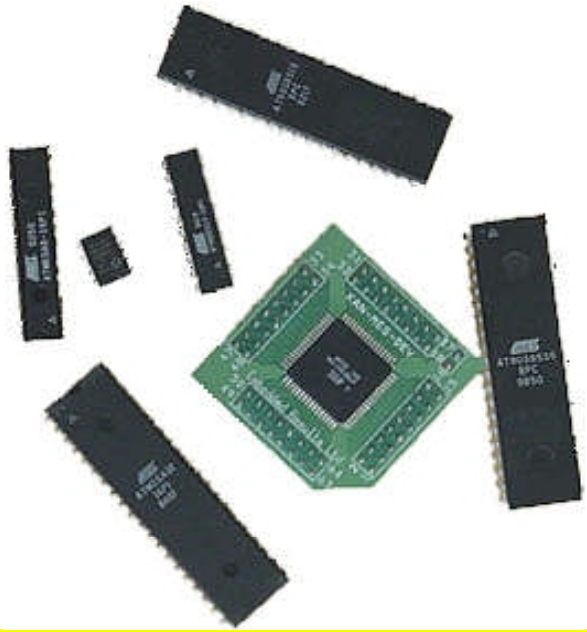
BASIC des microcontrôleurs

Jean-Pierre Duval

1ere partie : Le BASIC

2eme partie :
Le système de développement
BASIC BASCOM-AVR



3eme Partie : Le dictionnaire

4eme partie :
les microcontrôleurs
R.I.S.C. ATMEL

BASIC BASCOM DICTIONNAIRE

NOTES PRELIMINAIRES

Symboles utilisés dans ce dictionnaire:

Exemple : SHIFT Var, LEFT/RIGHT [,shifts]
SHIFT est l'instruction ou la fonction;
LEFT/RIGHT veut dire qu'il faut préciser un choix.
[,shifts] veut dire que le nombre "shifts" est optionnel.

Composant = composant contenu dans le µP (un timer, le watchdog…)

Quand plusieurs instructions sont similaires ou complémentaires, il n'y a qu'une seule explication (LTRIM, TRIM, RTRIM par exemple)

Les fichiers exemples (nnnn.bas) n'ont pas été recopiés, ils sont tous dans le répertoire Samples. De petits exemples aidant à la compréhension, ont néanmoins été ajoutés à certaines instructions.
Les instructions de compilations ne sont pas dans la partie dictionnaire, voir le 1° livre consacré au BASIC.

A la différence du dictionnaire BASCOM-AVR nous avons éclaté le notre en trois parties :
- ⇨ Les instructions de CONFIG,
- ⇨ Les instructions et fonctions proprement dites,
- ⇨ Les fonctions mathématiques.

Dans le Basic, des tableaux par ordre alphabétique reprennent les fonctions les plus utilisées, conversions de variables, manipulation de chaînes de caractères…

CONFIG

Les instructions CONFIG sont utilisées pour configurer les composants Hardware ou soft comme : 1WIRE, ADC, CLOCK, DEBOUNCE, GRAPHLCD, I2CDELAY, INTx, KBD, KEYBOARD, LCD, LCDBUS, LCDMODE, LCDPIN, RC5, PORT, SERIALIN, SERIALOUT, SERVOS, SDA, SCL, SPI, TIMER0, TIMER1,TIMER2, WATCHDOG, WAITSUART.

CONFIG 1WIRE

Action
Configure la broche à utiliser pour les instructions 1WIRE

Syntaxe
CONFIG 1WIRE= broche

Remarques
Broche la broche du port utilisé, exemple portD.0
Cette instruction supprime et remplace le réglage compilateur.
Normalement il ne peut y avoir qu'une seule broche 1WIRE puisque plusieurs composants peuvent être rattachés sur le BUS. En fait il est possible d'utiliser d'autres broches grâce aux instructions 1WRESET, 1WREAD, 1WWRITE.

Voir Aussi
1WRESET, 1WREAD, 1WWRITE, 1WIRECOUNT,1WSEARCHFIRST,1WSEARCHNEXT
1WVERIFY, Programme 1Wiresearch.bas, 1wire.bas

CONFIG ACI

Action
Configuration du Comparateur Analogique

Syntaxe
CONFIG ACI = ON|OFF, COMPARE = ON|OFF,TRIGGER=TOGGLE|RISING|FALLING

Remarques

ACI	Peut être ON ou OFF
COMPARE	Peut être ON ou OFF, le TIMER1 en mode CAPTURE place le Comparateur en état ON.
TRIGGER	Spécifie sur quel événements de comparaison l'interruption se produit

---

CONFIG ADC

Action
Configure le convertisseur Analogique/digital

Syntaxe
CONFIG ADC=SINGLE, PRESCALER=AUTO, REFERENCE=opt

Remarques
ADC Mode de travail peut-être SINGLE ou FREE
PRESCALER pre-diviseur, une constante numérique pour le diviseur d'horloge. Utiliser AUTO pour laisser le compilateur générer la meilleure valeur.
REFERENCE Quelque µP comme le M163 ont des options de références additionnelles. Leur valeur peut être OFF AVCC ou INTERNAL. Voir la datasheet de chaque µP.

Voir Aussi
Exemple ADC.bas et ADC-int.bas (avec un peu de commandes en assembleur !)

CONFIG CLOCK

Action
Configure le timer qui doit être utilisé pour TIME$ et DATE$

Syntaxe
CONFIG CLOCK=soft |USER [,gosub=sectic]

Remarques
SOFT|USER SOFT utilise les optionset routines incluses dans Bascom. USER permet d'écrire ses propres routines en combinaison avec une horloge I2C par exemple.
SECTIC Cette option permet de sauter à une routine utilisateur avec l'étiquette SECTIC. Il est important d'utiliser cette étiquette et de ne pas oublier le RETURN.

L'interruption arrive toutes les secondes, il est donc possible de réaliser beaucoup de tâches pendant cette période. Le programme doit être terminer par STOP (voir Stop)
Cette option utilise 30 Bytes de stack Hardware.

Le compilateur dimensionne automatiquement les variables suivantes:
_sec ; _min ; _hour ; _day ; _month ; _year et les variables TIME$ et DATE$

Le compilateur crée aussi un ISR[1] qui sera mis à jour toutes les secondes. Ceci ne fonctionne qu'avec les µP 8535, M163, M103, M603 ou autres µP qui peuvent fonctionner en mode asynchrone.

Voir Aussi
TIME$, DATE$ et le programme exemple MEGACLOCK.bas

---

[1] In Service Register

CONFIG COM1

Action
Configuration de l'UART des µP AVR qui possèdent une UART étendue comme le M8

Syntaxe
CONFIG COM1 = dummy , synchrone=SY, parity = Pa, stopbits=SB, Databits=DB, clockpol=CP

Remarques

SY	0 pour les opérations synchrones (défaut) et 1 pour les opérations asynchrones.
Pa	None (sans), Disabled(invalide), even(pair) ou Odd(impair)
SB	Le nombre de bit de stop : 1 ou 2
DB	Le nombre de bit de Data : 4,6,7,8,9.
CP	La polarité d'horloge, 0 ou 1

Voir Aussi
DATASHEET du M8

CONFIG COM2

Action
Configuration de la seconde UART des µP AVR qui possèdent des UART étendues comme le M128

Syntaxe
CONFIG COM2 = dummy , synchrone=SY, parity = Pa stopbits=SB, Databits=DB, clockpol=CP

Remarques

SY	0 pour les opérations synchrones (défaut) et 1 pour les opérations asynchrones.
Pa	None (sans), Disabled(invalide), even(pair) ou Odd(impair)
SB	Le nombre de bit de stop : 1 ou 2
DB	Le nombre de bit de Data : 4,6,7,8,9.
CP	La polarité d'horloge, 0 ou 1

Voir Aussi
DATASHEET du M128

CONFIG DATE

Action
Configure le format selon lequel la date sera affichée.

Syntaxe
CONFIG DATE = DMY , Séparateur = char

Remarques
DMY jour(Day), le mois(Month) and l'année(Year) . Usage DMY, MDY or YMD.
Char un caractère séparateur : / , - or . (point)
The next table shows the common formats of date and the associated statements.
Le tableau suivant montre les principaux formats internationaux.

Pays	Format	Expression
Amerique (or USA)	mm/dd/yy	Config Date = MDY, Separator = /
ANSI	yy.mm.dd	Config Date = YMD, Separator = .
Angleterre/France	dd/mm/yy	Config Date = DMY, Separator = /
Allemagne	dd.mm.yy	Config Date = DMY, Separator = .
Italle	dd-mm-yy	Config Date = DMY, Separator = -
Japon/Taiwan	yy/mm/dd	Config Date = YMD, Separator = /
USA	mm-dd-yy	Config Date = MDY, Separator = -

Voir Aussi
CONFIG CLOCK , DATE TIME , DayOfWeek , DayOfYear , SecOfDay , SecElapsed , SysDay
SysSec , SysSecElapsed , Time , Date

CONFIG DEBOUNCE

Action
Permet de régler le temps de latence de l'instruction DEBOUNCE (Anti-rebond)

Syntaxe
CONFIG DEBOUNCE= time

Remarques
Time Une constante numérique en mS
Quand "Config Debounce = time" n'est pas configuré, 25 ms sera utilisé par défaut.

Voir Aussi
DEBOUNCE, programme exemple DEBOUN.bas

CONFIG GRAPHLCD

Action
Configure l'afficheur LCD graphique.

Syntaxe
Config GRAPHLCD = type, DATAPORT = port, CONTROLPORT=port ,
CE = broche, CD = broche, WR = broche, RD = broche, RESET= broche,
FS = broche, MODE = mode

Remarques

Type Drivers T6963C : 240 * 64, 128* 128, 128 * 64 , 160 * 48 or 240 * 128
Drivers SED1520 : 128 * 64sed ou 120* 64SED uniquement
Drivers KS0108 : 128*64 uniquement (voir le programme KS108.bas)

Dataport Nom du port utilisé pour transmettre les informations aux broches db0-db7. PORTA par exemple.

Controlport Nom du port utilisé pour les broches de contrôles. PORTC par exemple.

Ce Le numéro de la broche utilisée pour l'autorisation.

Cd Le numéro de la broche utilisée pour contrôler la broche Cd.

WR Le numéro de la broche utilisée pour contrôler la broche /WR.

RD Le numéro de la broche utilisée pour contrôler la broche /RD.

FS Le numéro de la broche utilisée pour contrôler la broche FS. Non nécessaire pour les afficheurs SED.

RESET Le numéro de la broche utilisée pour contrôler la broche RESET.

MODE Suivant le nombre de colonnes pour le mode texte. Mode est calculé par le nombre de pixels et le nombre de colonnes désiré. Exemple afficheur de 240 pixels pour 30 colonnes = 240/30 =mode=8 ; pour 40 colonnes 240/40=6

Toutes ces commandes sont valables pour les afficheurs pilotés par le T6963C ou les SED1520. Pour le moment, ce sont les seuls supportés par BASCOM.

Les branchements suivants ont été utilisés pour nos tests T6963C:
PORTA.0 à PORTA.7 pour DB0-DB7 du LCD
PORTC.5 à FS, choix de la police de caractères du LCD
PORTC.2 à CE, chip enable (autorisation) du LCD
PORTC.3 à CD, code/Data sélection du code ou Data
PORTC.0 à WR du LCD, write (écrire)
PORTC.1 à RD du LCD, read (lire)
PORTC.4 à RESET du LCD

Les LCD graphiques ont, en général, besoin d'une tension négative d' environ 17V pour le contraste - voir notice de l'afficheur -

Les afficheurs graphiques pilotés par T6963C ont une zone graphique et une zone texte, ces zones peuvent être utilisées ensemble. Les routines utilisent le mode XOR pour afficher l'une ou l'autre des couches.

Les instructions utilisables en mode graphiques sont :

CIRCLE(x0,y0) , RAYON, COULEUR X0 ,Y0 : centre, RAYON : rayon du cercle, COULEUR : Couleur =255 pour afficher, 0 pour masquer.

CLS efface le texte et la partie graphique.

CLS GRAPH efface la partie graphique seulement.

CLS TEXT efface la partie texte seulement.

CURSOR ON/OFF BLINK/NOBLINK Fonctionne comme pour les afficheurs LCD (texte)

LINE(x0,y0) – (x1,y1) , COULEUR Dessine une ligne à partir de la coordonnée (x0,y) jusqu'à la coordonnée (x1,y1) COULEUR=0 masque la ligne COULEUR=255 l'affiche.

LCD Fonctionne comme pour les afficheurs LCD (texte)

LOCATE Ligne,colonne Place le curseur : la ligne peut varier de 1 à 16
La colonne de 1 à 40, suivant la taille et le MODE de l'afficheur.

PSET X, Y , COULEUR Place un Pixel, l'affiche ou le masque. X varie de 0 à 239 et Y de 9 à 63 (suivant l'afficheur). Si COULEUR =0 le pixel est masqué, si COULEUR=255 le pixel est allumé.

SHOWPIC X, Y , ETIQUETTE X est la colonne, Y la ligne, ETIQUETTE est l'étiquette où se trouve l'information concernant l'image à afficher

$BGF "file.bgf" Insert un fichier BGF à la position actuelle.

Les routines Graphiques sont rangées dans les fichiers glib.lib ou glib.lbx.
On peut relier les broches FS et RESET à condition de changer le Glib.lib. Ces broches sont alors disponibles pour d'autres tâches.

Voir Aussi
Programmes : T6963_240_128.bas, T6963v3.bas, KS108.bas

SHOWPIC, PSET, $BGF, LINE, LCD, CIRCLE

CONFIG I2CDELAY

Action
Directive de compilation qui remplace la routine interne I2Cdelay

Syntaxe
CONFIG I2CDELAY=valeur

Remarques
Valeur Une valeur de 1 à 255
Plus grande est la valeur plus lente est l'horloge I2C.
Pour les routines I2C la valeur d'horloge est calculée en fonction du quartz.
Pour être compatible avec tous les composants I2C la valeur la plus basse est utilisée.

Voir Aussi
CONFIG SCL, CONFIG SDA exemple: I2C.bas, LCD_i2C_DEMO.bas

CONFIG INTx

Action
Configuration de la manière dont les interruptions INT0, INT1, INT4-7 (pour les MEGA) seront commandées.

Syntaxe
CONFIG INTx = etat

Remarques
Etat LOW LEVEL, pour générer une interruption quand le niveau est bas.
Mettre la broche concernée à 0 générera une interruption continuellement.
FALLING, pour générer une interruption sur le front descendant.
RISING, pour générer une interruption sur le front montant.
CHANGE Pour générer une interruption sur un changement .
Les MEGA ont aussi INT0-3 qui sont toujours commandés à 0 donc il n'est pas possible et pas besoin de les configurer.
La plupart des µP ont seulement INT0 et INT1

Voir Aussi
Exemple : INT0.bas

CONFIG KBG

Action
Configuration de la fonction GETKBD( ) et informe quel port utiliser

Syntaxe
CONFIG KBG = PORTx, DEBOUNCE= Valeur [, DELAY=valeur]

Remarques
PORTx Le nom du port à utiliser comme PORTA, PORTD..
DEBOUNCE Par défaut la valeur est 20 . On peut monter jusqu'à 255.
L'instruction GETKBG ( ) peut être utilisée pour lire la touche appuyée par un clavier matricé.
DELAY optionnel un délai d'environ 100 ms protége des éventuels problèmes d'électricité statique.
Il est désormais possible d'utiliser 6 ligne à la place de 4 la configuration devient :
CONFIG KBD = PORTx , DEBOUNCE = value , rows=6, row5=pinD.6, row6=pind.7
Ceci indique que port Row5 doit être connecté à la broche pind.6 et Row6 à pind.7
Note : il ne peut avoir que 4 ou 6 lignes.

Voir Aussi
GETKBG

CONFIG KEYBOARD

Action
Configuration de l'instruction GETATKBG ( )

Syntaxe
CONFIG KEYBOARD = PINX.y, DATA = PINX.y, KEYDATA = table

Remarques

KEYBOARD	La broche utilisée pour l'entrée clock
DATA	La broche utilisée pour l'entrée Data
KEYDATA	L'étiquette où la traduction des touches peut être trouvée.

Les claviers AT ne retournent pas des caractères ASCII, donc une table de traduction est nécessaire. BASCOM permet l'utilisation des touches "shiftées", les touches spéciales comme les touches de fonctions ne sont pas supportées.
Le clavier AT peut être connecté avec 4 fils :CLOCK, DATA, masse et VCC.

Voir Aussi
Les Datasheets ATMEL d'où sont tirées certaines informations concernant les claviers AT. Voir exemple getatkbg.bas et getatkbg_int.bas

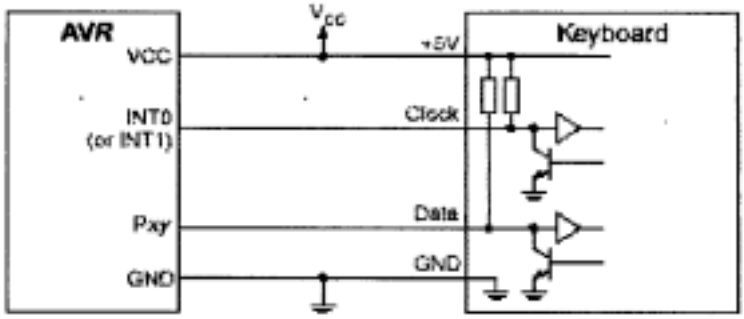


Table 1. AT Keyboard Connector Pin Assignments

AT Computer		
Signals	**DIN41524, Female at Computer, 5-pin DIN 180°**	**6-pin Mini DIN PS2 Style Female at Computer**
Clock	1	5
Data	2	1
nc	3	2,6
GND	4	3
+5V	5	4
Shield	Shell	Shell

CONFIG LCD

Action
Configure les afficheurs LCD et change les paramètres du compilateur.

Syntaxe
Config LCD=LCDtype

Remarques
LDCtype Le type d'afficheur utilisé : 40*4, 16*1, 20*2, 20*4, ou 16*2

16*2 est choisi par défaut. Dans ce cas il n'est pas nécessaire de l'indiquer.

16*1 : est en fait un 8*2 qui a pour adresse de ligne 2 : &H8.

Voir Aussi
Config LCDBUS, Config LCDMODE, Config LCDPIN

CONFIG LCDBUS

Action
Configure le Bus Data LCD et change les paramètres du compilateur.

Syntaxe
Config LCDBUS=constant

Remarques
Constant 4 pour une programmation 4 bis
8 pour une programmation 8 bis

LCDBUS est utilisé quand une RAM externe est utilisée, donc quand le système utilise un BUS d'adresses et un BUS de données. Cette commande fonctionne avec $LCD qui est l'adresse de commande E(Enable) et $LCDRS RS(Register-Select)

Voir Aussi
Config LCD

CONFIG LCDMODE

Action
Configure le mode opératoire du LCD et change les paramètres du compilateur.

Syntaxe
Config LCDMODE=Type

Remarques
Type PORT, le LCD sera piloté en Mode 4-bits (4-5-6-7) par défaut
Dans ce mode, le choix des broches utilisées pour les données et pour les commandes E et RS est libre

Type BUS, on choisira cette commande quand une RAM externe est utilisée, donc quand le système utilise un BUS d'adresses et un BUS de données. Cette commande fonctionne avec $LCD qui est l'adresse des commandes E(Enable) et $LCDRS qui est l'adresse de RS(Register-Select)

Voir Aussi
Config LCD Config LCDPIN

CONFIG LCDPIN

Action
Configure les broches à utiliser dans le mode opératoire PORT et change les paramètres du compilateur.

Syntaxe
Config LCDPIN=PIN, DB4=Pn, DB5=Pn, DB6=Pn, DB7=Pn, E=Pn, RS=Pn

Remarques
Pn le numéro de la broche concernée.
La configuration doit être écrite sur une ligne.

Voir Aussi
Config LCD

CONFIG PIN

voir CONFIG PORT

Config PORT

Action
Configure un port ou une broche du port.

Syntaxe
CONFIG PORTx=etat
CONFIG PINx.y=etat

Remarques
Etat Une constante qui peut être INPUT ou OUPUT
INPUT oriente le DDR (Data Direction Register) pour entrer des données par le port x.
OUTPUT oriente le DDR du port x en sortie.
Etat peut aussi prendre la valeur d'un nombre binaire par exemple :
&B00001111 oriente le quartet supérieur(MSB 0000) en entrée (0) et le quartet inférieur (LSB 1111) en sortie(1)

Pour le CONFIG PINx.y=etat, etat peut être INPUT ou OUTPUT ou 1 ou 0
Il est plus rapide de configurer un port par Config PORT.

Un port doit être à 0 avant d'être mis en INPUT.

Voir Aussi
le programme exemple Port.BAS,

CONFIG RC5

Action
Remplace la broche assignée à RC5 par "option compiler setting"

Syntaxe
CONFIG RC5 = Pin [,TIMER=2]

Remarques
Pin La broche où le récepteur RC5 est connecté
TIMER Doit être le TIMER 2, quand on utilise cette option, c'est pour libérer le TIMER0.
Quand on utilise différentes broches dans différents projets, on peut utiliser cette instruction pour remplacer la broche par défaut. En BASIC BASCOM-AVR les réglages sont dans le fichiers .CFG du projet.

Voir Aussi
GETRC5, RC5SEND les programmes RC5SEND.bas, RC6SEND.bas

CONFIG SCL

Action
Remplace la broche assignée à SCL par "option compiler setting"

Syntaxe
CONFIG SCL = pin

Remarques
Pin La broche où la ligne I2C-SCL est connectée.

Voir Aussi
CONFIG SDA, CONFIG I2CDELAY, I2C instructions, programme I2C.bas

---

CONFIG SDA

Action
Remplace la broche assignée à SDA par "option compiler setting"

Syntaxe
CONFIG SDA = pin

Remarques
Pin La broche où la ligne I2C-SDA est connectée.

Voir Aussi
CONFIG SCL, CONFIG I2CDELAY, I2C instructions, programme I2C.bas

---

CONFIG SERIALIN

Action
Configure l'UART hardware pour utiliser un tampon (Buffer) en entrée.

Syntaxe
CONFIG SERIALIN = BUFFERED, SIZE=size

Remarques
Size Une constante numérique qui indique la largeur du tampon. Cette constante sera incluse dans la mémoire SRAM
Les variables internes suivantes seront générées :
_RS_HEAD_PTR0 Byte : un pointeur vers la place mémoire où le tampon a été écrit.

_RS_TAIL_PTR0 Byte : un pointeur vers la place mémoire où le tampon a été lu.

_RS232INBUF0 Un tableau de Bytes utilisé comme tampon tournant pour recevoir les caractères

Voir Aussi
CONFIG SERIALOUT, exemple : RS232BUFFER.bas

CONFIG SERIALIN1

Action
Configuration de la seconde UART pour utiliser un tampon en entrée

Syntax
CONFIG SERIALIN1 = BUFFERED , SIZE = SZ

Remarques
SZ Une constante numérique qui indique la largeur du tampon. Cette constante sera incluse dans la mémoire SRAM.
Les variables internes suivantes seront générées :
_RS_HEAD_PTR1 Byte : un compteur qui accumule la tête du tampon
_RS_TAIL_PTR1 Byte : un compteur qui accumule la queue du tampon
_RS232INBUF1 Un tableau de Bytes utilisé comme tampon tournant pour recevoir les caractères.

Voir Aussi
CONFIG SERIALOUT1 et le programme exemple RS232BUFFEROUT1.bas

---

CONFIG SERIALOUT

Action
Configure l'UART hardware pour utiliser un tampon en sortie

Syntaxe
CONFIG SERIALOUT = BUFFERED, SIZE=SZ

Remarques
SZ Une constante numérique qui indique la largeur du tampon. Cette constante sera incluse dans la mémoire SRAM.
Les variables internes suivantes seront générées :
_RS_HEAD_PTRW0 Byte qui reçoit la tête du tampon
_RS_TAIL_PTRW0 Byte qui reçoit la queue du tampon
_RS_TAIL_PTRW0 Un tableau de Bytes utilisé comme tampon tournant pour recevoir les caractères à envoyer

Voir Aussi
CONFIG SERIALIN, programme exemple RS232BUFFEROUT.bas

CONFIG SERIALOUT1

Action
Configuration de la seconde UART pour utiliser un tampon en sortie.

Syntaxe
CONFIG SERIALOUT1 = BUFFERED, SIZE = SZ

Remarques
SZ = une constante numérique qui indique la largeur du tampon. Cette constante sera incluse dans la mémoire SRAM.
Les variables internes suivantes seront générées :
_RS_HEAD_PTRW1 Byte : un pointeur vers la tête du tampon
_RS_TAIL_PTRW1 Byte : un pointeur vers la queue du tampon
_RS232OUTBUF1 Un tableau de Bytes utilisé comme tampon tournant pour recevoir les caractères à envoyer .

Voir Aussi
CONFIG SERIALIN1 le programme RS232BUFFEROUT1.bas

CONFIG SERVOS

Action
Configuration d'un contrôle de servo-moteur.

Syntaxe
CONFIG SERVO = X, servo1= portn.x, Servo2=portm.y, reload=rl

Remarques

X	Le nombre de servos
Servo1	Le port utilisé par le servo1
RL	L'intervalle de rafraîchissement en µs

TIMER0 est utilisé pour l'interruption ISR (RL)

Voir Aussi
L'exemple Servos.bas

CONFIG SPI

Action
Configuration du composant SPI (Serial Protocol Interface)

Syntaxe pour une communication soft
CONFIG SPI=MODE, DIN=PIN, DOUT=PIN, SS=PIN, CLOCK=PIN

Syntaxe pour une communication hard
CONFIG SPI=MODE, DATA ORDER=DO, MASTER=M, POLARITY=PY, PHASE= PH, CLOCKRATE=CR

Remarques
MODE SOFT pour une émulation soft du SPI, ce qui laisse l'utilisateur libre de choisir les ports à utiliser.
HARD pour une utilisation utilisant la configuration du µP, les broches ne sont pas modifiables.

DIN	Broche pour Data input ou MISO
DOUT	Broche pour Data output ou MOSI
SS	Broche pour SLAVE SELECT
CLOCK	Broche pour Clock

DO	Choix du type de transmission effectué en premier: DO= MSB ou LSB
M	YES (maître) ou NO (esclave)
PY	HIGH pour mettre la ligne CLOCK à 1 quand le SPI est inoccupé LOW pour mettre la ligne CLOCK à 0 quand le SPI est inoccupé
PH	0 ou 1 Voir la datasheet pour connaître les combinaisons Polarity/phase.
CLOCKRATE	Facteur de division de la fréquence du quartz.

En mode HARD le réglage de départ est:
DATA ORDER=MSB, MASTER=YES, POLARITY=HIGH, PHASE= O, CLOCKRATE=4

Voir Aussi
SPIIN, SPIINIT, SPIMOVE, SPIOUT,les programmes : spi-slave.bas, sendspi.bas, spisoftware.bas

CONFIG TIMER0

Action
Configuration du TIMER0 en timer ou en compteur.

Syntaxe
CONFIG TIMER0=COUNTER, EDGE=ED
CONFIG TIMER0=TIMER, PRESCALE=PS

Remarques
Configuration Compteur sur 8 bits COUNTER
ED Rising ou Falling suivant que le compteur incrémentera sur le front montant ou descendant du signal déclenchant.

Configuration en TIMER
PS 1, 8, 64, 256, 1024, diviseur de la fréquence d'horloge.

Pour démarrer un Timer
START TIMER0

Pour arrêter un Timer
STOP TIMER0
Bien entendu le timer doit être configuré avant !
Il est possible d'utiliser plusieurs configurations du Timer dans le même programme.

Voir Aussi
Programmation Avancée, Programme Timer0.bas

CONFIG TIMER1

Action
Configuration du Timer1

Syntaxe
CONFIG TIMER1 =TM, EDGE=ED, PRESCALE=PS, NOISE CANCEL=NC, CAPTURE EDGE=CE, COMPARE A=CA, COMPARE B=CB, PWM=PW, COMPARE A PWM=CAP, COMPARE B PWM=CBP

Remarques
Le timer1 est un timer 16 bits, voir "programmation avancée"
Certains µP ne supportent pas le COMPARE B.

TM COUNTER, TIMER ou PWM
ED RISING ou FALLING suivant que le compteur incrémentera sur le front montant ou descendant du signal déclenchant.
PS 1, 8, 64, 256, 1024 diviseur de la fréquence d'horloge.
NC 0 ou 1, 1 permet la suppression des bruits pendant le comptage.
CE concerne la broche ICP, voir ED
Quand la valeur de Timer1 est égale à un registre de comparaison, on peut déclencher une action qui peut être :
CA , CB SET ajuste la broche 0C1X
CLEAR efface la broche OC1X
TOGGLE bascule la broche OCX1
DISCONNECT le Timer1 de la broche OC1X

Mode PWM

PW	8,9 ou 10
CAP,CBP	CLEAR UP, CLEAR DOWN, DISCONNECT

Si on utilise COMPARE A, COMPARE B, COMPARE A PMW, COMPARE B PWM, cela met la broche correspondante en sortie. Quand cet état n'est pas désiré, on peut utiliser la version NO_OUTPUT.
Ex : COMPARE A NO_OUTPUT, COMPARE A PWM NO_OUTPUT

Voir Aussi
Programme TIMER1.bas

CONFIG TIMER2

Action
Configuration du TIMER2

Syntaxe
Le timer2 est un timer 8 bits, voir "programmation avancée"
Il n'est présent que sur certains µP
La syntaxe décrite ci-dessous doit être écrite sur 1 ligne. Toutes les options ne sont pas toutes à préciser.

Pour le 8535

CONFIG TIMER2= mode, ASYNC=AS, PRESCALE=PS, COMPARE=C, PWM=P, COMPARE PWM=CP

Pour le M103

CONFIG TIMER2= mode, EDGE= ED, PRESCALE=PS, COMPARE=C, PWM=P, COMPARE PWM=CP

Remarques

Mode	TIMER ou PWM pour le 8535, TIMER, COUNTER, PWM pour le M103
AS	On ou Off uniquement pour le 8535
PS	1, 8, 32, 64, 128, 256, 1024 pour le 8535 1, 8, 64, 256, 1024 pour le M103
ED	Rising ou Falling uniquement pour le M103, et seulement pour le compteur.
C	CLEAR, SET, TOGGLE, DISCONNECT
P	On ou Off
CP	CLEAR UP, CLEAR DOWN, DISCONNECT

Exemple
```
Dim W As Byte
Config Timer2 = Timer , ASYNC = 1 , Prescale = 128
On TIMER2 Myisr                    'va à l'étiquette MYisr
ENABLE INTERRUPTS
ENABLE TIMER2
DO

LOOP

MYISR:
'Vient ici chaque seconde avec un quartz 32768 KHz
RETURN
```

on peut lire ou écrire dans le timer avec les variables COUNTER2 ou TIMER2
```
W = Timer2
Timer2 = W
```

CONFIG WAITSUART

Action
Directive de compilation qui indique que l'UART soft se met en attente après l'envoi du dernier Byte.

Syntaxe
CONFIG WAITSUART= valeur

Remarques
Valeur Une valeur numérique entre 1 et 255, une valeur importante indique un délai important.
Quand la routine UART soft est utilisée en même temps qu'un LCD SERIE, il faut spécifier un délai pour que l'afficheur puisse traiter les DATA.

Voir Aussi
OPEN et l'exemple : Open.bas

CONFIG WATCHDOG

Action
Configuration du timer watchdog (chien de garde)

Syntaxe
CONFIG WATCHDOG = time

Remarques
Time Intervalle constant en ms que comptera le timer watchdog avant de redémarrer le programme.
Réglage possible 16, 32, 64, 18, 256, 512, 1024, 2048
Quand le WD est démarré, un reset arrive après le nombre de ms spécifié.
Avec 2048 un reset arrivera après 2 secondes, donc on doit redémarrer le WD dans le programme périodiquement avec l'instruction RESET WATCHDOG

Voir Aussi
START WATCHDOG, STOP WATCHDOG, RESET WATCHDOG et le programme WATCHD.bas

---

CONFIG X10

Action
Configure les broches utilisé pour les transmission X10 (courants porteurs) Les équipements X10 doivent être homologués EDF, MCSELEC ni le traducteur ne peuvent être tenu responsables d'une utilisation non homologuées.

Syntaxe
CONFIG X10 = BrocheZC , TX = portpin

Remarques
PinZC La broche qui est connecté à la sortie Zéro-cross du TW-523 c'est la broche qui sera utilisé en INPUT
Portpin La broche qui est connecté au TX du TW-523. TX est utilisé pour envoyé des données X10 au TW-523 . Cette broche est utilisé en mode OUTPUT le connecteur RJ-11 du TW-523 à les connexions suivantes : TW-523 RJ-11 connector has the following pinout:

Broche	Description	Connexion micro
1	Zero Cross	broche Input. Ajouté une 5.1K pull up.
2	GND	GND
3	RX	Not used.
4	TX	broche Output. Ajouté une 1K pull up.

Voir Aussi
X10DETECT , X10SEND

INSTRUCTIONS GENERALES

1WIRECOUNT

Action
Cette instruction lit le nombre de composants 1WIRE reliés au bus.

Syntaxe
Var2=1WIRECOUNT( )
Var2=1WIRECOUNT(Pinname, pin)

Remarques

Var2	Une variable Word à qui est assigné le nombre de composants rattachés.
Pinname	Le nom de la broche (pinB, pinD)
Pin	Le numéro de la broche (0---7)

On utilise cette instruction pour connaître le nombre de fois que 1Wsearchnext( ) est utilisé pour récupérer tous les ID number[2] des composants sur le bus.
Cette instruction occupe 4 Bytes de la SRAM

_1wbitstorage Byte utilisé pour le stockage des bits :

lastdeviceflag	Bit0
id_bit	Bit1
cmp_id_bit	Bit2
Searchbit	Bit3

_1wid_bit_number	Byte
_1wlast_zero	Byte
_1wlast_discrepancy	Byte (divergence)

1WREAD

Action
Cette instruction lit les Data du bus 1Wire.

Syntaxe
Var2=1WREAD([Bytes])
Var2= 1WREAD(Bytes, pinname, pin)

Remarques

Var2	La fonction lit un Byte provenant du bus et l'attribue à la variable.
Pinname	Le nom de la broche (pinB, pinD)
Pin	Le numéro de la broche (0---7)

Cette fonction est supportée par différentes broches (voir l'exemple 1WIRE.bas)
La syntaxe dans ce dernier cas est la suivante:
1WRESET, pinname, pin
1WWRITE var/constant, Bytes, pinname, pin
var=1WREAD(Bytes, pinname,pin)

Voir Aussi
1WRESET, 1WWRITE, 1WIRECOUNT,1WSEARCHFIRST,1WSEARCHNEXT, 1WVERIFY, Programme 1Wiresearch.bas, 1wire.bas

[2] N° d'identification

1WRESET

Action
Cette instruction met la broche 1WIRE dans un état de fonctionnement correct en envoyant un RESET au bus.

Syntaxe
1WRESET
1WRESET, Pinname, pin (dans le cas d'une application multiple broches 1WIRE)

Remarques
Pinname Le nom de la broche (pinB, pinD)
Pin Le numéro de la broche (0---7)
La variable ERR est mise à 1 si une erreur intervient

Voir Aussi
1WREAD, 1WWRITE, 1WIRECOUNT,1WSEARCHFIRST,1WSEARCHNEXT
1WVERIFY, Programme 1Wiresearch.bas, 1wire.bas

1WSEARCHFIRST

Action
Cette instruction attribue le premier ID du bus 1WIRE dans un tableau

Syntaxe
Var=1WSEARCHFIRST( )
Var=1WSEARCHFIRST (Pinname, pin)

Remarques
Var Un tableau de 8 Bytes à qui seront assignés les 8 Bytes du premier composant sur le bus.
Pinname Le nom de la broche (pinB, pinD)
Pin Le numéro de la broche (0---7)

La fonction 1WSEARCHFIRSTdoit être appelée une fois à l'initialisation pour démarrer le processus de reconnaissance des ID. Ensuite il faut utiliser la fonction 1WSEARCHNEXT pour récupérer les autres ID qui sont sur le bus.

Voir Aussi
1WRESET,1WREAD, 1WWRITE, 1WIRECOUNT,1WSEARCHNEXT
1WVERIFY, Programme 1Wiresearch.bas, 1wire.bas

1WSEARCHNEXT

Action
Cette instruction attribue le prochain ID du bus 1WIRE dans un tableau

Syntaxe
Var= 1WSEARCHNEXT()
Var= 1WSEARCHNEXT (Pinname, pin)

Remarques
Var un tableau de 8 Bytes à qui seront assignés les 8 Bytes du prochain composant sur le bus.
Pinname Le nom de la broche (pinB, pinD)
Pin Le numéro de la broche (0---7)

La fonction 1WSEARCHFIRSTdoit être appelée une fois à l'initialisation pour démarrer le processus de reconnaissance des ID. Ensuite il faut utiliser la fonction 1WSEARCHNEXT pour récupérer les autres ID qui sont sur le bus.

Voir Aussi
1WRESET, 1WREAD, 1WWRITE, 1WIRECOUNT, 1WSEARCHFIRST, 1WVERIFY, Programme 1Wiresearch.bas, 1wire.bas

---

1WVERIFY

Action
Cette instruction vérifie si un ID est disponible sur le bus 1WIRE

Syntaxe
1Wverify ar(1)

Remarques
AR(1) un tableau de Bytes qui contient les ID à vérifier

Voir Aussi
1WRESET, 1WREAD, 1WWRITE, 1WIRECOUNT, 1WSEARCHFIRST, 1WSEARCHFIRST, Programme 1Wiresearch.bas, 1wire.bas

---

1WWRITE

Action
Cette instruction écrit la valeur d'une variable sur le bus 1Wire.

Syntaxe
1WWRITE var
1WWRITE var, Bytes
1WWRITE var, Bytes, pinname, pin

Remarques
Var La fonction lit un Byte provenant du bus et l'attribue à la variable.
Bytes Le nombre de Bytes à écrire, optionnel si pinname et pin ne sont pas utilisés
Pinname Le nom de la broche (pinB, pinD)
Pin Le numéro de la broche (0---7)

Cette fonction est supportée par différentes broches (voir l'exemple 1WIRE.bas)
La syntaxe dans ce dernier cas est la suivante:
1WRESET, pinname, pin
1WWRITE var/constant, Bytes, pinname, pin
var=1WREAD(Bytes, pinname,pin)

Voir Aussi
1WRESET, 1WWRITE, 1WIRECOUNT,1WSEARCHFIRST,1WSEARCHNEXT, 1WVERIFY, Programme 1Wiresearch.bas, 1wire.bas

---

ABS( )

Action
Retourne la valeur absolue d'un nombre signé

Syntaxe
Var=ABS(Var2)

Remarques

Var	Variable INTEGER ou LONG
Var2	Variable INTEGER ou LONG

Ne fonctionne pas avec les SINGLE en revanche Var peut être une WORD

Voir Aussi
programme conversions.bas

ALIAS

Action
Indique que la variable peut être référencée par une autre.

Syntaxe
Newvar ALIAS Oldvar

Remarques
Oldvar= par exemple **PortB.1**
Newvar= par exemple **direction**
If valeur=325 **then direction**=1 ‘change la valeur de portB.1
Donner un nom d’alias à un port permet de mieux le définir.

Voir Aussi
Const

ASC

Action
Retourne la valeur ASCII du premier caractère d'une String

Syntaxe
Var=ASC(String)

Remarques
Var Une variable Byte, Integer, Word ou Long
String peut aussi être une constante

Voir Aussi
CHR, programme conversions.bas

BAUD

Action
Change la vitesse de transfert de l'UART

Syntaxe
BAUD = Var
BAUD #x, const

Remarques
Var La vitesse qui sera utilisée
X Le N° de canal de L'UART(soft)
Const Une constante numérique pour la vitesse
Ne pas confondre l'instruction BAUD avec la directive de compilation $BAUD
Idem pour $Crystal et Crystal
$BAUD Change le réglage du compilateur tandis que BAUD change la vitesse en cours de programme.
BAUD=… concerne l'UART hardware
BAUD #x.yyyy concerne l'UART soft

Voir Aussi
$Crystal, $Baud, programmation avancée UART

BCD

Action
Pour convertir une variable au format BCD en String

Syntaxe
Print BCD(Var)
LCD BCD(Var)

Remarques
Var Byte, Integer, Word, Long, Constant
Le format BCD (Binary Code Decimal) est très utilisé par certains afficheurs et par le protocole I2C

Tableau de correspondance:

Poids →	**BCD ↓**	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**
Nombre ↓	**Puissance 2 →**					**3**	**2**	**1**	**0**
0	0					0	0	0	0
1	1					0	0	0	1
2	2					0	0	1	0
3	3					0	0	1	1
4	4					0	1	0	0
5	5					0	1	0	1
6	6					0	1	1	0
7	7					0	1	1	1
8	8					1	0	0	0
9	9					1	0	0	1
10	10	0	0	0	1	0	0	0	0
11	11				1	0	0	0	1
12	12				1	0	0	1	0
13	13				1	0	0	1	1
14	14				1	0	1	0	0
15	15				1	0	1	0	0

Changement de poids à chaque dizaine.

Voir Aussi
MAKEBCD, MAKEDEC, programme conversions.bas

BIN

Action
Convertit une variable numérique en représentation binaire sous forme de String.

Syntaxe
VarString=BIN(var)

Remarques
VarString La String créée, elle doit être dimensionnée en autant de bits que générera la transformation.
Var Une variable numérique

Voir Aussi
BINVAL, STR, VAL, HEX, HEXVAL., programme conversions.bas

BIN2GREY / GREY2BIN

Action
Renvoie le code GREY d'une variable. (BIN2GREY)
Renvoie le code binaire d'une variable en code GREY. (GREY2BIN)

Syntaxe
var1 = bin2grey(var2)
var3 = grey2bin(var2)

Remarques

var1	Variable qui reçoit le code GREY.
var2	Variable qui est convertie.
Var3	Variable qui reçoit le code binaire.

Le code GREY est utilisé pour les encodeurs rotatifs (roues codeuses)
BIN2GREY fonctionne avec des variables Bytes, Integer, Word, Long

Voir Aussi
Le programme exemple :Geycode.bas

BINVAL

Action
Pour convertir une variable String représentant une valeur binaire en variable numérique.

Syntaxe
Var=BINVAL(varString)

Remarques

VarString	La String à transformer.
Var	Une variable numérique

Voir Aussi
BIN, STR, VAL, HEX, HEXVAL., programme conversions.bas

BITS()

Action
Met tout les bits spécifié à 1.

Syntaxe
Var = Bits(b1 [,bn])

Remarques

Var	Le BYTE/PORT qui est utilisé .
B1 , bn	Une liste de bit qui doivent être mis à1.

Bien qu'il est simple de programmer un byte en notation binaire &B1000001
Il est plus simple et plus sûr d'écrire Bits(0,6)
C'est plus lisible sans prendre plus de code ni de place mémoire.
Les bits vont de 0 à 7

Voir Aussi
NBITS exemple dans l'aide du programme

BITWAIT

Action
Attend jusqu'à ce qu'un bit soit sélectionné ou désélectionné.

Syntaxe
Bitwait x, set /reset

Remarques
X Variable bit ou registre interne comme Portb.x où x varie de 0 à 7
Quand on utilise Bitwait, il faut s'assurer que la variable sera sélectionnée ou désélectionnée, sinon le programme tournera en boucle.
Ceci ne s'applique pas quand on utilise un bit qui peut être changé par Hardware comme Portb.0 qui peut être le résultat d'un appui sur un bouton.

Voir Aussi
Programme Exemple : Port.bas

BLOAD

Action
Ecrit le contenu d'un fichier dans la SRAM

Syntaxe
BLoad sFileName, wSRAMPointer

Remarques
sFileName (String) Nom du fichier qui doit être lu.
wSRAMPointer(Word) Variable, qui contient l'adresse SRAM d'où la SRAM doit réécrire le fichier.
Cette fonction écrit le contenu d'un fichier dans un espace de SRAM choisi.

Voir Aussi
Bsave et l'aide dans le fichier Help pour un exemple.

BSAVE

Action
Sauve un champ de la SRAM dans un fichier Save a range in SRAM to a File

Syntaxe
BSave sFileName, wSRAMPointer, wLength

Remarques
sFileName (String) Nom du fichier qui doit être écrit.
wSRAMPointer (Word) Variable, qui contient l'adresse SRAM d'où la SRAM doit écrire dans le fichier.
wLength (Word) Copteur de bytes de la SRAM qui doit être écrit dans le fichier

Cette fonction écrit un champs dans un fichier. Un fichier libre doit être disponible pour cette fonction.

Voir Aussi
INITFILESYSTEM , OPEN , CLOSE, FLUSH , PRINT, LINE INPUT, LOC, LOF , EOF , FREEFILE , FILEATTR , SEEK , BLOAD , KILL , DISKFREE , DISKSIZE , GET , PUT
FILEDATE , FILETIME , FILEDATETIME , DIR , FILELEN, WRITE , INPUT
L'aide dans le fichier Help pour un exemple.

BUFSPACE()

Action
Returne la quantité d 'espace libre dans le buffer d'une connexion série.

Syntaxe
Var = BufSpace(n)

Remarques
Var Une variable Word ou Integer.
N Une constante variant de 0-3 mais pour le moment, seule 0 qui correspond à la première UART est disponible. Bien qu'avec le buffer vous n'avez pas à attendre et donc à bloquer le µP, le buffer peut se remplir quand le µP n'a pas le temps de le vider. Avec cette fonction on peut surveiller la place restante avant de remettre des données dans le buffer.

Voir Aussi
Les fonctions relatives aux ports séries et spécialement config serialout

BYVAL / BYREF

Action
Spécifie que la variable est transmise par valeur.(Byval) ou par référence(Byref)

Syntaxe
Sub test(Byval var)
Sub test(Byref var)

Remarques
Var Nom de variable
BYVAL Var n'est pas transformée par la SUB ou la Function
BYREF : c'est par l'intermédiaire de son adresse (la référence) que la variable est transmise, donc elle sera transformée par la SUB ou la Function

Voir Aussi
BYREF, SUB, programme exemple : Declare.bas

CALL

Action
Appelle et exécute une procédure (SUB)

Syntaxe
Call Proc [(var1,varx)]

Remarques
Var1, Varx N'importe quelle variable ou constante

- ⇨ On peut appeler une SUB avec ou sans paramètres.
- ⇨ La déclaration « declare sub » doit se faire avant le CALL.
- ⇨ Le nombre de paramètres déclarés doit être égal au nombre de paramètres demandés.
- ⇨ Les constantes ne peuvent être appelées que « byval »
- ⇨ Call n'est pas obligatoire, dans ce cas il faut aussi supprimer les ( )

Exemple : mesure var1, Varx

Voir Aussi
Declare, sub, exit, function, local et programme DECLARE.Bas

CHECKSUM

Action
Renvoie un total de contrôle appelé « checksum »

Syntaxe
Print Checksum(var)
B=Checksum(var)

Remarques
Var Variable String
B Variable numérique calculée par l'instruction Checksum
Cette instruction calcule à partir de tous les Bytes de la String, Checksum est très utilisé dans les liaisons séries.

Voir Aussi
CRC8

CHR

Action
Pour convertir une variable ou une constante en une String de 1 caractère représentant la valeur ASCII de la valeur numérique.

Syntaxe
Svar=**Chr**(var)

Remarques
Svar Une String de 1 caractère
Var Un Byte ou un Integer de 0 à 255
En Association avec LCD, permet d'afficher les caractères personnalisés.
Numvar=52
Print numvar résultat = 52
Print **chr**(numvar) résultat= 4 52 est la valeur ASCII du caractère 4

Voir Aussi
En annexe le tableau de correspondance ASCII, le programme conversions.bas

---

CIRCLE

Action
Dessine un cercle sur un afficheur graphique.

Syntaxe
CIRCLE(x0,y0) , rayon, couleur

Remarques
X0 ,Y0 Centre
RAYON Rayon du cercle
COULEUR Couleur =255 pour afficher, 0 pour masquer.

Voir Aussi
SHOWPIC , PSET , $BGF , LINE , LCD et programme T6963_240_128.bas et T6963v3.bas

---

CLS

Action
Efface l'afficheur LCD et place le curseur en position 1

Syntaxe
CLS
Pour le LCD graphique
CLS
CLS TEXT
CLS GRAPH

Remarques
Effacer l'afficheur n'efface pas la CG-RAM où sont stockés les caractères personnalisés.
Pour les afficheurs graphiques, CLS efface le texte et le graphisme.

Voir Aussi
$LCD, LCD, SHIFTLCD, SHIFTCURSOR, DISPLAY.

CLOCKDIVISION

Action
Utilisable seulement sur les MEGA , règle le système d'horloge.

Syntaxe
Clockdivision=var

Remarques
Var Une variable ou constante qui règle la division d'horloge. Les valeurs autorisées varient de 2 à 127, une valeur de 0 supprime la division.
Sur les MEGA la fréquence d'horloge peut être divisée pour réduire la consommation par exemple. Les instructions reliées à la fréquence d'horloge ne fonctionnent plus correctement quand on utilise cette division, par exemple Waitms est multiplié par 2 pour une division de 2.

Voir Aussi
Powersafe.

CLOSE

Action
ferme un composant

Syntaxe
Open « Device » for Mode as #canal
Close #canal

Remarques
Les explications d'Open et Close sont données au mot Open.

Voir Aussi
Open, Print

CONST

Action
Déclaration d'une constante symbolique

Syntaxe
CONST symbole =valeur numérique
CONST symbole =valeur String
CONST symbole =expression

Remarques
Symbole Le nom de la constante
Les constantes ne consomment pas de mémoire programme, elles ne servent que de référence au compilateur.

Voir Aussi
ALIAS et le programme Const.bas

COUNTER0 & COUNTER1

Action
Ajuste ou récupère le registre interne16 bits

Syntaxe
COUNTER0 = var — on peut aussi utiliser TIMER0
Var = COUNTER0

COUNTER1 = var — on peut aussi utiliser TIMER1
Var = COUNTER1

CAPTURE1 = var — registre de capture de TIMER1
Var = CAPTURE1

COMPARE1A = var — registre COMPARE A de TIMER1
Var = COMPARE1A

COMPARE1B = var — registre COMPARE B de TIMER1
Var = COMPARE1B

PWM1A = var — registre COMPARE A de TIMER1, utilisé pour PWM
Var = PWM1A

PWM1B = var — registre COMPARE A de TIMER1, utilisé pour PWM
Var = PWM1B

Remarques
Var — Une constante ou variable Byte, Integer ou Word

Timer0 est un registre 8 bits mais la syntaxe est identique.

Voir Aussi
Le chapitre "Programmation avancée" et le programme Timer1.bas

---

CPEEK

Action
Retourne un Byte stocké dans la mémoire code

Syntaxe
Var= CPEEK(adresse)

Remarques
Var — Variable numérique à laquelle sera attribuée la valeur contenue à l'adresse mémoire.
Adresse — Constante ou variable numérique

Il n'y a pas de CPOKE car le programme ne peut pas se réécrire sur lui-même !
Cpeek(0) retourne le 1° Byte du fichier binaire, Cpeek(1) retourne le second.

Voir Aussi
PEEK, POKE, INP, OUT, programme PEEK.bas

CPEEKH

Action
Retourne un Byte stocké dans la page supérieure de la mémoire code du M103

Syntaxe
Var= CPEEKH(adresse)

Remarques

Var	Variable numérique à laquelle sera attribuée la valeur contenue à l'adresse mémoire.
Adresse	Constante ou variable numérique

Il n'y a pas de CPOKE car le programme ne peut pas se réécrire sur lui-même !
CpeekH(0) retourne le 1° Byte des 64 KBytes supérieures du fichier binaire,
Cette fonction ne s'applique qu'au M103.

CRC8 CRC16

Action
Renvoie la valeur CRC8 (CRC16) d'une variable ou d'un tableau.

Syntaxe
Var=CRC8(source, L)

Remarques

Var	la variable qui reçoit la valeur CRC8 (CRC16) de la source.
Source	la variable source ou le premier élément d'un tableau
L	le nombre de Bytes à contrôler.

CRC8 (CRC16) est utilisé dans les protocoles de communication pour contrôler les éventuelles erreurs de transmission.
Le protocole 1Wire, par exemple renvoie un bit CRC qui est le dernier Byte du n°ID.

Voir Aussi
CHECKSUM, l'exemple dans l'aide de BASCOM, cet exemple utilisant beaucoup de routines assembleur dépasse le cadre de cet ouvrage.

CRYSTAL

Action
Variable Byte qui peut servir à changer la vitesse de l' UART soft

Syntaxe
_crystal = var
baud#1, 2400

Remarques
Cette instruction est pratique dans les µp comportant un oscillateur intégré Attini22 par exemple, dont la fréquence n'est pas très précise.

Voir Aussi
L'aide en ligne, OPEN, CLOSE

CURSOR

Action
Règle l'état du curseur du LCD texte et graphique

Syntaxe
Cursor X Y

Remarques
X On ou OFF
Y BLINK (clignote) ou NOBLINK (ne clignote pas)
On peut utiliser l'un ou l'autre ou les deux paramètres.
Au démarrage l'état est ON et NOBLINK

Voir Aussi
DISPLAY, LCD et le chapitre concernant le LCD et le programme LCD.bas

---

DATA

Action
Spécifie les valeurs constantes à lire avec l'instruction READ

Syntaxe
DATA var, [varn]

Remarques
Var, varn valeur constante numérique ou alphanumérique.
Pour stocker un signe comme " ou un caractère spécial (chr(7)) on utilise $ suivi de la valeur ASCII du signe : $34

L'instruction DATA permet de générer un fichier EEP(eeprom), il existe deux directives de compilation $DATA et $EEPROM pour ce faire. Voir "Programmation avancée" et les programmes Eeprom.bas et Eeprom2.bas

Les Data doivent être écrites en fin de programme, sous l'instruction END.

Voir Aussi
READ, RESTORE, $DATA, $EEPROM et le programme READDATA.bas

DATE

Action
Retourne la valeur de la date soit sous forme de String ou sous 3 Bytes pour Day, Month et Year) suivant le type de la variable cible

Syntaxe
bDayMonthYear = Date(lSysSec)
bDayMonthYear = Date(lSysDay)
bDayMonthYear = Date(strDate)

strDate = Date(lSysSec)
strDate = Date(lSysDay)
strDate = Date(bDayMonthYear)

Remarques
StrDate Une variable String Date dans le format spécifié par l'instruction CONFIG DATE
LsysSec Une variable LONG qui contient le « System Second » (SysSec = TimeStamp)
LsysDay Une variable WORD qui contient alors « System Day » (SysDay)
BDayMonthYear Une variable BYTE qui contient « Days », suivi de « Month » (Byte) et « Year » (Byte)

Voir Aussi
DAYOFYEAR, SYSDAY, les nombreux exemples de l'aide.

Datetime

C'est la librairie qui fonctionne avec les instructions précédentes, écrite par Josef Franz Vögel. Elle augmente les fonctionnalités de gestion du temps.

Les fonctions suivantes sont disponibles :

var = DayOfWeek()
var = DayOfWeek(bDayMonthYear)
var = DayOfWeek(strDate)
var = DayOfWeek(wSysDay)
var = DayOfWeek(lSysSec)

Time
Date

Date et Time à ne pas confondre avec Date$ et Time$!

Voir les nombreux exemple de l'aide

DATE$ / TIME$

Action
Variable interne qui garde la date (l'heure)

Syntaxe
DATE$ = "mm/dd/yy"
TIME$ = "hh:mm:ss"
Var = DATE$
Var = TIME$

Remarques
Ces instructions sont utilisées en combinaison avec CONFIG CLOCK.
L'instruction CONFIG CLOCK utilise TIMER0 et TIMER2 en mode asynchrone pour créer une interruption chaque seconde. Dans cette interruption, les variables _dat, _month et _year sont mises à jour. Elles sont aussi mises à jour par _Sec, _Min et _Hour.
Le format est identique au format QuickBasic® ou VisualBasic®
Quand une variable String est attribuée aux valeurs de DATE$(TIME$), elle évolue avec ces valeurs, Idem pour une constante. La réciproque est vraie pour le réglage de date(heure).
Bien entendu, la date doit être réglée pour être mise à jour ensuite !

Les timers asynchrones ne sont accessibles que dans les µp M8,103,128 , 8535, M163 et M32.

Voir Aussi
TIMER, CONFIG CLOCK et le programme Megaclock.bas

DBG

Action
Donne des informations de debuggage sur l'UART

Syntaxe
DBG

Voir Aussi
$DBG

DEBOUNCE

Action
Anti-rebond sur une broche pour un switch.

Syntaxe
DEBOUNCE Portx.Y, etat, etiquette [, SUB]

Remarques

Portx.Y	Une broche de port
Etat	0 si le branchement se fait quand la broche est à 0, ou inversement.
Etiquette	Etiquette de branchement avec un GOTO si SUB n'est pas spécifié.
SUB	Branchement à l'étiquette de SUB

Voir l'utilisation des branchements GOTO dans les "généralités"
L'instruction DEBOUNCE attend que le port change d'état, quand cela se réalise, il y a un temps d'attente de 25 ms et un second contrôle pour éliminer les rebonds.
Si la condition est toujours bonne, il y a un branchement à l'étiquette.
Si DEBOUNCE est exécuté à nouveau, la broche doit être revenue dans son état initial avant un nouveau branchement.

Chaque instruction DEBOUNCE, pour différentes broches, utilise 1 bit de mémoire pour "se souvenir" de l'état.
Debouce n'attend pas le changement d'état, pour attendre un changement d'état, utiliser DEBOUNCE avec BITWAIT

Voir Aussi
CONFIG DEBOUNCE, programme DEBOUN.bas, BITWAIT

DDRx

Action
Oriente les ports du µ-contrôleur en entrée ou en sortie

Syntaxe
DDRx=&B01010101

Remarques
X Nom du port A,B,C,D…
Si un bit est à 1 la broche correspondante est en sortie, sinon elle est en entrée
Cette "instruction" est le nom du registre $11, un bon début dans le langage Assembleur.

Voir Aussi
Config port, config pin . Programme exemple port.bas…

DECLARE FUNCTION

Action
Déclare une fonction utilisateur.

Syntaxe
DECLARE FUNCTION mafonction[([mode] var as type)] as type

Remarques
Mode Méthode de transfert des variables à la fonction:
BYVAL pour passer une copie de la variable, cette variable ne sera pas modifiée par la fonction
BYREF pour passer l'adresse de la variable, si cette variable est modifiée par la fonction elle restera modifiée au retour.
Var Variable(s) transmise(s) à la fonction
Type Type des variables transmises et du résultat, Bytes, Integer, Word, long, Single, String ou tableaux.

⇨ Les bits sont des variables type global, ils ne peuvent pas être transmis à la fonction.

⇨ La fonction est assimilée à une nouvelle variable, elle doit être dimensionnée (as type) et dans le corps du programme, on donnera sa valeur :
Mafonction = j+val(mot)

⇨ On termine la rédaction d'une fonction par : End Function.

⇨ Les fonctions s'écrivent après le END du programme.

Exemple : Declare Function Mafunction(byval j As Integer , mot As String) As Integer

Voir Aussi
Call, Sub, programme déclare.bas et function.bas

DECLARE SUB

Action
Déclare une procédure utilisateur.

Syntaxe
DECLARE SUB masub[([mode] var as type)] as type

Remarques
Mode Méthode de transfert des variables à la fonction
BYVAL pour passer une copie de la variable, cette variable ne sera pas modifiée par la SUB
BYREF pour passer l'adresse de la variable, si cette variable est modifiée par la SUB elle restera modifiée au retour.
Var Variable(s) transmise(s) à la SUB
Type Type des variables transmises et du résultat, Bytes, Integer, Word, long, Single, String ou tableaux.

- ⇨ Les bits sont des variables type global, ils ne peuvent pas être transmis à la SUB.
- ⇨ La SUB est assimilée à une nouvelle instruction, elle n'est pas dimensionnée
- ⇨ En revanche les variables transférées doivent être dimensionnées
- ⇨ On termine la rédaction d'une SUB par : END SUB
- ⇨ Les fonctions s'écrivent après le END du programme.

La SUB réalise un ensemble d'instructions répétitif.
Exemple : Declare masub(byval j As Integer , mot As String)

Voir Aussi
Call, Function, programme déclare.bas

DECR (INCR)

Action
Décrémente (incremente) une variable d'une unité

Syntaxe
DECR var
INCR var

Remarques
VAR Variable à décrémenter, (n'importe quelle variable numérique sauf BIT)

Voir Aussi
INCR, programmes DECR.bas, INCR.bas

DEFxxx

Action
Déclare toutes les variables qui ne sont pas dimensionnées.

Syntaxe

DEFBIT b	Définit BIT
DEFBYTE c	Définit BYTE
DEFINT i	Définit INTEGER
DEFWORD x	Définit WORD
DEFLNG l	Définit LONG
DEFSNG s	Définit SINGLE

Remarques
Les variables commençant par les lettres qui suivent les DEFxxx sont considérées comme étant dimensionnées comme xxx.
Exemple :
Defbit b
Set B1
Reset bonjour
B1 et Bonjour sont des variables de type bit

QB supporte la définition des variables comme DEFINT A-D, BASCOM ne l'autorise pas. Pour la clarté du programme, nous déconseillons cette ancienne pratique, il est préférable de dimensionner chaque variable avec DIM. En effet les fautes de frappe sont redoutables dans ce cas (bonjour et bomjour sont deux variables bit, le compilateur les acceptera sans broncher !)

Voir Aussi
DIM

DEFLCDCHAR

Action
Définit un caractère LCD personnalisé.

Syntaxe
DEFLCDCHAR num_de_carac, r1,r2,r3,r4,r5,r6,r7,r8

Remarques
Num_de_carac Constante représentant le numéro du caractère défini (0 à 7)
R1 -- R8 La valeur de la ligne du caractère.
L'outil "LCD Designer", dans le menu TOOLS, est très pratique pour cette construction.
Un CLS doit suivre l'instruction DEFLCDCHAR
Le programme doit suivre le mode de rédaction de l'exemple LCDSTK200.bas

Utilisation de LCD DESIGNER

```
Deflcdchar 1 , 225 , 227 , 226 , 226 , 226 , 242 , 234 , 228      ' Carac perso n°1
Deflcdchar 0 , 240 , 224 , 224 , 255 , 254 , 252 , 248 , 240      ' Carac perso n°0
Cls             'obligatoire
Lcd Chr(0) ; Chr(1) 'envoi les caractères 0 et 1 à l'afficheur
```

```
'----------------- petite routine assembleur ------------
_temp1 = 1          'valeur dans ACC
!rCall _write_lcd   'écrit dans LCD
End
```

Voir Aussi
Programme exemple lcdstk200.bas

DELAY

Action
Suspend le programme pendant un temps très court

Syntaxe
DELAY

Remarques
La durée est d'environ 1000 µs

Voir Aussi
WAIT, WAITMS

DIM

Action
Dimensionne une variable

Syntaxe
DIM var AS [XRAM/SRAM/ERAM] type [AT location] [OVERLAY]

Remarques

Var	un nom de variable comme varentier, j, K1 ou un d'un tableau pression(10)
TYPE	Bit, Byte, Word, Integer, Long, Single, String.
XRAM	Optionnel, la variable sera stockée en mémoire externe
SRAM	Optionnel, la variable sera stockée en mémoire interne
ERAM	Optionnel, la variable sera stockée en mémoire EEPROM
AT location	Optionnel, Emplacement en mémoire
OVERLAY	Recouvrement

Une variable chaîne_de_caractère (String) doit être définie avec sa longueur. Elle occupe en mémoire un Byte de plus que la longueur définie. Exemple Mot as **sting*10** occupe 11 byte.

Utilisation classique de Dim :

Dim Jour(7) As String * 8 , tableau(25) as integer , An As Byte , Flagj As Bit
Dim Lejour As String * 9 , Lemois As String * 10 , Indexjour As Byte,Cosangle As Single Dim

On peut déclarer plusieurs variables sur une seule ligne. Les 2 premières déclarations sont des tableaux de string ou de chiffre.

En QB la déclaration de variable en mode implicite n'est pas obligatoire, à notre avis c'est une erreur, provoquant des bugs difficiles à trouver.
Le basic BASCOM n'autorise pas ce genre de pratique, sauf dans le cas de Defxxx que nous déconseillons.

Les bits ne peuvent être stockés qu'en mémoire SRAM

Le paramètre optionnel AT laisse à l'utilisateur le choix de l'emplacement mémoire où sera stocké la variable. Si l'emplacement mémoire est occupé, la variable occupera le premier emplacement libre suivant.

Overlay précise qu'une variable peut en recouvrir une autre (utiliser la même place mémoire)
L'option OVERLAY n'occupe aucun emplacement mémoire, il crée un pointeur.

Dim x as Long at $60 'long utilise 60,61,62 et 63 hex de la mémoire SRAM
Dim b1 as Byte at $60 OVERLAY
Dim b2 as Byte at $61 OVERLAY
B1 et B2 ne sont pas des variables réelles! Elles pointent vers un emplacement mémoire. Dans cet exemple de &H60 et &H61, en assignant le pointeur B1, on écrira à l'adresse mémoire &H60 qui est utilisée par la variable X.

On peut aussi lire la mémoire B1:
Print B1 : Cela imprimera le contenu de l'adresse mémoire &H60.
En utilisant un pointeur, on peut manipuler individuellement les Bytes constituant une variable.

Un autre exemple:
Dim L as Long at &H60
Dim W as Word at &H62 OVERLAY
W pointe maintenant vers les 2 Bytes supérieurs de la variable long.

Voir Aussi
DEFxxx, CONST, LOCAL

DISABLE /ENABLE

Action
Valide(Enable) ou invalide(desable) une interruption spécifiée.

Syntaxe
Disable *interruption* Enable *interruption*

Remarques

INTERRUPTION	DESCRIPTION
INTO	Interruption externe 0
INT1	Interruption externe 1
OVF0, TIMER0, COUNTER0	Interruption TIMER overflow 0
OVF1, TIMER1, COUNTER1	Interruption TIMER overflow 1
CAPTURE, ICP1	Interruption INPUT CAPTURE TIMER1
COMPARE1A, OC1A	Interruption TIMER1 OUTPUT COMPAREA
COMPARE1B, OC1B	Interruption TIMER1 OUTPUT COMPAREB
SPI	Interruption SPI
URXC	Interruption transmission série RX complète
UDRE	Interruption registre des données série vide
UTXC	Interruption transmission série TX complète
SERIAL	Invalide URXC, UDRE, UTXC
ACI	Interruption Comparateur Analogique
ADC	Interruption convertisseur Analogique/digital
INT2, INT3, etc..	Interruptions dépendantes des µp.

Par défaut, les interruptions sont invalides. Pour invalider ou valider toutes les interruptions,:
DISABLE(ENABLE) INTERRUPTS

Voir Aussi
Programme exemple : SERINT.bas

DISPLAY

Action
Met l'afficheur LCD en fonction ou non

Syntaxe
Display ON/OFF

Remarques
L'afficheur est en service (ON) à la mise sous tension.

Voir Aussi
LCD

DO --- LOOP

Action
Répétition d'un ensemble d'instructions jusqu'à réalisation d'une condition.

Syntaxe
DO
Instructions
LOOP [UNTIL condition]

Remarques
On peut sortir de cette boucle avec l'instruction EXIT DO

La boucle DO--- LOOP est exécutée au moins une fois.

UNTIL n'est pas obligatoire (dans le Basic Bascom), dans ce cas on tourne dans une boucle sans fin (cas de la lecture d'un clavier par exemple)

Voir Aussi
EXIT, WHILE---WEND, FOR---NEXT, Programme exemple Do-LOOP.bas

---

DTMFOUT

Action
Envoi d'un ton DTMF à la broche COMPARE1 de TIMER1

Syntaxe
DTMFOUT nombre, durée
DTMFOUT String, durée

Remarques

Nombre	Une variable ou une constante numérique qui est équivalent du "son" téléphone
Durée	Durée du ton
String	Une variable chaîne_de_caractère qui contient le digit à appeler.

L'instruction DTMF est basée sur la note d'application ATMEL (314).
L'utilisation de TIMER1 pour générer les sons a pour conséquence que TIMER1 ne peut plus être utilisé en mode interruption dans l'application, mais il peut être utilisé pour une autre tâche.
Puisque TIMER1 est utilisé en mode interruption on doit valider les interruptions par ENABLE INTERRUPTS.
La fréquence de travail est entre 4 et 8 MHz.
La sortie DTMF se trouve sur la broche TIMER1 OCA1
Attention au branchement sur la ligne téléphonique DC 48V !

Voir Aussi
Programme exemple DTMFOUT.bas

ECHO

Action
Met l'ECHO ON ou OFF dans l'entrée série par INPUT

Syntaxe
ECHO ON/OFF

Remarques
ON Valide l'ECHO
OFF Invalide l'ECHO
Par défaut l'ECHO est ON
Cette instruction remplace l'instruction NOECHO qui suivait l'instruction INPUT

Voir Aussi
INPUT, programme exemple INPUT.bas

ELSE
Voir IF---THEN

ENABLE
Voir Disable

END

Action
Met fin à l'exécution d'un programme.

Syntaxe
END

Remarques
On peut aussi employer STOP
Avec END, toutes les interruptions sont invalidées et une boucle sans fin est générée.
Avec STOP, seule une boucle sans fin est générée.

Voir Aussi
STOP

EXIT

Action
Pour sortir d'une boucle, d'une SUB ou d'une fonction

Syntaxe

EXIT FOR	boucle For--Next
EXIT DO	boucle Do--Loop
EXIT WHILE	boucle While--Wend
Exit SUB	sortir d'une SUBroutine
Exit Function	Sortir d'une Fonction

Remarques
Avec cette instruction on peut sortir d'une structure n'importe quand.

Voir Aussi
Programme Exit.bas

FIX / INT / ROUND

Action

FIX	Renvoie	Pour des valeurs <0 la valeur entière supérieure. Pour des valeurs >0 la valeur entière inférieure.
INT	Renvoie	La partie entière d'une variable.
ROUND	Renvoie	Pour des valeurs <0 la valeur entière inférieure. Pour des valeurs >0 la valeur entière supérieure.

Syntaxe

```
var = FIX( x )
var = INT( x )
var = ROUND( x )
```

Remarques

Var	Une variable Single qui prend la valeur de FIX(INT, ROUND) de la variable X
X	Une variable Single.

Exemple :

Dim J As Single , K As Single , H As Single

```
J = -10.5
K = -10.5
H = -10.5
H = Fix(h)
K = Int(k)
J = Round(j)
Print "-10.5 fix     int    round"
Print "   " ; H ; "    " ; K ; "     " ; J
' rem      -10       -11       -11
```

```
J = 10.5
K = 10.5
H = 10.5
H = Fix(h)
K = Int(k)
J = Round(j)
Print "10.5 fix     int    round"
Print "   " ; H ; "    " ; K ; "     " ; J
' rem       10       10        11
End
```

FOR---NEXT

Action
Boucle inconditionnelle, exécute un certain nombre d'instructions, un certain nombre de fois.

Syntaxe
FOR var=début TO fin [STEP valeur]
Instruction
Instruction
Instruction
Next

Remarques

Var	la variable-compteur utilisée
Début	la valeur de départ de la variable-compteur.
Fin	la valeur de fin la variable-compteur.
Valeur	La valeur d'incrémentation ou de décrémentation à chaque tour

Pour la décrémentation le pas (STEP) doit être négatif

La structure doit être terminée par NEXT, en cas de structure imbriquée, NEXT sera suivi de la VAR correspondante.

STEP est optionnel si STEP=1

Voir Aussi
Programme For_next.bas

FORMAT

Action
Formatage d'une Variable String numérique

Syntaxe
Cible =Format(source, "Masque")

Remarques

Cible	Variable qui recevra la source formatée
Source	String qui contient le nombre non formaté
Masque	masque de formatage de la source

Quand le masque comporte des espaces et que la source est d'une longueur inférieure, le résultat sera de la longueur du masque :
Masque = " " source = 126, cible = " 126"

Quand un + suit les espaces, un + sera mis devant les chiffres si le chiffre n'est pas négatif : masque = " +", source =126, cible = " +126"

Si des zéros sont mis à la place des espaces, on les retrouvera à la place des espaces : "+0000000", cible = "+00000126"
On peut aussi formater avec un point décimal : "000.00" donne "001.26"
La combinaison des espaces, +, zéros, et point décimal est possible mais doit respecter l'ordre : espaces, +, zéros, point décimal et zéros.

Voir Aussi
FUSING, programme exemple : Format.bas

FOURTHLINE/THIRDLINE

Action
Met le curseur de l'afficheur LCD au départ de la quatrième (troisième) ligne.

Syntaxe
FOURTHLINE
THIRDLINE

Remarques
Seulement pour les afficheurs 4 lignes

Voir Aussi
HOME, UPPERLINE LOWERLINE, LOCATE

FRAC

Action
Retourne la partie fractionnaire d'une variable Single.

Syntaxe
Var =FRAC(X)

Remarques
VAR une Single
X une Single

Voir Aussi
INT

FUSING

Action
Formatage d'une valeur Single en STRING

Syntaxe
Cible = FUSING(source, "Masque")

Remarques
Cible une variable String
Source une variable Single
Masque le masque de formatage.

Le masque doit toujours commencer par #.
#.### retourne un résultat du type 123.456 avec un arrondissement, 123.4567 donne 123.457
Pour ne pas arrondir on doit utiliser & à la place de # après le point décimal :
#.&&& et 123.4567 donne 123.456

Voir Aussi
Format, le programme chaine_de_caractere.bas

GETADC

Action
Retourne la valeur ADC (Analog-digital-converter) pour les broches ADC (0-7)

Syntaxe
Var= GETADC (broche, [offset])

Remarques
Var La variable associée à la conversion (word par exemple)
Broche La broche qui reçoit le signal
Offset Optionnel, une variable numérique qui spécifie le gain ou le mode. Cette option ne fonctionne que sur les nouveaux µP AVR. L'offset sera ajouté au canal désiré et insérré dans le registre ADMUX .

Attention cette fonction n'est disponible que sur certains µp (8535, M8,..)
Les broches sont utilisables en I/O mais il est important de ne pas utiliser les fonctions I/O pendant les conversions AD.

Voir Aussi
CONFIG ADC, programme exemple ADC.bas

GETATKBD

Action
Lecture d'un clavier PCAT

Syntaxe
Var = GETATKBD

Remarques
Var Une String ou un BYTE qui reçoit la valeur de la touche pressée.
Si aucune touche est pressée un 0 est retourné.
L'instruction utilise 2 broches et une table de traduction.

Voir Aussi
CONFIG KEYBOARD les programmes exemples GETATKBD et GETATKBD-int.bas
Ce dernier utilise une interruption.

GETKBD

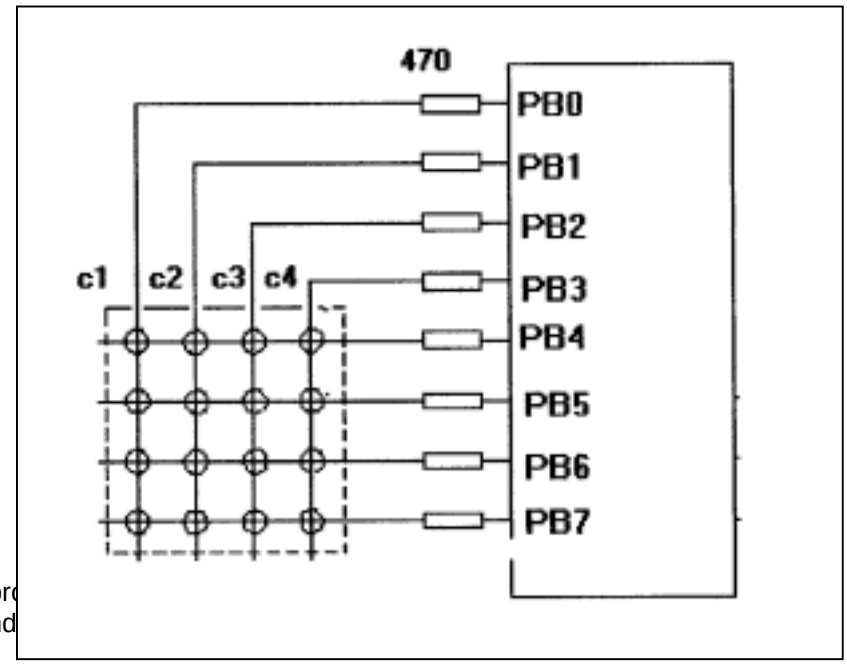
Action
Balaie un clavier matriciel 4 X 4 et renvoie la valeur de la touche pressée.

Syntaxe
Var = GETKBD( )

Remarques
Var La variable qui reçoit la valeur de la touche pressée.

La fonction GETKBD ( ) peut être attachée à un port du µP; celui-ci peut être défini par l'instruction CONFIG KBD

Schéma exemple pour le port B



Les br
Quand

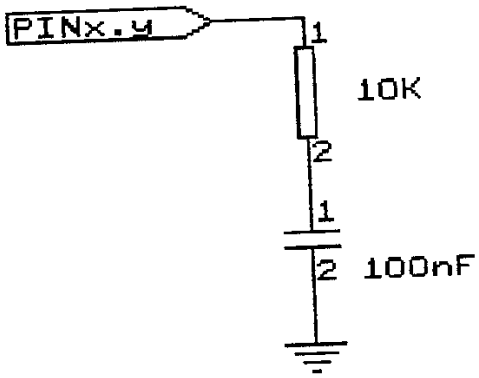
Voir Aussi
CONFIG KBG et Logiciel exemple : GETKBD.bas

GETRC

Action
Donne la valeur d'une résistance ou d'une capacité

Syntaxe
var = GETRC(Broche, nombre)

Remarques
Var La variable WORD qui reçoit la valeur
Broche Le nom de la broche pour la connexion RC
Number Le numéro du port.
Le nom du port d'entrée et son numéro (PIND.3 par exemple) doivent être donnés même si les autres broches sont configurées en sortie.



Le condensateur est chargé, et le temps pris pour qu'il se décharge est mesuré et stocké dans la variable. En faisant varier R ou C les valeurs changent.

La variable ne donne pas la valeur de la résistance ou la capacité.

Voir Aussi
Exemple : GETRC.bas

GETRC5

Action
Renvoie le code RC5 d'un transmetteur IR(Infrarouge)

Syntaxe
GETRC5 (adresse, commande)

Remarques

Adresse L'adresse du composant RC5
Commande La commande RC5
Utilise TIMER0
Voir la note d'application AVR 410
Les télécommandes IR utilisent le code RC5 (un mot de 14bits)

Voir Aussi
Datasheet du SFH506-36 (non fournie) de SIEMENS : le programme exemple RC5.bas

---

GOSUB

Action
Branche le programme vers une subroutine

Syntaxe
Gosub etiquette

End
Etiquette:
Instruction
Instruction
return

Remarques
Etiquette Un nom quelconque d'étiquette de branchement.
Avec Gosub le programme saute à l'étiquette de branchement et exécute les instructions à partir de cette étiquette jusqu'à l'instruction return.
L'étiquette doit se situer après l'instruction fin.
Ne pas confondre une instruction GOSUB avec une SUBroutine.

Voir Aussi
GOTO, CALL, RETURN et le programme exemple GOSUB.bas

GOTO

Action
Saute à une étiquette dans un programme.

Syntaxe
GOTO Etiquette

----
etiquette:

Remarques
Il n'y a pas d'instruction Return
Instruction à éviter, voir le chapitre "boucles et branchement"

Voir Aussi
GOSUB

GREY2BIN

Voir BIN2GREY

HEX

Action
Retourne une représentation hexadécimale "String" d'un nombre

Syntaxe
VarString=Hex(Var)

Remarques
VarString = une String
Var = Byte, Integer, word, long ou Single

Voir Aussi
HEXVAL, programme exemple : conversions.bas

HEXVAL

Action
Convertit une variable String représentant un nombre hexadécimal en variable numérique.

Syntaxe
Var = HEXVAL(String)

Remarques

Var	Une variable numérique
String	La chaîne_de_caractère qui doit être convertie

Voir Aussi
HEX, VAL, STR, BIN… et programme conversions.bas

HIGH/LOW HIGHW

Action
Retrouve le MSB (Byte le plus signifiant) pour HIGH et les LSB(moins signifiant) pour LOW d'une variable.
Retrouve le WORD le plus signifiant d'une variable LONG

Syntaxe
Var = HIGH(s)
Var = LOW(s)
VAR = HIGW(s)

Remarques

VAR	Une variable numérique qui contiendra le MSB, LSB, WORD
S	Une variable numérique qui contient le MSB, LSB, WORD

Voir Aussi

LOW, HIGHW, programme conversions.bas

HIGHW

Voir HIGH

HOME

Action
Place le curseur en position 1 de la ligne spécifiée.

Syntaxe
HOME UPPER
HOME LOWER
HOME THIRD
HOME FOURTH

Remarques
Si HOME est seul le curseur se place en haut à gauche de l'afficheur.

Voir Aussi
LCD, CLS, LOCATE… les programmes LCDxx.bas

I2CRECEIVE

Action
Reçoit des données depuis un composant I2C série.

Syntaxe
I2CRECEIVE esclave, var
I2CRECEIVE esclave, var, B2W, B2R

Remarques
EsclaveUne variable Byte, Word ou Integer ou encore une constante qui possède l'adresse esclave du composant I2C.
Var Une constante Byte, Word ou Integer qui reçoit les informations du composant.
B2W Le nombre de Bytes à écrire. Attention de ne pas le spécifier trop souvent.
B2R Le nombre de Bytes à recevoir; Attention de ne pas le spécifier trop souvent..
Quand une erreur intervient la variable interne ERR renvoie 1.

On peut spécifier l'adresse de base du composant esclave car le bit READ/WRITE est réglé (set/reset) par le software.

Voir Aussi
I2CSEND, I2CSTART, I2CSTOP, I2CRBYTE, I2CWBYTE et les programmes I2C.bas ou clock.bas

---

I2CSEND

Action
Envoie des données à un composant I2C série.

Syntaxe
I2CSEND esclave, var
I2CSEND esclave, var, Bytes

Remarques
Esclaveune variable Byte, Word ou Integer ou encore une constante qui possède l'adresse esclave du composant I2C.
Var Une constante Byte, Word ou Integer qui contient les informations à envoyer au composant.
Bytes Le nombre de Bytes à envoyer

Voir Aussi
I2CRECEIVE, I2CSTART, I2CSTOP, I2CRBYTE, I2CWBYTE et les programmes I2C.bas ou clock.bas

---

I2CSTART, I2CSTOP, I2CRBYTE, I2CWBYTE

Action

I2CSTART	génère une condition de démarrage I2C
I2CSTOP	génère une condition d'arrêt I2C
I2CRBYTE	reçoit un Byte depuis un composant I2C
I2CWBYTE	envoie un Byte à un composant I2C

Syntaxe

I2CSTART
I2CSTOP
I2CRBYTE var, ack, nack
I2CWBYTE val

Remarques

Var	Une variable qui reçoit des données d'un composant I2C
Ack/nack	ACK spécifie qu'il y a encore des Bytes à lire
	NACK spécifie qu'il n'y a plus de Byte à lire.
Val	une variable ou constante à écrire dans le composant I2C.

Ces instructions viennent en complément des instructions I2CRECEIVE et I2CSEND.

Voir Aussi

I2CRECEIVE, I2CSEND et les programmes I2C.bas ou clock.bas

IDLE

Action

Met le µP en mode inactif.

Syntaxe

IDLE

Remarques

Dans ce mode le système d'horloge est suspendu, seules les interruptions série ou timer/compteur fonctionnent.

Ce mode est suspendu aussi par le Watchdog, un niveau de déclenchement (Interruption ADC) ou encore par un RESET par la broche RESET.

Voir Aussi

POWERDOWN

IF---THEN---ELSEIF---ELSE----END IF

Action
SI---Alors----Sinon-SI ---Sinon---fin de test

Branchement basé sur une comparaison booléenne.

Syntaxe
```
If  comparaison1          Then
          ---instructions
          ---instructions
[Elseif  Comparaison2 Then]
          ---instructions
          ---instructions
[Else]
          ---instructions
          ---instructions
Endif
```

Remarques
Comparaison(1 ou 2…) n'importe qu'elles comparaisons booléennes

Les comparateurs ELSEIF et ELSE sont optionnels
Ce test peut être aussi utilisé avec des variables bits et des index.

Exemple :
```
Dim var as Byte, idx as Byte
Var = 255
Idx=1
If var.idx=1 then
          Print "bit 1 est 1"
End if
```

Voir Aussi
Programme exemple : IF-Then.bas

INCR

Voir DECR

INKEY

Action
Renvoie la valeur ASCII du premier caractère du tampon d'entrée série.

Syntaxe
```
Var = INKEY( )
Var = INKEY(#canal )
```

Remarques

Var	Variable Byte, Integer, word, long ou String
#canal	Une constante qui identifie le canal ouvert dans la liaison série Soft.

Si il n'y a pas de caractère en attente un 0 sera renvoyé. A partir de la version 1.11.6.9 La variable **ERR** n'est plus à un 1 quand il n'y a plus de caractère. Utiliser la fonction ISCHARWAITING quand la réception d'un chr(0)

```
If Err = 0 Then  ' quel est le caractère ?
    Print #1 , car       'l'affiche en voie série
End If
```

Cette instruction peut être utilisée si le µP possède une sortie RS232 (UART) les broches RXD (Receive Data) et TXD (Transmit Data) doivent alors être réservées à cet usage.

Voir Aussi
WAITKEY, ISCHARWAITING, les programmes exemples :
Open.bas, RS232inputBufer(1).bas, RS232outputBuffer(1).bas

INP

Action
Renvoie un Byte lu depuis un port hardware ou une mémoire interne ou externe.

Syntaxe
Var = INP(adresse)

Remarques

Var	Variable qui reçoit la valeur lue
Adresse	l'adresse où lire la valeur de : 0 à **&H**FFFF

La fonction PEEK( ) ne lit que les mémoires les plus basses (0-32) où se trouvent les registres.
La fonction INP( ) peut lire n'importe quelle mémoire.

Quand on désire lire la mémoire externe, il faut qu'elle soit configurée dans le menu : "compiler chip option"

Voir Aussi
OUT, PEEK et le programme exemple PEEK.bas

INPUT

Action
Permet de recevoir des données depuis un clavier pendant l'exécution d'un programme.

Syntaxe
INPUT [" Annonce"], var [, varn]

Remarques

Annonce	Une phrase, constante ou String optionnelle envoyée avant de recevoir les caractères.
Var, varn	Une variable qui recevra les données.

L'instruction INPUT peut être utilisée quand on possède un port RS232 sur le µP
Le TERMINAL EMULATOR est très pratique pour cette utilisation.

Voir Aussi
INPUTBIN, INPUTHEX , PRINT , INKEY, WAIKEY, ECHO et les programmes
Input.bas, RS232xxxx.Bas

INPUTBIN

Action
Lit des données binaires depuis un port série

Syntaxe
INPUTBIN var1, [var2]
INPUTBIN #canal, var1, [var2]

Remarques
Var1 la variable qui reçoit la donnée.
Var2 Une autre variable (ou plus) qui reçoit (reçoivent) les autres données
#canal Le canal utilisé pour le port série Soft utilisé, celui-ci doit être ouvert par Open et refermé par Close

Le nombre de Bytes lues dépend de la variable utilisée, une variable Byte, reçoit 1 Byte, une variable Integer, 2 une long 4….un tableau attendra jusqu'à ce que le tableau soit rempli.
Cette instruction attend le nombre de données demandé.

Voir Aussi
PRINTBIN

INPUTHEX

Action
Permet l'entrée de données hexadécimales depuis un clavier pendant l'exécution d'un programme.

Syntaxe
INPUTHEX [" Annonce"] , var [, varn]

Remarques
Annonce Une phrase, constante ou String optionnelle envoyée avant de recevoir les caractères.
Var, varn Une ou des variables qui recevront les données.
L'instruction INPUTHEX peut être utilisée quand le µP possède un port RS232.
Le TERMINAL EMULATOR est très pratique pour les tests de cette instruction.
Les valeurs entrées doivent être comprises entre 0--9 et A à F
Si VAR est un Byte la longueur maximum est 2 caractères
Un Integer, 4 caractères…

Voir Aussi
INPUT, INPUTBIN…

INSTR

Action
Retourne la position d'une (sous)String dans une autre String

Syntaxe
Var= INSTR(Start, String,Sous-str)
Var= INSTR(String,Sous-str)

Remarques
Var Variable numérique qui recevra la position de la Sous-String dans la String.
Start Un paramètre numérique optionnel qui peut recevoir la position à partir de laquelle sera trouvée la sous-String.
String La phrase qui contient
Sous-Str La sous-String

Voir Aussi
Programme Instr.bas

INT

Voir FIX

ISCHARWAITING

Action
Renvoie 1 quand un caractère est dans le tampon UART hard.

Syntaxe
var = ISCHARWAITING()
var = ISCHARWAITING(#canal)

Remarques
Var Variable Byte, Integer, Word or Long.
Canal un nombre ou une constante qui identifie le canal ouvert
Si aucun caractère est en attente un 0 est retourné.
Si un caractère est dans le tampon un 1 est retourné
Le caractère n'est pas envoyé à une variable, ni modifié par cette fonction
L'utilisation de ISCHARWAITING permet de savoir si un caractère est dans le buffer même si celui-ci est un caractère binaire 0 =&B00000000

Voir Aussi
INKEY, WAITKEY et les programmes exemples de la liaison série.

LCASE /UCASE

Action
Conversion de la casse d'une phase(tout en minuscule ou tout en majuscule)

Syntaxe
Cible = LCASE(source) 'minuscule
Cible = UCASE(source) 'majuscule

Remarques
Cible La String qui recevra la valeur modifiée par l'instruction.
Souce La String qui sera modifiée.

LCD

Action
Affiche une constante ou une variable sur un afficheur LCD.

Syntaxe
LCD x

Remarques
X La variable ou constante à afficher
Plus d'une variable peuvent être affichées séparées par un ;

Voir Aussi
Programme LCD et chapitre "Afficheur" dans la première partie.

LEFT / RIGHT

Action
Retourne la partie gauche (droite) d'une String

Syntaxe
Var=LEFT(String,var2)
Var=RIGHT(String,var2)

Remarques
Var une variable String qui reçoit la partie de "String"
String La String qui sera utilisée.
Var2 un nombre ou une variable représentant le nombre de caractères sélectionnés

Voir Aussi
MID, INSTR, programme chaine_de_caracteres.bas en annexe

LEN

Action
Retourne la longueur d'une String.

Syntaxe
Var = LEN(String)

Remarques
Var une variable numérique qui reçoit la longueur de la chaîne String
String une chaîne de caractère.

String ne peut dépasser 254 Bytes

LOAD

Action
Charge une période déterminée pour un timer.

Syntaxe
LOAD Timer, Var

Remarques
Timer Timer0, Timer1, Timer2
Var La variable contenant la valeur à prendre en compte, c'est le complément de la valeur maximum (256, 65536)

Timer 1 ne possède pas de mode de rechargement, mais on peut le charger avec l'instruction LOAD :
LOAD TIMER0, 10 Charge le timer avec 256-10=246 et donne un dépassement de timer après 10 "top"

LOADADR

Action
Charge une paire de registres avec l'adresse d'une variable.

Voir Aussi
Voir l'aide en ligne, cette instruction très orientée "assembleur" dépasse le cadre de cet ouvrage.

LOADLABEL

Action
Donne à une variable Word l'adresse d'une étiquette.

Syntaxe
Var = LOADLABEL(label)

Remarques

Var	La variable qui reçoit l'adresse de l'étiquette
Etiquette	Le nom de l'étiquette

Dans certains cas on peut avoir besoin de l'adresse d'un point pour faire un PEEK( ) par exemple.
On place une étiquette à ce point et on utilise LOADLABEL pour connaître l'adresse de cette étiquette.

Voir Aussi
PEEK

LOCAL

Action
Pour déclarer et dimensionner des variables dans un sous-programme (SUB-Function)

Syntaxe
LOCAL var AS TYPE

Remarques

Var	Le nom de la variable
As TYPE	Le type de donnée.

Il ne peut pas y avoir de LOCAL BIT ou TABLEAU
Une variable LOCAL est une variable temporaire qui est stockée dans la FRAME. Dés que le programme sort de la SUB, cette variable est retirée de la FRAME.

L'extension de dimensionnement vers le type de mémoire n'est pas possible :
AT, ERAM, SRAM, XRAM

Voir Aussi
DIM, le programme exemple : Declare.bas

LOCATE

Action
Déplace le curseur de l'afficheur LCD

Syntaxe
LOCATE Y, X

Remarques

Y	Constante ou variable avec le N° de LIGNE (1, 2, 3, 4)*
X	Constante ou variable avec le N° de LIGNE(1 à 64)*
*	Suivant l'afficheur

Voir Aussi
CONFIG LCD, LCD, SHIFTCURSOR, HOME, UPPERLINE…

LOOKDOWN

Action
Renvoie l'index d'une valeur appartenant à une série de données (DATA)

Syntaxe
Var=LOOKDOWN(Valeur, Etiquette, Nombre)

Remarques

Var	La valeur de l'index
Valeur	La valeur à rechercher
Etiquette	L'étiquette des données
Nombre	Le nombre de données qui doivent être recherchées

Quand on recherche une valeur dans une série de BYTE la variable Valeur doit être dimensionnée en Byte. Quand on recherche une valeur dans une série d'INTEGER ou WORD la variable Valeur doit être dimensionnée en INTEGER,

LOOKDOWN est l'instruction complémentaire de LOOKUP

LOOKDOWN cherche la donnée d'une valeur et retourne l'index quand la valeur est trouvée. Sinon LOOKDOWN retourne -1

Voir Aussi
LOOKUP et le programme exemple LOOKDOWN.bas

LOOKUP / LOOKUPSTR

Action
Renvoie une valeur contenue dans une table de nombres (LOOKUP) ou de String (LOOKUPSTR)

Syntaxe
Var=LOOKUP(index, etiquette)

Remarques

Var	la valeur renvoyée
Index	La valeur de l'emplacement (mini 0 max 65535)

Exemples:

```
Dim B1 As Byte , I As Integer
B1 = Lookup(2 , Dta)
Print B1                                ' envoi 3 (base zéro )
I = Lookup(0 , Dta2)                    ' print 1000
Print I
End

Dta:
Data 1 , 2 , 3 , 4 , 5
Dta2:
Data 1000% , 2000%
```

```
Dim S As String * 4 , Idx As Byte
Idx = 0 : S = Lookupstr(idx , SData)
Print S                                 'print 'This'
End

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

Voir Aussi
LOOKDOWN, LOOKUPSTR

LOW

Voir HIGH

LOWERLINE / THIRDLINE / FOURLINE / UPPERLINE

Action
Place le curseur en position 1 de la ligne spécifiée.

Syntaxe
UPPERLINE
LOWERLINE
THIRDLINE
FOURTHLINE

Voir Aussi
LCD, CLS, LOCATE… les programmes LCDxx.bas

LTRIM / TRIM / RTRIM

Action
Supprime les blancs, à gauche (LTRIM) ou à droite (RTRIM) ou de chaque côté (TRIM) d'une String

Syntaxe
Nouveaumot= LTRIM(mot)
Nouveaumot= RTRIM(mot)
Nouveaumot= TRIM(mot)

Remarques

Nouveaumot	Variable String cible
Mot	Variable String source

Exemple

```
Dim S As String * 6
S = "  AB "
Print Ltrim(s)
Print Rtrim(s)
Print Trim(s)
End
```

MAKEBCD / MAKEDEC

Action
Convertit une variable décimale en valeur BCD MAKEBCD ou
Convertit une variable BCD en variable décimale

Syntaxe
Varcible=MAKEBCD(Varsource)
Varcible=MAKEDEC(varsource)

Remarques
Utilisé par les composants I2C qui stockent les valeurs au format BCD.
Pour imprimer les valeurs BCD utiliser l'instruction BCD( ) qui convertit un nombre BCD en String.
Exemples :

```
Dim A As Byte
A = 65
Lcd A
Lcd Bcd(a)
A = Makebcd(a)
LCD "   " ; a
End
```

```
Dim A As Byte
a = 65
Print A
Print Bcd(a)
A = Makedec(a)
Print Spc(3) ; A
End
```

MAKEINT

Action
Réunit deux Bytes pour donner une INTEGER ou une WORD

Syntaxe
Var=(VarLSB, varMSB)

Remarques
Var La variable cible dimensionnée en INTEGER ou WORD
VarLSB Variable ou constante Byte contenant le Byte le moins signifiant
VarMSB Variable ou constante Byte contenant le Byte le plus signifiant

Le code équivalent est Var =(256*varMSB)+LSB

Voir Aussi
LOW HIGH

MAKEDEC

Voir MAKE BCD

MAX()

Action
Retourne la valeur maximum d'un tableau de byte ou word (seulement)

Syntaxe
var1 = MAX(tab2())
MAX(ar(1), m ,idx)

Remarques
var1 Variable qui prendra la valeur maximum.
tab2() Le tableau.
La fonction MAX peut retourner l'index de la valeur maximale
Ar(1) Élément de départ pour obtenir l'index et le maximum.
M La valeur maximale du tableau.
Idx L'index de la valeur maximale, égale 0 si il n'y a pas de maximale.

Voir Aussi
MIN()

MID

Action
La fonction MID renvoie une partie d'une variable String
L'instruction MID remplace une partie d'une variable String par une autre String.

Syntaxe
Var=MID(Varmot, départ, [quantité])
MID(var,départ, [quantité])=varmot

Remarques
Var String cible Z dans l'exemple de gauche
Varmot String source ABCDEFG transformé en A12DEFG dans le second exemple
Départ la lettre de départ en partant de la gauche
Quantité La quantité de lettres à prendre

Exemples:

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

```
Dim S As String * 15 , Z As String * 15
S = "ABCDEFG"
Z = "12345"
Mid(s , 2 , 2) = Z
Print S                          'A12DEFG
```

MIN()

Action
Retourne la valeur minimale d'un tableau de byte ou word (seulement) voir MAX()

Nbits()

Action
Inverse de Bits() met les bits sélectionnés à 0 (voir Bits()

ON INTERRUPT

Action
Exécute un sous-programme quand arrive l'interruption spécifiée.

Syntaxe
On INTERRUPT ETIQUETTE [NOSAVE]

Remarques
Interrupt INT0 INT1 INT2 INT3 INT4 INT5 TIMER0 TIMER1 TIMER2 ADC EEPROM CAPTURE1 COMPARE1A COMPARE1B COMPARE1.
On peut aussi utiliser la syntaxe AVR:
OC2 OVF2 ICP1 OC1A OC1B OVF1 OVF0 SPI URXC UDRE UTXE ADCC ERDY ACI

ETIQUETTE l'étiquette où commence le sous programme d'interruption.

NOSAVE Quand on spécifie NOSAVE les registres ne sont ni sauvés ni restaurés au retour donc il faut sauvegarder et restaurer les registres utilisés (voir explications dans l'aide en ligne)

Le sous-programme doit se terminer par RETURN

Voir Aussi
DISABLE/ENABLE, L'aide de BASCOM et programme exemple : INTO.bas

---

ON VALUE

Action
Branchement vers une ou plusieurs étiquettes dépendantes de la valeur de la variable.

Syntaxe
ON Var [goto] [gosub] etiquette1, etiquette2 [,CHECK]

Remarques
Var La valeur de la variable à tester peut être un registre comme PORTB
Etiquette1, etiquette2 Les étiquettes où se feront les sauts suivant la valeur de Var.
La valeur démarre à 0 donc le premier branchement se fait à 0

CHECK Un contrôle optionnel pour le nombre d'étiquettes de branchements. Le compilateur comparera la valeur de la variable par rapport au nombre d'étiquettes. S'il n'y a pas assez d'étiquettes, le programme continuera à la ligne suivante.

Exemple :

```
Dim X As Byte
X = 1                                   ‘donne une valeur d’interruption
On X Gosub Lbl1 ,  Lbl3                 'va à l’étiquette lbl3
X = 0
On X Goto Lbl1 ,  Lbl3
END
lbl3:
  Print "lbl3"
Return

Lbl1:
Print "lbl1"
return
```

OPEN

Action
Ouvre un composant

Syntaxe
Open "device" for MODE as #canal

Remarques
Device Le composant par défaut est com1, et il n'est pas nécessaire de l'ouvrir pour utiliser INPUT/OUTPUT pour celui-ci.
Avec l'implémentation d'un UART soft le compilateur doit savoir quelle broche est utilisée pour l'entrée (INPUT) et pour la sortie (OUTPUT)
Exemple COMB.0:9600,8,N,2
Utilisera le port B, broche 0, à 9600 baud, sur 8 bits, sans parité et avec 2 bits de stop. Un paramètre optionnel [,INVERTED] peut être utilisé pour spécifier une RS232 inversée.
MODE Peut être BINARY ou RANDOM pour COM1 mais pour l'UART soft doit être INPUT ou OUTPUT
Canal Doit être un nombre >0

Un composant accepte les instructions PRINT, INPUT, INPUTHEX, INPUTBIN, INKEY et WAITKEY

Chaque composant doit être fermé par l'instruction CLOSE #n
L'instruction INPUT en combinaison avec l'UART soft ne renvoie pas de caractère ECHO parce qu'il n'y a pas de broche associée pour le retour.
(dans le cas de l'UART hard TDX et RDX)

Voir Aussi
Crystal, programme exemple Open.bas

OUT

Action
Envoie un Byte vers un port hard ou une mémoire interne ou externe.

Syntaxe
OUT Adresse, Val

Remarques
Adresse L'adresse où envoyer le Byte entre 0 et **&H**FFFF
Val Valeur du Byte
OUT peut écrire dans n'importe quel emplacement mémoire.
Il est conseillé d'utiliser des variables WORD car des INTEGER peuvent être négatives. Dans ce cas le MSB est à 1.
Pour écrire en XRAM, la mémoire externe doit être validée dans "COMPILER CHIP OPTIONS.

Exemple :
```
Out &H8000 , 1          'send  1 to the Databus(d0-d7) at hex address 8000
```

Voir Aussi
INP, PEEK et le programme exemple PEEK.bas

PEEK

Action
Renvoie le contenu d'un registre.

Syntaxe
Var=PEEK(adresse)

Remarques

Var	Variable numérique qui reçoit le contenu du registre à l'adresse "Adresse"
adresse	Adresse du registre de 0 à 31

Peek lit le contenu d'un registre.
Out lit n'importe quel emplacement mémoire.

Voir Aussi
POKE, CPEEK, INP, OUT et le programme exemple PEEK.bas

POKE

Action
Ecrit un Byte dans un registre interne.

Syntaxe
POKE Adresse, Valeur

Remarques

Adresse	variable numérique de l'adresse du registre 0 à 31
Valeur	Valeur de 0 à 255

Cette instruction doit être maniée avec la plus grande prudence.

Voir Aussi
PEEK, CPEEK, INP, OUT et le programme exemple PEEK.bas

POPALL / PUSHALL

Action
Instruction de restauration et de sauvegarde des registres internes.

Syntaxe
POPALL (restauration)
PUSHALL (sauvegarde)

Remarques
A utiliser quand on a écrit et fait un mélange Basic et assembleur.

POWERDOWN

Action
Met le µP en mode arrêt.

Syntaxe
POWERDOWN

Remarques
Dans ce mode l'oscillateur externe est arrêté, le WATCHDOG ou un reset externe ou un niveau déclenchant (interruption ADC) peut être utilisé pour le relancer.

Voir Aussi
IDLE, POWERSAVE

POWERSAVE

Action
Met le µP en mode sommeil.

Syntaxe
POWERSAFE

Remarques
Seulement sur le 8535

Voir Aussi
IDLE, POWERDOWN

PRINT

Action
Envoie des données à la sortie RS232

Syntaxe
PRINT Var ; ["constant"]

Remarques

Var	La variable ou constante à envoyer

On peut utiliser un point-virgule pour séparer des données différentes.

Pour visualiser l'effet de PRINT on peut utiliser le TERMINAL EMULATOR

Voir Aussi
INPUT, OPEN, CLOSE, SPC et le logiciel exemple PRINT.bas

PRINTBIN

Action
Envoie le contenu binaire d'une variable

Syntaxe
PRINTBIN var [;Var2]
PRINTBIN [#canal,] var [;Var2]

Remarques

Var	la variable dont la valeur sera envoyée au port série
VAR2	variable supplémentaire optionnelle

PRINTBIN est équivalent à PRINT CHR(var);

Exemple : Quand on utilise une Variable LONG, 4 Bytes sont envoyées.

Voir Aussi
INPUTBIN

PSET

Action
Set (1) ou reset (0) un pixel d'un afficheur LCD graphique.

Syntaxe
PSET X, Y Valeur

Remarques

X	emplacement X de 0 à 239*
Y	emplacement Y de 0 à 63 *
Valeur	La valeur du pixel

*Xn et Yn dépendent de l'afficheur
PSET est utilisable pour créer un oscilloscope !

Voir Aussi
SHOWPIC, CONFIG GRAPHLCD, programme exemple : T6963_240_128.bas, T6963_V3.bas

---

PULSEIN

Action
Renvoie un nombre compté entre deux fronts montants ou descendants sur une entrée.

Syntaxe
PULSEIN var, PINX, PIN, état

Remarques

Var	Une variable WORD qui reçoit le nombre compté
PINX	Un registre d'entrée comme PIND
PIN	la broche qui reçoit l'impulsion
Etat	1 ou 0, 1 pour un front montant (0 à 1) 0 pour un front descendant

ERR sera positionnée à 1 en cas de "timeout" (dépassement de temps)
Soit 65535 comptages. = avec des unités de 10µs : 655.35 ms
On peut utiliser BITWAIT pour le démarrage mais cela risque de créer une boucle sans fin.
PULSEIN attend le front spécifié pour démarrer et pour s'arrêter. On ne se sert pas de TIMERx.

Voir Aussi
PULSEOUT

PULSEOUT

Action
Génère une impulsion d'une durée réglable sur la broche d'un port.

Syntaxe
PULSEOUT PORT, BROCHE, DUREE

Remarques

Port	le nom du port (PORTB par exemple)
Broche	la broche du port (0à 7) qui doit être configurée en Output.
Duree	La durée de l'impulsion, elle est en µs quand un quartz de 4MHz est utilisé.

Exemple :
```
Dim A As Byte
Config Portb = Output                    'PORTB all output pins
Portb = 0                                'toutes les broches à 0
For A = 0 To 7
  Pulseout Portb , A , 60000               ' génère l'impulsion
  Waitms 250                               'Attente
Next
```

Voir Aussi
PULSEIN

PUSHALL

Voir POPALL

RC5SEND

Action
Pilote un port infrarouge au code RC5, RC6

Syntaxe
RC5send Togbit , Address , Command

Remarques

Togglebit	construit le bit d'inversion 0 ou 32
Address	The RC5 adresse
Command	The RC5 commande.

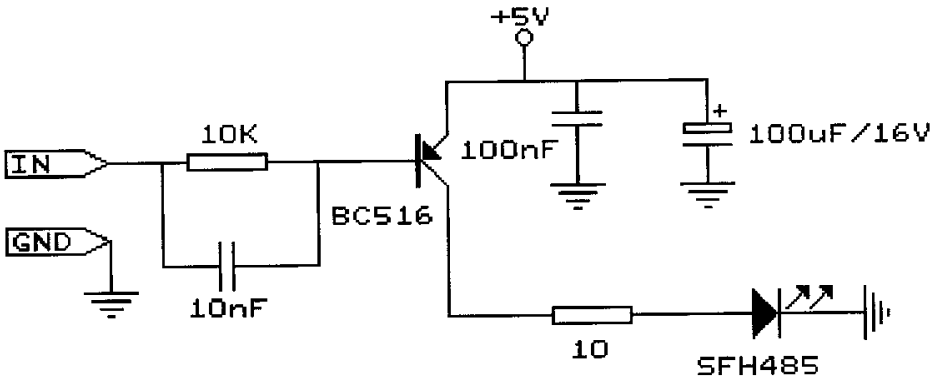
La résistance doit être connectée à la broche OC1A (exemple portB.3 pour un 2313)
Contrôler l'adresse de ce port dans la datasheet du µP.
Beaucoup d'appareils audio ou vidéo sont équipés avec une télécommande IR
Le code RC5 est un signal bi-phase composé d'un Word de 14 bits.
Le code RC6 est un Word de 16 bits.
Les 2 premiers bits sont les bits de start. Ils sont toujours à 1.
Le bit suivant est un bit de contrôle ou Toggle bit, qui est inversé à chaque fois q'un bouton est appuyé sur la télécommande.
Les 5 bits suivants sont les bits d'adresse du récepteur.

D'habitude la TV à l'adresse 0, Le magnétoscope l'adresse 5 etc..
La séquence de commande permet 64 combinaisons pour RC5 et 256 pour RC6

Les bits sont transmis en code bi-phase aussi appelé "Manchester code"

Nous vous invitons vivement à chercher des informations sur les codes RC5 et RC6 (sites PHILIPS, …)

Un exemple de circuit de télécommande :



Voir Aussi

CONFIG RC5, GETRC5, programme :SENDRC5.bas SENDRC6.bas

RC6SEND

Voir RC5SEND

READ / RESTORE

Action

Instruction de lecture des Data

Syntaxe

Restore etiquette	Pour positionner le pointeur où se trouvent les Data
Read var	Pour lire les Data

Remarques

Etiquette	Etiquette à partir de laquelle on lit les Data
Var	Variable du même type que la Data

Il est préférable (bien que non obligatoire) d'écrire les Data après l'instruction END.
Il faut savoir que RESTORE et READ ne fonctionnent pas avec les données stockées en EEPROM.
La directive de compilation $EEPROM permet seulement de créer une image mémoire de l'EEPROM
Pour retrouver des données en EEPROM on doit utiliser une variable ERAM

Voir Aussi

DATA, LOOKUP, Programme ReadData.bas

READEEPROM

Action
Lit une donnée en EEPROM et la place dans une variable.

Syntaxe
READEEPROM var, adresse

Remarques

Var	Le nom de la variable qui reçoit la variable
Adresse	L'adresse en EEPROM où la variable doit être lue

Cette instruction existe dans un but de compatibilité avec BASCOM -8051
On peut aussi utiliser :

```
Dim V as Eram Byte     'stocke en  EEPROM
Dim B As Byte          'variable normale
B = 10
V = B                        'met la variable in EEPROM
B = V                        'lit depuis EEPROM
```

Quand on utilise l'instruction READEEPROM les types doivent être identiques.
Ne pas utiliser l'adresse 0 qui peut être effacée pendant un reset.

ERAM peut être indexé :
Dim ar(10) as Eram Byte
Avec l'instruction READEEPROM, on peut omettre l'adresse si on ne travaille pas dans une boucle:

```
Readeeprom B , Label1
Print B
 Do
  Readeeprom B
  Print B
Loop Until B = 5
```

Ce qui précède ne fonctionne pas

```
Readeeprom B , Label1 ' etiquette précisée
Readeeprom B ' lit l'adresse suivante en EEPROM
Readeeprom B ' lit l'adresse suivante en EEPROM
```

Voir Aussi
$EEPROM dans "programmation avancée" et les programmes exemples :
EEPROM et EEPROM2.bas

READMAGCARD

Action
Lecture de données depuis une carte magnétique

Syntaxe
READMAGCARD var, Count, 5/7

Remarques

Var	Un tableau de Bytes qui reçoit les données
Count	Une variable Byte qui renvoie le nombre de Bytes lues
5/7	Une constante numérique qui spécifie le codage utilisé (5 ou 7 bits)

Il y a 3 pistes sur une carte magnétique.
La piste 1 stocke les données sur 7 bits incluant un bit de parité, elle est utilisée pour stocker une Data alphanumérique.
Sur les pistes 2 et 3 les Data sont stockés sous 5 bits (ISO7811-2)

Voir Aussi
Le programme exemple : MAGCARD.bas

REM

Action
Signale au compilateur de ne pas prendre en compte ce qui suit, c'est un commentaire.

Syntaxe

REM	Début du commentaire
'	Début du commentaire
'(	Début d'une partie de programme mise en commentaire
')	Fin d'une partie de programme mise en commentaire

Remarques
On peut mettre une partie du programme en REM pour débugger.
Ne pas oublier de "nettoyer" ensuite.

RESET / SET

Action
Met un bit à 1 (Set) ou le met à 0 (reset)

Syntaxe
Reset bit
Set bit
Reset Var.x
Set Var.x
Reset composant

Remarques

Bit	une variable Bit
Var.x	un port(var) et une broche(x) ou une variable et un poids (de 0 à 7) pour une variable Byte, 0 à 15 pour une Integer/Word ou 0 à 31 pour une long/Single
composant	Le timer watchdog par exemple.

```
Dim b1 as bit, b2 as Byte, I as Integer
Reset Portb.                    'reset bit 3 du port B
Reset                           'bitvariable
Reset B2.0                   'reset bit 0 de la variable Byte b2
Reset I.                   'reset MSB bit de I
Set Portb.1                     'set bit 1 du  port B
Reset Watchdog                  'reset le watchdog
```

RESTORE

Voir Read

RETURN

Action
Retourne au programme ou sous-programme appelant une sous-routine

Syntaxe
RETURN

Remarques
Sub et sous-routine d'interruption doivent être terminées par Return

Voir Aussi
GOSUB ON INTERRUPT

RIGHT

Voir LEFT

RND

Action
Renvoie un nombre aléatoire

Syntaxe
Var = RND (limit)

Remarques

Var	La variable qui reçoit le nombre aléatoire.
Limit	une variable Word qui limite le nombre aléatoire.

C'est une variable soft qui est créée, à chaque redémarrage, la même séquence est reproduite.

ROTATE

Action
Fait "tourner" les bits d'une variable.

Syntaxe
Rotate Var, LEFT/RIGHT [,Shift]

Remarques

Var	Une variable Byte, Integer/Word ou Long
Shift	Une variable qui précise le nombre de déplacement (1 par défaut)

Les bits sont préservés, si on désire supprimer des bits il faut utiliser SHIFT

Voir Aussi
SHIFT, SHIFTIN, SHIFTOUT programme exemple : ROTATE.bas

ROUND

Voir FIX

RTRIM

Voir LTRIM

SELECT-CASE-END-SELECT

Action

Exécute une ou plusieurs instructions suivant la valeur d'une expression ou d'une variable.

Syntaxe

```
SELECT CASE var
        Case test1
                …instruction
        [Case Test2
                …instruction
                …instruction]
        Case Else
                …instruction
END SELECT
```

Remarques

Var	Variable à tester
Test1	forme du test

Test peut avoir comme forme:
CASE IS >2
CASE 5
CASE 3 TO 9

Exemple :

```
Dim X As Byte

Do
  Input "X ? " , X
  Select Case X
    Case 1 To 3 : Print "1 , 2 or 3 will be ok"
    Case 4 : Print "4"
    Case Is > 10 : Print ">10"
    Case Else : Print "no"
  End Select
Loop

End
```

Voir Aussi

IF THEN

SHIFT

Action
Déplace les bits à droite ou à gauche d'un poids

Syntaxe
SHIFT Var, LEFT/RIGHT [,nombre]

Remarques
Var Variable Byte, Integer/word ou Long
nombre Le nombre de déplacements.
Quand on déplace vers la gauche, le bit le plus signifiant sera éliminé et le moins signifiant devient le second bit etc.. A droite l'inverse se réalise…

Voir Aussi
ROTATE, SHIFTIN, SHIFTOUT.

SHIFTCURSOR

Action
Déplace le curseur d'un afficheur LCD par une position.

Syntaxe
SHIFTCURSOR LEFT/RIGHT

Voir Aussi
SHIFTLCD, LCD…

SHIFTIN /SHIFTOUT

Action
Déplace un flux de bits d'un port dans une variable (SHIFTIN)
Déplace un flux de bits d'une variable vers une broche d'un port (SHIFTOUT)

Syntaxe
SHIFTIN broche, Pclock, var, option[,bits,delay]
SHIFTOUT broche, Pclock, var, option[,bits,delay]

Remarques
Broche La broche qui est utilisée comme entrée (SHIFTIN) ou en sortie (SHIFTOUT)
Pclock La broche qui génère l'horloge
Var La variable qui est concernée

OPTION :

0 ou 4	MSB se déplace en premier quand le bit d'horloge descend
1 ou 5	MSB se déplace en premier quand le bit d'horloge monte
2 ou 6	LSB se déplace en premier quand le bit d'horloge descend
3 ou 7	LSB se déplace en premier quand le bit d'horloge monte

En ajoutant 4 au paramètre OPTION on indique que c'est une horloge externe qui est utilisée. (Pclock)
Bits Nombre de bits à déplacer. Maxi 255 (Optionnel)
Delay Délai optionnel, en µs. Si ce délai est donné, le nombre de bits doit aussi être donné. Il représente en général 2 tours d'horloge

Si le nombre de bits n'est pas indiqué, le déplacement dépend du type de variable
8 bits pour un Byte, 16 pour un Integer

Voir Aussi
Le programme exemple : Shift.bas

SHIFTLCD

Action
Déplace le texte affiché à droite ou à gauche d'une position.

Syntaxe
SHIFTLCD LEFT/RIGHT

Voir Aussi
SHIFTCURSOR

SHOWPIC

Action
Montre un fichier BGF sur l'écran graphique.

Syntaxe
SHOWPIC X,Y,Etiquette

Remarques
X , Y Les coordonnées qui doivent être 0 ou un multiple de 8. La hauteur et la largeur de l'image doivent aussi être un multiple de 8.
Etiquette Pointeur vers le fichier graphique donné par la directive $BGF.

SHOWPIC peut afficher un fichier BMP qui doit être converti au préalable en fichier BGF par l'outil "tools graphic converter"

Voir Aussi
"Programmation avancée" pour la directive de compilation $BGF.
PSET, CONFIG GRAPHLCD et les logiciels exemples : T6963c xxx.bas

---

SOUND

Action
Envoie des impulsions à une broche d'un port.

Syntaxe
Sound Broche, Duree, frequence

Remarques
Broche broche d'un port en sortie
Duree Durée de l'envoi
Frequence Fréquence

Exemple :
Sound Portb.1, 10000, 10

SONYSEND

Action
Envoi les codes de télécommande IR type Sony®

Syntaxe
SONYSEND adresse [, bits]

Remarques
Adresse L'adresse du système Sony.
bits Paramètre optionnel, quand il est utilisé il doit être soit, 12,15 ou 20 .Si on utilise cette option, la variable Addresse doit être du type Long

Utilise TIMER1 !

Voir Aussi
CONFIG RC5 , GETRC5

Exemple :
des exemples de schéma et de code sont donnés dans l'aide de Bascom.

SPACE

Action
Donne à une variable String la valeur de x espaces

Syntaxe
Var =SPACE(x)

Remarques
Var une variable String
X Le nombre d'espaces (chr(32)) demandé, si 0 est donné il y aura 255 caractères.

Exemples :

```
Dim s as String * 15
s = Space(5)
Print " {"  ;s ; " }"  REM  imprime    '{    }
End
```

```
Dim A as Byte
A = 3
S = Space(a)
End
```

Voir Aussi
SPC

SPC

Action
Imprime le nombre d'espaces spécifiés

Syntaxe
Print SPC(x)

Remarques
X Le nombre d'espaces (chr(32)) demandé, si 0 est donné il y aura 255 caractères
SPC peut être utilisé aussi avec LCD

la fonction SPACE utilise une variable String, SPC n'en utilise pas.
SPC ne peut être utilisé que pour imprimer par PRINT ou LCD

Voir Aussi
SPACE

SPIIN / SPIOUT

Action
Pour lire (SPIIN) ou envoyer (SPIOUT) des données sur le Bus SPI

Syntaxe
SPIIN var , Bytes
SPIOUT var , Bytes

Remarques

Var	La variable qui reçoit les données lues (SPIIN) ou à envoyer(SPIOUT) au bus SPI
Bytes	Le nombre de Bytes à lire(SPIIN) ou à envoyer (SPIOUT)

Voir Aussi
CONFIG SPI, SPIINIT, SPIMOVE et les logiciels exemples : sendspi.bas, spi.bas, spi-slave.bas, spisoftslave.bas

SPIINIT

Action
Initialise le bus SPI.

Syntaxe
SPIINIT

Remarques
Après la configuration des broches SPI (CONFIG SPI), on doit les initialiser pour les régler dans la bonne direction.
Quand les broches ne servent qu'au bus SPI, on ne doit utiliser SPIINIT qu'une seule fois, sinon on doit réinitialiser avant chaque appel de SPIIN ou SPIOUT.

Voir Aussi
CONFIG SPI, SPIIN, SPIMOVE et les logiciels exemples : sendspi.bas, spi.bas, spi-slave.bas, spisoftslave.bas

---

SPIMOVE

Action
Envoie et reçoit une valeur ou une variable du bus SPI

Syntaxe
Var= SPIMOVE (Byte)

Remarques

Var	Variable qui reçoit les Bytes du bus SPI
Bytes	la variable ou constante dont le contenu doit être envoyé au bus SPI.

Voir Aussi
CONFIG SPI, SPIIN, SPIINIT et les logiciels exemples : sendspi.bas, spi.bas, spi-slave.bas, spisoftslave.bas

---

SPIOUT

Voir SPIIN

START

Action
Lance un composant

Syntaxe
START composant

Remarques
Composant TIMER0, TIMER1, COUNTER0, COUNTER1, WATCHDOG AC(comparateur analogique) ou ADC (Convertisseur Analogique digital)

On doit appliquer cette instruction à un composant dans le cas où une interruption interviendrait (quand l'entrée extérieure est désactivée)

TIMER0 et COUNTER0 sont des composants identiques.
Dans le cas de AC et de ADC, START alimente les composants pour que ceux-ci fonctionnent.

Voir Aussi
STOP et le programme exemple: ADC.bas

STCHECK

Action
Instruction de débogage, elle appelle une routine qui vérifie l'état des dépassements de piles (STACK OVERFLOW)

Syntaxe
STCHECK

Remarques
Voir utilisation de la mémoire, aide en ligne et programme stack.bas

---

STOP

Action
Arrête un composant
Arrête le programme

Syntaxe

STOP	Composant
STOP	

Remarques
L'instruction STOP seule, arrête le programme sans désactiver les interruptions à la différence de END.
STOP composant arrête le composant et suspend l'alimentation de celui-ci dans le cas des convertisseurs analogiques ou des comparateurs.

Voir Aussi
START et le programme exemple ADC.bas

STR

Action
Renvoie une représentation String d'un nombre

Syntaxe
Var= Str(nombre)

Remarques

Var	Une variable String suffisamment dimensionnée pour contenir le nombre
Nombre	Un nombre, une constante ou variable numérique.

Voir Aussi
VAL, HEX, HEXVAL …Le programme de conversion conversions.bas

STRING

Action
Renvoie une String composée de n fois le caractère ASCII m

Syntaxe
Var = STRING(n,m)

Remarques

Var	Une variable String suffisamment dimensionnée pour contenir le nombre n
n	Un nombre entre 1 et 254
m	le code ASCII du caractère

Voir Aussi
SPACE, la table de caractères ASCII en annexe.

SUB

Voir DECLARE SUB

SWAP

Action
Echange deux variables du même type.

Syntaxe
Swap var1,var2

Remarques

var1, var2	2 variables numériques ou String

Pas de variables tableaux.
Après un SWAP, Var1 garde la valeur précédente de Var2 tandis que Var2 garde celle de Var1.

THIRDLINE

Voir FOURTHLINE

TIME$

Voir DATE$

TOGGLE

Action
Change l'état d'une broche en sortie ou d'une variable bit.

Syntaxe
TOGGLE varbit

Remarques
Varbit Une broche d'un port PORTB.6 par exemple.
Le PORTB.6 doit avoir été configuré en sortie avant d'utiliser TOGGLE.

TOGGLE change simplement l'état d'un port : si le port est 1, TOGGLE le met à 0 si le port est à 0, TOGGLE le met à 1.

Voir Aussi
CONFIG PORT

TRIM

Voir LTRIM

UCASE

Voir LCASE

UPPERLINE

Voir LOWERLINE

VAL

Action
Convertit une variable String représentant une valeur numérique en nombre.

Syntaxe
Var = Val(s)

Remarques
Var Une variable numérique dont le type est suffisant pour contenir la valeur de la String.
S Une variable String ne contenant que des chiffres.

Exemple
```
Dim J as Integer, mot as String*4
Mot="1234"
J=val(mot)
Print J                 '1234
End
```

VARPTR

Action
Retrouve l'adresse mémoire d'une variable

Syntaxe
Var = VARPTR(var2)

Remarques
Var La variable qui reçoit l'adresse de var2
Var2 La variable dont on veut connaître l'adresse.

Version ()

Action
Retourne une String avec la date et l'heure de la dernière compilation.

Syntaxe
Var = Version(frm)

Remarques
Var une string qui reçoit une constante. Cette version est donné en MM-DD-YY hh:nn:ss
ou en format européen DD-MM-YY hh:nn:ss.

WAIT / WAITMS / WAITUS

Action
Suspend le déroulement du programme pour une durée de nnn secondes / milli -seconde /µ-seconde.

Syntaxe
WAIT nn
WAITMS nn
WAITUS nn c'est un U pas µ

Remarques
Nn Nombre ou une constante mais pas une variable (dans le cas WAITUS)

Les durées sont approximatives. Les interruptions ralentissent ces valeurs.

Voir Aussi
DELAY

WAITMS

Voir WAIT

WAITKEY

Action
Attend jusqu'à ce qu'un caractère soit présent dans le tampon série

Syntaxe
Var = WAITKEY ( )
Var = WAITKEY (#canal)

Remarques

Var	La variable qui reçoit le caractère ASCII
#canal	Le canal utilisé par l'UART soft

Voir Aussi
INKEY, INPUT et les programmes-exemples de la liaison série.

WHILE-WEND

Action
Exécute une série d'instructions dans une boucle TANT QUE une condition donnée est vraie.

Syntaxe
WHILE condition
…instructions
…instructions
WEND

Remarques
A la différence de la boucle "DO…LOOP UNTIL condition", la condition est testée AVANT exécution des instructions incluses. Donc si la condition est fausse AVANT le WHILE, la boucle ne sera jamais exécutée.

Voir Aussi
EXIT, DO…LOOP, FOR.. NEXT et le programme exemple WHILE_W.bas

---

WRITEEEPROM

Action
Ecrit le contenu d'une variable dans les données en EEPROM

Syntaxe
WRITEEEPROM var, adresse

Remarques

Var	Le nom de la variable qui doit être stockée.
Adresse	L'adresse où sera stockée la variable, il peut s'agir d'une étiquette voir l'exemple.

Si la variable a été configurée comme appartenant à l'EEPROM :
DIM var as ERAM Byte, on peut se passer de WRITEEEPROM qui est une instruction BASCOM 8051.
Les interruptions sont invalidés pendant l'écriture en EEPROM et rétablis ensuite.

Voir Aussi
READEEPROM et le programme exemple.EEPROM2.bas

LES FONCTIONS MATHEMATIQUES

Au chapitre "Qu'est ce qu'une variable", il est dit que les variables sont des variables entières, en fait les Singles peuvent aussi recevoir et calculer des valeurs décimales et aussi être utilisées pour les calculs en virgule flottante et les opérations mathématiques. Pour ce faire le compilateur BASCOM utilise l'une des 4 librairies ci-dessous en fonction des instructions demandées.

FP_trig.lib De Josef Franz Vögel
MCS.lib La librairie par défaut
SQR.lib
SQR_it.lib

Si aucune de ces fonctions n'est utilisée, il est conseillé de forcer le compilateur à utiliser des librairies plus simples :

MCSBYTE.lib pour Bytes uniquement
MCSBYTEINT.lib Pour Bytes, Integer et Word uniquement.

Les fonctions mathématiques utilisent toutes des SINGLES.

Toutes les fonctions trigonométriques sont en radians. On peut utiliser DEG2RAD et RAD2DEG pour convertir.

Programme exemple TEST_FPTRIG2.BAS et FP_for_French.bas

ACOS

Action
Renvoie l'arc-cosinus d'une Single en radians.

Syntaxe
var = ACOS(x)

Remarques

Var	Une variable Single qui reçoit la valeur de ACOS de la variable X
X	La variable Single dont on veut connaître la valeur ACOS

X doit être compris entre –1 et +1

ASIN

Action
Renvoie l'arc-sinus d'une Single en radians.

Syntaxe
var = ASIN(x)

Remarques

Var	Une variable Single qui reçoit la valeur de ASIN de la variable X
X	La variable Single dont on veut connaître la valeur ASIN

ATN

Action
Renvoie l'arc-tangente d'une Single en radians.

Syntaxe
var = ATN(X)

Remarques

VAR	Une variable Single qui reçoit la valeur de ATN de la variable X
X	La variable Single dont on veut connaître la valeur ATN

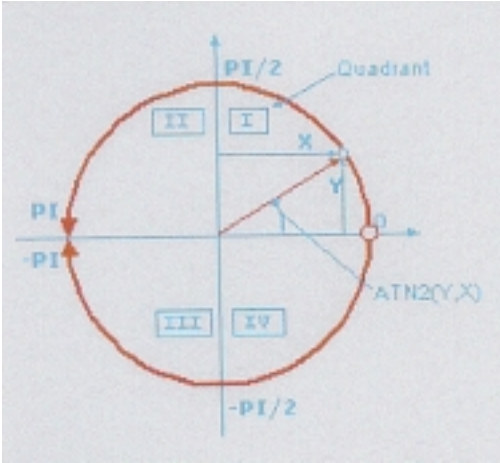
ATN2

Action
ATN2 est une fonction arc-tangente à 4 quadrants.
Tandis que ATN renvoie de -π/2 à π/2, l'ATN2 renvoie la valeur pour l'ensemble du cercle trigonométrique de -π à +π, le résultat dépend du ratio Y/X et des signes de X et Y.

Syntaxe
var = ATN2(x, y)

Remarques

VAR	Une variable Single qui reçoit la valeur de ATN2
X	La variable Single avec la distance dans la direction X
Y	La variable Single avec la distance dans la direction Y



Quadrant	Signe Y	Signe X	ATN2
I	+	+	0 to π/2
II	+	-	π/2 to π
III	-	-	-π/2 to -π
IV	-	+	0 to –π/2

On obtient le même résultat avec le rapport X/Y dans ATN pour X >0 (la droite du cercle) qu'avec ATN2. ATN2 utilise X et Y et peut donner des résultats pour des points dépassant 360°

COS

Action
Renvoie le cosinus d'une Single en radians.

Syntaxe
var = COS(x)

Remarques

Var	Une variable Single qui reçoit la valeur de COS de la variable X
X	La variable Single dont on veut connaître la valeur COS

COSH

Action
Renvoie le cosinus hyperbolique d'une Single en radians.

Syntaxe
var = COSH(x)

Remarques

Var	Une variable Single qui reçoit la valeur de COSH de la variable X
X	La variable Single dont on veut connaître la valeur COSH

DEG2RAD / RAD2DEG

Action
Convertit une variable de degrés en radians(DEG2RAD) ou inversement(RAD2DEG)

Syntaxe
Var=DEG2RAD(x)
Var=RAD2DEG(x)

Remarques

Var	Une variable Single qui reçoit la valeur de DEG2RAD / RAD2DEG de la variable X
X	La variable Single en radians dont on veut connaître la valeur en degrés

---

EXP

Action

Renvoie e (la base des logarithmes naturels) de la puissance d'une variable Single.

Syntaxe

Var=EXP(x)

Remarques

Var	Une variable Single qui reçoit la valeur de EXP de la variable X
X	La variable Single dont on veut connaître la valeur EXP

LOG/ LOG10

Action

Retourne le logarithme naturel (décimal) d'une variable SINGLE

Syntaxe

Cible =LOG(var)
Cible =LOG10(var)

Remarques

Cible	une variable Single
Var	une variable Single

Log et LOG10 utilisent des variables Single générées par l'instruction:
_SNGTMP1 et _SNGTMP4, ces variables peuvent être réutilisées par l'application.
Cette fonction peut prendre beaucoup de temps pour se réaliser, surtout quand les nombres sont grands. La précision diminue avec la grandeur des nombres.

POWER

Action
Renvoie la puissance d'un nombre

Syntaxe
Var = POWER(X, PUISSANCE)

Remarques

Var	Une Single qui reçoit le nombre X élevé à la puissance PUISSANCE
X	Nombre quelconque qui est élevé à la puissance
Puissance	Puissance à laquelle il faut élever le nombre

Par rapport à ^ ne fonctionne pas en nombre négatif, mais utilise moins de ressources.

SIN

Action
Renvoie le sinus d'une Single en radians.

Syntaxe
var = SIN(x)

Remarques

Var	Une variable Single qui reçoit la valeur de SIN de la variable X
X	La variable Single dont on veut connaître la valeur SIN

SINH

Action
Renvoie le sinus hyperbolique d'une Single en radians.

Syntaxe
var = SINH(x)

Remarques

Var	Une variable Single qui reçoit la valeur de SINH de la variable X
X	La variable Single dont on veut connaître la valeur SINH

SQR

Action
Renvoie la racine carrée d'un nombre

Syntaxe
var = SQR(x)

Remarques

Var	Une variable Single ou Double qui reçoit la racine carrée de la variable X
X	La variable Single ou Double dont on veut connaître la racine carrée.

Quand SQR est utilisé avec une Single, la directive $LIB = "FP_TRIG.LBX" doit être retenue.
Quand SQR est utilisé avec une Byte ou une Integer ou Word ou Long, MCS.LBX doit être retenue (par défaut). Quatre autres librairies peuvent être utilisées SQR_IT.LBX ou SQR.LBX Double.lbx et Double_trig.dbx ces 2 dernières pour les calculs sur Double et double avec trigo.

TAN

Action
Renvoie la tangente d'une Single en radians.

Syntaxe
var = TAN(x)

Remarques

Var	Une variable Single qui reçoit la valeur de TAN de la variable X
X	La variable Single dont on veut connaître la valeur TAN

TANH

Action
Renvoie la tangente hyperbolique d'une Single en radians.

Syntaxe
var = TANH(x)

Remarques

Var	Une variable Single qui reçoit la valeur de TANH de la variable X
X	La variable Single dont on veut connaître la valeur TANH
