Commodore

The Transactor

Commodore Canada's
Tech/News Periodical

VOLUME 3
Issue #6

Our Last Issue!

Well... this is it... the last issue of The Transactor. After considering the alternatives, we've decided to discontinue publication at Commodore Canada in favour of **Commodore, The Microcomputer Magazine**, our U.S. counterpart.

After only a few issues, the U.S. have established a circulation of over 10,000! With that much penetration, we believe that a combination of U.S. and Canadian material will not only reach more Commodore users, but make **The Microcomputer Magazine** the best resource for PET/CBM and VIC information!

I'd like to extend thanks to all contributing writers for all three volumes. Particular regards to Jim Law, Dave Berezowski, Henry Troup, Ted Evers, Kevin Erler, Charles McCarthy, Gord Campbell, and Sieg Deleu. Extra special thanks go out to Don White, Dave Hook, and Jim Butterfield; your exceptional brand of enthusiasm and undying efforts are found all too seldom, but appreciated by so many.

The spirit of The Transactor will live on! Material generated within Commodore Canada by Paul Higginbottom, Joe Ferrari, Donna Green, Dave Berezowski, Peter Velocci, and myself, will be sent stateside. Articles submitted from outside will, as always, be gratefully accepted... so keep 'em coming! Your feedback together with our input will produce a resonating support interface, amplified by **Commodore, The Microcomputer Magazine!**

Sincerely Yours,
Karl J. Hildon
Editor, The Transactor

6 issue subscription in : U.S.A. $10.00
(published bi-monthly) Canada & Mexico $25.00 US Dlrs
Outside N. America $25.00 US Dlrs
Or see your local Commodore dealer for a newstand copy!

Send name and return address with cheque or m/o payable to :

Commodore Business Machines, Inc.
681 Moore Road,
King of Prussia, PA, 19406
Attn: Editor, Commodore Magazine

The Transactor has been produced on the CBM 8032 with WordPro 4+ and the NEC Spinwriter.

Index Transactor #6

Bits & Pieces 3
  Faster Than A Speeding Cathode Ray. 3
  More One-Liners 3
  Deriving Mathematical Functions ... 4
  Graphics Tablet 4
  CB2 Amplifier 5
  SuperPET RS232 6
  Some VIC Notes 6
  Attention COMAL-80 Users! 7
Pretty Printing 8
Compiler Comments 11
BASIC Labels Re-Revisited 14
4022 Printer Notes 20
Turning The Switch?.. Allow Your PET!. 22
Machine Language Auto-Location 30
1981 PET Bibliography 34
Two Terminal Programs: IEEE & RS232 .. 40
BASIC-AID, A Super Editor For The PET. 52

Bits & Pieces

Faster Than A Speeding Cathode Ray!

These next one-liners come from Richard Griffith of Thunder Bay, Ontario.

```
10 PRINT"[CLR DN DN]IS NOT YOUR NAME ABE LINCOLN?":GOTO10
```

The string in the above statement prints and clears so fast that the screen can't keep up! You might expect the text to 'flash' on and off. But, as the trace is scanning the screen, the text actually prints, clears, and prints again before the trace gets a chance to erase. It's hard to say how many, but BASIC prints and clears several times during a single screen scan. Therefore, the text appears to be stationary, as if the Clear Screen character was not even there!

Then, they become 'un-syncronized'. At this point, the text appears to be drawn slowly across the line. The trace draws part of the text and then it's turned off again by the Clear Screen. The same thing happens next time around only a little farther to the left or right. It's rather hard to explain but not hard to imagine when you're looking at it.

Try different combinations by adding or removing CLRs, DNs, characters, commas and semicolons. For an interesting effect, add line 20 by simply duplicating line 10 (remove the GOTO 10 and add it at the end of line 20). Try this one too:

```
10 PRINT"[CLR 6DN]IS YOUR NAME ABE LINCOLN?";:GOTO10
```

6DN = 6 cursor downs. Different machines produce different results. These were done on forty column PETs. 80 column users will have to modify the statements slightly to get the right effect since the scan speed is somewhat different.

Richard also has a one-line game which surely could be expanded! It uses the SHIFT key as a control. The first line does all the work, the second merely gets it going.

```
1 POKE A+T, 81:PRINT SPC(RND(TI)*36)"###":T=T+PEEK(152)*2-1:
IF PEEK(A+T)=32 THEN 1
2 PRINT "[CLR 24DN" : T=0 : A=32768
```

More One-Liners

These ones from Dave Berezowski of Thunder Bay.

```
1 FOR X=0 TO 999 : POKE 32768+X,
      (PEEK(32768+X)+128)AND255:NEXT
```

```
1 a=32768:i=0:j=38
2 S=SGN(J-I) : FOR X=I TO J STEP S : POKE A+X, 32 :
POKE A+X+S, 87 : NEXTX : I=39-I : J=39-J : GOTO 2
```

Deriving Mathematical Functions

BASIC has some trignometric functions implemented but not all that may at some time be required. Here is a handy list:

Secant	SEC(X) = 1/COS(X)
Cosecant	CSC(X) = 1/SIN(X)
Cotangent	COT(X) = 1/TAN(X)
Inverse Sine	ARCSIN(X) = ATN(X/SQR(-X*X+1))
Inverse Cosine	ARCCOS(X) =-ATN(X/SQR(-X*X+1)) +π/2
Inverse Secant	ARCSEC(X) = ATN(X/SQR(X*X-1))
Inverse Cosecant	ARCCSC(X) = ATN(X/SQR(X*X-1)) +(SGN(X)-1*π/2
Inverse Cotangent	ARCCOT(X) = ATN(X)+π/2
Hyperbolic Sine	SINH(X) = (EXP(X)-EXP(-X))/2
Hyperbolic Cosine	COSH(X) = (EXP(X)+EXP(-X))/2
Hyperbolic Tangent	TANH(X) = EXP(-X)/(EXP(X) +EXP(-X))*2+1
Hyperbolic Secant	SECH(X) = 2/(EXP(X)+EXP(-X)
Hyperbolic Cosecant	CSCH(X) = 2/(EXP(X)-EXP(-X)
Hyperbolic Cotangent	COTH(X) = EXP(-X)/(EXP(X) -EXP(-X))*2+1
Inverse Hyperbolic Sine	ARCSINH(X) = LOG(X+SQR(X*X+1))
Inverse Hyperbolic Cosine	ARCCOSH(X) = LOG(X+SQR(X*X-1))
Inverse Hyperbolic Tangent	ARCTANH(X) = LOG((1+X)/(1-X))/2
Inverse Hyperbolic Secant	ARCSECH(X) = LOG((SQR(-X*X+1) +1/X))
Inverse Hyperbolic Cosecant	ARCCSCH(X) = LOG((SGN(X) *SQR(X*X+1/X))
Inv. Hyperbolic Cotangent	ARCCOTH(X) = LOG((X+1)/(X-1))/2

Graphics Tablet

Looking for a graphics tablet? KURTA seems to have the answer. It has an 8 1/2" x 11" surface with low radiation for diskette protection. Pen operated with 100 to 200 points per square inch. KURTA supplies all the software and interfaces making it completely compatible with PET/CBMs. For more information, contact:

In the U.S.

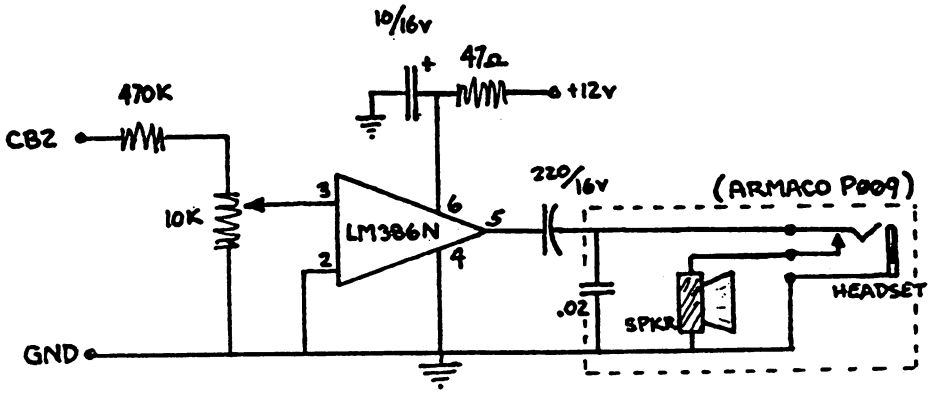
KURTA Corp.
206 S. River Dr.
Tempe, Arizona
85281
602 968 8709

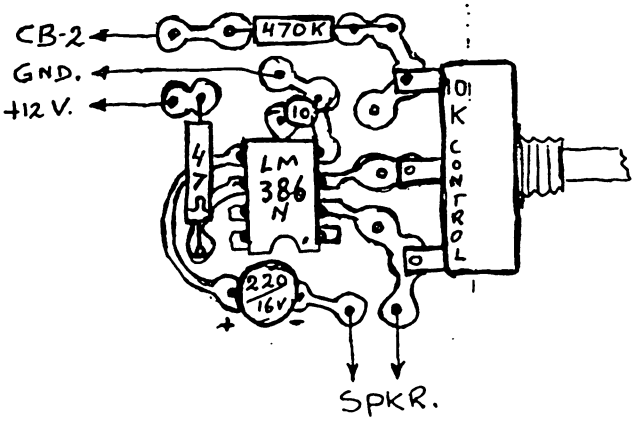
In Canada:

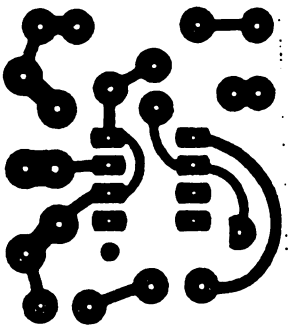
TCS Communications
1158 Victoria St. N.
Kitchener, Ontario
N2B 3C9
519 744 5071

CB2 Amplifier

This tidy little circuit cam from Ted Evers of Toronto. Connect it to the User Port CB2 line, ground, and one of the 12 volt pins inside the machine, and you've got CB2 sound (with optional headphones jack to prevent raging parents, teachers and wives).

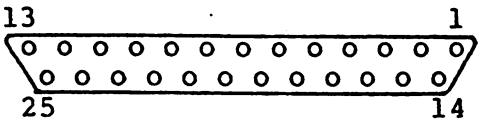






Attention SuperPET RS232 Interfacers!

Here are the pin connections for the RS232 Port of the SuperPET:



Pin	Description	Mnemonic	Comments
1	Protective Ground		
2	Transmitted Data	TXD	Output
3	Received Data	RXD	Input
4	Request To Send	RTS	Output
5	Clear To Send	CTS	Input
6	Data Set Ready	DSR	Input
7	Signal Ground		
8	Data Carrier Detect	DCD	Input
20	Data Terminal Ready	DTR	Output

The Data Set (Modem) controls inputs, CTS, DSR, and DCD are expected to operate in the normal way. If the device you connect to the SuperPET serial port does not control these pins, you should connect pins 6 and 8 to pin 20, and pin 5 to pin 4 within the connector attached to the SuperPET board. With these connections some devices may then be operated with a 4-wire cable using only TXD, RXD and the 2 grounds.

Another note to SuperPET users, COBOL should be available around the end of July '82. It will be made availble FREE to all existing SuperPET owners and included with any new SuperPET deliveries. For more details, contact your local dealer sometime in July.

Some VIC Notes

Vic-20 owners that wish to connect to colour monitors will need some extra cables. You could wire them up yourselves; for pin connections, see the VIC 20 Programmers Reference Guide, pp.282. Remember, the connectors shown here are as you look at them from the back of the VIC. The corresponding pins on the jacks will be "mirror image" looking into the jack.

You can buy the necessary cables at any Radio Shack. First you'll need a "Y" Adapter (Part# 42-2394); a 5 pin European plug to 4 phono jacks. This one goes on the audio/video connector. It's not very long, so you'll also need the 1.8m shielded stereo cable (42-2352). Most colour monitors use a female BNC connector for video in. In this case you'll need the ARCHER female RCA phono to male BNC adapter (278-254).

"Y" Adapter Connections

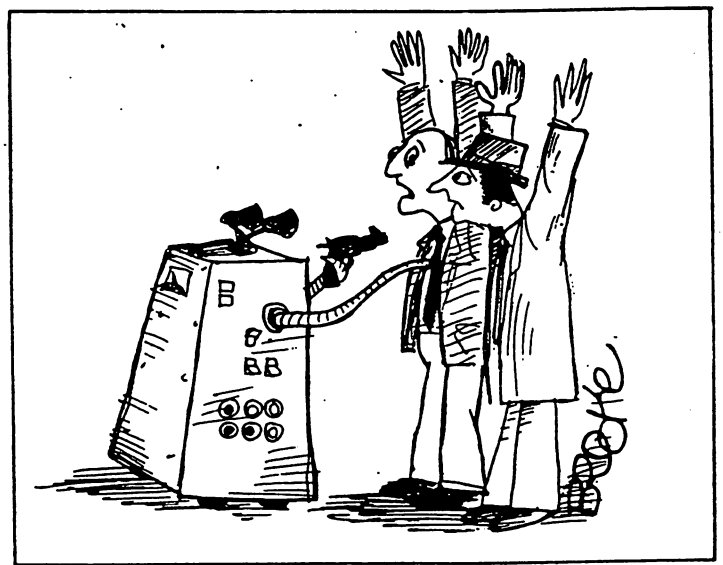
Black - Video low
Grey - Audio
White - Video High
Red - +5V Regulated @10 mA. max.

The 1.8m extension has black and grey connectors at each end. Use black for video and grey for audio.

Attention COMAL 80 Users!

We're collecting a list of bugs in COMAL-80. If you've been using COMAL, you've probably found some. Two that crash the machine are Division by Zero and Device Not Present. If you have any more, please send them to us and we'll pass them on. Hopefully version 00.12 will have them all corrected. Send any comments to:

COMAL 80 Feedback
Commodore Business Machines
3370 Pharmacy Ave.
Agincourt, Ontario
M1W 2K4



'The Increase in Computer Crime Is Frightening.'

Pretty Printing

Jim Butterfield, Toronto

When you are producing output, it's good to make it neat. The computer is there to help its human readers, and the more you can do to improve the information, the better job you'll be doing.

Printing in Columns

Beginners often arrange values in columns by using the screen tabulation functions: putting a comma into the PRINT statement, or using the TAB function. These methods work, but they have a pitfall: they won't behave properly if the output goes to other devices. The problem is that the computer always knows exactly where the screen cursor is, but it never knows on what column the external devices are located. It doesn't even try to keep track; so a TAB or a comma directed to the printer or other device won't behave properly.

It's my feeling that almost everything that goes the screen can be usefully directed to the printer, or written to a disk file with a view to transferring to the printer later. Once you have a report looking nice on the screen, you don't want to have to reprogram to get it looking nice in print. So ... stay away from TAB and commas - there's a better way.

Redirecting Output

While I'm on the subject of switching output from the screen to the printer, I'd like to share a little coding trick with you. Most programmers know that you can direct output to a printer by performing an OPEN to device number 4 (the printer) and then using PRINT#... That's fine for a finished program, but you can waste a lot of paper while you're checking out a program if you do everything to the printer.

Here's the trick: We can OPEN to device number 3 (the screen) and PRINT# to the screen, checking our program and fixing it up. When it's ready to go, all we need to do is to change the OPEN statement so that it names device number 4, and output goes to the printer. We save time and paper. Let's try it: we code:

```
100 OPEN 1,3
110 FOR J=1 TO 10
120  PRINT#1,J;SQR(J)
130 NEXT J
140 CLOSE 1
```

When we run this program, output is delivered to the screen. If everything looks good, we can now change line 100 to OPEN 1,4 ... and output is redirected.

It's not really a trick; it's good coding. We could allow the user to specify what output he wanted by coding something like: 100 INPUT"DEVICE NUMBER";N :OPEN 1,N so that the user could type in 3 or 4 to select the type of output he wants.

Neatness Counts

If I'm sternly discouraging TAB and the comma, how can you arrange things in columns? A few simple answers, but first some ground rules. The best way to arrange stuff in columns is to make sure that each "field" is always the same length; that way, each item will be printed neatly in the same place across the page.

How can we rechop two numbers as different as 3 and -32768 so that they occupy the same space? For that matter, how can we take two names as different as BUTTERFIELD and NG and make them the same length?

Let's take the names first. These "strings" could be neatly chopped down to a fixed length by means of the LEFT$(function ... if they were long enough. For example, we could slice out the first eight characters of string X$ with LEFT$(X$,8); but it won't work if X$ is less than eight characters long in the first place. So - pay attention - we must first pad out the name by adding spaces to the end. Sticking extra characters onto the end of a string is called "concatenation" - pronounced with emphasis on the cat - and is done with a plus sign. If we had a short name like M and wanted to tack eight spaces on the end, we'd do it by writing "M"+" " which would create a new string nine characters long. A name like BUTTERFIELD treated the same way would end up nineteen characters long, but this doesn't matter: we're going to chop them both down to the same length with LEFT$(.

Let's put it all together. If the name is held in variable N$, we code PRINT LEFT$(N$+" ",8); with a semicolon at the end. First we concatenate, adding the spaces; then we chop (or "truncate"), cutting to a fixed length; finally we print. Both long and short names will be printed as exactly eight characters; the next thing we print will be neatly lined up behind it. We might want to make the field more than eight characters long, since a splendid name like BUTTERFIELD would end up chopped to BUTTERFI - if we do increase the length we must remember to add more spaces, of course.

The above procedure is called Left Justification, since the strings are lined up neatly on the left with spaces filling out the right hand side. We can go the other way and produce Right Justification with a small adjustment: try PRINT RIGHT$(" "+N$,8); and you'll see how the left side fills with spaces and names line up on the right. This is the kind of alignment you will want with numbers; we'll deal with that in a moment. Remember that if you don't allow enough space you'll end up with chopped-off names like TERFIELD, and there's no justification for that...

If the numbers you are using are integers, you'll usually want to line them up with right justification. Once again, this is easy to do once you know the function that changes

numbers to strings. If your value is held in variable X, we can change it to a string with STR$(X); now we can do the right justification with PRINT RIGHT$("   "+STR$(X),6); everything will work out neatly. Study this statement and see how X builds up into a neatly justified string of length six.

If your numbers contain fractional values, you may want to try to line up the decimal points. That's much more challenging. Perhaps you'd like to try your hand at it. We'll tackle it here another time.

VIGIL



INTERACTIVE GRAPHICS/GAME LANGUAGE

FOR THE PET/CBM

VIGIL is an exciting new interactive language for your PET/CBM micro. VIGIL - Video Interactive Game Interpretive Language - is an easy to learn graphics and game language that lets you quickly create interactive applications.

* More than 60 powerful commands permit you to easily manipulate graphics figures on the screen
* Double density graphics give you 80 X 50 plot positions on your 40 column PET/CBM
* Large number display capability, access to two event timers and tone generation (if you have ext. speaker)
* Load and save your VIGIL programs to cassette
* Nine interactive programs demonstrate the power of VIGIL - Breakout, SpaceWar, AntiAircraft, U.F.O., SpaceBattle, Concentration, Maze, Kaleidoscope & Fortune
* Comprehensive user's manual with complete listings of enclosed programs

VIGIL comes on cassette, ready to run on any 40 column PET/CBM micro with at least 8K of memory. Specify ROM-set when ordering. 6502 listing of the VIGIL Interpreter available separately.

	US & Canada	Foreign
VIGIL for PET/CBM on CASSETTE (w/nine programs)	$35	$40
VIGIL User's Manual (refundable with software)	$10	$12
VIGIL Interpreter listing (6502 Assembly language)	$25	$30
PET MACHINE LANGUAGE GUIDE	$8	$10



ABACUS SOFTWARE
P. O. Box 7211
Grand Rapids, Michigan 49510

Prices include postage. Michigan residents include 4% sales tax. Orders must be prepaid or via bankcard (Mastercard, VISA, Eurocard, Access, etc.). Include card number and expiration date.

(C) 1981 by Roy Wainwright

Compiler Comments

Jim Butterfield, Toronto

I don't want to become involved in the Great Debate about compilers. On the other hand, it's almost irresistable to dive in and add a few footnotes. You'll find no product reviews here. Just a little talk about what's involved.

For BASIC?

Some languages were designed for compilers. In fact, the compiler was designed first, and whatever it turned out you had to type in ended up as the language. FORTRAN started more or less this way. To put compilers in perspective, we have to do a little historical work.

Once, long ago, there were no interactive computers. You punched up a deck of cards and if you were lucky an operator would run them sometime that week. Most of the results came back saying something like SYNTAX ERROR (does that sound familiar?). There was no point in having an interpreter language; you wouldn't be there to watch it happen. We had FORTRAN and COBOL and others...

The first FORTRANs, for example, were tricky. If you used a variable called DIGIT, it would turn out to be a floating-point number; on the other hand a variable called NUMBER would be fixed-point. Heaven help you if you typed TOTAL=TOTAL+1; you'd get a ?MIXED MODE error notice and have to recode TOTAL=TOTAL+1.0 to fix it. To input or output you needed to give more than the command: an extra line called FORMAT was needed, written in advanced gibberish. Honest.

Many of these problems have been fixed up over the years - you did know that there was more than one FORTRAN, didn't you? - but the style remains. The programmers have to adapt to the machine, and interactive is still an alien concept.

And Now, BASIC...

Along came BASIC. It's a loose language: you don't have to dimension some arrays; strings wander all over; sometimes you can have FOR and NEXT items that don't match (bad practice, but it can be done) ... and interactive users love it.

What's the problem? Things that are not clearly defined by BASIC. Let's look at a few of them.

Strings may be the worst thing that a compiler has to deal with. BASIC doesn't tell the compiler how big any string is likely to be - ever. INPUT X$ gives no hint as to the size of string X$. The poor compiler has a grim choice: allow maximum space for all strings and waste a lot of memory; or bounce the strings around as they change. The first alternative costs you program size; you write this little program that says DIM A$(1000) and the compiler immediatey reports OUT OF MEMORY since it tries to allocate 255000 bytes for the array. The second alternative costs you time; no matter what you call it, some sort of garbage collection will have to take place. And then people complain because they expect compilers to produce fast fast code.

At first glance we think that the whole object of compiling is to get speed. But we don't give the compiler enough information to work up a really fast program. It's obvious that FOR J=1 TO 10 can run faster if we treat J as an integer. Unfortunately, we're not allowed to code FOR J%... so the compiler will have to figure it out for itself. And what will it do with FOR J=A TO B? Until A is computed, we cannot know if it's integer or not.

It's obvious to us. We wrote the program. But the dumb compiler can't read our minds; and BASIC doesn't give enough explicit information to do the job.

One last example. It's one of the annoying things about BASIC that we sometimes have to code things like GET#1,X$: IF X$="" THEN X$=CHR$(0) mostly to cover failings in BASIC itself. If I were hand-coding into machine language, I could replace the whole thing with one instruction, because I know that Machine Language doesn't have the "fault" that's in BASIC. But a poor compiler can't know that. It sees the GET instruction and codes it... and it must add to the coding to generate the BASIC "fault" if it wants to be compatible. Then it must proceed to the IF statement and work through the coding to fix that same fault.

The Choices.

The compiler designer has a choice. He can code for 99% compatibility, tracking everything that BASIC does quite exactly (including the faults). In doing so, he'll create a package in whioh almost anything will compile sucessfully. But - the compiled machine language will be doing most of the things that BASIC does, and won't be much faster than BASIC.

On the other hand, the designer can ask the user to make changes to his program before compilation that will help the process. He may also have things that compile from BASIC in a non-standard manner. He may make arbitrary decisions on BASIC structures - all FOR loop variables will be fixed-point, for example. And the compiler may question the user during compilation: How large is string M$ likely to be? Can J be fixed-point? The user has to work harder, but the end product runs faster.

Either way, the compiled program is not likely to be smaller in size than its BASIC source. It's difficult to code 100 IFJ>5THENPRINT"J IS";J in less than the 19 bytes that BASIC uses. And good compilers add extra arithmetic - fixed-point addition, for example - that takes up overhead space.

Why Compile?

It's your choice. If you have a program that runs for five hours, you will probably be delighted with a paltry four-to-one compiler speedup. If you want protection against listing, a compiler will do a good job of instant obfuscation.

Don't lose perspective. A program that spends most of its time waiting for an operator or for a printer won't speed up much under compilation.

Machine Lanuage Programmers will be happy to know that they are not yet obsolete. Compilers can do a useful job. But until they get the brains equivalent to a human's judgment, they won't replace hand coding.

BASIC Labels Re-Revisited

Charles A. McCarthy
St. Paul, MN.

In Transactor Vol. 2, No. 12 (Best Of The Transactor Vol. 2, pp.172-177), J. Hoogstraat presented a program that makes a major contribution to the development and maintenance of PET BASIC programs: BASIC GOTO and GOSUB targets may be meaningful words. (A version for BASIC 4.0 appears in Transactor Vol. 3, No. 1).

For a number of reasons, I found it advisable to slightly re-work his program. The principal differences and the reasons for the changes are:

1. A bug involving stack handling has been corrected. In the original version, an interrupt that occurs during lines 1240-1270 (of the assembly listing in Vol. 2) will remove a flag from the stack. The result is that a BASIC GOSUB #Label command will be interpreted as a GOTO #Label. This happens randomly, on the average about once every one to two thousand times that a GOSUB #Label is encountered.

2. The original program makes free use of the .X and .Y registers, and in particular, destroys .Y each time the CHRGET routine at $0070 is called. I am loathe to use these registers until absolutely sure that it is safe to do so. Thus the present version returns with .X and .Y unchanged except in those cases that I am certain about.

3. I wanted to make this program available to our local PET users group. The code should be relocatable, which is easy: the only position dependent code in the original version is the call to the subroutine CORRECT which tests for a terminator to a #Label, and this can equally well be done inline. More importantly, for those who do wish to position this code in the second cassette buffer, the locations #03E0-$03F9 should be left free for Toolkit. Despite my best efforts at tightening up the code, the only way I could make it fit in $033A-03DF was to abandon the initialization routine HOOKUP. HOOKUP may either be placed following the present version, in which case a SYS 990 may be done before Toolkit is activated, or it may be placed in your BASIC program hidden in a REM statement (for details see F. VanDuinen, Transactor Vol. 2, No. 8 Program Plus in the section Within BASIC).

4. At the start of a target line (a BASIC line that begins with #Label), a comma seems inapropriate as a terminator because the call to SKPSTT will skip over the comma to the following colon or end-of-line null byte. Thus the present version does not test for a comma as a terminator here.

5. The program looks at the return address on the stack to decide what action it should take. These addresses are $C83E (THEN), $C7AC (GOTO), $C78F (GOSUB), and $C869 (ON..). The original version, and this one also, check only the low byte of this address. I am uneasy about this, so the version that I now use for myself tests the high byte also (it lives in high memory where space is not so improtant). I very much doubt that such caution is needed, but I want even less to find out that it is.

The following listing was made from working code, printed out by Supermon (Thank you Jim!). I have added labels and comments and have tried to follow Mr. Hoogstraats' original.

Charles A. McCarthy
1359 W. Idaho Av.
St. Paul, MN. 55108
(612) 645 6867

('*-' BASIC 4.0 users take note!)

```
.,  033A 4C EB C7   JMP $C7EB     ;UNDEF'D STATEMENT ERROR
                                  ;(*- $B86E) Issued when a
                                  ;searched for #Label is not
                                  ;found.  Placed here so an
                                  ;inadvertent SYS826 doesn't
                                  ;cause a crash
LABELS
.,  033D E6 77      INC $77       ;Perform TXPTR inc for
.,  033F D0 02      BNE $0343     ;CHRGET
.,  0341 E6 78      INC $78
.,  0343 8A         TXA           ;Save .X on stack
.,  0344 48         PHA
.,  0345 A2 FF      LDX #$FF      ;Test immediate mode
.,  0347 E4 37      CPX $37
.,  0349 F0 06      BEQ $0351     ;Go exit if so
.,  034B A1 78      LDA ($78,X)   ;Run mode: check for '#'
.,  034D C9 23      CMP #$23      ;Note (78,FF)=(77,00)
.,  034F F0 05      BEQ $0356     ;Go test if found
.,  0351 68         PLA           ;else exit to CHRGOT after
.,  0352 AA         TAX           ;restoring .X
.,  0353 4C 76 00   JMP $0076
```

```
CHKLAB
.,  0356 68          PLA          ;restore .X
.,  0357 AA          TAX
.,  0358 68          PLA          ;Low byte of calling addr.
.,  0359 C9 3E       CMP #$3E     ;(*- #$C1)
.,  035B F0 1A       BEQ $0377    ;do THEN (Carry set)
.,  035D C9 AC       CMP #$AC     ;(*- #$2F)
.,  035F F0 16       BEQ $0377    ;do GOTO (Carry set)
.,  0361 C9 8F       CMP #$8F     ;(*- #$12)
.,  0363 F0 11       BEQ $0376    ;do GOSUB
.,  0365 48          PHA          ;Low byte to stack in case
.,  0366 C9 69       CMP #$69     ;nothing to do... (*- #$EC)
.,  0368 D0 6E       BNE $03D8    ;then go SKPSTT and exit
.,  036A 68          PLA          ;do ON..  Pull rtn addr off
.,  036B 68          PLA          ;stack
.,  036C 20 70 00    JSR $0070    ;advance TXTPTR to comma
.,  036F C9 2C       CMP #$2C     ;following #Label
.,  0371 D0 F9       BNE $036C
.,  0373 4C 5F C8    JMP $C85F    ;(*- $B8E2) exit to ON.RET

;THEN/GOTO vs. GOSUB information is saved in carry bit

.,  0376 18          CLC          ;GOSUB entry
.,  0377 68          PLA          ;finish pulling rtn address
.,  0378 08          PHP          ;Save which (PLA doesn't
                                   affect Carry)

;At this point, free use may be made of .X and .Y  SETLAD
makes no use of .X and .Y, and SKPSTT needs neither on entry,
but uses both.

FLABEL
.,  0379 A5 28       LDA $28      ;Init BASIC text ptr to
.,  037B A6 29       LDX $29      ;search for line starting
.,  037D A0 00       LDY #$00     ;with #Label.  Hi byte in .X
.,  037F F0 04       BEQ $0385    ;Lo byte in .A

NXSTAT
.,  0381 A0 00       LDY #$00     ;Ptr to present line in $5C
.,  0383 B1 5C       LDA ($5C),Y  ;Ptr+4 points to text for
.,  0385 18          CLC          ;this line
.,  0386 85 5C       STA $5C
.,  0388 69 04       ADC #$04
.,  038A 85 5A       STA $5A
.,  038C 8A          TXA
.,  038D 85 5D       STA $5D
.,  038F 69 00       ADC #$00
.,  0391 85 5B       STA $5B
.,  0393 C8          INY          ;.Y=1
.,  0394 B1 5C       LDA ($5C),Y  ;chk for null link, endprog
.,  0396 F0 A2       BEQ $033A    ;if so, #Label not found
.,  0398 AA          TAX          ;save next line hi byt in .X
.,  0399 88          DEY          ;.Y=0
```

```
MATCH
.,  039A B1 5A      LDA ($5A),Y ;Test for #Label terminator
.,  039C F0 0F      BEQ $03AD   ;null (end of line)
.,  039E C9 3A      CMP #$3A    ;colon (end of statement)
.,  03A0 F0 0B      BEQ $03AD
.,  03A2 C9 20      CMP #$20
.,  03A4 F0 0B      BEQ $03AD   ;No terminator--test against
.,  03A6 D1 77      CMP ($77),Y ;given #Label--to NXSTAT if
.,  03A8 D0 D7      BNE $0381   ;not this one.  Match so far
.,  03AA C8         INY         ;test nest character.
.,  03AB D0 ED      BNE $039A   ;forced branch
.,  03AD B1 77      LDA ($77),Y ;Terminator found this line
.,  03AF F0 0C      BEQ $03BD   ;test given #Label for term.
.,  03B1 C9 2C      CMP #$2C    ;terminatrs are null, comma,
.,  03B3 F0 08      BEQ $03BD
.,  03B5 C9 3A      CMP #$3A    ;colon,
.,  03B7 F0 04      BEQ $03BD
.,  03B9 C9 20      CMP #$20    ;blank.
.,  03BB D0 C4      BNE $0381   ;No terminatr, try next line

;Match found.  We transfer BASIC execution to the appropriate
line of BASIC text with JSR SETLAD, then skip over to the
#Label that begins this line using JSR SKPSTT, and resume
normal execution.  First, however, if we have a GOSUB, we
must prepare the stack for the eventual RETURN statement.

.,  03BD 28         PLP         ;Recall THEN/GOTO vs GOSUB
.,  03BE B0 15      BCS $03D5   ;Carry set for THEN/GOTO
.,  03C0 A5 78      LDA $78     ;Carry clear, so put
.,  03C2 48         PHA         ;rtn data on stack
.,  03C3 A5 77      LDA $77
.,  03C5 48         PHA
.,  03C6 A5 37      LDA $37
.,  03C8 48         PHA
.,  03C9 A5 36      LDA $36
.,  03CB 48         PHA
.,  03CC A9 8D      LDA #$8D
.,  03CE 48         PHA
.,  03CF A9 C6      LDA #$C6    ;(*- #$B7)
.,  03D1 48         PHA
.,  03D2 A9 C3      LDA #$C3    ;(*- #$49)
.,  03D4 48         PHA
.,  03D5 20 CD C7   JSR $C7CD   ;JSR SETLAD (*- $B850)
.,  03D8 20 00 C8   JSR $C800   ;JSR SKPSTT (*- $B883)
.,  03DB 4C 76 00   JSR $0076   ;JMP to CHRGOT and return

HOOKUP
.,  03DE A9 4C      LDA #$4C    ;SYS here to engage (990)
.,  03E0 85 70      STA $70
.,  03E2 A9 47      LDA #$3D
.,  03E4 85 71      STA $71
.,  03E6 A9 03      LDA #$03
.,  03E8 85 72      STA $72
.,  03EA 60         RTS
```

Editor's Note

Here are two BASIC loaders for this newest rendition of the BASIC Label Support Interface; one for BASIC 2.0 and the other for BASIC 4.0. For those using disk with BASIC 4, you'll need to move the routine. BASIC 4.0 disk commands use parts of the 2nd cassette buffer (826-1017) and will clobber it good! You could move it up high in memory and seal it off, but the 1st cassette buffer (634-825) will do nicely (unless you're using cassette #1 too). To set up the routine here, simply change AD=826 to AD=634. In this case, the activating SYS will change from SYS 990 to SYS 798.

```
90 REM *** BASIC LABEL SUPPORT INTERFACE DEMO ***
100 FORI=1TO3
110 ON I GOSUB #SUB1, #SUB2, #SUB3
120 NEXT
130 GOTO #ALLDONE
140 :
150 #SUB1:PRINT"SUBROUTINE";I:RETURN
160 :
200 #SUB2
210 PRINT"SUBROUT";I
220 RETURN
300 :
310 #SUB3    : PRINT"SUBROUT 3
320 RETURN
500 :
510 #ALLDONE : PRINT"END ALLDONE
```

```
900 REM BASIC 4.0 LABEL SUPPORT INTERFACE
910 AD=826                      : REM DISK USERS, CHANGE TO AD=634
920 CH=0                        : REM RESET CHECKSUM
930 FOR J=AD TO AD+176
940 READ X : CH=CH+X            : REM ACCUMULATE SUM
950 POKE J, X
960 NEXT
970 PRINTCH                     : REM PRINT CHECKSUM
980 REM *** CHECKSUM SHOULD EQUAL 21952 ***
990 END
1011 DATA  76, 110, 184, 230, 119, 208,   2, 230, 120, 138,  72
1022 DATA 162, 255, 228,  55, 240,   6, 161, 120, 201,  35, 240
1033 DATA   5, 104, 170,  76, 118,   0, 104, 170, 104, 201, 193
1044 DATA 240,  26, 201,  47, 240,  22, 201,  18, 240,  17,  72
1055 DATA 201, 236, 208, 110, 104, 104,  32, 112,   0, 201,  44
1066 DATA 208, 249,  76, 226, 184,  24, 104,   8, 165,  40, 166
1077 DATA  41, 160,   0, 240,   4, 160,   0, 177,  92,  24, 133
1088 DATA  92, 105,   4, 133,  90, 138, 133,  93, 105,   0, 133
1099 DATA  91, 200, 177,  92, 240, 162, 170, 136, 177,  90, 240
1110 DATA  15, 201,  58, 240,  11, 201,  32, 240,  11, 209, 119
1121 DATA 208, 215, 200, 208, 237, 177, 119, 240,  12, 201,  44
1132 DATA 240,   8, 201,  58, 240,   4, 201,  32, 208, 196,  40
1143 DATA 176,  21, 165, 120,  72, 165, 119,  72, 165,  55,  72
1154 DATA 165,  54,  72, 169, 141,  72, 169, 183,  72, 169,  73
1165 DATA  72,  32,  80, 184,  32, 131, 184,  76, 118,   0, 169
1176 DATA  76, 133, 112, 169,  61, 133, 113, 169,   3, 133, 114
1177 DATA  96
```

```
900 REM BASIC 2.0 LABEL SUPPORT INTERFACE
910 AD=826
920 CH=0                        : REM RESET CHECKSUM
930 FOR J=AD TO AD+176
940 READ X : CH=CH+X            : REM ACCUMULATE SUM
950 POKE J, X
960 NEXT
970 PRINTCH                     : REM PRINT CHECKSUM
980 REM *** CHECKSUM SHOULD EQUAL 22127 ***
990 END
1011 DATA  76, 235, 199, 230, 119, 208,   2, 230, 120, 138,  72
1022 DATA 162, 255, 228,  55, 240,   6, 161, 120, 201,  35, 240
1033 DATA   5, 104, 170,  76, 118,   0, 104, 170, 104, 201,  62
1044 DATA 240,  26, 201, 172, 240,  22, 201, 143, 240,  17,  72
1055 DATA 201, 105, 208, 110, 104, 104,  32, 112,   0, 201,  44
1066 DATA 208, 249,  76,  95, 200,  24, 104,   8, 165,  40, 166
1077 DATA  41, 160,   0, 240,   4, 160,   0, 177,  92,  24, 133
1088 DATA  92, 105,   4, 133,  90, 138, 133,  93, 105,   0, 133
1099 DATA  91, 200, 177,  92, 240, 162, 170, 136, 177,  90, 240
1110 DATA  15, 201,  58, 240,  11, 201,  32, 240,  11, 209, 119
1121 DATA 208, 215, 200, 208, 237, 177, 119, 240,  12, 201,  44
1132 DATA 240,   8, 201,  58, 240,   4, 201,  32, 208, 196,  40
1143 DATA 176,  21, 165, 120,  72, 165, 119,  72, 165,  55,  72
1154 DATA 165,  54,  72, 169, 141,  72, 169, 198,  72, 169, 195
1165 DATA  72,  32, 205, 199,  32,   0, 200,  76, 118,   0, 169
1176 DATA  76, 133, 112, 169,  61, 133, 113, 169,   3, 133, 114
1177 DATA  96
```

4022 Printer Notes

First I'd like to mention that a new ROM is available for the 2022 printer. The 901472-07 replaces the original 03 ROM and the subsequent 04. Recall, the 03 forced a carriage travel for each line feed and the 04 would occasionally lock into lower case. The 07 fixes all previous bugs, but due to 2022 mechanics it can't give bi-directional print. For details on the 07, contact your nearest Commodore dealer.

Now on to the 4022. Faster, sleeker, quieter, a nicer character set, plus all the features of the 2022 make the 4022 a bargain at only $995 Canadian. The 4022P has since replaced the 4022. It has all the features of its predecessor with bi-directional print capability added. This new ROM for 4022s is available for retrofitting.

The manual has a few minor oversights which we'll clear up now:

1) Page 31 states, "...144 steps per inch, so a declared value of 18 produces 8 lines per inch." It should read, "...195 steps per inch, ...28 produces 8 lines per inch."

2) Page 31 also says, "PRINT#6,CHR$(144) produces lines spaced one inch apart." Secondary address 6 does not accept values over 127. Therefore CHR$(127) will result in maximum line spacing (approx. 1/2 inch).

3) Next it says, "default value is 24 for the standard 6 lines per inch." Change this to 36.

Here are some line spacing values for secondary address 6 (lpi = lines per inch):

CHR$ Value	Result	Comments
1	195 lpi	;characters will overlap
7	25 lpi	;characters still overlap
12	16.6 lpi	
14	14.2 lpi	
21	10 lpi	;characters stop overlapping
28	8 lpi	
36	6 lpi	
64	5 lpi	
66	4 lpi	
97	3 lpi	

The only other difference we've found occurs when "skipping" from one formatted field to the next. Like the 2022, the 4022(P) supports 'printing data according to a previously defined format'. In order to "skip" from one field to the beginning of the next, it was necessary to send a CHR$(29), or 'Cursor-Right'. This still applies to alpha fields, but when sending numerics to secondary address 1 on the 4022, the skip character is no longer needed. It seems

that the full designator
tags to the end of numerics and numeric variables when output. For example:

```
2022:   PRINT#1,A;CHR$(29);B;CHR$(29);C
4022:   PRINT#1,A;B;C
```

If the extra cursor-rights are sent to the 4022, the printer will skip two fields instead of just one.

Turning The Switch ?... Allow Your PET!

Kevin Erler,
Edson Alta.

After spending upwards of $3000 on your home computer, you'll find yourself with a brilliant machine that can perform fabulous feats of thinking, play master chess to perfection, balance your budget, and calculate the cost of heating your home, but, physically, can it DO anything?

What to Use:

The problem of computer control has been addressed with solutions ranging from the complex A.C. conducted control signals to the simple (and cumbersome) relay switching systems. I chose the former, however, as mean as it sounds, it is actually quite simple. The interface that follows implements an already developed, and readily accessible device called the BSR-X10 home control system. The system X-10 features control of sixteen different electrical appliances using the different modules. The control can be remote (through the command console) or local (by using the appliance's actual switch). Lights have additional ability to have their intensity varied. The BSR-X10 system is available for approximately $40 (plus modules).

The X-10's keyboard (figure 1) is very straight forward. The number of the appliance or lamp is entered, followed by the command (on, off, bright, or dim). The two other keys are to allow all of the devices attached to the system to be turned on or off (ie. ALL ON, ALL OFF). The variable intensity of the lights is determined by the amount of time the BRIGHT or DIM switch is held down. The vast increase of the computer's realm of control can be easily imagined if the computer could grasp control of these few buttons.

How To Do It:

Technically, the unit is built around a chip called the 542-C, which supports a 3 by 8 matrix keyboard. The chip first puts a -5 volt pulse on the first strobe line, and scans for the same pulse on the eight input lines. If it doesn't see one, the unit will carry on with the second strobe line, and so on. The rate at which it strobes these lines is about 3780 times per second, and the input on the appropriate pin must be present at that exact moment. As is easily seen, this incredible speed can provide slight problems. To overcome this huge timing problem, one could go to either software or hardware. While software would present us with very complicated programming tasks as well as the need for a program that would have to run constantly, the hardware approach proved quite simple. By using a 74LS153 multiplexing chip, the three strobe lines could be fed into it, and the computer could select which of the three could be gated through to the output pin (by use of a two bit binary input). With this done, the output would then be pulsed at the rate as the selected strobe line. This output could then be used as an enable line to a 3 to 8 line binary decoder (74LS138). With all this, the data from the computer could be fed right into the X-10. Simple, right? Right, but we're still not quite finished.

For the average person that isn't really interested in how the hardware works, all that has been done to this point is the formulation of a very simple two piece interface. No thinking required... at least not yet. Read on...

The Power:

The problem is that the BSR system operates on negative logic, while the computer operates on positive logic. Basically, the two power supplies are incompatible.

The Solution: we isolate the BSR system from the power line, and then tie the ground side of its chip's power into the computer power supply. The only additional part required is an isolation transformer, the cheapest of which is a shaver transformer like the one in your bathroom (available at your local hardware store). Cut off the plug from the BSR system and splice its power cord into one side of the isolation transformer. Now, after removing the cover from the X-10 (it will only come off part way), cut the plastic insert that holds the power cord in place (remove it totally), and take out the screws that hold the bottom PC board in place. Removing this board, locate the two printed circuit strips as indicated in figure 3, and very carefully scratch out these two lines with a small knife. Make sure there is no connection after you're done. Next, solder a piece of wire (about three feet) to each of the points shown in figure 3. Also solder a third wire onto the indicated jumper in figure 2. This wire connects to +5 volts in the computer so cut an appropriate length. Replace the PC board in the BSR system, allowing all three wires to exit out the same hole as the power cord. Run the first two wires over to the isolation transformer, and splice them (one to each side) along with an A.C. cord to the unused side. With all this assembled it should look something like figure 4.

Finished ??

The hard part is over. Now, with 5 minutes work, we can slap together the interface in figure 5. The numbers on the far right hand side are the pin numbers of the 542-C. Perhaps the easiest way to connect the interface to the X-10 is to remove the top PC board from the X-10, and solder the wires directly onto the chip. You are now completely finished with the hardware!

A few quick notes, this device is meant to connect to the User Port (ie. the 6522 VIA), not directly to the CPU. The keyboard of the BSR will not function with the interface attached, however, I believe the remote control unit available for certain BSRs would. Although it would be quite easy to build this interface on epoxy perfboard, I just used a breadboard (available any electronic shop). Finally, a quick rundown of the parts required:

BSR-X10 (model #014311)	approx. $40
74LS138	approx. $ 1
74LS153	approx. $ 1
Isolation transformer	approx. $ 5
Total	approx. $47

The Software:

All of the software required to operate the system is based on some simple numbers which represent the different keys of the X-10's keyboard. The following table lists these necessary codes:

Key	Code	Key	Code
1	100	12	56
2	36	13	116
3	112	14	52
4	48	15	124
5	104	16	60
6	40	ON	80
7	108	OFF	88
8	44	DIM	92
9	96	BRIGHT	76
10	32	ALL ON	72
11	120	ALL OFF	84

First, plug in your BSR system on turn on your computer. If there's no smoke then you've done everything right.

Next, set up the Data Direction Register (DDRA) of the 6522 to all outputs except PA0 and PA1: POKE 59459,252

Now, our first command can be entered with these stipulations:

- Commands are issued to the DATA OUTPUT REGISTER, 59471
- Directly after a command is issued, it should be turned off
- A command is turned off with: POKE 59471, 128
- After a command is turned off, another one should not be issued for at least 0.2 seconds

For example, to turn all appliances off, this command could be entered directly from the keyboard:

```
POKE 59471, 84 : POKE 59471, 128
```

For a sequence of two commands:

```
POKE 59471, 100 : POKE 59471, 128 : FOR I = 1 TO 200 :
NEXT : POKE 59471, 80 : POKE 59471, 128
```

The above would turn device #1 on. To tidy things up for a program, the following subroutine could be called after each command is sent:

```
10000 POKE 59471, 128 :FOR I = 1 TO 200 : NEXT : RETURN
```

One final note; when using the 'DIM' or 'BRIGHT' commands, a certain delay is necessary before the command is turned off:

```
POKE 59471, 92 : FOR I = 1TO400 : NEXT : POKE 59471, 128
```

This may dim your light about half, experiment as required.

But Wait!

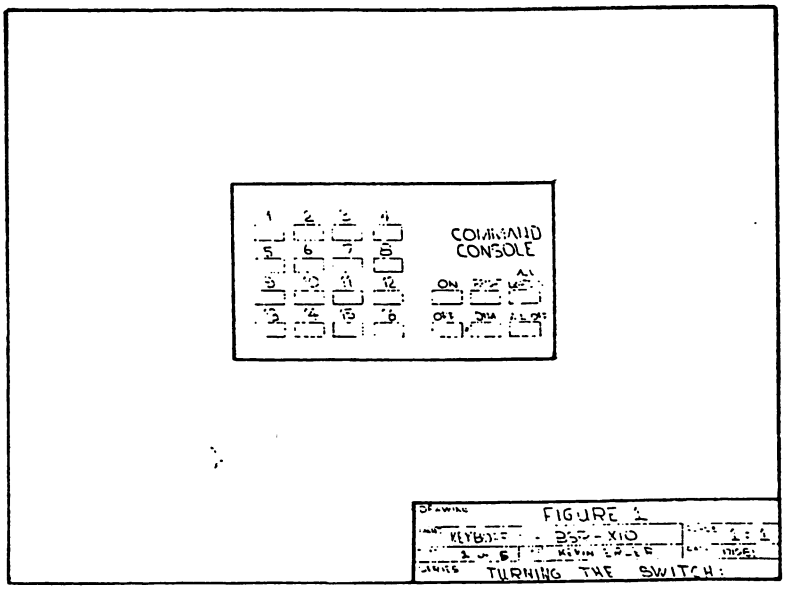
Imagine waking up to a dim light which is slowly gaining its proper form. The radio comes on with the morning news (a pre-determined time in your program), and downstairs your coffee is almost ready. As you leave your room, the light winks out and the room you enter is suddenly completely lit. Sitting down to catch the weather on TV, all that is necessary is a whisper, "TV, please". It's on.

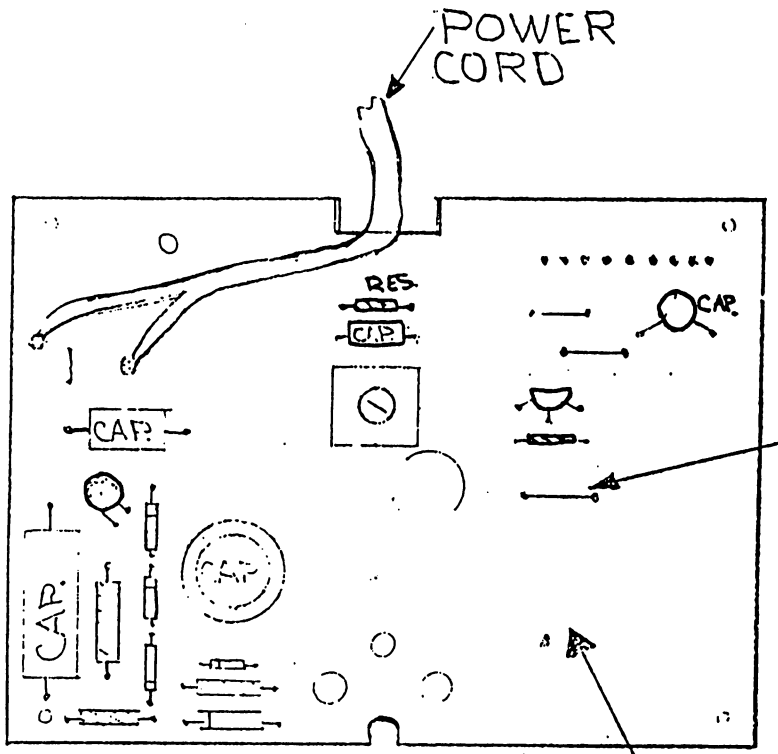
Sound like fantasy, it isn't so hard. Actually, with the system you have just built, the first half is already possible. With the addition of a couple of photo-cells, and a speech recognition unit, so is the rest. From here the possibilities are, of course, endless.

Editor's Note

Congratulations Kevin on an excellent idea AND implementation. A couple footnotes that deserve mention: The BSR system doesn't connect directly to your lights and appliances, but rather through modules that plug into the wall. The appliance then plugs in to the module which is serviced by a high frequency signal placed on your house wires by the main command unit. Several different types of modules are available for regular wall sockets, light switches and appliances. The BSR system is available at any Eatons hardware department.

Secondly, a unit known as the Cognivox will do voice recognition AND speech synthesis of up to 32 words and/or phrases. For more information, contact Voictek, PO Box 388, Goleta CA, 93116. Or call 805 685 1854.

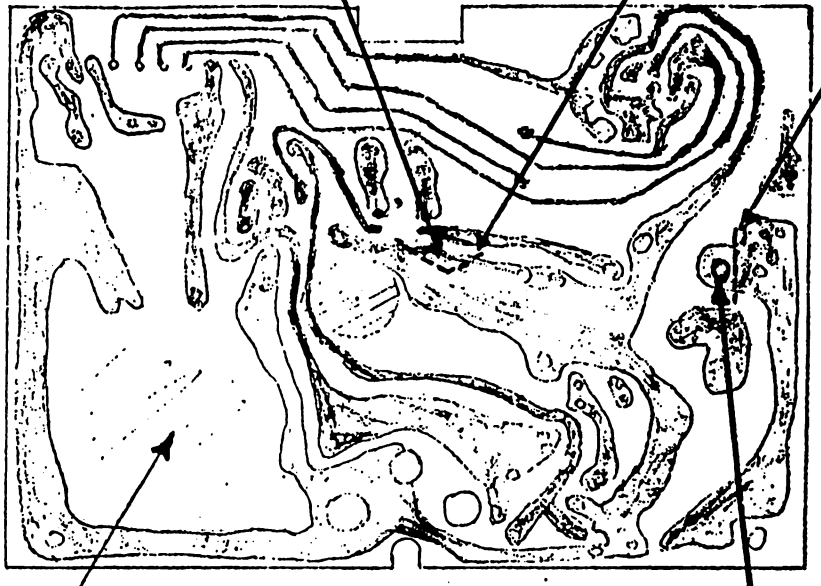




DRAWING: FIGURE 2		
PART: TOP VIEW - BOTTOM PCB		SCALE: 1 : 1
PLATE: 2 OF 5	NAME: KEVIN ERLER	DATE: 1/10/97
SERIES: - TURNING THE SWITCH:		

SOLDER FIRST WIRE TO THIS POINT.

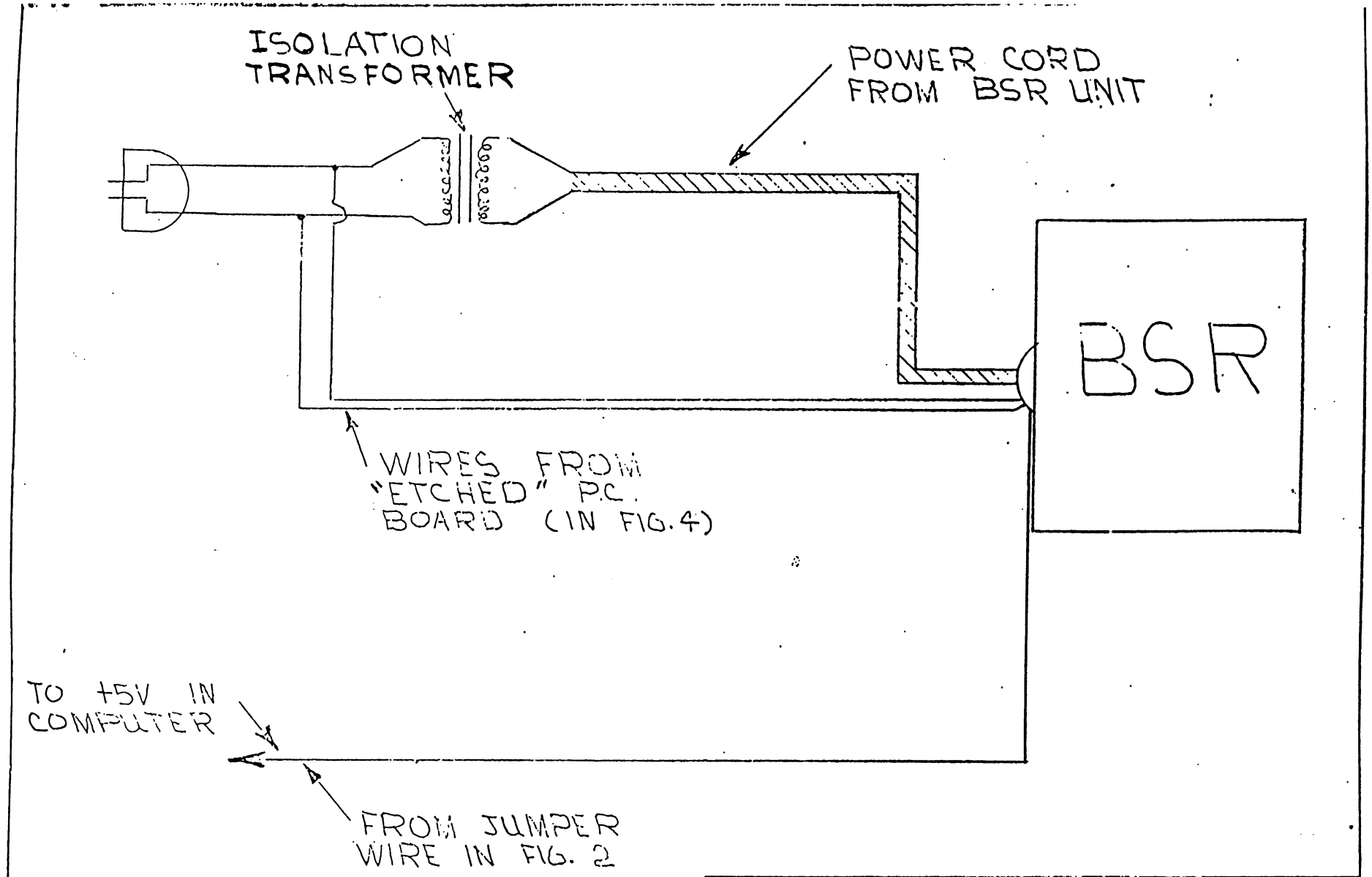
ETCH ALONG DOTTED LINES



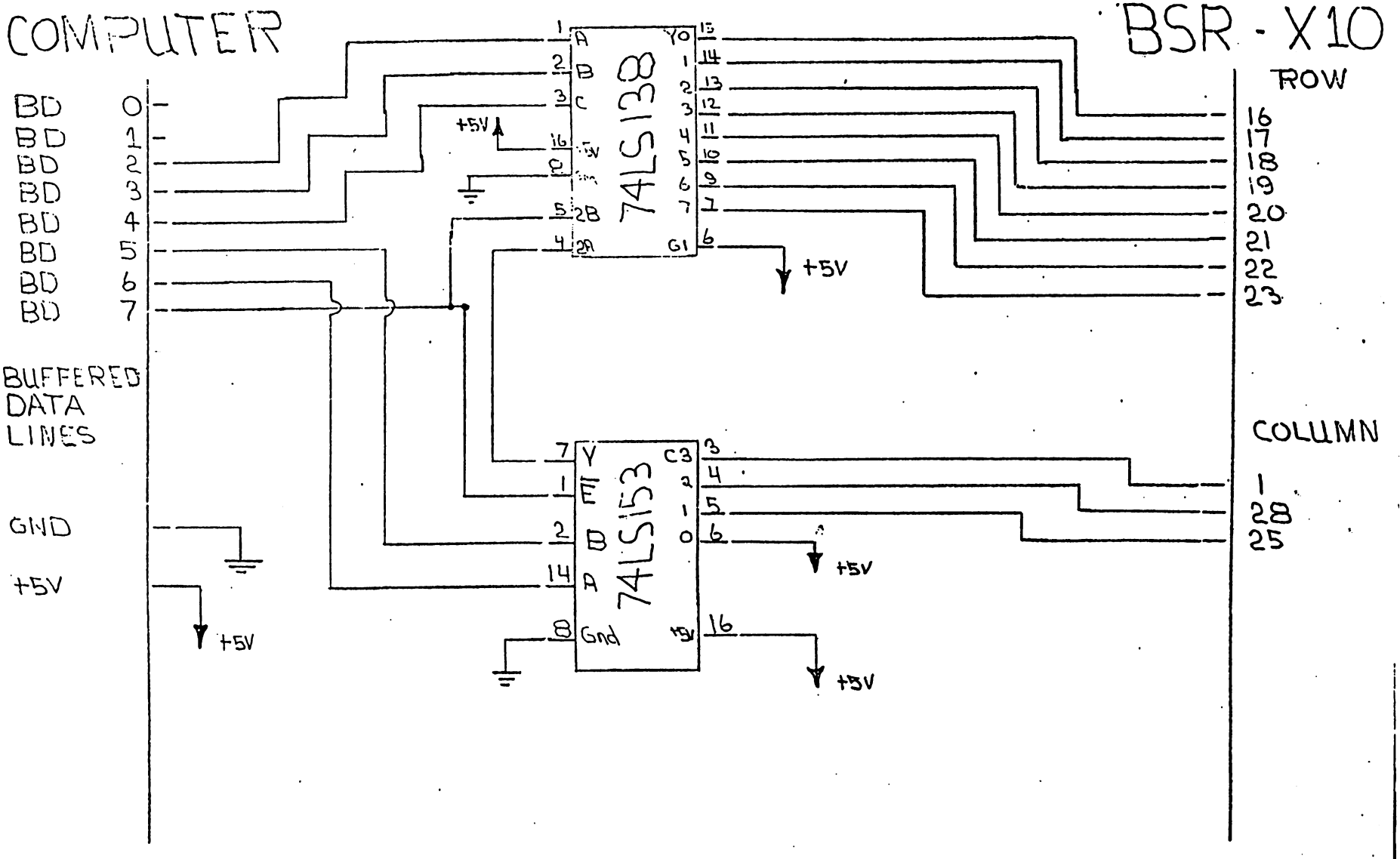
SOLDER OTHER WIRE TO THIS POINT

THIS AREA USED WITH ULTRASONIC BSR ONLY

DRAWING: FIGURE 3		
PART: BOTTOM VIEW - BOTTOM PCB		SCALE: 1:1
PLATE: 3 OF 5	NAME: KEVIN ERLER	DATE: 7/10/81
SERIES: TURNING THE SWITCH		



DRAWING: FIGURE 4		
PART: COMPLETED VERSION		SCALE: ___
PLATE: 4 OF 5	NAME: KEVIN ERLER	DATE 1/10/89
SERIES: TURNING THE SWITCH:		



DRAWING: FIGURE 5	
PART: SCHEMATIC	SCALE: ---
PLATE: 5 OF 5 NAME: KEVIN ERLER	DATE: 7/10/81
SERIES: TURNING THE SWITCH:	

Machine Langauge Auto-Location

Jim Butterfield,
Toronto

When a program like Supermon or Tinymon loads into its computer and RUN is given, it builds a copy of the "real" program in high memory. There's a need to do this: different computers have different memory sizes, and we want to find the top of memory wherever it is. More: the computer might already have something else near the top of memory (such as a wedge program) and we want the new program to fit neatly below it.

This calls for an auto-location program. The object program must be packed into high memory. This is often more than just moving the program, since some things may need to be changed with the move. If you have a program that uses only branches - no jumps, no in-program subroutines, no tables - you may be able to get away with a simple move operation. But any instruction that uses an in-program absolute address: jumps, subroutine calls, and tables - will need to be adjusted.

We need to build a relocatable program module. Something that says, "This byte is normal so we may just move it; but that pair of bytes is an address and must be recalculated for the new location".

Ground Rules.

We need a scheme which marks addresses so that the proper arithmetic may be performed. There's one requirment as to how you write the program: it may be summarized as "all addresses must be in one piece"..

The rule makes sense: it would be difficult to perform arithmetic on an address whose two bytes were scattered in different parts of the program. For users with assemblers, the rule translates to: never use the < or > functions for high and low byte.

So if we wanted to place the address of TABLE into indirect address INDAD, we would avoid coding: LDA #<TABLE : STA INDAD : LDA #>TABLE : STA INDAD+1. Instead, we'd define the table address in memory with TABLAD .WORD TABLE and perform the above setup with LDA TABLAD : STA INDAD : LDA TABLAD+1 : STA INDAD+1. We've used four more bytes but gained a major benefit: the two bytes representing the address of TABLE are now stored together (at TABLAD) and we can adjust this address easily when we wish to relocate.

The Method.

The way we build a relocatable module is quite easy. Any time we see an address that will need relocation, we place a zero above it. As we repack the program (from the top down) the zero will signal that a relocatable address follows.

That's all very well, but what do we do with real zeros? There will be many zeros in the program itself, and we don't want them to trigger a false relocation calculation. In this case, we change the zero to two zeros in the relocatable package. The relocation program will spot this and change it back to a single zero.

In order to do arithmetic on the addresses, we need to know where they are pointed in the first place. To relocate from $1000 to $4000, for example, we need to add $3000; but we must know that we are starting from $1000. I use the following convention: addresses are written so that the top of the program plus one is $0000 - that is, the last byte of the relocatable program is $FFFF. The program can't really go there, since that's ROM space, but it makes the arithmetic easy. We can look at an address in the relocation package as a signed number: address $FFC0 can be viewed as "64 bytes from the top of the program". If our real top-of-program turned out to be $8000, which would be correct for a 32K machine, we would translate the sequence 20 C0 FF 00 to 20 C0 7F ... note that the zero disappears; it's the relocation flag. How did we get the new address $7FC0? By adding the relocation address, $FFC0, to the top-of-program, $8000.

Generating the Relocatable Program.

How do we manufacture this package with zeros added and addresses recalculated, ready for relocation? With an assembler it's quite easy.

First, we assemble two versions of the program at two different locations. That's easy enough to do: we just change the *= statement at the start of our source code.

Then we run a simple compare program which compares the two object programs we have assembled, starting from the top. Each matching byte is copied into the relocation area unchanged; if it's a zero, an extra zero is added. If the bytes don't match, we have a relocatable address: in this case, we insert the zero plus the recalculated address into the relocation package. It's an easy job: my "relocate builder" is a BASIC program of about a dozen lines.

Stopping.

As we work down from the top we need to detect when we have reached the end of the program: this is true of both the relocate builder and the relocating program itself. There are many easy ways of doing it. The program can test to see if the last address has been reached. Alternatively, we can put some sort of "flag" into the coding itself to detect the end. In TINYMON, I use a value $BF which is never used in the program as a simple detection. A more complete method might be to use a zero with a value of 1 stored below it. It's up to you: whatever works is OK.

VIC Note.

In the VIC, we have one more problem to solve. We can find the top of memory (locations $37 and $38) but our program might fall into different memory space, depending on what's plugged in. Use pointers to find your own program (try $2D and $2E) and everything should work out nicely.

Summary.

You can pick apart the code of SUPERMON or TINYMON and see how it's done. You can develop your own programs. But if you understand the principles of a relocating program package, you can develop significantly more useful programs which will adapt to a wider variety of machine configurations.

Editor's Note

The machine code dissassembly to follow is Jim Butterfields relocator modified slightly by Dave Hook for use with his Vicloader for PET/CBMs (see Transactor #5, Vol3). Dave eliminated the JMPs and JSRs in Jim's original utility so that the relocator can be relocated. For Vicloader, it starts at $0640, but you can move it anywhere; higher if you want more BASIC underneath it, or lower for larger object programs.

Notice that the relocator starts with the end of the object program since this will be the first byte to be packed into high memory. This is conveniently pointed at by the Start of Variables pointer minus 1, which is set on completion of the LOAD (provided it is .Saved properly).

```
0400-063F   BASIC portion (title, sys address, etc)

0640 A5 2A      LDA $2A         ;store copy of
0642 85 1F      STA $1F         ;Start of Variables
0644 A5 2B      LDA $2B         ;pointer (last byte of
0646 85 20      STA $20         ;object program + 1).
0648 A5 34      LDA $34         ;store copy of
064A 85 21      STA $21         ;Top of Memory
064C A5 35      LDA #35         ;pointer (MemTop)
064E 85 22      STA $22
0650 A0 00      LDY #$00        ;zeroise Y index
0652 A5 1F      LDA $1F         ;dec pointer to last
0654 D0 02      BNE $0658       ;byte of object prog.
0656 C6 20      DEC $20         ;(1st byte to be
0658 C6 1F      DEC $1F         ;packed)
065A B1 1F      LDA ($1F),Y     ;get obj. prog. byte
065C D0 3C      BNE $069A       ;not 0, goto $069A
065E A5 1F      LDA $1F         ;if 0, dec pointer
0660 D0 02      BNE $0664
0662 C6 20      DEC $20
0664 C6 1F      DEC $1F
0666 B1 1F      LDA ($1F),Y     ;and get next byte
0668 F0 21      BEQ $068B       ;0? yes, true zero *
066A 85 23      STA $23         ;no, relocatable addr
066C A5 1F      LDA $1F         ;store high byte in
066E D0 02      BNE $0672       ;$23.  dec pointer
0670 C6 20      DEC $20         ;and
0672 C6 1F      DEC $1F
0674 B1 1F      LDA ($1F),Y     ;get next byte
0676 18         CLC             ;recalculate lo addr
0677 65 21      ADC $21         ;using MemTop lo
0679 AA         TAX             ;result in .X
067A A5 23      LDA $23         ;recalculate hi addr
067C 65 22      ADC $22         ;using MemTop hi
067E 48         PHA             ;result on stack
067F A5 34      LDA $34         ;dec MemTop
0681 D0 02      BNE $0685
0683 C6 35      DEC $35
0685 C6 34      DEC $34
0687 68         PLA             ;retrieve hi addr
0688 91 34      STA ($34),Y     ;pack at ($Memtop) .Y=0
068A 8A         TXA             ;retrieve lo addr
068B 48         PHA             ;* save on stack
068C A5 34      LDA $34         ;dec MemTop
068E D0 02      BNE $0692
0690 C6 35      DEC $35
0692 C6 34      DEC $34
0694 68         PLA             ;retrieve byte
0695 91 34      STA ($34),Y     ;pack at ($MemTop) .Y=0
0697 18         CLC             ;rather than
0698 90 B6      BCC $0650       ;a JMP
069A C9 BF      CMP #$BF        ;last byte?
069C D0 ED      BNE $068B       ;no, goto $068B *
069E A5 34      LDA $34         ;yes, set
06A0 85 30      STA $30         ;Bottom of Strings
06A2 A5 35      LDA $35         ;= MemTop
06A4 85 31      STA $31         ;pointer
06A6 6C 34 00   JMP ($0034)     ;jmp to program
06A9 BF                         ;end detector of obj prog
06AA ...                        ;start of object prog
```

1981 PET Bibliography

Don White
Ottawa 6502 User Group

The following is a bibliography of PET related articles published in 1981 in 'Creative Computing', 'Kilobaud Microcomputing' and 'COMPUTE!'.

Creative Computing 1981

January

Page 24 No PET Peeves-New Computers From Commodore
156 Personal Electronic Transactions

February

18 Music Editors for Personal Computers
154 Personal Electronic Transactions

March

26 Wordpro 1 vs CMC
78 A PET Lizzard (Game Listing)
166 Reading Level: Determination & Evaluation (Listing)

April

222 Personal Electronic Transactions

May

96 Break Even Analysis With Visicalc
208 Personal Electronic Transactions

June

26 The TNW 2000
30 Fantasy Games
36 Computer Warfare
88 Software Techniques of Digital Music Synthesis Pt 1

July

50 The Paper Mate
140 Software Techniques of Digital Music Synthesis Pt 2

August

24 The Last One
122 Tree (Game Listing)
144 PET Nuclear Power Plant (Game Listing)

September - Buyer's Guide

43 Commodore VIC-20

October

54 Educational Software and Books
160 Bombproofing the PET INPUT Statement

November

65 Dynacomp Bridge Challenger
192 PET Screen Line Length

December

76 Valdez: A Supertanker Simulation
240 Helping Students Think About Marriage & Education

Kilobaud Microcomputing 1981

January

10 PET-Pourri - New PET Monitor
48 Real-Time Spectrum Analyser
78 Scramble (Game Listing)
188 Second Cassette Interface (Hardware Modification)

February

12 PET-Pourri - Jinsam
56 London Computer Club A Huge Success
72 Portrait Of A Dynamic French Company

March

8 PET-Pourri - Handy Utilities
144 PET Shorthand Compleat

April

10 PET-Pourri - VIC-20 Debuts

May

10 PET-Pourri - Conversing In Assembly Language
179 A PIE Taster's Report
195 Soulful Software Sounds (Listing)
200 Find That Program! (Listing)

June

10 PET-Pourri - ROM Packages From Skyles
92 Once Upon A Time (Listing)
177 Expand PET Memory (Listing)

July

10 PET-Pourri - 8032 Data Handlers
104 A One-Two Punch For CBM/PET Graphics
167 Get On The PET Instrument Bus (Listing)

August

10 PET-Pourri - Commodore Colors NCC
152 What's The Difference (Listing)

September

12 PET-Pourri - CBM Utilities

October

10 PET-Pourri - Commodore's Big Push
195 PET Goes To The Polls (Listing)

November

14 PET-Pourri - VIC Expands Its Horizons
50 Popping And Pushing Permutations In BASIC

December

10 PET-Pourri - Word Pro Enhancement
66 Putting The Joy Back Into Programming (Listing)
114 A BASIC Assembler For The PET (Listing)
178 Put To The Test By A Computer

COMPUTE!

January

32 The Mysterious & Unpredictable RND Pt 1
38 CURSOR Classifications Revisited
44 ODDS & ENDS..re PET cassette tape
92 The Screen Squeeze Fix For CBM 8000
96 Horray For SYS
102 Machine Language:Scanning The Stack
108 The PET Revealed & Library Of PET Subroutines
110 A Visible Music Monitor
112 Disk-O-Pro
114 Detecting Loading Problems & Correcting Alignment
118 Spooling For PET With 2040 Disk Drive
118 Variable Dump For New ROM PETs
120 The 32K Bug
121 An Ideal Machine Language Save For The PET
122 PET Metronome
123 PET IEEE Bus:Standing Room Only?
124 PET/CBM IEEE Bus Error

February

16 LED - A Line-Oriented Text Editor
30 Simulated PRINT USING
34 The Mysterious & Unpredictable RND Pt 2
54 Basic Math For Fun & Profit
60 PET Spelling Lessons Your Student Can Prepare
97 Contour Plotting
103 Relocate
104 Mixing & Matching Commodore Disk Systems
109 Memory Calendar

114 Crash Prevention On The PET
116 Machine Language Printer Command
118 ODDS & ENDS On PET/CBM Files
120 Three PET Tricks
124 PASCAL On The PET
126 The PEDISK
127 A Disk Operating System For The CGRS PEDISK

March

20 Taking The Plunge-Machine Language Programming
42 Getting The Most From Your PET Cassette Deck
48 The Mysterious & Unpredictable RND Pt 3
54 A CAI Program Called LINEAR EQUATION
92 Keyprint Revisited
96 Learning About Garbage Collection
102 PET Machine Language Graphics
112 Disk File Recovery Program
124 PET Exec Hello
130 A Flexible Input Subroutine
132 Universal Tape Append for PET/CBM

April

26 Commodore VIC-20:A First Look
34 How To Be A VIC Expert
46 The Mysterious & Unpredictable RND Pt 4
52 Micros With The Handicapped
56 Matrix Row Operations
117 Partition and Load
122 Relative File Mechanics
126 COPLOT
130 ROM Expansion For The Commodore PET
136 Working With BASIC 4.0
138 Papermate Word Processor
142 Dissecting C.W. Moser's ASSM/TED 1.0
144 PET File I/O and Machine Language
146 How To Get Started In Machine Code And Not Go Crazy With A Routine For Two Joysticks
152 Machine LAnguage:The Wonderful Wedge

May

22 The Mysterious & Unpredictable RND Pt 5
30 Land Of The Lost - Cassette Filing System
46 EPIDEMIC
58 Naming Compounds
96 A Fast Visible Memory Dump
112 Getting To The Machine Language Program
116 A Thirteen Line BASIC Delete
118 Calculated Bar Graph Routines On The PET
120 The Revised PET/CBM Personal Computer Guide
124 Un-Compactor
126 Using The Hardware Interrupt Vector On The PET
128 PET As An IEEE-488 Logic Analyser
130 Running 40 Column Programs On A CBM 8032

June

4 RAM/ROMs-A New Style Of Memory?
22 Mapping & Modifying Unknown Machine Language
52 Ideal-Gas Law
94 Relocation Of BASIC Programs On The PET
98 Memory Partition Of BASIC Workspace
100 Machine Language Code For Appending Disk Files
102 Quadra-PET:Multitasking On Your PET
106 PET/CBM Disk Formats
110 Interfacing With The User Of Your PET Programs
116 Keeping Tabs On Your Printer
120 Assembler In BASIC For The PET
128 Uncrashing
130 Notes On The PET SAVE Command
131 Optimized Data System PH-001 2114 RAM Adaptor
132 Discovering Tape File Names
132 Petbug
133 Machine Language Utility Pac

July

142 Saving ML Programs on PET Tape Headers
146 Commodore ROM Systems: Terminology
150 SCREENER:4 Screen Utility Routines
155 Machine Language: Comparison Shopping
156 Using TAB, SPC and LEN

August

30 Minimize Code And Maximize Speed
50 Add A Programmable Sound Generator
105 The CBM "Fat 40" - Boon Or Bane?
109 Digital Arrayment
120 Keyword
124 CBM/PET Loading, Chaining, and Overlaying
128 Converting PET BASIC Programs To ASCII Files

August - Home & Educational Computing

4 Exploring The Rainbow Machine
9 VIC As A Super Calculator
11 Custom Characters For The VIC
16 The Confusing Quote

September

30 The Column Calculator
36 PET, Atari, Apple: On Speaking Terms
103 The Unwedge-Tape Append And Renumber
108 STP-488 A Smart Terminal Program
118 4.0 Garbage Collection: A Small Bug
120 Using The Monitor On The PET
122 Odds And Ends: Relative Files on BASIC 3.0
124 2040 Disk Program Listing
128 All About LOADing PET Cassettes
134 Graph Plotting Routine
136 Linelist

October

28 VIC-20 News
30 Various VIC Memory Locations
30 Update Floating Color, Floating Screen
48 More Machine Language For Beginners
62 Undeletable Lines
126 Practical PET Printing Primer
132 A Fat Forty Bug
138 Train Your PET To Run VIC Programs
140 Converting To Fat-40
143 High Resolution Bar Graphs For The PET
146 Waking Up The PET Screen
149 Interfacing A BSR X-10 AC Remote Control System
156 Using Non-Pin-Feed Forms In Tractor Printer
159 Why You Should Use PEEK(155) Instead Of GET

November

26 A Flower Sale Program
28 SuperPET's Super Software
38 SuperPET: A Preview
54 Bits, Bytes And Basic Boole
136 POWER
142 The PET Speaks
145 Machine Language: Monitoring Progress
147 Directory For 3.0
148 Inversion Partitioning
151 A Personal News Service
155 FOR/NEXT GOSUB/RETURN, And The Stack

December

38 Subscript Heap Sort
54 Maze Generator
130 A Look At SuperPET
134 SUPERMON
142 PET To PET Communication Over The User Port
150 Replacing The INPUT# Command
154 Typing Foreign Language Text With The CBM Printer
158 Three Reviews: Superchip, Spacemaker, Sort
160 Machine Language: Jumbo Numbers
163 File Recovery
166 Looney Line Numbers
168 Mine Maze
172 COMAL: Another Language

Two Terminal Programs: IEEE and RS232

Recall in Transactor #4 we mentioned Steve Punters' Bulletin Board System in Mississauga, Ontario. Commodore is now distributing Steves' BBS as a package and we've since delivered about 15 to date. Although all 15 may not be set up as public systems, any that are will be listed with their phone numbers on Steves board (see Transactor #4 for operating hours).

Steve has also written two terminal programs for use with PET/CBMs and this BBS software. Although any ASCII terminal can access the system, only these two programs are capable of up/downloading files to/from this BBS. Programs, SEQ and WordPro files are all transmitted flawlessly using a special checksum protocal. Once you've sent a program to the BBS, others can then download it with one of these programs. Likewise, you can receive programs that others have submitted.

The two terminal programs are identical in operation. The one you choose will depend on what type of modem (300 baud) you have; "TERMINAL.Ixx" is for use with IEEE modems (Commodore 8010, Livermore Star); "TERMINAL.Rxx" is for use with RS232 modems (Novation Cat, General DataComm, etc.). These files contain the BASIC part of the programs. The number "xx" denotes the version number. The ones listed here are version 11. As new versions are released (and Steve assures me there will be), you can obtain them from the BBS using one of these.

Each program has corresponding machine language subroutines that are LOADed by the BASIC part. "term.ieee" and "term.rs232" are PRG files that will be generated for you by the programs listed. These are also available for downloading from the BBS, but are listed there somewhat differently. "term.ieee" will be listed as "TERM.I11" and "term.rs232" will be "TERM.R11". Once again, the "11" represents the version number. You may be asking, "Why the different filenames and why does one show a version number and the other not?". A new edition of the BASIC does not necessarily mean a new machine language part, and vice versa. For instance, TERM.I12 may be released for use with the existing TERMINAL.I11. When you get it, simply rename "TERM.I12" to "term.ieee" and away you go; this way no editting is necessary for TERMINAL.I11.

There are two additional files associated with the RS232 terminal software. "rs3" and "rs4" are machine language subroutines that drive the Parallel User Port as an RS232 Port. The BASIC part (TERMINAL.R11) will load one of these automatically; "rs3" for BASIC 2.0 machines and "rs4" for BASIC 4.0. These programs don't produce 'true' RS232, only simulated RS232. Therefore, RS232 modem users will need a special cable to connect their modems to the User Port. For a description of this cable, see Henry Troups' article following this one.

In summary, you'll need to enter these programs:

IEEE Modem Users	RS232 Modem Users
TERMINAL.I11	TERMINAL.R11
term.ieee	term.rs232
	rs3 (for BASIC 2.0)
	rs4 (for BASIC 4.0)

Don't forget, those mnemonics inside square brackets should be replaced by their respective characters and make sure you SAVE everything before trying it! The programs that are mostly DATA statements generate the PRG files that are loaded by the BASIC programs. Before the actual run, put REMs in front of the OPEN and PRINT# commands (lines 500 & 540) and perform a test run to see if the checksums will match up.

Program Features

These Terminal programs have several features that make them ideal for communicating with other computer systems as well as the BBS.

On running the TERMINAL. program, the appropriate machine code support file(s) will load, and a menu will be displayed. Press the number of the desired option. Option 1 will always be "Terminal Mode". This engages the modem and gets you started. Now make your call. When you hear the tone, place your handset in the modem (unless you have a direct connect modem) and the carrier light should come on. Usually you have to hit RETURN once or twice to get a response. You're now ready to "tele-compute"!

At any time in Terminal Mode you can use the "HOME" key to display the menu. This does not mean you'll be disconnected; press 1 again for Terminal Mode and continue where you left off.

Control Key

The 'RVS' has been implemented as a 'Control Key'. One difference from an ASCII terminal control key is that it must be released before typing the Control Character.

Dump to Disk

If you wish to dump text to a disk file, select the Open Disk File option. The Terminal program will ask you for a filename which will be OPENed on drive 1 unless otherwise specified. Once back to Terminal Mode, hitting "CURSOR-DOWN" engages the disk log; "CURSOR-UP" halts disk log; and hitting "HOME" closes the files and returns the menu.

Print Disk File

Supply the name of an SEQ file and the contents will be sent to device #4.

Change Operating Modes

Here you can turn Line Feeds on or off, and change the parity of transmission.

Receive/Transmit Programs

As mentioned earlier, to send or receive programs, WordPro files or SEQ files to or from the BBS, you'll need to use one of these options. Of course, with other timesharing systems these functions will have no use since they won't be using Steves' checksum protocal. On the BBS however, the LOAD or SAVE commands will ask you for the filename, access code, etc., and will then display:

Waiting For START Signal!
(or 'A' for ABORT)

The START signal is the sequence of "HOME" followed by the desired option ie. Transmit or Receive. The Terminal program will ask for a filename and the transfer begins. When finished, you'll be returned to the BBS for your next action.

When you try a LOAD or SAVE on the BBS, you'll first be asked for a 'Program Access Code'. This is more or less a reminder that you must have the proper terminal software for up/downloading. The universal Program Access Code for public Bulletin Boards (using Steve Punters' software) is "EEZOO". If you give the code and attempt a transfer without one of these terminal programs, the BBS will abort shortly afterwards.

That's about it! Tele-computing is on the rise in North America. With this terminal program, you'll be able to communicate with any text oriented systems. Picture oriented systems such as Telidon require a totally different type of terminal software, but this also requires hardware with highly advanced colour graphics and sending data to Telidon is even more difficult (and expensive). For now though, systems like The Source and CompuServe are aquiring new users daily! Give them a try too, but with this program and the BBS, you can get terminal program updates FREE for the cost of a phone call!

```
10 REM TERMINAL.I11 FOR IEEE-488 MODEMS
15 IFPEEK(30976)<>76THENLOAD"TERM.IEEE",8
20 POKE53,120:CLR:RE$=" ":SE$=" ":MO%=134:R%=0:C$="0123456789ABCDEF"
22 RP=5:RS=0:WP=5:WS=0:OPEN5,RP,RS:OPEN6,WP,WS
23 POKE32767,RP:POKE32766,RS+96:POKE32765,WP:POKE32764,WS+96
25 ML=30976:POKE59468,14:POKE32761,0
30 OPEN1,8,15:POKE556,0:POKE552,0:POKE553,1
40 DN$="[DN DN]":GOTO80
70 GET#5,A$,A$:A=PEEK(59426):SYSML+0:CLOSE11:POKE32761,0
80 PRINT"[CLR]"X$:PRINT"[DN]FUNCTION:[DN]"
120 PRINT"1 - TERMINAL MODE"
130 PRINT"2 - RECEIVE A PROGRAM"
140 PRINT"3 - TRANSMIT A PROGRAM"
145 PRINT"4 - OPEN DISK FILE"
146 PRINT"5 - PRINT DISK FILE"
147 PRINT"6 - CHANGE OPERATING MODES"
160 GETA$:IFA$=""THEN160
170 ONVAL(A$)GOTO70,5000,6000,1000,2000,3000
180 GOTO160
1000 CLOSE11:PRINT"[DN]NAME OF DISK FILE":PRINT"DEFAULT IS DRIVE 0?
1010 PRINT">";:GOSUB8000:IFB$=""THEN80
1020 IFMID$(B$,2,1)<>":"THENB$="1:"+B$
1030 OPEN11,8,11,"@"+B$+",S,W":GOSUB9000:IFESTHENPRINTES$:GOTO1000
1040 POKE32761,1:GOTO80
2000 PRINT"[DN]NAME OF FILE?":PRINT">";:GOSUB8000:IFB$=""THEN80
2010 CLOSE11:OPEN11,8,11,B$:GOSUB9000:IFESTHENPRINTES$:GOTO2000
2020 PRINT"[DN]ASCII OR CBM TYPE OUTPUT?":PRINT">";:POKE555,0
2030 GETA$:IFA$=""THEN2030
2040 IFA$=CHR$(13)THEN80
2050 IFA$="A"THENPOKE555,0:GOTO2070
2060 IFA$<>"C"THEN2030
2070 PRINTCHR$(ASC(A$)OR128)"[DN]":SYSML+3:CLOSE11:GOTO80
3000 PRINT"[CLR]OPERATING MODES"
3010 PRINT"---------------[DN]"
3020 PRINT"1) AUTO LINE FEED:[DN]"
3030 PRINT"    OFF":PRINT"    ON[DN]"
3040 PRINT"2) PARITY:[DN]"
3050 PRINT"    MARK":PRINT"    EVEN":PRINT"    ODD[DN]"
3060 PRINT"3) EXIT                                    "[CLR]"  =  clear screen
3070 GOSUB3500                                        "[HOME]" =  cursor home
3090 GETA$:IFA$=""THEN3090                            "[UP]"   =  cursor up
3100 ONVAL(A$)GOTO3200,3300,80                        "[DN]"   =  cursor down
3110 GOTO3090                                         "[CL]"   =  cursor left
3200 GOSUB3510:A=PEEK(553):A=A+1:IFA=2THENA=0         "[CR]"   =  cursor right
3210 POKE553,A:GOTO3070                               "[RVS]"  =  reverse mode on
3300 GOSUB3510:A=PEEK(552):A=A+1:IFA=3THENA=0         "[OFF]"  =  reverse mode off
3310 POKE552,A:GOTO3070                               "[ ' ]"  =  1 space
3500 A$="[RVS ' OFF]":GOTO3520                        "[15CR]" =  15 cursor rights
3510 A$="['"
3520 PRINT"[HOME DN DN DN DN DN ' ' ']";LEFT$(DN$,PEEK(553));A$
3530 PRINT"[HOME 10DN ' ' ']";LEFT$(DN$,PEEK(552));A$:RETURN
5000 REM RECEIVE A PROGRAM
5010 PRINT"[DN]NAME OF FILE?"
5020 PRINT"DEFAULT DRIVE IS 0":PRINT">";:GOSUB8000:S$=B$
5025 IFS$=""THENPRINT#6,"A";:GOTO70
5030 IFMID$(S$,2,1)<>":"THENS$="0:"+S$
5040 GOSUB5800:S$="@"+S$+T$+",W"
5050 CLOSE2:OPEN2,8,2,S$:GOSUB9000:IFESTHENPRINTES$:CLOSE2:GOTO5010
5060 PRINT#6,"TTTTTTTTT";:GOTO5070
5065 GET#5,A$:IFST=0THEN5065
5070 SYSML+18:IFSTTHEN5140
5080 GET#5,A$:IFST=0THEN5080
5090 S1=PEEK(ML-2):S2=PEEK(ML-1):SYSML+12
5120 IFS1<>PEEK(ML-2)ORS2<>PEEK(ML-1)THEN5150
5130 SYSML+15:PRINT#6,"[15CR]";:PRINT"-";:GOTO5070
5140 CLOSE2:PRINT#6,"SSSSSSSSS";:PRINT:GOTO70
5150 PRINT#6,"[15DN]";:PRINT":";:GOTO5070
5500 PRINT"[DN]TYPE OF FILE:"
5510 PRINT"[DN](P)ROGRAM, (W)ORDPRO, OR (S)EQ?":PRINT">";
5520 GETB$:IFB$=""THEN5520
5530 TY$=B$:FL=0
5540 IFB$="P"THENT$=",P":POKE557,0:PRINT"PROGRAM":RETURN
5550 IFB$="S"THENT$=",S":POKE557,0:PRINT"SEQ":RETURN
5560 IFB$="W"THENT$=",P":POKE557,1:PRINT"WORDPRO":RETURN
5570 IFB$=CHR$(13)THENFL=1:RETURN
5580 GOTO5520
5800 PRINT#6,"UUUUUUUUU";
5810 GET#5,A$:IFST=2THEN5810
5820 IFA$="P"THENT$=",P":POKE557,0:A$="PROGRAM":GOTO5860
5830 IFA$="S"THENT$=",S":POKE557,0:A$="SEQ":GOTO5860
5840 IFA$="W"THENT$=",P":POKE557,1:A$="WORDPRO":GOTO5860
5850 GOTO5810
5860 PRINT"[DN]FILE TYPE: "A$"[DN]"
5870 GET#5,A$:IFST=0THEN5870
5880 RETURN
6000 REM SEND AN SEQ FILE TO BULLETIN BOARD
6010 PRINT"[DN]NAME OF FILE TO SEND?":PRINT">";:GOSUB8000:S$=B$
6015 IFS$=""THENPRINT#6,"A";:GOTO70
6017 GOSUB5500:IFFLTHENS$="":GOTO6015
6020 CLOSE2:OPEN2,8,2,S$+T$:GOSUB9000:IFESTHENPRINTES$:CLOSE2:GOTO6010
6030 FORX=1TO10:PRINT#6,TY$;:NEXTX:PRINT
6035 GET#5,A$:IFST=2ORA$<>"U"THEN6035
6040 SYSML+9:CK=ST
6050 GET#5,A$:IFST=0THEN6050
6055 FORX=1TO200:NEXT:REM DELAY LOOP
6057 FORT=0TO255:PRINT#6,CHR$(PEEK(ML-256+T));:NEXTT
6058 PRINT#6,"ZZZZZZZZZZZZZZZ";
6060 GET#5,A$:IFST=2THEN6060
6070 IFA$="[DN]"THENPRINT":";:GOTO6050
6080 IFA$<>"[CR]"THEN6060
6090 IFCK=0THENPRINT"-";:GOTO6040
6100 GET#5,A$:IFST=0THEN6100
6110 CLOSE2:GOTO70
7000 SYSML+21:A$=CHR$(PEEK(634)):RETURN
8000 PRINT"[RVS ' OFF CL]";:B$=""
8010 GETA$:IFA$=""THEN8010
8020 IFA$=CHR$(20)THEN8050
8030 IFA$=CHR$(13)THENPRINT" ":RETURN
8040 B$=B$+A$:PRINT"[' CL]"A$"[RVS ' OFF CL]";:GOTO8010
8050 IFLEN(B$)=0THEN8010
8060 B$=LEFT$(B$,LEN(B$)-1):PRINTA$;:GOTO8010
9000 REM GET ERROR CHANNEL
9010 INPUT#1,E1$,E2$,E3$,E4$
9020 ES$=E1$+","+E2$+","+E3$+","+E4$
9030 ES=VAL(E1$):RETURN
```

```
400 REM 'TERM.IEEE' PRG FILE GENERATOR
410 REM MACHINE LANGUAGE SUBROUTINES FOR TERMINAL.III
420 REM
500 OPEN 8,8,8,"0:TERM.IEEE,P,W"  :REM OPEN PRG FILE
510 CH=0                          :REM RESET CHECKSUM
520 FOR J=1 TO 1722
530 READ X : CH=CH+X
540 PRINT#8,CHR$(X);              :REM OUTPUT BYTE
550 NEXT
560 CLOSE 8                       :REM CLOSE PRG FILE
570 PRINT CH                      :REM PRINT CHECKSUM
580 REM *** CHECKSUM SHOULD EQUAL 193130 ***
590 END
1000 DATA   0, 121                :REM START ADDRESS
1008 DATA  76, 221, 121,  76, 141, 126,  76,  13
1016 DATA 125,  76,  13, 125,  76, 108, 125,  76
1024 DATA 165, 125,  76, 219, 125,  76,   7, 126
1032 DATA  76,  97, 126,  76,   0,   0,  76,   0
1040 DATA   0,  48,  48,  48,  48,  49,  50,  51
1048 DATA  52,  53,  54,  55,  56,  57,  65,  66
1056 DATA  67,  68,  69,  70,   0,   0,   0,  18
1064 DATA 193, 146,  18, 194, 146,  18, 195, 146
1072 DATA  18, 196, 146,  18, 197, 146,  18, 198
1080 DATA 146,   0,   0,   7,   0,   0,  20,  18
1088 DATA 201, 146,   0,   0,  10,  18, 203, 146
1096 DATA  18, 204, 146,   0,   0,  13,  18, 206
1104 DATA 146,  18, 207, 146,  18, 208, 146,  18
1112 DATA 209, 146,  18, 210, 146,  18, 211, 146
1120 DATA  18, 212, 146,  18, 213, 146,  18, 214
1128 DATA 146,  18, 215, 146,  18, 216, 146,  18
1136 DATA 217, 146,  18, 218, 146,  32, 114, 123
1144 DATA 208,   2,  73, 128,  96, 141,  76,   2
1152 DATA 152,  72, 138,  72, 169,   0, 133, 150
1160 DATA 173, 253, 127, 133, 212, 173, 252, 127
1168 DATA 133, 211,  32, 154, 123, 165, 211,  32
1176 DATA   8, 124, 173,  76,   2,  32,  54, 124
1184 DATA  32,  85, 124, 104, 170, 104, 168,  96
1192 DATA 152,  72, 138,  72, 169,   0, 133, 150
1200 DATA 162,   5,  32, 198, 255,  32, 228, 255
1208 DATA  41, 127, 141,  76,   2, 165, 150, 141
1216 DATA  73,   2,  32, 204, 255, 104, 170, 104
1224 DATA 168, 173,  76,   2,  96, 173, 193, 255
1232 DATA  24, 105,   3, 141,  28, 121, 173, 194
1240 DATA 255, 105,   0, 141,  29, 121, 173, 196
1248 DATA 255,  24, 105,   5, 141,  31, 121, 173
1256 DATA 197, 255, 105,   0, 141,  32, 121,  32
1264 DATA 204, 255, 169,  13,  32, 210, 255, 169
1272 DATA  18,  32, 210, 255, 169,  32,  32, 210
1280 DATA 255, 169, 146,  32, 210, 255, 169, 157
1288 DATA  32, 210, 255, 169,   0, 141,  68,   2
1296 DATA 141,  67,   2, 169,   0, 141, 250, 127
1304 DATA 141,  47,   2, 169,   1, 141,  44,   2
1312 DATA 173, 250, 127, 240,   3, 206, 250, 127
1320 DATA 173,  35, 232,  48,   3,  76, 228, 122
1328 DATA  32, 184, 121, 173,  34, 232, 173,  73
1336 DATA   2, 240,   3,  76,  64, 122, 173,  76
1344 DATA   2, 201,  97, 144,   7, 201, 123, 176
1352 DATA   3,  24, 105,  96,  32, 133, 121, 201
1360 DATA  27, 176,  26, 141,  66,   2,  24, 109
1368 DATA  66,   2, 109,  66,   2, 170, 189,  52
1376 DATA 121,  32, 104, 126, 189,  53, 121,  32
1384 DATA 104, 126, 189,  54, 121,  72,  41, 127
1392 DATA 201,  13, 208,  25,  32, 101, 127, 169
1400 DATA  13,  32, 104, 126, 173,  41,   2, 208
1408 DATA   5, 169, 145,  32, 210, 255,  32,  85
1416 DATA 127, 104,  76, 223, 122, 104,  72, 201
1424 DATA  10, 208,  16, 173,  41,   2, 208, 241
1432 DATA  32, 101, 127, 169,  17,  32, 210, 255
1440 DATA  76, 150, 122, 104, 141,  76,   2,  32
1448 DATA 104, 126, 201,  34, 208,  10, 169,  20
1456 DATA  32, 210, 255, 169,  34,  32, 210, 255
1464 DATA 173,  76,   2, 169,  18,  32, 210, 255
1472 DATA 169,  32,  32, 210, 255, 169, 146,  32
1480 DATA 210, 255, 169, 157,  32, 210, 255, 169
1488 DATA   0, 141, 250, 127, 173, 250, 127, 208
1496 DATA 118,  32, 228, 255, 240, 113, 162,   4
1504 DATA 142, 250, 127, 174,  46,   2, 240,  24
1512 DATA 141,  76,   2,  32, 114, 123, 208,   8
1520 DATA  41, 127,  56, 233,  64, 141,  76,   2
1528 DATA 169,   0, 141,  46,   2, 173,  76,   2
1536 DATA 201,  18, 208,   8, 169,   1, 141,  46
1544 DATA   2,  76,  48, 122, 201,  17, 208,  13
1552 DATA 173, 249, 127, 240,   8, 169,   1, 141
1560 DATA  47,   2,  76,  48, 122, 201, 145, 208
1568 DATA   8, 169,   0, 141,  47,   2,  76,  48
1576 DATA 122,  32, 133, 121, 201, 193, 144,   7
1584 DATA 201, 219, 176,   3,  56, 233,  96, 201
1592 DATA  19, 240,  23, 201,  20, 208,   4, 169
1600 DATA   8, 208,   6, 201, 131, 208,   2, 169
1608 DATA  16,  32, 112, 127,  32, 141, 121,  76
1616 DATA  48, 122,  32,  81, 124,  32,  85, 124
1624 DATA 169,   3, 133, 212, 169,   0, 141,  44
1632 DATA   2,  96, 133,   0,  41, 127, 201,  65
1640 DATA  48,  11, 201,  91,  16,   7, 169,   0
1648 DATA   8, 165,   0,  40,  96, 169,   1, 208
1656 DATA 247, 133,   0, 201,  48,  48, 246, 201
1664 DATA  58,  16, 242,  76, 126, 123, 169,  64
1672 DATA 208,   2, 169,  32,  72, 173,  64, 232
1680 DATA   9,   2, 141,  64, 232, 169,  60, 141
1688 DATA  33, 232,  36, 160, 240,  17, 169,  52
1696 DATA 141,  17, 232,  32, 206, 123, 169,   0
1704 DATA 133, 160, 169,  60, 141,  17, 232, 104
1712 DATA   5, 212, 133, 165, 173,  64, 232,  16
1720 DATA 251,  41, 251, 141,  64, 232, 169,  60
1728 DATA 141,  35, 232, 173,  64, 232,  41,  65
1736 DATA 201,  65, 240,  65, 165, 165,  73, 255
1744 DATA 141,  34, 232,  44,  64, 232,  80, 251
1752 DATA 169,  52, 141,  35, 232, 169, 255, 141
1760 DATA  69, 232, 173,  64, 232,  44,  77, 232
1768 DATA 112,  28,  74, 144, 245, 169,  60, 141
1776 DATA  35, 232, 169, 255, 141,  34, 232,  96
1784 DATA 133, 165,  32, 206, 123, 173,  64, 232
1792 DATA   9,   4, 141,  64, 232,  96, 169,   1
1800 DATA  32, 217, 124, 208, 224, 169, 128,  48
```

```
1808 DATA 247, 169,   2,  32, 217, 124, 173,  64
1816 DATA 232,  41, 253, 141,  64, 232, 169,  52
1824 DATA 141,  33, 232, 169,  13,  96,  36, 160
1832 DATA  48,   4, 198, 160, 208,   5,  72,  32
1840 DATA 206, 123, 104, 133, 165,  96, 133, 165
1848 DATA  32, 206, 123,  32,  38, 124,  76,  13
1856 DATA 124, 169,  95, 208,   2, 169,  63,  32
1864 DATA 156, 123, 208, 177, 169,  52, 141,  33
1872 DATA 232, 173,  64, 232,   9,   2, 141,  64
1880 DATA 232, 169, 255, 141,  69, 232,  44,  77
1888 DATA 232, 112, 174,  44,  64, 232,  48, 246
1896 DATA 173,  64, 232,  41, 253, 141,  64, 232
1904 DATA  44,  16, 232, 112,   5, 169,  64,  32
1912 DATA 217, 124, 173,  32, 232,  73, 255,  72
1920 DATA 169,  60, 141,  33, 232,  44,  64, 232
1928 DATA  16, 251, 169,  52, 141,  33, 232, 104
1936 DATA  96, 165, 211,  48, 251, 164, 209, 240
1944 DATA 247,  32, 154, 123, 165, 211,   9, 240
1952 DATA  32,   8, 124, 165, 209, 240,  12, 160
1960 DATA   0, 177, 218,  32,  54, 124, 200, 196
1968 DATA 209, 208, 246,  76,  85, 124,  36, 211
1976 DATA  48,  19,  32, 154, 123, 165, 211,  41
1984 DATA 239,   9, 224,  32,   8, 124,  76,  85
1992 DATA 124,   5, 150, 133, 150,  96, 173,  75
2000 DATA   2,  32, 247, 124, 141,  75,   2, 173
2008 DATA  74,   2,  32, 247, 124,  10,  10,  10
2016 DATA  10,  41, 240,  13,  75,   2,  96,  56
2024 DATA 233,  48, 201,  10,  48,   3,  56, 233
2032 DATA   7,  96,  24, 105,  48, 201,  58,  48
2040 DATA   3,  24, 105,   7,  96, 120, 160,   0
2048 DATA 169, 119, 153,   0, 120, 200, 192, 254
2056 DATA 208, 248, 160,   0, 169,   8, 133, 212
2064 DATA 169,  98, 133, 211, 169,   0, 133, 150
2072 DATA  32, 150, 123, 165, 211,  32,   8, 124
2080 DATA  32,  92, 124, 141,  76,   2, 174,  45
2088 DATA   2, 208,  23,  41,  15,  32,   2, 125
2096 DATA 153,   0, 120, 173,  76,   2,  41, 240
2104 DATA  74,  74,  74,  74,  41,  15,  32,   2
2112 DATA 125, 200, 153,   0, 120, 165, 150, 141
2120 DATA  73,   2, 173,  73,   2,  41,   2, 208
2128 DATA 187, 173,  73,   2, 208,   5, 200, 192
2136 DATA 254, 208, 197,  88, 160,   0, 169,   0
2144 DATA 141, 254, 120, 141, 255, 120,  24, 173
2152 DATA 254, 120, 121,   0, 120, 141, 254, 120
2160 DATA 144,   3, 238, 255, 120, 200, 192, 254
2168 DATA 208, 236, 173,  73,   2, 133, 150, 173
2176 DATA 254, 120,  16,   3, 238, 255, 120,  41
2184 DATA 127, 141, 254, 120, 173, 255, 120,  41
2192 DATA 127, 141, 255, 120,  96, 169,   8, 133
2200 DATA 212, 169,  98, 133, 211,  32, 154, 123
2208 DATA 165, 211,  32,   8, 124, 160,   0, 185
2216 DATA   0, 120, 201, 119, 240,  26, 174,  45
2224 DATA   2, 208,  13, 141,  75,   2, 200, 185
2232 DATA   0, 120, 141,  74,   2,  32, 222, 124
2240 DATA  32,  54, 124, 200, 192, 254, 208, 223
2248 DATA  76,  85, 124, 160,   0, 169,   0, 141
2256 DATA  72,   2, 238,  72,   2, 173,  72,   2
2264 DATA 201,  38, 240,  26,  32, 184, 121, 165
```

```
2272 DATA 150, 201,   2, 240, 237, 173,  76,   2
2280 DATA  41, 127, 153,   0, 120, 200, 192,   0
2288 DATA 208, 219, 169,   0, 133, 150,  96, 162
2296 DATA   9, 173,  44,   2, 240,   2, 162,   1
2304 DATA 142, 123,   2, 160,   2, 177,  42, 208
2312 DATA  19,  32,  86,  62, 160,   2, 177,  42
2320 DATA 208,  10, 206, 123,   2, 208, 236, 169
2328 DATA   2, 133, 150,  96, 133,   2, 200, 177
2336 DATA  42, 133,   0, 200, 177,  42, 133,   1
2344 DATA 160,   0, 177,   0, 141, 122,   2, 198
2352 DATA   2, 165,   2, 160,   2, 145,  42, 165
2360 DATA   0,  24, 105,   1, 133,   0, 144,   2
2368 DATA 198,   1, 200, 165,   0, 145,  42, 200
2376 DATA 165,   1, 145,  42, 169,   0, 133, 150
2384 DATA  96, 160,   2, 169,   0, 145,  42,  96
2392 DATA  32, 210, 255, 141,  76,   2, 173,  47
2400 DATA   2, 240,  25, 169,   8, 133, 212, 169
2408 DATA 107, 133, 211,  32, 154, 123, 165, 211
2416 DATA  32,   8, 124, 173,  76,   2,  32,  54
2424 DATA 124,  32,  85, 124,  96, 169,   0, 141
2432 DATA  42,   2, 169,   8, 133, 212, 169, 107
2440 DATA 133, 211,  32, 150, 123, 165, 211,  32
2448 DATA   8, 124, 169,   0, 133, 150,  32,  92
2456 DATA 124, 240, 247, 166, 150, 142,  73,   2
2464 DATA 208,  30, 201,  10, 240, 236, 201,  13
2472 DATA 240,  22, 201,  20, 208,   6, 206,  42
2480 DATA   2,  76, 162, 126, 172,  42,   2, 153
2488 DATA 122,   2, 238,  42,   2,  76, 162, 126
2496 DATA  32,  81, 124, 169,   4, 133, 212, 169
2504 DATA  96, 133, 211,  32, 154, 123, 165, 211
2512 DATA  32,   8, 124, 173,  42,   2, 240,  25
2520 DATA 160,   0, 185, 122,   2,  32, 210, 255
2528 DATA 174,  43,   2, 208,   3,  32,  42, 127
2536 DATA  32,  54, 124, 200, 204,  42,   2, 208
2544 DATA 233, 169,  13,  32, 210, 255,  32,  54
2552 DATA 124,  32,  85, 124, 173,  73,   2, 208
2560 DATA  42,  32, 228, 255, 240,  17, 201,  83
2568 DATA 208,  13,  32, 228, 255, 240, 251, 201
2576 DATA  65, 240,  24, 201,  83, 208, 243,  76
2584 DATA 141, 126,  32, 133, 121,  72,  41, 128
2592 DATA  74,  74, 141,  76,   2, 104,  41, 127
2600 DATA  13,  76,   2,  96, 164, 198, 177, 196
2608 DATA  72,  41,  64,  10, 141,  70,   2, 104
2616 DATA  41,  63, 201,  31,  16,   3,  24, 105
2624 DATA  64,  13,  70,   2,  96, 169,  18,  32
2632 DATA 210, 255,  32,  60, 127,  32, 210, 255
2640 DATA 169, 146,  76, 104, 127,  32,  60, 127
2648 DATA  32, 210, 255, 169, 157,  76, 210, 255
2656 DATA 141,  70,   2, 173,  40,   2, 240,  57
2664 DATA 152,  72, 160,   8, 173,  70,   2, 106
2672 DATA 144,   3, 238,  42,   2, 136, 208, 247
2680 DATA 173,  40,   2, 201,   1, 208,  14, 160
2688 DATA 128, 173,  42,   2,  41,   1, 208,   2
2696 DATA 160,   0,  76, 168, 127, 160, 128, 173
2704 DATA  42,   2,  41,   1, 240,   2, 160,   0
2712 DATA 152,  13,  70,   2, 141,  70,   2, 104
2720 DATA 168, 173,  70,   2,  96, 170, 170, 170
```

```
 8 REM TERMINAL.R11 FOR RS232 MODEMS
 10 IFPEEK(57345)<>72THENIFPEEK(15763)<>32THENLOAD"RS4",8
 12 IFPEEK(57345)=72THENIFPEEK(15763)<>32THENLOAD"RS3",8
 15 IFPEEK(30976)<>76THENLOAD"TERM.RS232",8
 20 POKE53,50:CLR:RE$=" ":SE$=" ":MO%=134:R%=0:C$="0123456789ABCDEF"
 22 RP=5:RS=0:WP=5:WS=0:OPEN5,RP,RS:OPEN6,WP,WS
 23 POKE32767,RP:POKE32766,RS+96:POKE32765,WP:POKE32764,WS+96
 25 ML=30976:SYS15641:POKE59468,14:POKE32761,0
 30 OPEN1,8,15:POKE556,0:POKE552,0:POKE553,1
 40 DN$="[DN DN]":GOTO80                          "[CLR]"  =  clear screen
 70 SE$="A"+"":SYSML+0:CLOSE11:POKE32761,0        "[HOME]" =  cursor home
 80 PRINT"[CLR]"X$:PRINT"[DN]FUNCTION:[DN]"       "[UP]"   =  cursor up
 120 PRINT"1 - TERMINAL MODE"                     "[DN]"   =  cursor down
 130 PRINT"2 - RECEIVE A PROGRAM"                 "[CL]"   =  cursor left
 140 PRINT"3 - TRANSMIT A PROGRAM"                "[CR]"   =  cursor right
 145 PRINT"4 - OPEN DISK FILE"                    "[RVS]"  =  reverse mode on
 146 PRINT"5 - PRINT DISK FILE"                   "[OFF]"  =  reverse mode off
 147 PRINT"6 - CHANGE OPERATING MODES"            "[ ' ]"  =  1 space
 160 GETA$:IFA$=""THEN160                         "[15CR]" =  15 cursor rights
 170 ONVAL(A$)GOTO70,5000,6000,1000,2000,3000
 180 GOTO160
1000 CLOSE11:PRINT"[DN]NAME OF DISK FILE":PRINT"DEFAULT IS DRIVE 0?
1010 PRINT">";:GOSUB8000:IFB$=""THEN80
1020 IFMID$(B$,2,1)<>":"THENB$="1:"+B$
1030 OPEN11,8,11,"@"+B$+",S,W":GOSUB9000:IFESTHENPRINTES$:GOTO1000
1040 POKE32761,1:GOTO80
2000 PRINT"[DN]NAME OF FILE?":PRINT">";:GOSUB8000:IFB$=""THEN80
2010 CLOSE11:OPEN11,8,11,B$:GOSUB9000:IFESTHENPRINTES$:GOTO2000
2020 PRINT"[DN]ASCII OR CBM TYPE OUTPUT?":PRINT">";:POKE555,0
2030 GETA$:IFA$=""THEN2030
2040 IFA$=CHR$(13)THEN80
2050 IFA$="A"THENPOKE555,0:GOTO2070
2060 IFA$<>"C"THEN2030
2070 PRINTCHR$(ASC(A$)OR128)"[DN]":SYSML+3:CLOSE11:GOTO80
3000 PRINT"[CLR]OPERATING MODES"
3010 PRINT"---------------[DN]"
3020 PRINT"1) AUTO LINE FEED:[DN]"
3030 PRINT"    OFF":PRINT"    ON[DN]"
3040 PRINT"2) PARITY:[DN]"
3050 PRINT"    MARK":PRINT"    EVEN":PRINT"    ODD[DN]"
3060 PRINT"3) EXIT
3070 GOSUB3500
3090 GETA$:IFA$=""THEN3090
3100 ONVAL(A$)GOTO3200,3300,80
3110 GOTO3090
3200 GOSUB3510:A=PEEK(553):A=A+1:IFA=2THENA=0
3210 POKE553,A:GOTO3070
3300 GOSUB3510:A=PEEK(552):A=A+1:IFA=3THENA=0
3310 POKE552,A:GOTO3070
3500 A$="[RVS ' OFF]":GOTO3520
3510 A$="[']"
3520 PRINT"[HOME DN DN DN DN DN ' ' ']";LEFT$(DN$,PEEK(553));A$
3530 PRINT"[HOME 10DN ' ' ']";LEFT$(DN$,PEEK(552));A$:RETURN
5000 REM RECEIVE A PROGRAM
5010 PRINT"[DN]NAME OF FILE?"
5020 PRINT"DEFAULT DRIVE IS #0":PRINT">";:GOSUB8000:S$=B$
5025 IFS$=""THENSE$="A":SYS15763:GOTO70
5030 IFMID$(S$,2,1)<>":"THENS$="0:"+S$
5040 GOSUB5800:S$="@"+S$+T$+",W"
5050 CLOSE2:OPEN2,8,2,S$:GOSUB9000:IFESTHENPRINTES$:CLOSE2:GOTO5010
5060 SE$="TTTTTTTTT":SYS15763:GOTO5070
5065 GOSUB7000:IFST=0THEN5065
5070 SYSML+18:IFSTTHEN5140
5080 GOSUB7000:IFST=0THEN5080
5090 S1=PEEK(ML-2):S2=PEEK(ML-1):SYSML+12
5120 IFS1<>PEEK(ML-2)ORS2<>PEEK(ML-1)THEN5150
5130 SYSML+15:SE$="[15CR]":SYS15763:PRINT"-";:GOTO5070
5140 CLOSE2:SE$="SSSSSSSSSS":SYS15763:PRINT:GOTO70
5150 SE$="[15DN]":SYS15763:PRINT";";:GOTO5070
5500 PRINT"[DN]TYPE OF FILE:"
5510 PRINT"[DN](P)ROGRAM, (W)ORDPRO, OR (S)EQ?":PRINT">";
5520 GETB$:IFB$=""THEN5520
5530 TY$=B$:FL=0
5540 IFB$="P"THENT$=",P":POKE557,0:PRINT"PROGRAM":RETURN
5550 IFB$="S"THENT$=",S":POKE557,0:PRINT"SEQ":RETURN
5560 IFB$="W"THENT$=",P":POKE557,1:PRINT"WORDPRO":RETURN
5570 IFB$=CHR$(13)THENFL=1:RETURN
5580 GOTO5520
5800 SE$="UUUUUUUUUU":SYS15763
5810 GOSUB7000:IFST=2THEN5810
5820 IFA$="P"THENT$=",P":POKE557,0:A$="PROGRAM":GOTO5860
5830 IFA$="S"THENT$=",S":POKE557,0:A$="SEQ":GOTO5860
5840 IFA$="W"THENT$=",P":POKE557,1:A$="WORDPRO":GOTO5860
5850 GOTO5810
5860 PRINT"[DN]FILE TYPE: "A$"[DN]"
5870 GOSUB7000:IFST=0THEN5870
5880 RETURN
6000 REM SEND AN SEQ FILE TO BULLETIN BOARD
6010 PRINT"[DN]NAME OF FILE TO SEND?":PRINT">";:GOSUB8000:S$=B$
6015 IFS$=""THENSE$="A":SYS15763:GOTO70
6017 GOSUB5500:IFFLTHENS$="":GOTO6015
6020 CLOSE2:OPEN2,8,2,S$+T$:GOSUB9000:IFESTHENPRINTES$:CLOSE2:GOTO6010
6030 FORX=1TO10:SE$=TY$:SYS15763:NEXTX:PRINT
6035 GOSUB7000:IFST=2ORA$<>"U"THEN6035
6040 SYSML+9:CK=ST
6050 GOSUB7000:IFST=0THEN6050
6055 FORX=1TO200:NEXT:REM DELAY LOOP
6057 FORT=0TO255:SE$=CHR$(PEEK(ML-256+T)):SYS15763:NEXTT
6058 SE$="ZZZZZZZZZZZZZZZ":SYS15763
6060 GOSUB7000:IFST=2THEN6060
6070 IFA$="[DN]"THENPRINT";";:GOTO6050
6080 IFA$<>"[CR]"THEN6060
6090 IFCK=0THENPRINT"-";:GOTO6040
6100 GOSUB7000:IFST=0THEN6100
6110 CLOSE2:GOTO70
7000 SYSML+21:A$=CHR$(PEEK(634)):RETURN
8000 PRINT"[RVS ' OFF CL]";:B$=""
8010 GETA$:IFA$=""THEN8010
8020 IFA$=CHR$(20)THEN8050
8030 IFA$=CHR$(13)THENPRINT" ":RETURN
8040 B$=B$+A$:PRINT"[' CL]"A$"[RVS ' OFF CL]";:GOTO8010
8050 IFLEN(B$)=0THEN8010
8060 B$=LEFT$(B$,LEN(B$)-1):PRINTA$;:GOTO8010
9000 REM GET ERROR CHANNEL
9010 INPUT#1,E1$,E2$,E3$,E4$
9020 ES$=E1$+","+E2$+","+E3$+","+E4$
9030 ES=VAL(E1$):RETURN
```

```
400 REM 'TERM.RS232' PRG FILE GENERATOR
410 REM MACHINE LANGUAGE SUBROUTINES FOR TERMINAL.R11
420 REM
500 OPEN 8,8,8,"0:TERM.RS232,P,W" :REM OPEN PRG FILE
510 CH=0                          :REM RESET CHECKSUM
520 FOR J=1 TO 1698
530 READ X : CH=CH+X
540 PRINT#8,CHR$(X);              :REM OUTPUT BYTE
550 NEXT
560 CLOSE 8                       :REM CLOSE PRG FILE
570 PRINT CH                      :REM PRINT CHECKSUM
580 REM *** CHECKSUM SHOULD EQUAL 190616 ***
590 END
1000 DATA   0, 121                :REM START ADDRESS
1008 DATA  76, 141, 121,  76, 120, 126,  76, 178
1016 DATA 124,  76, 178, 124,  76,  17, 125,  76
1024 DATA  74, 125,  76, 128, 125,  76, 172, 125
1032 DATA  76,   6, 126,  76,   0,   0,  76,   0
1040 DATA   0,  48,  48,  48,  48,  49,  50,  51
1048 DATA  52,  53,  54,  55,  56,  57,  65,  66
1056 DATA  67,  68,  69,  70,   0,   0,   0,  18
1064 DATA 193, 146,  18, 194, 146,  18, 195, 146
1072 DATA  18, 196, 146,  18, 197, 146,  18, 198
1080 DATA 146,   0,   0,   7,   0,   0,  20,  18
1088 DATA 201, 146,   0,   0,  10,  18, 203, 146
1096 DATA  18, 204, 146,   0,   0,  13,  18, 206
1104 DATA 146,  18, 207, 146,  18, 208, 146,  18
1112 DATA 209, 146,  18, 210, 146,  18, 211, 146
1120 DATA  18, 212, 146,  18, 213, 146,  18, 214
1128 DATA 146,  18, 215, 146,  18, 216, 146,  18
1136 DATA 217, 146,  18, 218, 146,  32,  23, 123
1144 DATA 208,   2,  73, 128,  96, 173, 193, 255
1152 DATA  24, 105,   3, 141,  28, 121, 173, 194
1160 DATA 255, 105,   0, 141,  29, 121, 173, 196
1168 DATA 255,  24, 105,   5, 141,  31, 121, 173
1176 DATA 197, 255, 105,   0, 141,  32, 121,  32
1184 DATA 204, 255, 169,  13,  32, 210, 255, 169
1192 DATA  18,  32, 210, 255, 169,  32,  32, 210
1200 DATA 255, 169, 146,  32, 210, 255, 169, 157
1208 DATA  32, 210, 255, 169,   0, 141,  68,   2
1216 DATA 141,  67,   2, 169,   0, 141, 250, 127
1224 DATA 141,  47,   2, 169,   1, 141,  44,   2
1232 DATA 173, 250, 127, 240,   3, 206, 250, 127
1240 DATA  32,  55, 126, 173,  73,   2, 240,   3
1248 DATA  76, 137, 122, 173,  76,   2, 201,  97
1256 DATA 144,   7, 201, 123, 176,   3,  24, 105
1264 DATA  96,  32, 133, 121, 201,  27, 176,  26
1272 DATA 141,  66,   2,  24, 109,  66,   2, 109
1280 DATA  66,   2, 170, 189,  52, 121,  32,  83
1288 DATA 126, 189,  53, 121,  32,  83, 126, 189
1296 DATA  54, 121,  72,  41, 127, 201,  13, 208
1304 DATA  25,  32,  80, 127, 169,  13,  32,  83
1312 DATA 126, 173,  41,   2, 208,   5, 169, 145
1320 DATA  32, 210, 255,  32,  64, 127, 104,  76
1328 DATA 132, 122, 104,  72, 201,  10, 208,  16
1336 DATA 173,  41,   2, 208, 241,  32,  80, 127
1344 DATA 169,  17,  32, 210, 255,  76,  59, 122
1352 DATA 104, 141,  76,   2,  32,  83, 126, 201
1360 DATA  34, 208,  10, 169,  20,  32, 210, 255
1368 DATA 169,  34,  32, 210, 255, 173,  76,   2
1376 DATA 169,  18,  32, 210, 255, 169,  32,  32
1384 DATA 210, 255, 169, 146,  32, 210, 255, 169
1392 DATA 157,  32, 210, 255, 169,   0, 141, 250
1400 DATA 127, 173, 250, 127, 208, 118,  32, 228
1408 DATA 255, 240, 113, 162,   4, 142, 250, 127
1416 DATA 174,  46,   2, 240,  24, 141,  76,   2
1424 DATA  32,  23, 123, 208,   8,  41, 127,  56
1432 DATA 233,  64, 141,  76,   2, 169,   0, 141
1440 DATA  46,   2, 173,  76,   2, 201,  18, 208
1448 DATA   8, 169,   1, 141,  46,   2,  76, 224
1456 DATA 121, 201,  17, 208,  13, 173, 249, 127
1464 DATA 240,   8, 169,   1, 141,  47,   2,  76
1472 DATA 224, 121, 201, 145, 208,   8, 169,   0
1480 DATA 141,  47,   2,  76, 224, 121,  32, 133
1488 DATA 121, 201, 193, 144,   7, 201, 219, 176
1496 DATA   3,  56, 233,  96, 201,  19, 240,  23
1504 DATA 201,  20, 208,   4, 169,   8, 208,   6
1512 DATA 201, 131, 208,   2, 169,  16,  32,  91
1520 DATA 127,  32,  13, 126,  76, 224, 121,  32
1528 DATA 246, 123,  32, 250, 123, 169,   3, 133
1536 DATA 212, 169,   0, 141,  44,   2,  96, 133
1544 DATA   0,  41, 127, 201,  65,  48,  11, 201
1552 DATA  91,  16,   7, 169,   0,   8, 165,   0
1560 DATA  40,  96, 169,   1, 208, 247, 133,   0
1568 DATA 201,  48,  48, 246, 201,  58,  16, 242
1576 DATA  76,  35, 123, 169,  64, 208,   2, 169
1584 DATA  32,  72, 173,  64, 232,   9,   2, 141
1592 DATA  64, 232, 169,  60, 141,  33, 232,  36
1600 DATA 160, 240,  17, 169,  52, 141,  17, 232
1608 DATA  32, 115, 123, 169,   0, 133, 160, 169
1616 DATA  60, 141,  17, 232, 104,   5, 212, 133
1624 DATA 165, 173,  64, 232,  16, 251,  41, 251
1632 DATA 141,  64, 232, 169,  60, 141,  35, 232
1640 DATA 173,  64, 232,  41,  65, 201,  65, 240
1648 DATA  65, 165, 165,  73, 255, 141,  34, 232
1656 DATA  44,  64, 232,  80, 251, 169,  52, 141
1664 DATA  35, 232, 169, 255, 141,  69, 232, 173
1672 DATA  64, 232,  44,  77, 232, 112,  28,  74
1680 DATA 144, 245, 169,  60, 141,  35, 232, 169
1688 DATA 255, 141,  34, 232,  96, 133, 165,  32
1696 DATA 115, 123, 173,  64, 232,   9,   4, 141
1704 DATA  64, 232,  96, 169,   1,  32, 126, 124
1712 DATA 208, 224, 169, 128,  48, 247, 169,   2
1720 DATA  32, 126, 124, 173,  64, 232,  41, 253
1728 DATA 141,  64, 232, 169,  52, 141,  33, 232
1736 DATA 169,  13,  96,  36, 160,  48,   4, 198
1744 DATA 160, 208,   5,  72,  32, 115, 123, 104
1752 DATA 133, 165,  96, 133, 165,  32, 115, 123
1760 DATA  32, 203, 123,  76, 178, 123, 169,  95
1768 DATA 208,   2, 169,  63,  32,  65, 123, 208
1776 DATA 177, 169,  52, 141,  33, 232, 173,  64
1784 DATA 232,   9,   2, 141,  64, 232, 169, 255
1792 DATA 141,  69, 232,  44,  77, 232, 112, 174
1800 DATA  44,  64, 232,  48, 246, 173,  64, 232
```

```
1808 DATA  41, 253, 141,  64, 232,  44,  16, 232
1816 DATA 112,   5, 169,  64,  32, 126, 124, 173
1824 DATA  32, 232,  73, 255,  72, 169,  60, 141
1832 DATA  33, 232,  44,  64, 232,  16, 251, 169
1840 DATA  52, 141,  33, 232, 104,  96, 165, 211
1848 DATA  48, 251, 164, 209, 240, 247,  32,  63
1856 DATA 123, 165, 211,   9, 240,  32, 173, 123
1864 DATA 165, 209, 240,  12, 160,   0, 177, 218
1872 DATA  32, 219, 123, 200, 196, 209, 208, 246
1880 DATA  76, 250, 123,  36, 211,  48,  19,  32
1888 DATA  63, 123, 165, 211,  41, 239,   9, 224
1896 DATA  32, 173, 123,  76, 250, 123,   5, 150
1904 DATA 133, 150,  96, 173,  75,   2,  32, 156
1912 DATA 124, 141,  75,   2, 173,  74,   2,  32
1920 DATA 156, 124,  10,  10,  10,  10,  41, 240
1928 DATA  13,  75,   2,  96,  56, 233,  48, 201
1936 DATA  10,  48,   3,  56, 233,   7,  96,  24
1944 DATA 105,  48, 201,  58,  48,   3,  24, 105
1952 DATA   7,  96, 120, 160,   0, 169, 119, 153
1960 DATA   0, 120, 200, 192, 254, 208, 248, 160
1968 DATA   0, 169,   8, 133, 212, 169,  98, 133
1976 DATA 211, 169,   0, 133, 150,  32,  59, 123
1984 DATA 165, 211,  32, 173, 123,  32,   1, 124
1992 DATA 141,  76,   2, 174,  45,   2, 208,  23
2000 DATA  41,  15,  32, 167, 124, 153,   0, 120
2008 DATA 173,  76,   2,  41, 240,  74,  74,  74
2016 DATA  74,  41,  15,  32, 167, 124, 200, 153
2024 DATA   0, 120, 165, 150, 141,  73,   2, 173
2032 DATA  73,   2,  41,   2, 208, 187, 173,  73
2040 DATA   2, 208,   5, 200, 192, 254, 208, 197
2048 DATA  88, 160,   0, 169,   0, 141, 254, 120
2056 DATA 141, 255, 120,  24, 173, 254, 120, 121
2064 DATA   0, 120, 141, 254, 120, 144,   3, 238
2072 DATA 255, 120, 200, 192, 254, 208, 236, 173
2080 DATA  73,   2, 133, 150, 173, 254, 120,  16
2088 DATA   3, 238, 255, 120,  41, 127, 141, 254
2096 DATA 120, 173, 255, 120,  41, 127, 141, 255
2104 DATA 120,  96, 169,   8, 133, 212, 169,  98
2112 DATA 133, 211,  32,  63, 123, 165, 211,  32
2120 DATA 173, 123, 160,   0, 185,   0, 120, 201
2128 DATA 119, 240,  26, 174,  45,   2, 208,  13
2136 DATA 141,  75,   2, 200, 185,   0, 120, 141
2144 DATA  74,   2,  32, 131, 124,  32, 219, 123
2152 DATA 200, 192, 254, 208, 223,  76, 250, 123
2160 DATA 160,   0, 169,   0, 141,  72,   2, 238
2168 DATA  72,   2, 173,  72,   2, 201,  38, 240
2176 DATA  26,  32,  55, 126, 165, 150, 201,   2
2184 DATA 240, 237, 173,  76,   2,  41, 127, 153
2192 DATA   0, 120, 200, 192,   0, 208, 219, 169
2200 DATA   0, 133, 150,  96, 162,   9, 173,  44
2208 DATA   2, 240,   2, 162,   1, 142, 123,   2
2216 DATA 160,   2, 177,  42, 208,  19,  32,  86
2224 DATA  62, 160,   2, 177,  42, 208,  10, 206
2232 DATA 123,   2, 208, 236, 169,   2, 133, 150
2240 DATA  96, 133,   2, 200, 177,  42, 133,   0
2248 DATA 200, 177,  42, 133,   1, 160,   0, 177
2256 DATA   0, 141, 122,   2, 198,   2, 165,   2
```

```
2264 DATA 160,   2, 145,  42, 165,   0,  24, 105
2272 DATA   1, 133,   0, 144,   2, 198,   1, 200
2280 DATA 165,   0, 145,  42, 200, 165,   1, 145
2288 DATA  42, 169,   0, 133, 150,  96, 160,   2
2296 DATA 169,   0, 145,  42,  96, 141,  76,   2
2304 DATA 152,  72, 138,  72, 169,   0, 133, 150
2312 DATA 160,   9, 169,   1, 145,  42, 200, 177
2320 DATA  42, 133,   0, 200, 177,  42, 133,   1
2328 DATA 160,   0, 173,  76,   2, 145,   0,  32
2336 DATA 147,  61, 104, 170, 104, 168,  96, 152
2344 DATA  72, 138,  72,  32, 172, 125, 173, 122
2352 DATA   2,  41, 127, 141,  76,   2, 165, 150
2360 DATA 141,  73,   2, 104, 170, 104, 168, 173
2368 DATA  76,   2,  96,  32, 210, 255, 141,  76
2376 DATA   2, 173,  47,   2, 240,  25, 169,   8
2384 DATA 133, 212, 169, 107, 133, 211,  32,  63
2392 DATA 123, 165, 211,  32, 173, 123, 173,  76
2400 DATA   2,  32, 219, 123,  32, 250, 123,  96
2408 DATA 169,   0, 141,  42,   2, 169,   8, 133
2416 DATA 212, 169, 107, 133, 211,  32,  59, 123
2424 DATA 165, 211,  32, 173, 123, 169,   0, 133
2432 DATA 150,  32,   1, 124, 240, 247, 166, 150
2440 DATA 142,  73,   2, 208,  30, 201,  10, 240
2448 DATA 236, 201,  13, 240,  22, 201,  20, 208
2456 DATA   6, 206,  42,   2,  76, 141, 126, 172
2464 DATA  42,   2, 153, 122,   2, 238,  42,   2
2472 DATA  76, 141, 126,  32, 246, 123, 169,   4
2480 DATA 133, 212, 169,  96, 133, 211,  32,  63
2488 DATA 123, 165, 211,  32, 173, 123, 173,  42
2496 DATA   2, 240,  25, 160,   0, 185, 122,   2
2504 DATA  32, 210, 255, 174,  43,   2, 208,   3
2512 DATA  32,  21, 127,  32, 219, 123, 200, 204
2520 DATA  42,   2, 208, 233, 169,  13,  32, 210
2528 DATA 255,  32, 219, 123,  32, 250, 123, 173
2536 DATA  73,   2, 208,  42,  32, 228, 255, 240
2544 DATA  17, 201,  83, 208,  13,  32, 228, 255
2552 DATA 240, 251, 201,  65, 240,  24, 201,  83
2560 DATA 208, 243,  76, 120, 126,  32, 133, 121
2568 DATA  72,  41, 128,  74,  74, 141,  76,   2
2576 DATA 104,  41, 127,  13,  76,   2,  96, 164
2584 DATA 198, 177, 196,  72,  41,  64,  10, 141
2592 DATA  70,   2, 104,  41,  63, 201,  31,  16
2600 DATA   3,  24, 105,  64,  13,  70,   2,  96
2608 DATA 169,  18,  32, 210, 255,  32,  39, 127
2616 DATA  32, 210, 255, 169, 146,  76,  83, 127
2624 DATA  32,  39, 127,  32, 210, 255, 169, 157
2632 DATA  76, 210, 255, 141,  70,   2, 173,  40
2640 DATA   2, 240,  57, 152,  72, 160,   8, 173
2648 DATA  70,   2, 106, 144,   3, 238,  42,   2
2656 DATA 136, 208, 247, 173,  40,   2, 201,   1
2664 DATA 208,  14, 160, 128, 173,  42,   2,  41
2672 DATA   1, 208,   2, 160,   0,  76, 147, 127
2680 DATA 160, 128, 173,  42,   2,  41,   1, 240
2688 DATA   2, 160,   0, 152,  13,  70,   2, 141
2696 DATA  70,   2, 104, 168, 173,  70,   2,  96
```

```
400 REM 'RS3' PRG FILE GENERATOR
410 REM TERMINAL.R11 USER PORT RS232 FOR BASIC 2.0
420 REM
500 OPEN 8,8,8,"0:RS3,P,W"          :REM OPEN PRG FILE
510 CH=0                            :REM RESET CHECKSUM
520 FOR J=1 TO 882
530 READ X : CH=CH+X
540 PRINT#8,CHR$(X);                :REM OUTPUT BYTE
550 NEXT
560 CLOSE 8                         :REM CLOSE PRG FILE
570 PRINT CH                        :REM PRINT CHECKSUM
580 REM *** CHECKSUM SHOULD EQUAL 98006 ***
590 END
1000 DATA 144,  60                  :REM START ADDRESS
1008 DATA 255,   0,  19,   2, 147,  18, 146,   6
1016 DATA   7, 157,  29,  17,  11, 145,  13,  14
1024 DATA  15,  16,  10,  18,   1, 127,  21,  22
1032 DATA  23,  24,  25,  26,  27,  28,   9,  30
1040 DATA  31,  32,  33,  34,  35,  36,  37,  38
1048 DATA  39,  40,  41,  42,  43,  44,  45,  46
1056 DATA  47,  48,  49,  50,  51,  52,  53,  54
1064 DATA  55,  56,  57,  58,  59,  60,  61,  62
1072 DATA  63,  64, 193, 194, 195, 196, 197, 198
1080 DATA 199, 200, 201, 202, 203, 204, 205, 206
1088 DATA 207, 208, 209, 210, 211, 212, 213, 214
1096 DATA 215, 216, 217, 218,  91,  92,  93,  94
1104 DATA  95,  64,  65,  66,  67,  68,  69,  70
1112 DATA  71,  72,  73,  74,  75,  76,  77,  78
1120 DATA  79,  80,  81,  82,  83,  84,  85,  86
1128 DATA  87,  88,  89,  90, 179, 221, 171, 219
1136 DATA  20, 209,  25,  35, 120,   2,   6,  12
1144 DATA  36, 120, 169, 159, 141, 144,   0, 169
1152 DATA  62, 141, 145,   0, 169,   0, 141, 139
1160 DATA  60, 141,  79, 232, 169, 127, 141,  77
1168 DATA 232, 141,  78, 232, 169, 255, 141, 144
1176 DATA  60, 169, 130, 141,  78, 232,  32, 151
1184 DATA  62, 169,   1, 141,  67, 232,  13,  76
1192 DATA 232, 141,  76, 232, 173,  75, 232,  41
1200 DATA  31, 141,  75, 232, 173, 134,  60,  41
1208 DATA   3, 136, 168, 185,  17,  61, 141,  68
1216 DATA 232, 141,  72, 232, 185,  21,  61, 141
1224 DATA 143,  60,  88,  96, 120, 169,  46, 141
1232 DATA 144,   0, 169, 230, 141, 145,   0, 169
1240 DATA 127, 141,  77, 232, 141,  78, 232, 169
1248 DATA   0, 141,  75, 232, 169,  61, 141,  19
1256 DATA 232, 169,  32, 141,  39, 128, 141,  38
1264 DATA 128,  88,  96,  32, 131,  62, 169,   0
1272 DATA 141, 142,  60, 173, 142,  60, 205, 140
1280 DATA  60, 240,  15, 168, 177, 184, 141, 138
1288 DATA  60,  32, 179,  61, 238, 142,  60,  76
1296 DATA 155,  61,  96, 169,  32,  44, 134,  60
1304 DATA 240,  21, 174, 138,  60, 160, 127, 185
1312 DATA 145,  60, 205, 138,  60, 208,   2, 152
1320 DATA 170, 136,  16, 243, 142, 138,  60, 169
1328 DATA  16,  44, 134,  60, 240,  33, 160,   8
1336 DATA 173, 138,  60, 141, 136,  60, 169,   0
1344 DATA  14, 136,  60, 105,   0, 136, 208, 248
1352 DATA  41,   1,  74, 106,  13, 138,  60, 141
1360 DATA 138,  60, 173, 144,  60,  16, 251, 120
1368 DATA 173, 143,  60, 141,  69, 232, 169, 192
1376 DATA 141,  78, 232, 169,   9, 141, 137,  60
1384 DATA  32,  80,  62,  88, 173, 137,  60,  16
1392 DATA 251,  96, 173,  78, 232,  41,  64, 208
1400 DATA   3,  76, 172,  62, 169,  64, 141,  77
1408 DATA 232, 206, 137,  60, 208,   6,  32,  74
1416 DATA  62,  76,  60,  62, 173, 137,  60,  16
1424 DATA   8, 169,  64, 141,  78, 232,  76, 249
1432 DATA  63,  32,  69,  62, 173, 143,  60, 141
1440 DATA  69, 232,  76, 249,  63,  78, 138,  60
1448 DATA 144,   6, 169,   0, 141,  79, 232,  96
1456 DATA 169,   1, 141,  79, 232,  96, 160,   0
1464 DATA 120, 174, 139,  60, 169,   0, 141, 139
1472 DATA  60, 185, 134,  59, 153, 134,  58, 200
1480 DATA  88, 208, 246, 160,   2, 138, 145,  42
1488 DATA 200, 169, 134, 145,  42, 200, 169,  58
1496 DATA 145,  42,  96, 160,  24, 173, 139,  60
1504 DATA 145,  42,  96, 160,   9, 177,  42, 141
1512 DATA 140,  60, 200, 177,  42, 141, 184,   0
1520 DATA 200, 177,  42, 141, 185,   0,  96, 160
1528 DATA  17, 177,  42, 141, 134,  60,  96, 173
1536 DATA  39, 128,  73, 128, 141,  39, 128, 173
1544 DATA  77, 232, 208, 120, 169,  60, 141,  19
1552 DATA 232,  88, 172, 197,   0, 173, 196,   0
1560 DATA  24, 109, 198,   0, 144,   1, 200, 192
1568 DATA 131, 144,  72, 201, 152, 144,  68, 160
1576 DATA  25, 185, 223,   0,   9, 128, 153, 223
1584 DATA   0, 136, 208, 245, 169, 128, 141,  35
1592 DATA   0, 162,   0, 142,  34,   0, 160,  40
1600 DATA 177,  34, 129,  34, 238,  34,   0, 208
1608 DATA   3, 238,  35,   0, 169, 192, 205,  34
1616 DATA   0, 208, 237, 173,  35,   0, 201, 131
1624 DATA 208, 230, 169,  32, 136, 145,  34, 208
1632 DATA 251, 169,   0, 141, 205,   0, 169, 145
1640 DATA  32, 210, 255, 169,  63,  72, 169,  24
1648 DATA  72,  72,  72,  72,  72,  76,  46, 230
1656 DATA 120, 169,   1,  13,  19, 232, 141,  19
1664 DATA 232,  76, 249,  63, 170,  41,   2, 208
1672 DATA  39, 138,  41,  32, 208,   8, 138,  41
1680 DATA  64, 240, 216,  76,  18,  62, 206, 144
1688 DATA  60,  48,  71, 173, 143,  60, 141,  73
1696 DATA 232, 173,  76, 232, 106, 110, 141,  60
1704 DATA 169,  32, 141,  77, 232,  76, 249,  63
1712 DATA 174, 144,  60,  16,  29, 173,  38, 128
1720 DATA   9, 128, 141,  38, 128, 169,   8, 141
1728 DATA 144,  60, 169, 160, 141,  78, 232, 173
1736 DATA 143,  60,  74,  24, 109, 143,  60, 141
1744 DATA  73, 232, 173,  76, 232,  73,   1, 141
1752 DATA  76, 232, 169,   2, 141,  77, 232,  76
1760 DATA 249,  63, 172, 139,  60, 238, 139,  60
1768 DATA 174, 141,  60, 173, 134,  60,  41,  16
1776 DATA 240,   4, 138,  41, 127, 170, 173, 134
1784 DATA  60,  41,  32, 240,   4, 189, 145,  60
1792 DATA 170, 173, 134,  60,  41,  64, 240,   4
1800 DATA 138,  41, 127, 170, 173, 134,  60,  16
1808 DATA  13, 224,  17, 208,   9, 173, 135,  60
1816 DATA 201,  13, 208,   2, 162,   0, 138, 141
1824 DATA 135,  60, 153, 134,  59, 173, 134,  60
1832 DATA  41,   8, 240,  16, 173, 158,   0, 201
1840 DATA  10,  16,   9, 168, 138, 153, 111,   2
1848 DATA 200, 140, 158,   0, 169,  34, 141,  77
1856 DATA 232, 169,   1,  13,  76, 232, 141,  76
1864 DATA 232, 169,  32, 141,  78, 232,  32, 123
1872 DATA  62, 173,  38, 128,  41, 127, 141,  38
1880 DATA 128, 104, 168, 104, 170, 104,  64, 170
```

```
400 REM 'RS4' PRG FILE GENERATOR
410 REM TERMINAL.R11 USER PORT RS232 FOR BASIC 4.0
420 REM
500 OPEN 8,8,8,"0:RS4,P,W"          :REM OPEN PRG FILE
510 CH=0                            :REM RESET CHECKSUM
520 FOR J=1 TO 882
530 READ X : CH=CH+X
540 PRINT#8,CHR$(X);                :REM OUTPUT BYTE
550 NEXT
560 CLOSE 8                         :REM CLOSE PRG FILE
570 PRINT CH                        :REM PRINT CHECKSUM
580 REM *** CHECKSUM SHOULD EQUAL 100066 ***
590 END
1000 DATA 144,  60                  :REM START ADDRESS
1008 DATA 255,   0,  19,   2, 147,  18, 146,   6
1016 DATA   7, 157,  29,  17,  11, 145,  13,  14
1024 DATA  15,  16,  10,  18,   1, 127,  21,  22
1032 DATA  23,  24,  25,  26,  27,  28,   9,  30
1040 DATA  31,  32,  33,  34,  35,  36,  37,  38
1048 DATA  39,  40,  41,  42,  43,  44,  45,  46
1056 DATA  47,  48,  49,  50,  51,  52,  53,  54
1064 DATA  55,  56,  57,  58,  59,  60,  61,  62
1072 DATA  63,  64, 193, 194, 195, 196, 197, 198
1080 DATA 199, 200, 201, 202, 203, 204, 205, 206
1088 DATA 207, 208, 209, 210, 211, 212, 213, 214
1096 DATA 215, 216, 217, 218,  91,  92,  93,  94
1104 DATA  95,  64,  65,  66,  67,  68,  69,  70
1112 DATA  71,  72,  73,  74,  75,  76,  77,  78
1120 DATA  79,  80,  81,  82,  83,  84,  85,  86
1128 DATA  87,  88,  89,  90, 179, 221, 171, 219
1136 DATA  20, 209,  25,  35, 120,   2,   6,  12
1144 DATA  36, 120, 169, 159, 141, 144,   0, 169
1152 DATA  62, 141, 145,   0, 169,   0, 141, 139
1160 DATA  60, 141,  79, 232, 169, 127, 141,  77
1168 DATA 232, 141,  78, 232, 169, 255, 141, 144
1176 DATA  60, 169, 130, 141,  78, 232,  32, 151
1184 DATA  62, 169,   1, 141,  67, 232,  13,  76
1192 DATA 232, 141,  76, 232, 173,  75, 232,  41
1200 DATA  31, 141,  75, 232, 173, 134,  60,  41
1208 DATA   3, 136, 168, 185,  17,  61, 141,  68
1216 DATA 232, 141,  72, 232, 185,  21,  61, 141
1224 DATA 143,  60,  88,  96, 120, 169,  85, 141
1232 DATA 144,   0, 169, 228, 141, 145,   0, 169
1240 DATA 127, 141,  77, 232, 141,  78, 232, 169
1248 DATA   0, 141,  75, 232, 169,  61, 141,  19
1256 DATA 232, 169,  32, 141,  79, 128, 141,  78
1264 DATA 128,  88,  96,  32, 131,  62, 169,   0
1272 DATA 141, 142,  60, 173, 142,  60, 205, 140
1280 DATA  60, 240,  15, 168, 177, 184, 141, 138
1288 DATA  60,  32, 179,  61, 238, 142,  60,  76
1296 DATA 155,  61,  96, 169,  32,  44, 134,  60
1304 DATA 240,  21, 174, 138,  60, 160, 127, 185
1312 DATA 145,  60, 205, 138,  60, 208,   2, 152
1320 DATA 170, 136,  16, 243, 142, 138,  60, 169
1328 DATA  16,  44, 134,  60, 240,  33, 160,   8
1336 DATA 173, 138,  60, 141, 136,  60, 169,   0
1344 DATA  14, 136,  60, 105,   0, 136, 208, 248
1352 DATA  41,   1,  74, 106,  13, 138,  60, 141
1360 DATA 138,  60, 173, 144,  60,  16, 251, 120
1368 DATA 173, 143,  60, 141,  69, 232, 169, 192
1376 DATA 141,  78, 232, 169,   9, 141, 137,  60
1384 DATA  32,  80,  62,  88, 173, 137,  60,  16
1392 DATA 251,  96, 173,  78, 232,  41,  64, 208
1400 DATA   3,  76, 172,  62, 169,  64, 141,  77
1408 DATA 232, 206, 137,  60, 208,   6,  32,  74
1416 DATA  62,  76,  60,  62, 173, 137,  60,  16
1424 DATA   8, 169,  64, 141,  78, 232,  76, 249
1432 DATA  63,  32,  69,  62, 173, 143,  60, 141
1440 DATA  69, 232,  76, 249,  63,  78, 138,  60
1448 DATA 144,   6, 169,   0, 141,  79, 232,  96
1456 DATA 169,   1, 141,  79, 232,  96, 160,   0
1464 DATA 120, 174, 139,  60, 169,   0, 141, 139
1472 DATA  60, 185, 134,  59, 153, 134,  58, 200
1480 DATA  88, 208, 246, 160,   2, 138, 145,  42
1488 DATA 200, 169, 134, 145,  42, 200, 169,  58
1496 DATA 145,  42,  96, 160,  24, 173, 139,  60
1504 DATA 145,  42,  96, 160,   9, 177,  42, 141
1512 DATA 140,  60, 200, 177,  42, 141, 184,   0
1520 DATA 200, 177,  42, 141, 185,   0,  96, 160
1528 DATA  17, 177,  42, 141, 134,  60,  96, 173
1536 DATA  79, 128,  73, 128, 141,  79, 128, 173
1544 DATA  77, 232, 208, 120, 169,  60, 141,  19
1552 DATA 232,  88, 172, 197,   0, 173, 196,   0
1560 DATA  24, 109, 198,   0, 144,   1, 200, 192
1568 DATA 135, 144,  72, 201, 208, 144,  68, 234
1576 DATA 234, 234, 234, 234, 234, 234, 234, 234
1584 DATA 234, 234, 234, 234, 169, 128, 141,  35
1592 DATA   0, 162,   0, 142,  34,   0, 160,  80
1600 DATA 177,  34, 129,  34, 238,  34,   0, 208
1608 DATA 247, 238,  35,   0, 169, 136, 205,  35
1616 DATA   0, 208, 237, 169, 135, 133,  35, 169
1624 DATA 128, 133,  34, 169,  32, 160,  80, 145
1632 DATA  34, 136, 208, 251, 169, 145,  32, 210
1640 DATA 255, 234, 234, 169,  63,  72, 169,  24
1648 DATA  72,  72,  72,  72,  72,  76,  85, 228
1656 DATA 120, 169,   1,  13,  19, 232, 141,  19
1664 DATA 232,  76, 249,  63, 170,  41,   2, 208
1672 DATA  39, 138,  41,  32, 208,   8, 138,  41
1680 DATA  64, 240, 216,  76,  18,  62, 206, 144
1688 DATA  60,  48,  71, 173, 143,  60, 141,  73
1696 DATA 232, 173,  76, 232, 106, 110, 141,  60
1704 DATA 169,  32, 141,  77, 232,  76, 249,  63
1712 DATA 174, 144,  60,  16,  29, 173,  78, 128
1720 DATA   9, 128, 141,  78, 128, 169,   8, 141
1728 DATA 144,  60, 169, 160, 141,  78, 232, 173
1736 DATA 143,  60,  74,  24, 109, 143,  60, 141
1744 DATA  73, 232, 173,  76, 232,  73,   1, 141
1752 DATA  76, 232, 169,   2, 141,  77, 232,  76
1760 DATA 249,  63, 172, 139,  60, 238, 139,  60
1768 DATA 174, 141,  60, 173, 134,  60,  41,  16
1776 DATA 240,   4, 138,  41, 127, 170, 173, 134
1784 DATA  60,  41,  32, 240,   4, 189, 145,  60
1792 DATA 170, 173, 134,  60,  41,  64, 240,   4
1800 DATA 138,  41, 127, 170, 173, 134,  60,  16
1808 DATA  13, 224,  17, 208,   9, 173, 135,  60
1816 DATA 201,  13, 208,   2, 162,   0, 138, 141
1824 DATA 135,  60, 153, 134,  59, 173, 134,  60
1832 DATA  41,   8, 240,  16, 173, 158,   0, 201
1840 DATA  10,  16,   9, 168, 138, 153, 111,   2
1848 DATA 200, 140, 158,   0, 169,  34, 141,  77
1856 DATA 232, 169,   1,  13,  76, 232, 141,  76
1864 DATA 232, 169,  32, 141,  78, 232,  32, 123
1872 DATA  62, 173,  78, 128,  41, 127, 141,  78
1880 DATA 128, 104, 168, 104, 170, 104,  64,  49
```

The User Port to RS-232 Cable

Henry Troup, Toronto

The RS232 version of the BBS TERMINAL programs works the Parallel User Port as an RS232 Port. However, a special cable is necessary to interface RS232 modems.

Components List:

1 - User Port connector
1 - RS-232 (Type D) connector
1 - 620 ohm resistor
1 - 680 ohm resistor
1 - 1 K ohm resistor
1 - Diode

The User port connections are all to the lower level: pins A (ground), B (CA1), C (PA0), and L (PA7) are used. On the RS-232 end, pins 7 (Signal Ground), 2 (Transmitted Data), and 3 (Received Data) are used.

- From pin A, to pin 7.
- Between pins A and B, a 680 ohm resistor. In parallel, a diode, with the anode to pin A.
- Pin B to Pin L.
- From pin C, through a 1 K ohm resistor, to pin 2.
- From pin L, through a 620 ohm resistor, to pin 3.

That's all for the wiring.

Explanation of Theory

This circuit will drive only TTL-level RS-232 hardware.

The diode protects the parallel port from negative swings of the received data line. The 620 and 680 ohm resistors form a voltage divider, sufficient to protect the CA1 pin. The 1 K resistor provides a load, and protection against mis-connected RS-232 cables.

Received data causes a transition on PA7, and an interrupt on CA1. Transmitted data is sent through PA0.

For any further details, phone Henry Troup at 416 624-3419.

BASIC-AID,
A Super Editor For The PET

F. Arthur Cochrane
Beech Island, S.C.

For those who may not know what Basic-Aid is, I will start with a little background. Basic-Aid is a BASIC program development tool for the PET and was orginally written by Bill Seiler and is very much like the Toolkit. It has the following commands:

Aid - A Help function when a BASIC program error occurs.
Auto - Auto line numbers for program entry.
Break - Break to the TIM machine language monitor in the PET.
Change - Search for an old string and replace it with a new string in a BASIC program.
Delete - Delete a range of lines from a BASIC program.
Find - Find a string in a BASIC program and print the lines where it occurs.
Kill - Disable Basic-Aid from use.
Number - Renumber a BASIC program correcting all GOTOs and GOSUBs.
Repeat - Enable repeat keys.
Trace - Enable the trace function, which prints the line number and token in a window when a program is run.

The program was a 2K program which loaded into the top 4K of a 32K PET and worked only on Upgrade BASIC (BASIC 2). The next version of Basic-Aid that I know about was a version from Commodore Canada. This version was upgraded for BASIC 4.0 and added the following commands:

Flist - List a BASIC program directly from the disk to the screen.
Hex - Convert HEX to decimal and decimal to HEX.
Lower - Put the PET into lower case.
Merge - Merge a program from the disk with the one in memory.
Read - Read a sequential file directly from the disk to the screen.
Start - Print the loading address of a program on the disk.
Upper - Put the PET into upper case.

The next version of Basic-Aid that I came across had these commands and functions added:

Dump - Dump the variables defined in the program.
Crt - Dump the screen to the printer.
Pack - Remove the extra spaces and REM's from a BASIC program.
Dos - Also the DOS Support commands (@, >, /, ↑) were included.

The ability to print the screen with [SHIFT-ESC] and to escape from the quote/insert mode with were also added.

My additions to Basic-Aid have been the following commands and functions;

Size - Give the size of a program in memory or on the disk.
Spool - Send a file from the disk directly to the printer.
Un-new - Restore a program after a NEW.

The ability to scroll the BASIC program with the cursor control keys was added. The scroll feature was adopted from code for a version of the CBM assembler editor by Bill Seiler.

Many bugs were also fixed. I would like to thank Jim Butterfield for the AID4 program which allowed me to fix a renumber bug in Basic-Aid. The DOS commands also had bugs which were fixed. Also when upgraded to BASIC 4.0 the trace would not function because a compare was now incorrect. The screen dump was modified to allow printing to an ASCII printer.

Basic-Aid is a very powerful BASIC program development aid, but how does it compare to others available for PET/CBMs?

Basic-Aid has more features than the Toolkit and is more useful than a Toolkit alone.

The Disk-O-Pro has some useful features. The most important is the addition of BASIC 4.0 commands to Upgrade BASIC. Also the Print Using command for formatted output is useful. The Disk-O-Pro will function with a Toolkit if one is present. A disadvantage is that the Disk-O-Pro must be in place for these commands to work in a program and it slows BASIC down. See Compute issue #8 page 112 for a complete review.

The Command-O adds the Print Using command, the Toolkit commands, and others to BASIC 4.0. The Renumber command is improved to allow renumbering in a line range instead of the whole program and the Trace function has been improved to show the whole line that is being traced. But again the Command-O must be enabled for the Print Using command and others to work in a program.

Power has some different commands also, and, like the others, comes on a ROM so no user RAM is taken away. It has the improved Renumber command and a very powerful Trace function. It has a Search and Replace command with the option for don't care characters in the search string. Power also has instant keywords and instant subroutines options which can be useful. The XEC command is very powerful and has many options, such as merging a program from disk. Power has the option for other commands to be added to it. For a full review of Power see the Overview in Compute issue #18 page 136.

So if you have a PET which super-editor is for you? The answer will depend on the BASIC your PET has and the features you want a super-editor to have. Upgrade BASIC users can choose from the Toolkit with a Disk-O-Pro, Power, or Basic-Aid. Basic 4.0 users can choose the Toolkit, Command-O, Power, or Basic-Aid. Each super-editor has some features not included in the others. The user should get all the information on each and decide for himself. In this evaluation Basic-Aid has a strong selling point in that it is in the public domain and is FREE. There are other super-editors not mentioned here but these are the ones most seen in ads and the ones the author is familiar with.

Note that Original BASIC users are limited to a Toolkit only. Because of vast zero page changes between BASIC 1.0 and BASIC 2.0, and the fact that Original BASIC will not work with the Commodore disk, Basic-Aid as it stands now will not assemble for Original BASIC.

Because the VIC-20 has BASIC 2.0, it will be possible to modify Basic-Aid for VIC use, however you'll need more memory than an "off-the-shelf" VIC. The modification will involve checking the subroutine calls and modifing the scroll for the screen size, but I believe a VIC Programmers Aid cartridge is already available. If anyone is successful in the modification of this public domain version, they should be sure to publish the results for others.

But where do you get Basic-Aid? A PET user group is the best source. Two user groups which can provide Basic-Aid are ATUG and TPUG (addresses below). They should also have source code in Carl Moser's MAE assembler format and a program that will convert this to Commodore assembler format. Basic-Aid can be assembled and burned into an EPROM and plugged into one of the empty sockets in the PET so it is available with a SYS and does not have to be loaded from disk each time the PET is reset or powered up.

I would also like to thank Jim Strasma of ATUG for his help and comments on the work I did on Basic-Aid.

I hope that you will pass Basic-Aid on to your friends. This program is in the public domain and should be passed around freely. If anyone finds bugs or has comments please contact me about them.

F. Arthur Cochrane
1402 Sand Bar Ferry Rd
Beech Island, S. C.
USA 29841

Home 803 827 1902
Bus. 803 725 3652

ATUG (ASM/TED Users Group)
c/o Brent Anderson
200 S. Century
Rantoul, Ill 61866
USA
217-893-4577

TPUG (Toronto PET Users Group)
c/o Chris Bennett
381 Lawrence Ave. West
Toronto, Ontario
Canada M5M 1B9
416-782-9252

Editor' Note

The two BASIC loaders that follow are Basic-Aid for the 8032 and for the fat 4032. Both use Commodore format for printer output. Versions for BASIC 2.0, 9" BASIC 4.0 machines and ASCII printer output are all available from either of the above user groups.

Both programs were generated using the DATA Line Generator from Transactor 1, Vol. 3, pg. 12. The SYS call in line 1000 will engage Basic-Aid. For a direct load version, use the monitor Save:

```
.S "Basic-Aid.bin",0x,7000,7FFF
```

where x=8 for disk, 1 or 2 for cassette. After loading this, use the same SYS to engage, but follow with a NEW else FRE(0) will misbehave.

After entering one of these loaders you'll probably see DATA statements in your sleep! You might have a friend read them out while you type. This will reduce entering time considerably. However, with over 4000 data elements, the possibility of error is still high. To make life a little easier, add this short checksum program to the beginning of the loader:

```
10 FOR I=1 TO 10              :REM 10 PAGES OF DATA
20 L=52 : IF I=10 THEN L=40 :REM LAST PAGE SHORTER
30 CH=0                       :REM RESET CHECKSUM
40 FOR J=1 TO L*8             :REM #LINES * 8 PER LINE
50 READ X : CH=CH+X           :REM ACCUMULATE
60 NEXT J
70 PRINT "CHECKSUM FOR BLOCK";I;"=";CH
80 NEXT I : END
```

It should produce these results:

Checksum for 8032 Basic-Aid

```
CHECKSUM FOR BLOCK 1 = 53306
CHECKSUM FOR BLOCK 2 = 51652
CHECKSUM FOR BLOCK 3 = 48818
CHECKSUM FOR BLOCK 4 = 51821
CHECKSUM FOR BLOCK 5 = 48292
CHECKSUM FOR BLOCK 6 = 54770
CHECKSUM FOR BLOCK 7 = 55381
CHECKSUM FOR BLOCK 8 = 52888
CHECKSUM FOR BLOCK 9 = 55393
CHECKSUM FOR BLOCK 10 = 34935
```

Checksum for fat 4032 Basic-Aid

```
CHECKSUM FOR BLOCK 1 = 53948
CHECKSUM FOR BLOCK 2 = 51417
CHECKSUM FOR BLOCK 3 = 48814
CHECKSUM FOR BLOCK 4 = 51819
CHECKSUM FOR BLOCK 5 = 48208
CHECKSUM FOR BLOCK 6 = 54546
CHECKSUM FOR BLOCK 7 = 54546
CHECKSUM FOR BLOCK 8 = 51798
CHECKSUM FOR BLOCK 9 = 55525
CHECKSUM FOR BLOCK 10 = 34939
```

Basic-Aid Command List (As of November 27, 1981)

AUTO LINE NUMBERS Syntax: AUTO [inc]
AUTO

Auto prints the BASIC line numbers for you as you key-in a program. Enter AUTO with an increment, then enter the first line of the program. After hitting RETURN Basic-Aid will provide the next line number. Turn AUTO off by entering the command with no increment. Increment can be from 1 to 127.

BREAK TO MONITOR Syntax: BREAK

The BREAK command calls the machine language monitor in the PET. This is a call to the monitor and not a break, so open files on BASIC 4.0 will remain open for monitor listings.

CHANGE TEXT Syntax: CHANGE @search@replace@,[range]

The CHANGE command will search through a BASIC program for a string and change it to a replacement. The changed lines are displayed as they are changed. Because BASIC lines are tokenized it may be necessary to enclose the strings in quotes to change them. The whole program is searched unless a line range is given. Line range has same format as LIST.

DUMP SCREEN Syntax: CRT

The screen is printed to a printer connected to the PET as device number 4. There are versions of Basic-Aid for ASCII and Commodore printers. The screen dump to a Commodore printer will be exactly like the screen. The screen dump to an ASCII printer will be in upper case only if the PET is in graphics mode or lower/upper case if the PET is in text mode.

DELETE LINES Syntax: DELETE [range]

Deletes a given line range of lines from a BASIC program. The line range has the same format as the LIST command.

DISPLAY VARIABLES Syntax: DUMP

DUMP will list the variables used in a BASIC program and their values. Dump does not list arrays or variables that have not yet been "seen" by BASIC. The variables are listed in the order in which they were created. The variables are printed in such a way that they can be edited with the screen editor so that a program can be stopped, the varibles dumped, edited to new values, and the CONT command given to continue the program with the new values.

FIND TEXT Syntax: FIND @search string@,[range]

The FIND command searches a BASIC program for a string and displays the lines where it occurs. Because BASIC lines are tokenized it may be necessary to enclose the search string in quotes to find it. The whole program is searched unless a line range is given. Line range has same format as LIST.

DISK LIST Syntax: FLIST "program filename"

This command will list a BASIC program on the disk directly to the screen without affecting the contents of memory. WARNING: DO NOT use the keyprint function to try and dump the screen to the printer while this command is executing.

DISPLAY ERROR Syntax: HELP

The HELP command ('AID' in older versions) will display the BASIC line that caused the BASIC program to stop and highlight where in the line the problem occured.

CONVERT HEX-DEC Syntax: HEX $ [hex number]
HEX [decimal number]

The HEX command will convert HEX to decimal and decimal to HEX. This can be very useful in figuring PEEK, POKE, and SYS addresses. If the number input is preceded by a dollar symbol then the number is taken to be HEX and the decimal value for it is printed. If a decimal number is entered then the HEX value for it is returned. The range for conversion is 0 to 65535 or $0000 to $FFFF.

KILL BASIC AID Syntax: KILL

Basic-Aid can be disabled with the KILL command. This restores the IRQ vector and CHRGET routine in zero-page. Re-enable Basic-Aid with a SYS to the start of the Basic-Aid machine code. (32K RAM version -- SYS 7*4096)

LOWER CASE MODE Syntax: LOWER

Puts the PET into lower case mode. (Same as POKE 59468,14.)

MERGE PROGRAMS Syntax: MERGE "program filename"

Merge a BASIC program from disk with the one in memory. The merging will be just like the program was typed in from the keyboard so lines are merged between ones in memory if necessary and duplicate lines in memory are replaced with the merged lines. The program is listed as it is merged.

KILL REPEAT/SCROLL Syntax: OFF

Restores the PETs normal IRQ vector. This will cancel repeat keys (except on FAT 40s and 8032s), scrolling, and keyprint.

PACK A PROGRAM Syntax: PACK

Removes remarks and waste spaces in a BASIC program. Note don't branch in a BASIC program to deleted remarks. This command is fooled easily so keep a copy of the orginal in case the packing does not function properly.

SEQ READ Syntax: READ "seq filename"

The READ command will read a sequential file from the disk and print it to the screen. This command can be very handy for viewing data created by programs. WARNING: DO NOT use the keyprint function to try and dump the screen to the printer while this command is executing.

LINE RENUMBER Syntax: RENUMBER
RENUMBER [start line#]
RENUMBER [start line#], [inc]

Renumbers a BASIC program correcting all GOTOs and GOSUBs in the program. The program is renumbered starting at 100 and with an increment of 10. A starting line number can be input other than 100 and an increment other than 10 can be input.

ENABLE REPEAT Syntax: REPEAT
SCROLL (FAT 40s & 8032s)

Enables repeat keys, scrolling, and keyprint. Repeat keys are set automatically when Basic-Aid is first called and automatically cancelled each time a program is loaded.

PROGRAM SIZE Syntax: SIZE
SIZE "program filename"

SIZE gives the size of a BASIC program in memory or any program on disk. The size of a program in memory is found by substracting the end of the program location from the start of the program location. The size of a program on disk is found by counting the bytes in the file. The size is given in decimal and HEX.

SEQ SPOOL Syntax: SPOOL "sequential filename"
SPOOL

Sends an SEQ file directly from the disk to the printer. The PET can then do other things, such as editing a program or running a program (but with no access to the IEEE bus). Basic-Aid opens the specified file and listens the printer then gets off the IEEE bus which allows the disk to talk directly to the printer. When the printer stops, enter SPOOL with no filename to unlisten the printer, untalk the disk, and close the file. Use the spool command to list a long program while you use the PET for something else. Create a file with:

```
OPEN8,8,8,"0:TEMP,S,W":CMD8:LIST
PRINT#8:CLOSE8
```

FIND LOAD ADDRESS Syntax: START "program filename"

Returns the load address of a program on the disk. The load address is found by reading the first two bytes of the file which is the address where the program is loaded. The load address is given in decimal and HEX. This command can be used to find where machine language programs load.

PROGRAM TRACE Syntax: TRACE [speed]
TRACE

The TRACE command enables or disables the tracing of a BASIC program. Tracing is enabled with the command and a number and disabled with the command alone. The number input controls the speed of the tracing. The number can be from 1 to 127 with 1 being the fastest and 127 the slowest. Tracing takes place in a window in the upper right of the screen with the last nine lines traced and the current line that is executing. The line number and what is executing in the line are listed.

If after a New command is entered it is discovered that a program has not been saved it can be recovered with this command.

PUT THE PET INTO UPPER CASE

UPPER CASE MODE Syntax: UPPER

Puts the PET into upper case mode. (same as POKE 59468,12)

DOS SUPPORT Syntax: > (or @)
>disk command
>$0
/program name
↑program name

The DOS support commands are supported. The at sign and greater than (@, >) symbols are used to read the error channel, send commands, and display the disk directory. The symbol alone will read the error channel and print it to the screen. The symbol followed by a disk command will send that command to the disk. The symbol followed by the dollor symbol will display the directory to the screen. WARNING: DO NOT use the keyprint function to try and dump the screen to the printer while this command is executing. The slash (/) will load a program from the disk. Repeat keys are not disabled by this load. The uparrow (↑) will load and execute a program from disk.

ESCAPE QUOTE MODE

Press the STOP key on graphics keyboards to escape quotes mode or cancel outstanding inserts (same as the ESCape key on business keyboards). This function will only work when repeat keys are enabled.

KEYPRINT

This function allows the screen to be printed to the printer with the press of one key. This is the same as the CRT command except that in can occur in a program. On graphics keyboards use the shifted backslash and on business use the shifted escape. This function is available only when repeat keys are enabled.

SCROLL A PROGRAM

The cursor up/down keys can be used to scroll through a BASIC program. The cursor must be in the first two columns for scrolling to take place. This function is only available when repeat keys are enabled (use REPEAT or SCROLL to enable).

NOTE: The commands which print to the screen (Change, Dump, Find, Flist, Merge, Read, Trace, and Directory (>$)) can be paused, held, or stopped. Pause with the SHIFT key, stop with the STOP key. Hold the display with the space bar on graphics keyboards and 6 on business keyboards. To continue use the < key on graphics keyboards and 9 on business keyboards.

Fat 4032 Basic-Aid Loader

```
1000 FOR J=28672 TO 32735 : READ X : POKE J,X : NEXT : SYS 7*4096
1008 DATA 169,   0, 141, 136,   3, 141, 137,   3
1016 DATA 173, 204, 127, 174, 205, 127, 224, 128
1024 DATA 176,   7, 133,  52, 134,  53,  32, 233
1032 DATA 181, 162,  15, 189,  43, 112, 149, 112
1040 DATA 202,  16, 248, 162,  19,  32,  59, 112
1048 DATA  76,  35, 114, 230, 119, 208,   2, 230
1056 DATA 120, 173, 255,   1,  76,  81, 113, 234
1064 DATA  76, 131, 113, 189, 247, 126, 240,   6
1072 DATA  32, 210, 255, 232, 208, 245,  96, 208
1080 DATA  14, 120, 162,  23, 189, 153, 211, 149
1088 DATA 112, 202,  16, 248,  76, 166, 120,  76
1096 DATA 239, 126, 169, 128, 133, 133, 169,  25
1104 DATA 133, 132, 169,  10, 133, 131, 162,   0
1112 DATA 160,  40, 169,  15, 133, 124, 177, 132
1120 DATA 129, 132,  32,  74, 113, 198, 124, 208
1128 DATA 245,  24, 169,  25, 101, 132, 133, 132
1136 DATA 144,   2, 230, 133, 198, 131, 208, 226
1144 DATA 166,  54, 165,  55, 134,  96,  32,  63
1152 DATA 113, 160,   0, 162,   5, 200, 185,   0
1160 DATA   1, 208, 250, 136, 240,  11, 202, 185
1168 DATA   0,   1,   9, 128, 157, 129, 129, 208
1176 DATA 242, 169, 160, 202,  48,   5, 157, 129
1184 DATA 129,  16, 248, 141, 134, 129, 232, 165
1192 DATA 128,  48,  38, 160,   0, 177, 119, 240
1200 DATA  20,  48,  15,  41, 191,   9, 128, 157
1208 DATA 135, 129, 232, 200, 192,   9, 208, 237
1216 DATA 240,  45, 169, 189,  44, 169, 160, 157
1224 DATA 135, 129, 232, 224,   9, 144, 246, 176
1232 DATA  30,  32,  96, 117, 200, 177, 132,  48
1240 DATA   9,  24, 105,  64, 157, 135, 129, 232
1248 DATA 208, 242,  41, 191,  44, 169, 160, 157
1256 DATA 135, 129, 232, 224,   9, 208, 246,  32
1264 DATA  20, 113, 169,  16, 174, 137,   3, 202
1272 DATA 240,  52, 141,  69, 232,  44,  77, 232
1280 DATA  80, 251, 112, 243,  36, 152, 208, 252
1288 DATA  32,  52, 113, 208,  33,  32,  52, 113
1296 DATA 240, 251, 201, 255, 240, 247,  72,  32
1304 DATA  52, 113, 201, 255, 208, 249, 169,   0
1312 DATA 133, 158, 104,  96, 173,  18, 232, 205
1320 DATA  18, 232, 208, 248, 201, 251,  96, 133
1328 DATA  95, 162, 144,  56,  32, 127, 205,  76
1336 DATA 147, 207, 230, 132, 208,   2, 230, 133
1344 DATA  96, 133, 128, 134, 129, 186, 189,   1
1352 DATA   1, 201,  15, 240,  48, 166, 120, 224
1360 DATA   2, 240,  24, 134, 135, 166, 119, 134
1368 DATA 134, 174, 137,   3, 240,  13,  48,  11
1376 DATA 201,  35, 208,   7, 132, 130,  32,  90
1384 DATA 112, 164, 130, 166, 129, 165, 128, 201
1392 DATA  58, 176, 187, 201,  32, 240,   3,  76
1400 DATA 170, 211,  76, 112,   0, 189,   2,   1
1408 DATA 201, 180, 208, 231,  32, 123, 113, 144
1416 DATA  77, 165, 128,  16,   2, 230, 119, 162
```

BLOCK 1 (CHECKSUM FOR BLOCK 1 = 53948)

```
1424 DATA   0, 134, 124, 132, 130, 164, 119, 185
1432 DATA   0,   2,  56, 253,  30, 127, 240,  19
1440 DATA 201, 128, 240,  19, 230, 124, 232, 189
1448 DATA  29, 127,  16, 250, 189,  30, 127, 208
1456 DATA 228, 240, 182, 232, 200, 208, 224, 132
1464 DATA 119, 104, 104, 165, 124,  10, 170, 189
1472 DATA 149, 127,  72, 189, 148, 127,  72,  32
1480 DATA 121, 113,  76, 112,   0,  32, 198, 116
1488 DATA 141, 136,   3,  76, 247, 114, 104, 104
1496 DATA 165, 128,  32, 246, 184, 240,  47, 173
1504 DATA 136,   3, 240,  42,  24, 165,  17, 109
1512 DATA 136,   3, 133,  96, 165,  18, 105,   0
1520 DATA  32,  63, 113, 162,   0, 169,  32, 157
1528 DATA 111,   2, 232, 189,   0,   1, 240,   6
1536 DATA 157, 111,   2, 232, 208, 245, 169,  32
1544 DATA 157, 111,   2, 232, 134, 158,  76,  34
1552 DATA 180, 208,  20, 120, 173, 206, 127, 133
1560 DATA 144, 173, 207, 127, 133, 145, 169,   2
1568 DATA 141, 140,   3,  88,  76, 255, 179,  76
1576 DATA 239, 126,  36, 152, 240,  38,  32, 129
1584 DATA 119, 162,   0, 134, 205, 134, 220, 134
1592 DATA 159, 134, 158, 240,  40, 189,   5,   1
1600 DATA 201,  43, 208,  10, 189,   6,   1, 201
1608 DATA 249, 208,   3,  32, 192, 252, 165, 151
1616 DATA 201,  92, 240, 214, 201,   3, 240, 217
1624 DATA 205, 138,   3, 240,   8, 141, 138,   3
1632 DATA 169,  16, 141, 139,   3, 173, 209, 127
1640 DATA  72, 173, 208, 127,  72,   8,  72,  72
1648 DATA  72,  76,  85, 228, 201, 255, 240, 237
1656 DATA 173, 139,   3, 240,   5, 206, 139,   3
1664 DATA 208, 227, 206, 140,   3, 208, 222, 169
1672 DATA   2, 141, 140,   3, 165, 158, 208, 213
1680 DATA 169,   0, 133, 151, 169,   2, 133, 168
1688 DATA 208, 203,  32,  52, 116, 165,  92, 166
1696 DATA  93, 133,  33, 134,  34,  32, 163, 181
1704 DATA 165,  92, 166,  93, 144,  10, 160,   1
1712 DATA 177,  92, 240,   4, 170, 136, 177,  92
1720 DATA 133, 119, 134, 120, 165,  33,  56, 229
1728 DATA 119, 170, 165,  34, 229, 120, 168, 176
1736 DATA  30, 138,  24, 101,  42, 133,  42, 152
1744 DATA 101,  43, 133,  43, 160,   0, 177, 119
1752 DATA 145,  33, 200, 208, 249, 230, 120, 230
1760 DATA  34, 165,  43, 197,  34, 176, 239,  32
1768 DATA 182, 180, 165,  31, 166,  32,  24, 105
1776 DATA   2, 133,  42, 144,   1, 232, 134,  43
1784 DATA  32, 233, 181,  76, 255, 179,  32, 251
1792 DATA 180,  32, 112,   0, 133, 128, 162,   0
1800 DATA 134,  70,  32, 244, 115, 165, 124, 201
1808 DATA   3, 208,   7, 162,   2, 134,  70,  32
1816 DATA 244, 115,  32, 112,   0, 240,   3,  32
1824 DATA 245, 190,  32,  52, 116, 165,  92, 166
1832 DATA  93, 133, 119, 134, 120,  32, 223, 186
```

BLOCK 2 (CHECKSUM FOR BLOCK 2 = 51417)

```
1840 DATA 208,  11, 200, 152,  24, 101, 119, 133
1848 DATA 119, 144,   2, 230, 120,  32, 196, 118
1856 DATA 240,   5,  32,  94, 116, 176,   3,  76
1864 DATA 247, 114, 132,  82, 230,  82, 164,  82
1872 DATA 166,  46, 165,  47, 133, 128, 177, 119
1880 DATA 240, 216, 221,   0,   2, 208, 237, 232
1888 DATA 200, 198, 128, 208, 241, 136, 132,   5
1896 DATA 132, 129, 165,  70, 240,  91,  32, 113
1904 DATA 116, 165,  49,  56, 229,  47, 133, 180
1912 DATA 240,  40, 200, 240, 202, 177, 119, 208
1920 DATA 249,  24, 152, 101, 180, 201,   2, 144
1928 DATA  64, 201,  75, 176,  60, 165, 180,  16
1936 DATA   2, 198, 128,  24, 101,   5, 133, 129
1944 DATA 176,   5,  32, 171, 116, 240,   3,  32
1952 DATA 147, 116, 165, 129,  56, 229,  49, 168
1960 DATA 200, 165,  49, 240,  15, 133, 130, 166
1968 DATA  48, 189,   0,   2, 145, 119, 232, 200
1976 DATA 198, 130, 208, 245,  24, 165,  42, 101
1984 DATA 180, 133,  42, 165,  43, 101, 128, 133
1992 DATA  43, 165, 119, 166, 120, 133,  92, 134
2000 DATA  93, 166,  64, 165,  65,  32,  18, 117
2008 DATA  32,  20, 113, 201, 239, 240, 156, 164
2016 DATA 129,  76,  90, 115, 164, 119, 200, 148
2024 DATA  46, 169,   0, 149,  47, 185,   0,   2
2032 DATA 240,  47, 197, 128, 240,   5, 246,  47
2040 DATA 200, 208, 242, 132, 119,  96, 208,  33
2048 DATA 141,   3,   2, 142,   4,   2, 140,   5
2056 DATA   2,   8, 104, 141,   2,   2, 169, 179
2064 DATA  72, 169, 255,  72,  56,  76, 114, 212
2072 DATA 201, 171, 240,   4, 201,  45, 208,   1
2080 DATA  96,  76, 239, 126, 144,   5, 240,   3
2088 DATA  32,  40, 116,  32, 246, 184,  32, 163
2096 DATA 181,  32, 118,   0, 240,  11,  32,  40
2104 DATA 116,  32, 112,   0,  32, 246, 184, 208
2112 DATA 224, 165,  17,   5,  18, 208,   6, 169
2120 DATA 255, 133,  17, 133,  18,  96,  32, 196
2128 DATA 118, 133,  64,  32, 196, 118, 133,  65
2136 DATA 165,  17, 197,  64, 165,  18, 229,  65
2144 DATA  96, 165, 119, 133,  31, 165, 120, 133
2152 DATA  32, 165,  42, 133,  33, 165,  43, 133
2160 DATA  34,  96, 165,  31, 197,  33, 208,   4
2168 DATA 165,  32, 197,  34,  96, 230,  31, 208
2176 DATA   2, 230,  32, 164,   5, 200, 177,  31
2184 DATA 164, 129, 200, 145,  31,  32, 130, 116
2192 DATA 208, 235,  96, 165,  33, 208,   2, 198
2200 DATA  34, 198,  33, 164,   5, 177,  33, 164
2208 DATA 129, 145,  33,  32, 130, 116, 208, 235
2216 DATA  96, 201,  34, 208,   8,  72, 165,   9
2224 DATA  73, 128, 133,   9, 104,  96,  32, 246
2232 DATA 184, 165,  18, 208,   4, 165,  17,  16
2240 DATA   2, 169, 127,  96,  32, 198, 116, 141
2248 DATA 137,   3,  76, 255, 179,  76, 239, 126
```

BLOCK 3 (CHECKSUM FOR BLOCK 3 = 48814)

```
2256 DATA 208, 251,  32, 232, 116,  76, 255, 179
2264 DATA  32, 223, 186, 133,  70, 166,  40, 165
2272 DATA  41, 134,  92, 133,  93, 160,   0, 177
2280 DATA  92, 170, 200, 177,  92, 208,   3,  76
2288 DATA 112,   0, 197, 135, 144, 235, 228, 134
2296 DATA 144, 231, 200, 177,  92, 170, 200, 177
2304 DATA  92,  44, 160,   0, 132, 124, 132,   9
2312 DATA  32, 131, 207, 169,  32, 164, 124,  41
2320 DATA 127,  32, 210, 255,  32, 185, 116, 169
2328 DATA   0, 133, 159, 200,  36,  70,  16,  23
2336 DATA 166,  93, 152,  56, 101,  92, 144,   1
2344 DATA 232, 228, 135, 144,  10, 197, 134, 144
2352 DATA   6, 169,   1, 133,  70, 133, 159, 177
2360 DATA  92, 240,  17,  16, 212, 201, 255, 240
2368 DATA 208,  36,   9,  48, 204, 132, 124,  32
2376 DATA 122, 117,  48, 193,  76, 223, 186,  96
2384 DATA 162, 176, 160, 177, 134, 133, 132, 132
2392 DATA  56, 233, 127, 170, 160,   0, 202, 240
2400 DATA 238,  32,  74, 113, 177, 132,  16, 249
2408 DATA  48, 244,  32,  96, 117, 200, 177, 132
2416 DATA  48, 221,  32, 210, 255, 208, 246,  32
2424 DATA 246, 184, 164,  17, 166,  18, 152,   5
2432 DATA  18, 208,   2, 160, 100, 132,  50, 134
2440 DATA  51, 162,   0, 161, 119, 208,   4, 169
2448 DATA  10, 208,  10,  32, 245, 190,  32, 246
2456 DATA 184, 165,  17, 166,  18, 133,  48, 134
2464 DATA  49,  32,  34, 182,  32, 196, 118,  32
2472 DATA 196, 118, 208,  33,  32, 166, 118,  32
2480 DATA 196, 118,  32, 196, 118, 208,   3,  76
2488 DATA 247, 114,  32, 196, 118, 165,  96, 145
2496 DATA 119,  32, 196, 118, 165,  95, 145, 119
2504 DATA  32, 177, 118, 240, 226,  32, 196, 118
2512 DATA  32, 196, 118,  32, 196, 118, 201,  34
2520 DATA 208,  11,  32, 196, 118, 240, 197, 201
2528 DATA  34, 208, 247, 240, 238, 170, 240, 188
2536 DATA  16, 233, 162,   4, 221,  25, 127, 240
2544 DATA   5, 202, 208, 248, 240, 221, 165, 119
2552 DATA 133,  56, 165, 120, 133,  57,  32, 112
2560 DATA   0, 176, 211,  32, 246, 184,  32,  80
2568 DATA 118, 165,  57, 133, 120, 165,  56, 133
2576 DATA 119, 160,   0, 162,   0, 189,   1,   1
2584 DATA 201,  48, 144,  17,  72,  32, 112,   0
2592 DATA 144,   3,  32, 124, 118, 104, 160,   0
2600 DATA 145, 119, 232, 208, 232,  32, 112,   0
2608 DATA 176,   8,  32, 139, 118,  32, 118,   0
2616 DATA 144, 248, 201,  44, 240, 184, 208, 150
2624 DATA  32, 166, 118,  32, 196, 118,  32, 196
2632 DATA 118, 208,   8, 169, 255, 133,  96, 133
2640 DATA  95,  48,  14,  32, 196, 118, 197,  17
2648 DATA 208,  10,  32, 196, 118, 197,  18, 208
2656 DATA   6,  76, 135, 207,  32, 196, 118,  32
2664 DATA 177, 118, 240, 215,  32, 156, 118, 230
```

BLOCK 4 (CHECKSUM FOR BLOCK 4 = 51819)

```
2672 DATA 129,  32, 171, 116, 230,  42, 208,   2
2680 DATA 230,  43,  96,  32, 156, 118, 198, 129
2688 DATA  32, 147, 116, 165,  42, 208,   2, 198
2696 DATA  43, 198,  42,  96,  32, 113, 116, 160
2704 DATA   0, 132,   5, 132, 129,  96, 165,  50
2712 DATA 133,  96, 165,  51, 133,  95,  76,  34
2720 DATA 182, 165,  96,  24, 101,  48, 133,  96
2728 DATA 165,  95, 101,  49, 133,  95,  32, 196
2736 DATA 118, 208, 251,  96, 160,   0, 230, 119
2744 DATA 208,   2, 230, 120, 177, 119,  96,  76
2752 DATA 255, 179, 208,  89, 165,  42, 133,  92
2760 DATA 165,  43, 133,  93, 165,  92, 197,  44
2768 DATA 165,  93, 229,  45, 176, 233, 160,   0
2776 DATA 132,  33, 200, 177,  92,  10, 102,  33
2784 DATA  74, 153,  66,   0, 136,  16, 244,  36
2792 DATA  33, 240,  30,  16,  51,  80,  89,  32
2800 DATA 108, 119, 162,  37, 169,  61,  32,  49
2808 DATA 215, 160,   2, 177,  92,  72, 200, 177
2816 DATA  92, 168, 104,  32, 188, 196,  76,  39
2824 DATA 119,  32, 108, 119, 169,  61,  32, 210
2832 DATA 255,  32, 185, 194,  32, 216, 204,  32
2840 DATA 141, 207,  76,  85, 119,  76, 239, 126
2848 DATA  32, 108, 119, 162,  36, 169,  61,  32
2856 DATA  49, 215, 169,  34,  32, 210, 255, 160
2864 DATA   4, 177,  92, 133,  32, 136, 177,  92
2872 DATA 133,  31, 136, 177,  92,  32,  35, 187
2880 DATA 169,  34,  32, 210, 255,  32, 223, 186
2888 DATA  32, 225, 255,  32,  20, 113,  24, 165
2896 DATA  92, 105,   7, 133,  92, 144,   2, 230
2904 DATA  93,  76, 220, 118, 165,  66,  32, 210
2912 DATA 255, 165,  67, 240,   3,  32, 210, 255
2920 DATA  96, 208, 178,  32, 129, 119,  76, 255
2928 DATA 179, 169, 128, 133,  32, 169,   0, 133
2936 DATA  31, 169,   4, 133, 212,  32, 213, 240
2944 DATA  32,  72, 241, 169,  25, 133,  34, 169
2952 DATA  13, 133,   9,  32, 158, 241, 169,  17
2960 DATA 174,  76, 232, 224,  12, 208,   2, 169
2968 DATA 145,  32, 158, 241, 160,   0, 177,  31
2976 DATA  41, 127, 170, 177,  31,  69,   9,  16
2984 DATA  11, 177,  31, 133,   9,  41, 128,  73
2992 DATA 146,  32, 158, 241, 138, 201,  32, 176
3000 DATA   4,   9,  64, 208,  14, 201,  64, 144
3008 DATA  10, 201,  96, 176,   4,   9, 128, 208
3016 DATA   2,  73, 192,  32, 158, 241, 200, 192
3024 DATA  40, 144, 203, 165,  31, 105,  39, 133
3032 DATA  31, 144,   2, 230,  32, 198,  34, 208
3040 DATA 166, 169,  13,  32, 158, 241,  32, 158
3048 DATA 241,  76, 185, 241,  76, 239, 126,  76
3056 DATA 247, 114, 208, 248, 169,   1, 133, 132
3064 DATA 133,  31, 169,   4, 133, 133, 133,  32
3072 DATA 160,   3, 177, 132, 145,  31, 136,  16
3080 DATA 249, 200, 132,   9, 177,  31, 200,  17
```

BLOCK 5 (CHECKSUM FOR BLOCK 5 = 48208)

```
3088 DATA  31, 240, 220, 160,   4, 177, 132, 201
3096 DATA  58, 208,   6, 200, 177, 132, 240,  48
3104 DATA 136, 177, 132, 201, 143, 240,  41,  36
3112 DATA   9, 112,   4, 201,  58, 240,   8, 201
3120 DATA  32, 208,   9,  36,   9,  48,   5,  32
3128 DATA  74, 113, 208, 229, 170, 169,  64,   5
3136 DATA   9, 133,   9, 138, 145,  31, 200, 201
3144 DATA   0, 240,  46,  32, 185, 116, 208, 209
3152 DATA 136,  36,   9, 112,  13, 160,   0,  24
3160 DATA 165, 132, 105,   4, 133, 132, 144,   2
3168 DATA 230, 133, 177, 132, 240,   5,  32,  74
3176 DATA 113, 208, 247,  36,   9,  80,   5, 145
3184 DATA  31, 200,  16,   5,  32,  74, 113, 208
3192 DATA 135, 152, 170, 160,   1, 177,  31, 133
3200 DATA 133, 136, 177,  31, 133, 132,  24, 138
3208 DATA 101,  31, 133,  31, 144,   2, 230,  32
3216 DATA  76,  16, 120, 208,  10, 120, 162,  12
3224 DATA  32, 224, 252,  88,  76, 255, 179,  76
3232 DATA 239, 126, 208, 251, 169,   4, 141,   2
3240 DATA   4,  76, 247, 114, 160,   0, 132, 131
3248 DATA 132,  13, 201,   0, 240,  32, 201,  36
3256 DATA 240,  64, 169,   8, 133, 212,  32, 213
3264 DATA 240, 169, 111,  32,  67, 241, 177, 119
3272 DATA 240,   6,  32, 158, 241, 200, 208, 246
3280 DATA  32, 185, 241,  76, 255, 179, 169,   8
3288 DATA 133, 212,  32, 210, 240, 169, 111,  32
3296 DATA 147, 241,  32, 192, 241, 201,  13, 240
3304 DATA   5,  32, 210, 255, 208, 244,  32, 174
3312 DATA 241,  76, 112, 121, 169,  94, 133, 131
3320 DATA 160,   0, 200, 177, 119, 208, 251, 132
3328 DATA 209, 165, 119, 133, 218, 165, 120, 133
3336 DATA 219, 169,   8, 133, 212, 165, 131, 208
3344 DATA  85, 169,  96, 133, 211,  32, 204, 255
3352 DATA  32, 165, 244,  32, 210, 240, 165, 211
3360 DATA  32, 147, 241,  32, 223, 186, 169,   0
3368 DATA 133, 150, 160,   3, 132, 209,  32, 192
3376 DATA 241, 170, 164, 150, 208,  39,  32, 192
3384 DATA 241, 164, 150, 208,  32, 198, 209, 208
3392 DATA 237,  32, 131, 207,  32,  62, 187,  32
3400 DATA 192, 241, 240,   5,  32, 210, 255, 208
3408 DATA 246,  32, 223, 186, 160,   2,  32,  20
3416 DATA 113, 201, 239, 208, 207,  32, 206, 243
3424 DATA  32, 223, 186,  76, 255, 179, 169,   0
3432 DATA 133, 150, 133, 157,  32,  86, 243, 165
3440 DATA 150,  41, 191, 208,  40,  44, 128, 241
3448 DATA  16,   6, 230, 201, 208,   2, 230, 202
3456 DATA 165, 202, 133,  43, 165, 201, 133,  42
3464 DATA  32, 233, 181,  32, 182, 180, 165, 131
3472 DATA 201,  94, 240,   3,  76, 255, 179,  32
3480 DATA  34, 182,  76,  74, 183,  76,  37, 244
3488 DATA  76,   0, 230, 165, 158, 240, 249, 165
3496 DATA 198, 201,   2, 176, 243, 173, 111,   2
```

BLOCK 6 (CHECKSUM FOR BLOCK 6 = 54546)

```
3504 DATA  41, 127, 201,  17, 208, 234, 173,   1
3512 DATA   4,  13,   2,   4, 240, 226, 169,   0
3520 DATA 141, 132,   3, 141, 134,   3, 169,  39
3528 DATA 141, 135,   3, 169,  24, 141, 133,   3
3536 DATA  32, 225, 123, 173, 111,   2,  48,  92
3544 DATA 166, 216, 224,  24, 208,  83, 142, 131
3552 DATA   3, 142, 129,   3, 202,  48,  59, 180
3560 DATA 224,  16, 249, 189, 152, 231, 133, 119
3568 DATA 132, 120,  32, 240, 123, 176, 237,  32
3576 DATA 246, 184, 230,  17, 208,   2, 230,  18
3584 DATA  32, 163, 181, 176,  16, 208,  14,  32
3592 DATA 241, 122,  32, 241, 122, 169,   0, 133
3600 DATA  17, 133,  18, 240, 235,  32, 241, 122
3608 DATA 206, 129,   3, 165, 224,  16, 246,  32
3616 DATA  58, 123, 162,   0, 189, 112,   2, 157
3624 DATA 111,   2, 232, 228, 158, 208, 245, 198
3632 DATA 158,  76,   0, 230, 166, 216, 208, 249
3640 DATA 142, 129,   3, 142, 131,   3, 202, 232
3648 DATA 224,  25, 176, 222, 180, 224,  16, 247
3656 DATA 189, 152, 231, 133, 119, 132, 120,  32
3664 DATA 240, 123, 176, 235,  32, 246, 184,  32
3672 DATA 163, 181, 165,  92, 166,  93, 197,  40
3680 DATA 208,  18, 228,  41, 208,  14,  32, 185
3688 DATA 122,  32, 185, 122, 169, 255, 133,  17
3696 DATA 133,  18, 208, 227, 133, 218, 202, 134
3704 DATA 219, 160, 255, 200, 177, 218, 170, 208
3712 DATA 250, 200, 177, 218, 197,  92, 208, 246
3720 DATA 200, 177, 218, 197,  93, 208, 239, 136
3728 DATA 152,  24, 101, 218, 133,  92, 165, 219
3736 DATA 105,   0, 133,  93, 165, 248,  48,   3
3744 DATA  32, 185, 122,  32, 185, 122,  76,  47
3752 DATA 122, 174, 133,   3, 232, 202, 189, 152
3760 DATA 231, 133,  33, 181, 224,   9, 128, 133
3768 DATA  34, 180, 223,  48,   2,  41, 127, 149
3776 DATA 224, 172, 134,   3, 136, 236, 132,   3
3784 DATA 240,  79, 189, 151, 231, 133,  31, 181
3792 DATA 223,   9, 128, 133,  32, 200, 177,  31
3800 DATA 145,  33, 204, 135,   3, 144, 246, 176
3808 DATA 204, 174, 132,   3, 202, 232, 189, 152
3816 DATA 231, 133,  33, 181, 224,   9, 128, 133
3824 DATA  34, 180, 225,  48,   2,  41, 127, 149
3832 DATA 224, 172, 134,   3, 136, 236, 133,   3
3840 DATA 176,  23, 189, 153, 231, 133,  31, 181
3848 DATA 225,   9, 128, 133,  32, 200, 177,  31
3856 DATA 145,  33, 204, 135,   3, 144, 246, 176
3864 DATA 204, 169,  32, 200, 145,  33, 204, 135
3872 DATA   3, 144, 248, 181, 224,   9, 128, 149
3880 DATA 224,  96, 162,   0, 134,   9, 142, 130
3888 DATA   3, 174, 129,   3, 189, 152, 231, 180
3896 DATA 224, 133, 218, 132, 219,  32,  20, 124
3904 DATA 133,  96,  32,  23, 124,  32,  63, 113
3912 DATA  32, 150, 123, 162,   0, 189,   1,   1
```

BLOCK 7 (CHECKSUM FOR BLOCK 7 = 54546)

```
3920 DATA 240,   6,  32, 152, 123, 232, 208, 245
3928 DATA 169,  32,  41, 127,  32, 152, 123,  32
3936 DATA  23, 124,   8,  32, 185, 116,  40,  48
3944 DATA   4, 240,  52, 208, 239, 201, 255, 240
3952 DATA 235,  36,   9,  48, 231,  32,  96, 117
3960 DATA 200, 177, 132,  48, 221, 132, 135,  32
3968 DATA 152, 123, 164, 135, 208, 242, 169,  32
3976 DATA  32,  34, 124, 160,   0, 145, 218, 230
3984 DATA 218, 208,   2, 230, 219, 238, 130,   3
3992 DATA 173, 130,   3, 201,  40, 240,   1,  96
4000 DATA 173, 131,   3, 240,  17, 172, 129,   3
4008 DATA 192,  24, 240,  36, 138,  72, 206, 133
4016 DATA   3,  32, 241, 122, 176,   8, 138,  72
4024 DATA 238, 132,   3,  32, 185, 122, 133, 219
4032 DATA  41, 127, 149, 224, 189, 152, 231, 133
4040 DATA 218, 169,   0, 141, 130,   3, 104, 170
4048 DATA  96, 165, 170, 240,  10, 160,   0, 132
4056 DATA 170, 164, 198, 165, 169, 145, 196,  96
4064 DATA 160,   0, 140, 130,   3, 240,  16, 230
4072 DATA 119, 208,   2, 230, 120, 238, 130,   3
4080 DATA 173, 130,   3, 201,  40, 176, 232, 177
4088 DATA 119, 201,  58, 176, 226, 201,  32, 240
4096 DATA 230,  76, 170, 211,  32,  23, 124, 230
4104 DATA  92, 208,   2, 230,  93, 160,   0, 177
4112 DATA  92,  96, 133, 134,  41, 127, 201,  32
4120 DATA   8,  41,  63,  40, 176,   2,   9, 128
4128 DATA  36, 134,  16,   2,   9,  64,  96, 104
4136 DATA 104,  32, 223, 186,  32, 204, 255, 169
4144 DATA   1,  32, 226, 242, 169,   2,  32, 226
4152 DATA 242,  76, 255, 179,  32, 207, 255,  32
4160 DATA 207, 255, 240, 227, 165, 150, 208, 223
4168 DATA 162, 255,  32, 207, 255, 133,  17,  32
4176 DATA 207, 255, 133,  18, 232, 224,  78, 176
4184 DATA   8,  32, 207, 255, 157,   0,   2, 208
4192 DATA 243, 232, 232, 157,   0,   2, 232, 232
4200 DATA 232, 134,   5,  96,  32, 193, 125, 162
4208 DATA   2, 134, 174,  32, 198, 255,  32,  76
4216 DATA 124,  32, 241, 125,  32, 163, 181, 144
4224 DATA  68, 160,   1, 177,  92, 133,  32, 165
4232 DATA  42, 133,  31, 165,  93, 133,  34, 165
4240 DATA  92, 136, 241,  92,  24, 101,  42, 133
4248 DATA  42, 133,  33, 165,  43, 105, 255, 133
4256 DATA  43, 229,  93, 170,  56, 165,  92, 229
4264 DATA  42, 168, 176,   3, 232, 198,  34,  24
4272 DATA 101,  31, 144,   3, 198,  32,  24, 177
4280 DATA  31, 145,  33, 200, 208, 249, 230,  32
4288 DATA 230,  34, 202, 208, 242,  32, 233, 181
4296 DATA  32, 182, 180, 173,   0,   2, 208,   3
4304 DATA  76, 127, 124,  24, 165,  42, 133,  87
4312 DATA 101,   5, 133,  85, 164,  43, 132,  88
4320 DATA 144,   1, 200, 132,  86,  32,  80, 179
4328 DATA 165,  17, 164,  18, 141, 254,   1, 140
```

BLOCK 8 (CHECKSUM FOR BLOCK 8 = 51798)

```
4336 DATA 255,   1, 165,  46, 164,  47, 133,  42
4344 DATA 132,  43, 164,   5, 136, 185, 252,   1
4352 DATA 145,  92, 136,  16, 248,  32, 233, 181
4360 DATA  32, 182, 180,  76, 127, 124, 162,   1
4368 DATA  32, 198, 255,  32, 207, 255, 133, 134
4376 DATA  32, 207, 255, 133, 135,   5, 134, 201
4384 DATA  48, 240,  22, 166, 134, 165, 135,  32
4392 DATA  49, 215,  32, 207, 255,  32, 210, 255
4400 DATA 201,  13, 208, 246, 104, 104,  76,  60
4408 DATA 124,  32, 207, 255, 201,  13, 208, 249
4416 DATA  76, 204, 255, 240,  68, 201,  36, 240
4424 DATA  23,  32, 118,   0,  32,  41, 206,  32
4432 DATA  45, 201, 165,  18, 133, 252, 165,  17
4440 DATA 133, 251,  32,  36, 126,  76, 255, 179
4448 DATA 169,   0, 133,  95, 133,  96, 169,   4
4456 DATA 133,  97,  32, 196, 118, 240,  20,  32
4464 DATA 141, 215, 162,   4,   6,  96,  38,  95
4472 DATA 202, 208, 249,   5,  96, 133,  96, 198
4480 DATA  97, 208, 231,  32, 135, 207,  76, 112
4488 DATA 121,  76, 239, 126, 208, 251, 169,   2
4496 DATA  13,  76, 232, 208,   7, 208, 242, 169
4504 DATA 253,  45,  76, 232, 141,  76, 232,  76
4512 DATA 255, 179,  32, 207, 255, 133, 252, 162
4520 DATA   0,  32,  59, 112, 104, 104,  76,  23
4528 DATA 126,  32, 170, 126,  32, 207, 255, 133
4536 DATA 251, 201,   1, 208, 229,  32, 207, 255
4544 DATA 201,   4, 208, 225,  76, 223, 186,  32
4552 DATA 193, 125,  32,  76, 124,  32, 241, 125
4560 DATA 208, 248, 104, 104, 160,   0, 185,  27
4568 DATA 179, 240,  47,  32, 210, 255, 200, 208
4576 DATA 245, 169, 255, 133,  92, 169,   1, 133
4584 DATA  93, 133,  70, 166,  17, 165,  18,  32
4592 DATA  18, 117,  32,  20, 113, 201, 239, 240
4600 DATA 217,  96,  32, 170, 126,  32, 207, 255
4608 DATA 133, 251,  32, 207, 255, 133, 252,  32
4616 DATA  29, 126,  76,  60, 124, 166, 251, 165
4624 DATA 252,  32, 131, 207, 162,  32, 169,  36
4632 DATA  32,  49, 215,  32,  23, 215,  76, 223
4640 DATA 186, 240,  22, 162, 254, 134, 251, 232
4648 DATA 134, 252,  32, 170, 126,  32, 207, 255
4656 DATA  32,  57, 213, 164, 150, 240, 246, 208
4664 DATA 206,  56, 165,  42, 229,  40, 133, 251
4672 DATA 165,  43, 229,  41, 133, 252,  32,  29
4680 DATA 126,  76, 255, 179,  32, 174, 126,  32
4688 DATA 207, 255, 164, 150,   8,  32, 210, 255
4696 DATA  40, 208,  10,  32,  20, 113, 201, 239
4704 DATA 208, 237,  76, 228, 125,  76,  57, 124
4712 DATA 240,  30,  32, 174, 126, 173,  64, 232
4720 DATA  41, 251, 141,  64, 232, 169,   4, 133
4728 DATA 212,  32, 213, 240,  32,  72, 241, 169
4736 DATA   0, 133, 175, 133, 174,  76, 255, 179
4744 DATA 169,   2, 133, 174, 169,   4, 133, 176
```

BLOCK 9 (CHECKSUM FOR BLOCK 9 = 55525)

```
4752 DATA 169,   8, 133, 175,  32, 204, 255,  76
4760 DATA  60, 124, 169,   0, 240,   2, 169,   2
4768 DATA 133, 131,  32, 231, 255,  32,  60, 245
4776 DATA 166, 209, 240,  51, 134,   9, 169,   1
4784 DATA 133, 210, 169,   8, 133, 212, 169,  15
4792 DATA 133, 211, 169,   0, 133, 209,  32,  99
4800 DATA 245,  32, 204, 255, 165,   9, 133, 209
4808 DATA 169,   2, 133, 210, 169,   8, 133, 212
4816 DATA 165, 131, 133, 211,  32,  99, 245,  32
4824 DATA  30, 125, 162,   2,  76, 198, 255, 169
4832 DATA 255, 133,  55,  76,   0, 191, 167,  13
4840 DATA  78,  79,  84,  32,  66,  65,  83,  73
4848 DATA  67,  44,  32,  83,  84,  65,  82,  84
4856 DATA  61,   0, 147,  66,  65,  83,  73,  67
4864 DATA  45,  65,  73,  68,  32,  52,  13,  17
4872 DATA   0,   0, 137, 138, 141, 167,  72,  69
4880 DATA  76, 208,  65,  85,  84, 207,  66,  82
4888 DATA  69,  65, 203,  67,  72,  65,  78,  71
4896 DATA 197,  68,  69,  76,  69,  84, 197,  70
4904 DATA  76,  73,  83, 212,  68,  85,  77, 208
4912 DATA  70,  73,  78, 196,  72,  69, 216,  67
4920 DATA  82, 212,  75,  73,  76, 204,  76,  79
4928 DATA  87,  69, 210,  77,  69,  82,  71, 197
4936 DATA  82,  69,  78,  85,  77,  66,  69, 210
4944 DATA  79,  70, 198,  80,  65,  67, 203,  82
4952 DATA  69,  65, 196,  83,  67,  82,  79,  76
4960 DATA 204,  83,  84,  65,  82, 212,  84,  82
4968 DATA  65,  67, 197,  85,  80,  80,  69, 210
4976 DATA 190, 192, 175, 222,  83,  73,  90, 197
4984 DATA  85,  78,  45,  78,  69, 215,  83,  80
4992 DATA  79,  79, 204,   0, 223, 116, 220, 113
5000 DATA  13, 116,  13, 115, 169, 114, 214, 125
5008 DATA 209, 118,  13, 115,  82, 125, 120, 119
5016 DATA  70, 112, 155, 125, 123, 124, 134, 117
5024 DATA 162, 120,   1, 120,  91, 126,  32, 114
5032 DATA   9, 126, 211, 116, 164, 125, 187, 120
5040 DATA 187, 120,   5, 121,   3, 121,  48, 126
5048 DATA 177, 120, 119, 126,   0, 112,  77, 114
5056 DATA 179, 121,  49,  49,  48,  52,  56,  49
5064 DATA 170, 170, 170, 170, 170, 170, 170, 170
```

BLOCK 10 (CHECKSUM FOR BLOCK 10 = 34939)

8032 Basic-Aid Loader

```
1000 FOR J=28672 TO 32735 : READ X : POKE J,X : NEXT : SYS 7*4096
1008 DATA 169,   0, 141, 136,   3, 141, 137,   3
1016 DATA 173, 206, 127, 174, 207, 127, 224, 128
1024 DATA 176,   7, 133,  52, 134,  53,  32, 233
1032 DATA 181, 162,  15, 189,  43, 112, 149, 112
1040 DATA 202,  16, 248, 162,  19,  32,  59, 112
1048 DATA  76,  35, 114, 230, 119, 208,   2, 230
1056 DATA 120, 173, 255,   1,  76,  81, 113, 234
1064 DATA  76, 131, 113, 189, 249, 126, 240,   6
1072 DATA  32, 210, 255, 232, 208, 245,  96, 208
1080 DATA  14, 120, 162,  23, 189, 153, 211, 149
1088 DATA 112, 202,  16, 248,  76, 166, 120,  76
1096 DATA 241, 126, 169, 128, 133, 133, 169,  65
1104 DATA 133, 132, 169,  10, 133, 131, 162,   0
1112 DATA 160,  80, 169,  15, 133, 124, 177, 132
1120 DATA 129, 132,  32,  74, 113, 198, 124, 208
1128 DATA 245,  24, 169,  65, 101, 132, 133, 132
1136 DATA 144,   2, 230, 133, 198, 131, 208, 226
1144 DATA 166,  54, 165,  55, 134,  96,  32,  63
1152 DATA 113, 160,   0, 162,   5, 200, 185,   0
1160 DATA   1, 208, 250, 136, 240,  11, 202, 185
1168 DATA   0,   1,   9, 128, 157,  17, 131, 208
1176 DATA 242, 169, 160, 202,  48,   5, 157,  17
1184 DATA 131,  16, 248, 141,  22, 131, 232, 165
1192 DATA 128,  48,  38, 160,   0, 177, 119, 240
1200 DATA  20,  48,  15,  41, 191,   9, 128, 157
1208 DATA  23, 131, 232, 200, 192,   9, 208, 237
1216 DATA 240,  45, 169, 189,  44, 169, 160, 157
1224 DATA  23, 131, 232, 224,   9, 144, 246, 176
1232 DATA  30,  32,  96, 117, 200, 177, 132,  48
1240 DATA   9,  24, 105,  64, 157,  23, 131, 232
1248 DATA 208, 242,  41, 191,  44, 169, 160, 157
1256 DATA  23, 131, 232, 224,   9, 208, 246,  32
1264 DATA  20, 113, 169,  16, 174, 137,   3, 202
1272 DATA 240,  52, 141,  69, 232,  44,  77, 232
1280 DATA  80, 251, 112, 243,  36, 152, 208, 252
1288 DATA  32,  52, 113, 208,  33,  32,  52, 113
1296 DATA 240, 251, 201, 255, 240, 247,  72,  32
1304 DATA  52, 113, 201, 255, 208, 249, 169,   0
1312 DATA 133, 158, 104,  96, 173,  18, 232, 205
1320 DATA  18, 232, 208, 248, 201, 251,  96, 133
1328 DATA  95, 162, 144,  56,  32, 127, 205,  76
1336 DATA 147, 207, 230, 132, 208,   2, 230, 133
1344 DATA  96, 133, 128, 134, 129, 186, 189,   1
1352 DATA   1, 201,  15, 240,  48, 166, 120, 224
1360 DATA   2, 240,  24, 134, 135, 166, 119, 134
1368 DATA 134, 174, 137,   3, 240,  13,  48,  11
1376 DATA 201,  35, 208,   7, 132, 130,  32,  90
1384 DATA 112, 164, 130, 166, 129, 165, 128, 201
1392 DATA  58, 176, 187, 201,  32, 240,   3,  76
1400 DATA 170, 211,  76, 112,   0, 189,   2,   1
1408 DATA 201, 180, 208, 231,  32, 123, 113, 144
1416 DATA  77, 165, 128,  16,   2, 230, 119, 162
```

BLOCK 1 (CHECKSUM FOR BLOCK 1 = 53306)

```
1424 DATA   0, 134, 124, 132, 130, 164, 119, 185
1432 DATA   0,   2,  56, 253,  32, 127, 240,  19
1440 DATA 201, 128, 240,  19, 230, 124, 232, 189
1448 DATA  31, 127,  16, 250, 189,  32, 127, 208
1456 DATA 228, 240, 182, 232, 200, 208, 224, 132
1464 DATA 119, 104, 104, 165, 124,  10, 170, 189
1472 DATA 151, 127,  72, 189, 150, 127,  72,  32
1480 DATA 121, 113,  76, 112,   0,  32, 198, 116
1488 DATA 141, 136,   3,  76, 247, 114, 104, 104
1496 DATA 165, 128,  32, 246, 184, 240,  47, 173
1504 DATA 136,   3, 240,  42,  24, 165,  17, 109
1512 DATA 136,   3, 133,  96, 165,  18, 105,   0
1520 DATA  32,  63, 113, 162,   0, 169,  32, 157
1528 DATA 111,   2, 232, 189,   0,   1, 240,   6
1536 DATA 157, 111,   2, 232, 208, 245, 169,  32
1544 DATA 157, 111,   2, 232, 134, 158,  76,  34
1552 DATA 180, 208,  20, 120, 173, 208, 127, 133
1560 DATA 144, 173, 209, 127, 133, 145, 169,   2
1568 DATA 141, 140,   3,  88,  76, 255, 179,  76
1576 DATA 241, 126,  36, 152, 240,  38,  32, 129
1584 DATA 119, 162,   0, 134, 205, 134, 220, 134
1592 DATA 159, 134, 158, 240,  40, 189,   5,   1
1600 DATA 201,  43, 208,  10, 189,   6,   1, 201
1608 DATA 249, 208,   3,  32, 192, 252, 165, 151
1616 DATA 201, 155, 240, 214, 201, 155, 240, 217
1624 DATA 205, 138,   3, 240,   8, 141, 138,   3
1632 DATA 169,  16, 141, 139,   3, 173, 211, 127
1640 DATA  72, 173, 210, 127,  72,   8,  72,  72
1648 DATA  72,  76,  85, 228, 201, 255, 240, 237
1656 DATA 173, 139,   3, 240,   5, 206, 139,   3
1664 DATA 208, 227, 206, 140,   3, 208, 222, 169
1672 DATA   2, 141, 140,   3, 165, 158, 208, 213
1680 DATA 169,   0, 133, 151, 169,   2, 133, 168
1688 DATA 208, 203,  32,  52, 116, 165,  92, 166
1696 DATA  93, 133,  33, 134,  34,  32, 163, 181
1704 DATA 165,  92, 166,  93, 144,  10, 160,   1
1712 DATA 177,  92, 240,   4, 170, 136, 177,  92
1720 DATA 133, 119, 134, 120, 165,  33,  56, 229
1728 DATA 119, 170, 165,  34, 229, 120, 168, 176
1736 DATA  30, 138,  24, 101,  42, 133,  42, 152
1744 DATA 101,  43, 133,  43, 160,   0, 177, 119
1752 DATA 145,  33, 200, 208, 249, 230, 120, 230
1760 DATA  34, 165,  43, 197,  34, 176, 239,  32
1768 DATA 182, 180, 165,  31, 166,  32,  24, 105
1776 DATA   2, 133,  42, 144,   1, 232, 134,  43
1784 DATA  32, 233, 181,  76, 255, 179,  32, 251
1792 DATA 180,  32, 112,   0, 133, 128, 162,   0
1800 DATA 134,  70,  32, 244, 115, 165, 124, 201
1808 DATA   3, 208,   7, 162,   2, 134,  70,  32
1816 DATA 244, 115,  32, 112,   0, 240,   3,  32
1824 DATA 245, 190,  32,  52, 116, 165,  92, 166
1832 DATA  93, 133, 119, 134, 120,  32, 223, 186
```

BLOCK 2 (CHECKSUM FOR BLOCK 2 = 51652)

```
1840 DATA 200,  11, 200, 152,  24, 101, 119, 133
1848 DATA 119, 144,   2, 230, 120,  32, 196, 118
1856 DATA 240,   5,  32,  94, 116, 176,   3,  76
1864 DATA 247, 114, 132,  82, 230,  82, 164,  82
1872 DATA 166,  46, 165,  47, 133, 128, 177, 119
1880 DATA 240, 216, 221,   0,   2, 208, 237, 232
1888 DATA 200, 198, 128, 208, 241, 136, 132,   5
1896 DATA 132, 129, 165,  70, 240,  91,  32, 113
1904 DATA 116, 165,  49,  56, 229,  47, 133, 180
1912 DATA 240,  40, 200, 240, 202, 177, 119, 208
1920 DATA 249,  24, 152, 101, 180, 201,   2, 144
1928 DATA  64, 201,  75, 176,  60, 165, 180,  16
1936 DATA   2, 198, 128,  24, 101,   5, 133, 129
1944 DATA 176,   5,  32, 171, 116, 240,   3,  32
1952 DATA 147, 116, 165, 129,  56, 229,  49, 168
1960 DATA 200, 165,  49, 240,  15, 133, 130, 166
1968 DATA  48, 189,   0,   2, 145, 119, 232, 200
1976 DATA 198, 130, 208, 245,  24, 165,  42, 101
1984 DATA 180, 133,  42, 165,  43, 101, 128, 133
1992 DATA  43, 165, 119, 166, 120, 133,  92, 134
2000 DATA  93, 166,  64, 165,  65,  32,  18, 117
2008 DATA  32,  20, 113, 201, 239, 240, 156, 164
2016 DATA 129,  76,  90, 115, 164, 119, 200, 148
2024 DATA  46, 169,   0, 149,  47, 185,   0,   2
2032 DATA 240,  47, 197, 128, 240,   5, 246,  47
2040 DATA 200, 208, 242, 132, 119,  96, 208,  33
2048 DATA 141,   3,   2, 142,   4,   2, 140,   5
2056 DATA   2,   8, 104, 141,   2,   2, 169, 179
2064 DATA  72, 169, 255,  72,  56,  76, 114, 212
2072 DATA 201, 171, 240,   4, 201,  45, 208,   1
2080 DATA  96,  76, 241, 126, 144,   5, 240,   3
2088 DATA  32,  40, 116,  32, 246, 184,  32, 163
2096 DATA 181,  32, 118,   0, 240,  11,  32,  40
2104 DATA 116,  32, 112,   0,  32, 246, 184, 208
2112 DATA 224, 165,  17,   5,  18, 208,   6, 169
2120 DATA 255, 133,  17, 133,  18,  96,  32, 196
2128 DATA 118, 133,  64,  32, 196, 118, 133,  65
2136 DATA 165,  17, 197,  64, 165,  18, 229,  65
2144 DATA  96, 165, 119, 133,  31, 165, 120, 133
2152 DATA  32, 165,  42, 133,  33, 165,  43, 133
2160 DATA  34,  96, 165,  31, 197,  33, 208,   4
2168 DATA 165,  32, 197,  34,  96, 230,  31, 208
2176 DATA   2, 230,  32, 164,   5, 200, 177,  31
2184 DATA 164, 129, 200, 145,  31,  32, 130, 116
2192 DATA 208, 235,  96, 165,  33, 208,   2, 198
2200 DATA  34, 198,  33, 164,   5, 177,  33, 164
2208 DATA 129, 145,  33,  32, 130, 116, 208, 235
2216 DATA  96, 201,  34, 208,   8,  72, 165,   9
2224 DATA  73, 128, 133,   9, 104,  96,  32, 246
2232 DATA 184, 165,  18, 208,   4, 165,  17,  16
2240 DATA   2, 169, 127,  96,  32, 198, 116, 141
2248 DATA 137,   3,  76, 255, 179,  76, 241, 126
```

BLOCK 3 (CHECKSUM FOR BLOCK 3 = 48818)

```
2256 DATA 208, 251,  32, 232, 116,  76, 255, 179
2264 DATA  32, 223, 186, 133,  70, 166,  40, 165
2272 DATA  41, 134,  92, 133,  93, 160,   0, 177
2280 DATA  92, 170, 200, 177,  92, 208,   3,  76
2288 DATA 112,   0, 197, 135, 144, 235, 228, 134
2296 DATA 144, 231, 200, 177,  92, 170, 200, 177
2304 DATA  92,  44, 160,   0, 132, 124, 132,   9
2312 DATA  32, 131, 207, 169,  32, 164, 124,  41
2320 DATA 127,  32, 210, 255,  32, 185, 116, 169
2328 DATA   0, 133, 159, 200,  36,  70,  16,  23
2336 DATA 166,  93, 152,  56, 101,  92, 144,   1
2344 DATA 232, 228, 135, 144,  10, 197, 134, 144
2352 DATA   6, 169,   1, 133,  70, 133, 159, 177
2360 DATA  92, 240,  17,  16, 212, 201, 255, 240
2368 DATA 208,  36,   9,  48, 204, 132, 124,  32
2376 DATA 122, 117,  48, 193,  76, 223, 186,  96
2384 DATA 162, 176, 160, 177, 134, 133, 132, 132
2392 DATA  56, 233, 127, 170, 160,   0, 202, 240
2400 DATA 238,  32,  74, 113, 177, 132,  16, 249
2408 DATA  48, 244,  32,  96, 117, 200, 177, 132
2416 DATA  48, 221,  32, 210, 255, 208, 246,  32
2424 DATA 246, 184, 164,  17, 166,  18, 152,   5
2432 DATA  18, 208,   2, 160, 100, 132,  50, 134
2440 DATA  51, 162,   0, 161, 119, 208,   4, 169
2448 DATA  10, 208,  10,  32, 245, 190,  32, 246
2456 DATA 184, 165,  17, 166,  18, 133,  48, 134
2464 DATA  49,  32,  34, 182,  32, 196, 118,  32
2472 DATA 196, 118, 208,  33,  32, 166, 118,  32
2480 DATA 196, 118,  32, 196, 118, 208,   3,  76
2488 DATA 247, 114,  32, 196, 118, 165,  96, 145
2496 DATA 119,  32, 196, 118, 165,  95, 145, 119
2504 DATA  32, 177, 118, 240, 226,  32, 196, 118
2512 DATA  32, 196, 118,  32, 196, 118, 201,  34
2520 DATA 208,  11,  32, 196, 118, 240, 197, 201
2528 DATA  34, 208, 247, 240, 238, 170, 240, 188
2536 DATA  16, 233, 162,   4, 221,  27, 127, 240
2544 DATA   5, 202, 208, 248, 240, 221, 165, 119
2552 DATA 133,  56, 165, 120, 133,  57,  32, 112
2560 DATA   0, 176, 211,  32, 246, 184,  32,  80
2568 DATA 118, 165,  57, 133, 120, 165,  56, 133
2576 DATA 119, 160,   0, 162,   0, 189,   1,   1
2584 DATA 201,  48, 144,  17,  72,  32, 112,   0
2592 DATA 144,   3,  32, 124, 118, 104, 160,   0
2600 DATA 145, 119, 232, 208, 232,  32, 112,   0
2608 DATA 176,   8,  32, 139, 118,  32, 118,   0
2616 DATA 144, 248, 201,  44, 240, 184, 208, 150
2624 DATA  32, 166, 118,  32, 196, 118,  32, 196
2632 DATA 118, 208,   8, 169, 255, 133,  96, 133
2640 DATA  95,  48,  14,  32, 196, 118, 197,  17
2648 DATA 208,  10,  32, 196, 118, 197,  18, 208
2656 DATA   6,  76, 135, 207,  32, 196, 118,  32
2664 DATA 177, 118, 240, 215,  32, 156, 118, 230
```

BLOCK 4 (CHECKSUM FOR BLOCK 4 = 51821)

```
2672 DATA 129,  32, 171, 116, 230,  42, 208,   2
2680 DATA 230,  43,  96,  32, 156, 118, 198, 129
2688 DATA  32, 147, 116, 165,  42, 208,   2, 198
2696 DATA  43, 198,  42,  96,  32, 113, 116, 160
2704 DATA   0, 132,   5, 132, 129,  96, 165,  50
2712 DATA 133,  96, 165,  51, 133,  95,  76,  34
2720 DATA 182, 165,  96,  24, 101,  48, 133,  96
2728 DATA 165,  95, 101,  49, 133,  95,  32, 196
2736 DATA 118, 208, 251,  96, 160,   0, 230, 119
2744 DATA 208,   2, 230, 120, 177, 119,  96,  76
2752 DATA 255, 179, 208,  89, 165,  42, 133,  92
2760 DATA 165,  43, 133,  93, 165,  92, 197,  44
2768 DATA 165,  93, 229,  45, 176, 233, 160,   0
2776 DATA 132,  33, 200, 177,  92,  10, 102,  33
2784 DATA  74, 153,  66,   0, 136,  16, 244,  36
2792 DATA  33, 240,  30,  16,  51,  80,  89,  32
2800 DATA 108, 119, 162,  37, 169,  61,  32,  49
2808 DATA 215, 160,   2, 177,  92,  72, 200, 177
2816 DATA  92, 168, 104,  32, 188, 196,  76,  39
2824 DATA 119,  32, 108, 119, 169,  61,  32, 210
2832 DATA 255,  32, 185, 194,  32, 216, 204,  32
2840 DATA 141, 207,  76,  85, 119,  76, 241, 126
2848 DATA  32, 108, 119, 162,  36, 169,  61,  32
2856 DATA  49, 215, 169,  34,  32, 210, 255, 160
2864 DATA   4, 177,  92, 133,  32, 136, 177,  92
2872 DATA 133,  31, 136, 177,  92,  32,  35, 187
2880 DATA 169,  34,  32, 210, 255,  32, 223, 186
2888 DATA  32, 225, 255,  32,  20, 113,  24, 165
2896 DATA  92, 105,   7, 133,  92, 144,   2, 230
2904 DATA  93,  76, 220, 118, 165,  66,  32, 210
2912 DATA 255, 165,  67, 240,   3,  32, 210, 255
2920 DATA  96, 208, 178,  32, 129, 119,  76, 255
2928 DATA 179, 169, 128, 133,  32, 169,   0, 133
2936 DATA  31, 169,   4, 133, 212,  32, 213, 240
2944 DATA  32,  72, 241, 169,  25, 133,  34, 169
2952 DATA  13, 133,   9,  32, 158, 241, 169,  17
2960 DATA 174,  76, 232, 224,  12, 208,   2, 169
2968 DATA 145,  32, 158, 241, 160,   0, 177,  31
2976 DATA  41, 127, 170, 177,  31,  69,   9,  16
2984 DATA  11, 177,  31, 133,   9,  41, 128,  73
2992 DATA 146,  32, 158, 241, 138, 201,  32, 176
3000 DATA   4,   9,  64, 208,  14, 201,  64, 144
3008 DATA  10, 201,  96, 176,   4,   9, 128, 208
3016 DATA   2,  73, 192,  32, 158, 241, 200, 192
3024 DATA  80, 144, 203, 165,  31, 105,  79, 133
3032 DATA  31, 144,   2, 230,  32, 198,  34, 208
3040 DATA 166, 169,  13,  32, 158, 241,  32, 158
3048 DATA 241,  76, 185, 241,  76, 241, 126,  76
3056 DATA 247, 114, 208, 248, 169,   1, 133, 132
3064 DATA 133,  31, 169,   4, 133, 133, 133,  32
3072 DATA 160,   3, 177, 132, 145,  31, 136,  16
3080 DATA 249, 200, 132,   9, 177,  31, 200,  17
```

BLOCK 5 (CHECKSUM FOR BLOCK 5 = 48292)

```
3088 DATA  31, 240, 220, 160,   4, 177, 132, 201
3096 DATA  58, 208,   6, 200, 177, 132, 240,  48
3104 DATA 136, 177, 132, 201, 143, 240,  41,  36
3112 DATA   9, 112,   4, 201,  58, 240,   8, 201
3120 DATA  32, 208,   9,  36,   9,  48,   5,  32
3128 DATA  74, 113, 208, 229, 170, 169,  64,   5
3136 DATA   9, 133,   9, 138, 145,  31, 200, 201
3144 DATA   0, 240,  46,  32, 185, 116, 208, 209
3152 DATA 136,  36,   9, 112,  13, 160,   0,  24
3160 DATA 165, 132, 105,   4, 133, 132, 144,   2
3168 DATA 230, 133, 177, 132, 240,   5,  32,  74
3176 DATA 113, 208, 247,  36,   9,  80,   5, 145
3184 DATA  31, 200,  16,   5,  32,  74, 113, 208
3192 DATA 135, 152, 170, 160,   1, 177,  31, 133
3200 DATA 133, 136, 177,  31, 133, 132,  24, 138
3208 DATA 101,  31, 133,  31, 144,   2, 230,  32
3216 DATA  76,  16, 120, 208,  10, 120, 162,  12
3224 DATA  32, 224, 252,  88,  76, 255, 179,  76
3232 DATA 241, 126, 208, 251, 169,   4, 141,   2
3240 DATA   4,  76, 247, 114, 160,   0, 132, 131
3248 DATA 132,  13, 201,   0, 240,  32, 201,  36
3256 DATA 240,  64, 169,   8, 133, 212,  32, 213
3264 DATA 240, 169, 111,  32,  67, 241, 177, 119
3272 DATA 240,   6,  32, 158, 241, 200, 208, 246
3280 DATA  32, 185, 241,  76, 255, 179, 169,   8
3288 DATA 133, 212,  32, 210, 240, 169, 111,  32
3296 DATA 147, 241,  32, 192, 241, 201,  13, 240
3304 DATA   5,  32, 210, 255, 208, 244,  32, 174
3312 DATA 241,  76, 112, 121, 169,  94, 133, 131
3320 DATA 160,   0, 200, 177, 119, 208, 251, 132
3328 DATA 209, 165, 119, 133, 218, 165, 120, 133
3336 DATA 219, 169,   8, 133, 212, 165, 131, 208
3344 DATA  85, 169,  96, 133, 211,  32, 204, 255
3352 DATA  32, 165, 244,  32, 210, 240, 165, 211
3360 DATA  32, 147, 241,  32, 223, 186, 169,   0
3368 DATA 133, 150, 160,   3, 132, 209,  32, 192
3376 DATA 241, 170, 164, 150, 208,  39,  32, 192
3384 DATA 241, 164, 150, 208,  32, 198, 209, 208
3392 DATA 237,  32, 131, 207,  32,  62, 187,  32
3400 DATA 192, 241, 240,   5,  32, 210, 255, 208
3408 DATA 246,  32, 223, 186, 160,   2,  32,  20
3416 DATA 113, 201, 239, 208, 207,  32, 206, 243
3424 DATA  32, 223, 186,  76, 255, 179, 169,   0
3432 DATA 133, 150, 133, 157,  32,  86, 243, 165
3440 DATA 150,  41, 191, 208,  40,  44, 128, 241
3448 DATA  16,   6, 230, 201, 208,   2, 230, 202
3456 DATA 165, 202, 133,  43, 165, 201, 133,  42
3464 DATA  32, 233, 181,  32, 182, 180, 165, 131
3472 DATA 201,  94, 240,   3,  76, 255, 179,  32
3480 DATA  34, 182,  76,  74, 183,  76,  37, 244
3488 DATA  76,   0, 230, 165, 158, 240, 249, 165
3496 DATA 198,  56, 229, 226, 201,   2, 176, 240
```

BLOCK 6 (CHECKSUM FOR BLOCK 6 = 54770)

```
3504 DATA 173, 111,   2,  41, 127, 201,  17, 208
3512 DATA 231, 173,   1,   4,  13,   2,   4, 240
3520 DATA 223, 165, 224, 141, 132,   3, 165, 226
3528 DATA 141, 134,   3, 165, 213, 141, 135,   3
3536 DATA 165, 225, 141, 133,   3,  32, 225, 123
3544 DATA 173, 111,   2,  48,  93, 166, 216, 228
3552 DATA 225, 208,  84, 142, 131,   3, 142, 129
3560 DATA   3, 202,  48,  60, 188, 110, 231, 189
3568 DATA  85, 231,  24, 101, 226, 144,   1, 200
3576 DATA 133, 119, 132, 120,  32, 240, 123, 176
3584 DATA 232,  32, 246, 184, 230,  17, 208,   2
3592 DATA 230,  18,  32, 163, 181, 176,  16, 208
3600 DATA  14,  32, 242, 122,  32, 242, 122, 169
3608 DATA   0, 133,  17, 133,  18, 240, 235,  32
3616 DATA 242, 122, 206, 129,   3,  32,  46, 123
3624 DATA 162,   0, 189, 112,   2, 157, 111,   2
3632 DATA 232, 228, 158, 208, 245, 198, 158,  76
3640 DATA   0, 230, 166, 216, 228, 224, 208, 247
3648 DATA 142, 129,   3, 142, 131,   3, 160,   0
3656 DATA 140, 131,   3, 202, 232, 228, 225, 176
3664 DATA 215, 188, 110, 231, 189,  85, 231,  24
3672 DATA 101, 226, 144,   1, 200, 133, 119, 132
3680 DATA 120,  32, 240, 123, 176, 230,  32, 246
3688 DATA 184,  32, 163, 181, 165,  92, 166,  93
3696 DATA 197,  40, 208,  18, 228,  41, 208,  14
3704 DATA  32, 196, 122,  32, 196, 122, 169, 255
3712 DATA 133,  17, 133,  18, 208, 227, 133, 218
3720 DATA 202, 134, 219, 160, 255, 200, 177, 218
3728 DATA 170, 208, 250, 200, 177, 218, 197,  92
3736 DATA 208, 246, 200, 177, 218, 197,  93, 208
3744 DATA 239, 136, 152,  24, 101, 218, 133,  92
3752 DATA 165, 219, 105,   0, 133,  93,  32, 196
3760 DATA 122,  76,  53, 122, 174, 133,   3, 232
3768 DATA 202, 189,  85, 231, 133,  33, 189, 110
3776 DATA 231, 133,  34, 172, 134,   3, 136, 236
3784 DATA 132,   3, 240,  68, 189,  84, 231, 133
3792 DATA  31, 189, 109, 231, 133,  32, 200, 177
3800 DATA  31, 145,  33, 204, 135,   3, 144, 246
3808 DATA 176, 214, 174, 132,   3, 202, 232, 189
3816 DATA  85, 231, 133,  33, 189, 110, 231, 133
3824 DATA  34, 172, 134,   3, 136, 236, 133,   3
3832 DATA 176,  22, 189,  86, 231, 133,  31, 189
3840 DATA 111, 231, 133,  32, 200, 177,  31, 145
3848 DATA  33, 204, 135,   3, 144, 246, 176, 214
3856 DATA 169,  32, 200, 145,  33, 204, 135,   3
3864 DATA 144, 248, 189, 110, 231,  96, 162,   0
3872 DATA 134,   9, 166, 226, 142, 130,   3, 174
3880 DATA 129,   3, 189,  85, 231, 188, 110, 231
3888 DATA  24, 101, 226, 144,   1, 200, 133, 218
3896 DATA 132, 219,  32,  22, 124, 133,  96,  32
3904 DATA  25, 124,  32,  63, 113,  32, 147, 123
3912 DATA 162,   0, 189,   1,   1, 240,   6,  32
```

BLOCK 7 (CHECKSUM FOR BLOCK 7 = 55381)

```
3920 DATA 149, 123, 232, 208, 245, 169,  32,  41
3928 DATA 127,  32, 149, 123,  32,  25, 124,   8
3936 DATA  32, 185, 116,  40,  48,   4, 240,  52
3944 DATA 208, 239, 201, 255, 240, 235,  36,   9
3952 DATA  48, 231,  32,  96, 117, 200, 177, 132
3960 DATA  48, 221, 132, 135,  32, 149, 123, 164
3968 DATA 135, 208, 242, 169,  32,  32,  36, 124
3976 DATA 160,   0, 145, 218, 230, 218, 208,   2
3984 DATA 230, 219, 238, 130,   3, 173, 130,   3
3992 DATA 197, 213, 240,   1,  96, 173, 131,   3
4000 DATA 240,  17, 172, 129,   3, 196, 225, 240
4008 DATA  39, 138,  72, 206, 133,   3,  32, 242
4016 DATA 122, 176,   8, 138,  72, 238, 132,   3
4024 DATA  32, 196, 122, 133, 219, 189,  85, 231
4032 DATA  24, 101, 226, 133, 218, 144,   2, 230
4040 DATA 219, 165, 226, 141, 130,   3, 104, 170
4048 DATA  96, 165, 170, 240,  10, 160,   0, 132
4056 DATA 170, 164, 198, 165, 169, 145, 196,  96
4064 DATA 164, 226, 140, 130,   3, 160,   0, 240
4072 DATA  16, 230, 119, 208,   2, 230, 120, 238
4080 DATA 130,   3, 173, 130,   3, 197, 213, 176
4088 DATA 230, 177, 119, 201,  58, 176, 224, 201
4096 DATA  32, 240, 230,  76, 170, 211,  32,  25
4104 DATA 124, 230,  92, 208,   2, 230,  93, 160
4112 DATA   0, 177,  92,  96, 133, 134,  41, 127
4120 DATA 201,  32,   8,  41,  63,  40, 176,   2
4128 DATA   9, 128,  36, 134,  16,   2,   9,  64
4136 DATA  96, 104, 104,  32, 223, 186,  32, 204
4144 DATA 255, 169,   1,  32, 226, 242, 169,   2
4152 DATA  32, 226, 242,  76, 255, 179,  32, 207
4160 DATA 255,  32, 207, 255, 240, 227, 165, 150
4168 DATA 208, 223, 162, 255,  32, 207, 255, 133
4176 DATA  17,  32, 207, 255, 133,  18, 232, 224
4184 DATA  78, 176,   8,  32, 207, 255, 157,   0
4192 DATA   2, 208, 243, 232, 232, 157,   0,   2
4200 DATA 232, 232, 232, 134,   5,  96,  32, 195
4208 DATA 125, 162,   2, 134, 174,  32, 198, 255
4216 DATA  32,  78, 124,  32, 243, 125,  32, 163
4224 DATA 181, 144,  68, 160,   1, 177,  92, 133
4232 DATA  32, 165,  42, 133,  31, 165,  93, 133
4240 DATA  34, 165,  92, 136, 241,  92,  24, 101
4248 DATA  42, 133,  42, 133,  33, 165,  43, 105
4256 DATA 255, 133,  43, 229,  93, 170,  56, 165
4264 DATA  92, 229,  42, 168, 176,   3, 232, 198
4272 DATA  34,  24, 101,  31, 144,   3, 198,  32
4280 DATA  24, 177,  31, 145,  33, 200, 208, 249
4288 DATA 230,  32, 230,  34, 202, 208, 242,  32
4296 DATA 233, 181,  32, 182, 180, 173,   0,   2
4304 DATA 208,   3,  76, 129, 124,  24, 165,  42
4312 DATA 133,  87, 101,   5, 133,  85, 164,  43
4320 DATA 132,  88, 144,   1, 200, 132,  86,  32
4328 DATA  80, 179, 165,  17, 164,  18, 141, 254
```

BLOCK 8 (CHECKSUM FOR BLOCK 8 = 52880)

```
4336 DATA   1, 140, 255,   1, 165,  46, 164,  47
4344 DATA 133,  42, 132,  43, 164,   5, 136, 185
4352 DATA 252,   1, 145,  92, 136,  16, 248,  32
4360 DATA 233, 181,  32, 182, 180,  76, 129, 124
4368 DATA 162,   1,  32, 198, 255,  32, 207, 255
4376 DATA 133, 134,  32, 207, 255, 133, 135,   5
4384 DATA 134, 201,  48, 240,  22, 166, 134, 165
4392 DATA 135,  32,  49, 215,  32, 207, 255,  32
4400 DATA 210, 255, 201,  13, 208, 246, 104, 104
4408 DATA  76,  62, 124,  32, 207, 255, 201,  13
4416 DATA 208, 249,  76, 204, 255, 240,  68, 201
4424 DATA  36, 240,  23,  32, 118,   0,  32,  41
4432 DATA 206,  32,  45, 201, 165,  18, 133, 252
4440 DATA 165,  17, 133, 251,  32,  38, 126,  76
4448 DATA 255, 179, 169,   0, 133,  95, 133,  96
4456 DATA 169,   4, 133,  97,  32, 196, 118, 240
4464 DATA  20,  32, 141, 215, 162,   4,   6,  96
4472 DATA  38,  95, 202, 208, 249,   5,  96, 133
4480 DATA  96, 198,  97, 208, 231,  32, 135, 207
4488 DATA  76, 112, 121,  76, 241, 126, 208, 251
4496 DATA 169,   2,  13,  76, 232, 208,   7, 208
4504 DATA 242, 169, 253,  45,  76, 232, 141,  76
4512 DATA 232,  76, 255, 179,  32, 207, 255, 133
4520 DATA 252, 162,   0,  32,  59, 112, 104, 104
4528 DATA  76,  25, 126,  32, 172, 126,  32, 207
4536 DATA 255, 133, 251, 201,   1, 208, 229,  32
4544 DATA 207, 255, 201,   4, 208, 225,  76, 223
4552 DATA 186,  32, 195, 125,  32,  78, 124,  32
4560 DATA 243, 125, 208, 248, 104, 104, 160,   0
4568 DATA 185,  27, 179, 240,  47,  32, 210, 255
4576 DATA 200, 208, 245, 169, 255, 133,  92, 169
4584 DATA   1, 133,  93, 133,  70, 166,  17, 165
4592 DATA  18,  32,  18, 117,  32,  20, 113, 201
4600 DATA 239, 240, 217,  96,  32, 172, 126,  32
4608 DATA 207, 255, 133, 251,  32, 207, 255, 133
4616 DATA 252,  32,  31, 126,  76,  62, 124, 166
4624 DATA 251, 165, 252,  32, 131, 207, 162,  32
4632 DATA 169,  36,  32,  49, 215,  32,  23, 215
4640 DATA  76, 223, 186, 240,  22, 162, 254, 134
4648 DATA 251, 232, 134, 252,  32, 172, 126,  32
4656 DATA 207, 255,  32,  57, 213, 164, 150, 240
4664 DATA 246, 208, 206,  56, 165,  42, 229,  40
4672 DATA 133, 251, 165,  43, 229,  41, 133, 252
4680 DATA  32,  31, 126,  76, 255, 179,  32, 176
4688 DATA 126,  32, 207, 255, 164, 150,   8,  32
4696 DATA 210, 255,  40, 208,  10,  32,  20, 113
4704 DATA 201, 239, 208, 237,  76, 230, 125,  76
4712 DATA  59, 124, 240,  30,  32, 176, 126, 173
4720 DATA  64, 232,  41, 251, 141,  64, 232, 169
4728 DATA   4, 133, 212,  32, 213, 240,  32,  72
4736 DATA 241, 169,   0, 133, 175, 133, 174,  76
4744 DATA 255, 179, 169,   2, 133, 174, 169,   4
```

BLOCK 9 (CHECKSUM FOR BLOCK 9 = 55393)

```
4752 DATA 133, 176, 169,   8, 133, 175,  32, 204
4760 DATA 255,  76,  62, 124, 169,   0, 240,   2
4768 DATA 169,   2, 133, 131,  32, 231, 255,  32
4776 DATA  60, 245, 166, 209, 240,  51, 134,   9
4784 DATA 169,   1, 133, 210, 169,   8, 133, 212
4792 DATA 169,  15, 133, 211, 169,   0, 133, 209
4800 DATA  32,  99, 245,  32, 204, 255, 165,   9
4808 DATA 133, 209, 169,   2, 133, 210, 169,   8
4816 DATA 133, 212, 165, 131, 133, 211,  32,  99
4824 DATA 245,  32,  32, 125, 162,   2,  76, 198
4832 DATA 255, 169, 255, 133,  55,  76,   0, 191
4840 DATA 167,  13,  78,  79,  84,  32,  66,  65
4848 DATA  83,  73,  67,  44,  32,  83,  84,  65
4856 DATA  82,  84,  61,   0, 147,  66,  65,  83
4864 DATA  73,  67,  45,  65,  73,  68,  32,  52
4872 DATA  13,  17,   0,   0, 137, 138, 141, 167
4880 DATA  72,  69,  76, 208,  65,  85,  84, 207
4888 DATA  66,  82,  69,  65, 203,  67,  72,  65
4896 DATA  78,  71, 197,  68,  69,  76,  69,  84
4904 DATA 197,  70,  76,  73,  83, 212,  68,  85
4912 DATA  77, 208,  70,  73,  78, 196,  72,  69
4920 DATA 216,  67,  82, 212,  75,  73,  76, 204
4928 DATA  76,  79,  87,  69, 210,  77,  69,  82
4936 DATA  71, 197,  82,  69,  78,  85,  77,  66
4944 DATA  69, 210,  79,  70, 198,  80,  65,  67
4952 DATA 203,  82,  69,  65, 196,  83,  67,  82
4960 DATA  79,  76, 204,  83,  84,  65,  82, 212
4968 DATA  84,  82,  65,  67, 197,  85,  80,  80
4976 DATA  69, 210, 190, 192, 175, 222,  83,  73
4984 DATA  90, 197,  85,  78,  45,  78,  69, 215
4992 DATA  83,  80,  79,  79, 204,   0, 223, 116
5000 DATA 220, 113,  13, 116,  13, 115, 169, 114
5008 DATA 216, 125, 209, 118,  13, 115,  84, 125
5016 DATA 120, 119,  70, 112, 157, 125, 125, 124
5024 DATA 134, 117, 162, 120,   1, 120,  93, 126
5032 DATA  32, 114,  11, 126, 211, 116, 166, 125
5040 DATA 187, 120, 187, 120,   5, 121,   3, 121
5048 DATA  50, 126, 177, 120, 121, 126,   0, 112
5056 DATA  77, 114, 179, 121,  49,  48,  49,  57
5064 DATA  56,  49, 170, 170, 170, 170, 170, 170
```

BLOCK 10 (CHECKSUM FOR BLOCK 10 = 34935)