```
1    * PeerSoft v1.5.6 by Benoit Gilon - (c) 2006-2015 L.P.C.B.
2    * 30 Sep 2012: initial release
3    * 16 Oct 2012: 1.1, integ. divide support
4    * 30 Dec 2012: 1.2, integer arithmetic in FOR/NEXT loops
5    * & @ pseudo var)
6    * 3nd Jan 2013: 1.3 reorg subroutine #0
7    * 27 Jan 2013: 1.4 reorg subroutine #4 and MT kernel
8    * 6 Apr 2013: local error handling within MT kernel
9    * 1.5.5 addons:
10   * 31st July 2015: can concurrenlty define up to 11
11   * assembly language functions.. support for up to 2
12   * arguments instead of one originally.
13   * 3nd August 2015: support for Procedural functions
14   * 1.5.6 addons:
15   * 8th September 2015: byte new integer subtype added
16   * ToDo: Two new integer subtypes: 24 and
17   * 32 bits integer now understood (convenience for array
18   * variables of this integer subtypes).
19   * ToDo: Possibility to store indiv. array content
20   * within aux mem (auxiliary memory Apple and AE RAMWorks
21   * protocol)
22   * Merlin 8 assembler
26   * Constants
27   VERSION  =     $15
28   K6502    =     0
29   K65C02   =     1
30   K65816   =     1
31   * Generate either 65(816!C)02 compatible version
32   KOPT     =     K65C02
33   KNEW     =     1
34   KNEW2    =     1
35   KOPTLNG32 =    1
36   KOPTLNG33 =    0
37   * Cache size (# of entries) for simple variables
38   KSNCACH  =     4
39   * Cache size (# of entries) for array variables
40   KANCACH  =     4
41
43            XC
44   KOPT16   =     0
52
53   * Token equates
54   TOKEQUAL =     $D0
55   TOKADD   =     $C8
56   TOKMUL   =     $CA
57   TOKDIV   =     $CB
58   TOKDEF   =     $B8        Prefix for DEF(INT!STR!SNG)
59   TOKINT   =     $D3        DEFINT instr st. as 2 tokens
60   TOKUSR   =     $D5        DEFUSR...
61   TOKMINUS =     $C9
62   TOKREM   =     $B2
63   TOKDATA  =     $83
64   TOKIF    =     $AD
65   TOKFN    =     $C2
66   TOKTO    =     $C1
67   TOKSTRD  =     $E4
68   TOKCHRD  =     $E7
69   TOKSGN   =     $D2
70   TOKSCRN  =     $D7
```

```
71   TOKNOT   =     $C6
72   TOKSTEP  =     $C7
73   TOKGOSUB =     $B0
74   TOKGOTO  =     $AB
75   TOKFOR   =     $81
76
77   * Page zero and monitor equates
78   PCL      EQU   $3A
79   LENGTH   EQU   $2F
80   INSDS2   EQU   $F88C
81   PCADJ    EQU   $F953
82   A1L      EQU   $3C
83   A2L      EQU   $3E
84   A4L      EQU   $42
85   MOVE     EQU   $FE2C
86   CH       EQU   $24
87   XFER     EQU   $C314
88   VECZAUX  EQU   $03ED
89
90   * Applesoft equates
91   DIMFLG   EQU   $10        Input to PTRGET
92   * Output from PTRGET
93   VALTYP   EQU   $11        $FF if string, 0 if num.
94   INTTYP   EQU   $12        $80 if integer, 0 otherwise
95   VARNAM   EQU   $81        Encoded varname 1st char.
96   VARPNT   EQU   $83        Variable value pointer
97   SUBFLG   EQU   $14        Parameter for PTRGET routine
98   LINNUM   EQU   $50        Line # (output from LINGET)
99   CURLIN   EQU   $75        Current line # (being run)
100  INDEX    EQU   $5E        General ptr for ROM str. routines
101  LOWTR    EQU   $9B        Address of BASIC line (output fro
m FNDLIN)
102  FAC      EQU   $9D        Main floating point accumulator
103  DEST     EQU   $60        Used by NEXT
104  STREND   EQU   $6D        End of array memory
105  FACSIGN  EQU   $A2
106  FACLO    EQU   $A1
107  FACMO    EQU   $A0
108  TXTPTR   EQU   $B8        Pointer to BASIC program memory
109  OLDTPTR  EQU   $79
110  REMSTK   EQU   $F8
111  OLDTEXT  EQU   $79
112  ARYPNT   EQU   $94        Pointer to array structure
113  ERRFLG   EQU   $D8        ONERR activivty flag
114  ERRLIN   EQU   $DA        Offending line #
115  ERRPOS   EQU   $DC        Where in the offending line #..
116  ERRNUM   EQU   $DE        Error #
117  ERRSTK   EQU   $DF        Stack pntr of offending instr.
118  TXTPSV   EQU   $F4
119  CURLSV   EQU   $F6
120
121  TOKTABL  EQU   $D0D0      Address of internal Applesoft tok
en table
122  ISLETC   EQU   $E07D      Check whether current char alpha
123  SYNERR   EQU   $DEC9      Report a SYNTAX ERROR
124  VLET     EQU   $DA46
125  VPTRGET  EQU   $DFEF      PTRGET return adress (from stack)
```

```
                126  ISCNTC   EQU   $D858      Check for Ctrl-C keystroke
                127  ADDON    EQU   $D998      Add Y to TXTPTR
                128  LINGET   EQU   $DA0C      Get line number from TXTPTR
                129  CHKMEM   EQU   $D3D6      Check for A 16bit words on stack
                130  COMBYTE  EQU   $E74C      Check for comma and compute
                131
                132  * Applesoft output routines
                133  OUTDO    EQU   $DB5C      Generic
                134  CRDO     EQU   $DAFB      Carriage return
                135  OUTSPC   EQU   $DB57      Space
                136  FNDLIN   EQU   $D61A      From line number (LINNUM) to addr
ess
                137  NEWSTT   EQU   $D7D2      Applesoft main exec loop
                138  FORPNT   EQU   $85
                139  FRMEVL   EQU   $DD7B      Eval. expr pointed to by TXTPTR
                140  FRMNUM   EQU   $DD67      Eval. expr & ensure numeric resul
t
                141  GETADR   EQU   $E752      Expression to 16bits integer
                142  GETBYT   EQU   $E6F8      Eval. expr into single byte value
                143  * Some checking about FAC:must contain..
                144  CHKNUM   EQU   $DD6A      a scalar factor
                145  CHKSTR   EQU   $DD6C      a string factor
                146  AYINT    EQU   $E10C      Integer conversion from FP
                147  * Some floating point computing dst is FAC1
                148  FSUB     EQU   $E7A7      (Y,A) - FAC1
                149  FADD     EQU   $E7BE      (Y,A) + FAC1
                150  FMULT    EQU   $E97F      (Y,A) * FAC1
                151  FDIV     EQU   $EA66      (Y,A) / FAC1
                152  NEGOP    EQU   $EED0      -FAC1
                153  * Raise some Applesoft errors
                154  GOSTLERR EQU   $E5B2      STRING TOO LONG
                155  GOOVFERR EQU   $E8D5      OVERFLOW
                156  GOTMIERR EQU   $DD76      TYPE MISMATCH
                157  GODVZERR EQU   $EAE1      DIVIDE BY ZERO
                158  GOIQERR  EQU   $E199      ILLEGAL QUANTITY
                159  FREESPC  EQU   $71
                160  STRSPA   EQU   $E3DD      Get space from string pool for a
string of len A
                161  DSCTMP   EQU   $9D        Temporary string pointer
                162  STRING1  EQU   $AB        String pointer used by copy
                163  MOVINS   EQU   $E5D4      Move string(STRING1) into memory(
FRESPC)
                164  ERRDIR   EQU   $E306      Raises a illegal direct mode iif
required
                165  DATAN    EQU   $D9A3      Scan ahead to next EOI
                166  DATA     EQU   $D995      TXTPTR points to next separator
                167  VARTAB   EQU   $69        Begin of simple var. mem. area
                168  ARYTAB   EQU   $6B        Begin of array var. mem. area
                169
                170  FRMSTCK3 EQU   $DE20
                171
                172  * ZP slots used by integer signed 16bits mult/div subroutin
es
                173  MCAND    EQU   $C0
                174  MPLIER   EQU   $C2
                175  DIVEND   EQU   MPLIER
                176  DIVSOR   EQU   $C0
```

```
177  PARTIAL  EQU   $BE
178  AUXBANK  EQU   $BF
179  LETINF   EQU   $C0
180  TYPMOD   EQU   $C1
181  INTTYPSV EQU   $C7
182  VALTYPSV EQU   $C8
183
184  * DOS 3.3 equates
185  OPRND    EQU   $44
186  DBUFP    EQU   $9D00
187
188           ORG   $4000
189
190  AUXPTR   EQU   $06
191  IDMOCL   EQU   $BD
192  OFFSET   EQU   $C2
193  XSAV     EQU   $B4
194  YSAV     EQU   $B5
195  MODREM   EQU   $BE
196  MODDAT   EQU   $BF
197  GFLAG    EQU   $C0
198  IDX0     EQU   $C0
199  DEFFLG   EQU   $C1
200  NOPER    =     4
201
204  EMOV     MAC
205           LDA   ]1
206           STA   ]2
207           <<<
208
209  STD      MAC
210           EMOV  ]1;]2
211           EMOV  ]1+1;]2+1
212           <<<
213
214  * 16bits immediate store
215  STID     MAC
216           EMOV  #]1;]2
217           EMOV  #>]1;]2+1
218           <<<
219
220  * Copy a large memory area within
221  * adressable memory
222  MOVM     MAC
223           STID  ]1;A1L
224           STID  ]2;A2L
225           STID  ]3;A4L
226           JSR   MOVE
227           <<<
228
229  * Copy a small memory area within
230  * adressable memory
231  SMOVE    MAC
232           LDX   #]3
233  LOOP     LDA   ]1-1,X
234           STA   ]2-1,X
235           DEX
```

```
236           BNE   LOOP
237           <<<
238
239   * Macros for simulating 65C02 instructions
240   * on a 6502
241   MPHX    MAC
242           DO    KOPT-K65C02
243           TXA
244           PHA
245           ELSE
246           PHX
247           FIN
248           <<<
249
250   MPHY    MAC
251           DO    KOPT-K65C02
252           TYA
253           PHA
254           ELSE
255           PHY
256           FIN
257           <<<
258
259   MPLX    MAC
260           DO    KOPT-K65C02
261           PLA
262           TAX
263           ELSE
264           PLX
265           FIN
266           <<<
267
268   MPLY    MAC
269           DO    KOPT-K65C02
270           PLA
271           TAY
272           ELSE
273           PLY
274           FIN
275           <<<
276
277   MTSB    MAC
278           DO    KOPT-K65C02
279           ORA   ]1
280           STA   ]1
281           ELSE
282           TSB   ]1
283           FIN
284           <<<
285
286   GOTO    MAC
287           DO    KOPT-K6502
288           BRA   ]1
289           ELSE
290           JMP   ]1
291           FIN
292           <<<
```

```
                294
                295  * Do all the stuff for installing Peersoft
                296  * between DOS and its buffers
                297           PUT   PEERINSTALL
               >1    NEWY     EQU   $47
               >15
               >16   * This module deals with all installation stuff for the
               >17   * Peersoft suite
4000: A9 D3    >18   SUITE    LDA   #$9CD3     Compute the offset
4002: 38       >19           SEC               ;Put it in :0+1 (lobyte)
4003: ED 00 9D >20           SBC   DBUFP        and :1+1 (hibyte)
4006: 8D 47 40 >21           STA   :0+1
4009: A9 9C    >22           LDA   #>$9CD3
400B: ED 01 9D >23           SBC   DBUFP+1
400E: AA       >24           TAX
400F: 0D 47 40 >25           ORA   :0+1
               >26   * If first utility to ask for memory this way, then ask for
               >27   * one additional page for our own purpose (i.e. Bananasoft
               >28   * or Peersoft)
4012: F0 01    >29           BEQ   :6
4014: CA       >30           DEX
4015: 8E 4F 40 >31   :6       STX   :1+1
               >32
               >33   * Relocate code (don´t move it yet)
4018: A9 9F    >40           LDA   #AROMBA
401A: A0 47    >41           LDY   #>AROMBA
401C: 85 3A    >42   ]LOOP    STA   PCL
401E: C9 64    >43           CMP   #FCODE-FNDVAR2+AROMBA
4020: 98       >44           TYA
4021: E9 67    >45           SBC   #>FCODE-FNDVAR2+AROMBA
4023: B0 33    >46           BCS   :4
4025: 84 3B    >47           STY   PCL+1
4027: 20 01 42 >51           JSR   MINSDS2
402A: A4 2F    >52           LDY   LENGTH
402C: C0 02    >53           CPY   #2          Only relocates 3 bytes instr.
402E: D0 22    >54           BNE   :3
4030: B1 3A    >55           LDA   (PCL),Y
4032: AA       >56           TAX
4033: 88       >57           DEY
4034: B1 3A    >58           LDA   (PCL),Y
4036: A8       >59           TAY
4037: C9 00    >60           CMP   #FIN        Only if adress within range
4039: 8A       >61           TXA
403A: E9 9C    >62           SBC   #>FIN
403C: B0 14    >63           BCS   :3          Must be < FIN to be relocated
403E: C0 09    >64           CPY   #FNDVAR2
4040: 8A       >65           TXA
4041: E9 76    >66           SBC   #>FNDVAR2
4043: 90 0D    >67           BCC   :3          Must be >= FNDVAR2
4045: 98       >68           TYA               ;Relocates address
4046: E9 00    >69   :0       SBC   #0
4048: A0 01    >70           LDY   #1
404A: 91 3A    >71           STA   (PCL),Y     Low byte
404C: C8       >72           INY
404D: 8A       >73           TXA
404E: E9 00    >74   :1       SBC   #0
4050: 91 3A    >75           STA   (PCL),Y     High byte
```

```
4052: 20 53 F9 >76   :3       JSR   PCADJ      Adjust PCL to length byte
4055: 4C 1C 40 >77            JMP   ]LOOP      Loop
               >78
               >80
               >81   * Relocate some non trivial references (i.e. instructions
               >82   * with immediate adressing mode).
4058: A2 13    >83   :4       LDX   #ADPFT-ADPFB-1
405A: BD 6E 6D >84   ]LOOP    LDA   ADPFB+AROMBA-FNDVAR2,X
405D: 38       >85            SEC
405E: ED 47 40 >86            SBC   :0+1
4061: 9D 6E 6D >87            STA   ADPFB+AROMBA-FNDVAR2,X
4064: BD 82 6D >88            LDA   ADPFT+AROMBA-FNDVAR2,X
4067: ED 4F 40 >89            SBC   :1+1
406A: 9D 82 6D >90            STA   ADPFT+AROMBA-FNDVAR2,X
406D: CA       >91            DEX
406E: 10 EA    >92            BPL   ]LOOP
               >93
4070: A2 0C    >94            LDX   #ADT1-ADB1-1
4072: A9 00    >95            LDA   #0
4074: 85 3A    >96            STA   PCL
4076: BD 24 42 >97   ]LOOP    LDA   ADT1,X
4079: 85 3B    >98            STA   PCL+1
407B: BC 17 42 >99            LDY   ADB1,X
407E: B1 3A    >100           LDA   (PCL),Y
4080: 38       >101           SEC
4081: ED 47 40 >102           SBC   :0+1
4084: 91 3A    >103           STA   (PCL),Y
4086: BD 3E 42 >104           LDA   ADT2,X
4089: 85 3B    >105           STA   PCL+1
408B: BC 31 42 >106           LDY   ADB2,X
408E: B1 3A    >107           LDA   (PCL),Y
4090: ED 4F 40 >108           SBC   :1+1
4093: 91 3A    >109           STA   (PCL),Y
4095: CA       >110           DEX
4096: 10 DE    >111           BPL   ]LOOP
               >112
4098: A2 0F    >113           LDX   #OFFSTT-OFFSTB-1
409A: BD BB 67 >114  ]LOOP    LDA   OFFSTB+AROMBA-FNDVAR2,X
409D: 38       >115           SEC
409E: ED 47 40 >116           SBC   :0+1
40A1: 9D BB 67 >117           STA   OFFSTB+AROMBA-FNDVAR2,X
40A4: BD CB 67 >118           LDA   OFFSTT+AROMBA-FNDVAR2,X
40A7: ED 4F 40 >119           SBC   :1+1
40AA: 9D CB 67 >120           STA   OFFSTT+AROMBA-FNDVAR2,X
40AD: CA       >121           DEX
40AE: 10 EA    >122           BPL   ]LOOP
               >123  * Move the code
40B0: A9 09    >124           LDA   #CGARBAG
40B2: A2 76    >125           LDX   #>CGARBAG
40B4: 38       >126           SEC
40B5: ED 47 40 >127           SBC   :0+1
40B8: 85 42    >128           STA   A4L
40BA: 8A       >129           TXA
40BB: ED 4F 40 >130           SBC   :1+1
40BE: 85 43    >131           STA   A4L+1
               >132
40C0: A9 9F    >133           LDA   #CGARBAG+AROMBA-FNDVAR2
```

```
40C2: A2 47    >134           LDX   #>CGARBAG+AROMBA-FNDVAR2
40C4: 85 3C    >135           STA   A1L
40C6: 86 3D    >136           STX   A1L+1
               >137
40C8: A9 95    >138           LDA   #FIN-1+AROMBA-FNDVAR2
40CA: 85 3E    >138           STA   A2L
40CC: A9 6D    >138           LDA   #>FIN-1+AROMBA-FNDVAR2
40CE: 85 3F    >138           STA   A2L+1
               >139
40D0: A0 00    >140           LDY   #0
40D2: 2C 81 C0 >141           BIT   $C081
40D5: 2C 81 C0 >142           BIT   $C081
40D8: 20 2C FE >143           JSR   MOVE
               >144  * Reconstruct DOS buffers below PeerSoft
40DB: AD 00 9D >145           LDA   DBUFP
40DE: AE 01 9D >146           LDX   DBUFP+1
40E1: C9 D3    >147           CMP   #$9CD3
40E3: D0 05    >148           BNE   :7
40E5: E0 9C    >149           CPX   #>$9CD3
40E7: D0 01    >150           BNE   :7         One more page if first utility
40E9: CA       >151           DEX              ; to install this way
40EA: 38       >152  :7       SEC
40EB: E9 F7    >153           SBC   #LONGLANG
40ED: A8       >154           TAY
40EE: 8A       >155           TXA
40EF: E9 25    >156           SBC   #>LONGLANG
40F1: 8C 00 9D >157           STY   DBUFP      New DOS base buffer address
40F4: 8D 01 9D >158           STA   DBUFP+1
40F7: 20 D4 A7 >159           JSR   $A7D4
               >160
40FA: A9 15    >161           LDA   #VERSION
40FC: 8D DE 9C >162           STA   PVERSION
40FF: A9 80    >163           LDA   #$80
4101: 8D D0 9C >164           STA   OPTCGOTO
4104: 9C CF 9C >166           STZ   NEEDDEC
               >171
               >172  * Number of Applesoft instruction runs
               >173  * between two consecutives context switches
4107: A9 0A    >174           LDA   #10
4109: 8D DD 9C >175           STA   ICTRACTV
410C: 9C DC 9C >180           STZ   MTACTV
410F: A9 4C    >182           LDA   #$4C
4111: 8D DF 9C >183           STA   REVECTOR
4114: 8D D5 9C >184           STA   VGARBAG
4117: 38       >185           SEC
4118: A9 00    >186           LDA   #ROUTGEN
411A: ED 47 40 >187           SBC   :0+1
411D: 8D E0 9C >188           STA   REVECTOR+1
4120: A9 87    >189           LDA   #>ROUTGEN
4122: ED 4F 40 >190           SBC   :1+1
4125: 8D E1 9C >191           STA   REVECTOR+2
4128: A9 72    >192           LDA   #NPTRGL90
412A: ED 47 40 >193           SBC   :0+1
412D: 8D D3 9C >194           STA   VNPTRG90
4130: A9 7A    >195           LDA   #>NPTRGL90
4132: ED 4F 40 >196           SBC   :1+1
4135: 8D D4 9C >197           STA   VNPTRG90+1
```

```
4138: A9 7E    >198           LDA   #NARRGL91
413A: ED 47 40 >199           SBC   :0+1
413D: 8D D1 9C >200           STA   VNARRG91
4140: A9 7B    >201           LDA   #>NARRGL91
4142: ED 4F 40 >202           SBC   :1+1
4145: 8D D2 9C >203           STA   VNARRG91+1
4148: A9 E1    >204           LDA   #TABOFB
414A: ED 47 40 >205           SBC   :0+1
414D: 8D D8 9C >206           STA   ADADR
4150: A9 95    >207           LDA   #>TABOFB
4152: ED 4F 40 >208           SBC   :1+1
4155: 8D D9 9C >209           STA   ADADR+1
4158: A2 84    >210           LDX   #GARBAG
415A: A9 E4    >211           LDA   #>GARBAG
415C: 2C EF 9C >212           BIT   MEMORY
415F: 10 0B    >213           BPL   *+13
4161: A9 D8    >214           LDA   #NGARBAG
4163: ED 47 40 >215           SBC   :0+1
4166: AA       >216           TAX
4167: A9 7D    >217           LDA   #>NGARBAG
4169: ED 4F 40 >218           SBC   :1+1
416C: 8E D6 9C >219           STX   VGARBAG+1
416F: 8D D7 9C >220           STA   VGARBAG+2
4172: A9 04    >221           LDA   #NDSVCMD   New DOS Save for applesoft
4174: ED 47 40 >222           SBC   :0+1
4177: 8D A6 A3 >223           STA   $A3A6
417A: A9 92    >224           LDA   #>NDSVCMD
417C: ED 4F 40 >225           SBC   :1+1
417F: 8D A7 A3 >226           STA   $A3A7
4182: A9 0C    >227           LDA   #NDLVCMD   Part of routine for loading
4184: ED 47 40 >228           SBC   :0+1
4187: 8D 2E A4 >229           STA   $A42E
418A: A9 92    >230           LDA   #>NDLVCMD
418C: ED 4F 40 >231           SBC   :1+1
418F: 8D 2F A4 >232           STA   $A42F
4192: A9 20    >233           LDA   #$20
4194: 8D 9E 9E >234           STA   $9E9E
4197: A9 EB    >235           LDA   #NKBDINT
4199: ED 47 40 >236           SBC   :0+1
419C: 8D 9F 9E >237           STA   $9E9F
419F: A9 91    >238           LDA   #>NKBDINT
41A1: ED 4F 40 >239           SBC   :1+1
41A4: 8D A0 9E >240           STA   $9EA0
41A7: 20 4B 42 >241           JSR   BIGRECON
41AA: 20 F5 42 >242           JSR   MOUSEDET
41AD: 2C EF 9C >243           BIT   MEMORY
41B0: 50 09    >244           BVC   :44
               >245  * Copy $F8-$FF pages within ROM to main and aux
               >246  * memory banks
41B2: 20 59 43 >247           JSR   COPYROM
               >248  * Initialize BF page
41B5: 20 18 44 >249           JSR   INITBF
41B8: 20 D2 41 >250           JSR   MZRTAUX
41BB: 2C 80 C0 >251  :44      BIT   $C080
41BE: 2C 80 C0 >252           BIT   $C080
               >253  * If Applesoft is the active language, so
               >254  * install Peersoft CHRGET/CHRGOT patch
```

```
41C1: AD B6 AA >255  EK       LDA   $AAB6
41C4: F0 09    >256           BEQ   :11
41C6: 2C 81 C0 >257           BIT   $C081
41C9: 2C 81 C0 >258           BIT   $C081
41CC: 20 12 82 >259           JSR   SETUPB
41CF: 4C 29 82 >260  :11      JMP   SETUPD
               >261
41D2: A9 BF    >262  MZRTAUX  LDA   #$BF
41D4: A2 00    >263           LDX   #0
41D6: 8D EE 03 >264           STA   $03EE
41D9: 8E ED 03 >265           STX   $03ED
41DC: B8       >266           CLV
41DD: 38       >267           SEC
41DE: 4C 14 C3 >268           JMP   XFER
               >269
               >271  MC       DO    KOPT16
41E1: DA FA 04 >275           HEX   DAFA041A3A PHX/PLX/TSB d/INC/DEC
41E6: 7C 80 7A >276           HEX   7C807A5A   JMP (abs, X)/BRA d/PLY/PHY
41EA: 64 9E    >277           HEX   649E       STZ d/STZ a, X
41EC: 0C 9C    >278           HEX   0C9C       TSB a/STZ a
41EE: 1C 14    >279           HEX   1C14       TRB a/TRB d
41F0: B2       >280           HEX   B2         LDA (d)
               >287  LN       DO    KOPT16
41F1: 00 00 01 >291           HEX   0000010000 PHX/PLX/TSB d/INC/DEC
41F6: 02 01 00 >292           HEX   02010000   JMP (abs, X)/BRA d/PLY/PHY
41FA: 01 02    >293           HEX   0102       STZ d/STZ a, X
41FC: 02 02    >294           HEX   0202       TSB a/STZ a
41FE: 02 01    >295           HEX   0201       TRB a/TRB d
4200: 01       >296           HEX   01         LDA (d)
               >303  * Check 65C02/65802 used and new machine codes
4201: B2 3A    >304  MINSDS2  LDA   (PCL)
4203: A2 0F    >305           LDX   #LN-MC-1
4205: DD E1 41 >306  ]LOOP    CMP   MC,X
4208: F0 07    >307           BEQ   :0
420A: CA       >308           DEX
420B: 10 F8    >309           BPL   ]LOOP
420D: E8       >310           INX             ;X = 0
420E: 4C 8C F8 >326           JMP   INSDS2
4211: BD F1 41 >327  :0       LDA   LN,X
4214: 85 2F    >328           STA   LENGTH
4216: 60       >393           RTS
               >398
4217: CA       >405  ADB1     DFB   EK+9
4218: CD       >406           DFB   EK+12
4219: AF       >407           DFB   SETUPB+7+AROMBA-FNDVAR2
421A: B7       >408           DFB   SETUPB+15+AROMBA-FNDVAR2
421B: C0       >409           DFB   SETUPD+1+AROMBA-FNDVAR2
421C: F2       >410           DFB   STP1+1+AROMBA-FNDVAR2
421D: 90       >411           DFB   SFE1+1+AROMBA-FNDVAR2
421E: 71       >412           DFB   SETLTR+1
421F: 27       >416           DFB   GN65536+1+AROMBA-FNDVAR2
4220: 1D       >417           DFB   GN32768+1+AROMBA-FNDVAR2
4221: 2C       >418           DFB   GP65536+1+AROMBA-FNDVAR2
4222: B5       >422           DFB   NAMNTFND+5
4223: AE       >426           DFB   V3B+1+AROMBA-FNDVAR2
4224: 41       >428  ADT1     DFB   >EK+9
4225: 41       >429           DFB   >EK+12
```

```
4226: 53       >430           DFB   >SETUPB+7+AROMBA-FNDVAR2
4227: 53       >431           DFB   >SETUPB+15+AROMBA-FNDVAR2
4228: 53       >432           DFB   >SETUPD+1+AROMBA-FNDVAR2
4229: 60       >433           DFB   >STP1+1+AROMBA-FNDVAR2
422A: 60       >434           DFB   >SFE1+1+AROMBA-FNDVAR2
422B: 89       >435           DFB   >SETLTR+1
422C: 60       >439           DFB   >GN65536+1+AROMBA-FNDVAR2
422D: 60       >440           DFB   >GN32768+1+AROMBA-FNDVAR2
422E: 60       >441           DFB   >GP65536+1+AROMBA-FNDVAR2
422F: 7A       >445           DFB   >NAMNTFND+5
4230: 51       >449           DFB   >V3B+1+AROMBA-FNDVAR2
4231: CB       >451  ADB2     DFB   EK+10
4232: CE       >452           DFB   EK+13
4233: B3       >453           DFB   SETUPB+11+AROMBA-FNDVAR2
4234: BB       >454           DFB   SETUPB+19+AROMBA-FNDVAR2
4235: C5       >455           DFB   SETUPD+6+AROMBA-FNDVAR2
4236: F4       >456           DFB   STP1+3+AROMBA-FNDVAR2
4237: 92       >457           DFB   SFE1+3+AROMBA-FNDVAR2
4238: 75       >458           DFB   SETLTR+5
4239: 29       >462           DFB   GN65536+3+AROMBA-FNDVAR2
423A: 1F       >463           DFB   GN32768+3+AROMBA-FNDVAR2
423B: 2E       >464           DFB   GP65536+3+AROMBA-FNDVAR2
423C: BC       >468           DFB   NAMNTFND+12
423D: AB       >472           DFB   V3T+1+AROMBA-FNDVAR2
423E: 41       >474  ADT2     DFB   >EK+10
423F: 41       >475           DFB   >EK+13
4240: 53       >476           DFB   >SETUPB+11+AROMBA-FNDVAR2
4241: 53       >477           DFB   >SETUPB+19+AROMBA-FNDVAR2
4242: 53       >478           DFB   >SETUPD+6+AROMBA-FNDVAR2
4243: 60       >479           DFB   >STP1+3+AROMBA-FNDVAR2
4244: 60       >480           DFB   >SFE1+3+AROMBA-FNDVAR2
4245: 89       >481           DFB   >SETLTR+5
4246: 60       >485           DFB   >GN65536+3+AROMBA-FNDVAR2
4247: 60       >486           DFB   >GN32768+3+AROMBA-FNDVAR2
4248: 60       >487           DFB   >GP65536+3+AROMBA-FNDVAR2
4249: 7A       >491           DFB   >NAMNTFND+12
424A: 51       >495           DFB   >V3T+1+AROMBA-FNDVAR2
               >497
424B: 2C 81 C0 >498  BIGRECON BIT   $C081
424E: 2C 81 C0 >499           BIT   $C081
               >500  * What is the model/ROM version of the Apple
4251: A0 07    >501           LDY   #8-1
4253: AD B3 FB >502           LDA   $FBB3
4256: 4D C0 FB >503           EOR   $FBC0
4259: 4D BF FB >504           EOR   $FBBF
425C: D9 CB 42 >505  ]LOOP    CMP   MACMAT,Y
425F: F0 04    >506           BEQ   :1
4261: 88       >507           DEY
4262: 10 F8    >508           BPL   ]LOOP
4264: C8       >509           INY                ;Assuming default 2+
               >510  * Apple //e enhanced ROM and //gs have same signature,
               >511  * so we ll make the difference on $FC5C
               >512  * value ($EB in a //gs ROM)
4265: C0 02    >513  :1       CPY   #2
4267: D0 20    >514           BNE   :2
4269: AD 5C FC >515           LDA   $FC5C
426C: C9 EB    >516           CMP   #$EB
```

```
426E: D0 19    >517           BNE   :2
4270: A0 08    >518           LDY   #8          //gs!
4272: 18       >519           CLC
4273: FB       >520           HEX   FB          ;XCE: Enter native mode
4274: 08       >521           PHP               ;Push carry status (old emu bit)
4275: C2 30    >522           HEX   C230        Set 16bits mode
4277: 20 1F FE >523           JSR   $FE1F       Call ID firmware routine
427A: 84 47    >524           STY   NEWY
427C: 28       >525           PLP               ;Restore original emulation bit
427D: FB       >526           HEX   FB          ;XCE: Exit native mode
427E: A0 0C    >527           LDY   #12
4280: A5 48    >528           LDA   NEWY+1
4282: D0 05    >529           BNE   :2
4284: A5 47    >530           LDA   NEWY
4286: 09 08    >531           ORA   #8
4288: A8       >532           TAY
               >533
4289: B9 D3 42 >534  :2       LDA   MCODE,Y
428C: 8D ED 9C >535           STA   MACHINE
428F: 98       >536           TYA
4290: AA       >537           TAX
4291: D0 26    >538           BNE   :3          00 if Apple 2+
               >539  * Test for Apple2+, X=0 upon entry
               >540  * Possible language card being there..
4293: 2C 83 C0 >541           BIT   $C083
4296: 2C 83 C0 >542           BIT   $C083
4299: AD 00 D0 >543           LDA   $D000
429C: C8       >544           INY
429D: 8C 00 D0 >545           STY   $D000
42A0: CC 00 D0 >546           CPY   $D000       Read after write (1st)
42A3: D0 0A    >547           BNE   :5
42A5: EE 00 D0 >548           INC   $D000
42A8: C8       >549           INY
42A9: CC 00 D0 >550           CPY   $D000       Read after increment (2nd)
42AC: D0 01    >551           BNE   :5
42AE: E8       >552           INX
42AF: 8D 00 D0 >553  :5       STA   $D000
42B2: BD E5 42 >554           LDA   CFA,X
42B5: A2 00    >555           LDX   #0
42B7: F0 0B    >556           BEQ   :4
42B9: C9 04    >557  :3       CMP   #4          Apple //c or //gs?
42BB: A9 C0    >558           LDA   #$C0
42BD: A2 80    >559           LDX   #$80
42BF: B0 03    >560           BCS   :4          Yes
42C1: 20 95 43 >561           JSR   TEST2E
42C4: 8D EF 9C >562  :4       STA   MEMORY
42C7: 8E F0 9C >563           STX   VID80C
42CA: 60       >564           RTS
               >565
42CB: EA 2D E6 >566  MACMAT   HEX   EA2DE6E7F9060502
42D3: 00       >567  MCODE    HEX   00          Apple 2+
42D4: 40 41 42 >568           HEX   404142      Apple //e
42D7: 80 81 82 >569           HEX   80818283    Apple //c
42DB: C0 C1 C2 >570           HEX   C0C1C2C3C4C5 Apple //gs
42E1: 80 80 C0 >571  CFM      HEX   8080C0C0
42E5: 00 80 80 >572  CFA      HEX   008080C0
               >573
```

```
42E9: 05 07 0B >574  DATA1IDX DFB   5,7,11,12,17,251
42EF: 38 18 01 >575  DATA1VAL HEX   3818012000D6
               >576  * Routine to detect a mouse card
42F5: A2 C7    >577  MOUSEDET LDX   #$C7
42F7: 86 07    >578           STX   AUXPTR+1
42F9: 8E CE 9C >579           STX   MOSL       ;b7 of MOSL set to 1
42FC: 64 06    >581           STZ   AUXPTR
42FE: 9C B7 99 >582           STZ   MOCN
4301: 9C B5 99 >583           STZ   MON0
4304: A2 05    >590  ]LOOP    LDX   #DATA1VAL-DATA1IDX-1
4306: BC E9 42 >591  ]LOOP1   LDY   DATA1IDX,X
4309: BD EF 42 >592           LDA   DATA1VAL,X
430C: 51 06    >593           EOR   (AUXPTR),Y
430E: D0 3D    >594           BNE   :1
4310: CA       >595           DEX
4311: 10 F3    >596           BPL   ]LOOP1
4313: A5 07    >597           LDA   AUXPTR+1
4315: 8D B7 99 >598           STA   MOCN
4318: 29 0F    >599           AND   #$F
431A: 8D CE 9C >600           STA   MOSL
431D: 0A       >602           ASL
431E: 0A       >602           ASL
431F: 0A       >602           ASL
4320: 0A       >602           ASL
4321: 8D B5 99 >604           STA   MON0
4324: E8       >605           INX              ;X = 0
4325: EC ED 9C >606           CPX   MACHINE    Is host an Apple2 or 2+?
4328: D0 13    >607           BNE   :2
               >608  * Time to INITMOUSE..
432A: A0 19    >609           LDY   #$19       Offset to INIT mouse offset
432C: B1 06    >610           LDA   (AUXPTR),Y
432E: 85 06    >611           STA   AUXPTR
4330: A6 07    >612           LDX   AUXPTR+1
4332: AC B5 99 >613           LDY   MON0
4335: 20 56 43 >614           JSR   :0
4338: 90 03    >615           BCC   :2
433A: 6E CE 9C >616           ROR   MOSL       Let set b7 of mouse slot
433D: A2 07    >617  :2        LDX   #OM_INI-OM_DEB
433F: 64 06    >619           STZ   AUXPTR
4341: BC AD 99 >624  ]JLOOP   LDY   OM_DEB,X
4344: B1 06    >625           LDA   (AUXPTR),Y
4346: 9D AD 99 >626           STA   OM_DEB,X
4349: CA       >627           DEX
434A: 10 F5    >628           BPL   ]JLOOP
434C: 60       >629           RTS
434D: A6 07    >630  :1        LDX   AUXPTR+1
434F: E0 C1    >631           CPX   #$C1
4351: C6 07    >632           DEC   AUXPTR+1
4353: B0 AF    >633           BCS   ]LOOP
4355: 60       >634  :FIN      RTS
4356: 6C 06 00 >635  :0        JMP   (AUXPTR)
               >636
               >637  * Routine to copy ROM to bank switched RAM
4359: A0 00    >638  COPYROM  LDY   #0
435B: A9 F8    >639           LDA   #$F8
435D: 84 3C    >640           STY   A1L
435F: 85 3D    >641           STA   A1L+1
```

```
4361: 8D 09 C0 >642           STA   $C009      Write into aux ZP
4364: 84 3C    >643           STY   A1L
4366: 85 3D    >644           STA   A1L+1
4368: 8D 08 C0 >645           STA   $C008      Write back into main ZP
436B: 2C 89 C0 >646           BIT   $C089      Write into LC ram
436E: 2C 89 C0 >647           BIT   $C089
4371: B1 3C    >648  ]LOOP    LDA   (A1L),Y
4373: 91 3C    >649           STA   (A1L),Y    within main memory
4375: 8D 09 C0 >650           STA   $C009      Write into aux memory LC bank
4378: 91 3C    >651           STA   (A1L),Y
437A: 8D 08 C0 >652           STA   $C008      Back to writing to main memory
437D: C8       >653           INY
437E: D0 F1    >654           BNE   ]LOOP
4380: E6 3D    >655           INC   A1L+1
4382: A5 3D    >656           LDA   A1L+1
4384: 8D 09 C0 >657           STA   $C009
4387: 85 3D    >658           STA   A1L+1
4389: 8D 08 C0 >659           STA   $C008
438C: D0 E3    >660           BNE   ]LOOP
438E: 2C 81 C0 >661           BIT   $C081
4391: 2C 81 C0 >662           BIT   $C081
4394: 60       >663           RTS
               >664
               >665  * Routine to test //e configuration: 80 col. card?
               >666  * memory expansion?
4395: 08       >667  TEST2E   PHP
4396: 78       >668           SEI
4397: A2 00    >669           LDX   #0
4399: AD 17 C0 >670           LDA   $C017
439C: 30 6F    >671           BMI   :6
439E: E8       >672           INX
439F: AD 1D C0 >673           LDA   $C01D
43A2: 48       >674           PHA
43A3: AD 18 C0 >675           LDA   $C018
43A6: 48       >676           PHA
43A7: AD 1C C0 >677           LDA   $C01C
43AA: 48       >678           PHA
43AB: AD 19 C0 >679  ]LOOP    LDA   $C019
43AE: 30 FB    >680           BMI   ]LOOP
43B0: 8D 57 C0 >681           STA   $C057
43B3: 8D 01 C0 >682           STA   $C001
43B6: 8D 55 C0 >683           STA   $C055
43B9: AD 00 04 >684           LDA   $400
43BC: 48       >685           PHA
43BD: AD 00 24 >686           LDA   $2400
43C0: 48       >687           PHA
43C1: A9 EE    >688           LDA   #$EE
43C3: 8D 00 04 >689           STA   $0400
43C6: AD 00 24 >690           LDA   $2400
43C9: C9 EE    >691           CMP   #$EE
43CB: D0 0B    >692           BNE   :2
43CD: 0E 00 24 >693           ASL   $2400
43D0: AD 00 04 >694           LDA   $0400
43D3: CD 00 24 >695           CMP   $2400
43D6: F0 1B    >696           BEQ   :3
43D8: E8       >697  :2       INX
43D9: A9 0F    >698           LDA   #$0F
```

```
43DB: 8D B9 C0 >699           STA   $C0B9
43DE: 8D 54 C0 >700           STA   $C054
43E1: AD 00 04 >701           LDA   $0400
43E4: 8D 00 04 >702           STA   $0400
43E7: 8D B8 C0 >703           STA   $C0B8
43EA: 8D 55 C0 >704           STA   $C055
43ED: AD 00 04 >705           LDA   $0400
43F0: 30 01    >706           BMI   :3
43F2: E8       >707           INX
43F3: 68       >708  :3       PLA
43F4: 8D 00 24 >709           STA   $2400
43F7: 68       >710           PLA
43F8: 8D 00 04 >711           STA   $0400
43FB: 68       >712           PLA
43FC: 30 03    >713           BMI   :4
43FE: 8D 54 C0 >714           STA   $C054
4401: 68       >715  :4       PLA
4402: 30 03    >716           BMI   :5
4404: 8D 00 C0 >717           STA   $C000
4407: 68       >718  :5       PLA
4408: 30 03    >719           BMI   :6
440A: 8D 56 C0 >720           STA   $C056
               >721  * X=0: No 80 col. card in aux. slot
               >722  * X=1: 80 col. card w/o memory expansion
               >723  * X=2: 80 col. card with at least 64K mem. expansion
               >724  * X=3: Same as above + special video modes (Eve le chat mau
ve)
440D: BD E1 42 >725  :6       LDA   CFM,X
4410: 48       >726           PHA
4411: BD E5 42 >727           LDA   CFA,X
4414: AA       >728           TAX
4415: 68       >729           PLA
4416: 28       >730           PLP
4417: 60       >731           RTS
                298           PUT   PEERAUXINSTALL
               >1    STRNG2   EQU   $AD
               >2    FRETOP   EQU   $6F
               >3    HIMEM    EQU   $73
               >4    ALTZP    EQU   $C009
               >5    STDZP    EQU   $C008
               >6    RD80STOR EQU   $C018
               >7    RDLCRAM  EQU   $C012
               >8    RDLCBNK2 EQU   $C011
               >9    GARBAG   EQU   $E484
               >10
               >11   INITBF   STID  CODE1BF;A1L
4418: A9 8F    >11            LDA   #CODE1BF
441A: 85 3C    >11            STA   A1L
441C: A9 44    >11            LDA   #>CODE1BF
441E: 85 3D    >11            STA   A1L+1
4420: A0 00    >12            LDY   #0
4422: A9 00    >13            LDA   #GZAUXRT
4424: 85 3E    >13            STA   A2L
4426: A9 BF    >13            LDA   #>GZAUXRT
4428: 85 3F    >13            STA   A2L+1
442A: 8D 05 C0 >14            STA   $C005
442D: B1 3C    >15   ]LOOP    LDA   (A1L),Y
```

```
442F: 91 3E    >16            STA   (A2L),Y
4431: C8       >17            INY
4432: C0 BB    >18            CPY   #CODE2BF-CODE1BF
4434: D0 F7    >19            BNE   ]LOOP
4436: 8D 04 C0 >20            STA   $C004
4439: 08       >21            PHP
443A: 08       >22            PHP
443B: 68       >23            PLA
443C: 78       >24            SEI
443D: BA       >25            TSX
443E: 8E 09 C0 >26            STX   ALTZP
4441: 8E 00 01 >27            STX   $0100
4444: A2 FF    >28            LDX   #$FF
4446: 9A       >29            TXS
4447: 8E 01 01 >30            STX   $0101
444A: 29 04    >31            AND   #%100
444C: D0 01    >32            BNE   *+3
444E: 58       >33            CLI
444F: A9 4A    >34            LDA   #CODE1LC
4451: 85 3C    >34            STA   A1L
4453: A9 45    >34            LDA   #>CODE1LC
4455: 85 3D    >34            STA   A1L+1
4457: A9 00    >35            LDA   #$D000
4459: 85 3E    >35            STA   A2L
445B: A9 D0    >35            LDA   #>$D000
445D: 85 3F    >35            STA   A2L+1
445F: 2C 81 C0 >36            BIT   $C081
4462: 2C 81 C0 >37            BIT   $C081
4465: B2 3C    >39   ]LOOP    LDA   (A1L)
4467: 92 3E    >40            STA   (A2L)
4469: E6 3C    >46            INC   A1L
446B: D0 02    >47            BNE   *+4
446D: E6 3D    >48            INC   A1L+1
446F: A5 3C    >49            LDA   A1L
4471: C9 EB    >50            CMP   #CODE2LC
4473: A5 3D    >51            LDA   A1L+1
4475: E9 45    >52            SBC   #>CODE2LC
4477: B0 08    >53            BCS   :0
4479: E6 3E    >54            INC   A2L
447B: D0 E8    >55            BNE   ]LOOP
447D: E6 3F    >56            INC   A2L+1
447F: 90 E4    >57            BCC   ]LOOP      Always
4481: 78       >58   :0       SEI
4482: BA       >59            TSX
4483: 8E 01 01 >60            STX   $0101
4486: AE 00 01 >61            LDX   $0100
4489: 9A       >62            TXS
448A: 8E 08 C0 >63            STX   STDZP
448D: 28       >64            PLP
448E: 60       >65   ]RET     RTS
               >66
               >67   CODE1BF  ORG   $BF00
               >68   AXHIMEM  EQU   *
               >69   * Routine de redirection pour la gestion des tableaux en
               >70   * memoire auxiliaire.
               >71   * X:0 init the auxilary memory segment for storing
               >72   *     array elements
```

```
                >73   * X:1 check that enough room exists for storing an
                >74   *     array´s elements
                >75   * X:2 actually updates the STREND new array end and
                >76   *     initializes the area.
                >77   * X:3 returns the mem bank free space after a garbage c.
                >78   * X:4 retrieve an array´s element from memory.
                >79   * X:5 stores an array´s element into memory
BF00: BC B7 BF  >80   GZAUXRT  LDY   ZAUXOFFT,X offset into Y
BF03: A9 00     >81            LDA   #0
BF05: 2C 12 C0  >82            BIT   RDLCRAM
BF08: 10 09     >83            BPL   *+11
BF0A: 09 0C     >84            ORA   #12
BF0C: 2C 11 C0  >85            BIT   RDLCBNK2
BF0F: 10 02     >86            BPL   *+4
BF11: 49 06     >87            EOR   #6
BF13: 48        >88            PHA
BF14: 08        >89            PHP                ;Save I bit flag on main stk
BF15: 68        >90            PLA                ;Restore in b2 of accum.
BF16: BA        >91            TSX
BF17: 78        >92            SEI
BF18: 8D 09 C0  >93            STA   ALTZP
BF1B: 8E 00 01  >94            STX   $0100
BF1E: A2 FF     >95            LDX   #$FF
BF20: 8E 01 01  >96            STX   $0101
BF23: 9A        >97            TXS
BF24: 29 04     >98            AND   #%100        bit I mask
BF26: D0 01     >99            BNE   *+3
BF28: 58        >100           CLI
BF29: AD 18 C0  >101           LDA   RD80STOR
BF2C: 48        >102           PHA
BF2D: 8D 00 C0  >103           STA   $C000        Enable basic access to screens
                >104  * Read/Write enable LC bank 2 in aux. mem. bec. of ALTZP
BF30: 20 A3 BF  >105           JSR   G83          Read/Write enable LC bank 2 in
BF33: A9 BF     >113           LDA   #>ZAUXRET-1
BF35: 48        >114           PHA
BF36: A9 3C     >115           LDA   #ZAUXRET-1
BF38: 48        >116           PHA
BF39: 18        >118           CLC
BF3A: 4C 14 D0  >119           JMP   ZAUXRT
                >120
                >121  * Routine de retour general vers le composant principal
                >122  * de Peersoft (en memoire principale)
BF3D: 68        >126  ZAUXRET  PLA                ;Restore RD80STOR status
BF3E: 10 03     >128           BPL   *+5          from aux stack..
BF40: 8E 01 C0  >129           STX   $C001        If On, then set it back..
BF43: 08        >130           PHP                ;Save carry flag
BF44: 28        >137           PLP                ;Restore carry flag
BF45: AE 00 01  >138           LDX   $0100        Get back main stack pointer
BF48: 9A        >139           TXS                ; from $0100 aux stack byte
BF49: 8E 08 C0  >140           STX   STDZP        Return to Std stack/p0
BF4C: 68        >144           PLA                ;Restore configuration flag
BF4D: 08        >145           PHP                ;Carry back into main stack
BF4E: 20 AA BF  >146           JSR   G81
BF51: 0A        >147           ASL
BF52: F0 0E     >148           BEQ   :0
BF54: A0 05     >149           LDY   #5
BF56: BE B1 BF  >150  ]LOOP    LDX   IRQTBLE,Y
```

```
BF59: 88       >151           DEY
BF5A: 0A       >152           ASL
BF5B: 90 03    >153           BCC   *+5
BF5D: 9D 00 C0 >154           STA   $C000,X
BF60: D0 F4    >155           BNE   ]LOOP
BF62: 28       >156  :0       PLP
               >157  * X set to zero upon return according to carry flag
BF63: A2 00    >158           LDX   #0
BF65: 90 01    >159           BCC   *+3
BF67: E8       >160           INX
BF68: 68       >161           PLA             ;Get return address
BF69: 7A       >163           PLY             ; from main stack
BF6A: 1A       >164           INC
BF6B: 8D ED 03 >165           STA   $03ED
BF6E: D0 01    >166           BNE   *+3
BF70: C8       >167           INY
BF71: 8C EE 03 >168           STY   $03EE
BF74: 18       >179           CLC
BF75: B8       >180           CLV
BF76: 4C 14 C3 >181           JMP   XFER      Retour a l´envoyeur
               >182
BF79: 08       >183  ZGCPARMS PHP
BF7A: 78       >184           SEI
BF7B: 8E 08 C0 >185           STX   STDZP
BF7E: A5 AD    >196           LDA   STRNG2
BF80: A6 AE    >197           LDX   STRNG2+1
BF82: 69 07    >198           ADC   #7
BF84: 8E 09 C0 >199           STX   ALTZP
BF87: 90 06    >200           BCC   :0
BF89: E8       >201           INX
BF8A: D0 03    >202           BNE   :0
BF8C: 28       >203           PLP
BF8D: 38       >204           SEC
BF8E: 60       >205           RTS
BF8F: 28       >207  :0       PLP
BF90: 18       >208           CLC
BF91: 60       >209  ]RET     RTS
               >210
BF92: 8E 08 C0 >211  ZGCP2    STX   STDZP
BF95: 85 AE    >215           STA   STRNG2+1
BF97: 86 AD    >216           STX   STRNG2
BF99: 8E 09 C0 >218           STX   ALTZP
BF9C: 60       >219           RTS
               >220
BF9D: 20 AA BF >224  ZNG       JSR   G81
BFA0: 20 84 E4 >225           JSR   GARBAG
BFA3: 2C 83 C0 >226  G83       BIT   $C083
BFA6: 2C 83 C0 >227           BIT   $C083
BFA9: 60       >228           RTS
BFAA: 2C 81 C0 >229  G81       BIT   $C081
BFAD: 2C 81 C0 >230           BIT   $C081
BFB0: 60       >232           RTS
               >233
BFB1: 83 8B 8B >234  IRQTBLE   HEX   838B8B
BFB4: 05 03 55 >235           HEX   050355
BFB7: 00 21    >236  ZAUXOFFT DFB   ZAUXRT0-ZAUXB,ZAUXRT1-ZAUXB
BFB9: 2C E7    >237           DFB   ZAUXRT2-ZAUXB,ZAUXRT3-ZAUXB
```

```
                >241            ERR   */$C000
                >242            ORG
                >243  CODE2BF
                >244  CODE1LC   ORG   $D000
                >245  * Y offset correspondant a X
                >246  * X:0 init the auxilary memory segment for storing
                >247  *     array elements
                >248  * X:1 check that enough room exists for storing an
                >249  *     array´s elements
                >250  * X:2 actually updates the STREND new array end and
                >251  *     initializes the area.
                >252  * X:3 returns the mem bank free space after a garbage c.
                >253  * X:4 retrieve an array´s element from memory.
                >254  * X:5 stores an array´s element into memory
                >255
                >256  * Returns amount of free space in aux memory bank
                >257  * after calling ROM based garbage collection.
D000: 20 9D BF  >270  ZAUXRT3  JSR   ZNG        Fall in a main 48K routine
D003: 38        >271           SEC
D004: A5 6F     >272           LDA   FRETOP
D006: E5 6D     >273           SBC   STREND
D008: AA        >274           TAX
D009: A5 70     >275           LDA   FRETOP+1
D00B: E5 6E     >276           SBC   STREND+1
D00D: 08        >277           PHP
D00E: 78        >278           SEI
D00F: 20 92 BF  >279           JSR   ZGCP2
D012: 28        >281           PLP
D013: 60        >282  ]RET     RTS
                >283
D014: 8C 18 D0  >284  ZAUXRT   STY   *+4
D017: D0 00     >285           BNE   ZAUXRT0
                >286  * User subroutine is called with Aux mem. stack/p0,
                >287  * 16bits Accu/mem access if 65802/816.
                >288  * Stack pointer set to $FD (a return address ZAUXRET)
                >289  ZAUXB    EQU   *
                >290
                >291  * Do the init
D019: AD 97 D0  >302  ZAUXRT0  LDA   AXARTAB
D01C: 85 69     >303           STA   VARTAB
D01E: 85 6B     >304           STA   ARYTAB
D020: 85 6D     >305           STA   STREND
D022: AD 98 D0  >306           LDA   AXARTAB+1
D025: 85 6A     >307           STA   VARTAB+1
D027: 85 6C     >308           STA   ARYTAB+1
D029: 85 6E     >309           STA   STREND+1
D02B: 64 73     >311           STZ   HIMEM
D02D: 64 6F     >312           STZ   FRETOP
D02F: A9 BF     >318           LDA   #>AXHIMEM
D031: 85 74     >319           STA   HIMEM+1
D033: 85 70     >320           STA   FRETOP+1
D035: A2 55     >322           LDX   #$55       Pour le Garbage collector...
D037: 86 52     >323           STX   $52
D039: 60        >324  ]RET     RTS
                >325
                >326  * Ensure enough room within array segment
D03A: 20 83 D0  >327  ZAUXRT1  JSR   ZCOMRT12
```

```
D03D: B0 FA    >328           BCS   ]RET
D03F: C5 6F    >329           CMP   FRETOP
D041: 8A       >331           TXA
D042: E5 70    >332           SBC   FRETOP+1
D044: 60       >334  ]RET     RTS
               >335
D045: A5 6D    >336  ZAUXRT2  LDA   STREND
D047: 85 3C    >337           STA   A1L
D049: A5 6E    >339           LDA   STREND+1
D04B: 85 3D    >340           STA   A1L+1
D04D: 20 83 D0 >342           JSR   ZCOMRT12
D050: B0 F2    >343           BCS   ]RET
D052: A0 02    >344           LDY   #2
D054: 86 6E    >351           STX   STREND+1
D056: 85 6D    >352           STA   STREND
               >353  * Offset to next array (low byte)
D058: 38       >354           SEC
D059: E5 3C    >355           SBC   A1L
D05B: 91 3C    >356           STA   (A1L),Y
D05D: C8       >357           INY
               >358  * and hi byte
D05E: 8A       >359           TXA
D05F: E5 3D    >360           SBC   A1L+1
D061: 91 3C    >361           STA   (A1L),Y
               >368  * # of dimensions
D063: A9 01    >369           LDA   #1
D065: A0 04    >370           LDY   #4
D067: 91 3C    >371           STA   (A1L),Y
               >372  * Init segment where elms will be stored
D069: A2 00    >373           LDX   #0
D06B: A4 3C    >374           LDY   A1L
D06D: 86 3C    >375           STX   A1L
D06F: C4 6D    >376  ]LOOP    CPY   STREND
D071: A5 3D    >377           LDA   A1L+1
D073: E5 6E    >378           SBC   STREND+1
D075: B0 0A    >379           BCS   *+12
D077: 8A       >380           TXA
D078: 91 3C    >381           STA   (A1L),Y
D07A: C8       >382           INY
D07B: D0 F2    >383           BNE   ]LOOP
D07D: E6 3D    >384           INC   A1L+1
D07F: 90 EE    >385           BCC   ]LOOP      Always
D081: 18       >386           CLC
D082: 60       >392  ]RET     RTS
               >393
               >466
D083: 20 79 BF >467  ZCOMRT12 JSR   ZGCPARMS
D086: B0 FA    >468           BCS   ]RET
D088: 65 6D    >469           ADC   STREND
D08A: A8       >473           TAY
D08B: 8A       >474           TXA
D08C: 65 6E    >475           ADC   STREND+1
D08E: AA       >476           TAX
D08F: 98       >477           TYA
               >478  * Result in X,A
D090: 60       >480  ]RET     RTS
               >481
```

```
                    >482
D091: FF 80 80      >483  TELMS    HEX   FF8080808000
D097: 00 08         >484  AXARTAB  DA    $0800      0
                    >485  AXARYPNT EQU   AXARTAB    2
D099: 00 00         >486  AXOFFSET DS    2
D09B: 00            >487  ELMSIZ   DS    1          2
D09C: 00 00 00      >488  AXVALUE  DS    5
                    >489  AXARYPT2 EQU   AXVALUE
                    >490  *ZAUXRTF EQU *
                    >491          ORG
                    >492  CODE2LC  EQU   *
                     299          PUT   PEERFGC
                    >1    * Fast garbage collector
                    >2    * Credits: Randy Wiggington
                    >3    STRNG    EQU   $19
                    >4    XXSAV    EQU   $1B
                    >5    PTR2     EQU   $1C
                    >6    DSCLEN   EQU   $8F
                    >7    NUMELS   =     8
                    >8    NUMELS2  =     NUMELS*2
                    >9
45EB: A0 00         >10   INITLC   LDY   #0
45ED: A9 06         >14           LDA   #CODE1GC
45EF: 85 3C         >14           STA   A1L
45F1: A9 46         >14           LDA   #>CODE1GC
45F3: 85 3D         >14           STA   A1L+1
45F5: A9 9F         >15           LDA   #CODE1GCF
45F7: 85 3E         >15           STA   A2L
45F9: A9 47         >15           LDA   #>CODE1GCF
45FB: 85 3F         >15           STA   A2L+1
45FD: 84 42         >16           STY   A4L
45FF: A9 D0         >17           LDA   #>$D000
4601: 85 43         >18           STA   A4L+1
4603: 4C 2C FE      >19           JMP   MOVE
                    >20
                    >21   CODE1GC  ORG   $D000
D000: A6 73         >150          LDX   HIMEM
D002: A5 74         >151          LDA   HIMEM+1
D004: 86 6F         >152  FNDVAR   STX   FRETOP
D006: 85 70         >153          STA   FRETOP+1
D008: 4C E2 D0      >154          JMP   NZTAB
D00B: A5 6D         >155  FNDVARX2 LDA   STREND
D00D: A6 6E         >156          LDX   STREND+1
D00F: A9 55         >157          LDA   #$55
D011: 85 5E         >158          STA   INDEX
D013: 64 5F         >160          STZ   INDEX+1
D015: C5 52         >165  ]LOOP    CMP   $52
D017: F0 05         >166          BEQ   SVARS
D019: 20 FF D0      >167          JSR   DVAR
D01C: F0 F7         >168          BEQ   ]LOOP
D01E: A9 07         >169  SVARS    LDA   #7
D020: 85 8F         >170          STA   DSCLEN
D022: A5 69         >171          LDA   VARTAB
D024: A6 6A         >172          LDX   VARTAB+1
D026: 85 5E         >173          STA   INDEX
D028: 86 5F         >174          STX   INDEX+1
D02A: E4 6C         >175  ]LOOP    CPX   ARYTAB+1
```

```
D02C: D0 04    >176           BNE   *+6
D02E: C5 6B    >177           CMP   ARYTAB
D030: F0 05    >178           BEQ   ARYVAR
D032: 20 F2 D0 >179           JSR   DVARS
D035: F0 F3    >180           BEQ   ]LOOP
D037: 85 94    >181  ARYVAR   STA   ARYPNT
D039: 86 95    >182           STX   ARYPNT+1
D03B: A9 03    >183           LDA   #3
D03D: 85 8F    >184           STA   DSCLEN
D03F: A5 94    >185  ]LOOP    LDA   ARYPNT
D041: A6 95    >186           LDX   ARYPNT+1
D043: E4 6E    >187  ]LOOP1   CPX   STREND+1
D045: D0 04    >188           BNE   *+6
D047: C5 6D    >189           CMP   STREND
D049: F0 4D    >190           BEQ   GRBPAS
D04B: 85 5E    >191           STA   INDEX
D04D: 86 5F    >192           STX   INDEX+1
D04F: B2 5E    >194           LDA   (INDEX)
D051: A0 01    >195           LDY   #1
D053: AA       >201           TAX
D054: B1 5E    >202           LDA   (INDEX),Y  Name 2nd character
D056: 08       >203           PHP
D057: C8       >204           INY
D058: B1 5E    >205           LDA   (INDEX),Y
D05A: 65 94    >206           ADC   ARYPNT     Carry clear
D05C: 85 94    >207           STA   ARYPNT
D05E: C8       >208           INY
D05F: B1 5E    >209           LDA   (INDEX),Y
D061: 65 95    >210           ADC   ARYPNT+1
D063: 85 95    >211           STA   ARYPNT+1
D065: 28       >212           PLP
D066: 10 D7    >213           BPL   ]LOOP
D068: 8A       >214           TXA
D069: 30 D4    >215           BMI   ]LOOP
D06B: C8       >216           INY              ;Y vaut 4
D06C: B1 5E    >217           LDA   (INDEX),Y
D06E: AA       >218           TAX
D06F: 25 38    >219           AND   %111000
D071: 08       >220           PHP
D072: 8A       >221           TXA
D073: 29 07    >222           AND   #7
D075: 1A       >224           INC
D076: A0 00    >228           LDY   #0
D078: 0A       >229           ASL
D079: 28       >230           PLP
D07A: F0 03    >231           BEQ   *+5
D07C: 69 0B    >232           ADC   #5+6
D07E: 2C       >233           HEX   2C         Skip next two bytes
D07F: 69 05    >234           ADC   #5
D081: 65 5E    >235           ADC   INDEX
D083: 85 5E    >236           STA   INDEX
D085: 90 02    >237           BCC   *+4
D087: E6 5F    >238           INC   INDEX+1
D089: A6 5F    >239           LDX   INDEX+1
               >240  * End of the array?
D08B: E4 95    >241  ]LOOP    CPX   ARYPNT+1
D08D: D0 04    >242           BNE   *+6
```

```
D08F: C5 94    >243           CMP   ARYPNT
D091: F0 B0    >244           BEQ   ]LOOP1
D093: 20 FF D0 >245           JSR   DVAR
D096: F0 F3    >246           BEQ   ]LOOP
               >248
               >249  * Have made a complete pass thru the variables
               >250  * Now collect the ones in the list
D098: A2 0F    >251  GRBPAS    LDX   #NUMELS2-1
D09A: BD A9 D1 >266  ]LOOP     LDA   LENTHS,X
D09D: A8       >267           TAY
D09E: F0 F0    >268           BEQ   ]RET
D0A0: 38       >269           SEC
D0A1: A5 6F    >270           LDA   FRETOP
D0A3: FD A9 D1 >271           SBC   LENTHS,X
D0A6: 85 6F    >272           STA   FRETOP
D0A8: A5 70    >273           LDA   FRETOP+1
D0AA: E9 00    >274           SBC   #0
D0AC: 85 70    >275           STA   FRETOP+1
D0AE: BD 98 D1 >276           LDA   BTMEL-1,X  Get current place
D0B1: 85 1C    >277           STA   PTR2
D0B3: BD 99 D1 >278           LDA   BTMEL,X
D0B6: 85 1D    >279           STA   PTR2+1
D0B8: 88       >281  ]LOOP1    DEY
D0B9: C0 FF    >282           CPY   #$FF
D0BB: F0 06    >283           BEQ   *+8
D0BD: B1 1C    >284           LDA   (PTR2),Y
D0BF: 91 6F    >285           STA   (FRETOP),Y
D0C1: 90 F5    >286           BCC   ]LOOP1     Always
D0C3: BD A8 D1 >287           LDA   LENTHS-1,X Get size of variable
D0C6: 29 04    >288           AND   #4
D0C8: 4A       >289           LSR
D0C9: A8       >290           TAY
D0CA: C8       >291           INY
D0CB: BD B8 D1 >296           LDA   VARPT-1,X
D0CE: 85 1C    >297           STA   PTR2
D0D0: BD B9 D1 >299           LDA   VARPT,X
D0D3: 85 1D    >300           STA   PTR2+1
D0D5: A5 6F    >302           LDA   FRETOP
D0D7: 91 1C    >303           STA   (PTR2),Y
D0D9: C8       >305           INY
D0DA: A5 70    >306           LDA   FRETOP+1
D0DC: 91 1C    >307           STA   (PTR2),Y
D0DE: CA       >309           DEX
D0DF: CA       >310           DEX
D0E0: 10 B8    >311           BPL   ]LOOP
D0E2: A2 0F    >315  NZTAB     LDX   #NUMELS2-1
D0E4: A9 00    >325           LDA   #0
D0E6: 9D A9 D1 >326  ]LOOP     STA   LENTHS,X
D0E9: 9D 99 D1 >327           STA   BTMEL,X
D0EC: CA       >328           DEX
D0ED: 10 F7    >329           BPL   ]LOOP
D0EF: 4C 0B D0 >331           JMP   FNDVARX2
               >333  * Garbage collection for simple variables
D0F2: B1 5E    >334  DVARS     LDA   (INDEX),Y  Is it a string var
D0F4: 30 05    >335           BMI   GDVARTS
D0F6: C8       >336           INY
D0F7: B1 5E    >337           LDA   (INDEX),Y
```

```
D0F9: 30 03    >338           BMI   *+5
D0FB: 4C 89 D1 >339  GDVARTS  JMP   DVARTS
D0FE: C8       >340           INY
D0FF: B1 5E    >341  DVAR     LDA   (INDEX),Y
D101: F0 F8    >342           BEQ   GDVARTS    Skip Zero length strings
D103: 85 2F    >343           STA   LENGTH
D105: C8       >344           INY
D106: B1 5E    >345           LDA   (INDEX),Y
D108: 85 19    >346           STA   STRNG
D10A: C5 6F    >347           CMP   FRETOP     Is this above where we are?
D10C: C8       >348           INY
D10D: B1 5E    >349           LDA   (INDEX),Y
D10F: 85 1A    >350           STA   STRNG+1
D111: E5 70    >351           SBC   FRETOP+1
D113: B0 E6    >352           BCS   GDVARTS    This one´s been collected before
D115: A5 19    >353           LDA   STRNG      Is it in our range?
D117: CD 99 D1 >354           CMP   BTMEL      Compare to lowest value in list
D11A: A5 1A    >355           LDA   STRNG+1
D11C: ED 9A D1 >356           SBC   BTMEL+1
D11F: 90 68    >357           BCC   DVARTS     No, below lowest, go to next one
D121: A5 19    >358           LDA   STRNG
D123: C5 6D    >359           CMP   STREND
D125: A5 1A    >360           LDA   STRNG+1
D127: E5 6E    >361           SBC   STREND+1
D129: 90 D0    >362           BCC   GDVARTS    Inside the program...
D12B: A2 11    >363           LDX   #NUMELS2+1 Search thru list of elements
D12D: CA       >364  ]LOOP    DEX
D12E: CA       >365           DEX
D12F: A5 19    >366           LDA   STRNG
D131: DD 98 D1 >367           CMP   BTMEL-1,X
D134: A5 1A    >368           LDA   STRNG+1
D136: FD 99 D1 >369           SBC   BTMEL,X
D139: 90 F2    >370           BCC   ]LOOP
D13B: 86 1B    >371           STX   XXSAV
D13D: A2 03    >372           LDX   #3         Make room in table for entry
D13F: BD 98 D1 >373  ]LOOP    LDA   BTMEL-1,X
D142: 9D 96 D1 >374           STA   BTMEL-3,X
D145: BD 99 D1 >375           LDA   BTMEL,X
D148: 9D 97 D1 >376           STA   BTMEL-2,X  Ripple down
D14B: BD A9 D1 >377           LDA   LENTHS,X
D14E: 9D A7 D1 >378           STA   LENTHS-2,X
D151: BD A8 D1 >379           LDA   LENTHS-1,X
D154: 9D A6 D1 >380           STA   LENTHS-3,X
D157: BD B9 D1 >381           LDA   VARPT,X
D15A: 9D B7 D1 >382           STA   VARPT-2,X
D15D: BD B8 D1 >383           LDA   VARPT-1,X
D160: 9D B6 D1 >384           STA   VARPT-3,X
D163: E4 1B    >385           CPX   XXSAV
D165: E8       >386           INX
D166: E8       >387           INX
D167: 90 D6    >388           BCC   ]LOOP
D169: A6 1B    >389           LDX   XXSAV
D16B: A5 19    >390           LDA   STRNG
D16D: 9D 98 D1 >391           STA   BTMEL-1,X
D170: A5 1A    >392           LDA   STRNG+1
D172: 9D 99 D1 >393           STA   BTMEL,X
D175: A5 2F    >394           LDA   LENGTH
```

```
D177: 9D A9 D1 >395           STA   LENTHS,X
D17A: A5 5E    >396           LDA   INDEX
D17C: 9D B8 D1 >397           STA   VARPT-1,X
D17F: A5 5F    >398           LDA   INDEX+1
D181: 9D B9 D1 >399           STA   VARPT,X
D184: A5 8F    >400           LDA   DSCLEN
D186: 9D A8 D1 >401           STA   LENTHS-1,X
D189: A5 8F    >402  DVARTS   LDA   DSCLEN
D18B: 18       >403           CLC
D18C: 65 5E    >404           ADC   INDEX
D18E: 85 5E    >405           STA   INDEX
D190: 90 02    >406           BCC   *+4
D192: E6 5F    >407           INC   INDEX+1
D194: A6 5F    >408           LDX   INDEX+1
D196: A0 00    >409           LDY   #0
D198: 60       >410           RTS
               >415           DUMMY *
D199: 00 00 00 >416  BTMEL     DS    NUMELS*2
D1A9: 00 00 00 >417  LENTHS    DS    NUMELS*2
D1B9: 00 00 00 >418  VARPT     DS    NUMELS*2
               >419           DEND
               >420           ORG
               >421  CODE1GCF EQU   *
                300  * Here is the Peersoft real origine
                307  AROMBA   ORG   $8CFC+$C00-$96-80-$56-$37-$4C-$B7-$54A-$1B33

                310  FNDVAR2
                311  CGARBAG
                312
                313  * All calls to CHRGET fall into this routine
7609: 86 B4     314  DEBUTGET STX   XSAV
760B: 84 B5     315           STY   YSAV
                316  * Check return address
760D: BA        322           TSX
760E: BD 02 01  323           LDA   $0102,X    hi byte
7611: 85 C2     324           STA   OFFSET
7613: BD 01 01  325           LDA   $0101,X    lo byte
7616: A2 14     327           LDX   #ADAPFTET-ADAPFBET
7618: DD AF 9B  328  ]LOOP    CMP   ADAPFBET-1,X
761B: D0 07     329           BNE   :0
761D: BC C3 9B  330           LDY   ADAPFTET-1,X
7620: C4 C2     331           CPY   OFFSET     Test for a match upon
7622: F0 20     332           BEQ   OKP1GET    return address: proceed
7624: CA        333  :0       DEX              ;No match: loop till
7625: D0 F1     334           BNE   ]LOOP       all values exhaustion
7627: A4 B5     335           LDY   YSAV
7629: 2C        339           HEX   2C         Skip next two bytes
                341  * No address match: exit with a simulation of CHRGET
762A: 86 B4     342  RST100   STX   XSAV
                346  RST101
762C: E6 B8     348  LLOOP    INC   TXTPTR
762E: D0 02     349           BNE   COMRST
7630: E6 B9     350           INC   TXTPTR+1
                355  RST102
                356  RST103
7632: B2 B8     357  COMRST   LDA   (TXTPTR)
7634: C9 20     359           CMP   #$20
```

```
7636: F0 F4     360           BEQ   LLOOP
7638: A6 B4     361           LDX   XSAV
763A: C9 3A     362 COMRSTC   CMP   #´:´
763C: B0 05     363           BCS   :0
763E: E9 2F     364           SBC   #$30-1     Because of carry clear
7640: 38        365           SEC
7641: E9 D0     366           SBC   #$D0
7643: 60        367 :0        RTS
                372
                373 OKP1GET
                374 * Tricky way to replace the two bytes at the top of stack
                375 * Instead of doing PLA PLA followed by PHA PHA...
7644: 8A        382           TXA              ;X into Y
7645: A8        383           TAY
7646: BA        384           TSX
7647: B9 D7 9B  385           LDA   ADPFB-1,Y
764A: 9D 01 01  386           STA   $0101,X
764D: B9 EB 9B  387           LDA   ADPFT-1,Y
7650: 9D 02 01  388           STA   $0102,X
7653: D0 D7     390           BNE   RST101     Always
7655: 4C 08 7A  391 GNPTRGET  JMP   NPTRGET
7658: 86 B4     392 DEBUTGOT  STX   XSAV
765A: BA        393           TSX
765B: BD 01 01  397           LDA   $0101,X
765E: C9 EE     399           CMP   #VPTRGET-1
7660: D0 D0     400           BNE   RST103
7662: BD 02 01  404           LDA   $0102,X
7665: 49 DF     406           EOR   #>VPTRGET-1 A=0 upon matching address
7667: D0 C9     407           BNE   RST103
7669: E8        414           INX              ;Quick way to pull two bytes
766A: E8        415           INX              ; from stack
766B: BD 02 01  416           LDA   $0102,X
766E: C9 DA     418           CMP   #>VLET+2
7670: D0 03     419           BNE   :44
7672: E8        420           INX
7673: E8        421           INX              ;Carry set at this time
7674: 24        422           HEX   24         Skip next byte
7675: 18        423 :44       CLC
7676: 9A        424           TXS
7677: A2 00     425           LDX   #0
7679: 90 DA     426           BCC   GNPTRGET
                427 * The following routine handles the Applesoft
                428 * variable setting
                429 * (LET is the optional keyword)
767B: 20 08 7A  430 RLET      JSR   NPTRGET
767E: 85 85     431           STA   FORPNT
7680: 84 86     432           STY   FORPNT+1
7682: A6 BF     433           LDX   AUXBANK
7684: F0 24     434           BEQ   RLET1
7686: A5 9C     439           LDA   LOWTR+1
7688: 48        440           PHA
7689: A5 9B     441           LDA   LOWTR
768B: 48        442           PHA
768C: A5 AE     443           LDA   STRNG2+1
768E: 48        444           PHA
768F: A5 AD     445           LDA   STRNG2
7691: 48        446           PHA
```

```
7692: DA        448           PHX
7693: 20 AA 76  449           JSR   RLET1
7696: 68        450           PLA
7697: 85 BF     451           STA   AUXBANK
7699: 68        452           PLA
769A: 85 AD     453           STA   STRNG2
769C: 68        454           PLA
769D: 85 AE     455           STA   STRNG2+1
769F: 68        456           PLA
76A0: 85 9B     457           STA   LOWTR
76A2: 68        458           PLA
76A3: 85 9C     459           STA   LOWTR+1
76A5: A2 05     460           LDX   #5
76A7: 4C CB 7D  461           JMP   ZRTAUX
                462
76AA: B2 B8     467  RLET1    LDA   (TXTPTR)
76AC: A2 03     469           LDX   #3          New syntax scheme?
76AE: DD 13 96  470  ]LOOP    CMP   TOKENS,X
76B1: F0 26     471           BEQ   :0          yes so handle it
76B3: CA        472           DEX
76B4: 10 F8     473           BPL   ]LOOP
76B6: A9 D0     474           LDA   #TOKEQUAL
76B8: 20 76 7E  475           JSR   NSYNCHR2    Y vaut deja zero si 6502
76BB: A5 12     476           LDA   INTTYP
76BD: 10 14     477           BPL   :11
76BF: 48        478           PHA
76C0: 20 3B 85  479           JSR   NFRMNUM
76C3: 20 4E 78  480  ]LOOP    JSR   NROUT
76C6: 68        481           PLA
76C7: C9 81     482           CMP   #$81        Byte subtype?
76C9: D0 0B     483           BNE   :12
76CB: 20 EA 79  484           JSR   CONV1628
76CE: A5 A1     485           LDA   FAC+4
76D0: 92 85     487           STA   (FORPNT)
76D2: 60        492           RTS
76D3: 4C 52 DA  493  :11      JMP   VLET+12
76D6: 4C 6B DA  494  :12      JMP   $DA6B
                495
                496  * Save selected operation on stack (+,-,*,/)
                497  :0       MPHX
76D9: DA        497           PHX
76DA: 20 2C 76  498           JSR   RST101      Bump next character
                499  * Ensure that next char is ´=´ symbol token
76DD: A9 D0     500           LDA   #TOKEQUAL
76DF: 20 76 7E  501           JSR   NSYNCHR2    no need to reset Y to 0
                502  * Save variable type on stack
76E2: A5 12     503           LDA   INTTYP      $80 iif integer variable
76E4: 48        504           PHA
76E5: A5 11     505           LDA   VALTYP      $FF iif string
76E7: 48        506           PHA
76E8: 20 7B DD  507           JSR   FRMEVL
76EB: 68        508           PLA
76EC: 2A        509           ROL               ;Carry set iif var. type string
76ED: 20 6D DD  510           JSR   $DD6D       Check FRMEVL result type accordin
g to C
76F0: 68        511           PLA               ;Get INTTYP off stack
76F1: B0 22     512           BCS   HNDLESTR    String variable and expression
```

```
                    513  * From then on: we´ll handle numeric var. and expr.
76F3: 30 4B         514           BMI   HNDLEINT
76F5: A4 86         515  HNDLEREA LDY   FORPNT+1
76F7: 68            516           PLA
76F8: 0A            518           ASL
76F9: AA            520           TAX
76FA: A9 EB         524           LDA   #>$EB27-1
76FC: 48            525           PHA
76FD: A9 26         526           LDA   #$EB27-1
76FF: 48            527           PHA
7700: A5 85         530           LDA   FORPNT
7702: 7C 17 96      531           JMP   (FPROUTS,X)
                    540
7705: 4C 27 EB      541  ]LOOP1   JMP   $EB27      SETFOR
7708: A5 12         542  NLET2    LDA   INTTYP
770A: 10 F9         543           BPL   ]LOOP1
770C: 48            544           PHA
770D: 30 B4         545           BMI   ]LOOP       Always
                    546
                    547  * Includes module for handling integ. arithmetic
                    548  * and <op>= instructions
                    549           PUT   PEERINTEGARITH
                   >1    * Module handling all integer arithmetic
                   >2    * within Peersoft and all op= instructions
                   >3    FCOMP    EQU   $EBB2
                   >4
770F: 4C 76 DD     >5    ]ERR     JMP   GOTMIERR
7712: 4C B2 E5     >6    ]ERR1    JMP   GOSTLERR
                   >7
                   >8    * Handle += instruction for string variables
7715: 68           >9    HNDLESTR PLA              ;Get OP kind off stack
7716: D0 F7        >10           BNE   ]ERR        ;Only ADD operation allowed
7718: B2 A0        >18           LDA   ($A0)
771A: F0 60        >19           BEQ   RET1
771C: 18           >20           CLC
771D: 72 85        >21           ADC   (FORPNT)
771F: B0 F1        >23           BCS   ]ERR1
7721: 20 DD E3     >24           JSR   STRSPA
7724: A5 85        >25           LDA   FORPNT
7726: A4 86        >26           LDY   FORPNT+1
7728: 20 39 77     >27           JSR   NMOVINS
772B: A0 02        >28           LDY   #2
772D: B9 9D 00     >29   ]LOOP    LDA   DSCTMP,Y
7730: 91 85        >30           STA   (FORPNT),Y
7732: 88           >31           DEY
7733: 10 F8        >32           BPL   ]LOOP
7735: A5 A0        >33           LDA   $A0
7737: A4 A1        >34           LDY   $A1
7739: 85 AB        >35   NMOVINS  STA   STRING1
773B: 84 AC        >36           STY   STRING1+1
773D: 4C D4 E5     >37           JMP   MOVINS
                   >38
7740: 29 07        >39   HNDLEINT AND   #7         Integer subtype in A reg.
7742: C9 02        >40           CMP   #2         Correct if 16bits integer
7744: D0 02        >41           BNE   :0
7746: A9 00        >42           LDA   #0
                   >43   * On enclenche NROUT que si 8 ou 16bits
```

```
7748: C9 02    >44   :0       CMP   #2
774A: B0 0D    >45           BCS   :1
774C: 48       >46           PHA
774D: 20 4E 78 >47           JSR   NROUT
7750: 68       >48           PLA
7751: F0 07    >49           BEQ   :2
               >50   * Ensure correct value for 8bits integer
7753: AA       >51           TAX
7754: 20 EA 79 >52           JSR   CONV1628
7757: 8A       >53           TXA
7758: 24       >55           HEX   24         Skip next byte
7759: 3A       >56   :1       DEC
775A: 0A       >61   :2       ASL
775B: 0A       >62           ASL
775C: 85 B4    >63           STA   XSAV
775E: 68       >64           PLA              ;Retrieve ope. index in A reg.
775F: 05 B4    >65           ORA   XSAV
7761: 2C E7 9C >66           BIT   WMODE
7764: 10 02    >67           BPL   *+4
7766: 09 08    >68           ORA   #8
7768: AA       >69           TAX              ;Global operation offset into X
7769: BD 35 96 >70   HNDLEIY  LDA   OFFSTT,X
776C: 48       >71           PHA
776D: BD 25 96 >72           LDA   OFFSTB,X
7770: 48       >73           PHA
7771: A0 01    >74           LDY   #1
7773: 8A       >75           TXA
7774: 29 04    >76           AND   #4
7776: F0 01    >77           BEQ   *+3        Branch iif 16bits int operation
7778: 88       >78           DEY
7779: 18       >79           CLC
777A: B1 85    >80           LDA   (FORPNT),Y
777C: 60       >84   RET1     RTS
               >85
777D: 65 A1    >86   HNDLUIAD ADC   $A1
777F: AA       >87           TAX              ;Low byte in X reg.
7780: B2 85    >89           LDA   (FORPNT)
7782: 65 A0    >93           ADC   $A0
7784: 90 4D    >94           BCC   HNDLEIC
7786: 4C D5 E8 >95   ]ERR     JMP   GOOVFERR
7789: 38       >96   HNDLUIMI SEC
778A: E5 A1    >97           SBC   $A1
778C: AA       >98           TAX              ;Low byte in X reg.
778D: B2 85    >100          LDA   (FORPNT)
778F: E5 A0    >104          SBC   $A0
7791: 90 F3    >105          BCC   ]ERR
7793: B0 3E    >106          BCS   HNDLEIC
7795: 65 A1    >107  HNDLSIAD ADC   $A1        ADD operation
7797: AA       >108          TAX
7798: B2 85    >112          LDA   (FORPNT)
779A: 65 A0    >114          ADC   $A0
779C: 70 E8    >115          BVS   ]ERR
779E: 50 33    >116          BVC   HNDLEIC
77A0: 38       >117  HNDLSIMI SEC
77A1: E5 A1    >118          SBC   $A1
77A3: AA       >119          TAX
77A4: B2 85    >123          LDA   (FORPNT)
```

```
77A6: E5 A0    >125           SBC   $A0
77A8: 70 DC    >126           BVS   ]ERR
77AA: 50 27    >127           BVC   HNDLEIC
77AC: 38       >128  HNDLUIDV SEC
77AD: 20 30 78 >129  HNDLUIMU JSR   LBS49
77B0: 90 05    >130           BCC   :0
77B2: 20 95 79 >131           JSR   USDIV
77B5: 80 03    >133           BRA   *+5
77B7: 20 43 79 >137  :0        JSR   USMUL
77BA: D0 CA    >138           BNE   ]ERR
77BC: F0 10    >139           BEQ   HNDLEIX
77BE: 38       >140  HNDLSIDV SEC
77BF: 20 30 78 >141  HNDLSIMU JSR   LBS49
77C2: B0 05    >142           BCS   :0
77C4: 20 25 79 >143           JSR   SMUL
77C7: 80 03    >145           BRA   *+5
77C9: 20 6A 79 >149  :0        JSR   SDIV
77CC: 70 B8    >150           BVS   ]ERR
77CE: C8       >152  HNDLEIX  INY
77CF: A6 C2    >153           LDX   MPLIER
77D1: A5 C3    >157           LDA   MPLIER+1
77D3: 92 85    >159  HNDLEIC  STA   (FORPNT)
77D5: 8A       >163           TXA                ;Low byte from result
77D6: 91 85    >167           STA   (FORPNT),Y
77D8: A9 80    >168  SETITS   LDA   #$80
77DA: 85 C7    >169           STA   INTTYPSV
77DC: 60       >170           RTS
               >171
77DD: 65 A1    >172  HNDLUBAD ADC   $A1
77DF: 90 4C    >173           BCC   HNDLEBC
77E1: 4C D5 E8 >174  ]ERR     JMP   GOOVFERR
77E4: 65 A1    >175  HNDLSBAD ADC   $A1
77E6: 70 F9    >176           BVS   ]ERR
               >177  ]ERRS    EQU   *-2
77E8: 50 43    >178           BVC   HNDLEBC
77EA: 38       >179  HNDLUBMI SEC
77EB: E5 A1    >180           SBC   $A1
77ED: 90 F2    >181           BCC   ]ERR
77EF: B0 3C    >182           BCS   HNDLEBC
77F1: 38       >183  HNDLSBMI SEC
77F2: E5 A1    >184           SBC   $A1
77F4: 70 F0    >185           BVS   ]ERRS
77F6: 50 35    >186           BVC   HNDLEBC
77F8: 38       >187  HNDLUBMU SEC
77F9: 85 C2    >188  HNDLSBMU STA   MPLIER
77FB: A5 A1    >189           LDA   $A1
77FD: 85 C0    >190           STA   MCAND
77FF: 90 09    >191           BCC   :0
7801: 20 A8 78 >192           JSR   USMUL8
7804: D0 DB    >193           BNE   ]ERR
7806: A5 C2    >194           LDA   MPLIER
7808: 70 23    >195           BVS   HNDLEBC    Always (see USMUL8 routine)
780A: 20 8B 78 >196  :0        JSR   SMUL8
780D: 70 D7    >197           BVS   ]ERRS
780F: A5 C2    >198           LDA   MPLIER
7811: 50 1A    >199           BVC   HNDLEBC    Always
               >200
```

```
7813: 4C E1 EA >201  ]ERR     JMP   GODVZERR
7816: 38       >202  HNDLUBDV SEC
7817: 85 C2    >203  HNDLSBDV STA   DIVEND
7819: A5 A1    >204           LDA   $A1
781B: F0 F6    >205           BEQ   ]ERR
781D: 85 C0    >206           STA   DIVSOR
781F: 90 05    >207           BCC   :0
7821: 20 F0 78 >208           JSR   USDIV8
7824: 70 07    >209           BVS   HNDLEBC    Always (see USDIV8 routine)
7826: 20 C3 78 >210  :0       JSR   SDIV8
7829: 70 BB    >211           BVS   ]ERRS
782B: A5 C2    >212           LDA   DIVEND
782D: 92 85    >214  HNDLEBC  STA   (FORPNT)
782F: 60       >219  ]RET     RTS
               >220
7830: 08       >221  LBS49    PHP
7831: 85 C2    >222           STA   MPLIER
7833: B2 85    >224           LDA   (FORPNT)
7835: 85 C3    >228           STA   MPLIER+1
7837: A5 A0    >229           LDA   $A0
7839: 85 C1    >230           STA   MCAND+1
783B: A5 A1    >231           LDA   $A1
783D: 85 C0    >232           STA   MCAND
783F: 28       >233           PLP
7840: 60       >234           RTS
               >235
7841: 4C 99 E1 >236  ]ERR     JMP   $E199
7844: 20 F2 EB >238  ]LOOP    JSR   QINT
7847: 18       >239           CLC
7848: 60       >240           RTS
               >242  * LBS03 is called with carry flag as input parm
               >243  * Carry set: for catering with negative STEP values
               >244  * while unsigned arithmetic is active.
7849: 08       >245  LBS03    PHP
784A: 20 3B 85 >246           JSR   NFRMNUM
784D: 24       >247           HEX   24
784E: 08       >248  NROUT    PHP
784F: 20 72 EB >249           JSR   $EB72      Arrondit FAC
7852: 28       >250           PLP
7853: A5 9D    >251  NEWAYINT LDA   FAC
7855: 2C E7 9C >252           BIT   WMODE
7858: 30 0A    >253           BMI   :1
785A: C9 90    >254           CMP   #$90
785C: 90 E6    >255           BCC   ]LOOP
785E: 20 86 8E >256           JSR   GN32768
7861: 4C 16 E1 >257           JMP   $E116
               >258  * Unsigned mode
7864: 24 A2    >259  :1       BIT   FACSIGN
7866: B0 17    >260           BCS   :3
7868: 30 D7    >261           BMI   ]ERR
786A: C9 91    >262  :2       CMP   #$91
786C: 90 D6    >264           BCC   ]LOOP
786E: 20 8B 8E >268           JSR   GP32768
7871: 20 B2 EB >269           JSR   FCOMP
7874: A8       >270           TAY
7875: 30 CD    >272           BMI   ]LOOP      A = -1 so FAC < 32768
7877: 20 90 8E >276           JSR   GN65536
```

```
787A: 20 BE E7 >277           JSR   FADD
787D: 80 C5    >279           BRA   ]LOOP
787F: 10 E9    >285  :3       BPL   :2
7881: 20 D0 EE >286           JSR   NEGOP
7884: A5 9D    >287           LDA   FAC
7886: 20 6A 78 >288           JSR   :2
7889: 38       >289           SEC
788A: 60       >290           RTS
               >291
               >292  * Signed 8bits multiplication: result in 8bits
               >293  * with possible overflow exception
               >294  * MCAND and MPLIER set upon entry
               >295  * Result in MPLIER
               >296  * Credits: Randy Hyde
788B: A5 C0    >297  SMUL8    LDA   MCAND
788D: 45 C2    >298           EOR   MPLIER
788F: 48       >299           PHA                ;Bit N set if signs differ
7890: 20 0B 79 >300           JSR   ZPRT8
7893: 20 A8 78 >301           JSR   USMUL8
7896: FA       >302           PLX
7897: 98       >303           TYA
7898: D0 0D    >304           BNE   :0
789A: A5 C2    >305           LDA   MPLIER
789C: 30 09    >306           BMI   :0
789E: 8A       >307           TXA
789F: 10 05    >308           BPL   :1
78A1: A2 C2    >309           LDX   #MPLIER
78A3: 20 1A 79 >310           JSR   NEG8
78A6: B8       >311  :1       CLV
78A7: 60       >312  :0       RTS
               >313
78A8: A0 08    >314  USMUL8   LDY   #8
78AA: A5 C2    >315  ]LOOP    LDA   MPLIER     Get lsb of MPLIER
78AC: 4A       >316           LSR               ; into C
78AD: 90 07    >317           BCC   :4
78AF: 18       >318           CLC
78B0: A5 BE    >319           LDA   PARTIAL
78B2: 65 C0    >320           ADC   MCAND
78B4: 85 BE    >321           STA   PARTIAL
               >322  * Shift result into MPLIER
78B6: 66 BE    >323  :4       ROR   PARTIAL
78B8: 66 C2    >324           ROR   MPLIER
78BA: 88       >325           DEY               ;All MPLIER 8 bits
78BB: D0 ED    >326           BNE   ]LOOP        have been processed?
78BD: 2C 2F 78 >327           BIT   ]RET        Bit V set..
78C0: A4 BE    >328           LDY   PARTIAL
78C2: 60       >329  ]RET     RTS
               >330
               >331  * Signed 8bits integer divide routine
               >332  * with possible overflow and divide by zero exceptions
               >333  * DIVEND and DIVSOR set upon entry
               >334  * Result in DIVEND
               >335  * Credits: Randy Hyde
78C3: A5 C0    >336  SDIV8    LDA   DIVSOR
78C5: 49 80    >337           EOR   #$80
78C7: D0 0D    >338           BNE   :1
               >339  * On traite le cas ou le diviseur est -128
```

```
                >340  * Dans ce cas la si DIVEND vaut aussi -128, alors
                >341  * retourne 1 sinon 0
78C9: A8        >342          TAY
78CA: AA        >343          TAX             ;X forced to zero
78CB: A5 C2     >344          LDA   DIVEND
78CD: C9 80     >345          CMP   #$80
78CF: D0 01     >346          BNE   *+3
78D1: E8        >347          INX
78D2: 86 C2     >348          STX   DIVEND
78D4: D0 EC     >349          BNE   ]RET
78D6: A5 C0     >350  :1      LDA   DIVSOR
78D8: 45 C2     >351  :2      EOR   DIVEND
78DA: 48        >352          PHA             ;Sign bit on stack
78DB: 20 0B 79  >353          JSR   ZPRT8     ;Absolute value for operands
78DE: 20 F0 78  >354          JSR   USDIV8
78E1: 1A        >358          INC
78E2: F0 0A     >360          BEQ   :3        Keep V set and exit
78E4: 68        >361          PLA             ;Get back sign
78E5: 10 05     >362          BPL   *+7       No need to get result opposite
78E7: A2 C2     >363          LDX   #DIVEND
78E9: 20 1A 79  >364          JSR   NEG8
                >365  * Exit with V clear
78EC: B8        >366          CLV
78ED: 60        >367          RTS
78EE: 68        >368  :3      PLA
78EF: 60        >369  ]RET    RTS
                >370
78F0: A0 08     >371  USDIV8  LDY   #8
78F2: 06 C2     >372  ]LOOP   ASL   DIVEND
78F4: 26 BE     >373          ROL   PARTIAL
78F6: 38        >374          SEC
78F7: A5 BE     >375          LDA   PARTIAL
78F9: E5 C0     >376          SBC   DIVSOR
78FB: AA        >377          TAX
78FC: 90 04     >378          BCC   :3
78FE: 86 BE     >379          STX   PARTIAL
7900: E6 C2     >380          INC   DIVEND
7902: 88        >381  :3      DEY
7903: D0 ED     >382          BNE   ]LOOP
7905: 2C EF 78  >383          BIT   ]RET      V set by default
7908: A5 C2     >384          LDA   DIVEND
790A: 60        >385          RTS
                >386
790B: A0 00     >387  ZPRT8   LDY   #0
790D: 84 BE     >388          STY   PARTIAL
790F: A2 C0     >389          LDX   #MCAND
7911: 20 16 79  >390          JSR   ABSOL8
7914: A2 C2     >391          LDX   #MPLIER
7916: B5 00     >392  ABSOL8  LDA   0,X
7918: 10 D5     >393          BPL   ]RET
791A: 98        >394  NEG8    TYA
791B: 38        >395          SEC
791C: F5 00     >396          SBC   0,X
791E: 95 00     >397          STA   0,X
7920: 60        >398  ]RET    RTS
                >399
                >400  * Signed 16bits multiplication: result in 16bits
```

```
                >401  * with possible overflow exception
                >402  * MCAND and MPLIER set upon entry
                >403  * Result in MPLIER
                >404  * Credits: Randy Hyde
7921: 2C 20 79  >405  ]LOOP    BIT   ]RET
7924: 60        >406          RTS
7925: A5 C1     >407  SMUL     LDA   MCAND+1
7927: 45 C3     >408          EOR   MPLIER+1
7929: 48        >409          PHA               ;BitN set if signs differ
792A: 20 CA 79  >410          JSR   ZEROPRT     Get absolute values of operands
792D: 20 43 79  >411          JSR   USMUL
7930: A8        >412          TAY
7931: FA        >413          PLX
7932: 98        >414          TYA
7933: D0 EC     >415          BNE   ]LOOP
7935: A5 C3     >416          LDA   MPLIER+1
7937: 30 E8     >417          BMI   ]LOOP
7939: 8A        >418          TXA
793A: 10 05     >419          BPL   :8
793C: A2 C2     >420          LDX   #MPLIER
793E: 20 DB 79  >421          JSR   NEGATE
7941: B8        >422  :8       CLV               ;reset bit V to zero
7942: 60        >423  ]RET     RTS
                >424
7943: A0 10     >425  USMUL    LDY   #16
7945: A5 C2     >426  ]LOOP    LDA   MPLIER      Get lsb of MPLIER
7947: 4A        >427          LSR               ; into C
7948: 90 0D     >428          BCC   :4
794A: 18        >429          CLC
794B: A5 BE     >430          LDA   PARTIAL
794D: 65 C0     >431          ADC   MCAND
794F: 85 BE     >432          STA   PARTIAL
7951: A5 BF     >433          LDA   PARTIAL+1
7953: 65 C1     >434          ADC   MCAND+1
7955: 85 BF     >435          STA   PARTIAL+1
                >436  * Shift result into MPLIER
7957: 66 BF     >437  :4       ROR   PARTIAL+1
7959: 66 BE     >438          ROR   PARTIAL
795B: 66 C3     >439          ROR   MPLIER+1
795D: 66 C2     >440          ROR   MPLIER
795F: 88        >441          DEY               ;All MPLIER 16 bits
7960: D0 E3     >442          BNE   ]LOOP        have been processed?
7962: A5 BE     >443          LDA   PARTIAL
7964: 05 BF     >444          ORA   PARTIAL+1
7966: 60        >445  ]RET     RTS
                >446
7967: 4C E1 EA  >447  DVZERROR JMP   GODVZERR
                >448  * Signed 16bits integer divide routine
796A: A5 C1     >449  SDIV     LDA   DIVSOR+1
796C: 05 C0     >450          ORA   DIVSOR
796E: F0 F7     >451          BEQ   DVZERROR
7970: A5 C1     >452          LDA   DIVSOR+1
7972: C9 80     >453          CMP   #>$8000
7974: D0 19     >454          BNE   :2
7976: A5 C0     >455          LDA   DIVSOR
7978: D0 13     >456          BNE   :1
                >457  * On traite le cas ou le diviseur est -32768
```

```
                >458  * Dans ce cas la si DIVEND vaut aussi -32768, alors
                >459  * retourne 1 sinon 0
797A: A8        >460          TAY
797B: AA        >461          TAX             ;X forced to zero
797C: C5 C2     >462          CMP   DIVEND
797E: D0 07     >463          BNE   :0
7980: A5 C3     >464          LDA   DIVEND+1
7982: C9 80     >465          CMP   #>$8000
7984: D0 01     >466          BNE   :0
7986: E8        >467          INX
7987: 86 C2     >468  :0      STX   DIVEND
7989: 84 C3     >469          STY   DIVEND+1
798B: D0 39     >470          BNE   NRET      Always
798D: A5 C1     >471  :1      LDA   DIVSOR+1
798F: 45 C3     >472  :2      EOR   DIVEND+1
7991: 48        >473          PHA             ;Sign bit on stack
7992: 20 CA 79  >474          JSR   ZEROPRT   ;Absolute value for operands
7995: A0 10     >475  USDIV   LDY   #16
7997: 06 C2     >476  ]LOOP   ASL   DIVEND
7999: 26 C3     >477          ROL   DIVEND+1
799B: 26 BE     >478          ROL   PARTIAL
799D: 26 BF     >479          ROL   PARTIAL+1
799F: 38        >480          SEC
79A0: A5 BE     >481          LDA   PARTIAL
79A2: E5 C0     >482          SBC   DIVSOR
79A4: AA        >483          TAX
79A5: A5 BF     >484          LDA   PARTIAL+1
79A7: E5 C1     >485          SBC   DIVSOR+1
79A9: 90 06     >486          BCC   :3
79AB: 86 BE     >487          STX   PARTIAL
79AD: 85 BF     >488          STA   PARTIAL+1
79AF: E6 C2     >489          INC   DIVEND
79B1: 88        >490  :3       DEY
79B2: D0 E3     >491          BNE   ]LOOP
79B4: 2C C9 79  >492          BIT   ARET+1    V set by default
79B7: A5 C2     >493          LDA   DIVEND
79B9: 25 C3     >494          AND   DIVEND+1
79BB: 1A        >498          INC
79BC: F0 0A     >500          BEQ   ARET      Keep V set and exit
79BE: 68        >501          PLA             ;Get back sign
79BF: 10 05     >502          BPL   NRET      No need to get result opposite
79C1: A2 C2     >503          LDX   #DIVEND
79C3: 20 DB 79  >504          JSR   NEGATE
                >505  * Exit with V clear
79C6: B8        >506  NRET    CLV
79C7: 70        >507          HEX   70        Skip next byte
79C8: 68        >508  ARET    PLA
79C9: 60        >509  ]RET    RTS
                >510
                >511  * Zero partial and fall into ABSOPND
79CA: A0 00     >512  ZEROPRT LDY   #0
79CC: 84 BE     >513          STY   PARTIAL
79CE: 84 BF     >514          STY   PARTIAL+1
79D0: A2 C0     >515          LDX   #MCAND
79D2: 20 D7 79  >516          JSR   ABSOLUTE
79D5: A2 C2     >517          LDX   #MPLIER   ;Fall into ABSOLUTE
                >518  * Compute absolute value of integer pointed to by X
```

```
                >519  * in ZP
79D7: B5 01     >520  ABSOLUTE LDA   1,X
79D9: 10 EE     >521           BPL   ]RET       No need
79DB: 38        >522  NEGATE   SEC
79DC: 98        >523           TYA               ;Y set to 0 upon entry
79DD: F5 00     >524           SBC   0,X
79DF: 95 00     >525           STA   0,X
79E1: 98        >526           TYA
79E2: F5 01     >527           SBC   1,X
79E4: 95 01     >528           STA   1,X
79E6: 60        >529  ]RET     RTS
                >530
                >531  * Conversion from 16bits to 8bits with provision for
                >532  * ILLEGAL QUANTITY..
79E7: 4C 99 E1  >533  ]ERR     JMP   GOIQERR
79EA: A5 A0     >534  CONV1628 LDA   FAC+3       High byte
79EC: 2C E7 9C  >535           BIT   WMODE
79EF: 30 0B     >536           BMI   :0
79F1: A8        >537           TAY
79F2: C8        >538           INY
79F3: C0 02     >539           CPY   #2          Must be either -1 or 0
79F5: B0 F0     >540           BCS   ]ERR         in unsigned mode
79F7: 45 A1     >541           EOR   FAC+4       b7 of low byte should be
79F9: 30 EC     >542           BMI   ]ERR        set accordingly.
79FB: 60        >543           RTS
79FC: D0 E9     >544  :0       BNE   ]ERR        Must be zero if unsigned mode
79FE: 60        >545           RTS
79FF: 4C 99 E1  >546           JMP   GOIQERR
                 550  * New processing for variable lookup
                 551           PUT   PEERNPTRGET
                >1    MKNV     EQU   $E09C       Make new variable (ROM routine)
                >2    SETVYA   EQU   $E0DE       Set LOWTR and Y,A if var. found
                >3
7A02: A9 40     >4    NGETARPT LDA   #$40        $40: only look for arrays
7A04: 85 14     >5             STA   SUBFLG
                >6    * This routine is the new PTRGET routine from PEERSOFT
                >7    NPTRGTX
7A06: 64 10     >12            STZ   DIMFLG
                >14   NPTRGET
                >15   * Upon exit from the above routine, the X reg will
                >16   * contain the value X had upon call to CHRGOT (here zero)
7A08: 20 32 76  >17            JSR   COMRST
                >18   * First variable name character must be alphabetic
7A0B: 20 70 7E  >19            JSR   MISLETC
                >20
7A0E: 64 11     >28   NPTRGET1 STZ   VALTYP
7A10: 64 12     >29            STZ   INTTYP
7A12: 64 82     >30            STZ   VARNAM+1    Default zero for 2nd name char.
7A14: 64 BF     >31            STZ   AUXBANK
7A16: 85 81     >33            STA   VARNAM
7A18: 20 2A 76  >34            JSR   RST100
7A1B: 90 05     >35            BCC   GTLT        Branch if numeric digit
7A1D: 20 7D E0  >36            JSR   ISLETC
7A20: 90 1A     >37            BCC   EXPLIC?     Branch if not alpha character
7A22: AA        >38   GTLT     TAX               ;2nd character in X
7A23: 86 82     >39            STX   VARNAM+1     and into VARNAM+1
                >40   * Skip subsequent alphanumeric characters
```

```
7A25: 20 2A 76 >41   ]LOOP    JSR   RST100
7A28: 90 FB    >42           BCC   ]LOOP      branch if numeric
7A2A: 20 7D E0 >43           JSR   ISLETC
7A2D: B0 F6    >44           BCS   ]LOOP      branch if alphabetic
7A2F: 90 0B    >45           BCC   EXPLIC?    Always
7A31: 4C C9 DE >46   BADNAM   JMP   SYNERR
               >47   * Code run as no explicit type specifier found, get the
               >48   * default type specifier according to 1st varname char.
7A34: 20 09 82 >49   SCDCH2   JSR   DECTPTR
7A37: A6 81    >50           LDX   VARNAM
7A39: BD 55 9B >51           LDA   TYPLET-´A´,X
               >52   * Fall into implicit (2nd pass to EXPLIC?)
7A3C: 20 FE 81 >53   EXPLIC?  JSR   XFROMMOT   Get index from character
               >54   * No explicit type specifier found, so try implicit
               >55   * type specifier (cannot fail)
7A3F: D0 F3    >56           BNE   SCDCH2     Branch if no type spec. found
7A41: BD 8A 9B >58           LDA   TVTVAL,X
7A44: 85 11    >59           STA   VALTYP
7A46: BD 86 9B >60           LDA   TITVAL,X
7A49: 85 12    >61           STA   INTTYP
7A4B: BD 8E 9B >62           LDA   TVNORA,X
7A4E: 04 81    >63           TSB   VARNAM
7A50: BD 92 9B >64           LDA   TVN1ORA,X
7A53: 04 82    >65           TSB   VARNAM+1
7A55: E0 02    >66           CPX   #2         FP or string
7A57: 90 04    >67           BCC   :6
7A59: A5 14    >68           LDA   SUBFLG
7A5B: 30 D4    >69           BMI   BADNAM
7A5D: 20 2A 76 >70   :6       JSR   RST100     Get next character
7A60: 38       >71           SEC
7A61: 05 14    >72           ORA   SUBFLG
7A63: E9 28    >73           SBC   #´(´
7A65: D0 03    >74           BNE   :8
7A67: 4C 2F 7B >75   :7       JMP   NARRAY
7A6A: 24 14    >76   :8       BIT   SUBFLG
7A6C: 30 02    >77           BMI   :9
7A6E: 70 F7    >78           BVS   :7
               >79   :9       DO    KOPT-K6502
7A70: 64 14    >80           STZ   SUBFLG
7A72: AE 83 99 >85   NPTRGL90 LDX   SNCCH
7A75: F0 05    >86           BEQ   :90
7A77: 20 E2 7A >87           JSR   SLKCACH
7A7A: D0 63    >88           BNE   NAMFOUND   Found cache entry if Zbit clear
               >89   :90      DO    KOPT16
7A7C: A6 69    >95           LDX   VARTAB
7A7E: A5 6A    >96           LDA   VARTAB+1
7A80: 85 9C    >101  ]LOOP    STA   LOWTR+1
7A82: 86 9B    >102  ]LOOP1   STX   LOWTR
7A84: E4 6B    >107          CPX   ARYTAB
7A86: E5 6C    >108          SBC   ARYTAB+1
7A88: B0 26    >110          BCS   NAMNTFND
7A8A: B2 9B    >115          LDA   (LOWTR)
7A8C: 45 81    >117          EOR   VARNAM
7A8E: D0 14    >118          BNE   :1
7A90: A0 01    >121          LDY   #1
7A92: B1 9B    >125          LDA   (LOWTR),Y
7A94: 45 82    >126          EOR   VARNAM+1
```

```
7A96: D0 0C    >127           BNE   :1
7A98: A5 12    >131           LDA   INTTYP
7A9A: 10 43    >132           BPL   NAMFOUND
7A9C: A0 06    >133           LDY   #6
7A9E: B1 9B    >134           LDA   (LOWTR),Y
7AA0: 45 12    >135           EOR   INTTYP
7AA2: F0 3B    >136           BEQ   NAMFOUND
               >140  * Name not yet found: look for next variable in memory
7AA4: A5 9B    >141  :1       LDA   LOWTR
7AA6: 69 07    >147           ADC   #7         Carry already clear
7AA8: AA       >148           TAX
7AA9: A5 9C    >149           LDA   LOWTR+1
7AAB: 90 D5    >150           BCC   ]LOOP1
7AAD: 1A       >152           INC
7AAE: D0 D0    >153           BNE   ]LOOP      Always
               >159
7AB0: BA       >168  NAMNTFND TSX
7AB1: BD 01 01 >169           LDA   STACK+1,X
7AB4: C9 99    >170           CMP   #RFFVL
7AB6: D0 0A    >171           BNE   :0
7AB8: BD 02 01 >172           LDA   STACK+2,X
7ABB: C9 85    >173           CMP   #>RFFVL
7ABD: D0 03    >174           BNE   :0
7ABF: 4C 95 E0 >176           JMP   $E095      Return 0 constant
               >177  * Make new variable
7AC2: 18       >178  :0       CLC
7AC3: A5 6D    >185           LDA   STREND
7AC5: A4 6E    >186           LDY   STREND+1
7AC7: 69 07    >187           ADC   #7
7AC9: 90 01    >188           BCC   *+3
7ACB: C8       >189           INY
7ACC: 20 13 7B >190           JSR   NREASON
7ACF: 20 9C E0 >192           JSR   MKNV       Make new variable (ROM routine)
7AD2: A5 12    >193           LDA   INTTYP     FP or string?
7AD4: 10 06    >194           BPL   :1         Yes
7AD6: A0 06    >195           LDY   #6
7AD8: 91 9B    >196           STA   (LOWTR),Y
7ADA: A4 84    >197           LDY   VARPNT+1
7ADC: A5 83    >198  :1       LDA   VARPNT
7ADE: 60       >199           RTS
               >200
               >201  NAMFOUND
7ADF: 4C DE E0 >207           JMP   SETVYA
               >208
               >209  * Cache mechanism for simple variables
               >210  SCTR     EQU   LOWTR
7AE2: A4 82    >238  SLKCACH  LDY   VARNAM+1
7AE4: A5 81    >239           LDA   VARNAM
7AE6: 86 9B    >240           STX   SCTR
7AE8: A2 00    >241           LDX   #0
7AEA: DD 84 99 >242  ]LOOP    CMP   SVN,X
7AED: D0 0F    >243           BNE   :0
7AEF: 98       >244           TYA
7AF0: DD 88 99 >245           CMP   SVNP1,X
7AF3: D0 07    >246           BNE   :2
7AF5: A5 12    >247           LDA   INTTYP
7AF7: DD 8C 99 >248           CMP   SIT,X
```

```
7AFA: F0 08    >249           BEQ   :1
7AFC: A5 81    >250  :2       LDA   VARNAM
7AFE: E8       >251  :0       INX
7AFF: E4 9B    >252           CPX   SCTR
7B01: D0 E7    >253           BNE   ]LOOP
7B03: 60       >255           RTS
               >256
7B04: BD 90 99 >257  :1       LDA   SLTR,X
7B07: 85 9B    >258           STA   LOWTR
7B09: BD 94 99 >263           LDA   SLTRP1,X
7B0C: 85 9C    >264           STA   LOWTR+1
7B0E: 8A       >266           TXA
7B0F: 60       >267           RTS
               >268
7B10: 4C 10 D4 >269  ]ERR     JMP   MEMERR
               >287  * Pour le 65(C)02, Y,A nouveau STREND
7B13: C4 70    >288  NREASON  CPY   FRETOP+1
7B15: 90 17    >289           BCC   :0
7B17: D0 04    >290           BNE   :1
7B19: C5 6F    >291           CMP   FRETOP
7B1B: 90 11    >292           BCC   :0
7B1D: 48       >293  :1       PHA
7B1E: 5A       >294           PHY
7B1F: 20 D5 9C >295           JSR   VGARBAG
7B22: 7A       >296           PLY
7B23: 68       >297           PLA
7B24: C4 70    >298           CPY   FRETOP+1
7B26: 90 06    >299           BCC   :0
7B28: D0 E6    >300           BNE   ]ERR
7B2A: C5 6F    >301           CMP   FRETOP
7B2C: B0 E2    >302           BCS   ]ERR
7B2E: 60       >303  :0       RTS
                552  * New processing for array processing
                553           PUT   PEERNARRAY
               >1    * Module handling the new array processing strategy
               >2    ERR_BSCR =     $6B
               >3    ERR_RDIM =     $78
               >4    ERR_SYNT =     $10
               >5
               >6    NUMDIM   EQU   $0F
               >7    RESULT   EQU   $62
               >8    STACK    EQU   $0100
               >9    SUBERR   EQU   $E196      Raise a BAD SUBSCRIPT error
               >10   MEMERR   EQU   $D410
               >11   REASON   EQU   $D3E3
               >12   GETARY   EQU   $E0ED
               >13   GETARY2  EQU   $E0EF      Compute addr. of 1st elm value
               >14   QINT     EQU   $EBF2
               >15
               >16   * MULTPLSS multiplies (STRNG2) by ((LOWTR),Y) leaving
               >17   * result in A,X. Hi byte also in Y
               >18   MULTPLSS EQU   $E2AD
               >19   MULTPLY1 EQU   $E2B6
               >20
7B2F: A5 14    >28   NARRAY   LDA   SUBFLG
7B31: D0 4B    >30           BNE   NARRGL91
7B33: A5 10    >36           LDA   DIMFLG
```

```
7B35: 48       >37            PHA
7B36: A5 12    >38            LDA   INTTYP
7B38: 48       >39            PHA
7B39: A5 11    >40            LDA   VALTYP
7B3B: 48       >41            PHA
7B3C: A0 00    >43            LDY   #0
               >44   ]LOOP    MPHY
7B3E: 5A       >44            PHY
7B3F: A5 82    >51            LDA   VARNAM+1
7B41: 48       >52            PHA
7B42: A5 81    >53            LDA   VARNAM
7B44: 48       >54            PHA
7B45: 20 79 7D >56            JSR   NMAKINT
7B48: 68       >63            PLA
7B49: 85 81    >64            STA   VARNAM     Restore array name
7B4B: 68       >67            PLA
7B4C: 85 82    >68            STA   VARNAM+1
7B4E: 7A       >70            PLY
               >71   * Code below would transform the stack area
               >72   * from
               >73   *  DIMFLG
               >74   *  INTTYP
               >75   *  VALTYP
               >76   * SPtr ->
               >77   * to
               >78   *  (FAC+3)
               >79   *  (FAC+4)
               >80   *  DIMFLG
               >81   *  INTTYP
               >82   *  VALTYP
               >83   * SPtr ->
7B4F: BA       >98            TSX
7B50: BD 02 01 >99            LDA   STACK+2,X  Get INTTYP
7B53: 48       >100           PHA
7B54: BD 01 01 >101           LDA   STACK+1,X  Get VALTYP
7B57: 48       >102           PHA
7B58: BD 03 01 >103           LDA   STACK+3,X  Get DIMFLG
7B5B: 9D 01 01 >104           STA   STACK+1,X  In place of original VALTYP
7B5E: A5 A0    >105           LDA   FAC+3
7B60: 9D 03 01 >106           STA   STACK+3,X  In place of original DIMFLG
7B63: A5 A1    >107           LDA   FAC+4
7B65: 9D 02 01 >108           STA   STACK+2,X  In place of original INTTYP
               >110   * Now the stack frame looks like
               >111   *  FAC+4
               >112   *  FAC+3
               >113   *  DIMFLG
               >114   *  INTTYP
               >115   *  VALTYP
               >116   * SPtr ->
7B68: C8       >117           INY
7B69: 20 32 76 >118           JSR   RST102
7B6C: C9 2C    >119           CMP   #´,´
7B6E: F0 CE    >120           BEQ   ]LOOP
7B70: 84 0F    >121           STY   NUMDIM
7B72: 20 A3 86 >122           JSR   NCHKCLS
7B75: 68       >123           PLA
7B76: 85 11    >124           STA   VALTYP
```

```
7B78: 68       >125           PLA
7B79: 85 12    >126           STA   INTTYP
7B7B: 68       >127           PLA
7B7C: 85 10    >128           STA   DIMFLG
               >129
               >130
7B7E: AE 98 99 >131  NARRGL91 LDX   ANCCH
7B81: F0 05    >132           BEQ   :20
7B83: 20 9D 7D >133           JSR   ALKCACH
7B86: D0 3D    >134           BNE   USEOLDAR
7B88: A5 6C    >145  :20      LDA   ARYTAB+1
7B8A: A6 6B    >146           LDX   ARYTAB
7B8C: 86 9B    >147  ]LOOP    STX   LOWTR
7B8E: 85 9C    >148           STA   LOWTR+1
7B90: E4 6D    >149           CPX   STREND
7B92: E5 6E    >150           SBC   STREND+1
7B94: B0 2C    >152           BCS   GNARRAY
7B96: B2 9B    >154           LDA   (LOWTR)
7B98: 45 81    >159           EOR   VARNAM
7B9A: D0 18    >160           BNE   :5
7B9C: A0 01    >167           LDY   #1
7B9E: B1 9B    >171           LDA   (LOWTR),Y
7BA0: 45 82    >172           EOR   VARNAM+1
7BA2: D0 10    >173           BNE   :5
7BA4: A6 12    >175           LDX   INTTYP
7BA6: 10 1D    >176           BPL   USEOLDAR   If FP or string array
7BA8: 20 95 7D >177           JSR   CNVT1
7BAB: A0 04    >178           LDY   #4
7BAD: 51 9B    >179           EOR   (LOWTR),Y
7BAF: 29 C0    >180           AND   #$C0       only test b6 and b7
7BB1: F0 12    >181           BEQ   USEOLDAR
7BB3: 18       >189           CLC
               >190  :5
7BB4: A0 02    >192           LDY   #2
7BB6: B1 9B    >194           LDA   (LOWTR),Y
7BB8: 65 9B    >195           ADC   LOWTR
7BBA: AA       >197           TAX
7BBB: C8       >198           INY
7BBC: B1 9B    >199           LDA   (LOWTR),Y
7BBE: 65 9C    >200           ADC   LOWTR+1
7BC0: 90 CA    >202           BCC   ]LOOP      Always
               >203
               >204  GNARRAY
7BC2: 4C 33 7C >209           JMP   MKNARRAY
               >210
7BC5: A5 10    >211  USEOLDAR LDA   DIMFLG     Called from the DIM stmt.?
7BC7: D0 65    >212           BNE   RDIMERR
7BC9: A5 14    >213           LDA   SUBFLG     Subscripts given?
7BCB: F0 02    >214           BEQ   :1         Yes
7BCD: 38       >215           SEC              ;No: just return "array found"
7BCE: 60       >216           RTS
               >217  * Set ARYPNT to 1st elm. base addr
7BCF: A0 04    >218  :1       LDY   #4
7BD1: B1 9B    >219           LDA   (LOWTR),Y
7BD3: 29 07    >220           AND   #7
7BD5: AA       >221           TAX
7BD6: 20 EF E0 >222           JSR   GETARY2
```

```
7BD9: A5 0F    >223          LDA   NUMDIM
7BDB: C9 01    >224          CMP   #1
7BDD: F0 07    >225          BEQ   :3
7BDF: E4 0F    >226          CPX   NUMDIM
7BE1: D0 45    >227          BNE   SUBSERR
7BE3: 4C 13 7D >228          JMP   NFAEP
               >229
               >230  * Il s´agit de traiter de la reference unidimensionnelle
               >231  * sur un tableau potentiellement multi-dimensions
               >232  * Multiplier l´indice tire dans la pile par le elm size
               >233  * et comparer par rapport a l´offset du tableau (corrige
               >234  * de la taille du header).
7BE6: 68       >235  :3       PLA
7BE7: 85 AD    >236          STA   STRNG2
7BE9: 68       >237          PLA
7BEA: 85 AE    >238          STA   STRNG2+1
7BEC: 20 66 7D >239          JSR   KWELMSIZ
7BEF: 86 64    >240          STX   RESULT+2
7BF1: A9 00    >241          LDA   #0
7BF3: 20 B6 E2 >242          JSR   MULTPLY1
7BF6: 86 AD    >243          STX   STRNG2
7BF8: 84 AE    >244          STY   STRNG2+1
7BFA: A0 04    >245          LDY   #4
7BFC: B1 9B    >246          LDA   (LOWTR),Y  # of dimensions
7BFE: 29 07    >247          AND   #7         Mask out new Peersoft bits
7C00: 0A       >248          ASL              ;2 bytes per dimension
7C01: 69 05    >249          ADC   #5         Carry clear
               >250  * Add this to element offset from base address
7C03: 65 AD    >251          ADC   STRNG2
7C05: A6 AE    >252          LDX   STRNG2+1
7C07: 90 01    >253          BCC   :4
7C09: E8       >254          INX
7C0A: A0 02    >255  :4       LDY   #2
7C0C: D1 9B    >256          CMP   (LOWTR),Y
7C0E: 85 83    >257          STA   VARPNT
7C10: C8       >258          INY
7C11: 8A       >259          TXA
7C12: F1 9B    >260          SBC   (LOWTR),Y
7C14: B0 12    >261          BCS   SUBSERR
7C16: 86 84    >262          STX   VARPNT+1
7C18: A5 9B    >263          LDA   LOWTR
7C1A: 65 83    >264          ADC   VARPNT
7C1C: 85 83    >265          STA   VARPNT
7C1E: A5 84    >266          LDA   VARPNT+1
7C20: 65 9C    >267          ADC   LOWTR+1
7C22: 85 84    >268          STA   VARPNT+1
7C24: A8       >269          TAY
7C25: A5 83    >270          LDA   VARPNT
7C27: 60       >271          RTS
               >272
7C28: A2 6B    >273  SUBSERR  LDX   #ERR_BSCR
7C2A: 2C       >274          HEX   2C         Skip next two bytes
7C2B: A2 10    >275  SNERR    LDX   #ERR_SYNT
7C2D: 2C       >276          HEX   2C
7C2E: A2 78    >277  RDIMERR  LDX   #ERR_RDIM
7C30: 4C 12 D4 >278          JMP   $D412
               >279
```

```
7C33: A5 14    >280  MKNARRAY LDA   SUBFLG
7C35: F0 03    >281          BEQ   :0
7C37: 4C DC E1 >282          JMP   $E1DC      Raise OUT OF DATA error
7C3A: 20 ED E0 >283  :0      JSR   GETARY     Address 1st elm in ARYPNT&Y,A
7C3D: 20 66 7D >284          JSR   KWELMSIZ
7C40: 86 AD    >285          STX   STRNG2
7C42: 64 BF    >287          STZ   AUXBANK
7C44: A5 10    >292          LDA   DIMFLG
7C46: F0 03    >293          BEQ   :1
7C48: 20 F1 7D >294          JSR   ISAUXMEM
7C4B: A5 94    >302  :1      LDA   ARYPNT
7C4D: 20 13 7B >304          JSR   NREASON    Ensure enough memory for array
7C50: A5 81    >305          LDA   VARNAM
7C52: 64 AE    >307          STZ   STRNG2+1
7C54: 92 9B    >308          STA   (LOWTR)
7C56: A0 01    >309          LDY   #1
7C58: A5 82    >316          LDA   VARNAM+1
7C5A: 91 9B    >317          STA   (LOWTR),Y
7C5C: A0 04    >318          LDY   #4
7C5E: A5 12    >319          LDA   INTTYP
7C60: F0 04    >320          BEQ   :2
7C62: AA       >321          TAX
7C63: 20 95 7D >322          JSR   CNVT1
7C66: 05 0F    >323  :2      ORA   NUMDIM
7C68: A6 BF    >324          LDX   AUXBANK
7C6A: 85 BF    >325          STA   AUXBANK
7C6C: 8A       >326          TXA
7C6D: 0A       >327          ASL
7C6E: 0A       >328          ASL
7C6F: 0A       >329          ASL
7C70: 05 BF    >330          ORA   AUXBANK
7C72: 86 BF    >331          STX   AUXBANK
7C74: 91 9B    >332          STA   (LOWTR),Y
7C76: A9 00    >333  ]LOOP    LDA   #0         Hi byte of default dim
7C78: A2 0B    >334          LDX   #11        Lo byte of default dim
7C7A: 24 10    >335          BIT   DIMFLG
7C7C: 50 06    >336          BVC   :5
7C7E: FA       >338          PLX
7C7F: 68       >339          PLA
7C80: E8       >340          INX
7C81: D0 01    >341          BNE   *+3
7C83: 1A       >342          INC
7C84: C8       >351  :5      INY              ;Add this dimension to descr.
7C85: 91 9B    >352          STA   (LOWTR),Y
7C87: C8       >353          INY
7C88: 8A       >354          TXA
7C89: 91 9B    >355          STA   (LOWTR),Y
               >356  * Multiply this dimension by running size
               >357  * ((LOWTR),Y) * (STRNG2) --> A,X
7C8B: 20 AD E2 >358          JSR   MULTPLSS
7C8E: 86 AD    >359          STX   STRNG2
7C90: 85 AE    >360          STA   STRNG2+1
7C92: A4 5E    >361          LDY   INDEX
7C94: C6 0F    >362          DEC   NUMDIM
7C96: D0 DE    >363          BNE   ]LOOP
               >364
7C98: A4 BF    >365          LDY   AUXBANK
```

```
7C9A: F0 0F    >366           BEQ   :7
7C9C: A2 01    >367           LDX   #1          Ensure enough room in aux mem.
7C9E: 20 CB 7D >368           JSR   ZRTAUX
7CA1: E0 01    >369           CPX   #1          X set to 0 iif enough room
7CA3: B0 6B    >370           BCS   GME         otherwise -> MEMORY ERROR
7CA5: A5 94    >371           LDA   ARYPNT
7CA7: A4 95    >372           LDY   ARYPNT+1
7CA9: 90 0F    >373           BCC   :6          Always
               >374  * Now A,X has the total # of bytes of array elements
7CAB: 65 95    >375  :7       ADC   ARYPNT+1    Compute address of end of array
7CAD: B0 61    >376           BCS   GME         Too large: error
7CAF: 85 95    >377           STA   ARYPNT+1
7CB1: A8       >378           TAY
7CB2: 8A       >379           TXA
7CB3: 65 94    >380           ADC   ARYPNT
7CB5: 90 03    >381           BCC   :6
7CB7: C8       >382           INY
7CB8: F0 56    >383           BEQ   GME         Too large: error
7CBA: 20 E3 D3 >384  :6       JSR   REASON      Ensure enough room up to Y,A
7CBD: 85 6D    >385           STA   STREND
7CBF: 84 6E    >386           STY   STREND+1
7CC1: 38       >387           SEC
7CC2: E5 9B    >388           SBC   LOWTR
7CC4: A0 02    >389           LDY   #2
7CC6: 91 9B    >390           STA   (LOWTR),Y
7CC8: C8       >391           INY
7CC9: A5 6E    >392           LDA   STREND+1
7CCB: E5 9C    >393           SBC   LOWTR+1
7CCD: 91 9B    >394           STA   (LOWTR),Y
7CCF: A5 BF    >395           LDA   AUXBANK
7CD1: F0 25    >396           BEQ   :9
7CD3: 08       >397           PHP
7CD4: 78       >398           SEI
7CD5: 8D 09 C0 >399           STA   ALTZP
7CD8: A5 6D    >400           LDA   STREND
7CDA: A6 6E    >401           LDX   STREND+1
7CDC: 8D 08 C0 >402           STA   STDZP
7CDF: 28       >403           PLP
               >404  * AUXPTR a ete fixe dans ISAUXMEM a l´adresse du slot
               >405  * Adresse du 1er element en p0.
7CE0: 92 06    >407           STA   (AUXPTR)
7CE2: A0 01    >408           LDY   #1
7CE4: 8A       >414           TXA
7CE5: 91 06    >415           STA   (AUXPTR),Y
7CE7: C8       >416           INY
7CE8: A5 AD    >417           LDA   STRNG2
7CEA: 91 06    >418           STA   (AUXPTR),Y
7CEC: C8       >419           INY
7CED: A5 AE    >420           LDA   STRNG2+1
7CEF: 91 06    >421           STA   (AUXPTR),Y
7CF1: A2 02    >422           LDX   #2          Init memory slot for array
7CF3: 20 CB 7D >423           JSR   ZRTAUX
7CF6: 80 13    >424           BRA   :10
               >425  * Zero fill the element segment within the array
               >426  * (fast init).
7CF8: E6 AE    >427  :9       INC   STRNG2+1
7CFA: A4 AD    >428           LDY   STRNG2      # of byte mod 256
```

```
7CFC: F0 05    >429           BEQ   :8         Upon a page limit
7CFE: 88       >430  ]LOOP    DEY
7CFF: 91 94    >431           STA   (ARYPNT),Y
7D01: D0 FB    >432           BNE   ]LOOP
7D03: C6 95    >433  :8       DEC   ARYPNT+1   Point to next page
7D05: C6 AE    >434           DEC   STRNG2+1   Count the pages
7D07: D0 F5    >435           BNE   ]LOOP      Still more to clear
7D09: E6 95    >436           INC   ARYPNT+1   Rollback last Decrement
7D0B: A5 10    >437  :10      LDA   DIMFLG
7D0D: F0 04    >438           BEQ   NFAEP
7D0F: 60       >439           RTS
               >440
7D10: 4C 10 D4 >441  GME      JMP   MEMERR     MEMORY FULL error
7D13: A0 04    >442  NFAEP    LDY   #4
               >443  * New routine for ROM FIND.ARRAY.ELEMENT
               >444  * Y reg. should be 4 upon entry
7D15: B1 9B    >445           LDA   (LOWTR),Y
7D17: AA       >446           TAX
7D18: 4A       >448           LSR
7D19: 4A       >448           LSR
7D1A: 4A       >448           LSR
7D1B: 29 07    >450           AND   #7
7D1D: 85 BF    >451           STA   AUXBANK
7D1F: 8A       >452           TXA
7D20: 29 07    >453           AND   #7
7D22: 85 0F    >455           STA   NUMDIM
7D24: A9 00    >459           LDA   #0
7D26: 85 AD    >460           STA   STRNG2
7D28: 85 AE    >461  ]LOOP    STA   STRNG2+1
7D2A: C8       >462           INY              ;Pull next subscript from stack
7D2B: FA       >463           PLX
7D2C: 86 A0    >464           STX   FAC+3
7D2E: 68       >465           PLA
7D2F: 85 A1    >466           STA   FAC+4
7D31: D1 9B    >467           CMP   (LOWTR),Y
7D33: 90 0B    >468           BCC   FAE2
7D35: D0 06    >469           BNE   GSE        Subscript is too large
7D37: C8       >470           INY
7D38: 8A       >471           TXA
7D39: D1 9B    >472           CMP   (LOWTR),Y
7D3B: 90 04    >473           BCC   FAE3
7D3D: 4C 96 E1 >474  GSE      JMP   SUBERR     BAD SUBSCRIPT error
7D40: C8       >475  FAE2     INY
7D41: A5 AE    >476  FAE3     LDA   STRNG2+1   Bypass multiplication if
7D43: 05 AD    >477           ORA   STRNG2      value so far is zero
7D45: 18       >478           CLC
7D46: F0 0A    >479           BEQ   :1
7D48: 20 AD E2 >480           JSR   MULTPLSS
7D4B: 8A       >481           TXA              ;Add current subscript
7D4C: 65 A0    >482           ADC   FAC+3
7D4E: AA       >483           TAX
7D4F: 98       >484           TYA
7D50: A4 5E    >485           LDY   INDEX
7D52: 65 A1    >486  :1       ADC   FAC+4      Finish adding current subscript
7D54: 86 AD    >487           STX   STRNG2     Store accumulated offset
7D56: C6 0F    >488           DEC   NUMDIM     Last subscript yet?
7D58: D0 CE    >489           BNE   ]LOOP      No: loop till done
```

```
7D5A: 85 AE    >490           STA   STRNG2+1   Yes: multiply by element size
7D5C: 20 66 7D >491           JSR   KWELMSIZ
7D5F: A5 BF    >492           LDA   AUXBANK
7D61: F0 00    >493           BEQ   :2
7D63: 4C 98 E2 >494   :2      JMP   $E298
               >495
               >496   * Donne la taille de l´element en fonction
               >497   * de VARNAM,+1 et de INTTYP
               >498   * Result in X reg.
7D66: 24 81    >499   KWELMSIZ BIT   VARNAM
7D68: 10 06    >500           BPL   :0
7D6A: A5 12    >501           LDA   INTTYP
7D6C: 29 07    >502           AND   #7
7D6E: AA       >503           TAX
7D6F: 60       >504           RTS
7D70: A2 05    >505   :0      LDX   #5
7D72: 24 82    >506           BIT   VARNAM+1
7D74: 10 02    >507           BPL   :1
7D76: CA       >508           DEX                ;Back to 3 if string
7D77: CA       >509           DEX
7D78: 60       >510   :1      RTS
               >511
               >512   * Evaluate numeric formula at TXTPPTR
               >513   * Converting result to INTEGER 0<= X < 65536
               >514   * into FAC+3,4
7D79: 20 2A 76 >515   NMAKINT  JSR   RST100     Get next character
7D7C: 20 3B 85 >516           JSR   NFRMNUM
               >517   * Convert FAC to integer
7D7F: A5 A2    >518           LDA   FACSIGN
7D81: 30 0F    >519           BMI   :1
7D83: A5 9D    >520           LDA   FAC
7D85: C9 90    >521           CMP   #$90
7D87: 90 06    >522           BCC   :3          Branch if abs(value) < 32768
7D89: 20 90 8E >523           JSR   GN65536
7D8C: 20 BE E7 >524           JSR   FADD
7D8F: 4C F2 EB >525   :3      JMP   QINT
7D92: 4C 99 E1 >526   :1      JMP   GOIQERR
               >527
               >528   * Convert INTTYP (in X reg.) from $81 to $84
               >529   * to %0000_0000 to %1100_0000 (respectively)
               >530   * Output value could be ORA ed or EOR ed with
               >531   * NUMDIM slot with array structure
7D95: CA       >532   CNVT1    DEX
7D96: 8A       >533           TXA
7D97: 4A       >534           LSR                ;b0 into Carry, 0 into b7
7D98: 6A       >535           ROR                ;b0 into b7 and b1 into carry
7D99: 6A       >536           ROR                ;b0 into b6, b1 into b7
7D9A: 29 C0    >537           AND   #$C0         Only retain b6-b7
7D9C: 60       >538           RTS
               >539
               >540   * Cache mechanism for array variables
               >541   ACTR     EQU   LOWTR
7D9D: A4 82    >570   ALKCACH  LDY   VARNAM+1
7D9F: A5 81    >571           LDA   VARNAM
7DA1: 86 9B    >572           STX   SCTR
7DA3: A2 00    >573           LDX   #0
7DA5: DD 99 99 >574   ]LOOP    CMP   AVN,X
```

```
7DA8: D0 0F    >575           BNE   :0
7DAA: 98       >576           TYA
7DAB: DD 9D 99 >577           CMP   AVNP1,X
7DAE: D0 07    >578           BNE   :2
7DB0: A5 12    >579           LDA   INTTYP
7DB2: DD A1 99 >580           CMP   AIT,X
7DB5: F0 08    >581           BEQ   :1
7DB7: A5 81    >582  :2       LDA   VARNAM
7DB9: E8       >583  :0       INX
7DBA: E4 9B    >584           CPX   SCTR
7DBC: D0 E7    >585           BNE   ]LOOP
7DBE: 60       >587           RTS
               >588
7DBF: BD A5 99 >589  :1       LDA   ALTR,X
7DC2: 85 9B    >590           STA   LOWTR
7DC4: BD A9 99 >595           LDA   ALTRP1,X
7DC7: 85 9C    >596           STA   LOWTR+1
7DC9: 8A       >598           TXA
7DCA: 60       >599           RTS
               >600
               >601  * Common entry point for accessing array content
               >602  * within auxiliary memory.
7DCB: A9 BF    >603  ZRTAUX   LDA   #$BF
7DCD: 8D EE 03 >604           STA   $03EE
7DD0: 9C ED 03 >606           STZ   $03ED
7DD3: B8       >611           CLV
7DD4: 38       >612           SEC
7DD5: 4C 14 C3 >613           JMP   XFER
               >614
7DD8: 2C 83 C0 >615  NGARBAG  BIT   $C083
7DDB: 2C 83 C0 >616           BIT   $C083
7DDE: 20 00 D0 >617           JSR   $D000
7DE1: 2C 81 C0 >618           BIT   $C081
7DE4: 2C 81 C0 >619           BIT   $C081
7DE7: 60       >620           RTS
                554  * New strategy for array storage
                555           PUT   PEERNAUXMEM
               >1    * Module handling the new Peersoft array storage strategy
               >2
7DE8: 4C C9 DE >3    GSNERR2  JMP   SYNERR
7DEB: 4C 99 E1 >4    GIQERR2  JMP   GOIQERR
7DEE: 4C 76 DD >5    GTMERR2  JMP   GOTMIERR
               >6    * Routine to test whether the array will be located
               >7    * Outcome:
               >8    * Carry set iif aux. mem storage asked for
               >9    * AUXBANK: bank memory asked for (in bits b4..b5)
               >10   * ARYPNT,+1: incremented if aux mem. storage
               >11   * (placeholders for offset within aux memory and
               >12   *  one element of specified size for returning values
               >13   *  during value expressions
               >14   * Y,A: values incremented in case aux. mem storage
7DF1: B2 B8    >16   ISAUXMEM LDA   (TXTPTR)
7DF3: C9 23    >20           CMP   #´#´
7DF5: 18       >21           CLC
7DF6: D0 37    >22           BNE   :2
7DF8: 20 2A 76 >23           JSR   RST100     Next char. must be numeric
7DFB: B0 EB    >24           BCS   GSNERR2    otherwise SYNTAX ERROR
```

```
7DFD: 29 07    >25            AND   #7
               >26   * Pour le moment uniquement la memoire auxiliaire
               >27   * est autorisee
7DFF: C9 02    >28            CMP   #2
7E01: B0 E8    >29            BCS   GIQERR2
7E03: 85 BF    >30            STA   AUXBANK
7E05: 20 2A 76 >31            JSR   RST100     Point to next character
7E08: 18       >32            CLC
               >33   * test de conformance par rap. a la configuration hote
7E09: 2C EF 9C >34            BIT   MEMORY     b6 a 1 si carte mem aux.
7E0C: A9 01    >36            LDA   #1
7E0E: 50 01    >37            BVC   *+3
7E10: 3A       >38            DEC
7E11: 14 BF    >39            TRB   AUXBANK
7E13: A5 BF    >40            LDA   AUXBANK
7E15: F0 18    >49            BEQ   :2
7E17: A5 94    >50            LDA   ARYPNT
7E19: A4 95    >51            LDY   ARYPNT+1
7E1B: 85 06    >52            STA   AUXPTR
7E1D: 84 07    >53            STY   AUXPTR+1
7E1F: 65 AD    >54            ADC   STRNG2     Carry already clear
7E21: 90 02    >55            BCC   *+4
7E23: C8       >56            INY
7E24: 18       >57            CLC
7E25: 69 04    >58            ADC   #4
7E27: 90 01    >59            BCC   *+3
7E29: C8       >60            INY
7E2A: 84 95    >61            STY   ARYPNT+1
7E2C: 38       >62            SEC
7E2D: 85 94    >63            STA   ARYPNT
7E2F: A5 94    >64   :2       LDA   ARYPNT
7E31: 60       >65   ]LOOP    RTS
               >66
7E32: 2C EF 9C >67   RCLMAUX  BIT   MEMORY
7E35: 50 FA    >68            BVC   ]LOOP
7E37: A2 00    >69            LDX   #0         Init array storage in aux mem.
7E39: 4C CB 7D >70            JMP   ZRTAUX
                556
                557  * Upon init, all variables are floating point by default
7E3C: 08        558  LBS00    PHP
7E3D: A2 1A     559            LDX   #26
7E3F: A9 21     560            LDA   #´!´
7E41: 9D 95 9B  561  ]LOOP    STA   TYPLET-1,X
7E44: CA        562            DEX
7E45: D0 FA     563            BNE   ]LOOP
7E47: 8E 23 96  564            STX   AEI
7E4A: 8E 24 96  565            STX   AEI+1
                566  * Reinit variables lookup caches (simple & array)
7E4D: 8E 83 99  567            STX   SNCCH
7E50: 8E 98 99  568            STX   ANCCH
7E53: 8E E7 9C  569            STX   WMODE
7E56: 20 32 7E  570            JSR   RCLMAUX
7E59: 28        571            PLP
7E5A: 60        572            RTS
                573
                574  * Applesoft RUN command
7E5B: 20 3C 7E  575  RRUN      JSR   LBS00      Init the default vartype table
```

```
7E5E: 8E C1 99  576           STX   MONU       Rearms MOUSE instruction flag
7E61: 4C 12 D9  577           JMP   $D912
                578
                579   * Applesoft NEW command
7E64: 20 3C 7E  580   RNEW    JSR   LBS00
7E67: 4C 4B D6  581           JMP   $D64B
                582
                583   * Applesoft CLEAR command
7E6A: 20 3C 7E  584   RCLEAR  JSR   LBS00
7E6D: 4C 6C D6  585           JMP   $D66C
                586
7E70: 20 7D E0  587   MISLETC JSR   ISLETC
7E73: 90 08     588           BCC   GOSYNERR
7E75: 60        589           RTS
                590
                591   * New subroutine checking a character (code in A)
                592   * is pointed to by TXTPTR
                593   * Falls into SYNERR if not
                594   NSYNCHR  DO    KOPT-K65C02
7E76: D2 B8     598   NSYNCHR2 CMP   (TXTPTR)
7E78: D0 03     600           BNE   GOSYNERR
7E7A: 4C 2A 76  601           JMP   RST100
7E7D: 4C C9 DE  602   GOSYNERR JMP   SYNERR
                603
                604           PUT   PEERPROCFUN
               >1     * Module en charge des fonctions utilisateur
               >2     * et particulierement des PF
               >3     ARG      EQU   $A5
               >4     TRCFLG   EQU   $F2
               >5     BISVTYP  EQU   $BE
               >6     VECTUSR  EQU   $A
               >7     TMERR    EQU   $DD76
               >8     ULERR    EQU   $D97C
               >9     MOVFM    EQU   $EAF9
               >10    MOVFA    EQU   $EB53
               >11    LET2     EQU   $DA63
               >12
               >13             DUMMY 0
0000: 00       >14    USRMOD   DS    1
0001: 00 00    >15    ADRUSR   DS    2
0003: 00 00    >16    VSRTNAM  DS    2
0005: 00       >17    VSRTVT   DS    1
0006: 00       >18    VSRTIT   DS    1
0007: 00 00    >19    VSRTPTR  DS    2
0009: 00 00    >20    VENT1NAM DS    2
000B: 00       >21    VENT1VT  DS    1
000C: 00       >22    VENT1IT  DS    1
000D: 00 00    >23    VENT1PTR DS    2
000F: 00 00    >24    VENT2NAM DS    2
0011: 00       >25    VENT2VT  DS    1
0012: 00       >26    VENT2IT  DS    1
0013: 00 00    >27    VENT2PTR DS    2
               >28    LENREC   EQU   *
               >29             DEND
               >30    * Sous routine pour initialiser les routines USR de type
               >31    * PF.
7E80: A2 0A    >32    RAZPF    LDX   #10
```

```
                >33   ]LOOP    MPHX
7E82: DA        >33             PHX
7E83: 20 A9 7E  >34             JSR   COMPOFST
7E86: FA        >35             PLX
7E87: B2 06     >37             LDA   (AUXPTR)
7E89: 10 06     >42             BPL   :0
7E8B: A0 02     >43             LDY   #ADRUSR+1
7E8D: A9 00     >44             LDA   #0
7E8F: 91 06     >45             STA   (AUXPTR),Y
7E91: CA        >46   :0        DEX
7E92: 10 EE     >47             BPL   ]LOOP
7E94: 8E 81 99  >48             STX   PFINDIC
7E97: 9C 80 99  >50             STZ   ISPFACT
7E9A: 60        >55             RTS
                >56
7E9B: A2 0B     >57   SETINITX  LDX   #12-1
7E9D: BD 74 99  >58   ]LOOP     LDA   SINITX,X
7EA0: 95 69     >59             STA   $69,X
7EA2: 9D 54 97  >60             STA   SVALTNM,X
7EA5: CA        >61             DEX
7EA6: 10 F5     >62             BPL   ]LOOP
7EA8: 60        >63             RTS
                >64
                >65   * Indice de la fonction dans X, ramene dans A,Y
                >66   * L´adresse de debut de la structure
7EA9: A9 00     >67   COMPOFST  LDA   #0
7EAB: A8        >68             TAY
7EAC: F0 05     >69             BEQ   :00         Always
7EAE: 69 15     >70   ]LOOP     ADC   #LENREC
7EB0: 90 02     >71             BCC   :0
7EB2: C8        >72             INY
7EB3: 18        >73   :00       CLC
7EB4: CA        >74   :0        DEX
7EB5: 10 F7     >75             BPL   ]LOOP
7EB7: 69 45     >76             ADC   #ADRSTRUCT
7EB9: 48        >77             PHA
7EBA: 98        >78             TYA
7EBB: 69 96     >79             ADC   #>ADRSTRUCT
7EBD: A8        >80             TAY
7EBE: 68        >81             PLA
7EBF: 85 06     >82             STA   AUXPTR
7EC1: 84 07     >83             STY   AUXPTR+1
7EC3: 60        >84             RTS
                >85
7EC4: 18        >86   GOSVCUR   CLC
                >87   ]LOOP
                >88   * Connaitre tout d´une variable non encore enregistree
                >89   * A: offset du premier byte pour la var. dans structure
7EC5: 4C 76 DD  >90   ]ERR      JMP   TMERR
7EC8: 48        >91   FRSTIM    PHA
7EC9: 20 A6 86  >92             JSR   NCHKCOM
7ECC: B2 06     >94             LDA   (AUXPTR)
7ECE: 29 01     >99             AND   #1          Environnement dynamique oui/non
7ED0: 48        >100            PHA
7ED1: F0 0F     >101            BEQ   :0
7ED3: A2 0B     >102            LDX   #12-1
7ED5: B5 69     >103  ]LOOP     LDA   $69,X
```

```
7ED7: 9D 48 97 >104           STA   SVCURRM,X
7EDA: BD 68 97 >105           LDA   SDEF1,X
7EDD: 95 69    >106           STA   $69,X
7EDF: CA       >107           DEX
7EE0: 10 F3    >108           BPL   ]LOOP
7EE2: A5 07    >112  :0       LDA   AUXPTR+1
7EE4: 48       >113           PHA
7EE5: A5 06    >114           LDA   AUXPTR
7EE7: 48       >115           PHA
7EE8: 20 06 7A >117           JSR   NPTRGTX
7EEB: C5 6B    >118           CMP   ARYTAB
7EED: 98       >119           TYA
7EEE: E5 6C    >120           SBC   ARYTAB+1
7EF0: 68       >121           PLA
7EF1: 85 06    >122           STA   AUXPTR
7EF3: 68       >123           PLA
7EF4: 85 07    >124           STA   AUXPTR+1
7EF6: 68       >125           PLA
7EF7: F0 0A    >126           BEQ   :1
7EF9: A2 0B    >127           LDX   #12-1
7EFB: BD 48 97 >128  ]LOOP    LDA   SVCURRM,X
7EFE: 95 69    >129           STA   $69,X
7F00: CA       >130           DEX
7F01: 10 F8    >131           BPL   ]LOOP
7F03: B0 C0    >132  :1       BCS   ]ERR
7F05: 7A       >133           PLY
7F06: A5 81    >134           LDA   VARNAM
7F08: 91 06    >135           STA   (AUXPTR),Y
7F0A: C8       >136           INY
7F0B: A5 82    >137           LDA   VARNAM+1
7F0D: 91 06    >138           STA   (AUXPTR),Y
7F0F: C8       >139           INY
7F10: A5 11    >140           LDA   VALTYP
7F12: 91 06    >141           STA   (AUXPTR),Y
7F14: C8       >142           INY
7F15: A5 12    >143           LDA   INTTYP
7F17: 91 06    >144           STA   (AUXPTR),Y
7F19: C8       >145           INY
7F1A: A5 83    >146  COMX1    LDA   VARPNT
7F1C: 91 06    >147           STA   (AUXPTR),Y
7F1E: C8       >148           INY
7F1F: A5 84    >149           LDA   VARPNT+1
7F21: 91 06    >150           STA   (AUXPTR),Y
7F23: 60       >151           RTS
               >152
               >153  * Connaitre tout d´une variable deja enregistree
               >154  * Y offset dans structure... (adressage par
               >155  * (AUXPTR),Y
7F24: B1 06    >156  SCNDTIM  LDA   (AUXPTR),Y
7F26: 85 81    >157           STA   VARNAM
7F28: C8       >158           INY
7F29: B1 06    >159           LDA   (AUXPTR),Y
7F2B: 85 82    >160           STA   VARNAM+1
7F2D: C8       >161           INY
7F2E: B1 06    >162           LDA   (AUXPTR),Y
7F30: 85 11    >163           STA   VALTYP
7F32: C8       >164           INY
```

```
7F33: B1 06    >165           LDA   (AUXPTR),Y
7F35: 85 12    >166           STA   INTTYP
7F37: C8       >167           INY
7F38: 5A       >168           PHY
7F39: 20 72 7A >169           JSR   NPTRGL90
7F3C: 7A       >170           PLY
7F3D: 80 DB    >171           BRA   COMX1
               >172
               >173  * X,A adresse a sauver dans ADRUSR de la structure
7F3F: A0 01    >174  HNDLEADR LDY   #ADRUSR
7F41: 91 06    >175           STA   (AUXPTR),Y
7F43: 90 08    >176           BCC   :4
7F45: 85 0B    >177           STA   $0B
7F47: 86 0C    >178           STX   $0C
7F49: A9 4C    >179           LDA   #$4C
7F4B: 85 0A    >180           STA   $0A
7F4D: C8       >181  :4        INY
7F4E: 8A       >182           TXA
7F4F: 91 06    >183           STA   (AUXPTR),Y
7F51: 60       >184           RTS
               >185
7F52: B1 06    >186  COMLET2  LDA   (AUXPTR),Y
7F54: AA       >187           TAX              ;INTTYP dans X
7F55: C8       >188           INY
7F56: B1 06    >189           LDA   (AUXPTR),Y ;pointeur sur valeur
7F58: 85 85    >190           STA   FORPNT      dans FORPNT
7F5A: C8       >191           INY
7F5B: B1 06    >192           LDA   (AUXPTR),Y
7F5D: 85 86    >193           STA   FORPNT+1
7F5F: 8A       >194           TXA              ;Set bit N
7F60: 4C 63 DA >195           JMP   LET2
               >196
7F63: 4C 10 D4 >197  ]ERR      JMP   MEMERR
7F66: 20 2A 76 >198  RUSR      JSR   RST100
7F69: A2 0A    >199           LDX   #10
7F6B: B0 06    >200           BCS   :0         Not a digit
7F6D: E9 2F    >201           SBC   #´0´-1
7F6F: AA       >202           TAX
7F70: 20 2A 76 >203           JSR   RST100
               >204  :0        MPHX
7F73: DA       >204           PHX
7F74: 20 A9 7E >205           JSR   COMPOFST
7F77: B2 06    >207           LDA   (AUXPTR)
7F79: 29 40    >212           AND   #64
7F7B: F0 41    >213           BEQ   :1
7F7D: BA       >214           TSX
7F7E: E0 08    >215           CPX   #8         At least 8 bytes on stack OK
7F80: 90 E1    >216           BCC   ]ERR
7F82: 20 A9 86 >217           JSR   NCHKOPN
7F85: 20 7B DD >218           JSR   FRMEVL
7F88: BA       >219           TSX
7F89: A5 11    >220           LDA   VALTYP
7F8B: 9D 00 01 >221           STA   $0100,X
7F8E: 8A       >222           TXA
7F8F: 38       >223           SEC
7F90: E9 06    >224           SBC   #6
7F92: AA       >225           TAX
```

```
7F93: 9A       >226           TXS
7F94: E8       >227           INX
7F95: A0 01    >228           LDY   #1
7F97: 20 2B EB >229           JSR   MOVMF
7F9A: 20 A6 86 >230           JSR   NCHKCOM
7F9D: 20 A0 86 >231           JSR   NPARCHK+3  2nd arg value left in FAC
7FA0: BA       >232           TSX
7FA1: E8       >233           INX
7FA2: 8A       >234           TXA
7FA3: 48       >235           PHA
7FA4: A0 01    >236           LDY   #1
7FA6: 20 E3 E9 >237           JSR   $E9E3      Load ARG from Y,A/1st arg value
7FA9: 68       >238           PLA
7FAA: 18       >239           CLC
7FAB: 69 05    >240           ADC   #5          6 instead of 5 because of INX
7FAD: AA       >241           TAX
7FAE: BD 00 01 >242           LDA   $0100,X
7FB1: 85 BE    >243           STA   BISVTYP
7FB3: 9A       >244           TXS
7FB4: 80 0B    >245           BRA   :2
7FB6: A2 26    >246  ]ERR     LDX   #38
7FB8: 2C       >247           HEX   2C          Skip next two bytes
7FB9: A2 27    >248  ]ERR1    LDX   #39
7FBB: 4C 1C 8E >249           JMP   NERRH
7FBE: 20 9D 86 >250  :1       JSR   NPARCHK     1er ou 2eme parm dans FAC
               >251  :2       MPLX
7FC1: FA       >251           PLX
7FC2: DA       >253           PHX
7FC3: 20 A9 7E >257           JSR   COMPOFST    Set AUXPTR according index X
7FC6: A0 02    >258           LDY   #ADRUSR+1
7FC8: B1 06    >259           LDA   (AUXPTR),Y
7FCA: F0 EA    >260           BEQ   ]ERR
7FCC: FA       >261           PLX
7FCD: 8E 82 99 >262           STX   PFINDX
7FD0: B2 06    >264           LDA   (AUXPTR)
7FD2: 10 48    >269           BPL   V3
               >270  * Procedural function...
7FD4: 4A       >271           LSR
7FD5: 90 2A    >272           BCC   :10         Branchem. ssi pas de segment
7FD7: AD 80 99 >273           LDA   ISPFACT
7FDA: D0 DD    >274           BNE   ]ERR1
7FDC: DA       >275           PHX
7FDD: 20 5F 81 >276           JSR   SAVCURRM
7FE0: 68       >277           PLA
7FE1: CD 81 99 >278           CMP   PFINDIC
7FE4: F0 03    >279           BEQ   :11
7FE6: 20 9B 7E >280           JSR   SETINITX
7FE9: 20 54 81 >281  :11      JSR   RSTALTM
7FEC: A0 03    >282           LDY   #VSRTNAM
7FEE: 20 24 7F >283           JSR   SCNDTIM
7FF1: A0 09    >284           LDY   #VENT1NAM
7FF3: 20 24 7F >285           JSR   SCNDTIM
7FF6: B2 06    >287           LDA   (AUXPTR)
7FF8: 29 40    >292           AND   #64
7FFA: F0 05    >293           BEQ   :10
7FFC: A0 0F    >294           LDY   #VENT2NAM
7FFE: 20 24 7F >295           JSR   SCNDTIM
```

```
8001: A0 0C    >296  :10      LDY   #VENT1IT
8003: 20 52 7F >297           JSR   COMLET2
8006: B2 06    >299           LDA   (AUXPTR)
8008: 29 40    >304           AND   #64
800A: F0 08    >305           BEQ   :12
800C: 20 53 EB >306           JSR   MOVFA
800F: A0 12    >307           LDY   #VENT2IT
8011: 20 52 7F >308           JSR   COMLET2
               >309  :12      DO    KOPT16
8014: A9 80    >312  V3T      LDA   #>RETOUR-1
8016: 48       >313           PHA
8017: A9 F2    >314  V3B      LDA   #RETOUR-1
8019: 48       >315           PHA
801A: 80 16    >317           BRA   COMMONG
               >318
               >319  * Code run when parsing USR function that is not a PF
801C: E0 0A    >320  V3       CPX   #10
801E: B0 0F    >321           BCS   :4         Special case for original USR
8020: A0 02    >322           LDY   #ADRUSR+1
8022: B1 06    >323           LDA   (AUXPTR),Y
8024: AA       >324           TAX
8025: 88       >325           DEY
8026: B1 06    >326           LDA   (AUXPTR),Y
8028: D0 01    >327           BNE   *+3
802A: CA       >328           DEX
802B: 3A       >330           DEC
802C: DA       >336           PHX
802D: 48       >337           PHA
802E: 60       >343           RTS
802F: 4C 0A 00 >344  :4        JMP   VECTUSR
               >345
8032: A0 0D    >346  COMMONG   LDY   #FINOF-SVOFST-1
8034: BE 2C 97 >347  ]LOOP     LDX   SVOFST,Y
8037: B5 00    >348           LDA   0,X
8039: 99 3A 97 >349           STA   SVAREA,Y
803C: 88       >350           DEY
803D: 10 F5    >351           BPL   ]LOOP
803F: 64 F2    >353           STZ   TRCFLG
               >358  * This is the critical code segment
8041: A5 B9    >363           LDA   TXTPTR+1
8043: 48       >364           PHA
8044: A5 B8    >365           LDA   TXTPTR
8046: 48       >366           PHA
8047: A5 76    >367           LDA   CURLIN+1
8049: 48       >368           PHA
804A: A5 75    >369           LDA   CURLIN
804C: 48       >370           PHA
804D: A9 B0    >372           LDA   #TOKGOSUB
804F: 48       >373           PHA
8050: A0 01    >374           LDY   #ADRUSR
8052: B1 06    >375           LDA   (AUXPTR),Y
8054: 85 B8    >376           STA   TXTPTR
8056: C8       >377           INY
8057: B1 06    >378           LDA   (AUXPTR),Y
8059: 85 B9    >379           STA   TXTPTR+1
805B: 4C D2 D7 >380           JMP   NEWSTT
               >381
```

```
805E: 20 32 76 >382  RDEFUSR  JSR   RST102
8061: 90 05    >383           BCC   :1          Branch if digit
8063: A9 0A    >384           LDA   #10
8065: 48       >385           PHA
8066: D0 06    >386           BNE   :3          Always
8068: E9 2F    >387  :1       SBC   #´0´-1      ASCII digit to binary
806A: 48       >388           PHA
806B: 20 2A 76 >389           JSR   RST100
806E: A9 D0    >390  :3       LDA   #TOKEQUAL
8070: 20 76 7E >391           JSR   NSYNCHR
8073: 20 67 DD >392           JSR   FRMNUM
8076: 20 52 E7 >393           JSR   GETADR
8079: FA       >394           PLX
807A: DA       >396           PHX
807B: 20 A9 7E >400           JSR   COMPOFST
807E: 68       >401           PLA
807F: 48       >402           PHA
8080: C9 0A    >403           CMP   #10         Set carry flag
               >404  * If LINNUM high byte is zero, then must be the mode
8082: A5 50    >405           LDA   LINNUM
8084: A6 51    >406           LDX   LINNUM+1
8086: F0 11    >407           BEQ   :5
8088: 20 3F 7F >408           JSR   HNDLEADR
808B: 68       >409           PLA
808C: A9 00    >410           LDA   #0
808E: 92 06    >412           STA   (AUXPTR)
8090: 20 32 76 >417  ]LOOP    JSR   RST102
8093: D0 01    >418           BNE   *+3
8095: 60       >419           RTS
8096: 4C C9 DE >420  ]ERR     JMP   SYNERR
               >421  * DEFUSR=<mode>,<otherparms>
8099: 92 06    >423  :5       STA   (AUXPTR)
809B: A8       >428           TAY
809C: 30 24    >429           BMI   :6          Procedural function
809E: 29 3F    >430           AND   #$3F
80A0: D0 F4    >431           BNE   ]ERR
80A2: 20 A6 86 >432           JSR   NCHKCOM
80A5: 20 67 DD >433           JSR   FRMNUM
80A8: 20 52 E7 >434           JSR   GETADR
80AB: FA       >435           PLX
80AC: E0 0A    >436           CPX   #10
80AE: 08       >437           PHP
80AF: 20 A9 7E >438           JSR   COMPOFST
80B2: 28       >439           PLP
80B3: A5 50    >440           LDA   LINNUM
80B5: A6 51    >441           LDX   LINNUM+1
80B7: 4C 3F 7F >442  ]LOOP    JMP   HNDLEADR
80BA: 4C 7C D9 >443  ]ERR     JMP   ULERR
80BD: A2 28    >444  ]ERR1    LDX   #40
80BF: 4C 1C 8E >445           JMP   NERRH
80C2: 48       >446  :6       PHA
80C3: AD 80 99 >447           LDA   ISPFACT
80C6: D0 F5    >448           BNE   ]ERR1
80C8: A9 03    >449           LDA   #VSRTNAM
80CA: 20 C8 7E >450           JSR   FRSTIM
80CD: A9 09    >451           LDA   #VENT1NAM
80CF: 20 C8 7E >452           JSR   FRSTIM
```

```
80D2: 68       >453           PLA
80D3: 29 40    >454           AND   #64
80D5: F0 05    >455           BEQ   :7
80D7: A9 0F    >456           LDA   #VENT2NAM
80D9: 20 C8 7E >457           JSR   FRSTIM
80DC: 68       >458  :7       PLA             ;Do not care routine idx
80DD: 20 A6 86 >459           JSR   NCHKCOM
80E0: 20 0C DA >460           JSR   LINGET
80E3: 20 1A D6 >461           JSR   FNDLIN
80E6: 90 D2    >462           BCC   ]ERR
80E8: A6 9C    >463           LDX   LOWTR+1
80EA: A5 9B    >464           LDA   LOWTR
80EC: D0 01    >465           BNE   *+3
80EE: CA       >466           DEX
80EF: 3A       >468           DEC
80F0: 18       >472           CLC
80F1: 90 C4    >473           BCC   ]LOOP     Always
               >474
80F3: 20 13 81 >475  RETOUR   JSR   COMREST
80F6: AE 82 99 >476           LDX   PFINDX
80F9: DA       >477           PHX
80FA: 20 A9 7E >478           JSR   COMPOFST
80FD: 20 21 81 >479           JSR   COLLECTR
8100: FA       >480           PLX
8101: B2 06    >482           LDA   (AUXPTR)
8103: 9C 80 99 >483           STZ   ISPFACT
8106: 4A       >489           LSR
8107: 90 09    >490           BCC   :0
8109: 8E 81 99 >491           STX   PFINDIC
810C: 20 6A 81 >492           JSR   SAVALTM
810F: 4C 49 81 >493           JMP   RSTCURRM
8112: 60       >494  :0       RTS
               >495
8113: A0 0D    >496  COMREST  LDY   #FINOF-SVOFST-1
8115: BE 2C 97 >497  ]LOOP    LDX   SVOFST,Y
8118: B9 3A 97 >498           LDA   SVAREA,Y
811B: 95 00    >499           STA   0,X
811D: 88       >500           DEY
811E: 10 F5    >501           BPL   ]LOOP
8120: 60       >502           RTS
               >503
8121: A0 06    >504  COLLECTR LDY   #VSRTIT
8123: B1 06    >505           LDA   (AUXPTR),Y
8125: 0A       >506           ASL
8126: A0 07    >507           LDY   #VSRTPTR
8128: B1 06    >508           LDA   (AUXPTR),Y
812A: AA       >509           TAX
812B: C8       >510           INY
812C: B1 06    >511           LDA   (AUXPTR),Y
812E: A8       >512           TAY
812F: 8A       >513           TXA
8130: B0 07    >514           BCS   :0        Branch iif integer output var.
8132: 64 11    >516           STZ   VALTYP
8134: 64 12    >517           STZ   INTTYP
8136: 4C F9 EA >523           JMP   MOVFM
8139: 84 84    >524  :0       STY   VARPNT+1
813B: 85 83    >525           STA   VARPNT
```

```
813D: B2 83    >527           LDA   (VARPNT)
813F: A0 01    >528           LDY   #1
8141: AA       >534           TAX
8142: B1 83    >535           LDA   (VARPNT),Y
8144: A8       >536           TAY
8145: 8A       >537           TXA
8146: 4C F2 E2 >538           JMP   GIVAYF
               >539
8149: A2 0B    >540  RSTCURRM LDX   #12-1
814B: BD 48 97 >541  ]LOOP    LDA   SVCURRM,X
814E: 95 69    >542           STA   $69,X
8150: CA       >543           DEX
8151: 10 F8    >544           BPL   ]LOOP
8153: 60       >545           RTS
               >546
8154: A2 0B    >547  RSTALTM  LDX   #12-1
8156: BD 54 97 >548  ]LOOP    LDA   SVALTNM,X
8159: 95 69    >549           STA   $69,X
815B: CA       >550           DEX
815C: 10 F8    >551           BPL   ]LOOP
815E: 60       >552           RTS
               >553
815F: A2 0B    >554  SAVCURRM LDX   #12-1
8161: B5 69    >555  ]LOOP    LDA   $69,X
8163: 9D 48 97 >556           STA   SVCURRM,X
8166: CA       >557           DEX
8167: 10 F8    >558           BPL   ]LOOP
8169: 60       >559           RTS
               >560
816A: A2 0B    >561  SAVALTM  LDX   #12-1
816C: B5 69    >562  ]LOOP    LDA   $69,X
816E: 9D 54 97 >563           STA   SVALTNM,X
8171: CA       >564           DEX
8172: 10 F8    >565           BPL   ]LOOP
8174: 60       >566           RTS
                605           PUT   PEERDEF
               >1    * Nouvelle routine de traitement du DEF..
8175: 4C 5E 80 >2    ]LOOP    JMP   RDEFUSR
8178: A4 B9    >3    RDEF     LDY   TXTPTR+1
817A: A5 B8    >4             LDA   TXTPTR
817C: D0 01    >11            BNE   *+3
817E: 88       >12            DEY
817F: 3A       >13            DEC
8180: A2 01    >15            LDX   #1
8182: 20 C4 82 >16            JSR   RECON      Check which DEF pattern
8185: D0 03    >17            BNE   :1         None detected
8187: 4C 13 E3 >18            JMP   $E313
818A: 88       >19   :1       DEY
818B: 20 98 D9 >20            JSR   ADDON
818E: A6 BD    >21            LDX   IDMOCL
8190: E0 0A    >22            CPX   #OFFUSR-TOFFST Is it DEFUSR?
8192: F0 E1    >23            BEQ   ]LOOP
8194: BD 77 9B >24            LDA   MOTIF-NOPER-7,X Must be DEF(INT/STR/SNG)
               >25   * Below is the common code for all three new instructions
8197: 64 C0    >30            STZ   LETINF
8199: 85 C1    >32            STA   TYPMOD
819B: 20 09 82 >33            JSR   DECTPTR    Decrement TXTPTR
```

```
819E: 20 D4 81 >34   ]LOOP    JSR   :LBS00     Bump ptr. to 1st letter of next v
ar
81A1: 20 70 7E >35            JSR   MISLETC    Must be alphabetic
81A4: 85 C0    >36            STA   LETINF
81A6: 20 D4 81 >37            JSR   :LBS00     Exit if no further variable
81A9: C9 C9    >38            CMP   #TOKMINUS  means a letter range
81AB: F0 0B    >39            BEQ   :2
81AD: C9 2C    >40            CMP   #´,´       Character must be either ´,´
81AF: D0 34    >41            BNE   GSNERR3     or ´-´
81B1: A6 C0    >42            LDX   LETINF     Process current letter
81B3: 20 DF 81 >43            JSR   RDEFSUB
81B6: 10 E6    >44            BPL   ]LOOP      Always
81B8: 20 2A 76 >45   :2       JSR   RST100     Range:get the upper range let.
81BB: 20 70 7E >46            JSR   MISLETC
81BE: C5 C0    >47            CMP   LETINF     Must not < 1st letter
81C0: 90 23    >48            BCC   GSNERR3
81C2: AA       >49            TAX              ;Into X for processing
81C3: 20 DF 81 >50   ]JLOOP   JSR   RDEFSUB    process current letter within
81C6: CA       >51            DEX
81C7: E4 C0    >52            CPX   LETINF     Loop until 1st letter
81C9: B0 F8    >53            BCS   ]JLOOP
81CB: 20 D4 81 >54            JSR   :LBS00
81CE: C9 2C    >55            CMP   #´,´
81D0: D0 13    >56            BNE   GSNERR3
81D2: F0 CA    >57            BEQ   ]LOOP      Always
81D4: 20 2A 76 >58   :LBS00   JSR   RST100
81D7: D0 0B    >59            BNE   R          Do not return if EOI
81D9: 68       >60            PLA
81DA: 68       >61            PLA
81DB: A6 C0    >62   :FIN     LDX   LETINF
81DD: F0 06    >63            BEQ   GSNERR3    Whaever args, process last letter
81DF: A5 C1    >64   RDEFSUB  LDA   TYPMOD
81E1: 9D 55 9B >65            STA   TYPLET-´A´,X
81E4: 60       >66   R        RTS
81E5: 4C C9 DE >67   GSNERR3  JMP   SYNERR
               >68
               >125
81E8: 20 2A 76 >142  ROUT1Y   JSR   RST100
81EB: 48       >143           PHA
81EC: BD 8E 9B >144  ROUT1X   LDA   TVNORA,X
81EF: 04 81    >145           TSB   VARNAM
81F1: BD 92 9B >146           LDA   TVN1ORA,X
81F4: 04 82    >147           TSB   VARNAM+1
81F6: 20 53 E0 >148           JSR   $E053      Attention, il faudra chg.
81F9: 68       >149           PLA
81FA: 60       >150           RTS
               >151
               >179
                606
81FB: BD 55 9B  607  XFRMMOT1 LDA   TYPLET-´A´,X
                608  XFROMMOT
                610  * X=0 for ´%´, 1 for ´$´ and 2 for ´!´, 3 for ´.´
81FE: A2 03     611           LDX   #TITVAL-MOTIF-1
8200: DD 82 9B  615  ]LOOP    CMP   MOTIF,X
8203: F0 03     616           BEQ   :0
8205: CA        617           DEX
8206: 10 F8     618           BPL   ]LOOP
```

```
8208: 60        619  :0       RTS
                620
                621  * Decrement TXTPTR
8209: A5 B8     622  DECTPTR  LDA   TXTPTR
820B: D0 02     623           BNE   :0
820D: C6 B9     624           DEC   TXTPTR+1
820F: C6 B8     625  :0       DEC   TXTPTR
8211: 60        626           RTS
                627
                628  * Subroutine to patch CHRGET/CHRGOT in page zero
8212: A9 4C     629  SETUPB   LDA   #$4C       JMP absolute
8214: 85 B1     630           STA   $B1
8216: 85 BA     631           STA   $BA
8218: A9 09     632           LDA   #DEBUTGET
821A: 85 B2     632           STA   $B2
821C: A9 76     632           LDA   #>DEBUTGET
821E: 85 B3     632           STA   $B2+1
8220: A9 58     633           LDA   #DEBUTGOT
8222: 85 BB     633           STA   $BB
8224: A9 76     633           LDA   #>DEBUTGOT
8226: 85 BC     633           STA   $BB+1
8228: 60        634           RTS
                635
                636  SETUPD   STID  BANCLD;$9D72
8229: A9 34     636           LDA   #BANCLD
822B: 8D 72 9D  636           STA   $9D72
822E: A9 82     636           LDA   #>BANCLD
8230: 8D 73 9D  636           STA   $9D72+1
8233: 60        637           RTS
                638
                639  * Subr. called upon a BASIC cold boot (FP DOS command)
8234: A2 FF     640  BANCLD   LDX   #$FF
8236: 86 76     641           STX   $76
8238: A2 FB     642           LDX   #$FB
823A: 9A        643           TXS
823B: A9 28     644           LDA   #$28
823D: A0 F1     645           LDY   #$F1
823F: 85 01     646           STA   1
8241: 84 02     647           STY   2
8243: 85 04     648           STA   4
8245: 84 05     649           STY   5
8247: 20 73 F2  650           JSR   $F273
824A: A9 4C     651           LDA   #$4C       JMP absolute
824C: 85 00     652           STA   0
824E: 85 03     653           STA   3
8250: 85 90     654           STA   $90
8252: 85 0A     655           STA   $A
8254: A9 99     656           LDA   #$99
8256: A0 E1     657           LDY   #$E1
8258: 85 0B     658           STA   $B
825A: 84 0C     659           STY   $C
825C: 20 12 82  660           JSR   SETUPB     Install CHRGET/CHRGOT patch in pa
ge zero
825F: 4C 5C F1  661           JMP   $F15C      End of initialization in ROM
                662
                663  * Do the DOS init
                664  NOUVIN   STID  $E000;$9D72
```

```
8262: A9 00     664           LDA   #$E000
8264: 8D 72 9D  664           STA   $9D72
8267: A9 E0     664           LDA   #>$E000
8269: 8D 73 9D  664           STA   $9D72+1
826C: A9 4C     665           LDA   #$4C       JMP absolute
826E: 8D C8 A2  666           STA   $A2C8
8271: A9 0B     667           LDA   #$B
8273: 20 AA A2  668           JSR   $A2AA
8276: A9 20     669           LDA   #$20
8278: 8D C8 A2  670           STA   $A2C8
827B: A5 45     671           LDA   OPRND+1
827D: D0 06     672           BNE   :4         No error during DoClose
827F: 20 29 82  673           JSR   SETUPD     Reinstall Peersoft
8282: 4C C8 A6  674           JMP   $A6C8       before exiting
8285: A2 60     675  :4       LDX   #$60
8287: 8E E7 A2  676           STX   $A2E7
828A: 20 D2 A2  677           JSR   $A2D2      Copy file manager parmlist
828D: A9 4C     678           LDA   #$4C       JMP absolute
828F: 8D E7 A2  679           STA   $A2E7
8292: AD 00 9D  680           LDA   DBUFP
8295: 8D AD 82  681           STA   E06+1
8298: AD 01 9D  682           LDA   DBUFP+1
829B: 8D B2 82  683           STA   E06+6
829E: A9 D3     684           LDA   #$9CD3
82A0: 8D 00 9D  684           STA   DBUFP
82A3: A9 9C     684           LDA   #>$9CD3
82A5: 8D 01 9D  684           STA   DBUFP+1
82A8: 20 06 AB  685           JSR   $AB06      File manager main entry (INIT)
82AB: 08        686           PHP              ;Save status
                687  E06      STID  0;DBUFP    Reinstall Peersoft DOS features
82AC: A9 00     687           LDA   #0
82AE: 8D 00 9D  687           STA   DBUFP
82B1: A9 00     687           LDA   #>0
82B3: 8D 01 9D  687           STA   DBUFP+1
82B6: 20 29 82  688           JSR   SETUPD
82B9: 28        689           PLP
82BA: 20 EB A6  690           JSR   $A6EB      process possible error after FM c
all
82BD: 4C 97 A3  691           JMP   $A397      Goto SAVE (HELLO) command handler
                692
                693  * RECON is a subroutine which scans BASIC program area
                694  * or input buffer for a Peersoft new keyword
                695  * 2 entry points:
                696  * RECON1 (BASIC statement execution): the pointer is TXTPTR
                697  * RECON (BASIC statement listing): the pointer is in A,Y
                698  * X value of 0: search for every new keyword (LIST)
                699  *            1: search only DEF patterns
                700  *            2: search only function statements
                701  *               (IIF, MOUSE and TIMER)
                702  *            3: search only MOUSE and TIMER keywords
                703  * On exit, Z bit set means no keyword found
                704  *                clear means keyword (index in IDMOCL)
82C0: A5 B8     705  RECON1   LDA   TXTPTR
82C2: A4 B9     706           LDY   TXTPTR+1
82C4: 85 06     707  RECON    STA   AUXPTR
82C6: 84 07     708           STY   AUXPTR+1
82C8: BD 70 9B  709  RECON2   LDA   TIDMOCL,X
```

```
82CB: 85 BD     710           STA   IDMOCL
82CD: BD 76 9B  711           LDA   TOFFIN,X
82D0: 8D 42 9B  712           STA   IFDEF
82D3: BD 7C 9B  713           LDA   TOFFIN2,X
82D6: 8D 33 9B  714           STA   IFIIF
82D9: E6 BD     715  :1       INC   IDMOCL
82DB: A4 BD     716           LDY   IDMOCL
82DD: BE 5F 9B  717           LDX   TOFFST,Y
82E0: 86 C2     718           STX   OFFSET
82E2: A0 00     719           LDY   #0
82E4: BD 1F 9B  720  ]LOOP    LDA   TMOCL,X
82E7: F0 0C     721           BEQ   :4          Keyword found: exit
82E9: C9 FF     722           CMP   #$FF        End of table?
82EB: F0 08     723           BEQ   :4          Yes: no keyword found
82ED: D1 06     724           CMP   (AUXPTR),Y Current character match?
82EF: D0 E8     725           BNE   :1          no: try next keyword from table
82F1: E8        726           INX               ;Next char. from current keyword
82F2: C8        727           INY
82F3: D0 EF     728           BNE   ]LOOP
                729
                730  :4       DO    KOPT-K65C02
82F5: 1A        734           INC
82F6: 60        736  RETURN    RTS
                737
                738           PUT   PEERLIST,D1
82F7: 90 0A    >1    STDLIS   BCC   STRTRNG
               >2
82F9: F0 08    >3             BEQ   STRTRNG
82FB: C9 C9    >4             CMP   #TOKMINUS
82FD: F0 04    >5             BEQ   STRTRNG
82FF: C9 2C    >6             CMP   #´,´
8301: D0 F3    >7             BNE   RETURN
               >8
8303: 20 C1 93 >9    STRTRNG  JSR   DECOMPILE
8306: 20 0C DA >10             JSR   LINGET
8309: 20 1A D6 >11             JSR   FNDLIN
830C: 20 32 76 >12             JSR   RST102
830F: F0 10    >13             BEQ   MAINLIST
8311: C9 C9    >14             CMP   #TOKMINUS
8313: F0 04    >15             BEQ   ENDRNG
8315: C9 2C    >16             CMP   #´,´
8317: D0 DD    >17             BNE   RETURN
               >18
8319: 20 2A 76 >19   ENDRNG    JSR   RST100
831C: 20 0C DA >20             JSR   LINGET
831F: D0 D5    >21             BNE   RETURN
               >22
8321: 68       >23   MAINLIST PLA
8322: 68       >24             PLA
8323: A5 50    >25             LDA   LINNUM     In case no second line given,
8325: 05 51    >26             ORA   LINNUM+1    let it be 65535
8327: D0 04    >27             BNE   NXLST
8329: C6 50    >28             DEC   LINNUM
832B: C6 51    >29             DEC   LINNUM+1
               >30
832D: A0 01    >31   NXLST     LDY   #1
832F: B1 9B    >32             LDA   (LOWTR),Y
```

```
8331: F0 6B    >33            BEQ   LISTED     End of program found
8333: 20 58 D8 >34            JSR   ISCNTC     Check for Ctrl-C keystroke
8336: 20 FB DA >35            JSR   CRDO
8339: C8       >36            INY
833A: B1 9B    >37            LDA   (LOWTR),Y  Line number in X,A
833C: AA       >38            TAX
833D: C8       >39            INY
833E: B1 9B    >40            LDA   (LOWTR),Y
8340: C5 51    >41            CMP   LINNUM+1   Beyond last line number?
8342: D0 04    >42            BNE   LSTD?
8344: E4 50    >43            CPX   LINNUM
8346: F0 02    >44            BEQ   LST1LIN
8348: B0 54    >45   LSTD?    BCS   LISTED     Yes
               >46
834A: 84 85    >47   LST1LIN  STY   $85
834C: 64 BE    >55            STZ   MODREM
834E: 64 BF    >56            STZ   MODDAT
8350: 64 C0    >57            STZ   GFLAG
8352: 64 C1    >58            STZ   DEFFLG
8354: 20 A4 83 >60            JSR   VLINPRT    Print line #
8357: A9 20    >61   ]JLOOP   LDA   #32        Print space after line number
8359: A4 85    >62            LDY   $85
835B: 2C       >63            HEX   2C
835C: A9 2D    >64   L088     LDA   #´-´
835E: C9 22    >65   L08      CMP   #´"´       Is it ´"´?
8360: D0 08    >66            BNE   :9
8362: A5 C0    >67            LDA   GFLAG
8364: 49 FF    >68            EOR   #$FF
8366: 85 C0    >69            STA   GFLAG
8368: A9 22    >70            LDA   #´"´
               >71   * Now we test for an EOI
836A: 24 BE    >72   :9       BIT   MODREM     If a REM has been scanned in this
 line
836C: 30 0C    >73            BMI   SENDCHR
836E: 24 C0    >74            BIT   GFLAG      Are we within a string litteral?
8370: 30 08    >75            BMI   SENDCHR    Same output as for a REM
8372: C9 3A    >76            CMP   #´:´       Current char is EOI?
8374: D0 04    >77            BNE   SENDCHR
8376: 85 BF    >78            STA   MODDAT     MODDAT b7 forced to zero
8378: 85 C1    >79            STA   DEFFLG     DEFFLG b7 forced to zero
837A: 20 5C DB >80   SENDCHR  JSR   OUTDO      Print current char
837D: A5 24    >81            LDA   CH
837F: C9 21    >82            CMP   #33        Have we reached "right" edge of s
creen?
8381: 90 07    >83            BCC   NCR        No
8383: 20 FB DA >84            JSR   CRDO       Yes: print CR for next line
8386: A9 05    >85            LDA   #5
8388: 85 24    >86            STA   CH
               >87   * Next character from line
838A: C8       >88   NCR      INY
838B: B1 9B    >89            LDA   (LOWTR),Y
838D: D0 18    >90            BNE   TOKEN?     Not end of line
838F: 85 C1    >91            STA   DEFFLG
8391: B2 9B    >98            LDA   (LOWTR)    Update next line pointer
8393: AA       >99            TAX
8394: A0 01    >100           LDY   #1
8396: B1 9B    >102           LDA   (LOWTR),Y
```

```
8398: 86 9B    >103           STX   LOWTR
839A: 85 9C    >104           STA   LOWTR+1
839C: D0 8F    >105           BNE   NXLST      Branch if not at program´s end
               >106
839E: 20 FB DA >107  LISTED   JSR   CRDO
83A1: 4C D2 D7 >108           JMP   NEWSTT
83A4: 6C FA D6 >109  VLINPRT  JMP   ($D6FA)
83A7: AA       >110  TOKEN?   TAX                ;Character in X
83A8: A5 BE    >111           LDA   MODREM     Is litteral mode active?
83AA: 05 BF    >112           ORA   MODDAT
83AC: 05 C0    >113           ORA   GFLAG
83AE: 0A       >114           ASL
83AF: 8A       >115           TXA
83B0: B0 AC    >116           BCS   L08        Yes
83B2: 84 B5    >117           STY   YSAV
83B4: 98       >118           TYA                ;Compute Y, A = LOWTR + Y
83B5: A4 9C    >119           LDY   LOWTR+1
83B7: 65 9B    >120           ADC   LOWTR      Carry already clear
83B9: 90 01    >121           BCC   :14
83BB: C8       >122           INY
83BC: A2 00    >123  :14      LDX   #0
83BE: 20 C4 82 >124           JSR   RECON      New BASIC keyword?
83C1: D0 33    >125           BNE   :23        Yes
               >126
83C3: A4 B5    >127           LDY   YSAV       Y = offset within line
83C5: B1 9B    >128           LDA   (LOWTR),Y  Current character
83C7: 10 95    >129           BPL   L08        Not a token
83C9: 24 C1    >130           BIT   DEFFLG
83CB: 10 04    >131           BPL   :18
83CD: C9 C9    >132           CMP   #TOKMINUS
83CF: F0 8B    >133           BEQ   L088
83D1: C9 B2    >134  :18      CMP   #TOKREM    REM token?
83D3: D0 02    >135           BNE   :15
83D5: 66 BE    >136           ROR   MODREM     bit 7 to 1 in MODREM
83D7: C9 83    >137  :15      CMP   #TOKDATA   DATA token?
83D9: D0 02    >138           BNE   :16
83DB: 66 BF    >139           ROR   MODDAT     bit 7 to 1 in MODDAT
83DD: 48       >140  :16      PHA
83DE: 20 57 DB >141           JSR   OUTSPC
83E1: 68       >142           PLA
83E2: 48       >143           PHA
83E3: 20 44 84 >144           JSR   LTOKEN     Print Applesoft token
83E6: 68       >145           PLA
83E7: C9 D5    >146           CMP   #TOKUSR
83E9: 20 34 84 >147           JSR   COMLISO
83EC: B0 05    >148           BCS   :17
83EE: 84 85    >149           STY   $85
83F0: 20 5C DB >150           JSR   OUTDO
83F3: 4C 57 83 >151  :17      JMP   ]JLOOP
               >152  * LIST a new BASIC statement
83F6: 88       >153  :23      DEY
83F7: A5 BD    >154           LDA   IDMOCL
83F9: C9 0B    >155           CMP   #OFFDEF-TOFFST
83FB: 90 03    >156           BCC   :39
83FD: 66 C1    >157           ROR   DEFFLG
83FF: 18       >158           CLC
8400: 98       >159  :39      TYA
```

```
8401: 65 B5    >160           ADC   YSAV
8403: 85 B5    >161           STA   YSAV
8405: 20 57 DB >162           JSR   OUTSPC
8408: A6 C2    >163           LDX   OFFSET     Get offset from new keyword table
840A: BD 1F 9B >164  ]LOOP    LDA   TMOCL,X
840D: F0 11    >165           BEQ   :29        End of keyword
840F: 30 05    >166           BMI   :27        Applesoft token: print it
8411: 20 5C DB >167           JSR   OUTDO      Normal text to output
8414: D0 07    >168           BNE   :28        Always
8416: 86 B4    >169  :27      STX   XSAV       Save offset
8418: 20 44 84 >170           JSR   LTOKEN     Print Applesoft token
841B: A6 B4    >171           LDX   XSAV
841D: E8       >172  :28      INX
841E: D0 EA    >173           BNE   ]LOOP      Always
8420: A5 BD    >174  :29      LDA   IDMOCL
8422: C9 0A    >175           CMP   #OFFUSR-TOFFST
8424: 20 34 84 >176           JSR   COMLISO
8427: B0 03    >177           BCS   :30
8429: 20 5C DB >178           JSR   OUTDO
842C: 20 57 DB >179  :30      JSR   OUTSPC
842F: A4 B5    >180  :31      LDY   YSAV
8431: 4C 8A 83 >181           JMP   NCR
               >182
8434: 38       >183  COMLISO  SEC
8435: D0 0C    >184           BNE   :0
8437: A4 B5    >185           LDY   YSAV
8439: C8       >186           INY
843A: B1 9B    >187           LDA   (LOWTR),Y
843C: 20 3A 76 >188           JSR   COMRSTC
843F: B0 02    >189           BCS   :0
8441: 84 B5    >190           STY   YSAV
8443: 60       >191  :0       RTS
               >192
               >193  * Print Applesoft token
8444: 38       >194  LTOKEN   SEC
8445: E9 7F    >195           SBC   #$7F
8447: AA       >196           TAX             ;Index in X reg
8448: 84 85    >197           STY   $85
844A: A0 D0    >198           LDY   #TOKTABL-256
844C: 84 9D    >199           STY   FAC
               >200  * Line below is a substitute for LDY #>TOKTABL-256
844E: 88       >201           DEY
844F: 84 9E    >202           STY   FAC+1
8451: A0 FF    >203           LDY   #$FF
8453: CA       >204  :1       DEX
8454: F0 07    >205           BEQ   :3
8456: 20 2C D7 >206  ]LOOP    JSR   $D72C
8459: 10 FB    >207           BPL   ]LOOP
845B: 30 F6    >208           BMI   :1
845D: 20 2C D7 >209  :3       JSR   $D72C
8460: 30 05    >210           BMI   :4
8462: 20 5C DB >211           JSR   OUTDO
8465: D0 F6    >212           BNE   :3
8467: A4 85    >213  :4       LDY   $85
8469: 4C 5C DB >214           JMP   OUTDO
                739
846C: D0 07     740  RRETURN  BNE   :0
```

```
846E: A9 FF     741           LDA   #$FF
8470: 85 86     742           STA   FORPNT+1
8472: 4C 71 D9  743           JMP   $D971
8475: 60        744  :0       RTS
                745
8476: A9 AB     746  RONERR   LDA   #TOKGOTO
8478: 20 76 7E  747           JSR   NSYNCHR
847B: A5 B8     748           LDA   TXTPTR
847D: 85 F4     749           STA   TXTPSV
847F: A5 B9     750           LDA   TXTPTR+1
8481: 85 F5     751           STA   TXTPSV+1
8483: 38        752           SEC
8484: 66 D8     753           ROR   ERRFLG
8486: A5 75     754           LDA   CURLIN
8488: 85 F6     755           STA   CURLSV
848A: A5 76     756           LDA   CURLIN+1
848C: 85 F7     757           STA   CURLSV+1
848E: 4C 95 D9  758           JMP   DATA
                759
                760  * New FRMEVL processing
                761           PUT   PEERAROMBA,D2
               >1    TOKDIM   =     $86
               >2    TOKFRE   =     $D6
               >3    NEWGARBG EQU   $E484
               >4    FREFAC   EQU   $E600
               >5    ENDCHR   EQU   $0E
               >6    STRNG1   EQU   $AC
               >7    VPNT     EQU   $A0
               >8    * When used in USR functions w 2 args, holdsin n
               >9    * the first arg expression type
               >10   GIVAYF   EQU   $E2F2
               >11   SNGFLT   EQU   $E301
               >12   MOVMF    EQU   $EB2B
               >13   LEVELPAR EQU   IDMOCL
               >14
8491: 20 2A 76 >85   RDIM     JSR   RST100
8494: 20 A9 86 >86            JSR   NCHKOPN
8497: 20 02 7A >87            JSR   NGETARPT
849A: A0 04    >88            LDY   #4
849C: B1 9B    >89            LDA   (LOWTR),Y
849E: 29 0F    >90            AND   #$0F
84A0: 48       >91            PHA
84A1: B2 B8    >93            LDA   (TXTPTR)
84A3: C9 2C    >98            CMP   #´,´
84A5: D0 29    >99            BNE   :1
84A7: A5 9C    >103           LDA   LOWTR+1
84A9: 48       >104           PHA
84AA: A5 9B    >105           LDA   LOWTR
84AC: 48       >106           PHA
84AD: 20 2A 76 >108           JSR   RST100
84B0: 20 D0 86 >109           JSR   NGETBYT    Index of dimension in X&FACLO
84B3: 8A       >110           TXA
84B4: F0 24    >111           BEQ   GOIQ
84B6: 68       >112           PLA
84B7: 85 9B    >113           STA   LOWTR
84B9: 68       >114           PLA
84BA: 85 9C    >115           STA   LOWTR+1
```

```
84BC: 68       >116           PLA
84BD: 38       >117           SEC
84BE: E5 A1    >118           SBC   FACLO
84C0: 90 18    >119           BCC   GOIQ
84C2: 0A       >120           ASL             ;Incidently clears the carry
84C3: 69 05    >121           ADC   #5         Because of carry clear
84C5: A8       >122           TAY
84C6: B1 9B    >123           LDA   (LOWTR),Y
84C8: AA       >124           TAX
84C9: C8       >125           INY
84CA: B1 9B    >126           LDA   (LOWTR),Y
84CC: A8       >127           TAY
84CD: 8A       >128           TXA
84CE: 90 04    >129           BCC   :0         Always
               >130  :1        MPLY
84D0: 7A       >130           PLY
84D1: A9 00    >132           LDA   #0
84D3: 38       >136           SEC
84D4: 20 F2 E2 >137  :0        JSR   GIVAYF
84D7: 4C A3 86 >138           JMP   NCHKCLS
               >139
84DA: 4C 99 E1 >140  GOIQ      JMP   GOIQERR    Raise a ILLEGAL QUANTITY ERROR
               >141
84DD: 20 AE 86 >142  RVRAI     JSR   NFRMEVL    True: evaluate second argument
84E0: 20 A6 86 >143           JSR   NCHKCOM    Skip the comma and 3rd expr.
84E3: A9 29    >144           LDA   #´)´       until end of function detected
               >145
               >146  * This subroutine will skip program text until an
               >147  * end character is scanned.
84E5: 85 0E    >148  SKIPC     STA   ENDCHR
84E7: A0 00    >149           LDY   #0
84E9: 84 BD    >150           STY   LEVELPAR   Parenthesis level
84EB: 84 C0    >151           STY   GFLAG      String litteral parsing flag
84ED: 88       >152           DEY
84EE: C8       >153  ]LOOP     INY
84EF: B1 B8    >154           LDA   (TXTPTR),Y
84F1: F0 36    >155           BEQ   LGSYNERR
84F3: C9 22    >156           CMP   #´"´
84F5: D0 08    >157           BNE   :0
84F7: A5 C0    >158           LDA   GFLAG      Inverse GFLAG b7
84F9: 49 80    >159           EOR   #$80
84FB: 85 C0    >160           STA   GFLAG
84FD: B0 EF    >161           BCS   ]LOOP      Always
84FF: 24 C0    >162  :0        BIT   GFLAG      Within litteral string
8501: 30 EB    >163           BMI   ]LOOP       so loop for next character.
8503: C9 3A    >164           CMP   #´:´       End of instruction?
8505: F0 22    >165           BEQ   LGSYNERR   SYNTAX ERROR if so
8507: C9 28    >166           CMP   #´(´
8509: D0 04    >167           BNE   :1
850B: E6 BD    >168           INC   LEVELPAR
850D: B0 DF    >169           BCS   ]LOOP      Always
850F: C9 29    >170  :1        CMP   #´)´
8511: D0 08    >171           BNE   :2
8513: A6 BD    >172           LDX   LEVELPAR
8515: F0 08    >173           BEQ   :3
8517: C6 BD    >174           DEC   LEVELPAR
8519: 10 D3    >175           BPL   ]LOOP
```

```
851B: A6 BD    >176  :2       LDX   LEVELPAR
851D: D0 CF    >177          BNE   ]LOOP
851F: C5 0E    >178  :3       CMP   ENDCHR
8521: D0 CB    >179          BNE   ]LOOP
8523: 20 98 D9 >180          JSR   ADDON      Add Y to TXTPTR
8526: 4C 2A 76 >181          JMP   RST100
               >182
8529: 4C C9 DE >183  LGSYNERR JMP   SYNERR     Vector to SYNTAX ERROR
               >184
               >185  * Handles the IIF function
852C: 20 A6 86 >186  RIIF     JSR   NCHKCOM    Check for trailing comma
852F: A6 9D    >187          LDX   FAC        True or false value?
8531: D0 AA    >188          BNE   RVRAI      True: then skip second arg.
8533: A9 2C    >189          LDA   #´,´
8535: 20 E5 84 >190          JSR   SKIPC      Skip 2nd expression
               >191  * Evaluate 3rd arg. and check for closing parenthesis
8538: 4C A0 86 >192          JMP   NPARCHK+3
               >193
853B: 20 AE 86 >194  NFRMNUM  JSR   NFRMEVL    Get scalar valueH
853E: 4C 6A DD >195          JMP   CHKNUM     Ensure numeric value
               >196
8541: 4C F9 EA >197  ]LOOP    JMP   MOVFM
8544: B2 A0    >203  H16B     LDA   (VPNT)
8546: 48       >204          PHA
8547: 20 E9 DE >205          JSR   $DEE9
854A: 68       >213          PLA
854B: 20 C5 86 >214          JSR   LBS81
854E: 4C BF 85 >215          JMP   XSUITE
               >216
               >217  * Takes care of the ´@´ processing
               >218  * Refactor part of the FRMEVL ROM routine
8551: 20 2A 76 >219  FRMELMLP JSR   RST100
8554: B0 07    >220  FRMELM   BCS   :2         Branch iif not a digit
8556: 64 C7    >228  :1       STZ   INTTYPSV
8558: 64 C8    >229          STZ   VALTYPSV
855A: 4C 4A EC >230          JMP   $EC4A
855D: C9 2E    >232  :2       CMP   #´.´
855F: F0 F5    >233          BEQ   :1
8561: 20 7D E0 >234          JSR   ISLETC
8564: 90 5E    >235          BCC   L3
8566: AA       >236          TAX
8567: 30 28    >237          BMI   :77
8569: C9 49    >238          CMP   #´I´
856B: F0 08    >239          BEQ   :80
856D: C9 4D    >240          CMP   #´M´
856F: F0 04    >241          BEQ   :80
8571: C9 54    >242          CMP   #´T´
8573: D0 1C    >243          BNE   :77
               >244  * Might be the IIF() function
8575: A2 02    >245  :80      LDX   #2
8577: 20 C0 82 >246          JSR   RECON1
857A: F0 15    >247          BEQ   :77
857C: 20 98 D9 >248          JSR   ADDON
857F: A5 BD    >249          LDA   IDMOCL
8581: 48       >250          PHA
8582: 20 A9 86 >251          JSR   NCHKOPN
8585: 20 3B 85 >252          JSR   NFRMNUM    Get operand numeric value
```

```
8588: 68       >253           PLA             ;Recall IDMOCL from stack
8589: 38       >254           SEC
858A: E9 08    >255           SBC   #OFFMOU-TOFFST
858C: 90 9E    >256           BCC   RIIF
               >257  * Space for MOUSE and TIMER functions
               >258  * ...: to be continued
858E: 4C 9E 8C >259           JMP   MTFUNC
               >260  * Alphabetic character: variable name
8591: A2 00    >261  :77       LDX   #0
8593: 86 10    >262           STX   DIMFLG
8595: B2 B8    >266           LDA   (TXTPTR)
8597: 20 0E 7A >268           JSR   NPTRGET1
               >269  RFFVL     EQU   *-1
859A: 85 A0    >270           STA   VPNT
859C: 84 A1    >271           STY   VPNT+1
859E: A6 11    >272           LDX   VALTYP
85A0: F0 04    >273           BEQ   :41
85A2: 64 AD    >279           STZ   STRNG1+1
85A4: D0 19    >280           BNE   XSUITE    Always
85A6: A6 12    >282  :41       LDX   INTTYP
85A8: 10 97    >283           BPL   ]LOOP
85AA: E0 81    >284           CPX   #$81
85AC: D0 96    >285           BNE   H16B      Branch if int16bit variable
85AE: A2 00    >286           LDX   #0
85B0: B2 83    >288           LDA   (VARPNT)
85B2: 10 06    >292           BPL   *+8
85B4: 2C E7 9C >293           BIT   WMODE
85B7: 30 01    >294           BMI   *+3
85B9: CA       >295           DEX             ;Poids fort dans X
85BA: A8       >296           TAY             ;Poids faible dans Y
85BB: 8A       >297           TXA             ;Poids fort dans A
85BC: 20 F2 E2 >298           JSR   GIVAYF    Convert A, Y to FP
85BF: A5 11    >299  XSUITE    LDA   VALTYP
85C1: 85 C8    >300  RET3      STA   VALTYPSV
85C3: 60       >301  ]RET      RTS
               >302
85C4: C9 C8    >303  L3        CMP   #TOKADD   Unary + operator: loop
85C6: F0 89    >304           BEQ   FRMELMLP
85C8: C9 22    >305           CMP   #´"´
85CA: D0 0A    >306           BNE   :4
85CC: 20 81 DE >307           JSR   $DE81
85CF: A9 FF    >308           LDA   #$FF
85D1: 30 EE    >309           BMI   RET3      Always
85D3: 4C 66 7F >310  ]LOOP     JMP   RUSR
85D6: C9 D5    >311  :4        CMP   #TOKUSR
85D8: F0 F9    >312           BEQ   ]LOOP
85DA: A2 03    >313           LDX   #TOKMTIFE-TOKMOTIF-1
85DC: DD 07 96 >314  ]LOOP     CMP   TOKMOTIF,X
85DF: D0 08    >315           BNE   :NOK
85E1: A8       >325           TAY
85E2: 8A       >326           TXA
85E3: 0A       >327           ASL
85E4: AA       >328           TAX
85E5: 98       >329           TYA
85E6: 7C 0B 96 >330           JMP   (TOKMPF,X)
85E9: CA       >332  :NOK      DEX
85EA: 10 F0    >333           BPL   ]LOOP
```

```
85EC: C9 40    >334  :6       CMP   #´@´
85EE: D0 10    >335          BNE   :78
85F0: A5 C8    >336          LDA   VALTYPSV
85F2: 85 11    >337          STA   VALTYP
85F4: 30 04    >338          BMI   :60
85F6: A5 C7    >339          LDA   INTTYPSV
85F8: 85 12    >340          STA   INTTYP
85FA: 4C 2A 76 >341  :60      JMP   RST100
85FD: 4C 91 84 >342  :79      JMP   RDIM
8600: C9 86    >343  :78      CMP   #TOKDIM
8602: F0 F9    >344          BEQ   :79
               >345
8604: C9 D2    >346  :7       CMP   #TOKSGN
8606: B0 18    >347          BCS   :10
8608: C9 23    >348          CMP   #´#´
860A: F0 03    >349          BEQ   *+5
860C: 4C 9D 86 >350          JMP   NPARCHK
               >351  * Handle the ´#´ pattern in a FOREACH loop
860F: AC 23 96 >352          LDY   AEI
8612: AD 24 96 >353          LDA   AEI+1
8615: 48       >357          PHA
8616: 20 F2 E2 >359          JSR   GIVAYF
8619: 68       >363          PLA
861A: 20 C8 86 >365          JSR   LBS80
861D: 4C 2A 76 >366          JMP   RST100
8620: 0A       >367  :10      ASL
8621: 48       >368          PHA
8622: AA       >369          TAX
8623: 20 2A 76 >370          JSR   RST100
8626: E0 CF    >371          CPX   #$CF
8628: 90 12    >372          BCC   :11
862A: 20 A9 86 >373          JSR   NCHKOPN
862D: 20 AE 86 >374          JSR   NFRMEVL
8630: 20 A6 86 >375          JSR   NCHKCOM
8633: 20 6C DD >376          JSR   CHKSTR
8636: FA       >377          PLX
8637: 20 83 86 >378          JSR   COMCMPLX
863A: 80 0F    >382          BRA   :14
863C: 20 9D 86 >384  :11      JSR   NPARCHK
863F: 7A       >385          PLY
8640: C0 C8    >386          CPY   #TOKSTRD+TOKSTRD
8642: F0 04    >387          BEQ   :15
8644: C0 CE    >388          CPY   #TOKCHRD+TOKCHRD
8646: D0 31    >389          BNE   :13
8648: 20 8F 86 >390  :15      JSR   CALLFUNC
864B: A9 FF    >391  :14      LDA   #$FF
864D: 85 C8    >392          STA   VALTYPSV
864F: 60       >393  ]RET     RTS
8650: A5 11    >394  ]LOOP    LDA   VALTYP
8652: D0 1C    >395          BNE   :19
8654: 18       >396          CLC
8655: 20 4E 78 >397          JSR   NROUT
8658: A2 00    >398          LDX   #0
865A: A5 A0    >399          LDA   FAC+3
865C: D0 15    >400          BNE   :2
865E: A5 A1    >401          LDA   FAC+4
8660: C9 01    >402          CMP   #1
```

```
8662: D0 0F    >403           BNE   :2
8664: A2 03    >404           LDX   #3
8666: 20 CB 7D >405           JSR   ZRTAUX
8669: A5 AE    >406           LDA   STRNG2+1
866B: A4 AD    >407           LDY   STRNG2
866D: 4C C0 86 >408           JMP   NWGVAYF
8670: 20 00 E6 >409  :19      JSR   FREFAC
8673: 20 84 E4 >410  :2       JSR   NEWGARBG
8676: 4C B6 86 >411           JMP   HE2E8
               >412
8679: C0 AC    >413  :13      CPY   #TOKFRE+TOKFRE
867B: F0 D3    >414           BEQ   ]LOOP
867D: 20 8F 86 >415           JSR   CALLFUNC
8680: 4C 6A DD >416           JMP   CHKNUM
               >417
               >418  COMCMPLX DO    KOPT16
8683: A5 A1    >421           LDA   FACLO
8685: 48       >422           PHA
8686: A5 A0    >423           LDA   FACMO
8688: 48       >424           PHA
8689: DA       >426           PHX
868A: 20 D0 86 >427           JSR   NGETBYT
868D: 7A       >428           PLY
868E: DA       >429           PHX
               >430
868F: B9 DC CF >431  CALLFUNC LDA   $CFDC,Y
8692: 85 91    >432           STA   $91
8694: B9 DD CF >433           LDA   $CFDD,Y
8697: 85 92    >434           STA   $92
8699: 20 90 00 >435           JSR   $90
869C: 60       >436           RTS
               >437
869D: 20 A9 86 >438  NPARCHK  JSR   NCHKOPN
86A0: 20 AE 86 >439           JSR   NFRMEVL
               >440
86A3: A9 29    >441  NCHKCLS  LDA   #´)´
86A5: 2C       >442           HEX   2C
86A6: A9 2C    >443  NCHKCOM  LDA   #´,´
86A8: 2C       >444           HEX   2C
86A9: A9 28    >445  NCHKOPN  LDA   #´(´
86AB: 4C 76 7E >446           JMP   NSYNCHR
               >447
86AE: 20 7B DD >448  NFRMEVL  JSR   FRMEVL
86B1: A5 11    >449           LDA   VALTYP
86B3: 85 C8    >450           STA   VALTYPSV
86B5: 60       >451  ]RET     RTS
               >452
86B6: 38       >453  HE2E8    SEC
86B7: A5 6F    >454           LDA   FRETOP
86B9: E5 6D    >455           SBC   STREND
86BB: A8       >456           TAY
86BC: A5 70    >457           LDA   FRETOP+1
86BE: E5 6E    >458           SBC   STREND+1
86C0: 48       >459  NWGVAYF  PHA
86C1: 20 F2 E2 >460           JSR   GIVAYF
86C4: 68       >461           PLA
86C5: 2D E7 9C >462  LBS81    AND   WMODE
```

```
86C8: 10 EB    >463  LBS80    BPL   ]RET
86CA: 20 95 8E >464           JSR   GP65536
86CD: 4C BE E7 >465           JMP   FADD
               >466
86D0: 20 F8 E6 >467  NGETBYT  JSR   GETBYT
86D3: 48       >468           PHA
86D4: 20 D8 77 >469           JSR   SETITS
86D7: 64 C8    >474           STZ   VALTYPSV
86D9: 68       >476           PLA
86DA: 60       >477  MFIN     RTS
                762
86DB: 20 4C E7  763  ROUT11   JSR   COMBYTE    Get VTAB value in X
86DE: 20 59 F2  764           JSR   $F259      Do the VTAB
86E1: 20 4C E7  765           JSR   COMBYTE
86E4: 20 EA F7  766           JSR   $F7EA      Do the HTAB
86E7: 20 32 76  767           JSR   RST102
86EA: F0 13     768           BEQ   :0
86EC: 20 A6 86  769           JSR   NCHKCOM
86EF: A5 F1     770           LDA   $F1        Save current SPEED
86F1: 48        771           PHA
86F2: A9 01     772           LDA   #1         Fastest speed..
86F4: 85 F1     773           STA   $F1
86F6: 20 32 76  774           JSR   RST102
86F9: 20 D5 DA  775           JSR   $DAD5      Do the PRINT
86FC: 68        776           PLA              ;restore original SPEED
86FD: 85 F1     777           STA   $F1
86FF: 60        778  :0       RTS
                779
8700: 20 A6 86  780  ROUTGEN  JSR   NCHKCOM
8703: 20 D0 86  781           JSR   NGETBYT
8706: 8A        782           TXA
8707: F0 1F     783           BEQ   ROUT0
8709: E0 0B     784           CPX   #11
870B: F0 CE     785           BEQ   ROUT11
870D: E0 0A     786           CPX   #10
870F: D0 03     787           BNE   :2
8711: 4C B4 8A  788           JMP   ROUT10
8714: E0 08     789  :2       CPX   #8
8716: D0 03     790           BNE   :1
8718: 4C 1B 92  791           JMP   ROUT8
871B: E0 05     792  :1       CPX   #5
871D: D0 03     793           BNE   :0
871F: 4C FD 88  794           JMP   KILLEMAL
8722: B0 B6     795  :0       BCS   MFIN
8724: E0 04     796           CPX   #4
8726: F0 3D     797           BEQ   ROUT4
8728: A5 69     798  ROUT0    LDA   VARTAB
872A: 85 06     799           STA   AUXPTR
872C: A5 6A     800           LDA   VARTAB+1
872E: 85 07     801           STA   AUXPTR+1
                802
8730: 20 32 76  803  ]LOOP    JSR   RST102
8733: F0 A5     804           BEQ   MFIN
8735: 20 A6 86  805           JSR   NCHKCOM
8738: 20 9D 88  806           JSR   NPTRGETX
873B: A5 9B     807           LDA   LOWTR
873D: C5 06     808           CMP   AUXPTR
```

```
873F: A5 9C     809           LDA   LOWTR+1
8741: E5 07     810           SBC   AUXPTR+1
8743: 90 95     811           BCC   MFIN
8745: A0 00     812           LDY   #0
8747: B1 9B     813  ]JLOOP    LDA   (LOWTR),Y
8749: AA        814           TAX
874A: B1 06     815           LDA   (AUXPTR),Y
874C: 91 9B     816           STA   (LOWTR),Y
874E: 8A        817           TXA
874F: 91 06     818           STA   (AUXPTR),Y
8751: C8        819           INY
8752: C0 07     820           CPY   #7
8754: 90 F1     821           BCC   ]JLOOP
8756: 18        822           CLC
8757: 98        823           TYA
8758: 65 06     824           ADC   AUXPTR
875A: 85 06     825           STA   AUXPTR
875C: 90 D2     826           BCC   ]LOOP
875E: E6 07     827           INC   AUXPTR+1
8760: B0 CE     828           BCS   ]LOOP      Always
                829
8762: 4C 76 DD  830  GGO2TMER JMP   GOTMIERR
                831
8765: A9 04     832  ROUT4    LDA   #4         Ensure enough room on stack
8767: 20 D6 D3  833           JSR   CHKMEM     7 bytes so 4 16bit words
876A: 68        834           PLA              ;Pull return adress
876B: 68        835           PLA
876C: 20 A6 86  836           JSR   NCHKCOM
876F: 20 06 7A  837           JSR   NPTRGTX
8772: 24 12     838           BIT   INTTYP
8774: 10 EC     839           BPL   GGO2TMER
8776: A5 9B     840           LDA   LOWTR
8778: C5 6B     841           CMP   ARYTAB
877A: 8D F3 95  842           STA   ITVADDR
877D: A5 9C     843           LDA   LOWTR+1
877F: 8D F4 95  844           STA   ITVADDR+1
8782: E5 6C     845           SBC   ARYTAB+1
8784: B0 DC     846           BCS   GGO2TMER
8786: A5 F8     847           LDA   REMSTK
8788: 8D F2 95  848           STA   SPROOT
                849  * Reinit the alive context markers
878B: A9 FF     850           LDA   #$FF
878D: A2 08     851           LDX   #TABOFT-TABOFB
878F: 9D E8 95  852  ]LOOP    STA   TABOFT-1,X
8792: CA        853           DEX
8793: D0 FA     854           BNE   ]LOOP
8795: 86 C0     855           STX   IDX0       Starting index: 0
8797: 20 32 76  856  ]LOOP    JSR   RST102
879A: F0 0F     857           BEQ   XMFIN      End of instruction
879C: 20 A6 86  858           JSR   NCHKCOM
879F: 20 9A 8E  859           JSR   NGTA2
87A2: 90 30     860           BCC   XMFIN1
87A4: 20 D7 87  861           JSR   LBS04
87A7: E6 C0     862           INC   IDX0
87A9: D0 EC     863           BNE   ]LOOP
                864
87AB: A5 C0     865  XMFIN    LDA   IDX0
```

```
87AD: F0 21     866           BEQ   :0
87AF: A9 80     867           LDA   #$80
87B1: 8D DC 9C  868           STA   MTACTV
87B4: 20 70 89  869           JSR   SETLTR
87B7: 20 D4 87  870           JSR   XMFIN1
87BA: A9 00     878           LDA   #0
87BC: 24 D8     879           BIT   ERRFLG
87BE: 10 01     880           BPL   *+3
87C0: 1A        881           INC
87C1: A0 1A     883           LDY   #26
87C3: 91 9B     884           STA   (LOWTR),Y
87C5: 20 D7 89  885           JSR   SAVERC
87C8: A2 00     886           LDX   #0
87CA: 8E F1 95  887           STX   INDX
87CD: 4C 19 89  888           JMP   RESTOR1
87D0: 60        889  :0       RTS
                890
87D1: 28        891  XMFIN2   PLP
87D2: 68        892           PLA
87D3: 68        893           PLA
87D4: 4C 95 D9  894  XMFIN1   JMP   DATA
                895
                896  * Handle a single entry (index in IDX0)
                897  LBS04
                898  * Array base address in (LOWTR, LOWTR+1)
87D7: A6 C0     899           LDX   IDX0
87D9: A5 9B     900           LDA   LOWTR
87DB: 85 06     901           STA   AUXPTR
87DD: E5 6B     902           SBC   ARYTAB     C already set
87DF: 9D E1 95  903           STA   TABOFB,X
87E2: 08        904           PHP
87E3: A5 9C     905           LDA   LOWTR+1
87E5: 85 07     906           STA   AUXPTR+1
                907  * Is local error handling desired
87E7: 20 A6 86  908           JSR   NCHKCOM
87EA: 20 F8 E6  909           JSR   GETBYT
                910  * Offset 24 for local error handling flag
87ED: A0 1A     911           LDY   #26
87EF: E0 02     912           CPX   #2
87F1: D0 06     913           BNE   :0
87F3: CA        914           DEX
87F4: 24 D8     915           BIT   ERRFLG
87F6: 30 01     916           BMI   :0
87F8: CA        917           DEX
87F9: 8A        918  :0       TXA
87FA: 91 06     919           STA   (AUXPTR),Y
87FC: F0 0E     920           BEQ   :1
87FE: A0 19     921           LDY   #26-1
8800: BE ED 95  922  ]LOOP    LDX   P0OFFSET-8,Y
8803: B5 00     923           LDA   0,X
8805: 91 06     924           STA   (AUXPTR),Y
8807: 88        925           DEY
8808: E0 F4     926           CPX   #TXTPSV
880A: D0 F4     927           BNE   ]LOOP
                928  * Offsets 27 and 28 for swapped in machine code routine
880C: A9 1C     929  :1       LDA   #28
880E: 20 86 88  930           JSR   LBS041
```

```
                        931  * Offsets 29 and 30 for swapped out machine code routine
8811: A9 1E             932          LDA   #30
8813: 20 86 88          933          JSR   LBS041
8816: 20 A6 86          934          JSR   NCHKCOM
8819: 20 0C DA          935          JSR   LINGET
881C: 20 1A D6          936          JSR   FNDLIN
881F: 90 B0             937          BCC   XMFIN2     Non existent line: exit
                        938  * Offsets 0 and 1 for array name
                        939  * Offsets 2 and 3 for offset to next array
                        940  * Offset 4 for number of dimension
                        941  * Offsets 5 and 6 for last dimension value
8821: A0 04             942          LDY   #4
8823: B1 06             943          LDA   (AUXPTR),Y
8825: 49 41             944          EOR   #%01000001 Must be 16bits integer and
8827: D0 A8             945          BNE   XMFIN2     # of dimensions must be 1
8829: A5 07             946          LDA   AUXPTR+1
882B: 28                947          PLP                ;Restaure Carry from previous SBC
882C: E5 6C             948          SBC   ARYTAB+1
882E: A6 C0             949          LDX   IDX0
8830: 9D E9 95          950          STA   TABOFT,X
                        951  * Offset 7 and 8 for storing SP value
                        952  * Integer variable value storage order
8833: A0 07             953          LDY   #7
8835: A9 00             954          LDA   #0
8837: 91 06             955          STA   (AUXPTR),Y
8839: C8                956          INY
883A: A5 F8             957          LDA   REMSTK
883C: E9 07             958          SBC   #7         ;Carry already set
883E: 91 06             959          STA   (AUXPTR),Y
8840: C8                960          INY
                        961  * Offset 9 and 10 for LINNUM storage
                        962  * (natural storage order)
8841: A5 50             963          LDA   LINNUM
8843: 91 06             964          STA   (AUXPTR),Y
8845: C8                965          INY
8846: A5 51             966          LDA   LINNUM+1
8848: 91 06             967          STA   (AUXPTR),Y
884A: C8                968          INY
                        969  * Offset 11 and 12 for TXTPTR storage
                        970  * (natural storage order)
884B: A5 9B             971          LDA   LOWTR
884D: 69 03             972          ADC   #4-1       Because Carry already set
884F: 91 06             973          STA   (AUXPTR),Y
8851: C8                974          INY
8852: A5 9C             975          LDA   LOWTR+1
8854: 69 00             976          ADC   #0
8856: 91 06             977          STA   (AUXPTR),Y
8858: C8                978          INY
                        979  * Offset 13 and 14 for OLDTEXT storage
                        980  * (natural storage order)
8859: A5 9B             981          LDA   LOWTR
885B: 69 04             982          ADC   #4
885D: 91 06             983          STA   (AUXPTR),Y
885F: C8                984          INY
8860: A5 9C             985          LDA   LOWTR+1
8862: 69 00             986          ADC   #0
8864: 91 06             987          STA   (AUXPTR),Y
```

```
8866: A0 1F     988           LDY   #31
                989  * Offsset 31 and above for stack content storage
                990  * from current SP to SPROOT
                991  * For the time being (init), prepare a GOSUB frame
8868: A9 B0     992           LDA   #TOKGOSUB
886A: A2 03     993           LDX   #3
886C: 91 06     994  ]JLOOP   STA   (AUXPTR),Y Do not mind calling CURLIN
886E: C8        995           INY
886F: CA        996           DEX
8870: D0 FA     997           BNE   ]JLOOP
8872: A5 79     998           LDA   OLDTPTR
8874: 91 06     999           STA   (AUXPTR),Y
8876: C8        1000          INY
8877: A5 7A     1001          LDA   OLDTPTR+1
8879: 91 06     1002          STA   (AUXPTR),Y
887B: C8        1003          INY
887C: A9 D1     1004          LDA   #NEWSTT-1
887E: 91 06     1005          STA   (AUXPTR),Y
8880: C8        1006          INY
8881: A9 D7     1007          LDA   #>NEWSTT-1
8883: 91 06     1008          STA   (AUXPTR),Y
8885: 60        1009          RTS
                1010
8886: 48        1011 LBS041   PHA
8887: 20 A6 86  1012          JSR   NCHKCOM
888A: 20 67 DD  1013          JSR   FRMNUM
888D: 20 52 E7  1014          JSR   GETADR
8890: 7A        1015          PLY
8891: A5 51     1016          LDA   LINNUM+1
8893: 91 06     1017          STA   (AUXPTR),Y
8895: F0 05     1018          BEQ   :0
8897: 88        1019          DEY
8898: A5 50     1020          LDA   LINNUM
889A: 91 06     1021          STA   (AUXPTR),Y
889C: 60        1022 :0       RTS
                1023
                1024 NPTRGETX DO    KOPT-K65C02
889D: 64 82     1028          STZ   VARNAM+1
889F: 20 70 7E  1030          JSR   MISLETC
88A2: 85 81     1031          STA   VARNAM
88A4: 20 2A 76  1032          JSR   RST100
88A7: 90 05     1033          BCC   :0
88A9: 20 7D E0  1034          JSR   ISLETC
88AC: 90 16     1035          BCC   :3
88AE: 85 82     1036 :0       STA   VARNAM+1
88B0: 20 2A 76  1037 ]LOOP    JSR   RST100
88B3: 90 FB     1038          BCC   ]LOOP
88B5: 20 7D E0  1039          JSR   ISLETC
88B8: B0 F6     1040          BCS   ]LOOP
88BA: 90 08     1041          BCC   :3
88BC: 20 09 82  1042 :2       JSR   DECTPTR
88BF: A6 81     1043          LDX   VARNAM
88C1: BD 55 9B  1044          LDA   TYPLET-´A´,X
88C4: A2 03     1046 :3       LDX   #3
88C6: 20 00 82  1050          JSR   XFROMMOT+2
88C9: D0 F1     1051          BNE   :2
88CB: 4C E8 81  1052          JMP   ROUT1Y
```

```
                1053
88CE: 2C DC 9C  1054 RNEWISUI BIT   MTACTV
88D1: 10 40     1055          BPL   RESTORD
                1056
                1057          PUT   PEERMTK
                  >1    * Main Active MT entry point
88D3: BA          >2    RMTCTRL  TSX             ;Test for an exhausted thread?
88D4: EC F2 95    >3             CPX   SPROOT
88D7: AE F1 95    >4             LDX   INDX
88DA: 90 07       >5             BCC   :2
88DC: A9 FF       >6             LDA   #$FF       Mark the current thread
88DE: 9D E9 95    >7             STA   TABOFT,X    before switching to another
88E1: B0 15       >8             BCS   KX3        Always branch
88E3: 2C DA 9C    >9    :2       BIT   INHACTV
88E6: 30 2B       >10            BMI   RESTORD
88E8: CE DB 9C    >11            DEC   CTRACTV    Time for a context switch?
88EB: D0 26       >12            BNE   RESTORD    Not yet
88ED: BD E9 95    >13            LDA   TABOFT,X   Get BASIC array where to save
88F0: 20 92 89    >14            JSR   NEXTC2      content
88F3: DA          >16            PHX
88F4: 20 A0 89    >18            JSR   SAVER      Perform the SAVE
88F7: FA          >20            PLX              ;Get back the new context index
                  >21   KX3
88F8: 20 79 89    >25            JSR   NEXTCTX    Search for a new context index
88FB: 90 26       >26            BCC   RESTOR2    Found one
                  >27   * Restore context from calling BASIC line
88FD: 20 70 89    >28   KILLEMAL JSR   SETLTR     Restore context from calling
8900: 20 5C 89    >29            JSR   RESTORC     BASIC line
8903: AE F2 95    >30            LDX   SPROOT
8906: 86 F8       >31            STX   REMSTK
8908: 20 0F 89    >32            JSR   R0
890B: 9A          >33            TXS
890C: 4C D2 D7    >34            JMP   NEWSTT
890F: 4E DC 9C    >35   R0       LSR   MTACTV
8912: 60          >36            RTS
                  >37
8913: 20 A9 8B    >38   RESTORD  JSR   LBS10
8916: 4C 20 D8    >39            JMP   $D820
                  >40   * General purpose restore routine
                  >41   * Input: X register index of context
8919: BD E9 95    >42   RESTOR1  LDA   TABOFT,X
891C: C9 FF       >43            CMP   #$FF       Safe guard: do not restore a
891E: F0 3B       >44            BEQ   RESTORF    terminated thread..
8920: 20 92 89    >45            JSR   NEXTC2
                  >46
                  >47   * Input from caller: X: context index
8923: AD DD 9C    >48   RESTOR2  LDA   ICTRACTV   Reinit counter
8926: 8D DB 9C    >49            STA   CTRACTV     value
                  >50   * Update ITHREAD% variable value
8929: AD F4 95    >51            LDA   ITVADDR+1
892C: F0 0C       >52            BEQ   RESTOR     Skip if no var. defined
892E: 85 07       >53            STA   AUXPTR+1
8930: AD F3 95    >54            LDA   ITVADDR
8933: 85 06       >55            STA   AUXPTR
8935: 8A          >56            TXA
8936: A0 03       >57            LDY   #3
8938: 91 06       >58            STA   (AUXPTR),Y
```

```
893A: 18       >59   RESTOR   CLC
893B: A0 1C    >60            LDY   #28        Trigger the page in routine if
893D: 20 BB 89 >61            JSR   SWPIO       defined
8940: AE F1 95 >63            LDX   INDX
8943: B0 B3    >65            BCS   KX3
               >66   * Do the RESTOR itself
               >67   * Input: LOWTR: Array base address
8945: 20 5C 89 >68            JSR   RESTORC
               >69   * Do the Stack restore
8948: A0 1F    >70            LDY   #31        From offset 31 within context
894A: A6 F8    >71            LDX   REMSTK      array storage
894C: 9A       >72   RESTORX  TXS
894D: EC F2 95 >73   ]LOOP    CPX   SPROOT     Until SPROOT value is reached
8950: B0 C1    >74            BCS   RESTORD
8952: E8       >75            INX
8953: B1 9B    >76            LDA   (LOWTR),Y
8955: 9D 00 01 >77            STA   $0100,X
8958: C8       >78            INY
8959: 90 F2    >79            BCC   ]LOOP      Always
895B: 60       >80   RESTORF  RTS
               >81
895C: 20 CA 89 >83   RESTORC  JSR   LBS06
895F: 90 02    >84            BCC   *+4
8961: 85 D8    >85            STA   ERRFLG
8963: B1 9B    >93   ]LOOP    LDA   (LOWTR),Y
8965: BE ED 95 >94            LDX   P0OFFSET-8,Y
8968: 95 00    >95            STA   0,X
896A: 88       >96            DEY
896B: E0 F8    >97            CPX   #REMSTK
896D: D0 F4    >98            BNE   ]LOOP
896F: 60       >99            RTS
               >100
               >101  * Subroutine to get the context storage index for
               >102  * global (i.e. Perrsoft MT kernel calling line)
8970: A9 C6    >103  SETLTR   LDA   #SVPTR-8
8972: 85 9B    >104           STA   LOWTR
8974: A9 95    >105           LDA   #>SVPTR-8
8976: 85 9C    >106           STA   LOWTR+1
8978: 60       >107           RTS
               >108  * Subroutine to get the next context after the current one
               >109  * (index in X).
8979: A0 00    >110  NEXTCTX  LDY   #0         ctr. to avoid counting too far
897B: E8       >111  ]LOOP    INX              ;Wrap around the context ptr
897C: E0 08    >112           CPX   #TABOFT-TABOFB area..
897E: 90 02    >113           BCC   :0
8980: A2 00    >114           LDX   #0         Perform wrap...
8982: BD E9 95 >115  :0       LDA   TABOFT,X
8985: C9 FF    >116           CMP   #$FF       Got an active one (iif <> $FF)
8987: D0 06    >117           BNE   :1         Yes...
8989: C8       >118           INY              ;Bump counter
898A: C0 08    >119           CPY   #TABOFT-TABOFB till all scanned
898C: 90 ED    >120           BCC   ]LOOP      Not yet: see next context ptr
898E: 60       >121           RTS              ;Exit with carry set..
898F: 8E F1 95 >122  :1       STX   INDX       Memorize the new context index
8992: A8       >123  NEXTC2   TAY              ;From offset to absolute address
8993: BD E1 95 >124           LDA   TABOFB,X   by adding the ARYTAB base address
8996: 65 6B    >125           ADC   ARYTAB     for arrays within Applesoft
```

```
8998: 85 9B    >126           STA   LOWTR
899A: 98       >127           TYA
899B: 65 6C    >128           ADC   ARYTAB+1
899D: 85 9C    >129           STA   LOWTR+1    Result in LOWTR pointer..
899F: 60       >130           RTS              ;Exit with carry clear (always)
               >131
               >132  * Save the context into BASIC array
               >133  * Input: LOWTR: array base address
89A0: 20 D7 89 >134  SAVER    JSR   SAVERC
89A3: A0 1E    >135           LDY   #30        Possible trigger for page out
89A5: 20 BB 89 >136           JSR   SWPIO       event...
               >137  * Now it´s time to save the stack extension
89A8: A0 1F    >138           LDY   #31
               >139  * As a subroutine, do not depend on current stack ptr.
               >140  * But rather on memorized stack ptr. (within exec loop)
89AA: A6 F8    >141           LDX   REMSTK
89AC: EC F2 95 >142  ]LOOP    CPX   SPROOT
89AF: B0 09    >143           BCS   :0
89B1: E8       >144           INX
89B2: BD 00 01 >145           LDA   $0100,X
89B5: 91 9B    >146           STA   (LOWTR),Y
89B7: C8       >147           INY
89B8: 90 F2    >148           BCC   ]LOOP
89BA: 60       >149  :0       RTS
               >150
               >151  * Routine to possibly trigger page in/page out routine
               >152  * for every configured coroutine. Inputs are:
               >153  * LOWTR: context array base address
               >154  * Y either 30 or 28 for page in/out event
89BB: B1 9B    >155  SWPIO    LDA   (LOWTR),Y
89BD: F0 0A    >156           BEQ   :0         No routine defined
89BF: 85 07    >157           STA   AUXPTR+1
89C1: 88       >158           DEY
89C2: B1 9B    >159           LDA   (LOWTR),Y
89C4: 85 06    >160           STA   AUXPTR
               >161  * Called routine must preserve registers
89C6: 6C 06 00 >162           JMP   (AUXPTR)
89C9: 60       >163  :0       RTS
               >164
89CA: A0 1A    >165  LBS06    LDY   #26
89CC: B1 9B    >166  LBS061   LDA   (LOWTR),Y
89CE: D0 04    >167           BNE   :0
89D0: 38       >169           SEC
89D1: A0 0E    >171  :1       LDY   #PIOFFSET-P0OFFSET+8-1
89D3: 60       >172           RTS
89D4: 18       >174  :0       CLC
89D5: 88       >178           DEY              ;Shortcut for
89D6: 60       >179           RTS              ; LDY #PEOFFSET-P0OFFSET+8-1
               >180
89D7: 20 CA 89 >182  SAVERC   JSR   LBS06
89DA: BE ED 95 >187  ]LOOP    LDX   P0OFFSET-8,Y
89DD: B5 00    >188           LDA   0,X        Value to save
89DF: 91 9B    >189           STA   (LOWTR),Y
89E1: 88       >190           DEY
89E2: E0 F8    >191           CPX   #REMSTK
89E4: D0 F4    >192           BNE   ]LOOP
89E6: 60       >193           RTS
```

```
                1058
                1059            PUT   PEERMOUSTIME
                >1    * Base addresses for mouse interface
                >2    BAXLO     EQU   $0478       X low
                >3    BAYLO     EQU   $04F8       Y low
                >4    BAXHI     EQU   $0578       X high
                >5    BAYHI     EQU   $05F8       Y high
                >6    BAMBS     EQU   $0778       Button status
                >7
                >8    TRACE     EQU   $D805
                >9    IRQV      EQU   $03FE       Page 3 Interrupt vector
                >10
                >11   * Reason codes for entering Mouse interface
                >12   RSETM     =     0
                >13   RSRVM     =     1
                >14   RREAD     =     2
                >15   RCLR      =     3
                >16   RPOS      =     4
                >17   RCLM      =     5
                >18   RHOM      =     6
                >19   RINI      =     7
                >20
                >21   CONINT    EQU   $E6FB       FAC to single byte
                >22
                >23   * Interrupt servicing routine
89E7: A2 01     >24   IRQHDLR   LDX   #RSRVM
89E9: 20 85 8C  >25             JSR   TOMOUSE
89EC: B0 39     >26             BCS   :2          ; Not from mouse or spurious
89EE: AE CE 9C  >27             LDX   MOSL
89F1: BD 78 07  >28             LDA   BAMBS,X
89F4: 4A        >29             LSR
                >30   * Movement interrupt bit into b0 and
                >31   * button bit into b1, VBL interrupt bit
                >32   * into b2
89F5: 29 07     >33             AND   #7          mask out other bits
89F7: AA        >34             TAX
89F8: BD C7 99  >35             LDA   MSTATUS,X   Get internal status
89FB: 8D D1 99  >36             STA   WORKPL1
89FE: A2 02     >37             LDX   #RREAD
8A00: 20 85 8C  >38             JSR   TOMOUSE
8A03: 2C D1 99  >39             BIT   WORKPL1
8A06: 10 18     >40             BPL   :1
                >41   * Decrement runtime counter
8A08: AE F7 99  >55             LDX   TIINC
8A0B: D0 03     >56             BNE   :01
8A0D: CE F8 99  >57             DEC   TIINC+1
8A10: CA        >58   :01       DEX
8A11: 8E F7 99  >59             STX   TIINC
8A14: D0 05     >60             BNE   :02
8A16: AD F8 99  >61             LDA   TIINC+1
8A19: F0 1D     >62             BEQ   :00
                >63   :02
8A1B: A9 80     >66             LDA   #$80
8A1D: 1C D1 99  >67             TRB   WORKPL1
8A20: AD D1 99  >73   :1        LDA   WORKPL1
8A23: 0C D2 99  >75             TSB   MIRQST
8A26: 40        >80   ]LOOP     RTI
```

```
                >81
                >82   * No spurious interrupt is fatal to us..
                >83   * I´m afraid of no ghosts.... ;-)
8A27: AD D0 99  >84   :2       LDA   OLDVECT+1
8A2A: C9 FF     >85            CMP   #>$FF65
8A2C: D0 07     >86            BNE   :20
8A2E: AD CF 99  >87            LDA   OLDVECT
8A31: C9 65     >88            CMP   #$FF65
8A33: F0 F1     >89            BEQ   ]LOOP
8A35: 6C CF 99  >90   :20      JMP   (OLDVECT)
                >91
8A38: AD F5 99  >94   :00      LDA   KTINC
8A3B: 8D F7 99  >95            STA   TIINC
8A3E: AD F6 99  >96            LDA   KTINC+1
8A41: 8D F8 99  >97            STA   TIINC+1
8A44: 80 DA     >99            BRA   :1
                >104
                >105  * Install new IRQ handler and save the original handler
                >106  * to build a daisy chain..
                >107  * Nouveau mode dans MOMODE
8A46: AD B8 99  >108  INSIRQV  LDA   MOMODE
8A49: C9 02     >109           CMP   #2
8A4B: 90 20     >110           BCC   :1
8A4D: AD FE 03  >127           LDA   IRQV
8A50: AE FF 03  >128           LDX   IRQV+1
8A53: C9 E7     >129           CMP   #IRQHDLR
8A55: D0 04     >130           BNE   :0
8A57: E0 89     >131           CPX   #>IRQHDLR
8A59: F0 12     >132           BEQ   :1
8A5B: 78        >133  :0       SEI
8A5C: 8D CF 99  >134           STA   OLDVECT
8A5F: 8E D0 99  >135           STX   OLDVECT+1
8A62: A9 E7     >136           LDA   #IRQHDLR
8A64: 8D FE 03  >136           STA   IRQV
8A67: A9 89     >136           LDA   #>IRQHDLR
8A69: 8D FF 03  >136           STA   IRQV+1
8A6C: 58        >138           CLI
8A6D: 60        >139  :1       RTS
                >140
                >141  * Deinstall IRQ handler
8A6E: AD B8 99  >142  DINSIRQV LDA   MOMODE
8A71: C9 02     >143           CMP   #2
8A73: B0 12     >144           BCS   :1
8A75: 78        >145           SEI
8A76: AD D0 99  >159           LDA   OLDVECT+1
8A79: F0 0C     >160           BEQ   :1
8A7B: 8D FF 03  >161           STA   IRQV+1
8A7E: 9C D0 99  >163           STZ   OLDVECT+1
8A81: AD CF 99  >168           LDA   OLDVECT
8A84: 8D FE 03  >169           STA   IRQV
8A87: 60        >171  :1       RTS
                >172
8A88: 48        >173  CMPCLAMP PHA
                >174  * X/Y min% expression
8A89: 20 54 8B  >175           JSR   NEVAL
8A8C: 8D 78 05  >176           STA   $0578
8A8F: 8C 78 04  >177           STY   $0478
```

```
                >178  * X/Y max% expression
8A92: 20 54 8B  >179            JSR   NEVAL
8A95: 8D F8 05  >180            STA   $05F8
8A98: 8C F8 04  >181            STY   $04F8
8A9B: 68        >182            PLA
8A9C: A2 05     >183            LDX   #RCLM
8A9E: 4C 85 8C  >184            JMP   TOMOUSE
                >185
8AA1: C5 A1     >186  IVALARG   CMP   FAC+4
8AA3: 90 01     >187            BCC   *+3
8AA5: 60        >188            RTS
8AA6: 68        >189            PLA
8AA7: 68        >190            PLA
8AA8: 4C 99 E1  >191  ]ERR      JMP   $E199      Illegal quantity error
                >192
8AAB: A9 00     >193  COMCLAMP  LDA   #0
8AAD: 20 88 8A  >194            JSR   CMPCLAMP
8AB0: A9 01     >195            LDA   #1
8AB2: D0 D4     >196            BNE   CMPCLAMP
                >197
8AB4: 20 A6 86  >198  ROUT10    JSR   NCHKCOM
8AB7: 20 D0 86  >199            JSR   NGETBYT    Get reason code in X reg.
8ABA: CA        >200            DEX
8ABB: CA        >201            DEX
8ABC: 30 EA     >202            BMI   ]ERR
8ABE: E0 05     >203            CPX   #5
8AC0: B0 E6     >204            BCS   ]ERR
8AC2: 20 0D 8E  >205            JSR   ISMOUSH
8AC5: AD B8 99  >206            LDA   MOMODE
8AC8: 29 0F     >207            AND   #$F
8ACA: D0 05     >208            BNE   :1
8ACC: A2 25     >209            LDX   #37
8ACE: 4C 1C 8E  >210            JMP   NERRH
                >211  * Only READ (2), CLEAR (3), POS(4), CLAMP (5) and HOME (6)
                >212  * reason codes are valid.
8AD1: 8A        >213  :1        TXA
8AD2: F0 11     >214            BEQ   COMREAD
8AD4: CA        >215            DEX
8AD5: F0 09     >216            BEQ   COMCLEAR
8AD7: CA        >217            DEX
8AD8: F0 39     >218            BEQ   COMPOS
8ADA: CA        >219            DEX
8ADB: F0 CE     >220            BEQ   COMCLAMP
8ADD: A2 06     >221            LDX   #RHOM
8ADF: 2C        >222            HEX   2C         Skip next two bytes
8AE0: A2 6A     >223  COMCLEAR  LDX   #RCLEAR
8AE2: 4C 85 8C  >224  FINMOUSE  JMP   TOMOUSE
                >225
8AE5: AE D4 99  >226  COMREAD   LDX   MODERUN
8AE8: D0 05     >227            BNE   :1
8AEA: A2 02     >228            LDX   #RREAD
8AEC: 20 85 8C  >229            JSR   TOMOUSE
                >230  * Handles X% host variable
8AEF: AE CE 9C  >231  :1        LDX   MOSL
8AF2: BD 78 05  >232            LDA   BAXHI,X
8AF5: 20 2F 8B  >233            JSR   NPTRG
8AF8: BD 78 04  >234            LDA   BAXLO,X
```

```
8AFB: 91 83    >235           STA   (VARPNT),Y
               >236  * Handle Y% host variable
8AFD: BD F8 05 >237           LDA   BAYHI,X
8B00: 20 2F 8B >238           JSR   NPTRG
8B03: BD F8 04 >239           LDA   BAYLO,X
8B06: 91 83    >240           STA   (VARPNT),Y
               >241  * Handle S% for button status variable
8B08: A9 00    >242           LDA   #0
8B0A: 20 2F 8B >243           JSR   NPTRG
8B0D: BD 78 07 >244           LDA   BAMBS,X
8B10: 91 83    >245           STA   (VARPNT),Y
8B12: 60       >246           RTS
               >247
               >248  COMPOS
               >249  * X% expression
8B13: 20 54 8B >250           JSR   NEVAL
8B16: 9D 78 05 >251           STA   BAXHI,X
8B19: 98       >252           TYA
8B1A: 9D 78 04 >253           STA   BAXLO,X
               >254  * Y% expression
8B1D: 20 54 8B >255           JSR   NEVAL
8B20: 9D F8 05 >256           STA   BAYHI,X
8B23: 98       >257           TYA
8B24: 9D F8 04 >258           STA   BAYLO,X
8B27: A2 04    >259           LDX   #RPOS
8B29: 4C E2 8A >260           JMP   FINMOUSE
               >261
8B2C: 4C 76 DD >262  ]ERR     JMP   GOTMIERR   TYPE MISMATCH ERROR
8B2F: 48       >263  NPTRG    PHA
8B30: 20 A6 86 >264           JSR   NCHKCOM
8B33: 20 06 7A >265           JSR   NPTRGTX
8B36: A5 12    >266           LDA   INTTYP
8B38: 10 F2    >267           BPL   ]ERR
8B3A: 29 0F    >268           AND   #15        cater for integer subtypes
8B3C: F0 04    >269           BEQ   :1         only $80 and $82 are valid
8B3E: C9 02    >270           CMP   #2
8B40: D0 EA    >271           BNE   ]ERR
8B42: AE CE 9C >272  :1       LDX   MOSL
8B45: 68       >273           PLA
8B46: 92 83    >275           STA   (VARPNT)
8B48: A0 01    >276           LDY   #1
8B4A: 60       >282           RTS
               >283
               >284  * Result in FAC+3, FAC+4
8B4B: 20 A6 86 >285  NEVALC   JSR   NCHKCOM
8B4E: 20 3B 85 >286           JSR   NFRMNUM
8B51: 4C 4E 78 >287           JMP   NROUT      Replac. for ROUND.FAC/AYINT
               >288
8B54: 20 4B 8B >289  NEVAL    JSR   NEVALC
8B57: A5 A0    >290           LDA   FAC+3
8B59: A4 A1    >291           LDY   FAC+4
8B5B: AE CE 9C >292           LDX   MOSL
8B5E: 60       >293  ]RET     RTS
               >294
               >295  * Common subroutine for parsing new tokens
               >296  * X upon entry: 0: updates TXTPTR if token found
               >297  * 1: skip updating TXTPTR even when token found
```

```
8B5F: 86 C0    >298  COMLBS   STX   GFLAG
8B61: B2 B8    >300          LDA   (TXTPTR)
8B63: 30 19    >305          BMI   :2
8B65: C9 4D    >306          CMP   #´M´
8B67: F0 04    >307          BEQ   :1
8B69: C9 54    >308          CMP   #´T´
8B6B: D0 11    >309          BNE   :2
8B6D: A2 03    >310  :1       LDX   #3
8B6F: 20 C0 82 >311          JSR   RECON1
8B72: F0 EA    >312          BEQ   ]RET
8B74: 20 6B 8C >313          JSR   COMINT4    Check mouse hardware/reinit
8B77: A6 C0    >314          LDX   GFLAG
8B79: D0 E3    >315          BNE   ]RET
8B7B: 4C 98 D9 >316          JMP   ADDON      will exit with Z flag clear
               >317  :2
8B7E: A2 00    >319          LDX   #0
8B80: 60       >323  ]RET     RTS
               >324
               >325  * New instructions handling
               >326  * for MOUSE and TIMER instructions
8B81: 4C 32 76 >327  ]LOOP    JMP   RST102
8B84: 68       >328  ]ERR1    PLA             ;Pull IDMOCL from stack
8B85: 68       >329          PLA             ;Pull return address
8B86: 68       >330          PLA
8B87: 4C C9 DE >331  ]ERR     JMP   SYNERR
               >332  * MOUSE/TIMER STOP handler
8B8A: C0 09    >333  ]JLOOP   CPY   #OFFTIM-TOFFST
8B8C: A2 00    >334          LDX   #0
8B8E: 90 01    >335          BCC   *+3        Branch iif MOUSE
8B90: E8       >336          INX
8B91: AD B8 99 >337          LDA   MOMODE
8B94: 3D C3 99 >338          AND   MOETMSK,X
               >339  * Compare to minimum allowable value
8B97: DD C5 99 >340          CMP   MOCMPVAL,X
8B9A: B0 05    >341          BCS   :0         OK iif greater or equal
8B9C: A2 25    >342          LDX   #37
8B9E: 4C 1C 8E >343          JMP   NERRH
8BA1: A9 01    >344  :0       LDA   #1         Update MODEPEC configuration
8BA3: 9D D6 99 >345          STA   MODEPEC,X
8BA6: 4C D2 D7 >346          JMP   NEWSTT
8BA9: A2 00    >347  LBS10    LDX   #0
8BAB: 20 5F 8B >348          JSR   COMLBS
8BAE: F0 D1    >349          BEQ   ]LOOP
8BB0: A5 BD    >350          LDA   IDMOCL
8BB2: 48       >351          PHA
8BB3: B2 B8    >353          LDA   (TXTPTR)
8BB5: A0 01    >354          LDY   #1
8BB7: C9 B3    >360          CMP   #$B3       STOP token?
8BB9: F0 0F    >361          BEQ   :3
8BBB: C9 B4    >362          CMP   #$B4
8BBD: F0 0B    >363          BEQ   :3         ON token?
8BBF: C9 4F    >364          CMP   #´O´
8BC1: D0 C1    >365          BNE   ]ERR1
8BC3: A2 05    >366          LDX   #5         Look up possible OFF pattern
8BC5: 20 C0 82 >367          JSR   RECON1
8BC8: F0 BA    >368          BEQ   ]ERR1
8BCA: AA       >369  :3       TAX             ;X STOP/ON token or 0 (OFF)
```

```
8BCB: 86 B4    >370           STX   XSAV
8BCD: 20 98 D9 >371           JSR   ADDON
8BD0: 7A       >372           PLY
8BD1: 68       >373           PLA
8BD2: 68       >374           PLA
8BD3: 20 32 76 >375           JSR   RST102
8BD6: F0 15    >376           BEQ   :23        If EOI found
8BD8: E0 B4    >377           CPX   #$B4
8BDA: D0 AB    >378           BNE   ]ERR       SYNTAX ERR if not ON nor EOI
8BDC: DA       >379           PHX
8BDD: 5A       >380           PHY
8BDE: 20 4B 8B >381           JSR   NEVALC     Get factor/mode value after comma
8BE1: 7A       >382           PLY
8BE2: FA       >383           PLX
8BE3: 86 B4    >384           STX   XSAV
8BE5: C0 08    >385           CPY   #OFFMOU-TOFFST
8BE7: D0 06    >386           BNE   :20
8BE9: 20 FE E6 >387           JSR   $E6FE      FAC integer -> single byte
8BEC: 2C       >388           HEX   2C
8BED: A2 01    >389  :23      LDX   #1
8BEF: 86 C0    >390  :20      STX   GFLAG
8BF1: 84 BD    >391           STY   IDMOCL
8BF3: A5 B4    >392           LDA   XSAV        A: ON/OFF/STOP index
8BF5: C9 B3    >393           CMP   #$B3       STOP token?
8BF7: F0 91    >394           BEQ   ]JLOOP
               >395  * IDMOCL in page zero, STOP/ON/OFF indic. in A reg.
8BF9: A6 BD    >396           LDX   IDMOCL
8BFB: E0 08    >397           CPX   #OFFMOU-TOFFST
8BFD: D0 3F    >398           BNE   TIMEINST
               >399
               >400  * Mouse event handler
8BFF: C9 B4    >401           CMP   #$B4       MOUSE ON?
8C01: D0 04    >402           BNE   *+6        No
8C03: A2 00    >403           LDX   #0
8C05: F0 0D    >404           BEQ   :8
8C07: A2 07    >405           LDX   #7
8C09: E4 C0    >406  ]LOOP    CPX   GFLAG
8C0B: F0 07    >407           BEQ   :8
8C0D: CA       >408           DEX
8C0E: CA       >409           DEX
8C0F: 10 F8    >410           BPL   ]LOOP
8C11: 4C 1A 8E >411  ]LOOP    JMP   NILLM
8C14: A9 07    >413  :8       LDA   #7
8C16: 1C B8 99 >414           TRB   MOMODE
8C19: 8A       >415           TXA
8C1A: 0C B8 99 >416           TSB   MOMODE
8C1D: C9 02    >417           CMP   #2
8C1F: A9 00    >426           LDA   #0
8C21: A8       >427           TAY
8C22: 90 02    >428           BCC   *+4
8C24: A9 02    >429  COMMON9  LDA   #2
8C26: 99 D6 99 >430           STA   MODEPEC,Y
8C29: AD B8 99 >431  COMMON   LDA   MOMODE
8C2C: 48       >432           PHA
8C2D: 20 46 8A >433           JSR   INSIRQV
8C30: 68       >434           PLA
8C31: A2 00    >435           LDX   #RSETM
```

```
8C33: 20 85 8C >436           JSR   TOMOUSE
8C36: B0 D9    >437           BCS   ]LOOP
8C38: 20 6E 8A >438           JSR   DINSIRQV
8C3B: 4C D2 D7 >439           JMP   NEWSTT
               >440
8C3E: C9 B4    >441  TIMEINST CMP   #$B4       TIMER ON
8C40: A9 08    >443           LDA   #8
8C42: 1C B8 99 >444           TRB   MOMODE
8C45: 90 E2    >445           BCC   COMMON
8C47: 0C B8 99 >446           TSB   MOMODE
8C4A: 24 C0    >456           BIT   GFLAG
8C4C: 30 06    >457           BMI   *+8
8C4E: A2 01    >458           LDX   #1
8C50: A0 00    >459           LDY   #0
8C52: 10 04    >460           BPL   *+6        Always
8C54: A6 A1    >461           LDX   FAC+4
8C56: A4 A0    >462           LDY   FAC+3
8C58: 08       >463           PHP
8C59: 78       >464           SEI
8C5A: 8C F6 99 >465           STY   KTINC+1
8C5D: 8E F5 99 >466           STX   KTINC
8C60: 8C F8 99 >467           STY   TIINC+1
8C63: 8E F7 99 >468           STX   TIINC
8C66: 28       >469           PLP
8C67: A0 01    >470           LDY   #1
8C69: B0 B9    >471           BCS   COMMON9    Always
               >472
               >473  * Do we have suitable mouse hardware?
8C6B: 20 0D 8E >474  COMINT4  JSR   ISMOUSH    Fall into SWREINIT if yes
               >475  * Routine below to check whether we should init the
               >476  * MOUSE system?
               >477  SWREINIT
8C6E: A9 80    >479           LDA   #$80
8C70: 0C C1 99 >480           TSB   MONU
8C73: D0 0C    >481           BNE   :0
               >488  * INITMOUSE was performed on Peersoft boot when in an
               >489  * Apple 2,2+ host.
8C75: AD ED 9C >490           LDA   MACHINE
8C78: F0 07    >491           BEQ   :0
8C7A: 5A       >492           PHY
8C7B: A2 07    >493           LDX   #RINI
8C7D: 20 85 8C >494           JSR   TOMOUSE
8C80: 7A       >495           PLY
8C81: 60       >496  :0       RTS
               >497
8C82: 6C B6 99 >498  ]LOOP    JMP   (MVECTOR)
               >499
8C85: BC AD 99 >500  TOMOUSE  LDY   OM_DEB,X
8C88: AE B7 99 >501           LDX   MOCN
8C8B: 08       >502           PHP
8C8C: 78       >503           SEI
8C8D: 8C B6 99 >504           STY   MVECTOR
8C90: AC B5 99 >505           LDY   MON0
8C93: 20 82 8C >506           JSR   ]LOOP
8C96: B0 03    >507           BCS   *+5
8C98: 28       >508           PLP
8C99: 18       >509           CLC
```

```
8C9A: 60       >510           RTS
8C9B: 28       >511           PLP
8C9C: 38       >512           SEC
8C9D: 60       >513           RTS
               >514
               >515  * Entry routine for MOUSE functions (either MOUSE or
               >516  * TIMER).
8C9E: 48       >517  MTFUNC   PHA
8C9F: 20 FB E6 >518           JSR   CONINT
8CA2: 20 A3 86 >519           JSR   NCHKCLS
8CA5: 20 6B 8C >520           JSR   COMINT4
8CA8: 68       >521           PLA
8CA9: D0 31    >522           BNE   TFUNC
8CAB: A9 02    >523           LDA   #2
8CAD: 20 A1 8A >524           JSR   IVALARG
8CB0: AE D4 99 >525           LDX   MODERUN
8CB3: D0 05    >526           BNE   *+7        Branch if within interrupt
8CB5: A2 02    >527           LDX   #RREAD
8CB7: 20 85 8C >528           JSR   TOMOUSE
8CBA: AE CE 9C >529           LDX   MOSL
8CBD: A5 A1    >531           LDA   FAC+4
8CBF: 3A       >532           DEC
8CC0: 10 09    >537           BPL   :1
8CC2: BD 78 05 >538           LDA   BAXHI,X    MOUSE(0) means read X
8CC5: BC 78 04 >539           LDY   BAXLO,X
8CC8: 4C F2 E2 >540  ]LOOP    JMP   GIVAYF
               >541  :1       DO    KOPT-K6502
8CCB: 3A       >542           DEC
8CCC: 10 08    >546           BPL   :2
8CCE: BD F8 05 >547           LDA   BAYHI,X    MOUSE(1) means read Y
8CD1: BC F8 04 >548           LDY   BAYLO,X
8CD4: 80 F2    >550           BRA   ]LOOP
8CD6: BC 78 07 >554  :2       LDY   BAMBS,X    MOUSE(2) means read buttons
8CD9: 4C 01 E3 >555           JMP   SNGFLT
8CDC: A9 01    >556  TFUNC    LDA   #1
8CDE: 20 A1 8A >557           JSR   IVALARG
8CE1: 20 05 8E >558           JSR   ISHOSTOK
8CE4: A2 00    >559           LDX   #0
8CE6: A5 A1    >560           LDA   FAC+4
8CE8: F0 02    >561           BEQ   *+4
8CEA: A2 02    >562           LDX   #2
8CEC: BD F6 99 >563           LDA   KTINC+1,X
8CEF: BC F5 99 >564           LDY   KTINC,X
8CF2: 80 D4    >566           BRA   ]LOOP
               >570
               >571  * Desactive le traitement d´une interruption (sur RETURN)
               >572  * Y en entree: indice de l´interruption
8CF4: A9 00    >573  COMINT1  LDA   #0
8CF6: 99 D4 99 >574           STA   MODERUN,Y
8CF9: 3A       >576           DEC
8CFA: 8D D3 99 >580           STA   YICUR
               >581  * MODEPEC passe de STOP a ON
8CFD: B9 D6 99 >583           LDA   MODEPEC,Y
8D00: C9 01    >584           CMP   #1
8D02: D0 04    >585           BNE   :0
8D04: 1A       >586           INC
8D05: 99 D6 99 >594           STA   MODEPEC,Y
```

```
8D08: B9 E0 99 >595  :0       LDA   TPT_B,Y
8D0B: 85 B8    >596           STA   TXTPTR
8D0D: B9 E2 99 >597           LDA   TPT_T,Y
8D10: 85 B9    >598           STA   TXTPTR+1
8D12: B9 DC 99 >599           LDA   CLN_B,Y
8D15: 85 75    >600           STA   CURLIN
8D17: B9 DE 99 >601           LDA   CLN_T,Y
8D1A: 85 76    >602           STA   CURLIN+1
8D1C: B9 E4 99 >603           LDA   OTPT_B,Y
8D1F: 85 79    >604           STA   OLDTEXT
8D21: B9 E6 99 >605           LDA   OTPT_T,Y
8D24: 85 7A    >606           STA   OLDTEXT+1
8D26: AE C2 99 >607           LDX   SVMTACTV
8D29: AD D4 99 >608           LDA   MODERUN
8D2C: 0D D5 99 >609           ORA   MODERUN+1
8D2F: D0 06    >610           BNE   *+8
8D31: 8D C2 99 >611           STA   SVMTACTV
8D34: 8E DC 9C >612           STX   MTACTV
8D37: A0 05    >613           LDY   #5
8D39: CC F4 99 >614           CPY   FRGNDCTX
8D3C: D0 05    >615           BNE   :1
8D3E: 68       >616           PLA
8D3F: 68       >617           PLA
8D40: 4C 74 8E >618           JMP   RW2
8D43: 60       >619  :1       RTS
               >620
               >621  * Routine en charge de determiner si l´interruption peut
               >622  * ou non etre cascadee.
               >623  * Sortie: bitN a 0 ssi possibilite de cascade (indice
               >624  * dans Y)
8D44: A0 01    >625  COMINT2   LDY   #1          On commence par la TIMER
8D46: B9 D8 99 >626  ]LOOP     LDA   MSKINT,Y
8D49: 08       >627           PHP                ;Sauve le interrupt enable
8D4A: 78       >628           SEI                ;courant
8D4B: 2D D2 99 >629           AND   MIRQST
8D4E: F0 27    >630           BEQ   :3
               >631  * Uniquement si prise en compte immediate..
8D50: BE D6 99 >632           LDX   MODEPEC,Y
8D53: E0 02    >633           CPX   #2
8D55: D0 20    >634           BNE   :3
               >635  * Uniquement si routine non deja active
8D57: BE D4 99 >636           LDX   MODERUN,Y
8D5A: D0 1B    >637           BNE   :3
8D5C: 1C D2 99 >639           TRB   MIRQST
8D5F: 28       >646           PLP
8D60: A9 02    >647           LDA   #3-1         because from within a called subr
.
8D62: 20 D6 D3 >648           JSR   CHKMEM
8D65: 8C D3 99 >649           STY   YICUR
8D68: AD DC 9C >650           LDA   MTACTV
8D6B: 8D C2 99 >651           STA   SVMTACTV
8D6E: A9 01    >652           LDA   #1
8D70: 99 D6 99 >653           STA   MODEPEC,Y
8D73: 99 D4 99 >654           STA   MODERUN,Y
8D76: 60       >655           RTS
8D77: 28       >656  :3       PLP
8D78: 88       >657           DEY
```

```
8D79: 10 CB    >658           BPL   ]LOOP
8D7B: 60       >659           RTS
               >660
               >661  * Retour d´une interruption souris
8D7C: A0 00    >662  RETOURM  LDY   #0
8D7E: 2C       >663           HEX   2C         Skip next two bytes
8D7F: A0 01    >664  RETOURT  LDY   #1
8D81: BA       >665           TSX
8D82: 86 F8    >666           STX   REMSTK
8D84: 20 F4 8C >667           JSR   COMINT1
8D87: 20 09 82 >668           JSR   DECTPTR
8D8A: 20 58 D8 >669           JSR   ISCNTC
8D8D: 4C 05 D8 >670           JMP   TRACE
               >671
8D90: AD D4 99 >672  RNEWINST LDA   MODERUN
8D93: 0D D5 99 >673           ORA   MODERUN+1
8D96: F0 19    >674           BEQ   RNI2
               >675  * Y a la bonne valeur selon MOUSE ou TIMER actifs
8D98: AC D3 99 >676           LDY   YICUR
8D9B: 10 0A    >677           BPL   :1
8D9D: C8       >678           INY              ;Y passe de FF a 0
8D9E: AD D5 99 >679           LDA   MODERUN+1
8DA1: F0 01    >680           BEQ   *+3
8DA3: C8       >681           INY              ;Y passe a 1
8DA4: 8C D3 99 >682           STY   YICUR
8DA7: BA       >683  :1       TSX
8DA8: 8A       >684           TXA
               >685  * Routine terminee par RETURN/POP ayant ramene le SP
8DA9: D9 DA 99 >686           CMP   INTSPTR,Y
8DAC: 90 03    >687           BCC   RNI2
8DAE: 20 F4 8C >688           JSR   COMINT1
               >689  * ...
8DB1: AD D2 99 >690  RNI2     LDA   MIRQST
8DB4: F0 4C    >691           BEQ   :4
8DB6: 20 44 8D >692           JSR   COMINT2
8DB9: 30 47    >693           BMI   :4         ;
               >694  * Reminder of current stack pointer
8DBB: BA       >695           TSX
8DBC: 8A       >696           TXA
8DBD: 99 DA 99 >697           STA   INTSPTR,Y
               >698  * Builds the GOSUB stack frame
8DC0: C0 01    >699           CPY   #1         carry set iif TIMER int.
8DC2: B0 06    >706           BCS   *+8
8DC4: A2 7B    >707           LDX   #RETOURM-1
8DC6: A9 8D    >708           LDA   #>RETOURM-1
8DC8: D0 04    >709           BNE   *+6
8DCA: A2 7E    >710           LDX   #RETOURT-1
8DCC: A9 8D    >711           LDA   #>RETOURT-1
8DCE: 48       >712           PHA
8DCF: DA       >713           PHX
8DD0: A5 B9    >715           LDA   TXTPTR+1
8DD2: 99 E2 99 >716           STA   TPT_T,Y
8DD5: 48       >717           PHA
8DD6: A5 B8    >718           LDA   TXTPTR
8DD8: 99 E0 99 >719           STA   TPT_B,Y
8DDB: 48       >720           PHA
8DDC: A5 76    >721           LDA   CURLIN+1
```

```
8DDE: 99 DE 99 >722           STA   CLN_T,Y
8DE1: 48       >723           PHA
8DE2: A5 75    >724           LDA   CURLIN
8DE4: 99 DC 99 >725           STA   CLN_B,Y
8DE7: 48       >726           PHA
8DE8: A5 79    >727           LDA   OLDTEXT
8DEA: 99 E4 99 >728           STA   OTPT_B,Y
8DED: A5 7A    >729           LDA   OLDTEXT+1
8DEF: 99 E6 99 >730           STA   OTPT_T,Y
8DF2: A9 B0    >731           LDA   #TOKGOSUB
8DF4: 48       >732           PHA
               >733  * and initialize the context for irq handler
               >734  * (before falling into NEWSTT)
8DF5: BE BF 99 >735           LDX   AHNDHI,Y
8DF8: B9 BD 99 >736           LDA   AHNDLO,Y
8DFB: 85 B8    >737           STA   TXTPTR
8DFD: 86 B9    >738           STX   TXTPTR+1
8DFF: 4C D2 D7 >739           JMP   NEWSTT
               >740
8E02: 4C CE 88 >741  :4       JMP   RNEWISUI
               >742
8E05: AD ED 9C >743  ISHOSTOK LDA   MACHINE
8E08: C9 41    >744           CMP   #$41       Enhanced 2e ROM pattern
8E0A: 90 09    >745           BCC   HNOK
8E0C: 60       >746  ]RET     RTS
8E0D: AD B7 99 >747  ISMOUSH  LDA   MOCN
8E10: D0 FA    >748           BNE   ]RET
8E12: A2 20    >749           LDX   #32
8E14: 2C       >750           HEX   2C         Skip next two byte
8E15: A2 21    >751  HNOK     LDX   #33
8E17: 68       >752  NERRHP   PLA              ;Pull return address
8E18: 68       >753           PLA
8E19: 2C       >754           HEX   2C
8E1A: A2 24    >755  NILLM    LDX   #36
               >756  * Error handler for new reason codes
               >757  * Upon entry, possible values of X
               >758  * 32: MOUSE NOT DETECTED
               >759  * UNSUPPORTED HARDWARE CONFIG.
               >760  * UNKNOWN APPLESOFT MOUSE EVENT HANDLER
               >761  * Same for TIMER
               >762  * ILLEGAL MOUSE MODE
               >763  * ILLEGAL MOUSE OP.
8E1C: 24 D8    >764  NERRH    BIT   ERRFLG
8E1E: 10 03    >765           BPL   *+5
8E20: 4C F9 E2 >766           JMP   $E2F9      to ROM Error handler code
8E23: 20 FB DA >767           JSR   CRDO
8E26: 20 5A DB >768           JSR   $DB5A      Output question mark
8E29: BD E2 9A >769           LDA   CODR-32,X
8E2C: AA       >770           TAX
8E2D: BD F9 99 >771  ]LOOP    LDA   MESSERR,X
8E30: 48       >772           PHA
8E31: 20 5C DB >773           JSR   OUTDO
8E34: E8       >774           INX
8E35: 68       >775           PLA
8E36: 10 F5    >776           BPL   ]LOOP
8E38: 4C 2A D4 >777           JMP   $D42A      Fall into ROM code tail
               >778
```

```
8E3B: 20 46 E7 >779  RWAIT    JSR   $E746      Get address in LINNUM,
8E3E: 86 85    >780           STX   FORPNT      mask in X (saved)
8E40: A2 00    >781           LDX   #0
8E42: 20 B7 00 >782           JSR   $00B7
8E45: F0 03    >783           BEQ   *+5
8E47: 20 4C E7 >784           JSR   COMBYTE
8E4A: 86 86    >785           STX   FORPNT+1
               >789  COMWAIT
8E4C: AD D2 99 >790  ]LOOP    LDA   MIRQST
8E4F: D0 09    >791           BNE   :2
8E51: B2 50    >793           LDA   (LINNUM)
8E53: 45 86    >797           EOR   FORPNT+1
8E55: 25 85    >798           AND   FORPNT
8E57: F0 F3    >799           BEQ   ]LOOP
8E59: 60       >800           RTS
8E5A: 20 44 8D >801  :2       JSR   COMINT2
8E5D: 30 ED    >803           BMI   ]LOOP
8E5F: 5A       >809           PHY
8E60: A0 05    >810           LDY   #5
8E62: 8C F4 99 >811           STY   FRGNDCTX
8E65: BE E8 99 >812  ]LOOP    LDX   SVWOF,Y
8E68: B5 00    >813           LDA   0,X
8E6A: 99 EE 99 >814           STA   SVA,Y
8E6D: 88       >815           DEY
8E6E: 10 F5    >816           BPL   ]LOOP
8E70: 7A       >817           PLY
8E71: 4C BB 8D >818           JMP   RNI2+10
               >819
8E74: A0 06    >820  RW2      LDY   #6
8E76: BE E7 99 >821  ]LOOP    LDX   SVWOF-1,Y
8E79: B9 ED 99 >822           LDA   SVA-1,Y
8E7C: 95 00    >823           STA   0,X
8E7E: 88       >824           DEY
8E7F: D0 F5    >825           BNE   ]LOOP
8E81: 8C F4 99 >826           STY   FRGNDCTX
8E84: F0 C6    >827           BEQ   COMWAIT    Always
                1060
8E86: A9 10     1061 GN32768  LDA   #NEG32768
8E88: A0 9B     1062          LDY   #>NEG32768
8E8A: 60        1063          RTS
8E8B: A9 15     1064 GP32768  LDA   #POS32768
8E8D: A0 9B     1065          LDY   #>POS32768
8E8F: 60        1066          RTS
                1067
8E90: A9 0B     1068 GN65536  LDA   #NEG65536
8E92: A0 9B     1069          LDY   #>NEG65536
8E94: 60        1070          RTS
8E95: A9 1A     1071 GP65536  LDA   #POS65536
8E97: A0 9B     1072          LDY   #>POS65536
8E99: 60        1073          RTS
                1074
                1075 * Get address of array which name is pointed to by
                1076 * TXTPTR. If no array is found, then the called
                1077 * ROM routine would have created one so we´ll have
                1078 * to rollback such creation and exit.
                1079 NGTA2    DO    KOPT16
8E9A: A5 6E     1082          LDA   STREND+1
```

```
8E9C: 48        1083          PHA
8E9D: A5 6D     1084          LDA   STREND
8E9F: 48        1085          PHA
8EA0: 20 02 7A  1087          JSR   NGETARPT
8EA3: FA        1088          PLX
8EA4: 68        1089          PLA
8EA5: B0 04     1090          BCS   :1          found existing array
8EA7: 85 6E     1091          STA   STREND+1    Do the rollback
8EA9: 86 6D     1092          STX   STREND
                1093 :1       DO    KOPT-K65C02
8EAB: 64 14     1097          STZ   SUBFLG
8EAD: 60        1099          RTS
                1100
                1101          PUT   PEERFORNEXT
                 >1     * Module en charge du traitement de boucles FOR/NEXT
                 >2     * en variante classique comme en variante FOREACH
                 >3     GTFORPNT EQU   $D365
                 >4     GETSPA   EQU   $E452      Get mem. space for new string
                 >5
8EAE: 4C 76 DD   >6     ]ERR     JMP   GOTMIERR
8EB1: 20 A6 86   >7     FEFOR    JSR   NCHKCOM    Ensure trailing comma
8EB4: A5 86      >12             LDA   FORPNT+1
8EB6: 48         >13             PHA
8EB7: A5 85      >14             LDA   FORPNT
8EB9: 48         >15             PHA
8EBA: A5 12      >16             LDA   INTTYP
8EBC: 48         >17             PHA
8EBD: A5 11      >18             LDA   VALTYP
8EBF: 48         >19             PHA
8EC0: 20 9A 8E   >20             JSR   NGTA2
8EC3: 90 E9      >21             BCC   ]ERR       En attendant mieux..
                 >22    * Same element type for array and loop variable?
8EC5: 68         >23             PLA
8EC6: 45 11      >24             EOR   VALTYP
8EC8: D0 E4      >25             BNE   ]ERR
8ECA: 68         >26             PLA
8ECB: 45 12      >27             EOR   INTTYP
8ECD: D0 DF      >28             BNE   ]ERR
8ECF: 68         >29             PLA
8ED0: 85 85      >30             STA   FORPNT
8ED2: 68         >31             PLA
8ED3: 85 86      >32             STA   FORPNT+1
                 >33    * LOWTR address: array base address
                 >34    * FORPNT address: simple variable value address
8ED5: 20 65 D3   >35             JSR   GTFORPNT
8ED8: D0 05      >36             BNE   :1          Si pas trouvee
                 >37    * Si oui, on revient au debut de la struct. dans la pile
8EDA: 8A         >38             TXA
8EDB: 69 0F      >39             ADC   #15
8EDD: AA         >40             TAX
8EDE: 9A         >41             TXS
8EDF: 68         >42    :1       PLA               ;Pop return address
8EE0: 68         >43             PLA
                 >44    * Check enough space on stack and start computing
                 >45    * TXTPTR for body loop.
8EE1: 20 DE 91   >46             JSR   LBS60
8EE4: 48         >47             PHA
```

```
8EE5: A5 B9    >48            LDA   TXTPTR+1
8EE7: 69 00    >49            ADC   #0
8EE9: 48       >50            PHA
8EEA: A5 76    >54            LDA   CURLIN+1
8EEC: 48       >55            PHA
8EED: A5 75    >56            LDA   CURLIN
8EEF: 48       >57            PHA
               >59   * Analyse array: result $9D and $A0 in abs. form
8EF0: 20 C9 90 >60            JSR   LBS61
8EF3: 20 F8 90 >61            JSR   LBS63      Copy 1st elm into loop var.
8EF6: 20 87 91 >62            JSR   LBS68      From abs to offset(ARYTAB)
8EF9: A9 00    >63   SFE1     LDA   #FESTEP
8EFB: A0 8F    >64            LDY   #>FESTEP
8EFD: 4C F1 90 >65            JMP   LBS62
               >66
8F00: A9 A5    >67   FESTEP   LDA   #%10100101
8F02: 4C 98 8F >68            JMP   COMFOR
               >69
8F05: 4C C9 DE >70   ]ERR     JMP   SYNERR
8F08: 20 B5 91 >71   RFOR     JSR   ITEACH     ;FOREACH variant?
8F0B: 08       >72            PHP              ;Z bit on stack
8F0C: 64 14    >74            STZ   SUBFLG
8F0E: 20 08 7A >79            JSR   NPTRGET
8F11: 85 85    >80            STA   FORPNT
8F13: 84 86    >81            STY   FORPNT+1
8F15: C5 6B    >82            CMP   ARYTAB
8F17: 98       >83            TYA
8F18: E5 6C    >84            SBC   ARYTAB+1
8F1A: B0 E9    >85            BCS   ]ERR
8F1C: 28       >86            PLP
8F1D: F0 92    >87            BEQ   FEFOR
8F1F: A0 01    >88            LDY   #1
8F21: B1 9B    >89            LDA   (LOWTR),Y
8F23: 52 9B    >91            EOR   (LOWTR)
8F25: 30 DE    >96            BMI   ]ERR
8F27: A5 12    >97            LDA   INTTYP
8F29: 48       >98            PHA
8F2A: 20 AA 76 >99            JSR   RLET1
8F2D: 68       >100           PLA
8F2E: 85 C0    >101           STA   GFLAG
8F30: 20 65 D3 >102           JSR   GTFORPNT
8F33: D0 05    >103           BNE   :0
8F35: 8A       >104           TXA              ;Stackframe pointer in X
8F36: 69 0F    >105           ADC   #$0F       Carry already set, add 16
8F38: AA       >106           TAX              ;+2 bytes (lines below)
8F39: 9A       >107           TXS              ;= 18 bytes
8F3A: 68       >108  :0       PLA
8F3B: 68       >109           PLA
8F3C: 24 C0    >110           BIT   GFLAG
8F3E: 30 03    >111           BMI   :1
8F40: 4C 79 D7 >112           JMP   $D779
               >113  * Check that enough stack available and
               >114  * compute Y as offset to next separator
8F43: 20 DE 91 >115  :1       JSR   LBS60
8F46: 48       >116           PHA
8F47: A5 B9    >117           LDA   TXTPTR+1
8F49: 69 00    >118           ADC   #0
```

```
8F4B: 48       >119           PHA
8F4C: A9 C1    >120           LDA   #TOKTO
8F4E: 20 76 7E >121           JSR   NSYNCHR
8F51: A5 76    >125           LDA   CURLIN+1
8F53: 48       >126           PHA
8F54: A5 75    >127           LDA   CURLIN
8F56: 48       >128           PHA
8F57: 18       >130           CLC
8F58: 20 CF 91 >131           JSR   LBS033
8F5B: A9 62    >132  STP1     LDA   #STEP
8F5D: A0 8F    >133           LDY   #>STEP
8F5F: 4C F1 90 >134           JMP   LBS62
               >135
8F62: 20 32 76 >136  STEP     JSR   RST102
8F65: A0 01    >137           LDY   #1
8F67: 84 A1    >138           STY   FACLO
8F69: 64 A0    >143           STZ   FACMO
8F6B: C9 C7    >145           CMP   #TOKSTEP
8F6D: 18       >146           CLC
8F6E: D0 07    >147           BNE   *+9
8F70: 20 2A 76 >148           JSR   RST100
8F73: 38       >149           SEC
8F74: 20 CF 91 >150           JSR   LBS033
8F77: 08       >151           PHP
8F78: A0 01    >152           LDY   #1          Step value > 0 par defaut
8F7A: B0 09    >153           BCS   :1          Branch iif inversion de signe
8F7C: A5 A0    >154           LDA   FACMO
8F7E: 30 05    >155           BMI   :1
8F80: 05 A1    >156           ORA   FACLO
8F82: D0 03    >157           BNE   :2
8F84: 24       >158           HEX   24          Skip next byte
8F85: 88       >159  :1       DEY
8F86: 88       >160           DEY
8F87: 98       >161  :2       TYA
8F88: 49 80    >162           EOR   #$80        Tag for integer var.
8F8A: 29 C3    >163           AND   #%11000011
8F8C: 2C E7 9C >164           BIT   WMODE
8F8F: 10 02    >165           BPL   *+4
8F91: 09 20    >166           ORA   #%00100000 Set Unsigned arith. flag
8F93: 28       >167           PLP
8F94: 90 02    >168           BCC   *+4
8F96: 09 10    >169           ORA   #%00010000 Set reverse step value sign
8F98: 20 BA 90 >170  COMFOR   JSR   NFRMSTK2
8F9B: 4C C9 D7 >175           JMP   $D7C9
               >177
               >178  * Incrementation de l´index d´elm.
8F9E: EE 23 96 >179  ]LOOP    INC   AEI         Incrementation de l´index d´elm
8FA1: D0 03    >180           BNE   *+5
8FA3: EE 24 96 >181           INC   AEI+1
               >182  * From new array element to loop var (value)
8FA6: 20 F8 90 >183           JSR   LBS63
8FA9: A4 5E    >184           LDY   INDEX       Write back $9D,$9E to stack
8FAB: 20 71 91 >185           JSR   LBS67
8FAE: BA       >186           TSX
8FAF: 4C 3E DD >187           JMP   $DD3E
               >188
8FB2: 20 5E 91 >189  FENEXT   JSR   LBS64       Step FP value into FAC
```

```
8FB5: 20 9C 91 >190           JSR   LBS69      From offset(ARYTAB) to absol.
8FB8: 20 29 91 >191           JSR   LBS65      Loop var. back into array elm.
8FBB: A5 9F    >192           LDA   $9F
8FBD: 18       >193           CLC
8FBE: 65 9D    >194           ADC   $9D
8FC0: 85 9D    >195           STA   $9D
8FC2: 90 02    >196           BCC   *+4
8FC4: E6 9E    >197           INC   $9E
               >198  * Loop exhausted?
8FC6: C5 A0    >199           CMP   $A0
8FC8: A5 9E    >200           LDA   $9E
8FCA: E5 A1    >201           SBC   $A1
               >202  * Carry set iif loop exhausted
8FCC: 90 D0    >203           BCC   ]LOOP
8FCE: 9C 23 96 >205           STZ   AEI
8FD1: 9C 24 96 >206           STZ   AEI+1
8FD4: BA       >212           TSX
8FD5: 4C 8E 90 >213           JMP   COMNEXT    Always
               >214
8FD8: 4C 0B DD >215  ]LOOP    JMP   $DD0B      NEXT WITHOUT FOR error
8FDB: D0 04    >216  RNEXT    BNE   NEXT1
8FDD: A0 00    >217           LDY   #0
8FDF: F0 03    >218           BEQ   *+5
8FE1: 20 06 7A >219  NEXT1    JSR   NPTRGTX
8FE4: 85 85    >220           STA   FORPNT
8FE6: 84 86    >221           STY   FORPNT+1
8FE8: 20 65 D3 >222           JSR   $D365
8FEB: D0 EB    >223           BNE   ]LOOP
8FED: 9A       >224           TXS
8FEE: E8       >226           INX
8FEF: E8       >226           INX
8FF0: E8       >226           INX
8FF1: E8       >226           INX
8FF2: 8A       >228           TXA              ;Base address of STEP value
8FF3: E8       >230           INX
8FF4: E8       >230           INX
8FF5: E8       >230           INX
8FF6: E8       >230           INX
8FF7: E8       >230           INX
8FF8: E8       >230           INX
8FF9: 86 60    >232           STX   DEST       Base adress of TO value
8FFB: A8       >233           TAY
8FFC: BA       >234           TSX
8FFD: BD 09 01 >235           LDA   $0109,X
9000: 85 C0    >236           STA   GFLAG
9002: 0A       >237           ASL
9003: 90 08    >238           BCC   :1
9005: 10 08    >239           BPL   :2
9007: 98       >240  ]LOOP    TYA
9008: A6 60    >241           LDX   DEST
900A: 4C 1D DD >242           JMP   $DD1D      FP var: classic mechanic
900D: 10 F8    >243  :1       BPL   ]LOOP
900F: 29 08    >244  :2       AND   #%00001000 Voir ASL precedent..
9011: D0 9F    >245           BNE   FENEXT
9013: A2 00    >246           LDX   #0
9015: 20 A3 90 >247           JSR   LBS05      Step value into $A0, $A1
9018: D0 02    >248           BNE   *+4
```

```
901A: A2 04    >249           LDX   #4
901C: 50 01    >250           BVC   *+3
901E: E8       >251           INX
901F: 90 04    >252           BCC   *+6
9021: 8A       >253           TXA
9022: 09 08    >254           ORA   #8
9024: AA       >255           TAX
9025: 20 69 77 >256           JSR   HNDLEIY    Current value in FORPNT
9028: A2 FF    >257           LDX   #-1
902A: A4 60    >258           LDY   DEST
902C: 20 A7 90 >259           JSR   LBS051
902F: 08       >260           PHP
9030: A2 FF    >261           LDX   #-1        endvalue > FAC par defaut
9032: A0 01    >262           LDY   #1
9034: A5 A1    >263           LDA   $A1
9036: 28       >264           PLP
               >265  * A: -1 iif endvalue > current value
               >266  * A: 0 iif endvalue = current value
               >267  * A: 1 iif endvalue < current value
9037: 90 1D    >268           BCC   :SI        Branch iif signed arithmetic
9039: D0 0B    >269           BNE   :7
903B: F2 85    >271           SBC   (FORPNT)
903D: F0 03    >276           BEQ   *+5
903F: B0 02    >277           BCS   *+4
9041: E8       >278           INX
9042: E8       >279           INX
9043: 8A       >280  ]LOOP    TXA
9044: 80 33    >281           BRA   :10
9046: F1 85    >282  :7       SBC   (FORPNT),Y
9048: 85 3C    >283           STA   A1L
904A: A5 A0    >284           LDA   $A0
904C: F2 85    >286           SBC   (FORPNT)
904E: 05 3C    >291           ORA   A1L
9050: F0 F0    >292           BEQ   ]LOOP-1
9052: B0 EF    >293           BCS   ]LOOP
9054: 90 EB    >294           BCC   ]LOOP-2    Always
               >295
               >296  * Signed arithetic comparison
9056: 38       >297  :SI       SEC
9057: D0 07    >298           BNE   :6
9059: F2 85    >300           SBC   (FORPNT)
905B: D0 0C    >305           BNE   :5
905D: E8       >306           INX
905E: 80 09    >307           BRA   :5
9060: F1 85    >308  :6       SBC   (FORPNT),Y
9062: D0 01    >309           BNE   *+3
9064: E8       >310           INX
9065: A5 A0    >311           LDA   $A0
9067: F2 85    >316           SBC   (FORPNT)
9069: 70 0C    >318  :5       BVS   :C1
906B: 30 07    >319           BMI   :LT
906D: D0 02    >320  ]LOOP    BNE   :C20
906F: 8A       >321           TXA              ;A=0 if both bytes equal
9070: 2C       >322           HEX   2C         next two bytes
9071: A9 FF    >323  :C20     LDA   #-1
9073: 2C       >324           HEX   2C
9074: A9 01    >325  :LT      LDA   #1
```

```
9076: 2C       >326           HEX   2C         Skip next two bytes
9077: 10 F4    >327  :C1      BPL   ]LOOP
9079: A8       >328  :10      TAY
907A: A5 C0    >329           LDA   GFLAG
907C: 29 03    >330           AND   #%00000011
907E: AA       >331           TAX
907F: BD 1F 96 >332           LDA   MOTGF,X
9082: 85 C0    >333           STA   GFLAG
9084: 98       >334           TYA
9085: BA       >335           TSX
9086: 38       >336           SEC
9087: E5 C0    >337           SBC   GFLAG
9089: F0 03    >338           BEQ   *+5
908B: 4C 3E DD >339           JMP   $DD3E      Processing next loop iteration
908E: 8A       >340  COMNEXT  TXA               ;Arithmetic of frame pointer
908F: 69 11    >341           ADC   #17        Carry set so add 18
9091: AA       >342           TAX
9092: 9A       >343           TXS
9093: 20 32 76 >344           JSR   RST102
9096: C9 2C    >345           CMP   #´,´
9098: D0 06    >346           BNE   *+8
909A: 20 2A 76 >347           JSR   RST100
909D: 20 E1 8F >348           JSR   NEXT1      Does not return
90A0: 4C D2 D7 >349           JMP   NEWSTT
               >350
90A3: A9 01    >351  LBS05    LDA   #1
90A5: 85 5F    >352           STA   INDEX+1
90A7: 20 62 91 >353  LBS051   JSR   LBS641
90AA: A5 C0    >354           LDA   GFLAG
90AC: 0A       >355           ASL
90AD: 0A       >356           ASL
90AE: 0A       >357           ASL               ;Unsigned into carry and reverse
into ovf
90AF: B8       >358           CLV
90B0: 10 03    >359           BPL   *+5
90B2: 2C 0C 8E >360           BIT   ]RET
90B5: B1 85    >361           LDA   (FORPNT),Y Y a 4: pointe sur le subtype
90B7: 49 81    >362           EOR   #$81       Z a 1 ssi BYTE
90B9: 60       >363           RTS
               >364
               >365  NFRMSTK2
90BA: A8       >366           TAY               ;FAC sign or SGN(step value)
90BB: FA       >367           PLX
90BC: 68       >368           PLA
90BD: E8       >369           INX
90BE: 86 5E    >370           STX   INDEX
90C0: D0 01    >371           BNE   :1
90C2: 1A       >373           INC
90C3: 85 5F    >378  :1       STA   INDEX+1
90C5: 5A       >379           PHY
90C6: 4C 23 DE >380           JMP   FRMSTCK3+3
               >381
               >382  * Analyse array: 1st array elm into $9D,9E and
               >383  * address of next array in $A0,A1.
90C9: A0 04    >384  LBS61    LDY   #4
90CB: B1 9B    >385           LDA   (LOWTR),Y
90CD: 29 07    >386           AND   #7          Isolate # of dims.
```

```
90CF: 0A       >387           ASL             ;2 bytes per dimensions
90D0: 69 05    >388           ADC   #5        Carry clear
90D2: 65 9B    >389           ADC   LOWTR
90D4: 85 9D    >390           STA   $9D
90D6: A9 00    >391           LDA   #0
90D8: 65 9C    >395           ADC   LOWTR+1
90DA: 85 9E    >396           STA   $9E
90DC: A0 02    >397           LDY   #2
90DE: B1 9B    >398           LDA   (LOWTR),Y
90E0: C8       >399           INY
90E1: 65 9B    >400           ADC   LOWTR
90E3: 85 A0    >401           STA   $A0
90E5: B1 9B    >402           LDA   (LOWTR),Y
90E7: 65 9C    >403           ADC   LOWTR+1
90E9: 85 A1    >404           STA   $A1
               >405  * Taille d´un element
90EB: 20 66 7D >406           JSR   KWELMSIZ
90EE: 86 9F    >407           STX   $9F
90F0: 60       >408           RTS
               >409
90F1: 85 5E    >410  LBS62    STA   INDEX
90F3: 84 5F    >411           STY   INDEX+1
90F5: 4C 23 DE >412           JMP   FRMSTCK3+3
               >413
               >414  * From array element to loop var.
90F8: A4 9F    >415  LBS63    LDY   $9F
90FA: C0 03    >416           CPY   #3
90FC: F0 09    >417           BEQ   :0        Special handling for strings
90FE: 88       >418           DEY
90FF: B1 9D    >419  ]LOOP    LDA   ($9D),Y
9101: 91 85    >420           STA   (FORPNT),Y
9103: 88       >421           DEY
9104: 10 F9    >422           BPL   ]LOOP
9106: 60       >423  ]RET     RTS
               >424  * Special handling for strings
9107: B2 9D    >426  :0       LDA   ($9D)     Length byte
9109: 92 85    >427           STA   (FORPNT)
910B: F0 F9    >433           BEQ   ]RET      Nothing to do if length zero
910D: 48       >434           PHA
910E: 20 53 91 >435           JSR   LBS66
9111: 91 85    >436           STA   (FORPNT),Y
               >437  * A1L,A1H: adresse source
9113: B1 9D    >438           LDA   ($9D),Y
9115: 85 3D    >439           STA   A1L+1
9117: 88       >440           DEY
9118: 8A       >441           TXA
9119: 91 85    >442           STA   (FORPNT),Y
911B: B1 9D    >443           LDA   ($9D),Y
911D: 85 3C    >444           STA   A1L
               >445  * Do the string copy itself: recall string length
911F: 7A       >447           PLY
9120: 88       >448  COMCOPY  DEY
9121: B1 3C    >454  ]LOOP    LDA   (A1L),Y
9123: 91 3E    >455           STA   (A2L),Y
9125: 88       >456           DEY
9126: 10 F9    >457           BPL   ]LOOP
9128: 60       >458           RTS
```

```
                >459
                >460  * From loop var. to array elm.
9129: A4 9F     >461  LBS65    LDY   $9F
912B: C0 03     >462          CPY   #3
912D: F0 09     >463          BEQ   :0         Special handling for strings
912F: 88        >464          DEY
9130: B1 85     >465  ]LOOP    LDA   (FORPNT),Y
9132: 91 9D     >466          STA   ($9D),Y
9134: 88        >467          DEY
9135: 10 F9     >468          BPL   ]LOOP
9137: 60        >469  ]RET     RTS
                >470  * Special handling for strings
9138: B2 85     >472  :0       LDA   (FORPNT)   Length byte
913A: 92 9D     >473          STA   ($9D)
913C: F0 F9     >479          BEQ   ]RET       Nothing to do if length zero
913E: 48        >480          PHA
913F: 20 53 91  >481          JSR   LBS66
9142: 91 9D     >482          STA   ($9D),Y    High byte
                >483  * A1L,A1H: adresse source
9144: B1 85     >484          LDA   (FORPNT),Y
9146: 85 3D     >485          STA   A1L+1
9148: 88        >486          DEY
9149: 8A        >487          TXA
914A: 91 9D     >488          STA   ($9D),Y
914C: B1 85     >489          LDA   (FORPNT),Y
914E: 85 3C     >490          STA   A1L
                >491  * Do the string copy itself: recall string length
9150: 7A        >493          PLY
9151: D0 CD     >497          BNE   COMCOPY    Always
                >498
9153: 20 52 E4  >499  LBS66    JSR   GETSPA
                >500  * returns with Y,X pointer to new string
                >501  * A2L,A2H: adresse destination
9156: 84 3F     >502          STY   A2L+1
9158: 86 3E     >503          STX   A2L
915A: 98        >504          TYA
915B: A0 02     >505          LDY   #2
915D: 60        >506          RTS
                >507
                >508  * Subroutine: copy from stack to FAC in page zero
915E: A9 01     >509  LBS64    LDA   #1
9160: 85 5F     >510          STA   INDEX+1
9162: 84 5E     >511  LBS641   STY   INDEX
9164: A0 FF     >512          LDY   #-1
9166: C8        >513  ]LOOP    INY
9167: B1 5E     >514          LDA   (INDEX),Y
9169: 99 9D 00  >515          STA   $9D,Y
916C: C0 04     >516          CPY   #4
916E: 90 F6     >517          BCC   ]LOOP
9170: 60        >518          RTS
                >519
                >520  * From FAC to stack.. called from FENEXT
                >521  * $9D is expected to be in absolute mode
9171: A9 01     >522  LBS67    LDA   #1
9173: 85 5F     >523          STA   INDEX+1
9175: 84 5E     >524          STY   INDEX
9177: A5 9D     >525          LDA   $9D        Convert $9D$9E
```

```
9179: 38       >526           SEC              ; to offset(ARYTAB)
917A: E5 6B    >527           SBC   ARYTAB
917C: 92 5E    >529           STA   (INDEX)
917E: A0 01    >530           LDY   #1
9180: A5 9E    >536           LDA   $9E
9182: E5 6C    >537           SBC   ARYTAB+1
9184: 91 5E    >538           STA   (INDEX),Y
9186: 60       >539           RTS
               >540
               >541  * From absolute address to offset from ARYTAB
9187: A2 A0    >542  LBS68     LDX   #$A0
9189: 20 8E 91 >543           JSR   *+5
918C: A2 9D    >544           LDX   #$9D
918E: B5 00    >545           LDA   0,X
9190: 38       >546           SEC
9191: E5 6B    >547           SBC   ARYTAB
9193: 95 00    >548           STA   0,X
9195: B5 01    >549           LDA   1,X
9197: E5 6C    >550           SBC   ARYTAB+1
9199: 95 01    >551           STA   1,X
919B: 60       >552           RTS
               >553
               >554  * From offset to absolute address
919C: A0 A0    >555  LBS69     LDY   #$A0
919E: 20 A3 91 >556           JSR   *+5
91A1: A0 9D    >557           LDY   #$9D
91A3: B9 00 00 >558           LDA   0,Y
91A6: 18       >559           CLC
91A7: 65 6B    >560           ADC   ARYTAB
91A9: 99 00 00 >561           STA   0,Y
91AC: B9 01 00 >562           LDA   1,Y
91AF: 65 6C    >563           ADC   ARYTAB+1
91B1: 99 01 00 >564           STA   1,Y
91B4: 60       >565           RTS
               >566
               >567  * Return with Z flag set iif ´EACH´ string @ TXTPTR
               >568  * TXTPTR updated accordngly if so
91B5: A0 FF    >569  ITEACH    LDY   #-1
91B7: C8       >570  ]LOOP     INY
91B8: B1 B8    >571           LDA   (TXTPTR),Y
91BA: D9 59 9B >572           CMP   IFEACH,Y
91BD: D0 0F    >573           BNE   :0
91BF: C0 03    >574           CPY   #3
91C1: D0 F4    >575           BNE   ]LOOP
91C3: 98       >576           TYA
91C4: 65 B8    >577           ADC   TXTPTR
91C6: 85 B8    >578           STA   TXTPTR
91C8: 90 02    >579           BCC   *+4
91CA: E6 B9    >580           INC   TXTPTR+1
91CC: A0 00    >581           LDY   #0         Set Zflag
91CE: 60       >582  :0        RTS
               >583
91CF: 20 49 78 >584  LBS033    JSR   LBS03
91D2: 08       >585           PHP
91D3: A5 C0    >586           LDA   GFLAG
91D5: C9 81    >587           CMP   #$81
91D7: D0 03    >588           BNE   :0
```

```
91D9: 20 EA 79 >589           JSR   CONV1628
91DC: 28       >590  :0       PLP
91DD: 60       >591           RTS
               >592
               >593  * a) Enough space on stack?
91DE: A9 07    >594  LBS60    LDA   #9-2       -2 car on est dans une SUBR
91E0: 20 D6 D3 >595           JSR   CHKMEM
               >596  * b) Debut du calcul du nouveau TXTPTR
               >597  * Comme c´est une operation avec la pile, oblige de
               >598  * morceler l´operation
91E3: 20 A3 D9 >599           JSR   DATAN      Prochain separateur (offset Y)
91E6: 18       >600           CLC
91E7: 98       >601           TYA
91E8: 65 B8    >602           ADC   TXTPTR
91EA: 60       >603           RTS
                1102           PUT   PEERGOTO
               >1    * Module in charge of accelerating GOTO/GOSUB line address
               >2    * computations.
               >3    TXTTAB   EQU   $67
               >4    TOKTHEN  =     $C4
               >5    GOTOTAIL EQU   $D95E
               >6    FOUT     EQU   $ED34
               >7    RD2      EQU   $A47A      Read 2 first bytes from file
               >8
               >9    EXFLG    EQU   $AAB3      Exec file activity flag
               >10   WHCBASIC EQU   $AAB6      0 iif Integer BASIC active
               >11   ISBASRUN EQU   $A65E
               >12   * Part of the DOS 3.3 keyboard intercept routine
91EB: AD B6 AA >13   NKBDINT  LDA   WHCBASIC
91EE: F0 10    >14           BEQ   :0
91F0: 20 5E A6 >15           JSR   ISBASRUN
91F3: 90 0B    >16           BCC   :0         program running
91F5: AD D0 9C >17           LDA   OPTCGOTO
91F8: 2D CF 9C >18           AND   NEEDDEC
91FB: 10 03    >19           BPL   :0
91FD: 20 C1 93 >20           JSR   DECOMPILE
9200: AD B3 AA >21   :0       LDA   EXFLG
9203: 60       >22           RTS
               >23
               >24   * New DOS Applesoft SAVE command handler (or part of)
9204: 20 C1 93 >25   NDSVCMD  JSR   DECOMPILE
9207: A9 02    >26           LDA   #2         Restore original A value..
9209: 4C D5 A3 >27           JMP   $A3D5      Fall into $A3D5 (orig. content)
               >28
               >29   * Reset NEEDDEC upon DOS 3.3 Applesoft program loading
               >30   NDLVCMD  DO    KOPT-K6502
920C: 9C CF 9C >31           STZ   NEEDDEC
920F: 4C 7A A4 >36           JMP   RD2
               >37
9212: 9D D8 9B >38   ROUT8C   STA   ADPFB,X
9215: 98       >39           TYA
9216: 9D EC 9B >40           STA   ADPFT,X
9219: E8       >41           INX
921A: 60       >42   ]RET     RTS
               >43   * Programmer routine to set the precomputed GOTO behavior
               >44   * CALL RE!,8,<n>
               >45   * with n being 0 to inactivate precomputed GOTOs,
```

```
                        >46   * 128 to activate precomuted GOTOs w/o safeguard option
                        >47   * 192 to activate precomputed GOTOs w safeguard option.
921B: 20 A6 86          >48   ROUT8    JSR   NCHKCOM
921E: 20 D0 86          >49           JSR   NGETBYT    Reason code in X
9221: 8E D0 9C          >50           STX   OPTCGOTO
9224: 8A                >51           TXA
9225: A2 0D             >52           LDX   #OFSTGTO-ADPFB
9227: A8                >53           TAY
9228: 10 16             >54           BPL   :2
922A: A9 13             >55           LDA   #RGOTO-1
922C: A0 93             >56           LDY   #>RGOTO-1
922E: 20 12 92          >57           JSR   ROUT8C
9231: A9 EE             >58           LDA   #RIF-1
9233: A0 92             >59           LDY   #>RIF-1
9235: 20 12 92          >60           JSR   ROUT8C
9238: E8                >61           INX
9239: A9 CC             >62           LDA   #RGOSUB-1
923B: A0 92             >63           LDY   #>RGOSUB-1
923D: 20 12 92          >64           JSR   ROUT8C
9240: 2C D0 9C          >65   :2       BIT   OPTCGOTO
9243: 30 18             >66           BMI   :3
9245: 08                >67           PHP
9246: A9 3D             >68           LDA   #APRGOTO-1
9248: A0 D9             >69           LDY   #>APRGOTO-1
924A: 20 12 92          >70           JSR   ROUT8C
924D: A9 C8             >71           LDA   #APRIF-1
924F: A0 D9             >72           LDY   #>APRIF-1
9251: 20 12 92          >73           JSR   ROUT8C
9254: E8                >74           INX
9255: A9 20             >75           LDA   #APRGOSUB-1
9257: A0 D9             >76           LDY   #>APRGOSUB-1
9259: 20 12 92          >77           JSR   ROUT8C
925C: 28                >78           PLP
925D: 70 02             >79   :3       BVS   :0
925F: 30 B9             >80           BMI   ]RET
9261: 4C C1 93          >81   :0       JMP   DECOMPILE  in case reason code 0 or 192
                        >82
9264: 4C C9 DE          >83   ]ERR     JMP   SYNERR
9267: A2 01             >84   RON      LDX   #1
9269: 20 5F 8B          >85           JSR   COMLBS
926C: F0 35             >86           BEQ   :1
                        >87   * Function call: normal flow
926E: B1 B8             >88           LDA   (TXTPTR),Y
9270: C9 28             >89           CMP   #´(´
9272: F0 2F             >90           BEQ   :1         Normal function
                        >91   * ON MOUSE GOSUB or ON TIMER GOSUB pattern
9274: 20 98 D9          >92           JSR   ADDON
9277: A9 B0             >93           LDA   #TOKGOSUB
9279: 20 76 7E          >94           JSR   NSYNCHR
927C: 20 1B 93          >95           JSR   RGPART1    LOWTR: address of target line
927F: A5 BD             >96           LDA   IDMOCL
9281: 38                >97           SEC
9282: E9 08             >98           SBC   #OFFMOU-TOFFST
9284: AA                >99           TAX
9285: A5 9B             >100          LDA   LOWTR
9287: E9 01             >101          SBC   #1         Carry already set
9289: 9D BD 99          >102          STA   AHNDLO,X
```

```
928C: A5 9C    >103           LDA   LOWTR+1
928E: E9 00    >104           SBC   #0
9290: 9D BF 99 >105           STA   AHNDHI,X
9293: A5 50    >106           LDA   LINNUM
9295: 9D B9 99 >107           STA   CLNLO,X
9298: A5 51    >108           LDA   LINNUM+1
929A: 9D BB 99 >109           STA   CLNHI,X
929D: 20 32 76 >110           JSR   RST102
92A0: D0 C2    >111           BNE   ]ERR
92A2: 60       >112           RTS
92A3: 20 D0 86 >113  :1       JSR   NGETBYT
92A6: C9 B0    >114           CMP   #TOKGOSUB
92A8: F0 04    >115           BEQ   :2
92AA: 49 AB    >116           EOR   #TOKGOTO   TOKGOTO being < TOKGOSUB
92AC: D0 B6    >117           BNE   ]ERR       carry is already clear
92AE: 08       >118  :2       PHP
92AF: C6 A1    >119  ]LOOP    DEC   FAC+4
92B1: D0 0D    >120           BNE   :3
92B3: 28       >121           PLP
               >122  * Carry set iif GOSUB, else GOTO (carry clear)
92B4: 90 05    >123           BCC   :GOTO
92B6: 20 2A 76 >124           JSR   RST100
92B9: 80 12    >125           BRA   RGOSUB
92BB: 20 2A 76 >126  :GOTO    JSR   RST100
92BE: 80 54    >127           BRA   RGOTO
92C0: 20 99 94 >128  :3       JSR   LRST100
92C3: 90 FB    >129           BCC   :3         Loop till not digit
92C5: E0 2C    >130           CPX   #´,´
92C7: F0 E6    >131           BEQ   ]LOOP
92C9: 28       >132           PLP
92CA: 4C 65 94 >133           JMP   NDATAN
               >134
92CD: 08       >135  RGOSUB   PHP
92CE: 48       >136           PHA
92CF: A9 02    >137           LDA   #3-1       -1 because of PLA PLP below..
92D1: 20 D6 D3 >138           JSR   CHKMEM
92D4: 68       >139           PLA
92D5: 28       >140           PLP
92D6: 20 1B 93 >141           JSR   RGPART1
92D9: A5 B9    >146           LDA   TXTPTR+1
92DB: 48       >147           PHA
92DC: A5 B8    >148           LDA   TXTPTR
92DE: 48       >149           PHA
92DF: A5 76    >150           LDA   CURLIN+1
92E1: 48       >151           PHA
92E2: A5 75    >152           LDA   CURLIN
92E4: 48       >153           PHA
92E5: A9 B0    >155           LDA   #TOKGOSUB
92E7: 48       >156           PHA
92E8: 38       >157           SEC
92E9: 20 5E D9 >158           JSR   GOTOTAIL
92EC: 4C D2 D7 >159           JMP   NEWSTT
               >160
92EF: 20 7B DD >161  RIF      JSR   FRMEVL
92F2: A5 9D    >162           LDA   FAC
92F4: F0 0D    >163           BEQ   :20
92F6: B2 B8    >165           LDA   (TXTPTR)
```

```
92F8: C9 AB    >170           CMP   #TOKGOTO
92FA: F0 13    >171           BEQ   :4
92FC: C9 C4    >172           CMP   #TOKTHEN
92FE: F0 0F    >173           BEQ   :4
9300: 4C 2B 7C >174           JMP   SNERR
9303: 20 68 94 >175  :20      JSR   NREMN
9306: 4C 98 D9 >176           JMP   ADDON
9309: 20 A9 8B >177  :3       JSR   LBS10
930C: 4C 28 D8 >178           JMP   $D828
930F: 20 2A 76 >179  :4       JSR   RST100
9312: B0 F5    >180           BCS   :3
               >181
9314: 20 1B 93 >182  RGOTO    JSR   RGPART1
9317: 38       >183           SEC
9318: 4C 5E D9 >184           JMP   GOTOTAIL
               >185
               >186  * First part of GOTO..
               >187  * Upon entry: A contains first target line no. char.,
               >188  * C clear iif this character is a numeric digit.
               >189  * Upon exit: LOWTR set to base adress of target line,
               >190  * LINNUM set to target line no.
931B: 90 2A    >191  RGPART1  BCC   :2         if num. digit then process it
931D: C9 20    >192           CMP   #$20
931F: 90 03    >193           BCC   *+5
9321: 4C 2B 7C >194  :11      JMP   SNERR
               >195  * Offset of target line from beginning of program
               >196  * already computed (value within program text).
9324: E9 1C    >197           SBC   #$1D-1
9326: A8       >198           TAY
9327: C8       >199           INY
9328: B1 B8    >200           LDA   (TXTPTR),Y lo byte
932A: 18       >201           CLC
932B: 65 67    >202           ADC   TXTTAB     to absolute address lo byte
932D: 85 9B    >203           STA   LOWTR
932F: C8       >204           INY
9330: B1 B8    >205           LDA   (TXTPTR),Y hi byte
9332: 65 68    >206           ADC   TXTTAB+1   to absolute address
9334: 85 9C    >207           STA   LOWTR+1
9336: C8       >208           INY
9337: 5A       >212           PHY
9338: A0 02    >214           LDY   #2
933A: B1 9B    >215           LDA   (LOWTR),Y
933C: 85 50    >216           STA   LINNUM
933E: C8       >217           INY
933F: B1 9B    >218           LDA   (LOWTR),Y
9341: 85 51    >219           STA   LINNUM+1
9343: 7A       >223           PLY
9344: 4C 98 D9 >225           JMP   ADDON      Add Y to TXTPTR
9347: A6 B8    >226  :2       LDX   TXTPTR     Backup TXTPTR
9349: 86 06    >227           STX   AUXPTR     before calling LINGET
934B: A6 B9    >228           LDX   TXTPTR+1
934D: 86 07    >229           STX   AUXPTR+1
934F: 20 0C DA >230           JSR   LINGET
               >231  * Now TXTPTR points to the first non numeric character
               >232  * following line no: computes the offset from current
               >233  * to stored position.
9352: 20 68 94 >234           JSR   NREMN      Compute Y offset to EOL
```

```
9355: A5 76    >235           LDA   CURLIN+1
9357: C5 51    >236           CMP   LINNUM+1
9359: B0 0C    >237           BCS   :1
935B: 98       >238           TYA
935C: 38       >239           SEC
935D: 65 B8    >240           ADC   TXTPTR
935F: A6 B9    >241           LDX   TXTPTR+1
9361: 90 08    >242           BCC   :3
9363: E8       >243           INX
9364: B0 05    >244           BCS   :3         Always
9366: 60       >245  ]RET     RTS
9367: A5 67    >246  :1        LDA   TXTTAB
9369: A6 68    >247           LDX   TXTTAB+1
936B: 20 1A D6 >248  :3        JSR   FNDLIN
936E: 90 4E    >249           BCC   GOUNDEF
9370: 2C D0 9C >250           BIT   OPTCGOTO
9373: 10 F1    >251           BPL   ]RET       Optimization deactivated
9375: A5 B8    >252           LDA   TXTPTR
9377: E5 06    >253           SBC   AUXPTR
9379: A8       >254           TAY
               >255  * Y should be 3, 4 or 5 (line no from 100 to 99999)
937A: A5 B9    >256           LDA   TXTPTR+1
937C: E5 07    >257           SBC   AUXPTR+1   Carry deja a 1
937E: D0 E6    >258           BNE   ]RET       hi byte must be zero
9380: 88       >259           DEY
9381: 88       >260           DEY
9382: 88       >261           DEY
9383: 30 E1    >262           BMI   ]RET       If Y was below 3
9385: C0 03    >263           CPY   #3         If Y was above 5
9387: B0 DD    >264           BCS   ]RET
9389: 84 B5    >265           STY   YSAV       possible values: 0, 1 or 2
938B: A5 9B    >266           LDA   LOWTR
938D: 38       >267           SEC
938E: E5 67    >268           SBC   TXTTAB
9390: AA       >269           TAX
9391: A5 9C    >270           LDA   LOWTR+1
9393: E5 68    >271           SBC   TXTTAB+1   Leaves carry always set..
9395: 2C D0 9C >272           BIT   OPTCGOTO
9398: 50 0F    >273           BVC   :6          Configured to skip checks..
939A: A8       >274           TAY              ;Set Z flag after BIT op
939B: 20 60 94 >275           JSR   COMRG
939E: F0 C6    >276           BEQ   ]RET
93A0: 8A       >277           TXA
93A1: 20 60 94 >278           JSR   COMRG
93A4: F0 C0    >279           BEQ   ]RET
93A6: 98       >280           TYA
93A7: A4 B5    >281           LDY   YSAV
93A9: C8       >282  :6        INY
93AA: C8       >283           INY
93AB: 91 06    >284           STA   (AUXPTR),Y
93AD: 88       >285           DEY
93AE: 8A       >286           TXA
93AF: 91 06    >287           STA   (AUXPTR),Y
93B1: 88       >288           DEY
93B2: 98       >289           TYA
93B3: 69 1C    >290           ADC   #$1D-1     Carry originally set
93B5: 91 06    >291  ]LOOP    STA   (AUXPTR),Y
```

```
93B7: 88       >292           DEY
93B8: 10 FB    >293           BPL   ]LOOP
93BA: 8C CF 9C >294           STY   NEEDDEC    Set "Need Decompile" indic.
93BD: 60       >295  ]RET     RTS
               >296
93BE: 4C 7C D9 >297  GOUNDEF  JMP   $D97C
               >298
               >299  * Routine to restore things at their original state
               >300  * This routine should be called upon LIST or a SAVE
               >301  * command under DOS 3.3.
               >302  DECOMPILE
93C1: 08       >303           PHP
93C2: 48       >304           PHA
93C3: 2C CF 9C >305           BIT   NEEDDEC
93C6: 10 3F    >306           BPL   FINDEC
93C8: A5 67    >307           LDA   TXTTAB
93CA: A6 68    >308           LDX   TXTTAB+1
93CC: A0 00    >309           LDY   #0
93CE: 8C CF 9C >310           STY   NEEDDEC
93D1: 85 06    >311  ]LOOP    STA   AUXPTR
93D3: 86 07    >312           STX   AUXPTR+1
93D5: 84 C0    >313           STY   GFLAG      Set b7 to 0
93D7: 8A       >314           TXA
93D8: F0 2D    >315           BEQ   FINDEC
93DA: A0 03    >316           LDY   #3
93DC: C8       >317  ]LOOP1   INY
93DD: B1 06    >318  ]LOOP2   LDA   (AUXPTR),Y
93DF: F0 1D    >319           BEQ   FINLIGNE
93E1: C9 22    >320           CMP   #´"´
93E3: D0 08    >321           BNE   :0
93E5: AA       >322           TAX
93E6: A5 C0    >323           LDA   GFLAG
93E8: 49 80    >324           EOR   #$80
93EA: 85 C0    >325           STA   GFLAG
93EC: 8A       >326           TXA
93ED: 24 C0    >327  :0       BIT   GFLAG
93EF: 30 EB    >328           BMI   ]LOOP1
93F1: C9 20    >329           CMP   #$20
93F3: B0 E7    >330           BCS   ]LOOP1
93F5: E9 1C    >331           SBC   #$1D-1
93F7: 90 E3    >332           BCC   ]LOOP1
93F9: 20 0A 94 >333           JSR   TRAITEOK
93FC: F0 DF    >334           BEQ   ]LOOP2     Always
93FE: A0 01    >335  FINLIGNE LDY   #1
9400: B1 06    >336           LDA   (AUXPTR),Y
9402: AA       >337           TAX
9403: B2 06    >339           LDA   (AUXPTR)
9405: 80 CA    >344           BRA   ]LOOP
9407: 68       >345  FINDEC   PLA
9408: 28       >346           PLP
9409: 60       >347  ]RET     RTS
               >348
               >349  * A: 0, 1 or 2 depending of length of org target line no
               >350  * Y: offset from AUXPTR where first pattern byte appeared
               >351  * Carry: must be set upon entry
940A: 85 B4    >352  TRAITEOK STA   XSAV
940C: 5A       >353           PHY
```

```
940D: 98       >354           TYA
940E: 65 B4    >355           ADC   XSAV       Carry set upon entry
9410: A8       >356           TAY
9411: 18       >357           CLC
               >358  * Now Y: offset from AUXPTR where to get the
               >359  *        target line adress offset
               >360  * CLC (carry already clear after ADC above).
9412: B1 06    >375           LDA   (AUXPTR),Y or stick to 8bits arithmetic
9414: 65 67    >376           ADC   TXTTAB
9416: 85 9B    >377           STA   LOWTR
9418: C8       >378           INY
9419: B1 06    >379           LDA   (AUXPTR),Y
941B: 65 68    >380           ADC   TXTTAB+1
941D: 85 9C    >381           STA   LOWTR+1
941F: A0 03    >382           LDY   #3
9421: B1 9B    >383           LDA   (LOWTR),Y
9423: 85 9E    >384           STA   $9E
9425: 88       >385           DEY
9426: B1 9B    >386           LDA   (LOWTR),Y
9428: 85 9F    >387           STA   $9F
942A: A2 90    >389           LDX   #$90       Get line #
942C: 38       >390           SEC              ; in ASCII form
942D: 20 A0 EB >391           JSR   $EBA0      stored into $100
9430: 20 34 ED >392           JSR   FOUT
9433: 20 57 94 >393           JSR   CLENGTH    Length of string in X
9436: 86 B5    >394           STX   YSAV
9438: 7A       >395           PLY
9439: A5 B4    >397           LDA   XSAV
943B: 1A       >398           INC
943C: 1A       >399           INC
943D: 1A       >400           INC
943E: 38       >408           SEC
943F: E5 B5    >409           SBC   YSAV
9441: AA       >410           TAX
9442: F0 08    >411           BEQ   :0
9444: A9 30    >412           LDA   #´0´
9446: 91 06    >413  ]LOOP    STA   (AUXPTR),Y
9448: C8       >414           INY
9449: CA       >415           DEX
944A: D0 FA    >416           BNE   ]LOOP
944C: BD 00 01 >417  :0       LDA   $0100,X
944F: F0 B8    >418           BEQ   ]RET
9451: 91 06    >419           STA   (AUXPTR),Y
9453: C8       >420           INY
9454: E8       >421           INX
9455: D0 F5    >422           BNE   :0         Always
               >423
9457: A2 FF    >424  CLENGTH  LDX   #255
9459: E8       >425  ]LOOP    INX
945A: BD 00 01 >426           LDA   $0100,X
945D: D0 FA    >427           BNE   ]LOOP
945F: 60       >428           RTS
               >429
               >430  * Small subroutine to test for critical offset value
               >431  * against insert into program text
9460: F0 02    >432  COMRG    BEQ   *+4
9462: 49 3A    >433           EOR   #´:´
```

```
9464: 60       >434  ]RET     RTS
               >435
               >436  CHARAC   EQU   $0D
               >437
9465: A2 3A    >438  NDATAN   LDX   #´:´
9467: 2C       >439          HEX   2C          Skip next two bytes
9468: A2 00    >440  NREMN    LDX   #0
946A: 86 0D    >441          STX   CHARAC
946C: A0 00    >442          LDY   #0
946E: 84 0E    >443          STY   ENDCHR
9470: A5 0E    >444  ]LOOP1   LDA   ENDCHR     Trick to count for Quote Parity
9472: A6 0D    >445          LDX   CHARAC     Do not stop upon ´:´ within
9474: 85 0D    >446          STA   CHARAC     a string litteral
9476: 86 0E    >447          STX   ENDCHR
9478: B1 B8    >448  ]LOOP    LDA   (TXTPTR),Y
947A: F0 E8    >449          BEQ   ]RET
947C: C5 0E    >450          CMP   ENDCHR
947E: F0 E4    >451          BEQ   ]RET
9480: C8       >452          INY
9481: C9 22    >453          CMP   #´"´
9483: F0 EB    >454          BEQ   ]LOOP1
9485: C9 20    >455          CMP   #´ ´
9487: B0 EF    >456          BCS   ]LOOP
9489: E9 1C    >457          SBC   #$1D-1     Substract $1D (carry clear)
948B: 90 EB    >458          BCC   ]LOOP      Out of scope..
948D: C8       >459          INY
948E: C8       >463  ]LOOP1   INY
948F: 3A       >467          DEC
9490: 10 FC    >469          BPL   ]LOOP1
9492: 30 E4    >470          BMI   ]LOOP      Always
               >471
9494: C8       >472  ]LOOP    INY
9495: C8       >473          INY
9496: 20 98 D9 >474          JSR   ADDON
9499: 20 2A 76 >475  LRST100  JSR   RST100
949C: AA       >476          TAX
949D: 90 0A    >477          BCC   :RETS+1
949F: E9 1D    >478          SBC   #$1D
94A1: A8       >479          TAY
94A2: 90 04    >480          BCC   :RETS
94A4: C0 03    >481          CPY   #3
94A6: 90 EC    >482          BCC   ]LOOP
94A8: 38       >483  :RETS    SEC
94A9: 60       >484          RTS
                1103          PUT   PEERRGI
               >1    INPUTFLG EQU   $15
               >2    INPTR    EQU   $7F
               >3    DATPTR   EQU   $7D
               >4    TXPSV    EQU   $87
               >5    DATLIN   EQU   $7B
               >6
               >7    IBUFFER  EQU   $0200
               >8    STRTXT   EQU   $DE81
               >9    STRPRT   EQU   $DB3D
               >10   OUTQUES  EQU   $DB5A
               >11   INLIN    EQU   $D52C
               >12   RDKEY    EQU   $FD0C
```

```
                  >13   STRLT2   EQU   $E3ED
                  >14   NXIN     EQU   $DBDC
                  >15
94AA: 20 06 E3    >16   RGET     JSR   ERRDIR
94AD: A2 01       >17           LDX   #IBUFFER+1
94AF: A0 02       >18           LDY   #>IBUFFER+1
94B1: 9C 01 02    >20           STZ   IBUFFER+1
94B4: A9 40       >25           LDA   #$40        Setup INPUTFLG
94B6: D0 32       >26           BNE   PIL         for PROCESS.INPUT.LIST: always
                  >27
94B8: C9 22       >28   RINP     CMP   #´"´        Check for optional prompt
94BA: D0 0D       >29           BNE   :1           string
94BC: 20 81 DE    >30           JSR   STRTXT
94BF: A9 3B       >31           LDA   #´;´
94C1: 20 76 7E    >32           JSR   NSYNCHR
94C4: 20 3D DB    >33           JSR   STRPRT      Print the string
94C7: 80 03       >34           BRA   :2
94C9: 20 5A DB    >35   :1       JSR   OUTQUES
94CC: 20 06 E3    >36   :2       JSR   ERRDIR
94CF: A9 2C       >37           LDA   #´,´        Prime the buffer
94D1: 8D FF 01    >38           STA   IBUFFER-1
94D4: 20 2C D5    >39           JSR   INLIN
94D7: AD 00 02    >40           LDA   IBUFFER
94DA: C9 03       >41           CMP   #$03        Control-C?
94DC: D0 0A       >42           BNE   IFZ
94DE: 4C 63 D8    >43           JMP   $D863
                  >44
94E1: A6 7D       >45   RREAD2   LDX   DATPTR
94E3: A4 7E       >46           LDY   DATPTR+1
94E5: A9 98       >47           LDA   #$98
94E7: 2C          >48           HEX   2C          Skip next two bytes
94E8: A9 00       >49   IFZ      LDA   #0
                  >50
                  >51   * For PROCESS.INPUT.LIST
94EA: 85 15       >52   PIL      STA   INPUTFLG
94EC: 86 7F       >53           STX   INPTR
94EE: 84 80       >54           STY   INPTR+1
                  >55
                  >56   * For PROCESS.INPUT.ITEM
94F0: 20 06 7A    >57   PII      JSR   NPTRGTX
94F3: 85 85       >58           STA   FORPNT
94F5: 84 86       >59           STY   FORPNT+1
94F7: A5 B8       >71           LDA   TXTPTR      Save current TXTPTR
94F9: A4 B9       >72           LDY   TXTPTR+1
94FB: 85 87       >73           STA   TXPSV
94FD: 84 88       >74           STY   TXPSV+1
94FF: A5 7F       >75           LDA   INPTR
9501: A4 80       >76           LDY   INPTR+1     Set TXTPR to point to input
9503: 85 B8       >77           STA   TXTPTR       buffer or "DATA" line
9505: 84 B9       >78           STY   TXTPTR+1
9507: 20 32 76    >80           JSR   RST102      Get character at pointer
950A: D0 1E       >81           BNE   INSTART    Not eol or colon.
950C: 24 15       >82           BIT   INPUTFLG
950E: 50 0E       >83           BVC   :1          Not doing a GET
9510: 20 0C FD    >84           JSR   RDKEY
9513: 29 7F       >85           AND   #$7F
9515: 8D 00 02    >86           STA   IBUFFER
```

```
9518: A2 FF    >87            LDX   #IBUFFER-1
951A: A0 01    >88            LDY   #>IBUFFER-1
951C: D0 08    >89            BNE   :2          Always
951E: 30 7B    >90   :1       BMI   FINDATA     doing a READ
9520: 20 5A DB >91            JSR   OUTQUES
9523: 20 DC DB >92            JSR   NXIN        Print another "?" & input a line
9526: 86 B8    >93   :2       STX   TXTPTR
9528: 84 B9    >94            STY   TXTPTR+1
952A: 20 2A 76 >95   INSTART  JSR   RST100
952D: 24 11    >96            BIT   VALTYP      String or numeric variable?
952F: 10 34    >97            BPL   :5
9531: 24 15    >98            BIT   INPUTFLG
9533: 50 09    >99            BVC   :1          Not a "GET"
9535: E8       >100           INX                ;GET
9536: 86 B8    >101           STX   TXTPTR
9538: A9 00    >102           LDA   #0
953A: 85 0D    >103           STA   CHARAC      No other terminator character
953C: F0 10    >104           BEQ   :2
953E: 85 0D    >105  :1       STA   CHARAC
9540: C9 22    >106           CMP   #´"´
9542: F0 0B    >107           BEQ   :3
9544: A5 15    >108           LDA   INPUTFLG    Applesoft bug fix
9546: F0 02    >109           BEQ   *+4
9548: A9 3A    >110           LDA   #´:´
954A: 85 0D    >111           STA   CHARAC
954C: A9 2C    >112           LDA   #´,´
954E: 18       >113  :2       CLC
954F: 85 0E    >114  :3       STA   ENDCHR
9551: A5 B8    >115           LDA   TXTPTR
9553: A4 B9    >116           LDY   TXTPTR+1
9555: 69 00    >117           ADC   #0          Skip over quotation mark, if
9557: 90 01    >118           BCC   :4          there was one
9559: C8       >119           INY
955A: 20 ED E3 >120  :4       JSR   STRLT2      Build string starting at Y,A
955D: 20 3D E7 >121           JSR   $E73D       Set TXTPTR to point at string
9560: 20 7B DA >122           JSR   $DA7B       PUTSTR
9563: 80 0F    >123           BRA   PIM
9565: 48       >124  :5       PHA
9566: AD 00 02 >125           LDA   IBUFFER     ANything in buffer?
9569: F0 59    >126           BEQ   INPFIN     No: see if READ or INPUT
956B: 68       >127  INPDATA  PLA                ;READ
956C: 20 4A EC >128           JSR   $EC4A       FIN: get FP number at TXTPTR
956F: A5 12    >129           LDA   INTTYP
9571: 20 08 77 >130           JSR   NLET2
               >131  * For PROCESS.INPUT.MORE
9574: 20 32 76 >132  PIM      JSR   RST102
9577: F0 07    >133           BEQ   :1          End of line or colon
9579: C9 2C    >134           CMP   #´,´        Comma in input?
957B: F0 03    >135           BEQ   :1          Yes
957D: 4C 71 DB >136           JMP   $DB71       Nothing else will do
9580: A5 B8    >148  :1       LDA   TXTPTR
9582: A4 B9    >149           LDY   TXTPTR+1
9584: 85 7F    >150           STA   INPTR
9586: 84 80    >151           STY   INPTR+1
9588: A5 87    >152           LDA   TXPSV       Restore program pointer
958A: A4 88    >153           LDY   TXPSV+1
958C: 85 B8    >154           STA   TXTPTR
```

```
958E: 84 B9    >155           STY   TXTPTR+1
9590: 20 32 76 >157           JSR   RST102     next char from program
9593: F0 36    >158           BEQ   INPDONE    End if statement
9595: 20 A6 86 >159           JSR   NCHKCOM
9598: 4C F0 94 >160           JMP   PII
               >161
959B: 20 A3 D9 >162  FINDATA  JSR   DATAN      Get offset to next colon/eol
959E: C8       >163           INY
959F: AA       >164           TAX              ;Which colon or eol?
95A0: D0 15    >165           BNE   :1         Colon
95A2: C8       >166           INY              ;Check hi byte
95A3: B1 B8    >167           LDA   (TXTPTR),Y
95A5: D0 05    >168           BNE   *+7
95A7: A2 2A    >169           LDX   #$2A       NODATA ERROR
95A9: 4C 12 D4 >170           JMP   $D412
95AC: C8       >171           INY              ;Pick up the line #
95AD: B1 B8    >172           LDA   (TXTPTR),Y
95AF: 85 7B    >173           STA   DATLIN
95B1: C8       >174           INY
95B2: B1 B8    >175           LDA   (TXTPTR),Y
95B4: C8       >176           INY
95B5: 85 7C    >177           STA   DATLIN+1
95B7: B1 B8    >178  :1       LDA   (TXTPTR),Y Get 1st token of statement
95B9: AA       >179           TAX              ;Save token in X reg.
95BA: 20 98 D9 >180           JSR   ADDON      Add Y to TXTPTR
95BD: E0 83    >181           CPX   #TOKDATA
95BF: D0 DA    >182           BNE   FINDATA
95C1: 4C 2A 95 >183           JMP   INSTART
               >184
95C4: A5 15    >185  INPFIN   LDA   INPUTFLG
95C6: D0 A3    >186           BNE   INPDATA
95C8: 4C 86 DB >187           JMP   $DB86
               >188
95CB: 4C C6 DC >189  INPDONE  JMP   $DCC6
                1104
                1105 FCODE    EQU   *
                1106
                1107          PUT   PEERGDATA
95CE: 00 00 00 >1    SVPTR    DS    18
95E0: 00       >2    SVP2     DFB   0
               >3
95E1: 00 00 00 >4    TABOFB   DFB   0,0,0,0,0,0,0,0
95E9: 00 00 00 >5    TABOFT   DFB   0,0,0,0,0,0,0,0
95F1: 00       >6    INDX     DFB   0
95F2: 00       >7    SPROOT   DFB   0
95F3: 00 00    >8    ITVADDR  DA    0          Adresse de la var. ITHREAD%
95F5: F8 75 76 >9    P0OFFSET DFB   REMSTK,CURLIN,CURLIN+1,TXTPTR,TXTPTR+1
95FA: 79 7A    >10            DFB   OLDTEXT,OLDTEXT+1
               >11   PIOFFSET EQU   *
95FC: F4 F5 F6 >12            DFB   TXTPSV,TXTPSV+1,CURLSV,CURLSV+1,ERRNUM
9601: DF DA DB >13            DFB   ERRSTK,ERRLIN,ERRLIN+1,ERRPOS,ERRPOS+1
9606: D8       >14            DFB   ERRFLG
               >15   PEOFFSET EQU   *
9607: C9 C6 C2 >16   TOKMOTIF DFB   TOKMINUS,TOKNOT,TOKFN,TOKSCRN
               >17   TOKMTIFE
960B: CE DE 90 >22   TOKMPF   DA    $DECE,$DE90,$E354,$DEF9
9613: C8 C9 CA >24   TOKENS   DFB   TOKADD,TOKMINUS,TOKMUL,TOKDIV
```

```
                    >25
9617: BE E7 A7      >27   FPROUTS  DA    FADD,FSUB,FMULT,FDIV
                    >32   * Motifs used inside FOR/NEXT loop handling
                    >33   * to restore full byte patterns from two bits
961F: 00 01         >34   MOTGF    DFB   0,1
9621: 00            >35           DS    1
9622: FF            >36           HEX   FF
                    >37   * Where is stored the elm. index in a FOREACH loop
9623: 00 00         >38   AEI      DA    0
                    >39
9625: 94 9F         >40   OFFSTB   DFB   HNDLSIAD-1,HNDLSIMI-1
9627: BE BD         >41           DFB   HNDLSIMU-1,HNDLSIDV-1
9629: E3 F0         >42           DFB   HNDLSBAD-1,HNDLSBMI-1
962B: F8 16         >43           DFB   HNDLSBMU-1,HNDLSBDV-1
962D: 7C 88         >44           DFB   HNDLUIAD-1,HNDLUIMI-1
962F: AC AB         >45           DFB   HNDLUIMU-1,HNDLUIDV-1
9631: DC E9         >46           DFB   HNDLUBAD-1,HNDLUBMI-1
9633: F7 15         >47           DFB   HNDLUBMU-1,HNDLUBDV-1
9635: 77 77         >48   OFFSTT   DFB   >HNDLSIAD-1,>HNDLSIMI-1
9637: 77 77         >49           DFB   >HNDLSIMU-1,>HNDLSIDV-1
9639: 77 77         >50           DFB   >HNDLSBAD-1,>HNDLSBMI-1
963B: 77 78         >51           DFB   >HNDLSBMU-1,>HNDLSBDV-1
963D: 77 77         >52           DFB   >HNDLUIAD-1,>HNDLUIMI-1
963F: 77 77         >53           DFB   >HNDLUIMU-1,>HNDLUIDV-1
9641: 77 77         >54           DFB   >HNDLUBAD-1,>HNDLUBMI-1
9643: 77 78         >55           DFB   >HNDLUBMU-1,>HNDLUBDV-1
                    >56
9645: 00 00 00      >57   ADRSTRUCT DS   11*LENREC
972C: F8            >58   SVOFST   DFB   REMSTK
972D: B8 B9         >59           DFB   TXTPTR,TXTPTR+1
972F: 75 76         >60           DFB   CURLIN,CURLIN+1
9731: 79 7A         >61           DFB   OLDTEXT,OLDTEXT+1
9733: F2            >62           DFB   TRCFLG
9734: A5 A6 A7      >63           DFB   ARG,ARG+1,ARG+2,ARG+3,ARG+4,$AA
                    >64   FINOF    EQU   *
973A: 00 00 00      >65   SVAREA   DS    FINOF-SVOFST
                    >66
9748: 00 00 00      >67   SVCURRM  DS    12
9754: 00 00 00      >68   SVALTNM  DS    12
                    >69
                    >70   * Structure juste pour la prise en compte lors du DEFUSR
9760: 00 00 00      >71   ]DEBUT   DS    8
                    >72   ]FIN
9768: 60 97         >73   SDEF1    DA    ]DEBUT     pour VARTAB
976A: 60 97         >74           DA    ]DEBUT     pour ARYTAB
976C: 60 97         >75           DA    ]DEBUT     pour STREND
976E: 68 97         >76           DA    ]FIN       pour FRETOP
9770: 68 97         >77           DA    ]FIN       pour FRESPC
9772: 68 97         >78           DA    ]FIN       pour MEMSIZ
                    >79
                    >80   * Structure de stockage privee pour la derniere PF
                    >81   * dynamique.
9774: 00 00 00      >82   ]DEBUT   DS    512
                    >83   ]FIN
9974: 74 97         >84   SINITX   DA    ]DEBUT     pour VARTAB
9976: 74 97         >85           DA    ]DEBUT     pour ARYTAB
9978: 74 97         >86           DA    ]DEBUT     pour STREND
```

```
997A: 74 99     >87             DA    ]FIN       pour FRETOP
997C: 74 99     >88             DA    ]FIN       pour FRESPC
997E: 74 99     >89             DA    ]FIN       pour MEMSIZ
                >90
9980: 00        >91   ISPFACT   DS    1          Dynamic PF active?
9981: 00        >92   PFINDIC   DS    1          Last dynamic PF used..
9982: 00        >93   PFINDX    DS    1          Current PF index..
                >94
                >95   * Cache structure for simple variables
9983: 00        >96   SNCCH     DFB   0
9984: 00 00 00 >102   SVN       DS    KSNCACH
9988: 00 00 00 >103   SVNP1     DS    KSNCACH
998C: 00 00 00 >104   SIT       DS    KSNCACH
9990: 00 00 00 >105   SLTR      DS    KSNCACH
9994: 00 00 00 >106   SLTRP1    DS    KSNCACH
                >108  * Cache structure for array variables
9998: 00        >109  ANCCH     DFB   0
9999: 00 00 00 >115   AVN       DS    KSNCACH
999D: 00 00 00 >116   AVNP1     DS    KSNCACH
99A1: 00 00 00 >117   AIT       DS    KSNCACH
99A5: 00 00 00 >118   ALTR      DS    KSNCACH
99A9: 00 00 00 >119   ALTRP1    DS    KSNCACH
                1108            PUT   PEERMOTIDATA
                >1    * Data segment for the mouse/timer/interrupt module
                >2    * Mouse data (detected upon init)
                >3    * Offset table
99AD: 12 13 14 >4     OM_DEB    HEX   12131415161718
99B4: 19        >5    OM_INI    HEX   19
                >6
99B5: 00        >7    MON0      DS    1
99B6: 00        >8    MVECTOR   DS    1
99B7: 00        >9    MOCN      DS    1
                >10
99B8: 01        >11   MOMODE    DFB   1
                >12
99B9: 00 00     >13   CLNLO     DS    2          Line # of inter. handler lo
99BB: 00 00     >14   CLNHI     DS    2          Line # of inter. handler hi
99BD: 00 00     >15   AHNDLO    DS    2          Address of Applesoft line lo
99BF: 00 00     >16   AHNDHI    DS    2          Address of Applesoft line hi
                >17
99C1: 00        >18   MONU      DS    1          0 till 1st MOUSE/TIMER instr
99C2: 00        >19   SVMTACTV  DS    1
                >20
99C3: 07 0F     >21   MOETMSK   HEX   070F
99C5: 01 00     >22   MOCMPVAL  HEX   0100
                >23
99C7: 00 40 40 >24    MSTATUS   HEX   0040404080C0C0C0
99CF: 00 00     >25   OLDVECT   DA    0
                >26
99D1: 00        >27   WORKPL1   DS    1
99D2: 00        >28   MIRQST    DS    1
                >29   * YICUR: indique quel est le dernier
                >30   * handler d´interruption retenu
99D3: FF        >31   YICUR     DFB   $FF
                >32
                >33   * Deux slots pour chaque entree
                >34   * Indices:
```

```
                >35   * 0: pour l´API MOUSE
                >36   * 1: pour l´API TIMER
                >37   * MODERUN: 1 iif routine en cours, 0 sinon
99D4: 00 00     >38   MODERUN  DS    2
                >39   * MODEPEC:
                >40   * 0: non prise en compte de l´interruption
                >41   * 1: prise en compte retardee
                >42   * 2: prise en compte immediate
99D6: 00 00     >43   MODEPEC  DS    2
99D8: 40 80     >44   MSKINT   HEX   4080
                >45   * Values of S to cmp upon return from Applesoft
                >46   * handling routine (usually RETURN)
99DA: 00 00     >47   INTSPTR  DS    2
                >48
99DC: 00 00     >49   CLN_B    DS    2          Interrupted line # lo byte
99DE: 00 00     >50   CLN_T    DS    2          Interrupted line # hi byte
99E0: 00 00     >51   TPT_B    DS    2          Interrupted text ptr lo byte
99E2: 00 00     >52   TPT_T    DS    2          Interrupted text ptr hi byte
99E4: 00 00     >53   OTPT_B   DS    2          Interrupted OLDTEXT lo byte
99E6: 00 00     >54   OTPT_T   DS    2          Interrupted OLDTEXT hi byte
                >55
                >56   * Offsets from page zero to save for WAIT
99E8: 50 51     >57   SVWOF    DFB   LINNUM,LINNUM+1
99EA: 85 86     >58            DFB   FORPNT,FORPNT+1
99EC: B8 B9     >59            DFB   TXTPTR,TXTPTR+1
                >60   * Save area for WAIT
99EE: 00 00 00 >61    SVA      DS    6
99F4: 00        >62   FRGNDCTX DFB   0          5 pour WAIT
                >63
                >64   * KTINC factor for timer interrupt (default 1)
99F5: 01 00     >65   KTINC    DA    1          config. value for timer factor
99F7: 00 00     >66   TIINC    DA    0          runtime value for timer factor
                >67
                >68   * Error messages
                >69   MESSERR
                >70   MESER1   EQU   *-MESSERR
99F9: 4D 4F 55 >71            DCI   ´MOUSE HARDWARE NOT DETECTED´
                >72   MESER2   EQU   *-MESSERR
9A14: 55 4E 53 >73            DCI   ´UNSUPPORTED HARDWARE CONFIGURATION´
                >74   MESER3   EQU   *-MESSERR
9A36: 55 4E 4B >75            DCI   ´UNKNOWN APPLESOFT MOUSE EVENT HANDLER´
                >76   MESER4   EQU   *-MESSERR
9A5B: 55 4E 4B >77            DCI   ´UNKNOWN APPLESOFT TIMER EVENT HANDLER´
                >78   MESER5   EQU   *-MESSERR
9A80: 49 4C 4C >79            DCI   ´ILLEGAL MOUSE MODE´
                >80   MESER6   EQU   *-MESSERR
9A92: 49 4C 4C >81            DCI   ´ILLEGAL MOUSE OPERATION´
                >82   MESER7   EQU   *-MESSERR
9AA9: 5A 45 52 >83            DCI   ´ZERO TARGET ADDRESS´
                >84   MESER8   EQU   *-MESSERR
9ABC: 45 4D 42 >85            DCI   ´EMBEDDED PF NOT SUPPORTED IN THIS RELEASE´
                >86   MESER9   EQU   *-MESSERR
9AE5: 49 4C 4C >87            DCI   ´ILLEGAL OP WHILE PF IS ACTIVE´
9B02: 00 1B 3D >88    CODR     DFB   MESER1,MESER2,MESER3,MESER4,MESER5,MESER6
9B08: B0 C3 EC >89            DFB   MESER7,MESER8,MESER9
                >90
9B0B: 91 80 00 >91    NEG65536 HEX   9180000000
```

```
9B10: 90 80 00 >92   NEG32768 HEX   9080000000
9B15: 90 00 00 >93   POS32768 HEX   9000000000
9B1A: 91 00 00 >94   POS65536 HEX   9100000000
                1109
                1110 * Table of new Peersoft commands
9B1F: C8        1111 TMOCL    DFB   TOKADD
9B20: D0        1112          DFB   TOKEQUAL
9B21: 00        1113          DFB   0
9B22: C9        1114          DFB   TOKMINUS
9B23: D0        1115          DFB   TOKEQUAL
9B24: 00        1116          DFB   0
9B25: CA        1117          DFB   TOKMUL
9B26: D0        1118          DFB   TOKEQUAL
9B27: 00        1119          DFB   0
9B28: CB        1120          DFB   TOKDIV
9B29: D0        1121          DFB   TOKEQUAL
9B2A: 00        1122          DFB   0
9B2B: 40        1123          ASC   ´@´
9B2C: 00        1124          DFB   0
9B2D: 23        1125          ASC   ´#´
9B2E: 00        1126          DFB   0
9B2F: 4F 46 46  1127          ASC   ´OFF´
9B32: 00        1128          DFB   0
9B33: 49        1129 IFIIF    ASC   ´I´
9B34: AD        1130          DFB   TOKIF
9B35: 00        1131          DFB   0
9B36: 4D 4F 55  1132          ASC   ´MOUSE´
9B3B: 00        1133          DFB   0
9B3C: 54 49 4D  1134          ASC   ´TIMER´
9B41: 00        1135          DFB   0
9B42: B8        1136 IFDEF    DFB   TOKDEF
9B43: D5        1137          DFB   TOKUSR
9B44: 00        1138          DFB   0
9B45: B8        1139          DFB   TOKDEF
9B46: 53 54 52  1140          ASC   ´STR´
9B49: 00        1141          DFB   0
9B4A: B8        1142          DFB   TOKDEF
9B4B: 53 4E 47  1143          ASC   ´SNG´
9B4E: 00        1144          DFB   0
9B4F: B8        1145          DFB   TOKDEF
9B50: D3        1146          DFB   TOKINT
9B51: 00        1147          DFB   0
9B52: B8        1149          DFB   TOKDEF
9B53: 42 59 54  1150          ASC   ´BYTE´
9B57: 00        1151          DFB   0
9B58: 81        1153          DFB   TOKFOR
9B59: 45 41 43  1154 IFEACH   ASC   ´EACH´
9B5D: 00        1155          DFB   0
9B5E: FF        1156          HEX   FF
                1157
9B5F: 00 03 06  1158 TOFFST   DFB   0,3,6,9    Pour les 4 syntax schemes
                1159          ERR   NOPER-4
9B63: 0C        1160          DFB   12         Pour le symbole @
9B64: 0E        1161          DFB   14         Pour le symbole #
9B65: 10        1162 OFFOFF   DFB   16         Pour le mot cle OFF
9B66: 14        1163 OFFIIF   DFB   20         Pour la fonction IIF()
9B67: 17        1164 OFFMOU   DFB   23         Pour le mot cle MOUSE
```

```
9B68: 1D        1165 OFFTIM   DFB   29         Pour le mot cle TIMER
9B69: 23        1166 OFFUSR   DFB   35         Pour le mot cle DEFUSR
9B6A: 26 2B 30  1167 OFFDEF   DFB   38,43,48   pour les intr. DEFSTR,SNG,INT...
9B6D: 33        1168          DFB   51         Pour le DEFBYTE
9B6E: 39        1169          DFB   57         Pour le FOREACH
9B6F: 3F        1170          DFB   63
                1171
                1172 * Ou commencer la recherche?
                1173 * au debut (LIST)
9B70: FF        1174 TIDMOCL  DFB   0-1
                1175 * instruction DEF<pattern>
9B71: 09        1176          DFB   OFFUSR-TOFFST-1
                1177 * sur la premiere fonction (IIF/MOUSE/TIMER)
9B72: 06        1178          DFB   OFFIIF-TOFFST-1
                1179 * fonction MOUSE ou TIMER
9B73: 07        1180          DFB   OFFMOU-TOFFST-1
9B74: 08        1181          DFB   OFFTIM-TOFFST-1
                1182 * Juste mot-cle OFF
9B75: 05        1183          DFB   OFFOFF-TOFFST-1
                1184 * Quoi mettre a l´offset OFFDEF
9B76: B8        1185 TOFFIN   DFB   TOKDEF     si LIST
9B77: B8        1186          DFB   TOKDEF     si DEF<pattern>
9B78: FF        1187          HEX   FF         si IIF/MOUSE/TIMER
9B79: FF        1188          HEX   FF         si MOUSE/TIMER
9B7A: FF        1189          HEX   FF         si TIMER
9B7B: FF        1190          HEX   FF         si OFF
                1191 * Quoi mettre a l´offset OFFIFF
9B7C: 49        1192 TOFFIN2  DFB   ´I´        ;si LIST
9B7D: 49        1193          DFB   ´I´        ;si DEF<pattern>
9B7E: 49        1194          DFB   ´I´        ;si IFF/MOUSE/TIMER
9B7F: 49        1195          DFB   ´I´        ;si MOUSE/TIMER
9B80: 49        1196          DFB   ´I´        ;si TIMER
9B81: FF        1197          HEX   FF         si OFF
9B82: 24 21 25  1198 MOTIF    ASC   ($!%(
9B85: 2E        1200          ASC   (.(
9B86: 00 00 82  1201 TITVAL   HEX   00008281   What to store into INTTYP
9B8A: FF 00 00  1202 TVTVAL   HEX   FF000000   What to store into VALTYP
9B8E: 00 00 80  1203 TVNORA   HEX   00008080   Value to ORA with VARNAM
9B92: 80 00 80  1204 TVN1ORA  HEX   80008080   Value to ORA with VARNAM+1
                1210
9B96: 21 21 21  1211 TYPLET   DS    26,´!´
                1212
                1213 * Applesoft standard instructions entry points
                1214 APRWAIT  EQU   $E784      WAIT instruction entry point
                1215 APRRUN   EQU   $D912      RUN instruction entry point
                1216 APRLIST  EQU   $D6A5      LIST instruction entry point
                1217 APRCLEAR EQU   $D66A      CLEAR instruction entry point
                1218 APRDEF   EQU   $E313      DEF instruction entry point
                1219 APRLET   EQU   $DA46      LET instruction entry point
                1220 APRFOR   EQU   $D766      FOR instruction entry point
                1221 APRNEXT  EQU   $DCF9      NEXT instruction entry point
                1222 APFRMELM EQU   $DE67      Return address from FRMELM
                1223 APRETURN EQU   $D96B      RETURN/POP instr. entry point
                1224 APRONERR EQU   $F2CB      ONERR instruction entry point
                1225 APRNEW   EQU   $D649      NEW instruction entry point
                1226 APRGOTO  EQU   $D93E      GOTO instruction entry point
                1227 APRGOSUB EQU   $D921      GOSUB instruction entry point
```

```
                1228 APRIF    EQU   $D9C9      IF instruction entry point
                1229 APRON    EQU   $D9EC      ON expr GOTO/GOSUB entry point
                1230 APRGET   EQU   $DBA0
                1231 APRINP   EQU   $DBB2
                1232 APRREAD  EQU   $DBE2
                1233
9BB0: 83        1234 ADAPFBET DFB   APRWAIT-1
9BB1: 11 48 A4  1235          DFB   APRRUN-1,APRNEW-1,APRLIST-1,APRCLEAR-1
9BB5: CA 12 45  1236          DFB   APRONERR-1,APRDEF-1,APRLET-1
9BB8: E1 B1 9F  1237          DFB   APRREAD-1,APRINP-1,APRGET-1
9BBB: 65 EB 3D  1238          DFB   APRFOR-1,APRON-1,APRGOTO-1,APRIF-1,APRETURN-
1,APRGOSUB-1
9BC1: F8 66     1239          DFB   APRNEXT-1,APFRMELM-1
9BC3: 1F        1240          DFB   $D820-1
9BC4: E7        1241 ADAPFTET DFB   >APRWAIT-1
9BC5: D9 D6 D6  1242          DFB   >APRRUN-1,>APRNEW-1,>APRLIST-1,>APRCLEAR-1
9BC9: F2 E3 DA  1243          DFB   >APRONERR-1,>APRDEF-1,>APRLET-1
9BCC: DB DB DB  1244          DFB   >APRREAD-1,>APRINP-1,>APRGET-1
9BCF: D7 D9 D9  1245          DFB   >APRFOR-1,>APRON-1,>APRGOTO-1,>APRIF-1,>APRE
TURN-1,>APRGOSUB-1
9BD5: DC DE     1246          DFB   >APRNEXT-1,>APFRMELM-1
9BD7: D8        1247          DFB   >$D820-1
9BD8: 3A        1248 ADPFB    DFB   RWAIT-1
9BD9: 5A 63 F6  1249          DFB   RRUN-1,RNEW-1,STDLIS-1,RCLEAR-1
9BDD: 75 77 7A  1250          DFB   RONERR-1,RDEF-1,RLET-1
9BE0: E0 B7 A9  1251          DFB   RREAD2-1,RINP-1,RGET-1
9BE3: 07 66     1252          DFB   RFOR-1,RON-1
9BE5: 13 EE 6B  1253 OFSTGTO  DFB   RGOTO-1,RIF-1,RRETURN-1,RGOSUB-1
9BE9: DA 53     1254          DFB   RNEXT-1,FRMELM-1
9BEB: 8F        1255          DFB   RNEWINST-1
9BEC: 8E        1256 ADPFT    DFB   >RWAIT-1
9BED: 7E 7E 82  1257          DFB   >RRUN-1,>RNEW-1,>STDLIS-1,>RCLEAR-1
9BF1: 84 81 76  1258          DFB   >RONERR-1,>RDEF-1,>RLET-1
9BF4: 94 94 94  1259          DFB   >RREAD2-1,>RINP-1,>RGET-1
9BF7: 8F 92 93  1260          DFB   >RFOR-1,>RON-1,>RGOTO-1,>RIF-1,>RRETURN-1,>R
GOSUB-1
9BFD: 8F 85     1261          DFB   >RNEXT-1,>FRMELM-1
9BFF: 8D        1262          DFB   >RNEWINST-1
                1263 FIN
                1264 LONGLANG EQU   *-CGARBAG
                1265          ERR   *-$9C00
                1266
                1267          PUT   PEERGLOBALPAGE
               >1             DUMMY $9CBD
9CBD: 00       >2    FLGFN    DS    1
9CBE: 00 00 00 >3    WRKFA    DS    5          FAC work area A
9CC3: 00 00 00 >4    WRKFB    DS    5          FAC work area B
9CC8: 00 00 00 >5    WRKFC    DS    5          FAC work area C
9CCD: 50       >6    SVNUM    HEX   50         Subversion number..
9CCE: 00       >7    MOSL     DS    1          Mouse slot (b7 set to 1 if none)
9CCF: 00       >8    NEEDDEC  DFB   0
               >9    * Computed GOTO behavior: 0 iif inactive
               >10   * 64: cannot happen
               >11   * 128 iif active and no safeguard
               >12   * 192 iif active and safeguard
9CD0: 80       >13   OPTCGOTO HEX   80
               >14   * Some vectors
```

```
9CD1: 7E 7B    >15   VNARRG91 DA    NARGL91   Look up array name in memory
9CD3: 72 7A    >16   VNPTRG90 DA    NPTRGL90  Look up variable name in memory
9CD5: 4C 84 E4 >17   VGARBAG  JMP   GARBAG
               >18   * MT parameters
9CD8: E1 95    >19   ADADR    DA    TABOFB
9CDA: 00       >20   INHACTV  DFB   0         b7 set if switching inhibited
9CDB: 00       >21   CTRACTV  DFB   0         Counter run value
9CDC: 00       >22   MTACTV   DFB   0         b7 set if MT active
9CDD: 00       >23   ICTRACTV DFB   0         Number of ticks between 2 CTS
               >24   * General purpose constants
9CDE: 15       >25   PVERSION DFB   VERSION   Peersoft version number
9CDF: 4C 00 87 >26   REVECTOR JMP   ROUTGEN   Vector to utility routine
               >27            ERR   *-$9CE2   Must coincide with Bananasoft
               >28            DEND
               >29   WMODE    EQU   $9CE7     Bit 7 set iif unsigned for Ints
               >30            DUMMY $9CED
9CED: 00       >31   MACHINE  DS    1
9CEE: 00       >32            DS    1         CPU
9CEF: 00       >33   MEMORY   DS    1
9CF0: 00       >34   VID80C   DS    1
               >35            DEND


--End assembly, 11670 bytes, Errors: 0


Symbol table - alphabetical order:

   A1L     =$3C        A2L     =$3E        A4L     =$42        ABSOL8  =$7916
   ABSOLUTE=$79D7   ?  ACTR    =$9B        ADADR   =$9CD8      ADAPFBET=$9BB0
   ADAPFTET=$9BC4      ADB1    =$4217      ADB2    =$4231      ADDON   =$D998
   ADPFB   =$9BD8      ADPFT   =$9BEC      ADRSTRUCT=$9645      ADRUSR  =$01
   ADT1    =$4224      ADT2    =$423E      AEI     =$9623      AHNDHI  =$99BF
   AHNDLO  =$99BD      AIT     =$99A1      ALKCACH =$7D9D      ALTR    =$99A5
   ALTRP1  =$99A9      ALTZP   =$C009      ANCCH   =$9998      APFRMELM=$DE67
   APRCLEAR=$D66A      APRDEF  =$E313      APRETURN=$D96B      APRFOR  =$D766
   APRGET  =$DBA0      APRGOSUB=$D921      APRGOTO =$D93E      APRIF   =$D9C9
   APRINP  =$DBB2      APRLET  =$DA46      APRLIST =$D6A5      APRNEW  =$D649
   APRNEXT =$DCF9      APRON   =$D9EC      APRONERR=$F2CB      APRREAD =$DBE2
   APRRUN  =$D912      APRWAIT =$E784      ARET    =$79C8      ARG     =$A5
   AROMBA  =$479F      ARYPNT  =$94        ARYTAB  =$6B        ARYVAR  =$D037
   AUXBANK =$BF        AUXPTR  =$06        AVN     =$9999      AVNP1   =$999D
   AXARTAB =$D097   ?  AXARYPNT=$D097   ?  AXARYPT2=$D09C      AXHIMEM =$BF00
?  AXOFFSET=$D099      AXVALUE =$D09C   ?  AYINT   =$E10C      BADNAM  =$7A31
   BAMBS   =$0778      BANCLD  =$8234      BAXHI   =$0578      BAXLO   =$0478
   BAYHI   =$05F8      BAYLO   =$04F8      BIGRECON=$424B      BISVTYP =$BE
   BTMEL   =$D199      CALLFUNC=$868F      CFA     =$42E5      CFM     =$42E1
   CGARBAG =$7609      CH      =$24        CHARAC  =$0D        CHKMEM  =$D3D6
   CHKNUM  =$DD6A      CHKSTR  =$DD6C      CLENGTH =$9457      CLNHI   =$99BB
   CLNLO   =$99B9      CLN_B   =$99DC      CLN_T   =$99DE      CMPCLAMP=$8A88
   CNVT1   =$7D95      CODE1BF =$448F      CODE1GC =$4606      CODE1GCF=$479F
   CODE1LC =$454A      CODE2BF =$454A      CODE2LC =$45EB      CODR    =$9B02
   COLLECTR=$8121      COMBYTE =$E74C      COMCLAMP=$8AAB      COMCLEAR=$8AE0
   COMCMPLX=$8683      COMCOPY =$9120      COMFOR  =$8F98      COMINT1 =$8CF4
   COMINT2 =$8D44      COMINT4 =$8C6B      COMLBS  =$8B5F      COMLET2 =$7F52
   COMLISO =$8434      COMMON  =$8C29      COMMON9 =$8C24      COMMONG =$8032
   COMNEXT =$908E      COMPOFST=$7EA9      COMPOS  =$8B13      COMREAD =$8AE5
```

```
   COMREST =$8113      COMRG   =$9460      COMRST  =$7632      COMRSTC =$763A
   COMWAIT =$8E4C      COMX1   =$7F1A      CONINT  =$E6FB      CONV1628=$79EA
   COPYROM =$4359      CRDO    =$DAFB      CTRACTV =$9CDB      CURLIN  =$75
   CURLSV  =$F6        DATA    =$D995      DATA1IDX=$42E9      DATA1VAL=$42EF
   DATAN   =$D9A3      DATLIN  =$7B        DATPTR  =$7D        DBUFP   =$9D00
   DEBUTGET=$7609      DEBUTGOT=$7658      DECOMPILE=$93C1      DECTPTR =$8209
   DEFFLG  =$C1        DEST    =$60        DIMFLG  =$10        DINSIRQV=$8A6E
   DIVEND  =$C2        DIVSOR  =$C0        DSCLEN  =$8F        DSCTMP  =$9D
   DVAR    =$D0FF      DVARS   =$D0F2      DVARTS  =$D189      DVZERROR=$7967
   E06     =$82AC      EK      =$41C1   ?  ELMSIZ  =$D09B   MD EMOV    =$8000
   ENDCHR  =$0E        ENDRNG  =$8319      ERRDIR  =$E306      ERRFLG  =$D8
   ERRLIN  =$DA        ERRNUM  =$DE        ERRPOS  =$DC        ERRSTK  =$DF
   ERR_BSCR=$6B        ERR_RDIM=$78        ERR_SYNT=$10        EXFLG   =$AAB3
   EXPLIC? =$7A3C      FAC     =$9D        FACLO   =$A1        FACMO   =$A0
   FACSIGN =$A2        FADD    =$E7BE      FAE2    =$7D40      FAE3    =$7D41
   FCODE   =$95CE      FCOMP   =$EBB2      FDIV    =$EA66      FEFOR   =$8EB1
   FENEXT  =$8FB2      FESTEP  =$8F00      FIN     =$9C00      FINDATA =$959B
   FINDEC  =$9407      FINLIGNE=$93FE      FINMOUSE=$8AE2      FINOF   =$973A
?  FLGFN   =$9CBD      FMULT   =$E97F      FNDLIN  =$D61A   ?  FNDVAR  =$D004
   FNDVAR2 =$7609      FNDVARX2=$D00B      FORPNT  =$85        FOUT    =$ED34
   FPROUTS =$9617   ?  FREESPC =$71        FREFAC  =$E600      FRETOP  =$6F
   FRGNDCTX=$99F4      FRMELM  =$8554      FRMELMLP=$8551      FRMEVL  =$DD7B
   FRMNUM  =$DD67      FRMSTCK3=$DE20      FRSTIM  =$7EC8      FSUB    =$E7A7
   G81     =$BFAA      G83     =$BFA3      GARBAG  =$E484      GDVARTS =$D0FB
   GETADR  =$E752      GETARY  =$E0ED      GETARY2 =$E0EF      GETBYT  =$E6F8
   GETSPA  =$E452      GFLAG   =$C0        GGO2TMER=$8762      GIQERR2 =$7DEB
   GIVAYF  =$E2F2      GME     =$7D10      GN32768 =$8E86      GN65536 =$8E90
   GNARRAY =$7BC2      GNPTRGET=$7655      GODVZERR=$EAE1      GOIQ    =$84DA
   GOIQERR =$E199      GOOVFERR=$E8D5      GOSTLERR=$E5B2   ?  GOSVCUR =$7EC4
   GOSYNERR=$7E7D      GOTMIERR=$DD76   MD GOTO    =$8000      GOTOTAIL=$D95E
   GOUNDEF =$93BE      GP32768 =$8E8B      GP65536 =$8E95      GRBPAS  =$D098
   GSE     =$7D3D      GSNERR2 =$7DE8      GSNERR3 =$81E5      GTFORPNT=$D365
   GTLT    =$7A22   ?  GTMERR2 =$7DEE      GZAUXRT =$BF00      H16B    =$8544
   HE2E8   =$86B6      HIMEM   =$73        HNDLEADR=$7F3F      HNDLEBC =$782D
   HNDLEIC =$77D3      HNDLEINT=$7740      HNDLEIX =$77CE      HNDLEIY =$7769
?  HNDLEREA=$76F5      HNDLESTR=$7715      HNDLSBAD=$77E4      HNDLSBDV=$7817
   HNDLSBMI=$77F1      HNDLSBMU=$77F9      HNDLSIAD=$7795      HNDLSIDV=$77BE
   HNDLSIMI=$77A0      HNDLSIMU=$77BF      HNDLUBAD=$77DD      HNDLUBDV=$7816
   HNDLUBMI=$77EA      HNDLUBMU=$77F8      HNDLUIAD=$777D      HNDLUIDV=$77AC
   HNDLUIMI=$7789      HNDLUIMU=$77AD      HNOK    =$8E15      IBUFFER =$0200
   ICTRACTV=$9CDD      IDMOCL  =$BD        IDX0    =$C0        IFDEF   =$9B42
   IFEACH  =$9B59      IFIIF   =$9B33      IFZ     =$94E8      INDEX   =$5E
   INDX    =$95F1      INHACTV =$9CDA      INITBF  =$4418   ?  INITLC  =$45EB
   INLIN   =$D52C      INPDATA =$956B      INPDONE =$95CB      INPFIN  =$95C4
   INPTR   =$7F        INPUTFLG=$15        INSDS2  =$F88C      INSIRQV =$8A46
   INSTART =$952A      INTSPTR =$99DA      INTTYP  =$12        INTTYPSV=$C7
   IRQHDLR =$89E7      IRQTBLE =$BFB1      IRQV    =$03FE      ISAUXMEM=$7DF1
   ISBASRUN=$A65E      ISCNTC  =$D858      ISHOSTOK=$8E05      ISLETC  =$E07D
   ISMOUSH =$8E0D      ISPFACT =$9980      ITEACH  =$91B5      ITVADDR =$95F3
   IVALARG =$8AA1      K6502   =$00        K65816  =$01        K65C02  =$01
?  KANCACH =$04        KILLEMAL=$88FD      KNEW    =$01     ?  KNEW2   =$01
   KOPT    =$01        KOPT16  =$00        KOPTLNG32=$01        KOPTLNG33=$00
   KSNCACH =$04        KTINC   =$99F5      KWELMSIZ=$7D66      KX3     =$88F8
   L08     =$835E      L088    =$835C      L3      =$85C4      LBS00   =$7E3C
   LBS03   =$7849      LBS033  =$91CF      LBS04   =$87D7      LBS041  =$8886
   LBS05   =$90A3      LBS051  =$90A7      LBS06   =$89CA   ?  LBS061  =$89CC
   LBS10   =$8BA9      LBS49   =$7830      LBS60   =$91DE      LBS61   =$90C9
```

```
   LBS62   =$90F1      LBS63   =$90F8      LBS64   =$915E      LBS641  =$9162
   LBS65   =$9129      LBS66   =$9153      LBS67   =$9171      LBS68   =$9187
   LBS69   =$919C      LBS80   =$86C8      LBS81   =$86C5      LENGTH  =$2F
   LENREC  =$15        LENTHS  =$D1A9      LET2    =$DA63      LETINF  =$C0
   LEVELPAR=$BD        LGSYNERR=$8529      LINGET  =$DA0C      LINNUM  =$50
   LISTED  =$839E      LLOOP   =$762C      LN      =$41F1      LONGLANG=$25F7
M? LOOP    =$4000      LOWTR   =$9B        LRST100 =$9499      LST1LIN =$834A
   LSTD?   =$8348      LTOKEN  =$8444      MACHINE =$9CED      MACMAT  =$42CB
   MAINLIST=$8321      MC      =$41E1      MCAND   =$C0        MCODE   =$42D3
   MEMERR  =$D410      MEMORY  =$9CEF      MESER1  =$00        MESER2  =$1B
   MESER3  =$3D        MESER4  =$62        MESER5  =$87        MESER6  =$99
   MESER7  =$B0        MESER8  =$C3        MESER9  =$EC        MESSERR =$99F9
   MFIN    =$86DA      MINSDS2 =$4201      MIRQST  =$99D2      MISLETC =$7E70
   MKNARRAY=$7C33      MKNV    =$E09C      MOCMPVAL=$99C5      MOCN    =$99B7
   MODDAT  =$BF        MODEPEC =$99D6      MODERUN =$99D4      MODREM  =$BE
   MOETMSK =$99C3      MOMODE  =$99B8      MON0    =$99B5      MONU    =$99C1
   MOSL    =$9CCE      MOTGF   =$961F      MOTIF   =$9B82      MOUSEDET=$42F5
   MOVE    =$FE2C      MOVFA   =$EB53      MOVFM   =$EAF9      MOVINS  =$E5D4
MD?MOVM    =$8000      MOVMF   =$EB2B   MD MPHX    =$8000   MD MPHY    =$8000
   MPLIER  =$C2     MD MPLX    =$8000   MD MPLY    =$8000      MSKINT  =$99D8
   MSTATUS =$99C7      MTACTV  =$9CDC      MTFUNC  =$8C9E   MD MTSB    =$8000
   MULTPLSS=$E2AD      MULTPLY1=$E2B6      MVECTOR =$99B6      MZRTAUX =$41D2
   NAMFOUND=$7ADF      NAMNTFND=$7AB0      NARRAY  =$7B2F      NARRGL91=$7B7E
   NCHKCLS =$86A3      NCHKCOM =$86A6      NCHKOPN =$86A9      NCR     =$838A
   NDATAN  =$9465      NDLVCMD =$920C      NDSVCMD =$9204      NEEDDEC =$9CCF
   NEG32768=$9B10      NEG65536=$9B0B      NEG8    =$791A      NEGATE  =$79DB
   NEGOP   =$EED0      NERRH   =$8E1C   ?  NERRHP  =$8E17      NEVAL   =$8B54
   NEVALC  =$8B4B   ?  NEWAYINT=$7853      NEWGARBG=$E484      NEWSTT  =$D7D2
   NEWY    =$47        NEXT1   =$8FE1      NEXTC2  =$8992      NEXTCTX =$8979
   NFAEP   =$7D13      NFRMEVL =$86AE      NFRMNUM =$853B      NFRMSTK2=$90BA
   NGARBAG =$7DD8      NGETARPT=$7A02      NGETBYT =$86D0      NGTA2   =$8E9A
   NILLM   =$8E1A      NKBDINT =$91EB      NLET2   =$7708      NMAKINT =$7D79
   NMOVINS =$7739      NOPER   =$04     ?  NOUVIN  =$8262      NPARCHK =$869D
   NPTRG   =$8B2F      NPTRGET =$7A08      NPTRGET1=$7A0E      NPTRGETX=$889D
   NPTRGL90=$7A72      NPTRGTX =$7A06      NREASON =$7B13      NREMN   =$9468
   NRET    =$79C6      NROUT   =$784E      NSYNCHR =$7E76      NSYNCHR2=$7E76
   NUMDIM  =$0F        NUMELS  =$08        NUMELS2 =$10        NWGVAYF =$86C0
   NXIN    =$DBDC      NXLST   =$832D      NZTAB   =$D0E2      OFFDEF  =$9B6A
   OFFIIF  =$9B66      OFFMOU  =$9B67      OFFOFF  =$9B65      OFFSET  =$C2
   OFFSTB  =$9625      OFFSTT  =$9635      OFFTIM  =$9B68      OFFUSR  =$9B69
   OFSTGTO =$9BE5      OKP1GET =$7644      OLDTEXT =$79        OLDTPTR =$79
   OLDVECT =$99CF      OM_DEB  =$99AD      OM_INI  =$99B4      OPRND   =$44
   OPTCGOTO=$9CD0      OTPT_B  =$99E4      OTPT_T  =$99E6      OUTDO   =$DB5C
   OUTQUES =$DB5A      OUTSPC  =$DB57      P0OFFSET=$95F5      PARTIAL =$BE
   PCADJ   =$F953      PCL     =$3A     ?  PEOFFSET=$9607      PFINDIC =$9981
   PFINDX  =$9982      PII     =$94F0      PIL     =$94EA      PIM     =$9574
   PIOFFSET=$95FC      POS32768=$9B15      POS65536=$9B1A      PTR2    =$1C
   PVERSION=$9CDE      QINT    =$EBF2      R       =$81E4      R0      =$890F
?  RAZPF   =$7E80      RCLEAR  =$7E6A      RCLM    =$05        RCLMAUX =$7E32
?  RCLR    =$03        RD2     =$A47A      RD80STOR=$C018      RDEF    =$8178
   RDEFSUB =$81DF      RDEFUSR =$805E      RDIM    =$8491      RDIMERR =$7C2E
   RDKEY   =$FD0C      RDLCBNK2=$C011      RDLCRAM =$C012      REASON  =$D3E3
   RECON   =$82C4      RECON1  =$82C0   ?  RECON2  =$82C8      REMSTK  =$F8
   RESTOR  =$893A      RESTOR1 =$8919      RESTOR2 =$8923      RESTORC =$895C
   RESTORD =$8913      RESTORF =$895B   ?  RESTORX =$894C      RESULT  =$62
   RET1    =$777C      RET3    =$85C1      RETOUR  =$80F3      RETOURM =$8D7C
   RETOURT =$8D7F      RETURN  =$82F6      REVECTOR=$9CDF      RFFVL   =$8599
```

```
   RFOR     =$8F08      RGET     =$94AA      RGOSUB   =$92CD      RGOTO    =$9314
   RGPART1  =$931B      RHOM     =$06        RIF      =$92EF      RIIF     =$852C
   RINI     =$07        RINP     =$94B8      RLET     =$767B      RLET1    =$76AA
?  RMTCTRL  =$88D3      RNEW     =$7E64      RNEWINST =$8D90      RNEWISUI =$88CE
   RNEXT    =$8FDB      RNI2     =$8DB1      RON      =$9267      RONERR   =$8476
   ROUT0    =$8728      ROUT10   =$8AB4      ROUT11   =$86DB   ?  ROUT1X   =$81EC
   ROUT1Y   =$81E8      ROUT4    =$8765      ROUT8    =$921B      ROUT8C   =$9212
   ROUTGEN  =$8700      RPOS     =$04        RREAD    =$02        RREAD2   =$94E1
   RRETURN  =$846C      RRUN     =$7E5B      RSETM    =$00        RSRVM    =$01
   RST100   =$762A      RST101   =$762C      RST102   =$7632      RST103   =$7632
   RSTALTM  =$8154      RSTCURRM =$8149      RUSR     =$7F66      RVRAI    =$84DD
   RW2      =$8E74      RWAIT    =$8E3B      SAVALTM  =$816A      SAVCURRM =$815F
   SAVER    =$89A0      SAVERC   =$89D7      SCDCH2   =$7A34      SCNDTIM  =$7F24
   SCTR     =$9B        SDEF1    =$9768      SDIV     =$796A      SDIV8    =$78C3
   SENDCHR  =$837A      SETINITX =$7E9B      SETITS   =$77D8      SETLTR   =$8970
   SETUPB   =$8212      SETUPD   =$8229      SETVYA   =$E0DE      SFE1     =$8EF9
   SINITX   =$9974      SIT      =$998C      SKIPC    =$84E5      SLKCACH  =$7AE2
   SLTR     =$9990      SLTRP1   =$9994   MD?SMOVE    =$8000      SMUL     =$7925
   SMUL8    =$788B      SNCCH    =$9983      SNERR    =$7C2B      SNGFLT   =$E301
   SPROOT   =$95F2      STACK    =$0100   MD?STD      =$8000      STDLIS   =$82F7
   STDZP    =$C008      STEP     =$8F62   MD STID     =$8000      STP1     =$8F5B
   STREND   =$6D        STRING1  =$AB        STRLT2   =$E3ED      STRNG    =$19
   STRNG1   =$AC        STRNG2   =$AD        STRPRT   =$DB3D      STRSPA   =$E3DD
   STRTRNG  =$8303      STRTXT   =$DE81      SUBERR   =$E196      SUBFLG   =$14
   SUBSERR  =$7C28   ?  SUITE    =$4000      SVA      =$99EE      SVALTNM  =$9754
   SVAREA   =$973A      SVARS    =$D01E      SVCURRM  =$9748      SVMTACTV =$99C2
   SVN      =$9984      SVNP1    =$9988   ?  SVNUM    =$9CCD      SVOFST   =$972C
?  SVP2     =$95E0      SVPTR    =$95CE      SVWOF    =$99E8      SWPIO    =$89BB
?  SWREINIT =$8C6E      SYNERR   =$DEC9      TABOFB   =$95E1      TABOFT   =$95E9
?  TELMS    =$D091      TEST2E   =$4395      TFUNC    =$8CDC      TIDMOCL  =$9B70
   TIINC    =$99F7      TIMEINST =$8C3E      TITVAL   =$9B86      TMERR    =$DD76
   TMOCL    =$9B1F      TOFFIN   =$9B76      TOFFIN2  =$9B7C      TOFFST   =$9B5F
   TOKADD   =$C8        TOKCHRD  =$E7        TOKDATA  =$83        TOKDEF   =$B8
   TOKDIM   =$86        TOKDIV   =$CB        TOKEN?   =$83A7      TOKENS   =$9613
   TOKEQUAL =$D0        TOKFN    =$C2        TOKFOR   =$81        TOKFRE   =$D6
   TOKGOSUB =$B0        TOKGOTO  =$AB        TOKIF    =$AD        TOKINT   =$D3
   TOKMINUS =$C9        TOKMOTIF =$9607      TOKMPF   =$960B      TOKMTIFE =$960B
   TOKMUL   =$CA        TOKNOT   =$C6        TOKREM   =$B2        TOKSCRN  =$D7
   TOKSGN   =$D2        TOKSTEP  =$C7        TOKSTRD  =$E4        TOKTABL  =$D0D0
   TOKTHEN  =$C4        TOKTO    =$C1        TOKUSR   =$D5        TOMOUSE  =$8C85
   TPT_B    =$99E0      TPT_T    =$99E2      TRACE    =$D805      TRAITEOK =$940A
   TRCFLG   =$F2        TVN1ORA  =$9B92      TVNORA   =$9B8E      TVTVAL   =$9B8A
   TXPSV    =$87        TXTPSV   =$F4        TXTPTR   =$B8        TXTTAB   =$67
   TYPLET   =$9B96      TYPMOD   =$C1        ULERR    =$D97C      USDIV    =$7995
   USDIV8   =$78F0      USEOLDAR =$7BC5      USMUL    =$7943      USMUL8   =$78A8
?  USRMOD   =$00        V3       =$801C      V3B      =$8017      V3T      =$8014
   VALTYP   =$11        VALTYPSV =$C8        VARNAM   =$81        VARPNT   =$83
   VARPT    =$D1B9      VARTAB   =$69        VECTUSR  =$0A     ?  VECZAUX  =$03ED
   VENT1IT  =$0C        VENT1NAM =$09     ?  VENT1PTR =$0D     ?  VENT1VT  =$0B
   VENT2IT  =$12        VENT2NAM =$0F     ?  VENT2PTR =$13     ?  VENT2VT  =$11
   VERSION  =$15        VGARBAG  =$9CD5      VID80C   =$9CF0      VLET     =$DA46
   VLINPRT  =$83A4      VNARRG91 =$9CD1      VNPTRG90 =$9CD3      VPNT     =$A0
   VPTRGET  =$DFEF      VSRTIT   =$06        VSRTNAM  =$03        VSRTPTR  =$07
?  VSRTVT   =$05        WHCBASIC =$AAB6      WMODE    =$9CE7      WORKPL1  =$99D1
?  WRKFA    =$9CBE   ?  WRKFB    =$9CC3   ?  WRKFC    =$9CC8      XFER     =$C314
?  XFRMMOT1 =$81FB      XFROMMOT =$81FE      XMFIN    =$87AB      XMFIN1   =$87D4
   XMFIN2   =$87D1      XSAV     =$B4        XSUITE   =$85BF      XXSAV    =$1B
```

```
   YICUR   =$99D3      YSAV    =$B5        ZAUXB   =$D019      ZAUXOFFT=$BFB7
   ZAUXRET =$BF3D      ZAUXRT  =$D014      ZAUXRT0 =$D019      ZAUXRT1 =$D03A
   ZAUXRT2 =$D045      ZAUXRT3 =$D000      ZCOMRT12=$D083      ZEROPRT =$79CA
   ZGCP2   =$BF92      ZGCPARMS=$BF79      ZNG     =$BF9D      ZPRT8   =$790B
   ZRTAUX  =$7DCB   V  ]DEBUT  =$9774   V  ]ERR    =$9264   V  ]ERR1   =$8B84
V  ]ERRS   =$77E6   V  ]FIN    =$9974   V  ]JLOOP  =$8B8A   V  ]LOOP   =$9494
V  ]LOOP1  =$948E   V  ]LOOP2  =$93DD   V  ]RET    =$9464
```

Symbol table - numerical order:

```
   K6502   =$00      KOPTLNG33=$00       KOPT16  =$00     ?  USRMOD  =$00
   RSETM   =$00      MESER1  =$00        K65C02  =$01        K65816  =$01
   KOPT    =$01      KNEW    =$01     ?  KNEW2   =$01        KOPTLNG32=$01
   ADRUSR  =$01      RSRVM   =$01        RREAD   =$02        VSRTNAM =$03
?  RCLR    =$03      KSNCACH =$04     ?  KANCACH =$04        NOPER   =$04
   RPOS    =$04   ?  VSRTVT  =$05        RCLM    =$05        AUXPTR  =$06
   VSRTIT  =$06      RHOM    =$06        VSRTPTR =$07        RINI    =$07
   NUMELS  =$08      VENT1NAM=$09        VECTUSR =$0A     ?  VENT1VT =$0B
   VENT1IT =$0C   ?  VENT1PTR=$0D        CHARAC  =$0D        ENDCHR  =$0E
   NUMDIM  =$0F      VENT2NAM=$0F        DIMFLG  =$10        NUMELS2 =$10
   ERR_SYNT=$10      VALTYP  =$11     ?  VENT2VT =$11        INTTYP  =$12
   VENT2IT =$12   ?  VENT2PTR=$13        SUBFLG  =$14        VERSION =$15
   LENREC  =$15      INPUTFLG=$15        STRNG   =$19        XXSAV   =$1B
   MESER2  =$1B      PTR2    =$1C        CH      =$24        LENGTH  =$2F
   PCL     =$3A      A1L     =$3C        MESER3  =$3D        A2L     =$3E
   A4L     =$42      OPRND   =$44        NEWY    =$47        LINNUM  =$50
   INDEX   =$5E      DEST    =$60        RESULT  =$62        MESER4  =$62
   TXTTAB  =$67      VARTAB  =$69        ARYTAB  =$6B        ERR_BSCR=$6B
   STREND  =$6D      FRETOP  =$6F     ?  FREESPC =$71        HIMEM   =$73
   CURLIN  =$75      ERR_RDIM=$78        OLDTPTR =$79        OLDTEXT =$79
   DATLIN  =$7B      DATPTR  =$7D        INPTR   =$7F        TOKFOR  =$81
   VARNAM  =$81      TOKDATA =$83        VARPNT  =$83        FORPNT  =$85
   TOKDIM  =$86      TXPSV   =$87        MESER5  =$87        DSCLEN  =$8F
   ARYPNT  =$94      MESER6  =$99        LOWTR   =$9B        SCTR    =$9B
?  ACTR    =$9B      FAC     =$9D        DSCTMP  =$9D        FACMO   =$A0
   VPNT    =$A0      FACLO   =$A1        FACSIGN =$A2        ARG     =$A5
   TOKGOTO =$AB      STRING1 =$AB        STRNG1  =$AC        TOKIF   =$AD
   STRNG2  =$AD      TOKGOSUB=$B0        MESER7  =$B0        TOKREM  =$B2
   XSAV    =$B4      YSAV    =$B5        TOKDEF  =$B8        TXTPTR  =$B8
   IDMOCL  =$BD      LEVELPAR=$BD        PARTIAL =$BE        MODREM  =$BE
   BISVTYP =$BE      AUXBANK =$BF        MODDAT  =$BF        MCAND   =$C0
   DIVSOR  =$C0      LETINF  =$C0        GFLAG   =$C0        IDX0    =$C0
   TOKTO   =$C1      TYPMOD  =$C1        DEFFLG  =$C1        TOKFN   =$C2
   MPLIER  =$C2      DIVEND  =$C2        OFFSET  =$C2        MESER8  =$C3
   TOKTHEN =$C4      TOKNOT  =$C6        TOKSTEP =$C7        INTTYPSV=$C7
   TOKADD  =$C8      VALTYPSV=$C8        TOKMINUS=$C9        TOKMUL  =$CA
   TOKDIV  =$CB      TOKEQUAL=$D0        TOKSGN  =$D2        TOKINT  =$D3
   TOKUSR  =$D5      TOKFRE  =$D6        TOKSCRN =$D7        ERRFLG  =$D8
   ERRLIN  =$DA      ERRPOS  =$DC        ERRNUM  =$DE        ERRSTK  =$DF
   TOKSTRD =$E4      TOKCHRD =$E7        MESER9  =$EC        TRCFLG  =$F2
   TXTPSV  =$F4      CURLSV  =$F6        REMSTK  =$F8        STACK   =$0100
   IBUFFER =$0200 ?  VECZAUX =$03ED      IRQV    =$03FE      BAXLO   =$0478
   BAYLO   =$04F8    BAXHI   =$0578      BAYHI   =$05F8      BAMBS   =$0778
MD EMOV    =$8000  MD?STD    =$8000   MD STID    =$8000   MD?MOVM    =$8000
MD?SMOVE   =$8000    LONGLANG=$25F7   M? LOOP    =$4000   MD MPHX    =$8000
MD MPHY    =$8000  MD MPLX    =$8000   MD MPLY    =$8000   MD MTSB    =$8000
```

```
MD GOTO    =$8000   ?  SUITE   =$4000      EK      =$41C1      MZRTAUX =$41D2
   MC      =$41E1      LN      =$41F1      MINSDS2 =$4201      ADB1    =$4217
   ADT1    =$4224      ADB2    =$4231      ADT2    =$423E      BIGRECON=$424B
   MACMAT  =$42CB      MCODE   =$42D3      CFM     =$42E1      CFA     =$42E5
   DATA1IDX=$42E9      DATA1VAL=$42EF      MOUSEDET=$42F5      COPYROM =$4359
   TEST2E  =$4395      INITBF  =$4418      CODE1BF =$448F      CODE2BF =$454A
   CODE1LC =$454A      CODE2LC =$45EB   ?  INITLC  =$45EB      CODE1GC =$4606
   CODE1GCF=$479F      AROMBA  =$479F      FNDVAR2 =$7609      CGARBAG =$7609
   DEBUTGET=$7609      RST100  =$762A      RST101  =$762C      LLOOP   =$762C
   RST102  =$7632      RST103  =$7632      COMRST  =$7632      COMRSTC =$763A
   OKP1GET =$7644      GNPTRGET=$7655      DEBUTGOT=$7658      RLET    =$767B
   RLET1   =$76AA   ?  HNDLEREA=$76F5      NLET2   =$7708      HNDLESTR=$7715
   NMOVINS =$7739      HNDLEINT=$7740      HNDLEIY =$7769      RET1    =$777C
   HNDLUIAD=$777D      HNDLUIMI=$7789      HNDLSIAD=$7795      HNDLSIMI=$77A0
   HNDLUIDV=$77AC      HNDLUIMU=$77AD      HNDLSIDV=$77BE      HNDLSIMU=$77BF
   HNDLEIX =$77CE      HNDLEIC =$77D3      SETITS  =$77D8      HNDLUBAD=$77DD
   HNDLSBAD=$77E4   V  ]ERRS   =$77E6      HNDLUBMI=$77EA      HNDLSBMI=$77F1
   HNDLUBMU=$77F8      HNDLSBMU=$77F9      HNDLUBDV=$7816      HNDLSBDV=$7817
   HNDLEBC =$782D      LBS49   =$7830      LBS03   =$7849      NROUT   =$784E
?  NEWAYINT=$7853      SMUL8   =$788B      USMUL8  =$78A8      SDIV8   =$78C3
   USDIV8  =$78F0      ZPRT8   =$790B      ABSOL8  =$7916      NEG8    =$791A
   SMUL    =$7925      USMUL   =$7943      DVZERROR=$7967      SDIV    =$796A
   USDIV   =$7995      NRET    =$79C6      ARET    =$79C8      ZEROPRT =$79CA
   ABSOLUTE=$79D7      NEGATE  =$79DB      CONV1628=$79EA      NGETARPT=$7A02
   NPTRGTX =$7A06      NPTRGET =$7A08      NPTRGET1=$7A0E      GTLT    =$7A22
   BADNAM  =$7A31      SCDCH2  =$7A34      EXPLIC? =$7A3C      NPTRGL90=$7A72
   NAMNTFND=$7AB0      NAMFOUND=$7ADF      SLKCACH =$7AE2      NREASON =$7B13
   NARRAY  =$7B2F      NARRGL91=$7B7E      GNARRAY =$7BC2      USEOLDAR=$7BC5
   SUBSERR =$7C28      SNERR   =$7C2B      RDIMERR =$7C2E      MKNARRAY=$7C33
   GME     =$7D10      NFAEP   =$7D13      GSE     =$7D3D      FAE2    =$7D40
   FAE3    =$7D41      KWELMSIZ=$7D66      NMAKINT =$7D79      CNVT1   =$7D95
   ALKCACH =$7D9D      ZRTAUX  =$7DCB      NGARBAG =$7DD8      GSNERR2 =$7DE8
   GIQERR2 =$7DEB   ?  GTMERR2 =$7DEE      ISAUXMEM=$7DF1      RCLMAUX =$7E32
   LBS00   =$7E3C      RRUN    =$7E5B      RNEW    =$7E64      RCLEAR  =$7E6A
   MISLETC =$7E70      NSYNCHR =$7E76      NSYNCHR2=$7E76      GOSYNERR=$7E7D
?  RAZPF   =$7E80      SETINITX=$7E9B      COMPOFST=$7EA9   ?  GOSVCUR =$7EC4
   FRSTIM  =$7EC8      COMX1   =$7F1A      SCNDTIM =$7F24      HNDLEADR=$7F3F
   COMLET2 =$7F52      RUSR    =$7F66      V3T     =$8014      V3B     =$8017
   V3      =$801C      COMMONG =$8032      RDEFUSR =$805E      RETOUR  =$80F3
   COMREST =$8113      COLLECTR=$8121      RSTCURRM=$8149      RSTALTM =$8154
   SAVCURRM=$815F      SAVALTM =$816A      RDEF    =$8178      RDEFSUB =$81DF
   R       =$81E4      GSNERR3 =$81E5      ROUT1Y  =$81E8   ?  ROUT1X  =$81EC
?  XFRMMOT1=$81FB      XFROMMOT=$81FE      DECTPTR =$8209      SETUPB  =$8212
   SETUPD  =$8229      BANCLD  =$8234   ?  NOUVIN  =$8262      E06     =$82AC
   RECON1  =$82C0      RECON   =$82C4   ?  RECON2  =$82C8      RETURN  =$82F6
   STDLIS  =$82F7      STRTRNG =$8303      ENDRNG  =$8319      MAINLIST=$8321
   NXLST   =$832D      LSTD?   =$8348      LST1LIN =$834A      L088    =$835C
   L08     =$835E      SENDCHR =$837A      NCR     =$838A      LISTED  =$839E
   VLINPRT =$83A4      TOKEN?  =$83A7      COMLISO =$8434      LTOKEN  =$8444
   RRETURN =$846C      RONERR  =$8476      RDIM    =$8491      GOIQ    =$84DA
   RVRAI   =$84DD      SKIPC   =$84E5      LGSYNERR=$8529      RIIF    =$852C
   NFRMNUM =$853B      H16B    =$8544      FRMELMLP=$8551      FRMELM  =$8554
   RFFVL   =$8599      XSUITE  =$85BF      RET3    =$85C1      L3      =$85C4
   COMCMPLX=$8683      CALLFUNC=$868F      NPARCHK =$869D      NCHKCLS =$86A3
   NCHKCOM =$86A6      NCHKOPN =$86A9      NFRMEVL =$86AE      HE2E8   =$86B6
   NWGVAYF =$86C0      LBS81   =$86C5      LBS80   =$86C8      NGETBYT =$86D0
   MFIN    =$86DA      ROUT11  =$86DB      ROUTGEN =$8700      ROUT0   =$8728
```

```
   GGO2TMER=$8762      ROUT4   =$8765      XMFIN   =$87AB      XMFIN2  =$87D1
   XMFIN1  =$87D4      LBS04   =$87D7      LBS041  =$8886      NPTRGETX=$889D
   RNEWISUI=$88CE   ?  RMTCTRL =$88D3      KX3     =$88F8      KILLEMAL=$88FD
   R0      =$890F      RESTORD =$8913      RESTOR1 =$8919      RESTOR2 =$8923
   RESTOR  =$893A   ?  RESTORX =$894C      RESTORF =$895B      RESTORC =$895C
   SETLTR  =$8970      NEXTCTX =$8979      NEXTC2  =$8992      SAVER   =$89A0
   SWPIO   =$89BB      LBS06   =$89CA   ?  LBS061  =$89CC      SAVERC  =$89D7
   IRQHDLR =$89E7      INSIRQV =$8A46      DINSIRQV=$8A6E      CMPCLAMP=$8A88
   IVALARG =$8AA1      COMCLAMP=$8AAB      ROUT10  =$8AB4      COMCLEAR=$8AE0
   FINMOUSE=$8AE2      COMREAD =$8AE5      COMPOS  =$8B13      NPTRG   =$8B2F
   NEVALC  =$8B4B      NEVAL   =$8B54      COMLBS  =$8B5F   V  ]ERR1   =$8B84
V  ]JLOOP  =$8B8A      LBS10   =$8BA9      COMMON9 =$8C24      COMMON  =$8C29
   TIMEINST=$8C3E      COMINT4 =$8C6B   ?  SWREINIT=$8C6E      TOMOUSE =$8C85
   MTFUNC  =$8C9E      TFUNC   =$8CDC      COMINT1 =$8CF4      COMINT2 =$8D44
   RETOURM =$8D7C      RETOURT =$8D7F      RNEWINST=$8D90      RNI2    =$8DB1
   ISHOSTOK=$8E05      ISMOUSH =$8E0D      HNOK    =$8E15   ?  NERRHP  =$8E17
   NILLM   =$8E1A      NERRH   =$8E1C      RWAIT   =$8E3B      COMWAIT =$8E4C
   RW2     =$8E74      GN32768 =$8E86      GP32768 =$8E8B      GN65536 =$8E90
   GP65536 =$8E95      NGTA2   =$8E9A      FEFOR   =$8EB1      SFE1    =$8EF9
   FESTEP  =$8F00      RFOR    =$8F08      STP1    =$8F5B      STEP    =$8F62
   COMFOR  =$8F98      FENEXT  =$8FB2      RNEXT   =$8FDB      NEXT1   =$8FE1
   COMNEXT =$908E      LBS05   =$90A3      LBS051  =$90A7      NFRMSTK2=$90BA
   LBS61   =$90C9      LBS62   =$90F1      LBS63   =$90F8      COMCOPY =$9120
   LBS65   =$9129      LBS66   =$9153      LBS64   =$915E      LBS641  =$9162
   LBS67   =$9171      LBS68   =$9187      LBS69   =$919C      ITEACH  =$91B5
   LBS033  =$91CF      LBS60   =$91DE      NKBDINT =$91EB      NDSVCMD =$9204
   NDLVCMD =$920C      ROUT8C  =$9212      ROUT8   =$921B   V  ]ERR    =$9264
   RON     =$9267      RGOSUB  =$92CD      RIF     =$92EF      RGOTO   =$9314
   RGPART1 =$931B      GOUNDEF =$93BE      DECOMPILE=$93C1   V  ]LOOP2  =$93DD
   FINLIGNE=$93FE      FINDEC  =$9407      TRAITEOK=$940A      CLENGTH =$9457
   COMRG   =$9460   V  ]RET    =$9464      NDATAN  =$9465      NREMN   =$9468
V  ]LOOP1  =$948E   V  ]LOOP   =$9494      LRST100 =$9499      RGET    =$94AA
   RINP    =$94B8      RREAD2  =$94E1      IFZ     =$94E8      PIL     =$94EA
   PII     =$94F0      INSTART =$952A      INPDATA =$956B      PIM     =$9574
   FINDATA =$959B      INPFIN  =$95C4      INPDONE =$95CB      FCODE   =$95CE
   SVPTR   =$95CE   ?  SVP2    =$95E0      TABOFB  =$95E1      TABOFT  =$95E9
   INDX    =$95F1      SPROOT  =$95F2      ITVADDR =$95F3      P0OFFSET=$95F5
   PIOFFSET=$95FC   ?  PEOFFSET=$9607      TOKMOTIF=$9607      TOKMTIFE=$960B
   TOKMPF  =$960B      TOKENS  =$9613      FPROUTS =$9617      MOTGF   =$961F
   AEI     =$9623      OFFSTB  =$9625      OFFSTT  =$9635      ADRSTRUCT=$9645
   SVOFST  =$972C      FINOF   =$973A      SVAREA  =$973A      SVCURRM =$9748
   SVALTNM =$9754      SDEF1   =$9768   V  ]DEBUT  =$9774   V  ]FIN    =$9974
   SINITX  =$9974      ISPFACT =$9980      PFINDIC =$9981      PFINDX  =$9982
   SNCCH   =$9983      SVN     =$9984      SVNP1   =$9988      SIT     =$998C
   SLTR    =$9990      SLTRP1  =$9994      ANCCH   =$9998      AVN     =$9999
   AVNP1   =$999D      AIT     =$99A1      ALTR    =$99A5      ALTRP1  =$99A9
   OM_DEB  =$99AD      OM_INI  =$99B4      MON0    =$99B5      MVECTOR =$99B6
   MOCN    =$99B7      MOMODE  =$99B8      CLNLO   =$99B9      CLNHI   =$99BB
   AHNDLO  =$99BD      AHNDHI  =$99BF      MONU    =$99C1      SVMTACTV=$99C2
   MOETMSK =$99C3      MOCMPVAL=$99C5      MSTATUS =$99C7      OLDVECT =$99CF
   WORKPL1 =$99D1      MIRQST  =$99D2      YICUR   =$99D3      MODERUN =$99D4
   MODEPEC =$99D6      MSKINT  =$99D8      INTSPTR =$99DA      CLN_B   =$99DC
   CLN_T   =$99DE      TPT_B   =$99E0      TPT_T   =$99E2      OTPT_B  =$99E4
   OTPT_T  =$99E6      SVWOF   =$99E8      SVA     =$99EE      FRGNDCTX=$99F4
   KTINC   =$99F5      TIINC   =$99F7      MESSERR =$99F9      CODR    =$9B02
   NEG65536=$9B0B      NEG32768=$9B10      POS32768=$9B15      POS65536=$9B1A
   TMOCL   =$9B1F      IFIIF   =$9B33      IFDEF   =$9B42      IFEACH  =$9B59
```

```
  TOFFST  =$9B5F     OFFOFF  =$9B65     OFFIIF  =$9B66     OFFMOU  =$9B67
  OFFTIM  =$9B68     OFFUSR  =$9B69     OFFDEF  =$9B6A     TIDMOCL =$9B70
  TOFFIN  =$9B76     TOFFIN2 =$9B7C     MOTIF   =$9B82     TITVAL  =$9B86
  TVTVAL  =$9B8A     TVNORA  =$9B8E     TVN1ORA =$9B92     TYPLET  =$9B96
  ADAPFBET=$9BB0     ADAPFTET=$9BC4     ADPFB   =$9BD8     OFSTGTO =$9BE5
  ADPFT   =$9BEC     FIN     =$9C00  ?  FLGFN   =$9CBD  ?  WRKFA   =$9CBE
?  WRKFB   =$9CC3  ?  WRKFC   =$9CC8  ?  SVNUM   =$9CCD     MOSL    =$9CCE
  NEEDDEC =$9CCF     OPTCGOTO=$9CD0     VNARRG91=$9CD1     VNPTRG90=$9CD3
  VGARBAG =$9CD5     ADADR   =$9CD8     INHACTV =$9CDA     CTRACTV =$9CDB
  MTACTV  =$9CDC     ICTRACTV=$9CDD     PVERSION=$9CDE     REVECTOR=$9CDF
  WMODE   =$9CE7     MACHINE =$9CED     MEMORY  =$9CEF     VID80C  =$9CF0
  DBUFP   =$9D00     RD2     =$A47A     ISBASRUN=$A65E     EXFLG   =$AAB3
  WHCBASIC=$AAB6     AXHIMEM =$BF00     GZAUXRT =$BF00     ZAUXRET =$BF3D
  ZGCPARMS=$BF79     ZGCP2   =$BF92     ZNG     =$BF9D     G83     =$BFA3
  G81     =$BFAA     IRQTBLE =$BFB1     ZAUXOFFT=$BFB7     STDZP   =$C008
  ALTZP   =$C009     RDLCBNK2=$C011     RDLCRAM =$C012     RD80STOR=$C018
  XFER    =$C314     ZAUXRT3 =$D000  ?  FNDVAR  =$D004     FNDVARX2=$D00B
  ZAUXRT  =$D014     ZAUXB   =$D019     ZAUXRT0 =$D019     SVARS   =$D01E
  ARYVAR  =$D037     ZAUXRT1 =$D03A     ZAUXRT2 =$D045     ZCOMRT12=$D083
?  TELMS   =$D091     AXARTAB =$D097  ?  AXARYPNT=$D097     GRBPAS  =$D098
?  AXOFFSET=$D099  ?  ELMSIZ  =$D09B     AXVALUE =$D09C  ?  AXARYPT2=$D09C
  TOKTABL =$D0D0     NZTAB   =$D0E2     DVARS   =$D0F2     GDVARTS =$D0FB
  DVAR    =$D0FF     DVARTS  =$D189     BTMEL   =$D199     LENTHS  =$D1A9
  VARPT   =$D1B9     GTFORPNT=$D365     CHKMEM  =$D3D6     REASON  =$D3E3
  MEMERR  =$D410     INLIN   =$D52C     FNDLIN  =$D61A     APRNEW  =$D649
  APRCLEAR=$D66A     APRLIST =$D6A5     APRFOR  =$D766     NEWSTT  =$D7D2
  TRACE   =$D805     ISCNTC  =$D858     APRRUN  =$D912     APRGOSUB=$D921
  APRGOTO =$D93E     GOTOTAIL=$D95E     APRETURN=$D96B     ULERR   =$D97C
  DATA    =$D995     ADDON   =$D998     DATAN   =$D9A3     APRIF   =$D9C9
  APRON   =$D9EC     LINGET  =$DA0C     VLET    =$DA46     APRLET  =$DA46
  LET2    =$DA63     CRDO    =$DAFB     STRPRT  =$DB3D     OUTSPC  =$DB57
  OUTQUES =$DB5A     OUTDO   =$DB5C     APRGET  =$DBA0     APRINP  =$DBB2
  NXIN    =$DBDC     APRREAD =$DBE2     APRNEXT =$DCF9     FRMNUM  =$DD67
  CHKNUM  =$DD6A     CHKSTR  =$DD6C     GOTMIERR=$DD76     TMERR   =$DD76
  FRMEVL  =$DD7B     FRMSTCK3=$DE20     APFRMELM=$DE67     STRTXT  =$DE81
  SYNERR  =$DEC9     VPTRGET =$DFEF     ISLETC  =$E07D     MKNV    =$E09C
  SETVYA  =$E0DE     GETARY  =$E0ED     GETARY2 =$E0EF  ?  AYINT   =$E10C
  SUBERR  =$E196     GOIQERR =$E199     MULTPLSS=$E2AD     MULTPLY1=$E2B6
  GIVAYF  =$E2F2     SNGFLT  =$E301     ERRDIR  =$E306     APRDEF  =$E313
  STRSPA  =$E3DD     STRLT2  =$E3ED     GETSPA  =$E452     GARBAG  =$E484
  NEWGARBG=$E484     GOSTLERR=$E5B2     MOVINS  =$E5D4     FREFAC  =$E600
  GETBYT  =$E6F8     CONINT  =$E6FB     COMBYTE =$E74C     GETADR  =$E752
  APRWAIT =$E784     FSUB    =$E7A7     FADD    =$E7BE     GOOVFERR=$E8D5
  FMULT   =$E97F     FDIV    =$EA66     GODVZERR=$EAE1     MOVFM   =$EAF9
  MOVMF   =$EB2B     MOVFA   =$EB53     FCOMP   =$EBB2     QINT    =$EBF2
  FOUT    =$ED34     NEGOP   =$EED0     APRONERR=$F2CB     INSDS2  =$F88C
  PCADJ   =$F953     RDKEY   =$FD0C     MOVE    =$FE2C
```