```
* PeerSoft v1.5.6 by Benoit Gilon - (c) 2006-2015 L.P.C.B.
* 30 Sep 2012: initial release
* 16 Oct 2012: 1.1, integ. divide support
* 30 Dec 2012: 1.2, integer arithmetic in FOR/NEXT loops
* & @ pseudo var)
* 3nd Jan 2013: 1.3 reorg subroutine #0
* 27 Jan 2013: 1.4 reorg subroutine #4 and MT kernel
* 6 Apr 2013: local error handling within MT kernel
* 1.5.5 addons:
* 31st July 2015: can concurrenlty define up to 11
* assembly language functions.. support for up to 2
* arguments instead of one originally.
* 3nd August 2015: support for Procedural functions
* 1.5.6 addons:
* 8th September 2015: byte new integer subtype added
* ToDo: Two new integer subtypes: 24 and
* 32 bits integer now understood (convenience for array
* variables of this integer subtypes).
* ToDo: Possibility to store indiv. array content
* within aux mem (auxiliary memory Apple and AE RAMWorks
* protocol)
* Merlin 8 assembler
* Constants
VERSION  =     $15
K6502    =     0
K65C02   =     1
K65816   =     1
* Generate either 65(816!C)02 compatible version
KOPT     =     K6502
KNEW     =     1
KNEW2    =     1
KOPTLNG32 =    1
KOPTLNG33 =    0
* Cache size (# of entries) for simple variables
KSNCACH  =     4
* Cache size (# of entries) for array variables
KANCACH  =     4

KOPT16   =     0

* Token equates
TOKEQUAL =     $D0
TOKADD   =     $C8
TOKMUL   =     $CA
TOKDIV   =     $CB
TOKDEF   =     $B8        Prefix for DEF(INT!STR!SNG)
TOKINT   =     $D3        DEFINT instr st. as 2 tokens
TOKUSR   =     $D5        DEFUSR...
TOKMINUS =     $C9
TOKREM   =     $B2
TOKDATA  =     $83
TOKIF    =     $AD
TOKFN    =     $C2
TOKTO    =     $C1
TOKSTRD  =     $E4
TOKCHRD  =     $E7
TOKSGN   =     $D2
TOKSCRN  =     $D7
TOKNOT   =     $C6
```

```
TOKSTEP  =     $C7
TOKGOSUB =     $B0
TOKGOTO  =     $AB
TOKFOR   =     $81

* Page zero and monitor equates
PCL      EQU   $3A
LENGTH   EQU   $2F
INSDS2   EQU   $F88C
PCADJ    EQU   $F953
A1L      EQU   $3C
A2L      EQU   $3E
A4L      EQU   $42
MOVE     EQU   $FE2C
CH       EQU   $24
XFER     EQU   $C314
VECZAUX  EQU   $03ED

* Applesoft equates
DIMFLG   EQU   $10        Input to PTRGET
* Output from PTRGET
VALTYP   EQU   $11        $FF if string, 0 if num.
INTTYP   EQU   $12        $80 if integer, 0 otherwise
VARNAM   EQU   $81        Encoded varname 1st char.
VARPNT   EQU   $83        Variable value pointer
SUBFLG   EQU   $14        Parameter for PTRGET routine
LINNUM   EQU   $50        Line # (output from LINGET)
CURLIN   EQU   $75        Current line # (being run)
INDEX    EQU   $5E        General ptr for ROM str. routines
LOWTR    EQU   $9B        Address of BASIC line (output from FNDLIN)
FAC      EQU   $9D        Main floating point accumulator
DEST     EQU   $60        Used by NEXT
STREND   EQU   $6D        End of array memory
FACSIGN  EQU   $A2
FACLO    EQU   $A1
FACMO    EQU   $A0
TXTPTR   EQU   $B8        Pointer to BASIC program memory
OLDTPTR  EQU   $79
REMSTK   EQU   $F8
OLDTEXT  EQU   $79
ARYPNT   EQU   $94        Pointer to array structure
ERRFLG   EQU   $D8        ONERR activivty flag
ERRLIN   EQU   $DA        Offending line #
ERRPOS   EQU   $DC        Where in the offending line #..
ERRNUM   EQU   $DE        Error #
ERRSTK   EQU   $DF        Stack pntr of offending instr.
TXTPSV   EQU   $F4
CURLSV   EQU   $F6

TOKTABL  EQU   $D0D0      Address of internal Applesoft token table
ISLETC   EQU   $E07D      Check whether current char alpha
SYNERR   EQU   $DEC9      Report a SYNTAX ERROR
VLET     EQU   $DA46
VPTRGET  EQU   $DFEF      PTRGET return adress (from stack)
ISCNTC   EQU   $D858      Check for Ctrl-C keystroke
```

```
ADDON    EQU   $D998      Add Y to TXTPTR
LINGET   EQU   $DA0C      Get line number from TXTPTR
CHKMEM   EQU   $D3D6      Check for A 16bit words on stack
COMBYTE  EQU   $E74C      Check for comma and compute

* Applesoft output routines
OUTDO    EQU   $DB5C      Generic
CRDO     EQU   $DAFB      Carriage return
OUTSPC   EQU   $DB57      Space
FNDLIN   EQU   $D61A      From line number (LINNUM) to address
NEWSTT   EQU   $D7D2      Applesoft main exec loop
FORPNT   EQU   $85
FRMEVL   EQU   $DD7B      Eval. expr pointed to by TXTPTR
FRMNUM   EQU   $DD67      Eval. expr & ensure numeric result
GETADR   EQU   $E752      Expression to 16bits integer
GETBYT   EQU   $E6F8      Eval. expr into single byte value
* Some checking about FAC:must contain..
CHKNUM   EQU   $DD6A      a scalar factor
CHKSTR   EQU   $DD6C      a string factor
AYINT    EQU   $E10C      Integer conversion from FP
* Some floating point computing dst is FAC1
FSUB     EQU   $E7A7      (Y,A) - FAC1
FADD     EQU   $E7BE      (Y,A) + FAC1
FMULT    EQU   $E97F      (Y,A) * FAC1
FDIV     EQU   $EA66      (Y,A) / FAC1
NEGOP    EQU   $EED0      -FAC1
* Raise some Applesoft errors
GOSTLERR EQU   $E5B2      STRING TOO LONG
GOOVFERR EQU   $E8D5      OVERFLOW
GOTMIERR EQU   $DD76      TYPE MISMATCH
GODVZERR EQU   $EAE1      DIVIDE BY ZERO
GOIQERR  EQU   $E199      ILLEGAL QUANTITY
FREESPC  EQU   $71
STRSPA   EQU   $E3DD      Get space from string pool for a string of len A
DSCTMP   EQU   $9D        Temporary string pointer
STRING1  EQU   $AB        String pointer used by copy
MOVINS   EQU   $E5D4      Move string(STRING1) into memory(FRESPC)
ERRDIR   EQU   $E306      Raises a illegal direct mode iif required
DATAN    EQU   $D9A3      Scan ahead to next EOI
DATA     EQU   $D995      TXTPTR points to next separator
VARTAB   EQU   $69        Begin of simple var. mem. area
ARYTAB   EQU   $6B        Begin of array var. mem. area

FRMSTCK3 EQU   $DE20

* ZP slots used by integer signed 16bits mult/div subroutines
MCAND    EQU   $C0
MPLIER   EQU   $C2
DIVEND   EQU   MPLIER
DIVSOR   EQU   $C0
PARTIAL  EQU   $BE
```

```
AUXBANK  EQU   $BF
LETINF   EQU   $C0
TYPMOD   EQU   $C1
INTTYPSV EQU   $C7
VALTYPSV EQU   $C8

* DOS 3.3 equates
OPRND    EQU   $44
DBUFP    EQU   $9D00

         ORG   $4000

AUXPTR   EQU   $06
IDMOCL   EQU   $BD
OFFSET   EQU   $C2
XSAV     EQU   $B4
YSAV     EQU   $B5
MODREM   EQU   $BE
MODDAT   EQU   $BF
GFLAG    EQU   $C0
IDX0     EQU   $C0
DEFFLG   EQU   $C1
NOPER    =     4

EMOV     MAC
         LDA   ]1
         STA   ]2
         <<<

STD      MAC
         EMOV  ]1;]2
         EMOV  ]1+1;]2+1
         <<<

* 16bits immediate store
STID     MAC
         EMOV  #]1;]2
         EMOV  #>]1;]2+1
         <<<

* Copy a large memory area within
* adressable memory
MOVM     MAC
         STID  ]1;A1L
         STID  ]2;A2L
         STID  ]3;A4L
         JSR   MOVE
         <<<

* Copy a small memory area within
* adressable memory
SMOVE    MAC
         LDX   #]3
LOOP     LDA   ]1-1,X
         STA   ]2-1,X
         DEX
         BNE   LOOP
```

```
          <<<

* Macros for simulating 65C02 instructions
* on a 6502
MPHX      MAC
          DO    KOPT-K65C02
          TXA
          PHA
          ELSE
          PHX
          FIN
          <<<

MPHY      MAC
          DO    KOPT-K65C02
          TYA
          PHA
          ELSE
          PHY
          FIN
          <<<

MPLX      MAC
          DO    KOPT-K65C02
          PLA
          TAX
          ELSE
          PLX
          FIN
          <<<

MPLY      MAC
          DO    KOPT-K65C02
          PLA
          TAY
          ELSE
          PLY
          FIN
          <<<

MTSB      MAC
          DO    KOPT-K65C02
          ORA   ]1
          STA   ]1
          ELSE
          TSB   ]1
          FIN
          <<<

GOTO      MAC
          DO    KOPT-K6502
          BRA   ]1
          ELSE
          JMP   ]1
          FIN
          <<<

```

```
               * Do all the stuff for installing Peersoft
               * between DOS and its buffers
                        PUT   PEERINSTALL
               NEWY     EQU   $47

               * This module deals with all installation stuff for the
               * Peersoft suite
4000: A9 D3    SUITE    LDA   #$9CD3     Compute the offset
4002: 38                SEC              ;Put it in :0+1 (lobyte)
4003: ED 00 9D          SBC   DBUFP       and :1+1 (hibyte)
4006: 8D 49 40          STA   :0+1
4009: A9 9C             LDA   #>$9CD3
400B: ED 01 9D          SBC   DBUFP+1
400E: AA                TAX
400F: 0D 49 40          ORA   :0+1
               * If first utility to ask for memory this way, then ask for
               * one additional page for our own purpose (i.e. Bananasoft
               * or Peersoft)
4012: F0 01             BEQ   :6
4014: CA                DEX
4015: 8E 51 40 :6       STX   :1+1

               * Relocate code (don´t move it yet)
4018: A9 7B             LDA   #AROMBA
401A: A0 47             LDY   #>AROMBA
401C: 85 3A    ]LOOP    STA   PCL
401E: C9 25             CMP   #FCODE-FNDVAR2+AROMBA
4020: 98                TYA
4021: E9 68             SBC   #>FCODE-FNDVAR2+AROMBA
4023: B0 35             BCS   :4
4025: 84 3B             STY   PCL+1
4027: A2 00             LDX   #0
4029: 20 8C F8          JSR   MINSDS2
402C: A4 2F             LDY   LENGTH
402E: C0 02             CPY   #2         Only relocates 3 bytes instr.
4030: D0 22             BNE   :3
4032: B1 3A             LDA   (PCL),Y
4034: AA                TAX
4035: 88                DEY
4036: B1 3A             LDA   (PCL),Y
4038: A8                TAY
4039: C9 00             CMP   #FIN       Only if adress within range
403B: 8A                TXA
403C: E9 9C             SBC   #>FIN
403E: B0 14             BCS   :3         Must be < FIN to be relocated
4040: C0 24             CPY   #FNDVAR2
4042: 8A                TXA
4043: E9 75             SBC   #>FNDVAR2
4045: 90 0D             BCC   :3         Must be >= FNDVAR2
4047: 98                TYA              ;Relocates address
4048: E9 00    :0       SBC   #0
404A: A0 01             LDY   #1
404C: 91 3A             STA   (PCL),Y    Low byte
404E: C8                INY
404F: 8A                TXA
4050: E9 00    :1       SBC   #0
4052: 91 3A             STA   (PCL),Y    High byte
```

```
4054: 20 53 F9 >76   :3       JSR   PCADJ      Adjust PCL to length byte
4057: 4C 1C 40 >77            JMP   ]LOOP      Loop
               >78
               >80
               >81   * Relocate some non trivial references (i.e. instructions
               >82   * with immediate adressing mode).
405A: A2 13    >83   :4       LDX   #ADPFT-ADPFB-1
405C: BD 2F 6E >84   ]LOOP    LDA   ADPFB+AROMBA-FNDVAR2,X
405F: 38       >85            SEC
4060: ED 49 40 >86            SBC   :0+1
4063: 9D 2F 6E >87            STA   ADPFB+AROMBA-FNDVAR2,X
4066: BD 43 6E >88            LDA   ADPFT+AROMBA-FNDVAR2,X
4069: ED 51 40 >89            SBC   :1+1
406C: 9D 43 6E >90            STA   ADPFT+AROMBA-FNDVAR2,X
406F: CA       >91            DEX
4070: 10 EA    >92            BPL   ]LOOP
               >93
4072: A2 0C    >94            LDX   #ADT1-ADB1-1
4074: A9 00    >95            LDA   #0
4076: 85 3A    >96            STA   PCL
4078: BD F3 41 >97   ]LOOP    LDA   ADT1,X
407B: 85 3B    >98            STA   PCL+1
407D: BC E6 41 >99            LDY   ADB1,X
4080: B1 3A    >100           LDA   (PCL),Y
4082: 38       >101           SEC
4083: ED 49 40 >102           SBC   :0+1
4086: 91 3A    >103           STA   (PCL),Y
4088: BD 0D 42 >104           LDA   ADT2,X
408B: 85 3B    >105           STA   PCL+1
408D: BC 00 42 >106           LDY   ADB2,X
4090: B1 3A    >107           LDA   (PCL),Y
4092: ED 51 40 >108           SBC   :1+1
4095: 91 3A    >109           STA   (PCL),Y
4097: CA       >110           DEX
4098: 10 DE    >111           BPL   ]LOOP
               >112
409A: A2 0F    >113           LDX   #OFFSTT-OFFSTB-1
409C: BD 7C 68 >114  ]LOOP    LDA   OFFSTB+AROMBA-FNDVAR2,X
409F: 38       >115           SEC
40A0: ED 49 40 >116           SBC   :0+1
40A3: 9D 7C 68 >117           STA   OFFSTB+AROMBA-FNDVAR2,X
40A6: BD 8C 68 >118           LDA   OFFSTT+AROMBA-FNDVAR2,X
40A9: ED 51 40 >119           SBC   :1+1
40AC: 9D 8C 68 >120           STA   OFFSTT+AROMBA-FNDVAR2,X
40AF: CA       >121           DEX
40B0: 10 EA    >122           BPL   ]LOOP
               >123  * Move the code
40B2: A9 24    >124           LDA   #CGARBAG
40B4: A2 75    >125           LDX   #>CGARBAG
40B6: 38       >126           SEC
40B7: ED 49 40 >127           SBC   :0+1
40BA: 85 42    >128           STA   A4L
40BC: 8A       >129           TXA
40BD: ED 51 40 >130           SBC   :1+1
40C0: 85 43    >131           STA   A4L+1
               >132
40C2: A9 7B    >133           LDA   #CGARBAG+AROMBA-FNDVAR2
```

```
40C4: A2 47             LDX   #>CGARBAG+AROMBA-FNDVAR2
40C6: 85 3C             STA   A1L
40C8: 86 3D             STX   A1L+1

40CA: A9 56             LDA   #FIN-1+AROMBA-FNDVAR2
40CC: 85 3E             STA   A2L
40CE: A9 6E             LDA   #>FIN-1+AROMBA-FNDVAR2
40D0: 85 3F             STA   A2L+1

40D2: A0 00             LDY   #0
40D4: 2C 81 C0          BIT   $C081
40D7: 2C 81 C0          BIT   $C081
40DA: 20 2C FE          JSR   MOVE
               * Reconstruct DOS buffers below PeerSoft
40DD: AD 00 9D          LDA   DBUFP
40E0: AE 01 9D          LDX   DBUFP+1
40E3: C9 D3             CMP   #$9CD3
40E5: D0 05             BNE   :7
40E7: E0 9C             CPX   #>$9CD3
40E9: D0 01             BNE   :7         One more page if first utility
40EB: CA                DEX              ; to install this way
40EC: 38       :7       SEC
40ED: E9 DC             SBC   #LONGLANG
40EF: A8                TAY
40F0: 8A                TXA
40F1: E9 26             SBC   #>LONGLANG
40F3: 8C 00 9D          STY   DBUFP      New DOS base buffer address
40F6: 8D 01 9D          STA   DBUFP+1
40F9: 20 D4 A7          JSR   $A7D4

40FC: A9 15             LDA   #VERSION
40FE: 8D DE 9C          STA   PVERSION
4101: A9 80             LDA   #$80
4103: 8D D0 9C          STA   OPTCGOTO
4106: 0A                ASL              ;Let zero acc
4107: 8D CF 9C          STA   NEEDDEC

               * Number of Applesoft instruction runs
               * between two consecutives context switches
410A: A9 0A             LDA   #10
410C: 8D DD 9C          STA   ICTRACTV
410F: A9 00             LDA   #0
4111: 8D DC 9C          STA   MTACTV
4114: A9 4C             LDA   #$4C
4116: 8D DF 9C          STA   REVECTOR
4119: 8D D5 9C          STA   VGARBAG
411C: 38                SEC
411D: A9 A4             LDA   #ROUTGEN
411F: ED 49 40          SBC   :0+1
4122: 8D E0 9C          STA   REVECTOR+1
4125: A9 86             LDA   #>ROUTGEN
4127: ED 51 40          SBC   :1+1
412A: 8D E1 9C          STA   REVECTOR+2
412D: A9 B8             LDA   #NPTRGL90
412F: ED 49 40          SBC   :0+1
4132: 8D D3 9C          STA   VNPTRG90
4135: A9 79             LDA   #>NPTRGL90
```

```
4137: ED 51 40           SBC   :1+1
413A: 8D D4 9C           STA   VNPTRG90+1
413D: A9 CA              LDA   #NARRGL91
413F: ED 49 40           SBC   :0+1
4142: 8D D1 9C           STA   VNARRG91
4145: A9 7A              LDA   #>NARRGL91
4147: ED 51 40           SBC   :1+1
414A: 8D D2 9C           STA   VNARRG91+1
414D: A9 E1              LDA   #TABOFB
414F: ED 49 40           SBC   :0+1
4152: 8D D8 9C           STA   ADADR
4155: A9 95              LDA   #>TABOFB
4157: ED 51 40           SBC   :1+1
415A: 8D D9 9C           STA   ADADR+1
415D: A2 84              LDX   #GARBAG
415F: A9 E4              LDA   #>GARBAG
4161: 2C EF 9C           BIT   MEMORY
4164: 10 0B              BPL   *+13
4166: A9 2F              LDA   #NGARBAG
4168: ED 49 40           SBC   :0+1
416B: AA                 TAX
416C: A9 7D              LDA   #>NGARBAG
416E: ED 51 40           SBC   :1+1
4171: 8E D6 9C           STX   VGARBAG+1
4174: 8D D7 9C           STA   VGARBAG+2
4177: A9 F2              LDA   #NDSVCMD   New DOS Save for applesoft
4179: ED 49 40           SBC   :0+1
417C: 8D A6 A3           STA   $A3A6
417F: A9 91              LDA   #>NDSVCMD
4181: ED 51 40           SBC   :1+1
4184: 8D A7 A3           STA   $A3A7
4187: A9 FA              LDA   #NDLVCMD   Part of routine for loading
4189: ED 49 40           SBC   :0+1
418C: 8D 2E A4           STA   $A42E
418F: A9 91              LDA   #>NDLVCMD
4191: ED 51 40           SBC   :1+1
4194: 8D 2F A4           STA   $A42F
4197: A9 20              LDA   #$20
4199: 8D 9E 9E           STA   $9E9E
419C: A9 D9              LDA   #NKBDINT
419E: ED 49 40           SBC   :0+1
41A1: 8D 9F 9E           STA   $9E9F
41A4: A9 91              LDA   #>NKBDINT
41A6: ED 51 40           SBC   :1+1
41A9: 8D A0 9E           STA   $9EA0
41AC: 20 1A 42           JSR   BIGRECON
41AF: 20 C4 42           JSR   MOUSEDET
41B2: 2C EF 9C           BIT   MEMORY
41B5: 50 09              BVC   :44
               * Copy $F8-$FF pages within ROM to main and aux
               * memory banks
41B7: 20 2C 43           JSR   COPYROM
               * Initialize BF page
41BA: 20 EB 43           JSR   INITBF
41BD: 20 D7 41           JSR   MZRTAUX
41C0: 2C 80 C0 :44       BIT   $C080
41C3: 2C 80 C0           BIT   $C080
```

```
               >253  * If Applesoft is the active language, so
               >254  * install Peersoft CHRGET/CHRGOT patch
41C6: AD B6 AA >255  EK       LDA   $AAB6
41C9: F0 09    >256           BEQ   :11
41CB: 2C 81 C0 >257           BIT   $C081
41CE: 2C 81 C0 >258           BIT   $C081
41D1: 20 A0 81 >259           JSR   SETUPB
41D4: 4C B7 81 >260  :11      JMP   SETUPD
               >261
41D7: A9 BF    >262  MZRTAUX  LDA   #$BF
41D9: A2 00    >263           LDX   #0
41DB: 8D EE 03 >264           STA   $03EE
41DE: 8E ED 03 >265           STX   $03ED
41E1: B8       >266           CLV
41E2: 38       >267           SEC
41E3: 4C 14 C3 >268           JMP   XFER
               >269
               >271  MC       DO    KOPT16
               >287  LN       DO    KOPT16
               >396  MINSDS2  EQU   INSDS2
               >398
41E6: CF       >405  ADB1     DFB   EK+9
41E7: D2       >406           DFB   EK+12
41E8: FE       >407           DFB   SETUPB+7+AROMBA-FNDVAR2
41E9: 06       >408           DFB   SETUPB+15+AROMBA-FNDVAR2
41EA: 0F       >409           DFB   SETUPD+1+AROMBA-FNDVAR2
41EB: 8D       >410           DFB   STP1+1+AROMBA-FNDVAR2
41EC: 28       >411           DFB   SFE1+1+AROMBA-FNDVAR2
41ED: 17       >412           DFB   SETLTR+1
41EE: BC       >416           DFB   GN65536+1+AROMBA-FNDVAR2
41EF: B2       >417           DFB   GN32768+1+AROMBA-FNDVAR2
41F0: C1       >418           DFB   GP65536+1+AROMBA-FNDVAR2
41F1: FD       >422           DFB   NAMNTFND+5
41F2: E1       >426           DFB   V3B+1+AROMBA-FNDVAR2
41F3: 41       >428  ADT1     DFB   >EK+9
41F4: 41       >429           DFB   >EK+12
41F5: 53       >430           DFB   >SETUPB+7+AROMBA-FNDVAR2
41F6: 54       >431           DFB   >SETUPB+15+AROMBA-FNDVAR2
41F7: 54       >432           DFB   >SETUPD+1+AROMBA-FNDVAR2
41F8: 61       >433           DFB   >STP1+1+AROMBA-FNDVAR2
41F9: 61       >434           DFB   >SFE1+1+AROMBA-FNDVAR2
41FA: 89       >435           DFB   >SETLTR+1
41FB: 60       >439           DFB   >GN65536+1+AROMBA-FNDVAR2
41FC: 60       >440           DFB   >GN32768+1+AROMBA-FNDVAR2
41FD: 60       >441           DFB   >GP65536+1+AROMBA-FNDVAR2
41FE: 79       >445           DFB   >NAMNTFND+5
41FF: 51       >449           DFB   >V3B+1+AROMBA-FNDVAR2
4200: D0       >451  ADB2     DFB   EK+10
4201: D3       >452           DFB   EK+13
4202: 02       >453           DFB   SETUPB+11+AROMBA-FNDVAR2
4203: 0A       >454           DFB   SETUPB+19+AROMBA-FNDVAR2
4204: 14       >455           DFB   SETUPD+6+AROMBA-FNDVAR2
4205: 8F       >456           DFB   STP1+3+AROMBA-FNDVAR2
4206: 2A       >457           DFB   SFE1+3+AROMBA-FNDVAR2
4207: 1B       >458           DFB   SETLTR+5
4208: BE       >462           DFB   GN65536+3+AROMBA-FNDVAR2
4209: B4       >463           DFB   GN32768+3+AROMBA-FNDVAR2
```

```
420A: C3       >464           DFB   GP65536+3+AROMBA-FNDVAR2
420B: 04       >468           DFB   NAMNTFND+12
420C: DE       >472           DFB   V3T+1+AROMBA-FNDVAR2
420D: 41       >474  ADT2     DFB   >EK+10
420E: 41       >475           DFB   >EK+13
420F: 54       >476           DFB   >SETUPB+11+AROMBA-FNDVAR2
4210: 54       >477           DFB   >SETUPB+19+AROMBA-FNDVAR2
4211: 54       >478           DFB   >SETUPD+6+AROMBA-FNDVAR2
4212: 61       >479           DFB   >STP1+3+AROMBA-FNDVAR2
4213: 61       >480           DFB   >SFE1+3+AROMBA-FNDVAR2
4214: 89       >481           DFB   >SETLTR+5
4215: 60       >485           DFB   >GN65536+3+AROMBA-FNDVAR2
4216: 60       >486           DFB   >GN32768+3+AROMBA-FNDVAR2
4217: 60       >487           DFB   >GP65536+3+AROMBA-FNDVAR2
4218: 7A       >491           DFB   >NAMNTFND+12
4219: 51       >495           DFB   >V3T+1+AROMBA-FNDVAR2
               >497
421A: 2C 81 C0 >498  BIGRECON BIT   $C081
421D: 2C 81 C0 >499           BIT   $C081
               >500  * What is the model/ROM version of the Apple
4220: A0 07    >501           LDY   #8-1
4222: AD B3 FB >502           LDA   $FBB3
4225: 4D C0 FB >503           EOR   $FBC0
4228: 4D BF FB >504           EOR   $FBBF
422B: D9 9A 42 >505  ]LOOP    CMP   MACMAT,Y
422E: F0 04    >506           BEQ   :1
4230: 88       >507           DEY
4231: 10 F8    >508           BPL   ]LOOP
4233: 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)
4234: C0 02    >513  :1       CPY   #2
4236: D0 20    >514           BNE   :2
4238: AD 5C FC >515           LDA   $FC5C
423B: C9 EB    >516           CMP   #$EB
423D: D0 19    >517           BNE   :2
423F: A0 08    >518           LDY   #8          //gs!
4241: 18       >519           CLC
4242: FB       >520           HEX   FB          ;XCE: Enter native mode
4243: 08       >521           PHP               ;Push carry status (old emu bit)
4244: C2 30    >522           HEX   C230        Set 16bits mode
4246: 20 1F FE >523           JSR   $FE1F       Call ID firmware routine
4249: 84 47    >524           STY   NEWY
424B: 28       >525           PLP               ;Restore original emulation bit
424C: FB       >526           HEX   FB          ;XCE: Exit native mode
424D: A0 0C    >527           LDY   #12
424F: A5 48    >528           LDA   NEWY+1
4251: D0 05    >529           BNE   :2
4253: A5 47    >530           LDA   NEWY
4255: 09 08    >531           ORA   #8
4257: A8       >532           TAY
               >533
4258: B9 A2 42 >534  :2       LDA   MCODE,Y
425B: 8D ED 9C >535           STA   MACHINE
425E: 98       >536           TYA
425F: AA       >537           TAX
```

```
4260: D0 26    >538           BNE   :3         00 if Apple 2+
               >539  * Test for Apple2+, X=0 upon entry
               >540  * Possible language card being there..
4262: 2C 83 C0 >541           BIT   $C083
4265: 2C 83 C0 >542           BIT   $C083
4268: AD 00 D0 >543           LDA   $D000
426B: C8       >544           INY
426C: 8C 00 D0 >545           STY   $D000
426F: CC 00 D0 >546           CPY   $D000      Read after write (1st)
4272: D0 0A    >547           BNE   :5
4274: EE 00 D0 >548           INC   $D000
4277: C8       >549           INY
4278: CC 00 D0 >550           CPY   $D000      Read after increment (2nd)
427B: D0 01    >551           BNE   :5
427D: E8       >552           INX
427E: 8D 00 D0 >553  :5       STA   $D000
4281: BD B4 42 >554           LDA   CFA,X
4284: A2 00    >555           LDX   #0
4286: F0 0B    >556           BEQ   :4
4288: C9 04    >557  :3       CMP   #4         Apple //c or //gs?
428A: A9 C0    >558           LDA   #$C0
428C: A2 80    >559           LDX   #$80
428E: B0 03    >560           BCS   :4         Yes
4290: 20 68 43 >561           JSR   TEST2E
4293: 8D EF 9C >562  :4       STA   MEMORY
4296: 8E F0 9C >563           STX   VID80C
4299: 60       >564           RTS
               >565
429A: EA 2D E6 >566  MACMAT   HEX   EA2DE6E7F9060502
42A2: 00       >567  MCODE    HEX   00         Apple 2+
42A3: 40 41 42 >568           HEX   404142     Apple //e
42A6: 80 81 82 >569           HEX   80818283   Apple //c
42AA: C0 C1 C2 >570           HEX   C0C1C2C3C4C5 Apple //gs
42B0: 80 80 C0 >571  CFM      HEX   8080C0C0
42B4: 00 80 80 >572  CFA      HEX   008080C0
               >573
42B8: 05 07 0B >574  DATA1IDX DFB   5,7,11,12,17,251
42BE: 38 18 01 >575  DATA1VAL HEX   3818012000D6
               >576  * Routine to detect a mouse card
42C4: A2 C7    >577  MOUSEDET LDX   #$C7
42C6: 86 07    >578           STX   AUXPTR+1
42C8: 8E CE 9C >579           STX   MOSL       ;b7 of MOSL set to 1
42CB: A2 00    >585           LDX   #0
42CD: 86 06    >586           STX   AUXPTR
42CF: 8E B7 99 >587           STX   MOCN
42D2: 8E B5 99 >588           STX   MON0
42D5: A2 05    >590  ]LOOP    LDX   #DATA1VAL-DATA1IDX-1
42D7: BC B8 42 >591  ]LOOP1   LDY   DATA1IDX,X
42DA: BD BE 42 >592           LDA   DATA1VAL,X
42DD: 51 06    >593           EOR   (AUXPTR),Y
42DF: D0 3F    >594           BNE   :1
42E1: CA       >595           DEX
42E2: 10 F3    >596           BPL   ]LOOP1
42E4: A5 07    >597           LDA   AUXPTR+1
42E6: 8D B7 99 >598           STA   MOCN
42E9: 29 0F    >599           AND   #$F
42EB: 8D CE 9C >600           STA   MOSL
```

```
42EE: 0A       >602              ASL
42EF: 0A       >602              ASL
42F0: 0A       >602              ASL
42F1: 0A       >602              ASL
42F2: 8D B5 99 >604              STA   MON0
42F5: E8       >605              INX                ;X = 0
42F6: EC ED 9C >606              CPX   MACHINE      Is host an Apple2 or 2+?
42F9: D0 13    >607              BNE   :2
               >608  * Time to INITMOUSE..
42FB: A0 19    >609              LDY   #$19         Offset to INIT mouse offset
42FD: B1 06    >610              LDA   (AUXPTR),Y
42FF: 85 06    >611              STA   AUXPTR
4301: A6 07    >612              LDX   AUXPTR+1
4303: AC B5 99 >613              LDY   MON0
4306: 20 29 43 >614              JSR   :0
4309: 90 03    >615              BCC   :2
430B: 6E CE 9C >616              ROR   MOSL         Let set b7 of mouse slot
430E: A2 07    >617  :2          LDX   #OM_INI-OM_DEB
4310: A9 00    >621              LDA   #0
4312: 85 06    >622              STA   AUXPTR
4314: BC AD 99 >624  ]JLOOP      LDY   OM_DEB,X
4317: B1 06    >625              LDA   (AUXPTR),Y
4319: 9D AD 99 >626              STA   OM_DEB,X
431C: CA       >627              DEX
431D: 10 F5    >628              BPL   ]JLOOP
431F: 60       >629              RTS
4320: A6 07    >630  :1          LDX   AUXPTR+1
4322: E0 C1    >631              CPX   #$C1
4324: C6 07    >632              DEC   AUXPTR+1
4326: B0 AD    >633              BCS   ]LOOP
4328: 60       >634  :FIN        RTS
4329: 6C 06 00 >635  :0          JMP   (AUXPTR)
               >636
               >637  * Routine to copy ROM to bank switched RAM
432C: A0 00    >638  COPYROM     LDY   #0
432E: A9 F8    >639              LDA   #$F8
4330: 84 3C    >640              STY   A1L
4332: 85 3D    >641              STA   A1L+1
4334: 8D 09 C0 >642              STA   $C009        Write into aux ZP
4337: 84 3C    >643              STY   A1L
4339: 85 3D    >644              STA   A1L+1
433B: 8D 08 C0 >645              STA   $C008        Write back into main ZP
433E: 2C 89 C0 >646              BIT   $C089        Write into LC ram
4341: 2C 89 C0 >647              BIT   $C089
4344: B1 3C    >648  ]LOOP       LDA   (A1L),Y
4346: 91 3C    >649              STA   (A1L),Y      within main memory
4348: 8D 09 C0 >650              STA   $C009        Write into aux memory LC bank
434B: 91 3C    >651              STA   (A1L),Y
434D: 8D 08 C0 >652              STA   $C008        Back to writing to main memory
4350: C8       >653              INY
4351: D0 F1    >654              BNE   ]LOOP
4353: E6 3D    >655              INC   A1L+1
4355: A5 3D    >656              LDA   A1L+1
4357: 8D 09 C0 >657              STA   $C009
435A: 85 3D    >658              STA   A1L+1
435C: 8D 08 C0 >659              STA   $C008
435F: D0 E3    >660              BNE   ]LOOP
```

```
4361: 2C 81 C0 >661          BIT   $C081
4364: 2C 81 C0 >662          BIT   $C081
4367: 60       >663          RTS
               >664
               >665  * Routine to test //e configuration: 80 col. card?
               >666  * memory expansion?
4368: 08       >667  TEST2E   PHP
4369: 78       >668          SEI
436A: A2 00    >669          LDX   #0
436C: AD 17 C0 >670          LDA   $C017
436F: 30 6F    >671          BMI   :6
4371: E8       >672          INX
4372: AD 1D C0 >673          LDA   $C01D
4375: 48       >674          PHA
4376: AD 18 C0 >675          LDA   $C018
4379: 48       >676          PHA
437A: AD 1C C0 >677          LDA   $C01C
437D: 48       >678          PHA
437E: AD 19 C0 >679  ]LOOP    LDA   $C019
4381: 30 FB    >680          BMI   ]LOOP
4383: 8D 57 C0 >681          STA   $C057
4386: 8D 01 C0 >682          STA   $C001
4389: 8D 55 C0 >683          STA   $C055
438C: AD 00 04 >684          LDA   $400
438F: 48       >685          PHA
4390: AD 00 24 >686          LDA   $2400
4393: 48       >687          PHA
4394: A9 EE    >688          LDA   #$EE
4396: 8D 00 04 >689          STA   $0400
4399: AD 00 24 >690          LDA   $2400
439C: C9 EE    >691          CMP   #$EE
439E: D0 0B    >692          BNE   :2
43A0: 0E 00 24 >693          ASL   $2400
43A3: AD 00 04 >694          LDA   $0400
43A6: CD 00 24 >695          CMP   $2400
43A9: F0 1B    >696          BEQ   :3
43AB: E8       >697  :2       INX
43AC: A9 0F    >698          LDA   #$0F
43AE: 8D B9 C0 >699          STA   $C0B9
43B1: 8D 54 C0 >700          STA   $C054
43B4: AD 00 04 >701          LDA   $0400
43B7: 8D 00 04 >702          STA   $0400
43BA: 8D B8 C0 >703          STA   $C0B8
43BD: 8D 55 C0 >704          STA   $C055
43C0: AD 00 04 >705          LDA   $0400
43C3: 30 01    >706          BMI   :3
43C5: E8       >707          INX
43C6: 68       >708  :3       PLA
43C7: 8D 00 24 >709          STA   $2400
43CA: 68       >710          PLA
43CB: 8D 00 04 >711          STA   $0400
43CE: 68       >712          PLA
43CF: 30 03    >713          BMI   :4
43D1: 8D 54 C0 >714          STA   $C054
43D4: 68       >715  :4       PLA
43D5: 30 03    >716          BMI   :5
43D7: 8D 00 C0 >717          STA   $C000
```

```
43DA: 68        >718  :5        PLA
43DB: 30 03     >719            BMI   :6
43DD: 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)
43E0: BD B0 42 >725   :6        LDA   CFM,X
43E3: 48        >726            PHA
43E4: BD B4 42 >727             LDA   CFA,X
43E7: AA        >728            TAX
43E8: 68        >729            PLA
43E9: 28        >730            PLP
43EA: 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
43EB: A9 64     >11             LDA   #CODE1BF
43ED: 85 3C     >11             STA   A1L
43EF: A9 44     >11             LDA   #>CODE1BF
43F1: 85 3D     >11             STA   A1L+1
43F3: A0 00     >12             LDY   #0
43F5: A9 00     >13             LDA   #GZAUXRT
43F7: 85 3E     >13             STA   A2L
43F9: A9 BF     >13             LDA   #>GZAUXRT
43FB: 85 3F     >13             STA   A2L+1
43FD: 8D 05 C0 >14              STA   $C005
4400: B1 3C     >15   ]LOOP     LDA   (A1L),Y
4402: 91 3E     >16             STA   (A2L),Y
4404: C8        >17             INY
4405: C0 BC     >18             CPY   #CODE2BF-CODE1BF
4407: D0 F7     >19             BNE   ]LOOP
4409: 8D 04 C0 >20              STA   $C004
440C: 08        >21             PHP
440D: 08        >22             PHP
440E: 68        >23             PLA
440F: 78        >24             SEI
4410: BA        >25             TSX
4411: 8E 09 C0 >26              STX   ALTZP
4414: 8E 00 01 >27              STX   $0100
4417: A2 FF     >28             LDX   #$FF
4419: 9A        >29             TXS
441A: 8E 01 01 >30              STX   $0101
441D: 29 04     >31             AND   #%100
441F: D0 01     >32             BNE   *+3
4421: 58        >33             CLI
4422: A9 20     >34             LDA   #CODE1LC
```

```
4424: 85 3C             STA   A1L
4426: A9 45             LDA   #>CODE1LC
4428: 85 3D             STA   A1L+1
442A: A9 00             LDA   #$D000
442C: 85 3E             STA   A2L
442E: A9 D0             LDA   #>$D000
4430: 85 3F             STA   A2L+1
4432: 2C 81 C0          BIT   $C081
4435: 2C 81 C0          BIT   $C081
4438: A0 00             LDY   #0
443A: B1 3C    ]LOOP    LDA   (A1L),Y
443C: 91 3E             STA   (A2L),Y
443E: E6 3C             INC   A1L
4440: D0 02             BNE   *+4
4442: E6 3D             INC   A1L+1
4444: A5 3C             LDA   A1L
4446: C9 C3             CMP   #CODE2LC
4448: A5 3D             LDA   A1L+1
444A: E9 45             SBC   #>CODE2LC
444C: B0 08             BCS   :0
444E: E6 3E             INC   A2L
4450: D0 E8             BNE   ]LOOP
4452: E6 3F             INC   A2L+1
4454: 90 E4             BCC   ]LOOP      Always
4456: 78       :0       SEI
4457: BA                TSX
4458: 8E 01 01          STX   $0101
445B: AE 00 01          LDX   $0100
445E: 9A                TXS
445F: 8E 08 C0          STX   STDZP
4462: 28                PLP
4463: 60       ]RET     RTS

               CODE1BF  ORG   $BF00
               AXHIMEM  EQU   *
               * Routine de redirection pour la gestion des tableaux en
               * memoire auxiliaire.
               * X:0 init the auxilary memory segment for storing
               *     array elements
               * X:1 check that enough room exists for storing an
               *     array´s elements
               * X:2 actually updates the STREND new array end and
               *     initializes the area.
               * X:3 returns the mem bank free space after a garbage c.
               * X:4 retrieve an array´s element from memory.
               * X:5 stores an array´s element into memory
BF00: BC B8 BF GZAUXRT  LDY   ZAUXOFFT,X offset into Y
BF03: A9 00             LDA   #0
BF05: 2C 12 C0          BIT   RDLCRAM
BF08: 10 09             BPL   *+11
BF0A: 09 0C             ORA   #12
BF0C: 2C 11 C0          BIT   RDLCBNK2
BF0F: 10 02             BPL   *+4
BF11: 49 06             EOR   #6
BF13: 48                PHA
BF14: 08                PHP              ;Save I bit flag on main stk
BF15: 68                PLA              ;Restore in b2 of accum.
```

```
BF16: BA              TSX
BF17: 78              SEI
BF18: 8D 09 C0        STA   ALTZP
BF1B: 8E 00 01        STX   $0100
BF1E: A2 FF           LDX   #$FF
BF20: 8E 01 01        STX   $0101
BF23: 9A              TXS
BF24: 29 04           AND   #%100      bit I mask
BF26: D0 01           BNE   *+3
BF28: 58              CLI
BF29: AD 18 C0        LDA   RD80STOR
BF2C: 48              PHA
BF2D: 8D 00 C0        STA   $C000      Enable basic access to screens
           * Read/Write enable LC bank 2 in aux. mem. bec. of ALTZP
BF30: 20 A4 BF        JSR   G83        Read/Write enable LC bank 2 in
BF33: A9 BF           LDA   #>ZAUXRET-1
BF35: 48              PHA
BF36: A9 3C           LDA   #ZAUXRET-1
BF38: 48              PHA
BF39: 18              CLC
BF3A: 4C 14 D0        JMP   ZAUXRT

           * Routine de retour general vers le composant principal
           * de Peersoft (en memoire principale)
BF3D: 68   ZAUXRET    PLA              ;Restore RD80STOR status
BF3E: 10 03           BPL   *+5        from aux stack..
BF40: 8E 01 C0        STX   $C001      If On, then set it back..
BF43: 08              PHP              ;Save carry flag
BF44: 28              PLP              ;Restore carry flag
BF45: AE 00 01        LDX   $0100      Get back main stack pointer
BF48: 9A              TXS              ; from $0100 aux stack byte
BF49: 8E 08 C0        STX   STDZP      Return to Std stack/p0
BF4C: 68              PLA              ;Restore configuration flag
BF4D: 08              PHP              ;Carry back into main stack
BF4E: 20 AB BF        JSR   G81
BF51: 0A              ASL
BF52: F0 0E           BEQ   :0
BF54: A0 05           LDY   #5
BF56: BE B2 BF ]LOOP  LDX   IRQTBLE,Y
BF59: 88              DEY
BF5A: 0A              ASL
BF5B: 90 03           BCC   *+5
BF5D: 9D 00 C0        STA   $C000,X
BF60: D0 F4           BNE   ]LOOP
BF62: 28   :0         PLP
           * X set to zero upon return according to carry flag
BF63: A2 00           LDX   #0
BF65: 90 01           BCC   *+3
BF67: E8              INX
BF68: 68              PLA              ;Get return address
BF69: 18              CLC
BF6A: 69 01           ADC   #1
BF6C: 8D ED 03        STA   $03ED
BF6F: 68              PLA
BF70: A8              TAY
BF71: 90 01           BCC   *+3
BF73: C8              INY
```

```
BF74: 8C EE 03          STY   $03EE
BF77: 18                CLC
BF78: B8                CLV
BF79: 4C 14 C3          JMP   XFER       Retour a l´envoyeur

BF7C: 08       ZGCPARMS PHP
BF7D: 78                SEI
BF7E: 8E 08 C0          STX   STDZP
BF81: A5 AD             LDA   STRNG2
BF83: A6 AE             LDX   STRNG2+1
BF85: 69 07             ADC   #7
BF87: 8E 09 C0          STX   ALTZP
BF8A: 90 06             BCC   :0
BF8C: E8                INX
BF8D: D0 03             BNE   :0
BF8F: 28                PLP
BF90: 38                SEC
BF91: 60                RTS
BF92: 28       :0       PLP
BF93: 18                CLC
BF94: 60       ]RET     RTS

BF95: 8E 08 C0 ZGCP2    STX   STDZP
BF98: 85 AD             STA   STRNG2
BF9A: 8E 09 C0          STX   ALTZP
BF9D: 60                RTS

BF9E: 20 AB BF ZNG      JSR   G81
BFA1: 20 84 E4          JSR   GARBAG
BFA4: 2C 83 C0 G83      BIT   $C083
BFA7: 2C 83 C0          BIT   $C083
BFAA: 60                RTS
BFAB: 2C 81 C0 G81      BIT   $C081
BFAE: 2C 81 C0          BIT   $C081
BFB1: 60                RTS

BFB2: 83 8B 8B IRQTBLE  HEX   838B8B
BFB5: 05 03 55          HEX   050355
BFB8: 00 23    ZAUXOFFT DFB   ZAUXRT0-ZAUXB,ZAUXRT1-ZAUXB
BFBA: 2E E7             DFB   ZAUXRT2-ZAUXB,ZAUXRT3-ZAUXB
                        ERR   */$C000
                        ORG
               CODE2BF
               CODE1LC  ORG   $D000
               * Y offset correspondant a X
               * X:0 init the auxilary memory segment for storing
               *     array elements
               * X:1 check that enough room exists for storing an
               *     array´s elements
               * X:2 actually updates the STREND new array end and
               *     initializes the area.
               * X:3 returns the mem bank free space after a garbage c.
               * X:4 retrieve an array´s element from memory.
               * X:5 stores an array´s element into memory

               * Returns amount of free space in aux memory bank
               * after calling ROM based garbage collection.
```

```
D000: 20 9E 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 95 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 99 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 9A 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: A9 00    >314           LDA   #AXHIMEM
D02D: 85 73    >315           STA   HIMEM
D02F: 85 6F    >316           STA   FRETOP
D031: A9 BF    >318           LDA   #>AXHIMEM
D033: 85 74    >319           STA   HIMEM+1
D035: 85 70    >320           STA   FRETOP+1
D037: A2 55    >322           LDX   #$55        Pour le Garbage collector...
D039: 86 52    >323           STX   $52
D03B: 60       >324  ]RET     RTS
               >325
               >326  * Ensure enough room within array segment
D03C: 20 85 D0 >327  ZAUXRT1  JSR   ZCOMRT12
D03F: B0 FA    >328           BCS   ]RET
D041: C5 6F    >329           CMP   FRETOP
D043: 8A       >331           TXA
D044: E5 70    >332           SBC   FRETOP+1
D046: 60       >334  ]RET     RTS
               >335
D047: A5 6D    >336  ZAUXRT2  LDA   STREND
D049: 85 3C    >337           STA   A1L
D04B: A5 6E    >339           LDA   STREND+1
D04D: 85 3D    >340           STA   A1L+1
D04F: 20 85 D0 >342           JSR   ZCOMRT12
D052: B0 F2    >343           BCS   ]RET
D054: A0 02    >344           LDY   #2
D056: 86 6E    >351           STX   STREND+1
D058: 85 6D    >352           STA   STREND
               >353  * Offset to next array (low byte)
```

```
D05A: 38       >354           SEC
D05B: E5 3C    >355           SBC   A1L
D05D: 91 3C    >356           STA   (A1L),Y
D05F: C8       >357           INY
               >358  * and hi byte
D060: 8A       >359           TXA
D061: E5 3D    >360           SBC   A1L+1
D063: 91 3C    >361           STA   (A1L),Y
               >368  * # of dimensions
D065: A9 01    >369           LDA   #1
D067: A0 04    >370           LDY   #4
D069: 91 3C    >371           STA   (A1L),Y
               >372  * Init segment where elms will be stored
D06B: A2 00    >373           LDX   #0
D06D: A4 3C    >374           LDY   A1L
D06F: 86 3C    >375           STX   A1L
D071: C4 6D    >376  ]LOOP    CPY   STREND
D073: A5 3D    >377           LDA   A1L+1
D075: E5 6E    >378           SBC   STREND+1
D077: B0 0A    >379           BCS   *+12
D079: 8A       >380           TXA
D07A: 91 3C    >381           STA   (A1L),Y
D07C: C8       >382           INY
D07D: D0 F2    >383           BNE   ]LOOP
D07F: E6 3D    >384           INC   A1L+1
D081: 90 EE    >385           BCC   ]LOOP      Always
D083: 18       >386           CLC
D084: 60       >392  ]RET     RTS
               >393
               >466
D085: 20 7C BF >467  ZCOMRT12 JSR   ZGCPARMS
D088: B0 FA    >468           BCS   ]RET
D08A: 65 6D    >469           ADC   STREND
D08C: A8       >473           TAY
D08D: 8A       >474           TXA
D08E: 65 6E    >475           ADC   STREND+1
D090: AA       >476           TAX
D091: 98       >477           TYA
               >478  * Result in X,A
D092: 60       >480  ]RET     RTS
               >481
               >482
D093: FF 80 80 >483  TELMS    HEX   FF8080808000
D099: 00 08    >484  AXARTAB  DA    $0800      0
               >485  AXARYPNT EQU   AXARTAB    2
D09B: 00 00    >486  AXOFFSET DS    2
D09D: 00       >487  ELMSIZ   DS    1          2
D09E: 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   $8F
                >7    NUMELS   =     8
                >8    NUMELS2  =     NUMELS*2
                >9
45C3: A0 00     >10   INITLC   LDY   #0
45C5: A9 DE     >14            LDA   #CODE1GC
45C7: 85 3C     >14            STA   A1L
45C9: A9 45     >14            LDA   #>CODE1GC
45CB: 85 3D     >14            STA   A1L+1
45CD: A9 7B     >15            LDA   #CODE1GCF
45CF: 85 3E     >15            STA   A2L
45D1: A9 47     >15            LDA   #>CODE1GCF
45D3: 85 3F     >15            STA   A2L+1
45D5: 84 42     >16            STY   A4L
45D7: A9 D0     >17            LDA   #>$D000
45D9: 85 43     >18            STA   A4L+1
45DB: 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 E6 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: A0 00     >162           LDY   #0
D015: 84 5F     >163           STY   INDEX+1
D017: C5 52     >165  ]LOOP    CMP   $52
D019: F0 05     >166           BEQ   SVARS
D01B: 20 03 D1  >167           JSR   DVAR
D01E: F0 F7     >168           BEQ   ]LOOP
D020: A9 07     >169  SVARS    LDA   #7
D022: 85 8F     >170           STA   DSCLEN
D024: A5 69     >171           LDA   VARTAB
D026: A6 6A     >172           LDX   VARTAB+1
D028: 85 5E     >173           STA   INDEX
D02A: 86 5F     >174           STX   INDEX+1
D02C: E4 6C     >175  ]LOOP    CPX   ARYTAB+1
D02E: D0 04     >176           BNE   *+6
D030: C5 6B     >177           CMP   ARYTAB
D032: F0 05     >178           BEQ   ARYVAR
D034: 20 F6 D0  >179           JSR   DVARS
D037: F0 F3     >180           BEQ   ]LOOP
D039: 85 94     >181  ARYVAR   STA   ARYPNT
D03B: 86 95     >182           STX   ARYPNT+1
D03D: A9 03     >183           LDA   #3
D03F: 85 8F     >184           STA   DSCLEN
D041: A5 94     >185  ]LOOP    LDA   ARYPNT
D043: A6 95     >186           LDX   ARYPNT+1
D045: E4 6E     >187  ]LOOP1   CPX   STREND+1
D047: D0 04     >188           BNE   *+6
D049: C5 6D     >189           CMP   STREND
D04B: F0 4F     >190           BEQ   GRBPAS
```

```
D04D: 85 5E    >191           STA   INDEX
D04F: 86 5F    >192           STX   INDEX+1
D051: A0 00    >197           LDY   #0
D053: B1 5E    >198           LDA   (INDEX),Y  Name 1st character
D055: C8       >199           INY
D056: AA       >201           TAX
D057: B1 5E    >202           LDA   (INDEX),Y  Name 2nd character
D059: 08       >203           PHP
D05A: C8       >204           INY
D05B: B1 5E    >205           LDA   (INDEX),Y
D05D: 65 94    >206           ADC   ARYPNT     Carry clear
D05F: 85 94    >207           STA   ARYPNT
D061: C8       >208           INY
D062: B1 5E    >209           LDA   (INDEX),Y
D064: 65 95    >210           ADC   ARYPNT+1
D066: 85 95    >211           STA   ARYPNT+1
D068: 28       >212           PLP
D069: 10 D6    >213           BPL   ]LOOP
D06B: 8A       >214           TXA
D06C: 30 D3    >215           BMI   ]LOOP
D06E: C8       >216           INY              ;Y vaut 4
D06F: B1 5E    >217           LDA   (INDEX),Y
D071: AA       >218           TAX
D072: 25 38    >219           AND   %111000
D074: 08       >220           PHP
D075: 8A       >221           TXA
D076: 29 07    >222           AND   #7
D078: 69 01    >226           ADC   #1
D07A: A0 00    >228           LDY   #0
D07C: 0A       >229           ASL
D07D: 28       >230           PLP
D07E: F0 03    >231           BEQ   *+5
D080: 69 0B    >232           ADC   #5+6
D082: 2C       >233           HEX   2C         Skip next two bytes
D083: 69 05    >234           ADC   #5
D085: 65 5E    >235           ADC   INDEX
D087: 85 5E    >236           STA   INDEX
D089: 90 02    >237           BCC   *+4
D08B: E6 5F    >238           INC   INDEX+1
D08D: A6 5F    >239           LDX   INDEX+1
               >240  * End of the array?
D08F: E4 95    >241  ]LOOP    CPX   ARYPNT+1
D091: D0 04    >242           BNE   *+6
D093: C5 94    >243           CMP   ARYPNT
D095: F0 AE    >244           BEQ   ]LOOP1
D097: 20 03 D1 >245           JSR   DVAR
D09A: F0 F3    >246           BEQ   ]LOOP
               >248
               >249  * Have made a complete pass thru the variables
               >250  * Now collect the ones in the list
D09C: A2 0F    >251  GRBPAS   LDX   #NUMELS2-1
D09E: BD AD D1 >266  ]LOOP    LDA   LENTHS,X
D0A1: A8       >267           TAY
D0A2: F0 EE    >268           BEQ   ]RET
D0A4: 38       >269           SEC
D0A5: A5 6F    >270           LDA   FRETOP
D0A7: FD AD D1 >271           SBC   LENTHS,X
```

```
D0AA: 85 6F    >272             STA   FRETOP
D0AC: A5 70    >273             LDA   FRETOP+1
D0AE: E9 00    >274             SBC   #0
D0B0: 85 70    >275             STA   FRETOP+1
D0B2: BD 9C D1 >276             LDA   BTMEL-1,X  Get current place
D0B5: 85 1C    >277             STA   PTR2
D0B7: BD 9D D1 >278             LDA   BTMEL,X
D0BA: 85 1D    >279             STA   PTR2+1
D0BC: 88       >281  ]LOOP1     DEY
D0BD: C0 FF    >282             CPY   #$FF
D0BF: F0 06    >283             BEQ   *+8
D0C1: B1 1C    >284             LDA   (PTR2),Y
D0C3: 91 6F    >285             STA   (FRETOP),Y
D0C5: 90 F5    >286             BCC   ]LOOP1     Always
D0C7: BD AC D1 >287             LDA   LENTHS-1,X Get size of variable
D0CA: 29 04    >288             AND   #4
D0CC: 4A       >289             LSR
D0CD: A8       >290             TAY
D0CE: C8       >291             INY
D0CF: BD BC D1 >296             LDA   VARPT-1,X
D0D2: 85 1C    >297             STA   PTR2
D0D4: BD BD D1 >299             LDA   VARPT,X
D0D7: 85 1D    >300             STA   PTR2+1
D0D9: A5 6F    >302             LDA   FRETOP
D0DB: 91 1C    >303             STA   (PTR2),Y
D0DD: C8       >305             INY
D0DE: A5 70    >306             LDA   FRETOP+1
D0E0: 91 1C    >307             STA   (PTR2),Y
D0E2: CA       >309             DEX
D0E3: CA       >310             DEX
D0E4: 10 B8    >311             BPL   ]LOOP
D0E6: A2 0F    >315  NZTAB      LDX   #NUMELS2-1
D0E8: A9 00    >325             LDA   #0
D0EA: 9D AD D1 >326  ]LOOP      STA   LENTHS,X
D0ED: 9D 9D D1 >327             STA   BTMEL,X
D0F0: CA       >328             DEX
D0F1: 10 F7    >329             BPL   ]LOOP
D0F3: 4C 0B D0 >331             JMP   FNDVARX2
               >333  * Garbage collection for simple variables
D0F6: B1 5E    >334  DVARS      LDA   (INDEX),Y  Is it a string var
D0F8: 30 05    >335             BMI   GDVARTS
D0FA: C8       >336             INY
D0FB: B1 5E    >337             LDA   (INDEX),Y
D0FD: 30 03    >338             BMI   *+5
D0FF: 4C 8D D1 >339  GDVARTS    JMP   DVARTS
D102: C8       >340             INY
D103: B1 5E    >341  DVAR       LDA   (INDEX),Y
D105: F0 F8    >342             BEQ   GDVARTS    Skip Zero length strings
D107: 85 2F    >343             STA   LENGTH
D109: C8       >344             INY
D10A: B1 5E    >345             LDA   (INDEX),Y
D10C: 85 19    >346             STA   STRNG
D10E: C5 6F    >347             CMP   FRETOP     Is this above where we are?
D110: C8       >348             INY
D111: B1 5E    >349             LDA   (INDEX),Y
D113: 85 1A    >350             STA   STRNG+1
D115: E5 70    >351             SBC   FRETOP+1
```

```
D117: B0 E6    >352            BCS   GDVARTS    This one´s been collected before
D119: A5 19    >353            LDA   STRNG      Is it in our range?
D11B: CD 9D D1 >354            CMP   BTMEL      Compare to lowest value in list
D11E: A5 1A    >355            LDA   STRNG+1
D120: ED 9E D1 >356            SBC   BTMEL+1
D123: 90 68    >357            BCC   DVARTS     No, below lowest, go to next one
D125: A5 19    >358            LDA   STRNG
D127: C5 6D    >359            CMP   STREND
D129: A5 1A    >360            LDA   STRNG+1
D12B: E5 6E    >361            SBC   STREND+1
D12D: 90 D0    >362            BCC   GDVARTS    Inside the program...
D12F: A2 11    >363            LDX   #NUMELS2+1 Search thru list of elements
D131: CA       >364   ]LOOP    DEX
D132: CA       >365            DEX
D133: A5 19    >366            LDA   STRNG
D135: DD 9C D1 >367            CMP   BTMEL-1,X
D138: A5 1A    >368            LDA   STRNG+1
D13A: FD 9D D1 >369            SBC   BTMEL,X
D13D: 90 F2    >370            BCC   ]LOOP
D13F: 86 1B    >371            STX   XXSAV
D141: A2 03    >372            LDX   #3         Make room in table for entry
D143: BD 9C D1 >373   ]LOOP    LDA   BTMEL-1,X
D146: 9D 9A D1 >374            STA   BTMEL-3,X
D149: BD 9D D1 >375            LDA   BTMEL,X
D14C: 9D 9B D1 >376            STA   BTMEL-2,X  Ripple down
D14F: BD AD D1 >377            LDA   LENTHS,X
D152: 9D AB D1 >378            STA   LENTHS-2,X
D155: BD AC D1 >379            LDA   LENTHS-1,X
D158: 9D AA D1 >380            STA   LENTHS-3,X
D15B: BD BD D1 >381            LDA   VARPT,X
D15E: 9D BB D1 >382            STA   VARPT-2,X
D161: BD BC D1 >383            LDA   VARPT-1,X
D164: 9D BA D1 >384            STA   VARPT-3,X
D167: E4 1B    >385            CPX   XXSAV
D169: E8       >386            INX
D16A: E8       >387            INX
D16B: 90 D6    >388            BCC   ]LOOP
D16D: A6 1B    >389            LDX   XXSAV
D16F: A5 19    >390            LDA   STRNG
D171: 9D 9C D1 >391            STA   BTMEL-1,X
D174: A5 1A    >392            LDA   STRNG+1
D176: 9D 9D D1 >393            STA   BTMEL,X
D179: A5 2F    >394            LDA   LENGTH
D17B: 9D AD D1 >395            STA   LENTHS,X
D17E: A5 5E    >396            LDA   INDEX
D180: 9D BC D1 >397            STA   VARPT-1,X
D183: A5 5F    >398            LDA   INDEX+1
D185: 9D BD D1 >399            STA   VARPT,X
D188: A5 8F    >400            LDA   DSCLEN
D18A: 9D AC D1 >401            STA   LENTHS-1,X
D18D: A5 8F    >402   DVARTS   LDA   DSCLEN
D18F: 18       >403            CLC
D190: 65 5E    >404            ADC   INDEX
D192: 85 5E    >405            STA   INDEX
D194: 90 02    >406            BCC   *+4
D196: E6 5F    >407            INC   INDEX+1
D198: A6 5F    >408            LDX   INDEX+1
```

```
D19A: A0 00                LDY   #0
D19C: 60                   RTS
                           DUMMY *
D19D: 00 00 00   BTMEL     DS    NUMELS*2
D1AD: 00 00 00   LENTHS    DS    NUMELS*2
D1BD: 00 00 00   VARPT     DS    NUMELS*2
                           DEND
                           ORG
                 CODE1GCF  EQU   *
                 * Here is the Peersoft real origine
                 AROMBA    ORG   $9816-5-$56-$4C-$BB-$26-$B6-$13-$4D5-$1BCC
                 FNDVAR2
                 CGARBAG

                 * All calls to CHRGET fall into this routine
7524: 86 B4      DEBUTGET  STX   XSAV
7526: 84 B5                STY   YSAV
                 * Check return address
7528: BA                   TSX
7529: BD 02 01             LDA   $0102,X    hi byte
752C: 85 C2                STA   OFFSET
752E: BD 01 01             LDA   $0101,X    lo byte
7531: A2 14                LDX   #ADAPFTET-ADAPFBET
7533: DD AF 9B   ]LOOP     CMP   ADAPFBET-1,X
7536: D0 07                BNE   :0
7538: BC C3 9B             LDY   ADAPFTET-1,X
753B: C4 C2                CPY   OFFSET     Test for a match upon
753D: F0 2B                BEQ   OKP1GET    return address: proceed
753F: CA         :0        DEX              ;No match: loop till
7540: D0 F1                BNE   ]LOOP       all values exhaustion
7542: A4 B5                LDY   YSAV
7544: 4C 49 75             JMP   RST101
                 * No address match: exit with a simulation of CHRGET
7547: 86 B4      RST100    STX   XSAV
7549: A2 00      RST101    LDX   #0
754B: E6 B8      LLOOP     INC   TXTPTR
754D: D0 04                BNE   COMRST
754F: E6 B9                INC   TXTPTR+1
7551: A2 00      RST103    LDX   #0
7553: A1 B8      COMRST    LDA   (TXTPTR,X)
7555: C9 20                CMP   #$20
7557: F0 F2                BEQ   LLOOP
7559: A6 B4                LDX   XSAV
755B: C9 3A      COMRSTC   CMP   #´:´
755D: B0 05                BCS   :0
755F: E9 2F                SBC   #$30-1     Because of carry clear
7561: 38                   SEC
7562: E9 D0                SBC   #$D0
7564: 60         :0        RTS
7565: 86 B4      RST102    STX   XSAV
7567: 4C 51 75             JMP   RST103

                 OKP1GET
                 * Tricky way to replace the two bytes at the top of stack
                 * Instead of doing PLA PLA followed by PHA PHA...
756A: 8A                   TXA              ;X into Y
756B: A8                   TAY
```

```
756C: BA                 TSX
756D: B9 D7 9B           LDA   ADPFB-1,Y
7570: 9D 01 01           STA   $0101,X
7573: B9 EB 9B           LDA   ADPFT-1,Y
7576: 9D 02 01           STA   $0102,X
7579: D0 CE              BNE   RST101     Always
757B: 4C 46 79  GNPTRGET JMP   NPTRGET
757E: 86 B4     DEBUTGOT STX   XSAV
7580: BA                 TSX
7581: BD 01 01           LDA   $0101,X
7584: C9 EE              CMP   #VPTRGET-1
7586: D0 C9              BNE   RST103
7588: BD 02 01           LDA   $0102,X
758B: 49 DF              EOR   #>VPTRGET-1 A=0 upon matching address
758D: D0 C2              BNE   RST103
758F: E8                 INX              ;Quick way to pull two bytes
7590: E8                 INX              ; from stack
7591: BD 02 01           LDA   $0102,X
7594: C9 DA              CMP   #>VLET+2
7596: D0 03              BNE   :44
7598: E8                 INX
7599: E8                 INX              ;Carry set at this time
759A: 24                 HEX   24         Skip next byte
759B: 18        :44      CLC
759C: 9A                 TXS
759D: A2 00              LDX   #0
759F: 90 DA              BCC   GNPTRGET
                * The following routine handles the Applesoft
                * variable setting
                * (LET is the optional keyword)
75A1: 20 46 79  RLET     JSR   NPTRGET
75A4: 85 85              STA   FORPNT
75A6: 84 86              STY   FORPNT+1
75A8: A6 BF              LDX   AUXBANK
75AA: F0 25              BEQ   RLET1
75AC: A5 9C              LDA   LOWTR+1
75AE: 48                 PHA
75AF: A5 9B              LDA   LOWTR
75B1: 48                 PHA
75B2: A5 AE              LDA   STRNG2+1
75B4: 48                 PHA
75B5: A5 AD              LDA   STRNG2
75B7: 48                 PHA
75B8: 8A                 TXA
75B9: 48                 PHA
75BA: 20 D1 75           JSR   RLET1
75BD: 68                 PLA
75BE: 85 BF              STA   AUXBANK
75C0: 68                 PLA
75C1: 85 AD              STA   STRNG2
75C3: 68                 PLA
75C4: 85 AE              STA   STRNG2+1
75C6: 68                 PLA
75C7: 85 9B              STA   LOWTR
75C9: 68                 PLA
75CA: 85 9C              STA   LOWTR+1
75CC: A2 05              LDX   #5
```

```
75CE: 4C 20 7D           JMP   ZRTAUX

75D1: A0 00     RLET1    LDY   #0
75D3: B1 B8              LDA   (TXTPTR),Y
75D5: A2 03              LDX   #3           New syntax scheme?
75D7: DD 13 96  ]LOOP    CMP   TOKENS,X
75DA: F0 28              BEQ   :0           yes so handle it
75DC: CA                 DEX
75DD: 10 F8              BPL   ]LOOP
75DF: A9 D0              LDA   #TOKEQUAL
75E1: 20 D0 7D           JSR   NSYNCHR2     Y vaut deja zero si 6502
75E4: A5 12              LDA   INTTYP
75E6: 10 16              BPL   :11
75E8: 48                 PHA
75E9: 20 CE 84           JSR   NFRMNUM
75EC: 20 83 77  ]LOOP    JSR   NROUT
75EF: 68                 PLA
75F0: C9 81              CMP   #$81         Byte subtype?
75F2: D0 0D              BNE   :12
75F4: 20 26 79           JSR   CONV1628
75F7: A5 A1              LDA   FAC+4
75F9: A0 00              LDY   #0
75FB: 91 85              STA   (FORPNT),Y
75FD: 60                 RTS
75FE: 4C 52 DA  :11      JMP   VLET+12
7601: 4C 6B DA  :12      JMP   $DA6B

                * Save selected operation on stack (+,-,*,/)
                :0       MPHX
7604: 8A                 TXA
7605: 48                 PHA
7606: 20 49 75           JSR   RST101       Bump next character
                * Ensure that next char is ´=´ symbol token
7609: A9 D0              LDA   #TOKEQUAL
760B: 20 D0 7D           JSR   NSYNCHR2     no need to reset Y to 0
                * Save variable type on stack
760E: A5 12              LDA   INTTYP       $80 iif integer variable
7610: 48                 PHA
7611: A5 11              LDA   VALTYP       $FF iif string
7613: 48                 PHA
7614: 20 7B DD           JSR   FRMEVL
7617: 68                 PLA
7618: 2A                 ROL                ;Carry set iif var. type string
7619: 20 6D DD           JSR   $DD6D        Check FRMEVL result type accordin
g to C
761C: 68                 PLA                ;Get INTTYP off stack
761D: B0 27              BCS   HNDLESTR     String variable and expression
                * From then on: we´ll handle numeric var. and expr.
761F: 30 52              BMI   HNDLEINT
7621: A4 86     HNDLEREA LDY   FORPNT+1
7623: 68                 PLA
7624: AA                 TAX
7625: A9 EB              LDA   #>$EB27-1
7627: 48                 PHA
7628: A9 26              LDA   #$EB27-1
762A: 48                 PHA
762B: BD 1B 96           LDA   FPROUTST,X
```

```
762E: 48                 PHA
762F: BD 17 96           LDA   FPROUTSB,X
7632: 48                 PHA
7633: A5 85              LDA   FORPNT
7635: 60                 RTS

7636: 4C 27 EB  ]LOOP1   JMP   $EB27      SETFOR
7639: A5 12     NLET2    LDA   INTTYP
763B: 10 F9              BPL   ]LOOP1
763D: 48                 PHA
763E: 30 AC              BMI   ]LOOP      Always

                * Includes module for handling integ. arithmetic
                * and <op>= instructions
                         PUT   PEERINTEGARITH
                * Module handling all integer arithmetic
                * within Peersoft and all op= instructions
                FCOMP    EQU   $EBB2

7640: 4C 76 DD  ]ERR     JMP   GOTMIERR
7643: 4C B2 E5  ]ERR1    JMP   GOSTLERR

                * Handle += instruction for string variables
7646: 68        HNDLESTR PLA              ;Get OP kind off stack
7647: D0 F7              BNE   ]ERR       ;Only ADD operation allowed
7649: A0 00              LDY   #0
764B: B1 A0              LDA   ($A0),Y
764D: F0 63              BEQ   RET1       Do nothing if len(FAC1) is zero
764F: 18                 CLC
7650: 71 85              ADC   (FORPNT),Y
7652: B0 EF              BCS   ]ERR1
7654: 20 DD E3           JSR   STRSPA
7657: A5 85              LDA   FORPNT
7659: A4 86              LDY   FORPNT+1
765B: 20 6C 76           JSR   NMOVINS
765E: A0 02              LDY   #2
7660: B9 9D 00  ]LOOP    LDA   DSCTMP,Y
7663: 91 85              STA   (FORPNT),Y
7665: 88                 DEY
7666: 10 F8              BPL   ]LOOP
7668: A5 A0              LDA   $A0
766A: A4 A1              LDY   $A1
766C: 85 AB     NMOVINS  STA   STRING1
766E: 84 AC              STY   STRING1+1
7670: 4C D4 E5           JMP   MOVINS

7673: 29 07     HNDLEINT AND   #7         Integer subtype in A reg.
7675: C9 02              CMP   #2         Correct if 16bits integer
7677: D0 02              BNE   :0
7679: A9 00              LDA   #0
                * On enclenche NROUT que si 8 ou 16bits
767B: C9 02     :0       CMP   #2
767D: B0 0D              BCS   :1
767F: 48                 PHA
7680: 20 83 77           JSR   NROUT
7683: 68                 PLA
7684: F0 08              BEQ   :2
```

```
               >50   * Ensure correct value for 8bits integer
7686: AA       >51            TAX
7687: 20 26 79 >52            JSR   CONV1628
768A: 8A       >53            TXA
768B: 2C       >58            HEX   2C          Skip next two bytes
768C: E9 01    >59   :1       SBC   #1          Carry already set
768E: 0A       >61   :2       ASL
768F: 0A       >62            ASL
7690: 85 B4    >63            STA   XSAV
7692: 68       >64            PLA               ;Retrieve ope. index in A reg.
7693: 05 B4    >65            ORA   XSAV
7695: 2C E7 9C >66            BIT   WMODE
7698: 10 02    >67            BPL   *+4
769A: 09 08    >68            ORA   #8
769C: AA       >69            TAX               ;Global operation offset into X
769D: BD 35 96 >70   HNDLEIY  LDA   OFFSTT,X
76A0: 48       >71            PHA
76A1: BD 25 96 >72            LDA   OFFSTB,X
76A4: 48       >73            PHA
76A5: A0 01    >74            LDY   #1
76A7: 8A       >75            TXA
76A8: 29 04    >76            AND   #4
76AA: F0 01    >77            BEQ   *+3         Branch iif 16bits int operation
76AC: 88       >78            DEY
76AD: 18       >79            CLC
76AE: B1 85    >80            LDA   (FORPNT),Y
76B0: A0 00    >82            LDY   #0
76B2: 60       >84   RET1     RTS
               >85
76B3: 65 A1    >86   HNDLUIAD ADC   $A1
76B5: AA       >87            TAX               ;Low byte in X reg.
76B6: B1 85    >91            LDA   (FORPNT),Y Y set to zero upon entry
76B8: 65 A0    >93            ADC   $A0
76BA: 90 4E    >94            BCC   HNDLEIC
76BC: 4C D5 E8 >95   ]ERR     JMP   GOOVFERR
76BF: 38       >96   HNDLUIMI SEC
76C0: E5 A1    >97            SBC   $A1
76C2: AA       >98            TAX               ;Low byte in X reg.
76C3: B1 85    >102           LDA   (FORPNT),Y Y set to zero upon entry
76C5: E5 A0    >104           SBC   $A0
76C7: 90 F3    >105           BCC   ]ERR
76C9: B0 3F    >106           BCS   HNDLEIC
76CB: 65 A1    >107  HNDLSIAD ADC   $A1         ADD operation
76CD: AA       >108           TAX
76CE: B1 85    >110           LDA   (FORPNT),Y Y set to zero upon entry
76D0: 65 A0    >114           ADC   $A0
76D2: 70 E8    >115           BVS   ]ERR
76D4: 50 34    >116           BVC   HNDLEIC
76D6: 38       >117  HNDLSIMI SEC
76D7: E5 A1    >118           SBC   $A1
76D9: AA       >119           TAX
76DA: B1 85    >121           LDA   (FORPNT),Y Y set to zero upon entry
76DC: E5 A0    >125           SBC   $A0
76DE: 70 DC    >126           BVS   ]ERR
76E0: 50 28    >127           BVC   HNDLEIC
76E2: 38       >128  HNDLUIDV SEC
76E3: 20 6A 77 >129  HNDLUIMU JSR   LBS49
```

```
76E6: 90 06    >130          BCC   :0
76E8: 20 D0 78 >131          JSR   USDIV
76EB: 4C F1 76 >135          JMP   *+6
76EE: 20 7E 78 >137  :0      JSR   USMUL
76F1: D0 C9    >138          BNE   ]ERR
76F3: F0 11    >139          BEQ   HNDLEIX
76F5: 38       >140  HNDLSIDV SEC
76F6: 20 6A 77 >141  HNDLSIMU JSR  LBS49
76F9: B0 06    >142          BCS   :0
76FB: 20 5F 78 >143          JSR   SMUL
76FE: 4C 04 77 >147          JMP   *+6
7701: 20 A5 78 >149  :0      JSR   SDIV
7704: 70 B6    >150          BVS   ]ERR
7706: A6 C2    >155  HNDLEIX LDX   MPLIER
7708: A5 C3    >157          LDA   MPLIER+1
770A: 91 85    >161  HNDLEIC STA   (FORPNT),Y
770C: 8A       >163          TXA              ;Low byte from result
770D: C8       >165          INY
770E: 91 85    >167          STA   (FORPNT),Y
7710: A9 80    >168  SETITS  LDA   #$80
7712: 85 C7    >169          STA   INTTYPSV
7714: 60       >170          RTS
               >171
7715: 65 A1    >172  HNDLUBAD ADC  $A1
7717: 90 4C    >173          BCC   HNDLEBC
7719: 4C D5 E8 >174  ]ERR    JMP   GOOVFERR
771C: 65 A1    >175  HNDLSBAD ADC  $A1
771E: 70 F9    >176          BVS   ]ERR
               >177  ]ERRS   EQU   *-2
7720: 50 43    >178          BVC   HNDLEBC
7722: 38       >179  HNDLUBMI SEC
7723: E5 A1    >180          SBC   $A1
7725: 90 F2    >181          BCC   ]ERR
7727: B0 3C    >182          BCS   HNDLEBC
7729: 38       >183  HNDLSBMI SEC
772A: E5 A1    >184          SBC   $A1
772C: 70 F0    >185          BVS   ]ERRS
772E: 50 35    >186          BVC   HNDLEBC
7730: 38       >187  HNDLUBMU SEC
7731: 85 C2    >188  HNDLSBMU STA  MPLIER
7733: A5 A1    >189          LDA   $A1
7735: 85 C0    >190          STA   MCAND
7737: 90 09    >191          BCC   :0
7739: 20 E1 77 >192          JSR   USMUL8
773C: D0 DB    >193          BNE   ]ERR
773E: A5 C2    >194          LDA   MPLIER
7740: 70 23    >195          BVS   HNDLEBC    Always (see USMUL8 routine)
7742: 20 C3 77 >196  :0      JSR   SMUL8
7745: 70 D7    >197          BVS   ]ERRS
7747: A5 C2    >198          LDA   MPLIER
7749: 50 1A    >199          BVC   HNDLEBC    Always
               >200
774B: 4C E1 EA >201  ]ERR    JMP   GODVZERR
774E: 38       >202  HNDLUBDV SEC
774F: 85 C2    >203  HNDLSBDV STA  DIVEND
7751: A5 A1    >204          LDA   $A1
7753: F0 F6    >205          BEQ   ]ERR
```

```
7755: 85 C0    >206             STA   DIVSOR
7757: 90 05    >207             BCC   :0
7759: 20 2A 78 >208             JSR   USDIV8
775C: 70 07    >209             BVS   HNDLEBC    Always (see USDIV8 routine)
775E: 20 FC 77 >210  :0         JSR   SDIV8
7761: 70 BB    >211             BVS   ]ERRS
7763: A5 C2    >212             LDA   DIVEND
7765: A0 00    >216  HNDLEBC    LDY   #0
7767: 91 85    >217             STA   (FORPNT),Y
7769: 60       >219  ]RET       RTS
               >220
776A: 08       >221  LBS49      PHP
776B: 85 C2    >222             STA   MPLIER
776D: B1 85    >226             LDA   (FORPNT),Y Y set to zero upon entry
776F: 85 C3    >228             STA   MPLIER+1
7771: A5 A0    >229             LDA   $A0
7773: 85 C1    >230             STA   MCAND+1
7775: A5 A1    >231             LDA   $A1
7777: 85 C0    >232             STA   MCAND
7779: 28       >233             PLP
777A: 60       >234             RTS
               >235
777B: 4C 99 E1 >236  ]ERR       JMP   $E199
               >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.
777E: 08       >245  LBS03      PHP
777F: 20 CE 84 >246             JSR   NFRMNUM
7782: 24       >247             HEX   24
7783: 08       >248  NROUT      PHP
7784: 20 72 EB >249             JSR   $EB72      Arrondit FAC
7787: 28       >250             PLP
7788: A5 9D    >251  NEWAYINT   LDA   FAC
778A: 2C E7 9C >252             BIT   WMODE
778D: 30 0A    >253             BMI   :1
778F: C9 90    >254             CMP   #$90
7791: 90 1F    >258             BCC   :LOOP
7793: 20 5A 8E >260             JSR   GN32768
7796: 4C 16 E1 >261             JMP   $E116
               >262  * Unsigned mode
7799: 24 A2    >263  :1         BIT   FACSIGN
779B: B0 1A    >264             BCS   :3
779D: 30 DC    >265             BMI   ]ERR
779F: C9 91    >266  :2         CMP   #$91
77A1: 90 0F    >270             BCC   :LOOP
77A3: 20 5F 8E >272             JSR   GP32768
77A6: 20 B2 EB >273             JSR   FCOMP
77A9: A8       >274             TAY
77AA: 30 06    >278             BMI   :LOOP      A = -1 so FAC < 32768
77AC: 20 64 8E >280             JSR   GN65536
77AF: 20 BE E7 >281             JSR   FADD
77B2: 20 F2 EB >285  :LOOP      JSR   QINT
77B5: 18       >286             CLC
77B6: 60       >287             RTS
77B7: 10 E6    >289  :3         BPL   :2
77B9: 20 D0 EE >290             JSR   NEGOP
77BC: A5 9D    >291             LDA   FAC
```

```
77BE: 20 9F 77          JSR   :2
77C1: 38                SEC
77C2: 60                RTS

               * Signed 8bits multiplication: result in 8bits
               * with possible overflow exception
               * MCAND and MPLIER set upon entry
               * Result in MPLIER
               * Credits: Randy Hyde
77C3: A5 C0    SMUL8    LDA   MCAND
77C5: 45 C2             EOR   MPLIER
77C7: 48                PHA              ;Bit N set if signs differ
77C8: 20 45 78          JSR   ZPRT8
77CB: 20 E1 77          JSR   USMUL8
77CE: 68                PLA
77CF: AA                TAX
77D0: 98                TYA
77D1: D0 0D             BNE   :0
77D3: A5 C2             LDA   MPLIER
77D5: 30 09             BMI   :0
77D7: 8A                TXA
77D8: 10 05             BPL   :1
77DA: A2 C2             LDX   #MPLIER
77DC: 20 54 78          JSR   NEG8
77DF: B8       :1       CLV
77E0: 60       :0       RTS

77E1: A0 08    USMUL8   LDY   #8
77E3: A5 C2    ]LOOP    LDA   MPLIER     Get lsb of MPLIER
77E5: 4A                LSR              ; into C
77E6: 90 07             BCC   :4
77E8: 18                CLC
77E9: A5 BE             LDA   PARTIAL
77EB: 65 C0             ADC   MCAND
77ED: 85 BE             STA   PARTIAL
               * Shift result into MPLIER
77EF: 66 BE    :4       ROR   PARTIAL
77F1: 66 C2             ROR   MPLIER
77F3: 88                DEY              ;All MPLIER 8 bits
77F4: D0 ED             BNE   ]LOOP       have been processed?
77F6: 2C 69 77          BIT   ]RET       Bit V set..
77F9: A4 BE             LDY   PARTIAL
77FB: 60       ]RET     RTS

               * Signed 8bits integer divide routine
               * with possible overflow and divide by zero exceptions
               * DIVEND and DIVSOR set upon entry
               * Result in DIVEND
               * Credits: Randy Hyde
77FC: A5 C0    SDIV8    LDA   DIVSOR
77FE: 49 80             EOR   #$80
7800: D0 0D             BNE   :1
               * On traite le cas ou le diviseur est -128
               * Dans ce cas la si DIVEND vaut aussi -128, alors
               * retourne 1 sinon 0
7802: A8                TAY
7803: AA                TAX              ;X forced to zero
```

```
7804: A5 C2            LDA   DIVEND
7806: C9 80            CMP   #$80
7808: D0 01            BNE   *+3
780A: E8               INX
780B: 86 C2            STX   DIVEND
780D: D0 EC            BNE   ]RET
780F: A5 C0    :1      LDA   DIVSOR
7811: 45 C2    :2      EOR   DIVEND
7813: 48               PHA               ;Sign bit on stack
7814: 20 45 78         JSR   ZPRT8       ;Absolute value for operands
7817: 20 2A 78         JSR   USDIV8
781A: C9 FF            CMP   #$FF
781C: F0 0A            BEQ   :3          Keep V set and exit
781E: 68               PLA               ;Get back sign
781F: 10 05            BPL   *+7         No need to get result opposite
7821: A2 C2            LDX   #DIVEND
7823: 20 54 78         JSR   NEG8
               * Exit with V clear
7826: B8               CLV
7827: 60               RTS
7828: 68       :3      PLA
7829: 60       ]RET    RTS

782A: A0 08    USDIV8  LDY   #8
782C: 06 C2    ]LOOP   ASL   DIVEND
782E: 26 BE            ROL   PARTIAL
7830: 38               SEC
7831: A5 BE            LDA   PARTIAL
7833: E5 C0            SBC   DIVSOR
7835: AA               TAX
7836: 90 04            BCC   :3
7838: 86 BE            STX   PARTIAL
783A: E6 C2            INC   DIVEND
783C: 88       :3      DEY
783D: D0 ED            BNE   ]LOOP
783F: 2C 29 78         BIT   ]RET        V set by default
7842: A5 C2            LDA   DIVEND
7844: 60               RTS

7845: A0 00    ZPRT8   LDY   #0
7847: 84 BE            STY   PARTIAL
7849: A2 C0            LDX   #MCAND
784B: 20 50 78         JSR   ABSOL8
784E: A2 C2            LDX   #MPLIER
7850: B5 00    ABSOL8  LDA   0,X
7852: 10 D5            BPL   ]RET
7854: 98       NEG8    TYA
7855: 38               SEC
7856: F5 00            SBC   0,X
7858: 95 00            STA   0,X
785A: 60       ]RET    RTS

               * Signed 16bits multiplication: result in 16bits
               * with possible overflow exception
               * MCAND and MPLIER set upon entry
               * Result in MPLIER
               * Credits: Randy Hyde
```

```
785B: 2C 5A 78 >409  ]LOOP    BIT   ]RET
785E: 60       >410           RTS
785F: A5 C1    >411  SMUL     LDA   MCAND+1
7861: 45 C3    >412           EOR   MPLIER+1
7863: 48       >413           PHA              ;BitN set if signs differ
7864: 20 06 79 >414           JSR   ZEROPRT    Get absolute values of operands
7867: 20 7E 78 >415           JSR   USMUL
786A: A8       >416           TAY
786B: 68       >417           PLA
786C: AA       >417           TAX
786D: 98       >418           TYA
786E: D0 EB    >419           BNE   ]LOOP
7870: A5 C3    >420           LDA   MPLIER+1
7872: 30 E7    >421           BMI   ]LOOP
7874: 8A       >422           TXA
7875: 10 05    >423           BPL   :8
7877: A2 C2    >424           LDX   #MPLIER
7879: 20 17 79 >425           JSR   NEGATE
787C: B8       >426  :8       CLV              ;reset bit V to zero
787D: 60       >427  ]RET     RTS
               >428
787E: A0 10    >429  USMUL    LDY   #16
7880: A5 C2    >430  ]LOOP    LDA   MPLIER     Get lsb of MPLIER
7882: 4A       >431           LSR              ; into C
7883: 90 0D    >432           BCC   :4
7885: 18       >433           CLC
7886: A5 BE    >434           LDA   PARTIAL
7888: 65 C0    >435           ADC   MCAND
788A: 85 BE    >436           STA   PARTIAL
788C: A5 BF    >437           LDA   PARTIAL+1
788E: 65 C1    >438           ADC   MCAND+1
7890: 85 BF    >439           STA   PARTIAL+1
               >440  * Shift result into MPLIER
7892: 66 BF    >441  :4       ROR   PARTIAL+1
7894: 66 BE    >442           ROR   PARTIAL
7896: 66 C3    >443           ROR   MPLIER+1
7898: 66 C2    >444           ROR   MPLIER
789A: 88       >445           DEY              ;All MPLIER 16 bits
789B: D0 E3    >446           BNE   ]LOOP       have been processed?
789D: A5 BE    >447           LDA   PARTIAL
789F: 05 BF    >448           ORA   PARTIAL+1
78A1: 60       >449  ]RET     RTS
               >450
78A2: 4C E1 EA >451  DVZERROR JMP   GODVZERR
               >452  * Signed 16bits integer divide routine
78A5: A5 C1    >453  SDIV     LDA   DIVSOR+1
78A7: 05 C0    >454           ORA   DIVSOR
78A9: F0 F7    >455           BEQ   DVZERROR
78AB: A5 C1    >456           LDA   DIVSOR+1
78AD: C9 80    >457           CMP   #>$8000
78AF: D0 19    >458           BNE   :2
78B1: A5 C0    >459           LDA   DIVSOR
78B3: D0 13    >460           BNE   :1
               >461  * On traite le cas ou le diviseur est -32768
               >462  * Dans ce cas la si DIVEND vaut aussi -32768, alors
               >463  * retourne 1 sinon 0
78B5: A8       >464           TAY
```

```
78B6: AA                    TAX                   ;X forced to zero
78B7: C5 C2                 CMP   DIVEND
78B9: D0 07                 BNE   :0
78BB: A5 C3                 LDA   DIVEND+1
78BD: C9 80                 CMP   #>$8000
78BF: D0 01                 BNE   :0
78C1: E8                    INX
78C2: 86 C2       :0        STX   DIVEND
78C4: 84 C3                 STY   DIVEND+1
78C6: D0 3A                 BNE   NRET        Always
78C8: A5 C1       :1        LDA   DIVSOR+1
78CA: 45 C3       :2        EOR   DIVEND+1
78CC: 48                    PHA                   ;Sign bit on stack
78CD: 20 06 79              JSR   ZEROPRT     ;Absolute value for operands
78D0: A0 10       USDIV     LDY   #16
78D2: 06 C2       ]LOOP     ASL   DIVEND
78D4: 26 C3                 ROL   DIVEND+1
78D6: 26 BE                 ROL   PARTIAL
78D8: 26 BF                 ROL   PARTIAL+1
78DA: 38                    SEC
78DB: A5 BE                 LDA   PARTIAL
78DD: E5 C0                 SBC   DIVSOR
78DF: AA                    TAX
78E0: A5 BF                 LDA   PARTIAL+1
78E2: E5 C1                 SBC   DIVSOR+1
78E4: 90 06                 BCC   :3
78E6: 86 BE                 STX   PARTIAL
78E8: 85 BF                 STA   PARTIAL+1
78EA: E6 C2                 INC   DIVEND
78EC: 88          :3        DEY
78ED: D0 E3                 BNE   ]LOOP
78EF: 2C 05 79              BIT   ARET+1      V set by default
78F2: A5 C2                 LDA   DIVEND
78F4: 25 C3                 AND   DIVEND+1
78F6: C9 FF                 CMP   #$FF
78F8: F0 0A                 BEQ   ARET        Keep V set and exit
78FA: 68                    PLA                   ;Get back sign
78FB: 10 05                 BPL   NRET        No need to get result opposite
78FD: A2 C2                 LDX   #DIVEND
78FF: 20 17 79              JSR   NEGATE
                  * Exit with V clear
7902: B8          NRET      CLV
7903: 70                    HEX   70          Skip next byte
7904: 68          ARET      PLA
7905: 60          ]RET      RTS

                  * Zero partial and fall into ABSOPND
7906: A0 00       ZEROPRT   LDY   #0
7908: 84 BE                 STY   PARTIAL
790A: 84 BF                 STY   PARTIAL+1
790C: A2 C0                 LDX   #MCAND
790E: 20 13 79              JSR   ABSOLUTE
7911: A2 C2                 LDX   #MPLIER     ;Fall into ABSOLUTE
                  * Compute absolute value of integer pointed to by X
                  * in ZP
7913: B5 01       ABSOLUTE  LDA   1,X
7915: 10 EE                 BPL   ]RET        No need
```

```
7917: 38          NEGATE   SEC
7918: 98                   TYA                  ;Y set to 0 upon entry
7919: F5 00                SBC   0,X
791B: 95 00                STA   0,X
791D: 98                   TYA
791E: F5 01                SBC   1,X
7920: 95 01                STA   1,X
7922: 60          ]RET     RTS

                  * Conversion from 16bits to 8bits with provision for
                  * ILLEGAL QUANTITY..
7923: 4C 99 E1    ]ERR     JMP   GOIQERR
7926: A5 A0       CONV1628 LDA   FAC+3      High byte
7928: 2C E7 9C             BIT   WMODE
792B: 30 0B                BMI   :0
792D: A8                   TAY
792E: C8                   INY
792F: C0 02                CPY   #2         Must be either -1 or 0
7931: B0 F0                BCS   ]ERR        in unsigned mode
7933: 45 A1                EOR   FAC+4      b7 of low byte should be
7935: 30 EC                BMI   ]ERR       set accordingly.
7937: 60                   RTS
7938: D0 E9       :0       BNE   ]ERR       Must be zero if unsigned mode
793A: 60                   RTS
793B: 4C 99 E1             JMP   GOIQERR
                  * New processing for variable lookup
                           PUT   PEERNPTRGET
                  MKNV     EQU   $E09C      Make new variable (ROM routine)
                  SETVYA   EQU   $E0DE      Set LOWTR and Y,A if var. found

793E: A9 40       NGETARPT LDA   #$40       $40: only look for arrays
7940: 85 14                STA   SUBFLG
                  * This routine is the new PTRGET routine from PEERSOFT
                  NPTRGTX
7942: A2 00                LDX   #0
7944: 86 10                STX   DIMFLG
                  NPTRGET
                  * Upon exit from the above routine, the X reg will
                  * contain the value X had upon call to CHRGOT (here zero)
7946: 20 53 75             JSR   COMRST
                  * First variable name character must be alphabetic
7949: 20 C8 7D             JSR   MISLETC

794C: A2 00       NPTRGET1 LDX   #0
794E: 86 11                STX   VALTYP
7950: 86 12                STX   INTTYP
7952: 86 BF                STX   AUXBANK
7954: 86 82                STX   VARNAM+1   default zero for 2nd name char.
7956: 85 81                STA   VARNAM
7958: 20 47 75             JSR   RST100
795B: 90 05                BCC   GTLT       Branch if numeric digit
795D: 20 7D E0             JSR   ISLETC
7960: 90 1A                BCC   EXPLIC?    Branch if not alpha character
7962: AA          GTLT     TAX              ;2nd character in X
7963: 86 82                STX   VARNAM+1    and into VARNAM+1
                  * Skip subsequent alphanumeric characters
7965: 20 47 75    ]LOOP    JSR   RST100
```

```
7968: 90 FB    >42              BCC   ]LOOP      branch if numeric
796A: 20 7D E0 >43              JSR   ISLETC
796D: B0 F6    >44              BCS   ]LOOP      branch if alphabetic
796F: 90 0B    >45              BCC   EXPLIC?    Always
7971: 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.
7974: 20 97 81 >49   SCDCH2     JSR   DECTPTR
7977: A6 81    >50              LDX   VARNAM
7979: BD 55 9B >51              LDA   TYPLET-´A´,X
               >52   * Fall into implicit (2nd pass to EXPLIC?)
797C: 20 8C 81 >53   EXPLIC?    JSR   XFROMMOT   Get index from character
               >54   * No explicit type specifier found, so try implicit
               >55   * type specifier (cannot fail)
797F: D0 F3    >56              BNE   SCDCH2     Branch if no type spec. found
7981: BD 8A 9B >58              LDA   TVTVAL,X
7984: 85 11    >59              STA   VALTYP
7986: BD 86 9B >60              LDA   TITVAL,X
7989: 85 12    >61              STA   INTTYP
798B: BD 8E 9B >62              LDA   TVNORA,X
798E: 05 81    >63              ORA   VARNAM
7990: 85 81    >63              STA   VARNAM
7992: BD 92 9B >64              LDA   TVN1ORA,X
7995: 05 82    >65              ORA   VARNAM+1
7997: 85 82    >65              STA   VARNAM+1
7999: E0 02    >66              CPX   #2         FP or string
799B: 90 04    >67              BCC   :6
799D: A5 14    >68              LDA   SUBFLG
799F: 30 D0    >69              BMI   BADNAM
79A1: 20 47 75 >70   :6         JSR   RST100     Get next character
79A4: 38       >71              SEC
79A5: 05 14    >72              ORA   SUBFLG
79A7: E9 28    >73              SBC   #´(´
79A9: D0 03    >74              BNE   :8
79AB: 4C 79 7A >75   :7         JMP   NARRAY
79AE: 24 14    >76   :8         BIT   SUBFLG
79B0: 30 02    >77              BMI   :9
79B2: 70 F7    >78              BVS   :7
               >79   :9         DO    KOPT-K6502
79B4: A9 00    >82              LDA   #0
79B6: 85 14    >83              STA   SUBFLG
79B8: AE 83 99 >85   NPTRGL90 LDX     SNCCH
79BB: F0 05    >86              BEQ   :90
79BD: 20 2A 7A >87              JSR   SLKCACH
79C0: D0 65    >88              BNE   NAMFOUND   Found cache entry if Zbit clear
               >89   :90        DO    KOPT16
79C2: A6 69    >95              LDX   VARTAB
79C4: A5 6A    >96              LDA   VARTAB+1
79C6: 85 9C    >101  ]LOOP      STA   LOWTR+1
79C8: 86 9B    >102  ]LOOP1     STX   LOWTR
79CA: E4 6B    >107             CPX   ARYTAB
79CC: E5 6C    >108             SBC   ARYTAB+1
79CE: B0 28    >110             BCS   NAMNTFND
79D0: A0 00    >112             LDY   #0
79D2: B1 9B    >113             LDA   (LOWTR),Y
79D4: 45 81    >117             EOR   VARNAM
79D6: D0 13    >118             BNE   :1
```

```
79D8: C8                  INY
79D9: B1 9B               LDA   (LOWTR),Y
79DB: 45 82               EOR   VARNAM+1
79DD: D0 0C               BNE   :1
79DF: A5 12               LDA   INTTYP
79E1: 10 44               BPL   NAMFOUND
79E3: A0 06               LDY   #6
79E5: B1 9B               LDA   (LOWTR),Y
79E7: 45 12               EOR   INTTYP
79E9: F0 3C               BEQ   NAMFOUND
                * Name not yet found: look for next variable in memory
79EB: A5 9B     :1        LDA   LOWTR
79ED: 69 07               ADC   #7          Carry already clear
79EF: AA                  TAX
79F0: A5 9C               LDA   LOWTR+1
79F2: 90 D4               BCC   ]LOOP1
79F4: 69 00               ADC   #0
79F6: 90 CE               BCC   ]LOOP       Always

79F8: BA        NAMNTFND  TSX
79F9: BD 01 01            LDA   STACK+1,X
79FC: C9 31               CMP   #RFFVL
79FE: D0 0A               BNE   :0
7A00: BD 02 01            LDA   STACK+2,X
7A03: C9 85               CMP   #>RFFVL
7A05: D0 03               BNE   :0
7A07: 4C 95 E0            JMP   $E095       Return 0 constant
                * Make new variable
7A0A: 18        :0        CLC
7A0B: A5 6D               LDA   STREND
7A0D: A4 6E               LDY   STREND+1
7A0F: 69 07               ADC   #7
7A11: 90 01               BCC   *+3
7A13: C8                  INY
7A14: 20 5B 7A            JSR   NREASON
7A17: 20 9C E0            JSR   MKNV        Make new variable (ROM routine)
7A1A: A5 12               LDA   INTTYP      FP or string?
7A1C: 10 06               BPL   :1          Yes
7A1E: A0 06               LDY   #6
7A20: 91 9B               STA   (LOWTR),Y
7A22: A4 84               LDY   VARPNT+1
7A24: A5 83     :1        LDA   VARPNT
7A26: 60                  RTS

                NAMFOUND
7A27: 4C DE E0            JMP   SETVYA

                * Cache mechanism for simple variables
                SCTR      EQU   LOWTR
7A2A: A4 82     SLKCACH   LDY   VARNAM+1
7A2C: A5 81               LDA   VARNAM
7A2E: 86 9B               STX   SCTR
7A30: A2 00               LDX   #0
7A32: DD 84 99  ]LOOP     CMP   SVN,X
7A35: D0 0F               BNE   :0
7A37: 98                  TYA
7A38: DD 88 99            CMP   SVNP1,X
```

```
7A3B: D0 07            BNE   :2
7A3D: A5 12            LDA   INTTYP
7A3F: DD 8C 99         CMP   SIT,X
7A42: F0 08            BEQ   :1
7A44: A5 81    :2      LDA   VARNAM
7A46: E8       :0      INX
7A47: E4 9B            CPX   SCTR
7A49: D0 E7            BNE   ]LOOP
7A4B: 60               RTS

7A4C: BD 90 99 :1      LDA   SLTR,X
7A4F: 85 9B            STA   LOWTR
7A51: BD 94 99         LDA   SLTRP1,X
7A54: 85 9C            STA   LOWTR+1
7A56: 8A               TXA
7A57: 60               RTS

7A58: 4C 10 D4 ]ERR    JMP   MEMERR
               * Pour le 65(C)02, Y,A nouveau STREND
7A5B: C4 70    NREASON CPY   FRETOP+1
7A5D: 90 19            BCC   :0
7A5F: D0 04            BNE   :1
7A61: C5 6F            CMP   FRETOP
7A63: 90 13            BCC   :0
7A65: 48       :1      PHA
7A66: 98               TYA
7A67: 48               PHA
7A68: 20 D5 9C         JSR   VGARBAG
7A6B: 68               PLA
7A6C: A8               TAY
7A6D: 68               PLA
7A6E: C4 70            CPY   FRETOP+1
7A70: 90 06            BCC   :0
7A72: D0 E4            BNE   ]ERR
7A74: C5 6F            CMP   FRETOP
7A76: B0 E0            BCS   ]ERR
7A78: 60       :0      RTS
               * New processing for array processing
                       PUT   PEERNARRAY
               * Module handling the new array processing strategy
               ERR_BSCR =    $6B
               ERR_RDIM =    $78
               ERR_SYNT =    $10

               NUMDIM  EQU   $0F
               RESULT  EQU   $62
               STACK   EQU   $0100
               SUBERR  EQU   $E196      Raise a BAD SUBSCRIPT error
               MEMERR  EQU   $D410
               REASON  EQU   $D3E3
               GETARY  EQU   $E0ED
               GETARY2 EQU   $E0EF      Compute addr. of 1st elm value
               QINT    EQU   $EBF2

               * MULTPLSS multiplies (STRNG2) by ((LOWTR),Y) leaving
               * result in A,X. Hi byte also in Y
               MULTPLSS EQU  $E2AD
               MULTPLY1 EQU   $E2B6

7A79: A5 14    NARRAY   LDA   SUBFLG
7A7B: D0 4D             BNE   NARRGL91
7A7D: A5 10             LDA   DIMFLG
7A7F: 48                PHA
7A80: A5 12             LDA   INTTYP
7A82: 48                PHA
7A83: A5 11             LDA   VALTYP
7A85: 48                PHA
7A86: A0 00             LDY   #0
               ]LOOP    MPHY
7A88: 98                TYA
7A89: 48                PHA
7A8A: A5 82             LDA   VARNAM+1
7A8C: 48                PHA
7A8D: A5 81             LDA   VARNAM
7A8F: 48                PHA
7A90: 20 CE 7C          JSR   NMAKINT
7A93: 68                PLA
7A94: 85 81             STA   VARNAM     Restore array name
7A96: 68                PLA
7A97: 85 82             STA   VARNAM+1
7A99: 68                PLA
7A9A: A8                TAY
               * Code below would transform the stack area
               * from
               *  DIMFLG
               *  INTTYP
               *  VALTYP
               * SPtr ->
               * to
               *  (FAC+3)
               *  (FAC+4)
               *  DIMFLG
               *  INTTYP
               *  VALTYP
               * SPtr ->
7A9B: BA                TSX
7A9C: BD 02 01          LDA   STACK+2,X  Get INTTYP
7A9F: 48                PHA
7AA0: BD 01 01          LDA   STACK+1,X  Get VALTYP
7AA3: 48                PHA
7AA4: BD 03 01          LDA   STACK+3,X  Get DIMFLG
7AA7: 9D 01 01          STA   STACK+1,X  In place of original VALTYP
7AAA: A5 A0             LDA   FAC+3
7AAC: 9D 03 01          STA   STACK+3,X  In place of original DIMFLG
7AAF: A5 A1             LDA   FAC+4
7AB1: 9D 02 01          STA   STACK+2,X  In place of original INTTYP
               * Now the stack frame looks like
               *  FAC+4
               *  FAC+3
               *  DIMFLG
               *  INTTYP
               *  VALTYP
               * SPtr ->
7AB4: C8                INY
```

```
7AB5: 20 65 75 >118           JSR   RST102
7AB8: C9 2C    >119           CMP   #´,´
7ABA: F0 CC    >120           BEQ   ]LOOP
7ABC: 84 0F    >121           STY   NUMDIM
7ABE: 20 46 86 >122           JSR   NCHKCLS
7AC1: 68       >123           PLA
7AC2: 85 11    >124           STA   VALTYP
7AC4: 68       >125           PLA
7AC5: 85 12    >126           STA   INTTYP
7AC7: 68       >127           PLA
7AC8: 85 10    >128           STA   DIMFLG
               >129
               >130
7ACA: AE 98 99 >131  NARRGL91 LDX   ANCCH
7ACD: F0 05    >132           BEQ   :20
7ACF: 20 F2 7C >133           JSR   ALKCACH
7AD2: D0 3E    >134           BNE   USEOLDAR
7AD4: A5 6C    >145  :20      LDA   ARYTAB+1
7AD6: A6 6B    >146           LDX   ARYTAB
7AD8: 86 9B    >147  ]LOOP    STX   LOWTR
7ADA: 85 9C    >148           STA   LOWTR+1
7ADC: E4 6D    >149           CPX   STREND
7ADE: E5 6E    >150           SBC   STREND+1
7AE0: B0 2D    >152           BCS   GNARRAY
7AE2: A0 00    >156           LDY   #0
7AE4: B1 9B    >157           LDA   (LOWTR),Y
7AE6: 45 81    >159           EOR   VARNAM
7AE8: D0 17    >160           BNE   :5
7AEA: C8       >169           INY
7AEB: B1 9B    >171           LDA   (LOWTR),Y
7AED: 45 82    >172           EOR   VARNAM+1
7AEF: D0 10    >173           BNE   :5
7AF1: A6 12    >175           LDX   INTTYP
7AF3: 10 1D    >176           BPL   USEOLDAR   If FP or string array
7AF5: 20 EA 7C >177           JSR   CNVT1
7AF8: A0 04    >178           LDY   #4
7AFA: 51 9B    >179           EOR   (LOWTR),Y
7AFC: 29 C0    >180           AND   #$C0       only test b6 and b7
7AFE: F0 12    >181           BEQ   USEOLDAR
7B00: 18       >189           CLC
               >190  :5
7B01: A0 02    >192           LDY   #2
7B03: B1 9B    >194           LDA   (LOWTR),Y
7B05: 65 9B    >195           ADC   LOWTR
7B07: AA       >197           TAX
7B08: C8       >198           INY
7B09: B1 9B    >199           LDA   (LOWTR),Y
7B0B: 65 9C    >200           ADC   LOWTR+1
7B0D: 90 C9    >202           BCC   ]LOOP      Always
               >203
               >204  GNARRAY
7B0F: 4C 80 7B >209           JMP   MKNARRAY
               >210
7B12: A5 10    >211  USEOLDAR LDA   DIMFLG     Called from the DIM stmt.?
7B14: D0 65    >212           BNE   RDIMERR
7B16: A5 14    >213           LDA   SUBFLG     Subscripts given?
7B18: F0 02    >214           BEQ   :1         Yes
```

```
7B1A: 38                 SEC              ;No: just return "array found"
7B1B: 60                 RTS
                * Set ARYPNT to 1st elm. base addr
7B1C: A0 04     :1       LDY   #4
7B1E: B1 9B              LDA   (LOWTR),Y
7B20: 29 07              AND   #7
7B22: AA                 TAX
7B23: 20 EF E0           JSR   GETARY2
7B26: A5 0F              LDA   NUMDIM
7B28: C9 01              CMP   #1
7B2A: F0 07              BEQ   :3
7B2C: E4 0F              CPX   NUMDIM
7B2E: D0 45              BNE   SUBSERR
7B30: 4C 67 7C           JMP   NFAEP

                * Il s´agit de traiter de la reference unidimensionnelle
                * sur un tableau potentiellement multi-dimensions
                * Multiplier l´indice tire dans la pile par le elm size
                * et comparer par rapport a l´offset du tableau (corrige
                * de la taille du header).
7B33: 68        :3       PLA
7B34: 85 AD              STA   STRNG2
7B36: 68                 PLA
7B37: 85 AE              STA   STRNG2+1
7B39: 20 BB 7C           JSR   KWELMSIZ
7B3C: 86 64              STX   RESULT+2
7B3E: A9 00              LDA   #0
7B40: 20 B6 E2           JSR   MULTPLY1
7B43: 86 AD              STX   STRNG2
7B45: 84 AE              STY   STRNG2+1
7B47: A0 04              LDY   #4
7B49: B1 9B              LDA   (LOWTR),Y  # of dimensions
7B4B: 29 07              AND   #7         Mask out new Peersoft bits
7B4D: 0A                 ASL              ;2 bytes per dimension
7B4E: 69 05              ADC   #5         Carry clear
                * Add this to element offset from base address
7B50: 65 AD              ADC   STRNG2
7B52: A6 AE              LDX   STRNG2+1
7B54: 90 01              BCC   :4
7B56: E8                 INX
7B57: A0 02     :4       LDY   #2
7B59: D1 9B              CMP   (LOWTR),Y
7B5B: 85 83              STA   VARPNT
7B5D: C8                 INY
7B5E: 8A                 TXA
7B5F: F1 9B              SBC   (LOWTR),Y
7B61: B0 12              BCS   SUBSERR
7B63: 86 84              STX   VARPNT+1
7B65: A5 9B              LDA   LOWTR
7B67: 65 83              ADC   VARPNT
7B69: 85 83              STA   VARPNT
7B6B: A5 84              LDA   VARPNT+1
7B6D: 65 9C              ADC   LOWTR+1
7B6F: 85 84              STA   VARPNT+1
7B71: A8                 TAY
7B72: A5 83              LDA   VARPNT
7B74: 60                 RTS
```

```
                  >272
7B75: A2 6B       >273  SUBSERR  LDX    #ERR_BSCR
7B77: 2C          >274           HEX    2C          Skip next two bytes
7B78: A2 10       >275  SNERR    LDX    #ERR_SYNT
7B7A: 2C          >276           HEX    2C
7B7B: A2 78       >277  RDIMERR  LDX    #ERR_RDIM
7B7D: 4C 12 D4 >278           JMP    $D412
                  >279
7B80: A5 14       >280  MKNARRAY LDA    SUBFLG
7B82: F0 03       >281           BEQ    :0
7B84: 4C DC E1 >282           JMP    $E1DC       Raise OUT OF DATA error
7B87: 20 ED E0 >283  :0       JSR    GETARY      Address 1st elm in ARYPNT&Y,A
7B8A: 20 BB 7C >284           JSR    KWELMSIZ
7B8D: 86 AD       >285           STX    STRNG2
7B8F: A2 00       >289           LDX    #0
7B91: 86 BF       >290           STX    AUXBANK
7B93: A5 10       >292           LDA    DIMFLG
7B95: F0 03       >293           BEQ    :1
7B97: 20 48 7D >294           JSR    ISAUXMEM
7B9A: A5 94       >302  :1       LDA    ARYPNT
7B9C: 20 5B 7A >304           JSR    NREASON     Ensure enough memory for array
7B9F: A5 81       >305           LDA    VARNAM
7BA1: A0 00       >311           LDY    #0
7BA3: 84 AE       >312           STY    STRNG2+1
7BA5: 91 9B       >313           STA    (LOWTR),Y
7BA7: C8          >314           INY
7BA8: A5 82       >316           LDA    VARNAM+1
7BAA: 91 9B       >317           STA    (LOWTR),Y
7BAC: A0 04       >318           LDY    #4
7BAE: A5 12       >319           LDA    INTTYP
7BB0: F0 04       >320           BEQ    :2
7BB2: AA          >321           TAX
7BB3: 20 EA 7C >322           JSR    CNVT1
7BB6: 05 0F       >323  :2       ORA    NUMDIM
7BB8: A6 BF       >324           LDX    AUXBANK
7BBA: 85 BF       >325           STA    AUXBANK
7BBC: 8A          >326           TXA
7BBD: 0A          >327           ASL
7BBE: 0A          >328           ASL
7BBF: 0A          >329           ASL
7BC0: 05 BF       >330           ORA    AUXBANK
7BC2: 86 BF       >331           STX    AUXBANK
7BC4: 91 9B       >332           STA    (LOWTR),Y
7BC6: A9 00       >333  ]LOOP    LDA    #0          Hi byte of default dim
7BC8: A2 0B       >334           LDX    #11         Lo byte of default dim
7BCA: 24 10       >335           BIT    DIMFLG
7BCC: 50 08       >336           BVC    :5
7BCE: 68          >344           PLA
7BCF: 18          >345           CLC
7BD0: 69 01       >346           ADC    #1
7BD2: AA          >347           TAX
7BD3: 68          >348           PLA
7BD4: 69 00       >349           ADC    #0
7BD6: C8          >351  :5       INY                ;Add this dimension to descr.
7BD7: 91 9B       >352           STA    (LOWTR),Y
7BD9: C8          >353           INY
7BDA: 8A          >354           TXA
```

```
7BDB: 91 9B    >355          STA   (LOWTR),Y
               >356  * Multiply this dimension by running size
               >357  * ((LOWTR),Y) * (STRNG2) --> A,X
7BDD: 20 AD E2 >358          JSR   MULTPLSS
7BE0: 86 AD    >359          STX   STRNG2
7BE2: 85 AE    >360          STA   STRNG2+1
7BE4: A4 5E    >361          LDY   INDEX
7BE6: C6 0F    >362          DEC   NUMDIM
7BE8: D0 DC    >363          BNE   ]LOOP
               >364
7BEA: A4 BF    >365          LDY   AUXBANK
7BEC: F0 0F    >366          BEQ   :7
7BEE: A2 01    >367          LDX   #1          Ensure enough room in aux mem.
7BF0: 20 20 7D >368          JSR   ZRTAUX
7BF3: E0 01    >369          CPX   #1          X set to 0 iif enough room
7BF5: B0 6D    >370          BCS   GME         otherwise -> MEMORY ERROR
7BF7: A5 94    >371          LDA   ARYPNT
7BF9: A4 95    >372          LDY   ARYPNT+1
7BFB: 90 0F    >373          BCC   :6          Always
               >374  * Now A,X has the total # of bytes of array elements
7BFD: 65 95    >375  :7      ADC   ARYPNT+1    Compute address of end of array
7BFF: B0 63    >376          BCS   GME         Too large: error
7C01: 85 95    >377          STA   ARYPNT+1
7C03: A8       >378          TAY
7C04: 8A       >379          TXA
7C05: 65 94    >380          ADC   ARYPNT
7C07: 90 03    >381          BCC   :6
7C09: C8       >382          INY
7C0A: F0 58    >383          BEQ   GME         Too large: error
7C0C: 20 E3 D3 >384  :6      JSR   REASON      Ensure enough room up to Y,A
7C0F: 85 6D    >385          STA   STREND
7C11: 84 6E    >386          STY   STREND+1
7C13: 38       >387          SEC
7C14: E5 9B    >388          SBC   LOWTR
7C16: A0 02    >389          LDY   #2
7C18: 91 9B    >390          STA   (LOWTR),Y
7C1A: C8       >391          INY
7C1B: A5 6E    >392          LDA   STREND+1
7C1D: E5 9C    >393          SBC   LOWTR+1
7C1F: 91 9B    >394          STA   (LOWTR),Y
7C21: A5 BF    >395          LDA   AUXBANK
7C23: F0 27    >396          BEQ   :9
7C25: 08       >397          PHP
7C26: 78       >398          SEI
7C27: 8D 09 C0 >399          STA   ALTZP
7C2A: A5 6D    >400          LDA   STREND
7C2C: A6 6E    >401          LDX   STREND+1
7C2E: 8D 08 C0 >402          STA   STDZP
7C31: 28       >403          PLP
               >404  * AUXPTR a ete fixe dans ISAUXMEM a l´adresse du slot
               >405  * Adresse du 1er element en p0.
7C32: A0 00    >410          LDY   #0
7C34: 91 06    >411          STA   (AUXPTR),Y
7C36: C8       >412          INY
7C37: 8A       >414          TXA
7C38: 91 06    >415          STA   (AUXPTR),Y
7C3A: C8       >416          INY
```

```
7C3B: A5 AD    >417          LDA   STRNG2
7C3D: 91 06    >418          STA   (AUXPTR),Y
7C3F: C8       >419          INY
7C40: A5 AE    >420          LDA   STRNG2+1
7C42: 91 06    >421          STA   (AUXPTR),Y
7C44: A2 02    >422          LDX   #2          Init memory slot for array
7C46: 20 20 7D >423          JSR   ZRTAUX
7C49: 4C 5F 7C >424          JMP   :10
               >425  * Zero fill the element segment within the array
               >426  * (fast init).
7C4C: E6 AE    >427  :9      INC   STRNG2+1
7C4E: A4 AD    >428          LDY   STRNG2      # of byte mod 256
7C50: F0 05    >429          BEQ   :8          Upon a page limit
7C52: 88       >430  ]LOOP   DEY
7C53: 91 94    >431          STA   (ARYPNT),Y
7C55: D0 FB    >432          BNE   ]LOOP
7C57: C6 95    >433  :8      DEC   ARYPNT+1    Point to next page
7C59: C6 AE    >434          DEC   STRNG2+1    Count the pages
7C5B: D0 F5    >435          BNE   ]LOOP       Still more to clear
7C5D: E6 95    >436          INC   ARYPNT+1    Rollback last Decrement
7C5F: A5 10    >437  :10     LDA   DIMFLG
7C61: F0 04    >438          BEQ   NFAEP
7C63: 60       >439          RTS
               >440
7C64: 4C 10 D4 >441  GME     JMP   MEMERR      MEMORY FULL error
7C67: A0 04    >442  NFAEP   LDY   #4
               >443  * New routine for ROM FIND.ARRAY.ELEMENT
               >444  * Y reg. should be 4 upon entry
7C69: B1 9B    >445          LDA   (LOWTR),Y
7C6B: AA       >446          TAX
7C6C: 4A       >448          LSR
7C6D: 4A       >448          LSR
7C6E: 4A       >448          LSR
7C6F: 29 07    >450          AND   #7
7C71: 85 BF    >451          STA   AUXBANK
7C73: 8A       >452          TXA
7C74: 29 07    >453          AND   #7
7C76: 85 0F    >457          STA   NUMDIM
7C78: A9 00    >459          LDA   #0
7C7A: 85 AD    >460          STA   STRNG2
7C7C: 85 AE    >461  ]LOOP   STA   STRNG2+1
7C7E: C8       >462          INY               ;Pull next subscript from stack
7C7F: 68       >463          PLA
7C80: AA       >463          TAX
7C81: 86 A0    >464          STX   FAC+3
7C83: 68       >465          PLA
7C84: 85 A1    >466          STA   FAC+4
7C86: D1 9B    >467          CMP   (LOWTR),Y
7C88: 90 0B    >468          BCC   FAE2
7C8A: D0 06    >469          BNE   GSE         Subscript is too large
7C8C: C8       >470          INY
7C8D: 8A       >471          TXA
7C8E: D1 9B    >472          CMP   (LOWTR),Y
7C90: 90 04    >473          BCC   FAE3
7C92: 4C 96 E1 >474  GSE     JMP   SUBERR      BAD SUBSCRIPT error
7C95: C8       >475  FAE2    INY
7C96: A5 AE    >476  FAE3    LDA   STRNG2+1    Bypass multiplication if
```

```
7C98: 05 AD             ORA   STRNG2      value so far is zero
7C9A: 18                CLC
7C9B: F0 0A             BEQ   :1
7C9D: 20 AD E2          JSR   MULTPLSS
7CA0: 8A                TXA              ;Add current subscript
7CA1: 65 A0             ADC   FAC+3
7CA3: AA                TAX
7CA4: 98                TYA
7CA5: A4 5E             LDY   INDEX
7CA7: 65 A1    :1       ADC   FAC+4      Finish adding current subscript
7CA9: 86 AD             STX   STRNG2     Store accumulated offset
7CAB: C6 0F             DEC   NUMDIM     Last subscript yet?
7CAD: D0 CD             BNE   ]LOOP      No: loop till done
7CAF: 85 AE             STA   STRNG2+1   Yes: multiply by element size
7CB1: 20 BB 7C          JSR   KWELMSIZ
7CB4: A5 BF             LDA   AUXBANK
7CB6: F0 00             BEQ   :2
7CB8: 4C 98 E2 :2       JMP   $E298

               * Donne la taille de l´element en fonction
               * de VARNAM,+1 et de INTTYP
               * Result in X reg.
7CBB: 24 81    KWELMSIZ BIT   VARNAM
7CBD: 10 06             BPL   :0
7CBF: A5 12             LDA   INTTYP
7CC1: 29 07             AND   #7
7CC3: AA                TAX
7CC4: 60                RTS
7CC5: A2 05    :0       LDX   #5
7CC7: 24 82             BIT   VARNAM+1
7CC9: 10 02             BPL   :1
7CCB: CA                DEX              ;Back to 3 if string
7CCC: CA                DEX
7CCD: 60       :1       RTS

               * Evaluate numeric formula at TXTPPTR
               * Converting result to INTEGER 0<= X < 65536
               * into FAC+3,4
7CCE: 20 47 75 NMAKINT  JSR   RST100     Get next character
7CD1: 20 CE 84          JSR   NFRMNUM
               * Convert FAC to integer
7CD4: A5 A2             LDA   FACSIGN
7CD6: 30 0F             BMI   :1
7CD8: A5 9D             LDA   FAC
7CDA: C9 90             CMP   #$90
7CDC: 90 06             BCC   :3         Branch if abs(value) < 32768
7CDE: 20 64 8E          JSR   GN65536
7CE1: 20 BE E7          JSR   FADD
7CE4: 4C F2 EB :3       JMP   QINT
7CE7: 4C 99 E1 :1       JMP   GOIQERR

               * Convert INTTYP (in X reg.) from $81 to $84
               * to %0000_0000 to %1100_0000 (respectively)
               * Output value could be ORA ed or EOR ed with
               * NUMDIM slot with array structure
7CEA: CA       CNVT1    DEX
7CEB: 8A                TXA
```

```
7CEC: 4A                LSR              ;b0 into Carry, 0 into b7
7CED: 6A                ROR              ;b0 into b7 and b1 into carry
7CEE: 6A                ROR              ;b0 into b6, b1 into b7
7CEF: 29 C0             AND   #$C0       Only retain b6-b7
7CF1: 60                RTS

               * Cache mechanism for array variables
               ACTR     EQU   LOWTR
7CF2: A4 82    ALKCACH  LDY   VARNAM+1
7CF4: A5 81             LDA   VARNAM
7CF6: 86 9B             STX   SCTR
7CF8: A2 00             LDX   #0
7CFA: DD 99 99 ]LOOP    CMP   AVN,X
7CFD: D0 0F             BNE   :0
7CFF: 98                TYA
7D00: DD 9D 99          CMP   AVNP1,X
7D03: D0 07             BNE   :2
7D05: A5 12             LDA   INTTYP
7D07: DD A1 99          CMP   AIT,X
7D0A: F0 08             BEQ   :1
7D0C: A5 81    :2       LDA   VARNAM
7D0E: E8       :0       INX
7D0F: E4 9B             CPX   SCTR
7D11: D0 E7             BNE   ]LOOP
7D13: 60                RTS

7D14: BD A5 99 :1       LDA   ALTR,X
7D17: 85 9B             STA   LOWTR
7D19: BD A9 99          LDA   ALTRP1,X
7D1C: 85 9C             STA   LOWTR+1
7D1E: 8A                TXA
7D1F: 60                RTS

               * Common entry point for accessing array content
               * within auxiliary memory.
7D20: A9 BF    ZRTAUX   LDA   #$BF
7D22: 8D EE 03          STA   $03EE
7D25: A9 00             LDA   #0
7D27: 8D ED 03          STA   $03ED
7D2A: B8                CLV
7D2B: 38                SEC
7D2C: 4C 14 C3          JMP   XFER

7D2F: 2C 83 C0 NGARBAG  BIT   $C083
7D32: 2C 83 C0          BIT   $C083
7D35: 20 00 D0          JSR   $D000
7D38: 2C 81 C0          BIT   $C081
7D3B: 2C 81 C0          BIT   $C081
7D3E: 60                RTS
               * New strategy for array storage
                        PUT   PEERNAUXMEM
               * Module handling the new Peersoft array storage strategy

7D3F: 4C C9 DE GSNERR2  JMP   SYNERR
7D42: 4C 99 E1 GIQERR2  JMP   GOIQERR
7D45: 4C 76 DD GTMERR2  JMP   GOTMIERR
               * Routine to test whether the array will be located
               * Outcome:
               * Carry set iif aux. mem storage asked for
               * AUXBANK: bank memory asked for (in bits b4..b5)
               * ARYPNT,+1: incremented if aux mem. storage
               * (placeholders for offset within aux memory and
               *  one element of specified size for returning values
               *  during value expressions
               * Y,A: values incremented in case aux. mem storage
7D48: A1 B8    ISAUXMEM LDA   (TXTPTR,X) X vaut deja zero
7D4A: C9 23             CMP   #´#´
7D4C: 18                CLC
7D4D: D0 38             BNE   :2
7D4F: 20 47 75          JSR   RST100     Next char. must be numeric
7D52: B0 EB             BCS   GSNERR2    otherwise SYNTAX ERROR
7D54: 29 07             AND   #7
               * Pour le moment uniquement la memoire auxiliaire
               * est autorisee
7D56: C9 02             CMP   #2
7D58: B0 E8             BCS   GIQERR2
7D5A: 85 BF             STA   AUXBANK
7D5C: 20 47 75          JSR   RST100     Point to next character
7D5F: 18                CLC
               * test de conformance par rap. a la configuration hote
7D60: 2C EF 9C          BIT   MEMORY     b6 a 1 si carte mem aux.
7D63: A2 01             LDX   #1
7D65: 50 01             BVC   *+3
7D67: CA                DEX
7D68: 8A                TXA
7D69: 25 BF             AND   AUXBANK
7D6B: 85 BF             STA   AUXBANK
7D6D: F0 18             BEQ   :2
7D6F: A5 94             LDA   ARYPNT
7D71: A4 95             LDY   ARYPNT+1
7D73: 85 06             STA   AUXPTR
7D75: 84 07             STY   AUXPTR+1
7D77: 65 AD             ADC   STRNG2     Carry already clear
7D79: 90 02             BCC   *+4
7D7B: C8                INY
7D7C: 18                CLC
7D7D: 69 04             ADC   #4
7D7F: 90 01             BCC   *+3
7D81: C8                INY
7D82: 84 95             STY   ARYPNT+1
7D84: 38                SEC
7D85: 85 94             STA   ARYPNT
7D87: A5 94    :2       LDA   ARYPNT
7D89: 60       ]LOOP    RTS

7D8A: 2C EF 9C RCLMAUX  BIT   MEMORY
7D8D: 50 FA             BVC   ]LOOP
7D8F: A2 00             LDX   #0         Init array storage in aux mem.
7D91: 4C 20 7D          JMP   ZRTAUX

               * Upon init, all variables are floating point by default
7D94: 08       LBS00    PHP
7D95: A2 1A             LDX   #26
7D97: A9 21             LDA   #´!´
```

```
7D99: 9D 95 9B  ]LOOP    STA   TYPLET-1,X
7D9C: CA                 DEX
7D9D: D0 FA              BNE   ]LOOP
7D9F: 8E 23 96           STX   AEI
7DA2: 8E 24 96           STX   AEI+1
                * Reinit variables lookup caches (simple & array)
7DA5: 8E 83 99           STX   SNCCH
7DA8: 8E 98 99           STX   ANCCH
7DAB: 8E E7 9C           STX   WMODE
7DAE: 20 8A 7D           JSR   RCLMAUX
7DB1: 28                 PLP
7DB2: 60                 RTS

                * Applesoft RUN command
7DB3: 20 94 7D  RRUN     JSR   LBS00      Init the default vartype table
7DB6: 8E C1 99           STX   MONU       Rearms MOUSE instruction flag
7DB9: 4C 12 D9           JMP   $D912

                * Applesoft NEW command
7DBC: 20 94 7D  RNEW     JSR   LBS00
7DBF: 4C 4B D6           JMP   $D64B

                * Applesoft CLEAR command
7DC2: 20 94 7D  RCLEAR   JSR   LBS00
7DC5: 4C 6C D6           JMP   $D66C

7DC8: 20 7D E0  MISLETC  JSR   ISLETC
7DCB: 90 0A              BCC   GOSYNERR
7DCD: 60                 RTS

                * New subroutine checking a character (code in A)
                * is pointed to by TXTPTR
                * Falls into SYNERR if not
                NSYNCHR  DO    KOPT-K65C02
7DCE: A0 00              LDY   #0
7DD0: D1 B8     NSYNCHR2 CMP   (TXTPTR),Y
7DD2: D0 03              BNE   GOSYNERR
7DD4: 4C 47 75           JMP   RST100
7DD7: 4C C9 DE  GOSYNERR JMP   SYNERR

                         PUT   PEERPROCFUN
                * Module en charge des fonctions utilisateur
                * et particulierement des PF
                ARG      EQU   $A5
                TRCFLG   EQU   $F2
                BISVTYP  EQU   $BE
                VECTUSR  EQU   $A
                TMERR    EQU   $DD76
                ULERR    EQU   $D97C
                MOVFM    EQU   $EAF9
                MOVFA    EQU   $EB53
                LET2     EQU   $DA63

                         DUMMY 0
0000: 00        USRMOD   DS    1
0001: 00 00     ADRUSR   DS    2
0003: 00 00     VSRTNAM  DS    2
```

```
0005: 00        VSRTVT   DS    1
0006: 00        VSRTIT   DS    1
0007: 00 00     VSRTPTR  DS    2
0009: 00 00     VENT1NAM DS    2
000B: 00        VENT1VT  DS    1
000C: 00        VENT1IT  DS    1
000D: 00 00     VENT1PTR DS    2
000F: 00 00     VENT2NAM DS    2
0011: 00        VENT2VT  DS    1
0012: 00        VENT2IT  DS    1
0013: 00 00     VENT2PTR DS    2
                LENREC   EQU   *
                         DEND
                * Sous routine pour initialiser les routines USR de type
                * PF.
7DDA: A2 0A     RAZPF    LDX   #10
                ]LOOP    MPHX
7DDC: 8A                 TXA
7DDD: 48                 PHA
7DDE: 20 08 7E           JSR   COMPOFST
7DE1: 68                 PLA
7DE2: AA                 TAX
7DE3: A0 00              LDY   #USRMOD
7DE5: B1 06              LDA   (AUXPTR),Y
7DE7: 10 06              BPL   :0
7DE9: A0 02              LDY   #ADRUSR+1
7DEB: A9 00              LDA   #0
7DED: 91 06              STA   (AUXPTR),Y
7DEF: CA        :0       DEX
7DF0: 10 EA              BPL   ]LOOP
7DF2: 8E 81 99           STX   PFINDIC
7DF5: E8                 INX
7DF6: 8E 80 99           STX   ISPFACT
7DF9: 60                 RTS

7DFA: A2 0B     SETINITX LDX   #12-1
7DFC: BD 74 99  ]LOOP    LDA   SINITX,X
7DFF: 95 69              STA   $69,X
7E01: 9D 54 97           STA   SVALTNM,X
7E04: CA                 DEX
7E05: 10 F5              BPL   ]LOOP
7E07: 60                 RTS

                * Indice de la fonction dans X, ramene dans A,Y
                * L´adresse de debut de la structure
7E08: A9 00     COMPOFST LDA   #0
7E0A: A8                 TAY
7E0B: F0 05              BEQ   :00        Always
7E0D: 69 15     ]LOOP    ADC   #LENREC
7E0F: 90 02              BCC   :0
7E11: C8                 INY
7E12: 18        :00      CLC
7E13: CA        :0       DEX
7E14: 10 F7              BPL   ]LOOP
7E16: 69 45              ADC   #ADRSTRUCT
7E18: 48                 PHA
7E19: 98                 TYA
```

```
7E1A: 69 96    >79                ADC   #>ADRSTRUCT
7E1C: A8       >80                TAY
7E1D: 68       >81                PLA
7E1E: 85 06    >82                STA   AUXPTR
7E20: 84 07    >83                STY   AUXPTR+1
7E22: 60       >84                RTS
               >85
7E23: 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
7E24: 4C 76 DD >90    ]ERR     JMP   TMERR
7E27: 48       >91    FRSTIM   PHA
7E28: 20 49 86 >92             JSR   NCHKCOM
7E2B: A0 00    >96             LDY   #USRMOD
7E2D: B1 06    >97             LDA   (AUXPTR),Y
7E2F: 29 01    >99             AND   #1          Environnement dynamique oui/non
7E31: 48       >100            PHA
7E32: F0 0F    >101            BEQ   :0
7E34: A2 0B    >102            LDX   #12-1
7E36: B5 69    >103   ]LOOP    LDA   $69,X
7E38: 9D 48 97 >104            STA   SVCURRM,X
7E3B: BD 68 97 >105            LDA   SDEF1,X
7E3E: 95 69    >106            STA   $69,X
7E40: CA       >107            DEX
7E41: 10 F3    >108            BPL   ]LOOP
7E43: A5 07    >112   :0       LDA   AUXPTR+1
7E45: 48       >113            PHA
7E46: A5 06    >114            LDA   AUXPTR
7E48: 48       >115            PHA
7E49: 20 42 79 >117            JSR   NPTRGTX
7E4C: C5 6B    >118            CMP   ARYTAB
7E4E: 98       >119            TYA
7E4F: E5 6C    >120            SBC   ARYTAB+1
7E51: 68       >121            PLA
7E52: 85 06    >122            STA   AUXPTR
7E54: 68       >123            PLA
7E55: 85 07    >124            STA   AUXPTR+1
7E57: 68       >125            PLA
7E58: F0 0A    >126            BEQ   :1
7E5A: A2 0B    >127            LDX   #12-1
7E5C: BD 48 97 >128   ]LOOP    LDA   SVCURRM,X
7E5F: 95 69    >129            STA   $69,X
7E61: CA       >130            DEX
7E62: 10 F8    >131            BPL   ]LOOP
7E64: B0 BE    >132   :1       BCS   ]ERR
7E66: 68       >133            PLA
7E67: A8       >133            TAY
7E68: A5 81    >134            LDA   VARNAM
7E6A: 91 06    >135            STA   (AUXPTR),Y
7E6C: C8       >136            INY
7E6D: A5 82    >137            LDA   VARNAM+1
7E6F: 91 06    >138            STA   (AUXPTR),Y
7E71: C8       >139            INY
7E72: A5 11    >140            LDA   VALTYP
7E74: 91 06    >141            STA   (AUXPTR),Y
7E76: C8       >142            INY
```

```
7E77: A5 12    >143          LDA   INTTYP
7E79: 91 06    >144          STA   (AUXPTR),Y
7E7B: C8       >145          INY
7E7C: A5 83    >146 COMX1    LDA   VARPNT
7E7E: 91 06    >147          STA   (AUXPTR),Y
7E80: C8       >148          INY
7E81: A5 84    >149          LDA   VARPNT+1
7E83: 91 06    >150          STA   (AUXPTR),Y
7E85: 60       >151          RTS
               >152
               >153 * Connaitre tout d´une variable deja enregistree
               >154 * Y offset dans structure... (adressage par
               >155 * (AUXPTR),Y
7E86: B1 06    >156 SCNDTIM  LDA   (AUXPTR),Y
7E88: 85 81    >157          STA   VARNAM
7E8A: C8       >158          INY
7E8B: B1 06    >159          LDA   (AUXPTR),Y
7E8D: 85 82    >160          STA   VARNAM+1
7E8F: C8       >161          INY
7E90: B1 06    >162          LDA   (AUXPTR),Y
7E92: 85 11    >163          STA   VALTYP
7E94: C8       >164          INY
7E95: B1 06    >165          LDA   (AUXPTR),Y
7E97: 85 12    >166          STA   INTTYP
7E99: C8       >167          INY
7E9A: 98       >168          TYA
7E9B: 48       >168          PHA
7E9C: 20 B8 79 >169          JSR   NPTRGL90
7E9F: 68       >170          PLA
7EA0: A8       >170          TAY
7EA1: 4C 7C 7E >171          JMP   COMX1
               >172
               >173 * X,A adresse a sauver dans ADRUSR de la structure
7EA4: A0 01    >174 HNDLEADR LDY   #ADRUSR
7EA6: 91 06    >175          STA   (AUXPTR),Y
7EA8: 90 08    >176          BCC   :4
7EAA: 85 0B    >177          STA   $0B
7EAC: 86 0C    >178          STX   $0C
7EAE: A9 4C    >179          LDA   #$4C
7EB0: 85 0A    >180          STA   $0A
7EB2: C8       >181 :4       INY
7EB3: 8A       >182          TXA
7EB4: 91 06    >183          STA   (AUXPTR),Y
7EB6: 60       >184          RTS
               >185
7EB7: B1 06    >186 COMLET2  LDA   (AUXPTR),Y
7EB9: AA       >187          TAX              ;INTTYP dans X
7EBA: C8       >188          INY
7EBB: B1 06    >189          LDA   (AUXPTR),Y ;pointeur sur valeur
7EBD: 85 85    >190          STA   FORPNT      dans FORPNT
7EBF: C8       >191          INY
7EC0: B1 06    >192          LDA   (AUXPTR),Y
7EC2: 85 86    >193          STA   FORPNT+1
7EC4: 8A       >194          TXA              ;Set bit N
7EC5: 4C 63 DA >195          JMP   LET2
               >196
7EC8: 4C 10 D4 >197 ]ERR     JMP   MEMERR
```

```
7ECB: 20 47 75 >198  RUSR      JSR   RST100
7ECE: A2 0A    >199            LDX   #10
7ED0: B0 06    >200            BCS   :0          Not a digit
7ED2: E9 2F    >201            SBC   #´0´-1
7ED4: AA       >202            TAX
7ED5: 20 47 75 >203            JSR   RST100
               >204  :0        MPHX
7ED8: 8A       >204            TXA
7ED9: 48       >204            PHA
7EDA: 20 08 7E >205            JSR   COMPOFST
7EDD: A0 00    >209            LDY   #USRMOD
7EDF: B1 06    >210            LDA   (AUXPTR),Y
7EE1: 29 40    >212            AND   #64
7EE3: F0 42    >213            BEQ   :1
7EE5: BA       >214            TSX
7EE6: E0 08    >215            CPX   #8          At least 8 bytes on stack OK
7EE8: 90 DE    >216            BCC   ]ERR
7EEA: 20 4C 86 >217            JSR   NCHKOPN
7EED: 20 7B DD >218            JSR   FRMEVL
7EF0: BA       >219            TSX
7EF1: A5 11    >220            LDA   VALTYP
7EF3: 9D 00 01 >221            STA   $0100,X
7EF6: 8A       >222            TXA
7EF7: 38       >223            SEC
7EF8: E9 06    >224            SBC   #6
7EFA: AA       >225            TAX
7EFB: 9A       >226            TXS
7EFC: E8       >227            INX
7EFD: A0 01    >228            LDY   #1
7EFF: 20 2B EB >229            JSR   MOVMF
7F02: 20 49 86 >230            JSR   NCHKCOM
7F05: 20 43 86 >231            JSR   NPARCHK+3  2nd arg value left in FAC
7F08: BA       >232            TSX
7F09: E8       >233            INX
7F0A: 8A       >234            TXA
7F0B: 48       >235            PHA
7F0C: A0 01    >236            LDY   #1
7F0E: 20 E3 E9 >237            JSR   $E9E3       Load ARG from Y,A/1st arg value
7F11: 68       >238            PLA
7F12: 18       >239            CLC
7F13: 69 05    >240            ADC   #5          6 instead of 5 because of INX
7F15: AA       >241            TAX
7F16: BD 00 01 >242            LDA   $0100,X
7F19: 85 BE    >243            STA   BISVTYP
7F1B: 9A       >244            TXS
7F1C: 4C 2A 7F >245            JMP   :2
7F1F: A2 26    >246  ]ERR      LDX   #38
7F21: 2C       >247            HEX   2C          Skip next two bytes
7F22: A2 27    >248  ]ERR1     LDX   #39
7F24: 4C E9 8D >249            JMP   NERRH
7F27: 20 40 86 >250  :1        JSR   NPARCHK     1er ou 2eme parm dans FAC
               >251  :2        MPLX
7F2A: 68       >251            PLA
7F2B: AA       >251            TAX
7F2C: 48       >255            PHA
7F2D: 20 08 7E >257            JSR   COMPOFST   Set AUXPTR according index X
7F30: A0 02    >258            LDY   #ADRUSR+1
```

```
7F32: B1 06             LDA   (AUXPTR),Y
7F34: F0 E9             BEQ   ]ERR
7F36: 68                PLA
7F37: AA                TAX
7F38: 8E 82 99          STX   PFINDX
7F3B: A0 00             LDY   #USRMOD
7F3D: B1 06             LDA   (AUXPTR),Y
7F3F: 10 4E             BPL   V3
               * Procedural function...
7F41: 4A                LSR
7F42: 90 2D             BCC   :10        Branchem. ssi pas de segment
7F44: AD 80 99          LDA   ISPFACT
7F47: D0 D9             BNE   ]ERR1
7F49: 8A                TXA
7F4A: 48                PHA
7F4B: 20 E5 80          JSR   SAVCURRM
7F4E: 68                PLA
7F4F: CD 81 99          CMP   PFINDIC
7F52: F0 03             BEQ   :11
7F54: 20 FA 7D          JSR   SETINITX
7F57: 20 DA 80 :11      JSR   RSTALTM
7F5A: A0 03             LDY   #VSRTNAM
7F5C: 20 86 7E          JSR   SCNDTIM
7F5F: A0 09             LDY   #VENT1NAM
7F61: 20 86 7E          JSR   SCNDTIM
7F64: A0 00             LDY   #USRMOD
7F66: B1 06             LDA   (AUXPTR),Y
7F68: 29 40             AND   #64
7F6A: F0 05             BEQ   :10
7F6C: A0 0F             LDY   #VENT2NAM
7F6E: 20 86 7E          JSR   SCNDTIM
7F71: A0 0C    :10      LDY   #VENT1IT
7F73: 20 B7 7E          JSR   COMLET2
7F76: A0 00             LDY   #USRMOD
7F78: B1 06             LDA   (AUXPTR),Y
7F7A: 29 40             AND   #64
7F7C: F0 08             BEQ   :12
7F7E: 20 53 EB          JSR   MOVFA
7F81: A0 12             LDY   #VENT2IT
7F83: 20 B7 7E          JSR   COMLET2
               :12      DO    KOPT16
7F86: A9 80    V3T      LDA   #>RETOUR-1
7F88: 48                PHA
7F89: A9 71    V3B      LDA   #RETOUR-1
7F8B: 48                PHA
7F8C: 4C AA 7F          JMP   COMMONG

               * Code run when parsing USR function that is not a PF
7F8F: E0 0A    V3       CPX   #10
7F91: B0 14             BCS   :4         Special case for original USR
7F93: A0 02             LDY   #ADRUSR+1
7F95: B1 06             LDA   (AUXPTR),Y
7F97: AA                TAX
7F98: 88                DEY
7F99: B1 06             LDA   (AUXPTR),Y
7F9B: D0 01             BNE   *+3
7F9D: CA                DEX
```

```
7F9E: 38       >332            SEC
7F9F: E9 01    >333            SBC   #1
7FA1: A8       >339            TAY
7FA2: 8A       >340            TXA
7FA3: 48       >340            PHA
7FA4: 98       >341            TYA
7FA5: 48       >341            PHA
7FA6: 60       >343            RTS
7FA7: 4C 0A 00 >344  :4        JMP   VECTUSR
               >345
7FAA: A0 0D    >346  COMMONG   LDY   #FINOF-SVOFST-1
7FAC: BE 2C 97 >347  ]LOOP     LDX   SVOFST,Y
7FAF: B5 00    >348            LDA   0,X
7FB1: 99 3A 97 >349            STA   SVAREA,Y
7FB4: 88       >350            DEY
7FB5: 10 F5    >351            BPL   ]LOOP
7FB7: C8       >355            INY
7FB8: 84 F2    >356            STY   TRCFLG
               >358  * This is the critical code segment
7FBA: A5 B9    >363            LDA   TXTPTR+1
7FBC: 48       >364            PHA
7FBD: A5 B8    >365            LDA   TXTPTR
7FBF: 48       >366            PHA
7FC0: A5 76    >367            LDA   CURLIN+1
7FC2: 48       >368            PHA
7FC3: A5 75    >369            LDA   CURLIN
7FC5: 48       >370            PHA
7FC6: A9 B0    >372            LDA   #TOKGOSUB
7FC8: 48       >373            PHA
7FC9: A0 01    >374            LDY   #ADRUSR
7FCB: B1 06    >375            LDA   (AUXPTR),Y
7FCD: 85 B8    >376            STA   TXTPTR
7FCF: C8       >377            INY
7FD0: B1 06    >378            LDA   (AUXPTR),Y
7FD2: 85 B9    >379            STA   TXTPTR+1
7FD4: 4C D2 D7 >380            JMP   NEWSTT
               >381
7FD7: 20 65 75 >382  RDEFUSR   JSR   RST102
7FDA: 90 05    >383            BCC   :1          Branch if digit
7FDC: A9 0A    >384            LDA   #10
7FDE: 48       >385            PHA
7FDF: D0 06    >386            BNE   :3          Always
7FE1: E9 2F    >387  :1        SBC   #´0´-1      ASCII digit to binary
7FE3: 48       >388            PHA
7FE4: 20 47 75 >389            JSR   RST100
7FE7: A9 D0    >390  :3        LDA   #TOKEQUAL
7FE9: 20 CE 7D >391            JSR   NSYNCHR
7FEC: 20 67 DD >392            JSR   FRMNUM
7FEF: 20 52 E7 >393            JSR   GETADR
7FF2: 68       >394            PLA
7FF3: AA       >394            TAX
7FF4: 48       >398            PHA
7FF5: 20 08 7E >400            JSR   COMPOFST
7FF8: 68       >401            PLA
7FF9: 48       >402            PHA
7FFA: C9 0A    >403            CMP   #10         Set carry flag
               >404  * If LINNUM high byte is zero, then must be the mode
```

```
7FFC: A5 50    >405          LDA   LINNUM
7FFE: A6 51    >406          LDX   LINNUM+1
8000: F0 12    >407          BEQ   :5
8002: 20 A4 7E >408          JSR   HNDLEADR
8005: 68       >409          PLA
8006: A9 00    >410          LDA   #0
8008: A8       >414          TAY
8009: 91 06    >415          STA   (AUXPTR),Y
800B: 20 65 75 >417  ]LOOP   JSR   RST102
800E: D0 01    >418          BNE   *+3
8010: 60       >419          RTS
8011: 4C C9 DE >420  ]ERR    JMP   SYNERR
               >421  * DEFUSR=<mode>,<otherparms>
8014: A0 00    >425  :5      LDY   #USRMOD
8016: 91 06    >426          STA   (AUXPTR),Y
8018: A8       >428          TAY
8019: 30 25    >429          BMI   :6          Procedural function
801B: 29 3F    >430          AND   #$3F
801D: D0 F2    >431          BNE   ]ERR
801F: 20 49 86 >432          JSR   NCHKCOM
8022: 20 67 DD >433          JSR   FRMNUM
8025: 20 52 E7 >434          JSR   GETADR
8028: 68       >435          PLA
8029: AA       >435          TAX
802A: E0 0A    >436          CPX   #10
802C: 08       >437          PHP
802D: 20 08 7E >438          JSR   COMPOFST
8030: 28       >439          PLP
8031: A5 50    >440          LDA   LINNUM
8033: A6 51    >441          LDX   LINNUM+1
8035: 4C A4 7E >442  ]LOOP   JMP   HNDLEADR
8038: 4C 7C D9 >443  ]ERR    JMP   ULERR
803B: A2 28    >444  ]ERR1   LDX   #40
803D: 4C E9 8D >445          JMP   NERRH
8040: 48       >446  :6      PHA
8041: AD 80 99 >447          LDA   ISPFACT
8044: D0 F5    >448          BNE   ]ERR1
8046: A9 03    >449          LDA   #VSRTNAM
8048: 20 27 7E >450          JSR   FRSTIM
804B: A9 09    >451          LDA   #VENT1NAM
804D: 20 27 7E >452          JSR   FRSTIM
8050: 68       >453          PLA
8051: 29 40    >454          AND   #64
8053: F0 05    >455          BEQ   :7
8055: A9 0F    >456          LDA   #VENT2NAM
8057: 20 27 7E >457          JSR   FRSTIM
805A: 68       >458  :7      PLA               ;Do not care routine idx
805B: 20 49 86 >459          JSR   NCHKCOM
805E: 20 0C DA >460          JSR   LINGET
8061: 20 1A D6 >461          JSR   FNDLIN
8064: 90 D2    >462          BCC   ]ERR
8066: A6 9C    >463          LDX   LOWTR+1
8068: A5 9B    >464          LDA   LOWTR
806A: D0 01    >465          BNE   *+3
806C: CA       >466          DEX
806D: E9 01    >470          SBC   #1          Carry already set
806F: 18       >472          CLC
```

```
8070: 90 C3    >473              BCC    ]LOOP      Always
               >474
8072: 20 96 80 >475  RETOUR      JSR    COMREST
8075: AE 82 99 >476              LDX    PFINDX
8078: 8A       >477              TXA
8079: 48       >477              PHA
807A: 20 08 7E >478              JSR    COMPOFST
807D: 20 A4 80 >479              JSR    COLLECTR
8080: 68       >480              PLA
8081: AA       >480              TAX
8082: A0 00    >485              LDY    #USRMOD
8084: 8C 80 99 >486              STY    ISPFACT
8087: B1 06    >487              LDA    (AUXPTR),Y
8089: 4A       >489              LSR
808A: 90 09    >490              BCC    :0
808C: 8E 81 99 >491              STX    PFINDIC
808F: 20 F0 80 >492              JSR    SAVALTM
8092: 4C CF 80 >493              JMP    RSTCURRM
8095: 60       >494  :0          RTS
               >495
8096: A0 0D    >496  COMREST     LDY    #FINOF-SVOFST-1
8098: BE 2C 97 >497  ]LOOP       LDX    SVOFST,Y
809B: B9 3A 97 >498              LDA    SVAREA,Y
809E: 95 00    >499              STA    0,X
80A0: 88       >500              DEY
80A1: 10 F5    >501              BPL    ]LOOP
80A3: 60       >502              RTS
               >503
80A4: A0 06    >504  COLLECTR LDY    #VSRTIT
80A6: B1 06    >505              LDA    (AUXPTR),Y
80A8: 0A       >506              ASL
80A9: A0 07    >507              LDY    #VSRTPTR
80AB: B1 06    >508              LDA    (AUXPTR),Y
80AD: AA       >509              TAX
80AE: C8       >510              INY
80AF: B1 06    >511              LDA    (AUXPTR),Y
80B1: A8       >512              TAY
80B2: 8A       >513              TXA
80B3: B0 09    >514              BCS    :0         Branch iif integer output var.
80B5: A2 00    >519              LDX    #0
80B7: 86 11    >520              STX    VALTYP
80B9: 86 12    >521              STX    INTTYP
80BB: 4C F9 EA >523              JMP    MOVFM
80BE: 84 84    >524  :0          STY    VARPNT+1
80C0: 85 83    >525              STA    VARPNT
80C2: A0 00    >530              LDY    #0
80C4: B1 83    >531              LDA    (VARPNT),Y
80C6: C8       >532              INY
80C7: AA       >534              TAX
80C8: B1 83    >535              LDA    (VARPNT),Y
80CA: A8       >536              TAY
80CB: 8A       >537              TXA
80CC: 4C F2 E2 >538              JMP    GIVAYF
               >539
80CF: A2 0B    >540  RSTCURRM LDX    #12-1
80D1: BD 48 97 >541  ]LOOP       LDA    SVCURRM,X
80D4: 95 69    >542              STA    $69,X
```

```
80D6: CA                  DEX
80D7: 10 F8               BPL   ]LOOP
80D9: 60                  RTS

80DA: A2 0B     RSTALTM   LDX   #12-1
80DC: BD 54 97  ]LOOP     LDA   SVALTNM,X
80DF: 95 69               STA   $69,X
80E1: CA                  DEX
80E2: 10 F8               BPL   ]LOOP
80E4: 60                  RTS

80E5: A2 0B     SAVCURRM  LDX   #12-1
80E7: B5 69     ]LOOP     LDA   $69,X
80E9: 9D 48 97            STA   SVCURRM,X
80EC: CA                  DEX
80ED: 10 F8               BPL   ]LOOP
80EF: 60                  RTS

80F0: A2 0B     SAVALTM   LDX   #12-1
80F2: B5 69     ]LOOP     LDA   $69,X
80F4: 9D 54 97            STA   SVALTNM,X
80F7: CA                  DEX
80F8: 10 F8               BPL   ]LOOP
80FA: 60                  RTS
                          PUT   PEERDEF
                * Nouvelle routine de traitement du DEF..
80FB: 4C D7 7F  ]LOOP     JMP   RDEFUSR
80FE: A4 B9     RDEF      LDY   TXTPTR+1
8100: A5 B8               LDA   TXTPTR
8102: 38                  SEC
8103: E9 01               SBC   #1
8105: B0 01               BCS   *+3
8107: 88                  DEY
8108: A2 01               LDX   #1
810A: 20 52 82            JSR   RECON      Check which DEF pattern
810D: D0 03               BNE   :1         None detected
810F: 4C 13 E3            JMP   $E313
8112: 88        :1        DEY
8113: 20 98 D9            JSR   ADDON
8116: A6 BD               LDX   IDMOCL
8118: E0 0A               CPX   #OFFUSR-TOFFST Is it DEFUSR?
811A: F0 DF               BEQ   ]LOOP
811C: BD 77 9B            LDA   MOTIF-NOPER-7,X Must be DEF(INT/STR/SNG)
                * Below is the common code for all three new instructions
811F: A0 00               LDY   #0
8121: 84 C0               STY   LETINF
8123: 85 C1               STA   TYPMOD
8125: 20 97 81            JSR   DECTPTR    Decrement TXTPTR
8128: 20 5E 81  ]LOOP     JSR   :LBS00     Bump ptr. to 1st letter of next v
ar
812B: 20 C8 7D            JSR   MISLETC    Must be alphabetic
812E: 85 C0               STA   LETINF
8130: 20 5E 81            JSR   :LBS00     Exit if no further variable
8133: C9 C9               CMP   #TOKMINUS  means a letter range
8135: F0 0B               BEQ   :2
8137: C9 2C               CMP   #´,´       Character must be either ´,´
8139: D0 34               BNE   GSNERR3     or ´-´
```

```
813B: A6 C0              LDX   LETINF     Process current letter
813D: 20 69 81           JSR   RDEFSUB
8140: 10 E6              BPL   ]LOOP      Always
8142: 20 47 75  :2       JSR   RST100     Range:get the upper range let.
8145: 20 C8 7D           JSR   MISLETC
8148: C5 C0              CMP   LETINF     Must not < 1st letter
814A: 90 23              BCC   GSNERR3
814C: AA                 TAX              ;Into X for processing
814D: 20 69 81  ]JLOOP   JSR   RDEFSUB    process current letter within
8150: CA                 DEX
8151: E4 C0              CPX   LETINF     Loop until 1st letter
8153: B0 F8              BCS   ]JLOOP
8155: 20 5E 81           JSR   :LBS00
8158: C9 2C              CMP   #´,´
815A: D0 13              BNE   GSNERR3
815C: F0 CA              BEQ   ]LOOP      Always
815E: 20 47 75  :LBS00   JSR   RST100
8161: D0 0B              BNE   R          Do not return if EOI
8163: 68                 PLA
8164: 68                 PLA
8165: A6 C0     :FIN     LDX   LETINF
8167: F0 06              BEQ   GSNERR3    Whaever args, process last letter
8169: A5 C1     RDEFSUB  LDA   TYPMOD
816B: 9D 55 9B           STA   TYPLET-´A´,X
816E: 60        R        RTS
816F: 4C C9 DE  GSNERR3  JMP   SYNERR


8172: 20 47 75  ROUT1Y   JSR   RST100
8175: 48                 PHA
8176: BD 8E 9B  ROUT1X   LDA   TVNORA,X
8179: 05 81              ORA   VARNAM
817B: 85 81              STA   VARNAM
817D: BD 92 9B           LDA   TVN1ORA,X
8180: 05 82              ORA   VARNAM+1
8182: 85 82              STA   VARNAM+1
8184: 20 53 E0           JSR   $E053      Attention, il faudra chg.
8187: 68                 PLA
8188: 60                 RTS



8189: BD 55 9B  XFRMMOT1 LDA   TYPLET-´A´,X
                XFROMMOT
                * X=0 for ´%´, 1 for ´$´ and 2 for ´!´, 3 for ´.´
818C: A2 03              LDX   #TITVAL-MOTIF-1
818E: DD 82 9B  ]LOOP    CMP   MOTIF,X
8191: F0 03              BEQ   :0
8193: CA                 DEX
8194: 10 F8              BPL   ]LOOP
8196: 60        :0       RTS

                * Decrement TXTPTR
8197: A5 B8     DECTPTR  LDA   TXTPTR
8199: D0 02              BNE   :0
819B: C6 B9              DEC   TXTPTR+1
819D: C6 B8     :0       DEC   TXTPTR
```

```
819F: 60                 RTS

                 * Subroutine to patch CHRGET/CHRGOT in page zero
81A0: A9 4C      SETUPB  LDA   #$4C        JMP absolute
81A2: 85 B1              STA   $B1
81A4: 85 BA              STA   $BA
81A6: A9 24              LDA   #DEBUTGET
81A8: 85 B2              STA   $B2
81AA: A9 75              LDA   #>DEBUTGET
81AC: 85 B3              STA   $B2+1
81AE: A9 7E              LDA   #DEBUTGOT
81B0: 85 BB              STA   $BB
81B2: A9 75              LDA   #>DEBUTGOT
81B4: 85 BC              STA   $BB+1
81B6: 60                 RTS

                 SETUPD  STID  BANCLD;$9D72
81B7: A9 C2              LDA   #BANCLD
81B9: 8D 72 9D           STA   $9D72
81BC: A9 81              LDA   #>BANCLD
81BE: 8D 73 9D           STA   $9D72+1
81C1: 60                 RTS

                 * Subr. called upon a BASIC cold boot (FP DOS command)
81C2: A2 FF      BANCLD  LDX   #$FF
81C4: 86 76              STX   $76
81C6: A2 FB              LDX   #$FB
81C8: 9A                 TXS
81C9: A9 28              LDA   #$28
81CB: A0 F1              LDY   #$F1
81CD: 85 01              STA   1
81CF: 84 02              STY   2
81D1: 85 04              STA   4
81D3: 84 05              STY   5
81D5: 20 73 F2           JSR   $F273
81D8: A9 4C              LDA   #$4C        JMP absolute
81DA: 85 00              STA   0
81DC: 85 03              STA   3
81DE: 85 90              STA   $90
81E0: 85 0A              STA   $A
81E2: A9 99              LDA   #$99
81E4: A0 E1              LDY   #$E1
81E6: 85 0B              STA   $B
81E8: 84 0C              STY   $C
81EA: 20 A0 81           JSR   SETUPB      Install CHRGET/CHRGOT patch in pa
ge zero
81ED: 4C 5C F1           JMP   $F15C       End of initialization in ROM

                 * Do the DOS init
                 NOUVIN  STID  $E000;$9D72
81F0: A9 00              LDA   #$E000
81F2: 8D 72 9D           STA   $9D72
81F5: A9 E0              LDA   #>$E000
81F7: 8D 73 9D           STA   $9D72+1
81FA: A9 4C              LDA   #$4C        JMP absolute
81FC: 8D C8 A2           STA   $A2C8
81FF: A9 0B              LDA   #$B
```

```
8201: 20 AA A2            JSR   $A2AA
8204: A9 20               LDA   #$20
8206: 8D C8 A2            STA   $A2C8
8209: A5 45               LDA   OPRND+1
820B: D0 06               BNE   :4          No error during DoClose
820D: 20 B7 81            JSR   SETUPD      Reinstall Peersoft
8210: 4C C8 A6            JMP   $A6C8        before exiting
8213: A2 60     :4        LDX   #$60
8215: 8E E7 A2            STX   $A2E7
8218: 20 D2 A2            JSR   $A2D2       Copy file manager parmlist
821B: A9 4C               LDA   #$4C        JMP absolute
821D: 8D E7 A2            STA   $A2E7
8220: AD 00 9D            LDA   DBUFP
8223: 8D 3B 82            STA   E06+1
8226: AD 01 9D            LDA   DBUFP+1
8229: 8D 40 82            STA   E06+6
822C: A9 D3               LDA   #$9CD3
822E: 8D 00 9D            STA   DBUFP
8231: A9 9C               LDA   #>$9CD3
8233: 8D 01 9D            STA   DBUFP+1
8236: 20 06 AB            JSR   $AB06       File manager main entry (INIT)
8239: 08                  PHP               ;Save status
                E06       STID  0;DBUFP     Reinstall Peersoft DOS features
823A: A9 00               LDA   #0
823C: 8D 00 9D            STA   DBUFP
823F: A9 00               LDA   #>0
8241: 8D 01 9D            STA   DBUFP+1
8244: 20 B7 81            JSR   SETUPD
8247: 28                  PLP
8248: 20 EB A6            JSR   $A6EB       process possible error after FM c
all
824B: 4C 97 A3            JMP   $A397       Goto SAVE (HELLO) command handler

                * RECON is a subroutine which scans BASIC program area
                * or input buffer for a Peersoft new keyword
                * 2 entry points:
                * RECON1 (BASIC statement execution): the pointer is TXTPTR
                * RECON (BASIC statement listing): the pointer is in A,Y
                * X value of 0: search for every new keyword (LIST)
                *             1: search only DEF patterns
                *             2: search only function statements
                *                (IIF, MOUSE and TIMER)
                *             3: search only MOUSE and TIMER keywords
                * On exit, Z bit set means no keyword found
                *                 clear means keyword (index in IDMOCL)
824E: A5 B8     RECON1    LDA   TXTPTR
8250: A4 B9               LDY   TXTPTR+1
8252: 85 06     RECON     STA   AUXPTR
8254: 84 07               STY   AUXPTR+1
8256: BD 70 9B  RECON2    LDA   TIDMOCL,X
8259: 85 BD               STA   IDMOCL
825B: BD 76 9B            LDA   TOFFIN,X
825E: 8D 42 9B            STA   IFDEF
8261: BD 7C 9B            LDA   TOFFIN2,X
8264: 8D 33 9B            STA   IFIIF
8267: E6 BD     :1        INC   IDMOCL
8269: A4 BD               LDY   IDMOCL
```

```
826B: BE 5F 9B            LDX   TOFFST,Y
826E: 86 C2               STX   OFFSET
8270: A0 00               LDY   #0
8272: BD 1F 9B  ]LOOP     LDA   TMOCL,X
8275: F0 0C               BEQ   :4          Keyword found: exit
8277: C9 FF               CMP   #$FF        End of table?
8279: F0 08               BEQ   :4          Yes: no keyword found
827B: D1 06               CMP   (AUXPTR),Y  Current character match?
827D: D0 E8               BNE   :1          no: try next keyword from table
827F: E8                  INX               ;Next char. from current keyword
8280: C8                  INY
8281: D0 EF               BNE   ]LOOP

                :4        DO    KOPT-K65C02
8283: AA                  TAX
8284: E8                  INX
8285: 60        RETURN    RTS

                          PUT   PEERLIST,D1
8286: 90 0A     STDLIS    BCC   STRTRNG

8288: F0 08               BEQ   STRTRNG
828A: C9 C9               CMP   #TOKMINUS
828C: F0 04               BEQ   STRTRNG
828E: C9 2C               CMP   #´,´
8290: D0 F3               BNE   RETURN

8292: 20 B7 93  STRTRNG   JSR   DECOMPILE
8295: 20 0C DA            JSR   LINGET
8298: 20 1A D6            JSR   FNDLIN
829B: 20 65 75            JSR   RST102
829E: F0 10               BEQ   MAINLIST
82A0: C9 C9               CMP   #TOKMINUS
82A2: F0 04               BEQ   ENDRNG
82A4: C9 2C               CMP   #´,´
82A6: D0 DD               BNE   RETURN

82A8: 20 47 75  ENDRNG    JSR   RST100
82AB: 20 0C DA            JSR   LINGET
82AE: D0 D5               BNE   RETURN

82B0: 68        MAINLIST  PLA
82B1: 68                  PLA
82B2: A5 50               LDA   LINNUM      In case no second line given,
82B4: 05 51               ORA   LINNUM+1     let it be 65535
82B6: D0 04               BNE   NXLST
82B8: C6 50               DEC   LINNUM
82BA: C6 51               DEC   LINNUM+1

82BC: A0 01     NXLST     LDY   #1
82BE: B1 9B               LDA   (LOWTR),Y
82C0: F0 6D               BEQ   LISTED      End of program found
82C2: 20 58 D8            JSR   ISCNTC      Check for Ctrl-C keystroke
82C5: 20 FB DA            JSR   CRDO
82C8: C8                  INY
82C9: B1 9B               LDA   (LOWTR),Y   Line number in X,A
82CB: AA                  TAX
```

```
82CC: C8        >39             INY
82CD: B1 9B     >40             LDA   (LOWTR),Y
82CF: C5 51     >41             CMP   LINNUM+1   Beyond last line number?
82D1: D0 04     >42             BNE   LSTD?
82D3: E4 50     >43             CPX   LINNUM
82D5: F0 02     >44             BEQ   LST1LIN
82D7: B0 56     >45    LSTD?    BCS   LISTED     Yes
                >46
82D9: 84 85     >47    LST1LIN  STY   $85
82DB: A0 00     >49             LDY   #0
82DD: 84 BE     >50             STY   MODREM
82DF: 84 BF     >51             STY   MODDAT
82E1: 84 C0     >52             STY   GFLAG
82E3: 84 C1     >53             STY   DEFFLG
82E5: 20 35 83 >60              JSR   VLINPRT    Print line #
82E8: A9 20     >61    ]JLOOP   LDA   #32        Print space after line number
82EA: A4 85     >62             LDY   $85
82EC: 2C        >63             HEX   2C
82ED: A9 2D     >64    L088     LDA   #´-´
82EF: C9 22     >65    L08      CMP   #´"´       Is it ´"´?
82F1: D0 08     >66             BNE   :9
82F3: A5 C0     >67             LDA   GFLAG
82F5: 49 FF     >68             EOR   #$FF
82F7: 85 C0     >69             STA   GFLAG
82F9: A9 22     >70             LDA   #´"´
                >71    * Now we test for an EOI
82FB: 24 BE     >72    :9       BIT   MODREM     If a REM has been scanned in this
 line
82FD: 30 0C     >73             BMI   SENDCHR
82FF: 24 C0     >74             BIT   GFLAG      Are we within a string litteral?
8301: 30 08     >75             BMI   SENDCHR    Same output as for a REM
8303: C9 3A     >76             CMP   #´:´       Current char is EOI?
8305: D0 04     >77             BNE   SENDCHR
8307: 85 BF     >78             STA   MODDAT     MODDAT b7 forced to zero
8309: 85 C1     >79             STA   DEFFLG     DEFFLG b7 forced to zero
830B: 20 5C DB >80     SENDCHR  JSR   OUTDO      Print current char
830E: A5 24     >81             LDA   CH
8310: C9 21     >82             CMP   #33        Have we reached "right" edge of s
creen?
8312: 90 07     >83             BCC   NCR        No
8314: 20 FB DA >84              JSR   CRDO       Yes: print CR for next line
8317: A9 05     >85             LDA   #5
8319: 85 24     >86             STA   CH
                >87    * Next character from line
831B: C8        >88    NCR      INY
831C: B1 9B     >89             LDA   (LOWTR),Y
831E: D0 18     >90             BNE   TOKEN?     Not end of line
8320: 85 C1     >91             STA   DEFFLG
8322: A8        >93             TAY              ;Force Y to 0
8323: B1 9B     >94             LDA   (LOWTR),Y  Update next line pointer
8325: AA        >95             TAX
8326: C8        >96             INY
8327: B1 9B     >102            LDA   (LOWTR),Y
8329: 86 9B     >103            STX   LOWTR
832B: 85 9C     >104            STA   LOWTR+1
832D: D0 8D     >105            BNE   NXLST      Branch if not at program´s end
                >106
```

```
832F: 20 FB DA  LISTED   JSR   CRDO
8332: 4C D2 D7           JMP   NEWSTT
8335: 6C FA D6  VLINPRT  JMP   ($D6FA)
8338: AA        TOKEN?   TAX                ;Character in X
8339: A5 BE              LDA   MODREM       Is litteral mode active?
833B: 05 BF              ORA   MODDAT
833D: 05 C0              ORA   GFLAG
833F: 0A                 ASL
8340: 8A                 TXA
8341: B0 AC              BCS   L08          Yes
8343: 84 B5              STY   YSAV
8345: 98                 TYA                ;Compute Y, A = LOWTR + Y
8346: A4 9C              LDY   LOWTR+1
8348: 65 9B              ADC   LOWTR        Carry already clear
834A: 90 01              BCC   :14
834C: C8                 INY
834D: A2 00     :14      LDX   #0
834F: 20 52 82           JSR   RECON        New BASIC keyword?
8352: D0 33              BNE   :23          Yes

8354: A4 B5              LDY   YSAV         Y = offset within line
8356: B1 9B              LDA   (LOWTR),Y    Current character
8358: 10 95              BPL   L08          Not a token
835A: 24 C1              BIT   DEFFLG
835C: 10 04              BPL   :18
835E: C9 C9              CMP   #TOKMINUS
8360: F0 8B              BEQ   L088
8362: C9 B2     :18      CMP   #TOKREM      REM token?
8364: D0 02              BNE   :15
8366: 66 BE              ROR   MODREM       bit 7 to 1 in MODREM
8368: C9 83     :15      CMP   #TOKDATA     DATA token?
836A: D0 02              BNE   :16
836C: 66 BF              ROR   MODDAT       bit 7 to 1 in MODDAT
836E: 48        :16      PHA
836F: 20 57 DB           JSR   OUTSPC
8372: 68                 PLA
8373: 48                 PHA
8374: 20 D5 83           JSR   LTOKEN       Print Applesoft token
8377: 68                 PLA
8378: C9 D5              CMP   #TOKUSR
837A: 20 C5 83           JSR   COMLISO
837D: B0 05              BCS   :17
837F: 84 85              STY   $85
8381: 20 5C DB           JSR   OUTDO
8384: 4C E8 82  :17      JMP   ]JLOOP
                * LIST a new BASIC statement
8387: 88        :23      DEY
8388: A5 BD              LDA   IDMOCL
838A: C9 0B              CMP   #OFFDEF-TOFFST
838C: 90 03              BCC   :39
838E: 66 C1              ROR   DEFFLG
8390: 18                 CLC
8391: 98        :39      TYA
8392: 65 B5              ADC   YSAV
8394: 85 B5              STA   YSAV
8396: 20 57 DB           JSR   OUTSPC
8399: A6 C2              LDX   OFFSET       Get offset from new keyword table
```

```
839B: BD 1F 9B  ]LOOP    LDA   TMOCL,X
839E: F0 11              BEQ   :29          End of keyword
83A0: 30 05              BMI   :27          Applesoft token: print it
83A2: 20 5C DB           JSR   OUTDO        Normal text to output
83A5: D0 07              BNE   :28          Always
83A7: 86 B4     :27      STX   XSAV         Save offset
83A9: 20 D5 83           JSR   LTOKEN       Print Applesoft token
83AC: A6 B4              LDX   XSAV
83AE: E8        :28      INX
83AF: D0 EA              BNE   ]LOOP        Always
83B1: A5 BD     :29      LDA   IDMOCL
83B3: C9 0A              CMP   #OFFUSR-TOFFST
83B5: 20 C5 83           JSR   COMLISO
83B8: B0 03              BCS   :30
83BA: 20 5C DB           JSR   OUTDO
83BD: 20 57 DB  :30      JSR   OUTSPC
83C0: A4 B5     :31      LDY   YSAV
83C2: 4C 1B 83           JMP   NCR

83C5: 38        COMLISO  SEC
83C6: D0 0C              BNE   :0
83C8: A4 B5              LDY   YSAV
83CA: C8                 INY
83CB: B1 9B              LDA   (LOWTR),Y
83CD: 20 5B 75           JSR   COMRSTC
83D0: B0 02              BCS   :0
83D2: 84 B5              STY   YSAV
83D4: 60        :0       RTS

                * Print Applesoft token
83D5: 38        LTOKEN   SEC
83D6: E9 7F              SBC   #$7F
83D8: AA                 TAX                ;Index in X reg
83D9: 84 85              STY   $85
83DB: A0 D0              LDY   #TOKTABL-256
83DD: 84 9D              STY   FAC
                * Line below is a substitute for LDY #>TOKTABL-256
83DF: 88                 DEY
83E0: 84 9E              STY   FAC+1
83E2: A0 FF              LDY   #$FF
83E4: CA        :1       DEX
83E5: F0 07              BEQ   :3
83E7: 20 2C D7  ]LOOP    JSR   $D72C
83EA: 10 FB              BPL   ]LOOP
83EC: 30 F6              BMI   :1
83EE: 20 2C D7  :3       JSR   $D72C
83F1: 30 05              BMI   :4
83F3: 20 5C DB           JSR   OUTDO
83F6: D0 F6              BNE   :3
83F8: A4 85     :4       LDY   $85
83FA: 4C 5C DB           JMP   OUTDO

83FD: D0 07     RRETURN  BNE   :0
83FF: A9 FF              LDA   #$FF
8401: 85 86              STA   FORPNT+1
8403: 4C 71 D9           JMP   $D971
8406: 60        :0       RTS
```

```

8407: A9 AB     RONERR   LDA   #TOKGOTO
8409: 20 CE 7D           JSR   NSYNCHR
840C: A5 B8              LDA   TXTPTR
840E: 85 F4              STA   TXTPSV
8410: A5 B9              LDA   TXTPTR+1
8412: 85 F5              STA   TXTPSV+1
8414: 38                 SEC
8415: 66 D8              ROR   ERRFLG
8417: A5 75              LDA   CURLIN
8419: 85 F6              STA   CURLSV
841B: A5 76              LDA   CURLIN+1
841D: 85 F7              STA   CURLSV+1
841F: 4C 95 D9           JMP   DATA

                * New FRMEVL processing
                         PUT   PEERAROMBA,D2
                TOKDIM   =     $86
                TOKFRE   =     $D6
                NEWGARBG EQU   $E484
                FREFAC   EQU   $E600
                ENDCHR   EQU   $0E
                STRNG1   EQU   $AC
                VPNT     EQU   $A0
                * When used in USR functions w 2 args, holdsin n
                * the first arg expression type
                GIVAYF   EQU   $E2F2
                SNGFLT   EQU   $E301
                MOVMF    EQU   $EB2B
                LEVELPAR EQU   IDMOCL

8422: 20 47 75  RDIM     JSR   RST100
8425: 20 4C 86           JSR   NCHKOPN
8428: 20 3E 79           JSR   NGETARPT
842B: A0 04              LDY   #4
842D: B1 9B              LDA   (LOWTR),Y
842F: 29 0F              AND   #$0F
8431: 48                 PHA
8432: A2 00              LDX   #0
8434: A1 B8              LDA   (TXTPTR,X)
8436: C9 2C              CMP   #´,´
8438: D0 29              BNE   :1
843A: A5 9C              LDA   LOWTR+1
843C: 48                 PHA
843D: A5 9B              LDA   LOWTR
843F: 48                 PHA
8440: 20 47 75           JSR   RST100
8443: 20 73 86           JSR   NGETBYT    Index of dimension in X&FACLO
8446: 8A                 TXA
8447: F0 24              BEQ   GOIQ
8449: 68                 PLA
844A: 85 9B              STA   LOWTR
844C: 68                 PLA
844D: 85 9C              STA   LOWTR+1
844F: 68                 PLA
8450: 38                 SEC
8451: E5 A1              SBC   FACLO
```

```
8453: 90 18               BCC   GOIQ
8455: 0A                  ASL                ;Incidently clears the carry
8456: 69 05               ADC   #5           Because of carry clear
8458: A8                  TAY
8459: B1 9B               LDA   (LOWTR),Y
845B: AA                  TAX
845C: C8                  INY
845D: B1 9B               LDA   (LOWTR),Y
845F: A8                  TAY
8460: 8A                  TXA
8461: 90 04               BCC   :0           Always
              :1          MPLY
8463: 68                  PLA
8464: A8                  TAY
8465: 8A                  TXA
8466: 38                  SEC
8467: 20 F2 E2    :0      JSR   GIVAYF
846A: 4C 46 86            JMP   NCHKCLS

846D: 4C 99 E1    GOIQ    JMP   GOIQERR      Raise a ILLEGAL QUANTITY ERROR

8470: 20 51 86    RVRAI   JSR   NFRMEVL      True: evaluate second argument
8473: 20 49 86            JSR   NCHKCOM      Skip the comma and 3rd expr.
8476: A9 29               LDA   #´)´         until end of function detected

              * This subroutine will skip program text until an
              * end character is scanned.
8478: 85 0E       SKIPC   STA   ENDCHR
847A: A0 00               LDY   #0
847C: 84 BD               STY   LEVELPAR     Parenthesis level
847E: 84 C0               STY   GFLAG        String litteral parsing flag
8480: 88                  DEY
8481: C8          ]LOOP   INY
8482: B1 B8               LDA   (TXTPTR),Y
8484: F0 36               BEQ   LGSYNERR
8486: C9 22               CMP   #´"´
8488: D0 08               BNE   :0
848A: A5 C0               LDA   GFLAG        Inverse GFLAG b7
848C: 49 80               EOR   #$80
848E: 85 C0               STA   GFLAG
8490: B0 EF               BCS   ]LOOP        Always
8492: 24 C0       :0      BIT   GFLAG        Within litteral string
8494: 30 EB               BMI   ]LOOP         so loop for next character.
8496: C9 3A               CMP   #´:´         End of instruction?
8498: F0 22               BEQ   LGSYNERR     SYNTAX ERROR if so
849A: C9 28               CMP   #´(´
849C: D0 04               BNE   :1
849E: E6 BD               INC   LEVELPAR
84A0: B0 DF               BCS   ]LOOP        Always
84A2: C9 29       :1      CMP   #´)´
84A4: D0 08               BNE   :2
84A6: A6 BD               LDX   LEVELPAR
84A8: F0 08               BEQ   :3
84AA: C6 BD               DEC   LEVELPAR
84AC: 10 D3               BPL   ]LOOP
84AE: A6 BD       :2      LDX   LEVELPAR
84B0: D0 CF               BNE   ]LOOP
```

```
84B2: C5 0E    >178  :3       CMP   ENDCHR
84B4: D0 CB    >179           BNE   ]LOOP
84B6: 20 98 D9 >180           JSR   ADDON      Add Y to TXTPTR
84B9: 4C 47 75 >181           JMP   RST100
               >182
84BC: 4C C9 DE >183  LGSYNERR JMP   SYNERR     Vector to SYNTAX ERROR
               >184
               >185  * Handles the IIF function
84BF: 20 49 86 >186  RIIF     JSR   NCHKCOM    Check for trailing comma
84C2: A6 9D    >187           LDX   FAC        True or false value?
84C4: D0 AA    >188           BNE   RVRAI      True: then skip second arg.
84C6: A9 2C    >189           LDA   #´,´
84C8: 20 78 84 >190           JSR   SKIPC      Skip 2nd expression
               >191  * Evaluate 3rd arg. and check for closing parenthesis
84CB: 4C 43 86 >192           JMP   NPARCHK+3
               >193
84CE: 20 51 86 >194  NFRMNUM  JSR   NFRMEVL    Get scalar valueH
84D1: 4C 6A DD >195           JMP   CHKNUM     Ensure numeric value
               >196
84D4: 4C F9 EA >197  ]LOOP    JMP   MOVFM
84D7: A0 00    >208  H16B     LDY   #0
84D9: B1 A0    >209           LDA   (VPNT),Y
84DB: 48       >210           PHA
84DC: 20 ED DE >211           JSR   $DEED
84DF: 68       >213           PLA
84E0: 20 68 86 >214           JSR   LBS81
84E3: 4C 59 85 >215           JMP   XSUITE
               >216
               >217  * Takes care of the ´@´ processing
               >218  * Refactor part of the FRMEVL ROM routine
84E6: 20 47 75 >219  FRMELMLP JSR   RST100
84E9: B0 0A    >220  FRMELM   BCS   :2         Branch iif not a digit
84EB: 20 4A EC >222  :1       JSR   $EC4A
84EE: A9 00    >223           LDA   #0
84F0: 85 C7    >224           STA   INTTYPSV
84F2: F0 67    >225           BEQ   RET3
84F4: 60       >226           RTS
84F5: C9 2E    >232  :2       CMP   #´.´
84F7: F0 F2    >233           BEQ   :1
84F9: 20 7D E0 >234           JSR   ISLETC
84FC: 90 60    >235           BCC   L3
84FE: AA       >236           TAX
84FF: 30 28    >237           BMI   :77
8501: C9 49    >238           CMP   #´I´
8503: F0 08    >239           BEQ   :80
8505: C9 4D    >240           CMP   #´M´
8507: F0 04    >241           BEQ   :80
8509: C9 54    >242           CMP   #´T´
850B: D0 1C    >243           BNE   :77
               >244  * Might be the IIF() function
850D: A2 02    >245  :80      LDX   #2
850F: 20 4E 82 >246           JSR   RECON1
8512: F0 15    >247           BEQ   :77
8514: 20 98 D9 >248           JSR   ADDON
8517: A5 BD    >249           LDA   IDMOCL
8519: 48       >250           PHA
851A: 20 4C 86 >251           JSR   NCHKOPN
```

```
851D: 20 CE 84             JSR   NFRMNUM    Get operand numeric value
8520: 68                   PLA              ;Recall IDMOCL from stack
8521: 38                   SEC
8522: E9 08                SBC   #OFFMOU-TOFFST
8524: 90 99                BCC   RIIF
               * Space for MOUSE and TIMER functions
               * ...: to be continued
8526: 4C 62 8C             JMP   MTFUNC
               * Alphabetic character: variable name
8529: A2 00    :77         LDX   #0
852B: 86 10                STX   DIMFLG
852D: A1 B8                LDA   (TXTPTR,X)
852F: 20 4C 79             JSR   NPTRGET1
               RFFVL       EQU   *-1
8532: 85 A0                STA   VPNT
8534: 84 A1                STY   VPNT+1
8536: A6 11                LDX   VALTYP
8538: F0 06                BEQ   :41
853A: A2 00                LDX   #0
853C: 86 AD                STX   STRNG1+1
853E: F0 19                BEQ   XSUITE     Always
8540: A6 12    :41         LDX   INTTYP
8542: 10 90                BPL   ]LOOP
8544: E0 81                CPX   #$81
8546: D0 8F                BNE   H16B       Branch if int16bit variable
8548: A2 00                LDX   #0
854A: A1 83                LDA   (VARPNT,X)
854C: 10 06                BPL   *+8
854E: 2C E7 9C             BIT   WMODE
8551: 30 01                BMI   *+3
8553: CA                   DEX              ;Poids fort dans X
8554: A8                   TAY              ;Poids faible dans Y
8555: 8A                   TXA              ;Poids fort dans A
8556: 20 F2 E2             JSR   GIVAYF     Convert A, Y to FP
8559: A5 11    XSUITE      LDA   VALTYP
855B: 85 C8    RET3        STA   VALTYPSV
855D: 60       ]RET        RTS

855E: C9 C8    L3          CMP   #TOKADD    Unary + operator: loop
8560: F0 84                BEQ   FRMELMLP
8562: C9 22                CMP   #´"´
8564: D0 0A                BNE   :4
8566: 20 81 DE             JSR   $DE81
8569: A9 FF                LDA   #$FF
856B: 30 EE                BMI   RET3       Always
856D: 4C CB 7E ]LOOP       JMP   RUSR
8570: C9 D5    :4          CMP   #TOKUSR
8572: F0 F9                BEQ   ]LOOP
8574: A2 03                LDX   #TOKMTIFE-TOKMOTIF-1
8576: DD 07 96 ]LOOP       CMP   TOKMOTIF,X
8579: D0 0B                BNE   :NOK
857B: A8                   TAY
857C: BD 0F 96             LDA   TOKMPFT,X
857F: 48                   PHA
8580: BD 0B 96             LDA   TOKMPFB,X
8583: 48                   PHA
8584: 98                   TYA
```

```
8585: 60       >323             RTS
8586: CA       >332  :NOK       DEX
8587: 10 ED    >333             BPL   ]LOOP
8589: C9 40    >334  :6         CMP   #´@´
858B: D0 10    >335             BNE   :78
858D: A5 C8    >336             LDA   VALTYPSV
858F: 85 11    >337             STA   VALTYP
8591: 30 04    >338             BMI   :60
8593: A5 C7    >339             LDA   INTTYPSV
8595: 85 12    >340             STA   INTTYP
8597: 4C 47 75 >341  :60        JMP   RST100
859A: 4C 22 84 >342  :79        JMP   RDIM
859D: C9 86    >343  :78        CMP   #TOKDIM
859F: F0 F9    >344             BEQ   :79
               >345
85A1: C9 D2    >346  :7         CMP   #TOKSGN
85A3: B0 18    >347             BCS   :10
85A5: C9 23    >348             CMP   #´#´
85A7: F0 03    >349             BEQ   *+5
85A9: 4C 40 86 >350             JMP   NPARCHK
               >351  * Handle the ´#´ pattern in a FOREACH loop
85AC: AC 23 96 >352             LDY   AEI
85AF: AD 24 96 >353             LDA   AEI+1
85B2: 48       >357             PHA
85B3: 20 F2 E2 >359             JSR   GIVAYF
85B6: 68       >363             PLA
85B7: 20 6B 86 >365             JSR   LBS80
85BA: 4C 47 75 >366             JMP   RST100
85BD: 0A       >367  :10        ASL
85BE: 48       >368             PHA
85BF: AA       >369             TAX
85C0: 20 47 75 >370             JSR   RST100
85C3: E0 CF    >371             CPX   #$CF
85C5: 90 14    >372             BCC   :11
85C7: 20 4C 86 >373             JSR   NCHKOPN
85CA: 20 51 86 >374             JSR   NFRMEVL
85CD: 20 49 86 >375             JSR   NCHKCOM
85D0: 20 6C DD >376             JSR   CHKSTR
85D3: 68       >377             PLA
85D4: AA       >377             TAX
85D5: 20 23 86 >378             JSR   COMCMPLX
85D8: 4C EB 85 >380             JMP   :14
85DB: 20 40 86 >384  :11        JSR   NPARCHK
85DE: 68       >385             PLA
85DF: A8       >385             TAY
85E0: C0 C8    >386             CPY   #TOKSTRD+TOKSTRD
85E2: F0 04    >387             BEQ   :15
85E4: C0 CE    >388             CPY   #TOKCHRD+TOKCHRD
85E6: D0 31    >389             BNE   :13
85E8: 20 32 86 >390  :15        JSR   CALLFUNC
85EB: A9 FF    >391  :14        LDA   #$FF
85ED: 85 C8    >392             STA   VALTYPSV
85EF: 60       >393  ]RET       RTS
85F0: A5 11    >394  ]LOOP      LDA   VALTYP
85F2: D0 1C    >395             BNE   :19
85F4: 18       >396             CLC
85F5: 20 83 77 >397             JSR   NROUT
```

```
85F8: A2 00             LDX   #0
85FA: A5 A0             LDA   FAC+3
85FC: D0 15             BNE   :2
85FE: A5 A1             LDA   FAC+4
8600: C9 01             CMP   #1
8602: D0 0F             BNE   :2
8604: A2 03             LDX   #3
8606: 20 20 7D          JSR   ZRTAUX
8609: A5 AE             LDA   STRNG2+1
860B: A4 AD             LDY   STRNG2
860D: 4C 63 86          JMP   NWGVAYF
8610: 20 00 E6 :19      JSR   FREFAC
8613: 20 84 E4 :2       JSR   NEWGARBG
8616: 4C 59 86          JMP   HE2E8

8619: C0 AC    :13      CPY   #TOKFRE+TOKFRE
861B: F0 D3             BEQ   ]LOOP
861D: 20 32 86          JSR   CALLFUNC
8620: 4C 6A DD          JMP   CHKNUM

               COMCMPLX DO    KOPT16
8623: A5 A1             LDA   FACLO
8625: 48                PHA
8626: A5 A0             LDA   FACMO
8628: 48                PHA
8629: 8A                TXA
862A: 48                PHA
862B: 20 73 86          JSR   NGETBYT
862E: 68                PLA
862F: A8                TAY
8630: 8A                TXA
8631: 48                PHA

8632: B9 DC CF CALLFUNC LDA   $CFDC,Y
8635: 85 91             STA   $91
8637: B9 DD CF          LDA   $CFDD,Y
863A: 85 92             STA   $92
863C: 20 90 00          JSR   $90
863F: 60                RTS

8640: 20 4C 86 NPARCHK  JSR   NCHKOPN
8643: 20 51 86          JSR   NFRMEVL

8646: A9 29    NCHKCLS  LDA   #´)´
8648: 2C                HEX   2C
8649: A9 2C    NCHKCOM  LDA   #´,´
864B: 2C                HEX   2C
864C: A9 28    NCHKOPN  LDA   #´(´
864E: 4C CE 7D          JMP   NSYNCHR

8651: 20 7B DD NFRMEVL  JSR   FRMEVL
8654: A5 11             LDA   VALTYP
8656: 85 C8             STA   VALTYPSV
8658: 60       ]RET     RTS

8659: 38       HE2E8    SEC
865A: A5 6F             LDA   FRETOP
```

```
865C: E5 6D               SBC   STREND
865E: A8                  TAY
865F: A5 70               LDA   FRETOP+1
8661: E5 6E               SBC   STREND+1
8663: 48        NWGVAYF   PHA
8664: 20 F2 E2            JSR   GIVAYF
8667: 68                  PLA
8668: 2D E7 9C  LBS81     AND   WMODE
866B: 10 EB     LBS80     BPL   ]RET
866D: 20 69 8E            JSR   GP65536
8670: 4C BE E7            JMP   FADD

8673: 20 F8 E6  NGETBYT   JSR   GETBYT
8676: 48                  PHA
8677: 20 10 77            JSR   SETITS
867A: 0A                  ASL
867B: 85 C8               STA   VALTYPSV
867D: 68                  PLA
867E: 60        MFIN      RTS

867F: 20 4C E7  ROUT11    JSR   COMBYTE    Get VTAB value in X
8682: 20 59 F2            JSR   $F259      Do the VTAB
8685: 20 4C E7            JSR   COMBYTE
8688: 20 EA F7            JSR   $F7EA      Do the HTAB
868B: 20 65 75            JSR   RST102
868E: F0 13               BEQ   :0
8690: 20 49 86            JSR   NCHKCOM
8693: A5 F1               LDA   $F1        Save current SPEED
8695: 48                  PHA
8696: A9 01               LDA   #1         Fastest speed..
8698: 85 F1               STA   $F1
869A: 20 65 75            JSR   RST102
869D: 20 D5 DA            JSR   $DAD5      Do the PRINT
86A0: 68                  PLA              ;restore original SPEED
86A1: 85 F1               STA   $F1
86A3: 60        :0        RTS

86A4: 20 49 86  ROUTGEN   JSR   NCHKCOM
86A7: 20 73 86            JSR   NGETBYT
86AA: 8A                  TXA
86AB: F0 1F               BEQ   ROUT0
86AD: E0 0B               CPX   #11
86AF: F0 CE               BEQ   ROUT11
86B1: E0 0A               CPX   #10
86B3: D0 03               BNE   :2
86B5: 4C 63 8A            JMP   ROUT10
86B8: E0 08     :2        CPX   #8
86BA: D0 03               BNE   :1
86BC: 4C 0B 92            JMP   ROUT8
86BF: E0 05     :1        CPX   #5
86C1: D0 03               BNE   :0
86C3: 4C A6 88            JMP   KILLEMAL
86C6: B0 B6     :0        BCS   MFIN
86C8: E0 04               CPX   #4
86CA: F0 3D               BEQ   ROUT4
86CC: A5 69     ROUT0     LDA   VARTAB
86CE: 85 06               STA   AUXPTR
```

```
86D0: A5 6A               LDA   VARTAB+1
86D2: 85 07               STA   AUXPTR+1

86D4: 20 65 75  ]LOOP     JSR   RST102
86D7: F0 A5               BEQ   MFIN
86D9: 20 49 86            JSR   NCHKCOM
86DC: 20 43 88            JSR   NPTRGETX
86DF: A5 9B               LDA   LOWTR
86E1: C5 06               CMP   AUXPTR
86E3: A5 9C               LDA   LOWTR+1
86E5: E5 07               SBC   AUXPTR+1
86E7: 90 95               BCC   MFIN
86E9: A0 00               LDY   #0
86EB: B1 9B     ]JLOOP    LDA   (LOWTR),Y
86ED: AA                  TAX
86EE: B1 06               LDA   (AUXPTR),Y
86F0: 91 9B               STA   (LOWTR),Y
86F2: 8A                  TXA
86F3: 91 06               STA   (AUXPTR),Y
86F5: C8                  INY
86F6: C0 07               CPY   #7
86F8: 90 F1               BCC   ]JLOOP
86FA: 18                  CLC
86FB: 98                  TYA
86FC: 65 06               ADC   AUXPTR
86FE: 85 06               STA   AUXPTR
8700: 90 D2               BCC   ]LOOP
8702: E6 07               INC   AUXPTR+1
8704: B0 CE               BCS   ]LOOP      Always

8706: 4C 76 DD  GGO2TMER  JMP   GOTMIERR

8709: A9 04     ROUT4     LDA   #4         Ensure enough room on stack
870B: 20 D6 D3            JSR   CHKMEM     7 bytes so 4 16bit words
870E: 68                  PLA              ;Pull return adress
870F: 68                  PLA
8710: 20 49 86            JSR   NCHKCOM
8713: 20 42 79            JSR   NPTRGTX
8716: 24 12               BIT   INTTYP
8718: 10 EC               BPL   GGO2TMER
871A: A5 9B               LDA   LOWTR
871C: C5 6B               CMP   ARYTAB
871E: 8D F3 95            STA   ITVADDR
8721: A5 9C               LDA   LOWTR+1
8723: 8D F4 95            STA   ITVADDR+1
8726: E5 6C               SBC   ARYTAB+1
8728: B0 DC               BCS   GGO2TMER
872A: A5 F8               LDA   REMSTK
872C: 8D F2 95            STA   SPROOT
                * Reinit the alive context markers
872F: A9 FF               LDA   #$FF
8731: A2 08               LDX   #TABOFT-TABOFB
8733: 9D E8 95  ]LOOP     STA   TABOFT-1,X
8736: CA                  DEX
8737: D0 FA               BNE   ]LOOP
8739: 86 C0               STX   IDX0       Starting index: 0
873B: 20 65 75  ]LOOP     JSR   RST102
```

```
873E: F0 0F              BEQ   XMFIN      End of instruction
8740: 20 49 86           JSR   NCHKCOM
8743: 20 6E 8E           JSR   NGTA2
8746: 90 31              BCC   XMFIN1
8748: 20 7C 87           JSR   LBS04
874B: E6 C0              INC   IDX0
874D: D0 EC              BNE   ]LOOP

874F: A5 C0     XMFIN    LDA   IDX0
8751: F0 22              BEQ   :0
8753: A9 80              LDA   #$80
8755: 8D DC 9C           STA   MTACTV
8758: 20 16 89           JSR   SETLTR
875B: 20 79 87           JSR   XMFIN1
875E: A2 00              LDX   #0
8760: 24 D8              BIT   ERRFLG
8762: 10 01              BPL   :1
8764: E8                 INX
8765: 8A        :1       TXA
8766: A0 1A              LDY   #26
8768: 91 9B              STA   (LOWTR),Y
876A: 20 7D 89           JSR   SAVERC
876D: A2 00              LDX   #0
876F: 8E F1 95           STX   INDX
8772: 4C C2 88           JMP   RESTOR1
8775: 60        :0       RTS

8776: 28        XMFIN2   PLP
8777: 68                 PLA
8778: 68                 PLA
8779: 4C 95 D9  XMFIN1   JMP   DATA

                * Handle a single entry (index in IDX0)
                LBS04
                * Array base address in (LOWTR, LOWTR+1)
877C: A6 C0              LDX   IDX0
877E: A5 9B              LDA   LOWTR
8780: 85 06              STA   AUXPTR
8782: E5 6B              SBC   ARYTAB     C already set
8784: 9D E1 95           STA   TABOFB,X
8787: 08                 PHP
8788: A5 9C              LDA   LOWTR+1
878A: 85 07              STA   AUXPTR+1
                * Is local error handling desired
878C: 20 49 86           JSR   NCHKCOM
878F: 20 F8 E6           JSR   GETBYT
                * Offset 24 for local error handling flag
8792: A0 1A              LDY   #26
8794: E0 02              CPX   #2
8796: D0 06              BNE   :0
8798: CA                 DEX
8799: 24 D8              BIT   ERRFLG
879B: 30 01              BMI   :0
879D: CA                 DEX
879E: 8A        :0       TXA
879F: 91 06              STA   (AUXPTR),Y
87A1: F0 0E              BEQ   :1
```

```
87A3: A0 19              LDY   #26-1
87A5: BE ED 95  ]LOOP    LDX   P0OFFSET-8,Y
87A8: B5 00              LDA   0,X
87AA: 91 06              STA   (AUXPTR),Y
87AC: 88                 DEY
87AD: E0 F4              CPX   #TXTPSV
87AF: D0 F4              BNE   ]LOOP
                * Offsets 27 and 28 for swapped in machine code routine
87B1: A9 1C     :1       LDA   #28
87B3: 20 2B 88           JSR   LBS041
                * Offsets 29 and 30 for swapped out machine code routine
87B6: A9 1E              LDA   #30
87B8: 20 2B 88           JSR   LBS041
87BB: 20 49 86           JSR   NCHKCOM
87BE: 20 0C DA           JSR   LINGET
87C1: 20 1A D6           JSR   FNDLIN
87C4: 90 B0              BCC   XMFIN2     Non existent line: exit
                * Offsets 0 and 1 for array name
                * Offsets 2 and 3 for offset to next array
                * Offset 4 for number of dimension
                * Offsets 5 and 6 for last dimension value
87C6: A0 04              LDY   #4
87C8: B1 06              LDA   (AUXPTR),Y
87CA: 49 41              EOR   #%01000001 Must be 16bits integer and
87CC: D0 A8              BNE   XMFIN2     # of dimensions must be 1
87CE: A5 07              LDA   AUXPTR+1
87D0: 28                 PLP              ;Restaure Carry from previous SBC
87D1: E5 6C              SBC   ARYTAB+1
87D3: A6 C0              LDX   IDX0
87D5: 9D E9 95           STA   TABOFT,X
                * Offset 7 and 8 for storing SP value
                * Integer variable value storage order
87D8: A0 07              LDY   #7
87DA: A9 00              LDA   #0
87DC: 91 06              STA   (AUXPTR),Y
87DE: C8                 INY
87DF: A5 F8              LDA   REMSTK
87E1: E9 07              SBC   #7         ;Carry already set
87E3: 91 06              STA   (AUXPTR),Y
87E5: C8                 INY
                * Offset 9 and 10 for LINNUM storage
                * (natural storage order)
87E6: A5 50              LDA   LINNUM
87E8: 91 06              STA   (AUXPTR),Y
87EA: C8                 INY
87EB: A5 51              LDA   LINNUM+1
87ED: 91 06              STA   (AUXPTR),Y
87EF: C8                 INY
                * Offset 11 and 12 for TXTPTR storage
                * (natural storage order)
87F0: A5 9B              LDA   LOWTR
87F2: 69 03              ADC   #4-1       Because Carry already set
87F4: 91 06              STA   (AUXPTR),Y
87F6: C8                 INY
87F7: A5 9C              LDA   LOWTR+1
87F9: 69 00              ADC   #0
87FB: 91 06              STA   (AUXPTR),Y
```

```
87FD: C8                  INY
                * Offset 13 and 14 for OLDTEXT storage
                * (natural storage order)
87FE: A5 9B               LDA   LOWTR
8800: 69 04               ADC   #4
8802: 91 06               STA   (AUXPTR),Y
8804: C8                  INY
8805: A5 9C               LDA   LOWTR+1
8807: 69 00               ADC   #0
8809: 91 06               STA   (AUXPTR),Y
880B: A0 1F               LDY   #31
                * Offsset 31 and above for stack content storage
                * from current SP to SPROOT
                * For the time being (init), prepare a GOSUB frame
880D: A9 B0               LDA   #TOKGOSUB
880F: A2 03               LDX   #3
8811: 91 06     ]JLOOP    STA   (AUXPTR),Y Do not mind calling CURLIN
8813: C8                  INY
8814: CA                  DEX
8815: D0 FA               BNE   ]JLOOP
8817: A5 79               LDA   OLDTPTR
8819: 91 06               STA   (AUXPTR),Y
881B: C8                  INY
881C: A5 7A               LDA   OLDTPTR+1
881E: 91 06               STA   (AUXPTR),Y
8820: C8                  INY
8821: A9 D1               LDA   #NEWSTT-1
8823: 91 06               STA   (AUXPTR),Y
8825: C8                  INY
8826: A9 D7               LDA   #>NEWSTT-1
8828: 91 06               STA   (AUXPTR),Y
882A: 60                  RTS

882B: 48        LBS041    PHA
882C: 20 49 86            JSR   NCHKCOM
882F: 20 67 DD            JSR   FRMNUM
8832: 20 52 E7            JSR   GETADR
8835: 68                  PLA
8836: A8                  TAY
8837: A5 51               LDA   LINNUM+1
8839: 91 06               STA   (AUXPTR),Y
883B: F0 05               BEQ   :0
883D: 88                  DEY
883E: A5 50               LDA   LINNUM
8840: 91 06               STA   (AUXPTR),Y
8842: 60        :0        RTS

                NPTRGETX  DO    KOPT-K65C02
8843: A2 00               LDX   #0
8845: 86 82               STX   VARNAM+1
8847: 20 C8 7D            JSR   MISLETC
884A: 85 81               STA   VARNAM
884C: 20 47 75            JSR   RST100
884F: 90 05               BCC   :0
8851: 20 7D E0            JSR   ISLETC
8854: 90 16               BCC   :3
8856: 85 82     :0        STA   VARNAM+1
```

```
8858: 20 47 75  1037 ]LOOP    JSR   RST100
885B: 90 FB     1038          BCC   ]LOOP
885D: 20 7D E0  1039          JSR   ISLETC
8860: B0 F6     1040          BCS   ]LOOP
8862: 90 08     1041          BCC   :3
8864: 20 97 81  1042 :2       JSR   DECTPTR
8867: A6 81     1043          LDX   VARNAM
8869: BD 55 9B  1044          LDA   TYPLET-´A´,X
886C: A2 03     1046 :3       LDX   #3
886E: 20 8E 81  1050          JSR   XFROMMOT+2
8871: D0 F1     1051          BNE   :2
8873: 4C 72 81  1052          JMP   ROUT1Y
                1053
8876: 2C DC 9C  1054 RNEWISUI BIT   MTACTV
8879: 10 41     1055          BPL   RESTORD
                1056
                1057          PUT   PEERMTK
               >1    * Main Active MT entry point
887B: BA       >2    RMTCTRL  TSX              ;Test for an exhausted thread?
887C: EC F2 95 >3             CPX   SPROOT
887F: AE F1 95 >4             LDX   INDX
8882: 90 07    >5             BCC   :2
8884: A9 FF    >6             LDA   #$FF       Mark the current thread
8886: 9D E9 95 >7             STA   TABOFT,X    before switching to another
8889: B0 13    >8             BCS   KX3        Always branch
888B: 2C DA 9C >9    :2       BIT   INHACTV
888E: 30 2C    >10            BMI   RESTORD
8890: CE DB 9C >11            DEC   CTRACTV    Time for a context switch?
8893: D0 27    >12            BNE   RESTORD    Not yet
8895: BD E9 95 >13            LDA   TABOFT,X   Get BASIC array where to save
8898: 20 38 89 >14            JSR   NEXTC2      content
889B: 20 46 89 >18            JSR   SAVER      Perform the SAVE
889E: AE F1 95 >23   KX3      LDX   INDX       Get back the new context index
88A1: 20 1F 89 >25            JSR   NEXTCTX    Search for a new context index
88A4: 90 26    >26            BCC   RESTOR2    Found one
               >27   * Restore context from calling BASIC line
88A6: 20 16 89 >28   KILLEMAL JSR   SETLTR     Restore context from calling
88A9: 20 02 89 >29            JSR   RESTORC     BASIC line
88AC: AE F2 95 >30            LDX   SPROOT
88AF: 86 F8    >31            STX   REMSTK
88B1: 20 B8 88 >32            JSR   R0
88B4: 9A       >33            TXS
88B5: 4C D2 D7 >34            JMP   NEWSTT
88B8: 4E DC 9C >35   R0       LSR   MTACTV
88BB: 60       >36            RTS
               >37
88BC: 20 5A 8B >38   RESTORD  JSR   LBS10
88BF: 4C 20 D8 >39            JMP   $D820
               >40   * General purpose restore routine
               >41   * Input: X register index of context
88C2: BD E9 95 >42   RESTOR1  LDA   TABOFT,X
88C5: C9 FF    >43            CMP   #$FF       Safe guard: do not restore a
88C7: F0 38    >44            BEQ   RESTORF    terminated thread..
88C9: 20 38 89 >45            JSR   NEXTC2
               >46
               >47   * Input from caller: X: context index
88CC: AD DD 9C >48   RESTOR2  LDA   ICTRACTV   Reinit counter
```

```
88CF: 8D DB 9C >49              STA   CTRACTV     value
               >50     * Update ITHREAD% variable value
88D2: AD F4 95 >51              LDA   ITVADDR+1
88D5: F0 0C    >52              BEQ   RESTOR      Skip if no var. defined
88D7: 85 07    >53              STA   AUXPTR+1
88D9: AD F3 95 >54              LDA   ITVADDR
88DC: 85 06    >55              STA   AUXPTR
88DE: 8A       >56              TXA
88DF: A0 03    >57              LDY   #3
88E1: 91 06    >58              STA   (AUXPTR),Y
88E3: 18       >59     RESTOR   CLC
88E4: A0 1C    >60              LDY   #28         Trigger the page in routine if
88E6: 20 61 89 >61              JSR   SWPIO        defined
88E9: B0 B3    >65              BCS   KX3
               >66     * Do the RESTOR itself
               >67     * Input: LOWTR: Array base address
88EB: 20 02 89 >68              JSR   RESTORC
               >69     * Do the Stack restore
88EE: A0 1F    >70              LDY   #31         From offset 31 within context
88F0: A6 F8    >71              LDX   REMSTK       array storage
88F2: 9A       >72     RESTORX  TXS
88F3: EC F2 95 >73     ]LOOP    CPX   SPROOT      Until SPROOT value is reached
88F6: B0 C4    >74              BCS   RESTORD
88F8: E8       >75              INX
88F9: B1 9B    >76              LDA   (LOWTR),Y
88FB: 9D 00 01 >77              STA   $0100,X
88FE: C8       >78              INY
88FF: 90 F2    >79              BCC   ]LOOP       Always
8901: 60       >80     RESTORF  RTS
               >81
8902: 20 70 89 >83     RESTORC  JSR   LBS06
8905: 90 02    >84              BCC   *+4
8907: 85 D8    >85              STA   ERRFLG
8909: B1 9B    >93     ]LOOP    LDA   (LOWTR),Y
890B: BE ED 95 >94              LDX   P0OFFSET-8,Y
890E: 95 00    >95              STA   0,X
8910: 88       >96              DEY
8911: E0 F8    >97              CPX   #REMSTK
8913: D0 F4    >98              BNE   ]LOOP
8915: 60       >99              RTS
               >100
               >101    * Subroutine to get the context storage index for
               >102    * global (i.e. Perrsoft MT kernel calling line)
8916: A9 C6    >103    SETLTR   LDA   #SVPTR-8
8918: 85 9B    >104             STA   LOWTR
891A: A9 95    >105             LDA   #>SVPTR-8
891C: 85 9C    >106             STA   LOWTR+1
891E: 60       >107             RTS
               >108    * Subroutine to get the next context after the current one
               >109    * (index in X).
891F: A0 00    >110    NEXTCTX  LDY   #0          ctr. to avoid counting too far
8921: E8       >111    ]LOOP    INX               ;Wrap around the context ptr
8922: E0 08    >112             CPX   #TABOFT-TABOFB area..
8924: 90 02    >113             BCC   :0
8926: A2 00    >114             LDX   #0          Perform wrap...
8928: BD E9 95 >115    :0       LDA   TABOFT,X
892B: C9 FF    >116             CMP   #$FF        Got an active one (iif <> $FF)
```

```
892D: D0 06             BNE   :1         Yes...
892F: C8                INY              ;Bump counter
8930: C0 08             CPY   #TABOFT-TABOFB till all scanned
8932: 90 ED             BCC   ]LOOP      Not yet: see next context ptr
8934: 60                RTS              ;Exit with carry set..
8935: 8E F1 95 :1       STX   INDX       Memorize the new context index
8938: A8       NEXTC2   TAY              ;From offset to absolute address
8939: BD E1 95          LDA   TABOFB,X   by adding the ARYTAB base address
893C: 65 6B             ADC   ARYTAB     for arrays within Applesoft
893E: 85 9B             STA   LOWTR
8940: 98                TYA
8941: 65 6C             ADC   ARYTAB+1
8943: 85 9C             STA   LOWTR+1    Result in LOWTR pointer..
8945: 60                RTS              ;Exit with carry clear (always)

               * Save the context into BASIC array
               * Input: LOWTR: array base address
8946: 20 7D 89 SAVER    JSR   SAVERC
8949: A0 1E             LDY   #30        Possible trigger for page out
894B: 20 61 89          JSR   SWPIO       event...
               * Now it´s time to save the stack extension
894E: A0 1F             LDY   #31
               * As a subroutine, do not depend on current stack ptr.
               * But rather on memorized stack ptr. (within exec loop)
8950: A6 F8             LDX   REMSTK
8952: EC F2 95 ]LOOP    CPX   SPROOT
8955: B0 09             BCS   :0
8957: E8                INX
8958: BD 00 01          LDA   $0100,X
895B: 91 9B             STA   (LOWTR),Y
895D: C8                INY
895E: 90 F2             BCC   ]LOOP
8960: 60       :0       RTS

               * Routine to possibly trigger page in/page out routine
               * for every configured coroutine. Inputs are:
               * LOWTR: context array base address
               * Y either 30 or 28 for page in/out event
8961: B1 9B    SWPIO    LDA   (LOWTR),Y
8963: F0 0A             BEQ   :0         No routine defined
8965: 85 07             STA   AUXPTR+1
8967: 88                DEY
8968: B1 9B             LDA   (LOWTR),Y
896A: 85 06             STA   AUXPTR
               * Called routine must preserve registers
896C: 6C 06 00          JMP   (AUXPTR)
896F: 60       :0       RTS

8970: A0 1A    LBS06    LDY   #26
8972: B1 9B    LBS061   LDA   (LOWTR),Y
8974: D0 04             BNE   :0
8976: 38                SEC
8977: A0 0E    :1       LDY   #PIOFFSET-P0OFFSET+8-1
8979: 60                RTS
897A: 18       :0       CLC
897B: 88                DEY               ;Shortcut for
897C: 60                RTS              ; LDY #PEOFFSET-P0OFFSET+8-1
```

```

897D: 20 70 89 SAVERC   JSR   LBS06
8980: BE ED 95 ]LOOP    LDX   P0OFFSET-8,Y
8983: B5 00             LDA   0,X         Value to save
8985: 91 9B             STA   (LOWTR),Y
8987: 88                DEY
8988: E0 F8             CPX   #REMSTK
898A: D0 F4             BNE   ]LOOP
898C: 60                RTS

                        PUT   PEERMOUSTIME
               * Base addresses for mouse interface
               BAXLO    EQU   $0478       X low
               BAYLO    EQU   $04F8       Y low
               BAXHI    EQU   $0578       X high
               BAYHI    EQU   $05F8       Y high
               BAMBS    EQU   $0778       Button status

               TRACE    EQU   $D805
               IRQV     EQU   $03FE       Page 3 Interrupt vector

               * Reason codes for entering Mouse interface
               RSETM    =     0
               RSRVM    =     1
               RREAD    =     2
               RCLR     =     3
               RPOS     =     4
               RCLM     =     5
               RHOM     =     6
               RINI     =     7

               CONINT   EQU   $E6FB       FAC to single byte

               * Interrupt servicing routine
898D: A2 01    IRQHDLR  LDX   #RSRVM
898F: 20 49 8C          JSR   TOMOUSE
8992: B0 3F             BCS   :2          ; Not from mouse or spurious
8994: AE CE 9C          LDX   MOSL
8997: BD 78 07          LDA   BAMBS,X
899A: 4A                LSR
               * Movement interrupt bit into b0 and
               * button bit into b1, VBL interrupt bit
               * into b2
899B: 29 07             AND   #7          mask out other bits
899D: AA                TAX
899E: BD C7 99          LDA   MSTATUS,X  Get internal status
89A1: 8D D1 99          STA   WORKPL1
89A4: A2 02             LDX   #RREAD
89A6: 20 49 8C          JSR   TOMOUSE
89A9: 2C D1 99          BIT   WORKPL1
89AC: 10 1B             BPL   :1
               * Decrement runtime counter
89AE: AE F7 99          LDX   TIINC
89B1: D0 03             BNE   :01
89B3: CE F8 99          DEC   TIINC+1
89B6: CA       :01      DEX
89B7: 8E F7 99          STX   TIINC
```

```
89BA: D0 05    >60              BNE   :02
89BC: AD F8 99 >61              LDA   TIINC+1
89BF: F0 23    >62              BEQ   :00
               >63   :02
89C1: A9 7F    >69              LDA   #$7F
89C3: 2D D1 99 >70              AND   WORKPL1
89C6: 8D D1 99 >71              STA   WORKPL1
89C9: AD D1 99 >73   :1         LDA   WORKPL1
89CC: 0D D2 99 >77              ORA   MIRQST
89CF: 8D D2 99 >78              STA   MIRQST
89D2: 40       >80   ]LOOP      RTI
               >81
               >82   * No spurious interrupt is fatal to us..
               >83   * I´m afraid of no ghosts.... ;-)
89D3: AD D0 99 >84   :2         LDA   OLDVECT+1
89D6: C9 FF    >85              CMP   #>$FF65
89D8: D0 07    >86              BNE   :20
89DA: AD CF 99 >87              LDA   OLDVECT
89DD: C9 65    >88              CMP   #$FF65
89DF: F0 F1    >89              BEQ   ]LOOP
89E1: 6C CF 99 >90   :20        JMP   (OLDVECT)
               >91
89E4: AD F5 99 >94   :00        LDA   KTINC
89E7: 8D F7 99 >95              STA   TIINC
89EA: AD F6 99 >96              LDA   KTINC+1
89ED: 8D F8 99 >97              STA   TIINC+1
89F0: 4C C9 89 >101             JMP   :1
               >104
               >105  * Install new IRQ handler and save the original handler
               >106  * to build a daisy chain..
               >107  * Nouveau mode dans MOMODE
89F3: AD B8 99 >108  INSIRQV    LDA   MOMODE
89F6: C9 02    >109             CMP   #2
89F8: 90 20    >110             BCC   :1
89FA: AD FE 03 >127             LDA   IRQV
89FD: AE FF 03 >128             LDX   IRQV+1
8A00: C9 8D    >129             CMP   #IRQHDLR
8A02: D0 04    >130             BNE   :0
8A04: E0 89    >131             CPX   #>IRQHDLR
8A06: F0 12    >132             BEQ   :1
8A08: 78       >133  :0         SEI
8A09: 8D CF 99 >134             STA   OLDVECT
8A0C: 8E D0 99 >135             STX   OLDVECT+1
8A0F: A9 8D    >136             LDA   #IRQHDLR
8A11: 8D FE 03 >136             STA   IRQV
8A14: A9 89    >136             LDA   #>IRQHDLR
8A16: 8D FF 03 >136             STA   IRQV+1
8A19: 58       >138             CLI
8A1A: 60       >139  :1         RTS
               >140
               >141  * Deinstall IRQ handler
8A1B: AD B8 99 >142  DINSIRQV LDA   MOMODE
8A1E: C9 02    >143             CMP   #2
8A20: B0 14    >144             BCS   :1
8A22: 78       >145             SEI
8A23: AD D0 99 >159             LDA   OLDVECT+1
8A26: F0 0E    >160             BEQ   :1
```

```
8A28: 8D FF 03           STA   IRQV+1
8A2B: A9 00              LDA   #0
8A2D: 8D D0 99           STA   OLDVECT+1
8A30: AD CF 99           LDA   OLDVECT
8A33: 8D FE 03           STA   IRQV
8A36: 60        :1       RTS

8A37: 48        CMPCLAMP PHA
                * X/Y min% expression
8A38: 20 04 8B           JSR   NEVAL
8A3B: 8D 78 05           STA   $0578
8A3E: 8C 78 04           STY   $0478
                * X/Y max% expression
8A41: 20 04 8B           JSR   NEVAL
8A44: 8D F8 05           STA   $05F8
8A47: 8C F8 04           STY   $04F8
8A4A: 68                 PLA
8A4B: A2 05              LDX   #RCLM
8A4D: 4C 49 8C           JMP   TOMOUSE

8A50: C5 A1     IVALARG  CMP   FAC+4
8A52: 90 01              BCC   *+3
8A54: 60                 RTS
8A55: 68                 PLA
8A56: 68                 PLA
8A57: 4C 99 E1  ]ERR     JMP   $E199      Illegal quantity error

8A5A: A9 00     COMCLAMP LDA   #0
8A5C: 20 37 8A           JSR   CMPCLAMP
8A5F: A9 01              LDA   #1
8A61: D0 D4              BNE   CMPCLAMP

8A63: 20 49 86  ROUT10   JSR   NCHKCOM
8A66: 20 73 86           JSR   NGETBYT    Get reason code in X reg.
8A69: CA                 DEX
8A6A: CA                 DEX
8A6B: 30 EA              BMI   ]ERR
8A6D: E0 05              CPX   #5
8A6F: B0 E6              BCS   ]ERR
8A71: 20 DA 8D           JSR   ISMOUSH
8A74: AD B8 99           LDA   MOMODE
8A77: 29 0F              AND   #$F
8A79: D0 05              BNE   :1
8A7B: A2 25              LDX   #37
8A7D: 4C E9 8D           JMP   NERRH
                * Only READ (2), CLEAR (3), POS(4), CLAMP (5) and HOME (6)
                * reason codes are valid.
8A80: 8A        :1       TXA
8A81: F0 11              BEQ   COMREAD
8A83: CA                 DEX
8A84: F0 09              BEQ   COMCLEAR
8A86: CA                 DEX
8A87: F0 39              BEQ   COMPOS
8A89: CA                 DEX
8A8A: F0 CE              BEQ   COMCLAMP
8A8C: A2 06              LDX   #RHOM
8A8E: 2C                 HEX   2C         Skip next two bytes
```

```
8A8F: A2 C2     COMCLEAR LDX   #RCLEAR
8A91: 4C 49 8C  FINMOUSE JMP   TOMOUSE

8A94: AE D4 99  COMREAD  LDX   MODERUN
8A97: D0 05              BNE   :1
8A99: A2 02              LDX   #RREAD
8A9B: 20 49 8C           JSR   TOMOUSE
                * Handles X% host variable
8A9E: AE CE 9C  :1       LDX   MOSL
8AA1: BD 78 05           LDA   BAXHI,X
8AA4: 20 DE 8A           JSR   NPTRG
8AA7: BD 78 04           LDA   BAXLO,X
8AAA: 91 83              STA   (VARPNT),Y
                * Handle Y% host variable
8AAC: BD F8 05           LDA   BAYHI,X
8AAF: 20 DE 8A           JSR   NPTRG
8AB2: BD F8 04           LDA   BAYLO,X
8AB5: 91 83              STA   (VARPNT),Y
                * Handle S% for button status variable
8AB7: A9 00              LDA   #0
8AB9: 20 DE 8A           JSR   NPTRG
8ABC: BD 78 07           LDA   BAMBS,X
8ABF: 91 83              STA   (VARPNT),Y
8AC1: 60                 RTS

                COMPOS
                * X% expression
8AC2: 20 04 8B           JSR   NEVAL
8AC5: 9D 78 05           STA   BAXHI,X
8AC8: 98                 TYA
8AC9: 9D 78 04           STA   BAXLO,X
                * Y% expression
8ACC: 20 04 8B           JSR   NEVAL
8ACF: 9D F8 05           STA   BAYHI,X
8AD2: 98                 TYA
8AD3: 9D F8 04           STA   BAYLO,X
8AD6: A2 04              LDX   #RPOS
8AD8: 4C 91 8A           JMP   FINMOUSE

8ADB: 4C 76 DD  ]ERR     JMP   GOTMIERR   TYPE MISMATCH ERROR
8ADE: 48        NPTRG    PHA
8ADF: 20 49 86           JSR   NCHKCOM
8AE2: 20 42 79           JSR   NPTRGTX
8AE5: A5 12              LDA   INTTYP
8AE7: 10 F2              BPL   ]ERR
8AE9: 29 0F              AND   #15        cater for integer subtypes
8AEB: F0 04              BEQ   :1         only $80 and $82 are valid
8AED: C9 02              CMP   #2
8AEF: D0 EA              BNE   ]ERR
8AF1: AE CE 9C  :1       LDX   MOSL
8AF4: 68                 PLA
8AF5: A0 00              LDY   #0
8AF7: 91 83              STA   (VARPNT),Y
8AF9: C8                 INY
8AFA: 60                 RTS

                * Result in FAC+3, FAC+4
```

```
8AFB: 20 49 86  NEVALC   JSR   NCHKCOM
8AFE: 20 CE 84           JSR   NFRMNUM
8B01: 4C 83 77           JMP   NROUT      Replac. for ROUND.FAC/AYINT

8B04: 20 FB 8A  NEVAL    JSR   NEVALC
8B07: A5 A0              LDA   FAC+3
8B09: A4 A1              LDY   FAC+4
8B0B: AE CE 9C           LDX   MOSL
8B0E: 60        ]RET     RTS

                * Common subroutine for parsing new tokens
                * X upon entry: 0: updates TXTPTR if token found
                * 1: skip updating TXTPTR even when token found
8B0F: 86 C0     COMLBS   STX   GFLAG
8B11: A2 00              LDX   #0
8B13: A1 B8              LDA   (TXTPTR,X)
8B15: 30 19              BMI   :2
8B17: C9 4D              CMP   #´M´
8B19: F0 04              BEQ   :1
8B1B: C9 54              CMP   #´T´
8B1D: D0 11              BNE   :2
8B1F: A2 03     :1       LDX   #3
8B21: 20 4E 82           JSR   RECON1
8B24: F0 E8              BEQ   ]RET
8B26: 20 2B 8C           JSR   COMINT4    Check mouse hardware/reinit
8B29: A6 C0              LDX   GFLAG
8B2B: D0 E1              BNE   ]RET
8B2D: 4C 98 D9           JMP   ADDON      will exit with Z flag clear
                :2
8B30: 8A                 TXA
8B31: 60        ]RET     RTS

                * New instructions handling
                * for MOUSE and TIMER instructions
8B32: 4C 65 75  ]LOOP    JMP   RST102
8B35: 68        ]ERR1    PLA              ;Pull IDMOCL from stack
8B36: 68                 PLA              ;Pull return address
8B37: 68                 PLA
8B38: 4C C9 DE  ]ERR     JMP   SYNERR
                * MOUSE/TIMER STOP handler
8B3B: C0 09     ]JLOOP   CPY   #OFFTIM-TOFFST
8B3D: A2 00              LDX   #0
8B3F: 90 01              BCC   *+3        Branch iif MOUSE
8B41: E8                 INX
8B42: AD B8 99           LDA   MOMODE
8B45: 3D C3 99           AND   MOETMSK,X
                * Compare to minimum allowable value
8B48: DD C5 99           CMP   MOCMPVAL,X
8B4B: B0 05              BCS   :0         OK iif greater or equal
8B4D: A2 25              LDX   #37
8B4F: 4C E9 8D           JMP   NERRH
8B52: A9 01     :0       LDA   #1         Update MODEPEC configuration
8B54: 9D D6 99           STA   MODEPEC,X
8B57: 4C D2 D7           JMP   NEWSTT
8B5A: A2 00     LBS10    LDX   #0
8B5C: 20 0F 8B           JSR   COMLBS
8B5F: F0 D1              BEQ   ]LOOP
```

```
8B61: A5 BD    >350          LDA   IDMOCL
8B63: 48       >351          PHA
8B64: A0 00    >356          LDY   #0
8B66: B1 B8    >357          LDA   (TXTPTR),Y
8B68: C8       >358          INY
8B69: C9 B3    >360          CMP   #$B3       STOP token?
8B6B: F0 0F    >361          BEQ   :3
8B6D: C9 B4    >362          CMP   #$B4
8B6F: F0 0B    >363          BEQ   :3         ON token?
8B71: C9 4F    >364          CMP   #´O´
8B73: D0 C0    >365          BNE   ]ERR1
8B75: A2 05    >366          LDX   #5         Look up possible OFF pattern
8B77: 20 4E 82 >367          JSR   RECON1
8B7A: F0 B9    >368          BEQ   ]ERR1
8B7C: AA       >369  :3      TAX              ;X STOP/ON token or 0 (OFF)
8B7D: 86 B4    >370          STX   XSAV
8B7F: 20 98 D9 >371          JSR   ADDON
8B82: 68       >372          PLA
8B83: A8       >372          TAY
8B84: 68       >373          PLA
8B85: 68       >374          PLA
8B86: 20 65 75 >375          JSR   RST102
8B89: F0 19    >376          BEQ   :23        If EOI found
8B8B: E0 B4    >377          CPX   #$B4
8B8D: D0 A9    >378          BNE   ]ERR       SYNTAX ERR if not ON nor EOI
8B8F: 8A       >379          TXA
8B90: 48       >379          PHA
8B91: 98       >380          TYA
8B92: 48       >380          PHA
8B93: 20 FB 8A >381          JSR   NEVALC     Get factor/mode value after comma
8B96: 68       >382          PLA
8B97: A8       >382          TAY
8B98: 68       >383          PLA
8B99: AA       >383          TAX
8B9A: 86 B4    >384          STX   XSAV
8B9C: C0 08    >385          CPY   #OFFMOU-TOFFST
8B9E: D0 06    >386          BNE   :20
8BA0: 20 FE E6 >387          JSR   $E6FE      FAC integer -> single byte
8BA3: 2C       >388          HEX   2C
8BA4: A2 01    >389  :23     LDX   #1
8BA6: 86 C0    >390  :20     STX   GFLAG
8BA8: 84 BD    >391          STY   IDMOCL
8BAA: A5 B4    >392          LDA   XSAV        A: ON/OFF/STOP index
8BAC: C9 B3    >393          CMP   #$B3       STOP token?
8BAE: F0 8B    >394          BEQ   ]JLOOP
               >395  * IDMOCL in page zero, STOP/ON/OFF indic. in A reg.
8BB0: A6 BD    >396          LDX   IDMOCL
8BB2: E0 08    >397          CPX   #OFFMOU-TOFFST
8BB4: D0 42    >398          BNE   TIMEINST
               >399
               >400  * Mouse event handler
8BB6: C9 B4    >401          CMP   #$B4       MOUSE ON?
8BB8: D0 04    >402          BNE   *+6        No
8BBA: A2 00    >403          LDX   #0
8BBC: F0 0D    >404          BEQ   :8
8BBE: A2 07    >405          LDX   #7
8BC0: E4 C0    >406  ]LOOP   CPX   GFLAG
```

```
8BC2: F0 07    >407          BEQ   :8
8BC4: CA       >408          DEX
8BC5: CA       >409          DEX
8BC6: 10 F8    >410          BPL   ]LOOP
8BC8: 4C E7 8D >411  ]LOOP    JMP   NILLM
8BCB: AD B8 99 >419  :8       LDA   MOMODE
8BCE: 29 F8    >420          AND   #%11111000
8BD0: E0 02    >421          CPX   #2
8BD2: 86 C0    >422          STX   GFLAG
8BD4: 05 C0    >423          ORA   GFLAG
8BD6: 8D B8 99 >424          STA   MOMODE
8BD9: A9 00    >426          LDA   #0
8BDB: A8       >427          TAY
8BDC: 90 02    >428          BCC   *+4
8BDE: A9 02    >429  COMMON9  LDA   #2
8BE0: 99 D6 99 >430          STA   MODEPEC,Y
8BE3: AD B8 99 >431  COMMON   LDA   MOMODE
8BE6: 48       >432          PHA
8BE7: 20 F3 89 >433          JSR   INSIRQV
8BEA: 68       >434          PLA
8BEB: A2 00    >435          LDX   #RSETM
8BED: 20 49 8C >436          JSR   TOMOUSE
8BF0: B0 D6    >437          BCS   ]LOOP
8BF2: 20 1B 8A >438          JSR   DINSIRQV
8BF5: 4C D2 D7 >439          JMP   NEWSTT
               >440
8BF8: C9 B4    >441  TIMEINST CMP   #$B4       TIMER ON
8BFA: AD B8 99 >448          LDA   MOMODE
8BFD: B0 04    >449          BCS   *+6        Yes
8BFF: 29 07    >450          AND   #7
8C01: 90 02    >451          BCC   *+4        Always
8C03: 09 08    >452          ORA   #8
8C05: 8D B8 99 >453          STA   MOMODE
8C08: 90 D9    >454          BCC   COMMON
8C0A: 24 C0    >456          BIT   GFLAG
8C0C: 30 06    >457          BMI   *+8
8C0E: A2 01    >458          LDX   #1
8C10: A0 00    >459          LDY   #0
8C12: 10 04    >460          BPL   *+6        Always
8C14: A6 A1    >461          LDX   FAC+4
8C16: A4 A0    >462          LDY   FAC+3
8C18: 08       >463          PHP
8C19: 78       >464          SEI
8C1A: 8C F6 99 >465          STY   KTINC+1
8C1D: 8E F5 99 >466          STX   KTINC
8C20: 8C F8 99 >467          STY   TIINC+1
8C23: 8E F7 99 >468          STX   TIINC
8C26: 28       >469          PLP
8C27: A0 01    >470          LDY   #1
8C29: B0 B3    >471          BCS   COMMON9    Always
               >472
               >473  * Do we have suitable mouse hardware?
8C2B: 20 DA 8D >474  COMINT4  JSR   ISMOUSH    Fall into SWREINIT if yes
               >475  * Routine below to check whether we should init the
               >476  * MOUSE system?
               >477  SWREINIT
8C2E: 2C C1 99 >483          BIT   MONU
```

```
8C31: 30 12    >484            BMI   :0
8C33: 38       >485            SEC
8C34: 6E C1 99 >486            ROR   MONU
               >488  * INITMOUSE was performed on Peersoft boot when in an
               >489  * Apple 2,2+ host.
8C37: AD ED 9C >490            LDA   MACHINE
8C3A: F0 09    >491            BEQ   :0
8C3C: 98       >492            TYA
8C3D: 48       >492            PHA
8C3E: A2 07    >493            LDX   #RINI
8C40: 20 49 8C >494            JSR   TOMOUSE
8C43: 68       >495            PLA
8C44: A8       >495            TAY
8C45: 60       >496  :0        RTS
               >497
8C46: 6C B6 99 >498  ]LOOP     JMP   (MVECTOR)
               >499
8C49: BC AD 99 >500  TOMOUSE   LDY   OM_DEB,X
8C4C: AE B7 99 >501            LDX   MOCN
8C4F: 08       >502            PHP
8C50: 78       >503            SEI
8C51: 8C B6 99 >504            STY   MVECTOR
8C54: AC B5 99 >505            LDY   MON0
8C57: 20 46 8C >506            JSR   ]LOOP
8C5A: B0 03    >507            BCS   *+5
8C5C: 28       >508            PLP
8C5D: 18       >509            CLC
8C5E: 60       >510            RTS
8C5F: 28       >511            PLP
8C60: 38       >512            SEC
8C61: 60       >513            RTS
               >514
               >515  * Entry routine for MOUSE functions (either MOUSE or
               >516  * TIMER).
8C62: 48       >517  MTFUNC    PHA
8C63: 20 FB E6 >518            JSR   CONINT
8C66: 20 46 86 >519            JSR   NCHKCLS
8C69: 20 2B 8C >520            JSR   COMINT4
8C6C: 68       >521            PLA
8C6D: D0 32    >522            BNE   TFUNC
8C6F: A9 02    >523            LDA   #2
8C71: 20 50 8A >524            JSR   IVALARG
8C74: AE D4 99 >525            LDX   MODERUN
8C77: D0 05    >526            BNE   *+7         Branch if within interrupt
8C79: A2 02    >527            LDX   #RREAD
8C7B: 20 49 8C >528            JSR   TOMOUSE
8C7E: AE CE 9C >529            LDX   MOSL
8C81: A4 A1    >534            LDY   FAC+4
8C83: 88       >535            DEY
8C84: 10 09    >537            BPL   :1
8C86: BD 78 05 >538            LDA   BAXHI,X     MOUSE(0) means read X
8C89: BC 78 04 >539            LDY   BAXLO,X
8C8C: 4C F2 E2 >540  ]LOOP     JMP   GIVAYF
               >541  :1        DO    KOPT-K6502
8C8F: 88       >544            DEY
8C90: 10 09    >546            BPL   :2
8C92: BD F8 05 >547            LDA   BAYHI,X     MOUSE(1) means read Y
```

```
8C95: BC F8 04           LDY   BAYLO,X
8C98: 4C F2 E2           JMP   GIVAYF
8C9B: BC 78 07  :2       LDY   BAMBS,X    MOUSE(2) means read buttons
8C9E: 4C 01 E3           JMP   SNGFLT
8CA1: A9 01     TFUNC    LDA   #1
8CA3: 20 50 8A           JSR   IVALARG
8CA6: 20 D2 8D           JSR   ISHOSTOK
8CA9: A2 00              LDX   #0
8CAB: A5 A1              LDA   FAC+4
8CAD: F0 02              BEQ   *+4
8CAF: A2 02              LDX   #2
8CB1: BD F6 99           LDA   KTINC+1,X
8CB4: BC F5 99           LDY   KTINC,X
8CB7: B0 D3              BCS   ]LOOP      Carry set as ISHOSTOK res.

                * Desactive le traitement d´une interruption (sur RETURN)
                * Y en entree: indice de l´interruption
8CB9: A9 00     COMINT1  LDA   #0
8CBB: 99 D4 99           STA   MODERUN,Y
8CBE: A9 FF              LDA   #$FF
8CC0: 8D D3 99           STA   YICUR
                * MODEPEC passe de STOP a ON
8CC3: BE D6 99           LDX   MODEPEC,Y
8CC6: E0 01              CPX   #1
8CC8: D0 05              BNE   :0
8CCA: E8                 INX
8CCB: 8A                 TXA
8CCC: 99 D6 99           STA   MODEPEC,Y
8CCF: B9 E0 99  :0       LDA   TPT_B,Y
8CD2: 85 B8              STA   TXTPTR
8CD4: B9 E2 99           LDA   TPT_T,Y
8CD7: 85 B9              STA   TXTPTR+1
8CD9: B9 DC 99           LDA   CLN_B,Y
8CDC: 85 75              STA   CURLIN
8CDE: B9 DE 99           LDA   CLN_T,Y
8CE1: 85 76              STA   CURLIN+1
8CE3: B9 E4 99           LDA   OTPT_B,Y
8CE6: 85 79              STA   OLDTEXT
8CE8: B9 E6 99           LDA   OTPT_T,Y
8CEB: 85 7A              STA   OLDTEXT+1
8CED: AE C2 99           LDX   SVMTACTV
8CF0: AD D4 99           LDA   MODERUN
8CF3: 0D D5 99           ORA   MODERUN+1
8CF6: D0 06              BNE   *+8
8CF8: 8D C2 99           STA   SVMTACTV
8CFB: 8E DC 9C           STX   MTACTV
8CFE: A0 05              LDY   #5
8D00: CC F4 99           CPY   FRGNDCTX
8D03: D0 05              BNE   :1
8D05: 68                 PLA
8D06: 68                 PLA
8D07: 4C 48 8E           JMP   RW2
8D0A: 60        :1       RTS

                * Routine en charge de determiner si l´interruption peut
                * ou non etre cascadee.
                * Sortie: bitN a 0 ssi possibilite de cascade (indice
               >624  * dans Y)
8D0B: A0 01    >625  COMINT2  LDY   #1          On commence par la TIMER
8D0D: B9 D8 99 >626  ]LOOP    LDA   MSKINT,Y
8D10: 08       >627           PHP               ;Sauve le interrupt enable
8D11: 78       >628           SEI               ;courant
8D12: 2D D2 99 >629           AND   MIRQST
8D15: F0 2C    >630           BEQ   :3
               >631  * Uniquement si prise en compte immediate..
8D17: BE D6 99 >632           LDX   MODEPEC,Y
8D1A: E0 02    >633           CPX   #2
8D1C: D0 25    >634           BNE   :3
               >635  * Uniquement si routine non deja active
8D1E: BE D4 99 >636           LDX   MODERUN,Y
8D21: D0 20    >637           BNE   :3
               >641  * A containes either $40 or $80
8D23: 49 C0    >642           EOR   #$C0
8D25: 2D D2 99 >643           AND   MIRQST
8D28: 8D D2 99 >644           STA   MIRQST
8D2B: 28       >646           PLP
8D2C: A9 02    >647           LDA   #3-1        because from within a called subr
.
8D2E: 20 D6 D3 >648           JSR   CHKMEM
8D31: 8C D3 99 >649           STY   YICUR
8D34: AD DC 9C >650           LDA   MTACTV
8D37: 8D C2 99 >651           STA   SVMTACTV
8D3A: A9 01    >652           LDA   #1
8D3C: 99 D6 99 >653           STA   MODEPEC,Y
8D3F: 99 D4 99 >654           STA   MODERUN,Y
8D42: 60       >655           RTS
8D43: 28       >656  :3       PLP
8D44: 88       >657           DEY
8D45: 10 C6    >658           BPL   ]LOOP
8D47: 60       >659           RTS
               >660
               >661  * Retour d´une interruption souris
8D48: A0 00    >662  RETOURM  LDY   #0
8D4A: 2C       >663           HEX   2C          Skip next two bytes
8D4B: A0 01    >664  RETOURT  LDY   #1
8D4D: BA       >665           TSX
8D4E: 86 F8    >666           STX   REMSTK
8D50: 20 B9 8C >667           JSR   COMINT1
8D53: 20 97 81 >668           JSR   DECTPTR
8D56: 20 58 D8 >669           JSR   ISCNTC
8D59: 4C 05 D8 >670           JMP   TRACE
               >671
8D5C: AD D4 99 >672  RNEWINST LDA   MODERUN
8D5F: 0D D5 99 >673           ORA   MODERUN+1
8D62: F0 19    >674           BEQ   RNI2
               >675  * Y a la bonne valeur selon MOUSE ou TIMER actifs
8D64: AC D3 99 >676           LDY   YICUR
8D67: 10 0A    >677           BPL   :1
8D69: C8       >678           INY               ;Y passe de FF a 0
8D6A: AD D5 99 >679           LDA   MODERUN+1
8D6D: F0 01    >680           BEQ   *+3
8D6F: C8       >681           INY               ;Y passe a 1
8D70: 8C D3 99 >682           STY   YICUR
8D73: BA       >683  :1       TSX
```

```
8D74: 8A                 TXA
               * Routine terminee par RETURN/POP ayant ramene le SP
8D75: D9 DA 99           CMP   INTSPTR,Y
8D78: 90 03              BCC   RNI2
8D7A: 20 B9 8C           JSR   COMINT1
               * ...
8D7D: AD D2 99 RNI2      LDA   MIRQST
8D80: F0 4D              BEQ   :4
8D82: 20 0B 8D           JSR   COMINT2
8D85: 30 48              BMI   :4          ;
               * Reminder of current stack pointer
8D87: BA                 TSX
8D88: 8A                 TXA
8D89: 99 DA 99           STA   INTSPTR,Y
               * Builds the GOSUB stack frame
8D8C: C0 01              CPY   #1          carry set iif TIMER int.
8D8E: B0 06              BCS   *+8
8D90: A2 47              LDX   #RETOURM-1
8D92: A9 8D              LDA   #>RETOURM-1
8D94: D0 04              BNE   *+6
8D96: A2 4A              LDX   #RETOURT-1
8D98: A9 8D              LDA   #>RETOURT-1
8D9A: 48                 PHA
8D9B: 8A                 TXA
8D9C: 48                 PHA
8D9D: A5 B9              LDA   TXTPTR+1
8D9F: 99 E2 99           STA   TPT_T,Y
8DA2: 48                 PHA
8DA3: A5 B8              LDA   TXTPTR
8DA5: 99 E0 99           STA   TPT_B,Y
8DA8: 48                 PHA
8DA9: A5 76              LDA   CURLIN+1
8DAB: 99 DE 99           STA   CLN_T,Y
8DAE: 48                 PHA
8DAF: A5 75              LDA   CURLIN
8DB1: 99 DC 99           STA   CLN_B,Y
8DB4: 48                 PHA
8DB5: A5 79              LDA   OLDTEXT
8DB7: 99 E4 99           STA   OTPT_B,Y
8DBA: A5 7A              LDA   OLDTEXT+1
8DBC: 99 E6 99           STA   OTPT_T,Y
8DBF: A9 B0              LDA   #TOKGOSUB
8DC1: 48                 PHA
               * and initialize the context for irq handler
               * (before falling into NEWSTT)
8DC2: BE BF 99           LDX   AHNDHI,Y
8DC5: B9 BD 99           LDA   AHNDLO,Y
8DC8: 85 B8              STA   TXTPTR
8DCA: 86 B9              STX   TXTPTR+1
8DCC: 4C D2 D7           JMP   NEWSTT

8DCF: 4C 76 88 :4        JMP   RNEWISUI

8DD2: AD ED 9C ISHOSTOK  LDA   MACHINE
8DD5: C9 41              CMP   #$41        Enhanced 2e ROM pattern
8DD7: 90 09              BCC   HNOK
8DD9: 60       ]RET      RTS
```

```
8DDA: AD B7 99  ISMOUSH  LDA   MOCN
8DDD: D0 FA              BNE   ]RET
8DDF: A2 20              LDX   #32
8DE1: 2C                 HEX   2C          Skip next two byte
8DE2: A2 21     HNOK     LDX   #33
8DE4: 68        NERRHP   PLA               ;Pull return address
8DE5: 68                 PLA
8DE6: 2C                 HEX   2C
8DE7: A2 24     NILLM    LDX   #36
                * Error handler for new reason codes
                * Upon entry, possible values of X
                * 32: MOUSE NOT DETECTED
                * UNSUPPORTED HARDWARE CONFIG.
                * UNKNOWN APPLESOFT MOUSE EVENT HANDLER
                * Same for TIMER
                * ILLEGAL MOUSE MODE
                * ILLEGAL MOUSE OP.
8DE9: 24 D8     NERRH    BIT   ERRFLG
8DEB: 10 03              BPL   *+5
8DED: 4C F9 E2           JMP   $E2F9       to ROM Error handler code
8DF0: 20 FB DA           JSR   CRDO
8DF3: 20 5A DB           JSR   $DB5A       Output question mark
8DF6: BD E2 9A           LDA   CODR-32,X
8DF9: AA                 TAX
8DFA: BD F9 99  ]LOOP    LDA   MESSERR,X
8DFD: 48                 PHA
8DFE: 20 5C DB           JSR   OUTDO
8E01: E8                 INX
8E02: 68                 PLA
8E03: 10 F5              BPL   ]LOOP
8E05: 4C 2A D4           JMP   $D42A       Fall into ROM code tail

8E08: 20 46 E7  RWAIT    JSR   $E746       Get address in LINNUM,
8E0B: 86 85              STX   FORPNT       mask in X (saved)
8E0D: A2 00              LDX   #0
8E0F: 20 B7 00           JSR   $00B7
8E12: F0 03              BEQ   *+5
8E14: 20 4C E7           JSR   COMBYTE
8E17: 86 86              STX   FORPNT+1
8E19: A0 00              LDY   #0
                COMWAIT
8E1B: AD D2 99  ]LOOP    LDA   MIRQST
8E1E: D0 09              BNE   :2
8E20: B1 50              LDA   (LINNUM),Y
8E22: 45 86              EOR   FORPNT+1
8E24: 25 85              AND   FORPNT
8E26: F0 F3              BEQ   ]LOOP
8E28: 60                 RTS
8E29: 20 0B 8D  :2       JSR   COMINT2
8E2C: 10 03              BPL   *+5
8E2E: C8                 INY
8E2F: F0 EA              BEQ   ]LOOP       Always
8E31: 98                 TYA
8E32: 48                 PHA
8E33: A0 05              LDY   #5
8E35: 8C F4 99           STY   FRGNDCTX
8E38: BE E8 99  ]LOOP    LDX   SVWOF,Y
```

```
8E3B: B5 00                LDA   0,X
8E3D: 99 EE 99             STA   SVA,Y
8E40: 88                   DEY
8E41: 10 F5                BPL   ]LOOP
8E43: 68                   PLA
8E44: A8                   TAY
8E45: 4C 87 8D             JMP   RNI2+10

8E48: A0 06      RW2       LDY   #6
8E4A: BE E7 99   ]LOOP     LDX   SVWOF-1,Y
8E4D: B9 ED 99             LDA   SVA-1,Y
8E50: 95 00                STA   0,X
8E52: 88                   DEY
8E53: D0 F5                BNE   ]LOOP
8E55: 8C F4 99             STY   FRGNDCTX
8E58: F0 C1                BEQ   COMWAIT    Always

8E5A: A9 10      GN32768   LDA   #NEG32768
8E5C: A0 9B                LDY   #>NEG32768
8E5E: 60                   RTS
8E5F: A9 15      GP32768   LDA   #POS32768
8E61: A0 9B                LDY   #>POS32768
8E63: 60                   RTS

8E64: A9 0B      GN65536   LDA   #NEG65536
8E66: A0 9B                LDY   #>NEG65536
8E68: 60                   RTS
8E69: A9 1A      GP65536   LDA   #POS65536
8E6B: A0 9B                LDY   #>POS65536
8E6D: 60                   RTS

                 * Get address of array which name is pointed to by
                 * TXTPTR. If no array is found, then the called
                 * ROM routine would have created one so we´ll have
                 * to rollback such creation and exit.
                 NGTA2     DO    KOPT16
8E6E: A5 6E                LDA   STREND+1
8E70: 48                   PHA
8E71: A5 6D                LDA   STREND
8E73: 48                   PHA
8E74: 20 3E 79             JSR   NGETARPT
8E77: 68                   PLA
8E78: AA                   TAX
8E79: 68                   PLA
8E7A: B0 04                BCS   :1         found existing array
8E7C: 85 6E                STA   STREND+1   Do the rollback
8E7E: 86 6D                STX   STREND
                 :1        DO    KOPT-K65C02
8E80: A9 00                LDA   #0
8E82: 85 14                STA   SUBFLG
8E84: 60                   RTS

                           PUT   PEERFORNEXT
                 * Module en charge du traitement de boucles FOR/NEXT
                 * en variante classique comme en variante FOREACH
                 GTFORPNT EQU   $D365
                 GETSPA   EQU   $E452      Get mem. space for new string
```

```

8E85: 4C 76 DD   ]ERR     JMP   GOTMIERR
8E88: 20 49 86   FEFOR    JSR   NCHKCOM    Ensure trailing comma
8E8B: A5 86               LDA   FORPNT+1
8E8D: 48                  PHA
8E8E: A5 85               LDA   FORPNT
8E90: 48                  PHA
8E91: A5 12               LDA   INTTYP
8E93: 48                  PHA
8E94: A5 11               LDA   VALTYP
8E96: 48                  PHA
8E97: 20 6E 8E            JSR   NGTA2
8E9A: 90 E9               BCC   ]ERR       En attendant mieux..
                 * Same element type for array and loop variable?
8E9C: 68                  PLA
8E9D: 45 11               EOR   VALTYP
8E9F: D0 E4               BNE   ]ERR
8EA1: 68                  PLA
8EA2: 45 12               EOR   INTTYP
8EA4: D0 DF               BNE   ]ERR
8EA6: 68                  PLA
8EA7: 85 85               STA   FORPNT
8EA9: 68                  PLA
8EAA: 85 86               STA   FORPNT+1
                 * LOWTR address: array base address
                 * FORPNT address: simple variable value address
8EAC: 20 65 D3            JSR   GTFORPNT
8EAF: D0 05               BNE   :1         Si pas trouvee
                 * Si oui, on revient au debut de la struct. dans la pile
8EB1: 8A                  TXA
8EB2: 69 0F               ADC   #15
8EB4: AA                  TAX
8EB5: 9A                  TXS
8EB6: 68         :1       PLA              ;Pop return address
8EB7: 68                  PLA
                 * Check enough space on stack and start computing
                 * TXTPTR for body loop.
8EB8: 20 CC 91            JSR   LBS60
8EBB: 48                  PHA
8EBC: A5 B9               LDA   TXTPTR+1
8EBE: 69 00               ADC   #0
8EC0: 48                  PHA
8EC1: A5 76               LDA   CURLIN+1
8EC3: 48                  PHA
8EC4: A5 75               LDA   CURLIN
8EC6: 48                  PHA
                 * Analyse array: result $9D and $A0 in abs. form
8EC7: 20 B0 90            JSR   LBS61
8ECA: 20 E0 90            JSR   LBS63      Copy 1st elm into loop var.
8ECD: 20 75 91            JSR   LBS68      From abs to offset(ARYTAB)
8ED0: A9 D7      SFE1     LDA   #FESTEP
8ED2: A0 8E               LDY   #>FESTEP
8ED4: 4C D9 90            JMP   LBS62

8ED7: A9 A5      FESTEP   LDA   #%10100101
8ED9: 4C 73 8F            JMP   COMFOR

```

```
8EDC: 4C C9 DE  ]ERR      JMP   SYNERR
8EDF: 20 A3 91  RFOR      JSR   ITEACH      ;FOREACH variant?
8EE2: 08                  PHP               ;Z bit on stack
8EE3: A2 00               LDX   #0
8EE5: 86 14               STX   SUBFLG
8EE7: 20 46 79            JSR   NPTRGET
8EEA: 85 85               STA   FORPNT
8EEC: 84 86               STY   FORPNT+1
8EEE: C5 6B               CMP   ARYTAB
8EF0: 98                  TYA
8EF1: E5 6C               SBC   ARYTAB+1
8EF3: B0 E7               BCS   ]ERR
8EF5: 28                  PLP
8EF6: F0 90               BEQ   FEFOR
8EF8: A0 01               LDY   #1
8EFA: B1 9B               LDA   (LOWTR),Y
8EFC: 88                  DEY
8EFD: 51 9B               EOR   (LOWTR),Y
8EFF: 30 DB               BMI   ]ERR
8F01: A5 12               LDA   INTTYP
8F03: 48                  PHA
8F04: 20 D1 75            JSR   RLET1
8F07: 68                  PLA
8F08: 85 C0               STA   GFLAG
8F0A: 20 65 D3            JSR   GTFORPNT
8F0D: D0 05               BNE   :0
8F0F: 8A                  TXA               ;Stackframe pointer in X
8F10: 69 0F               ADC   #$0F       Carry already set, add 16
8F12: AA                  TAX               ;+2 bytes (lines below)
8F13: 9A                  TXS               ;= 18 bytes
8F14: 68        :0        PLA
8F15: 68                  PLA
8F16: 24 C0               BIT   GFLAG
8F18: 30 03               BMI   :1
8F1A: 4C 79 D7            JMP   $D779
                * Check that enough stack available and
                * compute Y as offset to next separator
8F1D: 20 CC 91  :1        JSR   LBS60
8F20: 48                  PHA
8F21: A5 B9               LDA   TXTPTR+1
8F23: 69 00               ADC   #0
8F25: 48                  PHA
8F26: A9 C1               LDA   #TOKTO
8F28: 20 CE 7D            JSR   NSYNCHR
8F2B: A5 76               LDA   CURLIN+1
8F2D: 48                  PHA
8F2E: A5 75               LDA   CURLIN
8F30: 48                  PHA
8F31: 18                  CLC
8F32: 20 BD 91            JSR   LBS033
8F35: A9 3C     STP1      LDA   #STEP
8F37: A0 8F               LDY   #>STEP
8F39: 4C D9 90            JMP   LBS62

8F3C: 20 65 75  STEP      JSR   RST102
8F3F: A0 01               LDY   #1
8F41: 84 A1               STY   FACLO
```

```
8F43: 88       >140             DEY
8F44: 84 A0    >141             STY   FACMO
8F46: C9 C7    >145             CMP   #TOKSTEP
8F48: 18       >146             CLC
8F49: D0 07    >147             BNE   *+9
8F4B: 20 47 75 >148             JSR   RST100
8F4E: 38       >149             SEC
8F4F: 20 BD 91 >150             JSR   LBS033
8F52: 08       >151             PHP
8F53: A0 01    >152             LDY   #1         Step value > 0 par defaut
8F55: B0 09    >153             BCS   :1         Branch iif inversion de signe
8F57: A5 A0    >154             LDA   FACMO
8F59: 30 05    >155             BMI   :1
8F5B: 05 A1    >156             ORA   FACLO
8F5D: D0 03    >157             BNE   :2
8F5F: 24       >158             HEX   24         Skip next byte
8F60: 88       >159  :1         DEY
8F61: 88       >160             DEY
8F62: 98       >161  :2         TYA
8F63: 49 80    >162             EOR   #$80       Tag for integer var.
8F65: 29 C3    >163             AND   #%11000011
8F67: 2C E7 9C >164             BIT   WMODE
8F6A: 10 02    >165             BPL   *+4
8F6C: 09 20    >166             ORA   #%00100000 Set Unsigned arith. flag
8F6E: 28       >167             PLP
8F6F: 90 02    >168             BCC   *+4
8F71: 09 10    >169             ORA   #%00010000 Set reverse step value sign
8F73: 20 9D 90 >170  COMFOR     JSR   NFRMSTK2
8F76: 4C C9 D7 >175             JMP   $D7C9
               >177
               >178  * Incrementation de l´index d´elm.
8F79: EE 23 96 >179  ]LOOP      INC   AEI        Incrementation de l´index d´elm
8F7C: D0 03    >180             BNE   *+5
8F7E: EE 24 96 >181             INC   AEI+1
               >182  * From new array element to loop var (value)
8F81: 20 E0 90 >183             JSR   LBS63
8F84: A4 5E    >184             LDY   INDEX      Write back $9D,$9E to stack
8F86: 20 5E 91 >185             JSR   LBS67
8F89: BA       >186             TSX
8F8A: 4C 3E DD >187             JMP   $DD3E
               >188
8F8D: 20 4B 91 >189  FENEXT     JSR   LBS64      Step FP value into FAC
8F90: 20 8A 91 >190             JSR   LBS69      From offset(ARYTAB) to absol.
8F93: 20 14 91 >191             JSR   LBS65      Loop var. back into array elm.
8F96: A5 9F    >192             LDA   $9F
8F98: 18       >193             CLC
8F99: 65 9D    >194             ADC   $9D
8F9B: 85 9D    >195             STA   $9D
8F9D: 90 02    >196             BCC   *+4
8F9F: E6 9E    >197             INC   $9E
               >198  * Loop exhausted?
8FA1: C5 A0    >199             CMP   $A0
8FA3: A5 9E    >200             LDA   $9E
8FA5: E5 A1    >201             SBC   $A1
               >202  * Carry set iif loop exhausted
8FA7: 90 D0    >203             BCC   ]LOOP
8FA9: A9 00    >208             LDA   #0
```

```
8FAB: 8D 23 96          STA   AEI
8FAE: 8D 24 96          STA   AEI+1
8FB1: BA                TSX
8FB2: 4C 71 90          JMP   COMNEXT    Always

8FB5: 4C 0B DD  ]LOOP   JMP   $DD0B      NEXT WITHOUT FOR error
8FB8: D0 04     RNEXT   BNE   NEXT1
8FBA: A0 00             LDY   #0
8FBC: F0 03             BEQ   *+5
8FBE: 20 42 79  NEXT1   JSR   NPTRGTX
8FC1: 85 85             STA   FORPNT
8FC3: 84 86             STY   FORPNT+1
8FC5: 20 65 D3          JSR   $D365
8FC8: D0 EB             BNE   ]LOOP
8FCA: 9A                TXS
8FCB: E8                INX
8FCC: E8                INX
8FCD: E8                INX
8FCE: E8                INX
8FCF: 8A                TXA              ;Base address of STEP value
8FD0: E8                INX
8FD1: E8                INX
8FD2: E8                INX
8FD3: E8                INX
8FD4: E8                INX
8FD5: E8                INX
8FD6: 86 60             STX   DEST       Base adress of TO value
8FD8: A8                TAY
8FD9: BA                TSX
8FDA: BD 09 01          LDA   $0109,X
8FDD: 85 C0             STA   GFLAG
8FDF: 0A                ASL
8FE0: 90 08             BCC   :1
8FE2: 10 08             BPL   :2
8FE4: 98        ]LOOP   TYA
8FE5: A6 60             LDX   DEST
8FE7: 4C 1D DD          JMP   $DD1D      FP var: classic mechanic
8FEA: 10 F8     :1      BPL   ]LOOP
8FEC: 29 08     :2      AND   #%00001000 Voir ASL precedent..
8FEE: D0 9D             BNE   FENEXT
8FF0: A2 00             LDX   #0
8FF2: 20 86 90          JSR   LBS05      Step value into $A0, $A1
8FF5: D0 02             BNE   *+4
8FF7: A2 04             LDX   #4
8FF9: 50 01             BVC   *+3
8FFB: E8                INX
8FFC: 90 04             BCC   *+6
8FFE: 8A                TXA
8FFF: 09 08             ORA   #8
9001: AA                TAX
9002: 20 9D 76          JSR   HNDLEIY    Current value in FORPNT
9005: A2 FF             LDX   #-1
9007: A4 60             LDY   DEST
9009: 20 8A 90          JSR   LBS051
900C: 08                PHP
900D: A2 FF             LDX   #-1        endvalue > FAC par defaut
900F: A0 01             LDY   #1
```

```
9011: A5 A1    >263           LDA   $A1
9013: 28       >264           PLP
               >265  * A: -1 iif endvalue > current value
               >266  * A: 0 iif endvalue = current value
               >267  * A: 1 iif endvalue < current value
9014: 90 20    >268           BCC   :SI        Branch iif signed arithmetic
9016: D0 0D    >269           BNE   :7
9018: 88       >273           DEY
9019: F1 85    >274           SBC   (FORPNT),Y
901B: F0 03    >276           BEQ   *+5
901D: B0 02    >277           BCS   *+4
901F: E8       >278           INX
9020: E8       >279           INX
9021: 8A       >280  ]LOOP    TXA
9022: 4C 5C 90 >281           JMP   :10
9025: F1 85    >282  :7       SBC   (FORPNT),Y
9027: 85 3C    >283           STA   A1L
9029: A5 A0    >284           LDA   $A0
902B: 88       >288           DEY
902C: F1 85    >289           SBC   (FORPNT),Y
902E: 05 3C    >291           ORA   A1L
9030: F0 EE    >292           BEQ   ]LOOP-1
9032: B0 ED    >293           BCS   ]LOOP
9034: 90 E9    >294           BCC   ]LOOP-2    Always
               >295
               >296  * Signed arithetic comparison
9036: 38       >297  :SI      SEC
9037: D0 09    >298           BNE   :6
9039: 88       >302           DEY
903A: F1 85    >303           SBC   (FORPNT),Y
903C: D0 0E    >305           BNE   :5
903E: E8       >306           INX
903F: 4C 4C 90 >307           JMP   :5
9042: F1 85    >308  :6       SBC   (FORPNT),Y
9044: D0 01    >309           BNE   *+3
9046: E8       >310           INX
9047: A5 A0    >311           LDA   $A0
9049: 88       >313           DEY
904A: F1 85    >314           SBC   (FORPNT),Y
904C: 70 0C    >318  :5       BVS   :C1
904E: 30 07    >319           BMI   :LT
9050: D0 02    >320  ]LOOP    BNE   :C20
9052: 8A       >321           TXA              ;A=0 if both bytes equal
9053: 2C       >322           HEX   2C         next two bytes
9054: A9 FF    >323  :C20     LDA   #-1
9056: 2C       >324           HEX   2C
9057: A9 01    >325  :LT      LDA   #1
9059: 2C       >326           HEX   2C         Skip next two bytes
905A: 10 F4    >327  :C1      BPL   ]LOOP
905C: A8       >328  :10      TAY
905D: A5 C0    >329           LDA   GFLAG
905F: 29 03    >330           AND   #%00000011
9061: AA       >331           TAX
9062: BD 1F 96 >332           LDA   MOTGF,X
9065: 85 C0    >333           STA   GFLAG
9067: 98       >334           TYA
9068: BA       >335           TSX
```

```
9069: 38       >336            SEC
906A: E5 C0    >337            SBC   GFLAG
906C: F0 03    >338            BEQ   *+5
906E: 4C 3E DD >339            JMP   $DD3E       Processing next loop iteration
9071: 8A       >340  COMNEXT   TXA               ;Arithmetic of frame pointer
9072: 69 11    >341            ADC   #17         Carry set so add 18
9074: AA       >342            TAX
9075: 9A       >343            TXS
9076: 20 65 75 >344            JSR   RST102
9079: C9 2C    >345            CMP   #´,´
907B: D0 06    >346            BNE   *+8
907D: 20 47 75 >347            JSR   RST100
9080: 20 BE 8F >348            JSR   NEXT1       Does not return
9083: 4C D2 D7 >349            JMP   NEWSTT
               >350
9086: A9 01    >351  LBS05     LDA   #1
9088: 85 5F    >352            STA   INDEX+1
908A: 20 4F 91 >353  LBS051    JSR   LBS641
908D: A5 C0    >354            LDA   GFLAG
908F: 0A       >355            ASL
9090: 0A       >356            ASL
9091: 0A       >357            ASL               ;Unsigned into carry and reverse
into ovf
9092: B8       >358            CLV
9093: 10 03    >359            BPL   *+5
9095: 2C D9 8D >360            BIT   ]RET
9098: B1 85    >361            LDA   (FORPNT),Y Y a 4: pointe sur le subtype
909A: 49 81    >362            EOR   #$81        Z a 1 ssi BYTE
909C: 60       >363            RTS
               >364
               >365  NFRMSTK2
909D: A8       >366            TAY               ;FAC sign or SGN(step value)
909E: 68       >367            PLA
909F: AA       >367            TAX
90A0: 68       >368            PLA
90A1: E8       >369            INX
90A2: 86 5E    >370            STX   INDEX
90A4: D0 03    >371            BNE   :1
90A6: 18       >375            CLC
90A7: 69 01    >376            ADC   #1
90A9: 85 5F    >378  :1        STA   INDEX+1
90AB: 98       >379            TYA
90AC: 48       >379            PHA
90AD: 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.
90B0: A0 04    >384  LBS61     LDY   #4
90B2: B1 9B    >385            LDA   (LOWTR),Y
90B4: 29 07    >386            AND   #7          Isolate # of dims.
90B6: 0A       >387            ASL               ;2 bytes per dimensions
90B7: 69 05    >388            ADC   #5          Carry clear
90B9: 65 9B    >389            ADC   LOWTR
90BB: 85 9D    >390            STA   $9D
90BD: A9 00    >391            LDA   #0
90BF: A8       >393            TAY
90C0: 65 9C    >395            ADC   LOWTR+1
```

```
90C2: 85 9E    >396              STA   $9E
90C4: A0 02    >397              LDY   #2
90C6: B1 9B    >398              LDA   (LOWTR),Y
90C8: C8       >399              INY
90C9: 65 9B    >400              ADC   LOWTR
90CB: 85 A0    >401              STA   $A0
90CD: B1 9B    >402              LDA   (LOWTR),Y
90CF: 65 9C    >403              ADC   LOWTR+1
90D1: 85 A1    >404              STA   $A1
               >405  * Taille d´un element
90D3: 20 BB 7C >406              JSR   KWELMSIZ
90D6: 86 9F    >407              STX   $9F
90D8: 60       >408              RTS
               >409
90D9: 85 5E    >410  LBS62       STA   INDEX
90DB: 84 5F    >411              STY   INDEX+1
90DD: 4C 23 DE >412              JMP   FRMSTCK3+3
               >413
               >414  * From array element to loop var.
90E0: A4 9F    >415  LBS63       LDY   $9F
90E2: C0 03    >416              CPY   #3
90E4: F0 09    >417              BEQ   :0          Special handling for strings
90E6: 88       >418              DEY
90E7: B1 9D    >419  ]LOOP       LDA   ($9D),Y
90E9: 91 85    >420              STA   (FORPNT),Y
90EB: 88       >421              DEY
90EC: 10 F9    >422              BPL   ]LOOP
90EE: 60       >423  ]RET        RTS
               >424  * Special handling for strings
90EF: A0 00    >429  :0          LDY   #0
90F1: B1 9D    >430              LDA   ($9D),Y
90F3: 91 85    >431              STA   (FORPNT),Y
90F5: F0 F7    >433              BEQ   ]RET        Nothing to do if length zero
90F7: 48       >434              PHA
90F8: 20 40 91 >435              JSR   LBS66
90FB: 91 85    >436              STA   (FORPNT),Y
               >437  * A1L,A1H: adresse source
90FD: B1 9D    >438              LDA   ($9D),Y
90FF: 85 3D    >439              STA   A1L+1
9101: 88       >440              DEY
9102: 8A       >441              TXA
9103: 91 85    >442              STA   (FORPNT),Y
9105: B1 9D    >443              LDA   ($9D),Y
9107: 85 3C    >444              STA   A1L
               >445  * Do the string copy itself: recall string length
9109: 68       >450              PLA
910A: A8       >451  COMCOPY     TAY
910B: 88       >452              DEY
910C: B1 3C    >454  ]LOOP       LDA   (A1L),Y
910E: 91 3E    >455              STA   (A2L),Y
9110: 88       >456              DEY
9111: 10 F9    >457              BPL   ]LOOP
9113: 60       >458              RTS
               >459
               >460  * From loop var. to array elm.
9114: A4 9F    >461  LBS65       LDY   $9F
9116: C0 03    >462              CPY   #3
```

```
9118: F0 09              BEQ   :0          Special handling for strings
911A: 88                 DEY
911B: B1 85     ]LOOP    LDA   (FORPNT),Y
911D: 91 9D              STA   ($9D),Y
911F: 88                 DEY
9120: 10 F9              BPL   ]LOOP
9122: 60        ]RET     RTS
                * Special handling for strings
9123: A0 00     :0       LDY   #0
9125: B1 85              LDA   (FORPNT),Y Length byte
9127: 91 9D              STA   ($9D),Y
9129: F0 F7              BEQ   ]RET        Nothing to do if length zero
912B: 48                 PHA
912C: 20 40 91           JSR   LBS66
912F: 91 9D              STA   ($9D),Y     High byte
                * A1L,A1H: adresse source
9131: B1 85              LDA   (FORPNT),Y
9133: 85 3D              STA   A1L+1
9135: 88                 DEY
9136: 8A                 TXA
9137: 91 9D              STA   ($9D),Y
9139: B1 85              LDA   (FORPNT),Y
913B: 85 3C              STA   A1L
                * Do the string copy itself: recall string length
913D: 68                 PLA
913E: D0 CA              BNE   COMCOPY     Always

9140: 20 52 E4  LBS66    JSR   GETSPA
                * returns with Y,X pointer to new string
                * A2L,A2H: adresse destination
9143: 84 3F              STY   A2L+1
9145: 86 3E              STX   A2L
9147: 98                 TYA
9148: A0 02              LDY   #2
914A: 60                 RTS

                * Subroutine: copy from stack to FAC in page zero
914B: A9 01     LBS64    LDA   #1
914D: 85 5F              STA   INDEX+1
914F: 84 5E     LBS641   STY   INDEX
9151: A0 FF              LDY   #-1
9153: C8        ]LOOP    INY
9154: B1 5E              LDA   (INDEX),Y
9156: 99 9D 00           STA   $9D,Y
9159: C0 04              CPY   #4
915B: 90 F6              BCC   ]LOOP
915D: 60                 RTS

                * From FAC to stack.. called from FENEXT
                * $9D is expected to be in absolute mode
915E: A9 01     LBS67    LDA   #1
9160: 85 5F              STA   INDEX+1
9162: 84 5E              STY   INDEX
9164: A5 9D              LDA   $9D         Convert $9D$9E
9166: 38                 SEC               ; to offset(ARYTAB)
9167: E5 6B              SBC   ARYTAB
9169: A0 00              LDY   #0
```

```
916B: 91 5E             STA   (INDEX),Y
916D: C8                INY
916E: A5 9E             LDA   $9E
9170: E5 6C             SBC   ARYTAB+1
9172: 91 5E             STA   (INDEX),Y
9174: 60                RTS

               * From absolute address to offset from ARYTAB
9175: A2 A0    LBS68    LDX   #$A0
9177: 20 7C 91          JSR   *+5
917A: A2 9D             LDX   #$9D
917C: B5 00             LDA   0,X
917E: 38                SEC
917F: E5 6B             SBC   ARYTAB
9181: 95 00             STA   0,X
9183: B5 01             LDA   1,X
9185: E5 6C             SBC   ARYTAB+1
9187: 95 01             STA   1,X
9189: 60                RTS

               * From offset to absolute address
918A: A0 A0    LBS69    LDY   #$A0
918C: 20 91 91          JSR   *+5
918F: A0 9D             LDY   #$9D
9191: B9 00 00          LDA   0,Y
9194: 18                CLC
9195: 65 6B             ADC   ARYTAB
9197: 99 00 00          STA   0,Y
919A: B9 01 00          LDA   1,Y
919D: 65 6C             ADC   ARYTAB+1
919F: 99 01 00          STA   1,Y
91A2: 60                RTS

               * Return with Z flag set iif ´EACH´ string @ TXTPTR
               * TXTPTR updated accordngly if so
91A3: A0 FF    ITEACH   LDY   #-1
91A5: C8       ]LOOP    INY
91A6: B1 B8             LDA   (TXTPTR),Y
91A8: D9 59 9B          CMP   IFEACH,Y
91AB: D0 0F             BNE   :0
91AD: C0 03             CPY   #3
91AF: D0 F4             BNE   ]LOOP
91B1: 98                TYA
91B2: 65 B8             ADC   TXTPTR
91B4: 85 B8             STA   TXTPTR
91B6: 90 02             BCC   *+4
91B8: E6 B9             INC   TXTPTR+1
91BA: A0 00             LDY   #0          Set Zflag
91BC: 60       :0       RTS

91BD: 20 7E 77 LBS033   JSR   LBS03
91C0: 08                PHP
91C1: A5 C0             LDA   GFLAG
91C3: C9 81             CMP   #$81
91C5: D0 03             BNE   :0
91C7: 20 26 79          JSR   CONV1628
91CA: 28       :0       PLP
```

```
91CB: 60                       RTS

                      * a) Enough space on stack?
91CC: A9 07           LBS60    LDA   #9-2        -2 car on est dans une SUBR
91CE: 20 D6 D3                 JSR   CHKMEM
                      * b) Debut du calcul du nouveau TXTPTR
                      * Comme c´est une operation avec la pile, oblige de
                      * morceler l´operation
91D1: 20 A3 D9                 JSR   DATAN       Prochain separateur (offset Y)
91D4: 18                       CLC
91D5: 98                       TYA
91D6: 65 B8                    ADC   TXTPTR
91D8: 60                       RTS
                               PUT   PEERGOTO
                      * Module in charge of accelerating GOTO/GOSUB line address
                      * computations.
                      TXTTAB   EQU   $67
                      TOKTHEN  =     $C4
                      GOTOTAIL EQU   $D95E
                      FOUT     EQU   $ED34
                      RD2      EQU   $A47A       Read 2 first bytes from file

                      EXFLG    EQU   $AAB3       Exec file activity flag
                      WHCBASIC EQU   $AAB6       0 iif Integer BASIC active
                      ISBASRUN EQU   $A65E
                      * Part of the DOS 3.3 keyboard intercept routine
91D9: AD B6 AA        NKBDINT  LDA   WHCBASIC
91DC: F0 10                    BEQ   :0
91DE: 20 5E A6                 JSR   ISBASRUN
91E1: 90 0B                    BCC   :0          program running
91E3: AD D0 9C                 LDA   OPTCGOTO
91E6: 2D CF 9C                 AND   NEEDDEC
91E9: 10 03                    BPL   :0
91EB: 20 B7 93                 JSR   DECOMPILE
91EE: AD B3 AA        :0       LDA   EXFLG
91F1: 60                       RTS

                      * New DOS Applesoft SAVE command handler (or part of)
91F2: 20 B7 93        NDSVCMD  JSR   DECOMPILE
91F5: A9 02                    LDA   #2          Restore original A value..
91F7: 4C D5 A3                 JMP   $A3D5       Fall into $A3D5 (orig. content)

                      * Reset NEEDDEC upon DOS 3.3 Applesoft program loading
                      NDLVCMD  DO    KOPT-K6502
91FA: A9 00                    LDA   #0
91FC: 8D CF 9C                 STA   NEEDDEC
91FF: 4C 7A A4                 JMP   RD2

9202: 9D D8 9B        ROUT8C   STA   ADPFB,X
9205: 98                       TYA
9206: 9D EC 9B                 STA   ADPFT,X
9209: E8                       INX
920A: 60              ]RET     RTS
                      * Programmer routine to set the precomputed GOTO behavior
                      * CALL RE!,8,<n>
                      * with n being 0 to inactivate precomputed GOTOs,
                      * 128 to activate precomuted GOTOs w/o safeguard option
```

```
                * 192 to activate precomputed GOTOs w safeguard option.
920B: 20 49 86  ROUT8    JSR   NCHKCOM
920E: 20 73 86           JSR   NGETBYT    Reason code in X
9211: 8E D0 9C           STX   OPTCGOTO
9214: 8A                 TXA
9215: A2 0D              LDX   #OFSTGTO-ADPFB
9217: A8                 TAY
9218: 10 16              BPL   :2
921A: A9 07              LDA   #RGOTO-1
921C: A0 93              LDY   #>RGOTO-1
921E: 20 02 92           JSR   ROUT8C
9221: A9 E0              LDA   #RIF-1
9223: A0 92              LDY   #>RIF-1
9225: 20 02 92           JSR   ROUT8C
9228: E8                 INX
9229: A9 BE              LDA   #RGOSUB-1
922B: A0 92              LDY   #>RGOSUB-1
922D: 20 02 92           JSR   ROUT8C
9230: 2C D0 9C  :2       BIT   OPTCGOTO
9233: 30 18              BMI   :3
9235: 08                 PHP
9236: A9 3D              LDA   #APRGOTO-1
9238: A0 D9              LDY   #>APRGOTO-1
923A: 20 02 92           JSR   ROUT8C
923D: A9 C8              LDA   #APRIF-1
923F: A0 D9              LDY   #>APRIF-1
9241: 20 02 92           JSR   ROUT8C
9244: E8                 INX
9245: A9 20              LDA   #APRGOSUB-1
9247: A0 D9              LDY   #>APRGOSUB-1
9249: 20 02 92           JSR   ROUT8C
924C: 28                 PLP
924D: 70 02     :3       BVS   :0
924F: 30 B9              BMI   ]RET
9251: 4C B7 93  :0       JMP   DECOMPILE  in case reason code 0 or 192

9254: 4C C9 DE  ]ERR     JMP   SYNERR
9257: A2 01     RON      LDX   #1
9259: 20 0F 8B           JSR   COMLBS
925C: F0 35              BEQ   :1
                * Function call: normal flow
925E: B1 B8              LDA   (TXTPTR),Y
9260: C9 28              CMP   #´(´
9262: F0 2F              BEQ   :1         Normal function
                * ON MOUSE GOSUB or ON TIMER GOSUB pattern
9264: 20 98 D9           JSR   ADDON
9267: A9 B0              LDA   #TOKGOSUB
9269: 20 CE 7D           JSR   NSYNCHR
926C: 20 0F 93           JSR   RGPART1    LOWTR: address of target line
926F: A5 BD              LDA   IDMOCL
9271: 38                 SEC
9272: E9 08              SBC   #OFFMOU-TOFFST
9274: AA                 TAX
9275: A5 9B              LDA   LOWTR
9277: E9 01              SBC   #1         Carry already set
9279: 9D BD 99           STA   AHNDLO,X
927C: A5 9C              LDA   LOWTR+1
```

```
927E: E9 00    >104            SBC   #0
9280: 9D BF 99 >105            STA   AHNDHI,X
9283: A5 50    >106            LDA   LINNUM
9285: 9D B9 99 >107            STA   CLNLO,X
9288: A5 51    >108            LDA   LINNUM+1
928A: 9D BB 99 >109            STA   CLNHI,X
928D: 20 65 75 >110            JSR   RST102
9290: D0 C2    >111            BNE   ]ERR
9292: 60       >112            RTS
9293: 20 73 86 >113  :1        JSR   NGETBYT
9296: C9 B0    >114            CMP   #TOKGOSUB
9298: F0 04    >115            BEQ   :2
929A: 49 AB    >116            EOR   #TOKGOTO   TOKGOTO being < TOKGOSUB
929C: D0 B6    >117            BNE   ]ERR       carry is already clear
929E: 08       >118  :2        PHP
929F: C6 A1    >119  ]LOOP     DEC   FAC+4
92A1: D0 0F    >120            BNE   :3
92A3: 28       >121            PLP
               >122  * Carry set iif GOSUB, else GOTO (carry clear)
92A4: 90 06    >123            BCC   :GOTO
92A6: 20 47 75 >124            JSR   RST100
92A9: 4C BF 92 >125            JMP   RGOSUB
92AC: 20 47 75 >126  :GOTO     JSR   RST100
92AF: 4C 08 93 >127            JMP   RGOTO
92B2: 20 95 94 >128  :3        JSR   LRST100
92B5: 90 FB    >129            BCC   :3         Loop till not digit
92B7: E0 2C    >130            CPX   #´,´
92B9: F0 E4    >131            BEQ   ]LOOP
92BB: 28       >132            PLP
92BC: 4C 60 94 >133            JMP   NDATAN
               >134
92BF: 08       >135  RGOSUB    PHP
92C0: 48       >136            PHA
92C1: A9 02    >137            LDA   #3-1       -1 because of PLA PLP below..
92C3: 20 D6 D3 >138            JSR   CHKMEM
92C6: 68       >139            PLA
92C7: 28       >140            PLP
92C8: 20 0F 93 >141            JSR   RGPART1
92CB: A5 B9    >146            LDA   TXTPTR+1
92CD: 48       >147            PHA
92CE: A5 B8    >148            LDA   TXTPTR
92D0: 48       >149            PHA
92D1: A5 76    >150            LDA   CURLIN+1
92D3: 48       >151            PHA
92D4: A5 75    >152            LDA   CURLIN
92D6: 48       >153            PHA
92D7: A9 B0    >155            LDA   #TOKGOSUB
92D9: 48       >156            PHA
92DA: 38       >157            SEC
92DB: 20 5E D9 >158            JSR   GOTOTAIL
92DE: 4C D2 D7 >159            JMP   NEWSTT
               >160
92E1: 20 7B DD >161  RIF       JSR   FRMEVL
92E4: A5 9D    >162            LDA   FAC
92E6: F0 0F    >163            BEQ   :20
92E8: A0 00    >167            LDY   #0
92EA: B1 B8    >168            LDA   (TXTPTR),Y
```

```
92EC: C9 AB              CMP   #TOKGOTO
92EE: F0 13              BEQ   :4
92F0: C9 C4              CMP   #TOKTHEN
92F2: F0 0F              BEQ   :4
92F4: 4C 78 7B           JMP   SNERR
92F7: 20 63 94  :20      JSR   NREMN
92FA: 4C 98 D9           JMP   ADDON
92FD: 20 5A 8B  :3       JSR   LBS10
9300: 4C 28 D8           JMP   $D828
9303: 20 47 75  :4       JSR   RST100
9306: B0 F5              BCS   :3

9308: 20 0F 93  RGOTO    JSR   RGPART1
930B: 38                 SEC
930C: 4C 5E D9           JMP   GOTOTAIL

                * First part of GOTO..
                * Upon entry: A contains first target line no. char.,
                * C clear iif this character is a numeric digit.
                * Upon exit: LOWTR set to base adress of target line,
                * LINNUM set to target line no.
930F: 90 2C     RGPART1  BCC   :2          if num. digit then process it
9311: C9 20              CMP   #$20
9313: 90 03              BCC   *+5
9315: 4C 78 7B  :11      JMP   SNERR
                * Offset of target line from beginning of program
                * already computed (value within program text).
9318: E9 1C              SBC   #$1D-1
931A: A8                 TAY
931B: C8                 INY
931C: B1 B8              LDA   (TXTPTR),Y lo byte
931E: 18                 CLC
931F: 65 67              ADC   TXTTAB      to absolute address lo byte
9321: 85 9B              STA   LOWTR
9323: C8                 INY
9324: B1 B8              LDA   (TXTPTR),Y hi byte
9326: 65 68              ADC   TXTTAB+1    to absolute address
9328: 85 9C              STA   LOWTR+1
932A: C8                 INY
932B: 98                 TYA
932C: 48                 PHA
932D: A0 02              LDY   #2
932F: B1 9B              LDA   (LOWTR),Y
9331: 85 50              STA   LINNUM
9333: C8                 INY
9334: B1 9B              LDA   (LOWTR),Y
9336: 85 51              STA   LINNUM+1
9338: 68                 PLA
9339: A8                 TAY
933A: 4C 98 D9           JMP   ADDON       Add Y to TXTPTR
933D: A6 B8     :2       LDX   TXTPTR      Backup TXTPTR
933F: 86 06              STX   AUXPTR      before calling LINGET
9341: A6 B9              LDX   TXTPTR+1
9343: 86 07              STX   AUXPTR+1
9345: 20 0C DA           JSR   LINGET
                * Now TXTPTR points to the first non numeric character
                * following line no: computes the offset from current
                >233  * to stored position.
9348: 20 63 94 >234           JSR   NREMN      Compute Y offset to EOL
934B: A5 76    >235           LDA   CURLIN+1
934D: C5 51    >236           CMP   LINNUM+1
934F: B0 0C    >237           BCS   :1
9351: 98       >238           TYA
9352: 38       >239           SEC
9353: 65 B8    >240           ADC   TXTPTR
9355: A6 B9    >241           LDX   TXTPTR+1
9357: 90 08    >242           BCC   :3
9359: E8       >243           INX
935A: B0 05    >244           BCS   :3         Always
935C: 60       >245  ]RET     RTS
935D: A5 67    >246  :1       LDA   TXTTAB
935F: A6 68    >247           LDX   TXTTAB+1
9361: 20 1A D6 >248  :3       JSR   FNDLIN
9364: 90 4E    >249           BCC   GOUNDEF
9366: 2C D0 9C >250           BIT   OPTCGOTO
9369: 10 F1    >251           BPL   ]RET       Optimization deactivated
936B: A5 B8    >252           LDA   TXTPTR
936D: E5 06    >253           SBC   AUXPTR
936F: A8       >254           TAY
                >255  * Y should be 3, 4 or 5 (line no from 100 to 99999)
9370: A5 B9    >256           LDA   TXTPTR+1
9372: E5 07    >257           SBC   AUXPTR+1   Carry deja a 1
9374: D0 E6    >258           BNE   ]RET       hi byte must be zero
9376: 88       >259           DEY
9377: 88       >260           DEY
9378: 88       >261           DEY
9379: 30 E1    >262           BMI   ]RET       If Y was below 3
937B: C0 03    >263           CPY   #3         If Y was above 5
937D: B0 DD    >264           BCS   ]RET
937F: 84 B5    >265           STY   YSAV       possible values: 0, 1 or 2
9381: A5 9B    >266           LDA   LOWTR
9383: 38       >267           SEC
9384: E5 67    >268           SBC   TXTTAB
9386: AA       >269           TAX
9387: A5 9C    >270           LDA   LOWTR+1
9389: E5 68    >271           SBC   TXTTAB+1   Leaves carry always set..
938B: 2C D0 9C >272           BIT   OPTCGOTO
938E: 50 0F    >273           BVC   :6         Configured to skip checks..
9390: A8       >274           TAY              ;Set Z flag after BIT op
9391: 20 5B 94 >275           JSR   COMRG
9394: F0 C6    >276           BEQ   ]RET
9396: 8A       >277           TXA
9397: 20 5B 94 >278           JSR   COMRG
939A: F0 C0    >279           BEQ   ]RET
939C: 98       >280           TYA
939D: A4 B5    >281           LDY   YSAV
939F: C8       >282  :6       INY
93A0: C8       >283           INY
93A1: 91 06    >284           STA   (AUXPTR),Y
93A3: 88       >285           DEY
93A4: 8A       >286           TXA
93A5: 91 06    >287           STA   (AUXPTR),Y
93A7: 88       >288           DEY
93A8: 98       >289           TYA
```

```
93A9: 69 1C              ADC  #$1D-1     Carry originally set
93AB: 91 06     ]LOOP    STA  (AUXPTR),Y
93AD: 88                 DEY
93AE: 10 FB              BPL  ]LOOP
93B0: 8C CF 9C           STY  NEEDDEC    Set "Need Decompile" indic.
93B3: 60        ]RET     RTS

93B4: 4C 7C D9  GOUNDEF  JMP  $D97C

                * Routine to restore things at their original state
                * This routine should be called upon LIST or a SAVE
                * command under DOS 3.3.
                DECOMPILE
93B7: 08                 PHP
93B8: 48                 PHA
93B9: 2C CF 9C           BIT  NEEDDEC
93BC: 10 41              BPL  FINDEC
93BE: A5 67              LDA  TXTTAB
93C0: A6 68              LDX  TXTTAB+1
93C2: A0 00              LDY  #0
93C4: 8C CF 9C           STY  NEEDDEC
93C7: 85 06     ]LOOP    STA  AUXPTR
93C9: 86 07              STX  AUXPTR+1
93CB: 84 C0              STY  GFLAG      Set b7 to 0
93CD: 8A                 TXA
93CE: F0 2F              BEQ  FINDEC
93D0: A0 03              LDY  #3
93D2: C8        ]LOOP1   INY
93D3: B1 06     ]LOOP2   LDA  (AUXPTR),Y
93D5: F0 1D              BEQ  FINLIGNE
93D7: C9 22              CMP  #´"´
93D9: D0 08              BNE  :0
93DB: AA                 TAX
93DC: A5 C0              LDA  GFLAG
93DE: 49 80              EOR  #$80
93E0: 85 C0              STA  GFLAG
93E2: 8A                 TXA
93E3: 24 C0     :0       BIT  GFLAG
93E5: 30 EB              BMI  ]LOOP1
93E7: C9 20              CMP  #$20
93E9: B0 E7              BCS  ]LOOP1
93EB: E9 1C              SBC  #$1D-1
93ED: 90 E3              BCC  ]LOOP1
93EF: 20 02 94           JSR  TRAITEOK
93F2: F0 DF              BEQ  ]LOOP2     Always
93F4: A0 01     FINLIGNE LDY  #1
93F6: B1 06              LDA  (AUXPTR),Y
93F8: AA                 TAX
93F9: 88                 DEY
93FA: B1 06              LDA  (AUXPTR),Y
93FC: 4C C7 93           JMP  ]LOOP
93FF: 68        FINDEC   PLA
9400: 28                 PLP
9401: 60        ]RET     RTS

                * A: 0, 1 or 2 depending of length of org target line no
                * Y: offset from AUXPTR where first pattern byte appeared
                * Carry: must be set upon entry
9402: 85 B4     TRAITEOK STA   XSAV
9404: 98                 TYA
9405: 48                 PHA
9406: 98                 TYA
9407: 65 B4              ADC   XSAV       Carry set upon entry
9409: A8                 TAY
940A: 18                 CLC
                * Now Y: offset from AUXPTR where to get the
                *        target line adress offset
                * CLC (carry already clear after ADC above).
940B: B1 06              LDA   (AUXPTR),Y or stick to 8bits arithmetic
940D: 65 67              ADC   TXTTAB
940F: 85 9B              STA   LOWTR
9411: C8                 INY
9412: B1 06              LDA   (AUXPTR),Y
9414: 65 68              ADC   TXTTAB+1
9416: 85 9C              STA   LOWTR+1
9418: A0 03              LDY   #3
941A: B1 9B              LDA   (LOWTR),Y
941C: 85 9E              STA   $9E
941E: 88                 DEY
941F: B1 9B              LDA   (LOWTR),Y
9421: 85 9F              STA   $9F
9423: A2 90              LDX   #$90       Get line #
9425: 38                 SEC              ; in ASCII form
9426: 20 A0 EB           JSR   $EBA0      stored into $100
9429: 20 34 ED           JSR   FOUT
942C: 20 52 94           JSR   CLENGTH    Length of string in X
942F: 86 B5              STX   YSAV
9431: 68                 PLA
9432: A8                 TAY
9433: A6 B4              LDX   XSAV
9435: E8                 INX
9436: E8                 INX
9437: E8                 INX
9438: 8A                 TXA
9439: 38                 SEC
943A: E5 B5              SBC   YSAV
943C: AA                 TAX
943D: F0 08              BEQ   :0
943F: A9 30              LDA   #´0´
9441: 91 06     ]LOOP    STA   (AUXPTR),Y
9443: C8                 INY
9444: CA                 DEX
9445: D0 FA              BNE   ]LOOP
9447: BD 00 01  :0       LDA   $0100,X
944A: F0 B5              BEQ   ]RET
944C: 91 06              STA   (AUXPTR),Y
944E: C8                 INY
944F: E8                 INX
9450: D0 F5              BNE   :0         Always

9452: A2 FF     CLENGTH  LDX   #255
9454: E8        ]LOOP    INX
9455: BD 00 01           LDA   $0100,X
9458: D0 FA              BNE   ]LOOP
```

```
945A: 60        >428           RTS
                >429
                >430  * Small subroutine to test for critical offset value
                >431  * against insert into program text
945B: F0 02     >432  COMRG    BEQ   *+4
945D: 49 3A     >433           EOR   #´:´
945F: 60        >434  ]RET     RTS
                >435
                >436  CHARAC   EQU   $0D
                >437
9460: A2 3A     >438  NDATAN   LDX   #´:´
9462: 2C        >439           HEX   2C          Skip next two bytes
9463: A2 00     >440  NREMN    LDX   #0
9465: 86 0D     >441           STX   CHARAC
9467: A0 00     >442           LDY   #0
9469: 84 0E     >443           STY   ENDCHR
946B: A5 0E     >444  ]LOOP1   LDA   ENDCHR      Trick to count for Quote Parity
946D: A6 0D     >445           LDX   CHARAC      Do not stop upon ´:´ within
946F: 85 0D     >446           STA   CHARAC      a string litteral
9471: 86 0E     >447           STX   ENDCHR
9473: B1 B8     >448  ]LOOP    LDA   (TXTPTR),Y
9475: F0 E8     >449           BEQ   ]RET
9477: C5 0E     >450           CMP   ENDCHR
9479: F0 E4     >451           BEQ   ]RET
947B: C8        >452           INY
947C: C9 22     >453           CMP   #´"´
947E: F0 EB     >454           BEQ   ]LOOP1
9480: C9 20     >455           CMP   #´ ´
9482: B0 EF     >456           BCS   ]LOOP
9484: E9 1C     >457           SBC   #$1D-1      Substract $1D (carry clear)
9486: 90 EB     >458           BCC   ]LOOP       Out of scope..
9488: C8        >459           INY
9489: AA        >461           TAX               ;Possible values for X: 0, 1 or 2
948A: C8        >463  ]LOOP1   INY
948B: CA        >465           DEX
948C: 10 FC     >469           BPL   ]LOOP1
948E: 30 E3     >470           BMI   ]LOOP       Always
                >471
9490: C8        >472  ]LOOP    INY
9491: C8        >473           INY
9492: 20 98 D9  >474           JSR   ADDON
9495: 20 47 75  >475  LRST100  JSR   RST100
9498: AA        >476           TAX
9499: 90 0A     >477           BCC   :RETS+1
949B: E9 1D     >478           SBC   #$1D
949D: A8        >479           TAY
949E: 90 04     >480           BCC   :RETS
94A0: C0 03     >481           CPY   #3
94A2: 90 EC     >482           BCC   ]LOOP
94A4: 38        >483  :RETS    SEC
94A5: 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
```

```

               IBUFFER  EQU   $0200
               STRTXT   EQU   $DE81
               STRPRT   EQU   $DB3D
               OUTQUES  EQU   $DB5A
               INLIN    EQU   $D52C
               RDKEY    EQU   $FD0C
               STRLT2   EQU   $E3ED
               NXIN     EQU   $DBDC

94A6: 20 06 E3 RGET     JSR   ERRDIR
94A9: A2 01             LDX   #IBUFFER+1
94AB: A0 02             LDY   #>IBUFFER+1
94AD: A9 00             LDA   #0
94AF: 8D 01 02          STA   IBUFFER+1
94B2: A9 40             LDA   #$40       Setup INPUTFLG
94B4: D0 33             BNE   PIL        for PROCESS.INPUT.LIST: always

94B6: C9 22    RINP     CMP   #´"´       Check for optional prompt
94B8: D0 0E             BNE   :1          string
94BA: 20 81 DE          JSR   STRTXT
94BD: A9 3B             LDA   #´;´
94BF: 20 CE 7D          JSR   NSYNCHR
94C2: 20 3D DB          JSR   STRPRT     Print the string
94C5: 4C CB 94          JMP   :2
94C8: 20 5A DB :1       JSR   OUTQUES
94CB: 20 06 E3 :2       JSR   ERRDIR
94CE: A9 2C             LDA   #´,´       Prime the buffer
94D0: 8D FF 01          STA   IBUFFER-1
94D3: 20 2C D5          JSR   INLIN
94D6: AD 00 02          LDA   IBUFFER
94D9: C9 03             CMP   #$03       Control-C?
94DB: D0 0A             BNE   IFZ
94DD: 4C 63 D8          JMP   $D863

94E0: A6 7D    RREAD2   LDX   DATPTR
94E2: A4 7E             LDY   DATPTR+1
94E4: A9 98             LDA   #$98
94E6: 2C                HEX   2C         Skip next two bytes
94E7: A9 00    IFZ      LDA   #0

               * For PROCESS.INPUT.LIST
94E9: 85 15    PIL      STA   INPUTFLG
94EB: 86 7F             STX   INPTR
94ED: 84 80             STY   INPTR+1

               * For PROCESS.INPUT.ITEM
94EF: 20 42 79 PII      JSR   NPTRGTX
94F2: 85 85             STA   FORPNT
94F4: 84 86             STY   FORPNT+1
94F6: A5 B8             LDA   TXTPTR     Save current TXTPTR
94F8: A4 B9             LDY   TXTPTR+1
94FA: 85 87             STA   TXPSV
94FC: 84 88             STY   TXPSV+1
94FE: A5 7F             LDA   INPTR
9500: A4 80             LDY   INPTR+1    Set TXTPR to point to input
9502: 85 B8             STA   TXTPTR      buffer or "DATA" line
```

```
9504: 84 B9              STY   TXTPTR+1
9506: 20 65 75           JSR   RST102     Get character at pointer
9509: D0 1E              BNE   INSTART    Not eol or colon.
950B: 24 15              BIT   INPUTFLG
950D: 50 0E              BVC   :1         Not doing a GET
950F: 20 0C FD           JSR   RDKEY
9512: 29 7F              AND   #$7F
9514: 8D 00 02           STA   IBUFFER
9517: A2 FF              LDX   #IBUFFER-1
9519: A0 01              LDY   #>IBUFFER-1
951B: D0 08              BNE   :2         Always
951D: 30 7C    :1        BMI   FINDATA    doing a READ
951F: 20 5A DB           JSR   OUTQUES
9522: 20 DC DB           JSR   NXIN       Print another "?" & input a line
9525: 86 B8    :2        STX   TXTPTR
9527: 84 B9              STY   TXTPTR+1
9529: 20 47 75 INSTART   JSR   RST100
952C: 24 11              BIT   VALTYP     String or numeric variable?
952E: 10 35              BPL   :5
9530: 24 15              BIT   INPUTFLG
9532: 50 09              BVC   :1         Not a "GET"
9534: E8                 INX              ;GET
9535: 86 B8              STX   TXTPTR
9537: A9 00              LDA   #0
9539: 85 0D              STA   CHARAC     No other terminator character
953B: F0 10              BEQ   :2
953D: 85 0D    :1        STA   CHARAC
953F: C9 22              CMP   #´"´
9541: F0 0B              BEQ   :3
9543: A5 15              LDA   INPUTFLG   Applesoft bug fix
9545: F0 02              BEQ   *+4
9547: A9 3A              LDA   #´:´
9549: 85 0D              STA   CHARAC
954B: A9 2C              LDA   #´,´
954D: 18       :2        CLC
954E: 85 0E    :3        STA   ENDCHR
9550: A5 B8              LDA   TXTPTR
9552: A4 B9              LDY   TXTPTR+1
9554: 69 00              ADC   #0         Skip over quotation mark, if
9556: 90 01              BCC   :4         there was one
9558: C8                 INY
9559: 20 ED E3 :4        JSR   STRLT2     Build string starting at Y,A
955C: 20 3D E7           JSR   $E73D      Set TXTPTR to point at string
955F: 20 7B DA           JSR   $DA7B      PUTSTR
9562: 4C 74 95           JMP   PIM
9565: 48       :5        PHA
9566: AD 00 02           LDA   IBUFFER    ANything in buffer?
9569: F0 59              BEQ   INPFIN     No: see if READ or INPUT
956B: 68       INPDATA   PLA              ;READ
956C: 20 4A EC           JSR   $EC4A      FIN: get FP number at TXTPTR
956F: A5 12              LDA   INTTYP
9571: 20 39 76           JSR   NLET2
               * For PROCESS.INPUT.MORE
9574: 20 65 75 PIM       JSR   RST102
9577: F0 07              BEQ   :1         End of line or colon
9579: C9 2C              CMP   #´,´       Comma in input?
957B: F0 03              BEQ   :1         Yes
```

```
957D: 4C 71 DB              JMP    $DB71      Nothing else will do
9580: A5 B8       :1        LDA    TXTPTR
9582: A4 B9                 LDY    TXTPTR+1
9584: 85 7F                 STA    INPTR
9586: 84 80                 STY    INPTR+1
9588: A5 87                 LDA    TXPSV      Restore program pointer
958A: A4 88                 LDY    TXPSV+1
958C: 85 B8                 STA    TXTPTR
958E: 84 B9                 STY    TXTPTR+1
9590: 20 65 75              JSR    RST102     next char from program
9593: F0 36                 BEQ    INPDONE    End if statement
9595: 20 49 86              JSR    NCHKCOM
9598: 4C EF 94              JMP    PII

959B: 20 A3 D9    FINDATA   JSR    DATAN      Get offset to next colon/eol
959E: C8                    INY
959F: AA                    TAX               ;Which colon or eol?
95A0: D0 15                 BNE    :1         Colon
95A2: C8                    INY               ;Check hi byte
95A3: B1 B8                 LDA    (TXTPTR),Y
95A5: D0 05                 BNE    *+7
95A7: A2 2A                 LDX    #$2A       NODATA ERROR
95A9: 4C 12 D4              JMP    $D412
95AC: C8                    INY               ;Pick up the line #
95AD: B1 B8                 LDA    (TXTPTR),Y
95AF: 85 7B                 STA    DATLIN
95B1: C8                    INY
95B2: B1 B8                 LDA    (TXTPTR),Y
95B4: C8                    INY
95B5: 85 7C                 STA    DATLIN+1
95B7: B1 B8       :1        LDA    (TXTPTR),Y Get 1st token of statement
95B9: AA                    TAX               ;Save token in X reg.
95BA: 20 98 D9              JSR    ADDON      Add Y to TXTPTR
95BD: E0 83                 CPX    #TOKDATA
95BF: D0 DA                 BNE    FINDATA
95C1: 4C 29 95              JMP    INSTART

95C4: A5 15       INPFIN    LDA    INPUTFLG
95C6: D0 A3                 BNE    INPDATA
95C8: 4C 86 DB              JMP    $DB86

95CB: 4C C6 DC    INPDONE   JMP    $DCC6

                  FCODE     EQU    *

                            PUT    PEERGDATA
95CE: 00 00 00    SVPTR     DS     18
95E0: 00          SVP2      DFB    0

95E1: 00 00 00    TABOFB    DFB    0,0,0,0,0,0,0,0
95E9: 00 00 00    TABOFT    DFB    0,0,0,0,0,0,0,0
95F1: 00          INDX      DFB    0
95F2: 00          SPROOT    DFB    0
95F3: 00 00       ITVADDR   DA     0          Adresse de la var. ITHREAD%
95F5: F8 75 76    P0OFFSET DFB     REMSTK,CURLIN,CURLIN+1,TXTPTR,TXTPTR+1
95FA: 79 7A                 DFB    OLDTEXT,OLDTEXT+1
                  PIOFFSET EQU     *
```

```
95FC: F4 F5 F6            DFB   TXTPSV,TXTPSV+1,CURLSV,CURLSV+1,ERRNUM
9601: DF DA DB            DFB   ERRSTK,ERRLIN,ERRLIN+1,ERRPOS,ERRPOS+1
9606: D8                  DFB   ERRFLG
                PEOFFSET EQU    *
9607: C9 C6 C2  TOKMOTIF DFB    TOKMINUS,TOKNOT,TOKFN,TOKSCRN
                TOKMTIFE
960B: CD 8F 53  TOKMPFB  DFB    $DECE-1,$DE90-1,$E354-1,$DEF9-1
960F: DE DE E3  TOKMPFT  DFB    >$DECE-1,>$DE90-1,>$E354-1,>$DEF9-1
9613: C8 C9 CA  TOKENS   DFB    TOKADD,TOKMINUS,TOKMUL,TOKDIV

9617: BD A6 7E  FPROUTSB DFB    FADD-1,FSUB-1,FMULT-1,FDIV-1
961B: E7 E7 E9  FPROUTST DFB    >FADD-1,>FSUB-1,>FMULT-1,>FDIV-1
                * Motifs used inside FOR/NEXT loop handling
                * to restore full byte patterns from two bits
961F: 00 01     MOTGF    DFB    0,1
9621: 00                 DS     1
9622: FF                 HEX    FF
                * Where is stored the elm. index in a FOREACH loop
9623: 00 00     AEI      DA     0

9625: CA D5     OFFSTB   DFB    HNDLSIAD-1,HNDLSIMI-1
9627: F5 F4              DFB    HNDLSIMU-1,HNDLSIDV-1
9629: 1B 28              DFB    HNDLSBAD-1,HNDLSBMI-1
962B: 30 4E              DFB    HNDLSBMU-1,HNDLSBDV-1
962D: B2 BE              DFB    HNDLUIAD-1,HNDLUIMI-1
962F: E2 E1              DFB    HNDLUIMU-1,HNDLUIDV-1
9631: 14 21              DFB    HNDLUBAD-1,HNDLUBMI-1
9633: 2F 4D              DFB    HNDLUBMU-1,HNDLUBDV-1
9635: 76 76     OFFSTT   DFB    >HNDLSIAD-1,>HNDLSIMI-1
9637: 76 76              DFB    >HNDLSIMU-1,>HNDLSIDV-1
9639: 77 77              DFB    >HNDLSBAD-1,>HNDLSBMI-1
963B: 77 77              DFB    >HNDLSBMU-1,>HNDLSBDV-1
963D: 76 76              DFB    >HNDLUIAD-1,>HNDLUIMI-1
963F: 76 76              DFB    >HNDLUIMU-1,>HNDLUIDV-1
9641: 77 77              DFB    >HNDLUBAD-1,>HNDLUBMI-1
9643: 77 77              DFB    >HNDLUBMU-1,>HNDLUBDV-1

9645: 00 00 00  ADRSTRUCT DS    11*LENREC
972C: F8        SVOFST   DFB    REMSTK
972D: B8 B9              DFB    TXTPTR,TXTPTR+1
972F: 75 76              DFB    CURLIN,CURLIN+1
9731: 79 7A              DFB    OLDTEXT,OLDTEXT+1
9733: F2                 DFB    TRCFLG
9734: A5 A6 A7           DFB    ARG,ARG+1,ARG+2,ARG+3,ARG+4,$AA
                FINOF    EQU    *
973A: 00 00 00  SVAREA   DS     FINOF-SVOFST

9748: 00 00 00  SVCURRM  DS     12
9754: 00 00 00  SVALTNM  DS     12

                * Structure juste pour la prise en compte lors du DEFUSR
9760: 00 00 00  ]DEBUT   DS     8
                ]FIN
9768: 60 97     SDEF1    DA     ]DEBUT     pour VARTAB
976A: 60 97              DA     ]DEBUT     pour ARYTAB
976C: 60 97              DA     ]DEBUT     pour STREND
976E: 68 97              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    OLDVECT  DA    0

99D1: 00       WORKPL1  DS    1
99D2: 00       MIRQST   DS    1
               * YICUR: indique quel est le dernier
               * handler d´interruption retenu
99D3: FF       YICUR    DFB   $FF

               * Deux slots pour chaque entree
               * Indices:
               * 0: pour l´API MOUSE
               * 1: pour l´API TIMER
               * MODERUN: 1 iif routine en cours, 0 sinon
99D4: 00 00    MODERUN  DS    2
               * MODEPEC:
               * 0: non prise en compte de l´interruption
               * 1: prise en compte retardee
               * 2: prise en compte immediate
99D6: 00 00    MODEPEC  DS    2
99D8: 40 80    MSKINT   HEX   4080
               * Values of S to cmp upon return from Applesoft
               * handling routine (usually RETURN)
99DA: 00 00    INTSPTR  DS    2

99DC: 00 00    CLN_B    DS    2          Interrupted line # lo byte
99DE: 00 00    CLN_T    DS    2          Interrupted line # hi byte
99E0: 00 00    TPT_B    DS    2          Interrupted text ptr lo byte
99E2: 00 00    TPT_T    DS    2          Interrupted text ptr hi byte
99E4: 00 00    OTPT_B   DS    2          Interrupted OLDTEXT lo byte
99E6: 00 00    OTPT_T   DS    2          Interrupted OLDTEXT hi byte

               * Offsets from page zero to save for WAIT
99E8: 50 51    SVWOF    DFB   LINNUM,LINNUM+1
99EA: 85 86             DFB   FORPNT,FORPNT+1
99EC: B8 B9             DFB   TXTPTR,TXTPTR+1
               * Save area for WAIT
99EE: 00 00 00 SVA      DS    6
99F4: 00       FRGNDCTX DFB   0          5 pour WAIT

               * KTINC factor for timer interrupt (default 1)
99F5: 01 00    KTINC    DA    1          config. value for timer factor
99F7: 00 00    TIINC    DA    0          runtime value for timer factor

               * Error messages
               MESSERR
               MESER1   EQU   *-MESSERR
99F9: 4D 4F 55          DCI   ´MOUSE HARDWARE NOT DETECTED´
               MESER2   EQU   *-MESSERR
9A14: 55 4E 53          DCI   ´UNSUPPORTED HARDWARE CONFIGURATION´
               MESER3   EQU   *-MESSERR
9A36: 55 4E 4B          DCI   ´UNKNOWN APPLESOFT MOUSE EVENT HANDLER´
               MESER4   EQU   *-MESSERR
9A5B: 55 4E 4B          DCI   ´UNKNOWN APPLESOFT TIMER EVENT HANDLER´
               MESER5   EQU   *-MESSERR
9A80: 49 4C 4C          DCI   ´ILLEGAL MOUSE MODE´
               MESER6   EQU   *-MESSERR
9A92: 49 4C 4C          DCI   ´ILLEGAL MOUSE OPERATION´
```

```
                    MESER7   EQU   *-MESSERR
9AA9: 5A 45 52               DCI   ´ZERO TARGET ADDRESS´
                    MESER8   EQU   *-MESSERR
9ABC: 45 4D 42               DCI   ´EMBEDDED PF NOT SUPPORTED IN THIS RELEASE´
                    MESER9   EQU   *-MESSERR
9AE5: 49 4C 4C               DCI   ´ILLEGAL OP WHILE PF IS ACTIVE´
9B02: 00 1B 3D      CODR     DFB   MESER1,MESER2,MESER3,MESER4,MESER5,MESER6
9B08: B0 C3 EC               DFB   MESER7,MESER8,MESER9

9B0B: 91 80 00      NEG65536 HEX   9180000000
9B10: 90 80 00      NEG32768 HEX   9080000000
9B15: 90 00 00      POS32768 HEX   9000000000
9B1A: 91 00 00      POS65536 HEX   9100000000

                    * Table of new Peersoft commands
9B1F: C8            TMOCL    DFB   TOKADD
9B20: D0                     DFB   TOKEQUAL
9B21: 00                     DFB   0
9B22: C9                     DFB   TOKMINUS
9B23: D0                     DFB   TOKEQUAL
9B24: 00                     DFB   0
9B25: CA                     DFB   TOKMUL
9B26: D0                     DFB   TOKEQUAL
9B27: 00                     DFB   0
9B28: CB                     DFB   TOKDIV
9B29: D0                     DFB   TOKEQUAL
9B2A: 00                     DFB   0
9B2B: 40                     ASC   ´@´
9B2C: 00                     DFB   0
9B2D: 23                     ASC   ´#´
9B2E: 00                     DFB   0
9B2F: 4F 46 46               ASC   ´OFF´
9B32: 00                     DFB   0
9B33: 49            IFIIF    ASC   ´I´
9B34: AD                     DFB   TOKIF
9B35: 00                     DFB   0
9B36: 4D 4F 55               ASC   ´MOUSE´
9B3B: 00                     DFB   0
9B3C: 54 49 4D               ASC   ´TIMER´
9B41: 00                     DFB   0
9B42: B8            IFDEF    DFB   TOKDEF
9B43: D5                     DFB   TOKUSR
9B44: 00                     DFB   0
9B45: B8                     DFB   TOKDEF
9B46: 53 54 52               ASC   ´STR´
9B49: 00                     DFB   0
9B4A: B8                     DFB   TOKDEF
9B4B: 53 4E 47               ASC   ´SNG´
9B4E: 00                     DFB   0
9B4F: B8                     DFB   TOKDEF
9B50: D3                     DFB   TOKINT
9B51: 00                     DFB   0
9B52: B8                     DFB   TOKDEF
9B53: 42 59 54               ASC   ´BYTE´
9B57: 00                     DFB   0
9B58: 81                     DFB   TOKFOR
9B59: 45 41 43      IFEACH   ASC   ´EACH´
```

```
9B5D: 00                  DFB   0
9B5E: FF                  HEX   FF

9B5F: 00 03 06  TOFFST    DFB   0,3,6,9      Pour les 4 syntax schemes
                          ERR   NOPER-4
9B63: 0C                  DFB   12           Pour le symbole @
9B64: 0E                  DFB   14           Pour le symbole #
9B65: 10        OFFOFF    DFB   16           Pour le mot cle OFF
9B66: 14        OFFIIF    DFB   20           Pour la fonction IIF()
9B67: 17        OFFMOU    DFB   23           Pour le mot cle MOUSE
9B68: 1D        OFFTIM    DFB   29           Pour le mot cle TIMER
9B69: 23        OFFUSR    DFB   35           Pour le mot cle DEFUSR
9B6A: 26 2B 30  OFFDEF    DFB   38,43,48     pour les intr. DEFSTR,SNG,INT...
9B6D: 33                  DFB   51           Pour le DEFBYTE
9B6E: 39                  DFB   57           Pour le FOREACH
9B6F: 3F                  DFB   63

                * Ou commencer la recherche?
                * au debut (LIST)
9B70: FF        TIDMOCL   DFB   0-1
                * instruction DEF<pattern>
9B71: 09                  DFB   OFFUSR-TOFFST-1
                * sur la premiere fonction (IIF/MOUSE/TIMER)
9B72: 06                  DFB   OFFIIF-TOFFST-1
                * fonction MOUSE ou TIMER
9B73: 07                  DFB   OFFMOU-TOFFST-1
9B74: 08                  DFB   OFFTIM-TOFFST-1
                * Juste mot-cle OFF
9B75: 05                  DFB   OFFOFF-TOFFST-1
                * Quoi mettre a l´offset OFFDEF
9B76: B8        TOFFIN    DFB   TOKDEF       si LIST
9B77: B8                  DFB   TOKDEF       si DEF<pattern>
9B78: FF                  HEX   FF           si IIF/MOUSE/TIMER
9B79: FF                  HEX   FF           si MOUSE/TIMER
9B7A: FF                  HEX   FF           si TIMER
9B7B: FF                  HEX   FF           si OFF
                * Quoi mettre a l´offset OFFIFF
9B7C: 49        TOFFIN2   DFB   ´I´          ;si LIST
9B7D: 49                  DFB   ´I´          ;si DEF<pattern>
9B7E: 49                  DFB   ´I´          ;si IFF/MOUSE/TIMER
9B7F: 49                  DFB   ´I´          ;si MOUSE/TIMER
9B80: 49                  DFB   ´I´          ;si TIMER
9B81: FF                  HEX   FF           si OFF
9B82: 24 21 25  MOTIF     ASC   ($!%(
9B85: 2E                  ASC   (.(
9B86: 00 00 82  TITVAL    HEX   00008281     What to store into INTTYP
9B8A: FF 00 00  TVTVAL    HEX   FF000000     What to store into VALTYP
9B8E: 00 00 80  TVNORA    HEX   00008080     Value to ORA with VARNAM
9B92: 80 00 80  TVN1ORA   HEX   80008080     Value to ORA with VARNAM+1

9B96: 21 21 21  TYPLET    DS    26,´!´

                * Applesoft standard instructions entry points
                APRWAIT   EQU   $E784        WAIT instruction entry point
                APRRUN    EQU   $D912        RUN instruction entry point
                APRLIST   EQU   $D6A5        LIST instruction entry point
                APRCLEAR  EQU   $D66A        CLEAR instruction entry point
                APRDEF   EQU   $E313      DEF instruction entry point
                APRLET   EQU   $DA46      LET instruction entry point
                APRFOR   EQU   $D766      FOR instruction entry point
                APRNEXT  EQU   $DCF9      NEXT instruction entry point
                APFRMELM EQU   $DE67      Return address from FRMELM
                APRETURN EQU   $D96B      RETURN/POP instr. entry point
                APRONERR EQU   $F2CB      ONERR instruction entry point
                APRNEW   EQU   $D649      NEW instruction entry point
                APRGOTO  EQU   $D93E      GOTO instruction entry point
                APRGOSUB EQU   $D921      GOSUB instruction entry point
                APRIF    EQU   $D9C9      IF instruction entry point
                APRON    EQU   $D9EC      ON expr GOTO/GOSUB entry point
                APRGET   EQU   $DBA0
                APRINP   EQU   $DBB2
                APRREAD  EQU   $DBE2

9BB0: 83        ADAPFBET DFB   APRWAIT-1
9BB1: 11 48 A4           DFB   APRRUN-1,APRNEW-1,APRLIST-1,APRCLEAR-1
9BB5: CA 12 45           DFB   APRONERR-1,APRDEF-1,APRLET-1
9BB8: E1 B1 9F           DFB   APRREAD-1,APRINP-1,APRGET-1
9BBB: 65 EB 3D           DFB   APRFOR-1,APRON-1,APRGOTO-1,APRIF-1,APRETURN-
1,APRGOSUB-1
9BC1: F8 66              DFB   APRNEXT-1,APFRMELM-1
9BC3: 1F                 DFB   $D820-1
9BC4: E7        ADAPFTET DFB   >APRWAIT-1
9BC5: D9 D6 D6           DFB   >APRRUN-1,>APRNEW-1,>APRLIST-1,>APRCLEAR-1
9BC9: F2 E3 DA           DFB   >APRONERR-1,>APRDEF-1,>APRLET-1
9BCC: DB DB DB           DFB   >APRREAD-1,>APRINP-1,>APRGET-1
9BCF: D7 D9 D9           DFB   >APRFOR-1,>APRON-1,>APRGOTO-1,>APRIF-1,>APRE
TURN-1,>APRGOSUB-1
9BD5: DC DE              DFB   >APRNEXT-1,>APFRMELM-1
9BD7: D8                 DFB   >$D820-1
9BD8: 07        ADPFB    DFB   RWAIT-1
9BD9: B2 BB 85           DFB   RRUN-1,RNEW-1,STDLIS-1,RCLEAR-1
9BDD: 06 FD A0           DFB   RONERR-1,RDEF-1,RLET-1
9BE0: DF B5 A5           DFB   RREAD2-1,RINP-1,RGET-1
9BE3: DE 56              DFB   RFOR-1,RON-1
9BE5: 07 E0 FC  OFSTGTO  DFB   RGOTO-1,RIF-1,RRETURN-1,RGOSUB-1
9BE9: B7 E8              DFB   RNEXT-1,FRMELM-1
9BEB: 5B                 DFB   RNEWINST-1
9BEC: 8E        ADPFT    DFB   >RWAIT-1
9BED: 7D 7D 82           DFB   >RRUN-1,>RNEW-1,>STDLIS-1,>RCLEAR-1
9BF1: 84 80 75           DFB   >RONERR-1,>RDEF-1,>RLET-1
9BF4: 94 94 94           DFB   >RREAD2-1,>RINP-1,>RGET-1
9BF7: 8E 92 93           DFB   >RFOR-1,>RON-1,>RGOTO-1,>RIF-1,>RRETURN-1,>R
GOSUB-1
9BFD: 8F 84              DFB   >RNEXT-1,>FRMELM-1
9BFF: 8D                 DFB   >RNEWINST-1
                FIN
                LONGLANG EQU   *-CGARBAG
                         ERR   *-$9C00

                         PUT   PEERGLOBALPAGE
                         DUMMY $9CBD
9CBD: 00        FLGFN    DS    1
9CBE: 00 00 00  WRKFA    DS    5          FAC work area A
9CC3: 00 00 00  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: CA 7A    >15   VNARRG91 DA    NARRGL91   Look up array name in memory
9CD3: B8 79    >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 A4 86 >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, 11863 bytes, Errors: 0


Symbol table - alphabetical order:

   A1L     =$3C         A2L     =$3E         A4L     =$42         ABSOL8  =$7850
   ABSOLUTE=$7913   ?  ACTR    =$9B         ADADR   =$9CD8       ADAPFBET=$9BB0
   ADAPFTET=$9BC4       ADB1    =$41E6       ADB2    =$4200       ADDON   =$D998
   ADPFB   =$9BD8       ADPFT   =$9BEC       ADRSTRUCT=$9645       ADRUSR  =$01
   ADT1    =$41F3       ADT2    =$420D       AEI     =$9623       AHNDHI  =$99BF
   AHNDLO  =$99BD       AIT     =$99A1       ALKCACH =$7CF2       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    =$7904       ARG     =$A5
   AROMBA  =$477B       ARYPNT  =$94         ARYTAB  =$6B         ARYVAR  =$D039
   AUXBANK =$BF         AUXPTR  =$06         AVN     =$9999       AVNP1   =$999D
   AXARTAB =$D099   ?  AXARYPNT=$D099   ?  AXARYPT2=$D09E       AXHIMEM =$BF00
?  AXOFFSET=$D09B       AXVALUE =$D09E   ?  AYINT   =$E10C       BADNAM  =$7971
   BAMBS   =$0778       BANCLD  =$81C2       BAXHI   =$0578       BAXLO   =$0478
   BAYHI   =$05F8       BAYLO   =$04F8       BIGRECON=$421A       BISVTYP =$BE
   BTMEL   =$D19D       CALLFUNC=$8632       CFA     =$42B4       CFM     =$42B0
```

```
    CGARBAG =$7524      CH      =$24        CHARAC  =$0D        CHKMEM  =$D3D6
    CHKNUM  =$DD6A      CHKSTR  =$DD6C      CLENGTH =$9452      CLNHI   =$99BB
    CLNLO   =$99B9      CLN_B   =$99DC      CLN_T   =$99DE      CMPCLAMP=$8A37
    CNVT1   =$7CEA      CODE1BF =$4464      CODE1GC =$45DE      CODE1GCF=$477B
    CODE1LC =$4520      CODE2BF =$4520      CODE2LC =$45C3      CODR    =$9B02
    COLLECTR=$80A4      COMBYTE =$E74C      COMCLAMP=$8A5A      COMCLEAR=$8A8F
    COMCMPLX=$8623      COMCOPY =$910A      COMFOR  =$8F73      COMINT1 =$8CB9
    COMINT2 =$8D0B      COMINT4 =$8C2B      COMLBS  =$8B0F      COMLET2 =$7EB7
    COMLISO =$83C5      COMMON  =$8BE3      COMMON9 =$8BDE      COMMONG =$7FAA
    COMNEXT =$9071      COMPOFST=$7E08      COMPOS  =$8AC2      COMREAD =$8A94
    COMREST =$8096      COMRG   =$945B      COMRST  =$7553      COMRSTC =$755B
    COMWAIT =$8E1B      COMX1   =$7E7C      CONINT  =$E6FB      CONV1628=$7926
    COPYROM =$432C      CRDO    =$DAFB      CTRACTV =$9CDB      CURLIN  =$75
    CURLSV  =$F6        DATA    =$D995      DATA1IDX=$42B8      DATA1VAL=$42BE
    DATAN   =$D9A3      DATLIN  =$7B        DATPTR  =$7D        DBUFP   =$9D00
    DEBUTGET=$7524      DEBUTGOT=$757E      DECOMPILE=$93B7      DECTPTR =$8197
    DEFFLG  =$C1        DEST    =$60        DIMFLG  =$10        DINSIRQV=$8A1B
    DIVEND  =$C2        DIVSOR  =$C0        DSCLEN  =$8F        DSCTMP  =$9D
    DVAR    =$D103      DVARS   =$D0F6      DVARTS  =$D18D      DVZERROR=$78A2
    E06     =$823A      EK      =$41C6    ? ELMSIZ  =$D09D   MD EMOV    =$8000
    ENDCHR  =$0E        ENDRNG  =$82A8      ERRDIR  =$E306      ERRFLG  =$D8
    ERRLIN  =$DA        ERRNUM  =$DE        ERRPOS  =$DC        ERRSTK  =$DF
    ERR_BSCR=$6B        ERR_RDIM=$78        ERR_SYNT=$10        EXFLG   =$AAB3
    EXPLIC? =$797C      FAC     =$9D        FACLO   =$A1        FACMO   =$A0
    FACSIGN =$A2        FADD    =$E7BE      FAE2    =$7C95      FAE3    =$7C96
    FCODE   =$95CE      FCOMP   =$EBB2      FDIV    =$EA66      FEFOR   =$8E88
    FENEXT  =$8F8D      FESTEP  =$8ED7      FIN     =$9C00      FINDATA =$959B
    FINDEC  =$93FF      FINLIGNE=$93F4      FINMOUSE=$8A91      FINOF   =$973A
?   FLGFN   =$9CBD      FMULT   =$E97F      FNDLIN  =$D61A    ? FNDVAR  =$D004
    FNDVAR2 =$7524      FNDVARX2=$D00B      FORPNT  =$85        FOUT    =$ED34
    FPROUTSB=$9617      FPROUTST=$961B    ? FREESPC =$71        FREFAC  =$E600
    FRETOP  =$6F        FRGNDCTX=$99F4      FRMELM  =$84E9      FRMELMLP=$84E6
    FRMEVL  =$DD7B      FRMNUM  =$DD67      FRMSTCK3=$DE20      FRSTIM  =$7E27
    FSUB    =$E7A7      G81     =$BFAB      G83     =$BFA4      GARBAG  =$E484
    GDVARTS =$D0FF      GETADR  =$E752      GETARY  =$E0ED      GETARY2 =$E0EF
    GETBYT  =$E6F8      GETSPA  =$E452      GFLAG   =$C0        GGO2TMER=$8706
    GIQERR2 =$7D42      GIVAYF  =$E2F2      GME     =$7C64      GN32768 =$8E5A
    GN65536 =$8E64      GNARRAY =$7B0F      GNPTRGET=$757B      GODVZERR=$EAE1
    GOIQ    =$846D      GOIQERR =$E199      GOOVFERR=$E8D5      GOSTLERR=$E5B2
?   GOSVCUR =$7E23      GOSYNERR=$7DD7      GOTMIERR=$DD76   MD GOTO    =$8000
    GOTOTAIL=$D95E      GOUNDEF =$93B4      GP32768 =$8E5F      GP65536 =$8E69
    GRBPAS  =$D09C      GSE     =$7C92      GSNERR2 =$7D3F      GSNERR3 =$816F
    GTFORPNT=$D365      GTLT    =$7962    ? GTMERR2 =$7D45      GZAUXRT =$BF00
    H16B    =$84D7      HE2E8   =$8659      HIMEM   =$73        HNDLEADR=$7EA4
    HNDLEBC =$7765      HNDLEIC =$770A      HNDLEINT=$7673      HNDLEIX =$7706
    HNDLEIY =$769D    ? HNDLEREA=$7621      HNDLESTR=$7646      HNDLSBAD=$771C
    HNDLSBDV=$774F      HNDLSBMI=$7729      HNDLSBMU=$7731      HNDLSIAD=$76CB
    HNDLSIDV=$76F5      HNDLSIMI=$76D6      HNDLSIMU=$76F6      HNDLUBAD=$7715
    HNDLUBDV=$774E      HNDLUBMI=$7722      HNDLUBMU=$7730      HNDLUIAD=$76B3
    HNDLUIDV=$76E2      HNDLUIMI=$76BF      HNDLUIMU=$76E3      HNOK    =$8DE2
    IBUFFER =$0200      ICTRACTV=$9CDD      IDMOCL  =$BD        IDX0    =$C0
    IFDEF   =$9B42      IFEACH  =$9B59      IFIIF   =$9B33      IFZ     =$94E7
    INDEX   =$5E        INDX    =$95F1      INHACTV =$9CDA      INITBF  =$43EB
?   INITLC  =$45C3      INLIN   =$D52C      INPDATA =$956B      INPDONE =$95CB
    INPFIN  =$95C4      INPTR   =$7F        INPUTFLG=$15        INSDS2  =$F88C
    INSIRQV =$89F3      INSTART =$9529      INTSPTR =$99DA      INTTYP  =$12
    INTTYPSV=$C7        IRQHDLR =$898D      IRQTBLE =$BFB2      IRQV    =$03FE
    ISAUXMEM=$7D48      ISBASRUN=$A65E      ISCNTC  =$D858      ISHOSTOK=$8DD2
    ISLETC  =$E07D      ISMOUSH =$8DDA      ISPFACT =$9980      ITEACH  =$91A3
    ITVADDR =$95F3      IVALARG =$8A50      K6502   =$00        K65816  =$01
    K65C02  =$01      ? KANCACH =$04        KILLEMAL=$88A6      KNEW    =$01
?   KNEW2   =$01        KOPT    =$00        KOPT16  =$00        KOPTLNG32=$01
    KOPTLNG33=$00        KSNCACH =$04        KTINC   =$99F5       KWELMSIZ=$7CBB
    KX3     =$889E      L08     =$82EF      L088    =$82ED      L3      =$855E
    LBS00   =$7D94      LBS03   =$777E      LBS033  =$91BD      LBS04   =$877C
    LBS041  =$882B      LBS05   =$9086      LBS051  =$908A      LBS06   =$8970
?   LBS061  =$8972      LBS10   =$8B5A      LBS49   =$776A      LBS60   =$91CC
    LBS61   =$90B0      LBS62   =$90D9      LBS63   =$90E0      LBS64   =$914B
    LBS641  =$914F      LBS65   =$9114      LBS66   =$9140      LBS67   =$915E
    LBS68   =$9175      LBS69   =$918A      LBS80   =$866B      LBS81   =$8668
    LENGTH  =$2F        LENREC  =$15        LENTHS  =$D1AD      LET2    =$DA63
    LETINF  =$C0        LEVELPAR=$BD        LGSYNERR=$84BC      LINGET  =$DA0C
    LINNUM  =$50        LISTED  =$832F      LLOOP   =$754B    ? LN      =$41E6
    LONGLANG=$26DC   M? LOOP    =$4000      LOWTR   =$9B        LRST100 =$9495
    LST1LIN =$82D9      LSTD?   =$82D7      LTOKEN  =$83D5      MACHINE =$9CED
    MACMAT  =$429A      MAINLIST=$82B0    ? MC      =$41E6      MCAND   =$C0
    MCODE   =$42A2      MEMERR  =$D410      MEMORY  =$9CEF      MESER1  =$00
    MESER2  =$1B        MESER3  =$3D        MESER4  =$62        MESER5  =$87
    MESER6  =$99        MESER7  =$B0        MESER8  =$C3        MESER9  =$EC
    MESSERR =$99F9      MFIN    =$867E      MINSDS2 =$F88C      MIRQST  =$99D2
    MISLETC =$7DC8      MKNARRAY=$7B80      MKNV    =$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=$42C4      MOVE    =$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  =$8C62
MD  MTSB    =$8000      MULTPLSS=$E2AD      MULTPLY1=$E2B6      MVECTOR =$99B6
    MZRTAUX =$41D7      NAMFOUND=$7A27      NAMNTFND=$79F8      NARRAY  =$7A79
    NARRGL91=$7ACA      NCHKCLS =$8646      NCHKCOM =$8649      NCHKOPN =$864C
    NCR     =$831B      NDATAN  =$9460      NDLVCMD =$91FA      NDSVCMD =$91F2
    NEEDDEC =$9CCF      NEG32768=$9B10      NEG65536=$9B0B      NEG8    =$7854
    NEGATE  =$7917      NEGOP   =$EED0      NERRH   =$8DE9    ? NERRHP  =$8DE4
    NEVAL   =$8B04      NEVALC  =$8AFB    ? NEWAYINT=$7788      NEWGARBG=$E484
    NEWSTT  =$D7D2      NEWY    =$47        NEXT1   =$8FBE      NEXTC2  =$8938
    NEXTCTX =$891F      NFAEP   =$7C67      NFRMEVL =$8651      NFRMNUM =$84CE
    NFRMSTK2=$909D      NGARBAG =$7D2F      NGETARPT=$793E      NGETBYT =$8673
    NGTA2   =$8E6E      NILLM   =$8DE7      NKBDINT =$91D9      NLET2   =$7639
    NMAKINT =$7CCE      NMOVINS =$766C      NOPER   =$04      ? NOUVIN  =$81F0
    NPARCHK =$8640      NPTRG   =$8ADE      NPTRGET =$7946      NPTRGET1=$794C
    NPTRGETX=$8843      NPTRGL90=$79B8      NPTRGTX =$7942      NREASON =$7A5B
    NREMN   =$9463      NRET    =$7902      NROUT   =$7783      NSYNCHR =$7DCE
    NSYNCHR2=$7DD0      NUMDIM  =$0F        NUMELS  =$08        NUMELS2 =$10
    NWGVAYF =$8663      NXIN    =$DBDC      NXLST   =$82BC      NZTAB   =$D0E6
    OFFDEF  =$9B6A      OFFIIF  =$9B66      OFFMOU  =$9B67      OFFOFF  =$9B65
    OFFSET  =$C2        OFFSTB  =$9625      OFFSTT  =$9635      OFFTIM  =$9B68
    OFFUSR  =$9B69      OFSTGTO =$9BE5      OKP1GET =$756A      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     =$94EF      PIL     =$94E9
    PIM     =$9574      PIOFFSET=$95FC      POS32768=$9B15      POS65536=$9B1A
```

```
                              ===== Page 122 =====

   PTR2    =$1C        PVERSION=$9CDE      QINT    =$EBF2      R       =$816E
   R0      =$88B8   ?  RAZPF   =$7DDA      RCLEAR  =$7DC2      RCLM    =$05
   RCLMAUX =$7D8A   ?  RCLR    =$03        RD2     =$A47A      RD80STOR=$C018
   RDEF    =$80FE      RDEFSUB =$8169      RDEFUSR =$7FD7      RDIM    =$8422
   RDIMERR =$7B7B      RDKEY   =$FD0C      RDLCBNK2=$C011      RDLCRAM =$C012
   REASON  =$D3E3      RECON   =$8252      RECON1  =$824E   ?  RECON2  =$8256
   REMSTK  =$F8        RESTOR  =$88E3      RESTOR1 =$88C2      RESTOR2 =$88CC
   RESTORC =$8902      RESTORD =$88BC      RESTORF =$8901   ?  RESTORX =$88F2
   RESULT  =$62        RET1    =$76B2      RET3    =$855B      RETOUR  =$8072
   RETOURM =$8D48      RETOURT =$8D4B      RETURN  =$8285      REVECTOR=$9CDF
   RFFVL   =$8531      RFOR    =$8EDF      RGET    =$94A6      RGOSUB  =$92BF
   RGOTO   =$9308      RGPART1 =$930F      RHOM    =$06        RIF     =$92E1
   RIIF    =$84BF      RINI    =$07        RINP    =$94B6      RLET    =$75A1
   RLET1   =$75D1   ?  RMTCTRL =$887B      RNEW    =$7DBC      RNEWINST=$8D5C
   RNEWISUI=$8876      RNEXT   =$8FB8      RNI2    =$8D7D      RON     =$9257
   RONERR  =$8407      ROUT0   =$86CC      ROUT10  =$8A63      ROUT11  =$867F
?  ROUT1X  =$8176      ROUT1Y  =$8172      ROUT4   =$8709      ROUT8   =$920B
   ROUT8C  =$9202      ROUTGEN =$86A4      RPOS    =$04        RREAD   =$02
   RREAD2  =$94E0      RRETURN =$83FD      RRUN    =$7DB3      RSETM   =$00
   RSRVM   =$01        RST100  =$7547      RST101  =$7549      RST102  =$7565
   RST103  =$7551      RSTALTM =$80DA      RSTCURRM=$80CF      RUSR    =$7ECB
   RVRAI   =$8470      RW2     =$8E48      RWAIT   =$8E08      SAVALTM =$80F0
   SAVCURRM=$80E5      SAVER   =$8946      SAVERC  =$897D      SCDCH2  =$7974
   SCNDTIM =$7E86      SCTR    =$9B        SDEF1   =$9768      SDIV    =$78A5
   SDIV8   =$77FC      SENDCHR =$830B      SETINITX=$7DFA      SETITS  =$7710
   SETLTR  =$8916      SETUPB  =$81A0      SETUPD  =$81B7      SETVYA  =$E0DE
   SFE1    =$8ED0      SINITX  =$9974      SIT     =$998C      SKIPC   =$8478
   SLKCACH =$7A2A      SLTR    =$9990      SLTRP1  =$9994   MD?SMOVE   =$8000
   SMUL    =$785F      SMUL8   =$77C3      SNCCH   =$9983      SNERR   =$7B78
   SNGFLT  =$E301      SPROOT  =$95F2      STACK   =$0100   MD?STD     =$8000
   STDLIS  =$8286      STDZP   =$C008      STEP    =$8F3C   MD STID    =$8000
   STP1    =$8F35      STREND  =$6D        STRING1 =$AB        STRLT2  =$E3ED
   STRNG   =$19        STRNG1  =$AC        STRNG2  =$AD        STRPRT  =$DB3D
   STRSPA  =$E3DD      STRTRNG =$8292      STRTXT  =$DE81      SUBERR  =$E196
   SUBFLG  =$14        SUBSERR =$7B75   ?  SUITE   =$4000      SVA     =$99EE
   SVALTNM =$9754      SVAREA  =$973A      SVARS   =$D020      SVCURRM =$9748
   SVMTACTV=$99C2      SVN     =$9984      SVNP1   =$9988   ?  SVNUM   =$9CCD
   SVOFST  =$972C   ?  SVP2    =$95E0      SVPTR   =$95CE      SVWOF   =$99E8
   SWPIO   =$8961   ?  SWREINIT=$8C2E      SYNERR  =$DEC9      TABOFB  =$95E1
   TABOFT  =$95E9   ?  TELMS   =$D093      TEST2E  =$4368      TFUNC   =$8CA1
   TIDMOCL =$9B70      TIINC   =$99F7      TIMEINST=$8BF8      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?  =$8338
   TOKENS  =$9613      TOKEQUAL=$D0        TOKFN   =$C2        TOKFOR  =$81
   TOKFRE  =$D6        TOKGOSUB=$B0        TOKGOTO =$AB        TOKIF   =$AD
   TOKINT  =$D3        TOKMINUS=$C9        TOKMOTIF=$9607      TOKMPFB =$960B
   TOKMPFT =$960F      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 =$8C49      TPT_B   =$99E0      TPT_T   =$99E2
   TRACE   =$D805      TRAITEOK=$9402      TRCFLG  =$F2        TVN1ORA =$9B92
   TVNORA  =$9B8E      TVTVAL  =$9B8A      TXPSV   =$87        TXTPSV  =$F4
   TXTPTR  =$B8        TXTTAB  =$67        TYPLET  =$9B96      TYPMOD  =$C1
   ULERR   =$D97C      USDIV   =$78D0      USDIV8  =$782A      USEOLDAR=$7B12
   USMUL   =$787E      USMUL8  =$77E1      USRMOD  =$00        V3      =$7F8F
   V3B     =$7F89      V3T     =$7F86      VALTYP  =$11        VALTYPSV=$C8
   VARNAM  =$81        VARPNT  =$83        VARPT   =$D1BD      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 =$8335      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=$8189      XFROMMOT=$818C
   XMFIN   =$874F      XMFIN1  =$8779      XMFIN2  =$8776      XSAV    =$B4
   XSUITE  =$8559      XXSAV   =$1B        YICUR   =$99D3      YSAV    =$B5
   ZAUXB   =$D019      ZAUXOFFT=$BFB8      ZAUXRET =$BF3D      ZAUXRT  =$D014
   ZAUXRT0 =$D019      ZAUXRT1 =$D03C      ZAUXRT2 =$D047      ZAUXRT3 =$D000
   ZCOMRT12=$D085      ZEROPRT =$7906      ZGCP2   =$BF95      ZGCPARMS=$BF7C
   ZNG     =$BF9E      ZPRT8   =$7845      ZRTAUX  =$7D20   V  ]DEBUT  =$9774
V  ]ERR    =$9254   V  ]ERR1   =$8B35   V  ]ERRS   =$771E   V  ]FIN    =$9974
V  ]JLOOP  =$8B3B   V  ]LOOP   =$9490   V  ]LOOP1  =$948A   V  ]LOOP2  =$93D3
V  ]RET    =$945F


Symbol table - numerical order:

   K6502   =$00        KOPT    =$00        KOPTLNG33=$00        KOPT16  =$00
   USRMOD  =$00        RSETM   =$00        MESER1  =$00        K65C02  =$01
   K65816  =$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=$26DC   M? LOOP    =$4000   MD MPHX    =$8000
MD MPHY    =$8000   MD MPLX    =$8000   MD MPLY    =$8000   MD MTSB    =$8000
MD GOTO    =$8000   ?  SUITE   =$4000      EK      =$41C6      MZRTAUX =$41D7
?  MC      =$41E6   ?  LN      =$41E6      ADB1    =$41E6      ADT1    =$41F3
   ADB2    =$4200      ADT2    =$420D      BIGRECON=$421A      MACMAT  =$429A
   MCODE   =$42A2      CFM     =$42B0      CFA     =$42B4      DATA1IDX=$42B8
   DATA1VAL=$42BE      MOUSEDET=$42C4      COPYROM =$432C      TEST2E  =$4368
   INITBF  =$43EB      CODE1BF =$4464      CODE2BF =$4520      CODE1LC =$4520
   CODE2LC =$45C3   ?  INITLC  =$45C3      CODE1GC =$45DE      CODE1GCF=$477B
   AROMBA  =$477B      FNDVAR2 =$7524      CGARBAG =$7524      DEBUTGET=$7524
   RST100  =$7547      RST101  =$7549      LLOOP   =$754B      RST103  =$7551
   COMRST  =$7553      COMRSTC =$755B      RST102  =$7565      OKP1GET =$756A
   GNPTRGET=$757B      DEBUTGOT=$757E      RLET    =$75A1      RLET1   =$75D1
?  HNDLEREA=$7621      NLET2   =$7639      HNDLESTR=$7646      NMOVINS =$766C
   HNDLEINT=$7673      HNDLEIY =$769D      RET1    =$76B2      HNDLUIAD=$76B3
   HNDLUIMI=$76BF      HNDLSIAD=$76CB      HNDLSIMI=$76D6      HNDLUIDV=$76E2
   HNDLUIMU=$76E3      HNDLSIDV=$76F5      HNDLSIMU=$76F6      HNDLEIX =$7706
   HNDLEIC =$770A      SETITS  =$7710      HNDLUBAD=$7715      HNDLSBAD=$771C
V  ]ERRS   =$771E      HNDLUBMI=$7722      HNDLSBMI=$7729      HNDLUBMU=$7730
   HNDLSBMU=$7731      HNDLUBDV=$774E      HNDLSBDV=$774F      HNDLEBC =$7765
   LBS49   =$776A      LBS03   =$777E      NROUT   =$7783   ?  NEWAYINT=$7788
   SMUL8   =$77C3      USMUL8  =$77E1      SDIV8   =$77FC      USDIV8  =$782A
   ZPRT8   =$7845      ABSOL8  =$7850      NEG8    =$7854      SMUL    =$785F
   USMUL   =$787E      DVZERROR=$78A2      SDIV    =$78A5      USDIV   =$78D0
   NRET    =$7902      ARET    =$7904      ZEROPRT =$7906      ABSOLUTE=$7913
   NEGATE  =$7917      CONV1628=$7926      NGETARPT=$793E      NPTRGTX =$7942
   NPTRGET =$7946      NPTRGET1=$794C      GTLT    =$7962      BADNAM  =$7971
   SCDCH2  =$7974      EXPLIC? =$797C      NPTRGL90=$79B8      NAMNTFND=$79F8
   NAMFOUND=$7A27      SLKCACH =$7A2A      NREASON =$7A5B      NARRAY  =$7A79
   NARRGL91=$7ACA      GNARRAY =$7B0F      USEOLDAR=$7B12      SUBSERR =$7B75
   SNERR   =$7B78      RDIMERR =$7B7B      MKNARRAY=$7B80      GME     =$7C64
   NFAEP   =$7C67      GSE     =$7C92      FAE2    =$7C95      FAE3    =$7C96
   KWELMSIZ=$7CBB      NMAKINT =$7CCE      CNVT1   =$7CEA      ALKCACH =$7CF2
   ZRTAUX  =$7D20      NGARBAG =$7D2F      GSNERR2 =$7D3F      GIQERR2 =$7D42
?  GTMERR2 =$7D45      ISAUXMEM=$7D48      RCLMAUX =$7D8A      LBS00   =$7D94
   RRUN    =$7DB3      RNEW    =$7DBC      RCLEAR  =$7DC2      MISLETC =$7DC8
   NSYNCHR =$7DCE      NSYNCHR2=$7DD0      GOSYNERR=$7DD7   ?  RAZPF   =$7DDA
   SETINITX=$7DFA      COMPOFST=$7E08   ?  GOSVCUR =$7E23      FRSTIM  =$7E27
   COMX1   =$7E7C      SCNDTIM =$7E86      HNDLEADR=$7EA4      COMLET2 =$7EB7
   RUSR    =$7ECB      V3T     =$7F86      V3B     =$7F89      V3      =$7F8F
   COMMONG =$7FAA      RDEFUSR =$7FD7      RETOUR  =$8072      COMREST =$8096
   COLLECTR=$80A4      RSTCURRM=$80CF      RSTALTM =$80DA      SAVCURRM=$80E5
   SAVALTM =$80F0      RDEF    =$80FE      RDEFSUB =$8169      R       =$816E
   GSNERR3 =$816F      ROUT1Y  =$8172   ?  ROUT1X  =$8176   ?  XFRMMOT1=$8189
   XFROMMOT=$818C      DECTPTR =$8197      SETUPB  =$81A0      SETUPD  =$81B7
   BANCLD  =$81C2   ?  NOUVIN  =$81F0      E06     =$823A      RECON1  =$824E
   RECON   =$8252   ?  RECON2  =$8256      RETURN  =$8285      STDLIS  =$8286
   STRTRNG =$8292      ENDRNG  =$82A8      MAINLIST=$82B0      NXLST   =$82BC
   LSTD?   =$82D7      LST1LIN =$82D9      L088    =$82ED      L08     =$82EF
   SENDCHR =$830B      NCR     =$831B      LISTED  =$832F      VLINPRT =$8335
   TOKEN?  =$8338      COMLISO =$83C5      LTOKEN  =$83D5      RRETURN =$83FD
   RONERR  =$8407      RDIM    =$8422      GOIQ    =$846D      RVRAI   =$8470
   SKIPC   =$8478      LGSYNERR=$84BC      RIIF    =$84BF      NFRMNUM =$84CE
   H16B    =$84D7      FRMELMLP=$84E6      FRMELM  =$84E9      RFFVL   =$8531
   XSUITE  =$8559      RET3    =$855B      L3      =$855E      COMCMPLX=$8623
   CALLFUNC=$8632      NPARCHK =$8640      NCHKCLS =$8646      NCHKCOM =$8649
   NCHKOPN =$864C      NFRMEVL =$8651      HE2E8   =$8659      NWGVAYF =$8663
   LBS81   =$8668      LBS80   =$866B      NGETBYT =$8673      MFIN    =$867E
   ROUT11  =$867F      ROUTGEN =$86A4      ROUT0   =$86CC      GGO2TMER=$8706
   ROUT4   =$8709      XMFIN   =$874F      XMFIN2  =$8776      XMFIN1  =$8779
   LBS04   =$877C      LBS041  =$882B      NPTRGETX=$8843      RNEWISUI=$8876
?  RMTCTRL =$887B      KX3     =$889E      KILLEMAL=$88A6      R0      =$88B8
   RESTORD =$88BC      RESTOR1 =$88C2      RESTOR2 =$88CC      RESTOR  =$88E3
?  RESTORX =$88F2      RESTORF =$8901      RESTORC =$8902      SETLTR  =$8916
   NEXTCTX =$891F      NEXTC2  =$8938      SAVER   =$8946      SWPIO   =$8961
   LBS06   =$8970   ?  LBS061  =$8972      SAVERC  =$897D      IRQHDLR =$898D
   INSIRQV =$89F3      DINSIRQV=$8A1B      CMPCLAMP=$8A37      IVALARG =$8A50
   COMCLAMP=$8A5A      ROUT10  =$8A63      COMCLEAR=$8A8F      FINMOUSE=$8A91
   COMREAD =$8A94      COMPOS  =$8AC2      NPTRG   =$8ADE      NEVALC  =$8AFB
   NEVAL   =$8B04      COMLBS  =$8B0F   V  ]ERR1   =$8B35   V  ]JLOOP  =$8B3B
   LBS10   =$8B5A      COMMON9 =$8BDE      COMMON  =$8BE3      TIMEINST=$8BF8
   COMINT4 =$8C2B   ?  SWREINIT=$8C2E      TOMOUSE =$8C49      MTFUNC  =$8C62
   TFUNC   =$8CA1      COMINT1 =$8CB9      COMINT2 =$8D0B      RETOURM =$8D48
   RETOURT =$8D4B      RNEWINST=$8D5C      RNI2    =$8D7D      ISHOSTOK=$8DD2
   ISMOUSH =$8DDA      HNOK    =$8DE2   ?  NERRHP  =$8DE4      NILLM   =$8DE7
   NERRH   =$8DE9      RWAIT   =$8E08      COMWAIT =$8E1B      RW2     =$8E48
   GN32768 =$8E5A      GP32768 =$8E5F      GN65536 =$8E64      GP65536 =$8E69
   NGTA2   =$8E6E      FEFOR   =$8E88      SFE1    =$8ED0      FESTEP  =$8ED7
   RFOR    =$8EDF      STP1    =$8F35      STEP    =$8F3C      COMFOR  =$8F73
   FENEXT  =$8F8D      RNEXT   =$8FB8      NEXT1   =$8FBE      COMNEXT =$9071
   LBS05   =$9086      LBS051  =$908A      NFRMSTK2=$909D      LBS61   =$90B0
   LBS62   =$90D9      LBS63   =$90E0      COMCOPY =$910A      LBS65   =$9114
   LBS66   =$9140      LBS64   =$914B      LBS641  =$914F      LBS67   =$915E
   LBS68   =$9175      LBS69   =$918A      ITEACH  =$91A3      LBS033  =$91BD
   LBS60   =$91CC      NKBDINT =$91D9      NDSVCMD =$91F2      NDLVCMD =$91FA
   ROUT8C  =$9202      ROUT8   =$920B   V  ]ERR    =$9254      RON     =$9257
   RGOSUB  =$92BF      RIF     =$92E1      RGOTO   =$9308      RGPART1 =$930F
   GOUNDEF =$93B4      DECOMPILE=$93B7   V  ]LOOP2  =$93D3      FINLIGNE=$93F4
   FINDEC  =$93FF      TRAITEOK=$9402      CLENGTH =$9452      COMRG   =$945B
V  ]RET    =$945F      NDATAN  =$9460      NREMN   =$9463   V  ]LOOP1  =$948A
V  ]LOOP   =$9490      LRST100 =$9495      RGET    =$94A6      RINP    =$94B6
   RREAD2  =$94E0      IFZ     =$94E7      PIL     =$94E9      PII     =$94EF
   INSTART =$9529      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      TOKMPFB =$960B
   TOKMPFT =$960F      TOKENS  =$9613      FPROUTSB=$9617      FPROUTST=$961B
   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=$BF7C      ZGCP2   =$BF95      ZNG     =$BF9E
  G83     =$BFA4      G81     =$BFAB      IRQTBLE =$BFB2      ZAUXOFFT=$BFB8
  STDZP   =$C008      ALTZP   =$C009      RDLCBNK2=$C011      RDLCRAM =$C012
  RD80STOR=$C018      XFER    =$C314      ZAUXRT3 =$D000    ? FNDVAR  =$D004
  FNDVARX2=$D00B      ZAUXRT  =$D014      ZAUXB   =$D019      ZAUXRT0 =$D019
  SVARS   =$D020      ARYVAR  =$D039      ZAUXRT1 =$D03C      ZAUXRT2 =$D047
  ZCOMRT12=$D085    ? TELMS   =$D093      AXARTAB =$D099    ? AXARYPNT=$D099
? AXOFFSET=$D09B      GRBPAS  =$D09C    ? ELMSIZ  =$D09D      AXVALUE =$D09E
? AXARYPT2=$D09E      TOKTABL =$D0D0      NZTAB   =$D0E6      DVARS   =$D0F6
  GDVARTS =$D0FF      DVAR    =$D103      DVARTS  =$D18D      BTMEL   =$D19D
  LENTHS  =$D1AD      VARPT   =$D1BD      GTFORPNT=$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  =$F88C      MINSDS2 =$F88C      PCADJ   =$F953      RDKEY   =$FD0C
MOVE    =$FE2C
```