadena con la instrucción DIM.

Dim S **As String** * 10

Esto reservará el espacio para la cadena S con una longitud de 10 bytes.
Realmente la longitud es de 11 bytes, porque un byte nul(0) se usa para terminar la cadena.

Usted puede encadenar la cadena con el signo **+**.

Dim S **As String** * 10 , Z **As String** * 10

S = "test"
Z = S + "abc"

En QB usted puede asignar una cadena con un valor y puede agregar la cadena original (o una parte de ella) también:.

S = "test"
S = "a" + s

El resultado es la cadena "atest"

En BASCOM esto no es posible porque esto requiere una copia de la cadena.
En BASCOM la cadena S se asigna con " a " y en ese momento la cadena original S se destruye. Así que usted debe hacer una copia de la cadena durante el evento si usted necesita esta funcionalidad.

1WRESET,1WREAD,1WWRITE

Acción
Estas rutinas se usan para comunicar con los dispositivos de 1 Wire de Dallas Semiconductors.

Sintaxis
1WRESET
1WWRITE var1
var2 **= 1WREAD()**

Nota

1WRESET	Reset del bus 1WIRE. En caso de error, la variable ERR tendrá el valor 1.
1WWRITE var1	Envía al bus el valor de la var1.
var2 = 1WREAD()	Lee un byte del bus y lo asocia a la var2.

var1 : Byte, Integer, Word, Long, Constant.
var2 : Byte, Integer, Word, Long.

Ejemplo:
```
'-----------------------------------------------------
' 1WIRE.BAS
' muestra el uso de 1wreset, 1wwrite y 1wread()
' pullup de 4K7 necesario en P.1 a VCC
' patilla serie del DS2401 conectada a P1.1
'-----------------------------------------------------
Config 1wire = P1.1     'usa este pin
Dim Ar(8) As Byte , A As Byte , I As Byte
1wreset                 'reset del bus
Print Err               'Print error 1 en caso de error
1wwrite &H33            'comando lectura ROM
For I = 1 To 8
      Ar(I) = 1wread()  'lee byte
Next
For I = 1 To 8
      Printhex Ar(I);   'print Output
Next
Print                   'linefeed
End
```

$ASM - $END ASM

Acción
Inicio de un bloque en lenguaje assembler.

Sintaxis
$ASM

Nota
$ASM debe ser implementado conjuntamente con **$END ASM** para incorporar un bloque en código assembler dentro de un programa BASIC.

Ejemplo:
```
Dim c As Byte
$ASM
Mov r0,#{C}        ;dirección de C
Mov a,#1
Mov @r0,a          ;almacena 1 en var C
$END ASM
Print c
End
```

$INCLUDE

Acción
Incorpora un archivo ASCII en el programa a partir de la posición actual.

Sintaxis
$INCLUDE file

Nota

file	Nombre del archivo ASCII que debe contener declaraciones de BASCOM válidas. Esta opción puede usarse si hace uso de las mismas rutinas en muchos programas, puede escribir módulos y e incluirlos en sus programas. Si hay cambios, sólo necesitará cambiar el archivo del módulo y no todos sus programas de BASCOM. ¡El archivo debe de ser en formato ASCII!

Ejemplo:
```
'-----------------------------------------------------------
' (c) 1997,1998 MCS Electronics
'-----------------------------------------------------------
' file: INCLUDE.BAS
' demo: $INCLUDE
'-----------------------------------------------------------
Print "INCLUDE.BAS"
$include c:\bascom\123.bas    'incluye el archivo Hello
Print "Back in INCLUDE.BAS"
End
```

$BAUD

Acción
Fuerza al compilador a utilizar un valor específico de baud rate, ignorando lo predefinido en el menú de opciones.

Sintaxis
$BAUD = var

Nota

var	Es el valor (baud rate) que se desea utilizar.

var : Constant.
Esta directiva puede ser útil cuando es necesario seleccionar algunos valores dependiendo de la frecuencia del cristal oscilador (cristal/baud rate), que no está disponible en el menú de las opciones.
La directiva $CRYSTAL siempre se usa en combinación.
En el informe generado en fase de compilación, se puede ver qué proporción del *baud rate* realmente se genera.
El valor de *baud rate* sólo se muestra si existen instrucciones que hacen uso del puerto asíncrono serie de comunicación (RS-232) como por ejemplo: PRINTA, INPUT, etc.

Ver también
$CRYSTAL

Ejemplo:
```
$BAUD = 2400
$CRYSTAL = 12000000     'cristal 12 MHz
PRINT "Hola"
END
```

$CRYSTAL

Acción
Fuerza al compilado a utilizar un valor específico del valor del cristal del oscilador, ignorando el predefinido en el menú de opciones.

Sintaxis
$CRYSTAL = var

Nota

var	Frecuencia del cristal.

var : Constant.
Esta directiva puede ser útil cuando sea necesario seleccionar un valor de la frecuencia del cristal (*crystal/baud rate*) que no está disponible en las opciones del menú.
La directiva $BAUD siempre es usada en combinación.

Ver también
$BAUD

Ejemplo:
```
$BAUD = 2400
$CRYSTAL = 12000000
PRINT "Hola"
END
```

$DEFAULT XRAM

Acción
Directiva para el compilador para forzar la memorización de la variable dimensionada en la RAM externa (XRAM).

Sintaxis
$DEFAULT XRAM

Nota
Cuando se usan muchas variables en la memoria RAM externa, en vez de especificar cada vez la opción XRAM, es posible instruir al compilador que la memoria por defecto es la XRAM.
En este caso, si se desea memorizar una variable en la RAM interna, será posible indicando la opción IRAM.

Ejemplo:
```
$DEFAULT XRAM
Dim X As Integer         'memorizado en XRAM
Dim Z As IRAM Integer    'memorizado en IRAM
```

$IRAMSTART

Acción
Directiva para el compilador conteniendo el inicio de partida del la memoria RAM interna.

Sintaxis
$IRAMSTART = constant

Nota

constant	Constante con el valor de inicio de la RAM (0-255)

Ver también
$NOINIT $RAMSTART

Ejemplo:
```
$NOINIT
$NOSP
$IRAMSTART = &H60       'primera localización de memoria utilizable
SP = 80
DIM I As Integer
```

$LARGE

Acción
Fuerza al compilador al uso de la instrucción LCALL.

Sintaxis
$LARGE

Nota
Se usa internamente cuando se llama a un subprograma con la declaración de ACALL y se necesita desplazar más de 2048 bytes.
La instrucción de ACALL necesita sólo dos bytes (LCALL necesita tres bytes).
La declaración de ACALL puede dirigirse sólo a rutinas con un desplazamiento máximo de 2048.
Los micros AT89C2051 no tienen ningún problema estas llamadas.
Cuando el código se genera para otro, las llamadas al subprograma, si es mayor de 2048 bytes, el compilador generará un error, cuando esto ocurre, se le dice al compilador que use la declaración de $LARGE, la declaración de LCALL puede dirigirse a los 64K del direccionamiento de la memoria.

Ejemplo:
```
$LARGE      'Ha recibido un error 148 para que necesite esta opción
```

$LCD

Acción
Fuerza al compilador a generar código para la gestión del display LCD a 8 bit, mapeado en memoria y conectado al bus de datos.

Sintaxis
$LCD = [&H] address

Nota

address	La dirección en la cual está disponible el display LCD. Las líneas de datos db0-db7 del display LCD deben ser conectadas al bus de datos D0-D7. La señal RS del display LCD debe ser conectada a la línea A9 del bus de dirección. En sistemas con RAM/ROM externa, puede resultar un inconveniente derivar el display LCD al bus. Sería conveniente la habilitación de una línea para decodificar la dirección.

Ejemplo:
```
$LCD = &HA000      'Escribiendo en esta dirección ponemos la línea E del
                   'display LCD en alto.

LCD "Hola mundo"
```

$MAP

Acción
Genera el informe en el archivo del informe con la dirección en hexadecimal de cada línea fuente.

Sintaxis
$MAP

Nota
Para la depuración puede ser útil saber a que dirección empiezan las líneas fuente.

Ver también

Ejemplo:
$MAP

```
Print "Hola"
Print "Test"
```

Genera el siguiente archivo de informe:

```
Code map
------------------------------------------------------------
Line                          Address(hex)
------------------------------------------------------------
 2                            52
 3                            69
 5                            80
```

$NOBREAK

Acción
Fuerza al compilador a ignorar la instrucción BREAK.

Sintaxis
$NOBREAK

Nota
Con la instrucción BREAK, se genera un código de operación reservado que usa el simulador para hacer una pausa en la simulación.
Cuando se quiere compilar sin estos '*opcodes*' , no tiene que quitar la instrucción BREAK : use la directiva $NOBREAK para lograr lo mismo.

Ver también
BREAK

Ejemplo:
```
$NOBREAK
BREAK        'Esto no será compilado, aquí el simulador no hará pausa
End
```

$NOINIT

Acción
Fuerza al compilador a no seguir alguna inicialización.

Sintaxis
$NOINIT

Nota
BASCOM inicializa el procesador en función del la instrucción utilizada.
Si se desea proveer autónomamente de la inicialización es necesario especificar **$NOINIT**.
En este caso el compilador se limitará a inicializar el '*stack pointer*' (Puntero del stack) y el display LCD (siempre que se usen instrucciones relativas al display LCD).

Ver también
$NOSP

Ejemplo:
```
$NOINIT
.
'aquí va el programa
End
```

$NOSP

Acción
Fuerza al compilador a no inicializar el *stack pointer*.

Sintaxis
$NOSP

Nota
BASCOM inicializa el procesador en función del la instrucción utilizada.
Se desea proveer autónomamente de la inicialización es necesario especificar **$NOINIT**.
En este caso el compilador se limitará a inicializar el '*stack pointer'* (Puntero del stack) y el display LCD (siempre que se usen instrucciones relativas al display LCD).
Con la directiva **$NOSP** no será inicializado el *stack pointer*.

Ver también
$NOINIT

Ejemplo:
```
$NOSP
$NOINIT
End
```

$NONAN

Acción
Directiva del compilador para cambiar ***nan*** (not a number) (no es un número), devuelve 0.0

Sintaxis
$NONAN

Nota
TIMERx TI

Una variable con formato single puede devolver un NAN cuando no está considerada para ser un número.
Con la directiva $NONAN devolverá 0.0.

Ver también

Ejemplo:

$OBJ

Acción
Inclusión de código objeto en formato Intel.

Sintaxis
$OBJ obj

Nota
obj es el código objeto a incluir

Ejemplo:
```
$OBJ D291   'equivalente a SET P1.1
```

$RAMSTART

Acción
Especifica la localización de inicio del la memoria RAM externa.

Sintaxis
$RAMSTART = [&H]address

Nota

address	La dirección en (HEX) de inicio de la memoria de datos (la dirección mas baja que permite el chip de la RAM). Esta opción será tomada por el sistema para hacer uso de la RAM externa.

address : Constant.

Ver también
$RAMSIZE

Ejemplo:
```
$ROMSTART = &H4000
$RAMSTART = 0
$RAMSIZE = &H1000
```

$RAMSIZE

Acción
Especifica la dimensión del la memoria RAM externa.

Sintaxis
$RAMSIZE = [&H] size

Nota
size	Dimensión del la RAM externa.

size : Constant.

Ver también
$RAMSTART

Ejemplo:
```
$ROMSTART = &H4000
$RAMSTART = 0
$RAMSIZE = &H1000
DIM x AS XRAM Byte 'Especifica XRAM para salvar la variable in XRAM
```

$ROMSTART

Acción
Directiva pera el compilador conteniendo la dirección de la memoria ROM.

Sintaxis
$ROMSTART = [&H] address

Nota

address	La dirección (HEX) de inicio de la memoria conteniendo el código. Cuando no se especifica con la directiva se entiende por defecto=0. Esta opción puede ser empleada cuando se desea testear el código en RAM. El código debe ser cargado en una dirección especifica y llamado desde un programa monitor residente. El programa monitor debe proveer la relocalización de las interrupciones a las direcciones correctas. Cuando se especifica $ROMSTART = &H4000 el programa monitor debe efectuar un LJMP. La dirección 3 debe corresponder a &H4003. En caso contrario, no pueden manejarse las interrupciones correctamente. Esto depende del programa monitor.

Ver también
$RAMSTART

Ejemplo:
```
$ROMSTART = &H4000 'ROM habilitada a partir de 4000 hex
```

$REGFILE

Acción
Le dice al compilador que use el archivo del registro especificado.

Sintaxis
$REGFILE = "file"

Nota

file	El nombre del archivo a usar

La instrucción **$REGFILE** debe ponerse antes de cualquier otra instrucción ejecutable o directivas del compilador.

Ver también

Ejemplo:
```
'Los comentarios no causan problemas antes de la instrucción $REGFILE
$REFILE = "8052.DAT"     'usa el archivo 8052.DAT
```

$SERIALINPUT

Acción
Especifica la redirección para la entrada de la señal serie.

Sintaxis
$SERIALINPUT = label

Nota

label	Con la redirección del comando INPUT, es posible utilizar rutinas específicas. De ésta manera puede usar otros dispositivos como dispositivos de INPUT. Note que la instrucción INPUT se termina cuando un código del RETORNO (13) se recibe.

Ver también
$SERIALOUTPUT

Ejemplo:
```
$SERIALINPUT = Mi_input
'aqui va el programa
.
END


!mi_input:
;efectúe aquí las operaciones que necesite
.
mov a, sbuf ;mueve en el acc la señal del buffer serie
ret
```

$SERIALINPUT2LCD

Acción
Esta directiva permite el reenviar al display LCD toda la entrada serie en vez de hacer eco al puerto serie.

Sintaxis
$SERIALINPUT2LCD

Nota
Puede escribir en las entradas de costumbre o drivers de salida con las declaraciones $SERIALINPUT y $SERIALOUTPUT, el $SERIALINPUT2LCD es habilitado cuando se usan visualizadores LCD.

Ver también
$SERIALINPUT , $SERIALOUTPUT

Ejemplo:
```
$SERIALINPUT2LCD
Dim v As Byte
CLS
INPUT "Number ", v      'Esto se visualizará en el display LCD
```

$SERIALOUTPUT

Acción
Especifica que salida serie debe ser redireccionada.

Sintaxis
$SERIALOUTPUT = label

Nota

label	El nombre de la rutina en asembler que debe ser llamada cuando un carácter es enviado al bufer del puerto serie (SBUF). El carácter es puesto en el acumulador ACC. Con la redirección de la instrucción PRINT o cualquier otro comando todas las salidas del puerto serie será posible hacer uso en rutinas personalizadas. De este modo es posible utilizar cualquier otro dispositivo como destinatario de la salida.

Ejemplo:
```
$SERIALOUTPUT = MyOutput
'Aquí va el programa
.
END

!myoutput:
; efectuar aquí las operaciones necesarias
.
mov sbuf, a ;buffer de salida serie(por defecto)
ret
```

$SIM

Acción
Genera código sin bucles de espera para el uso con el simulador.

Sintaxis
$SIM

Nota
Cuando ejecuta la instrucción WAIT en el simulador, el tiempo de espera es excesivo, puede quitar estas variables, otra alternativa es la directiva $SIM
Es absolutamente necesario remover la instrucción $SIM cuando se efectúe la compilación definitiva para transferir el código al chip (EPROM)

Ver también
-

Ejemplo:
```
$SIM          'no se compila WAIT y WAITMS
WAIT 2        'el simulador ahora es más veloz
```

ABS()

Acción
Devuelve el valor absoluto de una variable numérica

Sintaxis
var **= ABS(**var2**)**

Nota

var	Variable a asignar el valor absoluto de var2.
Var2	Variable original desde la cual extraerá el valor absoluto.

var : Byte, Integer, Word, Long.
var2 : Integer, Long.

EI valor absoluto de un número es siempre positivo.

Ver también

-Diferencia con QB
No es posible hacer uso de constantes numéricas porque el valor absoluto es obvio para las mismas.
No es posible efectuar operaciones con *Singles*.

Ejemplo:
```
Dim a As Integer, c As Integer
a = -1000
c = Abs(a)
Print c
End
```

Output
1000

ALIAS

Acción
Indica que la variable puede ser referenciada con otro nombre.

Sintaxis
newvar **ALIAS** oldvar

Nota

oldvar	Nombre de la variable, ejemplo: P1.1
newvar	Nuevo nombre de la variable, ejemplo: Salida

Definir un nombre alternativo a los pins del puerto ayuda a ser más comprensible la programación.

Ver también
CONST

Ejemplo:
```
Salida ALIAS P1.1 'Ahora P1.1 puede ser reclamada como variable Salida
SET Salida        'produce el mismo efecto que SET P1.1
END
```

ASC()

Acción
Convierte un cadena en su correspondiente valor ASCII.

Sintaxis
var = ASC(string**)**

Nota

var	Variable de destino del la conversión.
String	Cadena, variable o constante de la cual obtendrá el valor ASCII.

var : Byte, Integer, Word, Long.
string : String, Constant.

En cualquier caso, la conversión se efectuará sólo en el primer carácter de la cadena. Si la cadena resultase vacía el valor de la conversión será cero.

Ver también
CHR()

Ejemplo:
```
Dim a As Byte, s As String * 10
s = ABC
a = Asc(s)
Print a
End
```

Output
65

BCD()

Acción
Convierte una variable en el correspondiente valor en BCD.

Sintaxis
PRINT BCD(var)
LCD BCD(var)

Nota

var	Variable a convertir.

var1 : Byte, Integer, Word, Long, Constant.

En caso de utilizar dispositivos *real time clock* I2C con valores almacenados en BCD, ésta función permite ver los valores correctamente.
BCD() visualizará un valor en formato *trailing* cero, ejemplo: 0012
La función BCD() se destina al uso en combinación con la instrucción PRINT/LCD.
Use la función MAKEBCD para convertir variables.

Ver también
MAKEBCD , MAKEDEC

Ejemplo:
```
Dim a As Byte
a = 65
LCD a
Lowerline
LCD BCD(a)
End
```

BITWAIT

Acción
Espera hasta que un bit sea *set* o *reset*

Sintaxis
BITWAIT x **SET/RESET**

Nota

x	Variable del tipo Bit o registro interno (ej. P1.x), para x cualquier valor de 0÷7.

Utilizando variables de Bit, asegurar que puedan ser SET/RESET (SET= pone el bit a 1, RESET= pone el bit a 0) por software para evitar estados indefinidos. Cuando se usan registros internos que pueden ser SET/RESET por hardware como P1.0 esto no hace falta.

Ver también
-

Ejemplo
```
Dim a As Bit
BITWAIT a , SET          'Espera hasta que el bit es puesto a 1.
BITWAIT P1.7, RESET      'Espera hasta que el bit 7 del Por 1 es 0.
End
```

ASM
```
BITWAIT P1.0 , SET       'produce:
Jnb h'91,*+0

BITWAIT P1.0 , RESET     'produce:
Jb h'91,*+0
```

BREAK

Acción
Genera una pausa en la ejecución del simulador.

Sintaxis
BREAK

Nota
Es posible definir breakpoints (Puntos de ruptura del programa) en el simulador, mas es posible insertar breakpoints en el programa por medio de la instrucción BREAK. Asegurarse de eliminar la instrucción BREAK al término de la operación de depuración del programa ó utilizar el metacomando $NOBREAK.

El *opcode* (Código de operación) reservado usado es A5.

Ver también
$NOBREAK

Ejemplo:
```
PRINT "Hola"
BREAK             'el simulador hará una pausa ahora
.
.
.
End
```

CALL

Acción
Llamar y ejecuta una subrutina.

Sintaxis
CALL Test [(var1, var-n**)]**

Nota

var1	Cualquier variable o constante BASCOM.
var-n	Cualquier variable o constante BASCOM
Test	Nombre del la subrutina. En este caso Test

Con la instrucción CALL es posible llamar un procedimiento o una subrutina.
En la llamada, pueden pasarse hasta 10 parámetros, pero también puede llamar un subprograma sin parámetros.
Por ejemplo : **Call Test2**

La declaración de la llamada le permite que lleve a cabo sus propias instrucciones personalizadas.

No es obligatorio especificar la instrucción CALL :
Test2 en este caso llama a la subrutina test2.

Cuando no se proporciona la declaración de la LLAMADA, se debe omitir el paréntesis.
Para que Llama Routine(x,y,z) debe escribirse como x,y,x Rutinario

Si no proporciona la instrucción CALL, debe ser omitido el paréntesis.
Para Call Routine(x,y,z) debe de escribirse como Routine x,y,x

Ver también
DECLARE, SUB

Ejemplo:
```
Dim a As Byte, b As Byte
Declare Sub Test(b1 as byte)
a = 65
Call test (a)           'Llama a test pasando el parámetro A.
test a                  'otro modo de efectuar la llamada.
End

SUB Test(b1 as byte)    'usa la misma variable en una declaración.
LCD b                   'la envía al display LCD.
Lowerline
LCD BCD(b1)
End SUB
```

CHR()

Acción
Convierte una variable de tipo Byte, Integer/Word ó una constante en un carácter.

Sintaxis
PRINT CHR(var**)**
s = CHR(var**)**

Nota

var	Variable tipo Byte, Integer/Word o constante numérica.
s	Variable de tipo String.

Cuando se quiere imprimir un carácter a la pantalla o en el visualizador LCD, debe de convertirlo con la función CHR ()

Ver también
ASC()

Ejemplo:
```
Dim a As Byte
a = 65
LCD a
Lowerline
LCDHEX a
LCD Chr(a)
End
```

CLS

Acción
Limpia el display LCD y posiciona el cursor al inicio (primera línea, primera columna), (CLS= Clear Screen).

Sintaxis
CLS

Nota
Cuando se limpia el display LCD, la memoria CG-RAM que contiene los caracteres especiales no se cancelan.

Ver también
$LCD , LCD

Ejemplo:
```
Cls
LCD " Hola"
End
```

CONST

Acción
Declara una constante simbólica.

Sintaxis
DIM symbol AS CONST value

Nota

symbol	Nombre del símbolo.
Value	Valor asignado al símbolo.

La asignación de una constante no requiere espacio de la memoria de programa. En fase de compilación, todas las constantes serán sustituidas con el valor asignado.

Ver también
DIM

Ejemplo:
```
'-----------------------------------------------------
' (c) 1997,1998 MCS Electronics
' CONST.BAS
'-----------------------------------------------------
Dim A As Const 5              'declara A como costante
Dim B1 As Const &B1001
Waitms A                      'espera 5 milisegundos
Print A
Print B1
End
```

CONFIG

La instrucción CONFIG consiente la configuración de las instrucciones ligadas al hardware.

Seleccione uno de los temas siguientes para aprender más sobre la instrucción específica de CONFIG.

CONFIG TIMER0, TIMER1
CONFIG TIMER2 (para dispositivos compatibles con el 8052)
CONFIG LCD
CONFIG LCDBUS
CONFIG LCDPIN
CONFIG BAUD
CONFIG 1WIRE
CONFIG SDA
CONFIG SCL
CONFIG DEBOUNCE
CONFIG WATCHDOG
CONFIG SPI

CONFIG TIMER0, TIMER1

Acción
Configura el TIMER0 ó TIMER1.

Sintaxis
CONFIG TIMERx = COUNTER/TIMER , GATE=INTERNAL/EXTERNAL , MODE=0/3

Nota

TIMERx	TIMER0 o TIMER1. COUNTER configurará TIMERx como COUNTER y TIMER configurará TIMERx como TIMER. El TIMER utiliza el *clock* (reloj) interno del microprocesador, mientras que COUNTER recibe el clock del exterior.
GATE	INTERNAL o EXTERNAL. Especificando EXTERNAL se habilita el control GATE a través de la entrada INT.
MODE	Time/counter modalidad 0÷3. Para mayor detalle, consultar la sección del hardware.

Cuando CONFIG TIMER0 = COUNTER, GATE = INTERNAL, MODE=2 configurará TIMER0 como CONTADOR sin control externo *gate* en modalidad 2 (auto recarga).

Durante la operación de configuración el timer/counter se ponen en stop, cuando deba inicializarlo, es necesario reactivarlo con la instrucción específica START TIMERx.

Ver la instrucción adicional disponible para otros microprocesadores cuando use la instrucción CONFIG.

Ejemplo:
```
CONFIG TIMER0=COUNTER, MODE=1, GATE=INTERNAL
COUNTER0 = 0                    'reset del contador 0
START COUNTER0                  'habilita el contador
DELAY                           'espera un rato
PRINT COUNTER0                  'imprime el valor del contador
END
```

CONFIG TIMER2

Algunos microprocesadores incorporan temporizadores adicionales: **TIMER2**.

Esta sección describe el TIMER2 para microprocesadores 8032 y compatibles y no se aplica al TIMER2 presente en los microprocesadores 80C535 y otros.
TIMER2 es un timer/counter a 16-bit que puede funcionar en modo temporizador o como contador (**Counter**) TIMER2 cuenta con 3 modalidades de funcionamiento: captura, auto-recarga (cuenta adelante o atrás), y generador de baudios.

Modo capture
En modalidad capture existen dos opciones:
_ timer/counter a 16-bit que al completar la cuenta pone a 1 el bit TF2, que es el bit de overflow (rebosamiento) del TIMER2. Este bit puede ser utilizado para generar una interrupción.
Modalidad Counter :
CONFIG TIMER2 = COUNTER, GATE = INTERNAL, MODE = 1
Modalidad Timer :
CONFIG TIMER2=TIMER, GATE= INTERNAL,MODE =1
_ Como lo anterior, pero con la función adicional que una transición de 1 a 0 sobre la entrada T2EX produce el traslado del contenido de los registros TL2 y TH2 de TIMER2 dentro de los registros de capture RCAP2L e RCAP2H.
Modalidad Counter :
CONFIG TIMER2 = COUNTER, GATE = EXTERNAL, MODE = 1
Modalidad Timer :
CONFIG TIMER2=TIMER,GATE=EXTERNAL,MODE=1
Además una transición de T2EX pone a 1 el bit EXF2 en T2CON. EXF2 como TF2, pueden generar una interrupción.
La rutina de interrupción de TIMER2 puede comprobar los bits TF2 y EXF2 para determinar cual acontecimiento ha causado la interrupción. (en esta modalidad no existe valor de recarga. También cuando un acontecimiento de captura ocurre sobre T2EX el contador continúa la cuenta de las transiciones sobre T2EX o bien los impulsos de reloj del temporizador obtenidos con osc/12).

Modalidad Auto recarga
En modalidad auto-recarga a 16-bit, TIMER2 puede ser configurado como temporizador o contador programable adelante/atrás. La dirección del contaje es determinada por el bit DCEN.
TIMER2 contará hasta &HFFFF alcanzado el fin de la cuenta pondrá a 1 el bit TF2 que es el flag (bandera) de overflow (rebosamiento). Éste produce una recarga de los registros de TIMER2 con el valor a 16-bit presente en RCAP2L y RCAP2H.
Los valores en RCAP2L y RCAP2H se ajustan mediante software.
Modalidad Counter :
CONFIG TIMER2=COUNTER,GATE=INTERNAL,MODE=0
Modalità Timer :
CONFIG TIMER2=COUNTER,GATE=INTERNAL,MODE=0
Cuándo EXEN2=1 un overflow o bien una transición de 1 a 0 sobre la entrada T2EX puede causar el recarga de 16-bit. Esta transición también pone a 1 el bit EXF2.

Si la interrupción de TIMER2 es habilitada puede ser generada cuando TF2 o bien EXF2 son al estado lógico 1.
Modalidad Counter :
CONFIG TIMER2=COUNTER,GATE=EXTERNAL,MODE=0
Modalidad Timer :
CONFIG TIMER2=TIMER,GATE=EXTERNAL,MODE=0
TIMER2 además puede contar adelante o atrás. En esta modalidad el pin T2EX puede ser utilizado para controlar la dirección de la cuenta. Cuando a T2EX es aplicado un estado lógico 1 TIMER2 cuenta hacia adelante (incremento).
El overflow del TIMER2 ocurre al logro de &HFFFF y al pasar a 1 el flag TF2, que puede generar una interrupción si está habilitada. El overflow del temporizador también produce la recarga de los registros TL2 y TH2 con los valores contenidos en RCAP2L y RCAP2H (16-bit).
Modo Counter:
CONFIG TIMER2=COUNTER, GATE=INTERNAL/EXTERNAL, MODE=0,DIRECTION=UP
Modo Timer:
CONFIG TIMER2=COUNTER, GATE=INTERNAL/EXTERNAL, MODE=0,DIRECTION=UP
Un estado lógico 0 aplicado al pin T2EX fuerza el TIMER2 a contar atrás (disminución).
El underflow de TIMER2 ocurrirá cuando TL2 y TH2 asuman el mismo valor presente en RCAP2L y RCAP2H. El underflow de TIMER2 pondrá al estado lógico 1 el flag TF2 y recargará el valor &HFFFF en los registros TL2 y TH2 del temporizador.
Modalidad Counter :
CONFIG TIMER2=COUNTER, GATE=INTERNAL/EXTERNAL, MODE=0,DIRECTION=DOWN
Modalidad Timer :
CONFIG TIMER2=COUNTER, GATE=INTERNAL/EXTERNAL, MODE=0,DIRECTION=DOWN
El flag externo TF2 cambia de estado a cada overflow o bien underflow de TIMER2. El flag EXF2 no genera interrupción en modalidad contador adelante/atrás.

Generador de Baud rate (velocidad en baudios)
Esta modalidad puede ser seleccionada para generar la velocidad en baudios del puerto serie asíncrono, liberando TIMER1 para otras funciones.
CONFIG TIMER2=TIMER,GATE=INTERNAL,MODE=2

Sólo Recepción
Esta modalidad puede ser seleccionada para generar sólo los baudios en modo recepción.
TIMER1 puede ser usado para generar una velocidad en baudios diferente en transmisión.
CONFIG TIMER2=TIMER,GATE=INTERNAL,MODE=3
TIMER1 tendrá que ser predispuesto para este empleo con instrucciones en assembler.

Sólo Trasmisión
Esta modalidad puede ser seleccionada sólo para generar la velocidad (baudios) para la transmisión.
TIMER1 puede ser usado para generar una velocidad (baudios) diferente en recepción.

CONFIG TIMER2=TIMER,GATE=INTERNAL,MODE=4
TIMER1 tendrá que ser predispuesto para este empleo con instrucciones en assembler.

Salida de Clock (reloj)
Algunas variantes de procesadores 8052 tienen la posibilidad de generar sobre P1.0 una señal de reloj con la duración del ciclo igual al 50%.
CONFIG TIMER2=TIMER,MODE=5
La frecuencia producida es igual a (fOSC **/** 4) **/** (65536-CAPTURE)
CAPTURE = valor programado en el registro de captura.

Como determinar qué a causado una interrupción
Analizando el bit T2CON.7 se puede determinar si la interrupción ha sido causada por un overflow.
Analizando el bit T2CON.6 se puede determinar si la interrupción ha sido causada por una transición negativa sobre T2EX.

```
Timer2_ISR:
If T2CON.7 = 1 Then
        Print “overflow del timer”
        Else
        If T2CON.6 = 1 Then
        Print “transición externa”
        End if
End If
Return
```

CONFIG LCD

Acción
Configura el display LCD.

Sintaxis
CONFIG LCD = LCDtype

Nota

LCDtype	El tipo de display LCD utilizado, seleccionando uno de los siguientes: 40 * 4, 16 * 1, 16 * 2, 16 * 4, 16 * 4, 20 * 2 ó 20 * 4. Por defecto se asume 16 * 2 .

Ejemplo:
```
CONFIG LCD = 40 * 4
LCD "Hola"                'visualiza “Hola” sobre el display LCD
FOURTHLINE                'selecciona la línea 4
LCD "4"                   'visualiza 4
END
```

CONFIG LCDBUS

Acción
Configura el bus de datos para el display LCD.

Sintaxis
CONFIG LCDBUS = constant

Nota

constant	4 para modalidad a 4-bits, 8 para modalidad a 8-bits (por defecto)

Utilizar esta instrucción conjuntamente con la directiva $LCD = address.

Cuando se conecta un display LCD al bus de datos, por defecto se considera la utilización de las 8 líneas del bus de datos. Si se selecciona la modalidad de 4 bits, debe ser conectado a las lineas de datos d7÷d4.

Ver también
CONFIG LCD

Ejemplo:
```
$LCD = &H8000                 'dirección de la línea de selección del LCD
Config LCDBUS = 4             'modalidad a 4 bits
LCD "hola"
```

CONFIG LCD PIN

Acción
Ignora las opciones relativas al display LCD para definir las acciones del programa.

Sintaxis
CONFIG LCDPIN , DB4= P1.1, DB5=P1.2, DB6=P1.3, DB7=P1.4, E=P1.5, RS=P1.6

Nota
P1.1 etc. es justo un ejemplo de la sintaxis.

Ver también
CONFIG LCD

Ejemplo:
```
CONFIG LCDPIN ,DB4= P1.1,DB5=P1.2,DB6=P1.3,DB7=P1.4,E=P1.5,RS=P1.6
```

CONFIG BAUD

Acción

Configura el microprocesador para seleccionar el generador interno de *baudrate* (velocidad en baudios para la comunicación serie).
Este generador de baudios está sólo disponible en los 80535, 80537 y chips compatibles.

Sintaxis

CONFIG BAUD = baudrate

Nota

baudrate	Velocidad en baudios a utilizar : 4800 ó 9600

Ejemplo:

```
CONFIG BAUD = 9600      'usa el generador interno para generar 9600 baudios
Print "Hola"
End
```

CONFIG 1WIRE

Acción
Configura el pin que debe ser utilizado con la instrucción 1WIRE.

Sintaxis
CONFIG 1WIRE = pin

Nota

pin	Pin del puerto a utilizar, por ejemplo: P1.0

Ver también
1WRESET , 1WREAD , 1WWRITE

Ejemplo:
```
Config 1WIRE = P1.0             'P1.0 usado por el bus 1-wire
1WRESET                         'reset del bus
```

CONFIG SDA

Acción
Ignora la asignación efectuada en *Option Settings* (desde el programa BASCOM-8051) y selecciona el pin para la línea SDA.

Sintaxis
CONFIG SDA = pin

Nota

pin	Pin del puerto a utilizar por la linea I2C-SDA.

Cuando usa diferentes pins en diferentes proyectos, puede usar esta instrucción para ignorar la opciones del compilador definida para el pin SDA.

De este modo el programa tendrá esplícitamente direccionado el pin utilizado y no será necesario anotarlo separadamente para modificarlo en *Option Settings*.

Ver también
CONFIG SCL

Ejemplo:
```
CONFIG SDA = P3.7              'P3.7 es la línea SDA
```

CONFIG SCL

Acción
Ignora la asignación efectuada en *Option Settings* (desde el programa BASCOM-8051) y selecciona el pin para la línea SCL.

Sintaxis
CONFIG SCL = pin

Nota

pin	Pin del puerto a utilizar por la linea I2C-SCL.

Cuando usa diferentes pins en diferentes proyectos, puede usar esta instrucción para ignorar la opciones del compilador definida para el pin SDA.

De este modo el programa tendrá esplícitamente direccionado el pin utilizado y no será necesario anotarlo separadamente para modificarlo en *Option Settings*.

Ver también
CONFIG SDA

Ejemplo:
```
CONFIG SCL = P3.5              'P3.5 es la línea SCL
```

CONFIG DEBOUNCE

Acción
Configura el tiempo de retardo para la instrucción DEBOUNCE.

Sintaxis
CONFIG DEBOUNCE = time

Nota

time	Costante numérica que especifica el retardo en mS.

Si el tiempo de ***debounce*** no es configurado, se aplica por defecto en 25mS.
Este tiempo es referido a una frecuencia de reloj del procesador a 12 MHz.

Ver también
DEBOUNCE

Ejemplo:
```
Config Debounce = 25 mS              '25 mS por defecto
```

CONFIG SPI

Acción
Configura la istrucción relativa al puerto SPI.

Sintaxis
CONFIG SPI = SOFT, DIN = PIN, DOUT = PIN , CS = PIN, CLK = PIN

Nota

DIN	Pin a utilizar por Data input, por Ejemplo: P1.0
DOUT	Pin a utilizar por Data output, por Ejemplo: P1.1
CS	Pin a utilizar por Chip select, por Ejemplo: P1.2
CLK	Pin a utilizar por Clock, por Ejemplo: P1.3

Ver también
SPIIN SPIOUT

Ejemplo:
```
Config SPI = SOFT, DIN = P1.0 , DOUT = P1.1, CS = P1.2, CLK = P1.3
SPIOUT var , 1    'envia 1 byte
```

CONFIG WATCHDOG

Acción
Configura el timer de watchdog del dispositivo AT89C8252

Sintaxis
CONFIG WATCHDOG = time

Nota

time	Costante que define el intervalo del tiempo para el timer del watchdog, expresado en mS, con posibilidad de las siguientes opciones: 16 , 32, 64 , 128 , 256 , 512 , 1024 y 2048.

Cuando el WatchDog es activado, será generado un reset en el intervalo del tiempo especificado . (WatchDog= perro guardián)
Con el valor 2048, un reset ocurrirá después de 2 segundos, para evitar que esto suceda, sera necesario resetear periódicamente el timer del WD durante la ejecución del del programa.

Ver también
START WATCHDOG , STOP WATCHDOG , RESET WATCHDOG

Ejemplo:
```
'-----------------------------------------------------
' (c) 1998 MCS Electronics
' WATCHD.BAS muestra el uso del timer del watchdog del
' AT89S8252 seleccionar el 89s8252.dat !!!
'-----------------------------------------------------
Config Watchdog = 2048        'reset despues de 2048 mSec
Start Watchdog                'start del timer del watchdog
Dim I As Word
For I = 1 To 10000
      Print I                 'imprime el valor
' Reset Watchdog
'podrá notar que el ciclo for next no será completado debido al reset,
'si habilita la istrucción Reset Watchdog, y después de reiniciar el
'programa, será posible ver terminar el ciclo porque wd-timer será
'reseteado antes de alcanzar los 2048 msec
Next
End
```

COUNTERx

Acción

Ajusta o lee la variable COUNTER0 ó COUNTER1.
Para procesadores 8052 y compatibles con TIMER2, puede usarse también COUNTER2.

Sintaxis

COUNTERX = var ó
var = **COUNTERX**

Nota

var	Una variable de tipo Byte, Integer/Word ó una constante cuyo valor será asignado al contador.
counterX	COUNTER0 , COUNTER1 or COUNTER2.

Con la instrucción COUNTERX = 0 es posible resetear el contador.

El contador puede contar desde 0 a 255 en modalidad 2 (8-bit auto recarga) y fino hasta 65535 en modalidad 1 (16-bit).

La variabile COUNTERX se destina a ajustar (*set*) o leer el registro interno TIMER/COUNTER desde BASCOM. COUNTER0 = TL0 y TH0.

COUNTERx es una variable reservada de 16 bits de longitud.
Para ajustar TLx ó THx, puede ser usado por Ejemplo: TL0 = 5.

Note que la variable de COUNTERx opera en los TIMERS y COUNTER, porque los TIMERS y COUNTER son la misma cosa salvo el modo en el que trabajan. Para ajustar un valor de recarga, usar la declaración LOAD.

Ejemplo:

```
'-----------------------------------------------------------
' (c) 1997,1998 MCS Electronics
'-----------------------------------------------------------
' file: COUNTER.BAS
'-----------------------------------------------------------
' Conectar a la entrada del timer P3.4 un generador de frecuencia
' *TIMER/COUNTER 1 es usado como generador de baudrate
'-----------------------------------------------------------

Dim A As Byte , C As Integer
Config Timer0 = Counter , Gate = Internal , Mode = 1
'Timer0 = counter : timer0 trabaja como countador
'Gate = Internal : control de gate (entrada de disparo) no externo.
'Mode = 1 : 16-bit counter
Counter0 = 0            'resetea el contador, lo pone a 0
Start Counter0          'habilita el countador
Do                      'inicia un bucle (loop)
      A = Inkey         'lee una entrada desde el teclado
      C = Counter0      'resetea el contador, lo pone a 0
      Print C           'imprime el valor del contador
      Loop Until A = 27 'si la tecla pulsada es escape sale del bucle.
```

End

```
Por Ejemplo: la siguiente instruccion esta traducida en ASM:
COUNTER0 = 1000

Código generado :
Clr TCON.4
Mov tl0,#232
Mov th0,#3
```

CPEEK()

Acción
Devuelve un byte almacenado en la memoria de programma.

Sintaxis
var **= CPEEK(** address **)**

Nota

var	Variable numérica a la que és asignado el contenido de la dirección localizada en address
address	Variable numérica o constante que contiene la dirección de la localización.

No existe istrucción CPOKE porque es imposibile escribir en la memoria del programa.

Ver también
PEEK , POKE , INP , OUT

Ejemplo:
```
'-----------------------------------------------------
' (c) 1998 MCS Electronics
' PEEK.BAS
' muestra el uso de PEEK, POKE, CPEEK, INP y OUT
'
'-----------------------------------------------------
Dim I As Integer , B1 As Byte

'volcado de la memoria interna.
For I = 0 To 127              'para un 8052 se puede usar 225
' Break
      B1 = Peek(i)            'toma un byte de la memoria interna
      Printhex B1 ; " ";
      'Poke I , 1             'escribe un valor en la memoria interna.
Next
Print                         'nueva línea (linefeed)

'Atención cuando se escribe en la memoria interna !!
```

CURSOR

Acción
Fija el estado del cursor del display LCD.

Sintaxis
CURSOR ON / OFF BLINK / NOBLINK

Nota
Pueden ser usados tambien ambos parámetros a la vez ON / OFF y BLINK / NOBLINK.

Al iniciar la alimentación al display LCD, es ajustado por defecto el cursor en modo ON y NOBLINK.

Ver también
DISPLAY

Ejemplo:
```
Dim a As Byte
a = 255
LCD a
CURSOR OFF              'oculta el cursor.
Wait 1                  'espera 1 segundo.
CURSOR BLINK            'cursor intermitente.
End
```

DATA

Acción
Especifica el valor que será leido sucesivamente con la instrucción READ.

Sintaxis
DATA var [, varn]

Nota

var	Costante numérica ó *String* (Cadena de texto).

Diferencia con QB
Costante de tipo Integer ó Word debe de terminar con el signo **%**.
Costante de tipo Long debe de terminar con el signo **&**.
Costante de tipo Single debe de terminar con el signo **!**

Ver también
READ , RESTORE

Ejemplo:
```
DIM a AS BYTE, I AS BYTE, L AS Long, S As XRAM STRING * 15
RESTORE DTA                    'inicializa el puntero de datos.
FOR a = 1 TO 3
      READ a : PRINT a         'lee el dato y lo visualiza
NEXT
RESTORE DTA2                   'inicializa el puntero de datos.

READ I : PRINT I
READ I : PRINT I
RESTORE DTA3
READ L : PRINT L
RESTORE DTA4
READ S : PRINT S
END

DTA1:
DATA 5, 10, 100
DTA2:
DATA -1%, 1000%
'Costante de tipo Integer ó Word debe de terminar con el signo %
'(Integer : <0 or >255)
DTA3:
DATA 1235678&
'Costanti di tipo Long debe de terminar con el signo &
DTA4:
DATA "Hola mundo"
REM Sobre la misma línea pueden coexistir constantes de diverso tipo.
DATA "TEST" , 5 , 1000% , -1& , 1.1!
```

DEBOUNCE

Acción
Es un filtro (debounce) para el pin del puerto conectado a un interruptor.

Sintaxis
DEBOUNCE Px.y , state , label [, SUB]

Nota

Px.y	Pin del puerto a examinar, por Ejemplo: P1.0.
state	0 salta cuando Px.y es bajo, 1 cuando Px.y es alto.
label	La etiqueta de (GOTO) cuando el estado especificado es detectado.
SUB	La etiqueta de (GOSUB) cuando el estado especificado es detectado.

Si se especifica el parámetro opcional SUB, el salto a la etiqueta lo realiza con la instrucción GOSUB en vez del GOTO.

La instrucción DEBOUNCE espera que un pin del puerto sea alto (1) o bajo (0).

Cuando es leido el estado del pin, espera 25 mS hasta chequear de nuevo el estado del pin (esto elimina rebotes producidos por el contacto de un interruptor).

Cuando la condición todavía es verdad y no había nigún salto antes, salta a la etiqueta especificada.

Cuando DEBOUNCE es ejecutado otra vez, el estado del interruptor se debe de haber colocado en la posición original antes de que pueda realizar otro salto a la etiqueta especificada.

Cada pin asociado a la instrucción DEBOUNCE comporta la utilización de 1 bit de la memoria interna para memorizar el estado.

Lo que también debe mencionarse es que P2.2-P2.7 y P3 tienen resistencias internas *pull up* (ver manual del hardware del microprocesador que se use).

Esto puede influir sobre las instrucciones de debounce. Con estos pins es mejor utilizar la instrucción de debounce del siguiente modo: **Debounce P1.1, 0, Pr [, sub],** cuando no requiere resistencias de pull up externas. Para evitar que se produzcan accidentalmentes saltos a la etiqueta a causa de las resistencias de pull up.

Ver también
CONFIG DEBOUNCE

Ejemplo:

```
'-----------------------------------------------------
' DEBOUN.BAS
' muestra el uso de DEBOUNCE
'-----------------------------------------------------
CONFIG DEBOUNCE = 30    'sin la instrucción config, es usado el valor por
                        'defecto de 25mS

Do
'Debounce P1.1 , 1 , Pr 'esto para saltar cuando la entrada es alto (1)
Debounce P1.0 , 0 , Pr,SUB
                    ' ^----- etiqueta al cual saltará
               ' ^---------- salta cuando P1.0 va a nivel bajo 0)
        ' ^---------------- analiza P1.0
'cuando P1.0 va a nivel bajo salta a la subrutina Pr
'P1.0 debe volver alto antes de habilitar de nuevo.
'salta a la etiqueta Pr
Loop
End

Pr:
Print "P1.0 es a nivel bajo"
Return
```

DECR

Acción
Decrementa una variable en uno.

Sintaxis
DECR var

Nota

Var	Variable numérica a decrementar.

var : Byte, Integer, Word, Long, Single.

Hay a menudo situaciones donde se necesita disminuir un número en 1.
La instrucción **DECR** realiza esta operación en modo más rápido que el clásico var = var - 1.

Ver también
INCR

Ejemplo:
```
'-----------------------------------------------------------
' (c) 1997,1998 MCS Electronics
'-----------------------------------------------------------
' file: DEC.BAS
' demo: DECR
'-----------------------------------------------------------
Dim A As Byte
A = 5              'asigna un valor a la variable A
Decr A             'decrementa (en uno)
Print A            'visualiza al valor de A
End
```

DECLARE SUB

Acción
Declara una subrutina.

Sintaxis
DECLARE SUB TEST[(var as type)]

Nota

test	Nombre del procedimiento.
Var	Nombre de la/las variable/es. Hasta un máximo de 10.
Type	Tipo de la/las variable/es. Bit, Byte,Word/Integer, Long ó String.

Es necesario declarar cada subrutina antes de escribir el procedimiento que deberá de ser seguido por la subrutina.

Ver también
CALL, SUB

Ejemplo:
```
Dim a As Byte, b1 As Byte, c As Byte
Declare Sub Test(a As Byte)
a = 1 : b1 = 2: c = 3
Print a ; b1 ; c
Call Test(b1)
Print a ;b1 ; c
End

Sub Test(a As Byte)
Print a ; b1 ; c
End Sub
```

Defint, DefBit, DefByte, DefWord

Acción
Declara el tipo para todas las variables no dimensionadas por **DefXXX type**.

Sintaxis
DEFBIT b
DEFBYTE c
DEFINT I
DEFWORD x

Diferencia con QB
QB permite especificar un campo, por ejemplo: DEFINT A - D. En BASCOM esto no está permitido.

Ejemplo:
```
Defbit b : DefInt c     'tipo por defecto para bit e integers
Set b1                  'set bit a 1
c = 10                  'Toma el valor c = 10
```

DEFLCDCHAR

Acción
Define un carácter especial personalizado para el display LCD

Sintaxis
DEFLCDCHAR char,r1,r2,r3,r4,r5,r6,r7,r8

Nota

char	Variable que representa el carácter (0÷7).
r1-r8	El valor asignado a la fila del caráctere.

char : Byte, Integer, Word, Long, Constant.
r1-r8 : Constant.

Es posibie utilizar el **LCD designer** para costruir en forma gráfica un carácter.
Es importante que después de una instrucción DEFLCDCHAR siga un CLS.
Un caráctere especial definido podrá ser visualizado con la funcion Chr().

Ver también
Edit LCD designer

Ejemplo:
```
DefLCDchar 0,1,2,3,4,5,6,7,8  'define el carácter especial.
Cls                           'selecciona la RAM de DATOS del display LCD.
LCD Chr(0)                    'muestra el carácter.
End
```

DELAY

Acción
Produce un breve retardo en la ejecución del programa.

Sintaxis
DELAY

Nota
DELAY es indicado para producir un breve retardo.
El tiempo de retardo producido es de 100 microsegundos si se emplea en el sistema una frecuencia de reloj de 12 MHz.

Ver también
WAIT , WAITMS

Ejemplo:
```
P1 = 5       'escribe 5 en el port 1.
DELAY        'espera a que el hardware esté preparado.
```

DIM

Acción
Dimensiona una variable.

Sintaxis
DIM var **AS [XRAM/IRAM]** type

Nota

var	Cualquier nombre válido para una variable: b1. **Var** puede ser una *array* (matriz), por ejemplo : ar(10).
type	Bit, Byte, Word, Integer, Long, Single ó String
XRAM	XRAM si la variable se memoriza en la memoria externa.
IRAM	IRAM si la variable se memoriza en la memoria interna (default)

Una variable de tipo *string* (cadena) requiere un parámetro adicional que defina la longitud:
Dim s As XRAM **String * 10**
En este caso la cadena podrá tener una longitud de 10 carácteres.
Las variables de tipo BITS son siempre salvadas en la memoria interna.

Nota:
Cada 8 bits usados ocupa 1 byte en memoria.
Cada byte usado ocupa 1 byte en memoria.
Cada integer/word usado ocupa 2 bytes en memoria.
Cada Long/Single usado ocupa 4 bytes en memoria.

Differenze da QB
En QB no es necesario dimensionar cada una de las variables que intenta usar.
En BASCOM es obligatorio dimensionar cada variable antes de ser usada.

XRAM/IRAM son opciones no disponibles en QB.

See Also
CONST , ERASE

Ejemplo:
```
'-----------------------------------------------------------
' (c) 1997-1999 MCS Electronics
'-----------------------------------------------------------
' file: DIM.BAS
' demo: DIM
'-----------------------------------------------------------
Dim B1 As Bit             'bit puede ser 0 o 1
Dim A As Byte             'byte válido entre 0 ÷ 255
Dim C As Integer          'integer válido entre -32767 ÷ +32768
Dim A As String * 10      'cadena con un longitud de 10 carácteres.
Dim ar(10) As Byte        'dimensiona un array (matriz).
'asigna bits
B1 = 1                    'pone el bit a 1 ó también
```

```
Set B1                    'puede usar la instrucción set


'asigna bytes
A = 12
A = A + 1


'asigna integer
C = -12
C = C + 100
Print C
End
```

DISABLE

Acción
Deshabilita la interrupción esfecificada.

Sintaxis
DISABLE interrupt

Nota
Interrupt : **INT0**, **INT1**, **SERIAL**, **TIMER0**, **TIMER1 or TIMER2**.

Por defecto todas las interrupciones son deshabilitadas.

Para deshabilitar TODAS las interrupciones es suficiente con especificar INTERRUPTS.

Para activar la posibilidad de habilitación ó deshabilitación de cada una de las interrupciones debe de ser previamente especificado ENABLE INTERRUPTS.

Pueden ser gestionadas mas o menos interrupciones en función del chip utilizado.

Ver las especificaciones del microprocesador usado para ampliar detalles.

Ver también
ENABLE

Ejemplo:
```
ENABLE INTERRUPTS       'habilita el control de las interrupciones
ENABLE TIMER0           'habilita TIMER0
DISABLE SERIAL          'deshabilita la interrupción serie
DISABLE INTERRUPTS      'deshabilita todas las interrupciones
```

DISPLAY

Acción
Conecta ó desconecta el **display** LCD (visualizador LCD).

Sintaxis
DISPLAY ON / OFF

Nota
Al conectar la alimentación el display siempre es puesto a ON.

Ver también
-

Ejemplo:
```
Dim a As Byte
a = 255
LCD a
DISPLAY OFF
Wait 1
DISPLAY ON
End
```

DO .. LOOP

Acción
Repite un bloque de instrucciones hasta que la condición sea verdadera.

Sintaxis
DO
istrucciones
. . .
LOOP [UNTIL expression **]**

Nota
Se puede abandonar un ciclo DO..LOOP no completado con la instrucción EXIT DO.

Ver también
EXIT , WHILE WEND , FOR , NEXT

Ejemplo:
```
Dim A As Byte
DO                          'empieza el bucle
      A = A + 1             'incrementa A
      PRINT A               'lo visualiza
LOOP UNTIL A = 10           'Repite el loop (bucle) hasta que A = 10
Print A                     'A valdrá 10
```

ELSE

Acción
Ejecuta una instrucción alternativa cuando la espresión IF-THEN resulta falsa.

Sintaxis
ELSE

Nota
La instrucción ELSE no debe de ser utilizada en la estructura IF THEN **...** END IF. Para verificar ulteriores condiciones puede ser implementada la instrucción ELSEIF.

IF a = 1 THEN
...
ELSEIF a = 2 THEN
..
ELSEIF b1 > a THEN
...
ELSE
...
END IF

Ver también
IF , END IF SELECT CASE

Ejemplo:
```
A = 10                  'La variable a es igual a 10
IF A > 10 THEN          'toma una decisión
      PRINT " A >10"    'esto no será visualido
      ELSE              'en caso alternativo...
      PRINT " A no es mayor que 10"      'esto será visualizado
END IF
```

ENABLE

Acción
Habilita una interrupción especificada.

Sintaxis
ENABLE interrupt

Nota
Interrupt **INT0**, **INT1**, **SERIAL**, **TIMER0, TIMER1** o **TIMER2**

Por defecto todas las interrupciones son deshabilitadas.
Para activar la posibilidad de habilitar o deshabilitar cada una de las interrupciones se debe de especificar ENABLE INTERRUPTS.

Para activar la posibilidad de habilitación ó deshabilitación de cada una de las interrupciones debe de ser previamente especificado ENABLE INTERRUPTS.

Pueden ser gestionadas más o menos interrupciones en función del chip utilizado.
Ver las especificaciones del microprocesador usado para ampliar detalles.

Ver también
DISABLE

Ejemplo:
```
ENABLE INTERRUPTS 'permite activar, habilitar (enable) las interrupciones.
ENABLE TIMER1     'habilita la interrupción de TIMER1.
```

END

Acción
Finaliza la ejecución del programa.

Sintaxis
END

Nota
Para terminar un programa puede ser utilizada también la instrucción **STOP**.
Cuando es encontrada un instrucción END ó STOP se produce un bucle sin fin.

Ver también
STOP

Ejemplo:
```
PRINT " Hola"      'visualiza esto.
END                'fin de la ejecución.
```

END IF

Acción
Termina una struttura IF .. THEN.

Sintaxis
END IF or **ENDIF**

Nota
Es indispensable terminar un estructura IF .. THEN utilizando la instrucción END IF.
Es posible anidar instrucciones IF ..THEN.

El uso de ELSE es opcional.
Si es habilitada la opción “reformat” el editor de BASCOM convierte ENDIF en End If.

Ejemplo:
```
Dim nmb As Byte
AGAIN:                                  'Etiqueta
INPUT " Numero " , nmb                  'Espera la entrada de un número
IF a = 10 THEN                          'lo compara
      PRINT " Numero es 10"             'corresponde
      ELSE                              'no corresponde
      IF nmb > 10 THEN                  'es mayor ?
            PRINT " Numero > 10"        'si
            ELSE                        'no
            PRINT " Numero < 10"        'lo visualiza
      END IF                            'fin del la estructura
END IF                                  'fin del la structura
END                                     'fin del programa
```

ERASE

Acción
Cancela, borra, elimina una variable liberando la memoria utilizada.

Sintaxis
ERASE var

Nota

var	Nombre de la variable a cancelar.

Después de eliminar una variable, esta puede ser nuevamente dimensionada. Cuando resulte conveniente hacer uso de variables temporales, una vez utilizada, se elimina liberando espacio en memoria. Esto permite usar menos memoria.

Es posible cancelar sólo la variable dimensionada con la última instrucción DIM. Cuando se han dimensionado 2 variables para su uso como temporales, se pueden eliminar estas variables. El orden para la eliminación no es relevante.

Ejemplo :
```
Dim a1 as byte , a2 as byte , a3 as byte , a4 as byte 'uso las variables.
ERASE a3 : ERASE a4    'cancelación de las 2 últimas variables porque eran
                       'temporales.

Dim a5 as Byte         'Dimensionamiento de una nueva variable.
```

Ahora no será posible cancelar las variables a1 y a2 !
La variable cancelada no vendrá reportada en el archivo de informe del simulador.

Ejemplo:
```
DIM A As Byte      'dimensiona una variable
A = 255            'le asigna un valor
Print A            'la visualiza
ERASE A            'elimina la variable
DIM A AS INTEGER   'la redimensiona como INT
PRINT A            'la visualiza nuevamente

REM A usa el espacio asignado antes de eliminar la variable, este espacio.
REM es asignado a la variable A que todavía retiene el valor asignado.
REM anteriormente.
```

EXIT

Acción
Sale de un bucle FOR..NEXT, DO..LOOP , WHILE ..WEND o SUB..END SUB.

Sintaxis
EXIT [**FOR**] [**DO**] [**WHILE**] [**SUB**]

Nota
Con la instrucción EXIT ... es posible salir en cualquier momento de una extructura.

Ejemplo:
```
IF a >= b1 THEN          'Una condición cualquiera para demostración...
   DO                    'inicia DO..LOOP
   A = A + 1             'incrementa a
     IF A = 100 THEN     'analiza si a = 100
       EXIT DO           'sale de DO..LOOP
     END IF              'fin de la estructura IF..THEN
   LOOP                  'fin de la estructura DO
END IF                   'fin de la estructura IF..THEN
```

FOR

Acción
Ejecuta un bloque de instrucciones un cierto número de veces.

Sintaxis
FOR var = start **TO/DOWNTO** end [**STEP** value]

Nota

var	Variable que utilizará como contador
start	Valor inicial del la variable var
end	Valor final del la variable var
value	Valor del incremento o decremento a aplicar a la variable var cada vuelta seguida de NEXT.

var : Byte, Integer, Word, Long, Single.
start: Byte, Integer, Word, Long, Single, Constant.
end : Byte, Integer, Word, Long, Single, Constant.
step : Byte, Integer, Word, Long, Single, Constant.

Para bucles de incremento se debe de utilizar TO.
Para bucles de decremento se debe de utilizar DOWNTO.

Una structura FOR debe de terminar con la propia instrución NEXT.

STEP es opcional y puede ser omitido, por defecto su valor es 1.

Ver también
NEXT , EXIT FOR

Ejemplo:
```
y = 10                          'Asigna el valor 10 a la variable y.
      FOR a = 1 TO 10           'repite 10 veces.
            FOR x = y TO 1      'esto también.
              PRINT x ; a       'visualiza el valor de x e y.
            NEXT                'siguiente next x (cuenta atrás).
      NEXT                      'siguiente next a (cuenta adelante).

Dim S As Single
      For S = 1 To 2 Step 0.1
        Print S
      Next
END
```

FOURTHLINE

Acción
Posiciona el cursor del display LCD al inicio de la cuarta línea.

Sintaxis
FOURTHLINE

Nota
Válido sólo para display LCD con 4 líneas.

Ver también
HOME , UPPERLINE , LOWERLINE , THIRDLINE , LOCATE

Ejemplo:
```
Dim a As Byte
a = 255
LCD a
Fourthline
LCD a
Upperline
END
```

FUSING

Acción
Formatea un valor en punto flotante **(floating-point)**.

Sintaxis
var = Fusing(fuente, máscara)

Nota

var	La cadena que es asignado el resultado.
source	Las Variables de tipo single han de ser formateadas.
mask	Máscara de formateo. ###.## El signo # es utilizado para indicar el número de cifras antes y después del punto decimal. El resultado será redondeado.

Ver también
STR

Ejemplo:
```
$large
Dim X As Single , Y As Single , Result As Single
Dim I As Integer
Dim Buf As String * 16
Input "Entre x " , X                          'introducir 2 valores.
Input "Entre y " , Y
Print "X+Y=" ; : Result = X + Y : Print Result 'calcula
Print "X-Y=" ; : Result = X - Y : Print Result
Print "X/Y=" ; : Result = X / Y : Print Result
Print "X*Y=" ; : Result = X * Y : Print Result
Buf = Fusing(result , #.##)                   'formatea una cadena
Print Buf                                     'y la visualiza
```

GET

Ación
Devuelve un byte del software UART.

Sintaxis
GET #channel , var

Nota

Channel	Constante numérica positiva que se refiere al canal abierto.
Var	La variable que recibe el valor desde el software UART.

Tenga en cuenta que el canal debe abrirse con la instrucción OPEN.
También, la instrucción CLOSE, debe ser la última en su programa. Por favor vea el comentario en la instrucción OPEN y el ejemplo en esta página.

Ver también
PUT

Ejemplo:
```
'En este ejemplo se usan los pins asignados normalmente para comunicación
Open "com3.1:9600" For Output As #1          'p3.1 normalmente se usa para Tx
Open "com3.0:9600" For Input As #2           'p3.0 normalmente se usa para Rx

S = "test this"                 'asignamos una cadena
Dum = Len(s)                    'asignamos a Dum la longitud de la cadena

For I = 1 To Dum                'para todos los caracteres desde la izquierda a la derecha
  A = Mid(s , I , 1)            'asignamos a A el caracter
  Put #1 , A                    'escribe el valor de A por el puerto de comunicaciones
Next

Do
  Get #2 , A                    'lee un caracter desde el puerto de comunicaciones
  Put #1 , A                    'lo envía por el puerto
  Print A                       'use el canal normal
Loop

Printbin #1, a                  ' Printbin también es soportado
Inputbin #2, a                  ' Inputbin también es soportado
Close #1                        ' finaliza y cierra el dispositivo
Close #2
End
```

GETRC

Acción
Devuelve el valor de una resistencia o condensador.

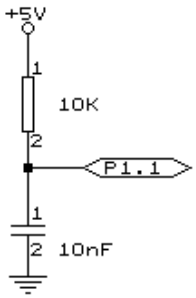
Sintaxis
var = GETRC(pin)

Nota

var	La cadena que recibirá el resultado.
pin	El pin del puerto que se conectará la R/C.

Ver también
–

Ejemplo de conexión:



Ejemplo:
```
'-----------------------------------------------------------------
' GETRC.BAS
' adquiere el valor de una resistencia.
' Conectar una resistencia variable de 10KOhm entre +5V e P1.7.
' Conectar un condensador de 10nF entre P1.7 y negativo (masa).
' La instrucción GETRC(pin) mide el tiempo necesario para la
' carga del condensador.
'-----------------------------------------------------------------
Config Timer0 = Timer , Gate = Internal , Mode = 1 'GETRC() usa timer 0
$baud = 9600                    'si es necesario usar el port serie
$crystal = 11059200
Dim W As Word                   'reserva espacio para la variable
Do                              'bucle contínuo
      W = Getrc(p1.7)           'adquiere el valor de la RC
      Print W                   'lo visualiza
      Wait 1                    'espera un segundo
Loop

'valor restituido con cap=10nF. Valores de resistencia medido con un DVM
' 250 per 10K9
' 198 per 9K02
' 182 per 8K04
```

```
' 166 per 7K
' 154 per 6K02
' 138 per 5K04
' 122 per 4K04
' 106 per 3K06
' 86 per 2K16
' 54 per 1K00
' 22 per 198 ohm
' 18 per 150 ohm
' 10 per 104 ohm
' 6 per 1 ohm (minimo)
'Como se puede apreciar la conversión es bastante lineal, se puede hacer un
'poco de matematicas para conseguir valores de R/C.
'Pero piense que la función sirve como una indicación aproximada para el
'valor de la resistencia, usted también puede cambiar el valor del
'condensador para conseguir valores más grandes.
'Con 10nF, el valor devuelto ocupa un byte
'Naturalmente R o C deberá de ser conocido para determinar otros valores.
```

GETRC5

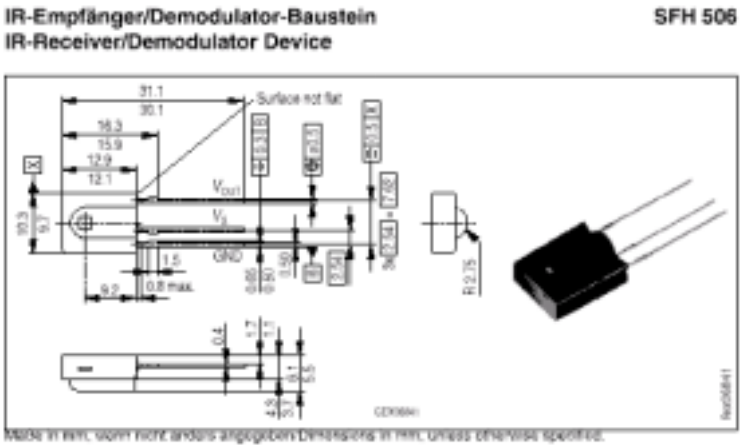
Acción
Devuelve el código y subdirección de un dispositivo de infrarojos RC5.

Sintaxis
GETRC5(address , command)

Nota

address	sub dirección recibida de RC5.
command	Comando (command) recibido de RC5.

Para utilizar esta función es necesario conectar un dispositivo de recepción de infrarojos RC5 tipo SHARP SFH506-36 al pin 2 del puerto 3 (conexión 3.2). Esta instrucción trabaja junto con la interrupción INT0. Vea el ejemplo siguiente cómo usarla.



Ejemplo:

```
'-----------------------------------------------------------------------
' RC5.BAS (c) 1999 MCS Electronics
' conectar un ricevitore InfraRosso SFH506-36 alla PORTA 3.2 (INT0)
'-----------------------------------------------------------------------
Dim New As Bit
Dim Command As Byte , Subaddress As Byte
clr tcon.0
On Int0 Receiverc5 Nosave
Enable Int0
Enable Interrupts
Do
      If New = 1 Then                'nuevo código recibido
            Print Command ; " " ; Subaddress
            New = 0                  'resetea el bit new
      End If
Loop
Receiverc5:                          'rutina de interrupción
Getrc5(Subaddress, command)
New = 1
Return
```

GOSUB

Acción
Salta y ejecuta una subrutina.

Sintaxis
GOSUB label

Nota

label	Nombre de la etiqueta a saltar.

Con GOSUB, el programa salta a la etiqueta especificada continuando la ejecucución desde la etiqueta.
Cuando encuentra la instrucción RETURN, el programa vuelve a la siguiente línea después de GOSUB y continúa con la ejecución del programa.

Ver también
GOTO CALL

Ejemplo:
```
GOSUB Rutina        'salta a la etiqueta Rutina
Print "Hola"        'continúa al retorno del salto
END                 'termina el programa

Rutina:             'esta es una subrutina
x = x + 2           'efectúa un cálculo
PRINT X             'visualiza el resultado
RETURN              'vuelve de la subrutina
```

GOTO

Acción
Salta a la etiqueta especificada.

Sintaxis
GOTO label

Nota
La etiqueta puede ser de hasta una longitud máxima de 32 carácteres
El compilador avisa cuando se duplica una etiqueta en el curso del programa.
Una etiqueta debe de terminar siempre con dos puntos ':'

Ver también
GOSUB

Ejemplo:
```
Start:              'la etiqueta ha de terminar con dos puntos
A = A + 1           'incrementa a
IF A < 10 THEN      'es inferior a 10?
GOTO Start          'inicia de nuevo, salta a la etiqueta
END IF              'finaliza IF
PRINT " Ready"      'es todo
```

HEX()

Acción
Devuelve un número hexadecimal en forma de cadena

Sintaxis
var = Hex(x)

Nota

var	Variable de tipo cadena.
X	Variable numérica de tipo Byte, Integer ó Word.

Ver también
HEXVAL

Ejemplo:
```
Dim a As Byte, S As String * 10
a = 123
s = Hex(a)
Print s
End
```

HEXVAL()

Acción
Convierte una cadena que representa un número Hexadecimal en una variable numérica

Sintaxis
var **= HEXVAL(** x **)**

Nota

var	Variable numérica a asignar el valor.
X	Cadena que contiene la expresión en Hexadecimal.

var : Byte, Integer, Word, Long.
x : String.

La cadena a convertir debe de tener una longitud de 2 bytes, 4 bytes ó 8 bytes, para Bytes, Integers/Words y Longs respectivamente.

Diferencia en QB
In QB se puede usar la función VAL() para convertir cadenas hexadecimales. Esto requeriría efectuar test extas para las señales &H, en BASCOM ha sido diseñada una función separada.

Ver también
HEX , VAL , STR

Ejemplo:
```
Dim a As Integer, s As String * 15
s = "000A"
a = Hexval(s) : Print a
End
```

HIGH

Acción
Extrae el byte más significativo de una variable.

Sintaxis
var = HIGH (s)

Nota

var	Variable a la cual será asignada el valor MSB del la variable S.
s	Variable a extraer el valor MSB: byte más significativo.

Ver también
LOW

Ejemplo:
```
Dim I As Integer , Z As Byte
I = &H1001
Z = High(I)       'es 16
```

HIGHW

Acción
Recupera los dos bytes mas significativos de una cadena tipo **long**.

Sintaxis
var = HIGHW(s)

Nota

var	La variable que es asignada con los dos MSB de la variable s. El formato debe ser un Entero (**Integer**) o Palabra (**Word**).
s	La variable fuente para conseguir los bytes MSB. Debe ser del tipo largo (**long**).

Ver también
LOW HIGH LOWW

Ejemplo:
```
Dim I As Long , Z As Word
I = &H10011001
Z = High(I)
```

HOME

Acción
Posiciona el cursor al la localización 1 del la línea specificada.

Sintaxis
HOME UPPER / LOWER /THIRD / FOURTH

Nota
Si se utiliza solo HOME el cursor será posicionado a la primera posición de la primera línea.
Está permitido la abreviación de la línea utilizando una sola inicial, por ejemplo: HOME U para indicar HOME UPPER.

Ver también
CLS , LOCATE , LCD

Ejemplo:
```
Lowerline
LCD " Hello"
Home Upper          'También es válido: Home U
LCD " Upper"
```

I2CRECEIVE

Acción
Recibe un dato desde un dispositivo serie I2C bus.

Sintaxis
I2CRECEIVE slave, var
I2CRECEIVE slave, var ,b2W, b2R

Nota

slave	Variable ó Constante de tipo Byte, Word/Integer que contiene la dirección del dispositivo slave (esclavo) I2C.
Var	Variable de tipo Byte ó Integer/Word que recibirá la información que proviene del dispositivo I2C direccionado.
b2W	Número de bytes a escribir. Atención a no especificar un número excesivo de bytes !
b2R	Número de bytes a recibir. Atención a no especificar un número excesivo de bytes !

En BASCOM LT es posible definir DATA como var, pero, desde que soporta arrays (matrices), es preferible utilizar arrays en vez de DATA.

Este comando requiere hardware adicional. Ver apéndice D.

Ver también
I2CSEND

Ejemplo:
```
x = 0                          'reset del la variable
slave = &H40                   'dirección de un slave tipo PCF 8574 (I/O)
I2CRECEIVE slave, x            'toma el valor
PRINT x                        'lo visualiza
Dim buf(10) As String
buf(1) = 1 : buf(2) = 2
I2CRECEIVE slave, buf(), 2, 1 'envía do bytes y recibe uno
Print buf(1)                   'visualiza el byte recibido
```

I2CSEND

Acción
Envía datos hacia un dispositivo I2C.

Sintaxis
I2CSEND slave, var
I2CSEND slave, var , bytes

Nota

slave	Dirección del dispositivo esclavo I2C bus.
var	Byte, Integer/Word ó número que representa el valor a envíar al dispositivo I2C bus.
bytes	Número de bytes a enviar.

Este comando requiere hardware adicional. Ver apéndice D.

Ver también
I2CRECEIVE

Ejemplo:
```
x = 5                   'asigna 5 al la variable
Dim ax(10) As Byte
slave = &H40            'dirección del esclavo tipo PCF 8574 (I/O)
bytes = 1               'envía 1 byte
I2CSEND slave, x        'envía el valor o
For a = 1 to 10
      ax(a) = a         'Llena la matriz con datos.
Next
bytes = 10
I2CSEND slave,ax(),bytes
END
```

I2START,I2CSTOP, I2CRBYTE, I2CWBYTE

Acción
I2CSTART genera una condición de start (inicio) sobre el bus I2C.
I2CSTOP genera una condición de stop (paro) sobre el bus I2C.
I2CRBYTE recibe un byte de un dispositivo I2C.
I2CWBYTE envia un byte a un dispositivo I2C.

Sintaxis
I2CSTART
I2CSTOP
I2CRBYTE var, 8/9
I2CWBYTE val

Nota

var	Variable que recibe el valor del dispositivo I2C.
8/9	Specifica 8 ó ACK si hay mas bytes para leer. (**ACK**) Specifica 9 ó NACK si es el último byte para leer. (**NACK**)
val	Variable ó constante a escribir en el dispositivo I2C.

Este comando requiere hardware adicional. Ver apéndice D.

Estas instrucciones están previstas como complemento de las funciones I2CSEND e I2CRECEIVE.

Ver también
I2CRECEIVE, I2CSEND

Ejemplo:
```
-------- Lectura y escritura de un byte en una EEPROM 2404 ----------------
DIM a As Byte
DIM adresW AS CONST 174 'comando write para 2404
DIM adresR AS CONST 175 'dirección read para 2404
I2CSTART                'genera el comando de start del bus I2C
I2CWBYTE adresW         'envía la dirección del esclavo
I2CWBYTE 1              'envía la dirección de la EEPROM a escribir
I2CWBYTE 3              'envía el valor
I2CSTOP                 'genera el comando de stop del bus I2C
WaitMS 10               'espera 10 mS necesarios para la escritura del chip

----------Ahora lee el valore asignándolo al la variable a ----------------
I2CSTART          'genera el comando start del bus I2C
I2CWBYTE adresW   'envía la dirección del escalvo
I2CWBYTE 1        'envía la dirección de la EEPROM a leer
I2CSTART          'repite el comando start
I2CWBYTE adresR   'envía la dirección esclavo de la EEPROM
I2CRBYTE a,9      'recibe el valor en a. 9 es el ultimo byte para recibir
I2CSTOP           'genera el comando de stop del bus I2C
PRINT a           'visualiza el valor recibido
END
```

IDLE

Acción
Fuerza al procesador al estado idle.

Sintaxis
IDLE

Nota
Cuando el procesador funciona en modo IDLE el reloj del sistema es removido desde la CPU, no desde las interrupciones lógicas, puerto serie ó temporizadores/contadores.

El procesador entra en funcionamiento normal cuando es recibida una interrupción ó se produce un reset por hardware a través del pin RESET.

Ver también
POWERDOWN

Ejemplo:
```
IDLE
```

IF

Acción
Permite ejecución condicional o salto, basado en la evaluación de una expresión de tipo Boolean.

Sintaxis
IF expresión **THEN**
[**ELSEIF** expresión **THEN**]
[**ELSE**]
END IF

Nota

expresión	Cualquier expresión estimable como verdadera ó falsa.

Ahora es posible usar una simple línea en IF:
IF expresión THEN instrucción [ELSE instrucción]
[ELSE] es opcional.

También es posible evaluar un solo bit como expresión:
IF var.bit = 1 THEN

Ver también
ELSE , END IF

Ejemplo:
```
DIM A AS INTEGER
A = 10
IF A = 10 THEN                                  'expresiones de prueba
      PRINT " Esta parte viene enseguida."      'Esto será visualizado
      ELSE
      PRINT " Esta parte no viene enseguida."   'Esto no
END IF
IF A = 10 THEN PRINT "Nuevo en BASCOM"
IF A = 10 THEN GOTO LABEL1 ELSE PRINT "A<>10"
LABEL1:


REM El ejemplo: siguiente muestra el uso avanzado de IF THEN
IF A.15 = 1 THEN                                'analiza un bit
      PRINT "BIT 15 = 1"
END IF


'El ejemplo: siguiente muestra el uso de una sola linea de IF THEN [ELSE]
IF A.15 = 0 THEN PRINT "BIT 15 es Reset" ELSE PRINT "BIT 15 es set"
```

INCR

Acción
Incrementa en una unidad el valor de la variable.

Sintaxis
INCR var

Nota

Var	Variabile numérica a incrementar.

Hay a menudo situaciones donde se necesite incrementar una variable en 1 unidad. La instrucción **Incr** realiza esta operación más rapido que el clásico var=var+1.

Ver también
DECR

Ejemplo:
```
Do                          'inicio del bucle (loop)
      Incr a                'incrementa a en 1
      Print a               'visualiza a
Loop Until a > 10           'repite hasta que a es mayor de 10
```

INKEY

Acción
Restituye el valor ASCII del primer carácter presente en el buffer de entrada del puerto serie.

Sintaxis
var = **INKEY**

var	Variable de tipo Byte, Integer, Word, Long o String.

Nota
Si no hay presente un carácter será restituido por un 0
INKEY puede ser usada sólo si hay presente un interface serie (RS-232) en el uP.
Consultar el manual para el diseño de un interface serie.
El interface RS-232 puede ser conectado a un puerto de comunicación de un PC.

Ver también
WAITKEY

Ejemplo:
```
DO                          'inicio del bucle
      A = INKEY             'toma el carácter del puerto serie
      IF A > 0 THEN         'la variable es > 0?
            PRINT A         'sí , la visualiza
      END IF
LOOP                        'bucle sin fin
```

INP()

Acción
Ingresa un byte leido de un port hardware o de una loclalización de memoria externa.

Sintaxis
var = **INP**(address)

Nota

var	Variable numérica al la cual será asignado el valor.
address	Direccón donde leerá el valor.

La instrucción INP() sólo puede usarse en sistemas de microprocesador con direccionamiento de memoria externa.

Ver también
OUT

Ejemplo:
```
Dim a As Byte
a = INP(&H8000)    'lee un valor del bus de datos (d0÷d7)
                   'en la dirección hex 8000
Print a
End
```

INPUTBIN

Acción
Lee un valor binario del puerto serie.

Sintaxis
INPUTBIN var1 [,var2]

Nota

var1	Variable al la cual será asignado el valor leido del puerto serie.
var2	Segunda (o más) variable opcional a la cual será asignado ulteriores carácteres del puerto serie.

El número de bytes a leer depende del tipo de variable usada.

Con una variable de tipo Byte, desde el puerto serie será leido un carácter.

Una variable de tipo Integer esperará 2 carácteres y una **array** (matriz) esperará hasta que la misma esté completa.

La instrucción INPUTBIN recibirá el número exacto de bytes indicado sin necesidad de esperar un <RETURN>.

Ver también
PRINTBIN

Ejemplo:
```
Dim a As Byte, C as Integer
Inputbin a, c     'si aspetta 3 caratteri
End
```

INPUTHEX

Acción
Permite la introducción de datos desde un teclado durante la ejecución del programa.

Sintaxis
INPUTHEX [" prompt"] , var [, varn] [NOECHO]

Nota

prompt	Cadena opcional que se visualiza antes del cursor que permite la introducción de datos.
Var,varn	Variable numérica al la cual será asignada el valor introducido.
NOECHO	Deshabilita el eco hacia el puerto de comunicación.

La variable INPUTHEX puede ser utilizada cuando existe un interface serie RS-232. Consultar el manual para la información de la realización de un interface serie RS-232.

El interface serie RS-232 puede ser conectado al puerto de comunicación serie de un PC.

De este modo es posible usar un emulador de terminal y un teclado como dispositivo de entrada. En BASCOM es integrado un emulador de terminal.

Si var es de tipo Byte debe ser introducido 2 carácteres desde el teclado.
Si var es de tipo Integer/Word debe ser introducido 4 carácteres desde el teclado.
Si var es de tipo Long debe ser introducido 8 carácteres desde el teclado.

Differenze da QB
En QB es posible especificar la opción &H para aceptar el carácter en formato hexadecimal.

BASCOM implementa una nueva instrucción: INPUTHEX.

Ver también
INPUT

Ejemplo:
```
Dim x As Byte
INPUTHEX " Enter a number " , x     'espera una entrada
```

INPUT

Acción
Permite la introducción de datos del teclado durante la ejecición del programa.

Sintaxis
INPUT [" prompt"] , var [, varn] [NOECHO]

Nota

prompt	Cadena opcional que se visualiza antes del cursor que permite la introducción de datos.
Var,varn	Variable que aceptará el valor de entrada o cadena introducida.
NOECHO	Deshabilita el eco hacia el puerto de comunicación.

La rutina INPUT puede ser utilizada cuando existe un interface serie RS-232. Consultar el manual para la información de la realización de un interface serie RS-232.
El interface serie RS-232 puede ser conectado al puerto de comunicación serie de un PC.
De este modo es posible usar un emulador de terminal y un teclado como dispositivo de entrada. En BASCOM es integrado un emulador de terminal.

Diferencia con QB
En QB es posible especificar la opción &H para reconocer que la cadena usada es en formato hexadecimal.

BASCOM implementa una nueva instrucción: INPUTHEX.

Ver también
INPUTHEX PRINT

Ejemplo:
```
'-----------------------------------------------------------
' (c) 1997,1998 MCS Electronics
'-----------------------------------------------------------
' file: INPUT.BAS
' demo: INPUT, INPUTHEX
'-----------------------------------------------------------
'Para diversos baudrate y frecuencias del  cuarzo usar la
'directiva $BAUD = y $CRYSTAL =
$baud = 1200                'prueba y Ejemplo: 1200 baudios
$crystal = 12000000         '12 MHz

Dim V As Byte , B1 As Byte
Dim C As Integer , D As Byte
Dim S As String * 15        'solo para uP que soporta XRAM

Input "Use esto para hacer una pregunta " , V
Input B1                    'omitir si no sirve
Input "Entre integer " , C
```

```
Print C
Inputhex "Entre numero hex (4 bytes) " , C
Print C
Inputhex "Entre byte en hex(2 bytes) " , D
Print D
Input "Mas variables " , C , D
Print C ; " " ; D
Input C Noecho                'suprime el eco
Input "Entre su nombre " , S
Print "Hola " ; S
Input S Noecho                'sin  eco
Print S
End
```

INSTR

Acción
Devuelve la posición de una subcadena dentro de una cadena.

Sintaxis
var = INSTR(start , string , substr)
var = INSTR(string , substr)

Nota

var	Variable numérica que se asignará con la posición de la subcadena en la cadena. Ingresa 0 cuando la subcadena no se encuentra.
Start	Un parámetro numérico optativo que puede asignarse con la primera posición dónde debe buscar en la cadena. Por defecto (cuando no es usado) se busca en la cadena entera iniciando la búsqueda en la posición 1.
String	La cadena para buscar.
Substr	La cadena objeto de la búsqueda.

De momento INSTR () sólo trabaja con cadenas internas.
El soporte para cadenas externas también se agregará.

Diferencia con QB
Ninguna constante puede usarse para cadenas y subcadenas.

Ver también

Ejemplo:
```
Dim S As String * 10 , Z As String * 5
Dim bP As Byte
s = "This is a test"
Z = "is"
bP = Instr(s,z) : Print bP          'debe imprimir 3
bP = Instr(4,s,z) : Print bP        'debe imprimir 6
End
```

LCASE

Acción
Convierte una cadena de texto en mayúsculas (**UCASE**) ó minúsculas (**LCASE**).

Sintaxis
dest = LCASE(source)
dest = UCASE(source)

Nota

dest	La variable de la cadena a la que asignará el texto en mayúsculas ó minúsculas de la cadena fuente.
source	La cadena fuente. La cadena original quedará inalterada.

Ver también
UCASE

Ejemplo:
```
Dim s As String * 12 , Z As String * 12

INPUT "Hello " , s              'asigna la cadena
s = LCASE(s)                    'convierte a minúsculas

Print s                         'imprime la cadena

s = UCASE(s)                    'convierte mayúsculas

Print s                         'imprime la cadena
```

LCD

Acción
Envía una constante o bien una variable al display LCD.

Sintaxis
LCD x

Nota

x	Variable o constante a visualizar.

Pueden ser mostradas más variables, separándolas entre ellas con la señal **;** en el siguiente modo:
LCD a **;** b1**;** constante

La instrucción LCD se comporta como PRINT.

Ver también
LCDHEX , $LCD CONFIG LCD

Ejemplo:
```
'-----------------------------------------------------------
' (c) 1997,1998 MCS Electronics
'-----------------------------------------------------------
' file: LCD.BAS
' demo: LCD, CLS, LOWERLINE, SHIFTLCD, SHIFTCURSOR, HOME
' CURSOR, DISPLAY
'-----------------------------------------------------------
Dim A As Byte
Config Lcd = 16 * 2     'configura el tipo de lcd

'otras opciones son 16 * 4 y 20 * 4, 20 * 2
'Omitiendo la configuración se asume por defecto 16 * 2
'$LCD = para utilizar el visualizador LCD con 8 bits sobre el bus de datos
' solo para uP con RAM y/o ROM externa

Cls                     'clear (limpia) del display LCD
Lcd " Hola mundo "      'muestra esto en la primera línea
Wait 1
Lowerline               'selecciona la línea inferior
Wait 1
Lcd "SE desplaza"       'y muestra esto
Wait 1
For A = 1 To 10
Shiftlcd Right          'desplaza el texto a la derecha
Wait 1                  'espera un segundo
Next
For A = 1 To 10
Shiftlcd Left           'desplaza el texto a la izquierda
Wait 1                  'espera un segundo
Next
Locate 2 , 1            'posiciona el cursor
Lcd "*"                 'muestra esto
Wait 1                  'espera un segundo
Shiftcursor Right       'desplaza el cursor
Lcd "@"                 'muestra esto
```

```
Wait 1                    'espera un segundo
Home Upper                'selecciona la linea 1 y se sitúa al inicio
Lcd "Reemplaza"           'sustituye el texto precedente
Wait 1                    'espera un segundo
Cursor Off Noblink        'esconde el cursor
Wait 1                    'espera un segundo
Cursor On Blink           'muestra el cursor
Wait 1                    'espera un segundo
Display Off               'apaga el display LCD
Wait 1                    'espera un segundo
Display On                'enciende el display LCD


'----------------- NUEVO soporte para LCD a 4 líneas ----------------
Thirdline
Lcd "Linea 3"
Fourthline
Lcd "Linea 4"
Home Third                'al inicio de la tercera línea
Home Fourth
Home F                    'es suficiente también la sola inicial
Locate 4 , 1 : Lcd "Linea 4"
Wait 1
'Ahora permite configurar un carácter especial
'el primer número es el número de carácter (0-7)
'Los otros números son los valores de las filas
'Use las herramientes de LCD para insertar estas líneas
Deflcdchar 0 , 31 , 17 , 17 , 17 , 17 , 17 , 31 , 0  'cambia ? con (0÷7)
Deflcdchar 1 , 16 , 16 , 16 , 16 , 16 , 16 , 16 , 31 'cambia ? con (0÷7)
Cls                       'cls es necesario después de Deflcdchar
Lcd Chr(0) ; Chr(1)       'visualiza el carácter especial


'----------------- Ahora usa una rutina interna -------------------
Acc = 1                   'valor en ACC
Call Write_lcd            'transfiere al display LCD
End
```

LCDHEX

Acción
Envía una variable en formato hexadecimal al visualizador LCD.

Sintaxis
LCDHEX var

Nota

var	variable a visualizar

var1 : Byte, Integer, Word, Long, Single, Constant.
Son aplicables las mismas reglas definidas por PRINTHEX.

Ver también
LCD

Ejemplo:
```
Dim a As Byte
a = 255
LCD a
Lowerline
LCDHEX a
End
```

LEFT()

Acción
Devuelve el número de carácteres especificado de una cadena, partiendo de la izquierda.

Sintaxis
var = **Left(**var1 , n **)**

Nota

var	Cadena a la cual será asignado el resultado.
Var1	Cadena de origen sobre la que operar.
n	Numero de carácteres a extraer de la cadena.

n : Byte, Integer, Word, Long, Constant.
Para las operaciones con variables de tipo String, todas las variables tienen que ser mismo tipo:
interna o externa.

Ejemplo:
```
Dim s As XRAM String * 15, z As XRAM String * 15
s = "ABCDEFG"
z = Left(s,5)
Print z 'ABCDE
End
```

LEN

Acción
Devuelve la longitud de una cadena..

Sintaxis
var = LEN(string)

Nota

var	Variabile numerica a la que asignará el valor de la longitud de la cadena.
string	Cadena a la cual se desea calcular la longitud.

Ejemplo:
```
Dim S As String * 12
Dim A As Byte
S = "test"
A = Len(s)
Print A           'Visualiza 4
```

LOAD

Acción
Carga el valor de auto-recarga del TIMER especificado.

Sintaxis
LOAD TIMER , value

Nota

TIMER	TIMER0, TIMER1 o TIMER2.
Value	Variable o valor de carga.

Cuando se utiliza la instrucción **ON TIMERx** con el **TIMER/COUNTER** en modalidad **2**, es posible precisar a cuál intervalo tiene que ser generada la interrupción.

El valor puede ser comprendido entre 1÷255 para **TIMER0** y **TIMER1**, mientras para el **TIMER2** es posible precisar valores entre 1÷65535.

La instrucción **LOAD** calcula el valor correcto de reload (recarga) en el siguiente modo:
TLx = THx = (256-valor)

Para **TIMER2**: RCAP2L = RCAP2H = (65536 - valor)

La instrucción **LOAD** no debe ser utilizada para asignar o leer un valor de los timers/counters.
Para este objetivo tiene que ser empleada la instrucción **COUNTERx**.
Ver el Hardware suplementario para mayores detalles.

Ejemplo:
```
LOAD TIMER0, 100        'carga TIMER0 con 100

Genera el código siguiente :
Mov tl0,#h'9C
Mov th0,#h'9C
LOAD TIMER2, 1000
Will generate:
Mov RCAP2L,#24
Mov RCAP2H,#252
```

LOCATE

Acción
Mueve el cursor del visualizador LCD a la localización especificada.

Sintaxis
LOCATE y **,** x

Nota

x	Constante o variable relativa a la columna. (1-64*)
y	Constante o variable relativa a la línea (1 - 4*)

* depende del visualizador LCD conectado

Ver también
CONFIG LCD , LCD , HOME , CLS

Ejemplo:
```
LCD "Hola"
Locate 1,10
LCD "*"
```

LOOKUP

Acción
Devuelve el valor contenido en una tabla.

Sintaxis
var **=LOOKUP(** value**,** label **)**

Nota

var	Valor devuelto
value	Valor a usar como índice de la tabla.
label	Etiqueta en la cual está presente el dato

var : Byte, Integer, Word, Long, Single.
value : Byte, Integer, Word, Long, Costante.

Differenze da BASCOM LT
En **BASCOM LT**, la instrucción lookup trabaja exclusivamente con tablas compuestas por bytes.

En **BASCOM-8051**, es posible también emplear Integer, Word, Long y Single.

Ver también
LOOKUPSTR

Ejemplo:
```
DIM b1 As Byte , I As Integer
b1 = Lookup(1, dta)
Print b1                'Visualiza 2 (sobre base cero)
I = Lookup(0,DTA2)
End


DTA:
DATA 1,2,3,4,5

DTA2:                   'de tipo integer
1000% , 2000%
```

LOOKUPSTR

Acción
Devuelve una cadena contenida en una tabla.

Sintaxis
var =LOOKUPSTR(value, label)

Nota

var	Cadena devuelta
value	Valor a usar como índice de la tabla. El índice parte de 0, luego el primer elemento de la tabla es en el índice 0.
label	Etiqueta en la cual está presente el dato

value : Byte, Integer, Word, Long, Costante incluidos entre 0÷255

Ver también
LOOKUP

Ejemplo:
```
Dim s As String, idx As Byte
idx = 0 : s = LookupStr(idx,Sdata)
Print s                  'visualizará 'Esto'
End

Sdata:
Data "Esto" , "es" ,"un test"
```

LOW

Acción
Devuelve el byte menos significativo de una variable.

Sintaxis
var = LOW (s)

Nota

var	Variable a la cual será asignado el LSB de la variable S.
s	Variable fuente de la cual se extrae el byte LSB

Ver también
HIGH

Ejemplo:
```
Dim I As Integer , Z As Byte
I = &H1001
Z = Low(I)          'es 1
```

LOWW

Acción
Devuelve los dos bytes menos significativos (LSB) de una variable tipo **long** (largo).

Sintaxis
var = LOWW(s)

Nota

var	La variable que se asigna con los dos LSB de la variable s.
s	La variable fuente para conseguir los dos LSB.

Ver también
HIGHW HIGH LOW

Ejemplo:
```
Dim L As Integer , Z As Long

L = &H1001
Z = LowW(L)
```

LOWERLINE

Acción
Posiciona el cursor del display LCD al inicio de la línea más baja.

Sintaxis
LOWERLINE

Nota
-

Ver también
UPPERLINE , THIRDLINE , FOURTHLINE , HOME

Ejemplo:
```
LCD "Test"
LOWERLINE
LCD "Hola"
End
```

MakeBCD()

Acción
Convierte una variable en el correspondiente valor BCD.

Sintaxis
var1 = **MAKEBCD**(var2)

Nota

var1	Variable a la cual se asignará el valor convertido.
Var2	Variable conteniendo el valor decimal original.

Cuando se hace empleo del **reloj en tiempo real** por el bus I2C es necesario adaptar el tamaño de los datos del tipo DECIMAL al BCD y esta instrucción sirve para una fácil conversión.
Para imprimir ó visualizar el valor BCD de una variable tiene que ser utilizada la instrucción BCD ().

Ver también
MAKEDEC BCD()

Ejemplo:
```
Dim a As Byte
a = 65
LCD a
Lowerline
LCD BCD(a)
a = MakeBCD(a)
LCD " " ; a
End
```

MakeINT()

Acción
Convierte 2 bytes en un individual Word o Integer.

Sintaxis
varn = MAKEINT(LSB , MSB)

Nota

varn	Variable a la cual será asignado el valor convertido.
LSB	Variable o constante con el byte menos significativo (LSB).
MSB	Variable o constante con el byte más significativo (MSB).

El código es equivalente a:
varn = (256 * MSB) + LSB

Ver también
MAKEDEC BCD()

Ejemplo:
```
Dim a As Integer , I As Integer
a = 2
I = MakeINT(a , 1) 'I = (1 * 256) + 2 = 258
End
```

MakeDEC()

Acción
Convierte una variable de tipo Byte o Integer/Word en formado BCD con el correspondiente valor DECIMAL.

Sintaxis
var1 = **MAKEDEC**(var2)

Nota

var1	Variable a la cual asignará el valor convertido.
var2	Variable original conteniendo el valor en BCD.

Cuando se hace empleo del **reloj en tiempo real** por el bus I2C es necesario readaptar el tamaño de los datos del tipo BCD a DECIMAL y esta instrucción sirve para una fácil conversión.

Ver también
MAKEBCD

Ejemplo:
```
Dim a As Byte
a = 65
LCD a
Lowerline
LCD BCD(a)
a = MakeDEC(a)
LCD " " ; a
End
```

MAX

Acción
Devuelve el valor más alto de una serie, matriz (**array**).

Sintaxis
var = MAX(ar(1))

Nota

var	Variable numérica que se asignará con el valor más alto de la serie.
ar()	El primer elemento de la serie a devolver el valor más alto.

Actualmente MAX() sólo trabaja con formato en BYTE.
El soporte para otros tipos de datos también se agregará.

Ver también
MIN

Ejemplo:
```
Dim ar(10) As Byte
Dim bP As Byte

For bP = 1 to 10
  ar(bP) = bP
Next

bP = Max(ar(1))
Print bP              'debe de imprimir 10
End
```

MID()

Acción

La instrucción MID restituye parte de una cadena **(substring).**
La instrucción MID puede ser usada para sustituir parte de una cadena

Sintaxis

var = **MID(**var1 ,st [, l] **)**
MID(var ,st [, l] **)** = var1

Nota

var	Cadena a la cual será asignado el resultado.
Var1	Cadena de orígen.
st	Posición de inicio de la estracción / sustitución.
l	Número de carácteres a retirar o cambiar.

Las cadenas a operar tienen que ser del mismo tipo **(interna o externa).**

Ver también

LEFT , RIGHT

Ejemplo:

```
Dim s As XRAM String * 15, z As XRAM String * 15
s = "ABCDEFG"
z = Mid(s,2,3)
Print z                 'BCD

z="12345"
Mid(s,2,2) = z
Print s                 'A12DEFG
End
```

MIN

Ación
Devuelve el valor más bajo de una serie, matriz (**array**).

Sintaxis
var = MIN(ar(1))

Nota

Var	Variable numérica que se asignará con el valor más bajo de la serie.
ar()	El primer elemento de la serie para devolver el valor más bajo.

Actualmente MIN() sólo trabaja con formato en BYTE.
El soporte para otros tipos de datos también se agregará.

Ver también
MAX

Ejemplo:
```
Dim ar(10) As Byte
Dim bP As Byte

For bP = 1 to 10
  ar(bP) = bP
Next

bP = Min(ar(1))
Print bP              'debe de imprimir 1
End
```

MOD

Acción
Devuelve el módulo (**resto**) de una división.

Sintaxis
ret = var1 **MOD** var2

Nota

ret	Variable a la cual sera asignado el módulo (resto).
var1	Variable a dividir (dividendo).
var2	Variable por la cual se dividirá (divisor).

Cuando se divide un número entero, el resto que queda del resultado entre el dividendo y el divisor será el asignado a la variable ret.

Ejemplo:
```
a = 10 MOD 3       'divide 10 entre 3
PRINT a            'visualiza el resto (1)
```

NEXT

Acción
Termina una estructura FOR .. NEXT.

Sintaxis
NEXT [var]

Nota

var	Variable índice utilizada como contador en una estructura **FOR-NEXT** y especificado conjuntamente a **FOR**. Var es opcional y no indispensable.

Cada instrucción **FOR** tiene que ser acabada sucesivamente con un **NEXT**.

Ver también
FOR

Ejemplo:
```
y = 10                  'Ajusta y = 10
FOR a = 1 TO 10         'repite 10 vueltas
      FOR x = y TO 1    'también éste
         PRINT x ; a    'visualiza el valor de x y a
      NEXT              'x siguiente (cuenta atrás)
NEXT a                  'a siguiente (cuenta adelante)
END
```

ON Interrupt

Acción
Ejecuta una subrutina cuando detecta la interrupción especificada.

Sintaxis
ON interrupt label [NOSAVE]

Nota

interrupt	INT0, INT1, SERIAL, TIMER0 ,TIMER1 o TIMER2. Pueden encontrarse chips con interrupciones específicas bajo el soporte del microprocesador..
Label	Etiqueta a la cual saltrá la ejecución del programa al detectar la ionterrupción.
NOSAVE	Especificando NOSAVE, ningún registro será salvado y restablecido en la rutina de interrupción. Usando esta opción nos aseguramos de cumplir separadamente esta operacción sobre los registros utilizados.

Una rutina de interrupción tiene que ser acabada con la instrucción RETURN.
Sólo una instrucción RETURN puede estar presente en una rutina de interrupción ya que el compilador provee a restablecer los registros y genera un RETI cada vez que encuentra una instrucción RETURN en la rutina de interrupción.

TIMER1 no puede ser utilizado al mismo tiempo a las rutinas de gestión de la comunicacción serie ya que esta última hace empleo del TIMER1 para la generacción del BAUDRATE.

Empleando las interrupciones INT0 ó INT1 es posible precisar cual condición producirá la interrupción. Es posible emplear la instrucción Set/reset sobre el registro TCON para definir el comportamiento deseado:

SET TCON.0	trigger INT0 sobre la transición.
RESET TCON.0	trigger INT0 a nivel bajo (0).
SET TCON.2	trigger INT1 sobre la transición.
RESET TCON.2	trigger INT1 a nivel bajo (0).

Ver Hardware para mayores detalles

Ejemplo:
```
ENABLE INTERRUPTS
ENABLE INT0              'Habilita la interrupción
ON INT0 Label2 nosave    'Cuando INT0 salta a la etiqueta label2
DO                       'loop continuo
LOOP
END
Label2:
PRINT " Ha ocurrido una interrupción por hardware!" 'visualiza el mensaje
RETURN
```

ON Value

Acción
Salta a una o más etiquetas especificadas en función del valor de una variable.

Sintaxis
ON var [**GOTO**] [**GOSUB**] label1 **[**, label2 **]**

Nota

var	Variable numérica de analizar. Esta también puede ser un SFR como P1.
label1, label2	Etiquetas a saltar en relacción al valor de la variable var.

La definición de los valores parte del número 0, por lo tanto cuando var=0 la ejecución del programa salta a la primera etiqueta especificada.

Ver también

Ejemplo:
```
x = 2                           'asigna la variable para la interrupción
ON x GOSUB lbl1, lbl2,lbl3      'salta a la etiqueta lbl3
x=0
ON x GOTO lbl1, lbl2 , lbl3
END

lbl3:
PRINT " lbl3"
RETURN

Lbl1:


Lbl2:
```

OPEN – CLOSE

Acción
Abre y cierra un dispositivo.

Sintaxis
OPEN "device" for MODE As #channel
CLOSE #channel

Nota

device	Son soportados dos dispositivos por hardware: COM1 Y COM2. Con el software UART, es necesario precisar el pin del puerto y el baudrate. COM3.0: 9600 usará el PORT 3.0 a 9600 baud.
MODE	Para COM1 y COM2 puede ser especificado BINARY, INPUT o OUTPUT, mientras que para el software UART, tiene que ser especificada la dirección del pin INPUT o OUTPUT.
channel	Numero del canal a abrir. Debe de ser una constante positiva.

Ya que existen microprocesadores como el 80537, que disponen de 2 canales seriales a bordo, el compilador tiene que ser informado sobre la puerta que se quiere usar. Ésta es la razón por la que existe la instrucción **OPEN**. Si existe una sola puerta serial a bordo del microprocesador no es necesario precisar cual se quiere usar y la instrucción **OPEN** puede ser omitida.

Las instrucciones unidas a los devices seriales son **PRINT, PRINTHEX, INPUT e INPUTHEX.**
Cada dispositivo abiertos tiene que ser cerrado sucesivamente a través de la instrucción **CLOSE #channel**. Naturalmente especificando el mismo número por channel.

El software **UART** sólo soporta las instrucciones **PUT y GET** para mandar y recibir datos.
COM1: Y COM2: son puertos hardware y pueden ser empleados con **PRINT** etcétera.

Ver también

Ejemplo: 1
```
REM Sólo trabaja con el 80517 o 80537
CONFIG BAUD1 = 9600                 'baudrate serial 1
OPEN "COM2:" FOR BINARY AS #1       'abre el puerto
PRINT #1, "Hola"                    'imprime sobre el puerto serie 1
PRINT "Hola"                        'imprime sobre el puerto serie 0
CLOSE #1                            'cierra el canal 1
```

Ejemplo: 2
```
'Funciona con cualquier pin del puerto
```

```
Dim A As Byte , S As String * 16 , I As Byte , Dum As Byte

'un puerto de comunicación software toma el nombre del pin utilizado
'por Ejemplo: usando P3.0 se obtendrá "COM3.0:" (P es omitido)
'para los puertos de comunicación software tiene que ser especificado el
'baudrate
'Por lo tanto para 9600 baudios, el nombre del dispositivo es COM3.0:9600
'Cuando se usa el pin para transmitir el dispositivo tiene que ser abierto
'como OUTPUT
'Cuando se usa el pin para recibir el dispositivo tiene que ser abierto
'como INPUT
'Por ahora es sólo posible el mandar y recibir variables a través de PUT y
'GET.
'En el futuro también será completada la función con PRINT etcétera

Open "com3.1:9600" For Output As #1 'p3.1 generalmente es usado por TX por
                                    'lo tanto la verificación es fácil

Open "com3.0:9600" For Input As #2  'p3.0 generalmente es usado por RX por
                                    'lo tanto la verificación es fácil

S = "test this"                 'asigna una cadena a la variable
Dum = Len(s)                    'nota el largo de la cadena

For I = 1 To Dum                'para todos los carácteres de SX a DX
      A = Mid(s , I ,1)         'toma el carácter
      Put #1 , A                'lo envía al puerto de comunicación
Next

Do
      Get #2 , A                'toma el carácter del puerto
      Put #1 , A                'y lo escribe
      Print A                   'también sobre el canal normal
Loop

Close #1                        'cierra el dispositivo
Close #2
End
```

OUT

Acción
Envía un byte a un puerto hardware o bien a una dirección de memoria externa.

Sintaxis
OUT address, value

Nota

address	Dirección a la cual enviará el byte.
value	Variable o valor a enviar.

La instrucción OUT trabaja solo con microprocesadores que pueden direccionar memoria externa.

Ver también
INP

Ejemplo:
```
Dim a As Byte
OUT &H8000,1 'envia 1 a la dirección HEX 8000 sobre el bus de datos (d0÷d7)
END

Genera el código siguiente:
Mov A,#1
Mov dptr,#h'8000
Movx @dptr,a
```

P1,P3

Acción
P1 y P3 son registros especiales de funcion (SFR) que pueden ser gestionados como variables.

Sintaxis
Px = var
var = **Px**

Nota

x	Número de port (1 or 3). **P3.6 no puede ser empleado en un AT89C2051!**
Var	Variable a adquirir o asignar.

Otros microprocesadores pueden disponer de más puertos como **P0, P2, P4** etc.

Elegiendo oportunamente el **archivo.DAT** que define cuál microprocesador se está empleando será posible también usar estos puertos adicionales como variables.

En la práctica es posible utilizar cualquier **SFR** como una variable.
ACC = 0 por ejemplo: resetea el acumulador

Ver hardware para mayores detalles.

Ejemplo:
```
Dim a As Byte, b1 As Bit
a = P1                  'toma el valor del puerto 1
a = a OR 2              'lo manipula
P1 = a                  'y lo escribe al puerto 1
P1 = &B10010101         'usa la notación binaria
P1 = &HAF               'usa la notación hexadecimal
b1 = P1.1               'lee el pin 1.1
P1.1 = 0                'y lo pone a 0
```

PEEK()

Acción
Devuelve un byte memorizado en la memoria interna.

Sintaxis
var **= PEEK(** address **)**

Nota

var	Variable numérica a la que asignar el contenido de la celda apuntada de address.
address	Variable numérica o constante que define la dirección. (0÷255)

Ver también
POKE , CPEEK , INP , OUT

Ejemplo:
```
DIM a As Byte
a = Peek( 0 )     'devuelve el primer byte de la memoria interna (r0)
End
```

POKE

Acción
Escribe un byte en una localización de la memoria interna.

Sintaxis
POKE address , value

Nota

address	Variable numérica conteniendo la dirección de la celda de la memoria a escribir. (0÷255)
value	Valor a asignar. (0÷255)

Se tiene que poner la máxima atención en el empleo de la instrucción POKE, ya que la memoria interna es utilizada para memorizar las variables y una operación errónea puede llevar al funcionamiento defectuoso del programa.

Ver también
PEEK , CPEEK , INP , OUT

Ejemplo:
```
POKE 127, 1       'escribe 1 en la dirección 127
End
```

POWERDOWN

Acción
Fuerza al procesador en modo powerdown.

Sintaxis
POWERDOWN

Nota
La modalidad powerdown para completamente el reloj del sistema.
El único modo para reactivar el procesador es haciendo un reset.

Ver también
IDLE

Ejemplo:
```
POWERDOWN
```

PRINT

Acción
Envía, escribe sobre el puerto serie asíncrono (RS-232).

Sintaxis
PRINT var ; " constant"

Nota

var	Variable o constante a transmitir (print).

Para mandar más variables con la misma instrucción, utilizar punto y coma como separador **;**
Acabando una línea con punto y coma no será mandado el carácter de cambio de línea (**linefeed**).

La instrucción **PRINT** puede ser utilizada cuando existe una interfaz serie RS-232. Consultar el manual para informaciones sobre la realización de una interfaz RS-232.

La interfaz RS-232 puede ser unida a la puerta de comunicación serie de un PC. De este modo es posible emplear un emulador de terminal como aparato de salida. En BASCOM es integrado un emulador de terminal.

Ver también
PRINTHEX , INPUT , OPEN , CLOSE

Ejemplo:
```
'-----------------------------------------------------------
' (c) 1997,1998 MCS Electronics
'-----------------------------------------------------------
' file: PRINT.BAS
' demo: PRINT, PRINTHEX
'-----------------------------------------------------------
Rem El Integer puede asumir valores entre -32767 y 32768

Dim A As Byte , B1 As Byte , C As Integer
A = 1
Print "print variable a " ; A
Print 'nuova linea
Print "Text to print."        'costante a visualizar
B1 = 10
Printhex B1                   'visualiza en notación hex
C = &HA000                    'asigna el valor a c%
Printhex C                    'visualiza en notación hex
Print C                       'visualiza en notación decimal
C = -32000
Print C
Printhex C
End
```

PRINTBIN

Acción
Envía el contenido de una variable sobre el puerto serie.

Sintaxis
PRINTBIN var [,varn]

Nota

var	Variable conteniendo el valor a enviar por el puerto serie
varn	Variables a enviar. (opcional)

PRINTBIN equivale a **PRINT CHR**(var); pero de este modo pueden imprimirse matrices enteras (**array**).

Usando una variable tipo **Long**, por ejemplo:, seran enviados 4 bytes.

Ver también
INPUTBIN

Ejemplo:
```
Dim a(10) As Byte, c As Byte
For c = 1 To 10
      a(c) = a    'rellena la matriz (array)
Next
PRINTBIN a(1)     'visualiza el contenido
```

PRINTHEX

Acción
Envía una variable al puerto serie en formato hexadecimal.

Sintaxis
PRINTHEX var

Nota

var	Variable conteniendo el valor a enviar por el puerto serie.

Se aplican las mismas reglas válidas por PRINT.

La instrucción PRINTHEX puede ser utilizada cuando existe una interfaz serie RS-232.

Consultar el manual por informaciones sobre el realización de un interfaz RS-232.

La interfaz RS-232 puede ser unida a la puerta de comunicación serie de un PC.
De este modo es posible emplear un emulador de terminal como aparato de salida.

En BASCOM es integrado un emulador de terminal.

Ver también
PRINT , INPUTHEX

Ejemplo:
```
Dim x As Byte
INPUT x                 'espera ala entrada de una variable
PRINT x                 'la visualiza en formato decimal
PRINTHEX "Hex " ; x     'y luego en formado Hexadecimal
```

PRIORITY

Acción
Define los niveles de prioridad de las interrupciones.

Sintaxis
PRIORITY SET / RESET interrupt

Nota

SET	Activa la prioridad de la interupción al nivel más alto.
RESET	Activa la prioridad de la interrupcion al nivel más bajo.
Interrupt	La interrupción a la cual define la prioridad.

Las interrupciones son: **INT0, INT1, SERIAL, TIMER0, TIMER1 e TIMER2.**

Otros microprocesadores pueden tener interrupciones adicionales o diferentes.
Ver el hardware de los Microprocesadores para mayores detalles.

La interrupción INT0 tiene siempre la prioridad más alta.
Cuando ocurren más interrupciones al mismo tiempo, el orden siguiente será utilizado en la gestión de las llamadas.

Prioridad en las Interrupciones:

Interrupción	Prioridad
INT0	1 (más alta)
TIMER0	2
INT1	3
TIMER1	4
SERIAL	5 (más baja)

Ejemplo:
```
PRIORITY SET SERIAL             'int serie al nivel mas alto
ENABLE SERIAL                   'habilita int serie
ENABLE TIMER0                   'habilita int timer0
ENABLE INTERRUPTS               'activa la gestion de las interrupciones
ON SERIAL label                 'salta a la etiqueta en caso de int serie
DO                              'bucle continuo
LOOP

Label:                          'cuando hay int serie salta a esta etiqueta
PRINT " Interrupción Serie"     'visualiza un mensaje
RETURN                          'regresa de la interrupción
```

PUT

Ación
Envía un byte al software UART.

Sintaxis
PUT #channel , var

Nota

channel	Constante numérica positiva que se refiere al canal abierto.
var	Una variable o constante que contiene el valor a envíar al software UART.

Tenga en cuenta que el canal debe abrirse con la instrucción OPEN.
También, la instrucción CLOSE, debe ser la última en su programa. Por favor vea el comentario en la instrucción OPEN y el ejemplo en esta misma página.

Ver también
GET

Ejemplo
```
'En este ejemplo se usan los pins asignados normalmente para comunicación
Open "com3.1:9600" For Output As #1          'p3.1 normalmente se usa para Tx
Open "com3.0:9600" For Input As #2           'p3.0 normalmente se usa para Rx

S = "test this"                 'asignamos una cadena
Dum = Len(s)                    'asignamos a Dum la longitud de la cadena

For I = 1 To Dum                'para todos los caracteres desde la izquierda a la derecha
  A = Mid(s , I , 1)            'asignamos a A el caracter
  Put #1 , A                    'escribe el valor de A por el puerto de comunicaciones
Next

Do
  Get #2 , A                    'lee un caracter desde el puerto de comunicaciones
  Put #1 , A                    'lo envía por el puerto
  Print A                       'usa el canal normal
Loop

Close #1                        ' finaliza y cierra el dispositivo
Close #2
End
```

READ

Acción
Lee un valor definido en DATA y lo asigna a la variable.

Sintaxis
READ var

Nota

var	Variable a la cual será asignado el valor leido.

Differenze da QB
Es indispensable que las variables sean del mismo tipo de los valores memorizado con **DATA**.

Ver también
DATA , RESTORE

Ejemplo:
```
Dim A As Byte, I As Byte, C As Integer, S As XRAM String * 10
RESTORE dta

FOR a = 1 TO 3
      READ i : PRINT i
NEXT

RESTORE DTA2
READ C : PRINT C
READ C : PRINT C
Restore dta3 : Read s : Print s
END

dta:
Data 5,10,15

dta2:
Data 1000%, -2000%

dta3:
Data "Hola"
```

REM

Acción
Inserta un comentario.

Sintaxis
REM ó **'**

Nota
Un programa comentado resulta más comprensible.

Para insertar un comentario usar **REM** o bien **'** seguido por el texto del comentario.

Todas las instrucciones puestas después de **REM** o **'** son consideradas comentarios y no serán compiladas.

Ahora es posible insertar como bloques de texto comentado:
'(principio de un bloque de comentario
print "Esto no será compilado"
') fin del bloque de comentario

El símbolo inicial garantiza la compatibilidad con QB.

Ejemplo:
```
REM TEST.BAS version 1.00
PRINT a     ' " Esto es un comentario : PRINT " Hola "
                ^--- esto no será compilado!
```

RESET

Acción
Resetea (pone a cero) un bit del puerto (P1.x, P3.x) o bien una variable interna de tipo Bit/Byte/Integer/Word.

Sintaxis
RESET bit
RESET var.x

Nota

bit	Puede ser P1.x, P3.x o cualquier bit de la variable dónde x=0÷7.
var	Puede ser una variable de tipo Byte, Integer o Word.
x	Constante indicando el bit de la variable a resetear. (0÷7) para bytes y (0÷15) para Integer/Word

Ver también
SET

Ejemplo:
```
Dim b1 As Bit, b2 As Byte, I As Integer

RESET P1.3          'reset del bit 3 del puerto 1
RESET b1            'variable de tipo bit
RESET b2.0          'reset del bit 0 de la variable tipo byte b2
RESET I.15          'reset del bit MS de I
```

RESTORE

Acción
Permite releer los datos leidos con READ. Borra el indicador de DATA.

Sintaxis
RESTORE label

Nota

label	Etiqueta que es posicionada con la instrucción DATA.

Ver también
DATA , READ

Ejemplo:
```
La variable Integer debe de terminar con el signo % (Integer : <0 or >255)

DIM a AS BYTE, I AS BYTE
RESTORE dta
FOR a = 1 TO 3
      READ a : PRINT a
NEXT

RESTORE DTA2

READ I : PRINT I
READ I : PRINT I
END

DTA1:
Data 5, 10, 100

DTA2:
Data -1%, 1000%
```

RETURN

Acción
Regresa de una subrutina.

Sintaxis
RETURN

Nota
Cada Subrutina tiene que ser acabada con la instrucción **RETURN**.

También las subrutinas de Interrupciones tienen que ser acabadas con la instrucción **RETURN**.

Ver también
GOSUB

Ejemplo:
```
y = 2
GOSUB Pr                'salta a la subrutina
PRINT result            'visualiza el resultado
END                     'fin del programa

Pr:                     'etiqueta de inicio de la subrutina
result = 5 * y
result = result + 100   'Para demostrar algo
RETURN                  'Regresa
```

RIGHT()

Acción
Devuelve el número de carácteres especificado de una cadena, partiendo de derecha.

Sintaxis
var = **RIGHT(**var1 ,st **)**

Nota

Var	Cadena a la cual será asignado el resultado.
Var1	Cadena de origen sobre el que operar.
St	Posición desde la cual se iniciará la estracción.

Todas las cadenas tienen que ser del mismo tipo: internas ó externas.

Ver también
LEFT , MID

Ejemplo:
```
Dim s As XRAM String * 15, z As XRAM String * 15
s = "ABCDEFG"
z = Right(s,2)
Print z                 'Visualiza  --->   FG
End
```

RND

Acción
Devuelve un número aleatorio.

Sintaxis
var = **RND(**limit**)**

Nota

Limit	número máximo que se asignará al número de selección aleatoria.

La función RND () usa 2 bytes internos para guardar el valor de lo calculado aleatoriamente.

Ver también

Ejemplo:
```
'-----------------------------------------------------------
'                     (c) 2000 MCS Electronics
'                           RND.BAS
'-----------------------------------------------------------
Dim W As Word

Do
      'Captura un número aleatorio limitado a un valor máximo de 100.
  W = Rnd(100)
  Print W
Loop
End
```

ROTATE

Acción
Desplaza todos los bits una posición a la derecha ó a la izquierda.

Sintaxis
ROTATE var **, LEFT/RIGHT** [**,** shifts]

Nota

var	Variable de tipo Byte, Integer/Word o Long.
shifts	Número de desplazamientos a efectuar.

Note que el flag de acarreo entra en el byte **LSB** o **MSB** dependendiendo de la dirección del desplazamiento.
El resultado es idéntico al producido con una instrucción assembler **RLC** o bien **RRC**.
Si el comportamiento es indeseado, borrar el bit de acarreo antes de la instrucción de desplazamiento con la instrucción **CLR C** (Clear Carry).

Ver también
SHIFTIN , SHIFTOUT

Ejemplo:
```
Dim a As Byte
a = 128
ROTATE a, LEFT , 2      'Rota dos bits
Print a                 'Visualiza 1
```

Código generado:
```
Mov R7,#2
Mov R0,#h'21
Mov a,@r0
Rlc a
Djnz r7,*-1
Mov @r0,a
```

SELECT

Acción
Ejecuta uno de varios bloques de instrucciones, según el valor de una expresión.

Sintaxis
SELECT CASE var
CASE test1 **:** statements
[**CASE** test2 **:** statements]
CASE ELSE : statements
END SELECT

Nota

var	Variable a analizar.
Test1	Valor para comparar con la variable.
Test2	Valor para comparar con la variable.

Ver también
-

Ejemplo:
```
Dim b2 As Byte
SELECT CASE b2                  'set bit 1 del puerto 1
      CASE 2 : PRINT "2"
      CASE 4 : PRINT "4"
      CASE IS >5 : PRINT ">5"     ' una prueba solicita la palabra clave IS
      CASE ELSE
END SELECT
END
```

SET

Acción
Setea (pone a 1) un bit del puerto (P1.x,P3.x) o bien una variable interna de tipo Bit/Byte/Integer/Word.

Sintaxis
SET bit
SET var.x

Nota

bit	Puede ser P1.x, P3.x o cualquier bit de variable dónde x=0÷7.
var	Puede ser una variable de tipo Byte, Integer o Word.
x	Bit de una variable (0÷7) a setear. (0÷15 para Integer/Word)

Ver también
RESET

Ejemplo:
```
Dim b1 As Bit, b2 As Byte, c As Word
SET P1.1    'set bit 1 del puerto 1
SET b1      'variable de tipo bit
SET b2.1    'set del bit 1 de la variable b2
SET C.15    'set del bit mas alto del la variable tipo Word
```

SHIFT

Action
Shifts all bits one place to the left or right.

Syntax
SHIFT var , LEFT/RIGHT [, shifts]

Remarks

Var Byte, Integer/Word or Long variable.
Shifts The number of shifts to perform.
The SHIFT statements shifts all bits to the left or right and so for a byte after 8 shifts, the byte will be zero.

See also
SHIFTIN , SHIFTOUT ROTATE

Example
```
Dim a as Word
a = 128
SHIFT a, LEFT , 1
Print a          '256
```

SHIFTCURSOR

Acción
Desplaza el cursor del display LCD a la izquierda o derecha una posición.

Sintaxis
SHIFTCURSOR LEFT / RIGHT

Ver también
SHIFTLCD

Ejemplo:
```
LCD "Hello"
SHIFTCURSOR LEFT
End
```

SHIFTIN y SHIFTOUT

Acción
Envía o recibe una variable por un pin del puerto, en forma de un tren de bits seriales con envío síncrono a una señal de clock.

Sintaxis
SHIFTIN pin , pclock , var , option
SHIFTOUT pin , pclock , var , option

Nota

pin	Pin del puerto a utilizar como input/output.
pclock	Pin del puerto que generará la señal del clock.
var	Variable asignada a la operación de desplazamiento.
Option	Opciones posibles : 0 - MSB se desplaza in/out primero cuando el clock está a nivel bajo 1 - MSB se desplaza in/out primero cuando clock está a nivel alto 2 - LSB se desplaza in/out primero cuando clock está a nivel bajo 3 - LSB se desplaza in/out primero cuando cock está a nivel alto Añadiendo la opción 4 a SHIFTIN se entiende que el clock para el desplazamiento sea generado externamente.

En relación al tipo de variable serán necesarios más o menos shift para completar el envío.
Por un byte necesitarán 8 shifts mientras 16 shifts servirán para un Integer.

Ver también

Ejemplo:
```
Dim a As Byte
SHIFTIN P1.0 , P1.1 , a , 0
SHIFTOUT P1.2 , P1.1 , a , 0

Por Ejemplo: de SHIFTIN éste es el código generado:
Setb P1.1
Mov R0,#h'21
Mov r2,#h'01
__UNQLBL1:
Mov r3,#8
__UNQLBL2:
Clr P1.1
Nop
Nop
Mov c,P1.0
Rlc a
Setb P1.1
Nop
Nop
Djnz r3,__UNQLBL2
Mov @r0,a
Dec r0
```

```
Djnz r2,__UNQLBL1
```

El código generado depende, naturalmente, de los parámetros.

Para desplazamientos con una señal externa de clock:

```
SHIFTIN P1.0, P1.1, a, 4  'añadir 4 para señal de reloj externa.
```

Código generado:

```
Mov R0,#h'21
Mov r2,#h'01
__UNQLBL1:
Mov r3,#8
__UNQLBL2:
Jnb P1.1,*+0
Mov c,P1.0
Rlc a
Jb P1.1,*+0
Djnz r3,__UNQLBL2
Mov @r0,a
Dec r0
Djnz r2,__UNQLBL1
```

SHIFTLCD

Acción
Desplaza el display LCD a la derecha ó a la izquierda una posición.

Sintaxis
SHIFTLCD LEFT / RIGHT

Nota

Ver también
SHIFTCURSOR

Ejemplo:
```
LCD "Texto muy largo"
SHIFTLCD LEFT
Wait 1
SHIFTLCD RIGHT
End
```

SOUND

Acción
Envía impulsos a un pin del puerto.

Sintaxis
SOUND pin, duration, frequency

Nota

pin	Cualquier pin del puerto como P1.0, etc.
duration	Número de impulsos a enviar. Byte, Integer/Word o constante comprendida entre 1÷ 32768.
Frequency	Tiempo por el cual el pin es forzado alto y bajo.

Conectando un altavoz o bien un timbre eléctrico a un pin del puerto (ver hardware), es posible utilizar la instrucción SOUND para producir tonos.
El pin del puerto es conmutado alto y bajo por el número de microsegundos especificado en frequency (frecuencia).
Este bucle es ejecutado el número de veces especificado en duration (duración).

Ver también
-

Ejemplo:
```
SOUND P1.1 , 10000, 10          'Produce un BEEP
End
```

SPACE()

Acción
Devuelve una cadena de espacios.

Sintaxis
var = **SPACE(**x **)**

Nota

x	Número de carácteres de espaciado.
Var	Cadena a la cual será asignado el resultado

Usando 0 por x, el resultado es una cadena de 255 bytes ya que no es efectuado ningún control sobre asignaciones de cadenas con largo cero.

Ejemplo:
```
Dim s As XRAM String * 15, z As XRAM String * 15
s = Space(5)
Print " {" ;s ; " }" '{ }
Dim A As Byte
A = 3
S = Space(a)


Código generado por las ultimas 2 líneas :
;---------- rutina de la librería -----------
_sStr_String:
Mov @r1,a
Inc r1
Djnz r2,_sStr_String
Clr a
Mov @r1,a
Ret
;---------------------------------------
Mov R1,#h'22 ; localización de la cadena
Mov R2,h'21 ;  número de espacios
Mov a,#32
Acall _sStr_String
```

SPIIN

Acción
Lee un valor desde el bus SPI.

Sintaxis
SPIIN var, bytes

Nota

var	Variable a la cual asignará el valor leido desde el bus SPI.
bytes	Numero de bytes a leer.

Ver también
SPIOUT , CONFIG SPI

Ejemplo:
```
Dim a(10) As Byte
CONFIG SPI = SOFT, DIN = P1.0, DOUT = P1.1, CS=P1.2, CLK = P1.3

SPIIN a(1) , 4    'lee 4 bytes
```

SPIOUT

Acción
Envía el valor de una variable sobre el bus SPI.

Sintaxis
SPIOUT var , bytes

Nota

var	Variable que contiene el valor que envía sobre el bus SPI.
bytes	Número de bytes a enviar.

Ver también
SPIIN , CONFIG SPI

Ejemplo:
```
CONFIG SPI = SOFT, DIN = P1.0, DOUT = P1.1, CS=P1.2, CLK = P1.3
Dim a(10) As Byte , X As Byte

SPIOUT a(1) , 5   'envia 5 bytes
SPIOUT X , 1      'envia 1 byte
```

START

Acción
Pone en marcha el timer/counter especificado.

Sintaxis
START timer

Nota

timer	TIMER0, TIMER1, TIMER2, COUNTER0 o COUNTER1.

Para que sea producida una interrupción es necesario primero poner en marcha un timer/counter (cuando el disparo externo es inhabilitado).

TIMER0 y **COUNTER0** son el mismo dispositivo.

Ver también
STOP TIMERx

Ejemplo:
```
ON TIMER0 label2
LOAD TIMER0, 100
START TIMER0
DO          'inicia el bucle de loop
LOOP        'loop continuo

label2:     'aquí continúa el programa cuando se produzca una interrupción.
.
.
.
RETURN
```

STOP

Acción
Para la ejecución de un programa.

Sintaxis
STOP

Nota
Para acabar un programa puede ser también empleada la instrucción END.
Cuando se alcanza una instrucción END o STOP es producido un BUCLE sin fin.

Ejemplo:
```
var=5
PRINT var   'visualiza 5
STOP        'La ejecución del programa es detenida.
```

STOP TIMERx

Acción
Para el timer/counter especificado.

Sintaxis
STOP timer

Nota

timer	TIMER0, TIMER1, TIMER2, COUNTER0 or COUNTER1.

Para suspender la ejecución de una interrupción, también es posible parando el temporizador.
TIMER0 y COUNTER0 son el mismo dispositivo.

Ver también
START TIMERx

Ejemplo:
```
'-----------------------------------------------------------------
' (c) 1997,1998 MCS Electronics
' file: TIMER0.BAS
' demo: ON TIMER0
' *   TIMER1 es usado como generador de baudrate para RS-232
'-----------------------------------------------------------------
Dim Count As Byte , Gt As Byte
Config Timer0 = Timer , Gate = Internal , Mode = 2
'Timer0 = counter : timer0 opera como contador
'Gate = Internal : el control de disparo (gate) es interno
'Mode = 2 : 8-bit auto reload (autorecarga, por defecto)
On Timer0 Timer_0_int
Load Timer0 , 100       'cuando el temporizador alcanza 100 ocurre una
                        'interrupción
Enable Interrupts       'habilita el uso de las interrupciones
Enable Timer0           'habilita el timer
Rem Setting Of Priority (fija la prioridad)
Priority Set Timer0     'a la prioridad mas alta
Start Timer0            'activa el timer (lo pone en marcha)
Count = 0               'reset del contador
Do
Input "Number " , Gt
Print "You entered : " ; Gt
Loop Until Gt = 1       'bucle hasta que se pulse ESC
Stop Timer0
End
Rem The Interrupt Handler For The Timer0 Interrupt
Timer_0_int:
Inc Count
If Count = 2 Then
Print " A ocurrido una interrupción en Timer0 "
Count = 0
End If
Return
```

STR()

Acción
Devuelve una representación de un número en forma de cadena.

Sintaxis
var **= Str(** x **)**

Nota

var	Variable de tipo String (cadena de carácteres).
X	Variable numerica.

x : Byte, Integer, Word, Long, Single.
La cadena debe de ser lo suficientemente grande para contener el resultado.

Ver también
VAL

Differenze da QB
En QB STR() devuelve una cadena con un espacio en la cabecera (leading).
En BASCOM esto no ocurre.

Ejemplo:
```
Dim a As Byte, S As XRAM String * 10
a = 123
s = Str(a)
Print s
End
```

STRING()

Acción
Devuelve una cadena conteniendo el carácter ASCII **n** repetido **m** veces.

Sintaxis
var = **STRING(**m ,n **)**

Nota

var	Cadena a la cual será asignado el resultado.
n	Código ASCII que asignará a la cadena.
m	Número de carácteres idénticos para asignar.

No es posible utilizar el código ASCII cero, ya que es utilizado por el compilador para señalar el fin de una cadena.
Precisando cero por **m**, se conseguirá una cadena de 255 bytes ya que no es efectuado ningún control sobre asignaciones de cadenas con largo cero.
En caso de que fuera necesaria esta función infórmese en MCS Electronics.

Ver también
SPACE

Ejemplo:
```
Dim s As XRAM String * 15
s = String(5,65)
Print s 'AAAAA
End
```

SUB

Acción
Definisce una Subroutine.

Sintaxis
SUB Name[(var1)**]**

Nota

name	Nombre de la subrutine. Puede ser empleada cualquiera palabra no reservada.
var1	Nombre del parámetro.

Cada Subrutina tiene que ser acabada con la instrucción END SUB.
La declaración de una Subrutina tiene que ser efectuada antes de escribir el procedimiento relativo a la Subrutina.
El tipo y el nombre de los parámetros tienen que ser idénticos sea en la declaración sea en el procedimiento de la Subrutina.
Los parámetros son GLOBALES para toda la aplicación y tienen que ser dimensionadas con la instrucción DIM como cualquiera otra variable.
Este permite compartir los parámetros de un Subrutina con el programa principal, como lo ilustrado en el ejemplo siguiente:

```
Dim a As Byte, b1 As Byte, c As Byte      'dimensiona la variable
Declare Sub Test(a As Byte)             'declara la subrutina
a = 1 : b1 = 2: c = 3                   'asigna la variable
Print a ; b1 ; c                        'visualiza la variable
Call Test(b1)                           'llama la subrutina
Print a ;b1 ; c                         'visualiza la variable de nuevo
End

Sub Test(a As Byte)                     'inicio del procedimiento/subrutina
print a ; b1 ; c                        'visualiza la variable
End Sub
```

Ver también
CALL, DECLARE

Ejemplo:
-

SWAP

Acción
Intercambia los valores entre dos variables del mismo tipo.

Sintaxis
SWAP var1, var2

Nota

var1	Variable de tipo Bit, Byte, Integer o Word.
var2	Variable del mismo tipo que var1.

Después del cambio var1 contendrá el valor de var2 mientras var2 aquel de var1.

Ver también
-

Ejemplo:
```
Dim a As Integer, b1 As Integer
a = 1 : b1 = 2          'asigna dos integers
SWAP a, b1              'los cambia
PRINT a ; b1
```

THIRDLINE

Acción
Posiciona el cursor del display al inicio de la tercera línea.

Sintaxis
THIRDLINE

Nota
-

Ver también
UPPERLINE , LOWERLINE , FOURTHLINE

Ejemplo:
```
Dim a As Byte
a = 255
LCD a
Thirdline
LCD a
Upperline
End
```

TIMEOUT

Acción
Incluya el código para verificar por una interrupción cuando espera carácteres por el puerto en serie.

Sintaxis
TIMEOUT = var1

Nota

var1	Constante con rango desde 1- 65535.

Cada unidad esperará por 100 uS. Así que la interrupción máxima es de 6 segundos.
El uso de TIMEOUT lo puede utilizar con INPUT, INPUTHEX y declaraciones de INPUTBIN usadas en su programa.
Cuando ocurre timeout (ningún carácter recibido dentro del tiempo dado) un RETORNO es emulado para abortar la rutina. La variable ERR se usarà para indicar que ha occurrido una interrupción por timeout.
Nota que TIMER0 es usado y aquí no se usa. La opción de TIMEOUT usa 4 bytes de la memoria interna.
La aplicación actual es una posibilidad dado a tener una interrupción. Podría cambiarse en el futuro.

Ver también
TIMER0

Ejemplo:
```
Dim a as integer, b1 as integer
TIMEOUT = 60000
INPUT "Number ", b1           'espera un máximo de 6 segundos
PRINT  b1  ; ERR
End
```

UCASE

Acción
Convierte una cadena de texto en mayúsculas (**UCASE**) ó minúsculas (**LCASE**).

Sintaxis
dest = LCASE(source)
dest = UCASE(source)

Nota

dest	La variable de la cadena a la que asignará el texto en mayúsculas ó minúsculas de la cadena fuente.
source	La cadena fuente. La cadena original quedará inalterada.

Ver también
UCASE

Ejemplo:

```
Dim s As String * 12 , Z As String * 12

INPUT "Hello " , s  'asigna la cadena
s = LCASE(s)        'convierte a minúsculas

Print s             'imprime la cadena

s = UCASE(s)        'convierte mayúsculas

Print s             'imprime la cadena
```

UPPERLINE

Acción
Posiciona el cursor del display LCD al inicio de la línea más alta.

Sintaxis
UPPERLINE

Nota
-

Ver también
LOWERLINE THIRDLINE FOURTHLINE

Ejemplo:
```
Dim a As Byte
a = 255
LCD a
Lowerline
LCD a
Upperline
End
```

VAL()

Acción
Convierte un cadena representando un número en número.

Sintaxis
var **= Val(** s **)**

Nota

var	Variable numérica a la cual se le asignará el valor de s.
s	Variable de tipo cadena.

var : Byte, Integer, Word, Long, Single.

Ver también
STR

Ejemplo:
```
Dim a As Byte, s As XRAM String * 10
s = "123"
a = Val(s) 'convierte la cadena
Print a
End
```

VARPTR()

Acción
Devuelve la dirección de memoria en la cual está memorizada una variable.

Sintaxis
var = VARPTR(var2)

Nota

var	Variable a la cual se asignará la dirección de var2.
var2	Variabile a la que se solicita la dirección.

Ver también
PEEK POKE

Ejemplo:
```
Dim I As Integer , B1 As Byte
B1 = Varptr(I)


Codigo generado :
Mov h'23,#h'21
```

WAIT

Acción
Suspende la ejecución del programa por un tiempo predefinido.

Sintaxis
WAIT seconds

Nota

seconds	Número en espera.

El retraso es basado en la frecuencia de reloj de 12 Mhz.
Este mando no permite temporizaciones precisas.
En el caso se averiguaran interrupciones en el curso del retraso, éste será alargado el tiempo necesario a la ejecución de las interrupciones.

Ver también
DELAY

Ejemplo:
```
WAIT 3      'espera durante tres segundos
Print "*"   'han pasado los tres sgundos
```

WAITKEY

Acción
Espera hasta que sea recibido un carácter en el buffer serie.

Sintaxis
var = **WAITKEY**

Nota

var	

Variable al que asignar el valor ASCII del carácter recibido en el buffer serial.
var : Byte, Integer, Word, Long, String.

Ver también
INKEY

Ejemplo:
```
Dim A As Byte
A = Waitkey       'espera un carácter
Print A
```

WAITMS

Acción
Sospende l'esecuzione del programma per un tempo predefinito in millisecondi.

Sintaxis
WAITMS mS

Nota

mS	Número en milisegundos de espera. (1÷255)

El retraso es basado en un frecuencia de reloj de 12 Mhz.

Este mando no permite temporizaciones precisas.

En el caso de verificar interrupciones en el curso del retraso, éste será alargado el tiempo necesario a la ejecución de la interrupción.

Esta instrucción está prevista para el empleo con las instrucciones del bus I2C. Cuando se realiza una escritura sobre una EEPROM conectada a un bus I2C es necesario esperar 10 mS antes de la siguiente instrucción de escritura.

Ver también
DELAY WAIT

Ejemplo:
```
WAITMS 10   'espera 10 milisegundos
Print "*"
```

WHILE .. WEND

Acción
Ejecuta una serie de instrucciones siempre que la condición especificada tenga el estado verdadero.

Sintaxis
WHILE condition
instrucciones
.
.
WEND

Nota
Las instrucciones contenidas entre WHILE y WEND serán ejecutadas mientras que la condición especificada en WHILE sea verdadera.
Si la condición no es verdadera el programa continúa con la instrucción que sigue a WEND.

Ver también
DO .. LOOP

Ejemplo:
```
a = 1
WHILE a <= 10     'Ejecuta el bucle hasta que a >10
      PRINT a
      INC a
WEND
```

Hardware - display LCD

El display LCD puede ser conectado de la siguiente manera:

DISPLAY LCD	PUERTO	PIN (Display)
DB7	P1.7	14
DB6	P1.6	13
DB5	P1.5	12
DB4	P1.4	11
E	P1.3	6
RS	P1.2	4
RW	Ground	5
Vss	Ground	1
Vdd	+5 Volt	2
Vo	0-5 Volt	3

Esta conexión deja disponibles P1.1 y P1.0 y P3 para otros empleos.
Es posible definir conexiones diferentes especificando los pin del puerto en el menú Options LCD.
El tipo de display LCD puede ser seleccionado con la instrucción CONFIG LCD.
El display LCD trabaja en modalidad 4 bits.
Ver la instrucción $LCD por el funcionamiento en modalidad 8 bites.
BASCOM cuenta con muchas instrucciones por la gestión del display LCD.
Para los que necesiten un mayor control, el ejemplo siguiente puede ser de ayuda:

```
Acc = 5              'carga un valor en el registro A
Call Lcd_control     ' es el valor de control del display
Acc = 65             'carga un nuevo valor (la letra A)
Call Write_lcd       'lo escribe en el display LCD
```

Lcd_control y Write_lcd son subroutines assembler que pueden ser llamadas por BASCOM.
Averiguar las especificaciones del display LCD utilizado para un correcto empleo.

Microprocesadores soportados

Algunos microprocesadores cuentan con características adicionales comparadas con el AT89C2051/8051.

8032/8052/AT89S8252
TIMER2

AT89S8252
WATCHDOG
DATA EEPROM
Pin del puerto con funciones alternativas

80515,80535,80517,80535
GETAD
WATCHDOG
BAUDRATE GENERATOR
INTERRUPCIONES y PRIORIDADES

80517,80537
GETAD
WATCHDOG
BAUDRATE GENERATOR
BAUDRATE GENERATOR1
INTERRUPCIONES e PRIORIDADES

AT89S8252 WATCHDOG

El microprocesador AT89S8252 incorpora un temporizador de watchdog.

Un temporizador de watchdog es un temporizador que provee a resetear el microprocesador al final de cierto valor (tiempo).

Por lo tanto, durante la ejecución del programa es necesario resetear este temporizador antes de que complete la cuenta. Esta función es utilizada para garantizar que el programa esté trabajando correctamente.

Cuando un programa no funciona o bien se estanca en un bucle sin fin el temporizador de watchdog no es borrado y por lo tanto se ocasiona una reposición automática con el consiguiente reinicio del programa.

START WATCHDOG activa el timer del watchdog.
STOP WATCHDOG para el timer del watchdog.
RESET WATCHDOG resetea (reinicia) el timer del watchdog.

Ver también

CONFIG WATCHDOG

Ejemplo:

```
'-----------------------------------------------------
' (c) 1998 MCS Electronics
' WATCHD.BAS demonstrates the AT89S8252 watchdog timer
' select 89s8252.dat !!!
'-----------------------------------------------------
Config Watchdog = 2048  'reset después de 2048 mSec
Start Watchdog          'start del timer del watchdog
Dim I As Word
For I = 1 To 10000
      Print I           'visualiza el valor

' Reset Watchdog
'el bucle for next no será completado por la intervención del timer del
'watchdog

      Next
End
```

WATCHDOG

El microprocesador AT89S8252 incorpora un temporizador de watchdog.

Un temporizador de watchdog es un temporizador que provee a resetear el microprocesador al final de cierto valor (tiempo).

Por lo tanto, durante la ejecución del programa es necesario resetear este temporizador antes de que complete la cuenta. Esta función es utilizada para garantizar que el programa esté trabajando correctamente.

Cuando un programa no funciona o bien se estanca en un bucle sin fin el temporizador de watchdog no es borrado y por lo tanto se ocasiona una reposición automática con el consiguiente reinicio del programa.

CONFIG WATCHDOG = value

value	Tiempo transcurrido al cual el temporizador de WD generará un reset. Valores posibes, en milisegundos: 16,32,64,128,256,512,1024 ó 2048

START WATCHDOG activa el timer del watchdog.
STOP WATCHDOG para el timer del watchdog.
RESET WATCHDOG resetea (reinicia) el timer del watchdog.

Ejemplo:

```
DIM A AS INTEGER
CONFIG WATCHDOG = 2048          'produce un reset después de 2 segundos
START WATCHDOG                  'activa el WD
DO
      PRINT a
      a = a + 1                 'para en caso de reset
      REM RESET WATCHDOG        'eliminando REM trabajará correctamente
LOOP
END
```

DATA EEPROM

El microprocesador AT89S8252 incorpora 2Kbytes de memoria flash EEPROM. Esta memoria puede ser utilizada para memorizar datos.

Por esto son previstas dos instrucciones específicas : WRITEEEPROM y READEEPROM.

WRITEEEPROM var [, address]

var	Cualquier nombre de Variable BASCOM.
Address	Dirección de la EEPROM donde escribir el dato, entre 0÷2047. Si es omitida la dirección ésta será asignada por BASCOM automáticamente y será posible localizarla leyendo en el ráport que genera el compilador.

READEEPROM var [, address]

var	Cualquier nombre de Variable BASCOM.
Address	La dirección de la EEPROM donde leer el dato, entre 0÷2047. La dirección puede ser omitida si los datos han sido memorizados anteriormente con la instrucción WRITEEEPROM ya que en este caso el compilador es capaz de remontar a la dirección asignada automáticamente.

Ejemplo:

```
Dim S As String * 15 , S2 As String * 10
S = "Hola" : S2 = "test"
Dim L As Long
L = 12345678
Writeeeprom S
Writeeeprom S2                      'escribe en formato string
Writeeeprom L                       'escribe en long
S = "" : S2 = "" : L = 0            'clear la variable
Readeeprom L : Print L
Readeeprom S : Print S
Readeeprom S2 : Print S2
End
```

Pin del puerto con funciones alternativas

Los puertos del microprocesador AT89S8252 tienen funciones alternativas y son resumidas en la siguiente tabla.

Puerto pin	**Función alternativa**
P1.0	T2 entrada de contaje externo para timer.counter 2, salida señal de reloj
P1.1	T2EX timer/counter 2 triger (**disparo**) captura/recarga y flag (**bandera**) de dirección.
P1.4	/SS Entrada de selección de esclavo
P1.5	MOSI salida datos maestro, entrada datos esclavo para SPI
P1.6	MISO entrada datos maestro, salida datos esclavo para SPI
P1.7	SCK salida reloj maestro, entrada reloj esclavo para SPI
P3.0	RxD entrada puerto serie asíncrono
P3.1	TxD salida puerto serie serie asíncrono
P3.2	/INT0 entrada interrupción 0
P3.3	/INT1 entrada interrupción 1
P3.4	T0 entrada externa timer 0
P3.5	T1 entrada externa timer 1
P3.6	/WR strobe de escritura para memoria de datos externa
P3.7	/RD strobe de lectura para memoria de datos externa

/ indica entrada negada, activo bajo

Página en blanco

80515 INTERRUPCIONES Y PRIORIDADES

Los microprocesadores 80515 y 80535 dispone de más interrupciones y el nivel de las prioridades es administrado de otra manera con respecto al clásico 8051.

Habilitación de interrupciones:

```
ENABLE AD                          'convertidor A/D
ENABLE INT2|INT3|INT4|INT5|INT6    ‘interrupción externa 2-6
ENABLE TIMER2EX                    'recarga externa del timer2
```

Deshabilitación de las interrupciones:

```
DISABLE AD                         'convertidor A/D
DISABLE INT2|INT3|INT4|INT5|INT6   ‘interrupción externa 2-6
DISABLE TIMER2EX                   'recarga externa del timer2
```

Selección de la prioridad:

PRIORITY SET|RESET fuente , nivel
El nivel puede ser 0,1,2 o 3.(0=más baja, 3=más alta)

Las fuentes pueden ser:

INT0/ADC
TIMER0/INT2
INT0/INT3
TIMER1/INT4
SERIAL/INT5
TIMER2/INT6

Atención, sólo una de una pareja puede ser seleccionada.

```
PRIORITY SET INT4,3     ' INT4 programará a la máxima prioridad.
```

Cuando dos interrupciones ocurren al mismo tiempo la primera a ser ejecutada será aquella con la prioridad mayor y en caso de idéntica prioridad será la primera en orden de lista a ser servida. Luego a una solicitud conteniendo TIMER1 e INT4, y se programan para tener la misma prioridad, TIMER1 será servida antes que INT4.

Para mayores detalles ver las especificaciones de la documentación técnica.

GETAD

Acción

Recupera il valore di una conversione analogica da uno dei canali AD 0÷7.
Per microprocessori 80517 o 80537 sono disponibili canali tra 0÷11.

Sintaxis

var = GETAD(channel, range)

Nota

var	Variable a la cual asignará el valor A/D leido
channel	Identificativo del canal a medir
range	Seleción del campo de medida 0 = 0-5 Volt 192 = 0 - 3.75 Volt 128 = 0 - 2.5 Volt 64 = 0 - 1.25 Volt 12 = 3.75 - 5 Volt 200 = 2.5 - 3.75 Volt 132 = 1.25 - 2.5 Volt

La instrucción GETAD() está disponibile solo con microprocesadores 80515, 80535, 80517 y 80535 porque hace uso de características específicas del chip.

Ver también

Ejemplo:

```
Dim b1 As Byte, Channel As Byte, ref As Byte
channel=0                'Entrada por P6.0
ref=0                    'rango de 0 ÷ 5 Voltios
b1=getad(channel,ref)    'guarda el valor A/D leido en b1
```

80515 WATCHDOG

Los microprocesadores 80515 y 80535 disponen de un temporizador de watchdog. Este es un temporizador a 16 bits que no puede ser parado y producirá una reposición después de 65535 uS con un reloj a 12MHz !

```
START WATCHDOG      'activa el temporizador del watchdog.
RESET WATCHDOG      'resetea (pone a cero) el timer del watchdog.
```

80537 INTERRUPCIONES Y PRIORIDAD

Los microprocesadores 80517 y 80537 dispone de más interrupciones y el nivel de las prioridades es administrado de otra manera con respecto al clásico 8051.

Habilitación de las interrupciones:

```
ENABLE AD                          'convertidor A/D
ENABLE INT2|INT3|INT4|INT5|INT6    'interrupción externa 2-6
ENABLE TIMER2EX                    'recarga externa del timer2
ENABLE CTF                         'interrupción de tiempo comparado
ENABLE SERIAL1                     'interrupción serie a síncrona 1
```

Deshabilitación de las interrupciones:

```
DISABLE AD                         'convertidor A/D
DISABLE INT2|INT3|INT4|INT5|INT6   'interrupción externa 2-6
DISABLE TIMER2EX                   'recarga externa del timer2
DISABLE CTF                        'interrupción de tiempo comparado
DISABLE SERIAL1                    'interrupción serie asíncrona 1
```

Selección de la prioridad :

PRIORITY SET|RESET fuente , nivel

Atención, sólo una de un tripleta puede ser seleccionada.

```
PRIORITY SET INT4,3                ' INT4 se programará a la máxima prioridad.
```

Cuando dos interrupciones ocurren al mismo tiempo la primera a ser ejecutada será aquella con la prioridad mayor y en caso de idéntica prioridad será la primera en orden de lista a ser servida. Luego a una solicitud conteniendo TIMER1 e INT4, y se programan para tener la misma prioridad, TIMER1 será servida antes que INT4.

Para mayores detalles ver las especificaciones de la documentación técnica del microprocesador que se use.

CONFIG BAUD1

Acción
Configura el microprocesador para utilizar el generador interno de baud_rate (velocidad en baudios) sobre el canal serie asíncrono 1.
Este generador de baud_rate está disponible solo en los microprocesadores 80517 y 80537.

Sintaxis
CONFIG BAUD1 = baudrate

Nota
baudrate	Velocidad en baudios a utilizar entre : 2048 ÷ 37500

El microprocesador 80517 y el 80537 disponen de 2 puertos serie asíncrono integrados en el mismo chip.

Ver también
CONFIG BAUD

Ejemplo:
```
CONFIG BAUD1 = 9600     'usa el generador de baudios interno
Print "HOLA"
End
```

ANEXO 1

Página intencionada en blanco

ANEXO 2

Página intencionada en blanco

INDICE

Palabras reservadas en BASCOM

ooOOoo