In particulare voglio ringraziare Giovanni Pedruzzi della ditta Contrive che ha tradotto questa documentazione in Italiano.

BASCOM-8051

GUIDA DI RIFERIMENTO DEL LINGUAGGIO

1WRESET,1WREAD,1WWRITE

Azione

Queste routines possono essere impiegate per comunicare con i dispositivi 1 Wire prodotti da Dallas Semiconductors.

Sintassi

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

Note

1WRESET	Reset del bus 1WIRE. In caso di errore la variabile ERR conterrà il valore 1.
1WWRITE var1	Invia sul bus il valore di var1.
var2 = 1WREAD()	Legge un byte dal bus e lo associa a var2.

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

Esempio

```
'-----------------------------------------------------
'                    1WIRE.BAS
' mostra l'uso di 1wreset, 1wwrite and 1wread()
' pullup di 4K7 necessario da P.1 alla VCC
' bottone seriale DS2401 collegato a P1.1
'-----------------------------------------------------
Config 1wire = P1.1                'usa questo pin
Dim Ar(8) As Byte , A As Byte , I As Byte

1wreset                            'reset del bus
Print Err                          'print error 1 in caso di errore
1wwrite &H33                       'comando lettura ROM
For I = 1 To 8
  Ar(I) = 1wread()                 'legge byte
Next
For I = 1 To 8
  Printhex Ar(I);                  'print output
Next
Print                              'linefeed
End
```

$ASM - $END ASM

Azione
Inizio di un blocco in linguaggio assembler.

Sintassi
$ASM

Note
$ASM deve essere impiegato congiuntamente ad $END ASM per incorporare un blocco in codice assembler all'interno di un programma BASIC.

Esempio
```
Dim c as Byte
$ASM
  Mov r0,#{C}  ;indirizzo di c
  Mov a,#1
  Mov @r0,a    ;salva 1 in var c
$END ASM
Print c
End
```

$INCLUDE

Azione
Incorpora un file ASCII nel programma a partire dalla posizione corrente.

Sintassi
$INCLUDE file

Note

file	Nome del file ASCII che deve contenere istruzioni BASCOM. Questa opzione può essere utilizzata quando siano impiegate le stesse routines in più programmi, in questo modo possono essere preparati dei moduli da incorporare nei programmi. Ogni volta che dovrete aggiornare una routine sarà sufficiente aggiornare il singolo modulo e non tutti i programmi che ne fanno uso. Il file deve essere in formato ASCII!

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

$BAUD

Azione
Forza il compilatore ad utilizzare uno specifico valore di baudrate, ignorando quello predefinito nel menu delle opzioni.

Sintassi
$BAUD = var

Note
var	Questo è il baudrate che si intende usare.

var : Constant.

Questa direttiva può rivelarsi utile quando sia necessario selezionare dei valori per crystal/baudrate che non siano disponibili nel menu delle opzioni.
La direttiva $CRYSTAL è sempre usata in combinazione.

Nel file report generato in fase di compilazione è contenuta l'informazione sul valore di baudrate attualmente prodotto.
Il valore di baudrate è mostrato solo se esistono effettivamente istruzioni che facciano uso della comunicazione seriale asincrona (RS-232) come ad esempio PRINT, INPUT ecc.

Vedere anche
$CRYSTAL

Esempio
```
$BAUD = 2400
$CRYSTAL = 14000000   ' cristallo da 14 MHz
PRINT "Hello"
END
```

$CRYSTAL

Azione

Forza il compilatore ad utilizzare uno specifico valore per il cristallo dell'oscillatore, ignorando quello predefinito nel menu delle opzioni.

Sintassi

$CRYSTAL = var

Note

var	Frequenza del cristallo.

var : Constant.

Questa direttiva può rivelarsi utile quando sia necessario selezionare dei valori per crystal/baudrate che non siano disponibili nel menu delle opzioni.
La direttiva $BAUD è sempre usata in combinazione.

Vedere anche

$BAUD

Esempio

```
$BAUD = 2400
$CRYSTAL = 14000000
PRINT "Hello"
END
```

$IRAMSTART

Azione
Direttiva per il compilatore contenente l'indirizzo di partenza della memoria RAM interna.

Sintassi
$IRAMSTART = constant

Note

constant	Costante con il valore d'inizio RAM (0-255)

Vedere anche
$NOINIT $RAMSTART

Esempio
```
$NOINIT
$NOSP
$IRAMSTART = &H60           'prima locazione di memoria utilizzabile
SP = 80
DIM I As Integer
```

$DEFAULT XRAM

Azione
Direttiva per il compilatore per forzare la memorizzazione delle variabili dimensionate nella RAM esterna (XRAM).

Sintassi
$DEFAULT XRAM

Note
Quando siano definite molte variabili nella RAM esterna, invece di specificare per ognuna l'opzione XRAM è possibile istruire il compilatore per impiegare questa memoria come default di memorizzazione.
In questo caso, se si deve memorizzare una variabile nella RAM interna sarà possibile indicare l'opzione IRAM.

Esempio
```
$DEFAULT XRAM
Dim X As Integer            'memorizzato in XRAM
Dim Z As IRAM Integer       'memorizzato in IRAM
```

$LARGE

Azione
Forza il compilatore all’uso di istruzioni LCALL.

Sintassi
$LARGE

Note
Normalmente, ogni volta che viene richiamata una subroutine il compilatore fa uso di istruzioni di tipo ACALL, che presentano il vantaggio di impiegare solo 2 bytes per la loro memorizzazione (LCALL richiede invece 3 bytes).
Una istruzione ACALL può quindi indirizzare routines con un offset fino a 2048.
Impiegando microprocessori AT89C2051 l’offset consentito da una istruzione ACALL copre l’intera memoria disponibile, non esiste quindi ragione di fare uso di LCALL.

Per contro, generando codice per altri microprocessori, le subroutine possono essere memorizzate anche molto più avanti rispetto alla posizione di chiamata ed una istruzione di ACALL può rivelarsi inadeguata, producendo un errore in esecuzione.
Con la direttiva $LARGE si forza il compilatore a fare uso di LCALL che consentono di indirizzare fino a 64K di memoria.

Esempio
```
$LARGE          'Ho ricevuto un errore 148 quindi serve questa opzione
```

$LCD

Azione

Forza il compilatore a generare codice per la gestione del display LCD a 8 bit, mappato in memoria e connesso al bus.

Sintassi

$LCD = [&H]*address*

Note

address	L'indirizzo al quale è disponibile il display LCD. Le linee dati db0-db7 del display LCD devono essere collegate al bus dati D0-D7. La segnale RS del display LCD deve essere collegata alla linea A9 del bus indirizzi. In sistemi con RAM/ROM esterne può risultare più conveniente derivare il display LCD dal bus. Sarà necessario prevedere una linea di abilitazione specifica proveniente dalla decodifica degli indirizzi.

Esempio

```
$LCD = &HA000 'scrivendo a questo indirizzo alzo la linea E del display LCD
LCD "Hello world"
```

$NOBREAK

Azione
Forza il compilatore ad ignorare le istruzioni BREAK.

Sintassi
$NOBREAK

Note
Le istruzioni BREAK vengono inserite per generare una pausa nel simulatore.
Quando si desidera compilare tralasciando i BREAK senza doverli rimuovere dal programma è sufficiente fare uso della direttiva $NOBREAK.

Vedere anche
BREAK

Esempio
```
$NOBREAK
BREAK          ' questo non sarà compilato quindi il simulatore non farà pause
End
```

$NOINIT

Azione
Forza il compilatore a non eseguire alcuna inizializzazione.

Sintassi
$NOINIT

Note

BASCOM inizializza il processore in funzione delle istruzioni utilizzate.
Se si desidera provvedere autonomamente alla inizializzazione è necessario specificare **$NOINIT**.
In questo caso il compilatore si limiterà ad inizializzare lo stack pointer ed il display LCD (sempre che siano state utilizzate istruzioni relative al display LCD).

Vedere anche
$NOSP

Esempio
```
$NONIT
'qui va il programma
End
```

$NOSP

Azione
Forza il compilatore a non inizializzare lo stack pointer.

Sintassi
$NOSP

Note
BASCOM inizializza il processore in funzione delle istruzioni utilizzate.
Se si desidera provvedere autonomamente alla inizializzazione è necessario specificare **$NOINIT**.
In questo caso il compilatore si limiterà ad inizializzare lo stack pointer ed il display LCD (sempre che siano state utilizzate istruzioni relative al display LCD).
Se viene specificato **$NOSP** non sarà inizializzato neppure lo stack.

Vedere anche
$NOINIT

Esempio
```
$NOSP
$NOINIT
End
```

$OBJ

Azione
Inclusione di codice oggetto in formato Intel.

Sintassi
$OBJ obj

Note
obj è il codice oggetto da includere.

Esempio
```
$OBJ D291  'equivalente a SET P1.1
```

$RAMSTART

Azione
Specifica la locazione d'inizio della memoria RAM esterna.

Sintassi
$RAMSTART = [&H]address

Note

address	L'indirizzo (hex) d'inizio della memoria dati (l'indirizzo più basso che abilita il chip della RAM). Questa opzione sarà impiegata in sistemi che fanno uso di RAM esterna.

address : Constant.

Vedere anche
$RAMSIZE

Esempio
```
$ROMSTART = &H4000
$RAMSTART = 0
$RAMSIZE = &H1000
```

$RAMSIZE

Azione
Specifica la dimensione della memoria RAM esterna.

Sintassi
$RAMSIZE = [&H] size

Note
size	Dimensione della RAM esterna.

size : Constant.

Vedere anche
$RAMSTART

Esempio
```
$ROMSTART = &H4000
$RAMSTART = 0
$RAMSIZE = &H1000
DIM x AS XRAM Byte    'specifico XRAM per salvare la variabile in XRAM
```

$ROMSTART

Azione
Direttiva per il compilatore contenente l'indirizzo di partenza della memoria ROM.

Sintassi
$ROMSTART = [&H] *address*

Note

address	L'indirizzo (HEX) d'inizio della memoria contenente il codice. Quando non specificato con la direttiva s'intende per default = 0. Questa opzione può essere impiegata quando si desidera testare il codice in RAM. Il codice deve essere caricato ad uno specifico indirizzo e richiamato da un programma monitor residente. Il programma monitor deve provvedere a spostare le interrupts all'indirizzo corretto! Quando si specifica $ROMSTART = &H4000 il programma di monitor deve effettuare un LJMP. L'indirizzo 3 deve corrispondere a &H4003. Diversamente le interrupt non potranno essere correttamente richiamati. Ma a questo deve provvedere il programma di monitor.

Vedere anche
$RAMSTART

Esempio
```
$ROMSTART = &H4000    'ROM abilitata a partire da 4000 hex
```

$SERIALINPUT

Azione
Specifica una diversa sorgente per l'ingresso seriale.

Sintassi
$SERIALINPUT = label

Note

label	Il nome della routine assembler che deve essere richiamata quando viene eseguita un'istruzione di INPUT. Il carattere ricevuto dovrà essere prelevato dall'ACCumulatore.

Con il reindirizzamento dell'INPUT è possibile impiegare routines specifiche.
Con questa opzione risulta possibile impiegare dispositivi diversi come sorgenti di INPUT.
L'istruzione di INPUT viene terminata quando nell'esecuzione della routine si incontra RETURN codice (13).

Vedere anche
$SERIALOUTPUT

Esempio
```
$SERIALINPUT = Myinput
 'qui va il programma
END

!myinput:
 ;effettuare qui le operazioni richieste
  mov a, sbuf  ;buffer ingresso seriale in acc
ret
```

$SERIALINPUT2LCD

Azione
Questa direttiva consente di inviare al display LCD invece che alla porta seriale l'eco dei caratteri ricevuti.

Sintassi
$SERIALINPUT2LCD

Note
E' anche possibile realizzare degli specifici driver che gestiscano l'input e l'output facendo uso delle direttive $SERIALINPUT e $SERIALOUTPUT, mentre la ridirezione dell'output su display LCD è già preconfezionata ed utilizzabile mediante la direttiva $SERIALINPUT2LCD.

Vedere anche
$SERIALINPUT , $SERIALOUTPUT

Esempio
```
$SERIALINPUT2LCD
Dim v as Byte
CLS
INPUT "Number ", v    'questo andrà al display LCD
```

$SERIALOUTPUT

Azione
Specifica che l'output seriale deve essere ridiretto su uno specifico driver.

Sintassi
$SERIALOUTPUT = label

Note

label	Nome della routine assembler che sarà richiamata ogni volta che un carattere viene inviato al buffer seriale (SBUF). Il carattere è posto nell'ACCumulatore.

Con la ridirezione dell'istruzione PRINT e degli altri comandi legati all'output seriale è possibile fare uso di routines personalizzate.
In questo modo è possibile utilizzare qualsiasi altro dispositivo come destinatario dell'output.

Esempio

```
$SERIALOUTPUT = MyOutput
 'qui va il programma
END

!myoutput:
 ; effettuare qui le operazioni richieste
  mov sbuf, a  ;buffer di uscita seriale (default)
ret
```

$SIM

Azione
Produce la generazione di codice senza loop di attesa destinato all'uso con il simulatore.

Sintassi
$SIM

Note
Ogni volta che viene eseguita un'istruzione di WAIT nel simulatore si potrà notare che il tempo di attesa è in realtà molto lungo. E' possibile anche disabilitare l'aggiornamento delle variabili, che prende diverso tempo, ma l'alternativa più conveniente consiste nell'impiego della direttiva $SIM.

E' assolutamente necessario rimuovere $SIM quando si effettua la compilazione definitiva del codice da trasferire nel dispositivo.

Vedere anche
-

Esempio
```
$SIM          'non compilare WAIT e WAITMS
WAIT 2        'il simulatore ora è più veloce
```

ABS()

Azione
Restituisce il valore assoluto di una variabile.

Sintassi
var **= ABS(**var2**)**

Note

var	Variabile di destinazione del valore assoluto di var2.
Var2	Variabile originale dalla quale estrarre il valore assoluto.

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

Il valore assoluto di un numero è sempre positivo.

Vedere anche
-

Differenze da QB
Non è possibile fare uso di costanti numeriche poiché per queste ultime è intrinsecamente definito il formato in valore assoluto.
Non è neppure possibile effettuare l'operazione su Singles.

Esempio
```
Dim a as Integer, c as Integer
a = -1000
c = Abs(a)
Print c
End
```

Output
1000

ALIAS

Azione
Definisce un nome alternativo che sarà utilizzato come riferimento per una variabile.

Sintassi
newvar **ALIAS** oldvar

Note

oldvar	Nome della variabile, ad esempio P1.1
newvar	Nuovo nome della variabile, ad esempio *direction*

Definire dei nomi alternativi per i pin di porta può rendere più comprensibile l'impiego al quale sono destinati.

Vedere anche
CONST

Esempio
```
direction ALIAS P1.1  'ora P1.1 può essere richiamato come variable direction
SET direction         'produce lo stesso effetto di SET P1.1
END
```

ASC()

Azione
Converte una stringa nel suo corrispondente valore ASCII.

Sintassi
var = ASC(string**)**

Note

var	Variabile di destinazione della conversione.
String	Stringa variabile o costante dalla quale ottenere il valore ASCII.

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

In ogni caso la conversione sarà effettuata solo sul primo carattere di una stringa.
Se la stringa risultasse vuota il risultato della conversione sarà zero.

Vedere anche
CHR()

Esempio
```
Dim a as byte, s as String * 10
s = ABC
a = Asc(s)
Print a
End
```

Output
65

BCD()

Azione
Converte una variabile nel corrispondente valore BCD.

Sintassi
PRINT BCD(var)
LCD BCD(var)

Note

var	Variabile da convertire.

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

Nel caso si desideri interfacciare un dispositivo real time clock, questa funzione consentirà una rapida conversione dal formato BCD utilizzato da questo componente.
BCD() visualizzerà un valore in formato trailing zero.

La funzione BCD() è destinata all'uso in abbinamento alle istruzioni PRINT/LCD.
Per convertire delle variabili esiste la funzione MAKEBCD.

Vedere anche
MAKEBCD , MAKEDEC

Esempio
```
Dim a as byte
a = 65
LCD a
Lowerline
LCD BCD(a)
End
```

BITWAIT

Azione
Attende finché un bit sia settato o resettato.

Sintassi
BITWAIT x **SET/RESET**

Note

x	Variabile tipo Bit o registro interno (es. P1.x), per x nell'intervallo 0÷7.

Utilizzando variabili di tipo Bit, assicuratevi che possano essere settate/resettate dal software per evitare di rimanere in attesa indefinita.
Definendo dei registri legati all'hardware, come P1.0 non si corre questo tipo di rischio.

Vedere anche
-

Esempio
```
Dim a as bit
BITWAIT a , SET              'attende fino a quando il bit a è settato a 1
BITWAIT P1.7, RESET          'attende fino a quando il bit 7 della Port 1 è 0
End
```

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

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

BREAK

Azione
Genera una pausa nell’esecuzione del simulatore.

Sintassi
BREAK

Note
E’ possibile definire dei breakpoints nel simulatore, ma è altresì possibile impostare breakpoints per mezzo dell’istruzione BREAK.
Assicuratevi di avere rimosso ogni BREAK al termine delle operazioni di debug, oppure utilizzate il metacomando $NOBREAK.

Il corrispondente opcode riservato è A5.

Vedere anche
$NOBREAK

Esempio
```
PRINT "Hello"
BREAK          'il simulatore farà una pausa ora
.....
.....
End
```

CALL

Azione
Richiama ed esegue una subroutine.

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

Note

var1	Qualsiasi variabile o costante BASCOM.
var-n	Qualsiasi variabile o costante BASCOM.
Test	Nome della subroutine. In questo caso Test

Con l'istruzione CALL è possibile richiamare ed eseguire una procedura o subroutine.
Fino ad un massimo di 10 parametri possono essere passati alla subroutine, ma ovviamente è anche possibile effettuare una chiamata senza passare alcun parametro.
Per esempio : **Call Test2**

Impiegando l'istruzione CALL è in quindi possibile definire delle istruzioni personalizzate.

Non è obbligatorio specificare l'istruzione CALL :
Test2 richiama ed esegue in ogni caso la subroutine test2

Se viene omessa l'istruzione CALL, devono essere omesse anche le parentesi.
Quindi Call Routine(x,y,z) equivale a Routine x,y,x

Vedere anche
DECLARE, SUB

Esempio
```
Dim a as byte, b as byte
Declare Sub Test(b1 as byte)
a = 65
Call test (a)         'chiama test passando il parametro A
test a                'alto modo di effettuare la chiamata
End

SUB Test(b1 as byte) 'usa la variabile come precedentemente dichiarata
  LCD b               'la inia al display LCD
  Lowerline
  LCD BCD(b1)
End SUB
```

CHR()

Azione
Converte una variabile di tipo Byte, Integer/Word oppure una costante in un carattere.

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

Note

var	Variabile tipo Byte, Integer/Word o costante numerica.
s	Variabile di tipo Stringa.

Ogni volta che si deve visualizzare un carattere sullo schermo oppure sul display LCD, deve essere convertito con la funzione CHR().

Vedere anche
ASC()

Esempio
```
Dim a as byte
a = 65
LCD a
Lowerline
LCDHEX a
LCD Chr(a)
End
```

CLS

Azione
Cancella il display LCD e posiziona il cursore a capo (prima riga, prima colona).

Sintassi
CLS

Note
Quando si cancella il display LCD la memoria RAM che contiene i caratteri speciali non viene cancellata.

Vedere anche
$LCD , LCD

Esempio
```
Cls
LCD " Hello"
End
```

CONST

Azione
Dichiarazione di una costante simbolica.

Sintassi
DIM symbol AS CONST value

Note

symbol	Nome del simbolo.
Value	Valore da assegnare al simbolo.

L’assegnazione di una costante non richiede spazio nella memoria di programma.
In fase di compilazione, tutte le costanti saranno sostituite con il valore assegnato.

Vedere anche
DIM

Esempio
```
'------------------------------------------------------
'              (c) 1997,1998 MCS Electronics
'              CONST.BAS
'------------------------------------------------------
Dim A As Const 5                        'dichiara a come costante
Dim B1 As Const &B1001
Waitms A                                'attende 5 millisecondi
Print A
Print B1
End
```

CONFIG

L’istruzione config consente la configurazione delle istruzioni legate all’hardware.
Segue è un elenco delle possibili opzioni per l’istruzione config, ognuna delle quali viene poi descritta individualmente.

CONFIG TIMER0, TIMER1
CONFIG TIMER2 (per dispositivi 8052 compatibili)
CONFIG LCD
CONFIG LCDBUS
CONFIG LCDPIN
CONFIG BAUD
CONFIG 1WIRE
CONFIG SDA
CONFIG SCL
CONFIG DEBOUNCE
CONFIG WATCHDOG
CONFIG SPI

CONFIG TIMER0, TIMER1

Azione

Configura TIMER0 o TIMER1.

Sintassi

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

Note

TIMERx	TIMER0 o TIMER1. COUNTER configurerà TIMERx come COUNTER e TIMER configurerà TIMERx come TIMER. Un TIMER utilizza il clock interno al microprocessore mentre un COUNTER riceve il clock dall'esterno.
GATE	INTERNAL o EXTERNAL. Specificando EXTERNAL si abilita il controllo GATE attraverso l'ingresso INT.
MODE	Time/counter modalità 0÷3. Maggiori dettagli nella sezione Hardware.

Quindi CONFIG TIMER0 = COUNTER, GATE = INTERNAL, MODE=2 configurerà TIMER0 come CONTATORE senza controllo esterno di gate, in modalità 2 (auto reload).

Durante l'operazione di configurazione il timer/counter si arresta, quindi sarà necessario riavviarlo in seguito con la specifica istruzione START TIMERx.

Vedere le istruzioni addizionali disponibili per altri microprocessori che fanno uso dell'istruzione CONFIG.

Esempio

```
CONFIG TIMER0=COUNTER, MODE=1, GATE=INTERNAL
COUNTER0 = 0          'reset del counter 0
START COUNTER0        'abilita il counter
DELAY                 'attende un attimo
PRINT COUNTER0        'print del counter
END
```

CONFIG LCD

Azione
Configura il display LCD.

Sintassi
CONFIG LCD = LCDtype

Note

LCDtype	Il tipo di display LCD utilizzato, selezionato tra i seguenti: 40 * 4,16 * 1, 16 * 2, 16 * 4, 16 * 4, 20 * 2 or 20 * 4 Per default si assume 16 * 2 .

Esempio
```
CONFIG LCD = 40 * 4
LCD "Hello"           'visualizza sul display LCD
FOURTHLINE            'seleziona la linea 4
LCD "4"               'visualizza 4
END
```

CONFIG LCDBUS

Azione
Configura il bus dati per il display LCD.

Sintassi
CONFIG LCDBUS = constant

Note

constant	4 per modalità a 4-bit, 8 per modalità 8-bit (default)

Utilizzare questa istruzione congiuntamente alla direttiva $LCD = address.
Quando è impiegato un display LCD derivato dal bus dati, per default si considerano utilizzate tutte le 8 linee dati. Se viene selezionata la modalità 4-bit dovranno essere impiegate le linee dati d7÷d4.

Vedere anche
CONFIG LCD

Esempio
```
$LCD = &H8000         'indirizzo della linea di enable
Config LCDBUS = 4     'modalità a 4 bit
LCD "hello"
```

CONFIG BAUD

Azione
Configura il microprocessore affinché utilizzi il generatore interno di baudrate.
Il generatore interno di baudrate è disponibile solo nei dispositivi 80535, 80537 e compatibili

Sintassi
CONFIG BAUD = baudrate

Note
baudrate	Baudrate da utilizzare : 4800 o 9600

Esempio
```
CONFIG BAUD = 9600    'usa il generatore interno di baud generator
Print "Hello"
End
```

CONFIG 1WIRE

Azione
Configura il pin che deve essere utilizzato dall'istruzione 1WIRE.

Sintassi
CONFIG 1WIRE = pin

Note
pin	Pin di porta da utilizzare, ad esempio P1.0

Vedere anche
1WRESET , 1WREAD , 1WWRITE

Esempio
```
Config 1WIRE = P1.0   'P1.0 usato per il bus 1-wire
1WRESET               'reset del bus
```

CONFIG SDA

Azione
Ignora le assegnazioni effettuate in Option Settings e seleziona un pin per la linea SDA.

Sintassi
CONFIG SDA = pin

Note
pin	Pin di porta da utilizzare per la linea I2C-SDA.

Quando sono utilizzati terminali diversi nei vari progetti, è possibile ignorare le predisposizioni e definire ogni volta il pin della linea SDA.
In questo modo il programma conterrà esplicitamente dichiarato il pin utilizzato e non sarà necessario annotarlo separatamente per modificarlo nelle Option Settings.

Vedere anche
CONFIG SCL

Esempio
```
CONFIG SDA = P3.7            'P3.7 è la linea SDA
```

CONFIG SCL

Azione
Ignora le assegnazioni effettuate in Option Settings e seleziona un pin per la linea SCL.

Sintassi
CONFIG SCL = pin

Note
pin	Pin di porta da utilizzare per la linea I2C-SCL.

Quando sono utilizzati terminali diversi nei vari progetti, è possibile ignorare le predisposizioni e definire ogni volta il pin della linea SCL.
In questo modo il programma conterrà esplicitamente dichiarato il pin utilizzato e non sarà necessario annotarlo separatamente per modificarlo nelle Option Settings.

Vedere anche
CONFIG SDA

Esempio
```
CONFIG SCL = P3.5            'P3.5 è la linea SCL
```

CONFIG DEBOUNCE

Azione
Configura il ritardo per l'istruzione DEBOUNCE.

Sintassi
CONFIG DEBOUNCE = time

Note

time	Costante numerica che specifica il ritardo in mS.

Se il tempo di debounce non è configurato, si applica quello di default pari a 25 mS.
Questo tempo è riferito ad una frequenza di clock del processore di 12 MHz.

Vedere anche
DEBOUNCE

Esempio
```
Config Debounce = 25 mS             '25 mS per default
```

CONFIG SPI

Azione
Configura le istruzioni relative alla porta SPI.

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

Note

DIN	Pin da utilizzare per Data input, ad esempio P1.0
DOUT	Pin da utilizzare per Data output, ad esempio P1.1
CS	Pin da utilizzare per Chip select, ad esempio P1.2
CLK	Pin da utilizzare per Clock, ad esempio P1.3

Vedere anche
SPIIN SPIOUT

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

CONFIG LCDPIN

Azione
Ignora le opzioni relative al display LCD e ne definisce di nuove all'interno del programma.

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

Note
P1.1 ecc. sono riportati solo a titolo di esempio.

Vedere anche
CONFIG LCD

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

CONFIG WATCHDOG

Azione
Configura il timer di watchdog nel dispositivo AT89C8252

Sintassi
CONFIG WATCHDOG = time

Note

time	Costante che definisce l'intervallo di tempo per il timer di watchdog, espresso in mS, tra le seguenti possibili opzioni : Impostazioni possibili : 16 , 32, 64 , 128 , 256 , 512 , 1024 e 2048.

Quando il WatchDog è avviato, sarà generato un reset dopo il tempo specificato. Impostando 2048, un reset avverrà dopo circa 2 secondi, per evitare che questo avvenga sarà necessario resettare periodicamente il timer di WD all'interno del programma.

Vedere anche
START WATCHDOG , STOP WATCHDOG , RESET WATCHDOG

Esempio
```
'-----------------------------------------------------
'                (c) 1998 MCS Electronics
' WATCHD.BAS mostra l'uso del timer di watchdog del
' AT89S8252 selezionare 89s8252.dat !!!
'-----------------------------------------------------
Config Watchdog = 2048                  'reset dopo 2048 mSec
Start Watchdog                          'start del timer di watchdog
Dim I As Word
For I = 1 To 10000
  Print I                               'print del valore
  ' Reset Watchdog
  'potete notare che il ciclo for next non sarà completato prima che
  'avvenga un reset, se abilitate l'istruzione Reset Watchdog togliendo
  'l'indicazione di commento sarà possibile vedere terminare il ciclo
  'perché wd-timer sarà resettato prima che raggiunga i 2048 msec
Next
End
```

COUNTERx

Azione

Imposta o legge la variabile COUNTER0 o COUNTER1.
Per microprocessori 8052 e compatibili sono disponibili anche TIMER2 e COUNTER2.

Sintassi

COUNTERX = var oppure
var = **COUNTERX**

Note

var	Una variabile Byte, Integer/Word oppure una costante il cui valore deve essere assegnato al contatore.
counterX	COUNTER0 , COUNTER1 or COUNTER2.

Con l'istruzione COUNTERX = 0 è possibile resettare il contatore.
Il contatore può contare da 0 a 255 in modalità 2 (8-bit auto reload) e fino a 65535 in modalità 1(16-bit).

La variabile COUNTERX è destinata ad impostare o leggere i registri TIMER/COUNTER internamente a BASCOM. COUNTER0 = TL0 e TH0.
Quindi COUNTERx è una variabile riservata lunga 16 bit.

Per impostare TLx o THx, può essere usato ad esempio : TL0 = 5.

Con COUNTERX è possibile assegnare entrambi i TIMERS e COUNTER poiché i TIMERS e i COUNTERS sono esattamente la stessa cosa, ad eccezione del loro proprio modo di lavorare. Per caricare o ricaricare un valore può essere impiegata l'istruzione LOAD.

Esempio

```
'-----------------------------------------------------------
'                     (c) 1997,1998 MCS Electronics
'-----------------------------------------------------------
'  file: COUNTER.BAS
'  demo: COUNTER
'-----------------------------------------------------------
'     Collegare l'ingresso di timer P3.4 ad un generatore di
'     frequenza
'     *TIMER/COUNTER 1 è usato come generatore di baudrate
'-----------------------------------------------------------
Dim A As Byte , C As Integer
Config Timer0 = Counter , Gate = Internal , Mode = 1
'Timer0 = counter : timer0 operates as a counter
'Gate = Internal  : no external gate control
'Mode = 1         : 16-bit counter

Counter0 = 0                              'azzera il counter
Start Counter0                            'abilita il counter
Do                                        'avvia un loop
  A = Inkey                               'attende un input
  C = Counter0                            'preleva il valore del counter
  Print C                                 'print del valore di counter
Loop Until A = 27                         'fino al tasto escape

End

Per l'esempio seguente è mostrato il codice ASM:
```

```
COUNTER0 = 1000

Codice generato :
Clr TCON.4
Mov tl0,#232
Mov th0,#3
```

CPEEK()

Azione
Restituisce un byte dalla memoria di programma.

Sintassi
var **= CPEEK(** address **)**

Note

var	Variabile numerica cui è assegnato il contenuto della locazione indirizzata da address
address	Variabile numerica o costante che contiene l'indirizzo della locazione

Non esiste istruzione CPOKE poiché è impossibile scrivere nella memoria di programma.

Vedere anche
PEEK , POKE , INP , OUT

Esempio
```
'------------------------------------------------------
'              (c) 1998 MCS Electronics
'                   PEEK.BAS
' mostra l'uso di PEEK, POKE, CPEEK, INP and OUT
'
'------------------------------------------------------
Dim I As Integer , B1 As Byte

'dump internal memory
For I = 0 To 127                        'per un 8052 si può usare 225
'  Break
  B1 = Peek(i)                          'prende un byte dalla memoria interna
  Printhex B1 ; "  ";
  'Poke I , 1                           'scrive un valore in memoria
Next
Print                                   'nuova linea
'attenzione quando si scrive nella memoria interna !!
```

CURSOR

Azione
Imposta lo stato del cursore sul display LCD.

Sintassi
CURSOR ON / OFF BLINK / NOBLINK

Note
Possono essere usati entrambi i parametri ON / OFF e BLINK / NOBLINK.
All'alimentazione lo stato del cursore è definito ON e NOBLINK.

Vedere anche
DISPLAY

Esempio
```
Dim a as byte
a = 255
LCD a
CURSOR OFF            'nasconde il cursor
Wait 1                'attende 1 secondo
CURSOR BLINK          'cursore lampeggiante
End
```

DATA

Azione
Specifica i valori che saranno letti successivamente con l'istruzione READ.

Sintassi
DATA var [, varn]

Note

var	Costante numerica o Stringa.

Differenze da QB
Costanti di tipo Integer e Word devono terminare con il segno **%**.
Costanti di tipo Long devono terminare con il segno **&**.
Costanti di tipo Single devono terminare con il segno **!**

Vedere anche
READ , RESTORE

Esempio
```
DIM a AS BYTE, I AS BYTE, L AS Long, S As XRAM  STRING * 15
RESTORE DTA          'azzera il puntatore dati
FOR a = 1 TO 3
  READ a : PRINT a   'legge i dati e li visualizza
NEXT
RESTORE DTA2         'azzera il puntatore dati
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%
'Costanti di tipo Integer e Word devono terminare con il segno %
(Integer : <0 or >255)

DTA3:
DATA 1235678&
'Costanti di tipo Long devono terminare con il segno &

DTA4:
DATA "Hello world"
REM Sulla stessa linea possono coesistere costanti di diverso tipo
DATA "TEST" , 5 , 1000% , -1& , 1.1!
```

DEBOUNCE

Azione
Esegue un filtro antirimbalzo (debounce) su un pin di porta collegato ad un contatto.

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

Note

Px.y	Pin di porta da esaminare, ad esempio P1.0.
state	0 esegue il salto quando Px.y è basso, 1 quando Px.y è alto.
label	La label da raggiungere (GOTO) quando lo stato viene rilevato.
SUB	La label da raggiungere (GOSUB) quando lo stato viene rilevato.

Quando si specifica il parametro opzionale SUB, il salto alla label avviene come per un'istruzione GOSUB invece di GOTO.
L'istruzione DEBOUNCE attende che un pin di porta sia alto (1) o basso (0).
Quando il pin si porta nello stato richiesto attende 25 mS ed esegue nuovamente una lettura del pin (questo elimina i rimbalzi prodotti da un contatto).
Quando la condizione viene riscontrata coerente in entrambe le letture, e non ci sono stati salti precedenti, il programma raggiunge la label specificata.
Quando DEBOUNCE viene nuovamente eseguito, il contatto dovrà prima essere tornato a riposo affinché possa avvenire un nuovo salto alla label specificata.
Ciascun pin associato all'istruzione di DEBOUNCE comporta l'utilizzo di 1 BIT della memoria interna per memorizzarne lo stato.

Deve anche essere ricordato che P2.2-P2.7 e P3 hanno resistori interni di pull up.
Questo può influire sull'istruzione di debounce. Con questi pin di porta è preferibile impiegare l'istruzione di debounce nel seguente modo: **Debounce P1.1, 0, Pr [, sub]** ,
così che il salto alla label non si produca accidentalmente a causa del resistore di pull up.

Vedere anche
CONFIG DEBOUNCE

Esempio
```
'-----------------------------------------------------
'                  DEBOUN.BAS
'           mostra l'uso di DEBOUNCE
'-----------------------------------------------------
CONFIG DEBOUNCE = 30        'senza istruzione config è usato il default di 25mS
Do
  'Debounce P1.1 , 1 , Pr    'questo per saltare quando l'ingresso è alto (1)
  Debounce P1.0 , 0 , Pr,SUB
  '                    ^----- label alla quale saltare
  '                ^---------- salta quando P1.0 diventa basso (0)
  '          ^---------------- analizza P1.0

  'quando P1.0 diventa basso salta alla subroutine Pr
  'P1.0 deve tornare alto prima di abilitare un altro
  'salto alla label Pr
Loop
End

Pr:
  Print "P1.0 era/è basso"
Return
```

DECR

Azione
Decrementa una variabile di uno.

Sintassi
DECR var

Note

Var	Variabile numerica da decrementare.

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

Si verificano spesso situazioni in cui sia richiesto di decrementare di uno un numero.
L'istruzione **DECR** consente questa operazione in modo più rapido del classico var = var - 1.

Vedere anche
INCR

Esempio
```
'-----------------------------------------------------------
'                     (c) 1997,1998 MCS Electronics
'-----------------------------------------------------------
'  file: DEC.BAS
'  demo: DECR
'-----------------------------------------------------------
Dim A As Byte

A = 5                   'assegna un valore ad a
Decr A                  'decrementa (di uno)
Print A                 'print del valore
End
```

DECLARE SUB

Azione
Dichiara una subroutine.

Sintassi
DECLARE SUB TEST[(var as type)]

Note

test	Nome della procedura.
Var	Nome della/e variabile/i. Fino ad un massimo di 10.
Type	Tipo della/e variabile/i. Bit, Byte,Word/Integer, Long o String.

E' necessario dichiarare ogni subroutine prima di scrivere la procedura che dovrà essere eseguita nella subroutine.

Vedere anche
CALL, SUB

Esempio
```
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

Azione
Dichiara il tipo per tutte le variabili non dimensionate.

Sintassi
DEFBIT b
DEFBYTE c
DEFINT I
DEFWORD x

Differenze da QB
QB permette di specificare un campo, per esempio DEFINT A - D. In BASCOM questo non è permesso.

Esempio
```
Defbit b : DefInt c     'tipo di default per bit e integers
Set b1                  'set bit a 1
c = 10                  'imposta c = 10
```

DEFLCDCHAR

Azione
Definisce un carattere speciale personalizzato per il display LCD.

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

Note

char	Variabile che rappresenta il carattere (0÷7).
r1-r8	Valore assegnato a ciascuna riga del carattere.

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

E' possibile utilizzare LCD designer per costruire in forma grafica un carattere.

E' importante che dopo una o più istruzioni DEFLCDCHAR segua un CLS.

Un carattere speciale così definito potrà essere visualizzato con la funzione Chr().

Vedere anche
Edit LCD designer

Esempio
```
DefLCDchar 0,1,2,3,4,5,6,7,8'define special character
Cls                                 'seleziona la RAM DATI del display LCD
LCD Chr(0)                          'mostra il carattere
End
```

DELAY

Azione
Produce un breve ritardo nell’esecuzione del programma.

Sintassi
DELAY

Note
DELAY è indicato per produrre brevi ritardi.
Il ritardo prodotto è pari a 100 microsecondi se s’impiega un clock di 12 MHz.

Vedere anche
WAIT , WAITMS

Esempio
```
P1 = 5          'scrive 5 sulla porta 1
DELAY           'attende che l’hardware sia pronto
```

DIM

Azione
Dimensiona una variabile.

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

Note

var	Qualsiasi nome valido per una variabile: b1, i, nomelungo. Var può anche essere un array : ar(10).
type	Bit, Byte, Word, Integer, Long, Single or String
XRAM	XRAM se la variabile va memorizzata nella memoria esterna
IRAM	IRAM se va memorizzata nella memoria interna (default)

Una variabile di tipo string richiede un parametro aggiuntivo che ne definisca la lunghezza :
Dim s As XRAM **String * 10**
In questo caso string potrà avere una lunghezza di 10 caratteri.

Variabili di tipo BIT sono sempre salvate nella memoria interna.

Differenze da QB
In QB non è necessario dimensionare ciascuna variabile s'intenda usare.
In BASCOM è obbligatorio dimensionare ogni variabile prima che sia utilizzata.
XRAM/IRAM sono opzioni non disponibili in QB.

See Also
CONST , ERASE

Esempio
```
'-----------------------------------------------------------
'                    (c) 1997-1999 MCS Electronics
'-----------------------------------------------------------
'  file: DIM.BAS
'  demo: DIM
'-----------------------------------------------------------
Dim B1 As Bit                             'bit può essere 0 o 1
Dim A As Byte                             'byte valido tra 0 ÷ 255
Dim C As Integer                          'integer valido tra -32767 ÷ +32768
Dim A As String * 10                      'stringa lunga 10 caratteri


Dim ar(10) As Byte                        'dimensiona un array
'assegna bits
B1 = 1                                    'oppure
Set B1                                    'usa set

'assegna bytes
A = 12
A = A + 1

'assegna integer
```

```
C = -12
C = C + 100
Print C
End
```

DISABLE

Azione
Disabilita una specifica interrupt.

Sintassi
DISABLE interrupt

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

Per default tutte le interrupts sono disabilitate.
Per disabilitare TUTTE le interrupts è sufficiente specificare INTERRUPTS.
Per attivare la possibilità di abilitazione e disabilitazione di ciascuna interrupt deve essere specificato ENABLE INTERRUPTS.

Possono essere gestite più o meno interrupts in funzione del chip selezionato.
Vedere alle specifiche dei microprocessori per ulteriori dettagli.

Vedere anche
ENABLE

Esempio
```
ENABLE INTERRUPTS    'abilita il settaggio delle interrupts
ENABLE TIMER0        'abilita TIMER0
DISABLE SERIAL       'disabilita l'interrupt seriale
DISABLE INTERRUPTS   'disabilita tutte le interrupts
```

DISPLAY

Azione
Accende e spegne il display LCD.

Sintassi
DISPLAY ON / OFF

Note
All’alimentazione il display è acceso.

Vedere anche
-

Esempio
```
Dim a as byte
a = 255
LCD a
DISPLAY OFF
Wait 1
DISPLAY ON
End
```

DO .. LOOP

Azione
Ripete un blocco d’istruzioni sino a quando è verificata una condizione.

Sintassi
DO
  istruzioni
LOOP [UNTIL expression **]**

Note
Si può abbandonare un ciclo DO..LOOP non ancora completato con l’istruzione EXIT DO.

Vedere anche
EXIT , WHILE WEND , FOR , NEXT

Esempio
```
Dim A As Byte
DO                          'avvia il loop
   A = A + 1                'incrementa A
   PRINT A                  'lo visualizza
LOOP UNTIL A = 10           'Ripete il loop finchè  A = 10
Print A                     'A rimarrà 10
```

ELSE

Azione
Esegue un'istruzione alternativa quando l'espressione IF-THEN risulta falsa.

Sintassi
ELSE

Note
L'istruzione ELSE non deve essere utilizzata nelle strutture IF THEN .. END IF.
Per verificare ulteriori condizioni può essere impiegata l'istruzione ELSEIF.

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

Vedere anche
IF , END IF SELECT CASE

Esempio
```
A = 10                                 'imposta a = 10
IF A > 10 THEN                         'prendi una decisione
   PRINT "  A >10"                     'questo non sarà visualizzato
ELSE                                   'in alternativa
   PRINT " A not greater than 10"      'questo sarà visualizzato
END IF
```

ENABLE

Azione
Abilita una specifica interrupt.

Sintassi
ENABLE interrupt

Note

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

Per default tutte le interrupt sono disabilitate.
Per attivare la possibilità di abilitazione e disabilitazione di ciascuna interrupt deve essere specificato ENABLE INTERRUPTS.

Possono essere gestite più o meno interrupts in funzione del chip selezionato.
Vedere alle specifiche dei microprocessori per ulteriori dettagli.

Vedere anche
DISABLE

Esempio
```
ENABLE INTERRUPTS           'permette il settaggio delle interrupts
ENABLE TIMER1               'abilita l'interrupt di TIMER1
```

END

Azione
Termina l’esecuzione di un programma.

Sintassi
END

Note
Per terminare un programma può essere impiegata anche l’istruzione **STOP**.

Quando si raggiunge un’istruzione END o STOP viene prodotto un loop senza fine.

Vedere anche
STOP

Esempio
```
PRINT " Hello"        'visualizza questo
END                   'fine dell’esecuzione
```

END IF

Azione
Termina una struttura IF .. THEN.

Sintassi
END IF or **ENDIF**

Note
E' indispensabile chiudere una struttura IF .. THEN utilizzando l'istruzione END IF.

E' possibile nidificare istruzioni IF ..THEN (nesting).
ELSE è opzionale.

Se è abilitata l'opzione "reformat" l'editor converte ENDIF in End If.

Esempio
```
Dim nmb As Byte
AGAIN:                          'label
INPUT " Number "  , nmb        'chiede un numero
IF a = 10 THEN                  'lo compara
   PRINT " Number is 10"        'corrisponde
ELSE                            'non corrisponde
    IF nmb > 10 THEN            'è maggiore ?
         PRINT " Number > 10"   'si
    ELSE                        'no
         PRINT " Number < 10"   'visualizzalo
    END IF                      'fine della struttura
END IF                          'fine della struttura
END                             'fine del programma
```

ERASE

Azione
Cancella una variabile liberando la memoria utilizzata.

Sintassi
ERASE var

Note
var Nome della variabile da cancellare.
Prima di potere cancellare una variabile, questa deve essere stata dimensionata.

Quando risulti conveniente fare uso di variabili temporanee è possibile, una volta utilizzate, cancellarle e liberare spazio in memoria. Questo consente un risparmio di memoria.

E' possibile cancellare solo le variabili dimensionate con l'ultima istruzione DIM. Quindi quando siano dimensionate 2 variabili per uso temporaneo, queste sono le variabili che possono essere cancellate. L'ordine con cui vengono cancellate non è rilevante.

Esempio :
Dim a1 as byte , a2 as byte , a3 as byte , a4 as byte
'utilizzo delle variabili
ERASE a3 : ERASE a4 'cancellazione delle 2 ultime variabili perché temporanee
Dim a5 as Byte 'Dimensionamento di una nuova variabile
Ora non sarà più possibile cancellare le variabili a1 ed a2 !

Le variabili cancellate non vengono riportate né nel report file né nel simulatore.

Esempio
```
DIM A As Byte         'dimensiona una variabile
A = 255               'assegna un valore
Print A               'visualizzalo
ERASE A               'elimina la variabile
DIM A AS INTEGER      'ridimensiona ma come INT
PRINT A               'visualizzalo nuovamente
REM A usa lo stesso spazio precedentemente assegnato alla variabile A
REM cancellata quindi contiene o stesso valore precedentemente assegnato
```

EXIT

Azione

Esce da un FOR..NEXT, DO..LOOP , WHILE ..WEND o SUB..END SUB.

Sintassi

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

Note

Con l'istruzione EXIT ... è possibile uscire in qualsiasi momento da una struttura.

Esempio

```
IF a >= b1 THEN              'del codice inutile, per dimostrare ...
   DO                        'innizia DO..LOOP
       A = A + 1             'incrementa a
       IF A = 100 THEN       'analizza se a = 100
          EXIT DO            'esce da DO..LOOP
       END IF                'fine della struttura IF..THEN
   LOOP                      'fine della struttura DO
END IF                       'fine della struttura IF..THEN
```

FOR

Azione
Esegue un blocco d'istruzioni per un certo numero di volte.

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

Note

var	Variabile da utilizzare come contatore
start	Valore iniziale della variabile var
end	Valore finale della variabile var
value	Valore dell'incremento o decremento da applicare alla variabile var ogni volta che viene eseguito 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.

Per loop in incremento si deve utilizzare TO.
Per loop in decremento si deve utilizzare DOWNTO.
Una struttura FOR deve terminare con la propria istruzione NEXT.
STEP è opzionale e può essere omesso, per default il suo valore è 1.

Vedere anche
NEXT , EXIT FOR

Esempio
```
y = 10                      'pone y = 10
FOR a = 1 TO 10             'ripete 10 volte
       FOR x = y TO 1       'anche questo
              PRINT x ; a   'visualizza i valori
       NEXT                 'x successivo (conta indietro)
NEXT                        'a successivo (conta avanti)


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

FOURTHLINE

Azione
Posiziona il cursore del display LCD a capo della quarta linea.

Sintassi
FOURTHLINE

Note
Valido solo per display LCD a 4 linee.

Vedere anche
HOME , UPPERLINE , LOWERLINE , THIRDLINE , LOCATE

Esempio
```
Dim a as byte
a = 255
LCD a
Fourthline
LCD a
Upperline
END
```

FUSING

Azione
Formatta un valore in virgola mobile.

Sintassi
var = Fusing(source, mask)

Note

var	La stringa che riceverà il risultato.
source	Variabile di tipo single che deve essere formattata.
mask	Maschera di formattazione. ###.## Il segno # è utilizzato per indicare il numero di cifre prima e dopo il punto decimale. Il risultato verrà arrotondato.

Vedere anche
STR

Esempio
```
$large
Dim X As Single , Y As Single , Result As Single
Dim I As Integer
Dim Buf As String * 16
Input "Enter x " , X                                    'chiede 2 valori
Input "Enter y " , Y
Print "X+Y=" ; : Result = X + Y : Print Result         'calcola
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 , #.##)                             'formatta una stringa
Print Buf                                               'e la visualizza
```

GETRC

Azione
Acquisisce il valore di un resistore o condensatore.

Sintassi
var = GETRC(pin)

Note

var	La stringa che riceverà il risultato.
pin	Pin di porta cui è connesso R/C.

Vedere anche

{bmc Getrc.bmp}

Esempio

```
'-----------------------------------------------------------------
'                            GETRC.BAS
'  acquisisce il valore di un resistore
'  Connettere un resistore variabile da 10KOhm tra +5V e P1.7
'  Connettere un condensatore da 10nF tra P1.7 e ground
'  L'istruzione GETRC(pin) misura il tempo necessario alla
'  carica del condensatore
'-----------------------------------------------------------------
Config Timer0 = Timer , Gate = Internal , Mode = 1      'GETRC() usa timer 0
$baud = 9600                                            'possibile settaggio
$crystal = 11059200
Dim W As Word                               'alloca spazio per la variabile

Do                                          'in continuazione
  W = Getrc(p1.7)                           'acquisisci il valore di RC
  Print W                                   'e stampalo
  Wait 1                                    'attendi un attimo
Loop

'valori restituiti con cap=10nF. Valori di resistenza misurati 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)

'Come si può riscontrare la conversione è ragionevolmente lineare, quindi è
'possibile usare qualche relazione matematica per risalire al valore di R/C
```

```
'La funzione è comunque intesa come indicazione grossolana di valori di R
'Si può intervenire sul valore di C per ottenere valori maggiori.
'Con 10nF, il valore reso occupa un byte
'Naturalmente R o C devono essere noti per determinare l’altro valore.
```

GETRC5

Azione
Acquisisce code e subaddress da un dispositivo infrarosso RC5.

Sintassi
GETRC5(address , command)

Note

address	Address (indirizzo) ricevuto da RC5.
command	Comando (command) ricevuto da RC5.

Per utilizzare questa funzione è necessario collegare un dispositivo ricevitore infrarosso SHARP SFH506-36 al pin di porta 3.2.
Questa istruzione impiega l'interrupt INT0. Vedere l'impiego corretto nell'esempio seguente.

{bmc sfh506.bmp}

Esempio
```
'-----------------------------------------------------------------------------
'                       RC5.BAS  (c) 1999 MCS Electronics
'  connettere 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                                          'nuovo codice ricevuto
     Print Command ; "  " ; Subaddress
     New = 0                                               'resetta il bit new
  End If
Loop



Receiverc5:                                                ' routine di interrupt
  Getrc5(Subaddress, command)
  New = 1
Return
```

GOSUB

Azione
Salta all'esecuzione di una subroutine.

Sintassi
GOSUB label

Note

label	Nome della label cui saltare.

Usando GOSUB, il programma prosegue dal punto indicato dalla label. Raggiunta un'istruzione RETURN, l'esecuzione del programma riprende dopo l'istruzione GOSUB.

Vedere anche
GOTO CALL

Esempio
```
GOSUB Routine          'salta alla routine
Print "Hello"          'al ritorno dalla routine
END                    'termina il programma

Routine:               'questa è una subroutine
       x = x + 2       'esegue qualche calcolo
       PRINT X         'visualizza il risultato
RETURN                 'ritorna
```

GOTO

Azione
Salta alla label specificata.

Sintassi
GOTO label

Note
Le Labels possono avere una lunghezza fino a 32 caratteri.
Il compilatore avvisa se una label viene duplicata nel corso del programma.

Vedere anche
GOSUB

Esempio
```
Start:                'una label deve terminare con due punti
A = A + 1             'incrementa a
IF A < 10 THEN        'è inferiore a 10?
       GOTO Start     'di nuovo
END IF                'chiudi IF
PRINT " Ready"        'è tutto
```

HEX()

Azione
Restituisce un numero esadecimale sotto forma di stringa.

Sintassi
var = Hex(x)

Note

var	Variabile di tipo String.
X	Variabile numerica di tipo Byte, Integer or Word.

Vedere anche
HEXVAL

Esempio
```
Dim a as Byte, S as String * 10
a = 123
s  = Hex(a)
Print s
End
```

HEXVAL()

Azione
Restituisce il valore esadecimale dall'analoga rappresentazione in forma di stringa.

Sintassi
var **= HEXVAL(** x **)**

Note

var	Variabile numerica cui assegnare il valore.
X	Stringa contenente la rappresentazione esadecimale.

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

La stringa da convertire deve avere una lunghezza di 2 bytes, 4 bytes o 8 bytes, per Bytes, Integers/Words and Longs rispettivamente.

Differenze da QB
In QB l'istruzione VAL() consente questo tipo di conversione, facendo uso dell'identificativo &H che specifica il formato numerico. Questo richiede un test ulteriore quindi per maggiore efficienza è stata prevista una funzione separata.

Vedere anche
HEX , VAL , STR

Esempio
```
Dim a as Integer, s as string * 15
s = "000A"
a = Hexval(s) : Print a
End
```

HIGH

Azione
Recupera il byte più significativo di una variabile.

Sintassi
var = HIGH (s)

Note

var	Variabile cui sarà assegnato il MSB della variabile S.
s	Variabile da cui prelevare MSB: byte più significativo.

Vedere anche
LOW

Esempio
```
Dim I As Integer , Z As Byte
I = &H1001
Z = High(I)  ' è 16
```

HOME

Azione
Posiziona il cursore alla locazione 1 della linea specificata.

Sintassi
HOME UPPER / LOWER /THIRD / FOURTH

Note
Se viene utilizzato solo HOME il cursore sarà posizionato alla prima riga.
E’ ammessa l’abbreviazione della linea impiegando la sola iniziale, ad esempio HOME U per indicare HOME UPPER.

Vedere anche
CLS , LOCATE , LCD

Esempio
```
Lowerline
LCD " Hello"
Home Upper
LCD " Upper"
```

I2CRECEIVE

Azione
Acquisisce dati da un dispositivo seriale sul bus I2C.

Sintassi
I2CRECEIVE slave, var
I2CRECEIVE slave, var ,b2W, b2R

Note

slave	Variabile o Costante di tipo Byte, Word/Integer variable or constant contenente l'indirizzo del dispositivo slave I2C.
Var	Variabile di tipo Byte or Integer/Word che riceverà le informazioni provenienti dal dispositivo I2C indirizzato.
b2W	Numero di bytes da scrivere. Attenzione a non specificare un eccessivo numero di bytes !
b2R	Numero di bytes da ricevere. Attenzione a non specificare un eccessivo numero di bytes !

In BASCOM LT è possibile definire DATA come var ma, giacchè sono supportati, è preferibile definire un array piuttosto che un DATA.

Questo comando richiede hardware addizionale. Vedere appendice D.

Vedere anche
I2CSEND

Esempio
```
x = 0                                   'reset della variabile
slave = &H40                            'indirizzo slave di un PCF 8574 (I/O)
I2CRECEIVE  slave, x                    'prendi il valore
PRINT x                                 'visualizzalo


Dim buf(10) as String
buf(1) = 1 : buf(2) = 2
I2CRECEIVE slave, buf(), 2, 1           'invia due bytes e ne riceve uno
Print buf(1)                            'visualizza il byte ricevuto
```

I2CSEND

Azione
Invia dati ad un dispositivoI2C.

Sintassi
I2CSEND slave, var
I2CSEND slave, var , bytes

Note

slave	Indirizzo del dispositivo slave I2C.
var	Byte, Integer/Word or numero che rappresenta il valore da inviare al dispositivo I2C.
bytes	Numero di bytes da inviare.

Questo comando richiede hardware addizionale. Vedere appendice D.

Vedere anche
I2CRECEIVE

Esempio
```
x = 5                                   'assegna 5 alla variabile
Dim ax(10) As Byte
slave = &H40                            'indirizzo slave di un PCF 8574 (I/O)
bytes = 1                               'invia 1 byte
I2CSEND slave, x                        'invia il valore  o


For a = 1 to 10
    ax(a) = a                           'Riempi lo spazio dati
Next
bytes = 10
I2CSEND slave,ax(),bytes
END
```

I2START,I2CSTOP, I2CRBYTE, I2CWBYTE

Azione

I2CSTART genera una condizione di start sul bus I2C.
I2CSTOP genera una condizione di stop sul bus I2C.
I2CRBYTE riceve un byte da un dispositivo I2C.
I2CWBYTE invia un byte ad un dispositivo I2C.

Sintassi

I2CSTART
I2CSTOP
I2CRBYTE var, 8/9
I2CWBYTE val

Note

var	Variabile che riceve il valore dal dispositivo I2C.
8/9	Specificare 8 o ACK se esistono ancora bytes da leggere. (**ACK**) Specificare 9 o NACK se è l'ultimo byte da leggere. (**NACK**)
val	Variabile o costante da scrivere nel dispositivo I2C.

Questo comando richiede hardware addizionale. Vedere appendice D.

Queste istruzioni sono previste come complemento alle funzioni I2CSEND e I2CRECEIVE.

Vedere anche

I2CRECEIVE I2CSEND

Esempio

```
-------- Lettura e scrittura di un byte in una EEPROM 2404 -----------------
DIM a As Byte
DIM adresW AS CONST 174      'comando write per 2404
DIM adresR AS CONST 175      'indirizzo read per 2404
I2CSTART                     'genera il comando di start del bus I2C
I2CWBYTE  adresW             'invia l'indirizzo dello slave
I2CWBYTE  1                  'invia l'indirizzo della EEPROM da scrivere
I2CWBYTE  3                  'invia il valore
I2CSTOP                      'genera il comando di stop del bus I2C
WaitMS 10                    'attende 10 mS necessari al chip per la scrittura

-----------ora rilegge il valore assegnandolo alla var a --------------------
I2CSTART              'genera il comando start del bus I2C
I2CWBYTE adresW       'invia l'indirizzo dello slave
I2CWBYTE 1            'invia l'indirizzo della EEPROM da leggere
I2CSTART              'ripete il comando start
I2CWBYTE adresR       'invia l'indirizzo slave della EEPROM
I2CRBYTE a,9          'riceve il valore in a. 9 perchè è l'ultimo byte da ricevere
I2CSTOP               'genera il comando di stop del bus I2C
PRINT a               'visualizza il valore ricevuto
END
```

IDLE

Azione
Forza il processore nello stato idle.

Sintassi
IDLE

Note
Quando il processore funziona in modo IDLE il clock di sistema viene rimosso dalla CPU ma non dalla logica che gestisce le interrupts, anche quelli provenienti dalla porta seriale e dai timers/counters. Il processore rientra in funzionamento normale quando riceve un’interrupt oppure un RESET hardware.

Vedere anche
POWERDOWN

Esempio
```
IDLE
```

IF

Azione
Consente l'esecuzione condizionata di salti basata sulla valutazione di un'espressione.

Sintassi
IF espressione **THEN**

[**ELSEIF** espressione **THEN**]
[**ELSE**]

END IF

Note

espressione	Qualsiasi espressione stimabile come vera o falsa.

E' ora possibile usare la forma su singola linea di IF :
IF espressione THEN istruzione [ELSE istruzione]
[ELSE] è opzionale.

E' ora possibile valutare anche un singolo bit come espressione :
IF var.bit = 1 THEN

Vedere anche
ELSE , END IF

Esempio
```
DIM A AS INTEGER
A = 10
IF A = 10 THEN                                    'espressione di prova
       PRINT " Questa parte viene eseguita."      'questo sarà visualizzato
ELSE
       PRINT " Qesta parte non viene eseguita."   'questo no
END IF
IF A = 10 THEN PRINT "Nuovo in BASCOM"
IF A = 10 THEN GOTO LABEL1 ELSE PRINT "A<>10"
LABEL1:

REM L'esempio seguente mostra l'uso avanzato di of IF THEN
IF A.15 = 1 THEN                                  'analizza un bit
   PRINT "BIT 15 = 1"
END IF
REM L'esempio seguente mostra l'uso su una sola linea di IF THEN [ELSE]
IF A.15 = 0 THEN PRINT "BIT 15 is cleared" ELSE PRINT "BIT 15 is set"
```

INCR

Azione
Incrementa una variabile di uno.

Sintassi
INCR var

Note

Var	Variabile numerica da incrementare.

Si verificano spesso situazioni in cui sia richiesto di incrementare di un numero.
L'istruzione **INCR** consente questa operazione in modo più rapido del classico var = var + 1.

Vedere anche
DECR

Esempio
```
DO                  'start del loop
       INCR a       'incrementa a di 1
       PRINT a      'visualizza a
LOOP UNTIL a > 10   'ripeti fnchè a è maggiore di 10
```

INKEY

Azione
Restituisce il valore ASCII del primo carattere presente nel buffer d'ingresso seriale.

Sintassi
var = **INKEY**

Note

var	Variabile di tipo Byte, Integer, Word, Long o String.

Se non sono presenti caratteri sarà restituito uno zero.

INKEY può essere impiegata solo se presente un'interfaccia seriale (RS-232).
Consultare il manuale per l'integrazione di un'interfaccia seriale.
L'interfaccia RS-232 può essere collegata ad una porta di comunicazione di un PC.

Vedere anche
WAITKEY

Esempio
```
DO                          'start del loop
       A = INKEY            'attesa di un carattere
       IF A > 0 THEN        'la variabile è > 0?
          PRINT A           'si , allora visualizzala
       END IF
LOOP                        'loop senza fine
```

INP()

Azione

Restituisce un byte letto da una porta hardware o da una locazione di memoria esterna.

Sintassi

var = **INP**(address)

Note

var	Variabile numerica alla quale sarà assegnato il valore.
address	Indirizzo dal quale leggere il valore.

L’istruzione INP può lavorare solo su sistemi con microprocessore in grado di indirizzare memoria esterna.

Vedere anche

OUT

Esempio

```
Dim a As Byte
a = INP(&H8000)       'legge un valore dal bus dati (d0÷d7)
                      'all’indirizzo hex 8000
PRINT a
END
```

INPUTBIN

Azione
Legge un valore binario dalla porta seriale.

Sintassi
INPUTBIN var1 [,var2]

Note

var1	Variabile alla quale sarà assegnato il valore letto dalla porta seriale.
var2	Seconda (o più) variabile opzionale alla quale assegnare ulteriori caratteri letti dalla porta seriale.

Il numero di bytes da leggere dipende dal tipo di variabile impiegata.
Con una variabile di tipo Byte, dalla porta seriale sarà letto un carattere.
Una variabile di tipo Integer si aspetterà 2 caratteri ed un array fino al completo riempimento dell'array stesso.

L'istruzione INPUTBIN riceverà l'esatto numero di bytes indicati, senza necessità di attendere un <RETURN> .

Vedere anche
PRINTBIN

Esempio
```
Dim a as Byte, C as Integer
INPUTBIN a, c        'si aspetta 3 caratteri
End
```

INPUTHEX

Azione
Consente l'introduzione di dati dalla tastiera durante l'esecuzione del programma.

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

Note

prompt	Stringa opzionale da visualizzare prima del carattere prompt.
Var,varn	Variabile/i numerica alla quale assegnare il valore introdotto.
NOECHO	Disabilita l'eco verso la porta di comunicazione.

L'istruzione INPUTHEX può essere utilizzata quando esiste un'interfaccia seriale RS-232.
Consultare il manuale per informazioni sulla realizzazione di un'interfaccia RS-232.
L'interfaccia RS-232 può essere collegata alla porta di comunicazione seriale di un PC.
In questo modo è possibile impiegare un emulatore di terminale ed una tastiera come dispositivo d'input.
In BASCOM è integrato un emulatore di terminale.

Se var è di tipo Byte devono essere digitati 2 caratteri.
Se var è di tipo Integer/Word devono essere digitati 4 caratteri.
Se var è di tipo Long devono essere digitati 8 caratteri.

Differenze da QB
In QB è possibile specificare l'opzione &H per accettare caratteri in formato esadecimale.
BASCOM impiega una specifica istruzione per questo scopo : INPUTHEX.

Vedere anche
INPUT

Esempio
```
Dim  x As Byte
INPUTHEX " Enter a number " , x     'chiede un input
```

INPUT

Azione
Consente l'introduzione di dati dalla tastiera durante l'esecuzione del programma.

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

Note

prompt	Stringa opzionale da visualizzare prima del carattere prompt.
Var,varn	Variabile/i numerica alla quale assegnare il valore introdotto.
NOECHO	Disabilita l'eco verso la porta di comunicazione.

L'istruzione INPUT può essere utilizzata quando esiste un'interfaccia seriale RS-232. Consultare il manuale per informazioni sulla realizzazione di un'interfaccia RS-232. L'interfaccia RS-232 può essere collegata alla porta di comunicazione seriale di un PC. In questo modo è possibile impiegare un emulatore di terminale ed una tastiera come dispositivo d'input. In BASCOM è integrato un emulatore di terminale.

Differenze da QB
In QB è possibile specificare l'opzione &H per accettare caratteri in formato esadecimale. BASCOM impiega una specifica istruzione per questo scopo : INPUTHEX.

Vedere anche
INPUTHEX PRINT

Esempio
```
'-----------------------------------------------------------
'                 (c) 1997,1998 MCS Electronics
'-----------------------------------------------------------
'  file: INPUT.BAS
'  demo: INPUT, INPUTHEX
'-----------------------------------------------------------
'Per diversi baudrate e frequenze di quarzo usare le
'direttive $BAUD =  e $CRYSTAL =
$baud = 1200                            'prova ad esempio 1200 baud
$crystal = 12000000                     '12 MHz

Dim V As Byte , B1 As Byte
Dim C As Integer , D As Byte
Dim S As String * 15                    'solo per uP che supportano XRAM

Input "Use this to ask a question " , V
Input B1                                'omettere se non serve

Input "Enter integer " , C
Print C

Inputhex "Enter hex number (4 bytes) " , C
Print C
Inputhex "Enter hex byte (2 bytes) " , D
Print D

Input "More variables " , C , D
Print C ; " " ; D

Input C Noecho                          'sopprime l'echo
```

```
Input "Enter your name " , S
Print "Hello " ; S

Input S Noecho                          'senza echo
Print S
End
```

LCD

Azione
Invia una costante oppure una variabile al display LCD.

Sintassi
LCD x

Note

x	Variabile o costante da visualizzare.

Possono essere mostrate più variabili, separandole tra loro con il segno **;** nel seguente modo
LCD a ; b1 ; "costante"
L'istruzione LCD si comporta come l'istruzione PRINT.

Vedere anche
LCDHEX , $LCD CONFIG LCD

Esempio
```
'-----------------------------------------------------------
'                 (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 il tipo di lcd
'altre opzioni sono 16 * 4 and 20 * 4, 20 * 2
'Omettendo la configurazione si assume per default 16 * 2

'$LCD = per utilizzare il display LCD ad 8-bit sul bus dati
'       solo per uP con RAM e/o ROM esterne

Cls                                     'clear del display LCD
Lcd "Hello world."                      'mostra questo in prima riga
Wait 1
Lowerline                               'seleziona la riga inferiore
Wait 1
Lcd "Shift this."                       'e mostra questo
Wait 1
For A = 1 To 10
   Shiftlcd Right                       'scorre il testo a destra
   Wait 1                               'attende un attimo
Next

For A = 1 To 10
   Shiftlcd Left                        'scorre il testo a sinistra
   Wait 1                               'attende un attimo
Next

Locate 2 , 1                            'posiziona il cursore
Lcd "*"                                 'mostra questo
Wait 1                                  'attende un attimo
```

```
Shiftcursor Right                       'scorre il cursore
Lcd "@"                                 'mostra questo
Wait 1                                  'attende un attimo

Home Upper                              'seleziona la linea 1 a capo
Lcd "Replaced."                         'sostituisce il testo precedente
Wait 1                                  'attende un attimo

Cursor Off Noblink                      'nasconde il cursore
Wait 1                                  'attende un attimo
Cursor On Blink                         'mostra il cursore
Wait 1                                  'attende un attimo
Display Off                             'spegne il display LCD
Wait 1                                  'attende un attimo
Display On                              'accende il display LCD
'----------------- NEW supporto per LCD a 4-linee ----------------
Thirdline
Lcd "Line 3"
Fourthline
Lcd "Line 4"
Home Third                              'a capo in linea tre
Home Fourth
Home F                                  'è sufficiente anche la sola iniziale
Locate 4 , 1 : Lcd "Line 4"
Wait 1

'Now lets build a special character
'the first number is the characternumber (0-7)
'The other numbers are the rowvalues
'Use the LCD tool to insert this line
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 è necessario dopo deflcdchar
Lcd Chr(0) ; Chr(1)                       'visualizza il carattere speciale

'----------------- Ora usa una routine interna -------------------
Acc = 1                                 'valore in ACC
Call Write_lcd                          'trasferisce al display LCD
End
```

LCDHEX

Azione
Invia una variabile in formato esadecimale al display LCD.

Sintassi
LCDHEX var

Note

var	Variabile da visualizzare.

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

S’intendono applicabili le stesse regole definite per PRINTHEX.

Vedere anche
LCD

Esempio
```
Dim a as byte
a = 255
LCD a
Lowerline
LCDHEX a
End
```

LEFT()

Azione

Restituisce il numero di caratteri specificati di una stringa, partendo da sinistra.

Sintassi

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

Note

var	Stringa alla quale verrà assegnato il risultato.
Var1	Stringa d'origine sulla quale operare.
n	Numero di caratteri da estrarre dalla stringa.

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

Per le operazioni variabili di tipo String, tutte le variabili devono essere dello stesso tipo : internal or external.

Esempio

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

LEN

Azione
Restituisce la lunghezza di una stringa.

Sintassi
var = LEN(string)

Note

var	Variabile numerica alla quale assegnare il valore di lunghezza della stringa.
string	Stringa per la quale si desidera calcolare la lunghezza.

Esempio
```
Dim S As String * 12
Dim A As Byte
S = "test"
A = Len(s)
Print A  ' prints 4
```

LOAD

Azione
Carica il valore di autoreload nel TIMER specificato.

Sintassi
LOAD TIMER , value

Note

TIMER	TIMER0, TIMER1 o TIMER2.
Value	Variabile o valore da caricare.

Quando si utilizza l'istruzione ON TIMERx con il TIMER/COUNTER in modalità 2, è possibile specificare a quale intervallo deve essere generata l'interrupt.
Il valore può essere compreso fra 1÷255 per TIMER0 e TIMER1, mentre per il TIMER2 è possibile specificare valori tra 1÷65535.

L'istruzione LOAD calcola il valore corretto di reload nel seguente modo.
TLx = THx = (256-valore)
Per TIMER2 : RCAP2L = RCAP2H = (65536 - valore)

L'istruzione LOAD non va utilizzata per assegnare o leggere un valore dei timers/counters.
Per questo scopo deve essere impiegata l'istruzione COUNTERx.

Vedere Hardware supplementare per maggiori dettagli.

Esempio
```
LOAD TIMER0, 100              'carica TIMER0 con 100

Genera il codice seguente :
Mov tl0,#h'9C
Mov th0,#h'9C

LOAD TIMER2, 1000
Will generate:
Mov RCAP2L,#24
Mov RCAP2H,#252
```

LOCATE

Azione
Muove il cursore del display LCD alla locazione specificata.

Sintassi
LOCATE y **,** x

Note

x	Costante o variabile relativa alla colonna. (1-64*)
y	Costante o variabile relativa alla linea (1 - 4*)

* dipende dal display LCD impiegato

Vedere anche
CONFIG LCD , LCD , HOME , CLS

Esempio
```
LCD "Hello"
Locate 1,10
LCD "*"
```

LOOKUP

Azione
Restituisce il valore contenuto in una tabella.

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

Note

var	Valore restituito
value	Valore da usare come indice della tabella
label	Label alla quale è presente la tabella dati

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

Differenze da BASCOM LT
In BASCOM LT, l'istruzione lookup lavora esclusivamente con tabelle composte da bytes.
In BASCOM-8051, è possibile impiegare anche Integer, Word, Long e Single.

Vedere anche
LOOKUPSTR

Esempio
```
DIM b1 As Byte ,  I as Integer
b1 = Lookup(1, dta)
Print b1              'Visualizza 2 (su base zero)

I = Lookup(0,DTA2)
End


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

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

LOOKUPSTR

Azione
Restituisce una stringa contenuta in una tabella.

Sintassi
var =LOOKUPSTR(value, label)

Note

var	Stringa restituita
value	Valore da usare come indice della tabella. L'indice parte da 0, quindi il primo elemento della tabella è all'indice 0.
label	Label alla quale è presente la tabella dati

value : Byte, Integer, Word, Long, Costante compresi tra 0÷255

Vedere anche
LOOKUP

Esempio
```
Dim s as string, idx as Byte
idx = 0 : s = LookupStr(idx,Sdata)
Print s              'visualizzerà 'This'
End

Sdata:
Data "This" , "is" ,"a test"
```

LOW

Azione
Restituisce il byte meno significativo di una variabile.

Sintassi
var = LOW (s)

Note

var	Variabile cui sarà assegnato il LSB della variabile S.
s	Variabile da cui prelevare MSB: byte meno significativo

Vedere anche
HIGH

Esempio
```
Dim I As Integer , Z As Byte
I = &H1001
Z = Low(I)  ' is 1
```

LOWERLINE

Azione
Posiziona il cursore del display LCD a capo della linea più bassa.

Sintassi
LOWERLINE

Note
-

Vedere anche
UPPERLINE , THIRDLINE , FOURTHLINE , HOME

Esempio
```
LCD "Test"
LOWERLINE
LCD "Hello"
End
```

MakeBCD()

Azione
Converte una variabile nel corrispondente valore BCD.

Sintassi
var1 = **MAKEBCD**(var2)

Note

var1	Variabile alla quale assegnare il valore convertito.
Var2	Variabile contenente il valore decimale originale.

Quando si fa uso di real time clock per bus I2C è necessario adattare il formato dei dati dal tipo DECIMALE al BCD e questa istruzione serve per un'agevole conversione.

Per stampare il valore bcd di una variabile deve essere utilizzata l'istruzione BCD().

Vedere anche
MAKEDEC BCD()

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

MAKEINT()

Azione
Compatta 2 bytes in un singolo Word o Integer.

Sintassi
varn = MAKEINT(LSB , MSB)

Note

varn	Variabile alla quale sarà assegnato il valore convertito.
LSB	Variabile o costante con il byte meno significativo (LSB).
MSB	Variabile o costante con il byte più significativo (MSB).

Il codice equivalente è :
varn = (256 * MSB) + LSB

Vedere anche
MAKEDEC BCD()

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

End
```

MakeDEC()

Azione

Converte una variabile di tipo Byte o Integer/Word in formato BCD nel corrispondente valore DECIMALE.

Sintassi

var1 = **MAKEDEC**(var2)

Note

var1	Variabile alla quale assegnare il valore convertito.
var2	Variabile origine contenente il valore in BCD.

Quando si fa uso di real time clock per bus I2C è necessario riadattare il formato dei dati dal tipo BCD a DECIMALE e questa istruzione serve per un'agevole conversione.

Vedere anche

MAKEBCD

Esempio

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

MID()

Azione

L'istruzione MID restituisce parte di una stringa (substring).
L'istruzione MID può essere usata per sostituire parte di una stringa.

Sintassi

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

Note

var	Stringa alla quale sarà assegnato il risultato.
Var1	Stringa di origine.
st	Posizione di inizio dell'estrazione/sostituzione.
l	Numero di caratteri da prelevare o cambiare.

Le stringhe su cui operare devono essere dello stesso tipo (internal or external).

Vedere anche

LEFT , RIGHT

Esempio

```
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
```

MOD

Azione

Restituisce il modulo (resto) di una divisione.

Sintassi

ret = var1 **MOD** var2

Note

ret	Variabile alla quale sarà assegnato il modulo (resto).
var1	Variabile da dividere (dividendo).
var2	Variabile per la quale dividere (divisore).

Esempio

```
a = 10 MOD 3          'divide 10 per 3
PRINT a               'visualizza il resto (1)
```

NEXT

Azione
Termina una struttura FOR..NEXT.

Sintassi
NEXT [var]

Note

var	Variabile indice utilizzata come contatore in una struttura FOR NEXT e specificato congiuntamente a FOR. Var è opzionale e non indispensabile.

Ogni istruzione FOR deve essere successivamente terminata con un NEXT.

Vedere anche
FOR

Esempio
```
y = 10                              'imposta y = 10
FOR a = 1 TO 10                     'ripete 10 volte
       FOR x = y TO 1               'anche questo
              PRINT x ; a           'visualizza i valori
       NEXT                         'x successivo (conta indietro)
NEXT a                              'a successivo (conta avanti)
END
```

ON Interrupt

Azione

Esegue una subroutine all'arrivo di una specifica interrupt.

Sintassi

ON interrupt label [NOSAVE]

Note

interrupt	INT0, INT1, SERIAL, TIMER0 ,TIMER1 o TIMER2. In 'Microprocessori supportati' sono elencati le specifiche interrupt associate ai chip.
Label	Label alla quale saltare all'arrivo dell'interrupt.
NOSAVE	Specificando NOSAVE, nessun registro sarà salvato e ripristinato nella routine di interrupt. Usando questa opzione assicuratevi di compiere separatamente questa operazione sui registri utilizzati.

Una routine di interrupt deve essere terminata con l'istruzione RETURN.
Solo un'istruzione RETURN può essere presente in una routine di interrupt poiché il compilatore provvede a ripristinare i registri e genera un RETI ogni volta che incontra un'istruzione RETURN nella routine di interrupt.

TIMER1 non può essere utilizzato contemporaneamente alle routine di gestione della comunicazione seriale poiché queste ultime fanno uso del TIMER1 per la generazione del BAUDRATE.

Impiegando le interrupts INT0 or INT1 è possibile specificare quale condizione produrrà l'interrupt. E' possibile impiegare l'istruzione Set/reset sul registro TCON per definire il comportamento desiderato:

SET TCON.0 : trigger INT0 sul fronte di discesa.
RESET TCON.0 : trigger INT0 a livello basso (0).
SET TCON.2 : trigger INT1 sul fronte di discesa.
RESET TCON.2 : trigger INT1 a livello basso (0).

vedere Hardware per maggiori dettagli

Esempio

```
ENABLE INTERRUPTS
ENABLE INT0                 'abilita l'interrupt
ON INT0 Label2 nosave       'salta a label2 su INT0
DO                          'loop continuo
LOOP
END

Label2:
    PRINT " An hardware interrupt occurred!"     'visualizza il messaggio
RETURN
```

ON Value

Azione

Salta ad una o più labels specificate in funzione del valore di una variabile.

Sintassi

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

Note

var	Variabile numerica da analizzare. Può anche essere uno SFR come P1.
label1, label2	Labels da raggiungere in relazione al valore *var*.

La definizione dei valori parte dal numero 0, quindi quando var=0 verrà richiamata la prima label specificata.

Esempio

```
x = 2                                   'assegna la variable per l'interrupt
ON x GOSUB lbl1, lbl2,lbl3              'salta alla label lbl3
x=0
ON x GOTO lbl1, lbl2 , lbl3
END


lbl3:
  PRINT " lbl3"
RETURN


Lbl1:

Lbl2:
```

OPEN - CLOSE

Azione
Apre e chiude un dispositivo.

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

Note

device	Sono supportati due devices hardware : COM1 e COM2. Con la UART software è necessario specificare il pin di porta ed il baudrate. COM3.0:9600 userà PORT 3.0 a 9600 baud.
MODE	Per COM1 e COM2 può essere specificato BINARY , INPUT o OUTPUT, mentre per le UART software deve essere definita la direzione del pin INPUT o OUTPUT.
channel	Numero del canale da aprire. Deve essere una costante positiva.

Poiché esistono microprocessori, come il 80537, che dispongono di 2 canali seriali a bordo, il compilatore deve essere informato sulla porta che s'intende usare. Questa è la ragione per la quale esiste l'istruzione OPEN. Se esiste una sola porta seriale a bordo del microprocessore non è necessario specificare quale s'intende usare e l'istruzione OPEN può essere omessa.
Le istruzioni collegate ai devices seriali sono PRINT , PRINTHEX, INPUT e INPUTHEX.

Ciascun device aperto deve successivamente essere chiuso per mezzo dell'istruzione CLOSE #channel. Naturalmente specificando lo stesso numero per channel.

La UART software supporta solo le istruzioni PUT e GET per inviare e ricevere dati.
COM1: e COM2: sono porte hardware e possono essere impiegate con PRINT ecc.

Vedere anche

Esempio 1
```
'only works with a 80517 or 80537
CONFIG BAUD1 = 9600                 'baudrate seriale 1
OPEN "COM2:" FOR BINARY AS #1       'apre la porta
PRINT #1, "Hello"                   'print sulla seriale 1
PRINT "Hello"                       'print sulla seriale 0
CLOSE #1                            'chiude il canale
```

Esempio 2
```
'funziona con qualsiasi pin di porta
Dim A As Byte , S As String * 16 , I As Byte , Dum As Byte

'una porta di comunicazione software prende il nome dal pin utilizzato
'ad esempio usando P3.0 si avrà "COM3.0:"  (viene omesso P)
'per le porte di comunicazione software deve essere specificato il baudrate
'Quindi per 9600 baud, il nome del device è "COM3.0:9600"
```

```
'Quando si usa il pin per trasmettere il device deve essere aperto come OUTPUT
'Quando si usa il pin per ricevere il device deve essere aperto come INPUT

'Per ora è possibile solo inviare e ricevere variabili per mezzo di PUT e GET.
'In futuro verrò completata la funzionalità anche con PRINT ecc.

Open "com3.1:9600" For Output As #1       'p3.1 è usata solitamente per TX quindi
                                          'la verifica è facile
Open "com3.0:9600" For Input As #2        'p3.0 è usata normalmente per RX quindi
                                          'la verifica è facile


S = "test this"                           'assegna una stringa
Dum = Len(s)                              'ne rileva la lunghezza
For I = 1 To Dum                          'per tutti i caratteri da SX a DX
  A = Mid(s , I , 1)                      'prende il carattere
  Put #1 , A                              'e lo invia sulla porta di comunicazione
Next

Do
  Get #2 , A                              'prende il carattere dalla porta
  Put #1 , A                              'e lo riscrive
  Print A                                 'anche sul canale normale
Loop


Close #1                                   'infine chiude il device
Close #2
End
```

OUT

Azione
Invia un byte ad una porta hardware oppure ad un indirizzo di memoria esterna.

Sintassi
OUT address, value

Note

address	Indirizzo al quale inviare il byte.
value	Variabile o valore da inviare.

L’istruzione OUT lavora solo con microprocessori che possono indirizzare memoria esterna.

Vedere anche
INP

Esempio
```
Dim a as byte
OUT &H8000,1          'invia  1 all’indirizzo hex 8000 del busdati (d0÷d7)
END

Genera il codice seguente :
Mov A,#1
Mov dptr,#h'8000
Movx @dptr,a
```

P1,P3

Azione

P1 e P3 sono registri speciali di funzione (SFR) che possono essere gestiti come variabili.

Sintassi

Px = var
var = **Px**

Note

x	Numero della porta (1 or 3). **P3.6 non può essere impiegata in un AT89C2051!**
Var	Variabile da acquisire o settare.

Altri microprocessori possono disporre di più porte, come P0, P2, P4 ecc.
Scegliendo opportunamente il file .**DAT** che definisce quale microprocessore si sta impiegando sarà possibile usare anche queste porte aggiuntive come variabili.
In pratica sarà possibile utilizzare qualsiasi SFR come una variabile.

ACC = 0 'ad esempio resetta l'accumulatore

Vedere hardware per maggiori dettagli.

Esempio

```
Dim a as BYTE, b1 as BIT
a = P1                'prende un valore dalla porta 1
a = a OR 2            'lo manipola
P1 = a                'e lo scrive sulla porta 1
P1 = &B10010101       'usa la notazione binaria
P1 = &HAF             'usa la notazione hex
b1 = P1.1             'legge il pin 1.1
P1.1 = 0              'e lo imposta a 0
```

PEEK()

Azione
Restituisce un byte memorizzato nella memoria interna.

Sintassi
var **= PEEK(** address **)**

Note

var	Variabile numerica alla quale assegnare il contenuto dalla cella puntata da address
address	Variabile numerica o costante che definisce l'indirizzo. (0÷255)

Vedere anche
POKE , CPEEK , INP , OUT

Esempio
```
DIM a As Byte
a = Peek( 0 ) 'restituisce il primo byte della memoria interna (r0)
End
```

POKE

Azione
Scrive un byte in una locazione della memoria interna.

Sintassi
POKE address , value

Note

address	Variabile numerica contenente l'indirizzo della cella di memoria da scrivere. (0÷255)
value	Valore da assegnare. (0÷255)

Si deve porre la massima attenzione nell'uso dell'istruzione POKE, poiché la memoria interna viene utilizzata per memorizzare le variabili e un'operazione maldestra può portare al funzionamento difettoso del programma.

Vedere anche
PEEK , CPEEK , INP , OUT

Esempio
```
POKE 127, 1          'scrive 1 all'indirizzo 127
End
```

POWERDOWN

Azione
Forza il processore in modalità powerdown.

Sintassi
POWERDOWN

Note
La modalità powerdown arresta completamente il clock di sistema.
L’unico modo per riattivare il processore è un reset.

Vedere anche
IDLE

Esempio
```
POWERDOWN
```

PRINT

Azione
Invia sulla porta seriale asincrona (RS-232).

Sintassi
PRINT var ; " constant"

Note

var	Variabile o costante da trasmettere (print).

Per inviare più variabili con la stessa istruzione, utilizzare punto e virgola come separatore **;**
Terminando una linea con punto e virgola non sarà inviato il carattere di avanzamento linea (linefeed).

L'istruzione PRINT può essere utilizzata quando esiste un'interfaccia seriale RS-232.
Consultare il manuale per informazioni sulla realizzazione di un'interfaccia RS-232.
L'interfaccia RS-232 può essere collegata alla porta di comunicazione seriale di un PC.
In questo modo è possibile impiegare un emulatore di terminale come dispositivo di output.
In BASCOM è integrato un emulatore di terminale.

Vedere anche
PRINTHEX , INPUT , OPEN , CLOSE

Esempio
```
'-----------------------------------------------------------
'               (c) 1997,1998 MCS Electronics
'-----------------------------------------------------------
'  file: PRINT.BAS
'  demo: PRINT, PRINTHEX
'-----------------------------------------------------------
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 da visualizzare


B1 = 10
Printhex B1                             'visualizza in notazione hex
C = &HA000                              'assegna il valore a c%
Printhex C                              'visualizza in notazione hex
Print C                                 'visualizza in notazione decimale

C = -32000
Print C
Printhex C
Rem Gli Integers possono assumere valori tra -32767 e 32768
End
```

PRINTBIN

Azione
Invia il contenuto binario di una variabile sulla porta seriale.

Sintassi
PRINTBIN var [,varn]

Note

var	Variabile contenente il valore da inviare alla porta seriale
varn	Ulteriori variabili da inviare. (opzionali)

PRINTBIN equivale a PRINT CHR(var); ma in questo modo si possono stampare interi array.

Usando un Long, ad esempio, saranno inviati 4 bytes.

Vedere anche
INPUTBIN

Esempio
```
Dim a(10) as Byte, c as Byte
For c = 1 To 10
  a(c) = a           'riempie l'array
Next
PRINTBIN a(1)        'visualizza il contenuto
```

PRINTHEX

Azione
Invia una variabile alla porta seriale in formato esadecimale.

Sintassi
PRINTHEX var

Note

var	Variabile contenente il valore da inviare alla porta seriale.

Si applicano le stesse regole valide per PRINT.

L'istruzione PRINTHEX può essere utilizzata quando esiste un'interfaccia seriale RS-232.
Consultare il manuale per informazioni sulla realizzazione di un'interfaccia RS-232.
L'interfaccia RS-232 può essere collegata alla porta di comunicazione seriale di un PC.
In questo modo è possibile impiegare un emulatore di terminale come dispositivo di output.
In BASCOM è integrato un emulatore di terminale.

Vedere anche
PRINT , INPUTHEX

Esempio
```
Dim x  As Byte
INPUT x                    'aspetta l'input di una variabile
PRINT x                    'la visualizza in formato decimale
PRINTHEX "Hex " ; x        'e poi in formato esadecimale
```

PRIORITY

Azione
Definisce i livelli di priorità delle interrupts.

Sintassi
PRIORITY SET / RESET interrupt

Note

SET	Porta la priorità dell'interupt al livello più alto.
RESET	Porta la priorità dell'interrupt al livello più basso.
Interrupt	L'interrupt per la quale definire la priorità.

Le interrupts sono: **INT0, INT1, SERIAL, TIMER0, TIMER1 e TIMER2.**

Altri microprocessori possono avere interrupt addizionali o diversi.
Vedere 'Microprocessori supportati' per maggiori dettagli.

Interrupt INT0 ha sempre la priorità più alta.
Quando più interrupts avvengano contemporaneamente l'ordine seguente sarà utilizzato nella gestione delle chiamate.

Interrupt	**Priorità**
INT0	1 (più alta)
TIMER0	2
INT1	3
TIMER1	4
SERIAL	5 (più bassa)

Esempio

```
PRIORITY SET SERIAL                 'int seriale al livello più alto
ENABLE SERIAL                       'abilita int seriale
ENABLE TIMER0                       'abilita int timer0
ENABLE INTERRUPTS                   'attiva la gestione delle interrupts
ON SERIAL label                     'salta alla label in caso di int seriale
DO                                  'loop continuo

LOOP

Label:                              'label
   PRINT " Serial int occurred."    'visualizza un messaggio
RETURN                              'return dall'interrupt
```

READ

Azione
Legge i valori definiti in DATA e li assegna a delle variabili.

Sintassi
READ var

Note

var	Variabile alla quale sarà assegnato il valore letto.

Differenze da QB
E' indispensabile che le variabili siano dello stesso tipo dei valori memorizzati con DATA.

Vedere anche
DATA , RESTORE

Esempio
```
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 " hello"
```

REM

Azione
Inserisce un commento.

Sintassi
REM o **'**

Note
Un programma commentato risulta più comprensibile.
Per inserire un commento usare REM oppure **'** seguito dal testo del commento.
Tutte le istruzioni poste dopo REM o **'** sono considerate commenti e non saranno compilate.

E' ora possibile inserire dei blocchi di testo come commento:
'(*inizio di un blocco di commento*
print "Questo non sarà compilato"
') *fine del blocco di commento*

Il simbolo iniziale ‘ garantisce la compatibilità con QB.

Esempio
```
REM  TEST.BAS version 1.00
PRINT a'    " this is comment   :  PRINT " hello"
                        ^--- questo non sarà eseguito!
```

RESET

Azione

Resetta (azzera) un bit di porta (P1.x, P3.x) oppure di una variabile interna di tipo Bit/Byte/Integer/Word.

Sintassi

RESET bit
RESET var.x

Note

bit	Può essere P1.x, P3.x o qualsiasi bit di variabile dove x=0÷7.
var	Può essere una variabile di tipo Byte, Integer o Word.
x	Costante indicante il bit di variabile da azzerare. (0÷7) per bytes e (0÷15) per Integer/Word

Vedere anche

SET

Esempio

```
Dim b1 as bit, b2 as byte, I as Integer
RESET P1.3                  'reset del bit 3 della porta 1
RESET b1                    'variable di tipo bit
RESET b2.0                  'reset bit 0 della variabile tipo byte b2
RESET I.15                  'reset del bit MS di I
```

RESTORE

Azione
Permette di rileggere dati già letti con READ. Azzera il puntatore di DATA.

Sintassi
RESTORE label

Note

label	Label cui è posizionata l'istruzione DATA.

Vedere anche
DATA , READ

Esempio
```
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%
Gli Integers devono terminare con il segno % (Integer : <0 or >255)
```

RETURN

Azione
Ritorna da una subroutine.

Sintassi
RETURN

Note
Ogni Subroutine deve essere terminata con la propria istruzione RETURN.
Anche le subroutines di Interrupt devono essere terminate con il proprio RETURN.

Vedere anche
GOSUB

Esempio
```
GOSUB  Pr                          'salta alla subroutine
PRINT result                       'visualizza result
END                                'fine del programma

Pr:                                'label di inizio della subroutine
       result = 5 * y              'del codice tanto
       result = result + 100       'per fare qualcosa
RETURN                             'ritorna
```

RIGHT()

Azione
Restituisce il numero di caratteri specificati di una stringa, partendo da destra.

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

Note

var	Stringa alla quale sarà assegnato il risultato.
Var1	Stringa di origine sulla quale operare.
st	Posizione dalla quale partire con l'estrazione.

Tutte le stringhe devono essere dello stesso tipo : internal or external.

Vedere anche
LEFT , MID

Esempio
```
Dim s As XRAM String * 15, z As XRAM String * 15
s = "ABCDEFG"
z = Right(s,2)
Print z                    'FG
End
```

ROTATE

Azione
Scorre tutti i bit a destra o a sinistra di una posizione.

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

Note

var	Variabile di tipo Byte, Integer/Word o Long.
shifts	Numero di scorrimenti da effettuare.

Il flag di carry viene interessato dall'operazione di scorrimento, occupando il posto del bit più significativo o meno significativo secondo la direzione dello scorrimento. Il risultato è identico a quello prodotto da un'istruzione assembler RLC oppure RRC.
Se questo comportamento è indesiderato, azzerare il bit di carry prima dello scorrimento per mezzo dell'istruzione CLR C (Clear Carry).

Vedere anche
SHIFTIN , SHIFTOUT

Esempio
```
Dim a as Byte
a = 128
ROTATE a, LEFT , 2
Print a       '1

Codice generato:

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

SELECT

Azione

Esegue un'istruzione da un elenco di istruzioni in relazione al valore di un valore o di un'espressione.

Sintassi

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

Note

var	Variabile da analizzare.
Test1	Valore da analizzare.
Test2	Valore da analizzare.

Vedere anche

-

Esempio

```
Dim b2 as byte
SELECT CASE b2              'set bit 1 della porta 1
  CASE 2 : PRINT "2"
  CASE 4 : PRINT "4"
  CASE IS >5 : PRINT ">5"   'un test richiede la parola chiave IS
  CASE ELSE
END SELECT
END
```

SET

Azione

Setta (pone a 1) un bit di porta (P1.x,P3.x) oppure di una variabile interna di tipo Bit/Byte/Integer/Word.

Sintassi

SET bit
SET var.x

Note

bit	Può essere P1.x, P3.x o qualsiasi bit di variabile dove x=0÷7.
var	Può essere una variabile di tipo Byte, Integer o Word.
x	Bit di una variabile (0÷7) da settare. (0÷15 per Integer/Word)

Vedere anche

RESET

Esempio

```
Dim b1 as Bit, b2 as byte, c as Word
SET P1.1      'set bit 1 della porta 1
SET b1        'variabile di tipo bit
SET b2.1      'set del bit 1 della var b2
SET C.15      'set del più alto bit della Word
```

SHIFTCURSOR

Azione
Scorre il cursore del display LCD a sinistra o a destra di una posizione.

Sintassi
SHIFTCURSOR LEFT / RIGHT

Vedere anche
SHIFTLCD

Esempio
```
LCD "Hello"
SHIFTCURSOR LEFT
End
```

SHIFTIN and SHIFTOUT

Azione

Invia o riceve una variabile attraverso un pin di porta, in forma di un treno di bit seriali con invio sincrono ad un segnale di clock.

Sintassi

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

Note

pin	Pin di porta da utilizzare come input/output.
pclock	Pin di porta su cui generare il clock.
var	Variabile assegnata all'operazione di shift.
Option	Opzioni possibili : 0 - MSB fatto scorrere in/out per primo quando clock va basso 1 - MSB fatto scorrere in/out per primo quando clock va alto 2 - LSB fatto scorrere in/out per primo quando clock va basso 3 - LSB fatto scorrere in/out per primo quando cock va alto Aggiungendo l'opzione 4 a SHIFTIN s'intende che il clock per lo scorrimento sia generato esternamente.

In relazione al tipo di variabile saranno necessari più o meno shift per completare l'invio. Per un byte necessiteranno 8 shifts mentre 16 shifts serviranno per un Integer.

Vedere anche

Esempio

```
Dim a as byte
SHIFTIN P1.0 , P1.1 , a , 0
SHIFTOUT P1.2 , P1.1 , a , 0

Per l'esempio di SHIFTIN questo è il codice generato:
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

Il codice generato dipende, naturalmente, dai parametri.
```

Per scorrere con un segnale esterno di clock:
SHIFTIN P1.0, P1.1 , a , 4 'add 4 for external clock

Codice generato:

```
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

Azione
Scorre il display LCD a sinistra o a destra di una posizione.

Sintassi
SHIFTLCD LEFT / RIGHT

Note
-

Vedere anche
SHIFTCURSOR

Esempio
```
LCD "Very long text"
SHIFTLCD LEFT
Wait 1
SHIFTLCD RIGHT
End
```

SOUND

Azione
Invia impulsi ad un pin di porta.

Sintassi
SOUND pin, duration, frequency

Note

pin	Qualsiasi pin di porta, come P1.0 ecc.
duration	Numero di impulsi da inviare. Byte, Integer/Word o costante compreso tra 1÷ 32768.
Frequency	Tempo per il quale il pin è forzato alto e basso.

Collegando un altoparlante oppure un cicalino ad un pin di porta (vedere hardware), è possibile utilizzare l'istruzione SOUND per produrre dei toni.

La porta è commutata alta e bassa per il numero di microsecondi specificato in *frequency*. Questo loop viene eseguito il numero di volte specificato in *duration*.

Vedere anche
-

Esempio
```
SOUND P1.1 , 10000, 10              'BEEP
End
```

SPACE()

Azione
Restituisce una stringa composta da più caratteri di spaziatura.

Sintassi
var = **SPACE(**x **)**

Note

x	Numero di caratteri di spaziatura.
Var	Stringa alla quale è assegnato il risultato.

Specificando zero per x, si otterrà una stringa di 255 bytes poiché non viene effettuato alcun controllo su assegnazioni di stringhe con lunghezza zero.

Esempio
```
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)

Codice generato per le ultime 2 linee :
;---------- library routine -----------
_sStr_String:
Mov @r1,a
Inc r1
Djnz r2,_sStr_String
Clr a
Mov @r1,a
Ret
;---------------------------------------
Mov R1,#h'22 ; location of string
Mov R2,h'21  ; number of spaces
Mov a,#32
Acall _sStr_String
```

SPIIN

Azione
Legge un valore dal bus SPI.

Sintassi
SPIIN var, bytes

Note

var	Variabile alla quale assegnare il valore letto dal bus SPI.
bytes	Numero di bytes da leggere.

Vedere anche
SPIOUT , CONFIG SPI

Esempio
```
Dim a(10) as byte
CONFIG SPI = SOFT, DIN = P1.0, DOUT = P1.1, CS=P1.2, CLK = P1.3
SPIIN a(1) , 4        'legge 4 bytes
```

SPIOUT

Azione
Invia il valore di una variabile sul bus SPI.

Sintassi
SPIOUT var , bytes

Note

var	Variabile che contiene il valore da inviare sul bus SPI.
bytes	Numero di bytes da inviare.

Vedere anche
SPIIN , CONFIG SPI

Esempio
```
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       'invia 5 bytes
SPIOUT X , 1          'invia 1 byte
```

START

Azione
Avvia il timer/counter specificato.

Sintassi
START timer

Note

timer	TIMER0, TIMER1, TIMER2, COUNTER0 o COUNTER1.

Affinché venga prodotta un'interrupt è necessario prima avviare un timer/counter (quando il gate esterno è disabilitato).

TIMER0 e COUNTER0 sono lo stesso dispositivo.

Vedere anche
STOP TIMERx

Esempio
```
ON TIMER0 label2
LOAD TIMER0, 100
START TIMER0
DO                    'start del loop
LOOP                  'loop continuo

label2:               'qui esegue una operazione

RETURN
```

STOP

Azione
Arresta l’esecuzione di un programma.

Sintassi
STOP

Note
Per terminare un programma può essere impiegata anche l’istruzione **END**.

Quando si raggiunge un’istruzione END o STOP viene prodotto un loop senza fine.

Esempio
```
PRINT var             'visualizza qualcosa
STOP                  'è tutto
```

STOP TIMERx

Azione
Arresta il timer/counter specificato.

Sintassi
STOP timer

Note

timer	TIMER0, TIMER1, TIMER2, COUNTER0 or COUNTER1.

Per sospendere l'esecuzione di un'interrupt è possibile anche arrestare il timer.

TIMER0 e COUNTER0 sono lo stesso dispositivo.

Vedere anche
START TIMERx

Esempio
```
'-----------------------------------------------------------
'                 (c) 1997,1998 MCS Electronics
'-----------------------------------------------------------
'  file: TIMER0.BAS
'  demo: ON TIMER0
'  *TIMER1 è usato come generatore di baudrate per RS-232
'-----------------------------------------------------------
Dim Count As Byte , Gt As Byte

Config Timer0 = Timer , Gate = Internal , Mode = 2
'Timer0 = counter : timer0 operates as a counter
'Gate = Internal  : no external gate control
'Mode = 2         : 8-bit auto reload (default)

On Timer0 Timer_0_int
Load Timer0 , 100                       'quando il timer raggiunge 100 avviene
                                        'un'interrupt
Enable Interrupts                       'abilita l'uso delle interrupts
Enable Timer0                           'abilita il timer


Rem Setting Of Priority
Priority Set Timer0                     'alla priorità più alta
Start Timer0                            'avvia il timer

Count = 0                               'reset del counter
Do
  Input "Number " , Gt
  Print "You entered : " ; Gt
Loop Until Gt = 1                       'loop finchè viene premuto ESC
Stop Timer0
End


Rem The Interrupt Handler For The Timer0 Interrupt
Timer_0_int:
   Inc Count
```

```
   If Count = 2 Then
      Print "Timer0 Interrupt avvenuto"
      Count = 0
   End If
Return
```

STR()

Azione
Restituisce un numero sotto forma di stringa.

Sintassi
var **= Str(** x **)**

Note

var	Variabile di tipo String.
X	Variabile numerica.

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

La stringa deve essere sufficientemente grande per contenere il risultato.

Vedere anche
VAL

Differenze da QB
In QB STR() restituisce una stringa con uno spazio in testa (leading).
In BASCOM questo non avviene.

Esempio
```
Dim a as Byte, S as XRAM String * 10
a = 123
s  = Str(a)
Print s
End
```

STRING()

Azione
Restituisce una stringa contenente il carattere ASCII n ripetuto m volte.

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

Note

var	Striga alla quale è assegnato il risultato.
n	Codice ASCII da assegnare alla stringa.
m	Numero di caratteri identici da assegnare.

Non è possibile utilizzare il codice ASCII zero, poiché viene utilizzato dal compilatore per segnalare la fine di una stringa.
Specificando zero per m, si otterrà una stringa di 255 bytes poiché non viene effettuato alcun controllo su assegnazioni di stringhe con lunghezza zero. Qualora fosse necessaria questa funzionalità informate MCS Electronics.

Vedere anche
SPACE

Esempio
```
Dim s as XRAM String * 15
s = String(5,65)
Print s                     'AAAAA
End
```

SUB

Azione
Definisce una Subroutine.

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

Note

name	Nome della subroutine. Può essere impiegata qualsiasi parola non riservata.
var1	Nome del parametro.

Ciascuna subroutine deve essere terminata con l'istruzione END SUB.

La dichiarazione di una Subroutine deve essere effettuata prima di scrivere la procedura relativa alla subroutine.
Il tipo ed il nome dei parametri devono essere identici sia nella dichiarazione sia nella procedura della subroutine.

I parametri sono GLOBALI per tutta l'applicazione e devono essere dimensionati con l'istruzione DIM come qualsiasi altra variabile.
Questo consente di condividere i parametri di una subroutine con il programma principale, come illustrato nell'esempio seguente :

```
Dim a as byte, b1 as byte, c as byte      'dimensiona le variabili
Declare Sub Test(a as byte)               'dichiara la subroutine
a = 1 : b1 = 2: c = 3                     'assegna le variabili

Print a ; b1 ; c                          'visualizza le variabili

Call Test(b1)                             'chiama la subroutine
Print a ;b1 ; c                           'visualizza le variabili di nuovo
End

Sub Test(a as byte)                       'inizio della procedura/subroutine
  print a ; b1 ; c                        'visualizza le variabili
End Sub
```

Vedere anche
CALL, DECLARE

Esempio
-

SWAP

Azione

Scambia tra loro due variabili dello stesso tipo.

Sintassi

SWAP var1, var2

Note

var1	Variabile di tipo Bit, Byte, Integer o Word.
var2	Variabile dello stesso tipo di var1.

Dopo lo scambio var1 conterrà il valore di var2 mentre var2 quello di var1.

Esempio

```
Dim a as integer,b1 as integer
a = 1 : b1 = 2              'assegna due integers
SWAP a, b1                  'li scambia
PRINT a ; b1
```

THIRDLINE

Azione
Posiziona il cursore del display LCD a capo della terza linea.

Sintassi
THIRDLINE

Note
-

Vedere anche
UPPERLINE , LOWERLINE , FOURTHLINE

Esempio
```
Dim a as byte
a = 255
LCD a
Thirdline
LCD a
Upperline
End
```

UPPERLINE

Azione
Posiziona il cursore del display LCD a capo della linea più alta.

Sintassi
UPPERLINE

Note
-

Vedere anche
LOWERLINE THIRDLINE FOURTHLINE

Esempio
```
Dim a as byte
a = 255
LCD a
Lowerline
LCD a
Upperline
End
```

VAL()

Azione
Converte una stringa rappresentante un numero in numero.

Sintassi
var **= Val(** s **)**

Note

var	Variabile numerica alla quale assegnare il valore di s.
s	Variabile di tipo stringa.

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

Vedere anche
STR

Esempio
```
Dim a as byte, s As XRAM string * 10
s = "123"
a = Val(s)           'converte la stringa
Print a
End
```

VARPTR()

Azione
Restituisce l’indirizzo di memoria al quale è memorizzata una variabile.

Sintassi
var = VARPTR(var2)

Note

var	Variabile alla quale assegnare l’indirizzo di var2.
var2	Variabile della quale si richiede l’indirizzo.

Vedere anche
PEEK POKE

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

Codice generato :
Mov h'23,#h'21
```

WAIT

Azione
Sospende l’esecuzione del programma per un tempo predefinito.

Sintassi
WAIT seconds

Note

seconds	Numero di secondi di attesa.

Il ritardo è basato sulla frequenza di clock di 12 Mhz.
Questo comando non consente temporizzazioni accurate.
Nel caso si verificassero delle interrupts nel corso del ritardo, questo sarà prolungato del tempo necessario all’esecuzione dell’interrupt.

Vedere anche
DELAY

Esempio
```
WAIT 3        'attende per tre secondi
Print "*"
```

WAITKEY

Azione
Attende finché sia ricevuto un carattere nel buffer seriale.

Sintassi
var = **WAITKEY**

Note

var	Variabile alla quale assegnare il valore ASCII del carattere ricevuto nel buffer seriale.

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

Vedere anche
INKEY

Esempio
```
Dim A As Byte
A = Waitkey           'attende un carattere
Print A
```

WAITMS

Azione
Sospende l’esecuzione del programma per un tempo predefinito in millisecondi.

Sintassi
WAITMS mS

Note

mS	Numero di millisecondi di attesa. (1÷255)

Il ritardo è basato sulla frequenza di clock di 12 Mhz.
Questo comando non consente temporizzazioni accurate.
Nel caso si verificassero delle interrupts nel corso del ritardo, questo sarà prolungato del tempo necessario all’esecuzione dell’interrupt.
Questa istruzione è prevista per l’uso con le istruzioni del bus I2C. Quando si effettua una scrittura su una EEPROM collegata ad un bus I2C è necessario attendere 10 mS prima della successiva istruzione di scrittura.

Vedere anche
DELAY WAIT

Esempio
```
WAITMS 10            'attende 10 millisecondi
Print "*"
```

WHILE .. WEND

Azione
Esegue una serie di istruzioni, in loop, fintanto che la condizione specificata rimane vera.

Sintassi
WHILE condition
        istruzioni
WEND

Note
Le istruzioni contenute tra WHILE e WEND saranno eseguite fino a quando la condizione specificata in WHILE è verificata.
Se la condizione non è verificata il programma riprende all'istruzione che segue WEND.

Vedere anche
DO .. LOOP

Esempio
```
WHILE a <= 10
       PRINT a
      INC a
WEND
```

Hardware - display LCD

Il display LCD può essere collegato come segue:

DISPLAY LCD	PORTA	PIN
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

Questa connessione lascia disponibili P1.1 e P1.0 e P3 per altri usi.

E' possibile definire collegamenti diversi specificando i pin di porta nel menu Options LCD.
Il tipo di display LCD può essere selezionato con l'istruzione CONFIG LCD.

Il display LCD lavora in modalità 4 bit.
Vedere l'istruzione $LCD per il funzionamento in modalità 8 bit.

BASCOM dispone di molte istruzioni per la gestione del display LCD.
Per quanti volessero un maggiore controllo l'esempio seguente può essere d'aiuto.

```
Acc = 5                 'carica un valore nel registro A
Call Lcd_control        'è il valore di controllo del display
Acc = 65                'carica un nuovo valore (la lettera A)
Call Write_lcd          'lo scrive sul display LCD
```

Lcd_control e Write_lcd sono subroutines assembler che possono essere richiamate da BASCOM.

Verificare le specifiche del display LCD utilizzato per un corretto impiego.

Microprocessori supportati

Alcuni microprocessori dispongono di caratteristiche aggiuntive rispetto agli AT89C2051/8051.

8032/8052/AT89S8252
TIMER2

AT89S8252
WATCHDOG
DATA EEPROM
Pin di porta con funzioni alternative

80515,80535,80517,80535
GETAD
WATCHDOG
BAUDRATE GENERATOR
INTERRUPTS e PRIORITY

80517,80537
GETAD
WATCHDOG
BAUDRATE GENERATOR
BAUDRATE GENERATOR1
INTERRUPTS e PRIORITY

AT89S8252 WATCHDOG

Il microprocessore AT89S8252 incorpora un timer di watchdog.
Un timer di watchdog è un timer che provvede a resettare il microprocessore al raggiungimento di un certo valore (tempo).
Quindi, durante l'esecuzione del programma è necessario resettare questo timer prima che completi il suo conteggio. Questa funzione viene utilizzata per garantire che il programma stia lavorando correttamente.
Quando un programma non funziona oppure si arena in un loop senza fine il timer di watchdog non viene azzerato e quindi si produce un reset automatico con conseguente riavvio del programma.

START WATCHDOG avvia il timer di watchdog.
STOP WATCHDOG arresta il timer di watchdog.
RESET WATCHDOG resetta (azzera) il timer di watchdog.

Vedere anche

CONFIG WATCHDOG

Esempio

```
'-----------------------------------------------------
'                (c) 1998 MCS Electronics
' WATCHD.BAS demonstrates the AT89S8252 watchdog timer
' select 89s8252.dat !!!
'-----------------------------------------------------
Config Watchdog = 2048                  'reset dopo 2048 mSec
Start Watchdog                          'start del timer di watchdog
Dim I As Word
For I = 1 To 10000
  Print I                               'visualizza il valore
  '  Reset Watchdog
  'il loop for next non sarà completato per l'intervento del timer di watchdog
  'togliendo il simbolo di commento all'istruzione RESET WATCHDOG verrà invece
  'completato perché il wd-timer sarà resettato prima che arrivi a 2048 msec
Next
End
```

WATCHDOG

Il microprocessore AT89S8252 incorpora un timer di watchdog.
Un timer di watchdog è un timer che provvede a resettare il microprocessore al raggiungimento di un certo valore (tempo).
Quindi, durante l'esecuzione del programma è necessario resettare questo timer prima che completi il suo conteggio. Questa funzione viene utilizzata per garantire che il programma stia lavorando correttamente.
Quando un programma non funziona oppure si arena in un loop senza fine il timer di watchdog non viene azzerato e quindi si produce un reset automatico con conseguente riavvio del programma.

CONFIG WATCHDOG = value

	Tempo trascorso il quale il timer di WD genera un reset.
value	Valori possibili, in millisecondi : 16,32,64,128,256,512,1024 o 2048

START WATCHDOG avvia il timer di watchdog.
STOP WATCHDOG arresta il timer di watchdog.
RESET WATCHDOG resetta (azzera) il timer di watchdog.

Esempio

```
DIM A AS INTEGER
CONFIG WATCHDOG = 2048              'produce un reset dopo circa 2 secondi
START WATCHDOG                      'avvia il WD
DO
  PRINT a
  a = a + 1                         'smette in caso di reset
  REM RESET WATCHDOG                'eliminando REM lavorerà correttamente
LOOP
END
```

DATA EEPROM

Il microprocessore AT89S8252 incorpora 2Kbytes di flash EEPROM.
Questa memoria può essere utilizzata per memorizzare dati.
Per questo sono previste due specifiche istruzioni : WRITEEEPROM e READEEPROM.

WRITEEEPROM var [, address]

var	Qualsiasi nome di Variabile BASCOM.
Address	Indirizzo di EEPROM dove scrivere i dati, tra 0÷2047. Se viene omesso l'indirizzo questo sarà assegnato da BASCOM automaticamente e sarà possibile individuarlo leggendo nel report file.

READEEPROM var [, address]

var	Qualsiasi nome di Variabile BASCOM.
Address	L'indirizzo di EEPROM dal quale leggere i dati, tra 0÷2047. L'indirizzo può essere omesso se i dati sono stati precedentemente memorizzati con l'istruzione **WRITEEEPROM** poiché in questo caso il compilatore è in grado di risalire all'indirizzo assegnato automaticamente.

Esempio

```
Dim S As String * 15 , S2 As String * 10
S = "Hello" : S2 = "test"

Dim L As Long
L = 12345678
Writeeeprom S
Writeeeprom S2                          'scrive le stringhe
Writeeeprom L                           'scrive il long

S = "" : S2 = "" : L = 0                'clear delle variabili
Readeeprom L : Print L
Readeeprom S : Print S
Readeeprom S2 : Print S2
End
```

Pin di porta con funzioni alternative

Le porte del microprocessore AT89S8252 hanno delle funzioni alternative e sono riassunte nella tabella seguente.

Port Pin	Funzione alternativa
P1.0	T2 ingresso di conteggio esterno per timer.counter 2, uscita del segnale di clock
P1.1	T2EX capture/reload trigger e flag di direzione di timer/counter 2
P1.4	/SS Ingresso di selezione Slave
P1.5	MOSI uscita dati master, ingresso dati slave per SPI
P1.6	MISO ingresso dati master, uscita dati slave per SPI
P1.7	SCK uscita clock master, ingresso clock slave per SPI
P3.0	RxD ingresso porta seriale asincrona
P3.1	TxD uscita porta seriale asincrona
P3.2	/INT0 ingresso interrupt 0
P3.3	/INT1 ingresso interrupt 1
P3.4	T0 ingresso esterno timer 0
P3.5	T1 ingresso esterno timer 1
P3.6	/WR strobe di scrittura per memoria dati esterna
P3.7	/RD strobe di lettura per memoria dati esterna

/ indica ingresso negato, attivo basso

TIMER2 in 8032 e compatibili

Alcuni microprocessori incorporano timer addizionali : TIMER2.
Questa sezione descrive il TIMER2 per microprocessori 8032 compatibili e non si applica al TIMER2 presente nei microprocessori 80C535 ed altri.
TIMER2 è un timer/counter a 16-bit which che può funzionare sia come timer sia come counter. TIMER2 dispone di 3 modalità di funzionamento : capture, auto-reload (conteggio avanti o indietro), e generatore di baud rate.

Capture mode

In modalità capture esistono due opzioni :

- timer/counter a 16-bit che al completamento del conteggio mette a 1 il bit TF2, che è il bit di overflow del TIMER2. Questo bit può essere utilizzato per generare una interrupt.

  Modalità Counter :
  CONFIG TIMER2 = COUNTER, GATE = INTERNAL, MODE = 1

  Modalità Timer :
  CONFIG TIMER2=TIMER, GATE= INTERNAL,MODE =1

- Come sopra, ma con la funzione aggiuntiva che una transizione da 1 a 0 sull'ingresso T2EX produce il trasferimento del contenuto dei registri TL2 e TH2 di TIMER2 dentro i registri di capture RCAP2L e RCAP2H.

  Modalità Counter :
  CONFIG TIMER2 = COUNTER, GATE = EXTERNAL, MODE = 1

  Modalità Timer :
  CONFIG TIMER2=TIMER,GATE=EXTERNAL,MODE=1

Inoltre una transizione di T2EX mette a 1 il bit EXF2 in T2CON. EXF2, come TF2, può generare un'interrupt.

La routine di interrupt di TIMER2 può verificare i bit TF2 e EXF2 per determinare quale evento ha causato l'interrupt.
(in questa modalità non esiste valore di reload. Anche quando un evento di capture avviene su T2EX il contatore prosegue il conteggio delle transizioni su T2EX oppure gli impulsi di clock del timer ottenuti con osc/12)

Modalità Auto reload

In modalità auto reload a 16-bit, TIMER2 può essere configurato come timer o contatore programmabile avanti/indietro. La direzione del conteggio è determinata dal bit DCEN.
TIMER2 conterà fino a **&H**FFFF raggiunta la fine del conteggio metterà a 1 il bit TF2 che è il flag di overflow. Questo produce il reload dei registri di TIMER2 con il valore a 16-bit presente in RCAP2L e RCAP2H.
I valori in RCAP2L e RCAP2H verranno preimpostati da software.

Modalità Counter :
CONFIG TIMER2=COUNTER,GATE=INTERNAL,MODE=0

Modalità Timer :
CONFIG TIMER2=COUNTER,GATE=INTERNAL,MODE=0

Quando EXEN2=1 un overflow oppure una transizione da 1 a 0 sull'ingresso T2EX possono causare il reload di 16-bit. Questa transizione mette anche a 1 il bit EXF2. Se l'interrupt di TIMER2 è abilitata può essere generata quando TF2 oppure EXF2 sono allo stato logico 1.

Modalità Counter :
CONFIG TIMER2=COUNTER,GATE=EXTERNAL,MODE=0

Modalità Timer :
CONFIG TIMER2=TIMER,GATE=EXTERNAL,MODE=0

TIMER2 può inoltre contare avanti o indietro. In questa modalità il pin T2EX può essere utilizzato per controllare la direzione del conteggio. Quando a T2EX è applicato uno stato logico 1 TIMER2 conta in avanti (incremento).
L'overflow del TIMER2 avviene al raggiungimento di **&H**FFFF e mete a 1 il flag TF2, che può generare un'interrupt, se abilitata. L'overflow del timer produce anche il reload dei registri TL2 e TH2 con i valori contenuti in RCAP2L e RCAP2H (16-bit).

Counter mode:
CONFIG TIMER2=COUNTER,GATE=INTERNAL/EXTERNAL,MODE=0,DIRECTION=UP

Timer mode:
CONFIG TIMER2=COUNTER,GATE=INTERNAL/EXTERNAL,MODE=0,DIRECTION=UP

Uno stato logico 0 applicato al pin T2EX forza il TIMER2 a contare indietro (decremento). L'underflow di TIMER2 avverrà quando TL2 e TH2 assumeranno lo stesso valore presente in RCAP2L e RCAP2H. L'underflow di TIMER2 metterà allo stato logico 1 il flag TF2 e ricaricherà il valore **&H**FFFF nei registri TL2 e TH2 del timer.

Modalità Counter :
CONFIG
TIMER2=COUNTER,GATE=INTERNAL/EXTERNAL,MODE=0,DIRECTION=DOWN

Modalità Timer :
CONFIG
TIMER2=COUNTER,GATE=INTERNAL/EXTERNAL,MODE=0,DIRECTION=DOWN

Il flag esterno TF2 cambia di stato ad ogni overflow oppure underflow di TIMER2.
Il flag EXF2 non genera interrupt in modalità contatore avanti/indietro.

Generatore di Baud rate

Questa modalità può essere selezionata per generare il baud rate per la porta seriale asincrona, liberando TIMER1 per altri compiti.
CONFIG TIMER2=TIMER,GATE=INTERNAL,MODE=2

Solo Ricezione

Questa modalità può essere selezionata per generare il baud rate solo per il ricevitore. TIMER1 può essere usato per generare un diverso baud rate in trasmissione.
CONFIG TIMER2=TIMER,GATE=INTERNAL,MODE=3

TIMER1 dovrà essere predisposto per questo uso con istruzioni assembler.

Solo Trasmissione

Questa modalità può essere selezionata per generare il baud rate solo per il trasmettitore. TIMER1 può essere usato per generare un diverso baud rate in ricezione.
CONFIG TIMER2=TIMER,GATE=INTERNAL,MODE=4

TIMER1 dovrà essere predisposto per questo uso con istruzioni assembler.

Uscita di Clock

Alcune varianti di processori 8052 hanno la possibilità di generare su P1.0 un segnale di clock con duty cycle pari al 50%.
CONFIG TIMER2=TIMER,MODE=5

La frequenza prodotta è pari a (fOSC **/** 4) **/** (65536-CAPTURE)

CAPTURE = valore impostato nel registro di capture.

Come determinare cosa ha causato un'interrupt

Analizzando il bit T2CON.7 si può determinare se l'interrupt è stata causata da un overflow.
Analizzando il bit T2CON.6 si può determinare se l'interrupt è stata causata da una transizione negativa su T2EX.

```
Timer2_ISR:
If T2CON.7 = 1 Then
  Print “overflow di timer”
Else
  If T2CON.6 = 1 Then
     Print “transizione esterna”
  End if
End If
Return
```

80515 INTERRUPTS E PRIORITÀ

I microprocessori 80515 e 80535 dispongono di più interrupts ed il livello delle priorità è gestito diversamente rispetto al classico 8051.

Abilitazione interrupts:

ENABLE AD	'convertitore A/D
ENABLE INT2|INT3|INT4|INT5|INT6	‘interrupt esterna 2-6
ENABLE TIMER2EX	'reload esterno timer2

Disabilitazione interrupts:

DISABLE AD	'convertitore A/D
DISABLE INT2|INT3|INT4|INT5|INT6	'interrupt esterna 2-6
DISABLE TIMER2EX	'reload esterno timer2

Selezione delle priorità:
PRIORITY SET|RESET source , level
level può essere 0,1,2 o 3.(0=più bassa, 3=più alta)

Le sorgenti possono essere :
INT0/ADC
TIMER0/INT2
INT0/INT3
TIMER1/INT4
SERIAL/INT5
TIMER2/INT6

Attenzione, solo una di una coppia può essere selezionata.
PRIORITY SET INT4,3 'imposterà INT4 alla massima priorità.
Quando due interrupts avvengono contemporaneamente la prima ad essere eseguita sarà quella a priorità maggiore e in caso di identica priorità sarà la prima in ordine di elenco ad essere servita. Quindi ad una richiesta contemporanea di TIMER1 ed INT4, impostate per avere la stessa priorità, TIMER1 sarà servita per prima.
Per maggiori dettagli vedere la specifica documentazione tecnica.

GETAD

Azione

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.

Sintassi

var = GETAD(channel, range)

Note

var	Variabile alla quale assegnare il valore A/D letto
channel	Identificativo del canale da misurare
range	Selezione del campo di misura 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

L'istruzione GETAD() è disponibile solo con microprocessori 80515, 80535,80517 e 80535 poiché fa uso di specifiche caratteristiche del chip.

Vedere anche

Esempio

```
Dim b1 as Byte, Channel as byte,ref as byte
channel=0                    'ingresso P6.0
ref=0                        'range 0 ÷ 5 Volt
b1=getad(channel,ref)        'salva A/D in b1
```

80515 WATCHDOG

I microprocessori 80515 e 80535 dispongono di un timer di watchdog.
Questo è un timer a 16 bit che non può essere arrestato e produrrà un reset dopo 65535 uS con un clock a 12MHz !

```
START WATCHDOG          'avvia il timer di watchdog.
RESET WATCHDOG          'resetta (azzera) il timer di watchdog.
```

80537 INTERRUPTS E PRIORITÀ

I microprocessori 80517 e 80537 dispongono di più interrupts ed il livello delle priorità è gestito diversamente rispetto al classico 8051.

Abilitazione interrupts:

ENABLE AD	'convertitore A/D
ENABLE INT2|INT3|INT4|INT5|INT6	'interrupt esterno 2-6
ENABLE TIMER2EX	'reload esterno timer2
ENABLE CTF	'interrupt di timer compare
ENABLE SERIAL1	'interrupt seriale asincrona 1

Disabilitazione interrupts:

DISABLE AD	'convertitore A/D
DISABLE INT2|INT3|INT4|INT5|INT6	interrupt esterno 2-6
DISABLE TIMER2EX	'reload esterno timer2
DISABLE CTF	'interrupt di timer compare
DISABLE SERIAL1	'interrupt seriale asincrona 1

Selezione delle priorità :
PRIORITY SET|RESET source , level
level può essere 0,1,2 o 3.(0=più bassa, 3=più alta)

Le sorgenti possono essere :
INT0/ADC/SERIAL1
TIMER0/INT2
INT0/INT3
TIMER1/CTF/INT4
SERIAL/INT5
TIMER2/INT6

Attenzione, sol una di una tripletta può essere selezionata.
PRIORITY SET INT4,3 'imposterà INT4 alla massima priorità.
Quando due interrupts avvengono contemporaneamente la prima ad essere eseguita sarà quella a priorità maggiore e in caso di identica priorità sarà la prima in ordine di elenco ad essere servita. Quindi ad una richiesta contemporanea di TIMER1 ed INT4, impostate per avere la stessa priorità, TIMER1 sarà servita per prima.
Per maggiori dettagli vedere la specifica documentazione tecnica.

CONFIG BAUD1

Azione

Configura il microprocessore per utilizzare il generatore interno di baud rate sul canale seriale asincrono 1.
Questo generatore di baud rate è disponibile solo nei microprocessori 80517 e 80537.

Sintassi

CONFIG BAUD1 = baudrate

Note

baudrate	Baudrate da utilizzare : 2048 ÷ 37500

I microprocessori 80517 e 80537 dispongono di 2 porte seriali asincrone integrate.

Vedere anche

CONFIG BAUD

Esempio

```
CONFIG BAUD1 = 9600   'usa il generatore di baud rate interno
Print "Hello"
End
```