



# EDITHA



## A REVISION OF THE FYLSTRA KIM-1 EDITOR PROGRAM

BY H. T. GORDON  
College of Natural Resources  
University of California  
Berkeley, CA 94720

Dear Jim, Tom, et al.,

Received: 78 April 3

I am herewith submitting an extensive recoding and enhancement of the editor section of Dan Fylstra's SWEETS program for the KIM-1, published in (and copyrighted by) BYTE. It poses several dilemmas of general interest. First, the problem of program names. SWEETS is Fylstra's acronym for 2 main programs, that share some subroutines. The entry label for the editor (the only one I have worked on) is CMD. I have given the more descriptive name EDITHA to my revision, on the grounds that its coding is mostly quite different and that it includes some major innovations (address information and modification capability, greater error-protection, and a richer command structure with modes not found in CMD). It is clearly an enhancement of the kind that Fylstra himself anticipated would grow out of CMD. I hesitated before adopting the name EDITHA, since renaming can easily become an idiotic game, involving personal vanity, causing confusion, and even trying to disguise software theft. I have restrained many of Fylstra's names in EDITHA whenever my recoding did not involve a major functional alteration, both as a courtesy to the originator of the basic concept and as an aid to comprehension by users. One of his subroutines (DETLN1) is an inefficient way of determining the bytecount of a 6502 opcode. In EDITHA I have replaced DETLN1 logic by the decoding logic of my subroutine BYTCNT (published in DDJ # 22), with minor retailoring to fit the special context. I re-use the name BYTCNT for this trivial modification, because I think it is better to extend the meaning of a subroutine name to include minor variations, instead of endlessly coining new names. There are, alas, no rules defining how great a dissimilarity is needed to justify a new name.

The second dilemma is the perennial one of software ownership. I assert copyright (qualified by a special non-monopolistic free-diffusion clause) for all components of EDITHA not either previously in the public domain or copyrighted by someone else. Exactly what these are is a moot question. Very little of the original CMD coding is left intact in EDITHA. None of the alterations is anything but functional (i.e., there is no intent to evade existing copyright by mere recoding). None is copied from any work I have had access to (although since there are relatively few ways of doing something efficiently, it is not impossible that some coding is an independent recreation of already existing logic). Here is the qualifying clause: The intent of my copyright is to remove all impediments to free diffusion, including (to whatever extent my copyright gives me leverage to do so) impediments that others may wish to impose on enhanced or different imple-

mentations, including systems other than the KIM-1. I therefore authorize any and all uses (including copying in any form, publication in whole or in part, and commercialization) if (and only if) all software associated with the EDITHA coding is declared to be equally available for totally unrestricted use. What I am trying to ensure is not only that my copyright will be no bar to progress but that it shall serve as an aid to it. Perhaps this is Utopian, but there's no harm in trying! It would be a tragicomedy if micro software had to be diffused underground, in the style of Russian samizdats, as a response to the censorship aspects of copyrighting.

The third dilemma is: where should revision of a program published in BYTE be published? First, let me say that I consider BYTE a valuable journal. I have had a note (on 6502 opcodes) published in it, and a second one (on a pseudorandom number generator, 8080-coded) is accepted and will (eventually!) get published. There are some advantages (a small remuneration, glossy paper, and a very large readership) and some disadvantages (among them are: loss of all rights to your work, and often long publication delay, and no indication of the date of receipt of articles). On the whole, my relationship with DDJ has been a more humanly satisfying one. Friends have told me its title is zany (but to me it's whimsical—I reserve zany [and its synonyms] for a world with tens of thousands of megatons of hydrogen bombs poised for launching). But publication is prompt, always includes date of receipt (a fine ethical touch), and the author retains the rights to his work. These small-is-beautiful qualities compensate for inexpensive paper and no remuneration!

Sincerely,  
H. T. Gordon

The SWEETS machine-language editor program CMD, created by Dan Fylstra (Feb. 1978 BYTE) is a major enhancement of the KIM-1 SBC. However, it has some coding inefficiencies, logic errors, and other limitations. The major ones are its inability to allow manipulation of addresses (only program instructions can be seen in the KIM display, and the lack of adequate safeguards against user errors. It is all too easy to ignore the CMD error-warning and cause disastrous upmoves or downmoves of immense blocks of RAM, including the CMD program itself. My revision gradually became a major overhaul, involving drastic alteration of both the main program (whose entry point I have renamed EDITHA) and the subroutines.

Insofar as possible, EDITHA (version 1.0, intended for the basic KIM-1) is located in the same memory areas used by CMD. This conforms with Fylstra's intent of leaving a large block of RAM free for new program construction under editor control. However, the enhancements provided by EDITHA increase its length to 478 bytes (71 more than CMD), and its subroutine READIN had to be located in 0200-0234. This leaves only the 459 locations from 0235-03FF free.

The revision relocates the original CMD pointers in zero-page: MOVAD (to DE-DF), BEGAD (to E0-E1), CURAD (to E2-E3), ENDA (to E4-E5), and adds a new one, LIMAD (at E6-E7). This allows operation of a new guard subroutine, INLMT, that protects memory locations outside preset program limits from being moved or overwritten. BEGAD is redefined as one location *below* the program start (lowest) address. LIMAD must be set 2 locations below the highest address that the program can enter. BEGAD and all lower memory locations, and LIMAD+3 and all higher memory locations, cannot be altered or moved by EDITHA. The guard subroutine blocks the action of the DA (delete instruction) key unless BEGAD < CURAD < ENDA. In addition, it blocks the AD (add instruction) key unless ENDA < LIMAD. When these conditions are not met, the only effect of pressing either key is to display CURAD in the usual KIM address display, and EE in the data display. CURAD is the address of the instruction currently displayed, that EDITHA is refusing to add to or delete. The same message will be displayed if any hex key is pressed when EDITHA is in the command-entry mode, but in this case the EE does not signify that CURAD is out-of-bounds but that hex keys cannot be interpreted and are being rejected.

Although EDITHA has instructions that allow the user to alter BEGAD or LIMAD, they are normally fixed and define the program limits. As in CMD, CURAD is advanced to the next higher instruction when the + key is pressed. However, while the + key is depressed, EDITHA displays the address of the next instruction (that appears when the key is released). If this address is within limits, 03 appears in the data display, but if it is out-of-bounds, EE is displayed.

During program writing, CURAD does not change, but ENDA moves upward when AD is used to insert an instruction and downward when DA is used to delete the instruction currently in CURAD. If deletions exceed insertions, ENDA may move below CURAD, and thereby block the action of the AD and DA keys. This is no problem, since the user can press a hex key to see where CURAD is and then use the pointer-resetting instruction to set ENDA to a higher location. In fact, it is sometimes useful to set ENDA to the program start address, so that the user can explore anywhere in memory with no risk of altering programs by accidental pressing of the AD or DA keys.

The operational changes in the + and DA key actions are achieved by quite trivial modification of the original (much less informative) CMD error-message, together with continual supervision by the guard subroutine. Users may notice one more trivial change: one-byte displays by EDITHA are of normal brightness, while those by CMD are extremely bright. CMD refreshes one-byte displays 3 times as often as 3-byte ones; this may or may not shorten the life of the LEDs, but I found it unpleasant. Therefore EDITHA inserts a very brief dark period in every refresh of a one-byte display. Both CMD and EDITHA display two-bytes at about 1.5 times normal brightness.

#### Implementation of Complex Commands in the AD and GO Operations.

Unlike the + and DA operations, which are simple commands executed by pressing one command key, the AD and GO operations require complex commands. Both call subroutine READIN, since the user is expected to key in more information. Although much more complex than its equivalent in CMD (subroutine RDBYTE), READIN also calls subroutine SCAN1 twice. Unlike RDBYTE (that always error-exits if a command key is pressed), the first call to SCAN1 by READIN will accept either a hex or a command key, the lat-

ter causing a return for interpretation by the calling routine. If the first key is hex, the second call to SCAN1 *ignores* all command keys but returns when a hex key is pressed, inserting the completed byte in the leftmost display (all others blank). ADKEY logic interprets this as an opcode; if it is a one-byte opcode, it is inserted in the program and the operation is over. If it is a two- or three-byte opcode, READIN is again called so the user can complete the instruction. At this point, it is possible to press a command key and cause a return. In version 1.0, EDITHA interprets such command-hex-command instructions as errors, simply wiping them out as if they had never been keyed-in at all. In an expanded version, however, more complex interpreting logic could open a cornucopia (or a Pandora's box?) of special commands. Very few of these possibilities are realized in version 1.0. One reason is the memory limitations of the basic KIM-1. Another is that I am not sure how powerful a machine-language editor ought to become, or what operations will prove to be the most useful.

#### The AD Key Operations

Pressing the AD key has no detectable effect if CURAD is within limits and the instruction currently in the display is one-byte. If it is two- or three-byte, it is left-shifted so that only the previously rightmost byte appears in the leftmost display (all others blank). When the opcode is keyed-in, it replaces the displayed byte. If it is a one-byte opcode the instruction is complete and written into the program (and being displayed by the program-scanning subroutines). Pressing a hex key will cause the usual error-display of CURAD. If the opcode is incorrect, it can be deleted by pressing the DA key.

If the opcode requires 1 or 2 operand bytes, pressing any command key will cause an error wipe-out (a quick way to erase a wrong opcode). If a hex key is pressed, nothing will happen until a second hex key is pressed. If the opcode was two-byte, the second keyed-in byte will appear in the *middle* display (normal order, an indication that the whole instruction has been written into the program). It can be deleted by the DA key. However, keying in the first operand of a three-byte opcode will cause the opcode to shift to the middle display and the operand to appear in the leftmost one (*reverse* order, instruction not yet completed). Again, pressing a command key will cause a wipeout. Keying in the third byte (second operand) will cause display in normal order (instruction written-in, deletable by DA).

A peculiar "double-command" operation is also possible, executed if the AD key is pressed *twice* (pressing AD, then any other command key, has the same effect but the AD-AD is safer). The effect is to insert the single-byte in the display (following the first AD) in the program, "completing" the instruction. The AD-AD operation is useful *only* if the instruction in the display before the first AD was a one-byte opcode. It allows *duplication* of this opcode, a convenient way to enter a string of one-byte opcodes (00, 0A, 4A, EA, etc.). The AD-AD was not a planned feature of EDITHA, but an accidental consequence of the planned GO-AD (to be discussed in the next section), retained because its implementation requires very little interpreting logic.

#### The GO Key Operations

These represent by far the most complex EDITHA enhancements, that still are only a minute fraction of a legion of possibilities. The only "double-command" is the GO-AD (the other 3 possible ones are interpreted as errors). This sequence of 2 command keyings resets CURAD to the program start (one location higher than BEGAD). (Note that, although the command keys are read by the KIM monitor subroutine

GETKEY as hex numbers from \$10 to \$13, the READIN decoder subtracts 4 from these values and returns command codes from \$0C to \$0F to the GOKEY routine.)

All other GO operations require keying-in 4 hex bytes for completion, the first 3 appearing in the display one after another, in reverse order. Wipeout can be caused by pressing any command key following entry of any byte into the display. The terminator byte (not displayed) is the command code that will cause execution of a special operation. Since all non-interpreted command codes cause an error-message, it is useful to keep the final key of this code depressed so that it can be seen displayed. If there is no error-message, the command has been executed. Error-messages are always XX 04 EE, where XX is the command code that could not be executed (except for codes 04 and 08, where XX is 00). The following is a list of the 5 valid command modes, with their codes:

#### Mode I, code 00

This is the ERASE command, that restores the original status before the GO key was pressed. Although the "wipeout" feature of command keys (a later development) makes it unnecessary, it might be useful in later versions.

#### Mode II, codes 01 to 03

These are IDPR (insert data into program) commands, 01 inserting only the first keyed-in byte (rightmost in the display), 02 the first and second, 03 all three. Before completion, display is in *reverse* order. When the command is completed, the inserted bytes will be displayed in *normal* order (as with the AD operation) but interpreted as data. Thus the sequence of keyings: GO-08-18-28-03, that is displayed as 28 18 08 before the command code is entered, will be displayed as 08 18 28 afterward. This sequence would need 3 separate entries in the AD operation, since all the entries are one-byte opcodes. Also, the *entire* sequence can be deleted by pressing the DA key once (if there has been an error).

In both CMD and EDITHA, to add a new program or data sequence *following* the one in the display, one must first press the + key to advance CURAD. This is necessary to allow the user to see in the display exactly what he has entered. In the EDITHA IDPR Mode II, all bytes that will be entered (and with codes 02 and 01, some that will *not* be) are visible in the display before the command is keyed-in, although in reverse order. Some users may wish to implement an automatic advance, so that more data can be entered by the Mode II GO without having to press the + key. This can be done by replacing the BEQ DACMD at 017B by a BEQ STPKEY (F0 37). The disadvantage is that one will not see displayed what has been entered, and not have a second chance to delete an erroneous entry.

#### Mode III, codes 05 to 07 and 09 to 0B

These are program SEARCH commands, with some kinship to the GO operation in CMD. There is unnecessary redundancy here (05 = 09, 06 = 0A, 07 = 0B) that saves a little GOKEY interpreting logic, but could be eliminated in expanded versions if codes 09 to 0B can be given another useful interpretation. One major difference from the search GO in CMD is that searching is done in the range from the current CURAD (the address of the instruction that was in the display before the GO) upward in memory to ENDAD. To search the *entire* program, one must first reset CURAD to the program start by a GO-AD. The reason for this modification is that the same instruction may recur several times in a program, and the GO logic in CMD can only find the *first* occurrence. When using the EDITHA search GO, the user should keep the final command key depressed. If a match is found, the display will show the address of this instruction that will be displayed when the key is released (while the data display will show the number of

bytes that were searched for). If this is not the one wanted, press the + key and redo the search GO to find the *next* identical instruction, higher in the program. If there is no match, the operation will display the ENDAD address, with EE in the data display.

The second major difference from CMD is that EDITHA can search for only the opcode (code 05), for the opcode and one following byte (code 06, identical with the CMD search GO), or for the opcode and *two* following bytes (code 07). E.g., the key sequence GO-85-FB-60-07 requires that an 85 FB 60 sequence exist in the program. This happens to be the terminal program sequence in the KIM monitor ROM. An interesting test of searching speed is possible if one sets BEGAD to 03FF, CURAD to 1FD4, and ENDAD and LIMAD both to 1FD5. When EDITHA is entered (set address to 019A and press GO) it will display the RTS (\$60 at 1FD4. A GO-AD sends it to 0400, and 0404 is displayed (although of course there is nothing there in the basic KIM). The GO-85-FB-60-70 keying sequence will darken the display for less than a second, then (with the final key still depressed) display 1F D2 03. This is the address of the last 85 FB instruction in the ROM, that appears when the 7 key is released. Nearly 7K of (mostly non-existent!) memory has been subjected to a complex search in a reasonably short time. If one uses the 06 command code to search for only 85 FB, one can find 5 recurrences of it lower in the KIM ROM than the terminal one.

#### Mode IV, code 0C

This works exactly like the GO-AD double-command. It was the awkwardness of resetting CURAD to the program start in this way (or in another way possible with Mode V) that led me to develop the GO-AD. In the original CMD design (largely retained in this pioneer version of EDITHA), keyings are recognized in 2 different ways. CMD calls the program-scanning routines that maintain display of the instruction at CURAD and detect a depressed key, which causes a return for interpretation by CMD logic. Hex keys are rejected (an EE EE EE error-message being displayed while they are depressed), while each of the 4 command keys causes control to shift to a special routine. With more elaborate logic, one could interpret double-commands here (or even, if one were ambitious, triple-commands). It would be a cleaner way than that in version 1.0, which re-uses the scaffolding of a primitive approach to implement a more sophisticated one (just to avoid one more reconstruction job).

The second kind of key-recognition in CMD is done by the GOKEY and ADKEY routines, via subroutine RDBYTE, which error-exits if a command key is pressed but accepts keyed-in hex bytes. My decision to implement double commands here was based on the fact that I had already included a lot of command-code interpreting logic and it was very simple just to modify READIN slightly. When I started out to revise CMD, I did not have a clear idea of how it worked and never dreamed that it would go so far! The structural flaws in EDITHA (1.0) are the result of its just growing like Topsy, with its growth halted when it started getting too big for the basic KIM-1.

#### Mode V, even-numbered codes from \$10 to \$8E

These are IDZP (insert data into zero-page) commands (64 in all), that write the first 2 keyed-in data bytes into contiguous zero-page locations specified by the command code. E.g., the keying sequence GO-12-34-10-10 will cause \$34 to be written in location 0010 and \$12 in location 0011. Before the keying-in of the command code 10, the first 3 bytes are displayed as 10 34 12 (reverse order); this is wiped out by the command code, since nothing is entered into the program. Since the effect is not visible and one has to be extra careful with zero-page, the third keyed-in (and so displayed) byte

Because important subroutines are located in zero page (0080 to 00D4), command codes 80 to 8E skip these locations and write into locations 00D8 to 00E7. E.g., the sequence GO-1C-00-8A-8A resets CURAD to address 1C00 in the KIM ROM, causing display of the instruction there. This program can then be scanned by the + key, since EDITHA does not object to its user going anywhere in memory just to *look* at it. Before wandering around in memory, ENDAD should be reset to the program-start address (easily checked by a GO-AD), using the 8C (reset ENDAD) command code, to "disconnect" the AD and DA keys. Codes 8E (reset LIMAD), 88 (reset BEGAD), and 86 (reset MOVAD) will be less often needed, and the 3 lower pointers (80, 82, 84) are available for future enhancements.

Since EDITHA interacts with molasses-slow human operators, it is not designed for speed but emphasizes coding efficiency, error-protection, and convenience. One major sub-routine (ASCNIT, 0100 to 011E) is *not* listed because its coding is basically identical with that of the Fylstra-BYTE copyrighted CMD (located in the original at 0103 to 0121) except that the downward relocation in EDITHA requires that the 3JSRs to SCAN3 be coded as 20 1F 01 instead of the original 20 22 01. The relocation is functional, since the JSR DETLEN that is the first instruction of the CMD sub-routine SCAN is now the second instruction in EDITHA (at 019B), so that the third one (JSR SCANIT, labeled DACMD) can be branched to to cause a display *not* interpreted as an instruction. My omission of SCANIT may strike some readers as ridiculous, but the intent is to avert the wrath of whatev-

Listings therefore start at 011F (SCAN3), that is significantly modified in order to reduce the brightness of single-byte displays (the JSR ONE just kills a little time). To be punctilious, I note that the terminal coding of the EDITHA subroutine MVDOWN (00C9-00D4) is identical with that of the CMD subroutine ADVAND. It seems to me that the logic of adding an 8-bit number to a 16-bit number is so commonly used that it is in the public domain.

### Comment on Enhancements

As I have indicated, EDITHA has many possibilities for restructurings and new operations, and everyone is free to do whatever he or she likes. It is quite possible that someone has already written something as good as or better than EDITHA. More likely, every further revision will have some fine ideas of its own that ought to be in the ultimate version(s). Perhaps *DDJ* can serve as a clearinghouse for pooling these ideas, and thus hasten development of a super-version (that will merit a super-name!).

[illegible]

```

0174 85 E8 STA BYTES (store for ops)
0176 20 DF 17 JSR LIMITS (OK to write?)
0179 90 3C BCC ADVAL (no, exit)
017B 20 E0 01 JSR WRITIN (write data)
017E F0 1E BEQ DACMD (display data)
0180 EA NOP (just a leftover!)

0181 09 8F HICMD CMP #8F (command too hi?)
0183 B0 38 BCS ERROR (yes, exit)
0185 C5 FB CMP POINTH (confirmed?)
0187 D0 34 BNE ERROR (no, exit)
0189 1A LSR A (even or odd #?)
018A B0 31 BCS ERROR (odd, exit)
018C 0A ASL A (restore command)
018D 10 02 BPL NOADD (<$80, use as is)
018F 69 58 ADC #58 (raise to $D8-up)
0191 4A NOADD TAX (move command to X index)
0192 A5 F9 LDA INH (pickup first byte)
0194 95 01 STA HIBASE+1,X (IDZP write)
0196 A5 FA LDA POINTL (pickup second)
0198 95 00 STA HIBASE,X (IDZP write)

019A D8 EDITHA CLD (clear decimal mode)
019B 20 80 00 JSR DETLEN (get opcode bytes)
019E 20 00 01 DACMD JSR SCANIT

01A1 09 12 GCMD CMP #12
01A3 F0 0F BEQ STPKEY (+ key = $12)
01A5 B0 A7 BCS COKEY (GO key = $13)
01A7 C9 10 CMP #10
01A9 90 0C BCC ADVAL (any hex key)
01AB F0 1F BEQ ADKEY (AD key = $10)

01AD 20 B2 17 JSR MOVEUP (DAKEY deletion)
01B0 B0 E8 BCS EDITHA (OK, done)
01B2 90 03 BCC ADVAL (can't be done)

01B4 20 8D 17 STPKEY JSR ADVANC (to next opcode)
01B7 A6 E2 ADVAL LDA CURAD (to be displayed)
01B9 A5 E3 LDA CURAD+1 ( " " )
01BB B0 02 BCS INFORM (not an error)

01BD A0 EE ERROR LDY #EE (display EE)
01BF 80 F9 INFORM STY INH (display Y in data)
01C1 86 FA STX POINTL (address low)
01C3 85 FB STA POINTH (address high)
01C5 20 1F 1F ERR1 JSR SCANDS (display message)
01C8 D0 FB BNE ERR1 (while key is held)
01CA F0 CE BEQ EDITHA (back to editor)
01CC 20 DF 17 ADKEY JSR LIMITS (write OK?)
01CF 90 E6 BCC ADVAL (no, exit)
01D1 86 EA STX COUNT (sets to 00)

01D3 20 00 02 JSR READIN (set key input)
01D6 F0 03 BEQ ACCEPT (all-hex, OK)
01D8 CA DEX (if AD-AD, X was 00)
01D9 10 E2 BPL ERROR (not AD-AD, exit)
01DB 20 E0 01 ACCEPT JSR WRITIN (write input)
01DE F0 BA BEQ EDITHA (back to editor)

01E0 20 A4 00 WRITIN JSR MVDOWN (open up space)
01E3 A6 E8 LDY BYTES (# bytes to write)
01E5 A0 00 LDY #0
01E7 B9 F9 00 STORIT LDA INH,Y (pickup byte)
01EA 91 E2 STA CURAD,Y (write it in)
01EC C8 INY (for next pickup)
01ED CA DEX (if = 0, all written)
01EE D0 F7 BNE STORIT (≠ 0, next byte)
01F0 60 RTS (return to caller)

(coding for subroutine READIN)

0200 A2 01 READIN LDY #1 (set X to 1)
0202 86 E8 STX BYTES (display 1 byte)
0204 CA DEX (X now = 00)
0205 86 EB NEXKEY STX XINDEX (saves X)
0207 20 0C 01 READHI JSR SCAN1 (read first key)
020A C9 10 CMP #10 (a hex key?)
020C 90 05 BCC SHIPT (yes, use it)
020E A6 EB LDY XINDEX (no, restore X)
0210 E9 04 SBC #4 (subtract, Acc - 4)
0212 60 RTS (return to caller)
0213 0A SHIFT ASL A (hex into hi nybble)
0214 0A ASL A
0215 0A ASL A
0216 0A ASL A
0217 85 E9 STA RDTMP (save hi nybble)
0219 20 0C JSR SCAN1 (read second key)
021C C9 10 CMP #10 (a hex key?)
021E B0 F9 BCS READLO (nc, try again)
0220 05 E9 ORA RDTMP (add hi nybble)
0222 A6 EB LDY XINDEX (restore X)
0224 95 F9 STA INH,X (byte into display)
0226 E8 INX (increment X for next op)
0227 A4 EA LDY COUNT (=0 only first AD)
0229 D0 03 BNE OPSKIP (not an opcode)
022B 20 84 JSR BYCNT (get bytecount)
022E 86 E8 STX BYTES (BYTES must = X)
0230 E4 EA CPX COUNT (is X = COUNT?)
0232 D0 D1 BNE NEXKEY (no, continue)
0234 60 RTS (return to caller)
0236 00 BEG-IN LDT BEGAD+1
0238 00 LDY BEGAD
0239 00 INX (true program start)
023A 00 STX CURAD (resets low)
023B 00 BNE NOINY (high is OK)
023C 00
023D 00
023E 00
023F 00
0240 00
0241 00
0242 00
0243 00
0244 00
0245 00
0246 00
0247 00
0248 00
0249 00
024A 00
024B 00
024C 00
024D 00
024E 00
024F 00
0250 00
0251 00
0252 00
0253 00
0254 00
0255 00
0256 00
0257 00
0258 00
0259 00
025A 00
025B 00
025C 00
025D 00
025E 00
025F 00
0260 00
0261 00
0262 00
0263 00
0264 00
0265 00
0266 00
0267 00
0268 00
0269 00
026A 00
026B 00
026C 00
026D 00
026E 00
026F 00
0270 00
0271 00
0272 00
0273 00
0274 00
0275 00
0276 00
0277 00
0278 00
0279 00
027A 00
027B 00
027C 00
027D 00
027E 00
027F 00
0280 00
0281 00
0282 00
0283 00
0284 00
0285 00
0286 00
0287 00
0288 00
0289 00
028A 00
028B 00
028C 00
028D 00
028E 00
028F 00
0290 00
0291 00
0292 00
0293 00
0294 00
0295 00
0296 00
0297 00
0298 00
0299 00
029A 00
029B 00
029C 00
029D 00
029E 00
029F 00
02A0 00
02A1 00
02A2 00
02A3 00
02A4 00
02A5 00
02A6 00
02A7 00
02A8 00
02A9 00
02AA 00
02AB 00
02AC 00
02AD 00
02AE 00
02AF 00
02B0 00
02B1 00
02B2 00
02B3 00
02B4 00
02B5 00
02B6 00
02B7 00
02B8 00
02B9 00
02BA 00
02BB 00
02BC 00
02BD 00
02BE 00
02BF 00
02C0 00
02C1 00
02C2 00
02C3 00
02C4 00
02C5 00
02C6 00
02C7 00
02C8 00
02C9 00
02CA 00
02CB 00
02CC 00
02CD 00
02CE 00
02CF 00
02D0 00
02D1 00
02D2 00
02D3 00
02D4 00
02D5 00
02D6 00
02D7 00
02D8 00
02D9 00
02DA 00
02DB 00
02DC 00
02DD 00
02DE 00
02DF 00
02E0 00
02E1 00
02E2 00
02E3 00
02E4 00
02E5 00
02E6 00
02E7 00
02E8 00
02E9 00
02EA 00
02EB 00
02EC 00
02ED 00
02EE 00
02EF 00
02F0 00
02F1 00
02F2 00
02F3 00
02F4 00
02F5 00
02F6 00
02F7 00
02F8 00
02F9 00
02FA 00
02FB 00
02FC 00
02FD 00
02FE 00
02FF 00
0300 00
0301 00
0302 00
0303 00
0304 00
0305 00
0306 00
0307 00
0308 00
0309 00
030A 00
030B 00
030C 00
030D 00
030E 00
030F 00
0310 00
0311 00
0312 00
0313 00
0314 00
0315 00
0316 00
0317 00
0318 00
0319 00
031A 00
031B 00
031C 00
031D 00
031E 00
031F 00
0320 00
0321 00
0322 00
0323 00
0324 00
0325 00
0326 00
0327 00
0328 00
0329 00
032A 00
032B 00
032C 00
032D 00
032E 00
032F 00
0330 00
0331 00
0332 00
0333 00
0334 00
0335 00
0336 00
0337 00
0338 00
0339 00
033A 00
033B 00
033C 00
033D 00
033E 00
033F 00
0340 00
0341 00
0342 00
0343 00
0344 00
0345 00
0346 00
0347 00
0348 00
0349 00
034A 00
034B 00
034C 00
034D 00
034E 00
034F 00
0350 00
0351 00
0352 00
0353 00
0354 00
0355 00
0356 00
0357 00
0358 00
0359 00
035A 00
035B 00
035C 00
035D 00
035E 00
035F 00
0360 00
0361 00
0362 00
0363 00
0364 00
0365 00
0366 00
0367 00
0368 00
0369 00
036A 00
036B 00
036C 00
036D 00
036E 00
036F 00
0370 00
0371 00
0372 00
0373 00
0374 00
0375 00
0376 00
0377 00
0378 00
0379 00
037A 00
037B 00
037C 00
037D 00
037E 00
037F 00
0380 00
0381 00
0382 00
0383 00
0384 00
0385 00
0386 00
0387 00
0388 00
0389 00
038A 00
038B 00
038C 00
038D 00
038E 00
038F 00
0390 00
0391 00
0392 00
0393 00
0394 00
0395 00
0396 00
0397 00
0398 00
0399 00
039A 00
039B 00
039C 00
039D 00
039E 00
039F 00
03A0 00
03A1 00
03A2 00
03A3 00
03A4 00
03A5 00
03A6 00
03A7 00
03A8 00
03A9 00
03AA 00
03AB 00
03AC 00
03AD 00
03AE 00
03AF 00
03B0 00
03B1 00
03B2 00
03B3 00
03B4 00
03B5 00
03B6 00
03B7 00
03B8 00
03B9 00
03BA 00
03BB 00
03BC 00
03BD 00
03BE 00
03BF 00
03C0 00
03C1 00
03C2 00
03C3 00
03C4 00
03C5 00
03C6 00
03C7 00
03C8 00
03C9 00
03CA 00
03CB 00
03CC 00
03CD 00
03CE 00
03CF 00
03D0 00
03D1 00
03D2 00
03D3 00
03D4 00
03D5 00
03D6 00
03D7 00
03D8 00
03D9 00
03DA 00
03DB 00
03DC 00
03DD 00
03DE 00
03DF 00
03E0 00
03E1 00
03E2 00
03E3 00
03E4 00
03E5 00
03E6 00
03E7 00
03E8 00
03E9 00
03EA 00
03EB 00
03EC 00
03ED 00
03EE 00
03EF 00
03F0 00
03F1 00
03F2 00
03F3 00
03F4 00
03F5 00
03F6 00
03F7 00
03F8 00
03F9 00
03FA 00
03FB 00
03FC 00
03FD 00
03FE 00
03FF 00
0400 00
0401 00
0402 00
0403 00
0404 00
0405 00
0406 00
0407 00
0408 00
0409 00
040A 00
040B 00
040C 00
040D 00
040E 00
040F 00
0410 00
0411 00
0412 00
0413 00
0414 00
0415 00
0416 00
0417 00
0418 00
0419 00
041A 00
041B 00
041C 00
041D 00
041E 00
041F 00
0420 00
0421 00
0422 00
0423 00
0424 00
0425 00
0426 00
0427 00
0428 00
0429 00
042A 00
042B 00
042C 00
042D 00
042E 00
042F 00
0430 00
0431 00
0432 00
0433 00
0434 00
0435 00
0436 00
0437 00
0438 00
0439 00
043A 00
043B 00
043C 00
043D 00
043E 00
043F 00
0440 00
0441 00
0442 00
0443 00
0444 00
0445 00
0446 00
0447 00
0448 00
0449 00
044A 00
044B 00
044C 00
044D 00
044E 00
044F 00
0450 00
0451 00
0452 00
0453 00
0454 00
0455 00
0456 00
0457 00
0458 00
0459 00
045A 00
045B 00
045C 00
045D 00
045E 00
045F 00
0460 00
0461 00
0462 00
0463 00
0464 00
0465 00
0466 00
0467 00
0468 00
0469 00
046A 00
046B 00
046C 00
046D 00
046E 00
046F 00
0470 00
0471 00
0472 00
0473 00
0474 00
0475 00
0476 00
0477 00
0478 00
0479 00
047A 00
047B 00
047C 00
047D 00
047E 00
047F 00
0480 00
0481 00
0482 00
0483 00
0484 00
0485 00
0486 00
0487 00
0488 00
0489 00
048A 00
048B 00
048C 00
048D 00
048E 00
048F 00
0490 00
0491 00
0492 00
0493 00
0494 00
0495 00
0496 00
0497 00
0498 00
0499 00
049A 00
049B 00
049C 00
049D 00
049E 00
049F 00
04A0 00
04A1 00
04A2 00
04A3 00
04A4 00
04A5 00
04A6 00
04A7 00
04A8 00
04A9 00
04AA 00
04AB 00
04AC 00
04AD 00
04AE 00
04AF 00
04B0 00
04B1 00
04B2 00
04B3 00
04B4 00
04B5 00
04B6 00
04B7 00
04B8 00
04B9 00
04BA 00
04BB 00
04BC 00
04BD 00
04BE 00
04BF 00
04C0 00
04C1 00
04C2 00
04C3 00
04C4 00
04C5 00
04C6 00
04C7 00
04C8 00
04C9 00
04CA 00
04CB 00
04CC 00
04CD 00
04CE 00
04CF 00
04D0 00
04D1 00
04D2 00
04D3 00
04D4 00
04D5 00
04D6 00
04D7 00
04D8 00
04D9 00
04DA 00
04DB 00
04DC 00
04DD 00
04DE 00
04DF 00
04E0 00
04E1 00
04E2 00
04E3 00
04E4 00
04E5 00
04E6 00
04E7 00
04E8 00
04E9 00
04EA 00
04EB 00
04EC 00
04ED 00
04EE 00
04EF 00
04F0 00
04F1 00
04F2 00
04F3 00
04F4 00
04F5 00
04F6 00
04F7 00
04F8 00
04F9 00
04FA 00
04FB 00
04FC 00
04FD 00
04FE 00
04FF 00
0500 00
0501 00
0502 00
0503 00
0504 00
0505 00
0506 00
0507 00
0508 00
0509 00
050A 00
050B 00
050C 00
050D 00
050E 00
050F 00
0510 00
0511 00
0512 00
0513 00
0514 00
0515 00
0516 00
0517 00
0518 00
0519 00
051A 00
051B 00
051C 00
051D 00
051E 00
051F 00
0520 00
0521 00
0522 00
0523 00
0524 00
0525 00
0526 00
0527 00
0528 00
0529 00
052A 00
052B 00
052C 00
052D 00
052E 00
052F 00
0530 00
0531 00
0532 00
0533 00
0534 00
0535 00
0536 00
0537 00
0538 00
0539 00
053A 00
053B 00
053C 00
053D 00
053E 00
053F 00
0540 00
0541 00
0542 00
0543 00
0544 00
0545 00
0546 00
0547 00
0548 00
0549 00
054A 00
054B 00
054C 00
054D 00
054E 00
054F 00
0550 00
0551 00
0552 00
0553 00
0554 00
0555 00
0556 00
0557 00
0558 00
0559 00
055A 00
055B 00
055C 00
055D 00
055E 00
055F 00
0560 00
0561 00
0562 00
0563 00
0564 00
0565 00
0566 00
0567 00
0568 00
0569 00
056A 00
056B 00
056C 00
056D 00
056E 00
056F 00
0570 00
0571 00
0572 00
0573 00
0574 00
0575 00
0576 00
0577 00
0578 00
0579 00
057A 00
057B 00
057C 00
057D 00
057E 00
057F 00
0580 00
0581 00
0582 00
0583 00
0584 00
0585 00
0586 00
0587 00
0588 00
0589 00
058A 00
058B 00
058C 00
058D 00
058E 00
058F 00
0590 00
0591 00
0592 00
0593 00
0594 00
0595 00
0596 00
0597 00
0598 00
0599 00
059A 00
059B 00
059C 00
059D 00
059E 00
059F 00
05A0 00
05A1 00
05A2 00
05A3 00
05A4 00
05A5 00
05A6 00
05A7 00
05A8 00
05A9 00
05AA 00
05AB 00
05AC 00
05AD 00
05AE 00
05AF 00
05B0 00
05B1 00
05B2 00
05B3 00
05B4 00
05B5 00
05B6 00
05B7 00
05B8 00
05B9 00
05BA 00
05BB 00
05BC 00
05BD 00
05BE 00
05BF 00
05C0 00
05C1 00
05C2 00
05C3 00
05C4 00
05C5 00
05C6 00
05C7 00
05C8 00
05C9 00
05CA 00
05CB 00
05CC 00
05CD 00
05CE 00
05CF 00
05D0 00
05D1 00
05D2 00
05D3 00
05D4 00
05D5 00
05D6 00
05D7 00
05D8 00
05D9 00
05DA 00
05DB 00
05DC 00
05DD 00
05DE 00
05DF 00
05E0 00
05E1 00
05E2 00
05E3 00
05E4 00
05E5 00
05E6 00
05E7 00
05E8 00
05E9 00
05EA 00
05EB 00
05EC 00
05ED 00
05EE 00
05EF 00
05F0 00
05F1 00
05F2 00
05F3 00
05F4 00
05F5 00
05F6 00
05F7 00
05F8 00
05F9 00
05FA 00
05FB 00
05FC 00
05FD 00
05FE 00
05FF 00
0600 00
0601 00
0602 00
0603 00
0604 00
0605 00
0606 00
0607 00
0608 00
0609 00
060A 00
060B 00
060C 00
060D 00
060E 00
060F 00
0610 00
0611 00
0612 00
0613 00
0614 00
0615 00
0616 00
0617 00
0618 00
0619 00
061A 00
061B 00
061C 00
061D 00
061E 00
061F 00
0620 00
0621 00
0622 00
0623 00
0624 00
0625 00
0626 00
0627 00
0628 00
0629 00
062A 00
062B 00
062C 00
062D 00
062E 00
062F 00
0630 00
0631 00
0632 00
0633 00
0634 00
0635 00
0636 00
0637 00
0638 00
0639 00
063A 00
063B 00
063C 00
063D 00
063E 00
063F 00
0640 00
0641 00
0642 00
0643 00
0644 00
0645 00
0646 00
0647 00
0648 00
0649 00
064A 00
064B 00
064C 00
064D 00
064E 00
064F 00
0650 00
0651 00
0652 00
0653 00
0654 00
0655 00
0656 00
0657 00
0658 00
0659 00
065A 00
065B 00
065C 00
065D 00
065E 00
065F 00
0660 00
0661 00
0662 00
0663 00
0664 00
0665 00
0666 00
0667 00
0668 00
0669 00
066A 00
066B 00
066C 00
066D 00
066E 00
066F 00
0670 00
0671 00
0672 00
0673 00
0674 00
0675 00
0676 00
0677 00
0678 00
0679 00
067A 00
067B 00
067C 00
067D 00
067E 00
067F 00
0680 00
0681 00
0682 00
0683 00
0684 00
0685 00
0686 00
0687 00
0688 00
0689 00
068A 00
068B 00
068C 00
068D 00
068E 00
068F 00
0690 00
0691 00
0692 00
0693 00
0694 00
0695 00
0696 00
0697 00
0698 00
0699 00
069A 00
069B 00
069C 00
069D 00
069E 00
069F 00
06A0 00
06A1 00
06A2 00
06A3 00
06A4 00
06A5 00
06A6 00
06A7 00
06A8 00
06A9 00
06AA 00
06AB 00
06AC 00
06AD 00
06AE 00
06AF 00
06B0 00
06B1 00
06B2 00
06B3 00
06B4 00
06B5 00
06B6 00
06B7 00
06B8 00
06B9 00
06BA 00
06BB 00
06BC 00
06BD 00
06BE 00
06BF 00
06C0 00
06C1 00
06C2 00
06C3 00
06C4 00
06C5 00
06C6 00
06C7 00
06C8 00
06C9 00
06CA 00
06CB 00
06CC 00
06CD 00
06CE 00
06CF 00
06D0 00
06D1 00
06D2 00
06D3 00
06D4 00
06D5 00
06D6 00
06D7 00
06D8 00
06D9 00
06DA 00
06DB 00
06DC 00
06DD 00
06DE 00
06DF 00
06E0 00
06E1 00
06E2 00
06E3 00
06E4 00
06E5 00
06E6 00
06E7 00
06E8 00
06E9 00
06EA 00
06EB 00
06EC 00
06ED 00
06EE 00
06EF 00
06F0 00
06F1 00
06F2 00
06F3 00
06F4 00
06F5 00
06F6 00
06F7 00
06F8 00
06F9 00
06FA 00
06FB 00
06FC 00
06FD 00
06FE 00
06FF 00
0700 00
0701 00
0702 00
0703 00
0704 00
0705 00
0706 00
0707 00
0708 00
0709 00
070A 00
070B 00
070C 00
070D 00
070E 00
070F 00
0710 00
0711 00
0712 00
0713 00
0714 00
0715 00
0716 00
0717 00
0718 00
0719 00
071A 00
071B 00
071C 00
071D 00
071E 00
071F 00
0720 00
0721 00
0722 00
0723 00
0724 00
0725 00
0726 00
0727 00
0728 00
0729 00
072A 00
072B 00
072C 00
072D 00
072E 00
072F 00
0730 00
0731 00
0732 00
0733 00
0734 00
0735 00
0736 00
0737 00
0738 00
0739 00
073A 00
073B 00
073C 00
073D 00
073E 00
073F 00
0740 00
0741 00
0742 00
0743 00
0744 00
0745 00
0746 00
0747 00
0748 00
0749 00
074A 00
074B 00
074C 00
074D 00
074E 00
074F 00
0750 00
0751 00
0752 00
0753 00
0754 00
0755 00
0756 00
0757 00
0758 00
0759 00
075A 00
075B 00
075C 00
075D 00
075E 00
075F 00
0760 00
0761 00
0762 00
0763 00
0764 00
0765 00
0766 0
```

1789	08	INY (increment high)	80	00	DETLEN LDY # 0
178A	8L E3	NOINY STY CURAD+1 (reset high)	81	E2	LDA (CURAD),Y
178C	60	RTS (return to caller)	82	01	BYTCNT LDY # 1
178D	18	ADVANC CLC	83	19	BIT TES3 (\$08 in 1981)
178E	A5 E2	LDA CURAD	84	08	ENE HALPOP (bit 3 = 1)
178F	65 E8	ADC BYTES	85	20	CMP #20 (is it JSR?)
1790	85 E8	STA CURAD	86	0F	BEQ THREE (length is 3)
1791	90 02	BCC PRELIM	87	9F	AND #9F (test bits 7-4)
1792	90 02	INC CURAD+1	88	0B	ENE TWO (if #0, length is 2)
1793	A2 04	PRELIM LDY # 4 (for 2 adchecks)	89	15	HALPOP AND #15 (retain bits 4-2)
1794	B5 E1	INLIMIT LDA BEGAD+1,X	90	05	CMP #1 (is only bit 0 = 1?)
1795	A8	TAY (save hi in Y)	91	05	BEQ TWO (if yes, length 2)
1796	D5 0F	CMP MOVAD+1,X	92	02	AND #5 (retain bits 2,0)
1797	00 0E	BCC LIMRET (error exit)	93	08	BEQ ONE1 (if =0, length one)
1798	D0 08	ENE TESTEX (hi ad is safe)	94	08	THREE INY (length 3 exit)
1799	E5 E0	LDA BEGAD,X (test lo ad)	95	08	TWO INY (length 2 exit)
17A0	D5 DE	CMP MOVAD,X	96	E8	ONE1 STY BYTES (save bytecount)
17A1	90 06	BCC LIMRET (error exit)	97	84	STY COUNT (save it again)
17A2	F0 05	BEQ ERREX (if ads=, error)	98	EA	RTS (return to caller)
17A3	CA	TESTEX DEX	99	60	MVDOWN LDA ENDAD+1 (this will set
17A4	CA	DEX	A0	E5	STA MOVAD+1 (MOVAD=ENDAD)
17A5	D0 EB	ENE INLIMIT (#0, continue)	A1	85	LDX ENDAD
17A6	60	LIMRET RTS (return to caller)	A2	E4	INX
17A7	18	ERREX CLC (clear carry, error)	A3	88	BEQ DECEX (this will
17A8	60	RTS	A4	05	decrement
17B2	20	MOVEUP JSR PRELIM (delete OK?)	A5	02	MOVAD)
17B3	90 27	BCC MVURET (no, exit)	A6	DF	DEC MOVAD+1
17B4	84 DF	STY MOVAD+1	A7	06	DEX
17B5	A6 E2	LDX CURAD	A8	DE	STX MOVAD
17B6	86 DE	STX MOVAD	A9	00	LDY # 0 (this will
17B7	A4 E8	LDY BYTES	B0	DE	move
17B8	B1 DE	LDA (MOVAD),Y	B1	DE	LDY BYTES (program)
17B9	A0 00	LDY # 0	B2	DE	STA (MOVAD),Y
17C0	91 DE	STA (MOVAD),Y	B3	E2	CPX CURAD (this will
17C1	91 DE	INX	B4	E2	exit when
17C2	E8	ENE CHKEND	B5	00	MOVAD=CURAD)
17C3	D0 02	INC MOVAD+1	B6	00	LDY MOVAD+1
17C4	E4 E4	BNE UPLOOP	B7	E8	CPY CURAD+1
17C5	D0 ED	LDY MOVAD+1	B8	00	BCC BRKERR (to 00 at 01B6)
17C6	A4 DF	CPY ENDAD+1	B9	00	ENE MVLOOP
17C7	00 E7	BCC UPLOOP	BA	00	CLC (readjust ENDAD)
17C8	8A	TXA (CURAD in X now=ENDAD)	BB	E4	LDA ENDAD
17C9	E5 E8	SBC BYTES	BC	E8	ADC BYTES
17D0	85 E4	STA ENDAD	BD	E4	STA ENDAD
17D1	B0 03	BCC MVURET	BE	02	BCC MVDRET
17D2	C6 E5	DEC ENDAD+1	BF	E5	INC ENDAD+1
17D3	38	SEC	C0	00	MVDRET RTS
17D4	60	MVURET RTS (return to caller)	C1	00	
17D5	A2 06	LIMITS LDY # 6 (for 3 adchecks)	C2	00	
17D6	00 B7	ENE INLIMIT	C3	00	
17D7			C4	00	
17D8			C5	00	
17D9			C6	00	
17DA			C7	00	
17DB			C8	00	
17DC			C9	00	
17DD			CA	00	
17DE			CB	00	
17DF			CC	00	
17E0			CD	00	
17E1			CE	00	
17E2			CF	00	
17E3			D0	00	
17E4			D1	00	
17E5			D2	00	
17E6			D3	00	
17E7			D4	00	
17E8			D5	00	
17E9			D6	00	
17EA			D7	00	
17EB			D8	00	
17EC			D9	00	
17ED			DA	00	
17EE			DB	00	
17EF			DC	00	
17F0			DD	00	
17F1			DE	00	
17F2			DF	00	
17F3			E0	00	
17F4			E1	00	
17F5			E2	00	
17F6			E3	00	
17F7			E4	00	
17F8			E5	00	
17F9			E6	00	
17FA			E7	00	
17FB			E8	00	
17FC			E9	00	
17FD			EA	00	
17FE			EB	00	
17FF			EC	00	