```
Page 001          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

0001 00000                    TITLE   Editeur, gestion du texte <xtext.as>
0002 00000
0003 00000           *
0004 00000           * Paramètres du tableau utilisé pour optimiser l'accès
0005 00000           * au texte.
0006 00000           *
0007 00000            BUFENT EQU     10      5 pour no de ligne, 5 pour adresse
0008 00000            DELTA  EQU     200     une entrée toutes les 200 lignes
0009 00000            TURBULENCE EQU 50
0010 00000           *
0011 00000           * Note : il faut que 0 <= DELTA + TURBULENCE <= 255
0012 00000           *
0013 00000
0014 00000           ***************************************************************
0015 00000           * inibuf
0016 00000           *
0017 00000           * But: initialiser le buffer d'accès au fichier
0018 00000           * Entrée:
0019 00000           *   - FILADR = adresse du début du fichier
0020 00000           * Sortie:
0021 00000           *   - BUFID = id du buffer
0022 00000           *   - BUFADR = adresse du buffer
0023 00000           *   - DERN = numéro de la dernière ligne
0024 00000           * Abime: A-D, R0-R3
0025 00000           * Appelle: I/OALL, IODAL, IOFSCR, I/OCON
0026 00000           * Niveaux: 3
0027 00000           * Historique:
0028 00000           *   88/05/15: PD/JT conception & codage
0029 00000           ***************************************************************
0030 00000
0031 00000           =inibuf
0032 00000           *
0033 00000           * Désallocation éventuelle du buffer
0034 00000           *
0035 00000 1B00000            D0=(5)  =BUFID
0036 00007 D2                 C=0     A
0037 00008 1563               C=DAT0  X       C(A) := bufid
0038 0000D 8AA                ?C#0    A
0039 00010 30                 GOYES   inib10
0040 00012 8F00000            GOSBVL  =I/ODAL Désallouer d'abord
0041 00019           *
0042 00019           * Allocation d'un nouveau buffer
0043 00019           *
0044 00019 8F00000    inib10 GOSBVL  =IOFSCR Find an available scratch buffer ID
0045 0001E 501                GONC    inib15  Ok
0046 00023 8D0000             GOVLNG  =CORUPT No available buffer id
0047 0002A
0048 0002A 8D0000     memerr GOVLNG  =MEMERR
0049 00031
0050 00031 1B00000    inib15 D0=(5)  =BUFID
```

```
Page 002          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

0051 00038 1543               DAT0=C  X       BUFID := id trouve
0052 0003C           *
0053 0003C           * Calculer la mémoire disponible
0054 0003C           *
0055 0003C 1F00000            D1=(5)  =AVMEME
0056 00043 143                A=DAT1  A
0057 00046 1C4                D1=D1-  5       D1=(5) =AVMEMS
0058 00049 147                C=DAT1  A
0059 0004C EA                 A=A-C   A       A(A) := mémoire disponible
0060 0004E D2                 C=0     A
0061 00050 3100               LC(2)   7+LEEWAY
0062 00054           *
0063 00054           * LEEWAY = place nécessaire pour récupérer le HP-71
0064 00054           * 7 = header du buffer
0065 00054           *
0066 00054 EA                 A=A-C   A       A(A) := mémoire - LEEWAY - 7
0067 00056 43D                GOC     memerr  On ne peut pas travailler...
0068 00059 31A0               LC(2)   BUFENT  C(4-2) = 000
0069 0005D 8B2                ?A<C    A
0070 00060 AC                 GOYES   memerr  Pas d'entrée suffisante
0071 00062 32FFF              LC(3)   4095    C(4-3) = 00
0072 00067 8B2                ?A<C    A
0073 0006A 40                 GOYES   inib20
0074 0006C DA                 A=C     A       A(A) := max (A(A), 4095)
0075 0006E           *
0076 0006E           * A(A) = taille du buffer à créer
0077 0006E           *
0078 0006E D8         inib20 B=A     A       B(A) := taille du buffer
0079 00070 101                R1=A            R1 := taille du buffer
0080 00073 1563               C=DAT0  X       C(4-3) = 00
0081 00077 8F00000            GOSBVL  =I/OALL Cy = 0 (no room) : impossible
0082 0007E           * B(A) = taille du buffer créé
0083 0007E           * D1 = ^ début des données
0084 0007E 1B00000            D0=(5)  =BUFADR
0085 00025 137                CD1EX
0086 00028 144                DAT0=C  A       BUFADR := adresse du buffer
0087 0002B 135                D1=C            D1 := ^ début du buffer
0088 0002E           *
0089 0002E           * last := fadr := adresse de la première ligne
0090 0002E           * badr := adresse du début du buffer
0091 0002E           * taille := taille du buffer
0092 0002E           * nl := DELTA
0093 0002E           * dernière := 0
0094 0002E           * tant que longueur_LIF (fadr) # FFFF
0095 0002E           *   faire
0096 0002E           *     si --nl < 0
0097 0002E           *       alors
0098 0002E           *         nl := DELTA
0099 0002E           *         dernière := dernière + DELTA
0100 0002E           *         si taille > 2*BUFENT
```

```
Page 003          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

0101 0002E           *           alors
0102 0002E           *             taille := taille - BUFENT
0103 0002E           *             *badr := (dernière, fadr - last)
0104 0002E           *             last := fadr
0105 0002E           *             badr++
0106 0002E           *           fin si
0107 0002E           *       fin si
0108 0002E           *       fadr := adresse de la ligne suivante
0109 0002E           * fin tant que
0110 0002E           * dernière := dernière + (DELTA - nl)
0111 0002E           * *badr := (0, 0)
0112 0002E           * compresser le buffer de (taille - BUFENT) quartets
0113 0002E           *
0114 0002E
0115 0002E           * R0 = dernière
0116 0002E           * D(B) = nl (limite à 255)
0117 0002E           * D0 = fadr
0118 0002E           * D1 = badr
0119 0002E           * R1 = taille (déjà mise)
0120 0002E           * R2 = last
0121 0002E           * B(A) = fadr limite
0122 0002E           *
0123 0002E AF2                C=0     W
0124 00091 108                R0=C            dernière := 0
0125 00094 317C               LC(2)   (DELTA)-1
0126 00098 AE7                D=C     B       nl := (DELTA)-1
0127 0009B
0128 0009B 1B00000            D0=(5)  =FILADR
0129 000A2 146                C=DAT0  A
0130 000A5 134                D0=C            D0 := ^ en-tête du fichier
0131 000A8 16F                D0=D0+  16
0132 000AB 16F                D0=D0+  16
0133 000AE 142                A=DAT0  A       A(A) := REL(5) FiLeNd
0134 000B1 136                CD0EX           C(A) := ^ REL(5)
0135 000B4 134                D0=C
0136 000B7 C2                 C=C+A   A       C(A) := ^ EOF
0137 000B9 D5                 B=C     A       B(A) := fadr limite
0138 000BB 164                D0=D0+  5       D0 := ^ première ligne
0139 000BE 136                CD0EX           C(A) := fadr
0140 000C1 10A                R2=C            last := fadr
0141 000C4 560                GONC    inib50  B.E.T.
0142 000C7
0143 000C7 6E80       inib89 GOTO    inib90
0144 000CB
0145 000CB           * C(A) := fadr
0146 000CB 8BD        inib50 ?C>=B   A       at EOF ?
0147 000CE 9F                 GOYES   inib89  fin tant que
0148 000D0 134                D0=C            D0 := ^ début de la ligne
0149 000D3 15A3               A=DAT0  4
0150 000D7           * SWPBYT (#17A24) recopie pour eviter un GOSBVL
```

```
Page 004          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

0151 000D7 D6                 C=A     A
0152 000D9 F0                 ASL     A
0153 000DB F0                 ASL     A
0154 000DD F6                 CSR     A
0155 000DF F6                 CSR     A       C(4) := 0 (entre autres)
0156 000E1 AEA                A=C     B       A(A) longueur LIF
0157 000E4           * fin du pompage
0158 000E4 23                 P=      3
0159 000E6 A96                C=A     WP      on sait que C(4) = 0
0160 000E9 DA                 A=C     A
0161 000EB B14                A=A+1   WP      Cy = 1 si C(A) = FFFF
0162 000EE 20                 P=      0
0163 000F0 456                GOC     inib90  fin tant que
0164 000F3           * conversion en nombre de quartets à sauter
0165 000F3 81C                ASRB
0166 000F6 E4                 A=A+1   A
0167 000F8 C4                 A=A+A   A
0168 000FA C4                 A=A+A   A       A(A) := nb de quartets à sauter
0169 000FC
0170 000FC A6F                D=D-1   B       --nl
0171 000FF 5D4                GONC    inib80  go if nl >= 0
0172 00102
0173 00102 103                R3=A            R3 := sauvegarde de A
0174 00105 D2                 C=0     A
0175 00107 317C               LC(2)   (DELTA)-1
0176 0010B AE7                D=C     B       nl := (DELTA)-1
0177 0010E 110                A=R0            A(A) := dernière
0178 00111 CA                 A=A+C   A
0179 00113 E4                 A=A+1   A
0180 00115 100                R0=A            dernière += DELTA
0181 00118
0182 00118 111                A=R1            A(A) := taille
0183 0011B 3141               LC(2)   2*BUFENT
0184 0011F 8BA                ?A<=C   A
0185 00122 B2                 GOYES   inib80  go if taille <= 20
0186 00124
0187 00124 31A0               LC(2)   BUFENT
0188 00128 EA                 A=A-C   A
0189 0012A 101                R1=A            taille -= 10
0190 0012D
0191 0012D 110                A=R0            A(A) := dernière
0192 00130 141                DAT1=A  A       *badr := (dernière,...
0193 00133 174                D1=D1+  5
0194 00136 112                A=R2            A(A) := last
0195 00139 136                CD0EX
0196 0013C 134                D0=C
0197 0013F 10A                R2=C            nouveau last := fadr
0198 00142 E2                 C=C-A   A       C(A) := fadr - ancien last
0199 00144 145                DAT1=C  A       ....            ..... fadr - last)
0200 00147 174                D1=D1+  5
```

```
Page 005          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

0201 0014A
0202 0014A 113              A=R3            A(A) := nb de quartets à sauter
0203 0014D
0204 0014D 136        inib80 CD0EX
0205 00150 C2               C=C+A   A       C(A) := fadr
0206 00152 687F             GOTO    inib50  fin tant que
0207 00156
0208 00156            inib90
0209 00156            *
0210 00156            * marque de fin de buffer
0211 00156            *
0212 00156 AF2              C=0     W
0213 00159 15D9             DAT1=C  BUFENT  *badr := (0, 0)
0214 0015D            *
0215 0015D            * Calcul du numéro de dernière ligne
0216 0015D            *
0217 0015D 317C             LC(2)   (DELTA)-1 on sait C(4-2) = 00
0218 00161 BGB              C=C-D   B       C(A) := (DELTA)-1 - nl
0219 00164 110              A=R0            A(A) := dernière
0220 00167 CA               A=A+C   A       dernière += (DELTA)-1 - nl
0221 00169 1B00000          D0=(5)  =DERN
0222 00170 140              DAT0=A  A       DERN := dernière
0223 00173            *
0224 00173            * Compresser le buffer
0225 00173            *
0226 00173 119              C=R1
0227 00176 D5               B=C     A       B(A) := taille du buffer
0228 00178 D2               C=0     A
0229 0017A 30A              LC(1)   BUFENT
0230 0017D E1               B=B-C   A       taille - 10
0231 0017F 1900             D0=(2)  =BUFID
0232 00183 1563             C=DAT0  X
0233 00187 8D00000          GOVLNG  =I/OCON
0234 0018E
0235 0018E            ****************************************************
0236 0018E            * seek
0237 0018E            *
0238 0018E            * But: chercher l'adresse de la ligne de numéro C (1..dern)
0239 0018E            * Entrée:
0240 0018E            *   - C(A) = numéro de la ligne à chercher
0241 0018E            * Sortie:
0242 0018E            *   - D0 =   début de la ligne
0243 0018E            * Abime: A-D, R0-R3, D0, D1, TMP5
0244 0018E            * Appelle: initbuf
0245 0018E            * Niveaux: 4
0246 0018E            * Detail:
0247 0018E            *   last := 0
0248 0018E            *   adr := FILADR
0249 0018E            *   p = &buffer [0]
0250 0018E            *   tant que (*p->numéro < numéro)
```

```
Page 006          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

0251 0018E            *     faire
0252 0018E            *       x = *p ->numéro - last
0253 0018E            *       si (x>DELTA+TURBULENCE) ou (x<DELTA-TURBULENCE)
0254 0018E            *         alors
0255 0018E            *           re-calculer le buffer
0256 0018E            *           recommencer au début
0257 0018E            *       fin si
0258 0018E            *       last := *p->numero
0259 0018E            *       adr += *p->adresse
0260 0018E            *       p++
0261 0018E            *   fin tant que
0262 0018E            *   entrée := numéro - last
0263 0018E            *   pour i := 1 à entrée
0264 0018E            *     faire
0265 0018E            *       adr = ligne suivante (adr)
0266 0018E            *   fin pour
0267 0018E            *   renvoyer adr
0268 0018E            * Affectation des registres:
0269 0018E            *   R0 = last
0270 0018E            *   D1 = accès
0271 0018E            *   B(A) = no
0272 0018E            *   D(A), puis D0 =   dans le fichier
0273 0018E            * Historique:
0274 0018E            *   88/05/18: PD/JT conception & codage
0275 0018E            *   88/06/05: PD/JT ajout intervalle pour test du recalcul
0276 0018E            *   88/11/20: PD/JT recalcul si trop de lignes à la fin
0277 0018E            ****************************************************
0278 0018E
0279 0018E D5         =seek   B=C     A
0280 00190 1F00000            D1=(5)  =BUFADR
0281 00197 147                C=DAT1  A
0282 0019A 135                D1=C            D1 := ^ buffer
0283 0019D 1B00000            D0=(5)  =FILADR
0284 001A4 142                A=DAT0  A
0285 001A7 D2                 C=0     A
0286 001A9 3152               LC(2)   37
0287 001AD C2                 C=A+C   A       A(A) := ^ début du buffer
0288 001AF D7                 D=C     A       D(A) := ^ début du fichier
0289 001B1 D2                 C=0     A
0290 001B3 E6                 C=C+1   A       last := 1
0291 001B5 108                R0=C            last := numéro de la première ligne
0292 001B8 594                GONC    =sek030 B.E.T.
0293 001BB
0294 001BB 1B00000    sek005  D0=(5)  =TMP5
0295 001C2 D9                 C=B     A
0296 001C4 144                DAT0=C  A       TMP5 := numéro à trouver
0297 001C7 753E               GOSUB   =initbuf recalcul...
0298 001CB 1B00000            D0=(5)  =TMP5
0299 001D2 146                C=DAT0  A
0300 001D5 68BF               GOTO    =seek
```

```
Page 007          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

0301 001D9
0302 001D9 110        sek010  A=R0            A(A) := last
0303 001DC 108                R0=C            R0 := nouveau last
0304 001DF DE                 ACEX    A
0305 001E1 EA                 A=A-C   A       A(A) := accès [i].no - last
0306 001E3 D2                 C=0     A
0307 001E5 31AF               LC(2)   (DELTA)+(TURBULENCE)
0308 001E9 8B6                ?A<C    A
0309 001EC FC                 GOYES   sek005  recalcul...
0310 001EE            * ajout du 5/06/1988
0311 001EE 3169               LC(2)   (DELTA)-(TURBULENCE)
0312 001F2 8B2                ?A<C    A
0313 001F5 6C                 GOYES   sek005  recalcul aussi
0314 001F7            * fin de l'ajout du 5/06/1988
0315 001F7
0316 001F7 174                D1=D1+  5
0317 001FA 147                C=DAT1  A       C(A) := adresse incrémentale
0318 001FD C3                 D=D+C   A
0319 001FF 174                D1=D1+  5
0320 00202
0321 00202 147        sek030  C=DAT1  A       C(A) := numéro
0322 00205 8AA                ?C=0    A
0323 00208 70                 GOYES   sek040
0324 0020A 8B1                ?C<B    A
0325 0020D CC                 GOYES   sek010
0326 0020F            *
0327 0020F            * Deuxième partie de la routine
0328 0020F            *
0329 0020F             sek040
0330 0020F            *
0331 0020F            * Sauvegarde du numéro de la ligne à atteindre dans R0
0332 0020F            *
0333 0020F D9                 C=B     A       C(A) := numéro de la ligne
0334 00211 128                CR0EX           C(A) := last (inférieur à no)
0335 00214            *
0336 00214            * R0 = numéro de la ligne à atteindre
0337 00214            * C(A) = last (<no)
0338 00214            *
0339 00214 E1                 B=B-C   A       B(A) := nombre de lignes à parcourir
0340 00216
0341 00216            *
0342 00216            * Y-a-t-il beaucoup trop de lignes à parcourir ?
0343 00216            *
0344 00216 D2                 C=0     A
0345 00218 31AF               LC(2)   (DELTA)+(TURBULENCE)
0346 0021C 8BD                ?B<=C   A
0347 0021F 02                 GOYES   sek045  non.
0348 00221            *
0349 00221            * Il n'y a pas suffisamment d'entrées dans le buffer. Est-ce
0350 00221            * par manque de place, ou parce qu'on a rajouté des lignes
```

```
Page 008          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

0351 00221            * à la fin ?
0352 00221            *
0353 00221 D4                 A=B     A       Sauvegarde de B(A) pendant MEMCKL
0354 00223 101                R1=A            R1(A) := nb de lignes à parcourir
0355 00226 D2                 C=0     A
0356 00228 30A                LC(1)   BUFENT  taille d'une entrée supplémentaire
0357 0022B 8F00000            GOSBVL  =MEMCKL assez de place ?
0358 00232 118                C=R0
0359 00235 D5                 B=C     A       B(A) := no de la ligne à atteindre
0360 00237 538                GONC    sek005  oui : recalcul
0361 0023A            *
0362 0023A            * Pas assez de place pour réinitialiser le buffer, il faut
0363 0023A            * donc faire avec.
0364 0023A            *
0365 0023A 111                A=R1            A(A) := sauvegarde avant MEMCKL
0366 0023D D8                 B=A     A       B(A) := nb de lignes à parcourir
0367 0023F
0368 0023F C8         sek045  C=D     A       C(A) := ^ ligne pointée
0369 00241 134                D0=C            D0 := ^ ligne courante
0370 00244 6030               GOTO    sek060
0371 00248
0372 00248 15A3       sek050  A=DAT0  4       A(3-0) := longueur LIF toute
0373 0024C            * SWPBYT (#17A24) recopié pour éviter un GOSBVL
0374 0024C D6                 C=A     A
0375 0024E F0                 ASL     A
0376 00250 F0                 ASL     A
0377 00252 F6                 CSR     A
0378 00254 F6                 CSR     A       C(4) := 0 (entre autres)
0379 00256 AEA                A=C     B       A(A) longueur LIF
0380 00259            * fin du pompage
0381 00259 23                 P=      3
0382 0025B A96                C=A     WP      on sait que C(4) = 0
0383 0025E 20                 P=      0
0384 00260 DA                 A=C     A
0385 00262 E4                 A=A+1   A
0386 00264 81C                ASRB
0387 00267 E4                 A=A+1   A
0388 00269 C4                 A=A+A   A
0389 0026B C4                 A=A+A   A       A(A) := nb de quartets à sauter
0390 0026D
0391 0026D 136                CD0EX           C(A) := ^ ligne courante
0392 00270 C2                 C=C+A   A       C(A) := ^ ligne suivante
0393 00272 134                D0=C
0394 00275
0395 00275 CD         sek060  B=B-1   A
0396 00277 50D                GONC    sek050
0397 0027A 01                 RTN
0398 0027C
0399 0027C            ****************************************************
0400 0027C            * LIFlen
```

```
Page 009          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

0401 0027C        *
0402 0027C        * But: calculer la longueur d'une ligne LIF
0403 0027C        * Entrée:
0404 0027C        *   - D0 = ^ ligne LIF
0405 0027C        * Sortie:
0406 0027C        *   - C(A) = longueur de la ligne en octets
0407 0027C        *   - A(A) = taille de la ligne en quartets (y compris LIF)
0408 0027C        * Abime: A, C(A), P
0409 0027C        * Niveaux: 0
0410 0027C        * Historique:
0411 0027C        *   88/05/18: PD/JT conception & codage
0412 0027C        ***************************************************
0413 0027C
0414 0027C 15A3   =LIFlen A=DAT0  4       A(3-0) := longueur LIF brute
0415 00280        * SWPBYT (#17A24) recopié pour éviter un GOSBVL
0416 00280 D6             C=A     A
0417 00282 F0             ASL     A
0418 00284 F0             ASL     A
0419 00286 F6             CSR     A
0420 00288 F6             CSR     A       C(4) := 0 (entre autres)
0421 0028A AEA            A=C     B       A(A) longueur LIF
0422 0028D        * fin du pompage
0423 0028D 23             P=      3
0424 0028F A96            C=A     WP      on sait que C(4) = 0
0425 00292 20             P=      0
0426 00294 DA             A=C     A
0427 00296 E4             A=A+1   A
0428 00298 81C            ASRB
0429 0029B E4             A=A+1   A
0430 0029D C4             A=A+A   A
0431 0029F C4             A=A+A   A       A(A) := nb de quartets à sauter
0432 002A1 01             RTN
0433 002A3
0434 002A3        ***************************************************
0435 002A3        * addlin
0436 002A3        *
0437 002A3        * But: ajoute une ou n lignes dans le buffer.
0438 002A3        * Entrée:
0439 002A3        *   - C(A) = nombre de lignes
0440 002A3        *   - A(A) = longueur de ce qui a été rajoute
0441 002A3        *   - B(A) = numéro de la nouvelle première ligne
0442 002A3        * Sortie: -
0443 002A3        * Abime: C(A), D(A), D1
0444 002A3        * Niveaux: 0
0445 002A3        * Note: DERN est réactualisé en conséquence
0446 002A3        * Historique:
0447 002A3        *   88/05/24: PD/JT conception & codage
0448 002A3        ***************************************************
0449 002A3
0450 002A3 D7     =addlin D=C     A       D(A) := nombre de lignes ajoutées
```

```
Page 010          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

0451 002A5
0452 002A5 1F0000         D1=(5)  =DERN
0453 002AC 147            C=DAT1  A
0454 002AF CB             C=C+D   A
0455 002B1 145            DAT1=C  A
0456 002B4
0457 002B4 1D00           D1=(2)  =BUFADR
0458 002B8 147            C=DAT1  A
0459 002BB 135            D1=C            D1 := ^ buffer
0460 002BE        *
0461 002BE        * Trouver l'entrée dans le buffer
0462 002BE        *
0463 002BE 147    adln10 C=DAT1  A
0464 002C1 8AA            ?C=0    A
0465 002C4 00             RTNYES          Ok, varienafair
0466 002C6 8B5            ?C>B    A
0467 002C9 30             GOYES   adln20  Trouve !
0468 002CB 179            D1=D1+  BUFENT
0469 002CE 5FE            GONC    adln10  B.E.T.
0470 002D1        *
0471 002D1        * L'entrée est trouvée. Reste à actualiser l'adresse, et le
0472 002D1        * numéro des lignes suivantes.
0473 002D1        *
0474 002D1 174    adln20 D1=D1+  5
0475 002D4 147            C=DAT1  A       C(A) := adresse
0476 002D7 C2             C=C+A   A
0477 002D9 145            DAT1=C  A       buffer := nouvelle adresse
0478 002DC 1C4            D1=D1-  5       D1 := ^ numéro
0479 002DF
0480 002DF 147    adln30 C=DAT1  A
0481 002E2 8AA            ?C=0    A
0482 002E5 00             RTNYES          finiiiiii !
0483 002E7 CB             C=C+D   A
0484 002E9 145            DAT1=C  A
0485 002EC 179            D1=D1+  BUFENT
0486 002EF 5FE            GONC    adln30  B.E.T.
0487 002F2
0488 002F2        ***************************************************
0489 002F2        * rpllin
0490 002F2        *
0491 002F2        * But: remplacer une ligne dans le buffer
0492 002F2        * Entrée:
0493 002F2        *   - A(A) = différence de taille (nouvelle - ancienne)
0494 002F2        *   - B(A) = numéro de la ligne remplacée
0495 002F2        * Sortie: -
0496 002F2        * Abime: C(A), D1
0497 002F2        * Niveaux: 0
0498 002F2        * Historique:
0499 002F2        *   88/05/24: PD/JT conception & codage
0500 002F2        ***************************************************
```

```
Page 011          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

0501 002F2
0502 002F2 1F0000 =rpllin D1=(5)  =BUFADR
0503 002F9 147            C=DAT1  A
0504 002FC 135            D1=C            D1 := ^ buffer
0505 002FF        *
0506 002FF        * Trouver l'entrée dans le buffer
0507 002FF        *
0508 002FF 147    rpln10 C=DAT1  A
0509 00302 8AA            ?C=0    A
0510 00305 00             RTNYES          Ok, varienafair
0511 00307 8B5            ?C>B    A
0512 0030A 80             GOYES   rpln20  Trouve !
0513 0030C 179            D1=D1+  BUFENT
0514 0030F 5FE            GONC    rpln10  B.E.T.
0515 00312        *
0516 00312        * L'entrée est trouvée. Il faut actualiser l'adresse de
0517 00312        * l'entrée suivante.
0518 00312        *
0519 00312 174    rpln20 D1=D1+  5
0520 00315 147            C=DAT1  A
0521 00318 C2             C=C+A   A       A peut être négatif
0522 0031A 145            DAT1=C  A
0523 0031D 01             RTN
0524 0031F
0525 0031F        ***************************************************
0526 0031F        * dellin
0527 0031F        *
0528 0031F        * But: supprime une ou n lignes dans le buffer
0529 0031F        * Entrée:
0530 0031F        *   - C(A) = nombre de lignes
0531 0031F        *   - A(A) = longueur de ce qui a été enlevé
0532 0031F        *   - B(A) = numéro de l'ancienne première ligne
0533 0031F        * Sortie:
0534 0031F        *   - DERN est réactualisée
0535 0031F        * Abime: A-C, D(A), D1
0536 0031F        * Appelle: -
0537 0031F        * Niveaux: 0
0538 0031F        * Algorithme:
0539 0031F        *   début := B(A)
0540 0031F        *   delta := A(A)
0541 0031F        *   nombre := C(A)
0542 0031F        *   last := B(A) + C(A) - 1
0543 0031F        *   avant := 1
0544 0031F        *
0545 0031F        *   DERN -= nombre
0546 0031F        *
0547 0031F        *   /* PHASE I : chercher la ligne de début */
0548 0031F        *
0549 0031F        *   p := &buffer [0]
0550 0031F        *   tant que p->numéro < début
```

```
Page 012          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

0551 0031F        *     faire
0552 0031F        *       si p->numéro == 0 alors sortir fin si
0553 0031F        *       avant := p->numéro
0554 0031F        *       p++
0555 0031F        *   fin tant que
0556 0031F        *   /* p est la première entrée >= début */
0557 0031F        *
0558 0031F        *   /* PHASE II : chevauchement de frontières */
0559 0031F        *
0560 0031F        *   tant que last >= p->numéro
0561 0031F        *     alors
0562 0031F        *       si p->numéro == 0 alors sortir fin si
0563 0031F        *       p->adresse := 0
0564 0031F        *       p->numéro := avant
0565 0031F        *       p++
0566 0031F        *   fin tant que
0567 0031F        *
0568 0031F        *   /* PHASE III : actualiser la fin du buffer */
0569 0031F        *
0570 0031F        *   tant que p != 0
0571 0031F        *     faire
0572 0031F        *       p->adr -= delta
0573 0031F        *       p->numéro -= nombre
0574 0031F        *       delta := 0
0575 0031F        *       p++
0576 0031F        *   fin tant que
0577 0031F        * Historique:
0578 0031F        *   88/05/24: PD/JT documentation de l'interface
0579 0031F        *   88/10/29: PD/JT conception & codage
0580 0031F        *   88/11/01: PD/JT déplacement de la modification de DERN
0581 0031F        ***************************************************
0582 0031F
0583 0031F        =dellin
0584 0031F        *
0585 0031F        * Sauvegarde des valeurs
0586 0031F        * - début (B(A)) dans B(4-0)
0587 0031F        * - delta (A(A)) dans B(9-5)
0588 0031F        * - nombre (C(A)) dans B(14-10)
0589 0031F        *
0590 0031F D7             D=C     A       D(A) := nombre
0591 00321 8F00000        GOSBVL  =CSLC5
0592 00328 D6             C=A     A
0593 0032A 8F00000        GOSBVL  =CSLC5
0594 00331 D9             C=B     A
0595 00333 AF5            B=C     W
0596 00336        *
0597 00336        * B(14-10) = nombre
0598 00336        * B(9-5) = delta
0599 00336        * B(4-0) = début
0600 00336        * D(A) = nombre
```

```
Page 013          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

0601 00336          •
0602 00336          •
0603 00336          •
0604 00336          • DERN -= nombre
0605 00336          •
0606 00336 1F00000          D1=(5)  =DERN
0607 0033D 147              C=DAT1  A       C(A) := dernière
0608 00340 EB               C=C-D   A       C(A) := dernière - nombre
0609 00342 145              DAT1=C  A       dernière -= nombre
0610 00345
0611 00345          •
0612 00345          • p = &buffer [0]
0613 00345          • avant := 1
0614 00345          •
0615 00345 1D00             D1=(2)  =BUFADR
0616 00349 147              C=DAT1  A       C(A) := & buffer [0]
0617 0034C 135              D1=C            D1 := & buffer [0]
0618 0034F
0619 0034F D3               D=0     A       avant := 0
0620 00351 E7               D=D+1   A       avant := 1
0621 00353          •
0622 00353          • tant que p->numéro < début
0623 00353          •   faire
0624 00353          •     si p->numéro == 0 alors sortir fin si
0625 00353          •     avant := p->numéro
0626 00353          •     p++
0627 00353          • fin tant que
0628 00353          •
0629 00353          • Invariant de boucle :
0630 00353          • - D1 = " entrée courante dans le buffer (p)
0631 00353          • - B(A) = début
0632 00353          •
0633 00353 147      d1100   C=DAT1  A
0634 00356 8AA              ?C=0    A
0635 00359 00               RTNYES          si p->numéro == 0 alors sortir
0636 0035B 8BD              ?C>=B   A
0637 0035E A0               GOYES   d1200
0638 00360 D7               D=C     A       avant := p->numéro
0639 00362 179              D1=D1+  BUFENT
0640 00365 5DE              GONC    d1100   B.E.T.
0641 00368          •
0642 00368          •
0643 00368          • Phase II. On rappelle que :
0644 00368          • - B(14-10) = nombre
0645 00368          • - B(9-5) = delta
0646 00368          • - B(4-0) = début
0647 00368          • - D(A) = avant
0648 00368          • - D1 = p
0649 00368          •
0650 00368          d1200
```

```
Page 014          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

0651 00368          •
0652 00368          • Calcul de last (début + nombre - 1)
0653 00368          •                   B(4-0)  B(14-10)
0654 00368          •
0655 00368 AF9              C=B     W       C(14-10) := nombre
0656 0036B 8F00000          GOSBVL  =CLC6   C(A) := nombre
0657 00372 C9               C=C+B   A       C(A) := début + nombre
0658 00374 CE               C=C-1   A       C(A) := last
0659 00376 DA               A=C     A       A(A) := last
0660 00378 521              GONC    d1220   B.E.T
0661 0037B          •
0662 0037B          • tant que last >= p->numéro
0663 0037B          •   alors
0664 0037B          •     si p->numéro == 0 alors sortir fin si
0665 0037B          •     p->adresse := 0
0666 0037B          •     p->numéro = avant
0667 0037B          •     p++
0668 0037B          • fin tant que
0669 0037B          •
0670 0037B          • Invariant de boucle :
0671 0037B          • - A(A) = last
0672 0037B          • - D1 = p
0673 0037B          •
0674 0037B DB       d1210   C=D     A       C(A) := début
0675 0037D 145              DAT1=C  A       p->numéro := début
0676 00380 174              D1=D1+  5
0677 00383 D2               C=0     A
0678 00385 145              DAT1=C  A       p->adresse := 0
0679 00388 174              D1=D1+  5
0680 0038B 147      d1220   C=DAT1  A       A(A) := p->numéro
0681 0038E 8AA              ?C=0    A
0682 00391 00               RTNYES          si p->numéro == 0 alors sortir
0683 00393 8BE              ?C>=A   A
0684 00396 5E               GOYES   d1210
0685 00398
0686 00398          •
0687 00398          • Phase III. On rappelle que :
0688 00398          • - B(14-10) = nombre
0689 00398          • - B(9-5) = delta
0690 00398          • - B(4-0) = début
0691 00398          • - D1 = p
0692 00398          •
0693 00398          d1300
0694 00398 AF4              A=B     W       A(14-10) := nombre; A(9-5) := delta
0695 0039B 8F00000          GOSBVL  =ASRW5  A(9-5) := nombre : A(A) := delta
0696 003A2 D8               B=A     A       B(A) := delta
0697 003A4 8F00000          GOSBVL  =ASRW5  A(A) := nombre
0698 003AB 6810             GOTO    d1350
0699 003AF          •
0700 003AF          • Invariant de boucle :
```

```
Page 015          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

0701 003AF          • - A(A) = nombre
0702 003AF          • - B(A) = delta
0703 003AF          • - D1 = p
0704 003AF          •
0705 003AF E2       d1310   C=C-A   A       C(A) := p->numéro - nombre
0706 003B1 145              DAT1=C  A       p->numéro -= nombre
0707 003B4 174              D1=D1+  5
0708 003B7
0709 003B7 147              C=DAT1  A       C(A) := p->adresse
0710 003BA E9               C=C-B   A       C(A) := p->adresse - delta
0711 003BC 145              DAT1=C  A       p->adresse -= delta
0712 003BF 174              D1=D1+  5
0713 003C2
0714 003C2 D1               B=0     A       delta := 0
0715 003C4
0716 003C4 147      d1350   C=DAT1  A       C(A) := p->numéro
0717 003C7 8AE              ?C#0    A
0718 003CA 5E               GOYES   d1310
0719 003CC
0720 003CC 01               RTN
0721 003CE
0722 003CE          ********************************************************
0723 003CE          • syncbf
0724 003CE          •
0725 003CE          • But: synchronise l'adresse du buffer après un mouvement
0726 003CE          •   mémoire.
0727 003CE          • Entrée:
0728 003CE          •   - BUFID = ID du buffer
0729 003CE          • Sortie:
0730 003CE          •   - BUFADR = nouvelle adresse du buffer
0731 003CE          • Abîme: A, C(A), C(S), D1
0732 003CE          • Appelle: IOFND0
0733 003CE          • Niveaux: 1
0734 003CE          • Historique:
0735 003CE          •   88/05/24: PD/JT conception & codage
0736 003CE          ********************************************************
0737 003CE
0738 003CE 1F00000  =syncbf D1=(5)  =BUFID
0739 003D5 1573             C=DAT1  X
0740 003D9 8F00000          GOSBVL  =IOFND0
0741 003E0 137              CD1EX
0742 003E3 1F00000          D1=(5)  =BUFADR
0743 003EA 145              DAT1=C  A
0744 003ED 01               RTN
0745 003EF
0746 003EF          ********************************************************
0747 003EF          • SRCLIN, SRCADR
0748 003EF          •
0749 003EF          • But: chercher une chaine (générique) dans le fichier texte
0750 003EF          •   La chaine à chercher est déjà compilée.
```

```
Page 016          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

0751 003EF          • Entrée:
0752 003EF          •   - R0(7-0) = caractéristique du buffer (chaine compilée)
0753 003EF          •   - A(A) = numéro de la ligne de début
0754 003EF          •   - C(A) = numéro de la ligne de fin
0755 003EF          •   SRCLIN : rien d'autre
0756 003EF          •   SRCADR :
0757 003EF          •       - B(A) = offset dans la ligne (en caractères)
0758 003EF          •       - D= " début ligne LIF (sur la longueur)
0759 003EF          • Sortie:
0760 003EF          •   Cy = 1 : match found
0761 003EF          •       - A(A) = numéro de la ligne trouvée
0762 003EF          •       - C(A) = longueur de l'occurrence trouvée (en octets)
0763 003EF          •       - D= " occurrence trouvée
0764 003EF          •       - D1 = " ligne (long. LIF) contenant l'occurrence
0765 003EF          •   Cy = 0 : match not found
0766 003EF          • Abîme: A-D, R0(12-8), R1, D0, D1, FUNCR0
0767 003EF          •   en plus, pour SRCLIN : R2-R3, FUNCR1, TMP5
0768 003EF          • Note: SRCLIN appelle "seek", qui est dévoreuse de place.
0769 003EF          • Appelle: seek, POSADR, LIFlen
0770 003EF          • Niveaux: 5 (seek) pour SRCLIN ou 4 (POSADR) pour SRCADR
0771 003EF          • Historique:
0772 003EF          •   88/07/06: PD/JT séparation de la routine de recherche
0773 003EF          •   88/07/10: PD/JT ajout de SRCADR
0774 003EF          •   88/10/03: PD/JT adaptation au nouveau POSADR
0775 003EF          ********************************************************
0776 003EF
0777 003EF          ldeb    EQU     (=FUNCR0)+00    5 quartets
0778 003EF          lfin    EQU     (=FUNCR0)+05    5 quartets
0779 003EF          curlin  EQU     (=FUNCR0)+10    5 quartets
0780 003EF          sauvR0  EQU     (=FUNCR0)+15    8 quartets
0781 003EF
0782 003EF          =SRCLIN
0783 003EF          •
0784 003EF          • Sauvegarde pendant seek
0785 003EF          •
0786 003EF 1F00000          D1=(5)  ldeb    ldeb := A(A)
0787 003F6 141              DAT1=A  A
0788 003F9 1D00             D1=(2)  lfin    lfin := C(A)
0789 003FD 145              DAT1=C  A
0790 00400 1D00             D1=(2)  sauvR0  sauvR0 := R0
0791 00404 118              C=R0
0792 00407 15D7             DAT1=C  8
0793 0040B
0794 0040B D6               C=A     A       C(A) := ...
0795 0040D 7D7D             GOSUB   =seek
0796 00411
0797 00411 1F00000          D1=(5)  sauvR0  R0 := sauvR0
0798 00418 15F7             C=DAT1  8
0799 0041C 108              R0=C    A
0800 0041F D1               B=0     A       offset := 0
```

```
Page 017          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

0801 00421 6410             GOTO    SRC100
0802 00425
0803 00425 1F0000  =SRCADR D1=(5)  ldeb    ldeb := A(A)
0804 0042C 141              DAT1=A  A
0805 0042F 1D00             D1=(2)  lfin    lfin := C(A)
0806 00433 145              DAT1=C  A
0807 00436
0808 00436           SRC100
0809 00436           *
0810 00436           * Invariant de boucle
0811 00436           *   ldeb = no ligne courante
0812 00436           *   lfin = no ligne fin
0813 00436           *   D0 = adresse ligne courante
0814 00436           *   B(A) = offset = offset dans la ligne courante
0815 00436           *
0816 00436 136              CD0EX           curlin := D0
0817 00439 1B0000           D0=(5)  curlin
0818 0043D 144              DAT0=C  A
0819 00443 134              D0=C            pour LIFlen
0820 00446
0821 00446 723E             GOSUB   =LIFlen
0822 0044A DA               A=C     A       A(A) := longueur en octets
0823 0044C E0               A=A-B   A       A(A) := longueur de ce qu'il reste
0824 0044E 163              D0=D0+  4       pour POSADR
0825 00451 136              CD0EX
0826 00454 109              R1=C            R1 := ^ vrai début
0827 00457 C9               C=C+B   A
0828 00459 C9               C=C+B   A
0829 0045B 134              D0=C            D0 := ^ début partie à chercher
0830 0045E           *
0831 0045E           *   D0 = ^ chaîne
0832 0045E           *   A(A) = longueur de la chaîne
0833 0045E           *   R0 = buffer
0834 0045E           *   R1(A) = ^ vrai début
0835 0045E           *
0836 0045E 8E0000           GOSUBL  =POSADR
0837 00464 404              GOC     SRC200  trouvé !
0838 00467
0839 00467 D1               B=0     A       offset := 0
0840 00469
0841 00469 1B0000           D0=(5)  curlin
0842 0046D 146              C=DAT0  A
0843 00473 134              D0=C            D0 := ^ ligne courante
0844 00476 720E             GOSUB   =LIFlen
0845 0047A 136              CD0EX
0846 0047D C2               C=C+A   A
0847 0047F 134              D0=C            D0 := ^ ligne suivante
0848 00482
0849 00482 1F0000           D1=(5)  lfin
0850 00489 143              A=DAT1  A       A(A) := lfin
```

```
Page 018          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

0851 0048C 1D00             D1=(2)  ldeb
0852 00490 147              C=DAT1  A       C(A) := ldeb
0853 00493 E6               C=C+1   A
0854 00495 145              DAT1=C  A       ^^ldeb
0855 00498 8BE              ?C?=A   A       ldeb ?= lfin ?
0856 0049B B9               GOYES   SRC100  oui : on continue
0857 0049D           *
0858 0049D           * Arrivé à la fin sans avoir trouvé
0859 0049D           *
0860 0049D 330000           LC(4)   (=id)"(=eNFND)  "Not Found"
0861 004A3 03               RTNCC
0862 004A5
0863 004A5           *
0864 004A5           * Arrivé à l'occurrence
0865 004A5           *
0866 004A5 D5        SRC200 B=C     A       B(A) := longueur de l'occurrence
0867 004A7 1F0000           D1=(5)  ldeb    A(A) := ^ numéro de ligne
0868 004AE 143              A=DAT1  A
0869 004B1 1D00             D1=(2)  curlin  D1 := ^ début ligne LIF
0870 004B5 147              C=DAT1  A
0871 004B8 135              D1=C
0872 004BB D9               C=B     A       C(A) := longueur de l'occurrence
0873 004BD 02               RTNSC           match found
0874 004BF
0875 004BF           ********************************************************
0876 004BF           * DELIM
0877 004BF           *
0878 004BF           * But: analyse une chaîne de recherche (ou de remplacement)
0879 004BF           *   suivant le mode (HP-UX ou HP-71). La chaîne est terminée
0880 004BF           *   par le délimiteur (le premier caractère de la chaîne),
0881 004BF           *   ou par la fin de la chaîne.
0882 004BF           * Entrée:
0883 004BF           *   - D1 = ^ premier caractère (le délimiteur)
0884 004BF           *   - D(A) = longueur totale de la chaîne
0885 004BF           * Sortie:
0886 004BF           *   - Cy = 0 : pas d'erreur
0887 004BF           *        - D1 et D(A) réactualisés (avant / final ou sur EOL)
0888 004BF           *        - A(A) = ^ début de la chaîne trouvée
0889 004BF           *        - C(A) = nombre de caractères dans la chaîne trouvée
0890 004BF           *   - Cy = 1 : erreur
0891 004BF           *        - C(4-0) = numéro d'erreur
0892 004BF           * Abîme: A-D, D0, D1
0893 004BF           * Appelle: getchr
0894 004BF           * Niveaux: 1
0895 004BF           * Note : en sortie, D1 pointe sur le "/", ou sur le EOL.
0896 004BF           *   Ceci est fait pour pouvoir distinguer le cas
0897 004BF           *   "R/toto/" du cas "R/toto". Cela est plus embêtant, car
0898 004BF           *   l'appelant doit explicitement passer le délimiteur. Mais
0899 004BF           *   c'est la seule manière de faire (simplement).
0900 004BF           * Historique:
```

```
Page 019          Editeur, gestion du texte <xtext.as>
AREUH'ASS. V2.4

0901 004BF           *   88/10/15: PD/JT séparation de GETSRC
0902 004BF           *   89/08/05: PD    correction de la documentation
0903 004BF           ********************************************************
0904 004BF
0905 004BF 330000    invprm LC(4)   =eILPAR "Invalid Parm"
0906 004C5 02               RTNSC           Erreur
0907 004C7
0908 004C7           =DELIM
0909 004C7           *
0910 004C7           * Chercher le caractère délimiteur
0911 004C7           *
0912 004C7 8E0000           GOSUBL  =getchr caractère délimiteur
0913 004CD 41F              GOC     invprm  pas présent : "Invalid Parm"
0914 004D0 AE8              B=A     B       B(A) := caractère délimiteur
0915 004D3           *
0916 004D3           * En fonction du mode, analyser en tenant compte des "\"
0917 004D3           * ou non :
0918 004D3           *
0919 004D3 1B0000           D0=(5)  =MODE71
0920 004DA 1564             C=DAT0  S
0921 004DE 137              CD1EX
0922 004E1 134              D0=C            Sauvegarde de D1 dans D0
0923 004E4 135              D1=C
0924 004E7 94A              ?C=0    S       mode HP-UX
0925 004EA 31               GOYES   DELI20
0926 004EC           *
0927 004EC           * Mode HP-71 : on recherche bêtement le caractère B(B)
0928 004EC           *
0929 004EC 8E0000    DELI10 GOSUBL  =getchr
0930 004F2 443              GOC     DELI70  EOL trouvé.
0931 004F5 964              ?A#B    B       délimiteur ?
0932 004F8 4F               GOYES   DELI10  non : on continue
0933 004FA 572              GONC    DELI60  B.E.T. délimiteur trouvé.
0934 004FD
0935 004FD           *
0936 004FD           * Mode HP-UX : le caractère B(B) peut être précédé par "\"
0937 004FD           *
0938 004FD 31C5      DELI20 LCASC   '\'
0939 00501 AC1       DELI30 B=0     S       état := 0
0940 00504 8E0000    DELI40 GOSUBL  =getchr
0941 0050A 4C1              GOC     DELI70  EOL trouvé.
0942 0050D 94D              ?B#0    S       selon état
0943 00510 1F               GOYES   DELI30  état 1 : on repasse en état 0
0944 00512           * etat 0 : pas de '\'
0945 00512 960              ?A=B    B       caractère délimiteur ?
0946 00515 D0               GOYES   DELI60  Délimiteur trouvé
0947 00517 966              ?A#C    B       '\' ?
0948 0051A AE               GOYES   DELI40  non : on continue en état 0
0949 0051C B45              B=B+1   S       état := 1
0950 0051F 54E              GONC    DELI40  B.E.T.
```

```
Page 020          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

0951 00522
0952 00522           *
0953 00522           * Pour le cas ou on est arrivé sur le délimiteur,
0954 00522           * on revient en arrière. A charge pour l'appelant de
0955 00522           * passer un caractère.
0956 00522           *
0957 00522 171       DELI60 D1=D1+  2       la M.S. va dans le sens inverse
0958 00525 E7               D=D+1   A
0959 00527           *
0960 00527           * Cas EOL : on laisse D1 et D(A) en l'état.
0961 00527           *
0962 00527 AD2       DELI70 C=0     M       pour le CSRB de tout à l'heure
0963 0052A 137              CD1EX
0964 0052D 135              D1=C
0965 00530           *
0966 00530           * D0 = ^ début
0967 00530           * C(A) = ^ dernier caractère de la chaîne
0968 00530           * D1 = ^ caractère suivant (sur le "/", ou sur EOL)
0969 00530           * D1 et D(A) sont à préserver
0970 00530           *
0971 00530 132              AD0EX
0972 00533 130              D0=A            on garde D0 encore un peu au chaud
0973 00536 EE               C=A-C   A
0974 00538 81E              CSRB            C(A) := nombre de caractères
0975 0053B 132              AD0EX           A(A) := ^ début de la chaîne
0976 0053E           *
0977 0053E           * A(A) = ^ début de la chaîne trouvée
0978 0053E           * C(A) = nombre de caractères dans la chaîne trouvée
0979 0053E           * D1 et D(A) réactualisés
0980 0053E           *
0981 0053E 03               RTNCC           pas d'erreur
0982 00540
0983 00540           ********************************************************
0984 00540           * GETSRC
0985 00540           *
0986 00540           * But: analyse et compile une chaîne de recherche suivant le
0987 00540           *   mode (HP-UX ou HP-71). La chaîne est terminée par le
0988 00540           *   délimiteur (le premier caractère de la chaîne), ou par
0989 00540           *   la fin de la chaîne.
0990 00540           * Entrée:
0991 00540           *   - D1 = ^ premier caractère (le délimiteur)
0992 00540           *   - D(A) = longueur totale de la chaîne
0993 00540           * Sortie:
0994 00540           *   - Cy = 0 : pas d'erreur
0995 00540           *        - R0(7-0) = caractéristiques du buffer compilation
0996 00540           *        - D1 et D(A) réactualisés (avant "/" ou sur EOL)
0997 00540           *   - Cy = 1 : erreur
0998 00540           *        - C(4-0) = numéro d'erreur
0999 00540           * Abîme: A-D, R0-R3, D0, D1
1000 00540           * Appelle: DELIM, COMPUX ou COMP71
```

```
Page 021          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1001 00540          • Niveaux: 5
1002 00540          • Note : l'appelant doit passer explicitement le "/" final.
1003 00540          • Historique:
1004 00540          •   88/07/09: PD/JT conception & codage
1005 00540          •   88/10/15: PD/JT séparation de DELIM
1006 00540          ********************************************************
1007 00540
1008 00540 738F     =GETSRC GOSUB   =DELIM
1009 00544 400              RTNC            erreur lors de la recherche du '/'
1010 00547          •
1011 00547          • A(A) = ^ début de la chaine trouvée
1012 00547          • C(A) = nombre de caractères dans la chaine trouvée
1013 00547          • D1 et D(A) réactualisés
1014 00547          •
1015 00547 06               RSTK=C          sauvegarder le nombre de caracteres
1016 00549 137              CD1EX
1017 0054C 8F00000          GOSBVL  =CSLC5
1018 00553 DB               C=D     A
1019 00555 10B              R3=C            R3(9-5) := D1 ; R3(A) := D(A)
1020 00558
1021 00558 131              D1=A            D1 :=  début de la chaine trouvée
1022 0055B 07               C=RSTK
1023 0055D DA               A=C     A       A(A) := nombre de caractères
1024 0055F          •
1025 0055F          • A(A) = nombre de caractères
1026 0055F          • D1 = ^ début de la chaine trouvée
1027 0055F          • R3(9-5,4-0) = partie restante après la chaine trouvée
1028 0055F          •
1029 0055F 1B00000          D0=(5)  =MODE71
1030 00566 1564             C=DAT0  S
1031 0056A 94A              ?C#0    S
1032 0056D C0               GOYES   GETS80
1033 0056F 8E0000           GOSUBL  =COMP71
1034 00575 6900             GOTO    GETS90
1035 00579 8E0000   GETS80 GOSUBL   =COMPHX
1036 0057F          •
1037 0057F          • Compilation terminée. Restauration de D1 et D(A) sans
1038 0057F          • abimer C(A) et Carry
1039 0057F          •
1040 0057F DA       GETS90 A=C      A       sauvegarde de C(A)
1041 00581 11B              C=R3
1042 00584 D7               D=C     A       restauration de D(A)
1043 00586 8F00000          GOSBVL  =CSRC5
1044 0058D 135              D1=C            restauration de D1
1045 00590 D6               C=A     A       restauration de C(A)
1046 00592 01               RTN             sans abimer la Cy
1047 00594
1048 00594          ********************************************************
1049 00594          • FMTLIF
1050 00594          •
```

```
Page 022          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1051 00594          • But: formatte une chaine placée en (OUTBS..AVMEMS) suivant
1052 00594          •   les conventions LIF. La ligne est déjà placée 4 quartets
1053 00594          •   après OUTBS (pour laisser de la place pour la longueur)
1054 00594          • Entrée:
1055 00594          •   - OUTBS et AVMEMS positionnes
1056 00594          •   - S0 = 1 si l'octet de padding doit être ajouté
1057 00594          • Sortie:
1058 00594          •   - Cy = 0 : pas d'erreur
1059 00594          •        AVMEMS réajusté si besoin est et si S0=1.
1060 00594          •        D0 = ^ début de la ligne (longueur LIF) dans OUTBS
1061 00594          •   - Cy = 1 : ligne trop longue (> 64 Ko)
1062 00594          •        C(3-0) = eL2LNG
1063 00594          • Abime: A(A), C(W), D0
1064 00594          • Appelle: LIFlen
1065 00594          • Niveaux: 1
1066 00594          • Historique:
1067 00594          •   88/10/15: PD/JT conception & codage
1068 00594          •   88/10/30: PD/JT ajout du flag 0
1069 00594          ********************************************************
1070 00594
1071 00594 1B00000  =FMTLIF D0=(5)  =AVMEMS
1072 0059B AD2              C=0     M
1073 0059E 146              C=DAT0  A
1074 005A1 184              D0=D0-  5       D0 := OUTBS
1075 005A4 142              A=DAT0  A
1076 005A7 E2               C=C-A   A       C(A) := longueur totale en quartets
1077 005A9 81E              CSRB            C(A) := longueur en octets * 2
1078 005AC CE               C=C-1   A
1079 005AE CE               C=C-1   A
1080 005B0          • Test de longueur < 65536
1081 005B0 D0               A=0     A
1082 005B2 23               P=      3
1083 005B4 A9A              A=C     WP      A(A) := longueur en octets (SWPBYT)
1084 005B7 20               P=      0
1085 005B9 8A6              ?A#C    A
1086 005BC 53               GOYES   FMTERR
1087 005BE          • Faut-il ajouter l'octet de padding ?
1088 005BE 860              ?ST=0   0
1089 005C1 D1               GOYES   FMTL10  non
1090 005C3 822              SB=0
1091 005C6 81E              CSRB
1092 005C9 832              ?SB=0           longueur paire ?
1093 005CC 21               GOYES   FMTL10  oui. C'est très bien comme ça.
1094 005CE          • Il faut ajouter 2 quartets à AVMEMS
1095 005CE 164              D0=D0+  5       D0 := AVMEMS
1096 005D1 146              C=DAT0  A
1097 005D4 E6               C=C+1   A
1098 005D6 E6               C=C+1   A
1099 005D8 144              DAT0=C  A       AVMEMS += 2
1100 005DB 184              D0=D0-  5       D0 := OUTBS
```

```
Page 023          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1101 005DE          • Placer la longueur LIF (qui est dans A) en OUTBS
1102 005DE 146      FMTL10 C=DAT0  A
1103 005E1 134              D0=C            D0 := ^ début ligne (long LIF)
1104 005E4 8F0000           GOSBVL  =SWPBYT
1105 005EB 1583             DAT0=A  4
1106 005EF 03               RTNCC           Cy = 0 : pas d'erreur
1107 005F1
1108 005F1 330000   FMTERR LC(4)   =eL2LNG "Line Too Long"
1109 005F7 02               RTNSC           Cy = 1
1110 005F9
1111 005F9          ********************************************************
1112 005F9          • MVFILE
1113 005F9          •
1114 005F9          • But: déplace une partie d'un fichier dans un autre, en
1115 005F9          •   supprimant éventuellement la partie "source"
1116 005F9          • Entrée:
1117 005F9          •   - A(A) = ^ header origine
1118 005F9          •   - R1(A) = ^ début partie à déplacer
1119 005F9          •   - R2(A) = ^ fin partie à déplacer
1120 005F9          •   - C(A) = ^ header destination
1121 005F9          •   - R0(A) = ^ emplacement où doit être inséré le bloc
1122 005F9          •   - sMOVE = 1 si l'origine doit être détruite après
1123 005F9          • Sortie:
1124 005F9          •   - Cy = 0 : pas d'erreur
1125 005F9          •        A(A) = ^ header origine mis-à-jour
1126 005F9          •        C(A) = ^ header destination mis-à-jour
1127 005F9          •   - Cy = 1 : erreur
1128 005F9          •        C(3-0) = numéro d'erreur
1129 005F9          •        Aucun des deux fichiers n'a bougé
1130 005F9          • Abime: A-D, R0-R3, D0, D1, FUNCRx
1131 005F9          • Appelle: ????? RAMROM, MOVE*M, RPLLIN, LIFlen
1132 005F9          • Niveaux: ???????
1133 005F9          • Algorithme:
1134 005F9          •   Paramètres :
1135 005F9          •     orghdr = header du fichier origine
1136 005F9          •     orgdeb = début de l'origine
1137 005F9          •     orgfin = fin de l'origine
1138 005F9          •     dsthdr = header du fichier destination
1139 005F9          •     dstadr = adresse destination courante
1140 005F9          •     taille = orgfin - orgdeb
1141 005F9          •
1142 005F9          •   /* Phase I : Allouer un buffer */
1143 005F9          •
1144 005F9          •   allouer un buffer temporaire et stocker les paramètres
1145 005F9          •
1146 005F9          •   /* Phase II : Assez de place pour la destination ? */
1147 005F9          •
1148 005F9          •   Tester la contrainte de place suivant tableau ci-dessous
1149 005F9          •   mem := MEM(device destination)
1150 005F9          •
```

```
Page 024          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1151 005F9          •   /* Phase III : Détection des cas particuliers */
1152 005F9          •
1153 005F9          •   si dstadr in [orgdeb...orgfin]
1154 005F9          •     alors
1155 005F9          •       si cas "copie"
1156 005F9          •         alors
1157 005F9          •           dstadr := valeur originale (sans updates)
1158 005F9          •           insérer
1159 005F9          •             taille = taille quartets
1160 005F9          •             début = dstadr
1161 005F9          •           fin insérer
1162 005F9          •           copier
1163 005F9          •             début source = orgdeb
1164 005F9          •             fin source = dstadr
1165 005F9          •             début dest = dstadr
1166 005F9          •           fin copier
1167 005F9          •           copier
1168 005F9          •             début source = dstadr + taille
1169 005F9          •             fin source = orgfin après update
1170 005F9          •             début dest = dstadr + (dstadr - orgdeb)
1171 005F9          •           fin copier
1172 005F9          •       fin si
1173 005F9          •       désallouer le buffer
1174 005F9          •       sortir
1175 005F9          •   fin si
1176 005F9          •
1177 005F9          •   /* Phase IV : Recopie dans le fichier */
1178 005F9          •
1179 005F9          •   tant que orgdeb # orgfin
1180 005F9          •
1181 005F9          •     /* Phase IV.a : prendre le maximum */
1182 005F9          •
1183 005F9          •     l2 := orgdeb
1184 005F9          •     tant que l2 # orgfin et (l2-orgdeb)+LIFlen(l2)<=mem
1185 005F9          •       l2 += LIFlen(l2)
1186 005F9          •     fin tant que
1187 005F9          •
1188 005F9          •     si orgdeb == l2
1189 005F9          •       alors
1190 005F9          •         /* cas IV.b : ligne trop longue */
1191 005F9          •
1192 005F9          •         l2 += LIFlen(l2)
1193 005F9          •         tant que orgdeb < l2
1194 005F9          •           x := min (mem, l2-orgdeb)
1195 005F9          •
1196 005F9          •           insérer
1197 005F9          •             taille = x quartets
1198 005F9          •             début dest = dstadr
1199 005F9          •           fin insérer
1200 005F9          •           copier
```

```
Page 025          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1201 005F9          *             début source = orgdeb
1202 005F9          *             fin source = orgdeb + x
1203 005F9          *             début dest = dstadr
1204 005F9          *           fin copier
1205 005F9          *           /* le cas "copie" n'est pas possible ici */
1206 005F9          *           détruire
1207 005F9          *             début = orgdeb
1208 005F9          *             fin = orgdeb + x
1209 005F9          *           fin détruire
1210 005F9          *           dstadr += x
1211 005F9          *         fin tant que
1212 005F9          *       sinon
1213 005F9          *         /* cas IV.c : on a trouvé des lignes */
1214 005F9          *
1215 005F9          *         x := 12 - orgdeb
1216 005F9          *
1217 005F9          *         insérer
1218 005F9          *           taille = x quartets
1219 005F9          *           début dest = dstadr
1220 005F9          *         fin insérer
1221 005F9          *         copier
1222 005F9          *           début source = orgdeb
1223 005F9          *           fin source = orgdeb + x
1224 005F9          *           début dest = dstadr
1225 005F9          *         fin copier
1226 005F9          *         si cas "copie"
1227 005F9          *           alors orgdeb += x
1228 005F9          *           sinon
1229 005F9          *             détruire
1230 005F9          *               début = orgdeb
1231 005F9          *               fin = orgdeb + x
1232 005F9          *             fin détruire
1233 005F9          *         fin si
1234 005F9          *         dstadr += x
1235 005F9          *     fin si
1236 005F9          *   fin tant que
1237 005F9          *   désallouer le buffer
1238 005F9          *   sortir
1239 005F9          *
1240 005F9          * Détail:
1241 005F9          *   Contraintes de mémoire
1242 005F9          *   - MEM = taille disponible dans le device destination
1243 005F9          *     Le device étant :MAIN ou :PORT
1244 005F9          *   - taille du bloc (du fichier origine) à copier
1245 005F9          *
1246 005F9          *       origine |       :PORT          |       :MAIN      |
1247 005F9          *  destination  |                       |                  |
1248 005F9          *  -------------+-----------------------+------------------|
1249 005F9          * |:PORT | copy | MEM > taille          | MEM > taille     |
1250 005F9          * |      |------+-----------------------+------------------|
```

```
Page 026          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1251 005F9          * |      | move | port = | MEM > 512    |                  |
1252 005F9          * |      |      |--------+--------------| MEM > taille     |
1253 005F9          * |      |      | port # | MEM > taille |                  |
1254 005F9          * |------+------+-----------------------+------------------|
1255 005F9          * |:MAIN | copy | MEM > taille          | MEM > taille     |
1256 005F9          * |      |------+-----------------------+------------------|
1257 005F9          * |      | move | MEM > taille          | MEM > 512        |
1258 005F9          *  ------------------------------------------------------
1259 005F9          *
1260 005F9          * On demande au moins MEM > 512 pour pouvoir transférer
1261 005F9          * les lignes raisonablement vite. Ceci n'arrive que lorsque
1262 005F9          * on fait un "move" à l'intérieur de la même unite.
1263 005F9          *
1264 005F9          * Cette mémoire est calculée une seule fois. En cas de copie
1265 005F9          * ou de déplacement dans des péripheriques différents, le
1266 005F9          * transfert est fait en une seule passe, d'où il n'y a pas
1267 005F9          * besoin de la recalculer. En cas de déplacement dans le
1268 005F9          * même périphérique, la mémoire n'est pas modifiée par
1269 005F9          * définition.
1270 005F9          *
1271 005F9          * Note: Les fichiers sont supposés inscriptibles.
1272 005F9          *
1273 005F9          * Historique:
1274 005F9          *   88/11/01: PD/JT conception de l'algorithme
1275 005F9          *   88/11/01: PD/JT début du codage
1276 005F9          *   88/11/06: PD/JT continuation du codage
1277 005F9          *   88/11/11: PD/JT debogage et fin du codage
1278 005F9          *   88/11/11: PD/JT ajout du test "Illegal Access"
1279 005F9          *   88/11/12: PD/JT retrait du test "Illegal Access"
1280 005F9          ************************************************************
1281 005F9          *
1282 005F9          *
1283 005F9          * Données placées dans le buffer. Les valeurs définies ici
1284 005F9          * sont les offsets par rapport au début du buffer.
1285 005F9          *
1286 005F9           orghdr EQU     00
1287 005F9           orgdeb EQU     05
1288 005F9           orgfin EQU     10
1289 005F9           dsthdr EQU     15
1290 005F9           dstadr EQU     20
1291 005F9           12     EQU     25      ;sauvegarde pendant cas pathologique
1292 005F9           rsti   EQU     30      3 niveaux de pile (15 q)
1293 005F9           MVBUFS EQU     45      taille du buffer
1294 005F9           
1295 005F9          *
1296 005F9          * Données en mémoire statique. Il s'agit ici de FUNCR0
1297 005F9          *
1298 005F9           mvbid  EQU     00+*FUNCR0      3 q : id du buffer
1299 005F9           mvbadr EQU     03+*FUNCR0      5 q : adresse du buffer
1300 005F9           taille EQU     08+*FUNCR0      5 q : orgfin - orgdeb
```

```
Page 027          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1301 005F9           mvmem  EQU     13+*FUNCR0      5 q : MEM (device dest.)
1302 005F9           xxxx   EQU     18+*FUNCR0      x q :
1303 005F9
1304 005F9 02        mverr  RTNSC
1305 005FB
1306 005FB          *+MOVE  EQU     0
1307 005FB
1308 005FB          *MVFILE
1309 005FB          ****************************************
1310 005FB          * Phase I : Allouer un buffer
1311 005FB          ****************************************
1312 005FB
1313 005FB           mvf100
1314 005FB          *
1315 005FB          * Sauver temporairement A(A) et C(A) dans R3 car les
1316 005FB          * routines de buffer les abimeraient.
1317 005FB          *
1318 005FB 8F00000          GOSBVL  =CSLC5  C(9-5) := C(A)  (dsthdr)
1319 00602 D6               C=A     A       C(4-0) := A(A)  (orghdr)
1320 00604 10B              R3=C            R3(9-5) := C(A) ; R3(4-0) := A(A)
1321 00607          *
1322 00607          * R3(4-0) = orghdr
1323 00607          * R3(9-5) = dsthdr
1324 00607          *
1325 00607 8F00000          GOSBVL  =IOFSCR find an available scratch buffer id
1326 0060E 590              GONC    mvf110  ok
1327 00611 8D00000          GOVLNG  =CORUPT No available buffer id
1328 00618          *
1329 00618          * C(X) = id disponible
1330 00618          *
1331 00618 1B00000   mvf110 D0=(5)  mvbid
1332 0061F 1543             DAT0=C  X       mvbid := id trouvé
1333 00623 AB5              B=C     X       B(X) := id trouvé
1334 00626 D2               C=0     A
1335 00628 31D2             LC(2)   MVBUFS  C(A) := taille du buffer
1336 0062C DD               BCEX    A       B(A) := taille ; C(A) := id
1337 0062E          *
1338 0062E          * C(X) = buffer id
1339 0062E          * B(A) = longueur demandée
1340 0062E          *
1341 0062E 8F00000          GOSBVL  =I/OALL Cy = 0 : no room
1342 00635 53C              GONC    mverr   RTNSC
1343 00638          *
1344 00638          * Recopie des paramètres dans le buffer
1345 00638          * D1 = ^ past buffer header
1346 00638          *
1347 00638 1B00000          D0=(5)  mvbadr
1348 0063F 137              CD1EX
1349 00642 135              D1=C
1350 00645 144              DAT0=C  A       mvbadr := adresse du buffer
```

```
Page 028          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1351 00648
1352 00648 1C6              D1=D1-  7
1353 0064B 309              LC(1)   (MVBUFS)/5 nb d'adresses à réactualiser
1354 0064E 1500             DAT1=C  1       nb adresses := 6
1355 00652 176              D1=D1+  7
1356 00655
1357 00655 113              A=R3            A(A) := orghdr
1358 00658 141              DAT1=A  A       orghdr
1359 0065B 174              D1=D1+  5
1360 0065E 111              A=R1
1361 00661 141              DAT1=A  A       orgdeb
1362 00664 174              D1=D1+  5
1363 00667 112              A=R2
1364 0066A 141              DAT1=A  A       orgfin
1365 0066D 174              D1=D1+  5
1366 00670
1367 00670 11B              C=R3
1368 00673 8F00000          GOSBVL  =CSRC5  C(A) := dsthdr
1369 0067A 145              DAT1=C  A       dsthdr
1370 0067D 174              D1=D1+  5
1371 00680 118              C=R0
1372 00683 145              DAT1=C  A       dstadr
1373 00686
1374 00686 1F00000          D1=(5)  taille
1375 0068D 11A              C=R2            C(A) := orgfin
1376 00690 111              A=R1            A(A) := orgdeb
1377 00693 E2               C=C-A   A       C(A) := taille
1378 00695 145              DAT1=C  A       taille ne doit pas être actualisée
1379 00698
1380 00698          ****************************************
1381 00698          * Phase II : Assez de place pour la destination ?
1382 00698          ****************************************
1383 00698
1384 00698          *
1385 00698          * Localiser le device du fichier origine
1386 00698          *
1387 00698 119       mvf200 C=R1            C(A) := orgdeb (dans origine)
1388 0069B 8F00000          GOSBVL  =LOCADR
1389 006A2          *
1390 006A2          * D(S) = 0 : main
1391 006A2          * D(S) # 0 :    D(S) =  1 : port ram
1392 006A2          *                      2 : port rom
1393 006A2          *                      3 : port eprom
1394 006A2          *              D(0) = extender
1395 006A2          *              D(1) = port number
1396 006A2          *              D(7-2) = rest of config entry
1397 006A2          *
1398 006A2 AFB              C=D     W
1399 006A5 10B              R3=C            R3 =: sauvegarde D(W) origine
1400 006A8          *
```

```
Page 029          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1401 006A8          * Localiser le device du fichier destination
1402 006A8          *
1403 006A8 118              C=R0            C(A) := dstadr
1404 006AB 8F00000          GOSBVL  =LOCADR
1405 006B2 7D72             GOSUB   memD    calcule la mémoire disponible
1406 006B6 1F00000          D1=(5)  mvmem
1407 006BD 145              DAT1=C  A       mvmem := MEM (device destination)
1408 006C0          *
1409 006C0          * A(A) = C(A) = MEM (destination device)
1410 006C0          * D(W) = device info of destination
1411 006C0          * R3 = device info of origine
1412 006C0          *
1413 006C0 860              ?ST=0   =sMOVE
1414 006C3 B3               GOYES   mvf230  tester MEM > TAILLE
1415 006C5          *
1416 006C5          * Cas "move". Est-ce la même unité ?
1417 006C5          *
1418 006C5 11B              C=R3            C(W) := device info of origine
1419 006C8 947              ?D#C    S
1420 006CB 33               GOYES   mvf230  tester MEM > TAILLE
1421 006CD 967              ?D#C    B
1422 006D0 E2               GOYES   mvf230  tester MEM > TAILLE
1423 006D2          *
1424 006D2          * Tester MEM > 512
1425 006D2          *
1426 006D2 3400200          LC(5)   512
1427 006D9 8BE              ?A>=C   A       MEM - 512
1428 006DC 13               GOYES   mvf250  c'est tout bon
1429 006DE
1430 006DE 330000   mvf220 LC(4)   =eMEM   "Insufficient Memory"
1431 006E4 108      mvfer* R0=C             R0(3-0) := numéro d'erreur
1432 006E7 1B00000          D0=(5)  mvbid
1433 006EE 1563             C=DAT0  X
1434 006F2 8F00000          GOSBVL  =I/OOAL
1435 006F9 118              C=R0            C(3-0) := numéro d'erreur
1436 006FC 02               RTNSC           Beeeeeep !
1437 006FE
1438 006FE          *
1439 006FE          * Tester MEM > taille
1440 006FE          *
1441 006FE 1B00000  mvf230 D0=(5)  taille
1442 00705 146              C=DAT0  A
1443 00708 8B2              ?A<C    A
1444 0070B 3D               GOYES   mvf220  "Insufficient Memory"
1445 0070D
1446 0070D          mvf250
1447 0070D
1448 0070D          ****************************************
1449 0070D          * Phase III : Détection des cas particuliers
1450 0070D          ****************************************
```

```
Page 030          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1451 0070D
1452 0070D 1B00000  mvf300 D0=(5)  mvbadr
1453 00714 146              C=DAT0  A       A(A) := ^ début du buffer
1454 00717 135              D1=C            D1 := ^ début du buffer
1455 0071A 174              D1=D1+  orgdeb
1456 0071D 147              C=DAT1  A       C(A) := orgdeb
1457 00720 D5               B=C     A       B(A) := orgdeb
1458 00722 174              D1=D1+  (orgfin)-(orgdeb)
1459 00725 147              C=DAT1  A       C(A) := orgfin
1460 00728
1461 00728 179              D1=D1+  (dstadr)-(orgfin)
1462 0072B 143              A=DAT1  A       A(A) := dstadr
1463 0072E          *
1464 0072E          * A(A) = dstadr
1465 0072E          * B(A) = orgdeb
1466 0072E          * C(A) = orgfin
1467 0072E          *
1468 0072E 8B4              ?A>B    A       dstadr > orgdeb
1469 00731 01               GOYES   mvf400
1470 00733 8B6              ?A<C    A       dstadr < orgfin
1471 00736 B0               GOYES   mvf400
1472 00738          *
1473 00738          * Cas particulier (chevauchement des zones.
1474 00738          *
1475 00738 860              ?ST=0   =sMOVE
1476 0073B A0               GOYES   mvf310  cas "copie"
1477 0073D 6C91             GOTO    mvf900  cas "move" : NOP
1478 00741
1479 00741 6D80     mvf400 GOTO    mvf400  Rallonge
1480 00745
1481 00745          mvf310
1482 00745          *
1483 00745          * insérer
1484 00745          *   taille = "taille" quartets          B(A)
1485 00745          *   début = dstadr                      A(A)
1486 00745          * fin insérer
1487 00745          *
1488 00745 1B00000          D0=(5)  taille
1489 0074C 146              C=DAT0  A       C(A) := taille
1490 0074F D5               B=C     A       B(A) := taille
1491 00751 1900             D0=(2)  mvbadr
1492 00755 142              A=DAT0  A
1493 00758 131              D1=A            D1 := ^ orghdr
1494 0075B 147              C=DAT1  A       C(A) := orghdr
1495 0075E 179              D1=D1+  (orgfin)-(orghdr)
1496 00761 179              D1=D1+  (dstadr)-(orgfin)       D1 := ^ dstadr
1497 00764 143              A=DAT1  A       A(A) := dstadr
1498 00767          *
1499 00767          * R3(A) = dstadr avant update (constante dans les 3 move)
1500 00767          *
```

```
Page 031          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1501 00767 103              R3=A            R3 := dstadr avant update
1502 0076A          *
1503 0076A          * A(A) = starting address to move up
1504 0076A          * B(A) = offset (dest addr - source addr)
1505 0076A          * C(A) = addr of header
1506 0076A          *
1507 0076A 8F00000          GOSBVL  =MGOSUB
1508 00771 00000            CON(5)  =MVMEM*
1509 00776          *
1510 00776          * Cy = 1 : erreur (Illegal Access ou Insufficient  Mem)
1511 00776          *   Impossible car ces deux conditions ont été testées
1512 00776          *   préalablement.
1513 00776          * Cy = 0 : ok
1514 00776          * R3 est préservé de l'update et contient toujours dstadr
1515 00776          *
1516 00776 7B81             GOSUB   syncmv  D1 = A(A) = ^ début du buffer
1517 0077A          *
1518 0077A          * Tester le cas particulier du cas particulier :
1519 0077A          * orgdeb avant = dstadr avant (ex: ...C)
1520 0077A          * Cette égalité est transmise après le MVMEM* dans
1521 0077A          * l'inégalité dstadr avant < orgdeb après.
1522 0077A          * Pourquoi ?
1523 0077A          * Si on avait dstadr avant > orgdeb avant, le MVMEM*
1524 0077A          * n'aurait pas modifié orgdeb. D'où :
1525 0077A          *   => orgdeb avant = orgdeb après
1526 0077A          *   => dstadr avant > orgdeb après
1527 0077A          * Ce qui est le contraire de l'inégalité en cause.
1528 0077A          * D'autre part, si on a dstadr avant = orgdeb avant,
1529 0077A          * orgdeb avant = orgdeb après. D'où
1530 0077A          * dstadr avant < orgdeb après. C.Q.F.D
1531 0077A          *
1532 0077A
1533 0077A          *
1534 0077A          * R3(A) = dstadr avant
1535 0077A          * D1 = ^ orghdr après
1536 0077A          *
1537 0077A 174              D1=D1+  (orgdeb)-(orghdr)
1538 0077D 143              A=DAT1  A       A(A) := orgdeb après
1539 00780 11B              C=R3            C(A) := dstadr avant
1540 00783 8B6              ?C<A    A       si dstadr avant < orgdeb après
1541 00786 54               GOYES   mvf900  sortie
1542 00788          *
1543 00788          * copier
1544 00788          *   début source = orgdeb
1545 00788          *   fin source = dstadr avant update
1546 00788          *   début dest = dstadr avant update
1547 00788          * fin copier
1548 00788          *
1549 00788
1550 00788          *
```

```
Page 032          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1551 00788          * D1 = ^ orgdeb
1552 00788          * A(A) = orgdeb après
1553 00788          * C(A) = dstadr avant
1554 00788          *
1555 00788 D8               B=A     A       B(A) := orgdeb
1556 0078A ED               B=C-B   A       B(A) := dstadr - orgdeb
1557 0078C          *
1558 0078C          * A(A) = source address          (orgdeb)
1559 0078C          * B(A) = length of block         (dstadr - orgdeb)
1560 0078C          * C(A) = dest address            (dstadr)
1561 0078C          *
1562 0078C 8F00000          GOSBVL  =MOVE*M
1563 00793          *
1564 00793          * Copie du dernier bloc (celui après dstadr) :
1565 00793          * copie
1566 00793          *   début source = dstadr + taille
1567 00793          *   fin source = orgfin après update
1568 00793          *   début dest = dstadr + (dstadr - orgdeb)
1569 00793          * fin copie
1570 00793          *
1571 00793 1F00000          D1=(5)  taille
1572 0079A 143              A=DAT1  A       A(A) := taille
1573 0079D 11B              C=R3            C(A) := dstadr
1574 007A0 D7               D=C     A       D(A) := dstadr
1575 007A2 CA               A=A+C   A       A(A) := dstadr + taille (source adr)
1576 007A4
1577 007A4 1D00             D1=(2)  mvbadr
1578 007A8 147              C=DAT1  A
1579 007AB 135              D1=C            D1 := ^ orghdr
1580 007AE 179              D1=D1+  (orgfin)-(orghdr)       D1 := ^ orgfin
1581 007B1 147              C=DAT1  A       C(A) := orgfin
1582 007B4 E2               C=C-A   A       C(A) := orgfin - (dstadr + taille)
1583 007B6 D5               B=C     A       B(A) := length of block
1584 007B8
1585 007B8 1C4              D1=D1-  (orgfin)-(orgdeb)       D1 := ^ orgdeb
1586 007BB 147              C=DAT1  A       C(A) := orgdeb
1587 007BE DF               PDEX    A       C(A) := dstadr ; D(A) := orgdeb
1588 007C0 EF               D=C-D   A       D(A) := dstadr - orgdeb
1589 007C2 CB               C=C+D   A       C(A) := dstadr + (dstadr - orgdeb)
1590 007C4          *
1591 007C4          * A(A) = source address          (dstadr + taille)
1592 007C4          * B(A) = length of block         (orgfin - (dstadr + taille))
1593 007C4          * C(A) = dest address            (dstadr + (dstadr - orgdeb))
1594 007C4          *
1595 007C4 8F00000          GOSBVL  =MOVE*M
1596 007CB          *
1597 007CB          * Sortie en beauté...
1598 007CB          *
1599 007CB 6E01     mvf900 GOTO    mvf900
1600 007CF
```

```
Page 033          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1601 007CF          ****************************************
1602 007CF          * Phase IV : Recopie dans le fichier    *
1603 007CF          ****************************************
1604 007CF
1605 007CF 6SE0      mvf400 GOTO    mvf430  aller directement au test
1606 007D3          *
1607 007D3          * Le test de sortie de boucle se trouve à la fin de
1608 007D3          * cette boucle (mvf490)
1609 007D3          *
1610 007D3 1F00000   mvf405 D1=(5)  mvmem
1611 007DA 147              C=DAT1  A       C(A) := MEM (dest)
1612 007DD 108              R0=C    A       R0(A) := MEM (dest) en quartets
1613 007E0
1614 007E0 1D00             D1=(2)  mvbadr
1615 007E4 147              C=DAT1  A
1616 007E7 135              D1=C            D1 :=  orghdr
1617 007EA 174              D1=D1+  (orgdeb)-(orghdr)
1618 007ED 147              C=DAT1  A       C(A) := orgdeb
1619 007F0 109              R1=C            R1 := orgdeb
1620 007F3          * C(A) = I2
1621 007F3 174              D1=D1+  (orgfin)-(orgdeb)
1622 007F6 143              A=DAT1  A
1623 007F9 102              R2=A            R2 := orgfin
1624 007FC
1625 007FC          ****************************************
1626 007FC          * Cas IV.a : prendre le nombre maximum de lignes
1627 007FC          ****************************************
1628 007FC          *
1629 007FC          * tant que I2 # orgfin et (I2 - orgdeb) + LIFlen(I2) <= MEM
1630 007FC          *   I2 += LIFlen(I2)
1631 007FC          * fin tant que
1632 007FC          *
1633 007FC 6800             GOTO    mvf420  aller directement au test
1634 00800          *
1635 00800          * Affectation des registres durant la boucle :
1636 00800          *   D0 = I2
1637 00800          *   R0 = MEM
1638 00800          *   R1 = orgdeb
1639 00800          *   R2 = orgfin
1640 00800          *
1641 00800           mvf410
1642 00800          *
1643 00800          * D0 = I2
1644 00800          * A(A) = LIFlen (I2)
1645 00800          *
1646 00800 136              CD0EX           C(A) := I2
1647 00803 C2               C=C+A   A       C(A) := I2 + LIFlen (I2)
1648 00805
1649 00805 134       mvf420 D0=C            D0 := I2
1650 00808          * D0 = I2
```

```
Page 034          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1651 00808 112              A=R2            A(A) := orgfin
1652 0080B 8A2              ?A=C    A
1653 0080E B1               GOYES   mvf430  fin tant que
1654 00810
1655 00810 111              A=R1            A(A) := orgdeb
1656 00813 E2               C=C-A   A       C(A) := (I2 - orgdeb)
1657 00815 D7               D=C     A       D(A) := (I2 - orgdeb)
1658 00817 716A             GOSUB   =LIFlen A(A) := taille totale en quartets
1659 0081B          *
1660 0081B          * A(A) = LIFlen (I2). Le contenu de A(A) n'est plus changé
1661 0081B          *
1662 0081B DB               C=D     A       C(A) := (I2 - orgdeb)
1663 0081D C2               C=C+A   A       C(A) := (I2 - orgdeb) + LIFlen(I2)
1664 0081F D5               B=C     A       B(A) := place nécessaire
1665 00821 118              C=R0            C(A) := MEM
1666 00824 8B5              ?B>C    A       place nécessaire > MEM ?
1667 00827 9D               GOYES   mvf410  oui : on continue
1668 00829
1669 00829          *
1670 00829          * fin tant que :
1671 00829          * D0 = I2
1672 00829          * si I2 # orgfin alors on a A(A) = LIFlen (I2)
1673 00829          * R0 = MEM
1674 00829          * R1 = orgdeb
1675 00829          * R2 = orgfin
1676 00829          *
1677 00829 136       mvf430 CD0EX           C(A) := I2
1678 0082C 134              D0=C            D0 := I2
1679 0082F D5               B=C     A       B(A) := I2
1680 00831 119              C=R1            C(A) := orgdeb
1681 00834 8A5              ?B#C    A       Cas pathologique ?
1682 00837 E1               GOYES   Mvf450  Non. On peut copier au moins 1 ligne
1683 00839
1684 00839          ****************************************
1685 00839          * Cas IV.b : ligne trop longue           *
1686 00839          ****************************************
1687 00839          *
1688 00839          * B(A) = D0 = I2
1689 00839          * A(A) = LIFlen (I2)
1690 00839          *   (car I2 == orgdeb et on sait que orgdeb # orgfin)
1691 00839          * R0 = MEM
1692 00839          * C(A) = R1 = orgdeb
1693 00839          * R2 = orgfin
1694 00839          *
1695 00839 C0               A=B+A   A       I2 += LIFlen (I2)
1696 0083B 1F0000           D1=(5)  mvbadr
1697 0083E 147              C=DAT1  A       C(A) :=  orghdr
1698 00841 135              D1=C
1699 00843 174              D1=D1+  (orgfin)-(orghdr)
1700 00846 17E              D1=D1+  (I2)-(orgfin)
```

```
Page 035          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1701 0084E 141              DAT1=A  A       I2 dans le buffer := A(A)
1702 00851 693A             GOTO    mvf445  directement sur le test du "while"
1703 00855
1704 00855 6A50      Mvf450 GOTO    mvf450
1705 00859          *
1706 00859          * Boucle :
1707 00859          * Tant que orgdeb # I2
1708 00859          *   x := min (mem, I2 - orgdeb)
1709 00859          *   insérer x dans le fichier (insfil)
1710 00859          * fin tant que
1711 00859          *
1712 00859 1F00000   mvf440 D1=(5)  mvmem
1713 00860 147              C=DAT1  A       C(A) := MEM (device destination)
1714 00863 D5               B=C     A       B(A) := MEM (device destination)
1715 00865
1716 00865 1D00             D1=(2)  mvbadr
1717 00869 147              C=DAT1  A
1718 0086C 135              D1=C
1719 0086F 174              D1=D1+  (orgdeb)-0
1720 00872 143              A=DAT1  A       A(A) := orgdeb
1721 00875 174              D1=D1+  (orgfin)-(orgdeb)
1722 00878 17E              D1=D1+  (I2)-(orgfin)
1723 0087B 147              C=DAT1  A       C(A) := I2
1724 0087E E2               C=C-A   A       C(A) := I2 - orgdeb
1725 00880          *
1726 00880          * B(A) = mem
1727 00880          * C(A) = I2 - orgdeb
1728 00880          *
1729 00880 8B9              ?C<=B   A       (I2 - orgdeb) <= mem ?
1730 00883 40               GOYES   mvf442  oui: min := I2 - orgdeb
1731 00885 D9               C=B     A       non: min := mem
1732 00887           mvf442
1733 00887          *
1734 00887          * C(A) = x
1735 00887          *
1736 00887 7EC0             GOSUB   insfil
1737 00888
1738 00888          *
1739 00888          * Tester la sortie de la boucle "tant que"
1740 00888          * Dans le buffer, on a :
1741 00888          *   I2 = adresse de fin de la boucle
1742 00888          *   orgdeb = adresse courante
1743 00888          * Test de fin : orgdeb == I2
1744 00888          *
1745 00888 1F00000   mvf445 D1=(5)  mvbadr
1746 00892 147              C=DAT1  A       C(A) := ^ début des données
1747 00895 135              D1=C
1748 00898 174              D1=D1+  (orgdeb)-0
1749 0089B 147              C=DAT1  A       C(A) := orgdeb
1750 0089E 174              D1=D1+  (orgfin)-(orgdeb)
```

```
Page 036          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1751 008A1 17E              D1=D1+  (I2)-(orgfin)
1752 008A4 143              A=DAT1  A       A(A) := I2
1753 008A7          *
1754 008A7          * C(A) = orgdeb
1755 008A7          * A(A) = I2
1756 008A7          *
1757 008A7 8B6              ?C<A    A       orgdeb < I2
1758 008AA FA               GOYES   mvf440  oui : alors on répète
1759 008AC          *
1760 008AC          * non : fin de la boucle.
1761 008AC          *
1762 008AC 6B00             GOTO    mvf490
1763 008B0
1764 008B0
1765 008B0          ****************************************
1766 008B0          * Cas IV.c : on peut copier au moins une ligne
1767 008B0          ****************************************
1768 008B0          *
1769 008B0          * B(A) = D0 = I2
1770 008B0          * R0 = MEM
1771 008B0          * C(A) = R1 = orgdeb
1772 008B0          * R2 = orgfin
1773 008B0          *
1774 008B0
1775 008B0 E1        mvf450 B=B-C   A       B(A) := I2 - orgdeb
1776 008B2 D9               C=B     A       C(A) := x
1777 008B4
1778 008B4 71A0             GOSUB   insfil
1779 008B8
1780 008B8          *
1781 008B8          * Test de sortie : "orgdeb == orgfin"
1782 008B8          *
1783 008B8 1F00000   mvf490 D1=(5)  mvbadr
1784 008BF 147              C=DAT1  A       C(A) := ^ début du buffer
1785 008C2 135              D1=C            D1 := ^ début du buffer
1786 008C5 174              D1=D1+  (orgdeb)-(orghdr)
1787 008C8 147              C=DAT1  A       C(A) := orgdeb
1788 008CB 174              D1=D1+  (orgfin)-(orgdeb)
1789 008CE 143              A=DAT1  A       A(A) := orgfin
1790 008D1 8A2              ?A=C    A       Est-on arrivé au bout ?
1791 008D4 60               GOYES   mvf900  oui : on sort
1792 008D6 6CFE             GOTO    mvf405  non : on continue
1793 008DA
1794 008DA          ****************************************
1795 008DA          * Sortie                                *
1796 008DA          ****************************************
1797 008DA
1798 008DA           mvf900
1799 008DA          *
1800 008DA          * Récupérer les valeurs de xxxhdr mises-à-jour et
```

```
Page 037          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1801 008DA          * supprimer le buffer
1802 008DA          *
1803 008DA 7720             GOSUB   syncmv
1804 008DE          *
1805 008DE          * D1 = ' début des données
1806 008DE          *
1807 008DE 143              A=DAT1  A       A(A) := orghdr
1808 008E1 100              R0=A
1809 008E4 17E              D1=D1+  (dsthdr)-(orghdr)
1810 008E7 147              C=DAT1  A       C(A) := dsthdr
1811 008EA D7               D=C     A
1812 008EC          *
1813 008EC          * R0(A) = orghdr
1814 008EC          * D(A) = dsthdr
1815 008EC          *
1816 008EC 1B00000          D0=(5)  mvbid
1817 008F3 1563             C=DAT0  X
1818 008F7 8F00000          GOSBVL  =I/ODAL
1819 008FE 110              A=R0
1820 00901 DB               C=D     A
1821 00903          *
1822 00903          * A(A) = orgdhr actualizé
1823 00903          * C(A) = dstdhr actualisé
1824 00903          *
1825 00903 03               RTNCC           Pas d'erreur
1826 00905
1827 00905          ****************************************************************
1828 00905          * syncmv
1829 00905          *
1830 00905          * But: recalculer l'adresse du buffer de "MVFILE" après un
1831 00905          *   mouvement de mémoire.
1832 00905          * Entrée:
1833 00905          *   - mvbid = id du buffer utilisé
1834 00905          * Sortie:
1835 00905          *   - D1 = A(A) = mvbadr = adresse du buffer
1836 00905          * Abime: A, C(A), C(S), D1
1837 00905          * Appelle: IOFND0
1838 00905          * Niveaux: 1
1839 00905          * Historique:
1840 00905          *   88/11/06: PD/JT conception & codage
1841 00905          ****************************************************************
1842 00905
1843 00905 1F00000   syncmv D1=(5)  mvbid
1844 0090C 1573             C=DAT1  X
1845 00910 8F00000          GOSBVL  =IOFND0
1846 00917 541              GONC    synerr  "System Error"
1847 0091A 133              ADIEX           A(A) := ' buffer (début des données)
1848 0091D 1F00000          D1=(5)  mvbadr
1849 00924 141              DAT1=A  A       mvbadr := ' buffer (début données)
1850 00927 131              D1=A            D1 := ' buffer (début des données)
```

```
Page 038          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1851 0092A 01               RTN
1852 0092C
1853 0092C 8D00000   synerr GOVLNG  =CORUPT C'est grave... On arrête tout
1854 00933
1855 00933          ****************************************************************
1856 00933          * memD
1857 00933          *
1858 00933          * But: calculer l'espace disponible (en quartets) dans le
1859 00933          *   périphérique identifié par sa caractéristique dans D(W).
1860 00933          *   Si le périphérique est ":MAIN", retranche le leeway.
1861 00933          * Entrée:
1862 00933          *   - D(W) = caractéristique du périphérique
1863 00933          * Sortie:
1864 00933          *   - D(W) = caractéristique du périphérique
1865 00933          *   - A(A) = C(A) = taille disponible en quartets
1866 00933          * Abime: A, C, D1.
1867 00933          * Appelle: FLADDR
1868 00933          * Niveaux: 3
1869 00933          * Historique:
1870 00933          *   88/11/06: PD/JT conception & codage
1871 00933          *   88/11/11: PD/JT prise en compte du Leeway
1872 00933          *   88/11/11: PD/JT supprime "memdev"
1873 00933          ****************************************************************
1874 00933
1875 00933 8F00000   memD   GOSBVL  =FLADDR
1876 0093A          *
1877 0093A          * A(A) := address of first nibble of available memory
1878 0093A          * C(A) := address of last nibble of available memory
1879 0093A          *
1880 0093A E2               C=C-A   A       C(A) := MEM (device)
1881 0093C DA               A=C     A       A(A) := MEM (device)
1882 0093E 94F              ?D#0    S
1883 00941 F0               GOYES   mem10   pas ":MAIN"
1884 00943          *
1885 00943          * On est en ":MAIN". Il faut retrancher le Leeway.
1886 00943          * Toutes les routines qui nous appellent appelleront aussi
1887 00943          * MGOSUB qui sauvegarde 6 quartets dans la pile des GOSUB.
1888 00943          * Il faut donc en tenir compte pour ne pas se heurter à un
1889 00943          * "Insufficient Memory" de la part de MVMEM+ par exemple.
1890 00943          *
1891 00943 D2               C=0     A
1892 00945 3100             LC(2)   (=LEEWAY)+7
1893 00949 EA               A=A-C   A
1894 0094B 460              GOC     mem20   "Insufficient Memory"
1895 0094E D6               C=A     A
1896 00950 03        mem10  RTNCC
1897 00952
1898 00952 8D00000   mem20  GOVLNG  =CORUPT Il ne reste même pas (=LEEWAY)
1899 00959
1900 00959          ****************************************************************
```

```
Page 039          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1901 00959          * insfil
1902 00959          *
1903 00959          * But: déplacer C quartets de l'adresse orgdeb dans le
1904 00959          *   fichier repéré par orghdr vers l'adresse dstadr dans le
1905 00959          *   fichier repéré par dsthdr.
1906 00959          * Entrée:
1907 00959          *   - C(A) = taille du bloc à insérer (symbolisé par "x")
1908 00959          *   - mvbadr = adresse du buffer contenant :
1909 00959          *       orghdr = adresse du header origine
1910 00959          *       orgdeb = adresse de début de l'origine
1911 00959          *       .... = 5 quartets inutilisés ici
1912 00959          *       dsthdr = adresse du header destination
1913 00959          *       dsthdr = adresse de la destination
1914 00959          *   - mvbid = id de ce buffer
1915 00959          *   - sMOVE = 1 s'il faut détruire le bloc origine après
1916 00959          * Sortie:
1917 00959          *   - mvbadr réajusté si besoin est
1918 00959          * Abime: A-D, D0, D1, R0-R3, SCRTCH(4-0)
1919 00959          * Appelle: MVMEM+, MOVE+M
1920 00959          * Niveaux: 3 (voir la note)
1921 00959          * Algorithme:
1922 00959          *   insérer
1923 00959          *     taille = x quartets
1924 00959          *     début = dstadr (qui est réactualisé)
1925 00959          *   fin insérer
1926 00959          *   copier
1927 00959          *     début source = orgdeb
1928 00959          *     taille = x quartets
1929 00959          *     début dest = dstadr (avant réactualisation)
1930 00959          *   fin copier
1931 00959          *   si cas "copie"
1932 00959          *     alors orgdeb += x
1933 00959          *     sinon
1934 00959          *       détruire
1935 00959          *         début = orgdeb
1936 00959          *         taille = x
1937 00959          *       fin détruire
1938 00959          *   fin si
1939 00959          * Note: 3 niveaux de retour sont sauvés dans un endroit où
1940 00959          *   ils sont réactualisés. La routine ne consomme donc
1941 00959          *   quasiment pas de niveau de retour.
1942 00959          * Note: "orgdeb" étant le point de départ de la suppression,
1943 00959          *   cette adresse est remise à 0 pendant la mise à jour dans
1944 00959          *   le buffer. Cette routine préserve cette adresse de cette
1945 00959          *   "mise à jour" intempestive... La valeur de "orgdeb"
1946 00959          *   stockée dans le buffer est donc fiable
1947 00959          * Historique:
1948 00959          *   88/11/06: PD/JT conception & codage
1949 00959          *   88/11/11: PD/JT ajout de la note sur "orgdeb"
1950 00959          *   88/11/11: PD/JT rectification de l'update de "dstadr"
```

```
Page 040          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

1951 00959          ****************************************************************
1952 00959
1953 00959           insfil
1954 00959          *
1955 00959          * On place la taille du bloc à deplacer (x) dans le
1956 00959          * registre R3 d'où elle ne bougera plus pendant tout
1957 00959          * reste de la routine.
1958 00959          *
1959 00959 10B              R3=C            R3 := taille du bloc (x)
1960 0095C          *
1961 0095C          * Pour commencer, on sauve des niveaux de pile dans le
1962 0095C          * buffer, ou ils sont mis à jour en cas de mouvement
1963 0095C          * mémoire.
1964 0095C          *
1965 0095C 1F00000          D1=(5)  mvbadr
1966 00963 143              A=DAT1  A
1967 00966 D2               C=0     A
1968 00968 31E1             LC(2)   (MVBUFS)-3+5
1969 0096C CA               A=A+C   A
1970 0096E 131              D1=A            D1 := ' place pour trois adresses
1971 00971
1972 00971 07               C=RSTK
1973 00973 145              DAT1=C  A
1974 00976 174              D1=D1+  5
1975 00979 07               C=RSTK
1976 0097B 145              DAT1=C  A
1977 0097E 174              D1=D1+  5
1978 00981 07               C=RSTK
1979 00983 145              DAT1=C  A
1980 00986          *
1981 00986          * Trois niveaux de pile sont sauvés
1982 00986          * R3 = taille du bloc à déplacer
1983 00986          *
1984 00986
1985 00986          *
1986 00986          * insérer
1987 00986          *   taille = x quartets
1988 00986          *   début = dstadr
1989 00986          * fin insérer
1990 00986          *
1991 00986 11B              C=R3
1992 00989 D5               B=C     A       B(A) := taille
1993 0098B
1994 0098B 1F00000          D1=(5)  mvbadr
1995 00992 143              A=DAT1  A
1996 00995 D2               C=0     A
1997 00997 31F0             LC(2)   dsthdr
1998 0099B CA               A=A+C   A
1999 0099D 131              D1=A            D1 := ' dsthdr
2000 009A0 147              C=DAT1  A       C(A) := header
```

```
Page 041          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

2001 009A3
2002 009A3 174              D1=D1+  (dstadr)-(dsthdr)
2003 009A6 143              A=DAT1  A       A(A) := starting address
2004 009A9          *
2005 009A9          * A(A) = starting address       (dstadr)
2006 009A9          * B(A) = taille                 (R3)
2007 009A9          * C(A) = header                 (dsthdr)
2008 009A9          *
2009 009A9 8F00000          GOSBVL  =MGOSUB
2010 009B0 00000            CON(5)  =MVMEM+
2011 009B5          *
2012 009B5          * Cy = 1 : erreur
2013 009B5          *   eILACS      (Illegal Access)
2014 009B5          *   eMEM        (Insufficient Memory)
2015 009B5          *   Aucune de ces deux erreurs ne peut se produire.
2016 009B5          * Cy = 0 : ok
2017 009B5          *   R0 = starting address of move (A(A) à l'entrée)
2018 009B5          *   R2 = header (C(A) à l'entrée)
2019 009B5          *
2020 009B5 7C4F             GOSUB   syncmv  le buffer a peut-être bougé
2021 009B9          * D1 = ' orghdr
2022 009B9
2023 009B9          *
2024 009B9          * copier
2025 009B9          *   début source = orgdeb
2026 009B9          *   taille = x quartets
2027 009B9          *   début dest = dstadr (avant update)
2028 009B9          * fin copier
2029 009B9          *
2030 009B9 174              D1=D1+  (orgdeb)-0
2031 009BC 143              A=DAT1  A       A(A) := orgdeb
2032 009BF
2033 009BF 11B              C=R3
2034 009C2 D5               B=C     A       B(A) := x
2035 009C4
2036 009C4 118              C=R0            C(A) := adresse destination
2037 009C7          *
2038 009C7          * A(A) = source                  (orgdeb)
2039 009C7          * B(A) = taille                 (R3)
2040 009C7          * C(A) = destination            (R0)
2041 009C7          *
2042 009C7 8F00000          GOSBVL  =MOVE•M La mémoire ne bouge pas
2043 009CE          *
2044 009CE          * En sortie : on a toutes les conditions d'entrée
2045 009CE          *
2046 009CE 1F00000          D1=(5)  mvbadr  Common Sub-expression Elimination
2047 009D5 147              C=DAT1  A
2048 009D8 135              D1=C
2049 009DB 174              D1=D1+  (orgdeb)-0
2050 009DE 147              C=DAT1  A       C(A) := orgdeb
```

```
Page 042          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

2051 009E1
2052 009E1 87A              ?ST=1   =sMOVE
2053 009E4 E0               GOYES   insf50
2054 009E6          *
2055 009E6          * Cas "copie"
2056 009E6          *
2057 009E6 113              A=R3            A(A) := x
2058 009E9 CC               C=C+A   A       C(A) := orgdeb + x
2059 009EB 145              DAT1=C  A       orgdeb += x
2060 009EE 6920             GOTO    insf90
2061 009F2
2062 009F2          *
2063 009F2          * Cas "move" : il faut détruire ce qu'on vient de copier.
2064 009F2          *
2065 009F2           insf50
2066 009F2          *
2067 009F2          * Note speciale : orgdeb est le début de la zone à détruire
2068 009F2          * Le système met "orgdeb" à 0 après la destruction, car il
2069 009F2          * considère qu'elle n'appartient plus à rien. Cela nous
2070 009F2          * oblige donc à la sauver. Où ? Dans RSTK. C'est le plus
2071 009F2          * simple...
2072 009F2          *
2073 009F2 86               RSTK=C          RSTK := orgdeb
2074 009F4
2075 009F4 DA               A=C     A       A(A) := orgdeb
2076 009F6 C0               A=A+B   A       A(A) := fin du bloc à détruire
2077 009F8 F9               B=-B    A       B(A) := -x
2078 009FA 1C4              D1=D1-  (orgdeb)-(orghdr)
2079 009FD 147              C=DAT1  A       C(A) := orghdr
2080 00A00          *
2081 00A00          * A(A) = starting address       (orgdeb)
2082 00A00          * B(A) = taille                 (-x)
2083 00A00          * C(A) = header                 (orghdr)
2084 00A00          *
2085 00A00 8F00000          GOSBVL  =MGOSUB
2086 00A07 00000            CON(5)  =MVMEM+
2087 00A0C          *
2088 00A0C          * Cy = 1 : erreur (Illegal Access ou Insufficient Mem)
2089 00A0C          *    Impossible eû égard aux tests préalables
2090 00A0C          * Cy = 0 : pas d'erreur
2091 00A0C          *
2092 00A0C 75FE             GOSUB   syncmv  ajuste mvbadr
2093 00A10 174              D1=D1+  (orgdeb)-0
2094 00A13 07               C=RSTK          C(A) := orgdeb
2095 00A15 145              DAT1=C  A       orgdeb := restaure d'après RSTK
2096 00A18          *
2097 00A18          * Sortie.
2098 00A18          *
2099 00A18           insf90
2100 00A18          *
```

```
Page 043          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4

2101 00A18          * Note : dstadr est actualisé par l'insertion du début. En
2102 00A18          * cas "move", dstadr est redescendu. Dans les deux cas, la
2103 00A18          * valeur actuelle de dstadr est la bonne, et il n'y a pas
2104 00A18          * besoin de la changer.
2105 00A18          *
2106 00A18          * Restauration de trois niveaux de pile. L'adresse du
2107 00A18          * buffer est bien dans mvbadr. Les adresses sont dépilées
2108 00A18          * à l'envers comme il se doit...
2109 00A18          *
2110 00A18 1F00000          D1=(5)  mvbadr
2111 00A1F 143              A=DAT1  A
2112 00A22 D2               C=0     A
2113 00A24 3182             LC(2)   (MVBUFS)-5
2114 00A28 CA               A=A+C   A
2115 00A2A 131              D1=A            D1 := ' dernière des trois adresses
2116 00A2D
2117 00A2D 147              C=DAT1  A
2118 00A30 86               RSTK=C
2119 00A32 1C4              D1=D1-  5
2120 00A35 147              C=DAT1  A
2121 00A38 86               RSTK=C
2122 00A3A 1C4              D1=D1-  5
2123 00A3D 147              C=DAT1  A
2124 00A40 86               RSTK=C
2125 00A42
2126 00A42 01               RTN
2127 00A44
2128 00A44                  END
```

```
Page 044          Editeur, gestion du texte <xtext.as>
AREUH ASS. V2.4   •••• SYMBOL TABLE ••••

=ASRW5          Extrn Ukn           0695 0697
=AVMEME         Extrn Ukn       -   0055
=AVMEMS         Extrn Ukn       -   1071
=BUFADR         Extrn Ukn       -   0084 0280 0457 0502 0615 0742
=BUFID          Extrn Ukn       -   0035 0050 0231 0738
 BUFENT         0000A Abs  0007 -   0068 0183 0187 0213 0229 0356 0468 0485 0513
                                +   0639
=COMP71         Extrn Ukn       -   1033
=COMPUX         Extrn Ukn       -   1035
=CORUPT         Extrn Ukn       -   0046 1327 1853 1898
=CSLC5          Extrn Ukn       -   0591 0593 1017 1318
=CSLC6          Extrn Ukn       -   0656
=CSRC5          Extrn Ukn       -   1043 1368
=DELIM          004C7 Rel  0908 -   1008
=DERN           Extrn Ukn       -   0221 0452 0606
 DELI10         004EC Rel  0929 -   0932
 DELI20         004FD Rel  0938 -   0925
 DELI30         00501 Rel  0939 -   0943
 DELI40         00504 Rel  0940 -   0948 0950
 DELI60         00522 Rel  0957 -   0933 0946
 DELI70         00527 Rel  0962 -   0930 0941
 DELTA          00AC8 Abs  0008 -   0125 0175 0217 0307 0311 0345
=FILADR         Extrn Ukn       -   0128 0283
=FLADDR         Extrn Ukn       -   1875
=FMTLIF         00594 Rel  1071 -
=FUNCR0         Extrn Ukn       -   0777 0778 0779 0780 1298 1299 1300 1301 1302
 FMTERR         005F1 Rel  1108 -   1086
 FMTL10         005DE Rel  1102 -   1089 1093
 FiLeNd         00A44 Rel  2128 -
=GETSRC         00540 Rel  1008 -
 GETS80         00579 Rel  1035 -   1032
 GETS90         0057F Rel  1040 -   1034
=I/OALL         Extrn Ukn       -   0081 1341
=I/OCON         Extrn Ukn       -   0233
=I/ODAL         Extrn Ukn       -   0040 1434 1818
=IOFND0         Extrn Ukn       -   0740 1845
=IOFSCR         Extrn Ukn       -   0044 1325
=LEEWAY         Extrn Ukn       -   0061 1892
=LIFlen         0027C Rel  0414 -   0821 0844 1658
=LOCADR         Extrn Ukn       -   1388 1404
=MEMCKL         Extrn Ukn       -   0357
=MEMERR         Extrn Ukn       -   0048
=MGOSUB         Extrn Ukn       -   1507 2009 2085
=MODE71         Extrn Ukn       -   0919 1029
=MOVE•M         Extrn Ukn       -   1562 1595 2042
=MVFILE         005FB Rel  1308 -
=MVMEM+         Extrn Ukn       -   1508 2010 2086
 MVBUFS         0002D Abs  1293 -   1335 1353 1968 2113
 Mvf450         00855 Rel  1704 -   1682
=POSADR         Extrn Ukn       -   0836
```

```
=SRCADR       00425 Rel  0803 -
=SRCLIN       003EF Rel  0782 -
=SWPBYT       Extrn Ukn        -  1104
 SRC100       00436 Rel  0808 -  0801 0856
 SRC200       004A5 Rel  0866 -  0837
=TMP5         Extrn Ukn        -  0294 0298
 TURBULENCE   00032 Abs  0009 -  0307 0311 0345
=addlin       002A3 Rel  0450 -
 adln10       002BE Rel  0463 -  0469
 adln20       002D1 Rel  0474 -  0467
 adln30       002DF Rel  0480 -  0486
 curlin       Unkun Ukn  0779 -  0817 0841 0869
=dellin       0031F Rel  0583 -
 dl100        00353 Rel  0633 -  0640
 dl200        00368 Rel  0650 -  0637
 dl210        0037B Rel  0674 -  0684
 dl220        0038B Rel  0680 -  0660
 dl300        00398 Rel  0693 -
 dl310        003AF Rel  0705 -  0718
 dl350        003C4 Rel  0716 -  0698
 dstadr       00014 Abs  1290 -  1461 1496 2002
 dsthdr       0000F Abs  1289 -  1309 1997 2002
=eILPAR       Extrn Ukn        -  0905
=eL2LNG       Extrn Ukn        -  1108
=eMEM         Extrn Ukn        -  1430
=eNFND        Extrn Ukn        -  0860
=getchr       Extrn Ukn        -  0912 0929 0940
=id           Extrn Ukn        -  0860
=inibuf       00000 Rel  0031 -  0297
 inib10       00019 Rel  0044 -  0039
 inib15       00031 Rel  0050 -  0045
 inib20       0006E Rel  0078 -  0073
 inib50       000CB Rel  0146 -  0141 0206
 inib80       0014D Rel  0204 -  0171 0185
 inib89       000C7 Rel  0143 -  0147
 inib90       00156 Rel  0208 -  0143 0163
 insf50       009F2 Rel  2065 -  2053
 insf90       00A18 Rel  2099 -  2060
 insfil       00959 Rel  1953 -  1736 1778
 invprm       004BF Rel  0905 -  0913
 l2           00019 Abs  1291 -  1700 1722 1751
 ldeb         Unkun Ukn  0777 -  0786 0803 0851 0867
 lfin         Unkun Ukn  0778 -  0788 0805 0849
 mem10        00950 Rel  1896 -  1883
 mem20        00952 Rel  1898 -  1894
 memD         00933 Rel  1875 -  1405
 memerr       0002A Rel  0048 -  0067 0070
 mvf400       00741 Rel  1479 -  1469 1471
 mvf900       007CB Rel  1599 -  1541
 mvbadr       Unkun Ukn  1299 -  1347 1452 1491 1577 1614 1696 1716 1745 1783
```



```
                                +  1843 1965 1994 2046 2110
 mvbid        Unkun Ukn  1298 -  1331 1432 1816 1843
 mverr        005F9 Rel  1304 -  1342
 mvf100       005FB Rel  1313 -
 mvf110       00618 Rel  1331 -  1326
 mvf200       00698 Rel  1387 -
 mvf220       006DE Rel  1430 -  1444
 mvf230       006FE Rel  1441 -  1414 1420 1422
 mvf250       0070D Rel  1446 -  1428
 mvf300       0070D Rel  1452 -
 mvf310       00745 Rel  1481 -  1476
 mvf400       007CF Rel  1605 -  1479
 mvf405       007D3 Rel  1610 -  1792
 mvf410       00800 Rel  1641 -  1667
 mvf420       00805 Rel  1649 -  1633
 mvf430       00829 Rel  1677 -  1653
 mvf440       00859 Rel  1712 -  1758
 mvf442       00887 Rel  1732 -  1730
 mvf445       0088B Rel  1745 -  1702
 mvf450       00880 Rel  1775 -  1704
 mvf490       00888 Rel  1783 -  1605 1762
 mvf900       008DA Rel  1798 -  1477 1599 1791
 mvfer+       006E4 Rel  1431 -
 mvmem        Unkun Ukn  1301 -  1406 1610 1712
 orgdeb       00005 Abs  1287 -  1455 1458 1537 1585 1617 1621 1719 1721 1748
                                +  1750 1786 1788 2030 2049 2078 2093
 orgfin       0000A Abs  1288 -  1458 1461 1495 1496 1580 1585 1621 1699 1700
                                +  1721 1722 1750 1751 1788
 orghdr       00000 Abs  1286 -  1495 1537 1580 1617 1699 1786 1809 2078
=rpllin       002F2 Rel  0502 -
 rpln10       002FF Rel  0508 -  0514
 rpln20       00312 Rel  0519 -  0512
 rstk         0001E Abs  1292 -
=sMOVE        00000 Abs  1306 -  1413 1475 2052
=seek         0018E Rel  0279 -  0300 0795
=syncbf       003CE Rel  0738 -
 sauvR0       Unkun Ukn  0780 -  0790 0797
 sek005       001BB Rel  0294 -  0309 0313 0360
 sek010       001D9 Rel  0302 -  0325
 sek030       00202 Rel  0321 -  0292
 sek040       0020F Rel  0329 -  0323
 sek045       0023F Rel  0368 -  0347
 sek050       00248 Rel  0372 -  0396
 sek060       00275 Rel  0395 -  0370
 syncmv       00905 Rel  1843 -  1516 1803 2020 2092
 synerr       0092C Rel  1853 -  1846
 taille       Unkun Ukn  1300 -  1374 1441 1488 1571
 xxxx         Unkun Ukn  1302 -
```



```
Source  : xtext.as

Object  : obj/xtext.ao

Listing : list/xtext.al

Date    : Sat Aug 12 17:41:23 1989

Errors  : 000


Areuh Assembler/Linker V2.4. (c) P. David & J. Taillandier 1986 Paris. France
```