Manuel d'utilisation EasyBuilder 8000

-

pour les MMIs série 8000

Siège social KEP France :
Z.A. Belle Aire
3 rue Vasco de Gama
17440 AYTRE
Tél : 05 46 07 44 40
Fax : 05 46 07 44 45
Site internet : www.kepfrance.fr
e-mail :
service commercial : commercial@kepfrance.fr
service technique : hotline@kepfrance.fr

Chapter 18 Macro

Sont décrites ici les synthaxes et les méthodes de programmation pour les scripts de macro.

Description des macros

1 Constantes et Variables

a Constantes

1. constante décimale
2. constante hexadécimale
3. code ASCII (constante de caractère)
4. booléen : VRAI (différent de zéro), FAUX (zéro)

b Variables

1. Règle de nommage
   Le nom d'une variable doit commencer par une lettre de l'alphabet et ne peut pas dépasser 32 caractères.
2. Type de variables
   char. Caractère (8 bits)
   bool. Booléen (1 bit)
   short entier (16 bits)
   int double mot (32 bits)
   float flottant (32 bits)

c Opérateur

1. Affectation : =
2. Opérations arithmétiques
   Addition : +
   Soustraction : -
   Multiplication : *
   Division : /
   Modulo : %
3. Comparaison
   strictement inférieur : <
   inférieur ou égal : <=
   strictement supérieur : >
   supérieur ou égal : >=
   Egal : ==
   différent : <>
4. Logique
   condition ET : and
   condition OU : or
   condition OU EXCLUSIF : xor
   booléen NON : not
5. décalage et opération sur les bits
   décalage à gauche : <<
   décalage à droite : >>
   ET : &
   OU : |
   OU Exclusif : ^
   complement à deux : ~

2 Priorité des opérations

L'ordre d'exécution de ces opérations se fait selon les critères de priorité suivants :

a Priorité de gauche à droite dans le cas où on a le même opérateur.
Arithmétique : ^ □ *, / □ mod □ +, -
Décalage : de gauche à droite
Comparaison : de gauche à droite
Logique : Not □ And □ Or □ Xor

b les opérations arithmétique ont la priorité sur les décalages
Les décalages ont la priorité sur les comparaisons
Les logiques ont la priorité sur les affectations

3 Tableau

seuls les tableaux à une dimension sont supportées.
Déclaration : NomTableau[taille]
La taille est un entier de 0 à 4 294 967 295
Valeur minimale de l'index : 0
Valeur maximale de l'index : Taille du tableau – 1

Exemple Tab[100]
Indice minimal : 0, maximal 99

4 Expression

a les opérations peuvent se faire sur
1. constantes
2. variables
3. tableau
4. fonction

b expression

une expression est une combinaison d'opération et d'objets

5 Etat

a déclaration
1. type nom : défini une variable
   ex: short a : défini la variable a de type entier
2. type nom[constante] : défini un tableau
   ex: short a[10] : défini un tableau de 10 entiers.

Affectation : variable=valeur
exemple : a=2

b logique et embranchement

une ligne

```
if condition then
    [expression]
end if
```

exemple :

```
if a==2 then
        b=1
else
        b=2
end if
```

bloc

```
if condition then
        [expression]
[else [if condition -n then
        [expression else if]
[else
        [expression else]
]]
end if

exemple :
if a==2 then
        b=1
else if a==3
        b=2
else
        b=3
end if
```

Description

Condition	Obligatoire. Le résultat de la condition doit être FAUX (0) ou VRAI (différent de 0)
Expression	Optionne. Cette partie est exécuté lorsque la condition précédente est VRAI
Condition – n	Optionnelle
Expression else if	Optionnelle.
Expression else	Optionnelle.

Boucle

1. boucle pour

à utiliser lorsque le nombre de tours de boucle est connu.

```
For compteur=début to fin [step pas]
        expression
next [compteur]

exemple :
pour a=0 to 10 step 2
        b=a
next a
```

Description

Compteur	Indice de boucle
Début	Valeur initiale du compteur
Fin	Valeur finale du compteur
Pas	Optionnel, pas d'incrémentation du compteur, par défaut 1.
Expression	Code qui sera exécuté à chaque tour de boucle

Note : pour décrémenter l'indice au lieu de l'incrémenter, remplacer le mot "to" par "down"

2. boucle tant que

à utiliser lorsque l'on ne connait pas le nombre de tour de boucle à faire, la sortie de la boucle se fait selon une condition.

```
While condition
	expression
wend
```

exemple :

```
while a==2
	b=b+1
	GetData(a, "local HMI", LB, 5, 1)
wend
```

Description

Condition	Obligatoire, tant qu'elle est VRAI la boucle continue.
Expression	Code qui sera exécuté à chaque tour de boucle

3. break

à utiliser pour dans une boucle pour quitter la boucle

4. continue

à utiliser pour dans une boucle pour passer à l'itération suivante

5. return

quitter la méthode courante

mots réservés :

les mots suivants sont réservés au système, ils ne peuvent pas être utilisés pour le nom d'un variable, une fonction, un tableau ...

+, -, *, /, ^, mod, >=, >, <, <=, <>, ==, and, or, xor, not, <<, >>, =, &, |, ~, if, then, else, endif, select, case, for, to, down, step, next, while, wend, break, continue, return.

Exemple de code source

1: boucle FOR, décallage et fonctions arithmérique

```
macro_command main()
int a[10], b[10], i
b[0]= (400 + 400 << 2) / 401
b[1]= 22 *2 - 30 % 7
b[2]= 111 >> 2
b[3]= 403 > 9 + 3 >= 9 + 3 < 4 + 3 <= 8 + 8 == 8
b[4]= not 8 + 1 and 2 + 1 or 0 + 1 xor 2
b[5]= 405 and 3 and not 0
b[6]= 8 & 4 + 4 & 4 + 8 | 4 + 8 ^ 4
b[7]= 6 – (~4)
b[8]= 0x11
b[9]= 409
for i = 0 to 4 step 1
if (a[0] == 400) then
GetData(a[0],”Device 1”, 3x, 0,9)
GetData(b[0],”Device 1”, 3x, 11,10)
end If
next i
end macro_command
```

2: while, if, break

```
macro_command main()
int b[10], i
i = 5
while i == 5 - 20 % 3
GetData(b[1], ”Device 1”, 3x, 11, 1)
if b[1] == 100 then
break
end if
wend
end macro_command
```

3: Variables globales et sous fonction

```
char g
sub int fun(int j, int k)
int y
SetData(j, “Local HMI”, LB, 14, 1)
GetData(y, “Local HMI”, LB, 15, 1)
g = y
return y
end Sub
macro_command main()
int a, b, i
a = 2
b = 3
i = fun(a, b)
SetData(i, “Local HMI”, LB, 16, 1)
end macro_command
```

4. If

```
macro_command main()
int k[10], j
for j = 0 to 10
k[j] = j
next j
if k[0] == 0 then
SetData(k[1], “Device 1”, 3x, 0, 1)
end if
if k[0] == 0 then
SetData(k[1], “Device 1”, 3x, 0, 1)
else
SetData(k[2], “Device 1”, 3x, 0, 1)
end if
if k[0] == 0 then
SetData(k[1], “Device 1”, 3x, 1, 1)
else if k[2] == 1 then
SetData(k[3], “Device 1”, 3x, 2, 1)
end If
if k[0] == 0 then
SetData(k[1], “Device 1”, 3x, 3, 1)
else if k[2] == 2 then
SetData(k[3], “Device 1”, 3x, 4, 1)
else
SetData(k[4], 3x_BIN, 5, 1)
end If
end macro_command
```

5. boucle while

```
macro_command main()
char i = 0
int a[13], b[14], c = 4848
b[0] = 13
while b[0]
a[i] = 20 + i * 10
if a[i] == 120 then
c =200
break
end if
i = i + 1
wend
SetData(c, “Device 1”, 3x, 2, 1)
end macro_command
```

6. break et continue

```
macro_command main()
char i = 0
int a[13], b[14], c = 4848
b[0] = 13
while b[0]
a[i] = 20 + i * 10
if a[i] == 120 then
c =200
i = i + 1
continue
```

```
end if
i = i + 1
if c == 200 then
SetData(c, “Device 1”, 3x, 2, 1)
break
end if
wend
end macro_command
```

7. tableau

```
macro_command main()
int a[25], b[25], i
b[0] = 13
for i = 0 to b[0] step 1
a[i] = 20 + i * 10
next i
SetData(a[0], “Device 1”, 3x, 0, 13)
end macro_command
```