```
* PeerSoft v1.5.5 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
* ToDo: Three new integer subtypes: byte, 24 and
* 32 bits integer now understood (convenience for array
* variables of this integer subtypes).
* Alternate way of array adressing (1dim)
* 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     =     K65C02
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

         XC
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

* 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
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      FAC1 + (Y,A)
FMULT    EQU   $E97F      FAC1 * (Y,A)
FDIV     EQU   $EA66      (Y,A) / 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   $5500

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
5500: A9 D3     SUITE    LDA   #$9CD3     Compute the offset
5502: 38                 SEC              ;Put it in :0+1 (lobyte)
5503: ED 00 9D           SBC   DBUFP       and :1+1 (hibyte)
5506: 8D 47 55           STA   :0+1
5509: A9 9C              LDA   #>$9CD3
550B: ED 01 9D           SBC   DBUFP+1
550E: AA                 TAX
550F: 0D 47 55           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)
5512: F0 01              BEQ   :6
5514: CA                 DEX
5515: 8E 4F 55  :6       STX   :1+1

                * Relocate code (don´t move it yet)
5518: A9 D4              LDA   #AROMBA
551A: A0 59              LDY   #>AROMBA
551C: 85 3A     ]LOOP    STA   PCL
551E: C9 19              CMP   #FCODE-FNDVAR2+AROMBA
5520: 98                 TYA
5521: E9 74              SBC   #>FCODE-FNDVAR2+AROMBA
5523: B0 33              BCS   :4
5525: 84 3B              STY   PCL+1
5527: 20 B1 56           JSR   MINSDS2
552A: A4 2F              LDY   LENGTH
552C: C0 02              CPY   #2         Only relocates 3 bytes instr.
552E: D0 22              BNE   :3
5530: B1 3A              LDA   (PCL),Y
5532: AA                 TAX
5533: 88                 DEY
5534: B1 3A              LDA   (PCL),Y
5536: A8                 TAY
5537: C9 00              CMP   #FIN       Only if adress within range
5539: 8A                 TXA
553A: E9 9C              SBC   #>FIN
553C: B0 14              BCS   :3         Must be < FIN to be relocated
553E: C0 CB              CPY   #FNDVAR2
5540: 8A                 TXA
5541: E9 7B              SBC   #>FNDVAR2
5543: 90 0D              BCC   :3         Must be >= FNDVAR2
5545: 98                 TYA              ;Relocates address
5546: E9 00     :0       SBC   #0
5548: A0 01              LDY   #1
554A: 91 3A              STA   (PCL),Y    Low byte
554C: C8                 INY
554D: 8A                 TXA
554E: E9 00     :1       SBC   #0
5550: 91 3A              STA   (PCL),Y    High byte
5552: 20 53 F9  :3       JSR   PCADJ      Adjust PCL to length byte
5555: 4C 1C 55           JMP   ]LOOP      Loop


```

```
                >81   * Relocate some non trivial references (i.e. instructions
                >82   * with immediate adressing mode).
5558: A2 10     >83   :4        LDX   #ADPFT-ADPFB-1
555A: BD E7 79 >84   ]LOOP     LDA   ADPFB+AROMBA-FNDVAR2,X
555D: 38        >85             SEC
555E: ED 47 55 >86             SBC   :0+1
5561: 9D E7 79 >87             STA   ADPFB+AROMBA-FNDVAR2,X
5564: BD F8 79 >88             LDA   ADPFT+AROMBA-FNDVAR2,X
5567: ED 4F 55 >89             SBC   :1+1
556A: 9D F8 79 >90             STA   ADPFT+AROMBA-FNDVAR2,X
556D: CA        >91             DEX
556E: 10 EA     >92             BPL   ]LOOP
                >93
5570: A2 09     >94             LDX   #ADT1-ADB1-1
5572: A9 00     >95             LDA   #0
5574: 85 3A     >96             STA   PCL
5576: BD D1 56 >97   ]LOOP     LDA   ADT1,X
5579: 85 3B     >98             STA   PCL+1
557B: BC C7 56 >99             LDY   ADB1,X
557E: B1 3A     >100            LDA   (PCL),Y
5580: 38        >101            SEC
5581: ED 47 55 >102            SBC   :0+1
5584: 91 3A     >103            STA   (PCL),Y
5586: BD E5 56 >104            LDA   ADT2,X
5589: 85 3B     >105            STA   PCL+1
558B: BC DB 56 >106            LDY   ADB2,X
558E: B1 3A     >107            LDA   (PCL),Y
5590: ED 4F 55 >108            SBC   :1+1
5593: 91 3A     >109            STA   (PCL),Y
5595: CA        >110            DEX
5596: 10 DE     >111            BPL   ]LOOP
                >112
                >113  * Move the code
5598: A9 CB     >114            LDA   #CGARBAG
559A: A2 7B     >115            LDX   #>CGARBAG
559C: 38        >116            SEC
559D: ED 47 55 >117            SBC   :0+1
55A0: 85 42     >118            STA   A4L
55A2: 8A        >119            TXA
55A3: ED 4F 55 >120            SBC   :1+1
55A6: 85 43     >121            STA   A4L+1
                >122
55A8: A9 D4     >123            LDA   #CGARBAG+AROMBA-FNDVAR2
55AA: A2 59     >124            LDX   #>CGARBAG+AROMBA-FNDVAR2
55AC: 85 3C     >125            STA   A1L
55AE: 86 3D     >126            STX   A1L+1
                >127
55B0: A9 08     >128            LDA   #FIN-1+AROMBA-FNDVAR2
55B2: 85 3E     >128            STA   A2L
55B4: A9 7A     >128            LDA   #>FIN-1+AROMBA-FNDVAR2
55B6: 85 3F     >128            STA   A2L+1
                >129
55B8: A0 00     >130            LDY   #0
55BA: 2C 81 C0 >131            BIT   $C081
55BD: 20 2C FE >132            JSR   MOVE
                >133  * Reconstruct DOS buffers below PeerSoft
55C0: AD 00 9D >134            LDA   DBUFP
```

```
55C3: AE 01 9D >135           LDX   DBUFP+1
55C6: C9 D3    >136           CMP   #$9CD3
55C8: D0 05    >137           BNE   :7
55CA: E0 9C    >138           CPX   #>$9CD3
55CC: D0 01    >139           BNE   :7         One more page if first utility
55CE: CA       >140           DEX              ; to install this way
55CF: 38       >141  :7       SEC
55D0: E9 35    >142           SBC   #LONGLANG
55D2: A8       >143           TAY
55D3: 8A       >144           TXA
55D4: E9 20    >145           SBC   #>LONGLANG
55D6: 8C 00 9D >146           STY   DBUFP      New DOS base buffer address
55D9: 8D 01 9D >147           STA   DBUFP+1
55DC: 20 D4 A7 >148           JSR   $A7D4
               >149
55DF: A9 15    >150           LDA   #VERSION
55E1: 8D DE 9C >151           STA   PVERSION
55E4: A9 80    >152           LDA   #$80
55E6: 8D D3 9C >153           STA   OPTCGOTO
55E9: 9C D2 9C >155           STZ   NEEDDEC
               >160
               >161  * Number of Applesoft instruction runs
               >162  * between two consecutives context switches
55EC: A9 0A    >163           LDA   #10
55EE: 8D DD 9C >164           STA   ICTRACTV
55F1: 9C DC 9C >169           STZ   MTACTV
55F4: A9 4C    >171           LDA   #$4C
55F6: 8D DF 9C >172           STA   REVECTOR
55F9: 38       >173           SEC
55FA: A9 B4    >174           LDA   #ROUTGEN
55FC: ED 47 55 >175           SBC   :0+1
55FF: 8D E0 9C >176           STA   REVECTOR+1
5602: A9 8B    >177           LDA   #>ROUTGEN
5604: ED 4F 55 >178           SBC   :1+1
5607: 8D E1 9C >179           STA   REVECTOR+2
560A: A9 FE    >180           LDA   #NPTRGL90
560C: ED 47 55 >181           SBC   :0+1
560F: 8D D6 9C >182           STA   VNPTRG90
5612: A9 7E    >183           LDA   #>NPTRGL90
5614: ED 4F 55 >184           SBC   :1+1
5617: 8D D7 9C >185           STA   VNPTRG90+1
561A: A9 DE    >186           LDA   #NARRGL91
561C: ED 47 55 >187           SBC   :0+1
561F: 8D D4 9C >188           STA   VNARRG91
5622: A9 7F    >189           LDA   #>NARRGL91
5624: ED 4F 55 >190           SBC   :1+1
5627: 8D D5 9C >191           STA   VNARRG91+1
562A: A9 23    >192           LDA   #TABOFB
562C: ED 47 55 >193           SBC   :0+1
562F: 8D D8 9C >194           STA   ADADR
5632: A9 96    >195           LDA   #>TABOFB
5634: ED 4F 55 >196           SBC   :1+1
5637: 8D D9 9C >197           STA   ADADR+1
563A: A9 67    >198           LDA   #NDSVCMD   New DOS Save for applesoft
563C: ED 47 55 >199           SBC   :0+1
563F: 8D A6 A3 >200           STA   $A3A6
5642: A9 93    >201           LDA   #>NDSVCMD
```

```
5644: ED 4F 55 >202           SBC   :1+1
5647: 8D A7 A3 >203           STA   $A3A7
564A: A9 6F    >204           LDA   #NDLVCMD   Part of routine for loading
564C: ED 47 55 >205           SBC   :0+1
564F: 8D 2E A4 >206           STA   $A42E
5652: A9 93    >207           LDA   #>NDLVCMD
5654: ED 4F 55 >208           SBC   :1+1
5657: 8D 2F A4 >209           STA   $A42F
565A: A9 20    >210           LDA   #$20
565C: 8D 9E 9E >211           STA   $9E9E
565F: A9 4E    >212           LDA   #NKBDINT
5661: ED 47 55 >213           SBC   :0+1
5664: 8D 9F 9E >214           STA   $9E9F
5667: A9 93    >215           LDA   #>NKBDINT
5669: ED 4F 55 >216           SBC   :1+1
566C: 8D A0 9E >217           STA   $9EA0
566F: 20 EF 56 >218           JSR   BIGRECON
5672: 20 99 57 >219           JSR   MOUSEDET
5675: 2C EF 9C >220           BIT   MEMORY
5678: 50 06    >221           BVC   :44
               >222  * Copy $F8-$FF pages within ROM to main and aux
               >223  * memory banks
567A: 20 FD 57 >224           JSR   COPYROM
               >225  * Initialize BF page
567D: 20 BC 58 >226           JSR   INITBF
5680: 2C 80 C0 >227  :44      BIT   $C080
               >228  * If Applesoft is the active language, so
               >229  * install Peersoft CHRGET/CHRGOT patch
5683: AD B6 AA >230  EK       LDA   $AAB6
5686: F0 06    >231           BEQ   :11
5688: 2C 81 C0 >232           BIT   $C081
568B: 20 FA 85 >233           JSR   SETUPB
568E: 4C 11 86 >234  :11      JMP   SETUPD
               >235
               >237  MC       DO    KOPT16
5691: DA FA 04 >241           HEX   DAFA041A3A PHX/PLX/TSB d/INC/DEC
5696: 7C 80 7A >242           HEX   7C807A5A   JMP (abs, X)/BRA d/PLY/PHY
569A: 64 9E    >243           HEX   649E       STZ d/STZ a, X
569C: 0C 9C    >244           HEX   0C9C       TSB a/STZ a
569E: 1C 14    >245           HEX   1C14       TRB a/TRB d
56A0: B2       >246           HEX   B2         LDA (d)
               >253  LN       DO    KOPT16
56A1: 00 00 01 >257           HEX   0000010000 PHX/PLX/TSB d/INC/DEC
56A6: 02 01 00 >258           HEX   02010000   JMP (abs, X)/BRA d/PLY/PHY
56AA: 01 02    >259           HEX   0102       STZ d/STZ a, X
56AC: 02 02    >260           HEX   0202       TSB a/STZ a
56AE: 02 01    >261           HEX   0201       TRB a/TRB d
56B0: 01       >262           HEX   01         LDA (d)
               >269  * Check 65C02/65802 used and new machine codes
56B1: B2 3A    >270  MINSDS2  LDA   (PCL)
56B3: A2 0F    >271           LDX   #LN-MC-1
56B5: DD 91 56 >272  ]LOOP    CMP   MC,X
56B8: F0 07    >273           BEQ   :0
56BA: CA       >274           DEX
56BB: 10 F8    >275           BPL   ]LOOP
56BD: E8       >276           INX              ;X = 0
56BE: 4C 8C F8 >292           JMP   INSDS2
```

```
56C1: BD A1 56 >293  :0       LDA   LN,X
56C4: 85 2F    >294           STA   LENGTH
56C6: 60       >359           RTS
               >364
56C7: 8C       >371  ADB1     DFB   EK+9
56C8: 8F       >372           DFB   EK+12
56C9: 0A       >373           DFB   SETUPB+7+AROMBA-FNDVAR2
56CA: 12       >374           DFB   SETUPB+15+AROMBA-FNDVAR2
56CB: 1B       >375           DFB   SETUPD+1+AROMBA-FNDVAR2
56CC: 08       >376           DFB   STP1+1+AROMBA-FNDVAR2
56CD: 25       >377           DFB   SETLTR+1
56CE: 9D       >382           DFB   LN65536+1+AROMBA-FNDVAR2
56CF: 41       >387           DFB   NAMNTFND+5
56D0: 36       >389           DFB   NEWAYINT+7
56D1: 56       >390  ADT1     DFB   >EK+9
56D2: 56       >391           DFB   >EK+12
56D3: 64       >392           DFB   >SETUPB+7+AROMBA-FNDVAR2
56D4: 64       >393           DFB   >SETUPB+15+AROMBA-FNDVAR2
56D5: 64       >394           DFB   >SETUPD+1+AROMBA-FNDVAR2
56D6: 65       >395           DFB   >STP1+1+AROMBA-FNDVAR2
56D7: 8E       >396           DFB   >SETLTR+1
56D8: 5F       >401           DFB   >LN65536+1+AROMBA-FNDVAR2
56D9: 7F       >406           DFB   >NAMNTFND+5
56DA: 7D       >408           DFB   >NEWAYINT+7
56DB: 8D       >409  ADB2     DFB   EK+10
56DC: 90       >410           DFB   EK+13
56DD: 0E       >411           DFB   SETUPB+11+AROMBA-FNDVAR2
56DE: 16       >412           DFB   SETUPB+19+AROMBA-FNDVAR2
56DF: 20       >413           DFB   SETUPD+6+AROMBA-FNDVAR2
56E0: 0A       >414           DFB   STP1+3+AROMBA-FNDVAR2
56E1: 29       >415           DFB   SETLTR+5
56E2: 9F       >420           DFB   LN65536+3+AROMBA-FNDVAR2
56E3: 48       >425           DFB   NAMNTFND+12
56E4: 38       >427           DFB   NEWAYINT+9
56E5: 56       >428  ADT2     DFB   >EK+10
56E6: 56       >429           DFB   >EK+13
56E7: 64       >430           DFB   >SETUPB+11+AROMBA-FNDVAR2
56E8: 64       >431           DFB   >SETUPB+19+AROMBA-FNDVAR2
56E9: 64       >432           DFB   >SETUPD+6+AROMBA-FNDVAR2
56EA: 65       >433           DFB   >STP1+3+AROMBA-FNDVAR2
56EB: 8E       >434           DFB   >SETLTR+5
56EC: 5F       >439           DFB   >LN65536+3+AROMBA-FNDVAR2
56ED: 7F       >444           DFB   >NAMNTFND+12
56EE: 7D       >446           DFB   >NEWAYINT+9
               >447
56EF: 2C 81 C0 >448  BIGRECON BIT   $C081
56F2: 2C 81 C0 >449           BIT   $C081
               >450  * What is the model/ROM version of the Apple
56F5: A0 07    >451           LDY   #8-1
56F7: AD B3 FB >452           LDA   $FBB3
56FA: 4D C0 FB >453           EOR   $FBC0
56FD: 4D BF FB >454           EOR   $FBBF
5700: D9 6F 57 >455  ]LOOP    CMP   MACMAT,Y
5703: F0 04    >456           BEQ   :1
5705: 88       >457           DEY
5706: 10 F8    >458           BPL   ]LOOP
5708: C8       >459           INY               ;Assuming default 2+
               >460  * Apple //e enhanced ROM and //gs have same signature,
               >461  * so we ll make the difference on $FC5C
               >462  * value ($EB in a //gs ROM)
5709: C0 02    >463  :1        CPY   #2
570B: D0 20    >464            BNE   :2
570D: AD 5C FC >465            LDA   $FC5C
5710: C9 EB    >466            CMP   #$EB
5712: D0 19    >467            BNE   :2
5714: A0 08    >468            LDY   #8         //gs!
5716: 18       >469            CLC
5717: FB       >470            HEX   FB         ;XCE: Enter native mode
5718: 08       >471            PHP              ;Push carry status (old emu bit)
5719: C2 30    >472            HEX   C230       Set 16bits mode
571B: 20 1F FE >473            JSR   $FE1F      Call ID firmware routine
571E: 84 47    >474            STY   NEWY
5720: 28       >475            PLP              ;Restore original emulation bit
5721: FB       >476            HEX   FB         ;XCE: Exit native mode
5722: A0 0C    >477            LDY   #12
5724: A5 48    >478            LDA   NEWY+1
5726: D0 05    >479            BNE   :2
5728: A5 47    >480            LDA   NEWY
572A: 09 08    >481            ORA   #8
572C: A8       >482            TAY
               >483
572D: B9 77 57 >484  :2        LDA   MCODE,Y
5730: 8D ED 9C >485            STA   MACHINE
5733: 98       >486            TYA
5734: AA       >487            TAX
5735: D0 26    >488            BNE   :3         00 if Apple 2+
               >489  * Test for Apple2+, X=0 upon entry
               >490  * Possible language card being there..
5737: 2C 83 C0 >491            BIT   $C083
573A: 2C 83 C0 >492            BIT   $C083
573D: AD 00 D0 >493            LDA   $D000
5740: C8       >494            INY
5741: 8C 00 D0 >495            STY   $D000
5744: CC 00 D0 >496            CPY   $D000      Read after write (1st)
5747: D0 0A    >497            BNE   :5
5749: EE 00 D0 >498            INC   $D000
574C: C8       >499            INY
574D: CC 00 D0 >500            CPY   $D000      Read after increment (2nd)
5750: D0 01    >501            BNE   :5
5752: E8       >502            INX
5753: 8D 00 D0 >503  :5        STA   $D000
5756: BD 89 57 >504            LDA   CFA,X
5759: A2 00    >505            LDX   #0
575B: F0 0B    >506            BEQ   :4
575D: C9 04    >507  :3        CMP   #4         Apple //c or //gs?
575F: A9 C0    >508            LDA   #$C0
5761: A2 80    >509            LDX   #$80
5763: B0 03    >510            BCS   :4         Yes
5765: 20 39 58 >511            JSR   TEST2E
5768: 8D EF 9C >512  :4        STA   MEMORY
576B: 8E F0 9C >513            STX   VID80C
576E: 60       >514            RTS
               >515
576F: EA 2D E6 >516  MACMAT    HEX   EA2DE6E7F9060502
```

```
5777: 00       >517  MCODE    HEX   00         Apple 2+
5778: 40 41 42 >518           HEX   404142     Apple //e
577B: 80 81 82 >519           HEX   80818283   Apple //c
577F: C0 C1 C2 >520           HEX   C0C1C2C3C4C5 Apple //gs
5785: 80 80 C0 >521  CFM      HEX   8080C0C0
5789: 00 80 80 >522  CFA      HEX   008080C0
               >523
578D: 05 07 0B >524  DATA1IDX DFB   5,7,11,12,17,251
5793: 38 18 01 >525  DATA1VAL HEX   3818012000D6
               >526  * Routine to detect a mouse card
5799: A2 C7    >527  MOUSEDET LDX   #$C7
579B: 86 07    >528           STX   AUXPTR+1
579D: 8E D1 9C >529           STX   MOSL       ;b7 of MOSL set to 1
57A0: 64 06    >531           STZ   AUXPTR
57A2: 9C D7 99 >532           STZ   MOCN
57A5: 9C D5 99 >533           STZ   MON0
57A8: A2 05    >540  ]LOOP    LDX   #DATA1VAL-DATA1IDX-1
57AA: BC 8D 57 >541  ]LOOP1   LDY   DATA1IDX,X
57AD: BD 93 57 >542           LDA   DATA1VAL,X
57B0: 51 06    >543           EOR   (AUXPTR),Y
57B2: D0 3D    >544           BNE   :1
57B4: CA       >545           DEX
57B5: 10 F3    >546           BPL   ]LOOP1
57B7: A5 07    >547           LDA   AUXPTR+1
57B9: 8D D7 99 >548           STA   MOCN
57BC: 29 0F    >549           AND   #$F
57BE: 8D D1 9C >550           STA   MOSL
57C1: 0A       >552           ASL
57C2: 0A       >552           ASL
57C3: 0A       >552           ASL
57C4: 0A       >552           ASL
57C5: 8D D5 99 >554           STA   MON0
57C8: E8       >555           INX              ;X = 0
57C9: EC ED 9C >556           CPX   MACHINE    Is host an Apple2 or 2+?
57CC: D0 13    >557           BNE   :2
               >558  * Time to INITMOUSE..
57CE: A0 19    >559           LDY   #$19       Offset to INIT mouse offset
57D0: B1 06    >560           LDA   (AUXPTR),Y
57D2: 85 06    >561           STA   AUXPTR
57D4: A6 07    >562           LDX   AUXPTR+1
57D6: AC D5 99 >563           LDY   MON0
57D9: 20 FA 57 >564           JSR   :0
57DC: 90 03    >565           BCC   :2
57DE: 6E D1 9C >566           ROR   MOSL       Let set b7 of mouse slot
57E1: A2 07    >567  :2       LDX   #OM_INI-OM_DEB
57E3: 64 06    >569           STZ   AUXPTR
57E5: BC CD 99 >574  ]JLOOP   LDY   OM_DEB,X
57E8: B1 06    >575           LDA   (AUXPTR),Y
57EA: 9D CD 99 >576           STA   OM_DEB,X
57ED: CA       >577           DEX
57EE: 10 F5    >578           BPL   ]JLOOP
57F0: 60       >579           RTS
57F1: A6 07    >580  :1       LDX   AUXPTR+1
57F3: E0 C1    >581           CPX   #$C1
57F5: C6 07    >582           DEC   AUXPTR+1
57F7: B0 AF    >583           BCS   ]LOOP
57F9: 60       >584  :FIN     RTS
```

```
57FA: 6C 06 00 >585  :0       JMP   (AUXPTR)
               >586
               >587  * Routine to copy ROM to bank switched RAM
57FD: A0 00    >588  COPYROM  LDY   #0
57FF: A9 F8    >589           LDA   #$F8
5801: 84 3C    >590           STY   A1L
5803: 85 3D    >591           STA   A1L+1
5805: 8D 09 C0 >592           STA   $C009      Write into aux ZP
5808: 84 3C    >593           STY   A1L
580A: 85 3D    >594           STA   A1L+1
580C: 8D 08 C0 >595           STA   $C008      Write back into main ZP
580F: 2C 89 C0 >596           BIT   $C089      Write into LC ram
5812: 2C 89 C0 >597           BIT   $C089
5815: B1 3C    >598  ]LOOP    LDA   (A1L),Y
5817: 91 3C    >599           STA   (A1L),Y    within main memory
5819: 8D 09 C0 >600           STA   $C009      Write into aux memory LC bank
581C: 91 3C    >601           STA   (A1L),Y
581E: 8D 08 C0 >602           STA   $C008      Back to writing to main memory
5821: C8       >603           INY
5822: D0 F1    >604           BNE   ]LOOP
5824: E6 3D    >605           INC   A1L+1
5826: A5 3D    >606           LDA   A1L+1
5828: 8D 09 C0 >607           STA   $C009
582B: 85 3D    >608           STA   A1L+1
582D: 8D 08 C0 >609           STA   $C008
5830: D0 E3    >610           BNE   ]LOOP
5832: 2C 81 C0 >611           BIT   $C081
5835: 2C 81 C0 >612           BIT   $C081
5838: 60       >613           RTS
               >614
               >615  * Routine to test //e configuration: 80 col. card?
               >616  * memory expansion?
5839: 08       >617  TEST2E   PHP
583A: 78       >618           SEI
583B: A2 00    >619           LDX   #0
583D: AD 17 C0 >620           LDA   $C017
5840: 30 6F    >621           BMI   :6
5842: E8       >622           INX
5843: AD 1D C0 >623           LDA   $C01D
5846: 48       >624           PHA
5847: AD 18 C0 >625           LDA   $C018
584A: 48       >626           PHA
584B: AD 1C C0 >627           LDA   $C01C
584E: 48       >628           PHA
584F: AD 19 C0 >629  ]LOOP    LDA   $C019
5852: 30 FB    >630           BMI   ]LOOP
5854: 8D 57 C0 >631           STA   $C057
5857: 8D 01 C0 >632           STA   $C001
585A: 8D 55 C0 >633           STA   $C055
585D: AD 00 04 >634           LDA   $400
5860: 48       >635           PHA
5861: AD 00 24 >636           LDA   $2400
5864: 48       >637           PHA
5865: A9 EE    >638           LDA   #$EE
5867: 8D 00 04 >639           STA   $0400
586A: AD 00 24 >640           LDA   $2400
586D: C9 EE    >641           CMP   #$EE
```

```
586F: D0 0B    >642             BNE   :2
5871: 0E 00 24 >643             ASL   $2400
5874: AD 00 04 >644             LDA   $0400
5877: CD 00 24 >645             CMP   $2400
587A: F0 1B    >646             BEQ   :3
587C: E8       >647  :2         INX
587D: A9 0F    >648             LDA   #$0F
587F: 8D B9 C0 >649             STA   $C0B9
5882: 8D 54 C0 >650             STA   $C054
5885: AD 00 04 >651             LDA   $0400
5888: 8D 00 04 >652             STA   $0400
588B: 8D B8 C0 >653             STA   $C0B8
588E: 8D 55 C0 >654             STA   $C055
5891: AD 00 04 >655             LDA   $0400
5894: 30 01    >656             BMI   :3
5896: E8       >657             INX
5897: 68       >658  :3         PLA
5898: 8D 00 24 >659             STA   $2400
589B: 68       >660             PLA
589C: 8D 00 04 >661             STA   $0400
589F: 68       >662             PLA
58A0: 30 03    >663             BMI   :4
58A2: 8D 54 C0 >664             STA   $C054
58A5: 68       >665  :4         PLA
58A6: 30 03    >666             BMI   :5
58A8: 8D 00 C0 >667             STA   $C000
58AB: 68       >668  :5         PLA
58AC: 30 03    >669             BMI   :6
58AE: 8D 56 C0 >670             STA   $C056
               >671  * X=0: No 80 col. card in aux. slot
               >672  * X=1: 80 col. card w/o memory expansion
               >673  * X=2: 80 col. card with at least 64K mem. expansion
               >674  * X=3: Same as above + special video modes (Eve le chat mau
ve)
58B1: BD 85 57 >675  :6         LDA   CFM,X
58B4: 48       >676             PHA
58B5: BD 89 57 >677             LDA   CFA,X
58B8: AA       >678             TAX
58B9: 68       >679             PLA
58BA: 28       >680             PLP
58BB: 60       >681             RTS
                294             PUT   PEERAUXINSTALL
               >1    INITBF     STID  CODE1;A1L
58BC: A9 ED    >1               LDA   #CODE1
58BE: 85 3C    >1               STA   A1L
58C0: A9 58    >1               LDA   #>CODE1
58C2: 85 3D    >1               STA   A1L+1
58C4: A0 00    >2               LDY   #0
58C6: A9 00    >3               LDA   #ZAUXRT
58C8: 85 3E    >3               STA   A2L
58CA: A9 BF    >3               LDA   #>ZAUXRT
58CC: 85 3F    >3               STA   A2L+1
58CE: 8D 05 C0 >4               STA   $C005
58D1: B1 3C    >5    ]LOOP      LDA   (A1L),Y
58D3: 91 3E    >6               STA   (A2L),Y
58D5: C8       >7               INY
58D6: C0 E7    >8               CPY   #CODE2-CODE1
```

```
58D8: D0 F7    >9             BNE   ]LOOP
58DA: 8D 04 C0 >10            STA   $C004
58DD: BA       >11            TSX
58DE: 8D 09 C0 >12            STA   $C009
58E1: 8E 00 01 >13            STX   $0100
58E4: A2 FF    >14            LDX   #$FF
58E6: 8E 01 01 >15            STX   $0101
58E9: 8D 08 C0 >16            STA   $C008
58EC: 60       >17   ]RET     RTS
               >18
               >19   CODE1    ORG   $BF00
               >20   AXHIMEM  EQU   *
BF00: AA       >21   ZAUXRT   TAX
BF01: BD D6 BF >22            LDA   ZAUXOFFT,X
BF04: BA       >23            TSX              ;Main stack pointer
BF05: 8D 09 C0 >24            STA   $C009
BF08: 8E 00 01 >25            STX   $0100        into $0100 aux stack
BF0B: A2 FF    >26            LDX   #$FF       Aux stack pointer
BF0D: 8E 01 01 >27            STX   $0101        into $0101 aux stack
BF10: 9A       >28            TXS
BF11: 8D 1B BF >29            STA   :0+1
BF14: A9 BF    >37            LDA   #>ZAUXRET-1
BF16: 48       >38            PHA
BF17: A9 B8    >39            LDA   #ZAUXRET-1
BF19: 48       >40            PHA
BF1A: D0 00    >42   :0       BNE   ZAUXRT0
               >43   ZAUXB    EQU   *
               >44
               >45   * Do the init
BF1C: AD DB BF >57   ZAUXRT0  LDA   AXARTAB
BF1F: 85 6B    >58            STA   ARYTAB
BF21: C9 00    >59            CMP   #AXHIMEM
BF23: AD DC BF >60            LDA   AXARTAB+1
BF26: 85 6C    >61            STA   ARYTAB+1
BF28: E9 BF    >62            SBC   #>AXHIMEM
BF2A: B0 0E    >63            BCS   :0
BF2C: AD DD BF >64            LDA   AXSTREND
BF2F: 85 6D    >65            STA   STREND
BF31: C9 00    >66            CMP   #AXHIMEM
BF33: AD DE BF >67            LDA   AXSTREND+1
BF36: 85 6E    >68            STA   STREND+1
BF38: E9 BF    >69            SBC   #>AXHIMEM
               >71   :0
BF3A: 60       >72   ]RET     RTS
               >73
               >74   * Ensure enough room within array segment
BF3B: AD DF BF >85   ZAUXRT1  LDA   AXSZ
BF3E: AE E0 BF >86            LDX   AXSZ+1
BF41: A0 01    >88            LDY   #1
BF43: 92 6D    >89            STA   (STREND)
BF45: 8A       >90            TXA
BF46: 91 6D    >91            STA   (STREND),Y
BF48: 18       >99            CLC
BF49: AD DF BF >100           LDA   AXSZ
BF4C: 69 02    >101           ADC   #2
BF4E: 90 02    >102           BCC   :1
BF50: E8       >103           INX
```

```
BF51: 18       >104          CLC
BF52: 65 6D    >105  :1      ADC   STREND
BF54: A8       >106          TAY
BF55: 8A       >107          TXA
BF56: 65 6E    >108          ADC   STREND+1
BF58: AA       >109          TAX
BF59: C0 00    >110          CPY   #AXHIMEM
BF5B: 8A       >111          TXA
BF5C: E9 BF    >112          SBC   #>AXHIMEM
BF5E: B0 DA    >113          BCS   ]RET
BF60: 84 6D    >114          STY   STREND
BF62: 86 6E    >115          STX   STREND+1
BF64: 60       >117  ]RET    RTS
               >118
               >119  * Retrieve an element value and store it in main memory
               >120  * AXARYPNT: base adress of memory segment dedicated to
               >121  * this array in aux memory.
               >122  * AXOFFSET: offset from adress to 1st elm to address of
               >123  * element which value is to collect
               >124  * ELMSIZE: size of element value
               >125  * AXARYPT2: address in main memory where to store value
BF65: 20 92 BF >126  ZAUXRT2  JSR   ZAUXRT23
BF68: B0 FA    >127          BCS   ]RET
BF6A: AD E2 BF >128          LDA   AXARYPT2
BF6D: 85 3C    >129          STA   A1L
BF6F: AD E3 BF >134          LDA   AXARYPT2+1
BF72: 85 3D    >135          STA   A1L+1
BF74: 8D 04 C0 >137          STA   $C004
BF77: B1 94    >138  ]LOOP   LDA   (ARYPNT),Y
BF79: 91 3C    >139          STA   (A1L),Y
BF7B: 88       >140          DEY
BF7C: 10 F9    >141          BPL   ]LOOP
BF7E: 8D 05 C0 >142          STA   $C005
BF81: 18       >143          CLC
BF82: 60       >144  ]RET    RTS
               >145
               >146  * Store an element value into aux memory
               >147  * AXARYPNT: base adress of memory segment dedicated to
               >148  * this array in aux memory.
               >149  * AXOFFSET: offset from adress to 1st elm to address of
               >150  * element where to store
               >151  * ELMSIZE: element size in # of bytes
               >152  * AXVALUE: value to store (5 bytes reserved)
BF83: 20 92 BF >153  ZAUXRT3  JSR   ZAUXRT23
BF86: B0 FA    >154          BCS   ]RET
BF88: B9 E2 BF >159  ]LOOP   LDA   AXVALUE,Y
BF8B: 91 3C    >160          STA   (A1L),Y
BF8D: 88       >161          DEY
BF8E: 10 F8    >162          BPL   ]LOOP
BF90: 18       >163          CLC
BF91: 60       >164  ]RET    RTS
               >165
BF92: AD DB BF >176  ZAUXRT23 LDA   AXARYPNT
BF95: AE DC BF >177          LDX   AXARYPNT+1
BF98: 18       >178          CLC
BF99: 69 02    >179          ADC   #2
BF9B: 90 02    >180          BCC   :1
```

```
BF9D: E8       >181           INX
BF9E: 18       >182           CLC
BF9F: 6D DF BF >183  :1       ADC   AXOFFSET
BFA2: A8       >184           TAY
BFA3: 8A       >185           TXA
BFA4: 6D E0 BF >186           ADC   AXOFFSET+1
BFA7: C4 6D    >187           CPY   STREND
BFA9: B0 E6    >188           BCS   ]RET
BFAB: AA       >189           TAX
BFAC: E5 6E    >190           SBC   STREND+1
BFAE: B0 E1    >191           BCS   ]RET
BFB0: 86 95    >192           STX   ARYPNT+1
BFB2: 84 94    >193           STY   ARYPNT
BFB4: AC E1 BF >195           LDY   ELMSIZ
BFB7: 88       >196           DEY
BFB8: 60       >197  ]RET     RTS
               >198
BFB9: AE 00 01 >199  ZAUXRET  LDX   $0100      Get back main stack pointer
BFBC: 9A       >200           TXS              ; from $0100 aux stack byte
BFBD: 8E 08 C0 >201           STX   $C008
BFC0: A2 00    >202           LDX   #0
BFC2: 90 01    >203           BCC   *+3
BFC4: E8       >204           INX
BFC5: AD D9 BF >209           LDA   AXRTMAIN
BFC8: 8D ED 03 >210           STA   $03ED
BFCB: AD DA BF >211           LDA   AXRTMAIN+1
BFCE: 8D EE 03 >212           STA   $03EE
BFD1: 18       >213           CLC
BFD2: B8       >214           CLV
BFD3: 4C 14 C3 >215           JMP   XFER
               >216
BFD6: 00 1F    >217  ZAUXOFFT DFB   ZAUXRT0-ZAUXB,ZAUXRT1-ZAUXB
BFD8: 49       >218           DFB   ZAUXRT2-ZAUXB
BFD9: 00 00    >219  AXRTMAIN DS    2
BFDB: 00 08    >220  AXARTAB  DA    $0800      0
               >221  AXARYPNT EQU   AXARTAB    2
BFDD: 00 08    >222  AXSTREND DA    $0800      0
BFDF: 00 00    >223  AXSZ     DS    2          1
               >224  AXOFFSET EQU   AXSZ       2
BFE1: 00       >225  ELMSIZ   DS    1          2
BFE2: 00 00 00 >226  AXVALUE  DS    5
               >227  AXARYPT2 EQU   AXVALUE
               >228  *ZAUXRTF EQU *
               >229           ERR   */$C000
               >230           ORG
               >231  CODE2    EQU   *
                295  * Here is the Peersoft real origine
                296  AROMBA   DO    KOPT-K65C02
                302           ORG   $8CFC+$C00-$96-80-$56-$37-$4C-$B7-$54A-$1571

                305  FNDVAR2
                306  CGARBAG
                307
                308  * All calls to CHRGET fall into this routine
7BCB: 86 B4     309  DEBUTGET STX   XSAV
7BCD: 84 B5     310           STY   YSAV
                311  * Check return address
```

```
7BCF: BA                 TSX
7BD0: BD 02 01           LDA   $0102,X    hi byte
7BD3: 85 C2              STA   OFFSET
7BD5: BD 01 01           LDA   $0101,X    lo byte
7BD8: A2 11              LDX   #ADAPFTET-ADAPFBET
7BDA: DD BB 9B  ]LOOP    CMP   ADAPFBET-1,X
7BDD: D0 07              BNE   :0
7BDF: BC CC 9B           LDY   ADAPFTET-1,X
7BE2: C4 C2              CPY   OFFSET     Test for a match upon
7BE4: F0 20              BEQ   OKP1GET    return address: proceed
7BE6: CA        :0       DEX              ;No match: loop till
7BE7: D0 F1              BNE   ]LOOP       all values exhaustion
7BE9: A4 B5              LDY   YSAV
7BEB: 2C                 HEX   2C         Skip next two bytes
                * No address match: exit with a simulation of CHRGET
7BEC: 86 B4     RST100   STX   XSAV
                RST101
7BEE: E6 B8     LLOOP    INC   TXTPTR
7BF0: D0 02              BNE   COMRST
7BF2: E6 B9              INC   TXTPTR+1
                RST102
                RST103
7BF4: B2 B8     COMRST   LDA   (TXTPTR)
7BF6: C9 20              CMP   #$20
7BF8: F0 F4              BEQ   LLOOP
7BFA: A6 B4              LDX   XSAV
7BFC: C9 3A     COMRSTC  CMP   #´:´
7BFE: B0 05              BCS   :0
7C00: E9 2F              SBC   #$30-1     Because of carry clear
7C02: 38                 SEC
7C03: E9 D0              SBC   #$D0
7C05: 60        :0       RTS

                OKP1GET
                * Tricky way to replace the two bytes at the top of stack
                * Instead of doing PLA PLA followed by PHA PHA...
7C06: 8A                 TXA              ;X into Y
7C07: A8                 TAY
7C08: BA                 TSX
7C09: B9 DD 9B           LDA   ADPFB-1,Y
7C0C: 9D 01 01           STA   $0101,X
7C0F: B9 EE 9B           LDA   ADPFT-1,Y
7C12: 9D 02 01           STA   $0102,X
7C15: D0 D7              BNE   RST101     Always
7C17: 4C 96 7E  GNPTRGET JMP   NPTRGET
7C1A: 86 B4     DEBUTGOT STX   XSAV
7C1C: BA                 TSX
7C1D: BD 01 01           LDA   $0101,X
7C20: C9 EE              CMP   #VPTRGET-1
7C22: D0 D0              BNE   RST103
7C24: BD 02 01           LDA   $0102,X
7C27: 49 DF              EOR   #>VPTRGET-1 A=0 upon matching address
7C29: D0 C9              BNE   RST103
7C2B: E8                 INX              ;Quick way to pull two bytes
7C2C: E8                 INX              ; from stack
7C2D: BD 02 01           LDA   $0102,X
7C30: C9 DA              CMP   #>VLET+2
```

```
7C32: D0 03             BNE   :44
7C34: E8                INX
7C35: E8                INX               ;Carry set at this time
7C36: 24                HEX   24          Skip next byte
7C37: 18        :44     CLC
7C38: 9A                TXS
7C39: A2 00             LDX   #0
7C3B: 90 DA             BCC   GNPTRGET
                * The following routine handles the Applesoft
                * variable setting
                * (LET is the optional keyword)
7C3D: 20 96 7E  RLET    JSR   NPTRGET
7C40: 85 85             STA   FORPNT
7C42: 84 86             STY   FORPNT+1
                RLET1   DO    KOPT-K65C02
7C44: B2 B8             LDA   (TXTPTR)
7C46: A2 03             LDX   #3          New syntax scheme?
7C48: DD 55 96  ]LOOP   CMP   TOKENS,X
7C4B: F0 29             BEQ   :0          yes so handle it
7C4D: CA                DEX
7C4E: 10 F8             BPL   ]LOOP
7C50: A6 12             LDX   INTTYP
7C52: E0 81             CPX   #$81        Byte integer subtype?
7C54: F0 06             BEQ   :10
7C56: 4C 4D DA          JMP   VLET+7      No: delegate to ROM routine
7C59: 4C 99 E1  :11     JMP   GOIQERR
7C5C: 20 4D DA  :10     JSR   VLET+7      Yes: call ROM routine
                * Convert from 16b to 8b
7C5F: A4 A0             LDY   FAC+3
7C61: 98                TYA
7C62: C8                INY
7C63: C0 02             CPY   #2
7C65: B0 F2             BCS   :11
7C67: 45 A1             EOR   FAC+4
7C69: 30 EE             BMI   :11
7C6B: A5 A1             LDA   FAC+4
7C6D: 92 85             STA   (FORPNT)
7C6F: A0 01             LDY   #1
7C71: A9 00             LDA   #0
7C73: 91 85             STA   (FORPNT),Y
7C75: 60                RTS
                * Save selected operation on stack (+,-,*,/)
                :0      MPHX
7C76: DA                PHX
7C77: 20 EE 7B          JSR   RST101      Bump next character
                * Ensure that next char is ´=´ symbol token
7C7A: A9 D0             LDA   #TOKEQUAL
7C7C: 20 62 82          JSR   NSYNCHR2    no need to reset Y to 0
                * Save variable type on stack
7C7F: A5 12             LDA   INTTYP      $80 iif integer variable
7C81: 48                PHA
7C82: A5 11             LDA   VALTYP      $FF iif string
7C84: 48                PHA
7C85: 20 7B DD          JSR   FRMEVL
7C88: 68                PLA
7C89: 2A                ROL               ;Carry set iif var. type string
7C8A: 20 6D DD          JSR   $DD6D       Check FRMEVL result type accordin
```

```
7C8D: 68         486           PLA               ;Get INTTYP off stack
7C8E: B0 68      487           BCS   HNDLESTR   String variable and expression
                 488  * From then on: we´ll handle numeric var. and expr.
7C90: 30 10      489           BMI   HNDLEINT
7C92: A4 86      490  HNDLEREA LDY   FORPNT+1
7C94: 68         491           PLA
7C95: 0A         493           ASL
7C96: AA         495           TAX
7C97: A9 EB      499           LDA   #>$EB27-1
7C99: 48         500           PHA
7C9A: A9 26      501           LDA   #$EB27-1
7C9C: 48         502           PHA
7C9D: A5 85      505           LDA   FORPNT
7C9F: 7C 59 96  506           JMP   (FPROUTS,X)
                 515
                 516  * Includes module for handling integ. arithmetic
                 517  * and <op>= instructions
                 518           PUT   PEERINTEGARITH
                >1    * Module handling all integer arithmetic
                >2    * within Peersoft and all op= instructions
7CA2: 20 2C 7D >3    HNDLEINT JSR   NROUT
                >4    * Get operation off stack into X reg.
7CA5: FA        >5             PLX
7CA6: BD 61 96 >6             LDA   OFFST,X
7CA9: 8D B1 7C >7    HNDLEIY  STA   HNDLEIB-1
7CAC: A0 01     >8             LDY   #1
7CAE: B1 85     >9             LDA   (FORPNT),Y
7CB0: 80 0C     >14            BRA   HNDLEIMI
                >16   HNDLEIB  EQU   *
                >17   HNDLEIAD DO    KOPT-K65C02
7CB2: 18        >19            CLC
7CB3: 65 A1     >21            ADC   $A1         ADD operation
7CB5: AA        >22            TAX
7CB6: B2 85     >27            LDA   (FORPNT)
7CB8: 65 A0     >29            ADC   $A0
7CBA: 70 67     >30            BVS   GOVERROR
7CBC: 50 30     >31            BVC   HNDLEIC
7CBE: 38        >32   HNDLEIMI SEC
7CBF: E5 A1     >33            SBC   $A1
7CC1: AA        >34            TAX
7CC2: B2 85     >39            LDA   (FORPNT)
7CC4: E5 A0     >41            SBC   $A0
7CC6: 70 5B     >42            BVS   GOVERROR
7CC8: 50 24     >43            BVC   HNDLEIC
7CCA: 38        >44   HNDLEIDV SEC
7CCB: 24        >45            HEX   24
7CCC: 18        >46   HNDLEIMU CLC
7CCD: 08        >47            PHP
7CCE: 85 C2     >48            STA   MPLIER
7CD0: B2 85     >53            LDA   (FORPNT)
7CD2: 85 C3     >55            STA   MPLIER+1
7CD4: A5 A0     >56            LDA   $A0
7CD6: 85 C1     >57            STA   MCAND+1
7CD8: A5 A1     >58            LDA   $A1
7CDA: 85 C0     >59            STA   MCAND
7CDC: 28        >60            PLP
```

```
7CDD: B0 05    >61              BCS   HNDLEDV
7CDF: 20 D1 7D >62              JSR   SMUL
7CE2: 80 03    >67              BRA   *+5
7CE4: 20 13 7E >69    HNDLEDV  JSR   SDIV
7CE7: 70 3A    >70              BVS   GOVERROR
               >71    HNDLEIX  DO    KOPT-K65C02
7CE9: C8       >73              INY
7CEA: A6 C2    >75              LDX   MPLIER
7CEC: A5 C3    >76              LDA   MPLIER+1
               >77    HNDLEIC  DO    KOPT-K65C02
7CEE: 92 85    >80              STA   (FORPNT)
7CF0: 8A       >82              TXA
7CF1: 91 85    >86              STA   (FORPNT),Y
7CF3: A9 80    >87    SETITS   LDA   #$80
7CF5: 85 C7    >88              STA   INTTYPSV
7CF7: 60       >89    RET1     RTS
               >90
               >91    * Handle += instruction for string variables
7CF8: 68       >92    HNDLESTR PLA               ;Get OP kind off stack
7CF9: D0 2B    >93              BNE   GTMERROR   ;Only ADD operation allowed
7CFB: B2 A0    >101             LDA   ($A0)
7CFD: F0 F8    >102             BEQ   RET1
7CFF: 18       >103             CLC
7D00: 72 85    >104             ADC   (FORPNT)
7D02: B0 25    >106             BCS   GSTERROR
7D04: 20 DD E3 >107             JSR   STRSPA
7D07: A5 85    >108             LDA   FORPNT
7D09: A4 86    >109             LDY   FORPNT+1
7D0B: 20 1C 7D >110             JSR   NMOVINS
7D0E: A0 02    >111             LDY   #2
7D10: B9 9D 00 >112   ]LOOP    LDA   DSCTMP,Y
7D13: 91 85    >113             STA   (FORPNT),Y
7D15: 88       >114             DEY
7D16: 10 F8    >115             BPL   ]LOOP
7D18: A5 A0    >116             LDA   $A0
7D1A: A4 A1    >117             LDY   $A1
7D1C: 85 AB    >118   NMOVINS  STA   STRING1
7D1E: 84 AC    >119             STY   STRING1+1
7D20: 4C D4 E5 >120             JMP   MOVINS
7D23: 4C D5 E8 >121   GOVERROR JMP   GOOVFERR
7D26: 4C 76 DD >122   GTMERROR JMP   GOTMIERR
7D29: 4C B2 E5 >123   GSTERROR JMP   GOSTLERR
               >124
7D2C: 20 72 EB >125   NROUT    JSR   $EB72      Arrondit FAC1
7D2F: A5 9D    >126   NEWAYINT LDA   FAC
7D31: C9 90    >127             CMP   #$90
7D33: 90 07    >128             BCC   :0
7D35: A9 9D    >129             LDA   #NEG32768
7D37: A0 9B    >130             LDY   #>NEG32768
7D39: 4C 16 E1 >131             JMP   $E116
7D3C: 4C F2 EB >132   :0       JMP   QINT
               >133
               >135   * Signed 8bits multiplication: result in 8bits
               >136   * with possible overflow exception
               >137   * MCAND and MPLIER set upon entry
               >138   * Result in MPLIER
               >139   * Credits: Randy Hyde
```

```
7D3F: A5 C0    >140  SMUL8    LDA   MCAND
7D41: 45 C2    >141           EOR   MPLIER
7D43: 48       >142           PHA              ;Bit N set if signs differ
7D44: 20 BB 7D >143           JSR   ZPRT8
7D47: A0 08    >144  USMUL8   LDY   #8
7D49: A5 C2    >145  ]LOOP    LDA   MPLIER     Get lsb of MPLIER
7D4B: 4A       >146           LSR              ; into C
7D4C: 90 07    >147           BCC   :4
7D4E: 18       >148           CLC
7D4F: A5 BE    >149           LDA   PARTIAL
7D51: 65 C0    >150           ADC   MCAND
7D53: 85 BE    >151           STA   PARTIAL
               >152  * Shift result into MPLIER
7D55: 66 BE    >153  :4       ROR   PARTIAL
7D57: 66 C2    >154           ROR   MPLIER
7D59: 88       >155           DEY              ;All MPLIER 8 bits
7D5A: D0 ED    >156           BNE   ]LOOP       have been processed?
7D5C: FA       >157           PLX
7D5D: 2C 71 7D >158           BIT   :7         Bit V set..
7D60: A5 BE    >159           LDA   PARTIAL
7D62: D0 0D    >160           BNE   :7
7D64: A5 C2    >161           LDA   MPLIER
7D66: 30 09    >162           BMI   :7
7D68: 8A       >163           TXA
7D69: 10 05    >164           BPL   :8
7D6B: A2 C2    >165           LDX   #MPLIER
7D6D: 20 CA 7D >166           JSR   NEG8
7D70: B8       >167  :8       CLV
7D71: 60       >168  :7       RTS
               >169
7D72: 4C E1 EA >170  DVZERR8  JMP   GODVZERR
               >171  * Signed 8bits integer divide routine
               >172  * with possible overflow and divide by zero exceptions
               >173  * DIVEND and DIVSOR set upon entry
               >174  * Result in DIVEND
               >175  * Credits: Randy Hyde
7D75: A5 C0    >176  SDIV8    LDA   DIVSOR
7D77: F0 F9    >177           BEQ   DVZERR8
7D79: 49 80    >178           EOR   #$80
7D7B: D0 0D    >179           BNE   :1
               >180  * On traite le cas ou le diviseur est -128
               >181  * Dans ce cas la si DIVEND vaut aussi -128, alors
               >182  * retourne 1 sinon 0
7D7D: A8       >183           TAY
7D7E: AA       >184           TAX              ;X forced to zero
7D7F: A5 C2    >185           LDA   DIVEND
7D81: C9 80    >186           CMP   #$80
7D83: D0 01    >187           BNE   :0
7D85: E8       >188           INX
7D86: 86 C2    >189  :0       STX   DIVEND
7D88: D0 30    >190           BNE   RETA8      Always
7D8A: A5 C0    >191  :1       LDA   DIVSOR
7D8C: 45 C2    >192  :2       EOR   DIVEND
7D8E: 48       >193           PHA              ;Sign bit on stack
7D8F: 20 BB 7D >194           JSR   ZPRT8      ;Absolute value for operands
7D92: A0 08    >195  USDIV8   LDY   #8
7D94: 06 C2    >196  ]LOOP    ASL   DIVEND
```

```
7D96: 26 BE    >197           ROL   PARTIAL
7D98: 38       >198           SEC
7D99: A5 BE    >199           LDA   PARTIAL
7D9B: E5 C0    >200           SBC   DIVSOR
7D9D: AA       >201           TAX
7D9E: 90 04    >202           BCC   :3
7DA0: 86 BE    >203           STX   PARTIAL
7DA2: E6 C2    >204           INC   DIVEND
7DA4: 88       >205  :3       DEY
7DA5: D0 ED    >206           BNE   ]LOOP
7DA7: 2C BA 7D >207           BIT   ARET8+1    V set by default
7DAA: A5 C2    >208           LDA   DIVEND
7DAC: 1A       >212           INC
7DAD: F0 0A    >214           BEQ   ARET8      Keep V set and exit
7DAF: 68       >215           PLA              ;Get back sign
7DB0: 10 05    >216           BPL   NRET8      No need to get result opposite
7DB2: A2 C2    >217           LDX   #DIVEND
7DB4: 20 CA 7D >218           JSR   NEG8
               >219  * Exit with V clear
7DB7: B8       >220  NRET8    CLV
7DB8: 70       >221           HEX   70         Skip next byte
7DB9: 68       >222  ARET8    PLA
7DBA: 60       >223  RETA8    RTS
               >224
7DBB: A0 00    >225  ZPRT8    LDY   #0
7DBD: 84 BE    >226           STY   PARTIAL
7DBF: A2 C0    >227           LDX   #MCAND
7DC1: 20 C6 7D >228           JSR   ABSOL8
7DC4: A2 C2    >229           LDX   #MPLIER
7DC6: B5 00    >230  ABSOL8   LDA   0,X
7DC8: 10 06    >231           BPL   :0
7DCA: 98       >232           TYA
7DCB: 38       >233           SEC
7DCC: F5 00    >234           SBC   0,X
7DCE: 95 00    >235           STA   0,X
7DD0: 60       >236  :0       RTS
               >237  NEG8     EQU   ABSOL8+4
               >239
               >240  * Signed 16bits multiplication: result in 16bits
               >241  * with possible overflow exception
               >242  * MCAND and MPLIER set upon entry
               >243  * Result in MPLIER
               >244  * Credits: Randy Hyde
7DD1: A5 C1    >245  SMUL     LDA   MCAND+1
7DD3: 45 C3    >246           EOR   MPLIER+1
7DD5: 48       >247           PHA              ;BitN set if signs differ
7DD6: 20 73 7E >248           JSR   ZEROPRT    Get absolute values of operands
7DD9: A0 10    >249  USMUL    LDY   #16
7DDB: A5 C2    >250  ]LOOP    LDA   MPLIER     Get lsb of MPLIER
7DDD: 4A       >251           LSR              ; into C
7DDE: 90 0D    >252           BCC   :4
7DE0: 18       >253           CLC
7DE1: A5 BE    >254           LDA   PARTIAL
7DE3: 65 C0    >255           ADC   MCAND
7DE5: 85 BE    >256           STA   PARTIAL
7DE7: A5 BF    >257           LDA   PARTIAL+1
7DE9: 65 C1    >258           ADC   MCAND+1
```

```
7DEB: 85 BF    >259           STA   PARTIAL+1
               >260  * Shift result into MPLIER
7DED: 66 BF    >261  :4       ROR   PARTIAL+1
7DEF: 66 BE    >262           ROR   PARTIAL
7DF1: 66 C3    >263           ROR   MPLIER+1
7DF3: 66 C2    >264           ROR   MPLIER
7DF5: 88       >265           DEY              ;All MPLIER 16 bits
7DF6: D0 E3    >266           BNE   ]LOOP       have been processed?
7DF8: FA       >267           PLX
7DF9: 2C 0F 7E >268           BIT   :7         bit V set
7DFC: A5 BE    >269           LDA   PARTIAL
7DFE: 05 BF    >270           ORA   PARTIAL+1
7E00: D0 0D    >271           BNE   :7
7E02: A5 C3    >272           LDA   MPLIER+1
7E04: 30 09    >273           BMI   :7
7E06: 8A       >274           TXA
7E07: 10 05    >275           BPL   :8
7E09: A2 C2    >276           LDX   #MPLIER
7E0B: 20 84 7E >277           JSR   NEGATE
7E0E: B8       >278  :8       CLV              ;reset it to zero
7E0F: 60       >279  :7       RTS
               >280
7E10: 4C E1 EA >281  DVZERROR JMP   GODVZERR
               >282  * Signed 16bits integer divide routine
7E13: A5 C1    >283  SDIV     LDA   DIVSOR+1
7E15: 05 C0    >284           ORA   DIVSOR
7E17: F0 F7    >285           BEQ   DVZERROR
7E19: A5 C1    >286           LDA   DIVSOR+1
7E1B: C9 80    >287           CMP   #>$8000
7E1D: D0 19    >288           BNE   :2
7E1F: A5 C0    >289           LDA   DIVSOR
7E21: D0 13    >290           BNE   :1
               >291  * On traite le cas ou le diviseur est -32768
               >292  * Dans ce cas la si DIVEND vaut aussi -32768, alors
               >293  * retourne 1 sinon 0
7E23: A8       >294           TAY
7E24: AA       >295           TAX              ;X forced to zero
7E25: C5 C2    >296           CMP   DIVEND
7E27: D0 07    >297           BNE   :0
7E29: A5 C3    >298           LDA   DIVEND+1
7E2B: C9 80    >299           CMP   #>$8000
7E2D: D0 01    >300           BNE   :0
7E2F: E8       >301           INX
7E30: 86 C2    >302  :0       STX   DIVEND
7E32: 84 C3    >303           STY   DIVEND+1
7E34: D0 39    >304           BNE   NRET       Always
7E36: A5 C1    >305  :1       LDA   DIVSOR+1
7E38: 45 C3    >306  :2       EOR   DIVEND+1
7E3A: 48       >307           PHA              ;Sign bit on stack
7E3B: 20 73 7E >308           JSR   ZEROPRT    ;Absolute value for operands
7E3E: A0 10    >309  USDIV    LDY   #16
7E40: 06 C2    >310  ]LOOP    ASL   DIVEND
7E42: 26 C3    >311           ROL   DIVEND+1
7E44: 26 BE    >312           ROL   PARTIAL
7E46: 26 BF    >313           ROL   PARTIAL+1
7E48: 38       >314           SEC
7E49: A5 BE    >315           LDA   PARTIAL
```

```
7E4B: E5 C0    >316          SBC   DIVSOR
7E4D: AA       >317          TAX
7E4E: A5 BF    >318          LDA   PARTIAL+1
7E50: E5 C1    >319          SBC   DIVSOR+1
7E52: 90 06    >320          BCC   :3
7E54: 86 BE    >321          STX   PARTIAL
7E56: 85 BF    >322          STA   PARTIAL+1
7E58: E6 C2    >323          INC   DIVEND
7E5A: 88       >324  :3      DEY
7E5B: D0 E3    >325          BNE   ]LOOP
7E5D: 2C 72 7E >326          BIT   ARET+1     V set by default
7E60: A5 C2    >327          LDA   DIVEND
7E62: 25 C3    >328          AND   DIVEND+1
7E64: 1A       >332          INC
7E65: F0 0A    >334          BEQ   ARET       Keep V set and exit
7E67: 68       >335          PLA              ;Get back sign
7E68: 10 05    >336          BPL   NRET       No need to get result opposite
7E6A: A2 C2    >337          LDX   #DIVEND
7E6C: 20 84 7E >338          JSR   NEGATE
               >339  * Exit with V clear
7E6F: B8       >340  NRET    CLV
7E70: 70       >341          HEX   70         Skip next byte
7E71: 68       >342  ARET    PLA
7E72: 60       >343          RTS
               >344  * Zero partial and fall into ABSOPND
7E73: A0 00    >345  ZEROPRT LDY   #0
7E75: 84 BE    >346          STY   PARTIAL
7E77: 84 BF    >347          STY   PARTIAL+1
7E79: A2 C0    >348          LDX   #MCAND
7E7B: 20 80 7E >349          JSR   ABSOLUTE
7E7E: A2 C2    >350          LDX   #MPLIER    ;Fall into ABSOLUTE
               >351  * Compute absolute value of integer pointed to by X
               >352  * in ZP
7E80: B5 01    >353  ABSOLUTE LDA  1,X
7E82: 10 0B    >354          BPL   :0         No need
7E84: 38       >355          SEC
7E85: 98       >356          TYA              ;Y set to 0 upon entry
7E86: F5 00    >357          SBC   0,X
7E88: 95 00    >358          STA   0,X
7E8A: 98       >359          TYA
7E8B: F5 01    >360          SBC   1,X
7E8D: 95 01    >361          STA   1,X
7E8F: 60       >362  :0      RTS
               >363  NEGATE  EQU   ABSOLUTE+4
                519  * New processing for variable lookup
                520          PUT   PEERNPTRGET
               >1    MKNV    EQU   $E09C      Make new variable (ROM routine)
               >2    SETVYA  EQU   $E0DE      Set LOWTR and Y,A if var. found
               >3
7E90: A9 40    >4    NGETARPT LDA  #$40       $40: only look for arrays
7E92: 85 14    >5            STA   SUBFLG
               >6    * This routine is the new PTRGET routine from PEERSOFT
               >7    NPTRGTX
7E94: 64 10    >12           STZ   DIMFLG
               >14   NPTRGET
               >15   * Upon exit from the above routine, the X reg will
               >16   * contain the value X had upon call to CHRGOT (here zero)
```

```
7E96: 20 F4 7B >17             JSR   COMRST
               >18    * First variable name character must be alphabetic
7E99: 20 5C 82 >19             JSR   MISLETC
               >20
7E9C: 64 11    >27    NPTRGET1 STZ   VALTYP
7E9E: 64 12    >28             STZ   INTTYP
7EA0: 64 82    >29             STZ   VARNAM+1   Default zero for 2nd name char.
7EA2: 85 81    >31             STA   VARNAM
7EA4: 20 EC 7B >32             JSR   RST100
7EA7: 90 05    >33             BCC   GTLT       Branch if numeric digit
7EA9: 20 7D E0 >34             JSR   ISLETC
7EAC: 90 1A    >35             BCC   EXPLIC?    Branch if not alpha character
7EAE: AA       >36    GTLT     TAX              ;2nd character in X
7EAF: 86 82    >37             STX   VARNAM+1    and into VARNAM+1
               >38    * Skip subsequent alphanumeric characters
7EB1: 20 EC 7B >39    ]LOOP    JSR   RST100
7EB4: 90 FB    >40             BCC   ]LOOP      branch if numeric
7EB6: 20 7D E0 >41             JSR   ISLETC
7EB9: B0 F6    >42             BCS   ]LOOP      branch if alphabetic
7EBB: 90 0B    >43             BCC   EXPLIC?    Always
7EBD: 4C C9 DE >44    BADNAM   JMP   SYNERR
               >45    * Code run as no explicit type specifier found, get the
               >46    * default type specifier according to 1st varname char.
7EC0: 20 F1 85 >47    SCDCH2   JSR   DECTPTR
7EC3: A6 81    >48             LDX   VARNAM
7EC5: BD 61 9B >49             LDA   TYPLET-´A´,X
               >50    * Fall into implicit (2nd pass to EXPLIC?)
7EC8: 20 E6 85 >51    EXPLIC?  JSR   XFROMMOT   Get index from character
               >52    * No explicit type specifier found, so try implicit
               >53    * type specifier (cannot fail)
7ECB: D0 F3    >54             BNE   SCDCH2     Branch if no type spec. found
7ECD: BD 8C 9B >56             LDA   TVTVAL,X
7ED0: 85 11    >57             STA   VALTYP
7ED2: BD 88 9B >58             LDA   TITVAL,X
7ED5: 85 12    >59             STA   INTTYP
7ED7: BD 90 9B >60             LDA   TVNORA,X
7EDA: 04 81    >61             TSB   VARNAM
7EDC: BD 94 9B >62             LDA   TVN1ORA,X
7EDF: 04 82    >63             TSB   VARNAM+1
7EE1: E0 02    >64             CPX   #2         FP or string
7EE3: 90 04    >65             BCC   :6
7EE5: A5 14    >66             LDA   SUBFLG
7EE7: 30 D4    >67             BMI   BADNAM
7EE9: 20 EC 7B >68    :6       JSR   RST100     Get next character
7EEC: 38       >69             SEC
7EED: 05 14    >70             ORA   SUBFLG
7EEF: E9 28    >71             SBC   #´(´
7EF1: D0 03    >72             BNE   :8
7EF3: 4C 8F 7F >73    :7       JMP   NARRAY
7EF6: 24 14    >74    :8       BIT   SUBFLG
7EF8: 30 02    >75             BMI   :9
7EFA: 70 F7    >76             BVS   :7
               >77    :9       DO    KOPT-K6502
7EFC: 64 14    >78             STZ   SUBFLG
7EFE: AE A3 99 >83    NPTRGL90 LDX   SNCCH
7F01: F0 05    >84             BEQ   :90
7F03: 20 61 7F >85             JSR   SLKCACH
```

```
7F06: D0 56    >86             BNE   NAMFOUND   Found cache entry if Zbit clear
               >87   :90       DO    KOPT16
7F08: A6 69    >94             LDX   VARTAB
7F0A: A5 6A    >95             LDA   VARTAB+1
7F0C: 85 9C    >100  ]LOOP     STA   LOWTR+1
7F0E: 86 9B    >101  ]LOOP1    STX   LOWTR
7F10: E4 6B    >106            CPX   ARYTAB
7F12: E5 6C    >107            SBC   ARYTAB+1
7F14: B0 26    >109            BCS   NAMNTFND
7F16: B2 9B    >114            LDA   (LOWTR)
7F18: 45 81    >116            EOR   VARNAM
7F1A: D0 14    >117            BNE   :1
7F1C: A0 01    >120            LDY   #1
7F1E: B1 9B    >124            LDA   (LOWTR),Y
7F20: 45 82    >125            EOR   VARNAM+1
7F22: D0 0C    >126            BNE   :1
7F24: A5 12    >131            LDA   INTTYP
7F26: 10 36    >132            BPL   NAMFOUND
7F28: A0 06    >133            LDY   #6
7F2A: B1 9B    >134            LDA   (LOWTR),Y
7F2C: 45 12    >135            EOR   INTTYP
7F2E: F0 2E    >136            BEQ   NAMFOUND
               >141  * Name not yet found: look for next variable in memory
7F30: A5 9B    >142  :1        LDA   LOWTR
7F32: 69 07    >147            ADC   #7         Carry already clear
7F34: AA       >148            TAX
7F35: A5 9C    >149            LDA   LOWTR+1
7F37: 90 D5    >150            BCC   ]LOOP1
7F39: 1A       >152            INC
7F3A: D0 D0    >153            BNE   ]LOOP      Always
               >159
7F3C: BA       >168  NAMNTFND TSX
7F3D: BD 01 01 >169            LDA   STACK+1,X
7F40: C9 AB    >170            CMP   #RFFVL
7F42: D0 0A    >171            BNE   :0
7F44: BD 02 01 >172            LDA   STACK+2,X
7F47: C9 8A    >173            CMP   #>RFFVL
7F49: D0 03    >174            BNE   :0
7F4B: 4C 95 E0 >176            JMP   $E095      Return 0 constant
               >177  * Make new variable
7F4E: 20 9C E0 >178  :0        JSR   MKNV       Make new variable (ROM routine)
7F51: A5 12    >179            LDA   INTTYP     FP or string?
7F53: 10 06    >180            BPL   :1         Yes
7F55: A0 06    >181            LDY   #6
7F57: 91 9B    >182            STA   (LOWTR),Y
7F59: A4 84    >183            LDY   VARPNT+1
7F5B: A5 83    >184  :1        LDA   VARPNT
7F5D: 60       >185            RTS
               >186
               >187  NAMFOUND
7F5E: 4C DE E0 >193            JMP   SETVYA
               >194
               >195  * Cache mechanism for simple variables
               >196  SCTR      EQU   LOWTR
7F61: A4 82    >226  SLKCACH   LDY   VARNAM+1
7F63: A5 81    >227            LDA   VARNAM
7F65: 86 9B    >228            STX   SCTR
```

```
7F67: A2 00    >229           LDX   #0
7F69: DD A4 99 >230  ]LOOP    CMP   SVN,X
7F6C: D0 0F    >231           BNE   :0
7F6E: 98       >232           TYA
7F6F: DD A8 99 >233           CMP   SVNP1,X
7F72: D0 07    >234           BNE   :2
7F74: A5 12    >235           LDA   INTTYP
7F76: DD AC 99 >236           CMP   SIT,X
7F79: F0 08    >237           BEQ   :1
7F7B: A5 81    >238  :2       LDA   VARNAM
7F7D: E8       >239  :0       INX
7F7E: E4 9B    >240           CPX   SCTR
7F80: D0 E7    >241           BNE   ]LOOP
7F82: 60       >243           RTS
               >244
7F83: BD B0 99 >245  :1       LDA   SLTR,X
7F86: 85 9B    >246           STA   LOWTR
7F88: BD B4 99 >252           LDA   SLTRP1,X
7F8B: 85 9C    >253           STA   LOWTR+1
7F8D: 8A       >255           TXA
7F8E: 60       >256           RTS
                521  * New processing for array processing
                522           PUT   PEERNARRAY
               >1    * Module handling the new array processing strategy
               >2    ERR_BSCR =     $6B
               >3    ERR_RDIM =     $78
               >4    ERR_SYNT =     $10
               >5
               >6    NUMDIM   EQU   $0F
               >7    RESULT   EQU   $62
               >8    STACK    EQU   $0100
               >9    SUBERR   EQU   $E196      Raise a BAD SUBSCRIPT error
               >10   MEMERR   EQU   $D410
               >11   FACSIGN  EQU   $A2
               >12   STRNG2   EQU   $AD
               >13   REASON   EQU   $D3E3
               >14   GETARY   EQU   $E0ED
               >15   GETARY2  EQU   $E0EF      Compute addr. of 1st elm value
               >16   QINT     EQU   $EBF2
               >17
               >18   * MULTPLSS multiplies (STRNG2) by ((LOWTR),Y) leaving
               >19   * result in A,X. Hi byte also in Y
               >20   MULTPLSS EQU   $E2AD
               >21   MULTPLY1 EQU   $E2B6
               >22
7F8F: A5 14    >30   NARRAY   LDA   SUBFLG
7F91: D0 4B    >32            BNE   NARRGL91
7F93: A5 10    >38            LDA   DIMFLG
7F95: 48       >39            PHA
7F96: A5 12    >40            LDA   INTTYP
7F98: 48       >41            PHA
7F99: A5 11    >42            LDA   VALTYP
7F9B: 48       >43            PHA
7F9C: A0 00    >45            LDY   #0
               >46   ]LOOP    MPHY
7F9E: 5A       >46            PHY
7F9F: A5 82    >53            LDA   VARNAM+1
```

```
7FA1: 48       >54          PHA
7FA2: A5 81    >55          LDA   VARNAM
7FA4: 48       >56          PHA
7FA5: 20 83 81 >58          JSR   NMAKINT
7FA8: 68       >65          PLA
7FA9: 85 81    >66          STA   VARNAM     Restore array name
7FAB: 68       >69          PLA
7FAC: 85 82    >70          STA   VARNAM+1
7FAE: 7A       >72          PLY
               >73    * Code below would transform the stack area
               >74    * from
               >75    *  DIMFLG
               >76    *  INTTYP
               >77    *  VALTYP
               >78    * SPtr ->
               >79    * to
               >80    *  (FAC+3)
               >81    *  (FAC+4)
               >82    *  DIMFLG
               >83    *  INTTYP
               >84    *  VALTYP
               >85    * SPtr ->
7FAF: BA       >100         TSX
7FB0: BD 02 01 >101         LDA   STACK+2,X  Get INTTYP
7FB3: 48       >102         PHA
7FB4: BD 01 01 >103         LDA   STACK+1,X  Get VALTYP
7FB7: 48       >104         PHA
7FB8: BD 03 01 >105         LDA   STACK+3,X  Get DIMFLG
7FBB: 9D 01 01 >106         STA   STACK+1,X  In place of original VALTYP
7FBE: A5 A0    >107         LDA   FAC+3
7FC0: 9D 03 01 >108         STA   STACK+3,X  In place of original DIMFLG
7FC3: A5 A1    >109         LDA   FAC+4
7FC5: 9D 02 01 >110         STA   STACK+2,X  In place of original INTTYP
               >112   * Now the stack frame looks like
               >113   *  FAC+4
               >114   *  FAC+3
               >115   *  DIMFLG
               >116   *  INTTYP
               >117   *  VALTYP
               >118   * SPtr ->
7FC8: C8       >119         INY
7FC9: 20 F4 7B >120         JSR   RST102
7FCC: C9 2C    >121         CMP   #´,´
7FCE: F0 CE    >122         BEQ   ]LOOP
7FD0: 84 0F    >123         STY   NUMDIM
7FD2: 20 71 8B >124         JSR   NCHKCLS
7FD5: 68       >125         PLA
7FD6: 85 11    >126         STA   VALTYP
7FD8: 68       >127         PLA
7FD9: 85 12    >128         STA   INTTYP
7FDB: 68       >129         PLA
7FDC: 85 10    >130         STA   DIMFLG
               >131
               >132
7FDE: AE B8 99 >133   NARRGL91 LDX   ANCCH
7FE1: F0 05    >134         BEQ   :20
7FE3: 20 B2 81 >135         JSR   ALKCACH
```

```
7FE6: D0 3D    >136              BNE   USEOLDAR
7FE8: A5 6C    >147  :20         LDA   ARYTAB+1
7FEA: A6 6B    >148              LDX   ARYTAB
7FEC: 86 9B    >149  ]LOOP       STX   LOWTR
7FEE: 85 9C    >150              STA   LOWTR+1
7FF0: E4 6D    >151              CPX   STREND
7FF2: E5 6E    >152              SBC   STREND+1
7FF4: B0 2C    >154              BCS   GNARRAY
7FF6: B2 9B    >156              LDA   (LOWTR)
7FF8: 45 81    >161              EOR   VARNAM
7FFA: D0 18    >162              BNE   :5
7FFC: A0 01    >169              LDY   #1
7FFE: B1 9B    >173              LDA   (LOWTR),Y
8000: 45 82    >174              EOR   VARNAM+1
8002: D0 10    >175              BNE   :5
8004: A6 12    >177              LDX   INTTYP
8006: 10 1D    >178              BPL   USEOLDAR   If FP or string array
8008: 20 AA 81 >179              JSR   CNVT1
800B: A0 04    >180              LDY   #4
800D: 51 9B    >181              EOR   (LOWTR),Y
800F: 29 C0    >182              AND   #$C0       only test b6 and b7
8011: F0 12    >183              BEQ   USEOLDAR
8013: 18       >191              CLC
               >192  :5
8014: A0 02    >194              LDY   #2
8016: B1 9B    >196              LDA   (LOWTR),Y
8018: 65 9B    >197              ADC   LOWTR
801A: AA       >199              TAX
801B: C8       >200              INY
801C: B1 9B    >201              LDA   (LOWTR),Y
801E: 65 9C    >202              ADC   LOWTR+1
8020: 90 CA    >204              BCC   ]LOOP      Always
               >205
               >206  GNARRAY
8022: 4C 8C 80 >211              JMP   MKNARRAY
               >212
8025: A5 10    >213  USEOLDAR LDA   DIMFLG     Called from the DIM stmt.?
8027: D0 5E    >214              BNE   RDIMERR
8029: A5 14    >215              LDA   SUBFLG     Subscripts given?
802B: F0 02    >216              BEQ   :1         Yes
802D: 38       >217              SEC              ;No: just return "array found"
802E: 60       >218              RTS
802F: 20 A0 81 >219  :1          JSR   NGETARY    Set ARYPNT to 1st elm. base addr
8032: A5 0F    >220              LDA   NUMDIM
8034: C9 01    >221              CMP   #1
8036: F0 07    >222              BEQ   :3
8038: E4 0F    >223              CPX   NUMDIM
803A: D0 45    >224              BNE   SUBSERR
803C: 4C 27 81 >225              JMP   NFAEP
               >226
               >227  * Il s´agit de traiter de la reference unidimensionnelle
               >228  * sur un tableau potentiellement multi-dimensions
               >229  * Multiplier l´indice tire dans la pile par le elm size
               >230  * et comparer par rapport a l´offset du tableau (corrige
               >231  * de la taille du header).
803F: 68       >232  :3          PLA
8040: 85 AD    >233              STA   STRNG2
```

```
8042: 68       >234         PLA
8043: 85 AE    >235         STA   STRNG2+1
8045: 20 70 81 >236         JSR   KWELMSIZ
8048: 86 64    >237         STX   RESULT+2
804A: A9 00    >238         LDA   #0
804C: 20 B6 E2 >239         JSR   MULTPLY1
804F: 86 AD    >240         STX   STRNG2
8051: 84 AE    >241         STY   STRNG2+1
8053: A0 04    >242         LDY   #4
8055: B1 9B    >243         LDA   (LOWTR),Y  # of dimensions
8057: 29 0F    >244         AND   #$0F       Mask out new Peersoft bits
8059: 0A       >245         ASL              ;2 bytes per dimension
805A: 69 05    >246         ADC   #5         Carry clear
               >247  * Add this to element offset from base address
805C: 65 AD    >248         ADC   STRNG2
805E: A6 AE    >249         LDX   STRNG2+1
8060: 90 01    >250         BCC   :4
8062: E8       >251         INX
8063: A0 02    >252  :4     LDY   #2
8065: D1 9B    >253         CMP   (LOWTR),Y
8067: 85 83    >254         STA   VARPNT
8069: C8       >255         INY
806A: 8A       >256         TXA
806B: F1 9B    >257         SBC   (LOWTR),Y
806D: B0 12    >258         BCS   SUBSERR
806F: 86 84    >259         STX   VARPNT+1
8071: A5 9B    >260         LDA   LOWTR
8073: 65 83    >261         ADC   VARPNT
8075: 85 83    >262         STA   VARPNT
8077: A5 84    >263         LDA   VARPNT+1
8079: 65 9C    >264         ADC   LOWTR+1
807B: 85 84    >265         STA   VARPNT+1
807D: A8       >266         TAY
807E: A5 83    >267         LDA   VARPNT
8080: 60       >268         RTS
               >269
8081: A2 6B    >270  SUBSERR  LDX   #ERR_BSCR
8083: 2C       >271         HEX   2C         Skip next two bytes
8084: A2 10    >272  SNERR    LDX   #ERR_SYNT
8086: 2C       >273         HEX   2C
8087: A2 78    >274  RDIMERR  LDX   #ERR_RDIM
8089: 4C 12 D4 >275         JMP   $D412
               >276
808C: A5 14    >277  MKNARRAY LDA   SUBFLG
808E: F0 03    >278         BEQ   :0
8090: 4C DC E1 >279         JMP   $E1DC      Raise OUT OF DATA error
8093: 20 ED E0 >280  :0     JSR   GETARY     Address 1st elm in ARYPNT&Y,A
8096: 20 70 81 >281         JSR   KWELMSIZ
8099: 86 AD    >282         STX   STRNG2
809B: A6 10    >283         LDX   DIMFLG
809D: 86 BF    >284         STX   AUXBANK
809F: F0 03    >285         BEQ   :1
80A1: 20 F6 81 >286         JSR   ISAUXMEM
80A4: A5 94    >287  :1     LDA   ARYPNT
80A6: 20 E3 D3 >288         JSR   REASON     Ensure enough memory for array
               >289
80A9: A5 81    >290         LDA   VARNAM
```

```
80AB: 64 AE    >292                STZ   STRNG2+1
80AD: 92 9B    >293                STA   (LOWTR)
80AF: A0 01    >294                LDY   #1
80B1: A5 82    >301                LDA   VARNAM+1
80B3: 91 9B    >302                STA   (LOWTR),Y
80B5: A0 04    >303                LDY   #4
80B7: A5 12    >304                LDA   INTTYP
80B9: F0 04    >305                BEQ   :2
80BB: AA       >306                TAX
80BC: 20 AA 81 >307                JSR   CNVT1
80BF: 05 0F    >308  :2            ORA   NUMDIM
80C1: 91 9B    >309                STA   (LOWTR),Y
80C3: A9 00    >310  ]LOOP         LDA   #0          Hi byte of default dim
80C5: A2 0B    >311                LDX   #11         Lo byte of default dim
80C7: 24 10    >312                BIT   DIMFLG
80C9: 50 08    >313                BVC   :5
80CB: 68       >314                PLA
80CC: 18       >315                CLC
80CD: 69 01    >316                ADC   #1
80CF: AA       >317                TAX
80D0: 68       >318                PLA
80D1: 69 00    >319                ADC   #0
80D3: C8       >320  :5            INY               ;Add this dimension to descr.
80D4: 91 9B    >321                STA   (LOWTR),Y
80D6: C8       >322                INY
80D7: 8A       >323                TXA
80D8: 91 9B    >324                STA   (LOWTR),Y
               >325  * Multiply this dimension by running size
               >326  * ((LOWTR),Y) * (STRNG2) --> A,X
80DA: 20 AD E2 >327                JSR   MULTPLSS
80DD: 86 AD    >328                STX   STRNG2
80DF: 85 AE    >329                STA   STRNG2+1
80E1: A4 5E    >330                LDY   INDEX
80E3: C6 0F    >331                DEC   NUMDIM
80E5: D0 DC    >332                BNE   ]LOOP
               >333  * Now A,X has the total # of bytes of array elements
80E7: 65 95    >334                ADC   ARYPNT+1    Compute address of end of array
80E9: B0 60    >335                BCS   GME         Too large: error
80EB: 85 95    >336                STA   ARYPNT+1
80ED: A8       >337                TAY
80EE: 8A       >338                TXA
80EF: 65 94    >339                ADC   ARYPNT
80F1: 90 03    >340                BCC   :6
80F3: C8       >341                INY
80F4: F0 55    >342                BEQ   GME         Too large: error
80F6: 20 E3 D3 >343  :6            JSR   REASON      Ensure enough room up to Y,A
80F9: 85 6D    >344                STA   STREND
80FB: 84 6E    >345                STY   STREND+1
               >346  * Zero fill the element segment within the array
               >347  * (fast init).
80FD: A9 00    >348                LDA   #0
80FF: E6 AE    >349                INC   STRNG2+1
8101: A4 AD    >350                LDY   STRNG2      # of byte mod 256
8103: F0 05    >351                BEQ   :8          Upon a page limit
8105: 88       >352  ]LOOP         DEY
8106: 91 94    >353                STA   (ARYPNT),Y
8108: D0 FB    >354                BNE   ]LOOP
```

```
810A: C6 95    >355  :8        DEC   ARYPNT+1   Point to next page
810C: C6 AE    >356            DEC   STRNG2+1   Count the pages
810E: D0 F5    >357            BNE   ]LOOP      Still more to clear
8110: E6 95    >358            INC   ARYPNT+1   Rollback last Decrement
8112: 38       >359            SEC
8113: A5 6D    >360            LDA   STREND
8115: E5 9B    >361            SBC   LOWTR
8117: A0 02    >362            LDY   #2
8119: 91 9B    >363            STA   (LOWTR),Y
811B: C8       >364            INY
811C: A5 6E    >365            LDA   STREND+1
811E: E5 9C    >366            SBC   LOWTR+1
8120: 91 9B    >367            STA   (LOWTR),Y
8122: A5 10    >368            LDA   DIMFLG
8124: F0 01    >369            BEQ   NFAEP
8126: 60       >370            RTS
               >371
8127: A0 04    >372  NFAEP     LDY   #4
               >373  * New routine for ROM FIND.ARRAY.ELEMENT
               >374  * Y reg. should be 4 upon entry
8129: B1 9B    >375            LDA   (LOWTR),Y
812B: 29 0F    >376            AND   #$0F
812D: 85 0F    >377            STA   NUMDIM
812F: A9 00    >378            LDA   #0
8131: 85 AD    >379            STA   STRNG2
8133: 85 AE    >380  FAE1      STA   STRNG2+1
8135: C8       >381            INY              ;Pull next subscript from stack
8136: FA       >382            PLX
8137: 86 A0    >383            STX   FAC+3
8139: 68       >384            PLA
813A: 85 A1    >385            STA   FAC+4
813C: D1 9B    >386            CMP   (LOWTR),Y
813E: 90 0E    >387            BCC   FAE2
8140: D0 06    >388            BNE   GSE        Subscript is too large
8142: C8       >389            INY
8143: 8A       >390            TXA
8144: D1 9B    >391            CMP   (LOWTR),Y
8146: 90 07    >392            BCC   FAE3
8148: 4C 96 E1 >393  GSE       JMP   SUBERR     BAD SUBSCRIPT error
814B: 4C 10 D4 >394  GME       JMP   MEMERR     MEMORY FULL error
814E: C8       >395  FAE2      INY
814F: A5 AE    >396  FAE3      LDA   STRNG2+1   Bypass multiplication if
8151: 05 AD    >397            ORA   STRNG2      value so far is zero
8153: 18       >398            CLC
8154: F0 0A    >399            BEQ   :1
8156: 20 AD E2 >400            JSR   MULTPLSS
8159: 8A       >401            TXA              ;Add current subscript
815A: 65 A0    >402            ADC   FAC+3
815C: AA       >403            TAX
815D: 98       >404            TYA
815E: A4 5E    >405            LDY   INDEX
8160: 65 A1    >406  :1        ADC   FAC+4      Finish adding current subscrit
8162: 86 AD    >407            STX   STRNG2     Store accumulated offset
8164: C6 0F    >408            DEC   NUMDIM     Last subscript yet?
8166: D0 CB    >409            BNE   FAE1       No: loop till done
8168: 85 AE    >410            STA   STRNG2+1   Yes: mmultiply by element size
816A: 20 70 81 >411            JSR   KWELMSIZ
```

```
816D: 4C 98 E2 >412          JMP   $E298
               >413
               >414  * Donne la taille de l´element en fonction
               >415  * de VARNAM,+1 et de INTTYP
               >416  * Result in X reg.
8170: 24 81    >417  KWELMSIZ BIT   VARNAM
8172: 10 06    >418          BPL   :0
8174: A5 12    >419          LDA   INTTYP
8176: 29 07    >420          AND   #7
8178: AA       >421          TAX
8179: 60       >422          RTS
817A: A2 05    >423  :0      LDX   #5
817C: 24 82    >424          BIT   VARNAM+1
817E: 10 02    >425          BPL   :1
8180: CA       >426          DEX              ;Back to 3 if string
8181: CA       >427          DEX
8182: 60       >428  :1      RTS
               >429
               >430  * Evaluate numeric formula at TXTPPTR
               >431  * Converting result to INTEGER 0<= X < 65536
               >432  * into FAC+3,4
8183: 20 EC 7B >433  NMAKINT  JSR   RST100     Get next character
8186: 20 5D 8A >434          JSR   NFRMNUM
               >435  * Convert FAC to integer
8189: A5 A2    >436          LDA   FACSIGN
818B: 30 10    >437          BMI   :1
818D: A5 9D    >438          LDA   FAC
818F: C9 90    >439          CMP   #$90
8191: 90 07    >440          BCC   :3         Branch if abs(value) < 32768
8193: A9 98    >441          LDA   #NEG65536
8195: A0 9B    >442          LDY   #>NEG65536
8197: 20 BE E7 >443          JSR   FADD
819A: 4C F2 EB >444  :3      JMP   QINT
819D: 4C 99 E1 >445  :1      JMP   GOIQERR
               >446  LN65536  EQU   *-13
               >447
81A0: A0 04    >448  NGETARY  LDY   #4
81A2: B1 9B    >449          LDA   (LOWTR),Y
81A4: 29 0F    >450          AND   #$0F
81A6: AA       >451          TAX
81A7: 4C EF E0 >452          JMP   GETARY2
               >453
               >454  * Convert INTTYP (in X reg.) from $81 to $84
               >455  * to %0000_0000 to %1100_0000 (respectively)
               >456  * Output value could be ORA ed or EOR ed with
               >457  * NUMDIM slot with array structure
81AA: CA       >458  CNVT1    DEX
81AB: 8A       >459          TXA
81AC: 4A       >460          LSR              ;b0 into Carry, 0 into b7
81AD: 6A       >461          ROR              ;b0 into b7 and b1 into carry
81AE: 6A       >462          ROR              ;b0 into b6, b1 into b7
81AF: 29 C0    >463          AND   #$C0       Only retain b6-b7
81B1: 60       >464          RTS
               >465
               >466  * Cache mechanism for array variables
               >467  ACTR     EQU   LOWTR
81B2: A4 82    >496  ALKCACH  LDY   VARNAM+1
```

```
81B4: A5 81    >497             LDA   VARNAM
81B6: 86 9B    >498             STX   SCTR
81B8: A2 00    >499             LDX   #0
81BA: DD B9 99 >500   ]LOOP     CMP   AVN,X
81BD: D0 0F    >501             BNE   :0
81BF: 98       >502             TYA
81C0: DD BD 99 >503             CMP   AVNP1,X
81C3: D0 07    >504             BNE   :2
81C5: A5 12    >505             LDA   INTTYP
81C7: DD C1 99 >506             CMP   AIT,X
81CA: F0 08    >507             BEQ   :1
81CC: A5 81    >508   :2        LDA   VARNAM
81CE: E8       >509   :0        INX
81CF: E4 9B    >510             CPX   SCTR
81D1: D0 E7    >511             BNE   ]LOOP
81D3: 60       >513             RTS
               >514
81D4: BD C5 99 >515   :1        LDA   ALTR,X
81D7: 85 9B    >516             STA   LOWTR
81D9: BD C9 99 >521             LDA   ALTRP1,X
81DC: 85 9C    >522             STA   LOWTR+1
81DE: 8A       >524             TXA
81DF: 60       >525             RTS
               >526
               >527   * Common entry point for accessing array content
               >528   * within auxiliary memory.
81E0: A2 BF    >529   ZRTAUX    LDX   #$BF
81E2: 8E EE 03 >530             STX   $03EE
81E5: 9C ED 03 >532             STZ   $03ED
81E8: B8       >537             CLV
81E9: 38       >538             SEC
81EA: 4C 14 C3 >539             JMP   XFER
                523   * New strategy for array storage
                524             PUT   PEERNAUXMEM
               >1     * Module handling the new Peersoft array storage strategy
               >2
81ED: 4C C9 DE >3     GSNERR2   JMP   SYNERR
81F0: 4C 99 E1 >4     GIQERR2   JMP   GOIQERR
81F3: 4C 76 DD >5     GTMERR2   JMP   GOTMIERR
               >6     * Routine to test whether the array will be located
               >7     * Outcome:
               >8     * Carry set iif aux. mem storage asked for
               >9     * AUXBANK: bank memory asked for (in bits b4..b5)
               >10    * ARYPNT,+1: incremented if aux mem. storage
               >11    * (placeholders for offset within aux memory and
               >12    *  one element of specified size for returning values
               >13    *  during value expressions
               >14    * Y,A: values incremented in case aux. mem storage
81F6: B2 B8    >16    ISAUXMEM LDA   (TXTPTR)
81F8: C9 23    >21              CMP   #´#´
81FA: 18       >22              CLC
81FB: D0 34    >23              BNE   :2
81FD: 20 EC 7B >24              JSR   RST100      Next char. must be numeric
8200: B0 EB    >25              BCS   GSNERR2     otherwise SYNTAX ERROR
8202: AA       >26              TAX
8203: 20 EC 7B >27              JSR   RST100      Point to next character
8206: 18       >28              CLC
```

```
8207: 2C EF 9C >29              BIT   MEMORY
820A: 50 25    >30              BVC   :2
820C: 8A       >31              TXA
820D: 0A       >32              ASL
820E: 0A       >33              ASL
820F: 0A       >34              ASL
8210: 0A       >35              ASL
8211: 85 BF    >36              STA   AUXBANK
8213: F0 1C    >37              BEQ   :2
8215: 24 11    >38              BIT   VALTYP
8217: 30 DA    >39              BMI   GTMERR2
8219: C9 20    >40              CMP   #$20
821B: B0 D3    >41              BCS   GIQERR2
821D: A5 94    >42              LDA   ARYPNT
821F: A4 95    >43              LDY   ARYPNT+1
8221: 65 AD    >44              ADC   STRNG2     Carry already clear
8223: 90 02    >45              BCC   :0
8225: C8       >46              INY
8226: 18       >47              CLC
8227: 69 02    >48    :0        ADC   #2
8229: 90 01    >49              BCC   :1
822B: C8       >50              INY
822C: 84 95    >51    :1        STY   ARYPNT+1
822E: 38       >52              SEC
822F: 85 94    >53              STA   ARYPNT
8231: A5 94    >54    :2        LDA   ARYPNT
8233: 60       >55              RTS
                525
                526   * Upon init, all variables are floating point by default
8234: 08        527   LBS00     PHP
8235: A2 1A     528             LDX   #26
8237: A9 21     529             LDA   #´!´
8239: 9D A1 9B  530   ]LOOP     STA   TYPLET-1,X
823C: CA        531             DEX
823D: D0 FA     532             BNE   ]LOOP
                533   * Reinit variables lookup caches (simple & array)
823F: 8E A3 99  534             STX   SNCCH
8242: 8E B8 99  535             STX   ANCCH
8245: 28        536             PLP
8246: 60        537             RTS
                538
                539   * Applesoft RUN command
8247: 20 34 82  540   RRUN      JSR   LBS00      Init the default vartype table
824A: 8E E1 99  541             STX   MONU       Rearms MOUSE instruction flag
824D: 4C 12 D9  542             JMP   $D912
                543
                544   * Applesoft NEW command
8250: 20 34 82  545   RNEW      JSR   LBS00
8253: 4C 4B D6  546             JMP   $D64B
                547
                548   * Applesoft CLEAR command
8256: 20 34 82  549   RCLEAR    JSR   LBS00
8259: 4C 6C D6  550             JMP   $D66C
                551
825C: 20 7D E0  552   MISLETC   JSR   ISLETC
825F: 90 08     553             BCC   GOSYNERR
8261: 60        554             RTS
```

```
                 555
                 556  * New subroutine checking a character (code in A)
                 557  * is pointed to by TXTPTR
                 558  * Falls into SYNERR if not
                 559  NSYNCHR  DO    KOPT-K65C02
8262: D2 B8      563  NSYNCHR2 CMP   (TXTPTR)
8264: D0 03      565           BNE   GOSYNERR
8266: 4C EC 7B   566           JMP   RST100
8269: 4C C9 DE   567  GOSYNERR JMP   SYNERR
                 568
                 569           PUT   PEERPROCFUN
                >1    * Module en charge des fonctions utilisateur
                >2    * et particulierement des PF
                >3    ARG      EQU   $A5
                >4    TRCFLG   EQU   $F2
                >5    BISVTYP  EQU   $BE
                >6    VECTUSR  EQU   $A
                >7    TMERR    EQU   $DD76
                >8    ULERR    EQU   $D97C
                >9    MOVFM    EQU   $EAF9
                >10   MOVFA    EQU   $EB53
                >11   LET2     EQU   $DA63
                >12
                >13            DUMMY 0
0000: 00        >14   USRMOD   DS    1
0001: 00 00     >15   ADRUSR   DS    2
0003: 00 00     >16   VSRTNAM  DS    2
0005: 00        >17   VSRTVT   DS    1
0006: 00        >18   VSRTIT   DS    1
0007: 00 00     >19   VSRTPTR  DS    2
0009: 00 00     >20   VENT1NAM DS    2
000B: 00        >21   VENT1VT  DS    1
000C: 00        >22   VENT1IT  DS    1
000D: 00 00     >23   VENT1PTR DS    2
000F: 00 00     >24   VENT2NAM DS    2
0011: 00        >25   VENT2VT  DS    1
0012: 00        >26   VENT2IT  DS    1
0013: 00 00     >27   VENT2PTR DS    2
                >28   LENREC   EQU   *
                >29            DEND
                >30   * Sous routine pour initialiser les routines USR de type
                >31   * PF.
826C: A2 0A     >32   RAZPF    LDX   #10
                >33   ]LOOP    MPHX
826E: DA        >33            PHX
826F: 20 95 82 >34             JSR   COMPOFST
8272: FA        >35            PLX
8273: B2 06     >37            LDA   (AUXPTR)
8275: 10 06     >42            BPL   :0
8277: A0 02     >43            LDY   #ADRUSR+1
8279: A9 00     >44            LDA   #0
827B: 91 06     >45            STA   (AUXPTR),Y
827D: CA        >46   :0       DEX
827E: 10 EE     >47            BPL   ]LOOP
8280: 8E A1 99 >48             STX   PFINDIC
8283: 9C A0 99 >50             STZ   ISPFACT
8286: 60        >55            RTS
```

```
                  >56
8287: A2 0B       >57   SETINITX LDX   #12-1
8289: BD 94 99 >58   ]LOOP    LDA   SINITX,X
828C: 95 69       >59            STA   $69,X
828E: 9D 74 97 >60            STA   SVALTNM,X
8291: CA          >61            DEX
8292: 10 F5       >62            BPL   ]LOOP
8294: 60          >63            RTS
                  >64
                  >65   * Indice de la fonction dans X, ramene dans A,Y
                  >66   * L´adresse de debut de la structure
8295: A9 00       >67   COMPOFST LDA   #0
8297: A8          >68            TAY
8298: F0 05       >69            BEQ   :00         Always
829A: 69 15       >70   ]LOOP    ADC   #LENREC
829C: 90 02       >71            BCC   :0
829E: C8          >72            INY
829F: 18          >73   :00      CLC
82A0: CA          >74   :0       DEX
82A1: 10 F7       >75            BPL   ]LOOP
82A3: 69 65       >76            ADC   #ADRSTRUCT
82A5: 48          >77            PHA
82A6: 98          >78            TYA
82A7: 69 96       >79            ADC   #>ADRSTRUCT
82A9: A8          >80            TAY
82AA: 68          >81            PLA
82AB: 85 06       >82            STA   AUXPTR
82AD: 84 07       >83            STY   AUXPTR+1
82AF: 60          >84            RTS
                  >85
82B0: 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
82B1: 4C 76 DD >90   ]ERR     JMP   TMERR
82B4: 48          >91   FRSTIM   PHA
82B5: 20 74 8B >92            JSR   NCHKCOM
82B8: B2 06       >94            LDA   (AUXPTR)
82BA: 29 01       >99            AND   #1          Environnement dynamique oui/non
82BC: 48          >100           PHA
82BD: F0 0F       >101           BEQ   :0
82BF: A2 0B       >102           LDX   #12-1
82C1: B5 69       >103  ]LOOP    LDA   $69,X
82C3: 9D 68 97 >104           STA   SVCURRM,X
82C6: BD 88 97 >105           LDA   SDEF1,X
82C9: 95 69       >106           STA   $69,X
82CB: CA          >107           DEX
82CC: 10 F3       >108           BPL   ]LOOP
82CE: A5 07       >112  :0       LDA   AUXPTR+1
82D0: 48          >113           PHA
82D1: A5 06       >114           LDA   AUXPTR
82D3: 48          >115           PHA
82D4: 20 94 7E >117           JSR   NPTRGTX
82D7: C5 6B       >118           CMP   ARYTAB
82D9: 98          >119           TYA
82DA: E5 6C       >120           SBC   ARYTAB+1
82DC: 68          >121           PLA
```

```
82DD: 85 06    >122          STA   AUXPTR
82DF: 68       >123          PLA
82E0: 85 07    >124          STA   AUXPTR+1
82E2: 68       >125          PLA
82E3: F0 0A    >126          BEQ   :1
82E5: A2 0B    >127          LDX   #12-1
82E7: BD 68 97 >128  ]LOOP    LDA   SVCURRM,X
82EA: 95 69    >129          STA   $69,X
82EC: CA       >130          DEX
82ED: 10 F8    >131          BPL   ]LOOP
82EF: B0 C0    >132  :1       BCS   ]ERR
82F1: 7A       >133          PLY
82F2: A5 81    >134          LDA   VARNAM
82F4: 91 06    >135          STA   (AUXPTR),Y
82F6: C8       >136          INY
82F7: A5 82    >137          LDA   VARNAM+1
82F9: 91 06    >138          STA   (AUXPTR),Y
82FB: C8       >139          INY
82FC: A5 11    >140          LDA   VALTYP
82FE: 91 06    >141          STA   (AUXPTR),Y
8300: C8       >142          INY
8301: A5 12    >143          LDA   INTTYP
8303: 91 06    >144          STA   (AUXPTR),Y
8305: C8       >145          INY
8306: A5 83    >146  COMX1    LDA   VARPNT
8308: 91 06    >147          STA   (AUXPTR),Y
830A: C8       >148          INY
830B: A5 84    >149          LDA   VARPNT+1
830D: 91 06    >150          STA   (AUXPTR),Y
830F: 60       >151          RTS
               >152
               >153  * Connaitre tout d´une variable deja enregistree
               >154  * Y offset dans structure... (adressage par
               >155  * (AUXPTR),Y
8310: B1 06    >156  SCNDTIM  LDA   (AUXPTR),Y
8312: 85 81    >157          STA   VARNAM
8314: C8       >158          INY
8315: B1 06    >159          LDA   (AUXPTR),Y
8317: 85 82    >160          STA   VARNAM+1
8319: C8       >161          INY
831A: B1 06    >162          LDA   (AUXPTR),Y
831C: 85 11    >163          STA   VALTYP
831E: C8       >164          INY
831F: B1 06    >165          LDA   (AUXPTR),Y
8321: 85 12    >166          STA   INTTYP
8323: C8       >167          INY
8324: 5A       >168          PHY
8325: 20 FE 7E >169          JSR   NPTRGL90
8328: 7A       >170          PLY
8329: 80 DB    >171          BRA   COMX1
               >172
               >173  * X,A adresse a sauver dans ADRUSR de la structure
832B: A0 01    >174  HNDLEADR LDY   #ADRUSR
832D: 91 06    >175          STA   (AUXPTR),Y
832F: 90 08    >176          BCC   :4
8331: 85 0B    >177          STA   $0B
8333: 86 0C    >178          STX   $0C
```

```
8335: A9 4C    >179          LDA   #$4C
8337: 85 0A    >180          STA   $0A
8339: C8       >181  :4       INY
833A: 8A       >182          TXA
833B: 91 06    >183          STA   (AUXPTR),Y
833D: 60       >184          RTS
               >185
833E: B1 06    >186  COMLET2  LDA   (AUXPTR),Y
8340: AA       >187          TAX              ;INTTYP dans X
8341: C8       >188          INY
8342: B1 06    >189          LDA   (AUXPTR),Y ;pointeur sur valeur
8344: 85 85    >190          STA   FORPNT      dans FORPNT
8346: C8       >191          INY
8347: B1 06    >192          LDA   (AUXPTR),Y
8349: 85 86    >193          STA   FORPNT+1
834B: 8A       >194          TXA              ;Set bit N
834C: 4C 63 DA >195          JMP   LET2
               >196
834F: 4C 10 D4 >197  ]ERR     JMP   MEMERR
8352: 20 EC 7B >198  RUSR     JSR   RST100
8355: A2 0A    >199          LDX   #10
8357: B0 06    >200          BCS   :0         Not a digit
8359: E9 2F    >201          SBC   #´0´-1
835B: AA       >202          TAX
835C: 20 EC 7B >203          JSR   RST100
               >204  :0       MPHX
835F: DA       >204          PHX
8360: 20 95 82 >205          JSR   COMPOFST
8363: B2 06    >207          LDA   (AUXPTR)
8365: 29 40    >212          AND   #64
8367: F0 41    >213          BEQ   :1
8369: BA       >214          TSX
836A: E0 08    >215          CPX   #8         At least 8 bytes on stack OK
836C: 90 E1    >216          BCC   ]ERR
836E: 20 77 8B >217          JSR   NCHKOPN
8371: 20 7B DD >218          JSR   FRMEVL
8374: BA       >219          TSX
8375: A5 11    >220          LDA   VALTYP
8377: 9D 00 01 >221          STA   $0100,X
837A: 8A       >222          TXA
837B: 38       >223          SEC
837C: E9 06    >224          SBC   #6
837E: AA       >225          TAX
837F: 9A       >226          TXS
8380: E8       >227          INX
8381: A0 01    >228          LDY   #1
8383: 20 2B EB >229          JSR   MOVMF
8386: 20 74 8B >230          JSR   NCHKCOM
8389: 20 6E 8B >231          JSR   NPARCHK+3  2nd arg value left in FAC
838C: BA       >232          TSX
838D: E8       >233          INX
838E: 8A       >234          TXA
838F: 48       >235          PHA
8390: A0 01    >236          LDY   #1
8392: 20 E3 E9 >237          JSR   $E9E3      Load ARG from Y,A/1st arg value
8395: 68       >238          PLA
8396: 18       >239          CLC
```

```
8397: 69 05    >240            ADC   #5          6 instead of 5 because of INX
8399: AA       >241            TAX
839A: BD 00 01 >242            LDA   $0100,X
839D: 85 BE    >243            STA   BISVTYP
839F: 9A       >244            TXS
83A0: 80 0B    >245            BRA   :2
83A2: A2 26    >246  ]ERR      LDX   #38
83A4: 2C       >247            HEX   2C          Skip next two bytes
83A5: A2 27    >248  ]ERR1     LDX   #39
83A7: 4C D0 92 >249            JMP   NERRH
83AA: 20 6B 8B >250  :1        JSR   NPARCHK     1er ou 2eme parm dans FAC
               >251  :2        MPLX
83AD: FA       >251            PLX
83AE: DA       >253            PHX
83AF: 20 95 82 >257            JSR   COMPOFST    Set AUXPTR according index X
83B2: A0 02    >258            LDY   #ADRUSR+1
83B4: B1 06    >259            LDA   (AUXPTR),Y
83B6: F0 EA    >260            BEQ   ]ERR
83B8: FA       >261            PLX
83B9: 8E A2 99 >262            STX   PFINDX
83BC: B2 06    >264            LDA   (AUXPTR)
83BE: 10 48    >269            BPL   :3
               >270  * Procedural function...
83C0: 4A       >271            LSR
83C1: 90 2A    >272            BCC   :10         Branchem. ssi pas de segment
83C3: AD A0 99 >273            LDA   ISPFACT
83C6: D0 DD    >274            BNE   ]ERR1
83C8: DA       >275            PHX
83C9: 20 47 85 >276            JSR   SAVCURRM
83CC: 68       >277            PLA
83CD: CD A1 99 >278            CMP   PFINDIC
83D0: F0 03    >279            BEQ   :11
83D2: 20 87 82 >280            JSR   SETINITX
83D5: 20 3C 85 >281  :11       JSR   RSTALTM
83D8: A0 03    >282            LDY   #VSRTNAM
83DA: 20 10 83 >283            JSR   SCNDTIM
83DD: A0 09    >284            LDY   #VENT1NAM
83DF: 20 10 83 >285            JSR   SCNDTIM
83E2: B2 06    >287            LDA   (AUXPTR)
83E4: 29 40    >292            AND   #64
83E6: F0 05    >293            BEQ   :10
83E8: A0 0F    >294            LDY   #VENT2NAM
83EA: 20 10 83 >295            JSR   SCNDTIM
83ED: A0 0C    >296  :10       LDY   #VENT1IT
83EF: 20 3E 83 >297            JSR   COMLET2
83F2: B2 06    >299            LDA   (AUXPTR)
83F4: 29 40    >304            AND   #64
83F6: F0 08    >305            BEQ   :12
83F8: 20 53 EB >306            JSR   MOVFA
83FB: A0 12    >307            LDY   #VENT2IT
83FD: 20 3E 83 >308            JSR   COMLET2
               >309  :12       DO    KOPT16
8400: A9 84    >312            LDA   #>RETOUR-1
8402: 48       >313            PHA
8403: A9 DA    >314            LDA   #RETOUR-1
8405: 48       >315            PHA
8406: 80 12    >317            BRA   COMMONG
```

```
                >318
8408: E0 0A     >319  :3        CPX   #10
840A: B0 0B     >320            BCS   :4
840C: A0 01     >321            LDY   #ADRUSR
840E: B1 06     >322            LDA   (AUXPTR),Y
8410: D0 01     >323            BNE   *+3
8412: CA        >324            DEX
8413: 3A        >326            DEC
8414: DA        >332            PHX
8415: 48        >333            PHA
8416: 60        >339            RTS
8417: 4C 0A 00 >340  :4        JMP   VECTUSR
                >341
841A: A0 0D     >342  COMMONG   LDY   #FINOF-SVOFST-1
841C: BE 4C 97 >343  ]LOOP     LDX   SVOFST,Y
841F: B5 00     >344            LDA   0,X
8421: 99 5A 97 >345            STA   SVAREA,Y
8424: 88        >346            DEY
8425: 10 F5     >347            BPL   ]LOOP
8427: 64 F2     >349            STZ   TRCFLG
                >354  * This is the critical code segment
8429: A5 B9     >359            LDA   TXTPTR+1
842B: 48        >360            PHA
842C: A5 B8     >361            LDA   TXTPTR
842E: 48        >362            PHA
842F: A5 76     >363            LDA   CURLIN+1
8431: 48        >364            PHA
8432: A5 75     >365            LDA   CURLIN
8434: 48        >366            PHA
8435: A9 B0     >368            LDA   #TOKGOSUB
8437: 48        >369            PHA
8438: A0 01     >370            LDY   #ADRUSR
843A: B1 06     >371            LDA   (AUXPTR),Y
843C: 85 B8     >372            STA   TXTPTR
843E: C8        >373            INY
843F: B1 06     >374            LDA   (AUXPTR),Y
8441: 85 B9     >375            STA   TXTPTR+1
8443: 4C D2 D7 >376            JMP   NEWSTT
                >377
8446: 20 F4 7B >378  RDEFUSR   JSR   RST102
8449: 90 05     >379            BCC   :1           Branch if digit
844B: A9 0A     >380            LDA   #10
844D: 48        >381            PHA
844E: D0 06     >382            BNE   :3           Always
8450: E9 2F     >383  :1        SBC   #´0´-1       ASCII digit to binary
8452: 48        >384            PHA
8453: 20 EC 7B >385            JSR   RST100
8456: A9 D0     >386  :3        LDA   #TOKEQUAL
8458: 20 62 82 >387            JSR   NSYNCHR
845B: 20 67 DD >388            JSR   FRMNUM
845E: 20 52 E7 >389            JSR   GETADR
8461: FA        >390            PLX
8462: DA        >392            PHX
8463: 20 95 82 >396            JSR   COMPOFST
8466: 68        >397            PLA
8467: 48        >398            PHA
8468: C9 0A     >399            CMP   #10          Set carry flag
```

```
                * If LINNUM high byte is zero, then must be the mode
846A: A5 50               LDA   LINNUM
846C: A6 51               LDX   LINNUM+1
846E: F0 11               BEQ   :5
8470: 20 2B 83            JSR   HNDLEADR
8473: 68                  PLA
8474: A9 00               LDA   #0
8476: 92 06               STA   (AUXPTR)
8478: 20 F4 7B  ]LOOP     JSR   RST102
847B: D0 01               BNE   *+3
847D: 60                  RTS
847E: 4C C9 DE  ]ERR      JMP   SYNERR
                * DEFUSR=<mode>,<otherparms>
8481: 92 06     :5        STA   (AUXPTR)
8483: A8                  TAY
8484: 30 24               BMI   :6          Procedural function
8486: 29 3F               AND   #$3F
8488: D0 F4               BNE   ]ERR
848A: 20 74 8B            JSR   NCHKCOM
848D: 20 67 DD            JSR   FRMNUM
8490: 20 52 E7            JSR   GETADR
8493: FA                  PLX
8494: E0 0A               CPX   #10
8496: 08                  PHP
8497: 20 95 82            JSR   COMPOFST
849A: 28                  PLP
849B: A5 50               LDA   LINNUM
849D: A6 51               LDX   LINNUM+1
849F: 4C 2B 83  ]LOOP     JMP   HNDLEADR
84A2: 4C 7C D9  ]ERR      JMP   ULERR
84A5: A2 28     ]ERR1     LDX   #40
84A7: 4C D0 92            JMP   NERRH
84AA: 48        :6        PHA
84AB: AD A0 99            LDA   ISPFACT
84AE: D0 F5               BNE   ]ERR1
84B0: A9 03               LDA   #VSRTNAM
84B2: 20 B4 82            JSR   FRSTIM
84B5: A9 09               LDA   #VENT1NAM
84B7: 20 B4 82            JSR   FRSTIM
84BA: 68                  PLA
84BB: 29 40               AND   #64
84BD: F0 05               BEQ   :7
84BF: A9 0F               LDA   #VENT2NAM
84C1: 20 B4 82            JSR   FRSTIM
84C4: 68        :7        PLA               ;Do not care routine idx
84C5: 20 74 8B            JSR   NCHKCOM
84C8: 20 0C DA            JSR   LINGET
84CB: 20 1A D6            JSR   FNDLIN
84CE: 90 D2               BCC   ]ERR
84D0: A6 9C               LDX   LOWTR+1
84D2: A5 9B               LDA   LOWTR
84D4: D0 01               BNE   *+3
84D6: CA                  DEX
84D7: 3A                  DEC
84D8: 18                  CLC
84D9: 90 C4               BCC   ]LOOP       Always
```

```
84DB: 20 FB 84 >471  RETOUR   JSR   COMREST
84DE: AE A2 99 >472           LDX   PFINDX
84E1: DA       >473           PHX
84E2: 20 95 82 >474           JSR   COMPOFST
84E5: 20 09 85 >475           JSR   COLLECTR
84E8: FA       >476           PLX
84E9: B2 06    >478           LDA   (AUXPTR)
84EB: 9C A0 99 >479           STZ   ISPFACT
84EE: 4A       >485           LSR
84EF: 90 09    >486           BCC   :0
84F1: 8E A1 99 >487           STX   PFINDIC
84F4: 20 52 85 >488           JSR   SAVALTM
84F7: 4C 31 85 >489           JMP   RSTCURRM
84FA: 60       >490  :0       RTS
               >491
84FB: A0 0D    >492  COMREST  LDY   #FINOF-SVOFST-1
84FD: BE 4C 97 >493  ]LOOP    LDX   SVOFST,Y
8500: B9 5A 97 >494           LDA   SVAREA,Y
8503: 95 00    >495           STA   0,X
8505: 88       >496           DEY
8506: 10 F5    >497           BPL   ]LOOP
8508: 60       >498           RTS
               >499
8509: A0 06    >500  COLLECTR LDY   #VSRTIT
850B: B1 06    >501           LDA   (AUXPTR),Y
850D: 0A       >502           ASL
850E: A0 07    >503           LDY   #VSRTPTR
8510: B1 06    >504           LDA   (AUXPTR),Y
8512: AA       >505           TAX
8513: C8       >506           INY
8514: B1 06    >507           LDA   (AUXPTR),Y
8516: A8       >508           TAY
8517: 8A       >509           TXA
8518: B0 07    >510           BCS   :0          Branch iif integer output var.
851A: 64 11    >512           STZ   VALTYP
851C: 64 12    >513           STZ   INTTYP
851E: 4C F9 EA >519           JMP   MOVFM
8521: 84 84    >520  :0       STY   VARPNT+1
8523: 85 83    >521           STA   VARPNT
8525: B2 83    >523           LDA   (VARPNT)
8527: A0 01    >524           LDY   #1
8529: AA       >530           TAX
852A: B1 83    >531           LDA   (VARPNT),Y
852C: A8       >532           TAY
852D: 8A       >533           TXA
852E: 4C F2 E2 >534           JMP   GIVAYF
               >535
8531: A2 0B    >536  RSTCURRM LDX   #12-1
8533: BD 68 97 >537  ]LOOP    LDA   SVCURRM,X
8536: 95 69    >538           STA   $69,X
8538: CA       >539           DEX
8539: 10 F8    >540           BPL   ]LOOP
853B: 60       >541           RTS
               >542
853C: A2 0B    >543  RSTALTM  LDX   #12-1
853E: BD 74 97 >544  ]LOOP    LDA   SVALTNM,X
8541: 95 69    >545           STA   $69,X
```

```
8543: CA       >546           DEX
8544: 10 F8    >547           BPL   ]LOOP
8546: 60       >548           RTS
               >549
8547: A2 0B    >550  SAVCURRM LDX   #12-1
8549: B5 69    >551  ]LOOP    LDA   $69,X
854B: 9D 68 97 >552           STA   SVCURRM,X
854E: CA       >553           DEX
854F: 10 F8    >554           BPL   ]LOOP
8551: 60       >555           RTS
               >556
8552: A2 0B    >557  SAVALTM  LDX   #12-1
8554: B5 69    >558  ]LOOP    LDA   $69,X
8556: 9D 74 97 >559           STA   SVALTNM,X
8559: CA       >560           DEX
855A: 10 F8    >561           BPL   ]LOOP
855C: 60       >562           RTS
                570           PUT   PEERDEF
               >1    * Nouvelle routine de traitement du DEF..
855D: 4C 46 84 >2    ]LOOP    JMP   RDEFUSR
8560: A4 B9    >3    RDEF     LDY   TXTPTR+1
8562: A5 B8    >4             LDA   TXTPTR
8564: D0 01    >11            BNE   *+3
8566: 88       >12            DEY
8567: 3A       >13            DEC
8568: A2 01    >15            LDX   #1
856A: 20 16 87 >16            JSR   RECON       Check which DEF pattern
856D: D0 03    >17            BNE   :1          None detected
856F: 4C 13 E3 >18            JMP   $E313
8572: 88       >19   :1       DEY
8573: 20 98 D9 >20            JSR   ADDON
8576: A6 BD    >21            LDX   IDMOCL
8578: E0 09    >22            CPX   #OFFUSR-TOFFST Is it DEFUSR?
857A: F0 E1    >23            BEQ   ]LOOP
857C: BD 7A 9B >24            LDA   MOTIF-NOPER-6,X Must be DEF(INT/STR/SNG)
               >25   * Below is the common code for all three new instructions
857F: 64 C0    >30            STZ   LETINF
8581: 85 C1    >32            STA   TYPMOD
8583: 20 F1 85 >33            JSR   DECTPTR     Decrement TXTPTR
8586: 20 BC 85 >34   ]LOOP    JSR   :LBS00      Bump ptr. to 1st letter of next v
ar
8589: 20 5C 82 >35            JSR   MISLETC     Must be alphabetic
858C: 85 C0    >36            STA   LETINF
858E: 20 BC 85 >37            JSR   :LBS00      Exit if no further variable
8591: C9 C9    >38            CMP   #TOKMINUS   means a letter range
8593: F0 0B    >39            BEQ   :2
8595: C9 2C    >40            CMP   #´,´        Character must be either ´,´
8597: D0 34    >41            BNE   GSNERR3      or ´-´
8599: A6 C0    >42            LDX   LETINF      Process current letter
859B: 20 C7 85 >43            JSR   RDEFSUB
859E: 10 E6    >44            BPL   ]LOOP       Always
85A0: 20 EC 7B >45   :2       JSR   RST100      Range:get the upper range let.
85A3: 20 5C 82 >46            JSR   MISLETC
85A6: C5 C0    >47            CMP   LETINF      Must not < 1st letter
85A8: 90 23    >48            BCC   GSNERR3
85AA: AA       >49            TAX               ;Into X for processing
85AB: 20 C7 85 >50   ]JLOOP   JSR   RDEFSUB     process current letter within
```

```
85AE: CA       >51               DEX
85AF: E4 C0    >52               CPX   LETINF     Loop until 1st letter
85B1: B0 F8    >53               BCS   ]JLOOP
85B3: 20 BC 85 >54               JSR   :LBS00
85B6: C9 2C    >55               CMP   #´,´
85B8: D0 13    >56               BNE   GSNERR3
85BA: F0 CA    >57               BEQ   ]LOOP      Always
85BC: 20 EC 7B >58   :LBS00      JSR   RST100
85BF: D0 0B    >59               BNE   R          Do not return if EOI
85C1: 68       >60               PLA
85C2: 68       >61               PLA
85C3: A6 C0    >62   :FIN        LDX   LETINF
85C5: F0 06    >63               BEQ   GSNERR3    Whaever args, process last letter
85C7: A5 C1    >64   RDEFSUB     LDA   TYPMOD
85C9: 9D 61 9B >65               STA   TYPLET-´A´,X
85CC: 60       >66   R           RTS
85CD: 4C C9 DE >67   GSNERR3     JMP   SYNERR
               >68
               >125
85D0: 20 EC 7B >142  ROUT1Y      JSR   RST100
85D3: 48       >143              PHA
85D4: BD 90 9B >144  ROUT1X      LDA   TVNORA,X
85D7: 04 81    >145              TSB   VARNAM
85D9: BD 94 9B >146              LDA   TVN1ORA,X
85DC: 04 82    >147              TSB   VARNAM+1
85DE: 20 53 E0 >148              JSR   $E053      Attention, il faudra chg.
85E1: 68       >149              PLA
85E2: 60       >150              RTS
               >151
               >179
                571
85E3: BD 61 9B  572  XFRMMOT1 LDA   TYPLET-´A´,X
                573  XFROMMOT
                575  * X=0 for ´%´, 1 for ´$´ and 2 for ´!´, 3 for ´.´
85E6: A2 03     576              LDX   #TITVAL-MOTIF-1
85E8: DD 84 9B  580  ]LOOP       CMP   MOTIF,X
85EB: F0 03     581              BEQ   :0
85ED: CA        582              DEX
85EE: 10 F8     583              BPL   ]LOOP
85F0: 60        584  :0          RTS
                585
                586  * Decrement TXTPTR
85F1: A5 B8     587  DECTPTR     LDA   TXTPTR
85F3: D0 02     588              BNE   :0
85F5: C6 B9     589              DEC   TXTPTR+1
85F7: C6 B8     590  :0          DEC   TXTPTR
85F9: 60        591              RTS
                592
                593  * Subroutine to patch CHRGET/CHRGOT in page zero
85FA: A9 4C     594  SETUPB      LDA   #$4C       JMP absolute
85FC: 85 B1     595              STA   $B1
85FE: 85 BA     596              STA   $BA
8600: A9 CB     597              LDA   #DEBUTGET
8602: 85 B2     597              STA   $B2
8604: A9 7B     597              LDA   #>DEBUTGET
8606: 85 B3     597              STA   $B2+1
8608: A9 1A     598              LDA   #DEBUTGOT
```

```
860A: 85 BB     598             STA   $BB
860C: A9 7C     598             LDA   #>DEBUTGOT
860E: 85 BC     598             STA   $BB+1
8610: 60        599             RTS
                600
                601   SETUPD    STID  BANCLD;$9D72
8611: A9 1C     601             LDA   #BANCLD
8613: 8D 72 9D  601             STA   $9D72
8616: A9 86     601             LDA   #>BANCLD
8618: 8D 73 9D  601             STA   $9D72+1
861B: 60        602             RTS
                603
                604   * Subr. called upon a BASIC cold boot (FP DOS command)
861C: A2 FF     605   BANCLD    LDX   #$FF
861E: 86 76     606             STX   $76
8620: A2 FB     607             LDX   #$FB
8622: 9A        608             TXS
8623: A9 28     609             LDA   #$28
8625: A0 F1     610             LDY   #$F1
8627: 85 01     611             STA   1
8629: 84 02     612             STY   2
862B: 85 04     613             STA   4
862D: 84 05     614             STY   5
862F: 20 73 F2  615             JSR   $F273
8632: A9 4C     616             LDA   #$4C        JMP absolute
8634: 85 00     617             STA   0
8636: 85 03     618             STA   3
8638: 85 90     619             STA   $90
863A: 85 0A     620             STA   $A
863C: A9 99     621             LDA   #$99
863E: A0 E1     622             LDY   #$E1
8640: 85 0B     623             STA   $B
8642: 84 0C     624             STY   $C
8644: 20 FA 85  625             JSR   SETUPB      Install CHRGET/CHRGOT patch in pa
ge zero
8647: 4C 5C F1  626             JMP   $F15C       End of initialization in ROM
                627
                628   * Do the DOS init
                629   NOUVIN    STID  $E000;$9D72
864A: A9 00     629             LDA   #$E000
864C: 8D 72 9D  629             STA   $9D72
864F: A9 E0     629             LDA   #>$E000
8651: 8D 73 9D  629             STA   $9D72+1
8654: A9 4C     630             LDA   #$4C        JMP absolute
8656: 8D C8 A2  631             STA   $A2C8
8659: A9 0B     632             LDA   #$B
865B: 20 AA A2  633             JSR   $A2AA
865E: A9 20     634             LDA   #$20
8660: 8D C8 A2  635             STA   $A2C8
8663: A5 45     636             LDA   OPRND+1
8665: D0 06     637             BNE   :4          No error during DoClose
8667: 20 11 86  638             JSR   SETUPD      Reinstall Peersoft
866A: 4C C8 A6  639             JMP   $A6C8        before exiting
866D: A2 60     640   :4        LDX   #$60
866F: 8E E7 A2  641             STX   $A2E7
8672: 20 D2 A2  642             JSR   $A2D2       Copy file manager parmlist
8675: A9 4C     643             LDA   #$4C        JMP absolute
```

```
8677: 8D E7 A2           STA   $A2E7
867A: AD 00 9D           LDA   DBUFP
867D: 8D 95 86           STA   E06+1
8680: AD 01 9D           LDA   DBUFP+1
8683: 8D 9A 86           STA   E06+6
8686: A9 D3              LDA   #$9CD3
8688: 8D 00 9D           STA   DBUFP
868B: A9 9C              LDA   #>$9CD3
868D: 8D 01 9D           STA   DBUFP+1
8690: 20 06 AB           JSR   $AB06          File manager main entry (INIT)
8693: 08                 PHP                  ;Save status
                E06      STID  0;DBUFP        Reinstall Peersoft DOS features
8694: A9 00              LDA   #0
8696: 8D 00 9D           STA   DBUFP
8699: A9 00              LDA   #>0
869B: 8D 01 9D           STA   DBUFP+1
869E: 20 11 86           JSR   SETUPD
86A1: 28                 PLP
86A2: 20 EB A6           JSR   $A6EB          process possible error after FM c
all
86A5: 4C 97 A3           JMP   $A397          Goto SAVE (HELLO) command handler

86A8: 4C C9 DE  GSNERR   JMP   SYNERR
                RFOR
86AB: 64 14              STZ   SUBFLG
86AD: 20 96 7E           JSR   NPTRGET
86B0: 85 85              STA   FORPNT
86B2: 84 86              STY   FORPNT+1
                * For the time being, array variables are forbidden
86B4: C5 6B              CMP   ARYTAB
86B6: 98                 TYA
86B7: E5 6C              SBC   ARYTAB+1
86B9: B0 ED              BCS   GSNERR
86BB: A0 01              LDY   #1
86BD: B1 9B              LDA   (LOWTR),Y
86BF: AA                 TAX
86C0: 52 9B              EOR   (LOWTR)
86C2: 30 E4              BMI   GSNERR
86C4: DA                 PHX
86C5: 20 44 7C           JSR   RLET1
86C8: 68                 PLA
86C9: 85 C0              STA   GFLAG
86CB: 20 65 D3           JSR   $D365
86CE: D0 05              BNE   :0
86D0: 8A                 TXA                  ;Stackframe pointer in X
86D1: 69 0F              ADC   #$0F           Carry already set, add 16
86D3: AA                 TAX                  ;+2 bytes (lines below)
86D4: 9A                 TXS                  ;= 18 bytes
86D5: 68        :0       PLA
86D6: 68                 PLA
86D7: A9 09              LDA   #9             Check for 18 bytes
86D9: 20 D6 D3           JSR   CHKMEM          available on stack
86DC: A5 C0              LDA   GFLAG
86DE: 30 03              BMI   :1
86E0: 4C 7E D7           JMP   $D77E
86E3: 20 A3 D9  :1       JSR   DATAN          Prochain separateur (offset Y)
86E6: 18                 CLC
```

```
86E7: 98                 TYA
86E8: 65 B8              ADC   TXTPTR
86EA: 48                 PHA
86EB: A5 B9              LDA   TXTPTR+1
86ED: 69 00              ADC   #0
86EF: 48                 PHA
86F0: A9 C1              LDA   #TOKTO
86F2: 20 62 82           JSR   NSYNCHR
86F5: A5 76              LDA   CURLIN+1
86F7: 48                 PHA
86F8: A5 75              LDA   CURLIN
86FA: 48                 PHA
86FB: 20 09 87           JSR   LBS03
86FE: A9 7C      STP1     LDA   #STEP
8700: A0 89              LDY   #>STEP
8702: 85 5E              STA   INDEX
8704: 84 5F              STY   INDEX+1
8706: 4C 23 DE           JMP   FRMSTCK3+3 Returns with a JMP (INDEX)

8709: 20 67 DD  LBS03    JSR   FRMNUM
870C: 20 72 EB           JSR   $EB72      Round FAC
870F: 4C 0C E1           JMP   AYINT      Result in FACLO,FACMO

                * 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)
8712: A5 B8     RECON1   LDA   TXTPTR
8714: A4 B9              LDY   TXTPTR+1
8716: 85 06     RECON    STA   AUXPTR
8718: 84 07              STY   AUXPTR+1
871A: BD 72 9B  RECON2   LDA   TIDMOCL,X
871D: 85 BD              STA   IDMOCL
871F: BD 78 9B           LDA   TOFFIN,X
8722: 8D 4C 9B           STA   IFDEF
8725: BD 7E 9B           LDA   TOFFIN2,X
8728: 8D 3D 9B           STA   IFIIF
872B: E6 BD     :1       INC   IDMOCL
872D: A4 BD              LDY   IDMOCL
872F: BE 63 9B           LDX   TOFFST,Y
8732: 86 C2              STX   OFFSET
8734: A0 00              LDY   #0
8736: BD 2B 9B  ]LOOP    LDA   TMOCL,X
8739: F0 0C              BEQ   :4         Keyword found: exit
873B: C9 FF              CMP   #$FF       End of table?
873D: F0 08              BEQ   :4         Yes: no keyword found
873F: D1 06              CMP   (AUXPTR),Y Current character match?
8741: D0 E8              BNE   :1         no: try next keyword from table
8743: E8                 INX              ;Next char. from current keyword
```

```
8744: C8        764           INY
8745: D0 EF     765           BNE   ]LOOP
                766
                767  :4       DO    KOPT-K65C02
8747: 1A        771           INC
8748: 60        773  RETURN   RTS
                774
                775           PUT   PEERLIST
8749: 90 0A    >1    STDLIS   BCC   STRTRNG
               >2
874B: F0 08    >3             BEQ   STRTRNG
874D: C9 C9    >4             CMP   #TOKMINUS
874F: F0 04    >5             BEQ   STRTRNG
8751: C9 2C    >6             CMP   #´,´
8753: D0 F3    >7             BNE   RETURN
               >8
8755: 20 24 95 >9    STRTRNG  JSR   DECOMPILE
8758: 20 0C DA >10            JSR   LINGET
875B: 20 1A D6 >11            JSR   FNDLIN
875E: 20 F4 7B >12            JSR   RST102
8761: F0 10    >13            BEQ   MAINLIST
8763: C9 C9    >14            CMP   #TOKMINUS
8765: F0 04    >15            BEQ   ENDRNG
8767: C9 2C    >16            CMP   #´,´
8769: D0 DD    >17            BNE   RETURN
               >18
876B: 20 EC 7B >19   ENDRNG   JSR   RST100
876E: 20 0C DA >20            JSR   LINGET
8771: D0 D5    >21            BNE   RETURN
               >22
8773: 68       >23   MAINLIST PLA
8774: 68       >24            PLA
8775: A5 50    >25            LDA   LINNUM     In case no second line given,
8777: 05 51    >26            ORA   LINNUM+1    let it be 65535
8779: D0 04    >27            BNE   NXLST
877B: C6 50    >28            DEC   LINNUM
877D: C6 51    >29            DEC   LINNUM+1
               >30
877F: A0 01    >31   NXLST    LDY   #1
8781: B1 9B    >32            LDA   (LOWTR),Y
8783: F0 6B    >33            BEQ   LISTED     End of program found
8785: 20 58 D8 >34            JSR   ISCNTC     Check for Ctrl-C keystroke
8788: 20 FB DA >35            JSR   CRDO
878B: C8       >36            INY
878C: B1 9B    >37            LDA   (LOWTR),Y  Line number in X,A
878E: AA       >38            TAX
878F: C8       >39            INY
8790: B1 9B    >40            LDA   (LOWTR),Y
8792: C5 51    >41            CMP   LINNUM+1   Beyond last line number?
8794: D0 04    >42            BNE   LSTD?
8796: E4 50    >43            CPX   LINNUM
8798: F0 02    >44            BEQ   LST1LIN
879A: B0 54    >45   LSTD?    BCS   LISTED     Yes
               >46
879C: 84 85    >47   LST1LIN  STY   $85
879E: 64 BE    >55            STZ   MODREM
87A0: 64 BF    >56            STZ   MODDAT
```

```
87A2: 64 C0    >57             STZ   GFLAG
87A4: 64 C1    >58             STZ   DEFFLG
87A6: 20 F6 87 >60             JSR   VLINPRT    Print line #
87A9: A9 20    >61   ]JLOOP    LDA   #32        Print space after line number
87AB: A4 85    >62             LDY   $85
87AD: 2C       >63             HEX   2C
87AE: A9 2D    >64   L088      LDA   #´-´
87B0: C9 22    >65   L08       CMP   #´"´       Is it ´"´?
87B2: D0 08    >66             BNE   :9
87B4: A5 C0    >67             LDA   GFLAG
87B6: 49 FF    >68             EOR   #$FF
87B8: 85 C0    >69             STA   GFLAG
87BA: A9 22    >70             LDA   #´"´
               >71   * Now we test for an EOI
87BC: 24 BE    >72   :9        BIT   MODREM     If a REM has been scanned in this
 line
87BE: 30 0C    >73             BMI   SENDCHR
87C0: 24 C0    >74             BIT   GFLAG      Are we within a string litteral?
87C2: 30 08    >75             BMI   SENDCHR    Same output as for a REM
87C4: C9 3A    >76             CMP   #´:´       Current char is EOI?
87C6: D0 04    >77             BNE   SENDCHR
87C8: 85 BF    >78             STA   MODDAT     MODDAT b7 forced to zero
87CA: 85 C1    >79             STA   DEFFLG     DEFFLG b7 forced to zero
87CC: 20 5C DB >80   SENDCHR   JSR   OUTDO      Print current char
87CF: A5 24    >81             LDA   CH
87D1: C9 21    >82             CMP   #33        Have we reached "right" edge of s
creen?
87D3: 90 07    >83             BCC   NCR        No
87D5: 20 FB DA >84             JSR   CRDO       Yes: print CR for next line
87D8: A9 05    >85             LDA   #5
87DA: 85 24    >86             STA   CH
               >87   * Next character from line
87DC: C8       >88   NCR       INY
87DD: B1 9B    >89             LDA   (LOWTR),Y
87DF: D0 18    >90             BNE   TOKEN?     Not end of line
87E1: 85 C1    >91             STA   DEFFLG
87E3: B2 9B    >98             LDA   (LOWTR)    Update next line pointer
87E5: AA       >99             TAX
87E6: A0 01    >100            LDY   #1
87E8: B1 9B    >102            LDA   (LOWTR),Y
87EA: 86 9B    >103            STX   LOWTR
87EC: 85 9C    >104            STA   LOWTR+1
87EE: D0 8F    >105            BNE   NXLST      Branch if not at program´s end
               >106
87F0: 20 FB DA >107  LISTED    JSR   CRDO
87F3: 4C D2 D7 >108            JMP   NEWSTT
87F6: 6C FA D6 >109  VLINPRT   JMP   ($D6FA)
87F9: AA       >110  TOKEN?    TAX              ;Character in X
87FA: A5 BE    >111            LDA   MODREM     Is litteral mode active?
87FC: 05 BF    >112            ORA   MODDAT
87FE: 05 C0    >113            ORA   GFLAG
8800: 0A       >114            ASL
8801: 8A       >115            TXA
8802: B0 AC    >116            BCS   L08        Yes
8804: 84 B5    >117            STY   YSAV
8806: 98       >118            TYA              ;Compute Y, A = LOWTR + Y
8807: A4 9C    >119            LDY   LOWTR+1
```

```
8809: 65 9B    >120             ADC   LOWTR      Carry already clear
880B: 90 01    >121             BCC   :14
880D: C8       >122             INY
880E: A2 00    >123  :14        LDX   #0
8810: 20 16 87 >124             JSR   RECON      New BASIC keyword?
8813: D0 33    >125             BNE   :23        Yes
               >126
8815: A4 B5    >127             LDY   YSAV       Y = offset within line
8817: B1 9B    >128             LDA   (LOWTR),Y  Current character
8819: 10 95    >129             BPL   L08        Not a token
881B: 24 C1    >130             BIT   DEFFLG
881D: 10 04    >131             BPL   :18
881F: C9 C9    >132             CMP   #TOKMINUS
8821: F0 8B    >133             BEQ   L088
8823: C9 B2    >134  :18        CMP   #TOKREM    REM token?
8825: D0 02    >135             BNE   :15
8827: 66 BE    >136             ROR   MODREM     bit 7 to 1 in MODREM
8829: C9 83    >137  :15        CMP   #TOKDATA   DATA token?
882B: D0 02    >138             BNE   :16
882D: 66 BF    >139             ROR   MODDAT     bit 7 to 1 in MODDAT
882F: 48       >140  :16        PHA
8830: 20 57 DB >141             JSR   OUTSPC
8833: 68       >142             PLA
8834: 48       >143             PHA
8835: 20 96 88 >144             JSR   LTOKEN     Print Applesoft token
8838: 68       >145             PLA
8839: C9 D5    >146             CMP   #TOKUSR
883B: 20 86 88 >147             JSR   COMLISO
883E: B0 05    >148             BCS   :17
8840: 84 85    >149             STY   $85
8842: 20 5C DB >150             JSR   OUTDO
8845: 4C A9 87 >151  :17        JMP   ]JLOOP
               >152  * LIST a new BASIC statement
8848: 88       >153  :23        DEY
8849: A5 BD    >154             LDA   IDMOCL
884B: C9 0A    >155             CMP   #OFFDEF-TOFFST
884D: 90 03    >156             BCC   :39
884F: 66 C1    >157             ROR   DEFFLG
8851: 18       >158             CLC
8852: 98       >159  :39        TYA
8853: 65 B5    >160             ADC   YSAV
8855: 85 B5    >161             STA   YSAV
8857: 20 57 DB >162             JSR   OUTSPC
885A: A6 C2    >163             LDX   OFFSET     Get offset from new keyword table
885C: BD 2B 9B >164  ]LOOP      LDA   TMOCL,X
885F: F0 11    >165             BEQ   :29        End of keyword
8861: 30 05    >166             BMI   :27        Applesoft token: print it
8863: 20 5C DB >167             JSR   OUTDO      Normal text to output
8866: D0 07    >168             BNE   :28        Always
8868: 86 B4    >169  :27        STX   XSAV       Save offset
886A: 20 96 88 >170             JSR   LTOKEN     Print Applesoft token
886D: A6 B4    >171             LDX   XSAV
886F: E8       >172  :28        INX
8870: D0 EA    >173             BNE   ]LOOP      Always
8872: A5 BD    >174  :29        LDA   IDMOCL
8874: C9 09    >175             CMP   #OFFUSR-TOFFST
8876: 20 86 88 >176             JSR   COMLISO
```

```
8879: B0 03                BCS   :30
887B: 20 5C DB             JSR   OUTDO
887E: 20 57 DB   :30       JSR   OUTSPC
8881: A4 B5      :31       LDY   YSAV
8883: 4C DC 87             JMP   NCR

8886: 38         COMLISO   SEC
8887: D0 0C                BNE   :0
8889: A4 B5                LDY   YSAV
888B: C8                   INY
888C: B1 9B                LDA   (LOWTR),Y
888E: 20 FC 7B             JSR   COMRSTC
8891: B0 02                BCS   :0
8893: 84 B5                STY   YSAV
8895: 60         :0        RTS

                 * Print Applesoft token
8896: 38         LTOKEN    SEC
8897: E9 7F                SBC   #$7F
8899: AA                   TAX              ;Index in X reg
889A: 84 85                STY   $85
889C: A0 D0                LDY   #TOKTABL-256
889E: 84 9D                STY   FAC
                 * Line below is a substitute for LDY #>TOKTABL-256
88A0: 88                   DEY
88A1: 84 9E                STY   FAC+1
88A3: A0 FF                LDY   #$FF
88A5: CA         :1        DEX
88A6: F0 07                BEQ   :3
88A8: 20 2C D7   ]LOOP     JSR   $D72C
88AB: 10 FB                BPL   ]LOOP
88AD: 30 F6                BMI   :1
88AF: 20 2C D7   :3        JSR   $D72C
88B2: 30 05                BMI   :4
88B4: 20 5C DB             JSR   OUTDO
88B7: D0 F6                BNE   :3
88B9: A4 85      :4        LDY   $85
88BB: 4C 5C DB             JMP   OUTDO

88BE: D0 07      RRETURN   BNE   :0
88C0: A9 FF                LDA   #$FF
88C2: 85 86                STA   FORPNT+1
88C4: 4C 71 D9             JMP   $D971
88C7: 60         :0        RTS

88C8: A9 AB      RONERR    LDA   #TOKGOTO
88CA: 20 62 82             JSR   NSYNCHR
88CD: A5 B8                LDA   TXTPTR
88CF: 85 F4                STA   TXTPSV
88D1: A5 B9                LDA   TXTPTR+1
88D3: 85 F5                STA   TXTPSV+1
88D5: 38                   SEC
88D6: 66 D8                ROR   ERRFLG
88D8: A5 75                LDA   CURLIN
88DA: 85 F6                STA   CURLSV
88DC: A5 76                LDA   CURLIN+1
88DE: 85 F7                STA   CURLSV+1
```

```
88E0: 4C 95 D9                JMP   DATA

88E3: 4C 0B DD  ]LOOP         JMP   $DD0B         NEXT WITHOUT FOR error
88E6: D0 04     RNEXT         BNE   NEXT1
88E8: A0 00                   LDY   #0
88EA: F0 03                   BEQ   *+5
88EC: 20 94 7E  NEXT1         JSR   NPTRGTX
88EF: 85 85                   STA   FORPNT
88F1: 84 86                   STY   FORPNT+1
88F3: 20 65 D3                JSR   $D365
88F6: D0 EB                   BNE   ]LOOP
88F8: 9A                      TXS
88F9: E8                      INX
88FA: E8                      INX
88FB: E8                      INX
88FC: E8                      INX
88FD: 8A                      TXA                 ;Base address of STEP value
88FE: E8                      INX
88FF: E8                      INX
8900: E8                      INX
8901: E8                      INX
8902: E8                      INX
8903: E8                      INX
8904: 86 60                   STX   DEST          Base adress of TO value
8906: A8                      TAY
8907: BA                      TSX
8908: BD 09 01                LDA   $0109,X
890B: 85 C0                   STA   GFLAG
890D: 0A                      ASL
890E: 90 08                   BCC   :1
8910: 10 08                   BPL   :2
8912: 98        :0            TYA
8913: A6 60                   LDX   DEST
8915: 4C 1D DD                JMP   $DD1D         FP var: classic mechanic
8918: 10 F8     :1            BPL   :0
891A: 20 6A 89  :2            JSR   LBS05         Step value into $A0, $A1
891D: A9 00                   LDA   #0
891F: 20 A9 7C                JSR   HNDLEIY       Current value in FORPNT
8922: A5 C0                   LDA   GFLAG         Retrofit to get normal sign
8924: 49 80                   EOR   #$80           value
8926: 85 C0                   STA   GFLAG
8928: 38                      SEC
8929: A4 60                   LDY   DEST
892B: 20 6E 89                JSR   LBS051
                * A: -1 iif endvalue > current value
                * A: 0 iif endvalue = current value
                * A: 1 iif endvalue < current value
892E: A2 FF                   LDX   #-1
8930: A0 01                   LDY   #1
8932: 38                      SEC
                * A and $A1 same content (result from previous call to
                * LBS05).
8933: F1 85                   SBC   (FORPNT),Y
8935: D0 01                   BNE   :C0
8937: E8                      INX
8938: A5 A0     :C0           LDA   $A0
893A: F2 85                   SBC   (FORPNT)
```

```
893C: 70 0C                BVS   :C1
893E: 30 07                BMI   :LT
8940: D0 02     :C2        BNE   :C20
8942: 8A                   TXA                 ;A=0 if both bytes equal
8943: 2C                   HEX   2C            next two bytes
8944: A9 FF     :C20       LDA   #-1
8946: 2C                   HEX   2C
8947: A9 01     :LT        LDA   #1
8949: 2C                   HEX   2C            Skip next two bytes
894A: 10 F4     :C1        BPL   :C2
894C: BA                   TSX
894D: 38                   SEC
894E: E5 C0                SBC   GFLAG
8950: F0 03                BEQ   :3
8952: 4C 3E DD             JMP   $DD3E         Processing next loop iteration
8955: 8A        :3         TXA                 ;Arithmetic of frame pointer
8956: 69 11                ADC   #17           Carry set so add 18
8958: AA                   TAX
8959: 9A                   TXS
895A: 20 F4 7B             JSR   RST102
895D: C9 2C                CMP   #´,´
895F: D0 06                BNE   :4
8961: 20 EC 7B             JSR   RST100
8964: 20 EC 88             JSR   NEXT1         Does not return
8967: 4C D2 D7  :4         JMP   NEWSTT

896A: A9 01     LBS05      LDA   #1
896C: 85 5F                STA   INDEX+1
896E: 84 5E     LBS051     STY   INDEX
8970: A0 03                LDY   #3
8972: B1 5E                LDA   (INDEX),Y
8974: 85 A0                STA   $A0
8976: C8                   INY
8977: B1 5E                LDA   (INDEX),Y
8979: 85 A1                STA   $A1
897B: 60                   RTS

897C: 20 F4 7B  STEP       JSR   RST102
897F: A0 01                LDY   #1
8981: 84 A1                STY   FACLO
8983: 64 A0                STZ   FACMO
8985: C9 C7                CMP   #TOKSTEP
8987: D0 06                BNE   *+8
8989: 20 EC 7B             JSR   RST100
898C: 20 09 87             JSR   LBS03
898F: A0 FF                LDY   #-1           Step negative by default
8991: A5 A0                LDA   FACMO
8993: 30 06                BMI   :2
8995: C8                   INY
8996: 05 A1                ORA   FACLO
8998: F0 01                BEQ   :2
899A: C8                   INY
899B: 98        :2         TYA
899C: 49 80                EOR   #$80          Tag for integer var.
899E: 20 A4 89             JSR   NFRMSTK2
89A1: 4C C9 D7             JMP   $D7C9
```

```
                NFRMSTK2
89A4: A8                 TAY                   ;FAC sign or SGN(step value)
89A5: FA                 PLX
89A6: 68                 PLA
89A7: E8                 INX
89A8: 86 5E              STX   INDEX
89AA: D0 01              BNE   :1
89AC: 1A                 INC
89AD: 85 5F     :1       STA   INDEX+1
89AF: 5A                 PHY
89B0: 4C 23 DE           JMP   FRMSTCK3+3

                * New FRMEVL processing
                         PUT   PEERAROMBA
                TOKDIM   =     $86

                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

89B3: 20 EC 7B  RDIM     JSR   RST100
89B6: 20 77 8B           JSR   NCHKOPN
89B9: 20 90 7E           JSR   NGETARPT
89BC: A0 04              LDY   #4
89BE: B1 9B              LDA   (LOWTR),Y
89C0: 29 0F              AND   #$0F
89C2: 48                 PHA
89C3: B2 B8              LDA   (TXTPTR)
89C5: C9 2C              CMP   #´,´
89C7: D0 29              BNE   :1
89C9: A5 9C              LDA   LOWTR+1
89CB: 48                 PHA
89CC: A5 9B              LDA   LOWTR
89CE: 48                 PHA
89CF: 20 EC 7B           JSR   RST100
89D2: 20 84 8B           JSR   NGETBYT    Index of dimension in X&FACLO
89D5: 8A                 TXA
89D6: F0 24              BEQ   GOIQ
89D8: 68                 PLA
89D9: 85 9B              STA   LOWTR
89DB: 68                 PLA
89DC: 85 9C              STA   LOWTR+1
89DE: 68                 PLA
89DF: 38                 SEC
89E0: E5 A1              SBC   FACLO
89E2: 90 18              BCC   GOIQ
89E4: 0A                 ASL              ;Incidently clears the carry
89E5: 69 05              ADC   #5         Because of carry clear
89E7: A8                 TAY
89E8: B1 9B              LDA   (LOWTR),Y
89EA: AA                 TAX
```

```
89EB: C8       >123            INY
89EC: B1 9B    >124            LDA   (LOWTR),Y
89EE: A8       >125            TAY
89EF: 8A       >126            TXA
89F0: 90 04    >127            BCC   :0          Always
               >128  :1        MPLY
89F2: 7A       >128            PLY
89F3: A9 00    >130            LDA   #0
89F5: 38       >134            SEC
89F6: 20 F2 E2 >135  :0        JSR   GIVAYF
89F9: 4C 71 8B >136            JMP   NCHKCLS
               >137
89FC: 4C 99 E1 >138  GOIQ      JMP   GOIQERR     Raise a ILLEGAL QUANTITY ERROR
               >139
89FF: 20 7C 8B >140  RVRAI     JSR   NFRMEVL     True: evaluate second argument
8A02: 20 74 8B >141            JSR   NCHKCOM     Skip the comma and 3rd expr.
8A05: A9 29    >142            LDA   #´)´        until end of function detected
               >143
               >144  * This subroutine will skip program text until an
               >145  * end character is scanned.
8A07: 85 0E    >146  SKIPC     STA   ENDCHR
8A09: A0 00    >147            LDY   #0
8A0B: 84 BD    >148            STY   LEVELPAR    Parenthesis level
8A0D: 84 C0    >149            STY   GFLAG       String litteral parsing flag
8A0F: 88       >150            DEY
8A10: C8       >151  ]LOOP     INY
8A11: B1 B8    >152            LDA   (TXTPTR),Y
8A13: F0 36    >153            BEQ   LGSYNERR
8A15: C9 22    >154            CMP   #´"´
8A17: D0 08    >155            BNE   :0
8A19: A5 C0    >156            LDA   GFLAG       Inverse GFLAG b7
8A1B: 49 80    >157            EOR   #$80
8A1D: 85 C0    >158            STA   GFLAG
8A1F: B0 EF    >159            BCS   ]LOOP       Always
8A21: 24 C0    >160  :0        BIT   GFLAG       Within litteral string
8A23: 30 EB    >161            BMI   ]LOOP        so loop for next character.
8A25: C9 3A    >162            CMP   #´:´        End of instruction?
8A27: F0 22    >163            BEQ   LGSYNERR    SYNTAX ERROR if so
8A29: C9 28    >164            CMP   #´(´
8A2B: D0 04    >165            BNE   :1
8A2D: E6 BD    >166            INC   LEVELPAR
8A2F: B0 DF    >167            BCS   ]LOOP       Always
8A31: C9 29    >168  :1        CMP   #´)´
8A33: D0 08    >169            BNE   :2
8A35: A6 BD    >170            LDX   LEVELPAR
8A37: F0 08    >171            BEQ   :3
8A39: C6 BD    >172            DEC   LEVELPAR
8A3B: 10 D3    >173            BPL   ]LOOP
8A3D: A6 BD    >174  :2        LDX   LEVELPAR
8A3F: D0 CF    >175            BNE   ]LOOP
8A41: C5 0E    >176  :3        CMP   ENDCHR
8A43: D0 CB    >177            BNE   ]LOOP
8A45: 20 98 D9 >178            JSR   ADDON       Add Y to TXTPTR
8A48: 4C EC 7B >179            JMP   RST100
               >180
8A4B: 4C C9 DE >181  LGSYNERR JMP   SYNERR      Vector to SYNTAX ERROR
               >182
               * Handles the IIF function
8A4E: 20 74 8B RIIF     JSR   NCHKCOM    Check for trailing comma
8A51: A6 9D             LDX   FAC        True or false value?
8A53: D0 AA             BNE   RVRAI      True: then skip second arg.
8A55: A9 2C             LDA   #´,´
8A57: 20 07 8A          JSR   SKIPC      Skip 2nd expression
               * Evaluate 3rd arg. and check for closing parenthesis
8A5A: 4C 6E 8B          JMP   NPARCHK+3

8A5D: 20 7C 8B NFRMNUM  JSR   NFRMEVL    Get scalar valueH
8A60: 4C 6A DD          JMP   CHKNUM     Ensure numeric value

               * Takes care of the ´@´ processing
               * Refactor part of the FRMEVL ROM routine
8A63: 20 EC 7B FRMELMLP JSR   RST100
8A66: B0 07    FRMELM   BCS   :2         Branch iif not a digit
               :1
8A68: 64 C7             STZ   INTTYPSV
8A6A: 64 C8             STZ   VALTYPSV
8A6C: 4C 4A EC          JMP   $EC4A
8A6F: C9 2E    :2       CMP   #´.´
8A71: F0 F5             BEQ   :1
8A73: 20 7D E0          JSR   ISLETC
8A76: 90 5C             BCC   L3
8A78: AA                TAX
8A79: 30 28             BMI   :77
8A7B: C9 49             CMP   #´I´
8A7D: F0 08             BEQ   :80
8A7F: C9 4D             CMP   #´M´
8A81: F0 04             BEQ   :80
8A83: C9 54             CMP   #´T´
8A85: D0 1C             BNE   :77
               * Might be the IIF() function
8A87: A2 02    :80      LDX   #2
8A89: 20 12 87          JSR   RECON1
8A8C: F0 15             BEQ   :77
8A8E: 20 98 D9          JSR   ADDON
8A91: A5 BD             LDA   IDMOCL
8A93: 48                PHA
8A94: 20 77 8B          JSR   NCHKOPN
8A97: 20 5D 8A          JSR   NFRMNUM    Get operand numeric value
8A9A: 68                PLA              ;Recall IDMOCL from stack
8A9B: 38                SEC
8A9C: E9 07             SBC   #OFFMOU-TOFFST
8A9E: 90 AE             BCC   RIIF
               * Space for MOUSE and TIMER functions
               * ...: to be continued
8AA0: 4C 52 91          JMP   MTFUNC
               * Alphabetic character: variable name
8AA3: A2 00    :77      LDX   #0
8AA5: 86 10             STX   DIMFLG
8AA7: B2 B8             LDA   (TXTPTR)
8AA9: 20 9C 7E          JSR   NPTRGET1
               RFFVL    EQU   *-1
8AAC: 85 A0             STA   VPNT
8AAE: 84 A1             STY   VPNT+1
8AB0: A6 11             LDX   VALTYP
```

```
8AB2: F0 04    >253             BEQ   :41
8AB4: 64 AD    >259             STZ   STRNG1+1
8AB6: D0 17    >260             BNE   :SUITE     Always
8AB8: A6 12    >262  :41        LDX   INTTYP
8ABA: E0 81    >263             CPX   #$81
8ABC: D0 0E    >264             BNE   :42        Branch if not byte variable
8ABE: A2 00    >265             LDX   #0
8AC0: B2 83    >267             LDA   (VARPNT)
8AC2: 10 01    >271             BPL   *+3
8AC4: CA       >272             DEX              ;Poids fort dans X
8AC5: A8       >273             TAY              ;Poids faible dans Y
8AC6: 8A       >274             TXA              ;Poids fort dans A
8AC7: 20 F2 E2 >275             JSR   GIVAYF     Convert A, Y to FP
8ACA: 80 03    >277             BRA   :SUITE
8ACC: 20 E5 DE >281  :42        JSR   $DEE5
8ACF: A5 11    >285  :SUITE     LDA   VALTYP
8AD1: 85 C8    >286  RET3       STA   VALTYPSV
8AD3: 60       >287             RTS
8AD4: C9 C8    >288  L3         CMP   #TOKADD    Unary + operator: loop
8AD6: F0 8B    >289             BEQ   FRMELMLP
8AD8: C9 22    >290             CMP   #´"´
8ADA: D0 0A    >291             BNE   :4
8ADC: 20 81 DE >292             JSR   $DE81
8ADF: A9 FF    >293             LDA   #$FF
8AE1: 30 EE    >294             BMI   RET3       Always
8AE3: 4C 52 83 >295  ]LOOP      JMP   RUSR
8AE6: C9 D5    >296  :4         CMP   #TOKUSR
8AE8: F0 F9    >297             BEQ   ]LOOP
8AEA: A2 03    >298             LDX   #TOKMTIFE-TOKMOTIF-1
8AEC: DD 49 96 >299  ]LOOP      CMP   TOKMOTIF,X
8AEF: D0 08    >300             BNE   :NOK
8AF1: A8       >310             TAY
8AF2: 8A       >311             TXA
8AF3: 0A       >312             ASL
8AF4: AA       >313             TAX
8AF5: 98       >314             TYA
8AF6: 7C 4D 96 >315             JMP   (TOKMPF,X)
8AF9: CA       >317  :NOK       DEX
8AFA: 10 F0    >318             BPL   ]LOOP
8AFC: C9 40    >319  :6         CMP   #´@´
8AFE: D0 10    >320             BNE   :78
8B00: A5 C8    >321             LDA   VALTYPSV
8B02: 85 11    >322             STA   VALTYP
8B04: 30 04    >323             BMI   :60
8B06: A5 C7    >324             LDA   INTTYPSV
8B08: 85 12    >325             STA   INTTYP
8B0A: 4C EC 7B >326  :60        JMP   RST100
8B0D: 4C B3 89 >327  :79        JMP   RDIM
8B10: C9 86    >328  :78        CMP   #TOKDIM
8B12: F0 F9    >329             BEQ   :79
               >330
8B14: C9 D2    >331  :7         CMP   #TOKSGN
8B16: B0 03    >332             BCS   :10
8B18: 4C 6B 8B >333             JMP   NPARCHK
               >334
8B1B: 0A       >335  :10        ASL
8B1C: 48       >336             PHA
```

```
8B1D: AA       >337           TAX
8B1E: 20 EC 7B >338           JSR   RST100
8B21: E0 CF    >339           CPX   #$CF
8B23: 90 12    >340           BCC   :11
8B25: 20 77 8B >341           JSR   NCHKOPN
8B28: 20 7C 8B >342           JSR   NFRMEVL
8B2B: 20 74 8B >343           JSR   NCHKCOM
8B2E: 20 6C DD >344           JSR   CHKSTR
8B31: FA       >345           PLX
8B32: 20 51 8B >346           JSR   COMCMPLX
8B35: 80 0F    >350           BRA   :14
8B37: 20 6B 8B >352  :11      JSR   NPARCHK
8B3A: 7A       >353           PLY
8B3B: C0 C8    >354           CPY   #TOKSTRD+TOKSTRD
8B3D: F0 04    >355           BEQ   :15
8B3F: C0 CE    >356           CPY   #TOKCHRD+TOKCHRD
8B41: D0 08    >357           BNE   :13
8B43: 20 5D 8B >358  :15      JSR   CALLFUNC
8B46: A9 FF    >359  :14      LDA   #$FF
8B48: 85 C8    >360           STA   VALTYPSV
8B4A: 60       >361           RTS
8B4B: 20 5D 8B >362  :13      JSR   CALLFUNC
8B4E: 4C 6A DD >363           JMP   CHKNUM
               >364
               >365  COMCMPLX DO    KOPT16
8B51: A5 A1    >368           LDA   FACLO
8B53: 48       >369           PHA
8B54: A5 A0    >370           LDA   FACMO
8B56: 48       >371           PHA
8B57: DA       >373           PHX
8B58: 20 84 8B >374           JSR   NGETBYT
8B5B: 7A       >375           PLY
8B5C: DA       >376           PHX
               >377
8B5D: B9 DC CF >378  CALLFUNC LDA   $CFDC,Y
8B60: 85 91    >379           STA   $91
8B62: B9 DD CF >380           LDA   $CFDD,Y
8B65: 85 92    >381           STA   $92
8B67: 20 90 00 >382           JSR   $90
8B6A: 60       >383           RTS
               >384
8B6B: 20 77 8B >385  NPARCHK  JSR   NCHKOPN
8B6E: 20 7C 8B >386           JSR   NFRMEVL
               >387
8B71: A9 29    >388  NCHKCLS  LDA   #´)´
8B73: 2C       >389           HEX   2C
8B74: A9 2C    >390  NCHKCOM  LDA   #´,´
8B76: 2C       >391           HEX   2C
8B77: A9 28    >392  NCHKOPN  LDA   #´(´
8B79: 4C 62 82 >393           JMP   NSYNCHR
               >394
8B7C: 20 7B DD >395  NFRMEVL  JSR   FRMEVL
8B7F: A5 11    >396           LDA   VALTYP
8B81: 85 C8    >397           STA   VALTYPSV
8B83: 60       >398           RTS
               >399
8B84: 20 F8 E6 >400  NGETBYT  JSR   GETBYT
```

```
8B87: 48       >401           PHA
8B88: 20 F3 7C >402           JSR   SETITS
8B8B: 64 C8    >407           STZ   VALTYPSV
8B8D: 68       >409           PLA
8B8E: 60       >410  MFIN     RTS
                939
8B8F: 20 4C E7  940  ROUT11   JSR   COMBYTE    Get VTAB value in X
8B92: 20 59 F2  941           JSR   $F259      Do the VTAB
8B95: 20 4C E7  942           JSR   COMBYTE
8B98: 20 EA F7  943           JSR   $F7EA      Do the HTAB
8B9B: 20 F4 7B  944           JSR   RST102
8B9E: F0 13     945           BEQ   :0
8BA0: 20 74 8B  946           JSR   NCHKCOM
8BA3: A5 F1     947           LDA   $F1        Save current SPEED
8BA5: 48        948           PHA
8BA6: A9 01     949           LDA   #1         Fastest speed..
8BA8: 85 F1     950           STA   $F1
8BAA: 20 F4 7B  951           JSR   RST102
8BAD: 20 D5 DA  952           JSR   $DAD5      Do the PRINT
8BB0: 68        953           PLA              ;restore original SPEED
8BB1: 85 F1     954           STA   $F1
8BB3: 60        955  :0       RTS
                956
8BB4: 20 74 8B  957  ROUTGEN  JSR   NCHKCOM
8BB7: 20 84 8B  958           JSR   NGETBYT
8BBA: 8A        959           TXA
8BBB: F0 1F     960           BEQ   ROUT0
8BBD: E0 0B     961           CPX   #11
8BBF: F0 CE     962           BEQ   ROUT11
8BC1: E0 0A     963           CPX   #10
8BC3: D0 03     964           BNE   :2
8BC5: 4C 68 8F  965           JMP   ROUT10
8BC8: E0 08     966  :2       CPX   #8
8BCA: D0 03     967           BNE   :1
8BCC: 4C 7E 93  968           JMP   ROUT8
8BCF: E0 05     969  :1       CPX   #5
8BD1: D0 03     970           BNE   :0
8BD3: 4C B1 8D  971           JMP   KILLEMAL
8BD6: B0 B6     972  :0       BCS   MFIN
8BD8: E0 04     973           CPX   #4
8BDA: F0 3D     974           BEQ   ROUT4
8BDC: A5 69     975  ROUT0    LDA   VARTAB
8BDE: 85 06     976           STA   AUXPTR
8BE0: A5 6A     977           LDA   VARTAB+1
8BE2: 85 07     978           STA   AUXPTR+1
                979
8BE4: 20 F4 7B  980  ]LOOP    JSR   RST102
8BE7: F0 A5     981           BEQ   MFIN
8BE9: 20 74 8B  982           JSR   NCHKCOM
8BEC: 20 51 8D  983           JSR   NPTRGETX
8BEF: A5 9B     984           LDA   LOWTR
8BF1: C5 06     985           CMP   AUXPTR
8BF3: A5 9C     986           LDA   LOWTR+1
8BF5: E5 07     987           SBC   AUXPTR+1
8BF7: 90 95     988           BCC   MFIN
8BF9: A0 00     989           LDY   #0
8BFB: B1 9B     990  ]JLOOP   LDA   (LOWTR),Y
```

```
8BFD: AA                 TAX
8BFE: B1 06              LDA   (AUXPTR),Y
8C00: 91 9B              STA   (LOWTR),Y
8C02: 8A                 TXA
8C03: 91 06              STA   (AUXPTR),Y
8C05: C8                 INY
8C06: C0 07              CPY   #7
8C08: 90 F1              BCC   ]JLOOP
8C0A: 18                 CLC
8C0B: 98                 TYA
8C0C: 65 06              ADC   AUXPTR
8C0E: 85 06              STA   AUXPTR
8C10: 90 D2              BCC   ]LOOP
8C12: E6 07              INC   AUXPTR+1
8C14: B0 CE              BCS   ]LOOP      Always

8C16: 4C 76 DD  GGO2TMER JMP   GOTMIERR

8C19: A9 04     ROUT4    LDA   #4         Ensure enough room on stack
8C1B: 20 D6 D3           JSR   CHKMEM     7 bytes so 4 16bit words
8C1E: 68                 PLA              ;Pull return adress
8C1F: 68                 PLA
8C20: 20 74 8B           JSR   NCHKCOM
8C23: 20 94 7E           JSR   NPTRGTX
8C26: 24 12              BIT   INTTYP
8C28: 10 EC              BPL   GGO2TMER
8C2A: A5 9B              LDA   LOWTR
8C2C: C5 6B              CMP   ARYTAB
8C2E: 8D 35 96           STA   ITVADDR
8C31: A5 9C              LDA   LOWTR+1
8C33: 8D 36 96           STA   ITVADDR+1
8C36: E5 6C              SBC   ARYTAB+1
8C38: B0 DC              BCS   GGO2TMER
8C3A: A5 F8              LDA   REMSTK
8C3C: 8D 34 96           STA   SPROOT
                * Reinit the alive context markers
8C3F: A9 FF              LDA   #$FF
8C41: A2 08              LDX   #TABOFT-TABOFB
8C43: 9D 2A 96  ]LOOP    STA   TABOFT-1,X
8C46: CA                 DEX
8C47: D0 FA              BNE   ]LOOP
8C49: 86 C0              STX   IDX0       Starting index: 0
8C4B: 20 F4 7B  ]LOOP    JSR   RST102
8C4E: F0 0F              BEQ   XMFIN      End of instruction
8C50: 20 74 8B           JSR   NCHKCOM
8C53: 20 3A 93           JSR   NGTA2
8C56: 90 30              BCC   XMFIN1
8C58: 20 8B 8C           JSR   LBS04
8C5B: E6 C0              INC   IDX0
8C5D: D0 EC              BNE   ]LOOP

8C5F: A5 C0     XMFIN    LDA   IDX0
8C61: F0 21              BEQ   :0
8C63: A9 80              LDA   #$80
8C65: 8D DC 9C           STA   MTACTV
8C68: 20 24 8E           JSR   SETLTR
8C6B: 20 88 8C           JSR   XMFIN1
```

```
8C6E: A9 00              LDA   #0
8C70: 24 D8              BIT   ERRFLG
8C72: 10 01              BPL   *+3
8C74: 1A                 INC
8C75: A0 1A              LDY   #26
8C77: 91 9B              STA   (LOWTR),Y
8C79: 20 8B 8E           JSR   SAVERC
8C7C: A2 00              LDX   #0
8C7E: 8E 33 96           STX   INDX
8C81: 4C CD 8D           JMP   RESTOR1
8C84: 60        :0       RTS

8C85: 28        XMFIN2   PLP
8C86: 68                 PLA
8C87: 68                 PLA
8C88: 4C 95 D9  XMFIN1   JMP   DATA

                * Handle a single entry (index in IDX0)
                LBS04
                * Array base address in (LOWTR, LOWTR+1)
8C8B: A6 C0              LDX   IDX0
8C8D: A5 9B              LDA   LOWTR
8C8F: 85 06              STA   AUXPTR
8C91: E5 6B              SBC   ARYTAB     C already set
8C93: 9D 23 96           STA   TABOFB,X
8C96: 08                 PHP
8C97: A5 9C              LDA   LOWTR+1
8C99: 85 07              STA   AUXPTR+1
                * Is local error handling desired
8C9B: 20 74 8B           JSR   NCHKCOM
8C9E: 20 F8 E6           JSR   GETBYT
                * Offset 24 for local error handling flag
8CA1: A0 1A              LDY   #26
8CA3: E0 02              CPX   #2
8CA5: D0 06              BNE   :0
8CA7: CA                 DEX
8CA8: 24 D8              BIT   ERRFLG
8CAA: 30 01              BMI   :0
8CAC: CA                 DEX
8CAD: 8A        :0       TXA
8CAE: 91 06              STA   (AUXPTR),Y
8CB0: F0 0E              BEQ   :1
8CB2: A0 19              LDY   #26-1
8CB4: BE 2F 96  ]LOOP    LDX   P0OFFSET-8,Y
8CB7: B5 00              LDA   0,X
8CB9: 91 06              STA   (AUXPTR),Y
8CBB: 88                 DEY
8CBC: E0 F4              CPX   #TXTPSV
8CBE: D0 F4              BNE   ]LOOP
                * Offsets 27 and 28 for swapped in machine code routine
8CC0: A9 1C     :1       LDA   #28
8CC2: 20 3A 8D           JSR   LBS041
                * Offsets 29 and 30 for swapped out machine code routine
8CC5: A9 1E              LDA   #30
8CC7: 20 3A 8D           JSR   LBS041
8CCA: 20 74 8B           JSR   NCHKCOM
8CCD: 20 0C DA           JSR   LINGET
```

```
8CD0: 20 1A D6  1113          JSR   FNDLIN
8CD3: 90 B0     1114          BCC   XMFIN2     Non existent line: exit
                1115 * Offsets 0 and 1 for array name
                1116 * Offsets 2 and 3 for offset to next array
                1117 * Offset 4 for number of dimension
                1118 * Offsets 5 and 6 for last dimension value
8CD5: A0 04     1119          LDY   #4
8CD7: B1 06     1120          LDA   (AUXPTR),Y
8CD9: 49 41     1121          EOR   #%01000001 Must be 16bits integer and
8CDB: D0 A8     1122          BNE   XMFIN2     # of dimensions must be 1
8CDD: A5 07     1123          LDA   AUXPTR+1
8CDF: 28        1124          PLP              ;Restaure Carry from previous SBC
8CE0: E5 6C     1125          SBC   ARYTAB+1
8CE2: A6 C0     1126          LDX   IDX0
8CE4: 9D 2B 96  1127          STA   TABOFT,X
                1128 * Offset 7 and 8 for storing SP value
                1129 * Integer variable value storage order
8CE7: A0 07     1130          LDY   #7
8CE9: A9 00     1131          LDA   #0
8CEB: 91 06     1132          STA   (AUXPTR),Y
8CED: C8        1133          INY
8CEE: A5 F8     1134          LDA   REMSTK
8CF0: E9 07     1135          SBC   #7         ;Carry already set
8CF2: 91 06     1136          STA   (AUXPTR),Y
8CF4: C8        1137          INY
                1138 * Offset 9 and 10 for LINNUM storage
                1139 * (natural storage order)
8CF5: A5 50     1140          LDA   LINNUM
8CF7: 91 06     1141          STA   (AUXPTR),Y
8CF9: C8        1142          INY
8CFA: A5 51     1143          LDA   LINNUM+1
8CFC: 91 06     1144          STA   (AUXPTR),Y
8CFE: C8        1145          INY
                1146 * Offset 11 and 12 for TXTPTR storage
                1147 * (natural storage order)
8CFF: A5 9B     1148          LDA   LOWTR
8D01: 69 03     1149          ADC   #4-1       Because Carry already set
8D03: 91 06     1150          STA   (AUXPTR),Y
8D05: C8        1151          INY
8D06: A5 9C     1152          LDA   LOWTR+1
8D08: 69 00     1153          ADC   #0
8D0A: 91 06     1154          STA   (AUXPTR),Y
8D0C: C8        1155          INY
                1156 * Offset 13 and 14 for OLDTEXT storage
                1157 * (natural storage order)
8D0D: A5 9B     1158          LDA   LOWTR
8D0F: 69 04     1159          ADC   #4
8D11: 91 06     1160          STA   (AUXPTR),Y
8D13: C8        1161          INY
8D14: A5 9C     1162          LDA   LOWTR+1
8D16: 69 00     1163          ADC   #0
8D18: 91 06     1164          STA   (AUXPTR),Y
8D1A: A0 1F     1165          LDY   #31
                1166 * Offsset 31 and above for stack content storage
                1167 * from current SP to SPROOT
                1168 * For the time being (init), prepare a GOSUB frame
8D1C: A9 B0     1169          LDA   #TOKGOSUB
```

```
8D1E: A2 03     1170          LDX   #3
8D20: 91 06     1171 ]JLOOP   STA   (AUXPTR),Y Do not mind calling CURLIN
8D22: C8        1172          INY
8D23: CA        1173          DEX
8D24: D0 FA     1174          BNE   ]JLOOP
8D26: A5 79     1175          LDA   OLDTPTR
8D28: 91 06     1176          STA   (AUXPTR),Y
8D2A: C8        1177          INY
8D2B: A5 7A     1178          LDA   OLDTPTR+1
8D2D: 91 06     1179          STA   (AUXPTR),Y
8D2F: C8        1180          INY
8D30: A9 D1     1181          LDA   #NEWSTT-1
8D32: 91 06     1182          STA   (AUXPTR),Y
8D34: C8        1183          INY
8D35: A9 D7     1184          LDA   #>NEWSTT-1
8D37: 91 06     1185          STA   (AUXPTR),Y
8D39: 60        1186          RTS
                1187
8D3A: 48        1188 LBS041   PHA
8D3B: 20 74 8B  1189          JSR   NCHKCOM
8D3E: 20 67 DD  1190          JSR   FRMNUM
8D41: 20 52 E7  1191          JSR   GETADR
8D44: 7A        1192          PLY
8D45: A5 51     1193          LDA   LINNUM+1
8D47: 91 06     1194          STA   (AUXPTR),Y
8D49: F0 05     1195          BEQ   :0
8D4B: 88        1196          DEY
8D4C: A5 50     1197          LDA   LINNUM
8D4E: 91 06     1198          STA   (AUXPTR),Y
8D50: 60        1199 :0       RTS
                1200
                1201 NPTRGETX DO    KOPT-K65C02
8D51: 64 82     1205          STZ   VARNAM+1
8D53: 20 5C 82  1207          JSR   MISLETC
8D56: 85 81     1208          STA   VARNAM
8D58: 20 EC 7B  1209          JSR   RST100
8D5B: 90 05     1210          BCC   :0
8D5D: 20 7D E0  1211          JSR   ISLETC
8D60: 90 16     1212          BCC   :3
8D62: 85 82     1213 :0       STA   VARNAM+1
8D64: 20 EC 7B  1214 ]LOOP    JSR   RST100
8D67: 90 FB     1215          BCC   ]LOOP
8D69: 20 7D E0  1216          JSR   ISLETC
8D6C: B0 F6     1217          BCS   ]LOOP
8D6E: 90 08     1218          BCC   :3
8D70: 20 F1 85  1219 :2       JSR   DECTPTR
8D73: A6 81     1220          LDX   VARNAM
8D75: BD 61 9B  1221          LDA   TYPLET-´A´,X
8D78: A2 03     1223 :3       LDX   #3
8D7A: 20 E8 85  1227          JSR   XFROMMOT+2
8D7D: D0 F1     1228          BNE   :2
8D7F: 4C D0 85  1229          JMP   ROUT1Y
                1230
8D82: 2C DC 9C  1231 RNEWISUI BIT   MTACTV
8D85: 10 40     1232          BPL   RESTORD
                1233
                1234          PUT   PEERMTK
                * Main Active MT entry point
8D87: BA        RMTCTRL  TSX              ;Test for an exhausted thread?
8D88: EC 34 96           CPX   SPROOT
8D8B: AE 33 96           LDX   INDX
8D8E: 90 07              BCC   :2
8D90: A9 FF              LDA   #$FF       Mark the current thread
8D92: 9D 2B 96           STA   TABOFT,X    before switching to another
8D95: B0 15              BCS   KX3        Always branch
8D97: 2C DA 9C  :2       BIT   INHACTV
8D9A: 30 2B              BMI   RESTORD
8D9C: CE DB 9C           DEC   CTRACTV    Time for a context switch?
8D9F: D0 26              BNE   RESTORD    Not yet
8DA1: BD 2B 96           LDA   TABOFT,X   Get BASIC array where to save
8DA4: 20 46 8E           JSR   NEXTC2      content
8DA7: DA                 PHX
8DA8: 20 54 8E           JSR   SAVER      Perform the SAVE
8DAB: FA                 PLX              ;Get back the new context index
                KX3
8DAC: 20 2D 8E           JSR   NEXTCTX    Search for a new context index
8DAF: 90 26              BCC   RESTOR2    Found one
                * Restore context from calling BASIC line
8DB1: 20 24 8E  KILLEMAL JSR   SETLTR     Restore context from calling
8DB4: 20 10 8E           JSR   RESTORC     BASIC line
8DB7: AE 34 96           LDX   SPROOT
8DBA: 86 F8              STX   REMSTK
8DBC: 20 C3 8D           JSR   R0
8DBF: 9A                 TXS
8DC0: 4C D2 D7           JMP   NEWSTT
8DC3: 4E DC 9C  R0       LSR   MTACTV
8DC6: 60                 RTS

8DC7: 20 5D 90  RESTORD  JSR   LBS10
8DCA: 4C 20 D8           JMP   $D820
                * General purpose restore routine
                * Input: X register index of context
8DCD: BD 2B 96  RESTOR1  LDA   TABOFT,X
8DD0: C9 FF              CMP   #$FF       Safe guard: do not restore a
8DD2: F0 3B              BEQ   RESTORF    terminated thread..
8DD4: 20 46 8E           JSR   NEXTC2

                * Input from caller: X: context index
8DD7: AD DD 9C  RESTOR2  LDA   ICTRACTV   Reinit counter
8DDA: 8D DB 9C           STA   CTRACTV     value
                * Update ITHREAD% variable value
8DDD: AD 36 96           LDA   ITVADDR+1
8DE0: F0 0C              BEQ   RESTOR     Skip if no var. defined
8DE2: 85 07              STA   AUXPTR+1
8DE4: AD 35 96           LDA   ITVADDR
8DE7: 85 06              STA   AUXPTR
8DE9: 8A                 TXA
8DEA: A0 03              LDY   #3
8DEC: 91 06              STA   (AUXPTR),Y
8DEE: 18        RESTOR   CLC
8DEF: A0 1C              LDY   #28        Trigger the page in routine if
8DF1: 20 6F 8E           JSR   SWPIO       defined
8DF4: AE 33 96           LDX   INDX
8DF7: B0 B3              BCS   KX3
```

```
               * Do the RESTOR itself
               * Input: LOWTR: Array base address
8DF9: 20 10 8E          JSR   RESTORC
               * Do the Stack restore
8DFC: A0 1F             LDY   #31        From offset 31 within context
8DFE: A6 F8             LDX   REMSTK      array storage
8E00: 9A       RESTORX  TXS
8E01: EC 34 96 ]LOOP    CPX   SPROOT     Until SPROOT value is reached
8E04: B0 C1             BCS   RESTORD
8E06: E8                INX
8E07: B1 9B             LDA   (LOWTR),Y
8E09: 9D 00 01          STA   $0100,X
8E0C: C8                INY
8E0D: 90 F2             BCC   ]LOOP      Always
8E0F: 60       RESTORF  RTS

8E10: 20 7E 8E RESTORC  JSR   LBS06
8E13: 90 02             BCC   *+4
8E15: 85 D8             STA   ERRFLG
8E17: B1 9B    ]LOOP    LDA   (LOWTR),Y
8E19: BE 2F 96          LDX   P0OFFSET-8,Y
8E1C: 95 00             STA   0,X
8E1E: 88                DEY
8E1F: E0 F8             CPX   #REMSTK
8E21: D0 F4             BNE   ]LOOP
8E23: 60                RTS

               * Subroutine to get the context storage index for
               * global (i.e. Perrsoft MT kernel calling line)
8E24: A9 08    SETLTR   LDA   #SVPTR-8
8E26: 85 9B             STA   LOWTR
8E28: A9 96             LDA   #>SVPTR-8
8E2A: 85 9C             STA   LOWTR+1
8E2C: 60                RTS
               * Subroutine to get the next context after the current one
               * (index in X).
8E2D: A0 00    NEXTCTX  LDY   #0         ctr. to avoid counting too far
8E2F: E8       ]LOOP    INX              ;Wrap around the context ptr
8E30: E0 08             CPX   #TABOFT-TABOFB area..
8E32: 90 02             BCC   :0
8E34: A2 00             LDX   #0         Perform wrap...
8E36: BD 2B 96 :0       LDA   TABOFT,X
8E39: C9 FF             CMP   #$FF       Got an active one (iif <> $FF)
8E3B: D0 06             BNE   :1         Yes...
8E3D: C8                INY              ;Bump counter
8E3E: C0 08             CPY   #TABOFT-TABOFB till all scanned
8E40: 90 ED             BCC   ]LOOP      Not yet: see next context ptr
8E42: 60                RTS              ;Exit with carry set..
8E43: 8E 33 96 :1       STX   INDX       Memorize the new context index
8E46: A8       NEXTC2   TAY              ;From offset to absolute address
8E47: BD 23 96          LDA   TABOFB,X   by adding the ARYTAB base address
8E4A: 65 6B             ADC   ARYTAB     for arrays within Applesoft
8E4C: 85 9B             STA   LOWTR
8E4E: 98                TYA
8E4F: 65 6C             ADC   ARYTAB+1
8E51: 85 9C             STA   LOWTR+1    Result in LOWTR pointer..
8E53: 60                RTS              ;Exit with carry clear (always)
```

```
                >131
                >132  * Save the context into BASIC array
                >133  * Input: LOWTR: array base address
8E54: 20 8B 8E >134  SAVER    JSR   SAVERC
8E57: A0 1E    >135           LDY   #30        Possible trigger for page out
8E59: 20 6F 8E >136           JSR   SWPIO       event...
                >137  * Now it´s time to save the stack extension
8E5C: A0 1F    >138           LDY   #31
                >139  * As a subroutine, do not depend on current stack ptr.
                >140  * But rather on memorized stack ptr. (within exec loop)
8E5E: A6 F8    >141           LDX   REMSTK
8E60: EC 34 96 >142  ]LOOP    CPX   SPROOT
8E63: B0 09    >143           BCS   :0
8E65: E8       >144           INX
8E66: BD 00 01 >145           LDA   $0100,X
8E69: 91 9B    >146           STA   (LOWTR),Y
8E6B: C8       >147           INY
8E6C: 90 F2    >148           BCC   ]LOOP
8E6E: 60       >149  :0       RTS
                >150
                >151  * Routine to possibly trigger page in/page out routine
                >152  * for every configured coroutine. Inputs are:
                >153  * LOWTR: context array base address
                >154  * Y either 30 or 28 for page in/out event
8E6F: B1 9B    >155  SWPIO    LDA   (LOWTR),Y
8E71: F0 0A    >156           BEQ   :0         No routine defined
8E73: 85 07    >157           STA   AUXPTR+1
8E75: 88       >158           DEY
8E76: B1 9B    >159           LDA   (LOWTR),Y
8E78: 85 06    >160           STA   AUXPTR
                >161  * Called routine must preserve registers
8E7A: 6C 06 00 >162           JMP   (AUXPTR)
8E7D: 60       >163  :0       RTS
                >164
8E7E: A0 1A    >165  LBS06    LDY   #26
8E80: B1 9B    >166  LBS061   LDA   (LOWTR),Y
8E82: D0 04    >167           BNE   :0
8E84: 38       >169           SEC
8E85: A0 0E    >171  :1       LDY   #PIOFFSET-P0OFFSET+8-1
8E87: 60       >172           RTS
8E88: 18       >174  :0       CLC
8E89: 88       >178           DEY              ;Shortcut for
8E8A: 60       >179           RTS              ; LDY #PEOFFSET-P0OFFSET+8-1
                >180
8E8B: 20 7E 8E >182  SAVERC   JSR   LBS06
8E8E: BE 2F 96 >187  ]LOOP    LDX   P0OFFSET-8,Y
8E91: B5 00    >188           LDA   0,X        Value to save
8E93: 91 9B    >189           STA   (LOWTR),Y
8E95: 88       >190           DEY
8E96: E0 F8    >191           CPX   #REMSTK
8E98: D0 F4    >192           BNE   ]LOOP
8E9A: 60       >193           RTS
                 1235
                 1236           PUT   PEERMOUSTIME
                >1    * Base addresses for mouse interface
                >2    BAXLO    EQU   $0478      X low
                >3    BAYLO    EQU   $04F8      Y low
                >4    BAXHI    EQU   $0578      X high
                >5    BAYHI    EQU   $05F8      Y high
                >6    BAMBS    EQU   $0778      Button status
                >7
                >8    TRACE    EQU   $D805
                >9    IRQV     EQU   $03FE      Page 3 Interrupt vector
                >10
                >11   * Reason codes for entering Mouse interface
                >12   RSETM    =     0
                >13   RSRVM    =     1
                >14   RREAD    =     2
                >15   RCLR     =     3
                >16   RPOS     =     4
                >17   RCLM     =     5
                >18   RHOM     =     6
                >19   RINI     =     7
                >20
                >21   CONINT   EQU   $E6FB      FAC to single byte
                >22
                >23   * Interrupt servicing routine
8E9B: A2 01     >24   IRQHDLR  LDX   #RSRVM
8E9D: 20 39 91  >25            JSR   TOMOUSE
8EA0: B0 39     >26            BCS   :2         ; Not from mouse or spurious
8EA2: AE D1 9C  >27            LDX   MOSL
8EA5: BD 78 07  >28            LDA   BAMBS,X
8EA8: 4A        >29            LSR
                >30   * Movement interrupt bit into b0 and
                >31   * button bit into b1, VBL interrupt bit
                >32   * into b2
8EA9: 29 07     >33            AND   #7         mask out other bits
8EAB: AA        >34            TAX
8EAC: BD E7 99  >35            LDA   MSTATUS,X  Get internal status
8EAF: 8D F1 99  >36            STA   WORKPL1
8EB2: A2 02     >37            LDX   #RREAD
8EB4: 20 39 91  >38            JSR   TOMOUSE
8EB7: 2C F1 99  >39            BIT   WORKPL1
8EBA: 10 18     >40            BPL   :1
                >41   * Decrement runtime counter
8EBC: AE 17 9A  >55            LDX   TIINC
8EBF: D0 03     >56            BNE   :01
8EC1: CE 18 9A  >57            DEC   TIINC+1
8EC4: CA        >58   :01      DEX
8EC5: 8E 17 9A  >59            STX   TIINC
8EC8: D0 05     >60            BNE   :02
8ECA: AD 18 9A  >61            LDA   TIINC+1
8ECD: F0 1D     >62            BEQ   :00
                >63   :02
8ECF: A9 80     >66            LDA   #$80
8ED1: 1C F1 99  >67            TRB   WORKPL1
8ED4: AD F1 99  >73   :1       LDA   WORKPL1
8ED7: 0C F2 99  >75            TSB   MIRQST
8EDA: 40        >80   ]LOOP    RTI
                >81
                >82   * No spurious interrupt is fatal to us..
                >83   * I´m afraid of no ghosts.... ;-)
8EDB: AD F0 99  >84   :2       LDA   OLDVECT+1
8EDE: C9 FF     >85            CMP   #>$FF65
```

```
8EE0: D0 07    >86            BNE   :20
8EE2: AD EF 99 >87            LDA   OLDVECT
8EE5: C9 65    >88            CMP   #$FF65
8EE7: F0 F1    >89            BEQ   ]LOOP
8EE9: 6C EF 99 >90   :20      JMP   (OLDVECT)
               >91
8EEC: AD 15 9A >94   :00      LDA   KTINC
8EEF: 8D 17 9A >95            STA   TIINC
8EF2: AD 16 9A >96            LDA   KTINC+1
8EF5: 8D 18 9A >97            STA   TIINC+1
8EF8: 80 DA    >99            BRA   :1
               >104
               >105  * Install new IRQ handler and save the original handler
               >106  * to build a daisy chain..
               >107  * Nouveau mode dans MOMODE
8EFA: AD D8 99 >108  INSIRQV  LDA   MOMODE
8EFD: C9 02    >109           CMP   #2
8EFF: 90 20    >110           BCC   :1
8F01: AD FE 03 >127           LDA   IRQV
8F04: AE FF 03 >128           LDX   IRQV+1
8F07: C9 9B    >129           CMP   #IRQHDLR
8F09: D0 04    >130           BNE   :0
8F0B: E0 8E    >131           CPX   #>IRQHDLR
8F0D: F0 12    >132           BEQ   :1
8F0F: 78       >133  :0       SEI
8F10: 8D EF 99 >134           STA   OLDVECT
8F13: 8E F0 99 >135           STX   OLDVECT+1
8F16: A9 9B    >136           LDA   #IRQHDLR
8F18: 8D FE 03 >136           STA   IRQV
8F1B: A9 8E    >136           LDA   #>IRQHDLR
8F1D: 8D FF 03 >136           STA   IRQV+1
8F20: 58       >138           CLI
8F21: 60       >139  :1       RTS
               >140
               >141  * Deinstall IRQ handler
8F22: AD D8 99 >142  DINSIRQV LDA   MOMODE
8F25: C9 02    >143           CMP   #2
8F27: B0 12    >144           BCS   :1
8F29: 78       >145           SEI
8F2A: AD F0 99 >159           LDA   OLDVECT+1
8F2D: F0 0C    >160           BEQ   :1
8F2F: 8D FF 03 >161           STA   IRQV+1
8F32: 9C F0 99 >163           STZ   OLDVECT+1
8F35: AD EF 99 >168           LDA   OLDVECT
8F38: 8D FE 03 >169           STA   IRQV
8F3B: 60       >171  :1       RTS
               >172
8F3C: 48       >173  CMPCLAMP PHA
               >174  * X/Y min% expression
8F3D: 20 08 90 >175           JSR   NEVAL
8F40: 8D 78 05 >176           STA   $0578
8F43: 8C 78 04 >177           STY   $0478
               >178  * X/Y max% expression
8F46: 20 08 90 >179           JSR   NEVAL
8F49: 8D F8 05 >180           STA   $05F8
8F4C: 8C F8 04 >181           STY   $04F8
8F4F: 68       >182           PLA
```

```
8F50: A2 05    >183          LDX   #RCLM
8F52: 4C 39 91 >184          JMP   TOMOUSE
               >185
8F55: C5 A1    >186  IVALARG CMP   FAC+4
8F57: 90 01    >187          BCC   *+3
8F59: 60       >188          RTS
8F5A: 68       >189          PLA
8F5B: 68       >190          PLA
8F5C: 4C 99 E1 >191  ]ERR    JMP   $E199      Illegal quantity error
               >192
8F5F: A9 00    >193  COMCLAMP LDA  #0
8F61: 20 3C 8F >194          JSR   CMPCLAMP
8F64: A9 01    >195          LDA   #1
8F66: D0 D4    >196          BNE   CMPCLAMP
               >197
8F68: 20 74 8B >198  ROUT10  JSR   NCHKCOM
8F6B: 20 84 8B >199          JSR   NGETBYT    Get reason code in X reg.
8F6E: CA       >200          DEX
8F6F: CA       >201          DEX
8F70: 30 EA    >202          BMI   ]ERR
8F72: E0 05    >203          CPX   #5
8F74: B0 E6    >204          BCS   ]ERR
8F76: 20 C1 92 >205          JSR   ISMOUSH
8F79: AD D8 99 >206          LDA   MOMODE
8F7C: 29 0F    >207          AND   #$F
8F7E: D0 05    >208          BNE   :1
8F80: A2 25    >209          LDX   #37
8F82: 4C D0 92 >210          JMP   NERRH
               >211  * Only READ (2), CLEAR (3), POS(4), CLAMP (5) and HOME (6)
               >212  * reason codes are valid.
8F85: 8A       >213  :1      TXA
8F86: F0 11    >214          BEQ   COMREAD
8F88: CA       >215          DEX
8F89: F0 09    >216          BEQ   COMCLEAR
8F8B: CA       >217          DEX
8F8C: F0 39    >218          BEQ   COMPOS
8F8E: CA       >219          DEX
8F8F: F0 CE    >220          BEQ   COMCLAMP
8F91: A2 06    >221          LDX   #RHOM
8F93: 2C       >222          HEX   2C         Skip next two bytes
8F94: A2 56    >223  COMCLEAR LDX  #RCLEAR
8F96: 4C 39 91 >224  FINMOUSE JMP  TOMOUSE
               >225
8F99: AE F4 99 >226  COMREAD LDX   MODERUN
8F9C: D0 05    >227          BNE   :1
8F9E: A2 02    >228          LDX   #RREAD
8FA0: 20 39 91 >229          JSR   TOMOUSE
               >230  * Handles X% host variable
8FA3: AE D1 9C >231  :1      LDX   MOSL
8FA6: BD 78 05 >232          LDA   BAXHI,X
8FA9: 20 E3 8F >233          JSR   NPTRG
8FAC: BD 78 04 >234          LDA   BAXLO,X
8FAF: 91 83    >235          STA   (VARPNT),Y
               >236  * Handle Y% host variable
8FB1: BD F8 05 >237          LDA   BAYHI,X
8FB4: 20 E3 8F >238          JSR   NPTRG
8FB7: BD F8 04 >239          LDA   BAYLO,X
```

```
8FBA: 91 83    >240          STA   (VARPNT),Y
               >241  * Handle S% for button status variable
8FBC: A9 00    >242          LDA   #0
8FBE: 20 E3 8F >243          JSR   NPTRG
8FC1: BD 78 07 >244          LDA   BAMBS,X
8FC4: 91 83    >245          STA   (VARPNT),Y
8FC6: 60       >246          RTS
               >247
               >248  COMPOS
               >249  * X% expression
8FC7: 20 08 90 >250          JSR   NEVAL
8FCA: 9D 78 05 >251          STA   BAXHI,X
8FCD: 98       >252          TYA
8FCE: 9D 78 04 >253          STA   BAXLO,X
               >254  * Y% expression
8FD1: 20 08 90 >255          JSR   NEVAL
8FD4: 9D F8 05 >256          STA   BAYHI,X
8FD7: 98       >257          TYA
8FD8: 9D F8 04 >258          STA   BAYLO,X
8FDB: A2 04    >259          LDX   #RPOS
8FDD: 4C 96 8F >260          JMP   FINMOUSE
               >261
8FE0: 4C 76 DD >262  ]ERR    JMP   GOTMIERR   TYPE MISMATCH ERROR
8FE3: 48       >263  NPTRG   PHA
8FE4: 20 74 8B >264          JSR   NCHKCOM
8FE7: 20 94 7E >265          JSR   NPTRGTX
8FEA: A5 12    >266          LDA   INTTYP
8FEC: 10 F2    >267          BPL   ]ERR
8FEE: 29 0F    >268          AND   #15        cater for integer subtypes
8FF0: F0 04    >269          BEQ   :1         only $80 and $82 are valid
8FF2: C9 02    >270          CMP   #2
8FF4: D0 EA    >271          BNE   ]ERR
8FF6: AE D1 9C >272  :1      LDX   MOSL
8FF9: 68       >273          PLA
8FFA: 92 83    >275          STA   (VARPNT)
8FFC: A0 01    >276          LDY   #1
8FFE: 60       >282          RTS
               >283
               >284  * Result in FAC+3, FAC+4
8FFF: 20 74 8B >285  NEVALC  JSR   NCHKCOM
9002: 20 5D 8A >286          JSR   NFRMNUM
9005: 4C 2C 7D >287          JMP   NROUT      Replac. for ROUND.FAC/AYINT
               >288
9008: 20 FF 8F >289  NEVAL   JSR   NEVALC
900B: A5 A0    >290          LDA   FAC+3
900D: A4 A1    >291          LDY   FAC+4
900F: AE D1 9C >292          LDX   MOSL
9012: 60       >293  ]RET    RTS
               >294
               >295  * Common subroutine for parsing new tokens
               >296  * X upon entry: 0: updates TXTPTR if token found
               >297  * 1: skip updating TXTPTR even when token found
9013: 86 C0    >298  COMLBS  STX   GFLAG
9015: B2 B8    >300          LDA   (TXTPTR)
9017: 30 19    >305          BMI   :2
9019: C9 4D    >306          CMP   #´M´
901B: F0 04    >307          BEQ   :1
```

```
901D: C9 54    >308            CMP   #´T´
901F: D0 11    >309            BNE   :2
9021: A2 03    >310  :1        LDX   #3
9023: 20 12 87 >311            JSR   RECON1
9026: F0 EA    >312            BEQ   ]RET
9028: 20 1F 91 >313            JSR   COMINT4    Check mouse hardware/reinit
902B: A6 C0    >314            LDX   GFLAG
902D: D0 E3    >315            BNE   ]RET
902F: 4C 98 D9 >316            JMP   ADDON      will exit with Z flag clear
               >317  :2
9032: A2 00    >319            LDX   #0
9034: 60       >323  ]RET      RTS
               >324
               >325  * New instructions handling
               >326  * for MOUSE and TIMER instructions
9035: 4C F4 7B >327  ]LOOP     JMP   RST102
9038: 68       >328  ]ERR1     PLA              ;Pull IDMOCL from stack
9039: 68       >329            PLA              ;Pull return address
903A: 68       >330            PLA
903B: 4C C9 DE >331  ]ERR      JMP   SYNERR
               >332  * MOUSE/TIMER STOP handler
903E: C0 08    >333  ]JLOOP    CPY   #OFFTIM-TOFFST
9040: A2 00    >334            LDX   #0
9042: 90 01    >335            BCC   *+3        Branch iif MOUSE
9044: E8       >336            INX
9045: AD D8 99 >337            LDA   MOMODE
9048: 3D E3 99 >338            AND   MOETMSK,X
               >339  * Compare to minimum allowable value
904B: DD E5 99 >340            CMP   MOCMPVAL,X
904E: B0 05    >341            BCS   :0         OK iif greater or equal
9050: A2 25    >342            LDX   #37
9052: 4C D0 92 >343            JMP   NERRH
9055: A9 01    >344  :0        LDA   #1         Update MODEPEC configuration
9057: 9D F6 99 >345            STA   MODEPEC,X
905A: 4C D2 D7 >346            JMP   NEWSTT
905D: A2 00    >347  LBS10     LDX   #0
905F: 20 13 90 >348            JSR   COMLBS
9062: F0 D1    >349            BEQ   ]LOOP
9064: A5 BD    >350            LDA   IDMOCL
9066: 48       >351            PHA
9067: B2 B8    >353            LDA   (TXTPTR)
9069: A0 01    >354            LDY   #1
906B: C9 B3    >360            CMP   #$B3       STOP token?
906D: F0 0F    >361            BEQ   :3
906F: C9 B4    >362            CMP   #$B4
9071: F0 0B    >363            BEQ   :3         ON token?
9073: C9 4F    >364            CMP   #´O´
9075: D0 C1    >365            BNE   ]ERR1
9077: A2 05    >366            LDX   #5         Look up possible OFF pattern
9079: 20 12 87 >367            JSR   RECON1
907C: F0 BA    >368            BEQ   ]ERR1
907E: AA       >369  :3        TAX              ;X STOP/ON token or 0 (OFF)
907F: 86 B4    >370            STX   XSAV
9081: 20 98 D9 >371            JSR   ADDON
9084: 7A       >372            PLY
9085: 68       >373            PLA
9086: 68       >374            PLA
```

```
9087: 20 F4 7B >375           JSR   RST102
908A: F0 15    >376           BEQ   :23        If EOI found
908C: E0 B4    >377           CPX   #$B4
908E: D0 AB    >378           BNE   ]ERR       SYNTAX ERR if not ON nor EOI
9090: DA       >379           PHX
9091: 5A       >380           PHY
9092: 20 FF 8F >381           JSR   NEVALC     Get factor/mode value after comma
9095: 7A       >382           PLY
9096: FA       >383           PLX
9097: 86 B4    >384           STX   XSAV
9099: C0 07    >385           CPY   #OFFMOU-TOFFST
909B: D0 06    >386           BNE   :20
909D: 20 FE E6 >387           JSR   $E6FE      FAC integer -> single byte
90A0: 2C       >388           HEX   2C
90A1: A2 01    >389  :23      LDX   #1
90A3: 86 C0    >390  :20      STX   GFLAG
90A5: 84 BD    >391           STY   IDMOCL
90A7: A5 B4    >392           LDA   XSAV        A: ON/OFF/STOP index
90A9: C9 B3    >393           CMP   #$B3       STOP token?
90AB: F0 91    >394           BEQ   ]JLOOP
               >395  * IDMOCL in page zero, STOP/ON/OFF indic. in A reg.
90AD: A6 BD    >396           LDX   IDMOCL
90AF: E0 07    >397           CPX   #OFFMOU-TOFFST
90B1: D0 3F    >398           BNE   TIMEINST
               >399
               >400  * Mouse event handler
90B3: C9 B4    >401           CMP   #$B4       MOUSE ON?
90B5: D0 04    >402           BNE   *+6        No
90B7: A2 00    >403           LDX   #0
90B9: F0 0D    >404           BEQ   :8
90BB: A2 07    >405           LDX   #7
90BD: E4 C0    >406  ]LOOP    CPX   GFLAG
90BF: F0 07    >407           BEQ   :8
90C1: CA       >408           DEX
90C2: CA       >409           DEX
90C3: 10 F8    >410           BPL   ]LOOP
90C5: 4C CE 92 >411  ]LOOP    JMP   NILLM
90C8: A9 07    >413  :8       LDA   #7
90CA: 1C D8 99 >414           TRB   MOMODE
90CD: 8A       >415           TXA
90CE: 0C D8 99 >416           TSB   MOMODE
90D1: C9 02    >417           CMP   #2
90D3: A9 00    >426           LDA   #0
90D5: A8       >427           TAY
90D6: 90 02    >428           BCC   *+4
90D8: A9 02    >429  COMMON9  LDA   #2
90DA: 99 F6 99 >430           STA   MODEPEC,Y
90DD: AD D8 99 >431  COMMON   LDA   MOMODE
90E0: 48       >432           PHA
90E1: 20 FA 8E >433           JSR   INSIRQV
90E4: 68       >434           PLA
90E5: A2 00    >435           LDX   #RSETM
90E7: 20 39 91 >436           JSR   TOMOUSE
90EA: B0 D9    >437           BCS   ]LOOP
90EC: 20 22 8F >438           JSR   DINSIRQV
90EF: 4C D2 D7 >439           JMP   NEWSTT
               >440
```

```
90F2: C9 B4    >441  TIMEINST CMP   #$B4        TIMER ON
90F4: A9 08    >443           LDA   #8
90F6: 1C D8 99 >444           TRB   MOMODE
90F9: 90 E2    >445           BCC   COMMON
90FB: 0C D8 99 >446           TSB   MOMODE
90FE: 24 C0    >456           BIT   GFLAG
9100: 30 06    >457           BMI   *+8
9102: A2 01    >458           LDX   #1
9104: A0 00    >459           LDY   #0
9106: 10 04    >460           BPL   *+6         Always
9108: A6 A1    >461           LDX   FAC+4
910A: A4 A0    >462           LDY   FAC+3
910C: 08       >463           PHP
910D: 78       >464           SEI
910E: 8C 16 9A >465           STY   KTINC+1
9111: 8E 15 9A >466           STX   KTINC
9114: 8C 18 9A >467           STY   TIINC+1
9117: 8E 17 9A >468           STX   TIINC
911A: 28       >469           PLP
911B: A0 01    >470           LDY   #1
911D: B0 B9    >471           BCS   COMMON9     Always
               >472
               >473  * Do we have suitable mouse hardware?
911F: 20 C1 92 >474  COMINT4  JSR   ISMOUSH     Fall into SWREINIT if yes
               >475  * Routine below to check whether we should init the
               >476  * MOUSE system?
               >477  SWREINIT
9122: A9 80    >479           LDA   #$80
9124: 0C E1 99 >480           TSB   MONU
9127: D0 0C    >481           BNE   :0
               >488  * INITMOUSE was performed on Peersoft boot when in an
               >489  * Apple 2,2+ host.
9129: AD ED 9C >490           LDA   MACHINE
912C: F0 07    >491           BEQ   :0
912E: 5A       >492           PHY
912F: A2 07    >493           LDX   #RINI
9131: 20 39 91 >494           JSR   TOMOUSE
9134: 7A       >495           PLY
9135: 60       >496  :0       RTS
               >497
9136: 6C D6 99 >498  ]LOOP    JMP   (MVECTOR)
               >499
9139: BC CD 99 >500  TOMOUSE  LDY   OM_DEB,X
913C: AE D7 99 >501           LDX   MOCN
913F: 08       >502           PHP
9140: 78       >503           SEI
9141: 8C D6 99 >504           STY   MVECTOR
9144: AC D5 99 >505           LDY   MON0
9147: 20 36 91 >506           JSR   ]LOOP
914A: B0 03    >507           BCS   *+5
914C: 28       >508           PLP
914D: 18       >509           CLC
914E: 60       >510           RTS
914F: 28       >511           PLP
9150: 38       >512           SEC
9151: 60       >513           RTS
               >514
```

```
                >515  * Entry routine for MOUSE functions (either MOUSE or
                >516  * TIMER).
9152: 48        >517  MTFUNC    PHA
9153: 20 FB E6 >518            JSR   CONINT
9156: 20 71 8B >519            JSR   NCHKCLS
9159: 20 1F 91 >520            JSR   COMINT4
915C: 68        >521            PLA
915D: D0 31     >522            BNE   TFUNC
915F: A9 02     >523            LDA   #2
9161: 20 55 8F >524            JSR   IVALARG
9164: AE F4 99 >525            LDX   MODERUN
9167: D0 05     >526            BNE   *+7         Branch if within interrupt
9169: A2 02     >527            LDX   #RREAD
916B: 20 39 91 >528            JSR   TOMOUSE
916E: AE D1 9C >529            LDX   MOSL
9171: A5 A1     >531            LDA   FAC+4
9173: 3A        >532            DEC
9174: 10 09     >537            BPL   :1
9176: BD 78 05 >538            LDA   BAXHI,X     MOUSE(0) means read X
9179: BC 78 04 >539            LDY   BAXLO,X
917C: 4C F2 E2 >540  ]LOOP     JMP   GIVAYF
                >541  :1        DO    KOPT-K6502
917F: 3A        >542            DEC
9180: 10 08     >546            BPL   :2
9182: BD F8 05 >547            LDA   BAYHI,X     MOUSE(1) means read Y
9185: BC F8 04 >548            LDY   BAYLO,X
9188: 80 F2     >550            BRA   ]LOOP
918A: BC 78 07 >554  :2        LDY   BAMBS,X     MOUSE(2) means read buttons
918D: 4C 01 E3 >555            JMP   SNGFLT
9190: A9 01     >556  TFUNC     LDA   #1
9192: 20 55 8F >557            JSR   IVALARG
9195: 20 B9 92 >558            JSR   ISHOSTOK
9198: A2 00     >559            LDX   #0
919A: A5 A1     >560            LDA   FAC+4
919C: F0 02     >561            BEQ   *+4
919E: A2 02     >562            LDX   #2
91A0: BD 16 9A >563            LDA   KTINC+1,X
91A3: BC 15 9A >564            LDY   KTINC,X
91A6: 80 D4     >566            BRA   ]LOOP
                >570
                >571  * Desactive le traitement d´une interruption (sur RETURN)
                >572  * Y en entree: indice de l´interruption
91A8: A9 00     >573  COMINT1   LDA   #0
91AA: 99 F4 99 >574            STA   MODERUN,Y
91AD: 3A        >576            DEC
91AE: 8D F3 99 >580            STA   YICUR
                >581  * MODEPEC passe de STOP a ON
91B1: B9 F6 99 >583            LDA   MODEPEC,Y
91B4: C9 01     >584            CMP   #1
91B6: D0 04     >585            BNE   :0
91B8: 1A        >586            INC
91B9: 99 F6 99 >594            STA   MODEPEC,Y
91BC: B9 00 9A >595  :0        LDA   TPT_B,Y
91BF: 85 B8     >596            STA   TXTPTR
91C1: B9 02 9A >597            LDA   TPT_T,Y
91C4: 85 B9     >598            STA   TXTPTR+1
91C6: B9 FC 99 >599            LDA   CLN_B,Y
```

```
91C9: 85 75    >600           STA   CURLIN
91CB: B9 FE 99 >601           LDA   CLN_T,Y
91CE: 85 76    >602           STA   CURLIN+1
91D0: B9 04 9A >603           LDA   OTPT_B,Y
91D3: 85 79    >604           STA   OLDTEXT
91D5: B9 06 9A >605           LDA   OTPT_T,Y
91D8: 85 7A    >606           STA   OLDTEXT+1
91DA: AE E2 99 >607           LDX   SVMTACTV
91DD: AD F4 99 >608           LDA   MODERUN
91E0: 0D F5 99 >609           ORA   MODERUN+1
91E3: D0 06    >610           BNE   *+8
91E5: 8D E2 99 >611           STA   SVMTACTV
91E8: 8E DC 9C >612           STX   MTACTV
91EB: A0 05    >613           LDY   #5
91ED: CC 14 9A >614           CPY   FRGNDCTX
91F0: D0 05    >615           BNE   :1
91F2: 68       >616           PLA
91F3: 68       >617           PLA
91F4: 4C 28 93 >618           JMP   RW2
91F7: 60       >619  :1       RTS
               >620
               >621  * Routine en charge de determiner si l´interruption peut
               >622  * ou non etre cascadee.
               >623  * Sortie: bitN a 0 ssi possibilite de cascade (indice
               >624  * dans Y)
91F8: A0 01    >625  COMINT2  LDY   #1          On commence par la TIMER
91FA: B9 F8 99 >626  ]LOOP    LDA   MSKINT,Y
91FD: 08       >627           PHP               ;Sauve le interrupt enable
91FE: 78       >628           SEI               ;courant
91FF: 2D F2 99 >629           AND   MIRQST
9202: F0 27    >630           BEQ   :3
               >631  * Uniquement si prise en compte immediate..
9204: BE F6 99 >632           LDX   MODEPEC,Y
9207: E0 02    >633           CPX   #2
9209: D0 20    >634           BNE   :3
               >635  * Uniquement si routine non deja active
920B: BE F4 99 >636           LDX   MODERUN,Y
920E: D0 1B    >637           BNE   :3
9210: 1C F2 99 >639           TRB   MIRQST
9213: 28       >646           PLP
9214: A9 02    >647           LDA   #3-1        because from within a called subr
.
9216: 20 D6 D3 >648           JSR   CHKMEM
9219: 8C F3 99 >649           STY   YICUR
921C: AD DC 9C >650           LDA   MTACTV
921F: 8D E2 99 >651           STA   SVMTACTV
9222: A9 01    >652           LDA   #1
9224: 99 F6 99 >653           STA   MODEPEC,Y
9227: 99 F4 99 >654           STA   MODERUN,Y
922A: 60       >655           RTS
922B: 28       >656  :3       PLP
922C: 88       >657           DEY
922D: 10 CB    >658           BPL   ]LOOP
922F: 60       >659           RTS
               >660
               >661  * Retour d´une interruption souris
9230: A0 00    >662  RETOURM  LDY   #0
```

```
9232: 2C       >663           HEX   2C          Skip next two bytes
9233: A0 01    >664  RETOURT  LDY   #1
9235: BA       >665           TSX
9236: 86 F8    >666           STX   REMSTK
9238: 20 A8 91 >667           JSR   COMINT1
923B: 20 F1 85 >668           JSR   DECTPTR
923E: 20 58 D8 >669           JSR   ISCNTC
9241: 4C 05 D8 >670           JMP   TRACE
               >671
9244: AD F4 99 >672  RNEWINST LDA   MODERUN
9247: 0D F5 99 >673           ORA   MODERUN+1
924A: F0 19    >674           BEQ   RNI2
               >675  * Y a la bonne valeur selon MOUSE ou TIMER actifs
924C: AC F3 99 >676           LDY   YICUR
924F: 10 0A    >677           BPL   :1
9251: C8       >678           INY               ;Y passe de FF a 0
9252: AD F5 99 >679           LDA   MODERUN+1
9255: F0 01    >680           BEQ   *+3
9257: C8       >681           INY               ;Y passe a 1
9258: 8C F3 99 >682           STY   YICUR
925B: BA       >683  :1       TSX
925C: 8A       >684           TXA
               >685  * Routine terminee par RETURN/POP ayant ramene le SP
925D: D9 FA 99 >686           CMP   INTSPTR,Y
9260: 90 03    >687           BCC   RNI2
9262: 20 A8 91 >688           JSR   COMINT1
               >689  * ...
9265: AD F2 99 >690  RNI2     LDA   MIRQST
9268: F0 4C    >691           BEQ   :4
926A: 20 F8 91 >692           JSR   COMINT2
926D: 30 47    >693           BMI   :4          ;
               >694  * Reminder of current stack pointer
926F: BA       >695           TSX
9270: 8A       >696           TXA
9271: 99 FA 99 >697           STA   INTSPTR,Y
               >698  * Builds the GOSUB stack frame
9274: C0 01    >699           CPY   #1          carry set iif TIMER int.
9276: B0 06    >706           BCS   *+8
9278: A2 2F    >707           LDX   #RETOURM-1
927A: A9 92    >708           LDA   #>RETOURM-1
927C: D0 04    >709           BNE   *+6
927E: A2 32    >710           LDX   #RETOURT-1
9280: A9 92    >711           LDA   #>RETOURT-1
9282: 48       >712           PHA
9283: DA       >713           PHX
9284: A5 B9    >715           LDA   TXTPTR+1
9286: 99 02 9A >716           STA   TPT_T,Y
9289: 48       >717           PHA
928A: A5 B8    >718           LDA   TXTPTR
928C: 99 00 9A >719           STA   TPT_B,Y
928F: 48       >720           PHA
9290: A5 76    >721           LDA   CURLIN+1
9292: 99 FE 99 >722           STA   CLN_T,Y
9295: 48       >723           PHA
9296: A5 75    >724           LDA   CURLIN
9298: 99 FC 99 >725           STA   CLN_B,Y
929B: 48       >726           PHA
```

```
929C: A5 79              LDA   OLDTEXT
929E: 99 04 9A           STA   OTPT_B,Y
92A1: A5 7A              LDA   OLDTEXT+1
92A3: 99 06 9A           STA   OTPT_T,Y
92A6: A9 B0              LDA   #TOKGOSUB
92A8: 48                 PHA
                * and initialize the context for irq handler
                * (before falling into NEWSTT)
92A9: BE DF 99           LDX   AHNDHI,Y
92AC: B9 DD 99           LDA   AHNDLO,Y
92AF: 85 B8              STA   TXTPTR
92B1: 86 B9              STX   TXTPTR+1
92B3: 4C D2 D7           JMP   NEWSTT

92B6: 4C 82 8D  :4       JMP   RNEWISUI

92B9: AD ED 9C  ISHOSTOK LDA   MACHINE
92BC: C9 41              CMP   #$41       Enhanced 2e ROM pattern
92BE: 90 09              BCC   HNOK
92C0: 60        ]RET     RTS
92C1: AD D7 99  ISMOUSH  LDA   MOCN
92C4: D0 FA              BNE   ]RET
92C6: A2 20              LDX   #32
92C8: 2C                 HEX   2C         Skip next two byte
92C9: A2 21     HNOK     LDX   #33
92CB: 68        NERRHP   PLA              ;Pull return address
92CC: 68                 PLA
92CD: 2C                 HEX   2C
92CE: 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.
92D0: 24 D8     NERRH    BIT   ERRFLG
92D2: 10 03              BPL   *+5
92D4: 4C F9 E2           JMP   $E2F9      to ROM Error handler code
92D7: 20 FB DA           JSR   CRDO
92DA: 20 5A DB           JSR   $DB5A      Output question mark
92DD: BD 02 9B           LDA   CODR-32,X
92E0: AA                 TAX
92E1: BD 19 9A  ]LOOP    LDA   MESSERR,X
92E4: 48                 PHA
92E5: 20 5C DB           JSR   OUTDO
92E8: E8                 INX
92E9: 68                 PLA
92EA: 10 F5              BPL   ]LOOP
92EC: 4C 2A D4           JMP   $D42A      Fall into ROM code tail

92EF: 20 46 E7  RWAIT    JSR   $E746      Get address in LINNUM,
92F2: 86 85              STX   FORPNT      mask in X (saved)
92F4: A2 00              LDX   #0
92F6: 20 B7 00           JSR   $00B7
92F9: F0 03              BEQ   *+5
```

```
92FB: 20 4C E7 >784              JSR   COMBYTE
92FE: 86 86    >785              STX   FORPNT+1
               >789  COMWAIT
9300: AD F2 99 >790  ]LOOP       LDA   MIRQST
9303: D0 09    >791              BNE   :2
9305: B2 50    >793              LDA   (LINNUM)
9307: 45 86    >797              EOR   FORPNT+1
9309: 25 85    >798              AND   FORPNT
930B: F0 F3    >799              BEQ   ]LOOP
930D: 60       >800              RTS
930E: 20 F8 91 >801  :2          JSR   COMINT2
9311: 30 ED    >803              BMI   ]LOOP
9313: 5A       >809              PHY
9314: A0 05    >810              LDY   #5
9316: 8C 14 9A >811              STY   FRGNDCTX
9319: BE 08 9A >812  ]LOOP       LDX   SVWOF,Y
931C: B5 00    >813              LDA   0,X
931E: 99 0E 9A >814              STA   SVA,Y
9321: 88       >815              DEY
9322: 10 F5    >816              BPL   ]LOOP
9324: 7A       >817              PLY
9325: 4C 6F 92 >818              JMP   RNI2+10
               >819
9328: A0 06    >820  RW2         LDY   #6
932A: BE 07 9A >821  ]LOOP       LDX   SVWOF-1,Y
932D: B9 0D 9A >822              LDA   SVA-1,Y
9330: 95 00    >823              STA   0,X
9332: 88       >824              DEY
9333: D0 F5    >825              BNE   ]LOOP
9335: 8C 14 9A >826              STY   FRGNDCTX
9338: F0 C6    >827              BEQ   COMWAIT    Always
                1237
                1238 * Get address of array which name is pointed to by
                1239 * TXTPTR. If no array is found, then the called
                1240 * ROM routine would have created one so we´ll have
                1241 * to rollback such creation and exit.
                1242 NGTA2       DO    KOPT16
933A: A5 6E     1245             LDA   STREND+1
933C: 48        1246             PHA
933D: A5 6D     1247             LDA   STREND
933F: 48        1248             PHA
9340: 20 90 7E  1250             JSR   NGETARPT
9343: FA        1251             PLX
9344: 68        1252             PLA
9345: B0 04     1253             BCS   :1         found existing array
9347: 85 6E     1254             STA   STREND+1   Do the rollback
9349: 86 6D     1255             STX   STREND
                1256 :1          DO    KOPT-K65C02
934B: 64 14     1260             STZ   SUBFLG
934D: 60        1262             RTS
                1263
                1264             PUT   PEERGOTO
               >1    * Module in charge of accelerating GOTO/GOSUB line address
               >2    * computations.
               >3    TXTTAB   EQU   $67
               >4    TOKTHEN  =     $C4
               >5    GOTOTAIL EQU   $D95E
               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
934E: AD B6 AA NKBDINT  LDA   WHCBASIC
9351: F0 10             BEQ   :0
9353: 20 5E A6          JSR   ISBASRUN
9356: 90 0B             BCC   :0         program running
9358: AD D3 9C          LDA   OPTCGOTO
935B: 2D D2 9C          AND   NEEDDEC
935E: 10 03             BPL   :0
9360: 20 24 95          JSR   DECOMPILE
9363: AD B3 AA :0       LDA   EXFLG
9366: 60                RTS

               * New DOS Applesoft SAVE command handler (or part of)
9367: 20 24 95 NDSVCMD  JSR   DECOMPILE
936A: A9 02             LDA   #2         Restore original A value..
936C: 4C D5 A3          JMP   $A3D5      Fall into $A3D5 (orig. content)

               * Reset NEEDDEC upon DOS 3.3 Applesoft program loading
               NDLVCMD  DO    KOPT-K6502
936F: 9C D2 9C          STZ   NEEDDEC
9372: 4C 7A A4          JMP   RD2

9375: 9D DE 9B ROUT8C   STA   ADPFB,X
9378: 98                TYA
9379: 9D EF 9B          STA   ADPFT,X
937C: E8                INX
937D: 60                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.
937E: 20 74 8B ROUT8    JSR   NCHKCOM
9381: 20 84 8B          JSR   NGETBYT    Reason code in X
9384: 8E D3 9C          STX   OPTCGOTO
9387: 8A                TXA
9388: A2 0A             LDX   #10
938A: A8                TAY
938B: 10 16             BPL   :2
938D: A9 77             LDA   #RGOTO-1
938F: A0 94             LDY   #>RGOTO-1
9391: 20 75 93          JSR   ROUT8C
9394: A9 52             LDA   #RIF-1
9396: A0 94             LDY   #>RIF-1
9398: 20 75 93          JSR   ROUT8C
939B: E8                INX
939C: A9 30             LDA   #RGOSUB-1
939E: A0 94             LDY   #>RGOSUB-1
93A0: 20 75 93          JSR   ROUT8C
93A3: 2C D3 9C :2       BIT   OPTCGOTO
93A6: 30 18             BMI   :3
```

```
93A8: 08                PHP
93A9: A9 3D             LDA   #APRGOTO-1
93AB: A0 D9             LDY   #>APRGOTO-1
93AD: 20 75 93          JSR   ROUT8C
93B0: A9 C8             LDA   #APRIF-1
93B2: A0 D9             LDY   #>APRIF-1
93B4: 20 75 93          JSR   ROUT8C
93B7: E8                INX
93B8: A9 20             LDA   #APRGOSUB-1
93BA: A0 D9             LDY   #>APRGOSUB-1
93BC: 20 75 93          JSR   ROUT8C
93BF: 28                PLP
93C0: 70 02    :3       BVS   :0
93C2: 30 03             BMI   :4
93C4: 4C 24 95 :0       JMP   DECOMPILE  in case reason code 0 or 192
93C7: 60       :4       RTS

93C8: 4C C9 DE ]ERR     JMP   SYNERR
93CB: A2 01    RON      LDX   #1
93CD: 20 13 90          JSR   COMLBS
93D0: F0 35             BEQ   :1
               * Function call: normal flow
93D2: B1 B8             LDA   (TXTPTR),Y
93D4: C9 28             CMP   #´(´
93D6: F0 2F             BEQ   :1         Normal function
               * ON MOUSE GOSUB or ON TIMER GOSUB pattern
93D8: 20 98 D9          JSR   ADDON
93DB: A9 B0             LDA   #TOKGOSUB
93DD: 20 62 82          JSR   NSYNCHR
93E0: 20 7F 94          JSR   RGPART1    LOWTR: address of target line
93E3: A5 BD             LDA   IDMOCL
93E5: 38                SEC
93E6: E9 07             SBC   #OFFMOU-TOFFST
93E8: AA                TAX
93E9: A5 9B             LDA   LOWTR
93EB: E9 01             SBC   #1         Carry already set
93ED: 9D DD 99          STA   AHNDLO,X
93F0: A5 9C             LDA   LOWTR+1
93F2: E9 00             SBC   #0
93F4: 9D DF 99          STA   AHNDHI,X
93F7: A5 50             LDA   LINNUM
93F9: 9D D9 99          STA   CLNLO,X
93FC: A5 51             LDA   LINNUM+1
93FE: 9D DB 99          STA   CLNHI,X
9401: 20 F4 7B          JSR   RST102
9404: D0 C2             BNE   ]ERR
9406: 60                RTS
9407: 20 84 8B :1       JSR   NGETBYT
940A: C9 B0             CMP   #TOKGOSUB
940C: F0 04             BEQ   :2
940E: 49 AB             EOR   #TOKGOTO   TOKGOTO being < TOKGOSUB
9410: D0 B6             BNE   ]ERR       carry is already clear
9412: 08       :2       PHP
9413: C6 A1    ]LOOP    DEC   FAC+4
9415: D0 0D             BNE   :3
9417: 28                PLP
               * Carry set iif GOSUB, else GOTO (carry clear)
```

```
9418: 90 05    >124           BCC   :GOTO
941A: 20 EC 7B >125           JSR   RST100
941D: 80 12    >127           BRA   RGOSUB
941F: 20 EC 7B >131  :GOTO    JSR   RST100
9422: 80 54    >133           BRA   RGOTO
9424: 20 F9 95 >137  :3       JSR   LRST100
9427: 90 FB    >138           BCC   :3          Loop till not digit
9429: E0 2C    >139           CPX   #´,´
942B: F0 E6    >140           BEQ   ]LOOP
942D: 28       >141           PLP
942E: 4C C9 95 >142           JMP   NDATAN
               >143
9431: 08       >144  RGOSUB   PHP
9432: 48       >145           PHA
9433: A9 02    >146           LDA   #3-1        -1 because of PLA PLP below..
9435: 20 D6 D3 >147           JSR   CHKMEM
9438: 68       >148           PLA
9439: 28       >149           PLP
943A: 20 7F 94 >150           JSR   RGPART1
943D: A5 B9    >155           LDA   TXTPTR+1
943F: 48       >156           PHA
9440: A5 B8    >157           LDA   TXTPTR
9442: 48       >158           PHA
9443: A5 76    >159           LDA   CURLIN+1
9445: 48       >160           PHA
9446: A5 75    >161           LDA   CURLIN
9448: 48       >162           PHA
9449: A9 B0    >164           LDA   #TOKGOSUB
944B: 48       >165           PHA
944C: 38       >166           SEC
944D: 20 5E D9 >167           JSR   GOTOTAIL
9450: 4C D2 D7 >168           JMP   NEWSTT
               >169
9453: 20 7B DD >170  RIF      JSR   FRMEVL
9456: A5 9D    >171           LDA   FAC
9458: F0 0D    >172           BEQ   :20
945A: B2 B8    >174           LDA   (TXTPTR)
945C: C9 AB    >179           CMP   #TOKGOTO
945E: F0 13    >180           BEQ   :4
9460: C9 C4    >181           CMP   #TOKTHEN
9462: F0 0F    >182           BEQ   :4
9464: 4C 84 80 >183           JMP   SNERR
9467: 20 CC 95 >184  :20      JSR   NREMN
946A: 4C 98 D9 >185           JMP   ADDON
946D: 20 5D 90 >186  :3       JSR   LBS10
9470: 4C 28 D8 >187           JMP   $D828
9473: 20 EC 7B >188  :4       JSR   RST100
9476: B0 F5    >189           BCS   :3
               >190
9478: 20 7F 94 >191  RGOTO    JSR   RGPART1
947B: 38       >192           SEC
947C: 4C 5E D9 >193           JMP   GOTOTAIL
               >194
               >195  * First part of GOTO..
               >196  * Upon entry: A contains first target line no. char.,
               >197  * C clear iif this character is a numeric digit.
               >198  * Upon exit: LOWTR set to base adress of target line,
               >199  * LINNUM set to target line no.
947F: 90 2A    >200  RGPART1  BCC   :2          if num. digit then process it
9481: C9 20    >201           CMP   #$20
9483: 90 03    >202           BCC   :10
9485: 4C 84 80 >203  :11      JMP   SNERR
               >204  * Offset of target line from beginning of program
               >205  * already computed (value within program text).
9488: E9 1C    >206  :10      SBC   #$1D-1
948A: A8       >207           TAY
948B: C8       >208           INY
948C: B1 B8    >209           LDA   (TXTPTR),Y lo byte
948E: 18       >210           CLC
948F: 65 67    >211           ADC   TXTTAB     to absolute address lo byte
9491: 85 9B    >212           STA   LOWTR
9493: C8       >213           INY
9494: B1 B8    >214           LDA   (TXTPTR),Y hi byte
9496: 65 68    >215           ADC   TXTTAB+1   to absolute address
9498: 85 9C    >216           STA   LOWTR+1
949A: C8       >217           INY
949B: 5A       >221           PHY
949C: A0 02    >223           LDY   #2
949E: B1 9B    >224           LDA   (LOWTR),Y
94A0: 85 50    >225           STA   LINNUM
94A2: C8       >226           INY
94A3: B1 9B    >227           LDA   (LOWTR),Y
94A5: 85 51    >228           STA   LINNUM+1
94A7: 7A       >232           PLY
94A8: 4C 98 D9 >234           JMP   ADDON      Add Y to TXTPTR
94AB: A6 B8    >235  :2       LDX   TXTPTR     Backup TXTPTR
94AD: 86 06    >236           STX   AUXPTR     before calling LINGET
94AF: A6 B9    >237           LDX   TXTPTR+1
94B1: 86 07    >238           STX   AUXPTR+1
94B3: 20 0C DA >239           JSR   LINGET
               >240  * Now TXTPTR points to the first non numeric character
               >241  * following line no: computes the offset from current
               >242  * to stored position.
94B6: 20 CC 95 >243           JSR   NREMN      Compute Y offset to EOL
94B9: A5 76    >244           LDA   CURLIN+1
94BB: C5 51    >245           CMP   LINNUM+1
94BD: B0 0B    >246           BCS   :1
94BF: 98       >247           TYA
94C0: 38       >248           SEC
94C1: 65 B8    >249           ADC   TXTPTR
94C3: A6 B9    >250           LDX   TXTPTR+1
94C5: 90 07    >251           BCC   :3
94C7: E8       >252           INX
94C8: B0 04    >253           BCS   :3         Always
94CA: A5 67    >254  :1       LDA   TXTTAB
94CC: A6 68    >255           LDX   TXTTAB+1
94CE: 20 1A D6 >256  :3       JSR   FNDLIN
94D1: 90 4E    >257           BCC   GOUNDEF
94D3: 2C D3 9C >258           BIT   OPTCGOTO
94D6: 10 48    >259           BPL   :RET       Optimization deactivated
94D8: A5 B8    >260           LDA   TXTPTR
94DA: E5 06    >261           SBC   AUXPTR
94DC: A8       >262           TAY
               >263  * Y should be 3, 4 or 5 (line no from 100 to 99999)
```

```
94DD: A5 B9    >264              LDA   TXTPTR+1
94DF: E5 07    >265              SBC   AUXPTR+1
94E1: D0 3D    >266              BNE   :RET       hi byte must be zero
94E3: 88       >267              DEY
94E4: 88       >268              DEY
94E5: 88       >269              DEY
94E6: 30 38    >270              BMI   :RET       If Y was below 3
94E8: C0 03    >271              CPY   #3         If Y was above 5
94EA: B0 34    >272              BCS   :RET
94EC: 84 B5    >273              STY   YSAV       possible values: 0, 1 or 2
94EE: A5 9B    >274              LDA   LOWTR
94F0: 38       >275              SEC
94F1: E5 67    >276              SBC   TXTTAB
94F3: AA       >277              TAX
94F4: A5 9C    >278              LDA   LOWTR+1
94F6: E5 68    >279              SBC   TXTTAB+1   Leaves carry always set..
94F8: 2C D3 9C >280              BIT   OPTCGOTO
94FB: 50 0F    >281              BVC   :6         Configured to skip checks..
94FD: A8       >282              TAY              ;Set Z flag after BIT op
94FE: 20 C4 95 >283              JSR   COMRG
9501: F0 1D    >284              BEQ   :RET
9503: 8A       >285              TXA
9504: 20 C4 95 >286              JSR   COMRG
9507: F0 17    >287              BEQ   :RET
9509: 98       >288              TYA
950A: A4 B5    >289              LDY   YSAV
950C: C8       >290  :6          INY
950D: C8       >291              INY
950E: 91 06    >292              STA   (AUXPTR),Y
9510: 88       >293              DEY
9511: 8A       >294              TXA
9512: 91 06    >295              STA   (AUXPTR),Y
9514: 88       >296              DEY
9515: 98       >297              TYA
9516: 69 1C    >298              ADC   #$1D-1     Carry originally set
9518: 91 06    >299  ]LOOP       STA   (AUXPTR),Y
951A: 88       >300              DEY
951B: 10 FB    >301              BPL   ]LOOP
951D: 8C D2 9C >302              STY   NEEDDEC    Set Need Decompile indic.
9520: 60       >303  :RET        RTS
               >304
9521: 4C 7C D9 >305  GOUNDEF     JMP   $D97C
               >306
               >307  * Routine to restore things at their original state
               >308  * This routine should be called upon LIST or a SAVE
               >309  * command under DOS 3.3.
               >310  DECOMPILE
9524: 08       >311              PHP
9525: 48       >312              PHA
9526: 2C D2 9C >313              BIT   NEEDDEC
9529: 10 3F    >314              BPL   FINDEC
952B: A5 67    >315              LDA   TXTTAB
952D: A6 68    >316              LDX   TXTTAB+1
952F: A0 00    >317              LDY   #0
9531: 8C D2 9C >318              STY   NEEDDEC
9534: 85 06    >319  ]LOOP       STA   AUXPTR
9536: 86 07    >320              STX   AUXPTR+1
```

```
9538: 84 C0    >321            STY   GFLAG      Set b7 to 0
953A: 8A       >322            TXA
953B: F0 2D    >323            BEQ   FINDEC
953D: A0 03    >324            LDY   #3
953F: C8       >325  ]LOOP1    INY
9540: B1 06    >326  ]LOOP2    LDA   (AUXPTR),Y
9542: F0 1D    >327            BEQ   FINLIGNE
9544: C9 22    >328            CMP   #´"´
9546: D0 08    >329            BNE   :0
9548: AA       >330            TAX
9549: A5 C0    >331            LDA   GFLAG
954B: 49 80    >332            EOR   #$80
954D: 85 C0    >333            STA   GFLAG
954F: 8A       >334            TXA
9550: 24 C0    >335  :0        BIT   GFLAG
9552: 30 EB    >336            BMI   ]LOOP1
9554: C9 20    >337            CMP   #$20
9556: B0 E7    >338            BCS   ]LOOP1
9558: E9 1C    >339            SBC   #$1D-1
955A: 90 E3    >340            BCC   ]LOOP1
955C: 20 6D 95 >341            JSR   TRAITEOK
955F: F0 DF    >342            BEQ   ]LOOP2     Always
9561: A0 01    >343  FINLIGNE LDY    #1
9563: B1 06    >344            LDA   (AUXPTR),Y
9565: AA       >345            TAX
9566: B2 06    >347            LDA   (AUXPTR)
9568: 80 CA    >348            BRA   ]LOOP
956A: 68       >354  FINDEC    PLA
956B: 28       >355            PLP
956C: 60       >356            RTS
               >357
               >358  * A: 0, 1 or 2 depending of length of org target line no
               >359  * Y: offset from AUXPTR where first pattern byte appeared
               >360  * Carry: must be set upon entry
956D: 85 B4    >361  TRAITEOK STA    XSAV
956F: 5A       >362            PHY
9570: 98       >363            TYA
9571: 65 B4    >364            ADC   XSAV       Carry set upon entry
9573: A8       >365            TAY
9574: 18       >366            CLC
               >367  * Now Y: offset from AUXPTR where to get the
               >368  *         target line adress offset
               >369  * CLC (carry already clear after ADC above).
9575: B1 06    >384            LDA   (AUXPTR),Y or stick to 8bits arithmetic
9577: 65 67    >385            ADC   TXTTAB
9579: 85 9B    >386            STA   LOWTR
957B: C8       >387            INY
957C: B1 06    >388            LDA   (AUXPTR),Y
957E: 65 68    >389            ADC   TXTTAB+1
9580: 85 9C    >390            STA   LOWTR+1
9582: A0 03    >391            LDY   #3
9584: B1 9B    >392            LDA   (LOWTR),Y
9586: 85 9E    >393            STA   $9E
9588: 88       >394            DEY
9589: B1 9B    >395            LDA   (LOWTR),Y
958B: 85 9F    >396            STA   $9F
958D: A2 90    >398            LDX   #$90       Get line #
```

```
958F: 38                 SEC               ; in ASCII form
9590: 20 A0 EB           JSR   $EBA0       stored into $100
9593: 20 34 ED           JSR   FOUT
9596: 20 BB 95           JSR   CLENGTH     Length of string in X
9599: 86 B5              STX   YSAV
959B: 7A                 PLY
959C: A5 B4              LDA   XSAV
959E: 1A                 INC
959F: 1A                 INC
95A0: 1A                 INC
95A1: 38                 SEC
95A2: E5 B5              SBC   YSAV
95A4: AA                 TAX
95A5: F0 08              BEQ   :0
95A7: A9 30              LDA   #´0´
95A9: 91 06    ]LOOP     STA   (AUXPTR),Y
95AB: C8                 INY
95AC: CA                 DEX
95AD: D0 FA              BNE   ]LOOP
95AF: BD 00 01 :0        LDA   $0100,X
95B2: F0 06              BEQ   :RET
95B4: 91 06              STA   (AUXPTR),Y
95B6: C8                 INY
95B7: E8                 INX
95B8: D0 F5              BNE   :0          Always
95BA: 60       :RET      RTS

95BB: A2 FF    CLENGTH   LDX   #255
95BD: E8       ]LOOP     INX
95BE: BD 00 01           LDA   $0100,X
95C1: D0 FA              BNE   ]LOOP
95C3: 60                 RTS

               * Small subroutine to test for critical offset value
               * against insert into program text
95C4: F0 02    COMRG     BEQ   :0
95C6: 49 3A              EOR   #´:´
95C8: 60       :0        RTS

               CHARAC    EQU   $0D

95C9: A2 3A    NDATAN    LDX   #´:´
95CB: 2C                 HEX   2C          Skip next two bytes
95CC: A2 00    NREMN     LDX   #0
95CE: 86 0D              STX   CHARAC
95D0: A0 00              LDY   #0
95D2: 84 0E              STY   ENDCHR
95D4: A5 0E    ]LOOP1    LDA   ENDCHR      Trick to count for Quote Parity
95D6: A6 0D              LDX   CHARAC      Do not stop upon ´:´ within
95D8: 85 0D              STA   CHARAC      a string litteral
95DA: 86 0E              STX   ENDCHR
95DC: B1 B8    ]LOOP     LDA   (TXTPTR),Y
95DE: F0 18              BEQ   :RET
95E0: C5 0E              CMP   ENDCHR
95E2: F0 14              BEQ   :RET
95E4: C8                 INY
95E5: C9 22              CMP   #´"´
```

```
95E7: F0 EB    >464              BEQ   ]LOOP1
95E9: C9 20    >465              CMP   #´ ´
95EB: B0 EF    >466              BCS   ]LOOP
95ED: E9 1C    >467              SBC   #$1D-1     Substract $1D (carry clear)
95EF: 90 EB    >468              BCC   ]LOOP      Out of scope..
95F1: C8       >469              INY
95F2: C8       >473  ]LOOP1      INY
95F3: 3A       >477              DEC
95F4: 10 FC    >479              BPL   ]LOOP1
95F6: 30 E4    >480              BMI   ]LOOP      Always
95F8: 60       >481  :RET        RTS
               >482
95F9: 20 EC 7B >483  LRST100     JSR   RST100
95FC: AA       >484              TAX
95FD: 90 10    >485              BCC   :RET
95FF: E9 1D    >486              SBC   #$1D
9601: A8       >487              TAY
9602: 90 0A    >488              BCC   :RETS
9604: C0 03    >489              CPY   #3
9606: B0 07    >490              BCS   :RET
9608: C8       >491              INY
9609: C8       >492              INY
960A: 20 98 D9 >493              JSR   ADDON
960D: 24       >494              HEX   24         Skip next byte
960E: 38       >495  :RETS       SEC
960F: 60       >496  :RET        RTS
                1265
                1266 FCODE       EQU   *
                1267
                1268             PUT   PEERGDATA
9610: 00 00 00 >1    SVPTR       DS    18
9622: 00       >2    SVP2        DFB   0
               >3
9623: 00 00 00 >4    TABOFB      DFB   0,0,0,0,0,0,0,0
962B: 00 00 00 >5    TABOFT      DFB   0,0,0,0,0,0,0,0
9633: 00       >6    INDX        DFB   0
9634: 00       >7    SPROOT      DFB   0
9635: 00 00    >8    ITVADDR     DA    0          Adresse de la var. ITHREAD%
9637: F8 75 76 >9    P0OFFSET    DFB   REMSTK,CURLIN,CURLIN+1,TXTPTR,TXTPTR+1
963C: 79 7A    >10               DFB   OLDTEXT,OLDTEXT+1
               >11   PIOFFSET    EQU   *
963E: F4 F5 F6 >12               DFB   TXTPSV,TXTPSV+1,CURLSV,CURLSV+1,ERRNUM
9643: DF DA DB >13               DFB   ERRSTK,ERRLIN,ERRLIN+1,ERRPOS,ERRPOS+1
9648: D8       >14               DFB   ERRFLG
               >15   PEOFFSET    EQU   *
9649: C9 C6 C2 >16   TOKMOTIF    DFB   TOKMINUS,TOKNOT,TOKFN,TOKSCRN
               >17   TOKMTIFE
964D: CE DE 90 >22   TOKMPF      DA    $DECE,$DE90,$E354,$DEF9
9655: C8 C9 CA >24   TOKENS      DFB   TOKADD,TOKMINUS,TOKMUL,TOKDIV
               >25
9659: BE E7 A7 >27   FPROUTS     DA    FADD,FSUB,FMULT,FDIV
               >32
9661: 00 0C    >33   OFFST       DFB   HNDLEIAD-HNDLEIB,HNDLEIMI-HNDLEIB
9663: 1A 18    >34               DFB   HNDLEIMU-HNDLEIB,HNDLEIDV-HNDLEIB
               >35
9665: 00 00 00 >36   ADRSTRUCT   DS    11*LENREC
974C: F8       >37   SVOFST      DFB   REMSTK
```

```
974D: B8 B9    >38            DFB   TXTPTR,TXTPTR+1
974F: 75 76    >39            DFB   CURLIN,CURLIN+1
9751: 79 7A    >40            DFB   OLDTEXT,OLDTEXT+1
9753: F2       >41            DFB   TRCFLG
9754: A5 A6 A7 >42            DFB   ARG,ARG+1,ARG+2,ARG+3,ARG+4,$AA
               >43   FINOF    EQU   *
975A: 00 00 00 >44   SVAREA   DS    FINOF-SVOFST
               >45
9768: 00 00 00 >46   SVCURRM  DS    12
9774: 00 00 00 >47   SVALTNM  DS    12
               >48
               >49   * Structure juste pour la prise en compte lors du DEFUSR
9780: 00 00 00 >50   ]DEBUT   DS    8
               >51   ]FIN
9788: 80 97    >52   SDEF1    DA    ]DEBUT     pour VARTAB
978A: 80 97    >53            DA    ]DEBUT     pour ARYTAB
978C: 80 97    >54            DA    ]DEBUT     pour STREND
978E: 88 97    >55            DA    ]FIN       pour FRETOP
9790: 88 97    >56            DA    ]FIN       pour FRESPC
9792: 88 97    >57            DA    ]FIN       pour MEMSIZ
               >58
               >59   * Structure de stockage privee pour la derniere PF
               >60   * dynamique.
9794: 00 00 00 >61   ]DEBUT   DS    512
               >62   ]FIN
9994: 94 97    >63   SINITX   DA    ]DEBUT     pour VARTAB
9996: 94 97    >64            DA    ]DEBUT     pour ARYTAB
9998: 94 97    >65            DA    ]DEBUT     pour STREND
999A: 94 99    >66            DA    ]FIN       pour FRETOP
999C: 94 99    >67            DA    ]FIN       pour FRESPC
999E: 94 99    >68            DA    ]FIN       pour MEMSIZ
               >69
99A0: 00       >70   ISPFACT  DS    1          Dynamic PF active?
99A1: 00       >71   PFINDIC  DS    1          Last dynamic PF used..
99A2: 00       >72   PFINDX   DS    1          Current PF index..
               >73
               >74   * Cache structure for simple variables
99A3: 00       >75   SNCCH    DFB   0
99A4: 00 00 00 >81   SVN      DS    KSNCACH
99A8: 00 00 00 >82   SVNP1    DS    KSNCACH
99AC: 00 00 00 >83   SIT      DS    KSNCACH
99B0: 00 00 00 >84   SLTR     DS    KSNCACH
99B4: 00 00 00 >85   SLTRP1   DS    KSNCACH
               >87   * Cache structure for array variables
99B8: 00       >88   ANCCH    DFB   0
99B9: 00 00 00 >94   AVN      DS    KSNCACH
99BD: 00 00 00 >95   AVNP1    DS    KSNCACH
99C1: 00 00 00 >96   AIT      DS    KSNCACH
99C5: 00 00 00 >97   ALTR     DS    KSNCACH
99C9: 00 00 00 >98   ALTRP1   DS    KSNCACH
                1269          PUT   PEERMOTIDATA
               >1    * Data segment for the mouse/timer/interrupt module
               >2    * Mouse data (detected upon init)
               >3    * Offset table
99CD: 12 13 14 >4    OM_DEB   HEX   12131415161718
99D4: 19       >5    OM_INI   HEX   19
               >6
```

```
99D5: 00        MON0     DS    1
99D6: 00        MVECTOR  DS    1
99D7: 00        MOCN     DS    1


99D8: 01        MOMODE   DFB   1

99D9: 00 00     CLNLO    DS    2          Line # of inter. handler lo
99DB: 00 00     CLNHI    DS    2          Line # of inter. handler hi
99DD: 00 00     AHNDLO   DS    2          Address of Applesoft line lo
99DF: 00 00     AHNDHI   DS    2          Address of Applesoft line hi

99E1: 00        MONU     DS    1          0 till 1st MOUSE/TIMER instr
99E2: 00        SVMTACTV DS    1

99E3: 07 0F     MOETMSK  HEX   070F
99E5: 01 00     MOCMPVAL HEX   0100

99E7: 00 40 40  MSTATUS  HEX   0040404080C0C0C0
99EF: 00 00     OLDVECT  DA    0

99F1: 00        WORKPL1  DS    1
99F2: 00        MIRQST   DS    1
                * YICUR: indique quel est le dernier
                * handler d´interruption retenu
99F3: 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
99F4: 00 00     MODERUN  DS    2
                * MODEPEC:
                * 0: non prise en compte de l´interruption
                * 1: prise en compte retardee
                * 2: prise en compte immediate
99F6: 00 00     MODEPEC  DS    2
99F8: 40 80     MSKINT   HEX   4080
                * Values of S to cmp upon return from Applesoft
                * handling routine (usually RETURN)
99FA: 00 00     INTSPTR  DS    2

99FC: 00 00     CLN_B    DS    2          Interrupted line # lo byte
99FE: 00 00     CLN_T    DS    2          Interrupted line # hi byte
9A00: 00 00     TPT_B    DS    2          Interrupted text ptr lo byte
9A02: 00 00     TPT_T    DS    2          Interrupted text ptr hi byte
9A04: 00 00     OTPT_B   DS    2          Interrupted OLDTEXT lo byte
9A06: 00 00     OTPT_T   DS    2          Interrupted OLDTEXT hi byte

                * Offsets from page zero to save for WAIT
9A08: 50 51     SVWOF    DFB   LINNUM,LINNUM+1
9A0A: 85 86              DFB   FORPNT,FORPNT+1
9A0C: B8 B9              DFB   TXTPTR,TXTPTR+1
                * Save area for WAIT
9A0E: 00 00 00  SVA      DS    6
9A14: 00        FRGNDCTX DFB   0          5 pour WAIT
```

```
               >64
               >65   * KTINC factor for timer interrupt (default 1)
9A15: 01 00    >66   KTINC    DA    1           config. value for timer factor
9A17: 00 00    >67   TIINC    DA    0           runtime value for timer factor
               >68
               >69   * Error messages
               >70   MESSERR
               >71   MESER1   EQU   *-MESSERR
9A19: 4D 4F 55 >72            DCI   ´MOUSE HARDWARE NOT DETECTED´
               >73   MESER2   EQU   *-MESSERR
9A34: 55 4E 53 >74            DCI   ´UNSUPPORTED HARDWARE CONFIGURATION´
               >75   MESER3   EQU   *-MESSERR
9A56: 55 4E 4B >76            DCI   ´UNKNOWN APPLESOFT MOUSE EVENT HANDLER´
               >77   MESER4   EQU   *-MESSERR
9A7B: 55 4E 4B >78            DCI   ´UNKNOWN APPLESOFT TIMER EVENT HANDLER´
               >79   MESER5   EQU   *-MESSERR
9AA0: 49 4C 4C >80            DCI   ´ILLEGAL MOUSE MODE´
               >81   MESER6   EQU   *-MESSERR
9AB2: 49 4C 4C >82            DCI   ´ILLEGAL MOUSE OPERATION´
               >83   MESER7   EQU   *-MESSERR
9AC9: 5A 45 52 >84            DCI   ´ZERO TARGET ADDRESS´
               >85   MESER8   EQU   *-MESSERR
9ADC: 45 4D 42 >86            DCI   ´EMBEDDED PF NOT SUPPORTED IN THIS RELEASE´
               >87   MESER9   EQU   *-MESSERR
9B05: 49 4C 4C >88            DCI   ´ILLEGAL OP WHILE PF IS ACTIVE´
9B22: 00 1B 3D >89   CODR     DFB   MESER1,MESER2,MESER3,MESER4,MESER5,MESER6
9B28: B0 C3 EC >90            DFB   MESER7,MESER8,MESER9
                1270
                1271 * Table of new Peersoft commands
9B2B: C8        1272 TMOCL    DFB   TOKADD
9B2C: D0        1273          DFB   TOKEQUAL
9B2D: 00        1274          DFB   0
9B2E: C9        1275          DFB   TOKMINUS
9B2F: D0        1276          DFB   TOKEQUAL
9B30: 00        1277          DFB   0
9B31: CA        1278          DFB   TOKMUL
9B32: D0        1279          DFB   TOKEQUAL
9B33: 00        1280          DFB   0
9B34: CB        1281          DFB   TOKDIV
9B35: D0        1282          DFB   TOKEQUAL
9B36: 00        1283          DFB   0
9B37: 40        1284          ASC   ´@´
9B38: 00        1285          DFB   0
9B39: 4F 46 46  1286          ASC   ´OFF´
9B3C: 00        1287          DFB   0
9B3D: 49        1288 IFIIF    ASC   ´I´
9B3E: AD        1289          DFB   TOKIF
9B3F: 00        1290          DFB   0
9B40: 4D 4F 55  1291          ASC   ´MOUSE´
9B45: 00        1292          DFB   0
9B46: 54 49 4D  1293          ASC   ´TIMER´
9B4B: 00        1294          DFB   0
9B4C: B8        1295 IFDEF    DFB   TOKDEF
9B4D: D5        1296          DFB   TOKUSR
9B4E: 00        1297          DFB   0
9B4F: B8        1298          DFB   TOKDEF
9B50: 53 54 52  1299          ASC   ´STR´
```

```
9B53: 00        1300          DFB   0
9B54: B8        1301          DFB   TOKDEF
9B55: 53 4E 47  1302          ASC   ´SNG´
9B58: 00        1303          DFB   0
9B59: B8        1304          DFB   TOKDEF
9B5A: D3        1305          DFB   TOKINT
9B5B: 00        1306          DFB   0
9B5C: B8        1308          DFB   TOKDEF
9B5D: 42 59 54  1309          ASC   ´BYTE´
9B61: 00        1310          DFB   0
9B62: FF        1312          HEX   FF
                1313
9B63: 00 03 06  1314 TOFFST   DFB   0,3,6,9      Pour les 4 syntax schemes
                1315          ERR   NOPER-4
9B67: 0C        1316          DFB   12           Pour le symbole @
9B68: 0E        1317 OFFOFF   DFB   14           Pour le mot cle OFF
9B69: 12        1318 OFFIIF   DFB   18           Pour la fonction IIF()
9B6A: 15        1319 OFFMOU   DFB   21           Pour le mot cle MOUSE
9B6B: 1B        1320 OFFTIM   DFB   27           Pour le mot cle TIMER
9B6C: 21        1321 OFFUSR   DFB   33           Pour le mot cle DEFUSR
9B6D: 24 29 2E  1322 OFFDEF   DFB   36,41,46     pour les intr. DEFSTR,SNG,INT...
9B70: 31        1324          DFB   49           Pour le DEFBYTE
9B71: 37        1325          DFB   55
                1329
                1330 * Ou commencer la recherche?
                1331 * au debut (LIST)
9B72: FF        1332 TIDMOCL  DFB   0-1
                1333 * instruction DEF<pattern>
9B73: 08        1334          DFB   OFFUSR-TOFFST-1
                1335 * sur la premiere fonction (IIF/MOUSE/TIMER)
9B74: 05        1336          DFB   OFFIIF-TOFFST-1
                1337 * fonction MOUSE ou TIMER
9B75: 06        1338          DFB   OFFMOU-TOFFST-1
9B76: 07        1339          DFB   OFFTIM-TOFFST-1
                1340 * Juste mot-cle OFF
9B77: 04        1341          DFB   OFFOFF-TOFFST-1
                1342 * Quoi mettre a l´offset OFFDEF
9B78: B8        1343 TOFFIN   DFB   TOKDEF       si LIST
9B79: B8        1344          DFB   TOKDEF       si DEF<pattern>
9B7A: FF        1345          HEX   FF           si IIF/MOUSE/TIMER
9B7B: FF        1346          HEX   FF           si MOUSE/TIMER
9B7C: FF        1347          HEX   FF           si TIMER
9B7D: FF        1348          HEX   FF           si OFF
                1349 * Quoi mettre a l´offset OFFIFF
9B7E: 49        1350 TOFFIN2  DFB   ´I´          ;si LIST
9B7F: 49        1351          DFB   ´I´          ;si DEF<pattern>
9B80: 49        1352          DFB   ´I´          ;si IFF/MOUSE/TIMER
9B81: 49        1353          DFB   ´I´          ;si MOUSE/TIMER
9B82: 49        1354          DFB   ´I´          ;si TIMER
9B83: FF        1355          HEX   FF           si OFF
9B84: 24 21 25  1356 MOTIF    ASC   ($!%(
9B87: 2E        1358          ASC   (.(
9B88: 00 00 82  1359 TITVAL   HEX   00008281     What to store into INTTYP
9B8C: FF 00 00  1360 TVTVAL   HEX   FF000000     What to store into VALTYP
9B90: 00 00 80  1361 TVNORA   HEX   00008080     Value to ORA with VARNAM
9B94: 80 00 80  1362 TVN1ORA  HEX   80008080     Value to ORA with VARNAM+1
                1368
```

```
9B98: 91 80 00  1370 NEG65536 HEX   9180000000
9B9D: 90 80 00  1372 NEG32768 HEX   9080000000
                1373
9BA2: 21 21 21  1374 TYPLET   DS    26,´!´
                1375
                1376 * Applesoft standard instructions entry points
                1377 APRWAIT  EQU   $E784      WAIT instruction entry point
                1378 APRRUN   EQU   $D912      RUN instruction entry point
                1379 APRLIST  EQU   $D6A5      LIST instruction entry point
                1380 APRCLEAR EQU   $D66A      CLEAR instruction entry point
                1381 APRDEF   EQU   $E313      DEF instruction entry point
                1382 APRLET   EQU   $DA46      LET instruction entry point
                1383 APRFOR   EQU   $D766      FOR instruction entry point
                1384 APRNEXT  EQU   $DCF9      NEXT instruction entry point
                1385 APFRMELM EQU   $DE67      Return address from FRMELM
                1386 APRETURN EQU   $D96B      RETURN/POP instr. entry point
                1387 APRONERR EQU   $F2CB      ONERR instruction entry point
                1388 APRNEW   EQU   $D649      NEW instruction entry point
                1389 APRGOTO  EQU   $D93E      GOTO instruction entry point
                1390 APRGOSUB EQU   $D921      GOSUB instruction entry point
                1391 APRIF    EQU   $D9C9      IF instruction entry point
                1392 APRON    EQU   $D9EC      ON expr GOTO/GOSUB entry point
                1393
9BBC: 83        1394 ADAPFBET DFB   APRWAIT-1
9BBD: 11 48 A4  1395          DFB   APRRUN-1,APRNEW-1,APRLIST-1,APRCLEAR-1
9BC1: CA 12 45  1396          DFB   APRONERR-1,APRDEF-1,APRLET-1
9BC4: 65 EB 3D  1397          DFB   APRFOR-1,APRON-1,APRGOTO-1,APRIF-1,APRETURN-
1,APRGOSUB-1
9BCA: F8 66     1398          DFB   APRNEXT-1,APFRMELM-1
9BCC: 1F        1399          DFB   $D820-1
9BCD: E7        1400 ADAPFTET DFB   >APRWAIT-1
9BCE: D9 D6 D6  1401          DFB   >APRRUN-1,>APRNEW-1,>APRLIST-1,>APRCLEAR-1
9BD2: F2 E3 DA  1402          DFB   >APRONERR-1,>APRDEF-1,>APRLET-1
9BD5: D7 D9 D9  1403          DFB   >APRFOR-1,>APRON-1,>APRGOTO-1,>APRIF-1,>APRE
TURN-1,>APRGOSUB-1
9BDB: DC DE     1404          DFB   >APRNEXT-1,>APFRMELM-1
9BDD: D8        1405          DFB   >$D820-1
9BDE: EE        1406 ADPFB    DFB   RWAIT-1
9BDF: 46 4F 48  1407          DFB   RRUN-1,RNEW-1,STDLIS-1,RCLEAR-1
9BE3: C7 5F 3C  1408          DFB   RONERR-1,RDEF-1,RLET-1
9BE6: AA CA 77  1409          DFB   RFOR-1,RON-1,RGOTO-1,RIF-1,RRETURN-1,RGOSUB-
1
9BEC: E5 65     1410          DFB   RNEXT-1,FRMELM-1
9BEE: 43        1411          DFB   RNEWINST-1
9BEF: 92        1412 ADPFT    DFB   >RWAIT-1
9BF0: 82 82 87  1413          DFB   >RRUN-1,>RNEW-1,>STDLIS-1,>RCLEAR-1
9BF4: 88 85 7C  1414          DFB   >RONERR-1,>RDEF-1,>RLET-1
9BF7: 86 93 94  1415          DFB   >RFOR-1,>RON-1,>RGOTO-1,>RIF-1,>RRETURN-1,>R
GOSUB-1
9BFD: 88 8A     1416          DFB   >RNEXT-1,>FRMELM-1
9BFF: 92        1417          DFB   >RNEWINST-1
                1418 FIN
                1419 LONGLANG EQU   *-CGARBAG
                1420          ERR   *-$9C00
                1421
                1422          PUT   PEERGLOBALPAGE
               >1             DUMMY $9CC0
```

```
9CC0: 00       >2    FLGFN    DS    1
9CC1: 00 00 00 >3    WRKFA    DS    5          FAC work area A
9CC6: 00 00 00 >4    WRKFB    DS    5          FAC work area B
9CCB: 00 00 00 >5    WRKFC    DS    5          FAC work area C
9CD0: 50       >6    SVNUM    HEX   50         Subversion number..
9CD1: 00       >7    MOSL     DS    1          Mouse slot (b7 set to 1 if none)
9CD2: 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
9CD3: 80       >13   OPTCGOTO HEX   80
               >14   * Some vectors
9CD4: DE 7F    >15   VNARRG91 DA    NARRGL91   Look up array name in memory
9CD6: FE 7E    >16   VNPTRG90 DA    NPTRGL90   Look up variable name in memory
               >17   * MT parameters
9CD8: 23 96    >18   ADADR    DA    TABOFB
9CDA: 00       >19   INHACTV  DFB   0          b7 set if switching inhibited
9CDB: 00       >20   CTRACTV  DFB   0          Counter run value
9CDC: 00       >21   MTACTV   DFB   0          b7 set if MT active
9CDD: 00       >22   ICTRACTV DFB   0          Number of ticks between 2 CTS
               >23   * General purpose constants
9CDE: 15       >24   PVERSION DFB   VERSION    Peersoft version number
9CDF: 4C B4 8B >25   REVECTOR JMP   ROUTGEN    Vector to utility routine
               >26            ERR   *-$9CE2    Must coincide with Bananasoft
               >27            DEND
               >28            DUMMY $9CED
9CED: 00       >29   MACHINE  DS    1
9CEE: 00       >30            DS    1          CPU
9CEF: 00       >31   MEMORY   DS    1
9CF0: 00       >32   VID80C   DS    1
               >33            DEND


--End assembly, 9481 bytes, Errors: 0


Symbol table - alphabetical order:

   A1L     =$3C        A2L     =$3E        A4L     =$42        ABSOL8  =$7DC6
   ABSOLUTE=$7E80   ?  ACTR    =$9B        ADADR   =$9CD8      ADAPFBET=$9BBC
   ADAPFTET=$9BCD      ADB1    =$56C7      ADB2    =$56DB      ADDON   =$D998
   ADPFB   =$9BDE      ADPFT   =$9BEF      ADRSTRUCT=$9665      ADRUSR  =$01
   ADT1    =$56D1      ADT2    =$56E5      AHNDHI  =$99DF      AHNDLO  =$99DD
   AIT     =$99C1      ALKCACH =$81B2      ALTR    =$99C5      ALTRP1  =$99C9
   ANCCH   =$99B8      APFRMELM=$DE67      APRCLEAR=$D66A      APRDEF  =$E313
   APRETURN=$D96B      APRFOR  =$D766      APRGOSUB=$D921      APRGOTO =$D93E
   APRIF   =$D9C9      APRLET  =$DA46      APRLIST =$D6A5      APRNEW  =$D649
   APRNEXT =$DCF9      APRON   =$D9EC      APRONERR=$F2CB      APRRUN  =$D912
   APRWAIT =$E784      ARET    =$7E71      ARET8   =$7DB9      ARG     =$A5
   AROMBA  =$59D4      ARYPNT  =$94        ARYTAB  =$6B        AUXBANK =$BF
   AUXPTR  =$06        AVN     =$99B9      AVNP1   =$99BD      AXARTAB =$BFDB
   AXARYPNT=$BFDB      AXARYPT2=$BFE2      AXHIMEM =$BF00      AXOFFSET=$BFDF
   AXRTMAIN=$BFD9      AXSTREND=$BFDD      AXSZ    =$BFDF      AXVALUE =$BFE2
   AYINT   =$E10C      BADNAM  =$7EBD      BAMBS   =$0778      BANCLD  =$861C
   BAXHI   =$0578      BAXLO   =$0478      BAYHI   =$05F8      BAYLO   =$04F8
   BIGRECON=$56EF      BISVTYP =$BE        CALLFUNC=$8B5D      CFA     =$5789
```

```
    CFM     =$5785      CGARBAG =$7BCB      CH      =$24        CHARAC  =$0D
    CHKMEM  =$D3D6      CHKNUM  =$DD6A      CHKSTR  =$DD6C      CLENGTH =$95BB
    CLNHI   =$99DB      CLNLO   =$99D9      CLN_B   =$99FC      CLN_T   =$99FE
    CMPCLAMP=$8F3C      CNVT1   =$81AA      CODE1   =$58ED      CODE2   =$59D4
    CODR    =$9B22      COLLECTR=$8509      COMBYTE =$E74C      COMCLAMP=$8F5F
    COMCLEAR=$8F94      COMCMPLX=$8B51      COMINT1 =$91A8      COMINT2 =$91F8
    COMINT4 =$911F      COMLBS  =$9013      COMLET2 =$833E      COMLISO =$8886
    COMMON  =$90DD      COMMON9 =$90D8      COMMONG =$841A      COMPOFST=$8295
    COMPOS  =$8FC7      COMREAD =$8F99      COMREST =$84FB      COMRG   =$95C4
    COMRST  =$7BF4      COMRSTC =$7BFC      COMWAIT =$9300      COMX1   =$8306
    CONINT  =$E6FB      COPYROM =$57FD      CRDO    =$DAFB      CTRACTV =$9CDB
    CURLIN  =$75        CURLSV  =$F6        DATA    =$D995      DATA1IDX=$578D
    DATA1VAL=$5793      DATAN   =$D9A3      DBUFP   =$9D00      DEBUTGET=$7BCB
    DEBUTGOT=$7C1A      DECOMPILE=$9524      DECTPTR =$85F1      DEFFLG  =$C1
    DEST    =$60        DIMFLG  =$10        DINSIRQV=$8F22      DIVEND  =$C2
    DIVSOR  =$C0        DSCTMP  =$9D        DVZERR8 =$7D72      DVZERROR=$7E10
    E06     =$8694      EK      =$5683      ELMSIZ  =$BFE1   MD EMOV    =$8000
    ENDCHR  =$0E        ENDRNG  =$876B    ? ERRDIR  =$E306      ERRFLG  =$D8
    ERRLIN  =$DA        ERRNUM  =$DE        ERRPOS  =$DC        ERRSTK  =$DF
    ERR_BSCR=$6B        ERR_RDIM=$78        ERR_SYNT=$10        EXFLG   =$AAB3
    EXPLIC? =$7EC8      FAC     =$9D        FACLO   =$A1        FACMO   =$A0
    FACSIGN =$A2        FADD    =$E7BE      FAE1    =$8133      FAE2    =$814E
    FAE3    =$814F      FCODE   =$9610      FDIV    =$EA66      FIN     =$9C00
    FINDEC  =$956A      FINLIGNE=$9561      FINMOUSE=$8F96      FINOF   =$975A
  ? FLGFN   =$9CC0      FMULT   =$E97F      FNDLIN  =$D61A      FNDVAR2 =$7BCB
    FORPNT  =$85        FOUT    =$ED34      FPROUTS =$9659    ? FREESPC =$71
    FRGNDCTX=$9A14      FRMELM  =$8A66      FRMELMLP=$8A63      FRMEVL  =$DD7B
    FRMNUM  =$DD67      FRMSTCK3=$DE20      FRSTIM  =$82B4      FSUB    =$E7A7
    GETADR  =$E752      GETARY  =$E0ED      GETARY2 =$E0EF      GETBYT  =$E6F8
    GFLAG   =$C0        GGO2TMER=$8C16      GIQERR2 =$81F0      GIVAYF  =$E2F2
    GME     =$814B      GNARRAY =$8022      GNPTRGET=$7C17      GODVZERR=$EAE1
    GOIQ    =$89FC      GOIQERR =$E199      GOOVFERR=$E8D5      GOSTLERR=$E5B2
  ? GOSVCUR =$82B0      GOSYNERR=$8269      GOTMIERR=$DD76   MD GOTO    =$8000
    GOTOTAIL=$D95E      GOUNDEF =$9521      GOVERROR=$7D23      GSE     =$8148
    GSNERR  =$86A8      GSNERR2 =$81ED      GSNERR3 =$85CD      GSTERROR=$7D29
    GTLT    =$7EAE      GTMERR2 =$81F3      GTMERROR=$7D26      HNDLEADR=$832B
    HNDLEDV =$7CE4      HNDLEIAD=$7CB2      HNDLEIB =$7CB2      HNDLEIC =$7CEE
    HNDLEIDV=$7CCA      HNDLEIMI=$7CBE      HNDLEIMU=$7CCC      HNDLEINT=$7CA2
  ? HNDLEIX =$7CE9      HNDLEIY =$7CA9    ? HNDLEREA=$7C92      HNDLESTR=$7CF8
    HNOK    =$92C9      ICTRACTV=$9CDD      IDMOCL  =$BD        IDX0    =$C0
    IFDEF   =$9B4C      IFIIF   =$9B3D      INDEX   =$5E        INDX    =$9633
    INHACTV =$9CDA      INITBF  =$58BC      INSDS2  =$F88C      INSIRQV =$8EFA
    INTSPTR =$99FA      INTTYP  =$12        INTTYPSV=$C7        IRQHDLR =$8E9B
    IRQV    =$03FE      ISAUXMEM=$81F6      ISBASRUN=$A65E      ISCNTC  =$D858
    ISHOSTOK=$92B9      ISLETC  =$E07D      ISMOUSH =$92C1      ISPFACT =$99A0
    ITVADDR =$9635      IVALARG =$8F55      K6502   =$00        K65816  =$01
    K65C02  =$01      ? KANCACH =$04        KILLEMAL=$8DB1      KNEW    =$01
    KNEW2   =$01        KOPT    =$01        KOPT16  =$00        KOPTLNG32=$01
    KOPTLNG33=$00        KSNCACH =$04        KTINC   =$9A15       KWELMSIZ=$8170
    KX3     =$8DAC      L08     =$87B0      L088    =$87AE      L3      =$8AD4
    LBS00   =$8234      LBS03   =$8709      LBS04   =$8C8B      LBS041  =$8D3A
    LBS05   =$896A      LBS051  =$896E      LBS06   =$8E7E    ? LBS061  =$8E80
    LBS10   =$905D      LENGTH  =$2F        LENREC  =$15        LET2    =$DA63
    LETINF  =$C0        LEVELPAR=$BD        LGSYNERR=$8A4B      LINGET  =$DA0C
    LINNUM  =$50        LISTED  =$87F0      LLOOP   =$7BEE      LN      =$56A1
    LN65536 =$8193      LONGLANG=$2035   M? LOOP    =$5500      LOWTR   =$9B
    LRST100 =$95F9      LST1LIN =$879C      LSTD?   =$879A      LTOKEN  =$8896
```

```
   MACHINE =$9CED      MACMAT  =$576F      MAINLIST=$8773      MC      =$5691
   MCAND   =$C0        MCODE   =$5777      MEMERR  =$D410      MEMORY  =$9CEF
   MESER1  =$00        MESER2  =$1B        MESER3  =$3D        MESER4  =$62
   MESER5  =$87        MESER6  =$99        MESER7  =$B0        MESER8  =$C3
   MESER9  =$EC        MESSERR =$9A19      MFIN    =$8B8E      MINSDS2 =$56B1
   MIRQST  =$99F2      MISLETC =$825C      MKNARRAY=$808C      MKNV    =$E09C
   MOCMPVAL=$99E5      MOCN    =$99D7      MODDAT  =$BF        MODEPEC =$99F6
   MODERUN =$99F4      MODREM  =$BE        MOETMSK =$99E3      MOMODE  =$99D8
   MON0    =$99D5      MONU    =$99E1      MOSL    =$9CD1      MOTIF   =$9B84
   MOUSEDET=$5799      MOVE    =$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  =$99F8      MSTATUS =$99E7      MTACTV  =$9CDC      MTFUNC  =$9152
MD MTSB    =$8000      MULTPLSS=$E2AD      MULTPLY1=$E2B6      MVECTOR =$99D6
   NAMFOUND=$7F5E      NAMNTFND=$7F3C      NARRAY  =$7F8F      NARRGL91=$7FDE
   NCHKCLS =$8B71      NCHKCOM =$8B74      NCHKOPN =$8B77      NCR     =$87DC
   NDATAN  =$95C9      NDLVCMD =$936F      NDSVCMD =$9367      NEEDDEC =$9CD2
   NEG32768=$9B9D      NEG65536=$9B98      NEG8    =$7DCA      NEGATE  =$7E84
   NERRH   =$92D0   ?  NERRHP  =$92CB      NEVAL   =$9008      NEVALC  =$8FFF
   NEWAYINT=$7D2F      NEWSTT  =$D7D2      NEWY    =$47        NEXT1   =$88EC
   NEXTC2  =$8E46      NEXTCTX =$8E2D      NFAEP   =$8127      NFRMEVL =$8B7C
   NFRMNUM =$8A5D      NFRMSTK2=$89A4      NGETARPT=$7E90      NGETARY =$81A0
   NGETBYT =$8B84      NGTA2   =$933A      NILLM   =$92CE      NKBDINT =$934E
   NMAKINT =$8183      NMOVINS =$7D1C      NOPER   =$04     ?  NOUVIN  =$864A
   NPARCHK =$8B6B      NPTRG   =$8FE3      NPTRGET =$7E96      NPTRGET1=$7E9C
   NPTRGETX=$8D51      NPTRGL90=$7EFE      NPTRGTX =$7E94      NREMN   =$95CC
   NRET    =$7E6F      NRET8   =$7DB7      NROUT   =$7D2C      NSYNCHR =$8262
   NSYNCHR2=$8262      NUMDIM  =$0F        NXLST   =$877F      OFFDEF  =$9B6D
   OFFIIF  =$9B69      OFFMOU  =$9B6A      OFFOFF  =$9B68      OFFSET  =$C2
   OFFST   =$9661      OFFTIM  =$9B6B      OFFUSR  =$9B6C      OKP1GET =$7C06
   OLDTEXT =$79        OLDTPTR =$79        OLDVECT =$99EF      OM_DEB  =$99CD
   OM_INI  =$99D4      OPRND   =$44        OPTCGOTO=$9CD3      OTPT_B  =$9A04
   OTPT_T  =$9A06      OUTDO   =$DB5C      OUTSPC  =$DB57      P0OFFSET=$9637
   PARTIAL =$BE        PCADJ   =$F953      PCL     =$3A     ?  PEOFFSET=$9649
   PFINDIC =$99A1      PFINDX  =$99A2      PIOFFSET=$963E      PVERSION=$9CDE
   QINT    =$EBF2      R       =$85CC      R0      =$8DC3   ?  RAZPF   =$826C
   RCLEAR  =$8256      RCLM    =$05     ?  RCLR    =$03        RD2     =$A47A
   RDEF    =$8560      RDEFSUB =$85C7      RDEFUSR =$8446      RDIM    =$89B3
   RDIMERR =$8087      REASON  =$D3E3      RECON   =$8716      RECON1  =$8712
?  RECON2  =$871A      REMSTK  =$F8        RESTOR  =$8DEE      RESTOR1 =$8DCD
   RESTOR2 =$8DD7      RESTORC =$8E10      RESTORD =$8DC7      RESTORF =$8E0F
?  RESTORX =$8E00      RESULT  =$62        RET1    =$7CF7      RET3    =$8AD1
   RETA8   =$7DBA      RETOUR  =$84DB      RETOURM =$9230      RETOURT =$9233
   RETURN  =$8748      REVECTOR=$9CDF      RFFVL   =$8AAB      RFOR    =$86AB
   RGOSUB  =$9431      RGOTO   =$9478      RGPART1 =$947F      RHOM    =$06
   RIF     =$9453      RIIF    =$8A4E      RINI    =$07        RLET    =$7C3D
   RLET1   =$7C44   ?  RMTCTRL =$8D87      RNEW    =$8250      RNEWINST=$9244
   RNEWISUI=$8D82      RNEXT   =$88E6      RNI2    =$9265      RON     =$93CB
   RONERR  =$88C8      ROUT0   =$8BDC      ROUT10  =$8F68      ROUT11  =$8B8F
?  ROUT1X  =$85D4      ROUT1Y  =$85D0      ROUT4   =$8C19      ROUT8   =$937E
   ROUT8C  =$9375      ROUTGEN =$8BB4      RPOS    =$04        RREAD   =$02
   RRETURN =$88BE      RRUN    =$8247      RSETM   =$00        RSRVM   =$01
   RST100  =$7BEC      RST101  =$7BEE      RST102  =$7BF4      RST103  =$7BF4
   RSTALTM =$853C      RSTCURRM=$8531      RUSR    =$8352      RVRAI   =$89FF
   RW2     =$9328      RWAIT   =$92EF      SAVALTM =$8552      SAVCURRM=$8547
   SAVER   =$8E54      SAVERC  =$8E8B      SCDCH2  =$7EC0      SCNDTIM =$8310
   SCTR    =$9B        SDEF1   =$9788      SDIV    =$7E13   ?  SDIV8   =$7D75
   SENDCHR =$87CC      SETINITX=$8287      SETITS  =$7CF3      SETLTR  =$8E24
   SETUPB  =$85FA      SETUPD  =$8611      SETVYA  =$E0DE      SINITX  =$9994
   SIT     =$99AC      SKIPC   =$8A07      SLKCACH =$7F61      SLTR    =$99B0
   SLTRP1  =$99B4   MD?SMOVE   =$8000      SMUL    =$7DD1   ?  SMUL8   =$7D3F
   SNCCH   =$99A3      SNERR   =$8084      SNGFLT  =$E301      SPROOT  =$9634
   STACK   =$0100   MD?STD     =$8000      STDLIS  =$8749      STEP    =$897C
MD STID    =$8000      STP1    =$86FE      STREND  =$6D        STRING1 =$AB
   STRNG1  =$AC        STRNG2  =$AD        STRSPA  =$E3DD      STRTRNG =$8755
   SUBERR  =$E196      SUBFLG  =$14        SUBSERR =$8081   ?  SUITE   =$5500
   SVA     =$9A0E      SVALTNM =$9774      SVAREA  =$975A      SVCURRM =$9768
   SVMTACTV=$99E2      SVN     =$99A4      SVNP1   =$99A8   ?  SVNUM   =$9CD0
   SVOFST  =$974C   ?  SVP2    =$9622      SVPTR   =$9610      SVWOF   =$9A08
   SWPIO   =$8E6F   ?  SWREINIT=$9122      SYNERR  =$DEC9      TABOFB  =$9623
   TABOFT  =$962B      TEST2E  =$5839      TFUNC   =$9190      TIDMOCL =$9B72
   TIINC   =$9A17      TIMEINST=$90F2      TITVAL  =$9B88      TMERR   =$DD76
   TMOCL   =$9B2B      TOFFIN  =$9B78      TOFFIN2 =$9B7E      TOFFST  =$9B63
   TOKADD  =$C8        TOKCHRD =$E7        TOKDATA =$83        TOKDEF  =$B8
   TOKDIM  =$86        TOKDIV  =$CB        TOKEN?  =$87F9      TOKENS  =$9655
   TOKEQUAL=$D0        TOKFN   =$C2        TOKGOSUB=$B0        TOKGOTO =$AB
   TOKIF   =$AD        TOKINT  =$D3        TOKMINUS=$C9        TOKMOTIF=$9649
   TOKMPF  =$964D      TOKMTIFE=$964D      TOKMUL  =$CA        TOKNOT  =$C6
   TOKREM  =$B2        TOKSCRN =$D7        TOKSGN  =$D2        TOKSTEP =$C7
   TOKSTRD =$E4        TOKTABL =$D0D0      TOKTHEN =$C4        TOKTO   =$C1
   TOKUSR  =$D5        TOMOUSE =$9139      TPT_B   =$9A00      TPT_T   =$9A02
   TRACE   =$D805      TRAITEOK=$956D      TRCFLG  =$F2        TVN1ORA =$9B94
   TVNORA  =$9B90      TVTVAL  =$9B8C      TXTPSV  =$F4        TXTPTR  =$B8
   TXTTAB  =$67        TYPLET  =$9BA2      TYPMOD  =$C1        ULERR   =$D97C
?  USDIV   =$7E3E   ?  USDIV8  =$7D92      USEOLDAR=$8025   ?  USMUL   =$7DD9
?  USMUL8  =$7D47   ?  USRMOD  =$00        VALTYP  =$11        VALTYPSV=$C8
   VARNAM  =$81        VARPNT  =$83        VARTAB  =$69        VECTUSR =$0A
?  VECZAUX =$03ED      VENT1IT =$0C        VENT1NAM=$09     ?  VENT1PTR=$0D
?  VENT1VT =$0B        VENT2IT =$12        VENT2NAM=$0F     ?  VENT2PTR=$13
?  VENT2VT =$11        VERSION =$15        VID80C  =$9CF0      VLET    =$DA46
   VLINPRT =$87F6      VNARRG91=$9CD4      VNPTRG90=$9CD6      VPNT    =$A0
   VPTRGET =$DFEF      VSRTIT  =$06        VSRTNAM =$03        VSRTPTR =$07
?  VSRTVT  =$05        WHCBASIC=$AAB6      WORKPL1 =$99F1   ?  WRKFA   =$9CC1
?  WRKFB   =$9CC6   ?  WRKFC   =$9CCB      XFER    =$C314   ?  XFRMMOT1=$85E3
   XFROMMOT=$85E6      XMFIN   =$8C5F      XMFIN1  =$8C88      XMFIN2  =$8C85
   XSAV    =$B4        YICUR   =$99F3      YSAV    =$B5        ZAUXB   =$BF1C
   ZAUXOFFT=$BFD6      ZAUXRET =$BFB9      ZAUXRT  =$BF00      ZAUXRT0 =$BF1C
   ZAUXRT1 =$BF3B      ZAUXRT2 =$BF65      ZAUXRT23=$BF92   ?  ZAUXRT3 =$BF83
   ZEROPRT =$7E73      ZPRT8   =$7DBB   ?  ZRTAUX  =$81E0   V  ]DEBUT  =$9794
V  ]ERR    =$93C8   V  ]ERR1   =$9038   V  ]FIN    =$9994   V  ]JLOOP  =$903E
V  ]LOOP   =$95DC   V  ]LOOP1  =$95F2   V  ]LOOP2  =$9540   V  ]RET    =$92C0


Symbol table - numerical order:

   K6502   =$00        KOPTLNG33=$00        KOPT16  =$00      ?  USRMOD  =$00
   RSETM   =$00        MESER1  =$00        K65C02  =$01        K65816  =$01
   KOPT    =$01        KNEW    =$01        KNEW2   =$01        KOPTLNG32=$01
   ADRUSR  =$01        RSRVM   =$01        RREAD   =$02        VSRTNAM =$03
?  RCLR    =$03        KSNCACH =$04     ?  KANCACH =$04        NOPER   =$04
   RPOS    =$04     ?  VSRTVT  =$05        RCLM    =$05        AUXPTR  =$06
   VSRTIT  =$06        RHOM    =$06        VSRTPTR =$07        RINI    =$07
   VENT1NAM=$09        VECTUSR =$0A     ?  VENT1VT =$0B        VENT1IT =$0C
```

```
?  VENT1PTR=$0D        CHARAC  =$0D        ENDCHR  =$0E        NUMDIM  =$0F
   VENT2NAM=$0F        DIMFLG  =$10        ERR_SYNT=$10        VALTYP  =$11
?  VENT2VT =$11        INTTYP  =$12        VENT2IT =$12     ?  VENT2PTR=$13
   SUBFLG  =$14        VERSION =$15        LENREC  =$15        MESER2  =$1B
   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     ?  FREESPC =$71
   CURLIN  =$75        ERR_RDIM=$78        OLDTPTR =$79        OLDTEXT =$79
   VARNAM  =$81        TOKDATA =$83        VARPNT  =$83        FORPNT  =$85
   TOKDIM  =$86        MESER5  =$87        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  =$D5        TOKSCRN =$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   ?  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=$2035   M? LOOP    =$5500   MD MPHX    =$8000
MD MPHY    =$8000   MD MPLX    =$8000   MD MPLY    =$8000   MD MTSB    =$8000
MD GOTO    =$8000   ?  SUITE   =$5500      EK      =$5683      MC      =$5691
   LN      =$56A1      MINSDS2 =$56B1      ADB1    =$56C7      ADT1    =$56D1
   ADB2    =$56DB      ADT2    =$56E5      BIGRECON=$56EF      MACMAT  =$576F
   MCODE   =$5777      CFM     =$5785      CFA     =$5789      DATA1IDX=$578D
   DATA1VAL=$5793      MOUSEDET=$5799      COPYROM =$57FD      TEST2E  =$5839
   INITBF  =$58BC      CODE1   =$58ED      CODE2   =$59D4      AROMBA  =$59D4
   FNDVAR2 =$7BCB      CGARBAG =$7BCB      DEBUTGET=$7BCB      RST100  =$7BEC
   RST101  =$7BEE      LLOOP   =$7BEE      RST102  =$7BF4      RST103  =$7BF4
   COMRST  =$7BF4      COMRSTC =$7BFC      OKP1GET =$7C06      GNPTRGET=$7C17
   DEBUTGOT=$7C1A      RLET    =$7C3D      RLET1   =$7C44   ?  HNDLEREA=$7C92
   HNDLEINT=$7CA2      HNDLEIY =$7CA9      HNDLEIB =$7CB2      HNDLEIAD=$7CB2
   HNDLEIMI=$7CBE      HNDLEIDV=$7CCA      HNDLEIMU=$7CCC      HNDLEDV =$7CE4
?  HNDLEIX =$7CE9      HNDLEIC =$7CEE      SETITS  =$7CF3      RET1    =$7CF7
   HNDLESTR=$7CF8      NMOVINS =$7D1C      GOVERROR=$7D23      GTMERROR=$7D26
   GSTERROR=$7D29      NROUT   =$7D2C      NEWAYINT=$7D2F   ?  SMUL8   =$7D3F
?  USMUL8  =$7D47      DVZERR8 =$7D72   ?  SDIV8   =$7D75   ?  USDIV8  =$7D92
   NRET8   =$7DB7      ARET8   =$7DB9      RETA8   =$7DBA      ZPRT8   =$7DBB
   ABSOL8  =$7DC6      NEG8    =$7DCA      SMUL    =$7DD1   ?  USMUL   =$7DD9
   DVZERROR=$7E10      SDIV    =$7E13   ?  USDIV   =$7E3E      NRET    =$7E6F
   ARET    =$7E71      ZEROPRT =$7E73      ABSOLUTE=$7E80      NEGATE  =$7E84
   NGETARPT=$7E90      NPTRGTX =$7E94      NPTRGET =$7E96      NPTRGET1=$7E9C
   GTLT    =$7EAE      BADNAM  =$7EBD      SCDCH2  =$7EC0      EXPLIC? =$7EC8
   NPTRGL90=$7EFE      NAMNTFND=$7F3C      NAMFOUND=$7F5E      SLKCACH =$7F61
   NARRAY  =$7F8F     NARRGL91=$7FDE     GNARRAY =$8022     USEOLDAR=$8025
   SUBSERR =$8081     SNERR   =$8084     RDIMERR =$8087     MKNARRAY=$808C
   NFAEP   =$8127     FAE1    =$8133     GSE     =$8148     GME     =$814B
   FAE2    =$814E     FAE3    =$814F     KWELMSIZ=$8170     NMAKINT =$8183
   LN65536 =$8193     NGETARY =$81A0     CNVT1   =$81AA     ALKCACH =$81B2
?  ZRTAUX  =$81E0     GSNERR2 =$81ED     GIQERR2 =$81F0     GTMERR2 =$81F3
   ISAUXMEM=$81F6     LBS00   =$8234     RRUN    =$8247     RNEW    =$8250
   RCLEAR  =$8256     MISLETC =$825C     NSYNCHR =$8262     NSYNCHR2=$8262
   GOSYNERR=$8269  ?  RAZPF   =$826C     SETINITX=$8287     COMPOFST=$8295
?  GOSVCUR =$82B0     FRSTIM  =$82B4     COMX1   =$8306     SCNDTIM =$8310
   HNDLEADR=$832B     COMLET2 =$833E     RUSR    =$8352     COMMONG =$841A
   RDEFUSR =$8446     RETOUR  =$84DB     COMREST =$84FB     COLLECTR=$8509
   RSTCURRM=$8531     RSTALTM =$853C     SAVCURRM=$8547     SAVALTM =$8552
   RDEF    =$8560     RDEFSUB =$85C7     R       =$85CC     GSNERR3 =$85CD
   ROUT1Y  =$85D0  ?  ROUT1X  =$85D4  ?  XFRMMOT1=$85E3     XFROMMOT=$85E6
   DECTPTR =$85F1     SETUPB  =$85FA     SETUPD  =$8611     BANCLD  =$861C
?  NOUVIN  =$864A     E06     =$8694     GSNERR  =$86A8     RFOR    =$86AB
   STP1    =$86FE     LBS03   =$8709     RECON1  =$8712     RECON   =$8716
?  RECON2  =$871A     RETURN  =$8748     STDLIS  =$8749     STRTRNG =$8755
   ENDRNG  =$876B     MAINLIST=$8773     NXLST   =$877F     LSTD?   =$879A
   LST1LIN =$879C     L088    =$87AE     L08     =$87B0     SENDCHR =$87CC
   NCR     =$87DC     LISTED  =$87F0     VLINPRT =$87F6     TOKEN?  =$87F9
   COMLISO =$8886     LTOKEN  =$8896     RRETURN =$88BE     RONERR  =$88C8
   RNEXT   =$88E6     NEXT1   =$88EC     LBS05   =$896A     LBS051  =$896E
   STEP    =$897C     NFRMSTK2=$89A4     RDIM    =$89B3     GOIQ    =$89FC
   RVRAI   =$89FF     SKIPC   =$8A07     LGSYNERR=$8A4B     RIIF    =$8A4E
   NFRMNUM =$8A5D     FRMELMLP=$8A63     FRMELM  =$8A66     RFFVL   =$8AAB
   RET3    =$8AD1     L3      =$8AD4     COMCMPLX=$8B51     CALLFUNC=$8B5D
   NPARCHK =$8B6B     NCHKCLS =$8B71     NCHKCOM =$8B74     NCHKOPN =$8B77
   NFRMEVL =$8B7C     NGETBYT =$8B84     MFIN    =$8B8E     ROUT11  =$8B8F
   ROUTGEN =$8BB4     ROUT0   =$8BDC     GGO2TMER=$8C16     ROUT4   =$8C19
   XMFIN   =$8C5F     XMFIN2  =$8C85     XMFIN1  =$8C88     LBS04   =$8C8B
   LBS041  =$8D3A     NPTRGETX=$8D51     RNEWISUI=$8D82  ?  RMTCTRL =$8D87
   KX3     =$8DAC     KILLEMAL=$8DB1     R0      =$8DC3     RESTORD =$8DC7
   RESTOR1 =$8DCD     RESTOR2 =$8DD7     RESTOR  =$8DEE  ?  RESTORX =$8E00
   RESTORF =$8E0F     RESTORC =$8E10     SETLTR  =$8E24     NEXTCTX =$8E2D
   NEXTC2  =$8E46     SAVER   =$8E54     SWPIO   =$8E6F     LBS06   =$8E7E
?  LBS061  =$8E80     SAVERC  =$8E8B     IRQHDLR =$8E9B     INSIRQV =$8EFA
   DINSIRQV=$8F22     CMPCLAMP=$8F3C     IVALARG =$8F55     COMCLAMP=$8F5F
   ROUT10  =$8F68     COMCLEAR=$8F94     FINMOUSE=$8F96     COMREAD =$8F99
   COMPOS  =$8FC7     NPTRG   =$8FE3     NEVALC  =$8FFF     NEVAL   =$9008
   COMLBS  =$9013  V  ]ERR1   =$9038  V  ]JLOOP  =$903E     LBS10   =$905D
   COMMON9 =$90D8     COMMON  =$90DD     TIMEINST=$90F2     COMINT4 =$911F
?  SWREINIT=$9122     TOMOUSE =$9139     MTFUNC  =$9152     TFUNC   =$9190
   COMINT1 =$91A8     COMINT2 =$91F8     RETOURM =$9230     RETOURT =$9233
   RNEWINST=$9244     RNI2    =$9265     ISHOSTOK=$92B9  V  ]RET    =$92C0
   ISMOUSH =$92C1     HNOK    =$92C9  ?  NERRHP  =$92CB     NILLM   =$92CE
   NERRH   =$92D0     RWAIT   =$92EF     COMWAIT =$9300     RW2     =$9328
   NGTA2   =$933A     NKBDINT =$934E     NDSVCMD =$9367     NDLVCMD =$936F
   ROUT8C  =$9375     ROUT8   =$937E  V  ]ERR    =$93C8     RON     =$93CB
   RGOSUB  =$9431     RIF     =$9453     RGOTO   =$9478     RGPART1 =$947F
   GOUNDEF =$9521     DECOMPILE=$9524  V  ]LOOP2  =$9540      FINLIGNE=$9561
   FINDEC  =$956A     TRAITEOK=$956D     CLENGTH =$95BB     COMRG   =$95C4
   NDATAN  =$95C9     NREMN   =$95CC  V  ]LOOP   =$95DC  V  ]LOOP1  =$95F2
   LRST100 =$95F9     FCODE   =$9610     SVPTR   =$9610  ?  SVP2    =$9622
   TABOFB  =$9623     TABOFT  =$962B     INDX    =$9633     SPROOT  =$9634
   ITVADDR =$9635     P0OFFSET=$9637     PIOFFSET=$963E  ?  PEOFFSET=$9649
```

```
    TOKMOTIF=$9649      TOKMTIFE=$964D      TOKMPF  =$964D      TOKENS  =$9655
    FPROUTS =$9659      OFFST   =$9661      ADRSTRUCT=$9665      SVOFST  =$974C
    FINOF   =$975A      SVAREA  =$975A      SVCURRM =$9768      SVALTNM =$9774
    SDEF1   =$9788  V   ]DEBUT  =$9794  V   ]FIN    =$9994      SINITX  =$9994
    ISPFACT =$99A0      PFINDIC =$99A1      PFINDX  =$99A2      SNCCH   =$99A3
    SVN     =$99A4      SVNP1   =$99A8      SIT     =$99AC      SLTR    =$99B0
    SLTRP1  =$99B4      ANCCH   =$99B8      AVN     =$99B9      AVNP1   =$99BD
    AIT     =$99C1      ALTR    =$99C5      ALTRP1  =$99C9      OM_DEB  =$99CD
    OM_INI  =$99D4      MON0    =$99D5      MVECTOR =$99D6      MOCN    =$99D7
    MOMODE  =$99D8      CLNLO   =$99D9      CLNHI   =$99DB      AHNDLO  =$99DD
    AHNDHI  =$99DF      MONU    =$99E1      SVMTACTV=$99E2      MOETMSK =$99E3
    MOCMPVAL=$99E5      MSTATUS =$99E7      OLDVECT =$99EF      WORKPL1 =$99F1
    MIRQST  =$99F2      YICUR   =$99F3      MODERUN =$99F4      MODEPEC =$99F6
    MSKINT  =$99F8      INTSPTR =$99FA      CLN_B   =$99FC      CLN_T   =$99FE
    TPT_B   =$9A00      TPT_T   =$9A02      OTPT_B  =$9A04      OTPT_T  =$9A06
    SVWOF   =$9A08      SVA     =$9A0E      FRGNDCTX=$9A14      KTINC   =$9A15
    TIINC   =$9A17      MESSERR =$9A19      CODR    =$9B22      TMOCL   =$9B2B
    IFIIF   =$9B3D      IFDEF   =$9B4C      TOFFST  =$9B63      OFFOFF  =$9B68
    OFFIIF  =$9B69      OFFMOU  =$9B6A      OFFTIM  =$9B6B      OFFUSR  =$9B6C
    OFFDEF  =$9B6D      TIDMOCL =$9B72      TOFFIN  =$9B78      TOFFIN2 =$9B7E
    MOTIF   =$9B84      TITVAL  =$9B88      TVTVAL  =$9B8C      TVNORA  =$9B90
    TVN1ORA =$9B94      NEG65536=$9B98      NEG32768=$9B9D      TYPLET  =$9BA2
    ADAPFBET=$9BBC      ADAPFTET=$9BCD      ADPFB   =$9BDE      ADPFT   =$9BEF
    FIN     =$9C00  ?   FLGFN   =$9CC0  ?   WRKFA   =$9CC1  ?   WRKFB   =$9CC6
?   WRKFC   =$9CCB  ?   SVNUM   =$9CD0      MOSL    =$9CD1      NEEDDEC =$9CD2
    OPTCGOTO=$9CD3      VNARRG91=$9CD4      VNPTRG90=$9CD6      ADADR   =$9CD8
    INHACTV =$9CDA      CTRACTV =$9CDB      MTACTV  =$9CDC      ICTRACTV=$9CDD
    PVERSION=$9CDE      REVECTOR=$9CDF      MACHINE =$9CED      MEMORY  =$9CEF
    VID80C  =$9CF0      DBUFP   =$9D00      RD2     =$A47A      ISBASRUN=$A65E
    EXFLG   =$AAB3      WHCBASIC=$AAB6      AXHIMEM =$BF00      ZAUXRT  =$BF00
    ZAUXB   =$BF1C      ZAUXRT0 =$BF1C      ZAUXRT1 =$BF3B      ZAUXRT2 =$BF65
?   ZAUXRT3 =$BF83      ZAUXRT23=$BF92      ZAUXRET =$BFB9      ZAUXOFFT=$BFD6
    AXRTMAIN=$BFD9      AXARTAB =$BFDB      AXARYPNT=$BFDB      AXSTREND=$BFDD
    AXSZ    =$BFDF      AXOFFSET=$BFDF      ELMSIZ  =$BFE1      AXVALUE =$BFE2
    AXARYPT2=$BFE2      XFER    =$C314      TOKTABL =$D0D0      CHKMEM  =$D3D6
    REASON  =$D3E3      MEMERR  =$D410      FNDLIN  =$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    =$DAFB      OUTSPC  =$DB57      OUTDO   =$DB5C
    APRNEXT =$DCF9      FRMNUM  =$DD67      CHKNUM  =$DD6A      CHKSTR  =$DD6C
    GOTMIERR=$DD76      TMERR   =$DD76      FRMEVL  =$DD7B      FRMSTCK3=$DE20
    APFRMELM=$DE67      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  =$E3DD      GOSTLERR=$E5B2      MOVINS  =$E5D4
    GETBYT  =$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   =$EB53      QINT    =$EBF2      FOUT    =$ED34
    APRONERR=$F2CB      INSDS2  =$F88C      PCADJ   =$F953      MOVE    =$FE2C
```