Dictionnaire Bascom

NOTES PRELIMINAIRES

Les grandes différences avec la version précédente sont :

- ajout des nouvelles instructions, une bonne cinquantaine quand même.
- Utilisation systématique de lien hypertexte pour démarrer les exemples, il est donc impératif que ce fichier Dictionnaire soit dans le répertoire BASCOM pour que les liens fonctionnent.
- Séparations des chapitres très spécifiques TCP/IP, I2C etc..
- Cette version en .Doc permet à l'utilisateur de corriger lui-même le dictionnaire, en revanche nous apprécierions de recevoir les corrections proposées.
- Il est vivement conseillé d'être connecté à Internet pour suivre certains liens.
- Les instructions redondantes ou inutiles « $init » par exemple ne sont pas expliquées.
- Cette version est encore incomplète, il manque, entre autre, la gestion des fichiers sur cartes mémoire…

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)

A la différence du dictionnaire BASCOM-AVR nous avons éclaté le notre en plusieurs parties :

⇨ Les directives de compilation.
⇨ Les instructions de CONFIG.
⇨ Les instructions et fonctions proprement dites,
⇨ Les fonctions mathématiques.
⇨ les explications et Les instructions concernant les bus 1 WIRE, I2C, SPI

Dans le chapitre « Introduction au Basic des microcontrôleurs », des tableaux par ordre alphabétique reprennent les fonctions les plus utilisées, conversions de variables, manipulation de chaînes de caractères…

Les fichiers exemples (nnnn.bas) n'ont pas été recopiés, ils sont tous dans le répertoire Samples et en général liés par un lien hypertexte. De petits exemples aidant à la compréhension, ont néanmoins été ajoutés à certaines instructions.

Dans cette version, nous avons laissé beaucoup de termes en anglais, en cas de difficultés, il y a des dictionnaires en lignes http://www.google.fr/language_tools?hl=fr et surtout l'indispensable http://jargonf.org/wiki/Accueil il est malheureusement impossible de traduire Socket par Chaussette ! CR « carriage return » par retour-chariot ou LF « line feed » par passer une ligne, le lecteur doit, se familiariser avec la langue de Charles Babbage et de Alan Tuning. Sans oublier George Boole les précurseurs. Néanmoins je n'oublie pas Bouchon, Pascal etc…

Pour mémoire, nous avons, dans une vie antérieure, et par respect pour nos anciens, appeler une machine « Pascaline » ce n'était pas un calculateur mais une machine à enregistrer l'activité des levures par vérification de la cinétique de production de CO^2 cette machine a rencontré un certain succès pour contrôler les bulles du champagne !

Les Directives de compilations

Les directives de compilations sont des repères pour le compilateur. Elles sont prioritaires sur les réglages effectués avec l'onglet « Options » de l'IDE. Il est conseillé d'utiliser ces directives plutôt que l'onglet « Options » car quand on perd la partie .CFG du programme (créé par le compilateur) on perd les réglages.
On les écrira en tête de programme.

1 $ASM

1.1 Action

Début d'un bloc en assembleur

1.2 Syntaxe

$ASM

1.3 Remarques

Utiliser $ASM avec l'instruction $END ASM pour insérer un bloc d'assembleur dans votre code Basic. Vous pouvez aussi précéder chaque ligne avec le signe !
La plupart des mnémoniques assembleurs peuvent être utilisés sans le le signe !

Voir aussi le chapitre "Mélanger BASIC and Assembleur » et les mnémoniques assembleur. Dans la documentation anglaise.

1.4 Exemple:

```
Dim C As Byte
Loadadr C , X 'load address of variable C into register X
$asm
Ldi R24,1 ; load register R24 with the constant 1
St X,R24 ; store 1 into variable c
$end Asm
Print C
End
```

---

2 $BAUD

2.1 Action

Demande au compilateur de réécrire le baud rate régler dans l'option menu.

2.2 Syntaxe

$BAUD = var

2.3 Remarques

Var le baud rate qui sera utilisé. Ceci doit être une constante numérique.
Le baud rate peut être choisi dans le réglage du compilateur. Il est écrit dans un fichier de configuration. Dans le rapport on peut voir quel baud rate est utilisé.
Il dépend aussi du quartz et du microprocesseur
le simulateur ne tient pas compte des éventuelles erreurs de choix. Un mauvais réglage peut donner des résultats faux. Pour de bons résultats il est préférable d'utiliser un quartz dont la fréquence est un multiple du baud rate.

2.4 Voir aussi :

$CRYSTAL , BAUD

---

3 $BAUD1

3.1 Action

Demande au compilateur de régler le baud rate pour le second UART hardware.

3.2 Syntaxe

$BAUD1 = var

Remarques
Var Le baud rate qui sera utilisé. Cela doit est une constante numérique.
Certains microcontrôleurs AVR ont 2 UART. Par exemple le Mega161, Mega162, Mega103 et le Mega128. Certains en ont même 4 UARTS.

3.3 Voir aussi

$CRYSTAL , BAUD , $BAUD

---

4 $BGF

4.1 Action

Inclut un fichier graphique BASCOM

4.2 Syntaxe

$BGF "fichier"

4.3 Remarques

Fichier : est le nom du fichier BGF à inclure.
Utiliser SHOWPIC pour montrer le fichier B G F. $BGF permet d'enregistrer l'image compressée au format graphique BASCOM (BGF).

Voir l'exemple :
showpicE.bas

---

5 $BOOT

5.1 Action

Signale au compilateur d'inclure un support pour un boot loader.

5.2 Syntaxe

$BOOT = Adresse

5.3 Remarques

Adresse l'adresse du boot loader.
Quelques nouveaux microprocesseurs AVR ont une section spéciale pour le boot loader dans la mémoire flash.
La taille de la section de boot est réglé par fuse bits.
La taille de la section de boot détermine aussi l'adresse du boot loader.
Le code de boot doit toujours être situé à la fin de votre programme.

5.4 Voir aussi

$LOADER , $loader est beaucoup plus simple.

---

6 $CRYSTAL

6.1 Action

Signale au compilateur de réécrire la fréquence du quartz.

6.2 Syntaxe

$CRYSTAL = var

6.3 Remarque

Var Une constante numérique avec la fréquence du quartz.

On peut aussi choisir la fréquence du quartz dans les options du compilateur.
Il est préférable de signaler la fréquence du quartz dans le programme ceci ne rend plus visible, en revanche il faut aussi régler la partie fuse bit.
La fréquence du quartz joue sur le baud rate et sur les temps d'attente comme WAITMS

Les nouveaux microprocesseurs AVR utilisent l'oscillateur interne par défaut, il faut donc impérativement contrôler les fuse bits.
Contrôler la data-sheet pour les valeurs par défaut

6.4 Voir aussi

$BAUD , BAUD , CONFIG CLOCKDIV

Exemple :
```
$regfile = "m48def.dat"
$crystal = 4000000
$baud = 19200
Config Com1 = Dummy , Synchrone = 0 , Parity = None , Stopbits = 1 ,
Databits = 8 , Clockpol = 0
Print "Hello world"
End
```

---

7 $DATA

7.1 Action

Signale au compilateur d'enregistrer les data après les lignes suivant la directive $DATA dans la mémoire flash.

7.2 Syntaxe

$DATA

7.3 Remarques

Par construction les AVR possèdent une EEPROM. Avec les instructions WRITEEEPROM et READEEPROM on peut lire et écrire l'EEPROM.
Pour charger des informations dans l'EEPROM, vous pouvez ajouter des lignes DATA dans votre programme. Un nouveau fichier est généré avec l'extension EEP ce fichier peut être utilisé pour programmer l'EEPROM.
Le compilateur doit connaître quelles data doivent être mises en mise en flash est mises en EEPROM. Deux directives de compilations sont ajoutées $EEPROM and $DATA.

$EEPROM dit au compilateur que les lignes data suivantes doivent être ajoutées à l'EEPROM est stockée dans le fichier EEP.
Pour retourner au comportement par défaut on doit utiliser la directive $DATA.

L'instruction READ qui est utilisée pour lire les data ne fonctionne qu'avec les data stockées dans la flash. Elle ne fonctionne pas avec les data stockées dans l'EEPROM.

💣 Ne pas confondre la directive $DATA avec l'instruction DATA.

7.4 Voir aussi

$EEPROM , READEEPROM, WRITEEEPROM , DATA

7.5 Exemple

```
Dim B As Byte
Readeeprom B , 0 'maintenant B = 1
End

Dta:
$eeprom
Data 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8
$data
```

---

8 $DBG

8.1 Action

Permet le débogage à partir de la liaison UART.

8.2 Syntaxe

$DBG

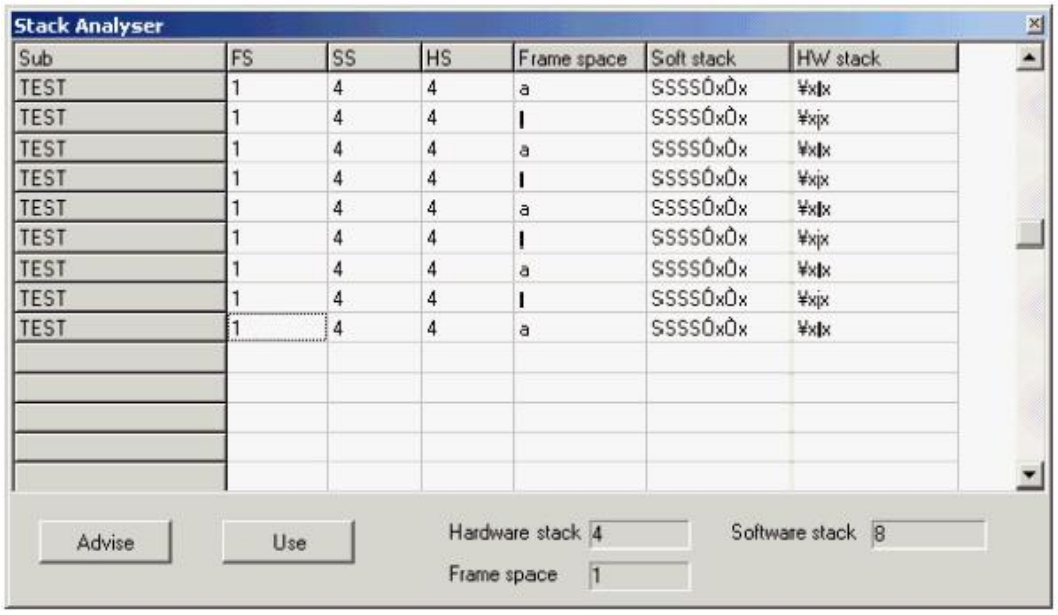
8.3 Remarques

Calculer l'espace hardware software et le cadre (Frame) peut-être une tâche difficile.
Avec cette instruction le compilateur insérera des caractères pour ces différents espaces.
Des F seront écrits pour les cadres. Par exemple un cadre de 4 on aura F. F. F. F.
Pour les espaces hardware des. H. seront écrits. Pour le software se seront des S.
A chaque fois que l'on utilise un espace où on réécrit ces lettres, il doit toujours en rester un peu.
Mais il est inutile d'en définir 100 si 10 sont nécessaires.

- Définir 40 cadres, 20 soft, et 50 hardware.
- Ajouter $DBG au début du programme
- Ajouter une instruction DBG à chaque sousroutine ou fonction

· ouvrir l'émulateur terminal et ouvrir un nouveau fichier, par défaut il aura le nom de votre programme avec l'extension..log
Lancer votre programme et noter les informations
Quand votre programme à exécuter toutes les routines
Arrêter d'enregistrement du fichier et arrêter le programme.
Choisir l'outil analyseur de piles.
Une fenêtre vous montrera les données du fichier .
En pressant le bouton Advise en détermine l'espace nécessaire.

- Appuyez sur le bouton USE pour utiliser le réglage indiqué.



sub	Nom des sousroutines et fonctions
FS	Espace Cadre
SS	Pile soft
HS	Pile hard

L'espace cadre est utilisé pour enregistrer les variables temporaires et locales, il enregistre aussi les variables qui sont passées aux sousroutines et fonctions passées par valeur(Byval).
Parce que les conversions PRINT, INPUT requièrent un buffer le compilateur utilise toujours 24 octets d'espace cadre.
Quand la proposition (advise) est utilisé 2 octets d'espace cadre sont ajoutés. Le réglage sera 24+2=26.

8.4 Voir aussi

DBG

---

9 $DEFAULT

9.1 Action

Règle à la valeur par défaut la dimension du type des variables.

9.2 Syntaxe

$DEFAULT = var

9.3 Remarque

Var SRAM, XRAM, ERAM

Chaque variable sera dimensionnée et stockée dans le type de mémoire indiqué par la directive de compilations.
SRAM = Mémoire interne (volatile)

XRAM = mémoire externe
ERAM = Mémoire EEPROM du microcontrôleur (non volatile)

---

10 $EEPLEAVE

10.1 Action

Signale au compilateur de ne pas recréer ou effacer le fichier EEP

10.2 Syntaxe

$EEPLEAVE

10.3 Remarque

On peut aussi utiliser le fuse bit du programmateur. Cette instruction est pratique en fin de mise au point.

---

11 $EEPROM

11.1 Action

Signale au compilateur d'enregistrer les data dans les lignes DATA suivant la directive de compilations $DATA dans le fichier in EEP.

11.2 Syntaxe

$EEPROM

11.3 Remarques

Les AVR auront une partie de mémoire en EEPROM. Avec les instructions WRITEEEPROM et READEEPROM on peut écrire et lire dans cette EEPROM.
Pour enregistrer des informations dans l'EEPROM, on peut ajouter des lignes de données dans le programme. Un fichier séparé est généré avec l'extension EEPROM.
On doit on doit enregistrer ce fichier séparément dans le microcontrôleur.
Le compilateur doit savoir quelles data doive être dirigé dans la mémoire de code (flash) et celles qui seront dirigées vers l'EEPROM.
En conséquence deux directives ont été ajoutées :
$EEPROM and $DATA.
$EEPROM signale au compilateur que les lignes data suivantes doivent être stocké dans le fichier EEP .
Pour revenir au comportement par défaut on doit utiliser la directive $DATA

L'instruction DATA qui est utilisée pour lire les données ne peut être utilisée que pour les données enregistrées dans le programme. Elle ne fonctionne pas avec les données enregistrées en EEPROM

Ne pas confondre la directive $DATA avec l'instruction DATA.

11.4 See also

$DATA , READEEPROM , WRITEEEPROM , DATA, EEPROMHEX

---

12 $EEPROMHEX

12.1 Action

Indique au compilateur d'enregistrer les données dans le fichier EEP file au format Intel HEX

12.2 syntaxe

$EEPROMHEX

---

13 $EXTERNAL

13.1 Action

Indique au compilateur d'inclure une routine assembleur à partir d'une librairie

13.2 syntaxe

$EXTERNAL Myroutine [, myroutine2]

13.3 remarque

On peut placer une routine assembleur dans un fichier librairie avec cette directive on signale au compilateur de l'inclure dans le programme.

13.4 Voir aussi

$LIB

---

14 $FRAMESIZE, $SWSTACK , $HWSTACK

14.1 Action

Ajuste la taille de l'espace cadre, de la pile soft, de la pile hard

14.2 Syntaxe

$FRAMESIZE = var

14.3 Remarques

Var une valeur numérique décimale.

Bien qu'on peut configurer la taille de ces espaces dans l'option compiler/chip, il est préférable de l'introduire dans le code.
Ces directives sont prioritaires sur les options de l'IDE.
Il est important que ces directives apparaissent dans votre programme, en revanche, elles ne sont pas nécessaires dans un fichier $include

---

15 $INC

15.1 Action

Inclure un fichier binaire dans le programme à la position courante.

15.2 Syntaxe

$INC etiquette, taille | pas de taille, "fichier"

15.3 Remarques

etiquette Nom de l'étiquette utilisée pour se référer aux données.
Taille C'est un repère pour connaître le nombre d'octets que l'on pourra retrouver.
Fichier C'est le nom du fichier à inclure.
Utiliser RESTORE pour mettre un pointeur sur les données, et utiliser READ, pour lire dans les données
l'instruction $ est une alternative pour l'instruction DATA.
Tandis que DATA fonctionne très bien pour les petites quantités de données c'est plus difficile pour les grands fichiers.

15.4 Voir aussi

RESTORE , DATA , READ, $include

---

16 $INCLUDE

16.1 Action

Introduit un fichier ASCII dans le programme à la position courante.

16.2 Syntaxe

$INCLUDE "fichier"

16.3 Remarques

« Fichier » et le nom du fichier ASCII, il doit contenir des instructions BASCOM valides cette option peut être utilisée si vous utilisez souvent les mêmes routines.

Cette directive ne fonctionne qu'avec des fichiers ASCII.
Utiliser $INC quand vous voulez inclure des fichiers binaires.
Voir aussi
$INC

---

17 $INITMICRO

17.1 Action

Pour appeler une routine utilisateur au démarrage qui réalise d'importantes initialisations .

17.2 Syntaxe

$INITMICRO

17.3 Remarques

Cette directive appellera une étiquette appelée _INIT_MICRO tout de suite après les plus importantes initialisations sont réalisées. Vous pouvez placer cette étiquette dans le programme, ou vous pouvez la placer dans une librairie. La librairie présente l'avantage d'être identique pour tous les programmes. Il faut terminer cette directive par RETURN puisqu'une étiquette est appelée on attend un retour. Cette directive peut être utilisée pour régler la direction des ports. La meilleure solution reste les résistances pull up/pull down.

17.4 Exemple

```
$regfile = "m48def.dat"
$crystal = 4000000
$baud = 19200
$initmicro

Print Portb
Do
 nop
Loop
End
'do not write a complete application in this routine.
'only perform needed init functions
_init_micro:
Config Portb = Output
Portb = 3
Return
```

---

18 $LCD, $LCDPUTCTRL, $LCDPUTDATA, $LCDRS

Ces quatre instructions concerne l'utilisation des afficheurs LCD relié au bus Data. Voir explication dans la documentation anglaise. L'utilisation des instructions config LCD est beaucoup plus simple.

---

19 $LCDVFO

19.1 Action

signale compilateur de générer une très courte impulsion Enable pour les afficheurs VFO.

19.2 Syntaxe

$LCDVFO

19.3 Remarque

Noritake est le plus grand constructeur de ses afficheurs.

Cette directive $LCDVFO doit être utilisée avec certaines routines LCD.

---

20 $LIB

20.1 Action

Indique au compilateur les librairies utilisées.

20.2 Syntaxe

$LIB "libname1" [, "libname2"]

20.3 Remarques

Libname1 est le nom de la librairie qui contient les routines assembleur utilisées par le programme. D'autres noms peuvent être utilisés en les séparant par des virgules.
Les librairies concernées seront recherchées quand vous spécifiez les routines à utiliser avec la directive $EXTERNAL. L'ordre de recherche est le même que l'ordre des noms.
La MCS.LBX sera recherchée en dernier, elle est toujours incluse, donc on n'a pas besoin de la spécifier.
Créer sa propre librairie : une librairie est un simple fichier ASCII elle peut être créée avec un éditeur comme BASCOM Notepad ou autres.

---

21 $LOADER

21.1 Action

Indique au compilateur de créer un boot loader à l'adresse spécifique

21.2 Syntaxe

$LOADER = adresse

21.3 Remarques

adresse : L'adresse où est située le boot loader, on peut trouver ces adresses dans la data sheet.
La plupart des microcontrôleurs AVR ont une section de boot. Normalement le microcontrôleur démarre à l'adresse 0 quand il est reseter. Ceci est aussi appelé le vecteur reset.
Les microcontrôleurs qui ont une section de boot séparent la flash mémoire en deux parties, en sélectionnant un fuse bit le programme démarre au vecteur de boot à la place du vecteur reset. Certains microcontrôleurs ont des fuse bits pour choisir la taille du boot loader.

L'exemple du boot loader MCS utilise le port série. Il utilise le protocole X-modem checksum pour recevoir les données. La plupart des émulateurs peuvent envoyer X-modem checksum. L'exemple donné supporte tous les microcontrôleurs avec une section de boot.
Comment procéder ?

- identifier la directive $regfile pour votre microcontrôleur.
- Rendre actif les lignes indiquées par la constante qui est utilisée pour la compilation conditionnelle.
- mettre en REM les autres $regfile et CONST
- compiler le fichier
- ajuster les fuse bits pour diriger le vecteur vers le boot loader
- Régler le fuse bit pour que la taille soit de 1024 octets.
- Choisir le MCS Boot loader programmateur.

Le boot loader est écrit pour travailler avec un baud rate de 57600. Ceci fonctionne pour la plupart des microcontrôleurs qui utilise l'oscillateur interne. Quand on utilise un quartz on peut aller plus vite.
Voir les cartes de programmation pour plus d'informations.

21.4 Voir aussi

$BOOT. $LOADER permet décrire le boot loader en Basic.

---

22 $LOADERSIZE

22.1 Action

Indique au compilateur qu'un boot loader est utilisé et qu'il ne faut pas réécrire sur cet espace.

22.2 Syntaxe

$LOADERSIZE = size

---

23 $MAP

23.1 Action

Générera des étiquettes d'information dans le rapport

23.2 Syntaxe

$MAP

---

24 $NOCOMPILE

24.1 Action

Indique au compilateur de ne pas compiler le fichier.

24.2 Syntaxe

$NOCOMPILE

24.3 Remarques

Cette directive peut paraître étrange, elle est utilisée pour les fichiers « Include »
Ceci permet d'éviter au compilateur de retourner des erreurs qui n'existent pas.
Le fichier sera compiler que lorsque le fichier qui « include » le fichier comportant cette directive sera compilé.

---

25 $NORAMCLEAR

25.1 Action

Indique au compilateur de ne pas réinitialiser la RAM

25.2 Syntaxe

$NORAMCLEAR

25.3 Remarque

Normalement la mémoire RAM est réinitialisée à l'initialisation du code.
Avec cette directive cela permet d'utiliser une batterie back-up et ainsi de garder des données.

---

26 $PROG

26.1 Action

Directive pour programmer automatiquement les Fuse bits.

---

27 $REGFILE

27.1 Action

Signale au compilateur d'utiliser le fichier spécifique des instructions correspondantes au microcontrôleur.

27.2 Syntaxe

$REGFILE = "nom"

27.3 Remarques

Nom : les fichiers des microcontrôleurs sont stockés dans le répertoire BASCOM-AVR et ils ont l'extension DAT.
Ce choix est stocké dans un dans un fichier ayant l'extension CFG
Cette directive doit être la première instruction dans votre programme. Elle ne doit pas être écrite dans un fichier à inclure.
Ce fichier permet aussi d'utiliser le PinOut viewer ou le PDF viewer. (onglet View ou Vue)

27.4 Exemple :

$REGFILE = "8515DEF.DAT"

---

28 $SERIALINPUT, $SERIALINPUT1

28.1 Action

Indique l'entrée série qui doit être utilisée.

28.2 Syntaxe

$SERIALINPUT = étiquette

28.3 Remarques

Etiquette le nom de la routine assembleur qui doit être appelée quand Input attend un caractère. (UDR)
ASM : Le caractère est mis dans R24.
Avec la redirection de la commande INPUT on peut utiliser ses propres routines.
Ainsi on peut utiliser d'autres modes comme mode d'entrée.
L'instruction INPUT est terminée quand un code « retour chariot » code (13) est reçu.
Par défaut quand on utilise INPUT ou INKEY(), le compilateur attende des données depuis le port COM.
À utiliser quand on veut utiliser le clavier ou une autre commande comme mode d'entrée. Voir l'exemple dans la documentation anglaise.

---

29 $SERIALINPUT2LCD

29.1 Action

Cette directive redirige les entrées série vers l'afficheur LCD display à la place de l'écho du port série.

---

30 $SERIALOUTPUT, $SERIALOUTPUT1

30.1 Action

Indique que la sortie série doit être dirigée.

---

31 $SIM

31.1 Action

Signale au compilateur de ne pas prendre en compte les temps d'attente WAIT, WAITMS pendant la simulation.

31.2 Syntaxe

$SIM

31.3 Remarque

Ne pas oublier de supprimer cette ligne en fin de mise au point.

---

32 $TIMEOUT

32.1 Action

Autorise un « timeout » (temps maximum d'attente) pour UART0 and UART1 (Hardware seulement)

32.2 Syntaxe

$TIMEOUT = valeur

32.3 Remarques

Valeur : une constante qui utilise un LONG, indique le temps maximum à attendre.

Toutes les instructions et fonctions RS-232 série (sauf INKEY) qui utilise le HW UART, arrête le programme au moment ou un caractère est reçu. Ce n'est pas le cas quand les fonctions sont bufferisées, la réception se fait alors en tâche de fond.
$TIMEOUT est une alternative pour des réceptions simples ; elle ne doit pas être utilisée avec un buffer.
La valeur n'est pas un nombre de secondes ou de µsecondes. C'est un nombre relatif fonction de l'oscillateur.

😐 $TIMEOUT, pour des µcontrôleur qui ont 2 UART (EX : Mega128), est active pour les 2 UART.

32.4 Exemple :

```
Dim Sname As String * 10
Dim B As Byte
Do
      $timeout = 1000000
      Input "Name : " , Sname
      Print "Hello " ; Sname
      $timeout = 5000000
      Input "Name : " , Sname
      Print "Hello " ; Sname
Loop
```

---

33 $TINY

33.1 Action

Cette directive ne concerne que les µcontrôleurs qui n'ont pas de ERAM , le compilateur ne configure pas les piles. Par exemple le Tiny 11.

33.2 Syntaxe

$TINY

33.3 Remarques

Quand on veut développer ces circuits en assembleur on doit utiliser cette directive.

💣 Bascom n'est pas optimisé pour ce genre de circuit.

```
'name : tiny15.bas
$regfile = "at15def.dat" ' specify the used micro
$crystal = 1000000 ' used crystal frequency
$tiny
$noramclear
Dim A As Iram Byte
Dim B As Iram Byte
A = 100 : B = 5
A = A + B
```

---

34 $WAITSTATE

34.1 Action

Directive de compilation pour activer la SRAM externe et pour insérer un temps d'attente (WAIT STATE) pour un signal ALE signal. (Address latch enable) plus lent.
On doit utiliser CONFIG XRAM désormais.

34.2 Syntaxe

$WAITSTATE

34.3 Remarques

$WAITSTATE était utilisé pour réécrire l'option de compilation.
Des états d'attente sont nécessaires dans les composants ne supportent pas la vitesse des Enable venant des microcontrôleurs AVR.

34.4 Voir aussi

$XA , CONFIG XRAM

---

35 $XA

35.1 Action

Directive de compilation pour activer l'accès à la mémoire XRAM externe, cette directive ne doit plus être utilisée

35.2 Voir Aussi

$WAITSTATE , CONFIG XRAM

---

36 $XRAMSIZE

36.1 Action

Pour spécifier la taille de la mémoire externe XRAM.

36.2 Syntaxe

$XRAMSIZE = [&H] taille

36.3 Remarques

Taille : Une constante avec la taille de la XRAM externe.
Cette taille peut aussi être configurée dans les options du compilateur
La directive $XRAMSIZE réécrit ce réglage, Ce qui est fortement recommandé.
C'est important que $XRAMSTART précède $XRAMSIZE

---

37 $XRAMSTART

37.1 Action

Spécifie l'adresse de la mémoire externe XRAM

37.2 Syntaxe

$XRAMSTART = [&H] adresse

37.3 Remarques

Adresse l'adresse en hexadécimale où sont stockées les data ou la plus petite adresse qu'autorise le RAM chip. Adresse doit être une constante.

Par défaut la mémoire externe démarre après la mémoire interne, aussi les adresses les plus basses ne peuvent être utilisées pour stocker des informations
On peut par exemple spécifier une adresse en &H400, ce qui est plus sûr que &H260.

37.4 Voir aussi

$XRAMSIZE

Les instructions de config

Les instructions CONFIG sont utilisées pour configurer les composants Hardware ou soft.
En général on écrit ces instructions au début du programme, juste après les directives de compilation qui commencent par $...)

Il est facilement compréhensible que certaines instructions soient reconfigurables à l'intérieur du programme et d'autre ne le soient pas (un afficheur LCD ne peut pas se voir changer les broches en cours de programme !) en revanche un Timer ou un port Com peut être adapté.

1 CONFIG 1WIRE

1.1 Action

Configure la broche à utiliser pour les instructions 1WIRE

1.2 Syntaxe

CONFIG 1WIRE= broche

1.3 Remarques

Broche Le 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.

1.4 Programme

Un chapitre entier est consacré à ces outils.

---

2 CONFIG ACI

2.1 Action

Configuration du Comparateur Analogique

2.2 Syntaxe

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

2.3 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 quels événements de comparaison l'interruption se produit.

3 CONFIG ADC

3.1 Action

Configure le convertisseur Analogique/digital

3.2 Syntaxe

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

3.3 Remarques

ADC Mode de travail peut-être SINGLE ou FREE
PRESCALER pré-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 addition-nelles. Leur valeur peut être OFF AVCC ou INTERNAL…

Chip	Modes	ADC_REFMODEL
2233,4433,4434,8535,m103, m603, m128103	OFF AVCC	0
m165, m169, m325,m3250, m645, m6450, m329,m3290, m649, m6490,m48,m88,m168	OFF AVCC INTERNAL or INTERNAL_1.1	1
tiny15,tiny26	AVCC OFF INTERNAL INTERNALEXTCAP	2
tiny13	AVCC INTERNAL	3
tiny24,tiny44,tiny85	AVCC EXTERNAL or OFF INTERNAL or INTERNAL_1.1	4
m164,m324,m644,m640,m1280, m1281,m2561,m2560	AREF or OFF AVCC INTERNAL1.1 INTERNAL_2.56	5
tiny261,tiny461,tiny861, tiny25, tiny45,tiny85	AVCC EXTERNAL or OFF INTERNAL_1.1 INTERNAL_2.56_NOCAP INTERNAL_2.56_EXTCAP	7
CAN128, PWM2_3,USB1287, m128, m16, m163, m32, m323, m64	AREF or OFF AVCC INTERNAL or INTERNAL_2.56	8
	You may also use VALUE=value	

3.4 Voir Aussi

GETADC , Exemple ADC.bas et ADC-int.bas (avec un peu d' assembleur !)

4 CONFIG ATEMU

4.1 Action

Configure les données et l'horloge du clavier PS/2

4.2 Syntaxe

CONFIG ATEMU = int , DATA = data, CLOCK=clock

4.3 Remarque

Ce Config nécessite un ADD ON payant voir le site www.mcselec.com
Voir la documentation en anglais.

---

5 CONFIG BCCARD

5.1 Action

Configure les broches du microcontrôleurs utilisées pour les BasicCard.

5.2 Syntax

CONFIG BCCARD = port, IO=broche, RESET=broche

5.3 Remarque

Ce Config nécessite un ADD ON payant voir le site www.mcselec.com
Voir la documentation en anglais.

5.4 Voir aussi

Bcccard.bas

---

6 CONFIG CLOCK

6.1 Action

Configure le Timer qui doit être utilisé pour TIME$ et DATE$

6.2 Syntaxe

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

6.3 Remarques

SOFT|USERSOFT utilise les options et 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. 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 c'est-à-dire qui ont la possibilité d'avoir un quartz 32Ko sur les broches TOSC1 et TOSC2.

6.4 Voir Aussi

TIME$, DATE$ et le programme exemple clockeng.bas ou autre…

---

7 CONFIG CLOCKDIV

7.1 Action

Règle le diviseur de la fréquence d'horloge

7.2 Syntax

CONFIG CLOCKDIV = Constant

7.3 Remarques

Constant le diviseur à utiliser. Valeur possible 1, 2, 4, 8, 16, 32, 64, 128 and 256. Ces options pour régler le diviseur d'horloge sont disponibles sur la plupart des nouveaux µP. la valeur par défaut est 1. Avec un diviseur de 8 et une fréquence de 8MHz la fréquence de travail sera de 1MHz. Des vitesses plus lentes peuvent être utilisées pour réduire la consommation et pour avoir une plus grande précision de fréquence. Certains µP ont des fuses-bits de réglage qui font double emploi avec cette fonction.
Quand on ajuste le diviseur, il faut aussi régler la directive $CRYSTAL.

7.4 Voir aussi

$CRYSTAL

7.5 Exemple

CONFIG CLOCKDIV = 8

---

8 CONFIG COM1, CONFIG COM2, config COMx

8.1 Action

Configuration de l'UART des µP AVR qui possèdent une UART étendue comme le M8 ou 2 UART comme le M128 ou d'autre pouvant gérer jusqu'à 4 ports série MEGA2560 par exemple.

8.2 Syntaxe

CONFIG COM1(2) = dummy , synchrone=SY, parity = Pa, stopbits=SB, Databits=DB, clockpol=CP

[1] In Service Register

8.3 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

8.4 Voir Aussi

DATASHEET des microcontrôleurs

8.5 Exemple

```
Config Com1 = 19200 , Synchrone = 0 , Parity = None , Stopbits = 1 , Databits = 8……
Config Com2 = 19200 , Synchrone = 0 , Parity = None , Stopbits = 1 , Databits = 8…..
Config Com3 = 19200 , Synchrone = 0 , Parity = None , Stopbits = 1 , Databits = 8…..
Config Com4 = 19200 , Synchrone = 0 , Parity = None , Stopbits = 1 , Databits = 8…..
'Ouvre toutes les  UARTS le port 1 est en #0 par défaut
Open "com2:" For Binary As #1
Open "Com3:" For Binary As #2
Open "Com4:" For Binary As #3
Print "Hello" 'first uart
Dim B As Byte
Dim Tel As Word
Do
Incr Tel
Print Tel ; " test serial port 1"
Print #1 , Tel ; " test serial port 2"
Print #2 , Tel ; " test serial port 3"
Print #3 , Tel ; " test serial port 4"
B = Inkey(#3)
If B <> 0 Then
Print #3 , B ; " from port 4"
End If
Waitms 500
Loop
Close #1
Close #2
Close #3
End
```

---

9 CONFIG DATE

9.1 Action

Configure le format selon lequel la date sera affichée.

9.2 Syntaxe

CONFIG DATE = DMY , Séparateur = char

9.3 Remarques

DMY jour(Day), le mois(Month) and l'année(Year) . Usage DMY, MDY or YMD.
Char un caractère séparateur : / , - or . (point)
Le tableau suivant montre les principaux formats internationaux.

Pays	Format	Expression
Amérique 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 = .
Italie	dd-mm-yy	Config Date = DMY, Separator = -
Japon/Taiwan	yy/mm/dd	Config Date = YMD, Separator = /
USA	mm-dd-yy	Config Date = MDY, Separator = -

9.4 Voir Aussi

CONFIG CLOCK

10 CONFIG DCF77

10.1 Action

Signale au compilateur d'utiliser les signaux de la radio DCF-77(les horloges atomiques)

10.2 Syntaxe

CONFIG DCF77 = pin , timer = timer [INVERTED=inv, CHECK=check,
UPDATE=upd, UPDATETIME=updtime , TIMER1SEC=tmr1sec,
SWITCHPOWER=swpwr,
POWERPIN=pin, POWERLEVEL = pwrlvl , SECONDTICKS=sectick ,DEBUG=dbg ,
GOSUB = Sectic]

10.3 Exemple

Test_DC577.Bas

---

11 CONFIG DEBOUNCE

11.1 Action

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

11.2 Syntaxe

CONFIG DEBOUNCE= time

11.3 Remarques

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

11.4 Voir Aussi

DEBOUNCE, programme exemple DEBOUN.bas

---

12 CONFIG HITAG

12.1 Action

Configure le timer et les variables HITAG pour les circuits RFID du type EM4095 par exemple voir l'exemple complet et le kit Mcselec :
http://www.mcselec.com/index.php?page=shop.product_details&flypage=shop.flypage&product_id=171&category_id=8&option=com_phpshop&Itemid=1

12.2 Syntaxe

CONFIG HITAG = prescale, TYPE=tp, DOUT = dout, DIN=din , CLOCK=clock, INT=int
CONFIG HITAG = prescale, TYPE=tp, DEMOD= demod, INT=@int

13 CONFIG I2CDELAY

13.1 Action

Directive de compilation qui remplace la routine interne I2Cdelay

13.2 Syntaxe

CONFIG I2CDELAY=valeur

13.3 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.

13.4 Voir Aussi

CONFIG SCL, CONFIG SDA et l'exemple : I2C.bas

---

14 CONFIG I2CSLAVE

14.1 Action

Configure le mode I2C esclave. Cette configuration demande l'ADD ON voir :
http://www.mcselec.com/index.php?page=shop.product_details&flypage=shop.flypage&product_id=34&category_id=6&option=com_phpshop&Itemid=1

14.2 Syntaxe

CONFIG I2CSLAVE = adresse , INT = interruption , TIMER = tmr

14.3 Remarques

Adresse	Une adresse paire par exemple 60 et inférieur à 256.
Interruption	l'interruption qui doit être utilisée INT0 par défaut
Tmr	Le timer qui doit être utilisé TIMER0 par défaut.

14.4 See Also

CONFIG TWI et l'exemple I2C-SLAVE.BAS

---

15 CONFIG INPUT

15.1 Action

Signale au compital de mofier le comportement de la ligne terminal Input

15.2 Syntaxe

CONFIG INPUT = terme , ECHO=echo

15.3 Remarques

Terme un paramètre qui peut prendre l'une des valeurs suivantes :
CR - Carriage Return (defaut) chr(13)
LF - Line Feed (chr10)
CRLF
LFCR
Echo un paramètre qui peut prendre l'une des valeurs suivantes :
CR - Carriage Return
LF - Line Feed
CRLF – CR+ LF (valeur par defaut)
LFCR
le 'terme' est un paramètre qui indique quel (s) est (sont) les caractères attendus pour terminer l'INPUT . il n'y a pas d'impact sur le fichier système DOS input

Quand NOECHO est utilisé il n'y a pas de caractère retourné, même si c'est indiqué dans CONFIG INPUT

15.4 Voir aussi

INPUT

---

16 CONFIG INTx

16.1 Action

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

16.2 Syntaxe

CONFIG INTx = etat

16.3 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

16.4 Voir Aussi

Exemple : INT0.bas

---

17 CONFIG GRAPHLCD

17.1 Action

Configure l'afficheur LCD graphique.

17.2 Syntaxe

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

17.3 Remarques

Type Drivers T6963C : 240 * 64, 128* 128, 128 * 64 , 160 * 48 or 240 * 128

Pour les afficheurs type SED : 128 * 64sed, 120* 64SED, SED180*32
Pour le 132x132 color :COLOR
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.

Les afficheurs couleurs, www.display3000 propose 2 afficheurs de très bonne qualité à un prix raisonnable sachant qu'ils sont fournit avec un ATMEGA128, nous déconseillons l'ATMEGA8 trop vite rempli. Voir sur l'aide anglaise a config GRAPHLCD quelques remarques importantes.

17.4 Voir Aussi

Programmes : T6963_240_128.bas, T6963v3.bas, KS108.bas , clockeng.bas

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

---

18 CONFIG KBG

18.1 Action

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

18.2 Syntaxe

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

18.3 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 lignes 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.

18.4 Voir Aussi

GETKBG

---

19 CONFIG KEYBOARD

19.1 Action

Configuration de l'instruction GETATKBG ( )

19.2 Syntaxe

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

19.3 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, GND et VCC.

19.4 Voir Aussi

La table de codes dans l'aide officielle. GETATKBG

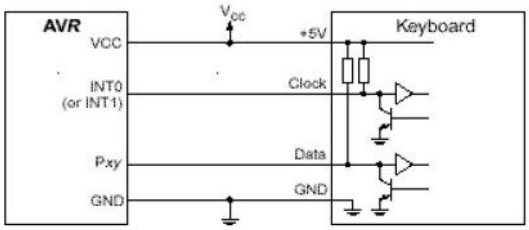


Table 1. AT Keyboard Connector Pin Assignments

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

20 Config LCD

20.1 Action

Configure les afficheurs LCD et change les paramètres du compilateur.

20.2 Syntaxe

CONFIG LCD = LCDtype , CHIPSET=KS077 | Dogm163v5 | DOG163V3 | DOG162V5 | DOG162V3 [,CONTRAST=value]

20.3 Remarques

LDCtype Le type d'afficheur utilisé : 40*4, 16*1, 20*2, 20*4, ou 16*2
CHIPSET KS077
La plupart des LCD-Texte utilise le même protocole, type Hitachi. Quelques un utilise le KS077 qui est presque compatible mais a besoin d'un fonction additionnelle pour être utilisé.

CHIPSET DOGM (idem KS077)

16 x 2 LCD est ajusté avec DOG162V3 pour une tension d'alimentation de 3V DOG162V5 pour une tension d'alimentation de 5V
16 x 3 LCD est ajusté avec DOG163V3 pour une tension d'alimentation de 3V
DoG163v5 pour une tension d'alimentation de 3V
CONTRAST cette option n'est nécessaire que pour les afficheurs EADOG

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.

On peut utiliser INITLCD pour réinitialiser le LCD n'importe où dans le programme.

20.4 Voir Aussi

Config LCDBUS, Config LCDMODE, Config LCDPIN exemples LCD.bas EADOG.bas

---

21 Config LCDBUS

21.1 Action

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

21.2 Syntaxe

Config LCDBUS=constante

21.3 Remarques

Constante 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)

21.4 Voir Aussi

Config LCD

---

22 Config LCDMODE

22.1 Action

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

22.2 Syntaxe

Config LCDMODE=Type

22.3 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)

22.4 Voir Aussi

Config LCD Config LCDPIN

---

23 Config LCDPIN

23.1 Action

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

23.2 Syntaxe

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

23.3 Remarques

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

23.4 Voir Aussi

Config LCD

---

24 Config PORT, Config PIN

24.1 Action

Configure un port ou une broche du port.

24.2 Syntaxe

CONFIG PORTx=etat
CONFIG PINx.y=etat

24.3 Remarques

Etat Une constante qui peut être INPUT ou OUPUT
INPUT oriente le DDR (Data Direction Register) pour entrer des données par le **portx**.
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.

24.4 Voir Aussi

le programme exemple Port.BAS,

---

25 CONFIG PRINT

25.1 Action

Configure l'UART pour être utilisé en RS485

25.2 Syntaxe

Config Print0 = Portb. 0 , Mode = Set
CONFIG PRINT0 = broche
CONFIG PRINT1 = broche

25.3 Remarques

Broche Le port qui sera utilisé pour contrôler la direction d'un driver RS485
mode SET or RESET
Utiliser PRINT or PRINT0 pour le premier port série et PRINT1 le second.

La communication RS-485 half duplex utilise une broche pour la direction des données.
Le CONFIG PRINT automatise le réglage manuel
Suivant le mode SET or RESET qui affecte la broche avant le transfert, Cette broche sera basculée en réception après transmission.

25.4 Voir aussi

CONFIG PRINTBIN

25.5 Exemple

rs485.bas

---

26 CONFIG PRINTBIN

26.1 Action

Configure PRINTBIN

26.2 Syntaxe

CONFIG PRINTBIN = extended

26.3 Remarques

extended il n'y a qu'un seul mode qui permet d'envoyer des tableaux de plus de 255 éléments vers le port série.

Sans l'option CONFIG PRINTBIN le nombre maximum d'éléments est de 255
Mais cette option est couteuse en mémoire aussi est-elle optionnelle.

26.4 Voir aussi

CONFIG PRINT exemple : printbin.bas

---

27 CONFIG PS2EMU

27.1 Action

Configure les données et pulse horloge pour une souris PS2

27.2 Syntaxe

CONFIG PS2EMU= int , DATA = data, CLOCK=clock

27.3 Remarques

Nécessite l'achat d'un ADD- ON chez www.mcselec.com

---

28 CONFIG RC5

28.1 Action

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

28.2 Syntaxe

CONFIG RC5 = Pin [,TIMER=2]

28.3 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 fichier .CFG du projet.

28.4 Voir Aussi

GETRC5, RC5SEND

---

29 CONFIG SCL

29.1 Action

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

29.2 Syntaxe

CONFIG SCL = pin

29.3 Remarques

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

29.4 Voir Aussi

CONFIG SDA, CONFIG I2CDELAY, Voir le chapitre concernant les interfaces
Et les programmes I2C

---

30 CONFIG SDA

30.1 Action

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

30.2 Syntaxe

CONFIG SDA = pin

30.3 Remarques

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

30.4 Voir Aussi

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

31 CONFIG SERIALIN, CONFIG SERIALIN1

31.1 Action

Configure l'UART hardware pour utiliser un tampon (Buffer) en entrée.
Serialin1 (pour la seconde UART pour les microcontrôleurs équipés.)

31.2 Syntaxe

CONFIG SERIALIN = BUFFERED, SIZE=size

31.3 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 [1] Byte : un pointeur vers la place mémoire où le tampon a été écrit.

_RS_TAIL_PTR0 [1] Byte : un pointeur vers la place mémoire où le tampon a été lu.

_RS232INBUF0 [1] Un tableau de Bytes utilisé comme tampon tournant pour recevoir les caractères

31.4 Voir Aussi

CONFIG SERIALOUT, exemple : RS232BUFFER.bas

---

32 CONFIG SERIALOUT

32.1 Action

Configure l'UART hardware pour utiliser un tampon en sortie

32.2 Syntaxe

CONFIG SERIALOUT = BUFFERED, SIZE=SZ

32.3 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 [1] Byte qui reçoit la tête du tampon
_RS_TAIL_PTRW0 [1] Byte qui reçoit la queue du tampon
_RS_TAIL_PTRW0 [1] Un tableau de Bytes utilisé comme tampon tournant pour recevoir les caractères à envoyer

32.4 Voir Aussi

CONFIG SERIALIN, programme exemple RS232BUFFEROUT.bas et les programmes exemples série

---

33 CONFIG SINGLE

33.1 Action

Signale au compilateur d'utiliser une conversion alternative pour les variables Single

33.2 Syntaxe

CONFIG SINGLE = SCIENTIFIC , DIGITS = valeur

33.3 Remarques

Digits Une constante numérique entre 0 et 7.
Pour représenter les Single en notation scientifique avec un nombre plus ou moins grand de chiffres après la virgule : 12e3 si DIGITS=1.

33.4 Exemple

```
Config Single = Scientific , Digits = 7
Dim S As Single
```

---

34 CONFIG SPI

34.1 Action

Configure les instructions relatives à la connexion SPI

34.2 Syntaxe pour une communication soft

CONFIG SPI = SOFT, DIN = broche, DOUT = broche, SS = broche|NONE, CLOCK = broche

34.3 Syntaxe pour une communication hard

CONFIG SPI=MODE, DATA ORDER=DO, MASTER=M, POLARITY=PY, PHASE= PH, CLOCKRATE=CR

CONFIG SPI = HARD, INTERRUPT=ON|OFF, DATA ORDER = LSB|MSB , MASTER =
YES|NO , POLARITY = HIGH|LOW , PHASE = 0|1, CLOCKRATE = 4|16|64|128 , NOSS=1|0

34.4 Remarques

MODE SOFT pour une émulation soft du SPI, ce qui laisse l'utilisateur libre de choisir les ports à utiliser. Ne fonctionne qu'en mode Master
HARD pour une utilisation utilisant la configuration du µP, les broches ne sont pas modifiables.
DIN Broche pour Data input ou MISO comme portB.0
DOUT Broche pour Data output ou MOSI
SS Broche pour SLAVE SELECT ou NONE quand on veut utiliser de multiple esclaves on utilise plusieurs broches ce qui signifie aussique pour choisir un esclave la Broche SS qui lui correspond devra être mise au niveau 0 avant puis remis à « 1 » après.

CLOCK Broche pour Clock
DO Choix du type de transmission effectué en premier: DO= MSB ou LSB
M YES (maître) ou NO (esclave)
PY **polarity** HIGH pour mettre la ligne CLOCK à 1 quand le SPI est inoccupé
Polarity LOW pour mettre la ligne CLOCK à 0 quand le SPI est inoccupé
PH **phase** 0 ou 1 Voir la datasheet pour connaître les combinaisons Polarity/phase.
CLOCKRATE Facteur de division de la fréquence du quartz.
DATA ORDER le choix entre MSB ou LSB sont transférer en premier (Mean Signifiant Bit les 4 bit les plus importants LSB less Signifiant bit les 4 bit les moins …)
MASTER Yes le µC est maitre, NO le µC est esclave
NOSS 1 ou 0 on utilise 1 quand on ne veut pas soit générer en Master Mode
INTERRUPT ON ou OFF. ON autorise les interruption . ENABLE SPI DISABLE SPI fon la même chose.

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

34.5 Voir Aussi

SPIIN, SPIINIT, SPIMOVE, SPIOUT , les programmes : spi-slave.bas, sendspi.bas, spisoftware.bas dans le répertoire samples SPI
Voir le chapitre concernant ce bus

---

35 CONFIG SERVOS

35.1 Action

Configuration d'un contrôle de servomoteur.

35.2 Syntaxe

Config servos = X, servo1= portn.x, Servo2=portm.y, reload=rl

35.3 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)

35.4 Voir Aussi

L'exemple Servos.bas

---

36 CONFIG TCPIP

Voir le chapitre TCP/IP

---

37 CONFIG TIMER0

37.1 Action

Configuration du TIMER0 en timer ou en compteur.

37.2 Syntaxe

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

37.3 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.

37.4 Voir Aussi

Programmation Avancée, Programme Timer0.bas

---

38 CONFIG TIMER1

38.1 Action

Configuration du Timer1

38.2 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

38.3 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

38.4 Voir Aussi

Programme TIMER1.bas

---

39 CONFIG TIMER2

39.1 Action

Configuration du TIMER2

39.2 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

39.3 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

39.4 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
```

---

40 CONFIG TWI

40.1 Action

Configure l'interface TWI (Two Wire serial Interface). (Voir le chapitre concernant les interfaces)

40.2 Syntaxe

CONFIG TWI = vitessse d'horloge

40.3 Remarques

Vitesse d'horloge la fréquence d'horloge désirée pour SCL

La frequence d'horloge est en général de 400KHz mais certains composants travaillent à 100KHz
Quand TWI est utilisé en mode Esclave, le Master doit avoir une fréquence dh'horloge plus rapide.
Attention à bien régler la fréquence du quartz $Crystal

40.4 Voir aussi

programmes TWI

---

41 CONFIG TWISLAVE

Configure l'adresse et les options de l'esclave TWI, nécessite l'ADD-ON payant pour cet usage : www.mcselec.com

---

42 CONFIG WAITSUART

42.1 Action

Directive de compilation qui indique le temps d'attente de l'UART soft après l'envoi du dernier Byte.

42.2 Syntaxe

CONFIG WAITSUART= valeur

42.3 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.

42.4 Voir Aussi

OPEN et l'exemple : Open.bas

---

43 CONFIG WATCHDOG

43.1 Action

Configuration du timer watchdog (chien de garde), un watchdog est un élément de sécurité dans le programme, c'est un peu difficile à utiliser mais indispensable dans les programmes où il y a des risques important de « plantage » liaison série par exemple…

43.2 Syntaxe

CONFIG WATCHDOG = time

43.3 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

43.4 Voir Aussi

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

---

44 CONFIG X10

44.1 Action

Configure les broches utilisées pour les transmissions X10 (courants porteurs) Les équipements X10 connectés sur le secteur 230V en Europe, doivent être homologués EDF, MCSELEC ni le traducteur ne peuvent être tenu responsables d'une utilisation non homologuées.

44.2 Syntaxe

CONFIG X10 = BrocheZC , TX = portpin

44.3 Remarques

PinZC La broche qui est connectée à la sortie Zéro-cross du TW-523 c'est la broche qui sera utilisé en INPUT
Portpin La broche qui est connectée au TX du TW-523. TX est utilisé pour envoyer des données X10 au TW-523 . Cette broche est utilisée en mode OUTPUT. Le connecteur RJ-11 du TW-523 présente les connexions suivantes : TW-523 RJ-11 :

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

Voir Aussi

X10DETECT , X10SEND , programme X10.bas

---

45 CONFIG XRAM

45.1 Action

Signale au compilateur de régler les options pour une mémoire RAM externe

45.2 Syntaxe

CONFIG XRAM = mode [, WaitstateLS=wls , WaitStateHS=whs]

45.3 Remarques

Mode Le mode mémoire, celui-ci est soit « enabled(autorisé) » ou « disabled » .Par défaut l'accès mémoire est « Disabled »

Wls Quand l'accès à la mémoire externe est Enabled, certains microcontrôleurs permettent l'usage d'un temps d'attente. Un microcontrôleur moderne comme le MEGA8515 à 4 modes

0 - pas de temps d'attente
1 - 1 cycle d'attente pendant un cycle écriture / lecture.
2 - 2 cycles d'attente pendant un cycle écriture / lecture.
3 - 2 cycles d'attente pendant un cycle écriture / lecture et 1 avant la sortie d'une nouvelle adresse.

WLS travaille sur le secteur le plus bas.

Whs Quand l'accès à la mémoire externe est Enabled, certains microcontrôleurs permettent l'usage d'un temps d'attente.
Les 4 modes du ATMEGA8515 par exemple sont :

0 - pas de temps d'attente
1 - 1 cycle d'attente pendant un cycle écriture / lecture.
2 - 2 cycles d'attente pendant un cycle écriture / lecture.
3 - 2 cycles d'attente pendant un cycle écriture / lecture et 1 avant la sortie d'une nouvelle adresse.

WHS travaille sur le secteur le plus haut.

Un temps d'attente est nécessaire quand on connecte d'autre équipement sur le BUS ce qui réduit la vitesse (LCD par exemple.
La directive $XA ne doit plus être utilisée. Elle remplissait la même fonction que CONFIG XRAM=Enabled

45.4 Voir aussi

$XA , $WAITSTATE, programme Xram.bas

LES INSTRUCTIONS GENERALES

Les instructions mathématiques sont regroupées à la suite des instructions générales.

1 Alias

1.1 Action

Indique que la variable peut être référencée par une autre.

1.2 Syntaxe

Newvar ALIAS Oldvar

1.3 Remarques

Oldvar = par exemple PortB.1
Newvar = par exemple direction

1.4 Exemple

```
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, exemple Buzzer, Led, relaisA
```

1.5 Voir Aussi

Const , programme Alias.bas

---

2 ASC

2.1 Action

Retourne la valeur ASCII du premier caractère d'une String

2.2 Syntaxe

Var=ASC(String)

2.3 Remarques

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

2.4 Voir Aussi

CHR , programme asc.bas

---

3 Base64DEC

Voir le chapitre TCP/IP

---

4 BASE64ENC

Voir le chapitre TCP/IP

---

5 BAUD, BAUD1

5.1 Action

Change la vitesse de transfert de ou des UART du microcontrôleur.

5.2 Syntaxe

BAUD = Var
BAUD #x, const
BAUD1 = Var
BAUD1 #x, const

5.3 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

5.4 Voir Aussi

$Crystal, $Baud, programmes exemples série

---

6 BCD

6.1 Action

Pour convertir une variable au format BCD en String

6.2 Syntaxe

Print BCD(Var)

LCD BCD(Var)

6.3 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
6.3..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.

6.4 Voir Aussi

MAKEBCD, MAKEDEC, programme bcd.bas

---

7 BIN

7.1 Action

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

7.2 Syntaxe

VarString=BIN(var)

7.3 Remarques

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

7.4 Voir Aussi

BINVAL, STR, VAL, HEX, HEXVAL., programme bin.bas

---

8 BIN2GREY / GREY2BIN

8.1 Action

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

8.2 Syntaxe

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

8.3 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

8.4 Voir Aussi

Le programme exemple :Greycode.bas

---

9 BINVAL

9.1 Action

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

9.2 Syntaxe

Var=BINVAL(varString)

9.3 Remarques

VarString La String à transformer.
Var Une variable numérique

9.4 Voir Aussi

BIN, STR, VAL, HEX, HEXVAL., programme binval.bas

---

10 BITS()

10.1 Action

Met tous les bits spécifiés à 1.

10.2 Syntaxe

Var = Bits(b1 [,bn])

10.3 Remarques

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

10.4 Voir Aussi

NBITS exemple programme bits-nbits.bas

---

11 BITWAIT

11.1 Action

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

11.2 Syntaxe

Bitwait x, set /reset

11.3 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.

11.4 Voir Aussi

Programme Exemple : Port.bas

---

12 BLOAD

12.1 Action

Ecrit le contenu d'un fichier dans la SRAM

12.2 Syntaxe

BLoad sFileName, wSRAMPointer

12.3 Remarques

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

12.4 Voir Aussi

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

---

13 BSAVE

13.1 Action

Sauve un champ de la SRAM dans un fichier

13.2 Syntaxe

BSave sFileName, wSRAMPointer, wLength

13.3 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) Compteur de bytes de la SRAM qui doit être écrit dans le fichier

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

13.4 Voir Aussi

BLOAD et l'aide dans le fichier Help pour un exemple.

---

14 BOX, BOXFILL

14.1 Action

Dessine une boite remplie sur un afficheur graphique.

14.2 Syntax

BOX (x1,y1) - (x2,y2) , color
BOXFILL (x1,y1) - (x2,y2) , color

14.3 Remarques

x1 → le coin haut gauche de la boite
y1 → la position haute de la boite
x2 → le coin droit bas de la boite
y2 → la position basse de la boite
color → la couleur qui remplie la boite

Color ne fonctionne qu'avec les écrans couleur, pour les écrans N&B la boite n'est pas remplie, elle n'est que dessiner.
En B&W on utilisera BOXFILL pour dessiner une boite pleine.

14.4 Voir aussi

LINE , CIRCLE

---

15 BUFSPACE()

15.1 Action

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

15.2 Syntaxe

Var = BufSpace(n)

15.3 Remarques

Var Une variable Word ou Integer qui est assignée avec la place libre dans le buffer.
N Une constante variant de 0-3.

0 : → Buffer de sortie de l' UART 1
1 : → Buffer d'entrée de l'UART 1
2 : → Buffer de sortie de l' UART 2
3 : → Buffer d'entrée de l'UART 2

Avec cette fonction on peut connaître la place disponible dans les Buffer, ce qui évite un Overflow.

15.4 Voir Aussi

Les fonctions relatives aux ports séries et spécialement config serialout : serie

---

16 Byval / BYREF

16.1 Action

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

16.2 Syntaxe

Sub test(Byval var)
Sub test(*Byref* var)
Sub test(var) ' byref

16.3 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
Attention, Byref est utilisé par défaut, il n'est plus nécessaire de l'écrire, le compilateur ne l'accepte plus.

16.4 Voir Aussi

SUB, programme exemple : Declare.bas

---

17 CALL

17.1 Action

Appelle et exécute une procédure (SUB)

17.2 Syntaxe

Call Proc [(var1,varx)]

17.3 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

17.4 Voir Aussi

Declare, sub, exit, function, local et programme Declare.Bas

---

18 CHECKSUM

18.1 Action

Renvoie un total de contrôle appelé « checksum », en français on féminise ce mot sans traduction.

18.2 Syntaxe

Print Checksum(var)
B=Checksum(var)

18.3 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.

18.4 Voir Aussi

CRC8

18.5 Exemple

la checksum est un octet. L'exemple VB suivant est équivalent :

```
Dim Check as Byte
Check = 255
For x = 1 To Len(s$)
  Check = check – ASC(mid$(s$,x,1))
Next
```

---

19 CHR

19.1 Action

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

19.2 Syntaxe

Svar=**Chr**(var)

19.3 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

19.4 Voir Aussi

ASC , et en annexe le tableau de correspondance ASCII, le programme conversions.bas sur le site de l'auteur.

---

20 CIRCLE

20.1 Action

Dessine un cercle sur un afficheur graphique.

20.2 Syntaxe

CIRCLE(x0,y0) , rayon, couleur

20.3 Remarques

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

20.4 Voir Aussi

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

---

21 Clear

21.1 Action

Efface ou le buffer l'entrée ou de sortie d'une liaison série

21.2 Syntax

CLEAR bufname

21.3 Remarques

Bufname Nom du buffer de la liaison série comme Serialin, Serialin1 , Serialout or Serialout1 pour les µ-contrôleur ayant plus d'une UARTS :SERIALIN2, SERIALIN3, SERIALOUT2, SERIALOUT3
Cette instruction est nécessaire quand on reçoit des données par la liaison série sur interruption, le buffer se rempli, mais on ne sais pas toujours où l'on en est.
Cette fonction a été introduite avec la version 1.11.8.3

21.4 Voir aussi

CONFIG SERIALIN, CONFIG SERIALOUT

21.1 ASM

Calls _BUF_CLEAR from MCS.LIB

21.2 Exemple

CLEAR SERIALIN

---

22 CLS

22.1 Action

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

22.2 Syntaxe

CLS
Pour le LCD graphique
CLS
CLS TEXT 'efface le texte seulement pour les afficheurs graphiques
CLS GRAPH 'efface les graphes seulement pour les afficheurs graphiques
(Il y a un bug pour ces 2 dernières lignes dans la version 1.1.9.11)

22.3 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.

22.4 Voir Aussi

LCD, SHIFTLCD, SHIFTCURSOR, DISPLAY.

---

23 CLOCKDIVISION

23.1 Remarques

Ne plus utiliser (réservé aux composants ATMEGA 103 et 603) qui sont obsolètes.

---

24 CLOSE

24.1 Action

Ferme un composant (port serie , fichier…)

24.2 Syntaxe

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

24.3 Remarques

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

24.4 Voir Aussi

Open, Print

---

25 CLOSESOCKET

Voir chapitre TCPIP

---

26 CONST

26.1 Action

Déclaration d'une constante symbolique

26.2 Syntaxe

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

26.3 Remarques

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

26.4 Voir Aussi

ALIAS et le programme Const.bas

27 Compact flash

27.1 Drivercheck, Drivegetidentity, DriveInit, DriveReset, DriveReadSector, DriveWriteSector

Toutes ces instructions sont spécifique au compact flash, ne pas confondre avec les SD card
Elles seront explicitées ultérieurement.

28 COUNTER0 & COUNTER1

28.1 Action

Ajuste ou récupère le registre interne16 bits

28.2 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

28.3 Remarques

Var Une constante ou variable Byte, Integer ou Word

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

28.4 Voir Aussi

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

29 CPEEK

29.1 Action

Retourne un Byte stocké dans la mémoire code

29.2 Syntaxe

Var= CPEEK(adresse)

29.3 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.

29.4 Voir Aussi

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

---

30 CPEEKH

30.1 Action

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

30.2 Syntaxe

Var= CPEEKH(adresse)

30.3 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 MC ayant une mémoire paginée, M128 par exemple.

---

31 CRC8, CRC16,CRC16UNI, CRC32

31.1 Action

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

31.2 Syntaxe

Var=CRC8(source, L)

31.3 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

31.4 Voir Aussi

CHECKSUM, l'exemple CRC8_16_32 .bas .

---

32 CRYSTAL

32.1 Remarques

Ne plus utiliser cette instruction voir $Crystal

---

33 CURSOR

33.1 Action

Règle l'état du curseur du LCD texte et graphique

33.2 Syntaxe

Cursor X Y

33.3 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

33.4 Voir Aussi

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

---

34 DATA

34.1 Action

Spécifie les valeurs constantes à lire avec l'instruction READ

34.2 Syntaxe

DATA var, [varn]

34.3 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.

34.4 Voir Aussi

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

---

35 DATE

35.1 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

35.2 Syntaxe

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

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

35.3 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)

35.4 Voir Aussi

DATETIME, SYSDAY, les nombreux exemples de l'aide.

---

36 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 exemples de l'aide.

---

37 DATE$ / TIME$

37.1 Action

Variable interne qui garde la date (l'heure)

37.2 Syntaxe

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

37.3 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.

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

---

38 DBG

38.1 Action
Donne des informations de débogage sur l'UART

38.2 Syntaxe
DBG

38.3 Voir Aussi
$DBG

---

39 DEBOUNCE

39.1 Action
Anti-rebond sur une broche pour un switch.

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

39.3 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

39.4 Voir Aussi

CONFIG DEBOUNCE, programme DEBOUN.bas, BITWAIT

---

40 DDRx

40.1 Action

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

40.2 Syntaxe

DDRx=&B01010101

40.3 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.

40.4 Voir Aussi

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

---

41 DECLARE FUNCTION

41.1 Action

Déclare une fonction utilisateur.

41.2 Syntaxe

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

41.3 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

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

---

42 DECLARE SUB

42.1 Action
Déclare une procédure utilisateur.

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

42.3 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)

42.4 Voir Aussi

Call, Function, programme déclare.bas

---

43 DECR (INCR)

43.1 Action

Décrémente (incremente) une variable d'une unité

43.2 Syntaxe

DECR var
INCR var

43.3 Remarques

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

43.4 Voir Aussi

INCR, programmes DECR.bas, INCR.bas

---

44 DEFxxx

44.1 Action

Déclare toutes les variables qui ne sont pas dimensionnées.

44.2 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

44.3 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 !)

44.4 Voir Aussi

DIM

---

45 DEFLCDCHAR

45.1 Action

Définit un caractère LCD personnalisé.

45.2 Syntaxe

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

45.3 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
```

44.4 Voir Aussi

Programme exemple lcdstk200.bas

---

46 DELAY

46.1 Action

Suspend le programme pendant un temps très court

46.2 Syntaxe

DELAY

46.3 Remarques

La durée est d'environ 1000 µs

46.4 Voir Aussi

WAIT, WAITMS

---

47 DIM

47.1 Action

Dimensionne une variable

47.2 Syntaxe

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

47.3 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.

47.4 Voir Aussi

DEFxxx, CONST, LOCAL

---

48 Disable /Enable

48.1 Action

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

48.2 Syntaxe

Disable *interruption* Enable *interruption*

48.3 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

48.4 Voir Aussi

Programme exemple : SERINT.bas

---

49 DISPLAY

49.1 Action

Met l'afficheur LCD en fonction ou non

49.2 Syntaxe

Display ON/OFF

49.3 Remarques

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

49.4 Voir Aussi

LCD

---

50 DO --- LOOP

50.1 Action

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

50.2 Syntaxe

```
DO
        Instructions
LOOP [UNTIL condition]
```

50.3 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)

50.4 Voir Aussi

EXIT, WHILE---WEND, FOR---NEXT, Programme exemple Do-Loop.bas

---

51 DTMFOUT

51.1 Action

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

51.2 Syntaxe

DTMFOUT nombre, durée
DTMFOUT String, durée

51.3 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 !

51.4 Voir Aussi

Programme exemple DTMFOUT.bas

---

52 ECHO

52.1 Action

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

52.2 Syntaxe

ECHO ON/OFF

52.3 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

52.4 Voir Aussi

INPUT, programme exemple INPUT.bas

---

53 ELSE

Voir IF---THEN

---

54 ENABLE

Voir Disable

---

55 END

55.1 Action

Met fin à l'exécution d'un programme.

55.2 Syntaxe

END

55.3 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.

55.4 Voir Aussi

STOP

---

56 EXIT

56.1 Action

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

56.2 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

56.3 Remarques

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

56.4 Voir Aussi

Programme Exit.bas

---

57 FIX / INT / ROUND

57.1 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.

57.2 Syntaxe

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

57.3 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
```

---

58 FOR---NEXT

58.1 Action

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

58.2 Syntaxe

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

58.3 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

58.4 Voir Aussi

Programme For_next.bas

---

59 FORMAT

59.1 Action

Formatage d'une Variable String numérique

59.2 Syntaxe

Cible =Format(source, "Masque")

59.3 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.

59.4 Voir Aussi

FUSING, programme exemple : Format.bas

---

60 FOURTHLINE/THIRDLINE

60.1 Action

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

60.2 Syntaxe

FOURTHLINE
THIRDLINE

60.3 Remarques

Seulement pour les afficheurs 4 lignes

60.4 Voir Aussi

HOME, UPPERLINE LOWERLINE, LOCATE

---

61 FRAC

61.1 Action

Retourne la partie fractionnaire d'une variable Single.

61.2 Syntaxe

Var =FRAC(X)

61.3 Remarques

VAR	une Single
X	une Single

61.4 Voir Aussi

INT

61.5 Exemple

```
Dim X As Single

X = 1.123456
Print X
Print Frac(x)  ' 0.123456
End
```

---

62 FUSING

62.1 Action

Formatage d'une valeur Single en STRING

62.2 Syntaxe

Cible = FUSING(source, "Masque")

62.3 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

62.4 Voir Aussi

Format, le programme chaine_de_caractere.bas sur notre site.

---

63 GET

63.1 Action

Lit un octet depuis l'UART hard ou soft.
Lit une donnée depuis un fichier ouvert en mode binaire.

63.2 Syntaxe

GET #channel, var
GET #channel, var , [pos] [, length]

63.3 Remarques

GET en combinaison avec l'UART soft ou hard lit un byte sur l'UART désigné.
GET in combinaison avec le système de fichier AVR-DOS est très flexible et versatile :
Il travaille sur les fichiers ouverts en mode BINAIRE, on peut lire tout les type de données

#channel	Le numéro de fichier qui peut être une variable ou une constante.
Var	La variable ou la variable-tableau qui recevra la donnée du fichier
Pos	Une option qui peut être utilisé pour spécifier la position à partir d'où les données seront les lues. C'est une variable de type Long.
Length	Une option qui spécifie le nombre d'octets qui doit être lu.

Par défaut , il n'est pas nécessaire de déclarer le type.
Si Var est un octet on lira un octet, un word ou integer on lira deux octets etc..

Quand la variable est une String , on lira le nombre d'octets correspondant à la dimension de la variable : Dim S as string *10, on lira 10 octets.
Note : Une string ne dépassera pas 254 octets, un tableau est limité à 65535 bytes

63.4 Voir aussi

Programme dans le dossier AVR-DOS et les explications complémentaires dans l'aide Bascom

---

64 GETADC

64.1 Action

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

64.2 Syntaxe

Var= GETADC (broche, [offset])

64.3 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 inséré 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.

64.4 Voir Aussi

CONFIG ADC, programme exemple ADC.bas

---

65 GETATKBD, GETATKBDAW

65.1 Action

Lecture d'un clavier PCAT

65.2 Syntaxe

Var = GETATKBD

65.3 Remarques

Var Une String ou un BYTE qui reçoit la valeur de la touche pressée.
Si aucune touche n'est pressée un 0 est retourné.
L'instruction utilise 2 broches et une table de traduction.
Dans l'aide officielle il y a une table de traduction de codes, comme nous utilisons des claviers AZERTY, nous ne savons pas ci celle-ci est opérationnelle.

65.4 Voir Aussi

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

---

66 GETKBD

66.1 Action

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

66.2 Syntaxe

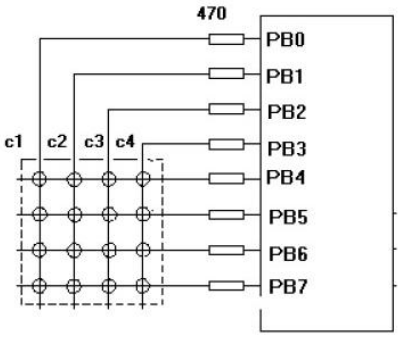
Var = GETKBD( )

66.3 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 broches du port peuvent être utilisées pour d'autres tâches.
Quand aucune touche n'est pressée 16 est retourné.

66.4 Voir Aussi

CONFIG KBG et Logiciel exemple : GETKBD.bas

---

67 GETDSTIP

Voir le chapitre TCP/IP

---

68 GETDSTPORT

Voir le chapitre TCP/IP

---

69 GETRC

69.1 Action

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

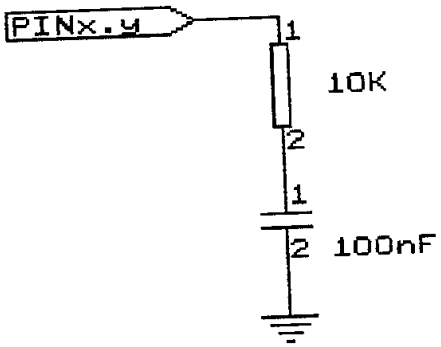
69.2 Syntaxe

var = GETRC(Broche, nombre)

69.3 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 de la capacité.

69.4 Voir Aussi

Exemple : GETRC.bas

---

70 GETRC5

70.1 Action

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

70.2 Syntaxe

GETRC5 (adresse, commande)

70.3 Remarques

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

70.4 Voir Aussi

Datasheet du SFH506-36 (non fournie) de SIEMENS : les programmes exemples dans le répertoire sample/ IR

71 GETTCPREGS

Voir le chapitre TCP/IP

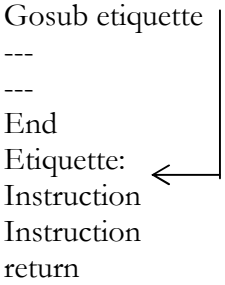
---

72 GOSUB

72.1 Action

Branche le programme vers une subroutine

72.2 Syntaxe



72.3 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.

72.4 Voir Aussi

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

---

73 GOTO

73.1 Action

Saute à une étiquette dans un programme.

73.2 Syntaxe

GOTO Etiquette

---
etiquette:

73.3 Remarques

Il n'y a pas d'instruction Return
Instruction à éviter, risque de saturer les piles d'instruction, voir le chapitre "boucles et branchement du livre 1"

73.4 Voir Aussi

GOSUB

---

74 GREY2BIN

Voir BIN2GREY

---

75 HEX

75.1 Action

Retourne une représentation hexadécimale "String" d'un nombre

---------- ----------

75.2 Syntaxe
VarString=Hex(Var)

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

75.4 Voir Aussi
HEXVAL, programme exemple : conversions.bas

---

76 HEXVAL

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

76.2 Syntaxe
Var = HEXVAL(String)

76.3 Remarques
Var	Une variable numérique
String	La chaîne_de_caractère qui doit être convertie

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

---

77 HIGH/LOW HIGHW

77.1 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

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

77.3 Remarques

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

77.4 Voir Aussi

LOW, HIGHW, programme conversions.bas

---

78 HIGHW

Voir HIGH

---

79 HOME

79.1 Action

Place le curseur en position 1 de la ligne spécifiée.

79.2 Syntaxe

HOME UPPER
HOME LOWER
HOME THIRD
HOME FOURTH

79.3 Remarques

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

79.4 Voir Aussi

LCD, CLS, LOCATE… les programmes LCDxx.bas

---

80 I2Cxxx

Un chapitre complet est consacré aux fonctions I2C

---

81 IDLE

81.1 Action

Met le µP en mode inactif.

81.2 Syntaxe

IDLE

81.3 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.

81.4 Voir Aussi

POWERDOWN

---

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

83 Action

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

Branchement basé sur une comparaison booléenne.

83.1 Syntaxe

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

83.2 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
```

83.3 Voir Aussi

Le cours Basic, Select-Case Programme exemple : IF-Then.bas

---

84 INCR

Voir DECR

---

85 INKEY

85.1 Action

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

85.2 Syntaxe

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

85.3 Remarques

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

S'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.

85.4 Voir Aussi

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

---

86 INP

86.1 Action

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

86.2 Syntaxe

Var = INP(adresse)

86.3 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"

86.4 Voir Aussi

OUT, PEEK et le programme exemple PEEK.bas

---

87 INPUT

87.1 Action

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

87.2 Syntaxe

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

87.3 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.

87.4 Voir Aussi

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

---

88 INPUTBIN

88.1 Action

Lit des données binaires depuis un port série

88.2 Syntaxe

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

88.3 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.

88.4 Voir Aussi

PRINTBIN, les programmes SERIAL

---

89 INPUTHEX

89.1 Action

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

89.2 Syntaxe

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

89.3 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…

89.4 Voir Aussi

INPUT, INPUTBIN… les programmes SERIAL

---

90 INSTR

90.1 Action

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

90.2 Syntaxe

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

90.3 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

90.4 Voir Aussi

Programme string

---

91 INT

Voir FIX

---

92 ISCHARWAITING

92.1 Action

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

92.2 Syntaxe

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

92.3 Remarques

Var Variable Byte, Integer, Word or Long.
Canal un nombre ou une constante qui identifie le canal ouvert
Si aucun caractère n'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

92.4 Voir Aussi

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

---

93 LCASE /UCASE

93.1 Action

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

93.2

93.3 Syntaxe

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

93.4 Remarques

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

---

94 LCD

94.1 Action

Affiche une constante ou une variable sur un afficheur LCD.

94.2 Syntaxe

LCD x

94.3 Remarques

X La variable ou constante à afficher
Plus d'une variable peuvent être affichées séparées par un ;
LCD est reconnu par le simulateur.

94.4 Voir Aussi

Exemple : locate 1,1 :LCD « coucou » ; « bonjour »
Programmes LCD et chapitre "Afficheur" dans la première partie.

---

95 LEFT / RIGHT

95.1 Action

Retourne la partie gauche (droite) d'une String

95.2 Syntaxe

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

95.3 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

95.4 Voir Aussi

MID, INSTR, programme chaine_de_caracteres.bas en annexe

---

96 LEN

96.1 Action

Retourne la longueur d'une String.

96.2 Syntaxe

Var = LEN(String)

96.3 Remarques

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

String ne peut dépasser 254 octets

---

97 LINE

97.1 Action

Dessine une ligne sur un afficheur graphique.

97.2 Syntaxe

LINE(x0,y0) - (x1,y1), color

97.3 Remarques

X0 → Début de la ligne sur le plan horizontal.
Y0 → Début de la ligne sur le plan Vertical.
X1 → Fin de la ligne sur le plan horizontal.
Y1 → Fin de la ligne sur le plan Vertical.
color La couleur pour les afficheurs couleurs supportés par Bascom ou 0 ou 255 pour les afficheurs B&W .

97.4 Voir Aussi

CIRCLE , CONFIG GRAPHLCD , BOX , BOXFILL

---

98 LOAD

98.1 Action

Charge une période déterminée pour un timer.

98.2 Syntaxe

LOAD Timer, Var

98.3 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"

---

99 LOADADR

99.1 Action

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

99.2 Voir Aussi

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

---

100 LOADLABEL

100.1 Action

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

100.2 Syntaxe

Var = LOADLABEL(label)

100.3 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.

100.4 Voir Aussi

PEEK

---

101 LOCAL

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

101.2 Syntaxe
LOCAL var AS TYPE

101.3 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

101.4 Voir Aussi
DIM, le programme exemple : Declare.bas

---

102 LOCATE

102.1 Action
Déplace le curseur de l'afficheur LCD

102.2 Syntaxe
LOCATE Y, X

102.3 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

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

---

103 LOOKDOWN

103.1 Action

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

103.2 Syntaxe

Var=LOOKDOWN(Valeur, Etiquette, Nombre)

103.3 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

103.4 Voir Aussi

LOOKUP et le programme exemple LOOKDOWN.bas

---

104 LOOKUP / LOOKUPSTR

104.1 Action

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

104.2 Syntaxe

Var=LOOKUP(index, etiquette)

104.3 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"
```

104.4 Voir Aussi

LOOKDOWN, LOOKUPSTR

---

105 LOW

Voir HIGH

---

106 LOWERLINE / THIRDLINE / FOURLINE / UPPERLINE

106.1 Action

Place le curseur en position 1 de la ligne spécifiée.

106.2 Syntaxe

UPPERLINE
LOWERLINE
THIRDLINE

FOURTHLINE

106.3 Voir Aussi

LCD, CLS, LOCATE… les programmes LCDxx.bas

107 LTRIM / TRIM / RTRIM

107.1 Action

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

107.2 Syntaxe

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

107.3 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
```

---

108 MAKEBCD / MAKEDEC

108.1 Action

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

108.2 Syntaxe

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

108.3 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

109 MAKEINT

109.1 Action

Réunit deux Bytes pour donner une INTEGER ou une WORD

109.2 Syntaxe

Var=(VarLSB, varMSB)

109.3 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

109.4 Voir Aussi

LOW HIGH

---

110 MAKEDEC

Voir MAKE BCD

---

111 MAX()

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

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

111.3 Remarques

111.4 var1 Variable qui prendra la valeur maximum.

111.5 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.

111.6

111.7 Voir Aussi
MIN()

---

112 MID

112.1 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.

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

112.3 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

---

113 MIN()

113.1 Action

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

---

114 Nbits()

114.1 Action

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

---

115 ON INTERRUPT

115.1 Action

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

115.2 Syntaxe

On INTERRUPT ETIQUETTE [NOSAVE]

115.3 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

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

---

116 ON VALUE

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

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

116.3 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
```

117 OPEN

117.1 Action
Ouvre un composant

117.2 Syntaxe

Open "device" for MODE as #canal

117.3 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)

117.4 Voir Aussi

Crystal, programme exemple Open.bas

---

118 OUT

118.1 Action

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

118.2 Syntaxe

OUT Adresse, Val

118.3 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.

118.4 Exemple :

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

118.5 Voir Aussi

INP, PEEK et le programme exemple PEEK.bas

---

119 PEEK

119.1 Action

Renvoie le contenu d'un registre.

119.2 Syntaxe

Var=PEEK(adresse)

119.3 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.

119.4 Voir Aussi

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

---

120 POKE

120.1 Action

Ecrit un Byte dans un registre interne.

120.2 Syntaxe

POKE Adresse, Valeur

120.3 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.

120.4 Voir Aussi

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

---

121 POPALL / PUSHALL

121.1 Action

Instruction de restauration et de sauvegarde des registres internes.

121.2 Syntaxe

POPALL (restauration)
PUSHALL (sauvegarde)

121.3 Remarques

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

---

122 POWERDOWN

122.1 Action

Met le µP en mode arrêt.

122.2 Syntaxe

POWERDOWN

122.3 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.

122.4 Voir Aussi
IDLE, POWERSAVE

---

123 POWERSAVE

123.1 Action
Met le µP en mode sommeil.

123.2 Syntaxe
POWERSAFE

123.3 Remarques
Seulement sur le 8535

123.4 Voir Aussi
IDLE, POWERDOWN

---

124 PRINT

124.1 Action
Envoie des données à la sortie RS232

124.2 Syntaxe
PRINT Var ; ["constant"]

124.3 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

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

125 PRINTBIN

125.1 Action
Envoie le contenu binaire d'une variable

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

125.3 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.

125.4 Voir Aussi
INPUTBIN

126 PSET

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

126.2 Syntaxe
PSET X, Y Valeur

126.3 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 !

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

---

127 PULSEIN

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

127.2 Syntaxe
PULSEIN var, PINX, PIN, état

127.3 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.

127.4 Voir Aussi
PULSEOUT

---

128 PULSEOUT

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

128.2 Syntaxe
PULSEOUT PORT, BROCHE, DUREE

128.3 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
```

128.4 Voir Aussi

PULSEIN

---

129 PUSHALL

Voir POPALL

---

130 Rc5send

130.1 Action

Pilote un port infrarouge au code RC5, RC6

130.2

130.3 Syntaxe

RC5send Togbit , Address , Command

130.4 Remarques

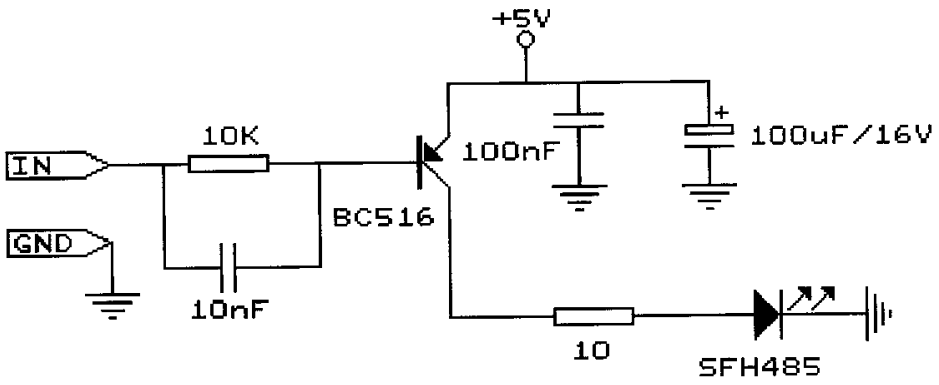
Togglebit construit le bit d'inversion 0 ou 32
Address The RC5 adresse
Command The RC5 commande.
La résistance doit être connectée à la broche OC1A (exemple portB.3 pour un 2313)
Contrôler l'adresse de ce port dans la datasheet du µP.
Beaucoup d'appareils audio ou vidéo sont équipés avec une télécommande 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 :



130.5 Voir Aussi

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

---

131 RC6SEND

Voir RC5SEND

---

132 READ / RESTORE

132.1 Action

Instruction de lecture des Data

132.2 Syntaxe

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

132.3 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

132.4 Voir Aussi

DATA, LOOKUP, Programme ReadData.bas

---

133 READEEPROM

133.1 Action

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

133.2 Syntaxe

READEEPROM var, adresse

133.3 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
```

133.4 Voir Aussi

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

---

134 READMAGCARD

134.1 Action

Lecture de données depuis une carte magnétique

134.2 Syntaxe

READMAGCARD var, Count, 5/7

134.3 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)

134.4 Voir Aussi

Le programme exemple : MAGCARD.bas

---

135 REM

135.1 Action

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

135.2 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

135.3 Remarques

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

---

136 RESET / SET

136.1 Action

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

136.2 Syntaxe

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

136.3 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
```

137 RESTORE

Voir Read

---

138 RETURN

138.1 Action

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

138.2 Syntaxe

RETURN

138.3 Remarques

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

138.4 Voir Aussi

GOSUB ON INTERRUPT

---

139 RIGHT

Voir LEFT

---

140 RND

140.1 Action

Renvoie un nombre aléatoire

140.2 Syntaxe

Var = RND (limit)

140.3 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.

140.4

---

141 ROTATE

141.1 Action

Fait "tourner" les bits d'une variable.

141.2 Syntaxe

Rotate Var, LEFT/RIGHT [,Shift]

141.3 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

141.4 Voir Aussi

SHIFT, SHIFTIN, SHIFTOUT programme exemple : ROTATE.bas

---

142 ROUND

Voir FIX

---

143 RTRIM

Voir LTRIM

---

144 SELECT-CASE-END-SELECT

144.1 Action

Exécute une ou plusieurs instructions suivant la valeur d'une expression ou d'une variabl

144.2 Syntaxe

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

144.3 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
```

144.4 Voir Aussi

IF THEN

---

145 SHIFT

145.1 Action

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

---------- ----------

145.2 Syntaxe

SHIFT Var, LEFT/RIGHT [,nombre]

145.3 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…

145.4 Voir Aussi

ROTATE, SHIFTIN, SHIFTOUT.

---

146 SHIFTCURSOR

146.1 Action

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

146.2 Syntaxe

SHIFTCURSOR LEFT/RIGHT

146.3 Voir Aussi

SHIFTLCD, LCD…

---

147 SHIFTIN /SHIFTOUT

147.1 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)

147.2 Syntaxe

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

147.3 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

147.4 Voir Aussi
Le programme exemple : Shift.bas

---

148 SHIFTLCD

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

148.2 Syntaxe
SHIFTLCD LEFT/RIGHT

148.3

148.4 Voir Aussi
SHIFTCURSOR

---

149 SHOWPIC, SHOWPICE

149.1 Action
Affiche un fichier BGF (Bascom Graphic File)sur un afficheur graphique. Si le fichier est en EEPROM on utilisera SHOWPICE

149.2 Syntaxe
SHOWPIC x, y , label
SHOWPICE x, y , label

149.3 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.
On peut utiliser plusieurs fichiers BGF, donc plusieurs directives $BGF .
SHOWPIC peut afficher un fichier BMP qui doit être converti au préalable en fichier BGF par l'outil "tools graphic converter", Outils→ convertisseur graphique

149.4 Voir Aussi

$BGF , CONFIG GRAPHLCD , exemples graphiques

---

150 SOUND

150.1 Action

Envoie des impulsions à une broche d'un port.

150.2 Syntaxe

Sound Broche, Duree, frequence

150.3 Remarques

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

Exemple :
Sound Portb.1, 10000, 10

---

151 SONYSEND

151.1 Action

Envoi les codes de télécommande IR type Sony®

151.2 Syntaxe

SONYSEND adresse [, bits]

151.3 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 !

151.4 Voir Aussi

CONFIG RC5 , GETRC5

151.5 Exemple :

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

---

152 SPACE

152.1 Action

Donne à une variable String la valeur de x espaces

152.2 Syntaxe

Var =SPACE(x)

152.3 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
```

152.4 Voir Aussi

SPC

---

153 SPC

153.1 Action
Imprime le nombre d'espaces spécifiés

153.2 Syntaxe
Print SPC(x)

153.3 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

153.4 Voir Aussi
SPACE

---

154 SPIIN / SPIOUT

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

154.2 Syntaxe
SPIIN var , Bytes
SPIOUT var , Bytes

154.3 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)

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

155 SPIINIT

155.1 Action
Initialise le bus SPI.

155.2 Syntaxe
SPIINIT

155.3 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.

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

---

156 SPIMOVE

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

156.2 Syntaxe
Var= SPIMOVE (Byte)

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

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

---

157 SPIOUT
Voir SPIIN

---

158 START

158.1 Action
Lance un composant

158.2 Syntaxe
START composant

158.3 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.

158.4 Voir Aussi
STOP et le programme exemple: ADC.bas

---

159 STCHECK

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

159.2 Syntaxe
STCHECK

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

---

160 STOP

160.1 Action

Arrête un composant
Arrête le programme

160.2 Syntaxe

STOP Composant
STOP

160.3 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.

160.4 Voir Aussi

START et le programme exemple ADC.bas

---

161 STR

161.1 Action

Renvoie une représentation String d'un nombre

161.2 Syntaxe

Var= Str(nombre)

161.3 Remarques

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

161.4 Voir Aussi

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

162 STRING

162.1 Action

Renvoie une String composée de n fois le caractère ASCII m

162.2 Syntaxe
Var = STRING(n,m)

162.3 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

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

---

163 SUB

Voir DECLARE SUB

---

164 SWAP

164.1 Action
Echange deux variables du même type.

164.2 Syntaxe
Swap var1,var2

164.3 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.

---

165 THIRDLINE

Voir FOURTHLINE

---

166 TIME$

Voir DATE$

167 TOGGLE

167.1 Action

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

167.2 Syntaxe

TOGGLE varbit

167.3 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.

167.4 Voir Aussi

CONFIG PORT

---

168 TRIM

Voir LTRIM

---

169 UCASE

169.1

Voir LCASE

---

170 UPPERLINE

Voir LOWERLINE

171 VAL

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

171.2 Syntaxe
Var = Val(s)

171.3 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
```

172 VARPTR

172.1 Action
Retrouve l'adresse mémoire d'une variable

172.2 Syntaxe
Var = VARPTR(var2)

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

173 Version ()

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

173.2 Syntaxe
Var = Version(frm)

173.3 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.

---

174 WAIT / WAITMS / WAITUS

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

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

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

Les durées sont approximatives. Les interruptions ralentissent ces valeurs.

174.4 Voir Aussi
DELAY

---

175 WAITMS

Voir WAIT

---

176 WAITKEY

176.1 Action

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

176.2 Syntaxe

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

176.3 Remarques

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

176.4 Voir Aussi

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

177 WHILE-WEND

177.1 Action

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

177.2 Syntaxe

WHILE condition
…instructions
…instructions
WEND

177.3 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.

177.4 Voir Aussi

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

178 WRITEEEPROM

178.1 Action

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

178.2 Syntaxe

WRITEEEPROM var, adresse

178.3 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.

178.4 Voir Aussi

READEEPROM et le programme exemple.EEPROM2.bas

Les Fonctions mathématiques

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.

Programmes exemples dans le dossier :MATH
Dans les exemples qui sont données π= Atn(1) * 4

---

1 ABS

1.1 Action

Retourne la valeur absolue d'un nombre signé

1.2 Syntaxe

Var=ABS(Var2)

1.3 Remarques

Var	Variable INTEGER ou LONG
Var2	Variable INTEGER ou LONG

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

---

2 ACOS

2.1 Action

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

2.2 Syntaxe

var = ACOS(x)

2.3 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

---

3 ASIN

3.1 Action

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

3.2 Syntaxe

var = ASIN(x)

3.3 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

---

4 ATN

4.1 Action

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

4.2 Syntaxe

var = ATN(X)

4.3 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

---

5 ATN2

5.1 Action

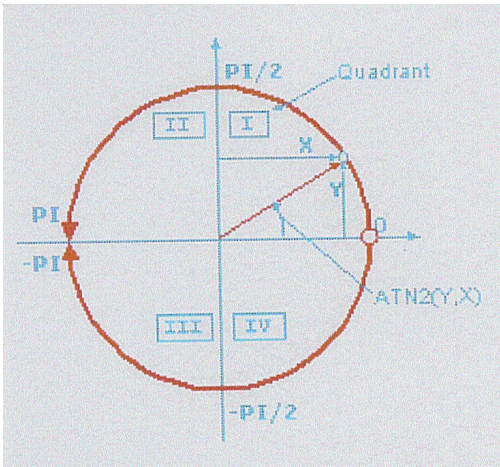
ATN2 est une fonction arc-tangente à 4 quadrants.
Tandis que ATN renvoie de -π/2 à π/2, l'ATN2 renvoie la valeur pour l'ensemble du cercle trigonométrique de -π à +π, le résultat dépend du ratio Y/X et des signes de X et Y.

5.2 Syntaxe

var = ATN2(x, y)

5.3 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°

---

6 COS

6.1 Action

Renvoie le cosinus d'une Single en radians.

6.2 Syntaxe

var = COS(x)

6.3 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

---

7 COSH

7.1 Action

Renvoie le cosinus hyperbolique d'une Single en radians.

7.2 Syntaxe

var = COSH(x)

7.3 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

---

8 DEG2RAD / RAD2DEG

8.1 Action

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

8.2 Syntaxe

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

8.3 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

9 EXP

9.1 Action

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

9.2 Syntaxe

Var=EXP(x)

9.3 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

10 LOG/ LOG10

10.1 Action

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

10.2 Syntaxe

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

10.3 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.

11 POWER

11.1 Action

Renvoie la puissance d'un nombre

11.2 Syntaxe

Var = POWER(X, PUISSANCE)

11.3 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.

---

12 SIN

12.1 Action

Renvoie le sinus d'une Single en radians.

12.2 Syntaxe

var = SIN(x)

12.3 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

---

13 SINH

13.1 Action

Renvoie le sinus hyperbolique d'une Single en radians.

13.2 Syntaxe

var = SINH(x)

13.3 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

---

14 SQR

14.1 Action

Renvoie la racine carrée d'un nombre

14.2 Syntaxe

var = SQR(x)

14.3 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.

---

15 TAN

15.1 Action

Renvoie la tangente d'une Single en radians.

15.2 Syntaxe

var = TAN(x)

15.3 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

---

16 TANH

16.1 Action

Renvoie la tangente hyperbolique d'une Single en radians.

16.2 Syntaxe

var = TANH(x)

16.3 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

```
'------------------------------------------------------------------------
'                          TEST_FPTRIG2.BAS
'          Demonstates FP trig library from Josef Franz Vögel
' The entire FP_TRIG.LIB is written by Josef Franz Vögel

'programme modifié  par Jean-Pierre DUVAL pour rentrer dans le cadre du
'didactitiel français
'on peut aller directement à une étiquette(label) ou supprimer le rem sur
'wait et stop à la fin de la fonction que l'on veut étudier

'if you want to see a function, go to the label, remove the rem from wait and stop
' compile(F7) and run(F2)
'------------------------------------------------------------------------------

$regfile = "8515def.dat"


Dim S1 As Single , S2 As Single , S3 As Single , S4 As Single , S5 As Single , S6 As Single
Dim Vcos As Single , Vsin As Single , Vtan As Single , Vatan As Single , S7 As Single
Dim Wi As Single , B1 As Byte
Dim Ms1 As Single
Dim X As Single , Y As Single , R As Single


Const Pi = 3.14159265358979 '(I'm addict about this number)

'calculate PI
Ms1 = Atn(1) * 4

'puissance instruction POWER----------------------------
'Testing_power:
Print "X =10.25    Y= 7.6"
X = 10.25 : Y = 7.6 : R = Power(x , Y)
Print " X puissance Y :" ; R
Print "attention valeur aprox!"
'Wait 1

' exponentielle et logarithme----------------------------
'Testing_exp_log:

Print "Test EXP and LOG pour X=5"
Print "x    exp(x)      log([exp(x)])    Erreur-abs    Erreur-rel"
Print "on retrouve le log à partir de l'exponentielle avec le calcul d'erreur"
X = 5

  S2 = Exp(x)
  S3 = Log(s2)
  S4 = S3 - X
  S5 = S4 \ X
  Print X ; "    " ; S2 ; "    " ; S3 ; "    " ; S4 ; "    " ; S5 ; " " ;

----------                                                    ----------
          ----------
```

```
Wait 1
Stop
' cos-- sin et tan--------------------------------------
'Testing_trig:
Print "Test COS, SIN and TAN pour un angle de 30°"
Print "Angle Degree   Angle Radiant        Cos        Sin        Tan"

' Valeur De L'angle :
 S1 = 30
   S2 = Deg2rad(s1)
   Vcos = Cos(s2)
   Vsin = Sin(s2)
   Vtan = Tan(s2)
   Print S1 ; "    " ; S2 ; "   " ; Vcos ; "   " ; Vsin ; "   " ; Vtan
'Wait 1
'Stop

'   arctangente-----------------------------------------------
'Testing_atan:
Print "Test de  Arctan"
Print "X    atan en Radian,    Degree"
S1 = 3584 / 1024
S2 = Atn(s1)
S3 = Rad2deg(s2)
Print S1 ; "     " ; S2 ; "     " ; S3
'Wait 1
'Stop

' test de la fonction INT et FRACT des single-------------------
'Testing_int_fract:
Print "Test Int et Fract de Single"
Print "Valeur          Int          Frac"
S2 = 7456.4658
   S3 = Int(s2)
   S4 = Frac(s2)
   Print S2 ; "      " ; S3 ; "      " ; S4
Print "valeur approx !"
'Wait 1
'Stop
' conversion deg2rad rad2deg-----------------------------
'Print "Test degree - radiant - degree pour un angle de 15°"
'Print "Degree   Radiant   Degree   Diff-abs   rel"
S1 = 15
  S2 = Deg2rad(s1)
  S3 = Rad2deg(s2)
  S4 = S3 - S1
  S5 = S4 \ S1
  Print S1 ; "   " ; S2 ; "   " ; S3 ; "   " ; S4 ; "   " ; S5
'Wait 1
'Stop
```

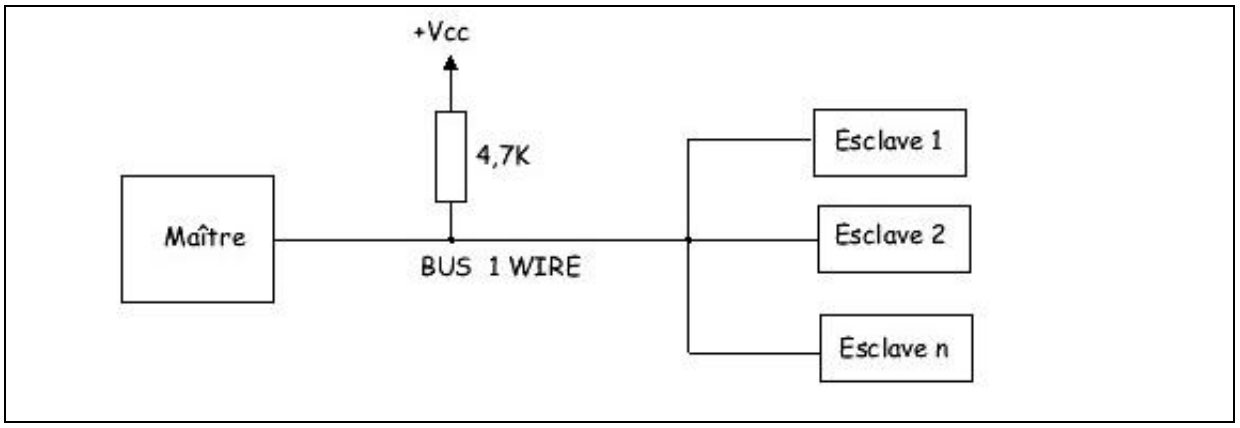
```
' fonctions cos, sin et tan hyperboliques----------------------------
'Testing_hyperbolicus:
Print "Test SINH, COSH et TANH pour une valeur de -12"
Print "X        sinh(x)         cosh(x)       tanh(x)"
 S1 = -12
  S3 = Sinh(s1)
  S2 = Cosh(s1)
  S4 = Tanh(s1)
  Print S1 ; "    " ; S3 ; "    " ; S2 ; "    " ; S4
'Wait 1
'Stop
'   log base 10----------------------------------------------------
Testing_log10:
Print "Test LOG10 de 14455"
Print "X        log10(x)"
X = 14455
S2 = Log10(x)
Print X ; "        " ; S2
'Wait 1
'Stop
'   MOD en vigule flottante-----------------------------------------
'test MOD on FP
S1 = 5
S2 = 3
S3 = S1 Mod S2
Print "Mod est le reste de la division"
Print "division 5 mod 3 = " ; S3
'Wait 1
'Stop
'   SQR-single------------------------------------------------------
Print "Test SQR-Single"
   S1 = 4.0625
   S2 = Sqr(s1)
   Print S1 ; " racine carrée " ; S2
'Wait 1
'Stop
'   Atn2------------------------------------------------------------
Test_atn2:
    Print "Test ATN2"
For S1 = -180 To 180 Step 30
    Print " degrée    Atn2     Deg of Atn2  "
   S2 = Deg2rad(s1)
   S3 = Sin(s2)
   S4 = Cos(s2)
   S5 = Atn2(s3 , S4)
   S6 = Rad2deg(s5)
   S7 = Round(s6)
   Print S1 ; "   " ; S5 ; " " ; S7
Next
Print "note: -180° is equivalent to +180°"
```

```
'Wait 1
'Stop
'   ASIN  ACOS----------------------------------------------------
Testing_asin_acos:
Print "Test ASIN, ACOS"
Print "X       asin(x)        acos(x)"
   S1 = 0.125
   'For S1 = -1.125 To 1.125 Step 0.0625
   S2 = Asin(s1)
   S3 = Acos(s1)
   Print S1 ; " " ; S2 ; " " ; S3
'Next
'Print "Note: > 1.0 and < -1.0 are invalid and shown here for error handling"
'Wait 1
'Stop

'Testing_shift:------------------------------------------------------
S1 = 12
For B1 = 1 To 20
   S2 = S1 : S3 = S1
   Shift S2 , Left , B1
   Shift S3 , Right , B1
   Print "S1        S2         S3"
   Print S1 ; " " ; S2 ; " " ; S3
Next

Print "End of testing"
End
```

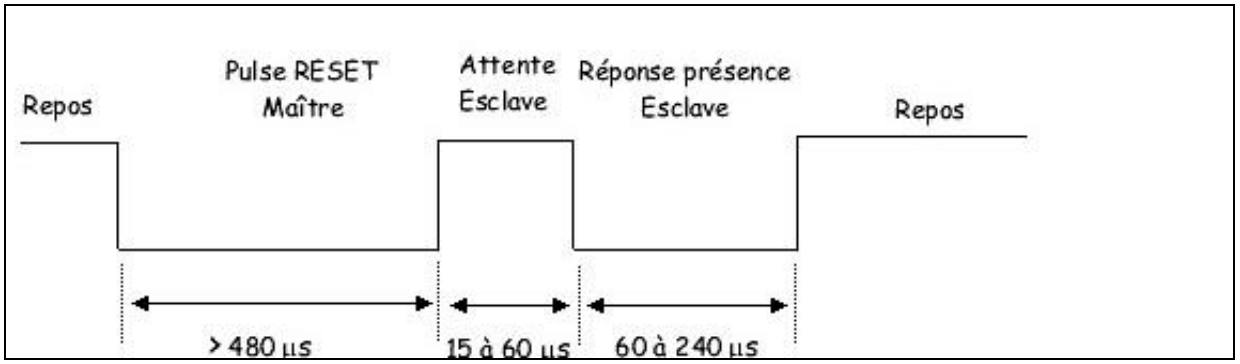
Le bus et les instructions 1WIRE

Sur l'excellent site, malheureusement consacré aux pics (rires)
http://daniel.menesplier.free.fr/index.htm
Il y deux explications sur les bus I2C et 1 Wire donc, et avec l'aimable autorisation de Mr Menesplier, nous allons l'utiliser pour la compréhension de ces bus. Tout ce qui suit est tiré de ce site.

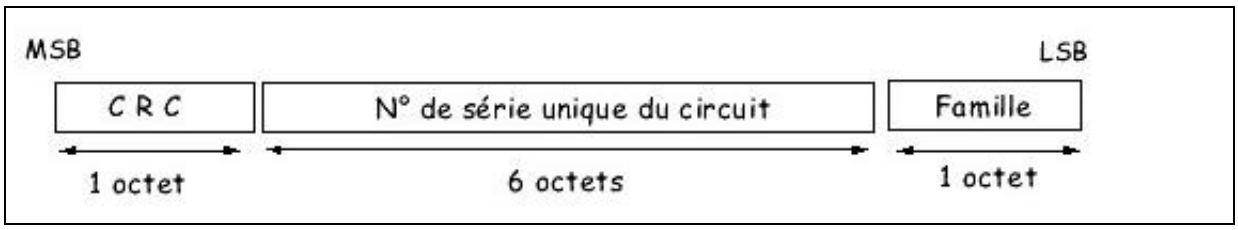
Le bus 1 WIRE de DALLAS, permet de connecter et de faire dialoguer entre eux des circuits sur un seul fil. (+ la masse)
Ce système de bus utilise un seul maître, qui pourra dialoguer avec un ou plusieurs esclaves.
Toutes les commandes et données sont envoyées avec le bit LSB en tête.
Le fil unique du bus doit être tiré au +Vcc par une résistance de 4,7K. L'état repos du bus est donc un état haut.



Si le bus est maintenu à l'état bas plus de 480 µs par le maître, tous les composants sur le bus sont remis à zéro. C'est le pulse d'initialisation ou de Reset.
Après un délai de 15 à 60 µs, le ou les esclaves raccordés, forcent le bus à l'état bas pendant 60 à 240 µs pour signaler leur présence.



Chaque circuit possède une adresse physique unique, gravée dans la puce à la fabrication.
Cette adresse est constituée de 64 bits soit 8 octets. Le premier octet détermine le type de famille auquel appartient le circuit. Les 6 octets suivants, constituent le code propre du circuit. Le dernier octet est le CRC. C'est un octet de contrôle calculé à partir des 56 bits précédents.

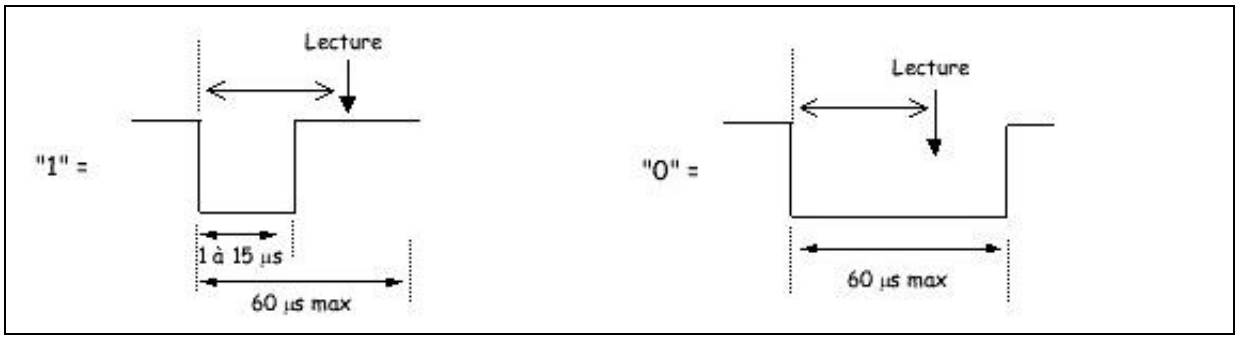


Toute transaction entre un maître et un ou plusieurs esclaves, débute par une initialisation, constituée par l'envoi du pulse de Reset par le maître.
Le maître doit ensuite envoyer une commande de type ROM qui est propre au protocole 1 Wire, et que tous les circuits de ce type vont reconnaître. Cela va permettre entre autre de sélectionner un circuit parmi les différents esclaves qui ont répondu présents au pulse de Reset.

Le dialogue et l'échange de données pourra ensuite commencer, entre le maître et l'esclave sélectionné.

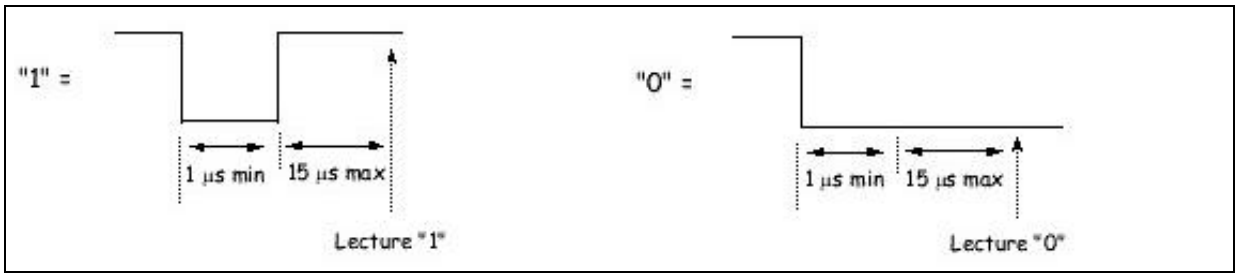
1 Emission d'un bit du maître vers l'esclave:

Le maître force le bus à "0" pendant 1 à 15 µs.
L'esclave va lire le bus entre 15 et 45 µs après le front descendant (valeur typique 30 µs).
Si on veut émettre un "1", il faut repasser le bus à "1" immédiatement, et
ne plus rien faire jusqu'à t = 60 µs.
Pour émettre un "0" il faut laisser le bus à "0" jusqu'à t = 60 µs, puis repasser le bus à "1".
La durée du bit est donc de 60 µs, ce qui donne un débit de 16 kbits/sec.



2 Réception d'un bit par le maître:

Le maître force le bus à "0" pendant au moins 1 µs. Si l'esclave veut émettre un "1", il laisse le bus libre donc tiré à "1".
Pour émettre un "0", l'esclave doit tirer le bus à "0" pendant 15 µs au minimum.
Le maître devra donc dans tous les cas lire le bus 15 µs maximum après avoir tiré le bus à "0" pendant 1 µs. L'état du bus donnera alors le bit transmis par l'esclave.



A partir de maintenant nous reprenons les explications de l'utilisation de BASCOM avec les outils 1WIRE.
On visitera les exemples dans le répertoire 1WIRE

2.1 Longueur du câble

Une longueur de 30 mètres ne pose pas de problème, ensuite (jusqu'à 300 voire 600 m ?) il faut prévoir un câble de bonne qualité et si possible type téléphone (paire twistée)

3 Avantages du 1 WIRE

⇒ 2 fils seulement
⇒ Un checksum de contrôle d'émission /réception des ordres ou des données (CRC)
⇒ Une distance assez confortable

4 Inconvénients

Technologie propriétaire, DALLAS uniquement
Lent = 14Kbits/S

---

5 Les outils BASCOM

Config 1wire
Pour configurer la broche utilisée.

---

6 1WIRECOUNT

6.1 Action

Cette instruction lit le nombre de circuits 1WIRE reliés au bus

6.2 Syntaxe

var2 = 1WIRECOUNT()
var2 = 1WIRECOUNT(port , pin)

var2 une variable WORD égale au nombre de circuits connecté sur le bus.
port le nom du port de la broche utilisée.
pin la broche du port de 0-7. peut être une constante ou une variable. La variable doit être du type WORD ou INTEGER.

6.3 Remarques

On utilisera cette fonction pour connaître le nombre de fois où la fonction 1Wsearchnext() doit être appelé pour connaître tous les ID' numbers (numéro d'identification) des circuits reliés au bus .cette fonction utilise 4byte 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)

---

7 1wread

7.1 Action

7.2 Cette instruction lit les Data du bus 1Wire.

7.3 Syntaxe

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

7.4 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)

---

8 1WRESET

8.1 Action

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

8.2 Syntaxe

8.3 1WRESET

8.4 1WRESET, Pinname, pin (dans le cas d'une application multiple broches 1WIRE)

8.5

8.6 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

---

9 1Wsearchfirst

9.1 Action

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

9.2 Syntaxe

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

9.3 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.

---

10 1Wsearchnext

10.1 Action

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

10.2 Syntaxe

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

10.3 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.

---

11 1Wverify

11.1 Action

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

11.2 Syntaxe

1Wverify ar(1)

11.3 Remarques

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

---

12 1wwrite

12.1 Action

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

12.2 Syntaxe

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

12.3 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)

Le bus et les instructions I2C

Nous revenons sur le site de Mr Menesplier, http://daniel.menesplier.free.fr/index.htm
Et nous exploitons la théorie du Bus

Il est constitué de deux fils. (+VCC et masse bien entendu) La transmission se fait en série de manière synchrone par rapport à une horloge. La fréquence max est de 100Khz (on peut pousser jusqu'à 400KHz suivant les circuits [NDLR]) soit 100 K/bits par seconde. La transmission d'un octet est suivie d'un 9eme bit d'accusé de réception : ACK.

SCL : Horloge série fournie par le maître. Le premier fil
SDA : Données dans les deux sens. Du maître vers l'esclave ou de l'esclave vers le maître. Le second fil.
Etat repos pour SDA et SCL = HIGH.
- Les data sur SDA ne changent que quand SCL est « LOW ». Si elles sont stables quand SCL est « HIGH » elles sont prises en compte.
- START : data sur SDA change (passage de 1 à 0) quand SCL est « HIGH ».
- STOP : data sur SDA change (passage de 0 à 1) quand SCL est « HIGH ».

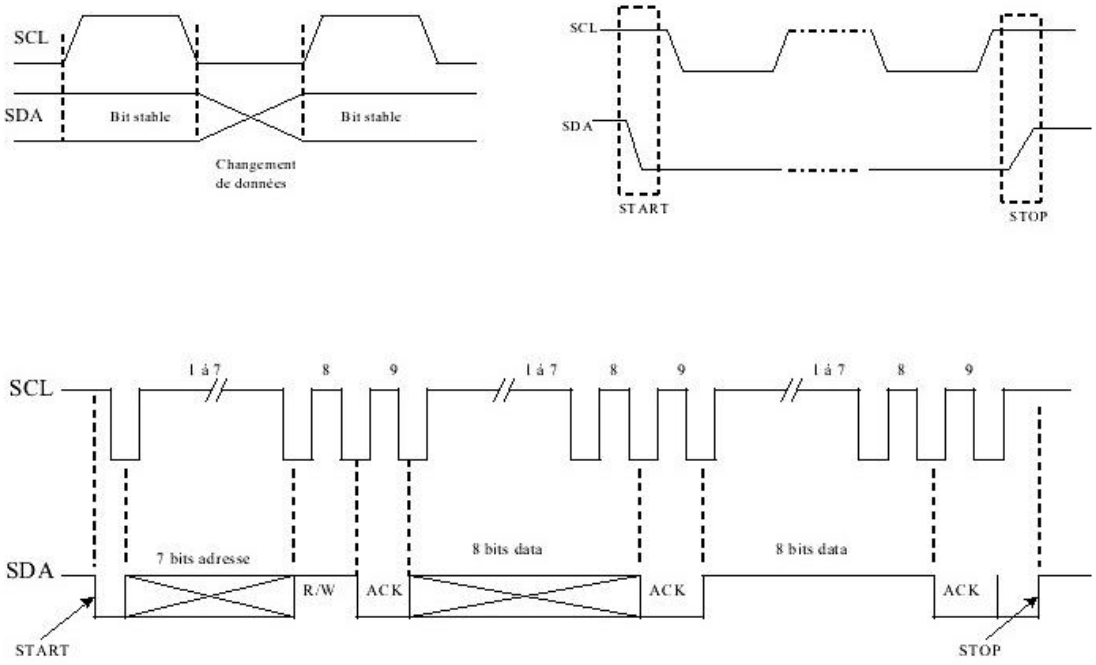
ACK : Le récepteur fait passer SDA de « High » à « Low » pendant la 9eme impulsion SCL.

1 TRANSMISSION :

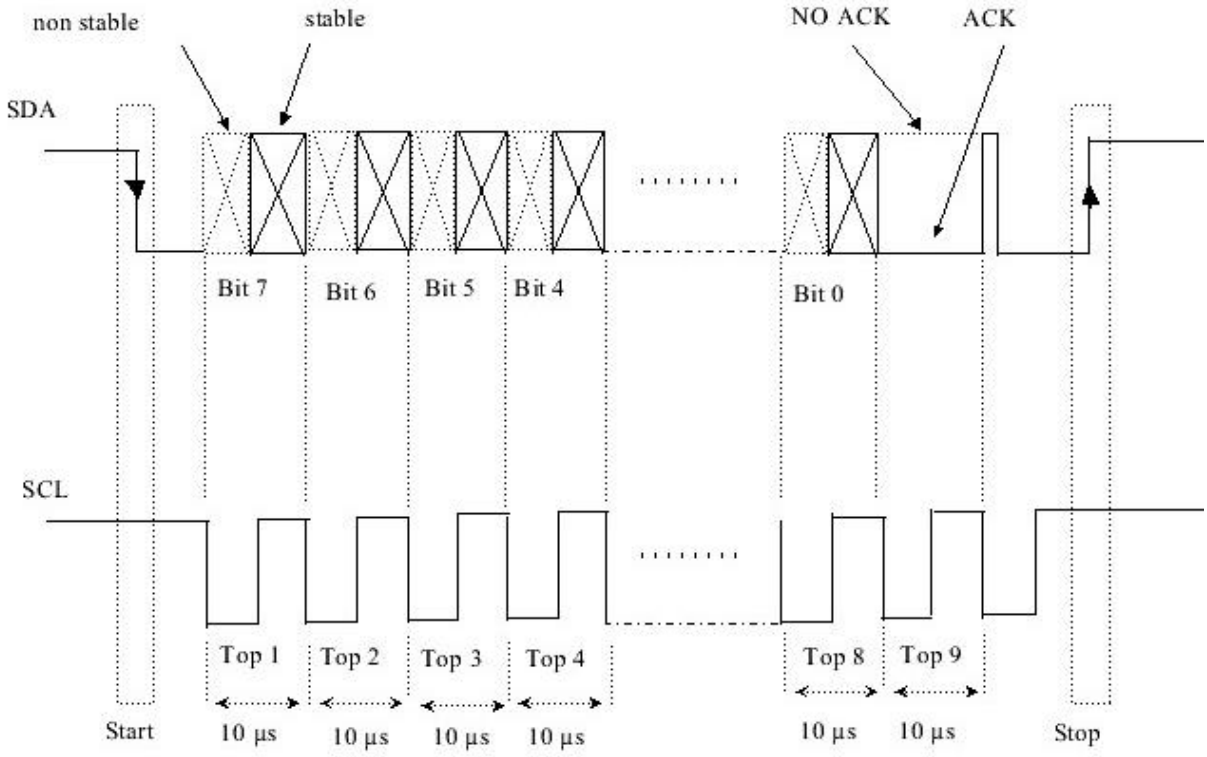
Le maître envoi le bit « START » et ensuite les 8 bits d'adressage sur SDA :
- Les 4 premiers bits déterminent le type de circuit et sont fixés par le fabriquant.
- Les 3 bits suivants dépendent, en général, du câblage de 3 pattes du CI.

Ces 7 bits constituent l'adresse.
- Le dernier bit est le R/W. Avec « 0 » on écrit dans le CI et un « 1 » permet la lecture.

L'esclave prend alors la ligne SDA qui est au repos donc à « 1 » et la fait passer à « 0 » pour un ACK.



2 EXEMPLE DE TRANSMISSION

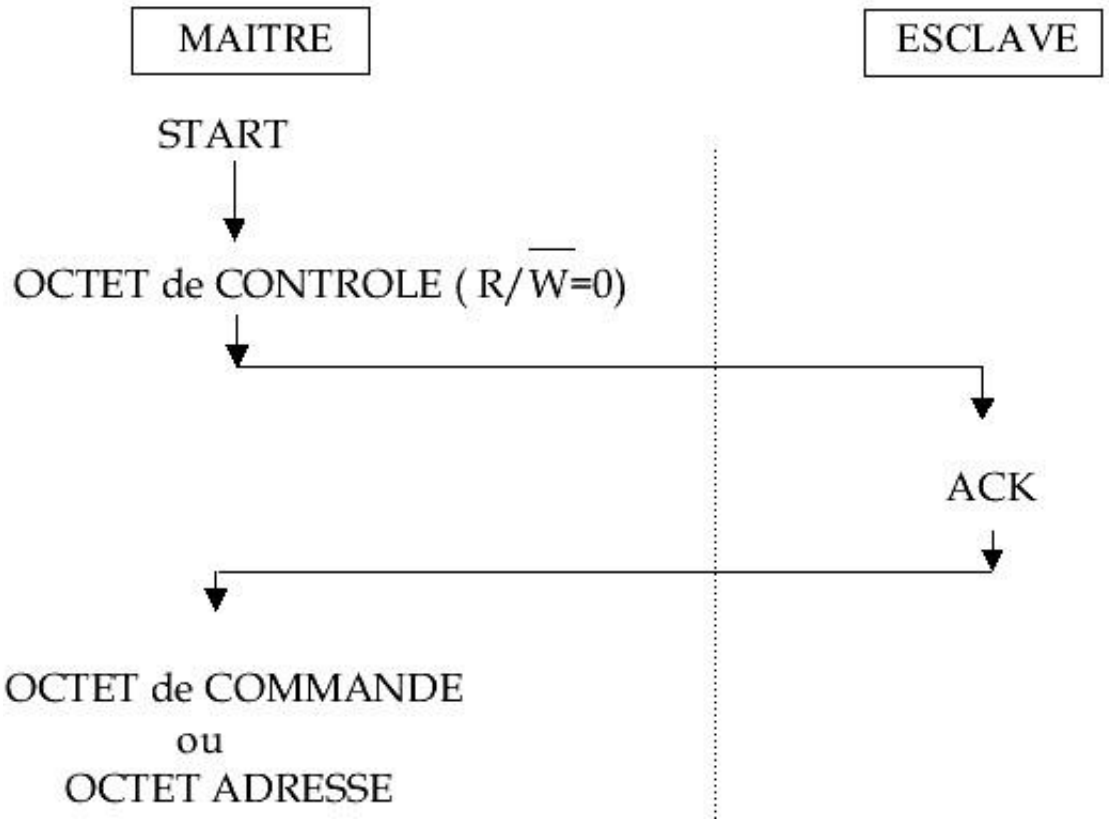


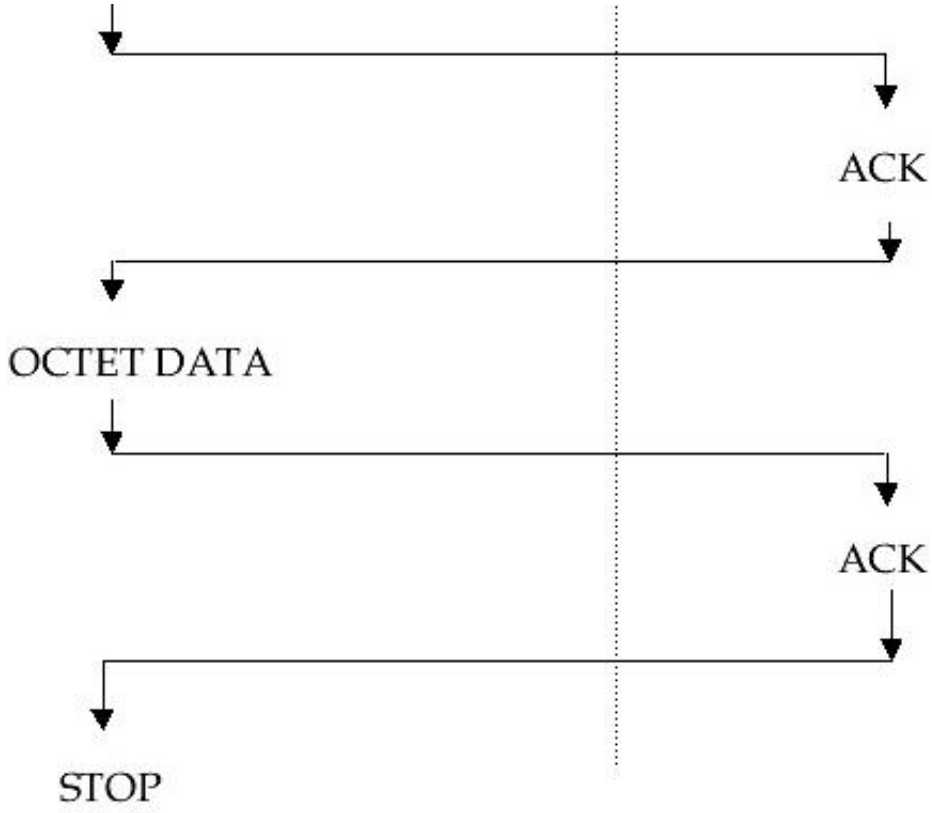
3 DEFINITIONS:

- Un circuit qui génère un message est un EMETTEUR, et le circuit qui le reçoit est un RECEPTEUR.
- Le circuit qui contrôle l'échange est le MAITRE et celui qui est contrôlé est l'ESCLAVE.
- Un ESCLAVE RECEPTEUR qui est adressé doit générer un ACK après réception de chaque octet.
- De même un MAITRE doit générer un ACK après réception de chaque octet venant de l'ESCLAVE.
- Un ACK est une mise à "0" de la ligne SDA pendant un pulse de la ligne SCL.
- Un MAITRE RECEPTEUR doit signaler la fin du transfert à l'ESCLAVE EMETTEUR en ne générant pas de ACK, c'est à dire en générant un NOACK qui est une mise à "1" de SDA pendant un pulse de SCL. Dans ce cas l'ESCLAVE doit laisser SDA libre pour permettre au MAITRE de générer ensuite un STOP

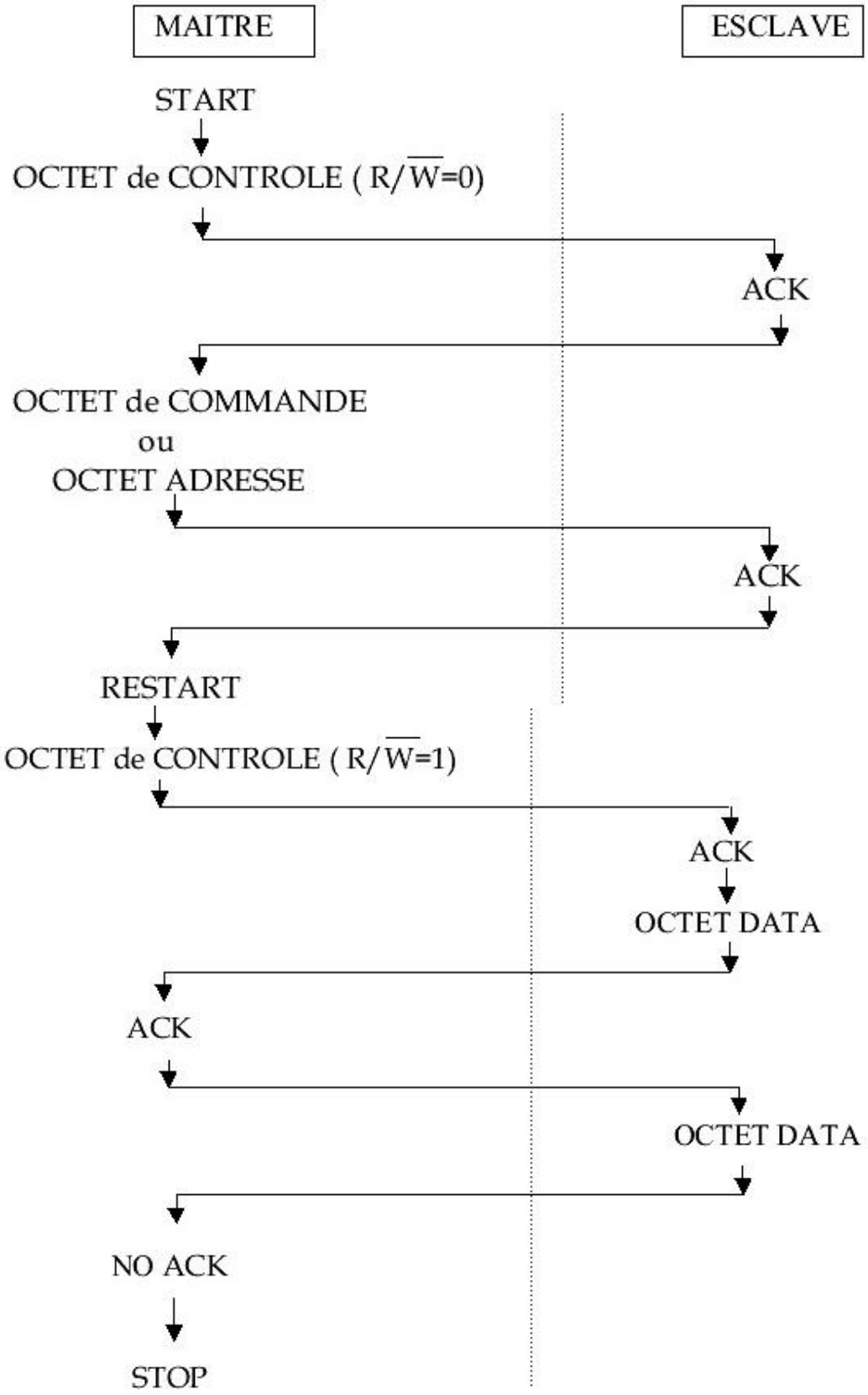
4 DIALOGUE :

ECRITURE du maître dans un esclave

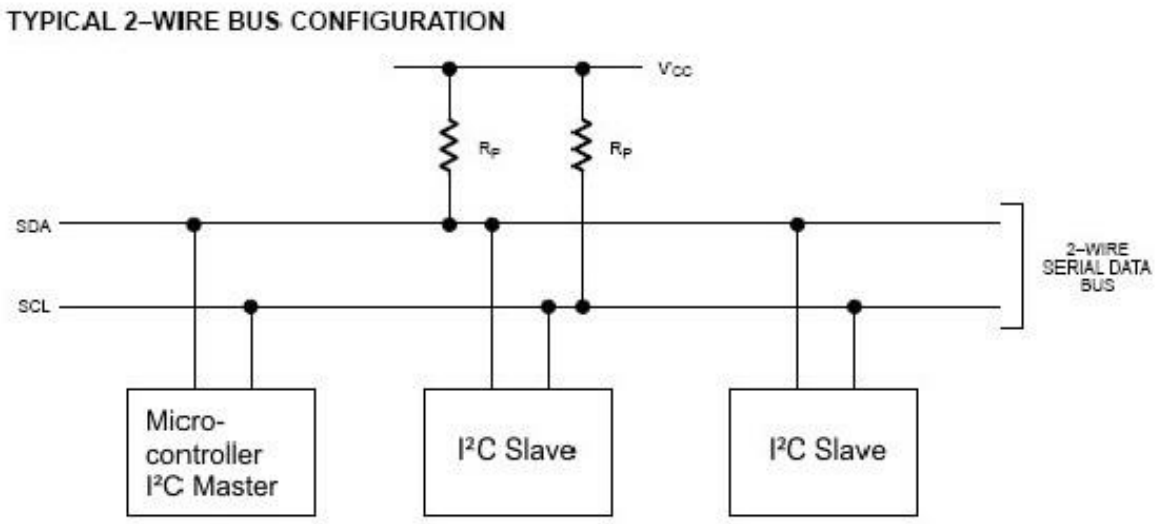




LECTURE de l'esclave par le maître


MAITRE
ESCLAVE
START
OCTET de CONTROLE (R/$\overline{W}$=0)
ACK
OCTET de COMMANDE
ou
OCTET ADRESSE
ACK
RESTART
OCTET de CONTROLE (R/$\overline{W}$=1)
ACK
OCTET DATA
ACK
OCTET DATA
NO ACK
STOP

Nous revenons maintenant sur les outils de BASCOM concernant ce bus.
Tout d'abord comment réaliser un bus I2C, VCC et GND sont commun à tous les circuits il peut y avoir des adaptations pour les bus mixes 3.3 et 5V
Le bus n'a pas besoin d'être blindé pour des petites distances. Si on utilise un câble en nappe, on pourra intercaler une masse entre chaque fil actif. La longueur du bus ne doit pas dépasser 2 ou 3 m surtout en milieu bruyant.



Rp sont des résistances « pull-up » d'environ 1K à 4.7KΩ suivant les circuits esclaves qui suivent.
Nous n'avons pas encore compris la différence entre TWI et I2C si un lecteur la connait nous lui serions reconnaissants de nous la transmettre.
Les µC Atmel , possèdent en général, des ports dédiés pour les fonctions I2C, mais il est possible de définir ses propres broches.

Config Sda = P o r t x . x	Configure la broche du port utilisé pour SDA
Config Scl = P o r t x . x	Idem pour SCL

---

5 I2CINIT

5.1 Action

Initialise les broches SCL et SDA .

5.2 Syntaxe

I2CINIT

5.3 Remarques

Par défauts les les bits du port Port et le DDR sont mis à 0 à l'initialisation du micro-contrôleur.
Avec I2CINIT on peut préparer les broches SDA et SCL pour le bus.

Voir le répertoire Samples\ I2C

---

6 I2CRECEIVE, I2CSEND

6.1 Action

Reçoit/ envoie des données depuis un composant I2C série.

6.2 Syntaxe

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

I2CSEND esclave, var , B2R

6.3 Remarques

Esclave Une 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 spécifier un nombre trop grand.
B2R Le nombre de Bytes à recevoir. Attention de ne pas spécifier un nombre trop grand.

On doit spécifier l'adresse du composant parce que le bit READ/WRITE est réglé par soft.

Quand une erreur intervient la variable interne ERR renvoie 1.
Cette méthode de communication est peu utilisée dans les exemples. Elle semble plus rapide.

```
Const Slave = &H40 'slave address of a PCF 8574 I/O IC
I2creceive Slave , X 'get the value
----------------------
For A = 1 To 10
        Ax(a) = préparation d'un tableau
Next
Bytes = 10
I2csend Slave , Ax(1) , Bytes
End
```

Voir Aussi

I2CSTART, I2CSTOP, I2CRBYTE, I2CWBYTE et les programmes I2C

---

7 I2CSTART, I2CSTOP, I2CRBYTE, I2CWBYTE

7.1 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

7.2 Syntaxe

I2CSTART
I2CSTOP
I2CRBYTE var, ack, nack
I2CWBYTE val

7.3 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.

7.4 Voir Aussi

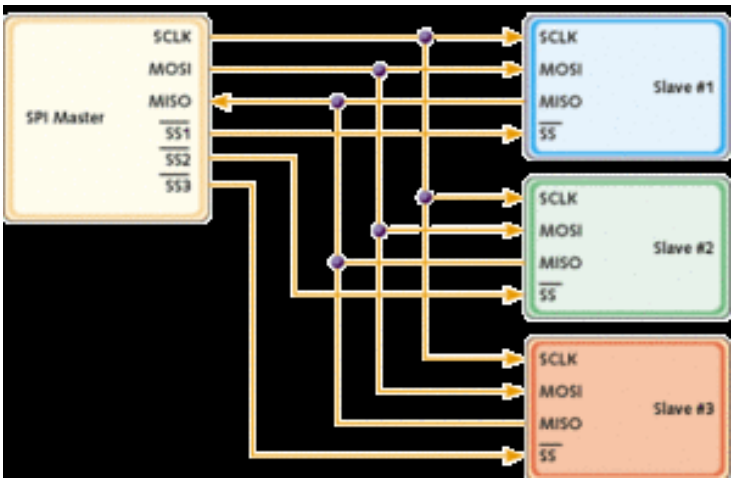
les programmes I2C

Le bus et les instructions SPI

J'ai de méme que précemment emprunté au blog :
http://embedded.over-blog.com/article-4591084.html une partie de la description du bus SPI.

SPI pour Serial Peripheral Interface
SPI est un bus de communication série synchrone standard établi par Motorola, supporté par de nombreux composants et repris par différentes marques. Le port d'interface SPI est disponible sur bon nombre de microprocesseurs et de microcontrôleurs : 68XX, 683XX, MCORE, MPC8260, DSP 56XXX de Motorola, mais aussi chez **Atmel**, Microchip, Texas Instruments etc ...

Tout comme l'I2C, le bus SPI est dédié pour établir une communication inter-composants, voir inter-cartes. Par ces caractéristiques le bus SPI est plus particulièrement dédié aux applications nécessitant des transferts de flots de données



Principe
Le bus SPI opère en mode "full duplex" (émission / réception simultanée) ; la méthode d'accès et du type maître / esclave et c'est toujours le maître qui a l'initiative des échanges : quand le maître sélectionne l'esclave et génère l'horloge, les données sont échangées dans les deux directions, simultanément. Le maître ne tient pas compte de la donnée reçue dans le cas d'un échange "écriture seule" ou il envoi un octet sans importance (0xFF) dans le cas d'un échange "lecture seule" ; la communication avec un esclave de type CODEC par exemple (coder - decoder), permet d'exploiter pleinement les capacités du bus SPI, sous un flot de données bidirectionnel.

L'interface SPI spécifie 4 signaux : SCLK (clock) ou horloge ; MOSI (master data output, slave data input) ou sortie donnée maître, entrée donnée esclave ; MISO (master data input, slave data output) ou entrée donnée maître, sortie donnée esclave ; SS (slave select) ou sélection esclave ; le schéma ci-dessous montre une connexion typique entre un maître et trois esclaves. SLCK, MOSI et MISO sont communs à tous les circuits, la sortie MISO des esclaves étant en haute impédance tant que le circuit n'est pas sélectionné par SS. MOSI envoi les données du maître vers l'esclave, MISO renvoi les données de l'esclave vers le maître. Un circuit esclave est sélectionné quand le maître valide le signal SS correspondant, par un port d'entrées / sorties : quand il plusieurs esclaves, le maître doit fournir autant de signaux SS qu'il y a d'esclaves.

---------- ----------

Trois paramètres principaux configure un port SPI : la fréquence d'horloge "bit" SCLK qui peut aller jusqu'à 10 MHz et qui est pré-divisée à partir de la fréquence de l'unité centrale ; la polarité de l'horloge, paramètre CPOL (Clock polarity) ; la phase de l'horloge, paramètre CPHA (Clock phase). Tous ces paramètres déterminent un diagramme de temps, pour chaque bit envoyé ou a chaque fois qu'un bit reçu est échantillonné. CPOL et CPHA ont deux état possible, donc cela fait 4 possibilités de configuration : chaque configuration étant incompatible entre elle, cela suppose que le maître doit avoir les mêmes paramètres que l'esclave avec lequel il dialogue, cela va de soit.

Au niveau supérieur
L'interface SPI ne dispose pas d'un mécanisme permettant de confirmer la réception des données par l'esclave. Il n'y a pas de protocole de communication particulier au niveau des trames envoyées ou reçues, et il n'y a pas non plus de contrôle de flux. Les esclaves sont vus comme des périphériques d'entrées / sorties par le maître, et il n'y a pas de protocole de haut niveau pour établir la communication entre le maître et l'esclave. C'est le travail du programmeur d'établir les règles de communication spécifiques pour chaque applications et / ou esclaves.

1.1 Retour à BASCOM

La première application de ce bus est l'utilisation de ce protocole pour programmer le microcontrôleur via le ISP[2].

En utilisation pour communiquer entre microcontrôleur, il faut noter que ce bus est très rapide, mais nécessite une broche à chaque esclave attaché.
Il faut noter que le SPI d'AVR peut être configuré comme étant maitre **et** esclave

Le mode dans lequel le microcontrôleur travaille est spécifié en ajustant le Bit du maitre (MSTR) dans le registre de protocole (SPCR)
La broche SS doit être prise en compte de façon particulière, elle est utilisée si différents composant sont reliés au bus SPI, l'Esclave autorisé à transmettre voie sa broche SS tirée vers le bas les autres doivent être haute. La table suivante résume l'état des broches SS

Mode	/SS Config	Niveau broche/SS	Description
Esclave	Toujours en input	haut	Esclave désactivé
	//	bas	Esclave activé
Maitre	Input	Haut	Maitre activé
	//	Bas	Maitre désactivé
	output	Haut	Maitre activé
	//	bas	//

⇒ Le Maitre est la partie active de ce système il délivre le signal d'horloge sur lequel les transmissions de data sont basées.

[2] [2] **In System Programing** le mode de programmation le plus populaire pour les AVR

⇒ Le ou les Esclaves ne sont pas capables de générer l'horloge et donc ne peut pas être actifs seul.

⇒ Le ou les Esclaves ne font qu'envoyer ou recevoir des data seulement si le Master envoie le signal d'horloge.

⇒ Le Maitre n'envoie le signal d'horloge que lorsqu'il envoie des données, ce qui signifie que le Maitre doit envoyer des données pour recevoir des données des esclaves.

Pour plus de théorie, se reporter à la notice anglaise. Et aux exemples dans le répertoire SPI.
Il est temps de passer aux instructions relatives à ce bus.

---

2 SPIINIT

2.1 Action

Initialise le bus SPI.

2.2 Syntaxe

SPIINIT

2.3 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.

---

3 SPIIN / SPIOUT

3.1 Action

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

3.2 Syntaxe

SPIIN var , Bytes
SPIOUT var , Bytes

3.3 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)

---

4 SPIMOVE

4.1 Action

Envoie et reçoit une valeur ou une variable du bus SPI, c'est l'instruction utilisé par le Maitre

4.2 Syntaxe

Var= SPIMOVE (Byte)

4.3 Remarques

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

Les routines, instructions et fonctions TCP/IP

Les routines TCP/IP peuvent être utilisées avec les composants ou modules W3100, IIM7000, IIM7010 . Voir le dossier exemple TCPIP

BASE64DEC , BASE64ENC , CLOSESOCKET, CONFIG TCIP , GETDSTIP ,GETDSTPORT, GETSOCKET , GETTCPREGS, , IP2STR , SETIPPROTOCOL ,SETTCP , SETTCPREGS , SOCKETCONNECT, SOCKETLISTEN SOCKETSTAT , TCPCHECKSUM, TCPREAD, TCPWRITE , TCPWRITESTR, UDPREAD , UDPWRITE ,UDPWRITESTR

---

1 Config TCPIP

Configure le circuit TCP/IP W3100A. Sur www.mcselec.com vous trouverez différents module tout prêt dont un qui utilise le port I2C (TWI)

1.1 Action

Configures the TCP/IP W3100A chip.

1.2 Syntaxe

CONFIG TCPIP = int , MAC = mac , IP = ip, SUBMASK = mask, GATEWAY = gateway, LOCALPORT= port, TX= tx, RX= rx , NOINIT= 0|1 , TWI=address , Clock = speed [, baseaddress = address] [,TimeOut=tmOut]

1.3 Remarques

Int L'interruption à utiliser comme INT0 or INT1, ne pas utiliser ces port à un autre usage. Pour le Easy TCP/IP PCB, utiliser INT0.

MAC L'adresse MAC que l'on assignera au W3100A, c'est un numéro unique qui l'identifie. On utilise une adresse différente pour chaque W3100A dans le réseau.

Exemple : 123.00.12.34.56.78

On doit définir 6 octets qui doivent être séparés par des points, ces octets doivent êtres spécifiés en notation décimale.

IP L'adresse IP que l'on assignera au W3100A, c'est un numéro unique qui l'identifie. On utilise une adresse differente pour chaque W3100A dans le réseau.

Pour votre réseau local LAN (Local Area Network), 192.168.0.10 peut être utilisé. 192.168.0.x est utilisé comme LAN puisque l'adresse n'est pas utilisé comme adresse internet.

SUBMASK Le submask que l'on assignera au W3100A.

Le submask est en général 255.255.255.0

GATEWAY L'adresse gateway que l'on assignera au W3100A.
L'adresse gateway peut être déterminée avec la commande IPCONFIG
C:\>ipconfig
Windows 2000 IP Configuration
Ethernet adapter Local Area Connection 2:
Connection-specific DNS Suffix . :
IP Address. . . . . : 192.168.0.3
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.0.1

Utiliser 192.168.0.1 dans ce cas.

LOCALPORT Une variable Word qui est assigné à LOCAL_PORT variable locale.
Voir aussi Getsocket.
On peut assigner une valeur de 5000.par défaut.

TX Un octet qui spécifie la taille du buffer de transmission du W3100A. Celui-ci possède 4 sockets.
Une valeur 00 qui assigne 1024 octets, 01 →2048 octets, 10 → , 11→8192 octets
C'est une notation binaire, par exemple pour assigner 2048 octets à chaque socket pour la transmission : TX = &B01010101
Le buffer de transmission ne peut pas exéder 8KB au total, on peut donc le découper en 4 partie de 2KB= 01
Si on veut utiliser les 8KB pour un socket on écrira TX = &B11. mais suel le Socket 0 pourra être utilisé.

RX Même procédure que pour TX
Consulter la documentation W3100A pour de plus amples informations.

Noinit (optionnel) mettre à 1 quand la configuration TCP, MAC, Subnetmask et GateWay doit être dynamique. Avec Noinit on devra passer par l'instruction SETTCP pour terminer le setup.

TWI L'adresse esclave du W3100A/NM7010. Quand on spécifie TWI on doit utiliser un micro qui possède cette interface Mega128, Mega88, Mega32…avec des R4.7k pull up

Clock La frequence d'horloge pour utiliser TWI W3100A. par defaut &H8000.
.
TimeOut On peut specifier un Timeout optionnel en utilisant les données UDP.
L'API Wiznet API attend le status CSEND . ce qui signifie qu'il bloquera votre application. On peut donc utiliser un timeout qui est décompté chaque fois que le status est. Quand on arrive à 0 on sort de la boucle.
Ce timeout n'a rien à voir avec les valeurs timeout des composants.

L'instruction CONFIG TCPIP ne peut être utilisé qu'une fois..
L'interruption doit être initialisé avant CONFIG TCPIP.
Après le CONFIG TCPIP, on peut immédiatement PINGuer the circuit. ([internet] Packet INternet Groper. « Faire un ping » consiste à envoyer une requête ICMP à une autre machine. Si

elle répond, c'est qu'on a des chances de pouvoir l'atteindre. Sinon, c'est qu'elle est en panne ou inaccessible. L'analogie avec le ping-pong est qu'au paquet ICMP (le ping), on attend une réponse (le pong). Mercie Jargon !

MCS Electronics fabrique un kit et un adpatateur Easy TCPIP

1.4 Exemple

Les exemples TCP-IP

Config Tcpip = Int0 , Mac = 00.00.12.34.56.78 , Ip = 192.168.0.8 , Submask = 255.255.255.0 , Gateway = 192.168.0.1 , Localport = 1000 , Tx = $55 , Rx = $55
`Now use PING at the command line to send a ping:
PING 192.168.0.8
Ou utiliser le easytcp pour pinguer le circuit.

Retour à la tête de chapitre

---

2 Base64DEC

2.1 Action

Pour Convertir des données Base-64 en valeur String.

2.2 Syntaxe

Result = BASE64DEC(source)

2.3 Remarques

Result Une variable string qui reçoit la valeur décodée

Source la string source qui est codé en Base-64

Base-64 n'est pas un protocole d'encryptage. Cela envoie les data au format ASCII sur 7 bit (0-127)
MIME, Les serveurs WEB et autres Internet client et serveurs utilisent le codage Base-64.
Cette fonction est un décodeur de la base-64, elle permet d'authentifier le WEB serveur.

2.4 Voir aussi

Les exemples TCP-IP

Retour à la tête de chapitre

---

3 BASE64ENC

3.1 Action

Pour Convertir des données string en Base-64

3.2 Syntaxe

Result = BASE64ENC (source)

3.3 Remarques

Result Une variable string qui reçoit la valeur à encoder

Source la string source qui est doit être codé en Base-64

On utilise cette instruction quand on veut envoyer des saisies avec POP3 par exemple.
La cible ajoutera un octet pour chaque 3 octets à convertir il faut donc dimensionner les variables en conséquence.
Retour à la tête de chapitre

---

4 CLOSESOCKET

4.1 Action

Ferme une connexion Soket. (on ne traduira pas Socket par chaussette à moins que vous y teniez vraiment !)

4.2 Syntaxe

CloseSocket socket [, prm]

4.3 Remarques

Socket Le numéro de socket que l'on veut fermé (entre 0 et 3)
Quand le Socket est déjà fermé rien ne se passe

Prm un paramètre optionnel pour changer le comportement de l'instruction

CloseSocket
Les valeurs suivantes sont possibles pour PRM
· 0 – le code se comporte comme si aucun parametre n'avait été ajusté.
· 1 – en fonctionnement normal, il y a un test pour vérifier si toutes les données écrites dans le circuit ont été transmises. Quand le bit 0 est mis à la valeur 1 ce test n'est pas effectué.
· 2 - – en fonctionnement normal, Il y a un test pour vérifier si le circuit est fermé après que la commande ait été envoyée au circuit. . Quand il n'est pas fermé, on ne peut pas réutiliser le socket. L'instruction bloque l'exécution du programme. Et on pourrait recontroler si la connexion a été fermée plus tard.
On peut combiner les valeurs 1 + 2 par exemple, il faut utiliser 1 avec précaution.

On doit fermer le socket quand on recoit le status SOCK_CLOSE_WAIT.

On peut aussi fermer le socket si c'est demandé par le protocole , on recevra alors un status SOCK_CLOSE_WAIT quand le serveur fermera la connexion.
Note :
Ce n'est pas nécessaire d'attendre un message SOCK_CLOSE_WAIT pour fermer la connexion.
Après la fermeture de la connexion il faut utiliser GetSocket pour pouvoir utiliser le socket fermé.

4.4 Voir Aussi

Les exemples TCP/IP
Retour à la tête de chapitre

---

5 GETDSTIP

5.1 Action

Revoie l'adresse IP du système Peer to Peer

5.2 Syntaxe

Result = GETDSTIP(socket)

5.3 Remarques

Result une variable LONG qui reçoit l'adresse IP(peer) ou de la destination (l'autre Peer).
Socket le numéro de prise « on parlera de Socket dans ce domaine IP (0-3)
Quand on est en mode serveur , il est intéressant de connaître l'adresse IP du client connecté ceci pour des raisons de sécurité, autorisation d'accés etc…
Le numéro IP MSB est stocké dans le LSB de la variable.

5.4 Voir aussi

Exemple partiel

```
Dim L as Long
L = Getdstip(i) ' store current IP number of socket i
```

Les programmes TCP-IP
Retour à la tête de chapitre

---

6 GETDSTPORT

6.1 Action

Renvoie le numéro de port de la liaison (PEER)

6.2 Syntaxe

Result = GETDSTPort(socket)

6.3 Remarques

Result une variable WORD qui reçoit le numéro de port du réseau (PEER) ou de du port de destination

Socket le numéro de Socket.

6.4 Voir aussi

Les programmes TCP-IP
Retour à la tête de chapitre

---

7 GETTCPREGS

7.1 Action

Lit une valeur de registre depuis le W3100A

7.2 Syntaxe

var = GETTCPREGS(address, bytes)

7.3 Remarques

Address l'adresse du registre du W3100A
bytes Le nombre d'octets à lire
La plupart des options W3100A sont compatible avec les instructions ou fonctions BASCOM Quand il est nécessaire de lire les registres du W3100A on peut utiliser la fonction GETTCPREGS
Il est important de spécifier la plus grande adresse parce que les registres doivent être lus en partant de cette adresse.

7.4 Voir aussi

Les programmes TCP-IP
Retour à la tête de chapitre

---

8 GETSOCKET

8.1 Action

Créer une socket pour les communications TCP/IP.

8.2 Syntaxe

Result = GETSOCKET(socket, mode, port, param)

8.3 Remarques

Result Un octet qui reçoit le numéro de Socket que l'on demande.Quand l'opération ne fonctionne pas Result= 255

Mode Le mode du Socket : on utilise sock_stream(1), sock_dgrm(2), sock_ipl_raw
(3), sock or macl_raw(4). Les modes sont définis avec des constantes.

Pour les communications TCP/IP on utilisera sock_stream ou sa valeur équivalente (1)

Pour les communications UDP sock_dgrm ou (2)

Port Ceci est le port local qui sera utilisé pour la communication, on peut utiliser n'importe quelle valeur, mais chaque Socket doit avoir son propre numéro de port local. Quand on utilise 0, la valeur de LOCAL_PORT sera utilisée. LOCAL_PORT est assigné dans CONFIG_TCPIP
Après
After l'affectation, LOCAL_PORT sera incrémenté par 1. Donc le moyen le plus simple est d'ajuster le Port Local avec Config TCPIP and utiliser 0 pour le port.

Param Paramètre optionnel. Valeur 0 par défaut.
128 : transmission/reception des message en mode UDP
64 : Utilise la valeur de registre avec une valeur de timeout designée
32 : Quand il n'est pas utiliser pas de ACK retardé
16: Quand il n'est pas « silly window syndrome » ?
Sin nous comprenons bien on peut additionner ces bits pour regler les paramétres

Consulter la documentation du W3100A documentation pour plus d'information.
Quand le Socket est initialisé, on utilise SocketConnect pour connecter le client, ou SocketListen pour servir comme serveur.

8.4 Voir aussi

Exemple partiel
I = Getsocket(0 , Sock_stream , 5000 , 0)' get a new socket

Les programmes TCP-IP
Retour à la tête de chapitre

---

9 IP2STR

9.1 Action

Convertit un nombre IP en représentation string.

9.2 Syntaxe

Var = IP2STR(num)

9.3 Remarques

Un nombre IP est représenté avec des points : 192.168.0.1.
Var La variable string qui reçoit le nombre IP

Num Une variable le numéro IP en format numérique.

9.4 Voir aussi

Les programmes TCP-IP
Retour à la tête de chapitre

---

10 SETIPPROTOCOL

10.1 Action

Configure le Socket en protocole RAW

10.2 Syntaxe

SETIPPROTOCOL socket, value

10.3 Remarques

Socket le numéro de socket. (0-3)
Value La valeur du protocole IP à régler.
Pour utiliser le IPL_RAW pour le W3100A's IPL_RAW Mode.
Voir l'exemple PING
Dans un premier temps on utilise :
Setipprotocol Idx , 1
Puis le socket est initialisé :
Idx = Getsocket(idx , 3 , 5000 , 0)

Malheureusement la data sheet du W3100A ne donne pas plus de détail !

10.4 Voir aussi

Les programmes TCP-IP
Retour à la tête de chapitre

---

11 SETTCP

11.1 Action

(Re) Configure le circuit TCP/IP W3100A.

11.2 Syntaxe

SETTCP MAC , IP , SUBMASK , GATEWAY

11.3 Remarques

MAC L'adresse MAC que l'on assignera au W3100A, c'est un numéro unique qui l'identifie. On utilise une adresse différente pour chaque W3100A dans le réseau.

Exemple : 123.00.12.34.56.78
On doit définir 6 octets qui doivent être séparés par des points, ces octets doivent êtres spécifiés en notation décimale.

IP L'adresse IP que l'on assignera au W3100A, c'est un numéro unique qui l'identifie. On utilise une adresse differente pour chaque W3100A dans le réseau.

Pour votre réseau local LAN (Local Area Network), 192.168.0.10 peut être utilisé. 192.168.0.x est utilisé comme LAN puisque l'adresse n'est pas utilisé comme adresse internet.

SUBMASK Le submask que l'on assignera au W3100A.
Le submask est en général 255.255.255.0

GATEWAY L'adresse gateway que l'on assignera au W3100A.
L'adresse gateway peut être déterminée avec la commande IPCONFIG
C:\>ipconfig
Windows 2000 IP Configuration
Ethernet adapter Local Area Connection 2:
Connection-specific DNS Suffix . :
IP Address. . . . . : 192.168.0.3
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.0.1

Utiliser 192.168.0.1 dans ce cas.

L'instruction CONFIG TCPIP ne peut être utilisé qu'une fois., on peut utiliser SETTCP à la place pour faire varier des paramètres, en plus SETTCP peut utiliser des constante ou des variables

11.4 Voir aussi

Les programmes TCP-IP
Retour à la tête de chapitre

---

12 SETTCPREGS

12.1 Action

Ecriture dans un registre du W3100A .

12.2 Syntaxe

SETTCPREGS address, var , bytes

12.3 Remarques

address l'adresse du registre du W3100A. Elle doit être la valeur du MSB, par exemple à l'adresse &H92 et &H93, le timeout est enregistré. On doit spécifier &H93 pour y accéder.

var La variable à écrire.
bytes le nombre d' octets à écrire.
La plupart des options W3100A sont compatible avec les instructions ou fonctions BASCOM Quand il est nécessaire d'écrire dans les registres du W3100A on peut utiliser la fonction SETTCPREGS
Il est important de spécifier la plus grande adresse parce que les registres doivent être lus en partant de cette adresse.

12.4 Voir aussi

Les programmes TCP-IP
Retour à la tête de chapitre

---

13 SOCKETCONNECT

13.1 Action

Etablir une connection à un serveur TCP/IP

13.2 Syntaxe

Result = SOCKETCONNECT(socket, IP, port)

13.3 Remarques

Result Un octet qui reçoit un 0 quand la connexion est réussie ou un 1 pour une erreur.

IP Le numéro IP du serveur où on veut se connecter. Ce peut être un nombre comme 192.168.0.2 ou une variable LONG à qui on a donné un nombre IP

Le LSB du long doit contenir le MSB du nombre IP

Port Le numéro du port où le serveur est connecté. Par exemple le protocole HTTP (web server) utilise le port 80.

13.4 Voir aussi

Les programmes TCP-IP
Retour à la tête de chapitre

---

14 SOCKETLISTEN

14.1 Action

Ouvrir un Socket du serveur en mode écoute.

14.2 Syntaxe

SOCKETLISTEN socket

14.3 Remarques

Socket Le numéro de socket number you want to open in the range of 0 -3.
Le socket écoutera le port spécifié avec la fonction GetSocket .
On peut écouter 4 sokets au maximum en même temps.
Après que la connexion soit fermer par le client ou le serveur, une nouvelle connexion doit être créé et l'instruction SOCKETLISTEN doit être utilisé.

14.4 Voir aussi

Les programmes TCP-IP
Retour à la tête de chapitre

---

15 SOCKETSTAT

15.1 Action

Pour connaître le status des Socket

15.2 Syntaxe

Result = SOCKETSTAT(socket , mode)

15.3 Remarques

Result Une Variable word qui reçoit le status.

Socket Le numéro de Socket où seront prises les informations nécessaires

Mode Un paramètre pour spécifier le type d'informations attendues
SEL_CONTROL ou 0 : retourne la valeur du status register

SEL_SEND or 1 : retourne le nombre d'octets qui peut être placés dans le buffer de transmission.
SEL_RECV or 2 : retourne le nombre d'octets qui peut sont placés dans le buffer de réception.

Quand on demande le mode 0 mode 0, un des octets suivant sera reçu :

Valeur	Etat	Description
0	SOCK_CLOSED	Connexion fermée
1	SOCK_ARP	Attente pour une réponse après transmission de ARP
2	SOCK_LISTEN	Attente pour setup de connexion au client en actionnant le mode Passif
3	SOCK_SYNSENT	Attente pour SYN,ACK après transmission de SYN pour le réglage de transmission en actionnant le mode Actif
4	SOCK_SYNSENT_ACK	Réglage de connexion après que SYN , ACK soit reçu dans le mode Actif.
5	SOCK_SYNRECV SYN,ACK	Est transmis après reception de SYN depuis le client en état d'écoute en mode Passif.
6	SOCK_ESTABLISHED	Le réglage de connexion est complet en mode actif ou passif.
7	SOCK_CLOSE_WAIT	La connexion étant terminée
8	SOCK_LAST_ACK	La connexion étant terminée
9	SOCK_FIN_WAIT1	La connexion étant terminée
10	SOCK_FIN_WAIT2	La connexion étant terminée
11	SOCK_CLOSING	La connexion étant terminée
12	SOCK_TIME_WAIT	La connexion étant terminée
13	SOCK_RESET	La connexion étant terminée après réception du packet depuis la paire (Peer)
14	SOCK_INIT	Initialisation du Socket
15	SOCK_UDP	Le socket est initialisé en mode UDP
16	SOCK_RAW	Le socket est initialisé en mode IP couche RAW
17	SOCK_UDP_ARP	Attente pour une réponse après transmission de ARP pour des transmissions
18	SOCK_UDP_DATA	Transmission des Data en cours dans le mode UDP RAW
19	SOCK_RAW_INIT	W3100A est initialisé en mode MAC couche RAW

Les fonctions de Socket sont aussi utilisées dans la librairie interne.

15.4 Voir aussi

Les programmes TCP-IP
Retour à la tête de chapitre

---

16 TCPCHECKSUM

16.1 Action

Pour retourner la TCP/IP checksum, appelée aussi Internet Checksum, ou IP Checksum.

16.2 Syntaxe

res= TCPCHECKSUM(buffer , bytes [,w1] [,w2])

16.3 Remarques

Res	Une variable Word qui reçoit la checksum TCP/IP du buffer.
Buffer	Une variable ou un tableau
Bytes	Le nombre d'octets à examiner.
w1,w2	Deux Word qui seront inclus dans la checksum

La checksum est utilisé dans beaucoup de protocole de communication c'est une méthode de contrôle de l'intégralité des transmissions. Dans les protocoles Internet (TCP, UDP, IP, ICMP ...)Une checksum spéciale est.
Normallement les données pour calculer la checksum sont stockés dans un tableau d'octets , mais dans certain cas comme TCP et UDP, une speudo entête (header) est ajoutée. WI et W2 sont alors utilisés dans ce cas .
Pour plus de détail sur le calcul voir l'aide en anglais.

16.4 Voir aussi

CRC8 , CRC16 , CRC32 , CHECKSUM
Les programmes TCP-IP et particulièrement le programme Ping_twi.bas
Retour à la tête de chapitre

---

17 TCPREAD

17.1 Action

Lire des données depuis une connexion Socket.

17.2 Syntaxe

Result = TCPREAD(socket , var, bytes)

17.3 Remarques

Result Un octet qui reçoit un 0 quand la connexion est réussie ou un 1 pour une erreur.
Quand il n'y a pas assez d'octets dans le buffer, la routine attend qu'il y a assez de données ou que le Socket soit fermé.

Socket Le numéro de socket. (0-3)

Var Le nom de la variable qui recevra les données depuis le socket.

Bytes Le nombre d'octets à lire.Seulement valide pour les variables non String.

Quand on utilise TCPread avec une variable string, la routine attendra CR + LF mais retournera les données sans CR + LF.
Pour les strings, la fonction ne réécrit pas la phrase. Par exemple la phrase est longue de 10 octets et la ligne que l'on reçoit est longue de 80 octets, on recevra seulement les 10 octets après que le CR+LF sera reçu. Aussi, si un CR+LF est attendu, il n'est pas nécessaire de spécifié le nombre d'octets à recevoir puisque la routine attend un CR+LF.

En revanche il faut le spécifié pour tous les autres types.

17.4 Voir aussi

Les programmes TCP-IP
Retour à la tête de chapitre

---

18 TCPWRITE

18.1 Action

Ecrire des données sur une connexion Socket.

18.2 Syntaxe

Result = TCPWRITE(socket , var , bytes)
Result = TCPWRITE(socket , EPROM, address , bytes)

18.3 Remarques

Result Une variable Word qui reçoit le nombre d'octets en cours d'écriture dans le Socket. Quand le buffer est suffisamment grand pour recevoir le nombre d'octets le résultat sera le mêm que BYTES en revanche quand il n'y a pas assez de place seul le nombre de byte écrit sera pris en compte. Quand il n'y a pas ou plus de place un 0 est renvoyé.

Socket Le numéro de socket. (0-3)

Var Une String constante comme "test" ou une variable.
Quand on envoie une constante le nombre d'octets ne dois pas être spécifié.

Bytes Une variable word ou une constante numérique constant qui spécifie le nombre d'octet qui seront envoyés
Address L'adresse des données dans l'EEPROM interne du circuit, on a besoin de spécifié EPROM aussi dans ce cas.
EPROM Une indicatio pour la compilation, ainsi les donné sontenvoyé à partir de l'EEPROM

18.4 Voir aussi

Les programmes TCP-IP
Retour à la tête de chapitre

---

19 TCPWRITESTR

19.1 Action

Pour envoyer une String à une connexion socket ouverte. La fonction TCPwriteStr est une variante de la fonction TCPwrite, elle l'utilise pour envoyer des données String.

19.2 Syntaxe

Result = TCPWRITESTR(socket , var , param)

19.3 Remarques

Var La variable string
Param Un paramètre qui doit être 0 pour envoyer la String seulement ou 255 pour ajouter un CR+LF additionnel. Beaucoup de protocole attendent ce CR+LF

19.4 Voir aussi

Les programmes TCP-IP
Retour à la tête de chapitre

20 UDPREAD

20.1 Action

Pour lire des données via un UDP protocole

20.2 Syntaxe

Result = UDPREAD(socket , var, bytes)

20.3 Remarques

Result Un octet qui reçoit un 0 quand la connexion est réussie ou un 1 pour une erreur.
Quand il n'y a pas assez d'octets dans le buffer, la routine attend qu'il y a assez de données ou que le Socket soit fermé.

Socket Le numéro de socket. (0-3)

Var Le nom de la variable qui recevra les données depuis le socket.

Bytes Le nombre d'octets à lire. Seulement valide pour les variables non String.

La lecture de Stings n'est pas supporté par UDP.
Quand on à besoin de lire des strings on peut utiliser l'option OVERLAY de DIM.
Bien respecter la méthode expliquée dans l'exemple UDPtest .bas et dans la documentation anglaise.

20.4 Voir aussi

Les programmes TCP-IP
Retour à la tête de chapitre

---

21 UDPWRITE

21.1 Action

Pour écrire des données dans un Socket en suivant le protocole UDP.

21.2 Syntaxe

Result = UDPwrite(IP, port, socket , var , bytes)
Result = UDPwrite(IP, port, socket , EPROM, address , bytes)

21.3 Remarques

Le fonctionnement est proche de TCPwrite, voir l'exemple UPDtest.bas UDP est une connexion sans protocole, on doit spécifier l'adresse IP et le numéro de port. UDP ne demande qu'un Socket ouvert. Il n'y a pas besoin de Connect ou Clode.

21.4 Voir aussi

Les programmes TCP-IP
Retour à la tête de chapitre

---

22 UDPWRITESTR

22.1 Action

Pour envoyer une String à une connexion socket ouverte. La fonction UDPwriteStr est une variante de la fonction UDPwrite, elle l'utilise pour envoyer des données String.
Cette fonction est très proche de la fonction TCPwritestr

22.2 Voir aussi

Les programmes TCP-IP
Retour à la tête de chapitre
