9/8.7

Défilement d'un message alphanumérique sur l'écran

Ce programme va être présenté dans deux versions : Basic et Assembleur. Il vous sera également possible d'entrer les données hexadécimales correspondant aux codes opératoires sous Basic.

Version Basic

Un message alphanumérique quelconque peut être affiché n'importe où sur l'écran, en mode 0, 1 ou 2. Pour cela, il faut placer le message à afficher dans la variable M$. Les coordonnées absolues d'affichage du premier caractère sur l'écran sont placées dans les variables X (abscisse) et Y (ordonnée). La vitesse de défilement est enfin indiquée dans la variable V. L'affichage est déclenché en faisant un « GOSUB 1000 », comme le montre l'exemple suivant :

```
10 MODE 2

20 V=80

30 M$="Texte quelconque pour tester la routine "

40 X=10:Y=15

50 CLS

60 GOSUB 1000

70 END
```

Le décalage se fait en trois étapes :
— isoler la première lettre de la chaîne ;
— copier les caractères suivants dans la chaîne de départ ;
— placer le caractère isolé en fin de chaîne.
Cette manipulation est faite lignes 1100 et 1110.

La boucle d'attente paramétrée se fait ligne 1120. Elle consiste à faire une boucle FOR NEXT dont la longueur dépend de la vitesse définie dans la variable V.

Le listing du programme est le suivant :

```
1000 '-------------------------------------------------------------------------
1010 '-                    MESSAGE DEFILANT                                   -
1020 '-------------------------------------------------------------------------
1030 '-Entree:V  = Vitesse (entre 0 et 100)                                    -
1040 '-        M$ = Message                                                    -
1050 '-        X  = Abcisse 1ere lettre du message                             -
1060 '-        Y  = Ordonnee 1ere lettre du message                            -
1070 '-------------------------------------------------------------------------
1080 LOCATE X,Y
1090 PRINT M$
1100 I$=MID$(M$,1,1):J$=MID$(M$,2,LEN(M$)-1)
1110 M$=J$+I$
1120 FOR I=1 TO (100-V)*3:NEXT I
1130 A$=INKEY$
1140 IF A$="" THEN 1080
1150 RETURN
```

Version Assembleur

Même si la vitesse d'exécution d'un programme écrit en Assembleur n'est plus à démontrer, vous serez peut-être surpris par le programme qui suit.

Il reprend les concepts exposés pour le programme de défilement Basic. Trois routines du FIRMWARE sont utilisées pour faciliter l'écriture du programme. Il s'agit de :

— TXT SET CURSOR (#BB75) qui positionne le curseur à un endroit quelconque de l'écran ;

— TXT OUTPUT qui affiche un caractère à la position courante du curseur ;

— KM READ KEY qui lit le code d'une éventuelle touche actionnée sur le clavier.

Le programme se décompose en cinq parties :

— recherche de la longueur du texte à afficher. Précisons à ce sujet que le texte à afficher doit être suivi du caractère terminateur « nul » de code ASCII 0 ;

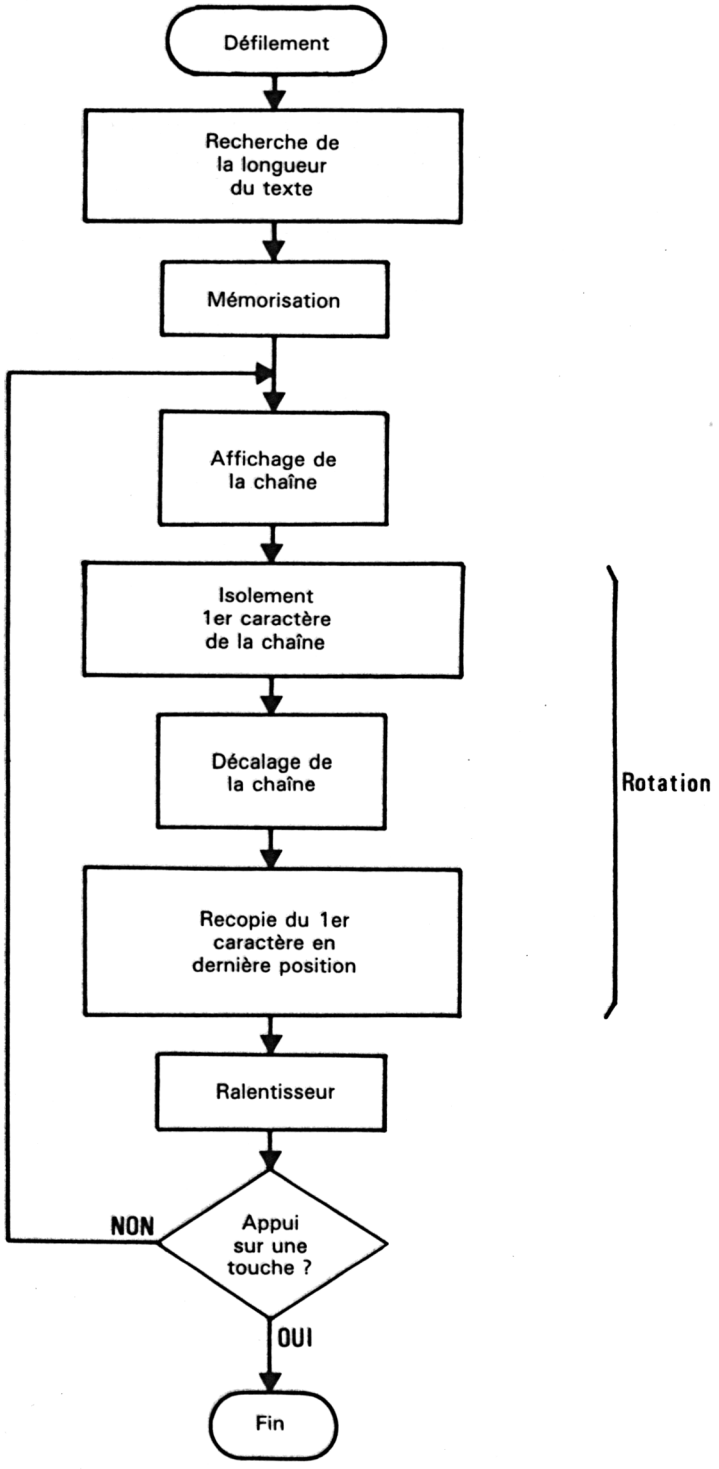
— affichage de la chaîne jusqu'au terminateur ;

— rotation de la chaîne,

— ralentisseur. Il consiste en une boucle qui utilise l'ordre DJNZ. Cette boucle contient 6 ordres NOP qui n'ont aucune action particulière, si ce n'est demander un cycle machine pour être exécutés.

— test, si une touche a été actionnée. Si oui, arrêt du programme et retour à l'appelant.

Ces diverses actions se retrouvent dans l'organigramme ci-dessous :



Le listing Assembleur est le suivant :

```
                            ORG  9000H
                            LOAD 9000H
                ;-----------------------------------
                ;Defilement d'un message sur
                ;l'ecran.
                ;-----------------------------------
                ;Entree: A=Vitesse (0 a 255)
                ;        H=Position en X
                ;        L=Position en Y
                ;-----------------------------------
                ;
                ;
                ;-----------------------------------
                ;Zone des constantes du programme
                ;-----------------------------------
                SETCUR:     EQU  OBB75H          ;TXT SET CURSOR
                WRCHAR:     EQU  OBB5AH          ;TXT OUTPUT
                RKEY:       EQU  OBB1BH          ;KM READ KEY
                ;
                ;
9000 326D90                 LD   (VIT),A         ;Vitesse
9003 226790                 LD   (POS),HL        ;Sauvegarde
9006 ED536990               LD   (TEXTE),DE      ;Sauvegarde à texte
900A EB                     EX   DE,HL
                ;
                ;Recherche de la longueur de TEXTE
                ;
                BP:         EQU  $               ;Boucle rech longueur
900B 7E                     LD   A,(HL)
900C B7                     OR   A
900D 23                     INC  HL
```

```
32 900E 20FB                JR   NZ,BP
33 9010 ED5B6990            LD   DE,(TEXTE)
34 9014 37                  SCF
35 9015 3F                  CCF
36 9016 ED52                SBC  HL,DE
37 9018 2B                  DEC  HL
38 9019 7D                  LD   A,L
39 901A 326B90              LD   (LEN),A          ;Longueur chaine
40              ;
41              MB:         EQU  $
42 901D 2A6790              LD   HL,(POS)
43 9020 CD75BB              CALL SETCUR           ;Locate
44 9023 2A6990              LD   HL,(TEXTE)
45              BP0:        EQU  $                ;Boucle d'affichage
46 9026 7E                  LD   A,(HL)
47 9027 B7                  OR   A
48 9028 2806                JR   Z,BP1
49 902A CD5ABB              CALL WRCHAR           ;Affichage
50 902D 23                  INC  HL
51 902E 18F6                JR   BP0
52              ;
53              BP1:        EQU  $                ;Rotation chaine
54 9030 2A6990              LD   HL,(TEXTE)
55 9033 7E                  LD   A,(HL)           ;1ere lettre
56 9034 326C90              LD   (SAUV),A         ;Sauvegarde
57 9037 3A6B90              LD   A,(LEN)
58 903A 47                  LD   B,A              ;Longueur decalage
59 903B 05                  DEC  B
60 903C ED5B6990            LD   DE,(TEXTE)
61 9040 1B                  DEC  DE
62              BP2:        EQU  $
63 9041 23                  INC  HL
```

```
64 9042 13                  INC   DE              ;Caract suivant
65 9043 7E                  LD    A,(HL)
66 9044 12                  LD    (DE),A
67 9045 10FA                DJNZ  BP2             ;Boucle de deplact
68 9047 3A6C90              LD    A,(SAUV)
69 904A 77                  LD    (HL),A
70 904B CD5490              CALL  WAIT            ;Ralentisseur
71 904E CD1BBB              CALL  RKEY            ;Test appui clavier
72 9051 30CA                JR    NC,MB           ;Pas d'appui => boucle
73 9053 C9                  RET
74                ;
75                ;Ralentisseur
76                ;
77                WAIT:     EQU   $
78 9054 3A6D90              LD    A,(VIT)         ;Vitesse
79                W0:       EQU   $               ;Boucle ralentisseur
80 9057 3D                  DEC   A
81 9058 2001                JR    NZ,W1
82 905A C9                  RET
83                W1:       EQU   $
84 905B 0600                LD    B,0
85                W2:       EQU   $
86 905D 00                  NOP
87 905E 00                  NOP
88 905F 00                  NOP
89 9060 00                  NOP
90 9061 00                  NOP
91 9062 00                  NOP                   ;Intructions NOP pour att
92 9063 10F8                DJNZ  W2
93 9065 18F0                JR    W0
94                ;-------------------------------------
95                ;Zone des variables du programme
96                ;-------------------------------------
```

```
97              POS:        DS   2              ;Position du curseur
98              TEXTE:      DS   2              ;à debut de texte
99              LEN:        DS   1              ;Longueur chaine
100             SAUV:       DS   1              ;Sauveg. caractere
101             VIT:        DS   1              ;Vitesse d'affichage
102                         END
```

Pour faciliter l'interfaçage, un programme Basic peut appeler ce programme Assembleur de la façon suivante :

```
100 MODE 2
110 FOR I=&8000 TO &800B:READ A:POKE I,A:NEXT
120 DATA &21,10,10,&11,0,&81,&3E,&70,&CD,&0,&90,&C9
130 A$="Exemple de message..."
140 FOR I=1 TO LEN(A$)
150   B$=MID$(A$,I,1)
160   B=ASC(B$) 'Code ASCII de chaque lettre
170   POKE &8100+I-1,B 'Mise en memoire
180 NEXT I
190 POKE &8100+I-1,0 'Code terminateur
```

L'abscisse correspond à la deuxième donnée, l'ordonnée à la troisième donnée.

Les données de la ligne 120 correspondent au programme Assembleur suivant :

```
1                           ORG  8000H
2                           LOAD 8000H
3 8000 210A0A               LD   HL,0A0AH          ;Position curseur
4 8003 110081               LD   DE,8100H          ;Position memoire
5 8006 3E70                 LD   A,70H             ;Vitesse d'affichage
6 8008 CD0090               CALL 9000H
7 800B C9                   RET
8                           END
```

Il est également possible de saisir les codes hexadécimaux correspondant au programme Assembleur dans un programme Basic. Le listing du programme est alors le suivant :

```
1000 'Defilement d'un message sur l'ecran
1010 '=====================================
1020 FOR I=&9000 TO &9066
1030   READ A$
1040   B$="&"+A$
1050   POKE I,VAL(B$)
1060 NEXT I
1070 '----------------------
1080 'Donnees hexadecimales
1090 '----------------------
1100 DATA 32,6D,90,22,67,90,ED,53,69,90,EB,7E,B7,28,20,FB
1110 DATA ED,5B,69,90,37,3F,ED,52,2B,7D,32,6B,90,2A,67,90
1120 DATA CD,75,BB,2A,69,90,7E,B7,28,06,CD,5A,BB,23,18,F6
1130 DATA 2A,69,90,7E,32,6C,90,3A,6B,90,47,05,ED,5B,69,90
1140 DATA 1B,23,13,7E,12,10,FA,3A,6C,90,77,CD,54,90,CD,18
1150 DATA BB,30,CA,C9,3A,6D,90,3D,20,01,C9,06,00,00,00,00
1160 DATA 00,00,00,10,F8,18,F0,00,00,00,00,00,00,00,00,00
```

Si vous décidez d'entrer les codes du programme assembleur sous Basic, vérifiez les codes entrés grâce au programme de checksum (voir Partie 9, chap. 8.4).

Pour cela, tapez « MERGE » suivi du nom sous lequel vous avez sauvegardé le programme de checksum. Exécutez le programme de checksum en tapant « RUN 50000 ». Les données de vérification sont les suivantes :

E6 F2 9D 97 37 E6 12

Si une ou plusieurs des données de vérification ne correspondent pas avec celles données ci-dessus, vérifiez la ligne correspondante.

Remarque :

Plusieurs données nulles ont été rajoutées en fin de listing afin d'assurer la compatibilité avec le programme de checksum.