Apple II Integer Basic Disassembly

Compilation by Paul R. Santa-Maria
With Comments by "WOZ"

This compilation was put together by Paul R, Santa-Maria for the purpose of educating users on the Internet. The comments by "WOZ" are from the woz.org web site.

Introduction

In 1977, Steven Wozniak wrote a basic interpreter for the Apple Computer in his hotel room. That basic was all hand coded and hand input into an Apple-1 computer. This basic led to other basics, the first of which was the basic for the new Apple II. Integer Basic. This basic would be the staple for games on the Apple II for several years until the introduction of Applesoft eventually did away with Integer Basic. However, for the purpose of documentation, we have introduced this book.

Much of the information in this book comes directly from "Woz" himself. This is in the form of emails, magazine interviews, notes and any other items which could be garnered on the subject.

For those Apple II aficionados who still have an original Apple II which boots to monitor, you will appreciate having this listing. Not only is it complete but it is also the only one known to exist other than in hand written form by "Woz" himself.

Notes About Integer Basic

The first section is from BYTE magazine where Woz described the Apple II. The second is from www.woz.org, where Woz has a section where he responds to e-mail; some of the questions asked about Integer BASIC. The third section is my disassembly of Integer BASIC.

Apple BASIC (BYTE 1977:5 p34)

Apple-II BASIC is implemented as a translator-interpreter combination. When a line is read from the input device, the translator analyzes it and generates a more efficient internal language facsimile. Syntax errors are detected at this time. The "nouns" of this internal language are variable names, integer constants (pre-converted to binary for execution speed enhancement), and string constants. The "verbs" are 1 byte tokens substituted for keywords, operators and delimiters. Because the translator distinguishes syntax, different verbs are assigned to different usages of the same symbol. For example, three distinct verbs represent the word PRINT, depending on whether it is immediately followed by a string source, an arithmetic expression or nothing. Thus this distinction need not be made at execution time. For each verb there exists a subroutine to perform that specific action. Listing a program actually involves de-compiling the internal language back to BASIC source code. Those statements with line numbers are stored as part of the user program, while those without line numbers are executed immediately. If desired, the Apple BASIC interpreter's editing functions can be set to generate line numbers automatically. Although some commands are valid only for immediate execution and others only for programmed execution, most can be employed in both ways. In the BASIC source programs, multiple statements may reside on the same line, separated by colons (':').

BASIC language statements are stored in user memory as they are accepted and variables are allocated space the first time they are encountered during immediate or programmed execution. When a program terminates, whether by completion, interruption or error conditions, all variables are preserved. Programs may be interrupted in execution by typing an ASCII control C; it is then possible to examine and modify a few variables in immediate mode, then continue execution at the point of interruption by typing the CONtinue command. BASIC provides the line number of the statement as the point of interruption when this sequence is used. The entire variable space is cleared to zero when BASIC is initialized by the CLR command, and prior to executing the RUN command. (It is possible to carry variables from one program to another, but to initiate the second program a GOTO command must be used instead of RUN in order to override the automatic clear at the beginning of execution of a new program.)

The interpreter consists of a standard expression evaluator and a symbol table routine for allocating variable storage similar to those described by Prof Maurer in his 2 part series in the February and March 1976 issues of BYTE. As statements are scanned, nouns and verbs are encountered. Variable names result in calls to the symbol table routine, which pushes address and length information on the noun stack (operand stack).

Constants are pushed directly onto this stack. Verbs are pushed onto the verb stack (operator stack) after popping and executing any verbs of greater priority. A verb is executed by calling its associated subroutine. Tables define priorities and routine entry addresses for all verbs.

Keywords such as THEN or STEP, and delimiters such as commas and parentheses, are dealt with just as though they were arithmetic operators. Verb routines obtain their arguments from the noun stack. Because verbs such as parentheses tend sometimes to be of low, and other times of high priority, each verb is actually assigned two priorities (left-hand, right-hand). One represents its tendency to force execution of other verbs, the second its tendency to be executed.

Woz Speaks Part 1

My own BASIC was the hardest task of developing the Apple I and][computers. I'd never studied compiler/interpreter writing and had only practiced my ideas on paper before. I'd read some good books on the subject. I'd never programmed in BASIC before the Apple I. I just sniffed the air and decided that the games that would drive personal computers were written in BASIC. I picked up a manual at Hewlett Packard and used their variant of BASIC as my model. Either they had good sub string syntax or I evolved my own based on theirs, but I much preferred it to the DEC style that Microsoft went with, using LEFT$ and MID$ and RIGHT$ functions. I laid out my syntax charts and made a decision to take floating point out so that I could finish slightly sooner and have the first BASIC for the 6502 processor ever. I mainly wanted it to be able to play games. Then I knew it was good enough for whatever else. I also wanted to program solutions to my Hewlett Packard engineering problems. That's where I worked as an engineer designing calculators.

I could go on. The BASIC turned out extremely modular, so I could easily add something by adding some syntax descriptions in near-text form, and write routines for the new functions or ops that were needed. The language didn't have to be rewritten.

Woz Speaks Part 2

I wrote the original Apple Integer BASIC. I had wanted it to be the very first BASIC for the 6502 microprocessor. I might then have something to be recognized for. I decided that it had to play games and let me solve engineering problems. I first wrote out a syntax with floating point but then figured that it might be done a few weeks sooner with just integers. I had to write it in the evenings as I worked at Hewlett Packard then. So I cut back to an Integer BASIC that I called "Game BASIC".

I'd never programmed in BASIC. My college had encountered Fortran, several machine languages, Algol, and a couple of special languages. But you could buy a book called "101 BASIC Games". Plus, the Gates/Allen BASIC was becoming the standard thing to get for your Altair computer, although very few people had these computers yet.

I'd never writing a computer language or taken a course in it, although I'd studied books on my own touching on the topic. I have no idea to this day if I wrote it as anyone else would. I broke the entire language down into a syntax table that was stored in memory, in modified text form. A word like "PRINT" was stored as the 5 letters. If you were allowed an unsigned expression after some word, I stored a pointer to the syntax of that type of expression, which specified what it could be made of. Each line was compared, letter-by-letter, through this syntax table to see if there was any valid BASIC statement.

I gave each symbol in the syntax table a particular code as on operator. The word "PRINT" might be operator number 5 and "FOR" might be operator number 13, etc. A plus sign had its code too. A symbol like a minus sign might have two different codes depending on whether it was prefix (like -5) or infix (like 9-6). A variable or a number was an operand. I pushed the operand references onto one-stack and operator codes onto another. But the operator codes each had 2 different priorities telling my BASIC whether to push them on top of the topmost operator already on the stack, or to pop that one off and generate the output program from it. Each operator had a value for it's tendency to push others off, and a value for its resistance to being pushed off. For example, plus tends to push divide off, causing the division to happen first. Strangely all this works.

Then I had to write one short routine for each of perhaps 100 operators. These included keywords like "PRINT", mathematical operators like 'plus', parenthesis, and other grammar symbols of BASIC.

It took a couple of months to get the BASIC to this shape, with an engine that ran the whole thing. Then I would define a Syntax sentence in the syntax table, along with any routines for any new operator symbols. I would test it; get it working, and move on to the next syntax sentence for the next BASIC statement. From this point on, things were very modular and I was only writing very short programs.

Well, the BASIC was a very big success. Especially when I was able to easily add statements and corresponding routines for color graphics and game commands in the Apple][.

Woz Speaks Part 3

I wrote it all in machine language without an assembler. It was the only way that I could afford. I looked at the sort of software that I wanted - games and puzzle solvers and logic simulators. Floating point wasn't the way to go here. My first syntax table was for floating point but I saw that I could make the language faster and tighter, and I could complete it perhaps a couple of weeks sooner, if I went integer only, so I backed off. Just because of this I did include some floating point routines in the Apple][ROM.

Anyway, I used a condensed syntax chart to scan lines that were typed in, to allocate tokens corresponding the syntax elements of a good statement. The interpreted collection of tokens would be executed from left to right, with a couple of tables holding 'push' and 'pull' priorities for each token (operators, parenthesis, etc.). Numbers and variable names weren't tokenized. This way I could just add commands or other items to a near-ASCII syntax table and then write small routines for each new token added. It was a very efficient approach that broke up a large task in a very orderly way.

I'd never studied compiler writing. I'd just worked out ideas on my own back in college days. I had read some papers and books on it, but never wrote one before or did any homework or other exercises in this regard. I feel very lucky that I was able to do it. For me it was a larger task by far than both the Apple I and Apple][combined.

The original version of this BASIC is in a binder in my own handwriting.

Disassembly Listing of Integer Basic

```
ORG $E000
LST OFF
XC OFF ;6502 only
EXP ON ;print only macro call
LSTDO OFF ;don't list conditional code
TR OFF ;don't truncate listing to 3 bytes
CYC OFF ;don't print cycle times
USE MACROS
LST ON
**************************************************
*                                                *
*                 INTEGER BASIC                  *
*                                                *
*                      WOZ                       *
*                                                *
**************************************************
*                                                *
* "That BASIC, which we shipped with the first   *
*  Apple II's, was never assembled -- ever.      *
*  There was one handwritten copy, all           *
*  handwritten, all hand assembled."             *
*                                                *
*              Steve Wozniak                     *
*              Call-A.P.P.L.E., October 1986     *
*                                                *
**************************************************

* Computer  Apple II family
* O/S       none needed, but usually DOS 3.3
* Language  6502 assembly -- Merlin assembler
* Disassembled by:
*           Paul R. Santa-Maria (paulrsm@buckeye-express.com)
*           P.O. Box 924
*           Monroe  MI  48161
* Revised   1 May 2000
* Reference "What's Where in the Apple"; William F. Luebbert
*           Peeking at Call-A.P.P.L.E.  Vol 2  1979; pp44-61

**************************************************

* zero-page

LOMEM = $004A ;ptr: start of vars
HIMEM = $004C ;ptr: end of BASIC program
NOUNSTKL = $0050 ;noun stack low bytes (80-87)
SYNSTKH = $0058 ;syntax stack high byte
NOUNSTKH = $0078 ;noun stack high bytes (78-97)
SYNSTKL = $0080 ;syntax stack low bytes (80-9F)
NOUNSTKC = $00A0 ;noun stack counter (A0-BF)
TXTNDXSTK = $00A8 ;text index stack (A8-C7)
TXTNDX = $00C8 ;text index val (OUTVAL)
LEADBL = $00C9 ;leading blanks index (YTEMP)
PP = $00CA ;ptr: start of program
PV = $00CC ;ptr: end of vars
ACC = $00CE ;word: main accumulator
SRCH = $00D0 ;ptr to search var tbl
TOKNDXSTK = $00D1 ;token index stack (D1-F0)
SRCH2 = $00D2 ;second var search ptr
IFFLAG = $00D4 ;IF/THEN fail flag
CRFLAG = $00D5 ;carriage return flag
VERBNOW = $00D6 ;verb currently in use
PRFLAG = $00D7 ;print it now flag
XSAVE = $00D8 ;temp Xreg save
RUNFLAG = $00D9 ;run mode flag
AUX = $00DA ;word: aux ctr
PR = $00DC ;word: current line value
*PN = $00DE ;ptr to current noun
PX = $00E0 ;ptr to current verb
P1 = $00E2 ;aux ptr 1 (delete line ptr)
P2 = $00E4 ;aux ptr 2 ...
*  (line num adr) (next line num) (general flag)
P3 = $00E6 ;aux ptr 3 (next ptr)
TOKNDX = $00F1 ;token index val
PCON = $00F2 ;continue ptr (PRDEC low/high)
AUTOINC = $00F4 ;auto line increment
AUTOLN = $00F6 ;current auto line
AUTOFLAG = $00F8 ;auto line mode flag ($FF = on)
CHAR = $00F9 ;current char
LEADZR = $00FA ;leading zeros index ($00,$A0,$B0)
FORNDX = $00FB ;FOR-NEXT loop index
GOSUBNDX = $00FC ;GOSUB index
SYNSTKDX = $00FD ;syntax stack index val
SYNPAG = $00FE ;ptr: syntax page
*if SYNPAG+1 <> 0 then error condition exists

STACK = $0100 ;6502 STACK

*   GOSUB/RETURN usage

STK_00 = STACK+$00
STK_10 = STACK+$10
STK_20 = STACK+$20
STK_30 = STACK+$30

*   FOR/NEXT/STEP usage

STK_40 = STACK+$40
STK_50 = STACK+$50
STK_60 = STACK+$60
STK_70 = STACK+$70
STK_80 = STACK+$80
STK_90 = STACK+$90
STK_A0 = STACK+$A0
STK_B0 = STACK+$B0
STK_C0 = STACK+$C0
STK_D0 = STACK+$D0

* I/O addresses

KBD = $C000
KBDSTRB = $C010

* Monitor zero page and low memory

WNDWDTH = $0021
CH = $0024
CV = $0025
GBAS = $0026
H2 = $002C
V2 = $002D
A1 = $003C
A2 = $003E
PROMPT = $0033
RNDL = $004E
RNDH = $004F

IN = $0200

* Monitor routines

PLOT = $F800
HLINE = $F819
VLINE = $F828
GBASCALC = $F847
SETCOL = $F864
PREAD = $FB1E
SETTXT = $FB39
SETGR = $FB40
VTAB = $FC22
WRITE = $FECD
WRITE0 = $FECF
READ = $FEFD
NXTCHAR = $FD75
CROUT = $FD8E
COUT = $FDED
INPORT = $FE8B
OUTPORT = $FE95
BELL = $FF3A

* ASCII (excess $8000 for xref listing)

ETX = $8003 ;CTRL-C
LF = $800A
CR = $800D
BLANK = $8020
DQT = $8022
SQT = $8027

**************************************************
* ;Z  = unreferenced area
* ;V  = referenced in verb table
* ;VO = referenced in verb table only
* ;solo = one reference only (could be in-line)


 PUT PART1
BASIC JSR COLD
BASIC2 JMP WARM

SetPrompt ;solo
  STA PROMPT
  JMP COUT
*>

  RTS ;Z
**

HE00C
  TXA ;?print a trailing blank?
  AND #$20
  BEQ HE034 ;=>RTS
HE011 ;solo
  LDA #BLANK+$80
  STA P2
  JMP COUT
*>

HE018 ;solo
  LDA #32 ;check line length
HE01A
  CMP CH
  BCS NextByte ;=HS> line too short
  LDA #CR+$80 ;print CR, then 7 blanks
  LDY #7
*!LOOP
    JSR COUT
    LDA #BLANK+$80
    DEY
*!UNTIL <EQ>

NextByte ;get next byte 16-bit ptr
  LDY #0
  LDA (P1),Y
  INCW P1
HE034
  RTS
**

* tkn $75 , (with tkn $74 LIST)
*   LIST 5,30

COMMA_LIST ;VO
  JSR GET16BIT
  JSR HE576
HE03B
  CMPW P1;P3
  BCS HE034 ;=>P1 <HS> P3, RTS
  JSR UNPACK
  JMP HE03B
*>


* tkn $76 LIST
*   list entire program

LIST ;VO
  MOVW PP;P1
  MOVW HIMEM;P3
  BNE HE03B ;=>always

* tkn $74 LIST
*   specific line number or range of numbers
*   LIST 10: LIST 5,30

LISTNUM ;VO
  JSR GET16BIT
  JSR HE56D
  MOVW P2;P1
  BCS HE034 ;=>RTS
UNPACK ;unpack tokens to mnemonics
  STX XSAVE
  LDA #BLANK+$80
  STA LEADZR
  JSR NextByte
  TYA
HE077
  STA P2
  JSR NextByte
  TAX
  JSR NextByte
  JSR PRDEC
*!LOOP
    JSR HE018
```

```
   STY LEADZR
   TAX
   BPL HE0A3 ;=>
   ASL
   BPL HE077 ;=>
   LDA P2
*!  IF <EQ>
     JSR HE011
*!  ENDIF
   TXA
*!  LOOP
     JSR COUT
HE099
     LDA #$25
     JSR HE01A
     TAX
*!  UNTIL <PL>
   STA P2
HE0A3
   CMP #$01
*!  IF <EQ>
     LDX XSAVE
     JMP CROUT
*!  ENDIF
   PHA
   STY ACC
   LDX #>SYNTABL2
   STX ACC+1
   CMP #$51 ;END tkn
*!  IF <HS>
     DEC ACC+1 ; in SYNTABL
     SBC #$50 ;TAB tkn
*!  ENDIF
*!  LOOP
     PHA
     LDA (ACC),Y
*!    LOOP
*!      LOOP
         TAX
         DEY
         LDA (ACC),Y
*!      UNTIL <MI>
        CPX #$C0
*!    WHILE <LO>
        CPX #0
*!    UNTIL <PL>
     TAX
     PLA
     SBC #1 ;carry is set
*!  UNTIL <EQ>
   BIT P2
*!  IF <PL>
     JSR HEFF8
*!  ENDIF
*!  LOOP
     LDA (ACC),Y
*!  WHILE <MI>
     TAX
     AND #$3F
     STA P2
     CLC
     ADC #BLANK+$80
     JSR COUT
     DEY
     CPX #$C0
*!  UNTIL <HS>
   JSR HE00C
   PLA
   CMP #$5D ;93 ]
   BEQ HE099 ;=>
   CMP #$28 ;40 (
*!UNTIL <EQ>
  BEQ HE099 ;=>always

* tkn $2A (
*   substring
*   PRINT A$(12,14)

PAREN_SUBSTR ;VO
  JSR HE118
  STA NOUNSTKL,X
  CMP NOUNSTKH,X
HE102
  BCC HE115 ;=LO>
HE104
  LDY #ErrMsg05 ;"STRING"
HE106
  JMP ERRMESS
*>

* tkn $23 ,
*   substring
*   PRINT A$(3,3)

COMMA_SUBSTR ;VO
  JSR GETBYTE
  CMP NOUNSTKL,X
  BCC HE104 ;=LO>"STRING"
  JSR HEFE4
  STA NOUNSTKH,X
HE115
  JMP HE823
*>

HE118
  JSR GETBYTE
  BEQ HE104 ;=>"STRING"
  SEC
  SBC #1
  RTS
**

* tkn $42 (
*   string array is destination of the data
*   A$(1)="HELLO"

HE121 ;VO
  JSR HE118
  STA NOUNSTKL,X
  CLC
  SBC NOUNSTKH,X
  JMP HE102
*>

HE12C
  LDY #ErrMsg03 ;"MEM FULL"
  BNE HE106 ;=>always

* tkn $43 ,
*   next var in DIM statement is string
*   DIM X(5),A$(5)

* tkn $4E DIM
*   string var.  uses tkn $22 (
*   DIM A$(5)

DIMSTR ;VO
  JSR HE118
  INX
HE134
  LDA NOUNSTKL,X
  STA AUX
  ADC ACC
  PHA
  TAY
  LDA NOUNSTKH,X
  STA AUX+1
  ADC ACC+1
  PHA
  CPY PP
  SBC PP+1
  BCS HE12C ;=HS>"MEM FULL" error
  LDA AUX ;AUX := AUX-2
  ADC #<0-2
  STA AUX
  LDA #>0-2
  TAY
  ADC AUX+1
  STA AUX+1
*!LOOP
    INY
    LDA (AUX),Y
    CMP PV,Y
    BNE DimErr ;=>
    TYA
*!UNTIL <NE>
*!LOOP
    PLA
    STA (AUX),Y
    STA PV,Y
    DEY
*!UNTIL <MI>
  INX
  RTS
**

  NOP ;Z
DimErr
  LDY #ErrMsg17 ;"DIM"
HE16F
  BNE HE106 ;=>always

INPUTSTR ;input a string
  LDA #0
  JSR HE70A
  LDY #$02
  STY NOUNSTKH,X
  JSR HE70A
  STX XSAVE
  TAX
  INC PROMPT ;change '>' to '?'
  JSR RDKEY
  DEC PROMPT ;change '?' to '>'
  TXA
  LDX XSAVE
  STA NOUNSTKH,X

* tkn $70 =
*   string - non-conditional
*   A$ = "HELLO"

HE18C ;VO
  LDA NOUNSTKL+1,X
  STA ACC
  LDA NOUNSTKH+1,X
  STA ACC+1
  INX
  INX
  JSR HE1BC
*!LOOP
    LDA NOUNSTKL-2,X
    CMP NOUNSTKH-2,X
*!WHILE <LO>
    INC NOUNSTKL-2,X
    TAY
    LDA (ACC),Y
    LDY NOUNSTKL,X
    CPY P2
*!  IF <HS>
      LDY #ErrMsg18 ;"STR OVFL"
      BNE HE16F ;=>always
*!  ENDIF
    STA (AUX),Y
    INC NOUNSTKL,X
*!UNTIL <CS>
  LDY NOUNSTKL,X
  TXA
  STA (AUX),Y
  JMP HF223
*>

HE1BC ;solo
  LDA NOUNSTKL+1,X
  STA AUX
  SEC
  SBC #2
  STA P2
  LDA NOUNSTKH+1,X
  STA AUX+1
  SBC #0
  STA P2+1
  LDY #0
  LDA (P2),Y
  CLC
  SBC AUX
  STA P2
  RTS
**

* tkn $39 =
*   string logic op
*   IF A$ = "CAT" THEN END

HE1D7 ;V
  LDA NOUNSTKL+3,X
  STA ACC
  LDA NOUNSTKH+3,X
  STA ACC+1
  LDA NOUNSTKL+1,X
  STA AUX
  LDA NOUNSTKH+1,X
  STA AUX+1
  INX
  INX
  INX
  LDY #0
  STY NOUNSTKH,X
  STY NOUNSTKC,X
  INY
  STY NOUNSTKL,X
*!LOOP
    LDA HIMEM+1,X
    CMP NOUNSTKH-3,X
    PHP
    PHA
    LDA NOUNSTKL-1,X
    CMP NOUNSTKH-1,X
*!  IF <HS>
      PLA
      PLP
*!    IF <CC>
HE203
        LSR NOUNSTKL,X
*!    ENDIF
      RTS

*!  ENDIF
    TAY
    LDA (ACC),Y
    STA P2
    PLA
    TAY
    PLP
    BCS HE203 ;=>EXIT LOOP
    LDA (AUX),Y
    CMP P2
    BNE HE203 ;=>EXIT LOOP
    INC NOUNSTKL-1,X
    INC HIMEM+1,X
*!UNTIL <LO>
* always

* tkn $3A #
*   string logic op
*   IF A$ # "CAT" THEN END

HE21C ;VO
  JSR HE1D7
  JMP NOT
*>

* tkn $14 *
*   num math op
*   A = 27 * 2

MULT ;V
  JSR HE254
*!LOOP
    ASL ACC
    ROL ACC+1 ;add partial product if C flag set
*!  IF <CS>
      ADDW P3;AUX;P3
*!  ENDIF
    DEY
    BEQ HE244 ;=>EXIT LOOP
    ASL P3
    ROL P3+1
*!UNTIL <MI>
  JMP HE77E
*>

HE244
  LDA P3
  JSR HE708
  LDA P3+1
  STA NOUNSTKC,X
  ASL P2+1
  BCC HE279 ;=>RTS
  JMP NEGATE
*>

HE254
  LDA #$55
  STA P2+1
  JSR HE25B
HE25B
  MOVW ACC;AUX
  JSR GET16BIT
  STY P3 ;P3 := 0
  STY P3+1
  LDA ACC+1
*!IF <MI>
    DEX
    ASL P2+1
    JSR NEGATE
    JSR GET16BIT
*!ENDIF
  LDY #$10
HE279
  RTS
**

* tkn $1F MOD
*   num op
*   IF X MOD 13 THEN END

MOD ;V
  JSR HEE6C
  BEQ HE244 ;=>always

  DB $FF ;Z

HE280 ;solo
  INC PROMPT ;change '>' to '?'
  LDY #0
  JSR GETCMD
  DEC PROMPT ;change '?' to '>'
  RTS
**

* tkn $3D SCRN(
*   PRINT SCRN(X,Y)

SCRN ;VO
  JSR GETBYTE
  LSR ;Areg := Areg/2
  PHP ;stash carry (lsb)
  JSR GBASCALC
  JSR GETBYTE
  TAY
  LDA (GBAS),Y ;get screen byte
  PLP ;retrieve carry
*!IF <CS>
    LSR ;odd, upper half
    LSR
    LSR
    LSR
*!ENDIF
  AND #$0F ;Areg := color number
  LDY #0
  JSR HE708
  STY NOUNSTKC,X
  DEY
  STY PRFLAG ;PRFLAG := $FF

* tkn $3E ,
*   PRINT SCRN(X,Y)

COMMA_SCRN ;VO
  RTS
**

  DB $FF,$FF,$FF,$FF ;Z

  JSR HEFD3 ;old 4K cold start ;Z

* Warm start

WARM ;main compile/execute code
  JSR CROUT ;emit blank line
HE2B6
  LSR RUNFLAG ;not running
  LDA #">"
  JSR SetPrompt ;set and print prompt char
  LDY #0
  STY LEADZR ;no leading zeros for AUTOLN
  BIT AUTOFLAG ;AUTO?
* if AUTOLN active
*!IF <MI>
    LDX AUTOLN ;yes, print line number
    LDA AUTOLN+1
    JSR PRDEC
    LDA #BLANK+$80 ;and a blank
    JSR COUT
*!ENDIF
  LDX #$FF ;init Sreg
  TXS
  JSR GETCMD
  STY TOKNDX
  TXA
  STA TXTNDX
  LDX #$20
  JSR HE491
  LDA TXTNDX ;PX := TXTNDX+$0200+C flag
  ADC #<$0200
  STA PX
  LDA #0
  TAX
  ADC #>$0200
  STA PX+1
  LDA (PX,X)
  AND #$F0
  CMP #"0"
*!IF <NE>
    JMP HE883
*!ENDIF
  LDY #2 ;move two bytes
*!LOOP
    LDA (PX),Y
    STA ACC-1,Y
    DEY
*!UNTIL <EQ>
  JSR HE38A
  LDA TOKNDX
  SBC TXTNDX
  CMP #$04
  BEQ HE2B6 ;=>
  STA (PX),Y
  LDA PP ;P2 := PP-(PX),Y
  SBC (PX),Y
  STA P2
  LDA PP+1
  SBC #0
  STA P2+1
  CMPW P2;PV
  BCC MEMFULL ;=>P2 <LT> PV
*!LOOP
    LDA PP ;P3 := PP-(PX),Y
    SBC (PX),Y
    STA P3
    LDA PP+1
    SBC #0
    STA P3+1
    LDA (PP),Y
    STA (P3),Y
    INCW PP
    CMPW P1;PP
*!UNTIL <LO>
*!LOOP
    LDA P2,X
    STA PP,X
    DEX
*!UNTIL <MI>
  LDA (PX),Y
  TAY
*!LOOP
    DEY
    LDA (PX),Y
    STA (P3),Y
    TYA
*!UNTIL <EQ>
  BIT AUTOFLAG ;auto line?
*!IF <MI>
* yes
*!  LOOP
      LDA AUTOLN+1,X ;AUTOLN :=
AUTOLN+AUTOINC
      ADC AUTOINC+1,X
      STA AUTOLN+1,X
      INX
*!  UNTIL <NE>
*!ENDIF
  BPL HE3E5 ;=>always

  DB $00,$00,$00,$00 ;Z

MEMFULL
  LDY #ErrMsg03 ;"MEM FULL"
  BNE ERRMESS ;=>always

* tkn $0A ,
*   DEL 0,10

COMMA_DEL ;VO
  JSR GET16BIT
  MOVW P1;P3
  JSR HE575
  MOVW P1;P2
  BNE HE395 ;=>always?

* tkn $09 DEL

DEL ;VO
  JSR GET16BIT
HE38A
  JSR HE56D
  MOVW P3;P1
HE395
  LDY #0
* memory move: P3<PP.P2 backwards
*!LOOP
    CMPW PP;P2
    BCS HE3B7 ;=>PP <HS> P2
    DECW P2
    DECW P3
    LDA (P2),Y
    STA (P3),Y
*!UNTIL <HS>
* always

HE3B7 ;solo
  MOVW P3;PP
  RTS
**

*!LOOP
    JSR COUT ;print error message
    INY
ERRORMESS ;print error message
    LDA ErrorMsgs,Y ;routine entry point
*!UNTIL <PL>
  ORA #$80
  JMP COUT
*>

GETCMD
  TYA
  TAX
  JSR NXTCHAR ;
  TXA
  TAY
  LDA #"_" ;underline problem?
  STA IN,Y
  LDX #$FF
  RTS
**

  RTS ;Z
**

HE3DE
  LDY #ErrMsg01 ;"TOO LONG"
ERRMESS ;print error message and goto mainline
  JSR PRINTERR
*$E3E3 DOS 3.3 chains here when processing errors
  BIT RUNFLAG
HE3E5
*!IF <PL>
    JMP HE2B6
*!ENDIF
  JMP HEB9A
*>

HE3ED ;solo
  ROL
  ADC #$A0
  CMP IN,X
  BNE HE448 ;=>
  LDA (SYNPAG),Y
  ASL
*!IF <PL>
    DEY
    LDA (SYNPAG),Y
    BMI HE428 ;=>
    INY
*!ENDIF
  STX TXTNDX
  TYA
  PHA
  LDX #0
  LDA (SYNPAG,X)
  TAX
*!LOOP
    LSR
    EOR #$40
    ORA (SYNPAG),Y
    CMP #$C0
*!  IF <HS>
      INX
*!  ENDIF
    INY
*!UNTIL <EQ>
  PLA
  TAY
  TXA
  JMP HF2F8
*>

HE41C
  INC TOKNDX
  LDX TOKNDX
  BEQ HE3DE ;=>"TOO LONG"
  STA IN,X
HE425
  RTS
**

HE426 ;solo
  LDX TXTNDX
HE428
  LDA #BLANK+$80
*!LOOP
    INX
    CMP IN,X
*!UNTIL <LO>
  LDA (SYNPAG),Y
  AND #$3F
  LSR
  BNE HE3ED ;=>
  LDA IN,X
*!IF <CC>
    ADC #$3F
    CMP #$1A
    BCC HE4B1 ;=LO>
*!ENDIF
  ADC #$4F
  CMP #$0A
  BCC HE4B1 ;=LO>
HE448
  LDX SYNSTKDX
*!LOOP
    INY
    LDA (SYNPAG),Y
    AND #$E0
    CMP #$20
    BEQ HE4CD ;=>
    LDA TXTNDXSTK,X
    STA TXTNDX
    LDA TOKNDXSTK,X
    STA TOKNDX
*!  LOOP
      DEY
      LDA (SYNPAG),Y
      ASL ;dbl
*!  UNTIL <MI>
    DEY
    BCS HE49C ;=>
    ASL ;dbl
    BMI HE49C ;=>
    LDY SYNSTKH,X
    STY SYNPAG+1
    LDY SYNSTKL,X
    INX
*!UNTIL <MI>
HE470
  BEQ HE425 ;=>RTS
  CMP #$7E
  BCS HE498 ;=HS>
  DEX
*!IF <MI>
    LDY #ErrMsg01 ;"TOO LONG"
* BUG FIX: ABOVE LINE SHOULD BE
*   LDY #ErrMsg04 ;"TOO MANY PARENS"
* REF: CALL-APPLE  MAR 1983  P.114
  BPL HE4A6 ;=>always
*!ENDIF
  STY SYNSTKL,X
  LDY SYNPAG+1
  STY SYNSTKH,X
  LDY TXTNDX
  STY TXTNDXSTK,X
  LDY TOKNDX
  STY TOKNDXSTK,X
  AND #$1F
  TAY
  LDA SYNTABLNDX,Y
HE491
  ASL ;dbl
  TAY
  LDA #>SYNTABL/2
  ROL
  STA SYNPAG+1
HE498
*!IF <EQ>
    INY
*!ENDIF
  INY
HE49C
  STX SYNSTKDX
  LDA (SYNPAG),Y
  BMI HE426 ;=>
*!IF <EQ>
    LDY #ErrMsg02 ;"SYNTAX"
HE4A6
    JMP ERRMESS
*!ENDIF
  CMP #$03
  BCS HE470 ;=HS>
  LSR ;half
  LDX TXTNDX
  INX
HE4B1
  LDA IN,X
  BCC HE4BA ;=>
  CMP #DQT+$80
  BEQ HE4C4 ;=>
HE4BA
  CMP #"_" ;underline problem?
  BEQ HE4C4 ;=>
  STX TXTNDX
HE4C0
*!LOOP
    JSR HE41C
    INY
HE4C4
    DEY
    LDX SYNSTKDX
*!  LOOP
      LDA (SYNPAG),Y
      DEY
      ASL
      BPL HE49C ;=>
HE4CD
      LDY SYNSTKH,X
      STY SYNPAG+1
      LDY SYNSTKL,X
      INX
      LDA (SYNPAG),Y
      AND #%10011111
*!  UNTIL <EQ>
    STA PCON
    STA PCON+1
    TYA
    PHA
    STX SYNSTKDX
    LDY TOKNDXSTK-1,X
    STY LEADBL
    CLC
*!  LOOP
      LDA #$0A
      STA CHAR
      LDX #0
      INY
      LDA IN,Y
      AND #$0F
*!    LOOP
        ADC PCON
        PHA
        TXA
        ADC PCON+1
        BMI HE517 ;=>
        TAX
        PLA
        DEC CHAR
*!    UNTIL <EQ>
      STA PCON
      STX PCON+1
      CPY TOKNDX
*!  UNTIL <EQ>
    LDY LEADBL
    INY
    STY TOKNDX
    JSR HE41C
    PLA
    TAY
    LDA PCON+1
*!UNTIL <CC>
HE517
  LDY #ErrMsg00 ;">32767"
  BPL HE4A6 ;=>always

*-----------------------------
* Name      PRDEC
* Purpose   Print a 16-bit number in decimal.
* Input     Areg = high byte
*           Xreg = low byte
* Output
* Uses
* Calls
* Note

PRDEC
  STA PCON+1
  STX PCON
  LDX #4
  STX LEADBL
*!LOOP
```

```
  LDA #"0"
  STA CHAR
*!  LOOP
   LDA PCON
   CMP NUMLOW,X
   LDA PCON+1
   SBC NUMHI,X
*!  WHILE <HS>
   STA PCON+1
   LDA PCON
   SBC NUMLOW,X
   STA PCON
   INC CHAR
*!  UNTIL <EQ>
*GETDIG
  LDA CHAR
  INX
  DEX
  BEQ PRDEC5 ;=>
  CMP #"0"
*!  IF <NE>
   STA LEADBL
*!  ENDIF
* if LEADBL is <MI> or LEADZR <NE> #0
  BIT LEADBL
  BMI PRDEC5 ;=>
  LDA LEADZR
  BEQ PRDEC6 ;=>
* then
PRDEC5 ;PRINT
  JSR COUT
  BIT AUTOFLAG ;auto line?
*!  IF <MI>
   STA IN,Y
   INY
*!  ENDIF
PRDEC6 ;NXTX
  DEX
*!UNTIL <MI>
 RTS
**

NUMLOW
 DB 1
 DB 10
 DB 100
 DB 1000
 DB 10000

NUMHI
 DB 1/$0100
 DB 10/$0100
 DB 100/$0100
 DB 1000/$0100
 DB 10000/$0100

HE56D
 MOVW PP;P3
HE575
 INX
HE576
*!LOOP
*   MOVW P3;P2
  LDA P3+1 ;P2 := P3
  STA P2+1
  LDA P3
  STA P2
*   CMPW P2;HIMEM
  CMP HIMEM ;is P2 <HS> HIMEM?
  LDA P2+1
  SBC HIMEM+1
*!WHILE <LO>
  LDY #1
  LDA (P2),Y
  SBC ACC
  INY
  LDA (P2),Y
  SBC ACC+1
*!WHILE <LO>
  LDY #0
  LDA P3 ;P3 := P3.W + (P2).B
  ADC (P2),Y
  STA P3
*!  IF <CS>
   INC P3+1
   CLC
*!  ENDIF
  INY
  LDA ACC :is ACC+1 <HS> (P2),Y ?
  SBC (P2),Y
  INY
  LDA ACC+1
  SBC (P2),Y
*!UNTIL <LO>
 RTS
**

* tkn $0B NEW
*   turn off AUTO
*   remove program
*   fall into CLR

NEW ;V
 LSR AUTOFLAG ;manual
 MOVW HIMEM;PP

* tkn $0C CLR
*   remove variables
*   remove FOR loops and GOSUBs

CLR ;V
 MOVW LOMEM;PV
 LDA #0
 STA FORNDX ;no FORs
 STA GOSUBNDX ;no GOSUBs
 STA SYNPAG
 LDA #0 ;Z
 STA $1D ;Z
 RTS
**

 LDA SRCH ;Z
HE5CE
 JMP MEMFULL
*>

*!LOOP
*!  LOOP
   LDY #$FF
HE5D3
   STY XSAVE
*!    LOOP
    INY
    LDA (PX),Y
*!      IF <PL>
     CMP #$40
     BNE HE646 ;=>EXIT LOOP
     STA XSAVE
*!      ENDIF
    CMP (SRCH),Y
*!    UNTIL <NE>
*!    LOOP
    LDA (SRCH),Y
HE5E6
    INY
    LSR
*!    UNTIL <EQ>
   LDA (SRCH),Y
   PHA
   INY
   LDA (SRCH),Y
   TAY
   PLA
HE5F2
   STA SRCH
   STY SRCH+1
   CMP PV
*!  UNTIL <EQ>
  CPY PV+1
*!UNTIL <EQ>
 LDY #0
*!LOOP
*!  LOOP
   INY
   LDA (PX),Y
*!  UNTIL <PL>
  EOR #$40
*!UNTIL <NE>
 TYA
 ADC #$04
 PHA
 ADC SRCH
 TAY
 LDA SRCH+1
 ADC #0
 PHA
 CPY PP
 SBC PP+1
 BCS HE5CE ;=HS>"MEM FULL" error
 STY PV
 PLA
 STA PV+1
 PLA
 TAY
 LDA #0
 DEY
 STA (SRCH),Y
 DEY
 STA (SRCH),Y
 DEY
 LDA PV+1
 STA (SRCH),Y
 DEY
 LDA PV
 STA (SRCH),Y
 DEY
 LDA #0
*!LOOP
  STA (SRCH),Y
  DEY
  BMI HE5D3 ;=>
  LDA (PX),Y
*!UNTIL <EQ>
HE640
 LDA LOMEM
 LDY LOMEM+1
 BNE HE5F2 ;=>always

HE646
 LDA (SRCH),Y
 CMP #$40
 BCS HE5E6 ;=HS>
 STA NOUNSTKC-1,X
 TYA
 ADC #$03
 PHA
 ADC SRCH
 JSR HE70A
*!LOOP
  JSR GETVERB
  DEY
*!UNTIL <EQ>
 TYA
 ADC SRCH+1
 STA NOUNSTKH,X
 PLA
 BIT XSAVE
 BMI HE684 ;=>
 TAY
 LDA #0
 JSR HE70A
 STA NOUNSTKH,X
*!LOOP
  LDA (SRCH),Y
  BPL HE682 ;=>EXIT LOOP
  INC NOUNSTKH,X
  INY
*!UNTIL <EQ>
* always

 DB 9 ;Z

HE679 ;solo
 LDA #0
 STA IFFLAG ;pos
 STA CRFLAG ;pos
 LDX #$20
HE681
 PHA
HE682
 LDY #0
HE684
 LDA (PX),Y
*!LOOP
  BPL HE6A0 ;=>EXIT LOOP
  ASL
  BMI HE640 ;=>
  JSR GETVERB
  JSR HE708
  JSR GETVERB
  STA NOUNSTKC,X
HE696
  BIT IFFLAG
*!  IF <MI>
   DEX
*!  ENDIF
HE69B
  JSR GETVERB
*!UNTIL <CC>
HE6A0
 CMP #$28
*!IF <EQ>
  LDA PX
```

```
    JSR HE70A
    LDA PX+1
    STA NOUNSTKH,X
    BIT IFFLAG
    BMI HE6BC ;=>
    LDA #$01
    JSR HE70A
    LDA #0
    STA NOUNSTKH,X
*!  LOOP
      INC NOUNSTKH,X
HE6BC
      JSR GETVERB
*!  UNTIL <PL>
    BCS HE696 ;=>
*!ENDIF
  BIT IFFLAG
*!IF <MI>
    CMP #$04
    BCS HE69B ;=HS>
    LSR IFFLAG ;pos
*!ENDIF
  TAY
  STA VERBNOW
  LDA HE980,Y
  AND #%01010101 ;even bits only
  ASL
  STA PRFLAG ;temp
HE6D8
  PLA
  TAY
  LDA HE980,Y
  AND #%10101010 ;odd bits only
  CMP PRFLAG
*!IF <LO>
    TYA
    PHA
    JSR HF3EB
    LDA VERBNOW
    BCC HE681 ;=LT> always
*!ENDIF

* BRANCH: get high/low then JSR

  LDA VERBADRL,Y
  STA ACC
  LDA VERBADRH,Y
  STA ACC+1
  JSR HE6FC
  JMP HE6D8
*>

HE6FC
  JMP (ACC)
*>

GETVERB ;get next verb to use
  INCW PX
  LDA (PX),Y
  RTS
**

HE708
  STY NOUNSTKH-1,X
HE70A
  DEX
*!IF <PL>
    STA NOUNSTKL,X
    RTS
*!ENDIF

  LDY #$66 ;"PPED AT" ;Z?
HE712
  JMP ERRMESS
*>

*---------
* Output    Yreg := 0

GET16BIT ;get a 16 bit value
  LDY #0
  LDA NOUNSTKL,X
  STA ACC
  LDA NOUNSTKC,X
  STA ACC+1
  LDA NOUNSTKH,X
*!IF <NE>
    STA ACC+1
    LDA (ACC),Y ;ACC := (ACC),Y
    PHA ;save low byte
    INY ;Yreg := 1
    LDA (ACC),Y
    STA ACC+1
    PLA ;restore low byte
    STA ACC
    DEY ;Yreg := 0
*!ENDIF
  INX
  RTS
**

* tkn $16 =
*   num var logic op
*   IF X = 13 THEN END

HE733 ;VO
  JSR HE74A

* tkn $37 NOT
*   numeric
*   IF NOT X THEN END

NOT ;V
  JSR GET16BIT
  TYA ;Areg := 0
  JSR HE708
  STA NOUNSTKC,X
  CMP ACC
*!IF <EQ>
    CMP ACC+1
*!  IF <EQ>
      INC NOUNSTKL,X
*!  ENDIF
*!ENDIF
  RTS
**

* tkn $17 #
*   num var logic op
*   IF X # 13 THEN END

* tkn $1B <>
*   num var logic op
*   IF X <> 13 THEN END

HE74A ;V
  JSR SUBTRACT
  JSR SGN

* tkn $31 ABS

ABS ;VO
  JSR GET16BIT
  BIT ACC+1
  BMI HE772 ;=>
HE757 ;solo
  DEX
HE758
  RTS
**

* tkn $30 SGN

SGN ;V
  JSR GET16BIT
  LDA ACC+1 ;is ACC zero?
*!IF <EQ>
    LDA ACC
    BEQ HE757 ;=>yes
*!ENDIF
  LDA #$FF
  JSR HE708
  STA NOUNSTKC,X
  BIT ACC+1
  BMI HE758 ;=>RTS

* tkn $36 -
*   unary sign of number
*   X = -5

NEGATE ;V
  JSR GET16BIT
HE772
  TYA ;Areg := 0
  SEC
  SBC ACC
  JSR HE708
  TYA
  SBC ACC+1
  BVC HE7A1 ;=>
HE77E
  LDY #ErrMsg00 ;">32767"
  BPL HE712 ;=>always

* tkn $13 -
*   num op
*   X=27-2

SUBTRACT ;V
  JSR NEGATE ;negate, then add

* tkn $12 +
*   num op
*   X=27+2

ADDITION ;VO
  JSR GET16BIT
  MOVW ACC;AUX
  JSR GET16BIT
HE793
  CLC
  LDA ACC
  ADC AUX
  JSR HE708
  LDA ACC+1
  ADC AUX+1
  BVS HE77E ;=>
HE7A1
  STA NOUNSTKC,X

* tkn $35 +
*   unary sign of number
*   X = +5

POSITIVE ;VO
  RTS
**

* tkn $50 TAB

TAB ;VO
  JSR GETBYTE
  TAY
*!IF <EQ>
    JMP HEECB ;range error?
*!ENDIF
  DEY
HE7AE ;solo
  JMP HF3F4
*>

* comma tab to next tab posn (every 8 spaces)

HE7B1
  LDA CH ;get horiz posn
  ORA #$07 ;set bits 0-2
  TAY
  INY ;incr, is it zero?
HE7B7 ;Z
  BNE HE7AE ;=>no, adjust CH
  INY ;yes, go to next tab posn
  BNE HE7B1 ;=>always
  BCS HE7B7 ;=>;Z
  RTS ;Z
**

  DB 0,0 ;Z

* tkn $49 ,
*   num print follows
*   PRINT A$,X

HE7C1 ;VO
  JSR HE7B1

* tkn $46 ;
*   num print follows
*   PRINT A$ ; X

* tkn $62 PRINT
*   num value
*   PRINT 123: PRINT X: PRINT ASC(A$)

PRNTNUM ;VO branch
  JSR GET16BIT
HE7C7 ;solo
  LDA ACC+1 ;is it positive?
*!IF <MI>
    LDA #"-" ;no, print minus sign
    JSR COUT
    JSR HE772
    BVC PRNTNUM ;=>always
*!ENDIF
```

```
16080   DEY ;Yreg := $FF
16090   STY CRFLAG ;CRFLAG := $FF
16100   STX ACC+1 ;save Xreg
16110   LDX ACC
16120   JSR PRDEC
16130   LDX ACC+1 ;restore Xreg
16140   RTS
16150  **
16160
16170  * tkn $0D AUTO
16180
16190  AUTO ;VO
16200   JSR GET16BIT
16210   MOVW ACC;AUTOLN
16220   DEY
16230   STY AUTOFLAG ;AUTOFLAG := $FF
16240   INY
16250   LDA #10 ;default increment
16260  HE7F3
16270   STA AUTOINC
16280   STY AUTOINC+1
16290   RTS
16300  **
16310
16320  * tkn $0E ,
16330  *   AUTO 10,20
16340
16350  COMMA_AUTO ;VO
16360   JSR GET16BIT
16370   LDA ACC
16380   LDY ACC+1
16390   BPL HE7F3 ;=>always
16400
16410  * tkn $56 =
16420  *   FOR X = 5 TO 10
16430
16440  * tkn $71 =
16450  *   num - non-conditional
16460  *   X = 5
16470
16480  HE801 ;V
16490   JSR GET16BIT
16500   LDA NOUNSTKL,X
16510   STA AUX
16520   LDA NOUNSTKH,X
16530   STA AUX+1
16540   LDA ACC
16550   STA (AUX),Y
16560   INY
16570   LDA ACC+1
16580   JMP HF207
16590  *>
16600
16610  * tkn $25 THEN
16620  *   IF X = 3 THEN Y = 5
16630
16640  * tkn $5E LET
16650
16660  LET ;VO
16670   RTS
16680  **
16690
16700  * tkn $00
16710  *   internal begin-of-line
16720
16730  BEGIN_LINE ;VO
16740   PLA
16750   PLA
16760
16770  * tkn $03 :
16780  *   statement separation
16790  *   X = 5: A$ = "HELLO"
16800
16810  COLON ;VO
16820   BIT CRFLAG
16830   BPL HE822 ;=>RTS
16840
16850  * tkn $63 PRINT
16860  *   dummy print
16870  *   PRINT: PRINT
16880
16890  PRINT_CR ;VO
16900   JSR CROUT
16910
16920  * tkn $47 ;
16930  *   end of print statement
16940  *   PRINT A$;
16950
16960  HE820 ;VO
16970   LSR CRFLAG ;pos
16980  HE822
16990   RTS
17000  **
17010
17020  * tkn $22 (
17030  *   string DIM
17040  *   DIM A$(X)
17050
17060  * tkn $34 (
17070  *   num DIM
17080  *   DIM X(5)
17090
17100  * tkn $38 (
17110  *   logic statements and num operations
17120  *   IF C AND (A=14 OR B=12) THEN
X=(27+3)/13
17130
17140  * tkn $3F (
17150  *   used after PEEK, RND, SGN, ABS, and PDL
17160
17170  HE823 ;V
17180   LDY #$FF
17190   STY PRFLAG ;PRFLAG := $FF
17200
17210  * tkn $72 )
17220  *   the only right parenthesis token
17230
17240  RIGHT_PAREN ;VO
17250   RTS
17260  **
17270
17280  * tkn $60 IF
17290
17300  IF ;VO
17310   JSR HEFCD
17320  *!IF <NE>
17330     LDA #$25 ;THEN token?
17340     STA VERBNOW
17350     DEY
17360     STY IFFLAG
17370  *!ENDIF
17380   INX
17390   RTS
17400  **
17410
17420  * RUN without CLR
17430  *   DOS 3.3 chains here to run a program
17440
17450  RUNWARM ;solo
17460   LDA PP
17470   LDY PP+1
17480   BNE HE896 ;=>always
17490
17500  * tkn $5C GOSUB
17510
17520  GOSUB ;VO
17530   LDY #ErrMsg08 ;"16 GOSUBS"
17540   LDA GOSUBNDX
17550   CMP #16 ;sixteen GOSUBs?
17560   BCS HE8A2 ;=HS> yes, error
17570   TAY
17580   INC GOSUBNDX
17590
17600   LDA PX
17610   STA STK_00,Y
17620   LDA PX+1
17630   STA STK_10,Y
17640
17650   LDA PR
17660   STA STK_20,Y
17670   LDA PR+1
17680   STA STK_30,Y
17690
17700  * tkn $24 THEN
17710  *   followed by a line number
17720  *   IF X=3 THEN 10
17730
17740  * tkn $5F GOTO
17750
17760  GOTO ;V
17770   JSR GET16BIT
17780   JSR HE56D
17790  *!IF <CS>
17800     LDY #ErrMsg07 ;"BAD BRANCH"
17810     BNE HE8A2 ;=>always
17820  *!ENDIF
17830   LDA P2
17840   LDY P2+1
17850
17860  * main loop for running Integer BASIC programs
17870
17880  *!LOOP
17890  *!  LOOP
17900       STA PR
17910       STY PR+1
17920       CLC
17930       ADC #$03
17940  *!    IF <CS>
17950         INY
17960  *!    ENDIF
17970  GETNEXT ;fetch next statement from text
source
17980       LDX #$FF
17990       STX RUNFLAG ;neg
18000       TXS
18010       STA PX
18020       STY PX+1
18030       JSR HF02E ;test for ctrl-C & TRACE mode
18040       LDY #0
18050  HE883
18060       JSR HE679 ;execute statement
18070       BIT RUNFLAG
18080       BPL END ;=>
18090       CLC
18100       LDY #0
18110       LDA PR
18120       ADC (PR),Y
18130       LDY PR+1
18140  *!    IF <CS>
18150         INY
18160  *!    ENDIF
18170  HE896
18180       CMP HIMEM
18190  *!  UNTIL <EQ>
18200      CPY HIMEM+1
18210  *!UNTIL <EQ>
18220   LDY #ErrMsg06 ;"NO END"
18230   LSR RUNFLAG ;pos
18240  HE8A2
18250   JMP ERRMESS
18260  *>
18270
18280  * tkn $5B RETURN
18290
18300  RETURN ;V
18310   LDY #ErrMsg09 ;"BAD RETURN"
18320   LDA GOSUBNDX
18330   BEQ HE8A2 ;=>
18340   DEC GOSUBNDX
18350   TAY
18360   LDA STK_20-1,Y
18370   STA PR
18380   LDA STK_30-1,Y
18390   STA PR+1
18400   LDX: STK_00-1,Y
18410   LDA STK_10-1,Y
18420  HE8BE
18430   TAY
18440   TXA
18450   JMP GETNEXT
18460  *>
18470
18480  STOPPED_AT
18490   LDY #ErrMsg12 ;"STOPPED AT "
18500   JSR ERRORMESS
18510   LDY #1
18520   LDA (PR),Y
18530   TAX
18540   INY
18550   LDA (PR),Y
18560   JSR PRDEC
18570
18580  * tkn $51 END
18590
18600  END ;V
18610   JMP WARM
18620  *>
18630
18640  *!LOOP
18650  *!  LOOP
18660       DEC FORNDX
18670
18680  * tkn $59 NEXT
18690
18700  * tkn $5A ,
18710  *   NEXT X,Y
18720
18730  NEXT ;VO
18740       LDY #ErrMsg11 ;"BAD NEXT"
18750       LDA FORNDX
18760  HE8DC
18770       BEQ HE8A2 ;=>no more FORs
18780       TAY
```

```
18790       LDA NOUNSTKL,X
18800       CMP STK_40-1,Y
18810 *!  UNTIL <EQ>
18820     LDA NOUNSTKH,X
18830     CMP STK_50-1,Y
18840 *!UNTIL <EQ>
18850
18860   LDA STK_60-1,Y
18870   STA AUX
18880   LDA STK_70-1,Y
18890   STA AUX+1
18900
18910   JSR GET16BIT
18920   DEX
18930   JSR HE793
18940   JSR HE801
18950   DEX
18960   LDY FORNDX
18970   LDA STK_D0-1,Y
18980   STA NOUNSTKC-1,X
18990   LDA STK_C0-1,Y
19000   LDY #0
19010   JSR HE708
19020   JSR SUBTRACT
19030   JSR SGN
19040   JSR GET16BIT
19050   LDY FORNDX
19060   LDA ACC
19070 *!IF <NE>
19080     EOR STK_70-1,Y
19090     BPL HE937 ;=>
19100 *!ENDIF
19110
19120   LDA STK_80-1,Y
19130   STA PR
19140   LDA STK_90-1,Y
19150   STA PR+1
19160
19170   LDX STK_A0-1,Y
19180   LDA STK_B0-1,Y
19190   BNE HE8BE ;=>
19200 HE937
19210   DEC FORNDX
19220   RTS
19230 **
19240
19250 * tkn $55 FOR
19260
19270 FOR ;VO
19280   LDY #ErrMsg10 ;"16 FORS"
19290   LDA FORNDX
19300   CMP #16 ;sixteen FORs?
19310   BEQ HE8DC ;=>yes, error
19320   INC FORNDX
19330   TAY
19340   LDA NOUNSTKL,X
19350   STA STK_40,Y
19360   LDA NOUNSTKH,X
19370   JMP HF288
19380 *>
19390
19400   RTS ;Z
19410 **
19420
19430 * tkn $57 TO
19440
19450 TO ;VO
19460   JSR GET16BIT
19470   LDY FORNDX
19480
19490   LDA ACC
19500   STA STK_C0-1,Y
19510   LDA ACC+1
19520   STA STK_D0-1,Y
19530
19540   LDA #<$0001
19550   STA STK_60-1,Y
19560   LDA #>$0001
19570 HE966 ;solo
19580   STA STK_70-1,Y
19590
19600   LDA PR
19610   STA STK_80-1,Y
19620   LDA PR+1
19630   STA STK_90-1,Y
19640
19650   LDA PX
19660   STA STK_A0-1,Y
19670   LDA PX+1
19680   STA STK_B0-1,Y
19690   RTS
```

```
19700 **
19710
19720   DB $20,$15 ;Z
19730
19740
19750   PUT TABLE1
19760 HE980
19770   DB $00,$00,$00,$AB,$03,$03,$03,$03
19780   DB $03,$03,$03,$03,$03,$03,$03,$03
19790   DB $03,$03,$3F,$3F,$C0,$C0,$3C,$3C
19800   DB $3C,$3C,$3C,$3C,$3C,$30,$0F,$C0
19810   DB $C3,$FF,$55,$00,$AB,$AB,$03,$03
19820   DB $FF,$FF,$55,$FF,$FF,$55,$CF,$CF
19830   DB $CF,$CF,$CF,$FF,$55,$C6,$C6,$C6
19840   DB $55,$F0,$F0,$CF,$CF,$55,$01,$55
19850   DB $FF,$FF,$55,$03,$03,$03,$03,$03
19860   DB $03,$03,$03,$03,$03,$03,$03,$03
19870   DB $03,$03,$03,$03,$03,$03,$03,$03
19880   DB $03,$03,$03,$03,$03,$00,$AB,$03
19890   DB $57,$03,$03,$03,$03,$07,$03,$03
19900   DB $03,$03,$03,$03,$03,$03,$03,$03
19910   DB $03,$03,$AA,$FF,$03,$03,$03,$03
19920   DB $03,$03,$03,$03,$03,$03,$03,$03
19930
19940 * token address tables (verb dispatch tables)
19950
19960 VERBADRL
19970   DB <BEGIN_LINE,<$FFFF,<$FFFF,<COLON
19980   DB <LOAD,<SAVE,<CON,<RUNNUM
19990   DB <RUN,<DEL,<COMMA_DEL,<NEW
20000   DB <CLR,<AUTO,<COMMA_AUTO,<MAN
20010   DB
<VHIMEM,<VLOMEM,<ADDITION,<SUBTRACT
20020   DB <MULT,<DIVIDE,<HE733,<HE74A
20030   DB <HF25B,<HF24E,<HF253,<HE74A
20040   DB <HF249,<VAND,<VOR,<MOD
20050   DB
<EXP,<$FFFF,<HE823,<COMMA_SUBSTR
20060   DB <GOTO,<LET,<HEFB6,<HEBCB
20070   DB
<$FFFF,<$FFFF,<PAREN_SUBSTR,<$FFFF
20080   DB <$FFFF,<HEF24,<PEEK,<RND
20090   DB <SGN,<ABS,<PDL,<$FFFF
20100   DB <HE823,<POSITIVE,<NEGATE,<NOT
20110   DB <HE823,<HE1D7,<HE21C,<LEN
20120   DB <ASC,<SCRN,<COMMA_SCRN,<HE823
20130   DB <$FFFF,<$FFFF,<HE121,<DIMSTR
20140   DB
<DIMNUM,<PRNTSTR,<PRNTNUM,<HE820
20150   DB <HEE00,<HE7C1,<HF3BA,<SETTXT
20160   DB <SETGR,<CALL,<DIMSTR,<DIMNUM
20170   DB <TAB,<END,<HEFB6,<INPUT_PROMPT
20180   DB <HEBAA,<FOR,<HE801,<TO
20190   DB <STEP,<NEXT,<NEXT,<RETURN
20200   DB <GOSUB,<$FFFF,<LET,<GOTO
20210   DB <IF,<PRNTSTR,<PRNTNUM,<PRINT_CR
20220   DB
<POKE,<GETVAL255,<COLOR,<GETVAL255
20230   DB
<COMMA_PLOT,<GETVAL255,<COMMA_HLIN,<AT_H
LIN
20240   DB
<GETVAL255,<COMMA_VLIN,<AT_VLIN,<IVTAB
20250   DB
<HE18C,<HE801,<RIGHT_PAREN,<$FFFF
20260   DB <LISTNUM,<COMMA_LIST,<LIST,<POP
20270   DB
<NODSP_STR,<NODSP_NUM,<NOTRACE,<DSP_NU
M
20280   DB <DSP_STR,<TRACE,<PRSLOT,<INSLOT
20290
20300 VERBADRH
20310   DB >BEGIN_LINE,>$FFFF,>$FFFF,>COLON
20320   DB >LOAD,>SAVE,>CON,>RUNNUM
20330   DB >RUN,>DEL,>COMMA_DEL,>NEW
20340   DB >CLR,>AUTO,>COMMA_AUTO,>MAN
20350   DB
>VHIMEM,>VLOMEM,>ADDITION,>SUBTRACT
20360   DB >MULT,>DIVIDE,>HE733,>HE74A
20370   DB >HF25B,>HF24E,>HF253,>HE74A
20380   DB >HF249,>VAND,>VOR,>MOD
20390   DB
>EXP,>$FFFF,>HE823,>COMMA_SUBSTR
20400   DB >GOTO,>LET,>HEFB6,>HEBCB
20410   DB
>$FFFF,>$FFFF,>PAREN_SUBSTR,>$FFFF
20420   DB >$FFFF,>HEF24,>PEEK,>RND
20430   DB >SGN,>ABS,>PDL,>$FFFF
20440   DB >HE823,>POSITIVE,>NEGATE,>NOT
20450   DB >HE823,>HE1D7,>HE21C,>LEN
20460   DB >ASC,>SCRN,>COMMA_SCRN,>HE823
```

```
20470   DB >$FFFF,>$FFFF,>HE121,>DIMSTR
20480   DB
>DIMNUM,>PRNTSTR,>PRNTNUM,>HE820
20490   DB >HEE00,>HE7C1,>HF3BA,>SETTXT
20500   DB >SETGR,>CALL,>DIMSTR,>DIMNUM
20510   DB >TAB,>END,>HEFB6,>INPUT_PROMPT
20520   DB >HEBAA,>FOR,>HE801,>TO
20530   DB >STEP,>NEXT,>NEXT,>RETURN
20540   DB >GOSUB,>$FFFF,>LET,>GOTO
20550   DB >IF,>PRNTSTR,>PRNTNUM,>PRINT_CR
20560   DB
>POKE,>GETVAL255,>COLOR,>GETVAL255
20570   DB
>COMMA_PLOT,>GETVAL255,>COMMA_HLIN,>AT_H
LIN
20580   DB
>GETVAL255,>COMMA_VLIN,>AT_VLIN,>IVTAB
20590   DB
>HE18C,>HE801,>RIGHT_PAREN,>$FFFF
20600   DB >LISTNUM,>COMMA_LIST,>LIST,>POP
20610   DB
>NODSP_STR,>NODSP_NUM,>NOTRACE,>DSP_NU
M
20620   DB >DSP_STR,>TRACE,>PRSLOT,>INSLOT
20630
20640 ErrorMsgs
20650
20660 ErrMsg00 = *-ErrorMsgs+$8100 ;00
20670   DCI ">32767"
20680
20690 ErrMsg01 = *-ErrorMsgs+$8100 ;06
20700   DCI "TOO LONG"
20710
20720 ErrMsg02 = *-ErrorMsgs+$8100 ;0E
20730   DCI "SYNTAX"
20740
20750 ErrMsg03 = *-ErrorMsgs+$8100 ;14
20760   DCI "MEM FULL"
20770
20780 ErrMsg04 = *-ErrorMsgs+$8100 ;1C
20790   DCI "TOO MANY PARENS"
20800
20810 ErrMsg05 = *-ErrorMsgs+$8100 ;2B
20820   DCI "STRING"
20830
20840 ErrMsg06 = *-ErrorMsgs+$8100 ;31
20850   DCI "NO END"
20860
20870 ErrMsg07 = *-ErrorMsgs+$8100 ;37
20880   DCI "BAD BRANCH"
20890
20900 ErrMsg08 = *-ErrorMsgs+$8100 ;41
20910   DCI "16 GOSUBS"
20920
20930 ErrMsg09 = *-ErrorMsgs+$8100 ;4A
20940   DCI "BAD RETURN"
20950
20960 ErrMsg10 = *-ErrorMsgs+$8100 ;54
20970   DCI "16 FORS"
20980
20990 ErrMsg11 = *-ErrorMsgs+$8100 ;5B
21000   DCI "BAD NEXT"
21010
21020 ErrMsg12 = *-ErrorMsgs+$8100 ;63
21030   DCI "STOPPED AT "
21040

21050 ErrMsg13 = *-ErrorMsgs+$8100 ;6E
21060   DCI "*** "
21070
21080 ErrMsg14 = *-ErrorMsgs+$8100 ;72
21090   ASC " ERR"
21100   DB  CR
21110
21120 ErrMsg15 = *-ErrorMsgs+$8100 ;77
21130   DCI ">255"
21140
21150 ErrMsg16 = *-ErrorMsgs+$8100 ;7B
21160   DCI "RANGE"
21170
21180 ErrMsg17 = *-ErrorMsgs+$8100 ;80
21190   DCI "DIM"
21200
21210 ErrMsg18 = *-ErrorMsgs+$8100 ;83
21220   DCI "STR OVFL"
21230
21240   ASC "\" ;8B
21250   DB  CR
21260
21270 ErrMsg20 = *-ErrorMsgs+$8100 ;8D
21280   ASC "RETYPE LINE"
```

```
21290  DB  CR+$80
21300
21310 ErrMsg21 = *-ErrorMsgs+$8100 ;99
21320  ASC '?'
21330
21340  PUT PART2
21350
21360 *continue run w/o deleting vars?
21370
21380 HEB9A ;solo
21390   LSR RUNFLAG ;pos
21400 *!IF <CS>
21410     JMP STOPPED_AT
21420 *!ENDIF
21430   LDX ACC+1
21440   TXS
21450   LDX ACC
21460   LDY #ErrMsg20 ;"RETYPE LINE",CR,"?"
21470   BNE HEBAC ;=>always
21480
21490 * tkn $54 INPUT
21500 *   num with no prompt
21510 *   INPUT X
21520
21530 HEBAA ;VO branch
21540   LDY #ErrMsg21 ;'?' for INPUT
21550 HEBAC
21560   JSR ERRORMESS
21570   STX ACC
21580   TSX
21590   STX ACC+1
21600   JSR HF366
21610   STY TOKNDX
21620   LDA #$FF
21630   STA TXTNDX
21640   ASL
21650   STA RUNFLAG ;neg
21660   LDX #$20
21670   LDA #$15
21680   JSR HE491
21690   INC RUNFLAG
21700   LDX ACC
21710
21720 * tkn $27 ,
21730 *   num inputs
21740 *   INPUT "QUANTITY",Q
21750
21760 HEBCB ;VO
21770   LDY TXTNDX
21780   ASL
21790 *!LOOP
21800     STA ACC
21810     INY
21820     LDA IN,Y
21830     CMP #$80
21840     BEQ HEBAA ;=>end of input?
21850     EOR #"0"
21860     CMP #10
21870 *!UNTIL <LO>
21880   INY
21890   INY
21900   STY TXTNDX
21910   LDA IN,Y
21920   PHA
21930   LDA IN-1,Y
21940   LDY #0
21950   JSR HE708
21960   PLA
21970   STA NOUNSTKC,X
21980   LDA ACC
21990   CMP #$33
22000 *!IF <EQ>
22010     JSR NEGATE
22020 *!ENDIF
22030   JMP HE801
22040 *>
22050
22060
22070
22080  DB $FF,$FF,$FF ;Z
22090
22100  PUT TABLE2
22110 * token/syntax table
22120
22130 SYNTABL
22140  DB $50
22150
22160  DB $20,$4F,$C0 ;Z
22170  DB "T"+32,"A"-32 ;Z
22180  DB "D"+32,"O"-32,"M"-32 ;Z
22190  DB "R"+32,"O"-32 ;Z
22200   DB "D"+32,"N"-32,"A"-32 ;Z
22210   DB "P"+32,"E"-32,"T"-32,"S"-32 ;Z
22220   DB "O"+32,"T"-32 ;Z
22230   DB "N"+32,"E"-32,"H"-32,"T"-32 ;Z
22240
22250   DB $5C,$80,$00,$40
22260   DB $60,$8D,$60,$8B,$7F,$1D,$20,$7E
22270   DB $8C,$33,$00,$00,$60,$03,$BF,$12
22280
22290   DB $47,"#"-32,"N"-32,"I"-32 ;IN#
22300   DB $67,"#"-32,"R"-32,"P"-32 ;PR#
22310   DB "E"+32,"C"-32,"A"-32,"R"-32,"T"-32 ;TRACE
22320   DB $79,"P"-32,"S"-32,"D"-32 ;DSP
22330   DB $69,"P"-32,"S"-32,"D"-32 ;DSP
22340   DB "E"+32,"C"-32,"A"-32,"R"-32,"T"-32,"O"-
32,"N"-32 ;NOTRACE
22350   DB $79,"P"-32,"S"-32,"D"-32,"O"-32,"N"-
32 ;NODSP
22360   DB $69,"P"-32,"S"-32,"D"-32,"O"-32,"N"-
32 ;NODSP
22370   DB "P"+32,"O"-32,"P"-32 ;POP
22380   DB "T"+32,"S"-32,"I"-32,"L"-32 ;LIST
22390   DB $60,","-32 ;
22400   DB $20,"T"-32,"S"-32,"I"-32,"L"-32 ;LIST
22410   DB 0
22420   DB $40,$89
22430   DB ")"+32 ;
22440   DB $47,"="-32 ;
22450   DB $17,$68,"="-32 ;
22460   DB $0A,$58,$7B,$67,"B"-32,"A"-32,"T"-32,"V"-
32 ;VTAB
22470   DB $67,"T"-32,"A"-32 ;AT
22480   DB $07,","-32 ;
22490   DB $07,"N"-32,"I"-32,"L"-32,"V"-32 ;VLIN
22500   DB $67,"T"-32,"A"-32 ;AT
22510   DB $07,","-32 ;
22520   DB $07,"N"-32,"I"-32,"L"-32,"H"-32 ;HLIN
22530   DB $67,","-32 ;
22540   DB $07,"T"-32,"O"-32,"L"-32,"P"-32 ;PLOT
22550   DB $67,"="-32,"R"-32,"O"-32,"L"-32,"O"-32,"C"-
32 ;COLOR=
22560   DB $67,","-32 ;
22570   DB $07,"E"-32,"K"-32,"O"-32,"P"-32 ;POKE
22580   DB "T"+32,"N"-32,"I"-32,"R"-32,"P"-32 ;PRINT
22590   DB $7F,$0E,$27,"T"-32,"N"-32,"I"-32,"R"-
32,"P"-32 ;PRINT
22600   DB $7F,$0E,$28,"T"-32,"N"-32,"I"-32,"R"-
32,"P"-32 ;PRINT
22610   DB $64,$07,"F"-32,"I"-32 ;IF
22620   DB $67,"O"-32,"T"-32,"O"-32,"G"-32 ;GOTO
22630   DB $78,"T"-32,"E"-32,"L"-32 ;LET
22640   DB $6B,$7F,$02,"M"-32,"E"-32,"R"-32 ;REM
22650   DB $67,"B"-32,"U"-32,"S"-32,"O"-32,"G"-
32 ;GOSUB
22660   DB "N"+32,"R"-32,"U"-32,"T"-32,"E"-32,"R"-
32 ;RETURN
22670   DB $7E,","-32 ;
22680   DB $39,"T"-32,"X"-32,"E"-32,"N"-32 ;NEXT
22690   DB $67,"P"-32,"E"-32,"T"-32,"S"-32 ;STEP
22700   DB $27,"O"-32,"T"-32 ;TO
22710   DB $07,"="-32 ;
22720   DB $19,"R"-32,"O"-32,"F"-32 ;FOR
22730   DB $7F,$05,$37,"T"-32,"U"-32,"P"-32,"N"-
32,"I"-32 ;INPUT
22740   DB $7F,$05,$28,"T"-32,"U"-32,"P"-32,"N"-
32,"I"-32 ;INPUT
22750   DB $7F,$05,$2A,"T"-32,"U"-32,"P"-32,"N"-
32,"I"-32 ;INPUT
22760   DB "D"+32,"N"-32,"E"-32 ;END (tkn $51)
22770
22780  SYNTABL2
22790   DB 0
22800   DB $47,"B"-32,"A"-32,"T"-32 ;TAB (tkn $50)
22810   DB $7F,$0D,$30,"M"-32,"I"-32,"D"-32 ;DIM
22820   DB $7F,$0D,$23,"M"-32,"I"-32,"D"-32 ;DIM
22830   DB $67,"L"-32,"L"-32,"A"-32,"C"-32 ;CALL
22840   DB "R"+32,"G"-32 ;GR
22850   DB "T"+32,"X"-32,"E"-32,"T"-32 ;TEXT
22860   DB 0 ;above are statements
22870   DB $4D,","+32 ;
22880   DB $67,","-32 ;
22890   DB $68,","-32 ;
22900   DB ";"+32 ;
22910   DB $67,";"-32 ;
22920   DB $68,";"-32 ;
22930   DB $50,","-32 ;
22940   DB $63,","-32 ;
22950   DB $7F,$01,$51,$07,"("-32 ;
22960   DB $29,$84
22970   DB $80,"$"+32 ;
22980   DB $19,$57,$71,$07,"("-32 ;
22990   DB $14,$71,$07,","-32 ;
23000   DB $07,"("-32,"N"-32,"R"-32,"C"-32,"S"-
32 ;SCRN(
23010   DB $71,$08,"("-32,"C"-32,"S"-32,"A"-32 ;ASC(
23020   DB $71,$08,"("-32,"N"-32,"E"-32,"L"-32 ;LEN(
23030   DB $68,"#"-32 ;
23040   DB $08,$68,"="-32 ;
23050   DB $08,$71,$07,"("-32 ;
23060   DB $60,$75,"T"-32,"O"-32,"N"-32 ;NOT
23070   DB $75,"-"-32 ;
23080   DB $75,"+"-32 ;
23090   DB $51,$07,"("-32,$19 ;
23100   DB "X"-32,"D"-32,"N"-32,"R"-32
23110   DB "L"+32,"D"-32,"P"-32 ;PDL
23120   DB "S"+32,"B"-32,"A"-32 ;ABS
23130   DB "N"+32,"G"-32,"S"-32 ;SGN
23140   DB "D"+32,"N"-32,"R"-32 ;RND
23150   DB "K"+32,"E"-32,"E"-32,"P"-32 ;PEEK
23160   DB $51,$07,"("-32 ;
23170   DB $39,$81,$C1,$4F,$7F,$0F,$2F
23180   DB 0 ;above are functions
23190   DB $51,$06,"("-32 ;
23200   DB $29,""""+32 ;open quote
23210   DB $0C,""""-32 ;close quote
23220   DB $57,","-32 ;
23230   DB $6A,","-32 ;
23240   DB $42,"N"-32,"E"-32,"H"-32,"T"-32 ;THEN
23250   DB $60,"N"-32,"E"-32,"H"-32,"T"-32 ;THEN
23260   DB $4F,$7E,$1E,$35,","-32 ;
23270   DB $27,$51,$07,"("-32 ;
23280   DB $09,"+"-32
23290   DB "^"+32 ;exponent
23300   DB "D"+32,"O"-32,"M"-32 ;MOD
23310   DB "R"+32,"O"-32 ;OR
23320   DB "D"+32,"N"-32,"A"-32 ;AND
23330   DB "<"+32 ;less than
23340   DB ">"+32,"<"-32 ;not equal
23350   DB "="+32,"<"-32 ;less or equal
23360   DB ">"+32 ;greater than
23370   DB "="+32,">"-32 ;greater or equal
23380   DB "#"+32 ;not equal
23390   DB "="+32 ;equal
23400   DB "/"+32 ;divide
23410   DB "*"+32 ;multiply
23420   DB "-"+32 ;subtract
23430   DB "+"+32 ;add
23440   DB 0 ;above 4 are num ops
23450   DB $47,":"-32,"M"-32,"E"-32,"M"-32,"O"-32,"L"-
32 ;LOMEM:
23460   DB $67,":"-32,"M"-32,"E"-32,"M"-32,"I"-32,"H"-
32 ;HIMEM:
23470   DB "N"+32,"A"-32,"M"-32 ;MAN
23480   DB $60,","-32 ;comma for AUTO
23490   DB $20,"O"-32,"T"-32,"U"-32,"A"-32 ;AUTO
23500   DB "R"+32,"L"-32,"C"-32 ;CLR
23510   DB "W"+32,"E"-32,"N"-32 ;NEW
23520   DB $60,","-32 ;comma for DEL
23530   DB $20,"L"-32,"E"-32,"D"-32 ;DEL
23540   DB "N"+32,"U"-32,"R"-32 ;RUN
23550   DB $60,"N"-32,"U"-32,"R"-32 ;RUN
23560   DB "N"+32,"O"-32,"C"-32 ;CON
23570   DB "E"+32,"V"-32,"A"-32,"S"-32 ;SAVE
23580   DB "D"+32,"A"-32,"O"-32,"L"-32 ;LOAD
23590  *above are commands
23600   DB $7A,$7E,$9A,$22,$20
23610   DB $00,$60,$03,$BF,$60,$03,$BF,$1F
23620
23630
23640   PUT PART3
23650  * tkn $48 ,
23660  *   string prints
23670  *   PRINT T,A$
23680
23690  HEE00 ;VO
23700    JSR HE7B1
23710
23720  * tkn $45 ;
23730  *   string prints
23740  *   PRINT anytype ; string
23750
23760  * tkn $61 PRINT
23770  *   string var or literal
23780  *   PRINT A$: PRINT "HELLO"
23790
23800  PRNTSTR ;V
23810    INX
23820    INX
23830    LDA NOUNSTKL-1,X
23840    STA AUX
23850    LDA NOUNSTKH-1,X
23860    STA AUX+1
```

```
23870   LDY NOUNSTKL-2,X
23880  HEE0F ;*!LOOP
23890     TYA
23900     CMP NOUNSTKH-2,X
23910     BCS HEE1D ;=HS>exit loop
23920     LDA (AUX),Y
23930     JSR COUT
23940     INY
23950     JMP HEE0F ;*!loop always
23960  HEE1D
23970   LDA #$FF
23980   STA CRFLAG ;CRFLAG := $FF
23990   RTS
24000  **
24010
24020  * tkn $3B LEN(
24030
24040  LEN ;VO
24050   INX
24060   LDA #0
24070   STA NOUNSTKH,X
24080   STA NOUNSTKC,X
24090   LDA NOUNSTKH-1,X
24100   SEC
24110   SBC NOUNSTKL-1,X
24120   STA NOUNSTKL,X
24130   JMP HE823
24140  *>
24150
24160   DB $FF ;Z
24170
24180  GETBYTE
24190   JSR GET16BIT
24200   LDA ACC+1
24210   BNE HI255ERR ;=>">255" error
24220   LDA ACC
24230   RTS
24240  **
24250
24260  * tkn $68 ,
24270  *   PLOT 20,15
24280
24290  COMMA_PLOT ;VO
24300   JSR GETBYTE
24310   LDY TXTNDX
24320   CMP #48
24330   BCS RANGERR ;=HS>
24340   CPY #40
24350   BCS RANGERR ;=HS>
24360   JMP PLOT
24370  *>
24380
24390  * tkn $66 COLOR=
24400
24410  COLOR ;VO
24420   JSR GETBYTE
24430   JMP SETCOL
24440  *>
24450
24460  * tkn $0F MAN
24470
24480  MAN
24490   LSR AUTOFLAG ;manual
24500   RTS
24510  **
24520
24530  * tkn $6F VTAB
24540
24550  IVTAB ;VO
24560   JSR HF3B3
24570   CMP #24
24580   BCS RANGERR ;=HS>
24590   STA CV
24600   JMP VTAB
24610  *>
24620
24630  HI255ERR
24640   LDY #ErrMsg15 ;">255"
24650  HEE65
24660   JMP ERRMESS
24670  *>
24680
24690  RANGERR
24700   LDY #ErrMsg16 ;"RANGE"
24710   BNE HEE65 ;=>always
24720
24730  * divide routine
24740
24750  HEE6C
24760   JSR HE254
24770   LDA AUX ;is AUX zero?
24780  *!IF <EQ>
24790     LDA AUX+1
24800  *!  IF <EQ>
24810     JMP HE77E ;yes, ">32767" error
24820  *!  ENDIF
24830  *!ENDIF
24840  *!LOOP
24850     ASL ACC
24860     ROL ACC+1
24870     ROL P3
24880     ROL P3+1
24890     CMPW P3;AUX
24900  *!  IF <HS>
24910       STA P3+1 ;P3 := P3-AUX
24920       LDA P3
24930       SBC AUX
24940       STA P3
24950       INC ACC
24960  *!  ENDIF
24970     DEY
24980  *!UNTIL <EQ>
24990   RTS
25000  **
25010
25020   DB $FF,$FF,$FF,$FF,$FF,$FF ;Z
25030
25040  * tkn $4D CALL
25050
25060  CALL ;VO
25070   JSR GET16BIT
25080   JMP (ACC)
25090  *>
25100
25110  * tkn $6A ,
25120  *   HLIN 10,20 AT 30
25130
25140  COMMA_HLIN ;VO
25150   JSR GETBYTE
25160   CMP TXTNDX
25170   BCC RANGERR ;=LO>
25180   STA H2
25190   RTS
25200  **
25210
25220  * tkn $6B AT
25230  *   HLIN 10,20 AT 30
25240
25250  AT_HLIN ;VO
25260   JSR GETBYTE
25270   CMP #48
25280   BCS RANGERR ;=HS>
25290   LDY TXTNDX
25300   JMP HLINE
25310  *>
25320
25330  * tkn $6D ,
25340  *   VLIN 10,20 AT 30
25350
25360  COMMA_VLIN ;VO
25370   JSR GETBYTE
25380   CMP TXTNDX
25390   BCC RANGERR ;=LO>
25400   STA V2
25410   RTS
25420  **
25430
25440  * tkn $6E AT
25450  *   VLIN 10,20 AT 30
25460
25470  AT_VLIN ;VO
25480   JSR GETBYTE
25490   CMP #40
25500  HEECB
25510   BCS RANGERR ;=HS>
25520   TAY
25530   LDA TXTNDX
25540   JMP VLINE
25550  *>
25560
25570  PRINTERR
25580   TYA
25590   TAX
25600   LDY #ErrMsg13 ;"*** "
25610   JSR ERRORMESS
25620   TXA
25630   TAY
25640   JSR ERRORMESS
25650   LDY #ErrMsg14 ;" ERR"
25660   JMP PRTERR
25670  *>
25680
25690  HEEE4
25700   JSR HF23F
25710  *!LOOP
25720     ASL ACC
25730     ROL ACC+1
25740  *!UNTIL <PL>
25750   BCS HEECB ;=>"RANGE" error
25760  *!IF <EQ>
25770     CMP ACC
25780     BCS HEECB ;=HS>"RANGE" error
25790  *!ENDIF
25800   RTS
25810  **
25820
25830  * tkn $2E PEEK
25840  *   uses tkn $3F (
25850
25860  PEEK ;VO
25870   JSR GET16BIT
25880   LDA (ACC),Y
25890   STY NOUNSTKC-1,X
25900   JMP HE708
25910  *>
25920
25930  * tkn $65 ,
25940  *   POKE 20000,5
25950
25960  * tkn $67 PLOT
25970
25980  * tkn $69 HLIN
25990
26000  * tkn $6C VLIN
26010
26020  GETVAL255 ;VO
26030   JSR GETBYTE
26040   LDA ACC
26050   STA TXTNDX
26060   RTS
26070  **
26080
26090  * tkn $64 POKE
26100
26110  POKE ;VO
26120   JSR GET16BIT
26130   LDA TXTNDX
26140   STA (ACC),Y
26150   RTS
26160  **
26170
26180  * tkn $15 /
26190  *   num op.  uses $38 (
26200  *   A = 27 / 2
26210
26220  DIVIDE ;VO
26230   JSR HEE6C
26240   MOVW ACC;P3
26250   JMP HE244
26260  *>
26270
26280  * tkn $44 ,
26290  *   next var in DIM is num
26300  *   DIM X(5),A(5)
26310
26320  * tkn $4F DIM
26330  *   num var.  uses tkn $22 (
26340  *   DIM A(5)
26350
26360  DIMNUM ;VO
26370   JSR HEEE4
26380   JMP HE134
26390  *>
26400
26410  * tkn $2D (
26420  *   var array
26430  *   X(12)
26440
26450  HEF24 ;VO
26460   JSR HEEE4
26470   LDY NOUNSTKH,X
26480   LDA NOUNSTKL,X
26490   ADC #$FE
26500  *!IF <CC>
26510     DEY
26520  *!ENDIF
26530   STA AUX
26540   STY AUX+1
26550   CLC
26560   ADC ACC
26570   STA NOUNSTKL,X
26580   TYA
26590   ADC ACC+1
```

```
26600   STA NOUNSTKH,X
26610   LDY #0
26620   LDA NOUNSTKL,X
26630   CMP (AUX),Y
26640   INY
26650   LDA NOUNSTKH,X
26660   SBC (AUX),Y
26670   BCS HEECB ;=HS>"RANGE" error
26680   JMP HE823
26690 *>
26700
26710 * tkn $2F RND
26720 *   uses tkn $3F (
26730
26740 RND ;VO
26750   JSR GET16BIT
26760   LDA RNDL
26770   JSR HE708
26780   LDA RNDH
26790 *!IF <EQ>
26800     CMP RNDL
26810     ADC #0
26820 *!ENDIF
26830   AND #$7F
26840   STA RNDH
26850   STA NOUNSTKC,X
26860   LDY #$11
26870 *!LOOP
26880     LDA RNDH
26890     ASL
26900     CLC
26910     ADC #$40
26920     ASL
26930     ROL RNDL
26940     ROL RNDH
26950     DEY
26960 *!UNTIL <EQ>
26970   LDA ACC
26980   JSR HE708
26990   LDA ACC+1
27000   STA NOUNSTKC,X
27010   JMP MOD
27020 *>
27030
27040   JSR GET16BIT ;Z
27050   LDY ACC ;is ACC <LO> LOMEM?
27060   CPY LOMEM
27070   LDA ACC+1
27080   SBC LOMEM+1
27090   BCC HEFAB ;=LO>yes
27100   STY HIMEM ;HIMEM := ACC
27110   LDA ACC+1
27120   STA HIMEM+1
27130 HEF93 ;Z
27140   JMP NEW
27150 *>
27160
27170   JSR GET16BIT ;Z
27180   LDY ACC ;is ACC <HS> LOMEM?
27190   CPY HIMEM
27200   LDA ACC+1
27210   SBC HIMEM+1
27220   BCS HEFAB ;=HS>yes
27230   STY LOMEM ;LOMEM := ACC
27240   LDA ACC+1
27250   STA LOMEM+1
27260   BCC HEF93 ;=LO>always
27270
27280 HEFAB ;Z
27290   JMP HEECB ;range error?
27300 *>
27310
27320   DB $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF ;Z
27330
27340 * tkn $26 ,
27350 *   string inputs
27360 *   INPUT "WHO",W$
27370
27380 * tkn $52 INPUT
27390 *   string with no prompt
27400 *   INPUT S$
27410
27420 HEFB6 ;VO
27430   JSR INPUTSTR
27440   JMP HEFBF
27450 *>
27460
27470 * tkn $53 INPUT
27480 *   string or num with prompt
27490 *   INPUT "WHO",W$: INPUT "QUANTITY",Q
27500
27510 INPUT_PROMPT ;VO
27520   JSR PRNTSTR
27530 HEFBF
27540   LDA #$FF
27550   STA TXTNDX
27560   LDA #$80
27570   STA IN
27580   RTS
27590 **
27600
27610 HEFC9
27620   JSR NOT
27630   INX
27640 HEFCD ;solo
27650   JSR NOT
27660   LDA NOUNSTKL,X
27670   RTS
27680 **
27690
27700 * old 4K cold start
27710
27720 HEFD3 ;Z
27730   LDA #0
27740   STA LOMEM ;LOMEM := $0800
27750   STA HIMEM ;HIMEM := $1000
27760   LDA #>$0800
27770   STA LOMEM+1
27780   LDA #>$1000
27790   STA HIMEM+1
27800   JMP NEW
27810 *>
27820
27830 HEFE4 ;solo
27840   CMP NOUNSTKH,X
27850 *!IF <EQ>
27860     CLC
27870 *!ENDIF
27880   JMP HE102
27890 *>
27900
27910 * tkn $08 RUN
27920 *   run from first line of program
27930
27940 RUN ;VO
27950   JSR CLR
27960   JMP RUNWARM
27970 *>
27980
27990 * tkn $07 RUN
28000 *   RUN 100
28010
28020 RUNNUM ;VO
28030   JSR CLR
28040   JMP GOTO
28050 *>
28060
28070 HEFF8 ;solo
28080   CPX #$80
28090 *!IF <EQ>
28100     DEY
28110 *!ENDIF
28120   JMP HE00C
28130 *>
28140
28142 * Cold start
28150 *   set LOMEM, find HIMEM
28152 *   fall into NEW
28160
28170 COLD
28180   LDY #<$0800
28190   STY NOUNSTKC
28200   STY LOMEM ;LOMEM := $0800
28210   STY HIMEM ;HIMEM := $0800
28220   LDA #>$0800
28230   STA LOMEM+1
28240   STA HIMEM+1
28250 *!LOOP
28260     INC HIMEM+1 ;find top of RAM
28270     LDA (HIMEM),Y
28280     EOR #$FF
28290     STA (HIMEM),Y
28300     CMP (HIMEM),Y
28310 *!WHILE <EQ>
28320     EOR #$FF
28330     STA (HIMEM),Y
28340     CMP (HIMEM),Y
28350 *!UNTIL <NE>
28360   JMP NEW
28370 *>
28380
28390 HF025 ;solo
28400   JMP HF179
28410 *>
28420
28430   JSR HF032 ;Z
28440   JMP HE8BE ;Z
28450 *>
28460
28470 HF02E ;solo
28480   LDX PX
28490   LDA PX+1
28500 HF032 ;Z
28510   LDY KBD ;get keypress
28520   CPY #ETX+$80 ;is it ctrl-C?
28530   BNE HF025 ;=>no
28540   BIT KBDSTRB ;yes, clear keypress
28550   STX NOUNSTKL
28560   STA NOUNSTKL+1
28570   MOVW PR;NOUNSTKH
28580   JMP STOPPED_AT
28590 *>
28600
28610   DB $FF,$FF ;Z
28620
28630 * tkn $10 HIMEM:
28640
28650 VHIMEM ;VO
28660   JSR GET16BIT
28670   STX XSAVE
28680   LDX #0-2
28690   SEC
28700
28710 *   MOVW ACC;P2
28720 *   SUBW HIMEM;ACC;AUX
28730
28740 *!LOOP
28750     LDA ACC+2,X
28760     STA P2+2,X
28770     LDA HIMEM+2,X
28780     SBC ACC+2,X
28790     STA AUX+2,X
28800     INX
28810 *!UNTIL <EQ>
28820   BCC HF0AF ;=>
28830   DEX ;Xreg := $FF
28840
28850 *   MOVW PP;P3
28860 *   SUBW PP;AUX;P2
28870
28880 *!LOOP
28890     LDA PP+1,X
28900     STA P3+1,X
28910     SBC AUX+1,X
28920     STA P2+1,X
28930     INX
28940 *!UNTIL <NE>
28950 *!IF <HS>
28960     CMPW PV;P2
28970     BCC HF08F ;=>PV <LO> P2
28980 *!ENDIF
28990 HF07C
29000   JMP MEMFULL
29010 *>
29020
29030 *!LOOP
29040     LDA (P3),Y
29050     STA (P2),Y
29060     INCW P2
29070     INCW P3
29080 HF08F ;solo
29090     CMPW P3;HIMEM
29100 *!UNTIL <HS>
29110 HF099 ;solo
29120   LDX #0-2
29130
29140 *   MOVW P2;HIMEM
29150 *   SUBW PP;AUX;PP
29160
29170 *!LOOP
29180     LDA P2+2,X
29190     STA HIMEM+2,X
29200     LDA PP+2,X
29210     SBC AUX+2,X
29220     STA PP+2,X
29230     INX
29240 *!UNTIL <EQ>
29250   LDX XSAVE
29260   RTS
29270 **
29280
29290 *!LOOP
29300     LDA (HIMEM),Y
```

```
    STA (ACC),Y
HF0AF ;solo
    DECW ACC
    DECW HIMEM
    CMP PP ;is PP <LO> HIMEM?
    LDA HIMEM+1
    SBC PP+1
*!UNTIL <HS>
  BCS HF099 ;=HS> always

* tkn $11 LOMEM:

VLOMEM ;VO
  JSR GET16BIT
  LDY ACC ;is ACC <HS> PP?
  CPY #PP
* BUG FIX: ABOVE LINE SHOULD BE
*   CPY PP
* REF: NONE.  FOUND BY INSPECTION.
  LDA ACC+1
  SBC PP+1
HF0D4
  BCS HF07C ;=HS> yes, MEM FULL error
  STY LOMEM ;LOMEM := ACC
  LDA ACC+1
  STA LOMEM+1
  JMP CLR
*>

* tkn $04 LOAD

LOAD ;VO
  STX XSAVE
  JSR SETHDR
  JSR READ
  LDX #$FF
  SEC
*!LOOP
    LDA HIMEM+1,X ;AUX := HIMEM-ACC
    SBC ACC+1,X
    STA AUX+1,X
    INX
*!UNTIL <NE>
  BCC HF07C ;=LO>MEM FULL error
  CMPW PV;AUX
  BCS HF0D4 ;=>PV <HS> AUX, MEM FULL error
  LDA ACC ;is ACC zero?
*!IF <EQ>
    LDA ACC+1
    BEQ HF118 ;=>yes
*!ENDIF
  MOVW AUX;PP
  JSR SETPRG
  JSR READ
HF115 ;solo
  LDX XSAVE
  RTS
**

HF118 ;solo
  JSR BELL
  JMP HF115
*>

SETHDR
  LDY #$CE
  STY A1 ;A1 := $00CE
  INY
  STY A2 ;A2 := $00CD
  LDY #0
  STY A1+1
  STY A2+1
  RTS
**

SETPRG
*!LOOP
    LDA PP,X
    STA A1,X
    LDY HIMEM,X
    STY A2,X
    DEX
*!UNTIL <MI>
  DECW A2
  RTS
**

  STX XSAVE ;Z

* tkn $05 SAVE

SAVE ;VO
  SEC ;ACC := HIMEM-PP
  LDX #0-1
*!LOOP
    LDA HIMEM+1,X
    SBC PP+1,X
    STA ACC+1,X
    INX
*!UNTIL <NE>
  JSR SETHDR
  JSR WRITE
  LDX #$01
  JSR SETPRG
  LDA #$1A
  JSR WRITE0
  LDX XSAVE
  RTS
**

PRTERR
  JSR ERRORMESS
  JMP BELL
*>

* tkn $77 POP

POP ;VO
  LDA GOSUBNDX
*!IF <EQ>
    JMP RETURN ;force error
*!ENDIF
  DEC GOSUBNDX
  RTS
**

* tkn $7D TRACE

TRACE ;VO
  LDA #$FF
  STA NOUNSTKC
  RTS
**

* tkn $7A NOTRACE

NOTRACE ;VO
  LSR NOUNSTKC ;clear bit 7
  RTS
**

HF179 ;solo
  BIT NOUNSTKC ;trace mode?
*!IF <MI>
HF17D
*yes, print line number
    LDA #"#"
    JSR COUT
    LDY #1
    LDA (PR),Y
    TAX
    INY
    LDA (PR),Y
    JSR PRDEC
    LDA #BLANK+$80
    JMP COUT
*>
    LDA PR ;Z
    LDY PR+1 ;Z
*!ENDIF
  RTS
**



SYNTABLNDX ;indices into SYNTABL
 DB $C1,$00,$7F,$D1,$CC,$C7,$CF,$CE
 DB $C5,$9A,$98,$8D,$96,$95,$93,$BF
 DB $B2,$32,$12,$0F,$BC,$B0,$AC,$BE
 DB $35,$0C,$61,$30,$10,$0B,$DD,$FB



HF1B7 ;solo
  LDY #0
  JSR HE7C7
  LDA #BLANK+$80
  JMP COUT
*>

  DB $00,$00,$00,$00,$00,$00,$00,$00 ;Z

HF1C9
  LDY LOMEM
  LDA LOMEM+1
*!LOOP
    PHA
    CPY AUX ;is LOMEM <HS> AUX?
    SBC AUX+1
    BCS HF1F0 ;=HS> yes, exit repeat
    PLA
    STY SRCH ;SRCH := LOMEM
    STA SRCH+1
    LDY #$FF
*!  LOOP
*!    LOOP
        INY
        LDA (SRCH),Y
*!    UNTIL <PL>
      CMP #$40
*!  UNTIL <NE>
    INY
    INY
    LDA (SRCH),Y
    PHA
    DEY
    LDA (SRCH),Y
    TAY
    PLA
*!UNTIL <EQ>
HF1F0
  PLA
  LDY #0
*!LOOP
    LDA (SRCH),Y
    BMI HF1FC ;=>
    LSR
    BEQ HF202 ;=>
    LDA #"$"
HF1FC
    JSR COUT
    INY
*!UNTIL <EQ>
HF202
  LDA #"="
  JMP COUT
*>

HF207 ;solo
  STA (AUX),Y
  INX
  LDA NOUNSTKC-1,X
  BEQ HF23E ;=>RTS
  JMP HF3D5
*>

  DB $A0 ;Z

HF212 ;solo
*!IF <PL>
    LDA PR
    LDY PR+1
    JSR HF17D
*!ENDIF
  JSR HF1C9
  LDX XSAVE
  JMP HF1B7
*>

HF223 ;solo
  INX
  INX
  LDA NOUNSTKC-1,X
  BEQ HF248 ;=>RTS
  JMP HF3E0
*>

HF22C ;solo
*!IF <PL>
    LDA PR
    LDY PR+1
    JSR HF17D
*!ENDIF
  JSR HF1C9
  LDX XSAVE
  JMP HF409
*>

  INX ;Z
HF23E
  RTS
**
```

```
32030
32040 HF23F ;solo
32050   JSR GET16BIT
32060   INCW ACC
32070 HF248
32080   RTS
32090 **
32100
32110 * tkn $1C <
32120 *   IF X < 13 THEN END
32130
32140 HF249 ;V
32150   JSR HF25B
32160   BNE HF263 ;=>NOT
32170
32180 * tkn $19 >
32190 *   IF X > 13 THEN END
32200
32210 HF24E ;VO
32220   JSR HF253
32230   BNE HF263 ;=>NOT
32240
32250 * tkn $1A <=
32260 *   IF X <= 13 THEN END
32270
32280 HF253 ;V
32290   JSR SUBTRACT
32300   JSR NEGATE
32310   BVC HF25E ;=>
32320
32330 * tkn $18 >=
32340 *   IF X >= 13 THEN END
32350
32360 HF25B ;V
32370   JSR SUBTRACT
32380 HF25E
32390   JSR SGN
32400   LSR NOUNSTKL,X
32410 HF263
32420   JMP NOT
32430 *>
32440
32450 * tkn $1D AND
32460
32470 VAND ;VO
32480   JSR HEFC9
32490   ORA NOUNSTKL-1,X
32500   BPL HF272 ;=>always?
32510
32520 * tkn $1E OR
32530
32540 VOR ;VO
32550   JSR HEFC9
32560   AND NOUNSTKL-1,X
32570 HF272 ;solo
32580   STA NOUNSTKL,X
32590   BPL HF263 ;=>NOT
32600   JMP HEFC9
32610 *>
32620
32630 * tkn $58 STEP
32640
32650 STEP ;VO
32660   JSR GET16BIT
32670   LDY FORNDX
32680   LDA ACC
32690   STA STK_60-1,Y
32700   LDA ACC+1
32710   JMP HE966
32720 *>
32730
32740 HF288 ;solo
32750   STA STK_50,Y
32760 *!LOOP
32770 *!  LOOP
32780       DEY
32790       BMI HF2DF ;=>RTS
32800       LDA STK_40,Y
32810       CMP NOUNSTKL,X
32820 *!  UNTIL <EQ>
32830     LDA STK_50,Y
32840     CMP NOUNSTKH,X
32850 *!UNTIL <EQ>
32860   DEC FORNDX
32870 *!LOOP
32880     LDA STK_40+1,Y
32890     STA STK_40,Y
32900     LDA STK_50+1,Y
32910     STA STK_50,Y
32920     LDA STK_C0+1,Y
32930     STA STK_C0,Y
32940     LDA STK_D0+1,Y
32950     STA STK_D0,Y
32960     LDA STK_60+1,Y
32970     STA STK_60,Y
32980     LDA STK_70+1,Y
32990     STA STK_70,Y
33000     LDA STK_80+1,Y
33010     STA STK_80,Y
33020     LDA STK_90+1,Y
33030     STA STK_90,Y
33040     LDA STK_A0+1,Y
33050     STA STK_A0,Y
33060     LDA STK_A0+1,Y
33070     STA STK_A0,Y
33080 * BUG FIX: ABOVE TWO LINES SHOULD BE
33090 *   LDA STK_B0+1,Y
33100 *   STA STK_B0,Y
33110 * REF: CHANGED IN DISK VERSION
33120     INY
33130     CPY FORNDX
33140 *!UNTIL <HS>
33150 HF2DF
33160   RTS
33170 **
33180
33190 * tkn $78 NODSP
33200 *   string var
33210
33220 NODSP_STR ;VO
33230   INX
33240
33250 * tkn $79 NODSP
33260 *   num var
33270
33280 NODSP_NUM ;VO
33290   LDA #0
33300 HF2E3
33310   PHA
33320   LDA NOUNSTKL,X
33330   SEC
33340   SBC #$03
33350   STA ACC
33360   LDA NOUNSTKH,X
33370   SBC #0
33380   STA ACC+1
33390   PLA
33400   LDY #0
33410   STA (ACC),Y
33420   INX
33430   RTS
33440 **
33450
33460 HF2F8 ;solo
33470   CMP #$85
33480 *!IF <LO>
33490     JMP HE4C0
33500 *!ENDIF
33510   LDY #$02
33520   JMP HE448
33530 *>
33540
33550 * tkn $7B DSP
33560 *   string var
33570
33580 DSP_NUM ;VO
33590   INX
33600
33610 * tkn $7C DSP
33620 *   num var
33630
33640 DSP_STR ;VO
33650   LDA #$01
33660   BNE HF2E3 ;=>always
33670
33680   INX ;Z
33690
33700 * tkn $06 CON
33710
33720 CON ;VO
33730   MOVW NOUNSTKH;PR
33740   LDA NOUNSTKL
33750   LDY NOUNSTKL+1
33760   JMP GETNEXT
33770 *>
33780
33790   LDA #$01 ;Z
33800   BNE HF2E3 ;=>always
33810
33820 * tkn $3C ASC(
33830
33840 ASC ;VO
33850   LDA NOUNSTKL,X
33860   CMP NOUNSTKH,X
33870 *!IF <HS>
33880     JMP RANGERR
33890 *!ENDIF
33900   TAY
33910   LDA NOUNSTKL+1,X
33920   STA ACC
33930   LDA NOUNSTKH+1,X
33940   STA ACC+1
33950   LDA (ACC),Y
33960   LDY #0
33970   INX
33980   INX
33990   JSR HE708
34000   JMP HF404
34010 *>
34020
34030 * tkn $32 PDL
34040
34050 PDL ;VO
34060   JSR GETBYTE
34070   STX XSAVE
34080   AND #$03
34090   TAX
34100   JSR PREAD
34110   LDX XSAVE
34120   TYA
34130   LDY #0
34140   JSR HE708
34150   STY NOUNSTKC,X
34160   RTS
34170 **
34180
34190 RDKEY ;solo
34200   JSR NXTCHAR
34210 HF354 ;solo
34220   TXA
34230   PHA
34240 *!LOOP
34250     LDA IN,X
34260     CMP #ETX+$80 ;is it ctrl-C?
34270 *!  IF <EQ>
34280       JMP BASIC2
34290 *!  ENDIF
34300     DEX
34310 *!UNTIL <MI>
34320   PLA
34330   TAX
34340   RTS
34350 **
34360
34370 HF366 ;solo
34380   JSR HE280
34390   TYA
34400   TAX
34410   JSR HF354
34420   TXA
34430   TAY
34440   RTS
34450 **
34460
34470 * tkn $20 ^
34480
34490 EXP ;VO
34500   JSR GET16BIT
34510   LDA ACC+1
34520 *!IF <MI>
34530     TYA ;Areg := 0
34540     DEX
34550     JSR HE708
34560     STY NOUNSTKC,X
34570 HF37F
34580     RTS
34590 *!ENDIF
34600   STA SRCH+1 ;SRCH := ACC
34610   LDA ACC
34620   STA SRCH
34630   JSR GET16BIT
34640   MOVW ACC;SRCH2
34650   LDA #$01
34660   JSR HE708
34670   STY NOUNSTKC,X
34680 HF398 ;*!LOOP
34690   LDA SRCH ;SRCH := SRCH-1
34700 *!IF <EQ>
34710     DEC SRCH+1 ;is SRCH negative?
34720     BMI HF37F ;=>yes, RTS
34730 *!ENDIF
34740   DEC SRCH
34750   LDA SRCH2
```

```
  LDY #0
  JSR HE708
  LDA SRCH2+1
  STA NOUNSTKC,X
  JSR MULT
  JMP HF398 ;*!loop forever
*>

HF3B3 ;solo
  JSR GETBYTE
  CLC ;Areg := Areg-1
  ADC #$FF
HF3B9
  RTS
**

* tkn $4A ,
*   end of PRINT statement
*   PRINT A$,

HF3BA ;VO
  JSR HE7B1
  LSR CRFLAG ;pos
  RTS
**

  STX RUNFLAG ;Z
  TXS
  JSR HF02E
  JMP HE883
*>

* tkn $7E PR#

PRSLOT ;VO
  JSR GETBYTE
  STX XSAVE
  JSR OUTPORT
  LDX XSAVE
  RTS
**

  DB $FE ;Z

HF3D5 ;solo
  BIT RUNFLAG
  BPL HF3B9 ;=>RTS
  STX XSAVE
  BIT NOUNSTKC
  JMP HF212
*>

HF3E0 ;solo
  BIT RUNFLAG
  BPL HF3B9 ;=>RTS
  STX XSAVE
  BIT NOUNSTKC
  JMP HF22C
*>

HF3EB ;solo
  LDY #0
  JMP GETVERB
*>

*!LOOP
    TAY
    JSR CROUT
HF3F4 ;solo
    TYA
    SEC
    SBC WNDWDTH
*!UNTIL <LO>
  STY CH
  RTS
**

  DB $00,$00,$00 ;Z
  DB $FF,$FF,$FF,$FF ;Z

HF404 ;solo
  STY NOUNSTKC,X
  JMP HE823
*>

HF409 ;solo
  LDY #0
  BEQ HF411 ;=>always
*!LOOP
    JSR COUT
    INY
HF411
    LDA (AUX),Y
*!UNTIL <PL>
  LDA #$FF
  STA CRFLAG ;CRFLAG := $FF
  RTS
**

* tkn $7F IN#

INSLOT ;VO
  JSR GETBYTE
  STX XSAVE
  JSR INPORT
  LDX XSAVE
  RTS
**
 LST OFF

*** END OF LISTING ***
```