

X15	07B4	X16	07E4	X17	081C	X19	086B
X19A	08E5	X20	0909	X21	094E	X22	098E
X23	09CA	X24	0A2C	X25	0A83	X26	0AE8
X27	0B30	X4	01CA	X5	0221	X5A	0286
X6	02DD	X6A	030A	X7	034A	X8	0376
X9	03EF	X9A	0439	X9B	04A4	XCH3P1	0639
XFER	0171	XFER1	0179	Z20001	101C	Z20002	1120
Z20003	0FBD	Z20004	0FBD	Z20005	0FBD	Z20006	0FBD
Z20007	0F37	Z20008	0F37	Z20009	0FBD	Z2000A	10D6
Z2000E	0FBD	Z2000C	101C	Z2000D	0FBD	Z2000E	0002
Z2000F	0002	Z20010	0006	Z20011	0002	Z20012	0003
Z20013	0002	Z20014	0002	Z20015	0002	Z20016	0002
Z20017	0005	Z20018	0004	Z20019	0002	Z2001A	0007
Z2001B	0004	Z2001C	0004	Z2001D	0003	Z2001E	0002
Z2001F	0006	Z20020	0005	Z20021	0002	Z20022	0003
Z20023	0006	Z20024	0005	Z20025	0002	Z20026	0003
Z20027	0005	Z20028	0002	Z20029	0002	Z2002A	0005
Z2002B	0003	Z2002C	0003	Z2002D	0007	Z2002E	0003
Z2002F	0004	Z20030	0003	Z20031	0002	Z20032	0005
Z20033	0002	Z20034	0003	Z20035	0002	Z20036	0003
Z20037	0003	Z20038	0002	Z20039	0006	Z2003A	0003
Z2003E	0003	Z2003C	0007	Z2003D	0003	Z2003E	0002
Z2003F	0002	Z20040	0002	Z20041	0002	Z20042	0002
Z20043	0002	Z20044	0002	Z20045	0002	Z20046	0002
Z20047	0002	Z20048	0002	Z20049	0002	Z2004A	0002
Z2004E	0002	Z2004C	0002	Z2004D	0002	Z2004E	0003
Z2004F	0002	Z20050	0003	Z20051	0002	Z20052	0003
Z20053	0003	Z20054	0003	Z20055	0004	Z20056	0004
Z20057	0008	Z20058	0003	Z20059	0002	Z2005A	0003
Z2005B	0005	\$0	002A	\$0	0420	\$0	076F
\$0	0996	\$0	0E91	\$1	0043	\$1	01C6
\$1	023D	\$1	0397	\$1	0443	\$1	05FA
\$1	06E7	\$1	0772	\$1	0838	\$1	0930
\$1	097E	\$1	099C	\$1	09FD	\$1	0A4C
\$1	0A8F	\$1	0E12	\$1	0B46	\$1	0BA3
\$1	0BE4	\$1	0F7B	\$10	0AA9	\$2	01FF
\$2	025A	\$2	03A8	\$2	0454	\$2	0707
\$2	07B2 *	\$2	0842	\$2	09A3	\$2	0ADD
\$2	0E28	\$2	0EFC	\$2	0F8F	\$3	0261
\$3	03D4	\$3	04AC	\$3	084A	\$3	0ADF
\$3	0C11	\$3	0F9C	\$4	03F1	\$4	0857
\$4	0FA0	\$5	085E	\$AB0R	06C6	\$ADD	08E3
\$ADD1	08FF	\$CALB	8000	\$CR	07D8	\$DOWN	0885
\$END	04C3	\$END	06AC	\$END	0E4C	\$ENT1	04A6
\$ENTE	068B	\$ENTE	07B6	\$EXIT	0402	\$FAIL	04FB
\$JMPB	4000	\$LETR	067C	\$LOOP	002C	\$LOOP	00DB
\$LOOP	0203	\$LOOP	0241	\$LOOP	03B6	\$LOOP	0460
\$LOOP	066C	\$LOOP	06F7	\$LOOP	09D9	\$LOOP	0AD1
\$LOOP	0B38	\$LOOP	0B68	\$LOOP	0BAA	\$MAYB	050D
\$MOVE	086F	\$MSG	0247	\$NEO	00EE	\$NOT	0628
\$OK	051A	\$OK	0688	\$OR	0616	\$POS	043B
\$PRNT	01EF	\$REDO	093C	\$REDO	0AF6	\$RET	06D6
\$RUB	07CE	\$SCAN	00C7	\$SHIF	0692	\$SHIF	0714
\$SKIP	0664	\$TSTB	2000	\$UP	0895	\$UP1	08A1
\$UP2	08C2	\$UP3	08C4	\$UP4	08D4	\$XH	07C8
\$XU	07A6						

Chapter 4

The Great Experiment Floppy ROM™ #1 Robert Uiterwyk's 6800 4K BASIC

By William Turner
and William Blomgren

The one thing that characterizes the microcomputer industry, is the number of innovations that appear on a daily basis. During the infancy of the industry, *Interface Age* in May of 1977 kept pace with the introduction of portable software; in the form of a plastic record called a Floppy ROM™.

This article is a chronicle of the first Floppy ROM™ with Robert Uiterwyk's 4K BASIC for the 6800. What made this first record significant was that it was the first time major systems software was published in machine readable form, that just about anyone could use. More importantly it opened up an era of cheap software distribution.

WHY "PLATTER BASIC" ON A FLOPPY-ROM™?

This is really a two-part question. First, why BASIC at all? The need for high level software is fairly self-evident, and BASIC will be explained in a second article later in this issue. For now, let's just settle with "Why a Floppy-ROM™ or platter?"

Low cost software distribution is a recurring problem. It just appears that "low cost" and "high reliability" just don't seem to mix. Shipping programs in ROM, at several dollars per copy, is too expensive. All magnetic media cost too much, are fairly bulky, and are fragile. Machine readable print requires a fair amount of hardware.

NO ERROR LINES
SOURCE CHECKSUM = 33FE
INPUT FILE 1:NIBL1127 SRC

The bar code reader that we have located costs well over \$100. Most people do not have a bar code reader, nor even access to one, but many do have some sort of record player. A conventional record costs a fair amount, and is relatively fragile. The "sound Sheet" or floppy-ROM is low in cost, and best of all, requires little more than a record player and your cassette interface. (Use the card at the end of this book to order the Floppy ROM™.)

What's On The Record?

There are two different sections on the record. First are the test patterns to align your cassette interface. The second section is the software you will want to load into your 6800 based microcomputer. This section of the record includes a binary loader program, and the binary dump of BASIC.

What Will I Need To Use The Record?

A 6800 based microprocessor system is a must. A minimum amount of memory considered should be 6K. In addition, a Kansas City Cassette interface is also a must, and some sort of 300-baud terminal is desirable to check the alignment of your cassette interface. A cassette machine should be used to save BASIC once it gets loaded, because the record will have limited life. Don't rely on it for more than ten or twenty loads because it will wear out. BASIC assumes that MIKBUG is resident in your system, with a scratchpad from A000 to A07F. Patching instructions are provided in an appendix to this article for those systems that do not have MIKBUG and the scratchpad. A separate loader program is also supplied for those without a MIKBUG loader. Several SWTPC 6800 systems have been used by various individuals in verifying the concept behind BASIC on a "Floppy-ROM™," so owners of those systems should not have trouble at all. Last but certainly not least, a 33 1/3 RPM turntable.

What Is The "Kansas City Standard?"

The "Kansas City" recording standard was established to allow interchange of data and programs between microcomputer users. It allows great latitude in playback speed, and can be used with almost any cassette machine. "Marks", or logic ones, are represented by a frequency of 2400 Hz, and "spaces", or logic zeros, are represented by 1200 Hz. Data is transferred using the standard UART format, (1 start bit, 1 stop bit) at 300 baud. (Roughly thirty characters per second.) These frequencies were chosen because the 16X clock for the UART can be easily derived from the data being fed into the interface. Speed tolerances of 20- to 30% are acceptable. Figure 1 shows the circuit of the Southwest Technical Products AC-30. It is a fairly simple circuit, which will allow entry of the program into your system. With this circuit, or its equivalent, the program on the "floppy-ROM" may be read.

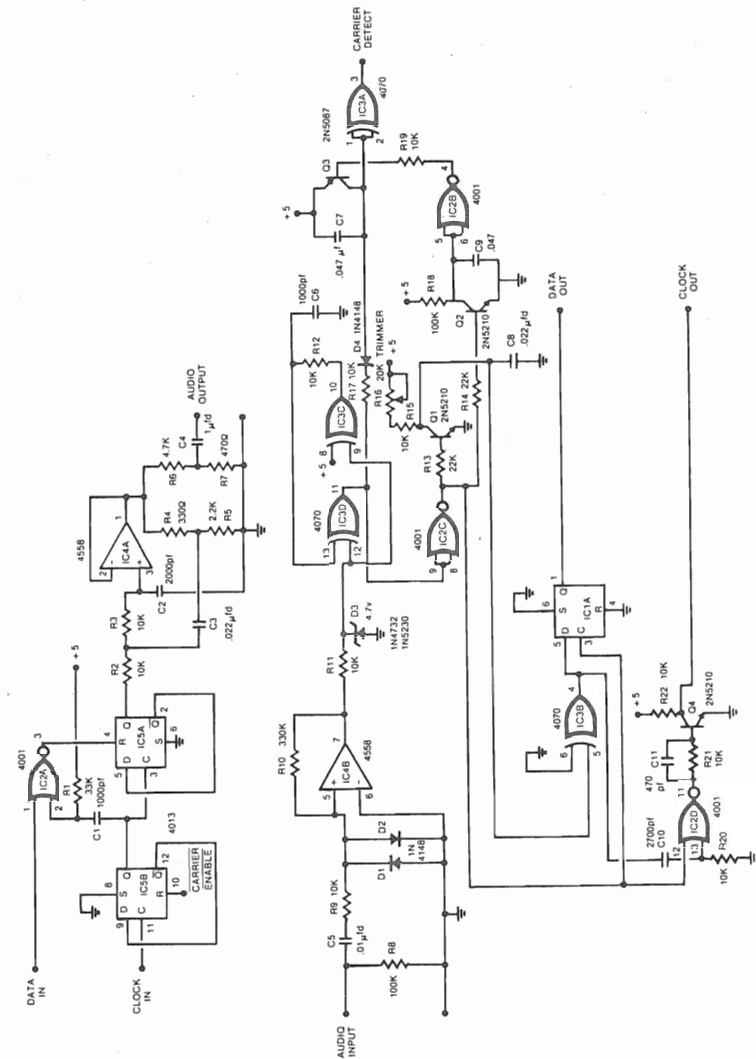


Figure 1. SWTPC AC-30 Cassette Tape Modulator/Demodulator Schematic

How Does It Work?

Audio is fed through the highpass filter made up of R9 and C5. It is clipped by the pair of diodes, and fed into IC-4B, which acts as a comparator. IC-4 should be fed a well regulated 7.5V supply, to ensure stable operation at this point. Zener diode D3 limits the output of IC-4 to levels acceptable to the CMOS gate that follows. When the comparator changes state, IC-3C and IC-3D generate a short (5 microsecond) negative pulse. When data is being received, these pulses repeat. This train of negative pulses grounds C7 through D4. The output of C7 is inverted, buffered, and can be used to generate a carrier detect signal. These negative pulses are found at pin 11 of IC-3D. They're inverted by IC-2A, where they feed four circuits. The first circuit is a missing pulse detector, made up of Q2 and IC-2B. They pull the Carrier detect signal low if several cycles of audio are lost.

The second circuit is an adjustable missing pulse detector, which "times out" when 1200 Hz data are received. R16 trims this time period. The third circuit is flip-flop IC-1A, which outputs the data. Its output is high when 1200 Hz is received. The last circuit is the clock synthesizer, which is made up of IC-2D and IC-3B. These generate the 16X clock for your UART.

How Do I Hook Up A Turntable?

The AC-30 requires about 5 volts of signal to decode the data on the floppy record. Speaker output jacks will easily supply this level. Before you hook up a phonograph, check to see if it has a transformer-isolated chassis. If it doesn't, *don't hook it up!* It would be all too easy to get 120V of your local power utility into your computer. This would tend to make a very expensive pile of write-only memory. If necessary, have a technician check out your stereo. Some component pre-amps are capable of driving 5 volts. A Dyna Pat-4 was used to test the records, and that worked nicely. However, it had to be turned up all the way. If your stereo set has a MONO—STEREO switch, set it on mono. A large percentage of noise on this type of disc is vertical. The vertical signal is essentially a "difference" signal for stereo, but is unwanted for this playback. It will cancel nicely, which will help load a very noisy disc.

Adjust your stereo set for 'flat' response. If your tone control is a 'High Cut' type, set the control for maximum. This is the starting point to align your system. If possible, transcribe the record the first time it is played. It should have reasonable life, 20 to 30 plays or more, but better safe than sorry.

What Are The Weird Signals At The Start Of The Record?

There are two test patterns that should be used to set up your AC-30. The first is a pattern of "5"s. Set your system up to echo the data from the tape to your terminal (local mode). Play back the tape. You should see a stream of "5"s. If you don't, adjust the tone and volume controls slowly and carefully. The "5"s should play back

perfectly. There should be no other characters intermixed with the "5"s.

If you are not able to get the "5"s to play back properly, check your interface circuitry next. If you built the interface shown in Figure 1, adjust the R16 to trim the test pattern.

The second test pattern is a "U" pattern, to check for jitter. If the "5"s worked properly, this should also. If you are not able to get the "5"s to read out, don't go further. Examine your connections, make sure you do not have any ground loops. Make sure you are getting a good clean signal out of the turntable. If you re-recorded the record on a cassette, make sure that you did not overload the cassette recorder when recording. Figure 2, shows a general view of how your system should be hooked up, with the turntable hooked into the AC-30 in place of a cassette machine. If you have arrived this far with flying colors, now is the time to load the program. First check your memory. Make sure all is well before going further.

A slightly high boost may be necessary in some systems. If your system has a rumble filter, switch it on. There will be no valid information below 100 Hz, and elimination of these noise components may be helpful.

If your record player has a good "scratch" filter, try it too. There is no information above 2400, but response at 2400 should be left as "Flat" as possible. If your scratch filter affects 5000 and above, it may be OK; 10000 and above would be better. Remember that most filters have a finite slope and frequencies 2 to 3 octaves away from the break frequency will be affected by level and phase changes. If you detect errors during program load, turn off the 'scratch' filter first.

What If My Turntable Runs Fast Or Slowly?

Again, the beauty of the "Kansas City" standard comes to the rescue. Very few turntables have speed errors larger than 10%, so there is little reason for concern. If your error is very large, adjust your cassette interface card. Run as close to 33⅓ as possible on an adjustable speed turntable.

How To Load The Program

After the test patterns, there is a 20-second space where the 'band' is cut. A string of 'L's follow, to start the loader function in MIKBUG. Place the tone arm down on the record near the end of the test patterns. There is a binary loader program in front of the data, which MIKBUG loads into memory at 1100 HEX. The computer then executes the binary load program, and loads BASIC. Very simple if you have MIKBUG. Figure 4, is a listing of the loader program which was loaded at 1100 HEX. You will note several MIKBUG routines were used. Patch these to fit your operating system. If your turntable has a center depression, place a regular record on the turntable first to support the Floppy-ROM™.

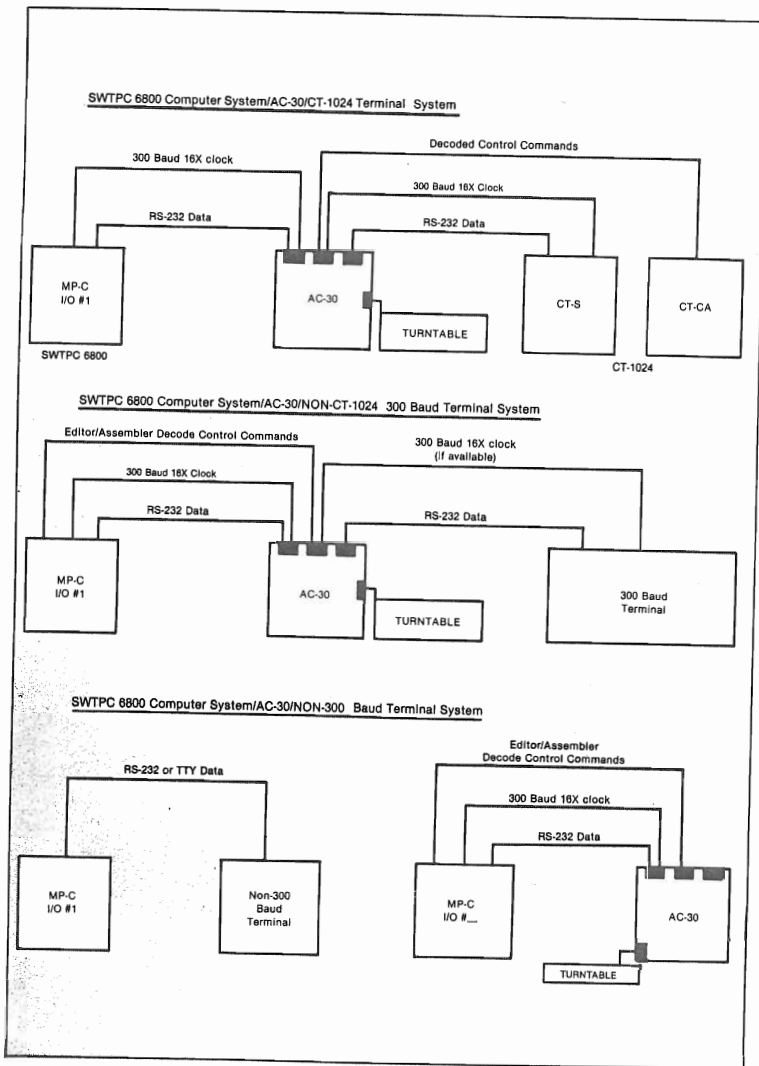


Figure 2. Interconnection Block Diagrams

Pointers And External Routines

Several subroutines in MIKBUG® were used. Copies of these routines may be found in Motorola Engineering Note #100, which is a listing of MIKBUG®. The routines are shown in Listing 1. Here is a short summary of the external subroutines, and a short explanation of what they do.

0272	7E	E0BF	OUT2H	JMP	\$E0BF	OUT2H IN MIKBUG
0275	7E	E0C8	OUT4HS	JMP	\$E0C8	OUT4HS IN MIKBUG
0278	8D	09	OUTCH	BSR	BREAK	
027A	7E	E1D1		JMP	\$E1D1	OUTEEE IN MIKBUG
027D	BD	E1AC	INCH	JSR	\$E1AC	INEEE IN MIKBUG
0280	36			PSH	A	
0281	20	08		BRA	BREAK0	
0283	36		BREAK	PSH	A	
0284	B6	8004		LDA	A	\$8004
0287	2B	09		BMI	BREAK1	
0289	8D	F2		BSR	INCH	
028B	81	03	BREAK0	CMP	A	#\$03
028D	26	03		BNE	BREAK1	
028F	7E	0815		JMP	READY	NOTE READY LOCATION
0292	32		BREAK1	PUL	A	
0293	39			RTS		DONE

Listing 1.

The first of these routines is "OUT2H." This routine prints the contents of memory pointed to by the index register. The contents are printed as two HEX characters. The index register is incremented before returning. A comparable routine should be found in almost all operating systems. The jump location is:

0272 (HEX): 7E, E0, BF

E0BF is the location of OUT2H in MIKBUG®, 7E is the jump instruction.

The second external routine called is "OUT4HS." This routine will print 4 HEX characters by calling the OUT2H routine twice. It then outputs a space. Again, there is an equivalent in almost all operating systems. This routine returns the Index Register incremented twice. The instruction is located at:

0275: 7E, E0, C8

E0C8 is the address of OUT4HS in MIKBUG®.

The input and output routines in MIKBUG® are also used. INEEE and OUTEEE transfer one character through 'A' when called. It should be noted that the control port on a MIKBUG® equipped system is a 'bitbanger' and thus may have different status bits than an ACIA. The jump locations are:

INEEE:

027D: BD, E1, AC (Note it is a jump to subroutine)

OUTEEE:

027A: 7E, E1, D1 (Note that this is a jump)

Listing 1 shows the block of code containing these calls. The listing also has a listing of the break routine. If you are using an ACIA, it will

be necessary to change the break routine to check for your status. You will probably have to add a short subroutine to the end of BASIC.

Another not-so-critical patch location is the carriage return-line feed string. The 15(HEX) is the erase to end of line character on a SWTPC CT-1024 terminal. This may be changed as necessary to allow an erase on your terminal. The string is located at:

02AC: 0D, 0A, 15, FF, FF, FF, FF, 1E

If you have to add a routine to the end of BASIC, the pointer for the beginning of program storage must be changed, or BASIC will dramatically change your code! The locations you must change are:

07F9, 07FA

They currently contain 1200 HEX. You would have to change the contents of these locations to point at a location after your routine(s). Any routines that you have to add, should be located beyond 1200.

Stack load and store operating may have to be changed if you don't have memory from A000 to A07F. There are four stack pointer instructions of note. These all load the stack pointer with A045 HEX. These locations are:

080F, 0848, 0B16, 0CC3

There is also a pair of stack references in the patch command routine. The first is at 08F8. This contains A040 currently. The instruction is located in 08FB then stores the stack in A008. Note... the "Patch" command has another jump that may have to be changed if you don't have MIKBUG®. 08E3 contains the jump to MIKBUG®. It contains E0E3 currently. Change this to the equivalent in your monitor.

There is an index register stack which has one reference. The index register stack sits at the top of MIKBUG's scratchpad. Location 0819 contains the Value A07F. Move this elsewhere as necessary.

There is a store index register command in the "Patch" routine. It is located at 08F5, and contains an A046. This location is the top of the stack, so the 'G' command will cause a return to BASIC.

Listing 2 shows a possible break routine for those with an ACIA for input. Note that it will not fit in the room that the current break routine occupies. Change the current break to have a jump subroutine at 1200 HEX. A return would follow that.

Listing 3 is a command listing of the biload program. It may be 'patched' to fit into most operating systems, and will be quite easy to implement on a system using an ACIA. The data is transferred through the A accumulator. In MITS 680b systems, you will have to transfer from B to A because they use B for input and output. The place to put your input one character routine call is INCHP. Load your BILOAD equivalent, then execute it. It will ignore the BILOAD program on the FLOPPY ROM™ and load BASIC (hopefully).

```

00001          NAM      BREAK
00002          OPT      S*P
00003          0B15    READY EQU  *0B15    INSIDE BASIC....
00004          E000    PORT  EQU  *E000
00005 1200          ORG      *1200
00006 1200 36      BREAK PSH A
00007 1201 B6 E000 LDA A      PORT      (THIS IS THE STATUS PORT)
00008 1204 48      ASL A      PUT DATA AVAILABLE FLAG IN
00009 1205 24 0C   BCC      RETURN NOTHING THERE..BRANCH
00010 1207 B6 E001 LDA A      PORT+1  BRING IN A CHAR IF THERE
00012 120A B4 7F   AND A      *%7F   GET RID OF PARITY...
00013 120C B1 03   CMP A      *%03   IS IT A BREAK CHARACTER?
00014 120E 26 03   BNE      RETURN  NO...
00015 1210 7E 0B15 JMP      READY  YES IT IS!! STOP!
00016          *
00017 1213 32      RETURN PUL A
00018 1214 39      RTS      GO BACK AGAIN...
00019          END

```

READY 0B15
PORT E000
BREAK 1200
RETURN 1213

Listing 2.

```

00030          NAM      BILOAD
00031          OPT      L
00032          *$1500  LDS    NEW STACK LOCATION
00033          19A4 8E 1500 BILOAD BSR    GET IO READY
00034          19A7 8D 49      BSR    LOAD
00035          19A9 8D 3C   OVER BSR    INPUT
00036          19AB B1 58   CMP A  *%X   GET A CHARACTER
00037          19AD 26 FA   BNE    OVER IS IT FIRST CHARACTER?
00038          19AF 8D 36   BSR    INPUT IF NOT, KEEP LOOKING....
00039          19B1 B1 31   CMP A  *%1   GET ANOTHER CHARACTER
00040          19B3 27 07   BEQ    READ SECOND CHARACTER?
00041          19B5 B1 39   CMP A  *%?   GET DATA AND MORE...
00042          19B7 26 F0   BNE    OVER END CHARACTER?
00043          19B9 7E E0E3 JMP    CONTRL KEEP LOOKING
00044          2895          * 2895 * PATCH HERE FOR OTHER
00045          19BC 7F 17FF READ CLR    CKSM 6800 SYSTEMS
00046          19BF 8D 26   BSR    INPUT START WITH ZERO
00047          19C1 16      TAB    GET BYTE COUNT
00048          19C2 5C      INC B  * PUT IT IN BP
00049          19C3 8D 22   BSR    INPUT INCREMENT IT ONCE
00050          19C5 B7 1800 STA A  TW  GET FIRST PART OF ADDRESS
00051          19C8 8D 1D   BSR    INPUT STORE IT TEMPORARILY
00052          19CA B7 1801 STA A  TW+1 GET SECOND PART
00053          19CD FE 1800 LDX    TW  STORE IT
00054          19D0 8D 15   STORE BSR    INPUT GET SOME DATA
00055          19D2 A7 00   STA A  X  STORE IT
00056          19D4 01      NOP    TIME TO FORGET..
00057          19D5 A1 00   CMP A  X  IF SAME, GO ON
00058          19D7 26 0B   BNE    OUT  IF NOT, NO MEMORY THERE
00059          19D9 08      INX    POINT AT NEXT LOCATION
00060          19DA 5A          DEC B  * SUBTRACT 1 BYTE FROM COUNT
00061          19DB 26 F3   BNE    STORE NOT DONE, DO MORE
00062          19DD 8D 08   BSR    INPUT OK, ADD IN TAPE CHECKSUM
00063          19DF 7C 17FF INC    CKSM  ADD ONE...
00064          19E2 27 C5   BEQ    OVER IF ZERO, ALL IS GOOD
00065          19E4 7E E040 OUT JMP    LOAD19 AN ERROR! PRINT GARBAGE
00066          19E7 8D 14   INPUT BSR    INCHP INPUT 1-B BIT CHARACTER
00067          19E9 36      PSH A  * SAVE IT
00068          19EA BB 17FF ADD A  CKSM  ADD SOME TO CHECKSUM
00069          19ED B7 17FF STA A  CKSM  STORE THE RESULTS
00070          19F0 32      PUL A  * RECALL THE DATA
00071          19F1 39      RTS    GO BACK

```

```

03220 19F2 86 11  LOAD  LDA A  ##11  START YOUR ENGINES
03230 19F4 BD E1D1  JSR   OUTEEE TELL THE MACHINE TO GO
03240 19F7 86 3C    LDA A  ##3C  PIA READER CONTROL
03250 19F9 B7 8007  STA A  #8007  PUT IT INTO PIA
03260 19FC 39      RTS

03271          3271 * NOTE...IF YOU HAVE A ROUTINE THAT WILL
03272          3272 * INPUT A FULL 8 BIT WORD WITH PARITY
03273          3273 * CALL YOUR ROUTINE IN PLACE OF INCHP...
03274          3274 * FOR EXAMPLE...
03275          3275 * INCHP PSH B      PROTECT B
03276          3276 *          JSR INCHAR THIS IS YOUR ROUTINE

-----
03277          3277 *          PUL B      GET B BACK
03278          3278 *          RTS
03279          3279 * PROTECT INDEX REGISTER AT ALL TIMES!!!
03280 19FD 37      INCHP PSH B *      THIS IS ALMOST SAME
03290 19FE BD E1A5  JSR   SAV   AS MIBUG,BUT SAVES PARITY
03300 1A01 A6 00  IN1   LDA A: X  BIT. SEE MOTOROLA ENG.100
03310 1A03 2B FC   BMI   IN1   FOR DETAILS OF HOW IT WORKS

03330 1A05 6F 02          CLR  2*X
03340 1A07 BD E1F3  JSR   DE
03350 1A0A BD E1EF  JSR   DEL
03360 1A0D C6 04          LDA B  #4
03370 1A0F E7 02          STA B  2*X
03380 1A11 5B          ASL B

03400 1A12 BD E1EF IN3  JSR   DEL
03410 1A15 0D          SEC
03420 1A16 69 00          ROL   X
03430 1A18 46          ROR A
03440 1A19 5A          DEC B
03450 1A1A 26 F6          BNE IN3
03460 1A1C BD E1EF  JSR   DEL
03470 1A1F 7E E1E3  JMP   IOUT2

```

Listing 3.

Memory Map for 4K Basic

```

0000-00FF  Input buffer and temporary variable storage
0100      Hard starting address of BASIC (Cold start—clears
          program)
0103      Soft starting address of BASIC (warm start—saves
          program)
0105-10FF  BASIC Interpreter
1100-11FF  Arithmetic and For-Next stack
A000-A045  Machine stack
A04A-A07F  Index register stack

```

Note: Binary loader program starts at 1100. It will be cleared by the Basic interpreter when 0100 is executed.

```

02775 1100          ORG   $1100
02780 1100 8E A047  BILDAD LDS   ##A047
02790 1103 8D 49   BSR   LOAD

02800 1105 8D 3C   OVER  BSR   INPUT
02810 1107 81 58          CMP A  #'X
02820 1109 26 FA          BNE  OVER
02830 110B 8D 36          BSR   INPUT
02840 110D 81 31          CMP A  #'1
02850 110F 27 07          BEQ  READ
02860 1111 81 39          CMP A  #'9

```

```

02870 1113 26 F0          BNE  OVER
02880 1115 7E E0E3       JMP   CONTRL

02900 1118 7F A016 READ  CLR   CKSM
02910 111B 8D 26          BSR   INPUT
02920 111D 16          TAB
02930 111E 5C          INC B
02940 111F 8D 22          BSR   INPUT
02950 1121 B7 A019       STA A  TW
02960 1124 8D 1D          BSR   INPUT
02970 1126 B7 A01A       STA A  TW+1
02980 1129 FE A019       LDX   TW

03000 112C 8D 15   STORE  BSR   INPUT
03010 112E A7 00       STA A  X
03020 1130 01          NOP
03040 1131 A1 00       CMP A  X
03050 1133 26 0B       BNE  OUT
03060 1135 08          INX
03080 1136 5A          DEC B
03090 1137 26 F3       BNE  STORE
03100 1139 8D 08       BSR   INPUT
03110 113B 7C A016     INC   CKSM
03120 113E 27 C5       BEQ  OVER

03130 1140 7E E040 OUT  JMP   LOAD19

03140 1143 8D 14   INPUT  BSR   INCHP
03150 1145 36       PSH A
03160 1146 BB A016   ADD A  CKSM
03170 1149 B7 A016   STA A  CKSM
03180 114C 32       PUL A
03190 114D 39       RTS

03210 114E 86 11  LOAD  LDA A  ##11
03220 1150 BD E1D1  JSR   OUTEEE
03230 1153 86 3C    LDA A  ##3C
03240 1155 B7 8007  STA A  #8007
03250 1158 39      RTS
03270 1159 37      INCHP PSH B
03280 115A BD E1A5  JSR   SAV

03290 115D A6 00  IN1   LDA A  X
03300 115F 2B FC   BMI   IN1
03320 1161 6F 02          CLR  2*X
03330 1163 BD E1F3  JSR   DE
03340 1166 BD E1EF  JSR   DEL
03350 1169 C6 04          LDA B  #4
03360 116B E7 02          STA B  2*X
03370 116D 5B          ASL B

03390 116E BD E1EF IN3  JSR   DEL
03400 1171 0D          SEC
03410 1172 69 00          ROL   X
03420 1174 46          ROR A
03430 1175 5A          DEC B
03440 1176 26 F6          BNE  IN3

03450 1178 BD E1EF  JSR   DEL
03460 117B 7E E1E3  JMP   IOUT2

```

Listing 4.

```

NAM BASIC 4K
*
*
***** VERSION 2.1 *****
*
* BY ROBERT H. UITERWYK
* PLACED IN THE PUBLIC
* DOMAIN JUNE, 1979 FOR
* THE PURPOSES OF EDUCATION
* FOR ALL MICROCOMPUTER
* USERS.
*
* THIS PROGRAM ASSUMES THAT THE
* MOTOROLA MIKBUG ROM IS INSTALLED
* AND THAT ITS ASSOCIATED 128 BYTE
* RAM IS ALSO PRESENT
* THE SP AND XSTACK WILL HAVE TO
* BE MOVED IF THIS IS NOT THE CASE.

```

```

00270 *SAVE FIRST $20 BYTES FOR
00280 *FUTURE DISK USE
00290 0020 ORG $20
00300 0020 0002 INDEX1 RMB 2
00310 0022 0002 INDEX2 RMB 2
00320 0024 0002 INDEX3 RMB 2
00330 0026 0002 INDEX4 RMB 2
00340 0028 0002 SAVESP RMB 2
00350 002A 0002 NEXTRA RMB 2
00360 002C 0002 WORKBA RMB 2
00370 002E 0002 SOURCE RMB 2
00380 *TO BE CHANGED AT PROGRAMMERS OPTION
00390 0030 0002 PACKLN RMB 2
00400 0032 0002 HIGHLN RMB 2
00410 0034 0002 RASPNT RMB 2
00420 0036 0002 RASLIN RMB 2
00430 0038 0002 PUSHTX RMB 2
00440 003A 0002 YSTACK RMB 2
00450 003C 0002 DATPNT RMB 2
00460 003E 0002 DIMPNT RMB 2
00470 0040 0001 PRCNT RMB 1
00480 0041 0003 RMB 3
00490 0044 0002 MEMEND RMB 2
00510 0046 0002 ARRTAR RMB 2
00520 0048 0002 KEYWD RMB 2
00530 004A 0001 TSIGN RMB 1
00540 004B 0001 EXP RMB 1
00550 004C 0001 SIGDIG RMB 1
00560 004D 0001 DECFLG RMB 1
00570 004E 0001 TOGGLE RMB 1
00580 004F 0001 NCMPR RMB 1
00590 0050 0001 TNJMR RMB 1
00600 0051 0001 ANJMR RMB 1
00610 0052 0001 RNJMR RMB 1
00620 0053 0001 LRNTNF RMB 1
00630 0054 0001 ROUNDV RMB 1
00640 0055 0001 ONLDDP RMB 1
00650 0056 0001 NORSGN RMB 1
00660 0057 0002 YLONC RMB 2
00670 0059 0002 MPYXX RMB 2
00680 005B 0002 RASNUM RMB 2
00690 005D 0002 AFSTK RMB 2
00700 005F 0002 FORPNT RMB 2
00710 0061 0002 FORNDW RMB 2
00720 0063 0002 VARPNT RMB 2
00730 0065 0002 SBRPNT RMB 2
00740 0067 0002 DUMMY RMB 2
00750 0069 0006 RANNUM RMB 6
00760 006F 0006 FORTMP RMB 6
00770 0075 0010 SBRSTK RMB 10
00780 0085 0001 LDAFLG RMB 1

00800 00AC ORG $00AC
00810 00AC 0002 RUFNXT RMB 2
00820 00AE 0002 FNDRUF RMB 2
00830 00B0 ORG $00B0
00840 00B0 0050 BUFFER RMB $50 I/O BUFFER = 80 DECIMAL

```

```

00860 0100 ORG $0100
00870 0100 FD 07F2 PROG JSR START
00880 0103 7F 07F6 JMP READY
00890 0106 54 INTER FCC /TO/
0107 4F
00900 0108 1F FCR $1F
00910 0109 9999 TOPNT FDR $9999
00920 010B 53 FCC /STFP/
010C 54
010D 45
010E 50
00930 010F 1F FCR $1F
00940 0110 8888 STFPNT FDB $8888
00950 0112 52 COMMAN FCC /RUN/
0113 55
0114 4F
00960 0115 1F FCR $1F
00970 0116 085R FDR RIN
00980 0118 4C FCC /LIST/
0119 49
011A 53
011B 54
00990 011C 1F FCB $1F
01000 011D 0882 FDR CLIST
01010 011F 4F FCC /NEW/
0120 45
0121 57
01020 0122 1F FCR $1F
01030 0123 07F2 FDR START
01040 0125 50 FCC /PAT/
0126 41
0127 54
01050 0128 1E FCB $1F
01060 0129 08D3 FDB PATCH
01070 012B 4C FCC /LOA/
012C 4F
012D 41
01080 012E 1F FCR $1F
01090 012F 090D FDR LOAD
01100 0131 53 FCC /SAV/
0132 41
0133 56
01110 0134 1F FCB $1F
01120 0135 08F5 FDR SAVE
01130 0137 47 GOLIST FCC /GOSUB/
0138 4F
0139 53
013A 55
013B 42
01140 013C 1E FCB $1F
01150 013D 0A5A FDB GOSUR
01160 013F 47 FCC /GOTO/
0140 4F
0141 54
0142 4F
01170 0143 1F FCR $1F
01180 0144 0A7C FDB GOTO
01190 0146 4F FCC /ON/
0147 4F
01200 0148 1F FCR $1F
01210 0149 0A3B FDB ONGO
01220 014B 54 FCC /THEN/
014C 48
014D 45
014E 4F
01230 014F 1E FCR $1F
01240 0150 0A29 FDB THEN

```

01250	0152 50	FCC	/PRINT/
	0153 52		
	0154 49		
	0155 4F		
	0156 54		
01260	0157 1F	FCR	\$1F
01270	0158 0C6D	FDR	PRINT
01280	015A 4C	FCC	/LEFT/
	0158 45		
	015C 54		
01290	015D 1F	FCR	\$1F
01300	015E 0DRC	FDR	LEFT
01310	0160 49	FCC	/INPUT/
	0161 4F		
	0162 50		
	0163 55		
	0164 54		
01320	0165 1F	FCB	\$1F
01330	0166 0AC4	FDR	INPUT
01340	0168 49	FCC	/IF/
	0169 46		
01350	016A 1F	FCR	\$1F
01360	016B 0FDD	FDR	IF
01370	016D 52	FCC	/READ/
	016E 45		
	016F 41		
	0170 44		
01380	0171 1F	FCB	\$1F
01390	0172 0C17	FDR	READ
01400	0174 44	FCC	/DATA/
	0175 41		
	0176 54		
	0177 41		
01410	0178 1F	FCB	\$1F
01420	0179 0DDE	FDR	REMARK
01430	017B 52	FCC	/RESTORE/
	017C 45		
	017D 53		
	017E 54		
	017F 4F		
	0180 52		
	0181 45		
01440	0182 1F	FCR	\$1F
01450	0183 0C64	FDR	RESTORE
01460	0185 45	FCC	/END/
	0186 4F		
	0187 44		
01470	0188 1F	FCR	\$1F
01480	0189 07F6	FDR	READY
01490	0188 52	FCC	/RETURN/
	018C 45		
	018D 54		
	018E 55		
	018F 52		
	0190 4F		
01500	0191 1F	FCB	\$1F
01510	0192 0AA2	FDR	RETURN
01520	0194 44	FCC	/DIM/
	0195 49		
	0196 4D		
01530	0197 1F	FCR	\$1F
01540	0198 0F06	FDR	DIM
01550	019A 46	FCC	/FOR/
	019B 4F		
	019C 52		
01560	019D 1F	FCB	\$1F
01570	019E 0F86	FDR	FOR
01580	01A0 4F	FCC	/NEXT/
	01A1 45		

	01A2 58		
	01A3 54		
01590	01A4 1F	FCB	\$1F
01600	01A5 0F50	FDR	NEXT
01610	01A7 52	FCC	/RFM/
	01A8 45		
	01A9 4D		
01620	01AA 1F	FCR	\$1F
01630	01AB 0DDE	FDR	REMARK
01640	01AD 41	FCC	/APP/
	01AE 50		
	01AF 50		
01650	01B0 1F	FCB	\$1F
01660	01B1 0910	FDR	APPEND
01670	01B3 53	STDMMSG FCC	/STOP/
	01B4 54		
	01B5 4F		
	01B6 50		
01680	01B7 1F	FCR	\$1F
01690	01B8 0AB6	FDR	STOP
01700	01BA 1F	COMFND FCR	\$1F
01710	01BB 0DRC	TMPLT FDR	LEFT
01730	01RD 52	FUNCT FCC	/RND(/
	01RE 4E		
	01BF 44		
	01CO 28		
01740	01C1 1F	FCR	\$1F
01750	01C2 10B7	FDR	RANDOM
01760	01C4 54	FCC	/TAB(/
	01C5 41		
	01C6 42		
	01C7 28		
01770	01C8 1F	FCB	\$1F
01780	01C9 0CR2	FDR	TAB
01790	01CB 49	FCC	/INT(/
	01CC 4F		
	01CD 54		
	01CE 28		
01800	01CF 1F	FCR	\$1F
01810	01D0 1079	FDR	INT
01820	01D2 43	FCC	/CHRS(/
	01D3 48		
	01D4 52		
	01D5 24		
	01D6 28		
01830	01D7 1F	FCR	\$1F
01840	01D8 0CA5	FDR	CHR
01850	01DA 41	FCC	/ARS(/
	01DB 42		
	01DC 53		
	01DD 28		
01860	01DE 1F	FCR	\$1F
01870	01DF 10AE	FDR	ARS
01880	01E1 53	FCC	/SGN(/
	01E2 47		
	01E3 4F		
	01E4 28		
01890	01E5 1F	FCR	\$1F
01900	01E6 1093	FDR	SGN
01910	01E8 55	FCC	/ISFR(/
	01E9 53		
	01FA 45		
	01FR 52		
	01EC 28		

```

01920 01ED 1F      FUNFND FCB      $1F
01930 01EE 106R    PDR          USFR
01950 01FO 13      RDYMSG FCB      $13
01960 01F1 14      FCB          $14
01970 01F2 0D      FCB          $0D
01980 01F3 0A      FCB          $0A
01990 01F4 15      FCB          $15
02000 01F5 52      FCC          /READY/
          01F6 45
          01F7 41
          01F8 44
          01F9 59
02010 01FA 1F      FCB          $1F
02020 01FR 20      DELMSG FCC      / DEL/
          01FC 44
          01FD 45
          01FE 4C
02030 01FF 1F      FCB          $1F
02040 0200 10      PGCNTL FCB      $10
02050 0201 16      FCB          $16
02060 0202 1F      FCB          $1F
02070 0203 52      WHAT FCC       /RF-ENTER/
          0204 45
          0205 2D
          0206 45
          0207 4F
          0208 54
          0209 45
          020A 52
02080 020B 1E      FCB          $1F
02090 020C 45      FRRMS1 FCC     /FRRDR# /
          020D 52
          020E 52
          020F 4F
          0210 52
          0211 23
          0212 20
02100 0213 1F      FCB          $1F
02110 0214 20      FRRMS2 FCC     / IN LINE /
          0215 49
          0216 4F
          0217 20
          0218 4C
          0219 49
          021A 4E
          021R 45
          021C 20
02120 021D 1F      FCB          $1F
02140 021F CE 01FR DEL LDX #DFMSG
02150 0221 8D 6F   RSR          OUTPUT
02160 0223 20 07   BRA          KEYRDO
02170
02180 0225 86 3F   *ALL PURPOSE KEYRD ROUTINE
          KEYBD LDA A #S3F  OUTPUT ""
02190 0227 8D 4D   RSR          OUTCH
02200 0229 8D 0350 JSR          PRINSP
02210
          *SUBROUTINE ENTRY WITHOUT ""
02220 022C CE 00R0 KEYRDO LDX #RUFFFR  START OF KEYRD ROUTINE
02230 022F 8D 4A   KEYRDI RSR   INCH  GET A CHARACTER
02240 0231 81 18   CMP A #S1R  CANCEL - 000F
02250 0233 27 E9   RFO          DEL
02260 0235 81 0D   CMP A #S0D   IS IT C/R
02270 0237 27 2A   RFO          IFEXIT IF SO EXIT
02280 0239 81 0F   CMP A #S0F  Backspace Code
02290 023B 26 0C   BNF          **14
02300 023D 86 5F   LDA A #S5F
02310 023F 8D 35   RSR          OUTCH
02320 0241 8C 00B0 CPX #RUFFFR  IS IT FIRST CHARACTER
02330 0244 27 F9   RFO          IF SO, JUST RFL0OP
02340 0246 09      DEX          OTHERWISE, BACKSPACE
    
```

```

02350 0247 20 F6   BRA          KEYRDI  AND THEN RFL0OP
02360 0249 7D 00B5 TST          LJAFLG
02370 024C 26 0A   RNF          KEYRDI
02380 024E 8C 00F8 CPX          #BUFFFF+72
02390 0251 26 05   RNF          **7
02400 0253 C6 21   LDA R       #S21
02410 0255 7E 0828 JMP          FRRDR
02420 0258 A7 00   KEYRDI STA A 0.X
02430 025A 08      INX
02440 025B 20 D2   BRA          KEYRDI
02450 025D 86 23   CNTLIN LDA A #/#
02460 025F 8D 15   RSR          OUTCH
02470 0261 20 C9   BRA          KEYRDI
02480 0263 86 1E   IFEXIT LDA A #S1F
02490 0265 A7 00   STA A X
02500 0267 DF AE   STX          FNRBUF
02510 0269 96 85   LDA A LJAFLG
02520 026B 26 02   RNF          **4
02530 026D 8D 31   RSR          CRLF
02540 026F 39      RTS
    
```

```

02560 *****
02570 *
02580 * THIS PAGE CONTAINS EXTERNAL ROUTINE CALLS
02590 *
02600 *****
02610 0270 7F 00BF OUT2H JMP $00BF OUT2H IN MIKBUG
    
```

```

02630 0273 7F 00C8 OUT4HS JMP $00C8 OUT4HS IN MIKBUG
    
```

```

02650 0276 8D 09 OUTCH BSR BRFAK
02660 0278 7F F1D1 JMP $F1D1 OUTFFF IN MIKBUG
    
```

```

02680 027B 8D F1AC INCH JSR $F1AC INFFF IN MIKBUG
02690 027E 36      PSH A
02700 027F 20 08   BRA          BRFAK0
    
```

```

02720 0281 36      BREAK PSH A
02730 0282 86 8004 LDA A $9004
02740 0285 2R 09   RMI          BRFAK1
02750 0287 8D F2   RSR          INCH
02760 0289 81 03   BREAKO CMP A #S03
02770 028B 26 03   RNF          BRFAK1
02780 028D 7E 07F6 JMP          READY
02790 0290 32      BREAKI PUL A
02800 0291 39      RTS
    
```

```

02830 *UNIVERSAL OUTPUT ROUTINE
02840 0292      OUTPUT FOU *
02850 0292 8D 05   BSR          OUTNCR
02860 0294 20 0A   BRA          CRLF
02870 0296 8D DE   BSR          OUTCH
02880 0298 08      INX
02890 0299 A6 00   OUTNCR LDA A 0.X
02900 029B 81 1E   CMP A #S1F
02910 029D 26 F7   RNF          *-7
02920 029F 39      RTS
02930 02A0 8D 10   CRLF RSR PUSHX
02940 02A2 CE 02AA LDA #CRLFST
02950 02A5 8D F2   RSR          OUTNCR
02960 02A7 8D 1F   RSR          PULLX
02970 02A9 39      RTS
02980 02AA 0D      CRLFST FCB $0D
02990 02AB 0A      FCB $0A
03000 02AC 15      FCB $15
03010 02AD FF      FCB $FF,$FF,$FF,$FF
          02AE FF
    
```

0273 7F 00C8
 0276 8D 09
 0278 7F F1D1
 0281 36
 0282 86 8004
 0285 2R 09
 0287 8D F2
 0289 81 03
 028B 26 03
 028D 7E 07F6
 0290 32
 0291 39
 0292 8D 05
 0294 20 0A
 0296 8D DE
 0298 08
 0299 A6 00
 029B 81 1E
 029D 26 F7
 029F 39
 02A0 8D 10
 02A2 CE 02AA
 02A5 8D F2
 02A7 8D 1F
 02A9 39
 02AA 0D
 02AB 0A
 02AC 15
 02AD FF
 02AE FF

7200

02 0120 32

02AF FF							
02B0 FF							
03020 02B1 1E	CRFND	FCR	SIF				
03040 02B2 DF 38	PUSHX	STX	PUSHTX				
03050 02B4 DF 3A		LDX	XSTACK				
03060 02B6 09		DFX					
03070 02B7 09		DFX					
03080 02B8 DF 3A		STX	XSTACK				
03090 02BA 36		PSH A					
03100 02BB 96 38		LDA A	PUSHTX				
03110 02BD A7 00		STA A	0.X				
03120 02BF 96 39		LDA A	PUSHTX+1				
03130 02C1 A7 01		STA A	1.X				
03140 02C3 32		PUL A					
03150 02C4 DE 38		LDX	PUSHTX				
03160 02C6 39		RTS					
03180 02C7 DE 3A	PULLX	LDX	XSTACK				
03190 02C9 FF 00		LDX	0.X				
03200 02CB 7C 003R		INC	XSTACK+1				
03210 02CF 7C 003R		INC	XSTACK+1				
03220 02D1 39		RTS					
03240 02D2 8D DE	STOREX	RSR	PUSHX				
03250 02D4 RD 0448		JSR	PULLAF				
03260 02D7 9F 28		STS	SAVFSP				
03270 02D9 35		TXS					
03280 02DA DF 3A		LDX	XSTACK				
03290 02DC FF 00		LDX	0.X				
03300 02DE 20 11		BRA	**+19				
03310 02E0 8D D0	STORF	RSR	PUSHX				
03320 02E2 RD 0448		JSR	PULLAF				
03330 02E5 9F 28		STS	SAVFSP				
03340 02E7 35		TXS					
03350 02E8 09		DFX					
03360 02E9 09		DFX					
03370 02EA 09		DFX					
03380 02EB 09		DFX					
03390 02EC 09		DFX					
03400 02ED 09		DFX					
03410 02EE 09		DFX					
03420 02EF FF 00		LDX	0.X				
03430 02F1 C6 05		LDA R	#5				
03440 02F3 32		PUL A					
03450 02F4 A7 00		STA A	0.X				
03460 02F6 08		INX					
03470 02F7 5A		DFC R					
03480 02F8 26 F9		BNF	*-5				
03490 02FA 32		PUL A					
03500 02FB 32		PUL A					
03510 02FC A7 00		STA A	0.X				
03520 02FE 9E 28		LDS	SAVFSP				
03530 0300 20 C5		RRA	PULLX				
03550 0302 8D AF	TNDX	RSR	PUSHX				
03560 0304 9F 28		STS	SAVFSP				
03570 0306 35		TXS					
03580 0307 20 0A		RRA	TND0				

03590 0309 8D A7	TND	RSR	PUSHX				
03600 030B BD 0448		JSR	PULLAF				
03610 030E 9F 28		STS	SAVFSP				
03620 0310 AE 00		LDS	0.X				
03630 0312 34		DFS					
03640 0313 C6 05	TND0	LDA R	#5				
03650 0315 DE 5D		LDX	AFSTK				
03660 0317 32		PUL A					
03670 0318 A7 00		STA A	0.X				
03680 031A 08		INX					
03690 031B 5A		DFC R					
03700 031C 26 F9		RNF	*-5				
03710 031E 32		PUL A					
03720 031F A7 01		STA A	1.X				
03730 0321 6F 00		CLR	0.X				
03740 0323 9F 28		LDS	SAVFSP				
03750 0325 RD 043F		JSR	PUSHAF				
03760 0328 20 9D		BRA	PULLX				
03780 032A RD 0273	OUTLIN	JSR	OUT4HS				
03790 032D A6 00		LDA A	0.X				
03800 032F 08		INX					
03810 0330 RD 02B2		JSR	PUSHX				
03820 0333 97 49		STA A	KEYWD+1				
03830 0335 DF 48		LDX	KEYWD				
03840 0337 09		DFX					
03850 0338 09		DFX					
03860 0339 A6 00		LDA A	0.X				
03870 033B 81 1F		CMP A	#51F				
03880 033D 26 F9		RNF	*-5				
03890 033F 08		INX					
03900 0340 08		INX					
03910 0341 08		INX					
03920 0342 RD 0299		JSR	OUTNCR				
03930 0345 8D 09		RSR	PRINSP				
03940 0347 7F 0040		CLR	PRCNT				
03950 034A RD 02C7		JSR	PULLX				
03960 034D 7F 0292		JMP	OUTPUT				
03980 0350 36	PRINSP	PSH A					
03990 0351 86 20		LDA A	#520				
04000 0353 RD 0CF6		JSR	OUTCHA				
04010 0356 32		PUL A					
04020 0357 39		RTS					
04040 0358 BD 0744	FINVAR	JSR	SKTSP				
04050 035B RD 042A		JSR	TSTLTR				
04060 035E 25 26		RCS	FINV11				
04070 0360 16	FINVI	TAR					
04080 0361 A6 01		LDA A	1.X				
04090 0363 BD 042A		JSR	TSTLTR				
04100 0366 25 20		RCS	FINV2				
04110 0368 BD 0789		JSR	FCODF				
04120 036B RD 02B2		JSR	PUSHX				
04130 036E DF 34		LDX	BASPNT				
04140 0370 PD 04R1		JSR	PABFXP				
04150 0373 DF 34		STX	BASPNT				
04160 0375 RD 02C7		JSR	PULLX				

04170	0378	FF 00		LIX	0.X	
04180	037A	AD 00		JSR	0.X	
04190	037C	RD 043F		JSR	PUSHAF	
04200	037F	DF 34		LIX	RASPNT	
04210	0381	31		INS		SINCE FUNCTION CAN ONLY BE
04220	0382	31		INS		ON RIGHT OF EXPR WE SKIP
04230	0383	31		INS		
04240	0384	31		INS		DIRECTLY FROM TERM
04250	0385	39		RTS		
04260	0386	0B	FINV11	SEV		
04270	0387	39		RTS		
04280	0388	BD 0432	FINV2	JSR	TFSTNO	
04290	038B	24 07		RCC	FINV3	
04300	038D	81 2B		CMP A	#'(
04310	038F	27 03		RFQ	FINV3	
04320	0391	09		DEF		
04330	0392	86 20		LDA A	#S20	
04340	0394	08	FINV3	INX		
04350	0395	08		INX		
04360	0396	BD 02B2		JSR	PUSHX	
04370	0399	DF 2A		LIX	NEXTBA	
04380	039B	9C 46	FINV4	CPX	ARRTAR	
04390	039D	0D		SFC		
04400	039E	27 09		RFQ	FINV5	
04410	03A0	E1 00		CMP R	0.X	
04420	03A2	26 0C		RNF	FINV6	
04430	03A4	A1 01		CMP A	1.X	
04440	03A6	26 08		RNF	FINV6	
04450	03A8	0C		CLC		
04460	03A9	DF 63	FINV5	STX	VARPNT	
04470	03AB	RD 02C7		JSR	PULLX	
04480	03AF	0A		CLV		
04490	03AF	39		RTS		
04510	03B0	36	FINV6	PSH A		
04520	03B1	A6 01		LDA A	1.X	
04530	03B3	81 28		CMP A	#'(
04540	03B5	32		PUL A		
04550	03B6	27 05		RFQ	FINV7	
04560	03B8	RD 0457		JSR	INXATF	
04570	03BB	20 DF		RRA	FINV4	
04580	03BD	FF 04	FINV7	LIX	4.X	
04590	03BF	20 DA		RRA	FINV4	
04610	03C1	BD 0281	TSTV	JSR	BRFAK	
04620	03C4	8D 92		RSR	FINVAR	
04630	03C6	28 01		RVC	TSTV1	
04640	03C8	39		RTS		
04650	03C9	36	TSTV1	PSH A		
04660	03CA	25 31		RCS	TSTV4	
04670	03CC	DF 34		STX	RASPNT	
04680	03CE	DF 63		LIX	VARPNT	
04690	03D0	81 28		CMP A	#'(
04700	03D2	27 15		RFQ	TSTV3	
04710	03D4	08	TSTV11	INX		
04720	03D5	08		INX		
04730	03D6	DF 63	TSTV2	STX	VARPNT	
04740	03D8	DF 5D		LIX	AESTK	
04750	03DA	96 63		LDA A	VARPNT	
04760	03DC	A7 00		STA A	0.X	
04770	03DE	96 64		LDA A	VARPNT+1	
04780	03E0	A7 01		STA A	1.X	
04790	03E2	8D 5A		RSR	PUSHAF	
04800	03E4	32		PUL A		
04810	03E5	DF 34		LIX	RASPNT	

04820	03F7	0C			CLC	
04830	03F8	39			RTS	
04840	03F9	BD 0DE4	TSTV3	JSR	GETPAR	
04850	03FC	24 04		BCC	TSTV33	
04860	03FE	8D 77	TSTV31	RSR	FIX	
04870	03F0	20 01		RRA	TSTV34	
04880	03F2	5F	TSTV33	CLR B		
04890	03F3	37	TSTV34	PSH P		
04900	03F4	8D 71		RSR	FIX	
04910	03F6	17		TRA		
04920	03F7	33		PUL B		
04930	03F8	RD 0F59		JSR	SHRCAL	
04940	03FB	20 09		RRA	TSTV2	
04950	03FD	DF 34	TSTV4	STX	RASPNT	
04960	03FF	DE 63		LIX	VARPNT	
04970	0401	F7 00		STA R	0.X	
04980	0403	A7 01		STA A	1.X	
04990	0405	81 28		CMP A	#'(
05000	0407	26 19		RNF	TSTV5	
05010	0409	C6 0A		LDA R	#10	
05020	040B	F7 02		STA R	2.X	
05030	040D	F7 03		STA B	3.X	
05040	040F	BD 0DE4		JSR	GETPAR	
05050	0412	25 03		RCS	TSTV44	
05060	0414	5F		CLR B		
05070	0415	E7 03		STA B	3.X	
05080	0417	A6 02	TSTV44	LDA A	2.X	
05090	0419	BD 0F41		JSR	DIMCAL	
05100	041C	6D 03		TST	3.X	
05110	041E	27 D2		RFQ	TSTV33	
05120	0420	20 CC		RRA	TSTV31	
05130	0422	8D 33	TSTV5	RSR	INXATF	
05140	0424	DF 46		STX	ARRTAR	
05150	0426	DF 63		LIX	VARPNT	
05160	0428	20 AA		BRA	TSTV11	
05190	042A	81 41	TSTLTR	CMP A	#S41	
05200	042C	2P 0C		RMI	NONO	
05210	042E	81 5A		CMP A	#S5A	
05220	0430	2F 0A		RLF	YESNO	
05240	0432	81 30	TFSTNO	CMP A	#S30	
05250	0434	2B 04		BMI	NONO	
05260	0436	81 39		CMP A	#S39	
05270	0438	2F 02		RLF	YESNO	
05280	043A	0D	NONO	SFC		
05290	043B	39		RTS		
05300	043C	0C	YESNO	CLC		
05310	043D	39		RTS		
05330	043E	DF 5D	PUSHAF	LIX	AESTK	
05340	0440	8D 16		RSR	INXSFV	
05350	0442	DF 5D		STX	AESTK	
05360	0444	39		RTS		
05380	0445	BD 02B2	PSHXPA	JSR	PUSHX	
05400	0448	DF 5D	PULLAF	LIX	AESTK	
05410	044A	8D 03		RSR	DEXSFV	
05420	044C	20 F4		RRA	*-10	
05440	044E	09	DEXATE	DEF		
05450	044F	09	DEXSFV	DEF		

```

05460 0450 09      DEXSIX DFX
05470 0451 09      DEXFIV DFX
05480 0452 09      DEXFOR DFX
05490 0453 09      DFX
05500 0454 09      DFX
05510 0455 09      DFX
05520 0456 39      RTS

```

```

05540 0457 08      INXATF INX
05550 0458 08      INXSFV INX
05560 0459 08      INXSIX INX
05570 045A 08      INXFIV INX
05580 045B 08      INXFOR INX
05590 045C 08      INX
05600 045D 08      INX
05610 045E 08      INX
05620 045F 9C 44  QVRTST MEMFND
05630 0461 2B 03      BMI    **5
05640 0463 7E 09FR  JMP    OVERFL
05650 0466 39      RTS

```

```

05670 0467 BD 0445  FIX  JSR  PSHXPA
05680 046A A6 06      LDA A 6.X
05690 046C 81 04      CMP A #4
05700 046E 2F 05      RLF  **7
05710 0470 C6 01      FIXERR LDA B #1
05720 0472 7E 0828  JMP  FRROR
05730 0475 4F      CLR A
05740 0476 5F      CLR B
05750 0477 6D 06      FIX2  TST 6.X
05760 0479 2F 19      BLF  FIXFX
05770 047B 58      ASL B
05780 047C 49      ROL A
05790 047D D7 52      STA B BNUMR
05800 047F 97 51      STA A ANUMR
05810 0481 58      ASL B
05820 0482 49      ROL A
05830 0483 58      ASL B
05840 0484 49      ROL A
05850 0485 DR 52      ADD R BNUMB
05860 0487 99 51      ADC A ANUMR
05870 0489 ER 00      ADD B 0.X
05880 048B 89 00      ADC A #0
05890 048D RD 0552  JSR  ALLFFT
05900 0490 6A 06      DFC 6.X
05910 0492 20 F3      RRA  FIX2
05920 0494 4D      FIXFX TST A
05930 0495 26 D9      BNE  FIXFRR
05940 0497 7E 02C7  JMP  PULLX

```

```

05960 049A RD 0744  FACT  JSR  SKIPSP
05970 049D RD 03C1  JSR  TSTV
05980 04A0 25 04      RCS  **6
05990 04A2 BD 0309  JSR  IND
06000 04A5 39      RTS
06010 04A6 BD 0B28  JSR  TSTN
06020 04A9 25 01      RCS  **3
06030 04AB 39      RTS
06040 04AC 81 2B      CMP A #'(
06050 04AE 26 11      RNF  FACT3
06060 04B0 08      INX

```

```

06070 04B1 BD 33      PAREXP RSR  EXPR
06080 04B3 BD 0744  JSR  SKIPSP
06090 04B6 81 29      CMP A #' )
06100 04B8 26 02      RNF  FACT?
06110 04BA 08      INX
06120 04BB 39      RTS
06130 04BC C6 13      FACT2 LDA R #S13
06140 04BE 7F 0B28  JMP  ERROR
06150 04C1 C6 06      FACT3 LDA B #S06
06160 04C3 20 F9      RRA  *-5

```

```

06180 04C5 BD D3      TERM  BSR  FACT
06190 04C7 RD 0744  TERMO JSR  SKIPSP
06200 04CA 81 2A      CMP A #'*
06210 04CC 26 0B      RNF  TERM1
06220 04CE 08      INX
06230 04CF RD 0744  JSR  SKIPSP
06240 04D2 8D C6      BSR  FACT
06250 04D4 RD 06D1  JSR  MPY
06260 04D7 20 FF      RRA  TERMO
06270 04D9 81 2F      TERM1 CMP A #' /
06280 04DB 26 08      BNF  TERM2
06290 04DD 08      INX
06300 04DE 8D BA      BSR  FACT
06310 04E0 RD 0659  JSR  DIV
06320 04E3 20 F2      RRA  TERMO
06330 04E5 39      TERM2 RTS

```

```

06350 04E6 RD 0744  EXPR  JSR  SKIPSP
06360 04E9 81 2D      CMP A #' -
06370 04EB 26 07      RNF  **0
06380 04ED 08      INX
06390 04EE 8D D5      BSR  TERM
06400 04F0 8D 2D      BSR  NEG
06410 04F2 20 07      RRA  EXPR1
06420 04F4 81 2B      CMP A #' +
06430 04F6 26 01      RNF  **3
06440 04F8 08      INX
06450 04F9 8D CA      RSR  TERM
06460 04FB BD 0744  EXPR1 JSR  SKIPSP
06470 04FE 81 2B      CMP A #' +
06480 0500 26 08      RNF  **10
06490 0502 08      INX
06500 0503 8D C0      RSR  TERM
06510 0505 BD 05FC  JSR  ADD
06520 0508 20 F1      BRA  EXPR1
06530 050A 81 2D      CMP A #' -
06540 050C 26 08      RNF  **10
06550 050E 08      INX
06560 050F 8D R4      BSR  TERM
06570 0511 BD 05F0  JSR  SUB
06580 0514 20 F5      BRA  EXPR1
06590 0516 39      RTS

```

```

06610 0517 BD 02B2  NEGDP JSR  PUSHX
06620 051A BD 0458  JSR  INXSFV
06630 051D 20 0C      BRA  **14

```

```

06640 051F BD 02B2 NEG JSR PIJSHX
06650 0522 DF 5D LDX AFSTK
06660 0524 09 DFX
06670 0525 09 DFX
06680 0526 20 06 BRA **8
06690 0528 RD 02B2 NEGACC JSR PIJSHX
06700 052B RD 045A JSR INXFIV
06710 052E 36 PSH A
06720 052F 37 PSH R
06730 0530 C6 06 LDA R #6
06740 0532 86 99 NEGALL LDA A #S99
06750 0534 A0 00 SHR A 0.X
06760 0536 A7 00 STA A 0.X
06770 0538 09 DFX
06780 0539 5A DFC R
06790 053A 26 F6 BNF NEGALL
06800 053C BD 0459 JSR INXSIX
06810 053F 0D SFC
06820 0540 C6 06 LDA R #6
06830 0542 86 00 NEGA2 LDA A #S00
06840 0544 A9 00 ADC A 0.X
06850 0546 19 DAA
06860 0547 A7 00 STA A 0.X
06870 0549 09 DFX
06880 054A 5A DFC R
06890 054B 26 F5 BNF NEGA2
06900 054D 33 PIJL R
06910 054E 32 PIJL A
06920 054F 7F 02C7 JMP PIJLLX

```

```

06940 0552 36 ALLFFT PSH A
06950 0553 8D 08 BSR LLFFT
06960 0555 A6 00 LDA A 0.X
06970 0557 84 0F AND A #S0F
06980 0559 A7 00 STA A 0.X
06990 055B 32 PIJL A
07000 055C 39 RTS

```

```

07020 055D 36 LLLEFT PSH A
07030 055E 37 PSH R
07040 055F DF 57 STX XLONG
07050 0561 C6 04 LDA R #4
07060 0563 DF 57 LLFFT2 LDX XLONG
07070 0565 BD 045A JSR INXFIV
07080 0568 68 00 ASL 0.X
07090 056A 86 05 LDA A #5
07100 056C 09 LLLEFT3 DFX
07110 056D 69 00 ROL 0.X
07120 056F 4A DFC A
07130 0570 26 FA BNF LLFFT3
07140 0572 5A DFC B
07150 0573 26 FF BNF LLFFT2
07160 0575 33 PIJL R
07170 0576 32 PIJL A
07180 0577 39 RTS

```

```

07200 0578 36 LRIGHT PSH A
07210 0579 37 PSH R
07220 057A DF 57 STX XLONG
07230 057C A6 00 LDA A 0.X

```

```

07240 057E 84 F0 AND A #SFO
07250 0580 97 53 STA A LRNTNF
07260 0582 C6 04 LDA R #4
07270 0584 DF 57 LRIGH2 LDX XLONG
07280 0586 64 00 LSR 0.X
07290 0588 86 05 LDA A #5
07300 058A 08 LRIGH3 INX
07310 058B 66 00 ROR 0.X
07320 058D 4A DFC A
07330 058F 26 FA BNF LRIGH3
07340 0590 5A DFC R
07350 0591 26 F1 BNF LRIGH2
07360 0593 DF 57 LDX XLONG
07370 0595 A6 00 LDA A 0.X
07380 0597 9B 53 ADD A LRNTNF
07390 0599 A7 00 STA A 0.X
07400 059B 33 PIJL R
07410 059C 32 PIJL A
07420 059D 39 RTS

```

```

07440 059E RD 043F NORMPA JSR PIJSHAF

```

```

07460 05A1 RD 0445 NORM JSR PSHXPA
07470 05A4 7F 0056 CLR NORSGN
07480 05A7 RD 0DA4 JSR TSTZFR
07490 05AA 24 19 RCC NORMFZ
07500 05AC A6 00 LDA A 0.X
07510 05AE 2A 0R RPL NORMI
07520 05B0 97 56 STA A NORSGN
07530 05B2 RD 0528 JSR NEGACC
07540 05B5 86 0F NORMOO LDA A #S0F
07550 05B7 A4 00 AND A 0.X
07560 05B9 A7 00 STA A 0.X
07570 05BB A6 00 NORMI LDA A 0.X
07580 05BD 26 08 BNF NORMFX
07590 05BF 8D 91 RSR ALLFFT
07600 05C1 6A 06 DFC 6.X
07610 05C3 20 F6 RRA NORMI
07620 05C5 6F 06 NORMFZ CLR 6.X
07630 05C7 6F 05 NORMFX CLR 5.X
07640 05C9 A6 06 LDA A 6.X
07650 05CB 81 65 CMP A #101
07660 05CD 2D 15 BLT NORMFO
07670 05CF 37 PSH R
07680 05D0 C6 05 LDA R #5
07690 05D2 86 99 LDA A #S99
07700 05D4 A7 00 NORMO1 STA A 0.X
07710 05D6 08 INX
07720 05D7 5A DFC R
07730 05D8 26 FA BNF NORMO1
07740 05DA 86 64 LDA A #100
07750 05DC A7 01 STA A 1.X
07760 05DE RD 0451 JSR DEXFIV
07770 05E1 33 PIJL R
07780 05E2 20 D1 RRA NORMOO
07790 05E4 81 9D NORMFO CMP A #-99
07800 05E6 2F 03 BGT NORMF1
07810 05E8 BD 0BDD JSR CLRACC
07820 05EB 7D 0056 NORME1 TST NORSGN
07830 05EE 27 03 BFO NORMF2
07840 05F0 BD 0528 JSR NEGACC
07850 05F3 BD 043F NORME2 JSR PIJSHAF
07860 05F6 7F 02C7 JMP PIJLLX

```

```

07880 05F9 BD 051F SUR JSR NFG

```

```

07900 05FC RD 0445 ADD JSR PSHXPA
07910 05FF RD 0DA4 JSR TSTZFR
07020 0602 24 25 RCC ADD4
07930 0604 RD 0578 JSR LRIGHT
07940 0607 6C 06 INC 6.X
07950 0609 BD 0448 JSR PULLAF
07960 060C BD 0DA4 JSR TSTZFR
07970 060F 24 11 RCC ADD1
07980 0611 BD 0578 ADD0 JSR LRIGHT
07990 0614 6C 06 INC 6.X
08000 0616 A6 06 LDA A 6.X
08010 0618 A1 0D CMP A 13.X
08020 061A 27 08 BFO ADD2
08030 061C 2D F3 BLT ADD0
08040 061F 8D 0C RSR SWAP
08050 0620 20 FF RRA ADD0
08060 0622 8D 08 ADD1 BSR SWAP
08070 0624 8D 1F ADD2 BSR ADDFR
08080 0626 RD 059F JSR NTRMPA
08090 0629 7F 02C7 ADD4 JMP PULLX

```

```

08110 062C DF 5D SWAP LDX AFSTK
08120 062E 36 PSH A
08130 062F 37 PSH R
08140 0630 C6 07 LDA R #7
08150 0632 A6 00 SWAP1 LDA A 0.X
08160 0634 36 PSH A
08170 0635 A6 07 LDA A 7.X
08180 0637 A7 00 STA A 0.X
08190 0639 32 PIUL A
08200 063A A7 07 STA A 7.X
08210 063C 08 INX
08220 063D 5A DEC R
08230 063E 26 F2 BNF SWAP1
08240 0640 DF 5D SWAP2 LDX AFSTK
08250 0642 33 PIUL R
08260 0643 32 PIUL A
08270 0644 39 RTS

```

```

08290 0645 36 ADDFR PSH A
08300 0646 37 PSH R
08310 0647 DF 5D LDX AFSTK
08320 0649 C6 06 LDA R #6
08330 064B 0C CLC
08340 064C A6 05 ADDER1 LDA A 5.X
08350 064E A9 0C ADC A 12.X
08360 0650 19 DAA
08370 0651 A7 05 STA A 5.X
08380 0653 09 DFX
08390 0654 5A DEC B
08400 0655 26 F5 BNF ADDFR1
08410 0657 20 F7 BRA SWAP2

```

```

08430 0659 BD 02B2 DIV JSR PUSHX
08440 065C BD 0BDD JSR CLRACC
08450 065F BD 0448 JSR PULLAF
08460 0662 BD 0DA4 JSR TSTZFR

```

```

08470 0665 25 05 RCS **7
08480 0667 C6 08 LDA R #S0B
08490 0669 7F 082B JMP FRROR
08500 066C 60 06 NFO 6.X
08510 066E 8D 38 RSR MDSIGN
08520 0670 BD 0578 JSR LRIGHT
08530 0673 BD 0517 JSR NFGOP
08540 0676 86 0R LDA A #11
08550 0678 5F DIV2 CLR R
08560 0679 5C DIV3 INC B
08570 067A 8D C9 BSR ADDFR
08580 067C 6D 00 TST 0.X
08590 067E 2A F9 BPL DIV3
08600 0680 5A DEC B
08610 0681 BD 0517 JSR NFGOP
08620 0684 8D BF BSR ADDFR
08630 0686 BD 0517 JSR NFGOP
08640 0689 BD 0458 JSR INXSEV
08650 068C BD 0458 JSR INXSEV
08660 068F BD 055D JSR LLFFT
08670 0692 FB 05 ADD R 5.X
08680 0694 F7 05 STA B 5.X
08690 0696 DF 5D LDX AFSTK
08700 0698 RD 055D JSR LLFFT
08710 069B 4A DEC A
08720 069C 26 DA BNF DIV2
08730 069E RD 043F JSR PUSHA
08740 06A1 8D 89 RSR SWAP
08750 06A3 7C 0050 INC TNUMP
08760 06A6 20 56 RRA MDEXIT
08780 06A8 BD 0578 MDSIGN JSR LRIGHT
08790 06AB 4F CLR A
08800 06AC 6D 00 TST 0.X
08810 06AE 2A 05 RPL MDS2
08820 06B0 BD 0528 JSR NFGACC
08830 06B3 86 80 LDA A #S80
08840 06B5 RD 0448 MDS2 JSR PULLAF
08850 06B8 6D 00 TST 0.X
08860 06BA 2A 05 RPL MDS3
08870 06BC BD 0528 JSR NFGACC
08880 06BF 8B 80 ADD A #S80
08890 06C1 97 4A MDS3 STA A TSIGN
08900 06C3 A6 06 LDA A 6.X
08910 06C5 AB 0D ADD A 13.X
08920 06C7 28 05 RVC MDS4
08930 06C9 86 78 LDA A #120
08940 06CB 24 01 BCC MDS4
08950 06CD 40 NEG A
08960 06CE 97 50 MDS4 STA A TNUMB
08970 06D0 39 RTS

```

```

08990 06D1 BD 0445 MPY JSR PSHXPA
09000 06D4 8D D2 RSR MDSIGN
09010 06D6 DF 59 STX MPYXX
09020 06D8 BD 043F JSR PUSHA
09030 06DB RD 062C JSR SWAP
09040 06DE BD 0RDD JSR CLRACC
09050 06E1 86 09 LDA A #9
09060 06E3 DF 59 MPY4 LDX MPYXX
09070 06E5 E6 00 LDA R 0.X
09080 06E7 RD 0552 JSR ALLFFT
09090 06EA 5D MPY5 TST R
09100 06EB 27 06 BFO MPY6
09110 06ED RD 0645 JSR ADDFR
09120 06F0 5A DEC B

```

```

09130 06F1 20 F7      RRA      MPY5
09140 06F3 DF 5D      MPY6   LDX      AFSSTK
09150 06F5 BD 0458    JSR      INXSFFV
09160 06F8 BD 0578    JSR      LRIGHT
09170 06FB 4A        DFC A
09180 06FC 26 F5      RNF      MPY4
09190 06FE BD 0448    MDEXIT JSR      PULLAF
09200 0701 BD 062C    JSR      SWAP
09210 0704 96 50      LDA A    TNUMR
09220 0706 A7 06      STA A    6.X
09230 0708 RD 059F    JSR      NORMPA
09240 070B 7D 004A    TST      TSIGN
09250 070E 2A 03      BPL      MDEX2
09260 0710 RD 051F    JSR      NEG
09270 0713 7E 02C7    MDEX2   JMP      PULLX

```

```

09300 0716 96 32      FINDND  LDA A    HIGHLN
09310 0718 D6 33      LDA B    HIGHLN+
09320 071A D0 31      SJR B    PACKLN+
09330 071C 92 30      SRC A    PACKLN
09340 071E 25 1E      RCS      HIRALL
09350 0720 DF 2F      FINDNI  LDX      SOURCEF
09360 0722 96 30      FINDO   LDA A    PACKLN
09370 0724 D6 31      LDA B    PACKLN+
09380 0726 F0 01      SJR B    1,X
09390 0728 A2 00      SRC A    0,X
09400 072A 25 14      BCS      **22
09410 072C 26 03      BNF      **5
09420 072E 5D        TST B
09430 072F 27 10      BFO      **18
09440 0731 08        INX
09450 0732 08        INX
09460 0733 A6 00      LDA A    0,X
09470 0735 81 1F      CMP A    $S1F
09480 0737 26 F9      RNF      *-5
09490 0739 08        INX
09500 073A 9C 2A      CPX      NEXTBA
09510 073C 26 F4      RNF      FINDO
09520 073E DF 2A      HITBALL LDX      NEXTRA
09530 0740 0D        SFC
09540 0741 DF 2C      STX      WORKRA
09550 0743 39        RTS

```

```

09570 0744 A6 00      SKIPSP  LDA A    0,X
09580 0746 81 20      CMP A    $S20
09590 0748 26 03      BNF      **5
09600 074A 08        INXSKP  INX
09610 074B 20 F7      BRA      SKIPSP
09620 074D 39        RTS

```

```

09640 074E BD 0B28    LINEND  JSR      TSTN
09650 0751 24 05      BCC      **7
09660 0753 C6 07      LINFO   LDA B    #7
09670 0755 7E 0828    JMP      ERROR
09680 0758 DF AC      STX      BUFNXT
09690 075A BD 0448    JSR      PULLAF
09700 075D E6 06      LDA R    6,X
09710 075F 23 F2      BLS      LINEO
09720 0761 C1 04      CMP R    #4
09730 0763 2F FE      BGT      LINFO
09740 0765 86 05      LDA A    #5
09750 0767 10        SRA
09760 0768 4D        TST A
09770 0769 27 06      BFO      **8
09780 076B RD 0578    JSR      LRIGHT
09790 076E 4A        DFC A

```

```

09800 076F 20 F7      BRA      *-7
09810 0771 A6 01      LDA A    1,X
09820 0773 97 30      STA A    PACKLN
09830 0775 A6 02      LDA A    2,X
09840 0777 97 31      STA A    PACKLN+1
09850 0779 DE AC      LDX      BUFNXT
09860 077B 0C        CLC
09870 077C 39        RTS

```

```

09890 077D DF 34      NXTLIN  LDX      BASPNT
09900 077F A6 00      LDA A    0,X
09910 0781 08        INX
09920 0782 81 1F      CMP A    $S1F
09930 0784 26 F9      RNF      *-5
09940 0786 DF 36      STX      BASLIN
09950 0788 39        RTS

```

```

09970 0789 DF AC      FCODE   STX      RUFNXT
09980 078B 9F 28      STS      SAVFSP
09990 078D CF 01BC    LDX      #FIJNCT-1
10000 0790 20 12      BRA      LOOP3

```

```

10020 0792 DF AC      CCINT   STX      RUFNXT
10030 0794 9F 28      STS      SAVFSP
10040 0796 CF 0105    LDX      #INTFR-1
10050 0799 20 09      BRA      LOOP3

```

```

10070 079B 8D A7      CCODE   BSR      SKIPSP
10080 079D DF AC      STX      RUFNXT
10090 079F 9F 28      STS      SAVFSP
10100 07A1 CF 0111    LDX      #COMMAN-1
10110 07A4 9F AC      LOOP3   LDS      BUFNXT
10120 07A6 34        DFS
10130 07A7 08        INX
10140 07A8 32        PUL A
10150 07A9 81 20      CMP A    $S20
10160 07AB 27 FB      BFO      *-3
10170 07AD F6 00      LDA B    0,X
10180 07AF C1 1F      CMP R    $S1F
10190 07B1 27 18      BFO      **26
10200 07B3 11        CBA
10210 07B4 27 F1      BFO      *-13
10220 07B6 08        LOOP5   INX
10230 07B7 8C 01BA    CPX      #COMFND

```

```

10240 07BA 27 17      BFO      CCFXIT
10250 07BC 8C 01ED    CPX      #FUNFND
10260 07BF 27 18      BFO      FIJNCFX
10270 07C1 F6 00      LDA B    0,X
10280 07C3 C1 1F      CMP R    $S1F
10290 07C5 26 FF      BNF      LOOP5
10300 07C7 08        INX
10310 07C8 08        INX
10320 07C9 20 D9      BRA      LOOP3
10330 07CB 08        INX
10340 07CC 9F AC      STS      BUFNXT
10350 07CE 9F 34      STS      BASPNT
10360 07D0 9F 28      LDS      SAVFSP
10370 07D2 39        RTS
10380 07D3 9F 28      CCFXIT  LDS      SAVFSP

```

```

10390 07D5 CF 01BB      LDX  #IMPLFT
10400 07D8 39          RTS
10410 07D9 7F 0B18     JMP  DRLLTR
10440 07DC CF 1200     LDX  #FILE
10450 07DF DF 2F       STX  SOURCE
10460 07E1 DF 2A       STX  NEXTRA
10470 07E3 DF 46       STX  ARRTAR
10480 07E5 4F         CLR  A
10490 07E6 97 32       STA  A  HIGHLN
10500 07E8 97 33       STA  A  HIGHLN+1
10510 07FA 97 40       STA  A  PRCNT
10520 07EC CF 1078     LDX  #DUMRTS
10530 07FF DF 67       STX  DIMMY
10540 07F1 39          RTS

```

```

10560          *START OF MAIN PROGRAM
10570          *
10580          *
10590 07F2 8D E8       START BSR  NEWSUR
10600 07F4 20 65       BRR  RIN
10610 07F6 8F A045     RFADY LNS  #SA045
10620 07F9 CF A07F     LDX  #SA07F
10630 07FC DF 3A       STX  XSTACK
10640 07FE 7F 0085     CLR  L7AFLG
10650 0801 CE 01F0     LDX  #RDYMSG
10660 0804 BD 0292     JSR  OUTPUT
10670 0807 7F 0040     NFWLIN CLR  PRCNT
10680 080A BD 025D     JSR  CNTLIN
10690 080D CF 00R0     LDX  #RIFFFFR
10700 0810 BD 0744     JSR  SKIPSP
10710 0813 BD 0432     JSR  TFSTND
10720 0816 25 05       RCS  **7
10730 0818 RD 0949     JSR  NIIMRFR
10740 081R 20 FA       BRR  NFWLIN
10750 081D 81 1F       CMP  A  #S1F
10760 081F 27 F6       RFO  NFWLIN
10770 0821 RD 079R     JSR  CODDF
10780 0824 FF 00       LDX  0.X
10790 0826 6F 00       JMP  0.X

```

```

10810 0828 8F A045     FRROR LNS  #SA045
10820 082B BD 02A0     JSR  CRLF
10830 082E CF 020C     LDX  #FRRMS1
10840 0831 RD 0299     JSR  O'ITNCR
10850 0834 D7 20       STA  B  INDFX1
10860 0836 CF 0020     LDX  #INDFX1
10870 0839 BD 0270     JSR  O'IT2H
10880 083C CF 0214     LDX  #FRRMS2
10890 083F RD 0299     JSR  O'ITNCR
10900 0842 DF 5B       LDX  RASNIM
10910 0844 FF 00       LDX  0.X
10920 0846 DF 30       STX  PACKLN
10930 0848 CF 0030     LDX  #PACKLN
10940 084R 96 34       LDA  A  BASPNT
10950 084D 26 04       RNF  **6
10960 084F 6F 00       CLR  0.X
10970 0851 6F 01       CLR  1.X
10980 0853 RD 0273     FRROR2 JSR  O'IT4HS
10990 0856 RD 02A0     JSR  CRLF
11000 0859 20 9R       BRR  READY

```

```

11020 085B DF 2F      RUN  LDX  SOURCE WITH TEXT BUFFER POINTER
11030 085D DF 3C      STX  DATPNT
11040 085F DF 36      STX  RASLIN
11050 0861 CF 0075     LDX  #SRRSTK
11060 0864 DF 65      STX  SRRPNT
11070 0866 CF 1180     LDX  #FORSTK
11080 0869 DF 5F      STX  FORPNT
11090 086B DF 2A      LDX  NEXTRA
11100 086D DF 46      STX  ARRTAR
11110 086F 4F         CLR  A
11120 0870 09         DFX
11130 0871 08         INX
11140 0872 A7 00      STA  A  0.X
11150 0874 9C 44      CPX  MEMFND
11160 0876 27 07      RFO  **9
11170 0878 A6 00      LDA  A  0.X
11180 087A 27 F5      RFO  *-9
11190 087C 09         DFX
11200 087D DF 44      STX  MEMFND
11210 087F 7F 0A00    JMP  BASIC

```

```

11230 0882 CF 0200     CLIST LDX  #PGCNTL
11240 0885 RD 0292     JSR  OUTPUT
11250 0888 DF 34       LDX  RASPNT
11260 088A BD 0744     JSR  SKIPSP
11270 088D 81 1F       CMP  A  #S1F
11280 088F 27 2B       RFO  CLIST4
11290 0891 RD 074F     JSR  LINFNO
11300 0894 DF 34       STX  RASPNT
11310 0896 RD 0720     JSR  FINDN1
11320 0899 DF 24       STX  INDFX3
11330 089B DF 34       LDX  BASPNT
11340 089D RD 0744     JSR  SKIPSP
11350 08A0 81 1F       CMP  A  #S1F
11360 08A2 27 04       RFO  CLIST3
11370 08A4 08         INX
11380 08A5 RD 074F     JSR  LINFNO
11390 08A8 4F         CLIST3 CLR  A
11400 08A9 C6 01      LDA  B  #1
11410 08AB DR 31      ADD  B  PACKLN+
11420 08AD 19         DAA
11430 08AE D7 31      STA  B  PACKLN+
11440 08B0 99 30      ADC  A  PACKLN
11450 08B2 19         DAA
11460 08B3 97 30      STA  A  PACKLN
11470 08B5 BD 0720     JSR  FINDN1
11480 08B8 DE 24       LDX  INDFX3
11490 08BA 20 06       BRA  CLIST5
11500 08BC DE 2A       CLIST4 LDX  NEXTRA
11510 08BE DF 2C       STX  WRKBA
11520 08C0 DE 2F       LDX  SOURCE
11530 08C2 9C 2C       CLIST5 CPX  WRKBA
11540 08C4 27 0A       RFO  CLFXT
11550 08C6 9C 2A       CPX  NEXTRA
11560 08C8 27 06       RFO  CLFXT
11570 08CA BD 032A     JSR  OUTLIN
11580 08CD 08         INX
11590 08CE 20 F2      RRR  CLIST5
11600 08D0 7F 0DDF    JMP  CLFXT REMARK

```

```

11620 08D3 RD 077D     PATCH JSR  NXLIN
11630 08D6 CF 0A00     LDX  #BASIC
11640 08D9 FF A046     STX  SA046

```

```

11650 08DC 8F A040   LDS   #S A040
11660 08DF 8F A008   STS   $A008
11670 08E2 7E F0E3   JMP   $F0E3

```

```

11690 08E5 DE 2F   SAVF  LDX   SOURCEF
11700 08E7 86 12   LDA  A   #S12
11710 08E9 BD 0276  JSR   OUTCH
11720 08EC 8D 45   RSR   DELAY2
11730 08EE 8D 43   RSR   DELAY2
11740 08F0 8D 4C   RSR   LEADFR
11750 08F2 9C 2A   SAVE1 CPX   NEXTRA
11760 08F4 27 0D   RFO   SAVEF5
11770 08F6 86 02   LDA  A   #S02
11780 08F8 BD 0276  JSR   OUTCH
11790 08FB BD 032A  JSR   OUTLIN
11800 08FE 08      INX
11810 08FF 8D 34   BSR   DELAY
11820 0901 20 FF   BRA   SAVE1
11830 0903 86 03   SAVE5 LDA  A   #S03
11840 0905 BD 0276  JSR   OUTCH
11850 0908 8D 34   BSR   LEADFR
11860 090A 7E 07F6  JMP   READY

```

```

11880 090D BD 07DC  LOAD  JSR   NEWSUR
11890 0910 86 11   APPEND LDA A   #S11
11900 0912 97 85   STA  A   L7AFLG
11910 0914 RD 0276  JSR   OUTCH
11920 0917 RD 027B  LOAD1 JSR   INCH
11930 091A 81 02   CMP  A   #S02
11940 091C 26 F9   BNF   L7AD1
11950 091E BD 027R  JSR   INCH
11960 0921 BD 0432  JSR   TESTND
11970 0924 25 F4   RCS   L7AD1+3
11980 0926 CF 00R0  LDX   #R1FFFF
11990 0929 BD 0231  JSR   KEYBD0+5
12000 092C CF 00R0  LDX   #R1FFFF
12010 092F 8D 18   BSR   NUMBER
12020 0931 20 F4   BRA   L7AD1

```

```

12040 0933 8D 00   DELAY2 BSR   DELAY

```

```

12060 0935 5F      DELAY CLR  R
12070 0936 4F      CLR  A
12080 0937 4A      DELAY3 DEC  A
12090 0938 26 FD   RNF
12100 093A 5A      DFC  B
12110 093B 26 F9   BNF   DELAY+1
12120 093D 39      RTS

```

```

12140 093F C6 32   LEADFR LDA R   #50
12150 0940 86 FF   LDA  A   #5FF
12160 0942 BD 0276  JSR   OUTCH
12170 0945 5A      DFC  B
12180 0946 26 F8   BNF   LEADFR+2
12190 0948 39      RTS

```

```

12210 0949 BD 074F  NUMBR JSR   LINFN0
12220 094C RD 0716  NUM1  JSR   FINDNO
12230 094F 24 13   BCC   DELRFP
12240      * YOU HAVE IDENTICAL NUMBER AND U MUST DELETE AND IN
12250 0951 DF AC   LDX   BIJFNXT
12260 0953 BD 0744  JSR   SKIPSP
12270 0956 81 1F   CMP  A   #S1F
12280 0958 27 21   BFO   NEXIT
12290 095A DF 2C   LDX   WORKBA
12300 095C 9C 2A   CPX   NEXTBA
12310 095E 27 1C   BFO   CAPPFN   IF AT END FILE
12320      * YOU MERELY APPEND
12330      * OTHERWISE JUST INSERT AND RETURN
12340 0960 8D 48   RSR   INSERT
12350 0962 20 17   RRA   NEXIT
12360 0964 DF AC   DELRFP LDX   BIJFNXT   GET FIRST CHARACTER
12370 0966 RD 0744  JSR   SKIPSP   AFTER LIN
12380 0969 81 1F   CMP  A   #S1F   IS FIRST CHARACTER RS
12390 096B 26 0A   BNF   REPLAC   IF NOT YOU ARE
12400 096D DF 2A   LDX   NEXTBA   REPLACING LINE
12410 096F 9C 2F   CPX   SOURCEF
12420 0971 27 08   RFO   NEXIT
12430 0973 8D 0F   BSR   DELETE   OTHERWISE ONLY DELETE
12440 0975 20 04   BRA   NEXIT   AND RET
12450 0977 8D 0B   REPLAC BSR   DELETE
12460 0979 8D 2F   BSR   INSERT
12470 097B 39      NEXIT RTS
12480 097C 8D 2C   CAPPEN BSR   INSERT
12490 097E DE 30   LDX   PACKLN   AND MAKE THIS LINE
12500 0980 DF 32   STX   HIGHLN   NEW HIGH L
12510 0982 20 F7   RRA   NEXIT

```

```

12530 0984 9F 28   DELFTF STS   SAVFSP
12540 0986 DF 2C   LDX   WORKRA
12550 0988 9F 2A   LDS   NEXTRA
12560 098A 08      INX
12570 098B 08      INX
12580 098C 34      DFS
12590 098D 34      DFS
12600 098E A6 00   DEL2  LDA  A   0.X
12610 0990 34      DFS
12620 0991 08      INX
12630 0992 81 1F   CMP  A   #S1F   IF NOT RS, DO IT AGAIN
12640 0994 26 F8   BNF   DEL2
12650 0996 9F 2A   STS   NEXTRA   STACK WAS DECREMENTED
12660 0998 9F 46   STS   ARRTAB   BY # OF
12670 099A 35      TXS
12680 099B DE 2C   LDX   WORKBA
12690 099D 9C 2A   DEL4  CPX   NEXTBA
12700 099F 27 06   RFO   DFLFX
12710 09A1 32      PUL  A
12720 09A2 A7 00   STA  A   0.X   HONEST 0 OFFSET
12730 09A4 08      INX
12740 09A5 20 F6   BRA   DEL4
12750 09A7 9F 28   DELEX LDS   SAVFSP
12760 09A9 39      RTS

```

```

12780 09AA DF AC   INSERT LDX   BIJFNXT
12790 09AC RD 079B  JSR   CCONF
12800 09AF 9F 28   STS   SAVFSP
12810 09B1 DF 48   STX   KEYWD
12820 09B3 DF 2A   LDX   NEXTRA
12830 09B5 DF 20   STX   INDFX1
12840 09B7 D6 AF   LDA  B   FNDRUF+1

```

```

12850 09B9 D0 AD      SIJB B RIJFNXT+1
12860 09BB CB 04      ADD B #S04
12870 09BD DB 2F      ADD B NEXTRA+1
12880 09BF 86 00      LDA A #S00
12890 09C1 99 2A      ADC A NEXTBA
12900 09C3 91 44      CMP A MFNFND
12910 09C5 22 34      RHI OVRFL
12920 09C7 D7 2R      STA R NEXTRA+1
12930 09C9 97 2A      STA A NEXTRA      THIS INCREMENTS NEXTRA
12940 09CB DF 2A      LDX NEXTRA      FOR DF
12950 09CD DF 46      STX ARRTAB
12960 09CF 35         TXS
12970 09D0 DE 20      LDX INDEXI
12980 09D2 9C 2C      CPX WWRKBA
12990 09D4 27 06      BFC RIJFWRT
13000 09D6 09         DFX
13010 09D7 A6 00      LDA A 0.X
13020 09D9 36         PSH A
13030 09DA 20 F6      BRR INS2
13040 09DC DE 2C      RUFWRT LDX WWRKBA      WE HAVE OPENED HOLE NOW
13050      *THE KEYRD BUFFER      CAN I
13060 09DE 96 30      LDA A PACKLN
13070 09E0 A7 00      STA A 0.X
13080 09E2 08         INX
13090 09E3 96 31      LDA A PACKLN+1
13100 09E5 A7 00      STA A 0.X
13110 09E7 08         INX
13120 09E8 96 49      LDA A KEYWD+1
13130 09EA A7 00      STA A 0.X
13140 09EC 08         INX
13150 09ED 9F AC      LDS RIJFNXT
13160 09EF 34         DFS
13170 09F0 32         PUL A
13180 09F1 A7 00      STA A 0.X
13190 09F3 08         INX
13200 09F4 81 1F      CMP A #S1F
13210 09F6 26 F8      RNF *-6
13220 09F8 9F 28      LDS SAVFSP
13230 09FA 39         RTS
13240 09FB C6 14      OVRFL LDA R #S14
13250 09FD 7F 082B    JMP FRROR

13270 0A00 CF 0112 BASIC LDX #COMMAN
13280 0A03 DF 48      STX KEYWD
13290 0A05 CF 1102    LDX #ASTACK
13300 0A08 DF 5D      STX AFSTK
13310 0A0A DF 36      LDX BASLIN
13320 0A0C DF 5R      STX BASNUM
13330 0A0E 9C 2A      CPX NEXTRA
13340 0A10 26 03      RNF **5
13350 0A12 7E 07F6    JMP READY
13360 0A15 7D 0036    TST BASLIN
13370 0A18 27 F8      RFO *-6
13380 0A1A 08         INX
13390 0A1B 08         INX
13400 0A1C A6 00      LDA A 0.X
13410 0A1E 08         INX
13420 0A1F DF 34      STX RASPNT
13430 0A21 97 49      STA A KEYWD+1
13440 0A23 DF 48      LDX KEYWD
13450 0A25 FF 00      LDX 0.X
13460 0A27 6F 00      JMP 0.X

```

```

13480 0A29 DF 34      THEN LDX RASPNT
13490 0A2B FD 0744    JSR SKIPSP
13500 0A2E FD 0432    JSR TESTND
13510 0A31 24 49      RCC GOT0
13520 0A33 7F 0FF2    JMP IF2

13540 0A36 C6 20      ONFRR LDA R #S20
13550 0A38 7F 0828    JMP FRROR

13570 0A3B DE 34      ONGD  LDX RASPNT
13580 0A3D FD 04E6    JSR EXPR
13590 0A40 FD 0467    JSR FIX
13600 0A43 5D         TST R
13610 0A44 27 F0      RFO ONFRR
13620 0A46 5A         DFC B
13630 0A47 D7 55      STA R ONLNDP
13640 0A49 BD 079R    JSR CONDF
13650 0A4C FF 00      LDX 0.X
13660 0A4E 8C 0A7C    CPX #GOTO
13670 0A51 27 2C      RFO GOTD+3
13680 0A53 8C 0A5A    CPX #GDSUR
13690 0A56 27 05      RFO GDSUR+3
13700 0A58 20 DC      BRA ONFRR

13720 0A5A 7F 0055 GDSUR CLR ONLNDP
13730 0A5D DF 36      LDX BASLIN
13740 0A5F FD 077D    JSR NYTLIN
13750 0A62 DF 65      LDX SRRPNT
13760 0A64 8C 0085    CPX #SRRSTK+16
13770 0A67 26 05      RNF GDSUR
13780 0A69 C6 09      LDA R #0
13790 0A6B 7F 0828    JMP FRROR
13800 0A6E 96 36      GDSUR LDA A BASLIN
13810 0A70 A7 00      STA A 0.X
13820 0A72 08         INX
13830 0A73 96 37      LDA A BASLIN+1
13840 0A75 A7 00      STA A 0.X
13850 0A77 08         INX
13860 0A78 DF 65      STX SRRPNT
13870 0A7A 20 03      BRA GOTD+3

13890 0A7C 7F 0055 GOTD CLR ONLNDP
13900 0A7F DF 34      LDX RASPNT
13910 0A81 FD 074F GOTD1 JSR LINFND
13920 0A84 7A 0055    DEC ONLNDP
13930 0A87 2B 0A      RMI **12
13940 0A89 FD 0744    JSR SKIPSP
13950 0A8C 81 2C      CMP A #7
13960 0A8E 26 A6      RNF ONFRR
13970 0A90 08         INX
13980 0A91 20 FF      BRR GOTD1
13990 0A93 FD 0716    JSR FINDND
14000 0A96 24 05      RCC GOTD2
14010 0A98 C6 07      LDA R #7
14020 0A9A 7F 0828    JMP FRROR
14030 0A9D DF 36      GOTD2 STX BASLIN
14040 0A9F 7F 0A0D    JMP BASIC

```

```

14060 0AA2 DE 65 RETURN LDX SRRPNT
14070 0AA4 8C 0075 CPX #SRRSTK
14080 0AA7 26 05 RNF **7
14090 0AA9 C6 10 LDA R #S10
14100 0AAB 7F 0828 JMP ERROR
14110 0AAE 09 DFX
14120 0AAF 09 DFX
14130 0AB0 DF 65 STX SRRPNT
14140 0AB2 EF 00 LDX O.X
14150 0AB4 20 E7 BRA GTT02

```

```

14180 0AB6 CF 01B3 STOP LDX #STOMSG
14190 0AB9 RD 0299 JSR OUTNCR
14200 0ABC RD 0350 JSR PRINSP
14210 0ABF DE 36 LDX RASLIN
14220 0AC1 7F 0853 JMP FRROR2

```

```

14240 0AC4 96 34 INPUT LDA A RASPNT
14250 0AC6 26 05 RNF **7
14260 0AC8 C6 02 LDA R #2
14270 0ACA 7E 0828 INPFERR JMP FRROR
14280 0ACD BD 0225 JSR KEYBD
14290 0ADO CE 00R0 LDX #BUFFFFR
14300 0AD3 DF AC STX BUFNXT
14310 0AD5 DE 34 LDX BASPNT
14320 0AD7 BD 03C1 INPUT1 JSR TSTV
14330 0ADA 25 31 BCS INPFX
14340 0ADC DF 34 STX RASPNT
14350 0ADE DE AC LDX BUFNXT
14360 0AE0 8D 3R INPUT2 BSR INNUM
14370 0AE2 24 12 BCC INPUT4
14380 0AE4 81 1E CMP A #S1E
14390 0AE6 27 06 BFO **8
14400 0AE8 CF 0203 LDX #WHAT
14410 0AEB RD 0292 JSR OUTPUT
14420 0AEE RD 0225 JSR KEYRD
14430 0AF1 CF 00B0 LDX #RUFFFFR
14440 0AF4 20 EA BRA INPUT2
14450 0AF6 RD 02E0 INPUT4 JSR STORF
14460 0AF9 RD 0744 JSR SKIPSP
14470 0AFC 81 2C CMP A #'
14480 0AFE 26 01 RNF INPUT5
14490 0B00 08 INX
14500 0B01 DF AC INPUT5 STX BUFNXT
14510 0B03 DF 34 LDX RASPNT
14520 0B05 RD 0744 JSR SKIPSP
14530 0B08 08 INX
14540 0B09 81 2C CMP A #'
14550 0B0B 27 CA BFO INPUT1
14560 0B0D 09 INPFX DFX
14570 0B0E 7F 0040 CLR PRCNT
14580 0B11 81 1E CMP A #S1F
14590 0B13 26 03 RNF DBLLTR
14600 0B15 7E 0DDF JMP RMARK
14610 0B18 C6 03 DBLLTR LDA R #3
14620 0B1A 7F 0828 JMP FRROR

```

```

14650 0B1D BD 0744 INNUM JSR SKIPSP
14660 0B20 97 4A STA A TSIGN
14670 0B22 08 INX
14680 0B23 81 2D CMP A #'
14690 0B25 27 04 BFO INNUM0
14700 0B27 09 DFX

```

```

14720 0B28 7F 004A TSTN CLR TSIGN
14730 0B2B 4F INNUM0 CLR A
14740 0B2C 97 4C STA A SIGDIG
14750 0B2E 97 4D STA A DFCFLG
14760 0B30 97 4E STA A TGGLE
14770 0B32 RD 0RDD JSR CLRACC
14780 0B35 D6 5D LDA B AFSTK
14790 0B37 D7 26 STA R INDFX4
14800 0B39 D6 5E LDA B AFSTK+1
14810 0B3B D7 27 STA B INDFX4+1
14820 0B3D C6 05 LDA B #5
14830 0B3F BD 0744 INTST1 JSR SKIPSP
14840 0B42 RD 0432 JSR TFSTND
14850 0B45 24 0R RCC INTST2-1
14860 0B47 81 2F CMP A #'
14870 0B49 26 05 BNF **7
14880 0B4B 97 4D STA A DFCFLG
14890 0B4D 08 INX
14900 0B4E 20 EF BRA INTST1
14910 0B50 0D SFC
14920 0B51 39 RTS
14930 0B52 09 DFX
14940 0B53 08 INTST2 INX
14950 0B54 A6 00 LDA A O.X
14960 0B56 BD 0432 JSR TFSTND
14970 0B59 25 42 RCS MAYFXP
14980 0B5B 80 30 SUR A #S30
14990 0B5D 26 0F RNF INTST3
15000 0B5F 7D 004C TST SIGDIG
15010 0B62 26 0C RNF INTST4
15020 0B64 7D 004D TST DFCFLG
15030 0B67 27 FA BFO INTST2
15040 0B69 7A 004P DFC FXP
15050 0B6C 20 F5 RRA INTST2
15060 0B6E 97 4C INTST3 STA A SIGDIG
15070 0B70 RD 02B2 INTST4 JSR PUSHX
15080 0B73 DE 26 LDX INDFX4
15090 0B75 5D TST B
15100 0B76 27 16 BFO INTST6
15110 0B78 7D 004F TST TGGLE
15120 0B7B 26 08 RNF INTST5
15130 0B7D AB 00 ADD A O.X
15140 0B7F A7 00 STA A O.X
15150 0B81 08 INX
15160 0B82 5A DFC B
15170 0B83 20 06 RRA INTS55
15180 0B85 48 INTST5 ASL A
15190 0B86 48 ASL A
15200 0B87 48 ASL A
15210 0B88 48 ASL A
15220 0B89 A7 00 STA A O.X
15230 0B8B 73 004F INTS55 CDM TGGLE
15240 0B8E DF 26 INTST6 STX INDFX4
15250 0B90 7D 004D TST DFCFLG
15260 0B93 26 03 RNF INTS66
15270 0B95 7C 004B INC FXP
15280 0B98 RD 02C7 INTS66 JSR PULLX
15290 0B9B 20 R6 RRA INTST2
15300 0B9D 7D 004D MAYFXP TST DFCFLG
15310 0BA0 26 08 RNF MAYFX2
15320 0BA2 81 2E CMP A #'

```

```

15330 DBA4 26 04      RNF    MAYFX2
15340 DBA6 97 4D      STA A  DFCFLG
15350 DBA8 20 A9      RRA    INTST2
15360 DBAA 5F          MAYFX2 CLR B
15370 DBAR 81 45      CMP A  #'F
15380 DBAD 26 14      RNF    INNFX
15390 DBAF 08          INX
15400 DBR0 A6 00      LDA A  0.X
15410 DBB2 81 2D      CMP A  #'-
15420 DBR4 27 09      BFO    INTST9
15430 DBR6 81 2B      CMP A  #'+'
15440 DBR8 26 01      RNF    INTST8
15450 DBRA 08          INX
15460 DBRB 8D 35      INTST8 BSR    FORMFX
15470 DBRD 20 04      BRA    INNFX
15480 DBRF 08          INTST9 INX
15490 DR00 8D 30      BSR    FORMFX
15500 DR02 50          NEG R
15510 DR03 96 4R      INNFX LDA A  EXP
15520 DR05 1B          ARA
15530 DR06 BD 02B2    JSR    PUSHX
15540 DR09 DE 5D      LDX    AFSTK
15550 DR0B A7 06      STA A  6.X
15560 DR0D BD 043F    JSR    PUSHAF
15570 DR0F BD 02C7    JSR    PULLX
15580 DR13 7D 004A    TST    TSIGN
15590 DR16 27 03      BFO    **5
15600 DR18 BD 051F    JSR    NEG
15610 DR1B 0C          CLC
15620 DR1C 39          RTS

```

```

15640 DR1D RD 02B2    CLRACC JSR    PUSHX
15650 DR1F 37          PSH B
15660 DR21 C6 07      LDA R  #7
15670 DR23 DE 5D      LDX    AFSTK
15680 DR25 6F 00      CLR    0.X
15690 DR27 08          INX
15700 DR29 5A          DFC B
15710 DR2B 26 FA      RNF    *-4
15720 DR2D 7F 004R    CLR    EXP
15730 DR2F 33          PUL B
15740 DR31 7E 02C7    JMP    PULLX

```

```

15760 DR33 A6 00      FORMEX LDA A  0.X
15770 DR35 BD 0432    JSR    TESTNO
15780 DR37 24 05      BCC    **7
15790 DR39 C6 02      INNER  LDA B  #2
15800 DR3B 7F 0828    JMP    FRROR
15810 DR3D 80 30      SJR A  #S30
15820 DR3F 16          TAR
15830 DR41 08          INX
15840 DR43 A6 00      LDA A  0.X
15850 DR45 BD 0432    JSR    TESTNO
15860 DR47 25 0D      BCS    **15
15870 DR49 58          ASL B
15880 DR4B 17          TRA
15890 DR4D 58          ASL B
15900 DR4F 58          ASL B
15910 DR51 1B          ARA
15920 DR53 16          TAR
15930 DR55 A6 00      LDA A  0.X
15940 DR57 C1 80 30    SJR A  #S30
15950 DR59 1B          ARA
15960 DR5B 16          TAR
15970 DR5D 08          INX
15980 DR5F 39          RTS

```

```

16000 OC17 DE 34      READ  LDX    RASPNT
16010 OC19 BD 03C1    READ1 JSR    TSTV
16020 OC1C 25 3F      RCS    READFX
16030 OC1E DF 34      STX    RASPNT
16040 OC20 DE 3C      LDX    DATPNT
16050 OC22 9C 2F      CPX    SOURCE
16060 OC24 27 1C      BFO    READ33
16070 OC26 BD 0B1D    READ2 JSR    INNUM
16080 OC29 24 22      BCC    READ4
16090 OC2B 09          DEX
16100 OC2C A6 00      LDA A  0.X
16110 OC2F 81 1F      CMP A  #S1F
16120 OC30 27 05      BFO    READ3
16130 OC32 C6 19      RDERR LDA B  #S19
16140 OC34 7F 0828    JMP    ERROR
16150 OC37 9C 2A      READ3 CPX    EXTRAR
16160 OC39 27 F7      BFO    RDERR
16170 OC3B A6 00      LDA A  0.X
16180 OC3D 08          INX
16190 OC3E 81 1F      CMP A  #S1F
16200 OC40 26 F5      BNF    READ3
16210 OC42 A6 02      READ33 LDA A  2.X
16220 OC44 81 79      CMP A  #S79
16230 OC46 26 EF      BNF    READ3
16240 OC48 08          INX
16250 OC49 08          INX
16260 OC4A 08          INX
16270 OC4B 20 D9      RRA    READ2
16280 OC4D BD 02E0    READ4 JSR    STORE
16290 OC4E 08          INX
16300 OC51 DF 3C      STX    DATPNT
16310 OC53 DF 34      LDX    RASPNT
16320 OC55 RD 0744    JSR    SKIPSP
16330 OC58 08          INX
16340 OC59 81 2C      CMP A  #'
16350 OC5B 27 RC      BFO    READ1
16360 OC5D 09          READEX DEX
16370 OC5E 81 1F      CMP A  #S1F
16380 OC60 26 D0      RNF    RDERR
16390 OC62 20 06      BRA    RESTO1

```

VECTOR OF DATA STATEMENT

```

16410 OC64 DE 2E      RESTOR LDX    SOURCE
16420 OC66 DF 3C      STX    DATPNT
16430 OC68 DE 34      LDX    RASPNT
16440 OC6A 7E 0DDF    RESTO1 JMP    REMARK

```

```

16460 OC6D DE 34      PRINT  LDX    RASPNT
16470 OC6F BD 0744    PRINTO JSR    SKIPSP
16480 OC72 81 22      CMP A  #'
16490 OC74 26 14      BNF    PRINT4
16500 OC76 08          INX
16510 OC77 A6 00      PRINT1 LDA A  0.X
16520 OC79 08          INX
16530 OC7A 81 22      CMP A  #'
16540 OC7C 27 4A      BFO    PRINRR
16550 OC7E 81 1F      CMP A  #S1F
16560 OC80 26 04      RNF    PRINT2
16570 OC82 C6 04      LDA R  #4
16580 OC84 20 1C      BRA    PRINTF
16590 OC86 8D 6E      PRINT2 BSR    DITCHA

```

```

16600 0C88 20 ED      BRA PRINT1
16610 0C8A 81 1F     PRINT4 CMP A #S1F
16620 0C8C 26 35      BNF PRINT8
16630 0C8E 09        DFX
16640 0C8F A6 00     LDA A 0,X
16650 0C91 08        INX
16660 0C92 81 3R     CMP A #'
16670 0C94 27 06     RFO PRINT5
16680 0C96 RD 02A0   JSR CRLF
16690 0C99 7F 0040   CLR PRCNT
16700 0C9C 08     PRINT5 INX
16710 0C9D DF 36     STX BASLIN
16720 0C9F 7F 0A00   JMP BASIC
16730 0CA2 7F 0828   PRINTF JMP ERROR
16740 0CA5 DF 34     CHR LDX BASPNT
16750 0CA7 BD 0467   JSR FIX
16760 0CAA 17     CHR2 TRA
16770 0CAB 8D 49     RSR OUTCHA
16780 0CAD 8F A045   PRINT6 LDS #SA045
16790 0CBB 20 16     RRA PRIN88
16800 0CB2 DF 34     TAB LDX BASPNT
16810 0CB4 RD 0467   JSR FIX
16820 0CB7 D0 40     SJR R PRCNT
16830 0CB9 23 F2     BLS PRINT6
16840 0CBB BD 0350   PRIN77 JSR PRINSP
16850 0CBE 5A        DEC R
16860 0CBF 26 FA     RNF PRIN77
16870 0CC1 20 FA     RRA PRIN76
16880 0CC3 BD 04E6   PRINT8 JSR EXPR
16890 0CC6 8D 4R     RSR PRN
16900 0CC8 BD 0744   PRIN88 JSR SKIPSP
16910 0CCB 81 2C     CMP A #'
16920 0CCD 26 0E     BNF PRIN99
16930 0CCF 08        INX
16940 0CD0 96 40     PRLOOP LDA A PRCNT
16950 0CD2 16        TAB
16960 0CD3 C4 F0     AND R #SFO
16970 0CD5 10        SBA
16980 0CD6 27 0A     BFO PRI999
16990 0CD8 BD 0350   JSR PRINSP

```

```

17010 0CDB 20 F3     BRA PRLOOP
17020 0CDD 81 3B     PRIN99 CMP A #'
17030 0CDF 26 04     RNE PRFND
17040 0CE1 08        TNX
17050 0CE2 7F 0C6F   PRI999 JMP PRINT0
17060 0CE5 81 1E     PRFND CMP A #S1F
17070 0CE7 27 A1     RFO PRINT4
17080 0CE9 C6 06     LDA B #6
17090 0CEB 20 B5     BRA PRINTF

```

```

17110 0CED 86 2F     OUTPNT LDA A #'
17120 0CEF 20 05     BRA OUTCHA
17130 0CF1 4F        OUTZER CLR A
17140 0CF2 84 0F     OUTDIG AND A #SOF
17150 0CF4 8B 30     ADD A #S30
17160 0CF6 36        OUTCHA PSH A
17170 0CF7 BD 0276   JSR OUTCH
17180 0CFA 32        PUL A

```

```

17200 0CFR 37     ENLINE PSH R
17210 0CFC D6 40     LDA R PRCNT
17220 0CFE 5C        INC R
17230 0CFF C1 30     CMP R #48
17240 0D01 23 0C     RLS ENLFXT
17250 0D03 C1 3F     CMP R #63
17260 0D05 27 04     BFO **6
17270 0D07 81 20     CMP A #S20
17280 0D09 26 04     RNF ENLFXT
17290 0D0B BD 02A0   JSR CRLF
17300 0D0E 5F        CLR R
17310 0D0F D7 40     ENLEXT STA B PRCNT
17320 0D11 33        PUL B
17330 0D12 39        RTS

```

```

17350 0D13 BD 0445   PRN JSR PSHXPA
17360 0D16 4F        CLR A
17370 0D17 97 4D     STA A DECFLG
17380 0D19 6D 00     TST 0,X
17390 0D1B 26 08     BNF PRN1
17400 0D1D 8D D2     RSR OUTZER
17410 0D1F BD 02C7   PRNO JSR PULLX
17420 0D22 7E 0350   JMP PRINSP
17430 0D25 6D 00     PRN1 TST 0,X
17440 0D27 2A 07     RPL PRN2
17450 0D29 86 2D     LDA A #'
17460 0D2B 8D C9     RSR OUTCHA
17470 0D2D BD 0528   JSR NEGACC
17480 0D30 A6 06     PRN2 LDA A 6,X
17490 0D32 81 09     CMP A #9
17500 0D34 2E 31     RGT PRISCI
17510 0D36 81 FF     CMP A #-1
17520 0D38 2D 2D     RLT PRISCI
17530 0D3A 4D        TST A
17540 0D3B 2E 10     RGT PRN4
17550 0D3D 8D R2     RSR OUTZER
17560 0D3F 8D AC     RSR OUTPNT
17570 0D41 97 4D     STA A DECFLG
17580 0D43 6D 06     PRN3 TST 6,X
17590 0D45 2A 1C     RPL PRN5
17600 0D47 8D A8     RSR OUTZER
17610 0D49 6C 06     INC 6,X
17620 0D4B 20 F6     BRA PRN3
17630 0D4D 7D 004D   PRN4 TST DECFLG
17640 0D50 26 11     RNF PRN5
17650 0D52 A6 00     PRN4 LDA A 0,X
17660 0D54 8D 9C     RSR OUTDIG
17670 0D56 BD 0552   JSR ALLFFT
17680 0D59 6A 06     DEC 6,X
17690 0D5B 26 F5     RNF PRN4
17700 0D5D 8D 45     RSR TSTZER
17710 0D5F 24 BF     BCC PRNO
17720 0D61 8D 8A     RSR OUTPNT
17730 0D63 8D 30     PRN5 RSR PRTAIL
17740 0D65 20 R8     RRA PRNO
17750 0D67 A6 00     PRISCI LDA A 0,X
17760 0D69 8D 87     RSR OUTDIG
17770 0D6B BD 0552   JSR ALLFFT
17780 0D6E BD 0CED   JSR OUTPNT
17790 0D71 8D 22     RSR PRTAIL
17800 0D73 86 45     LDA A #'
17810 0D75 RD 0CF6   JSR OUTCHA
17820 0D78 E6 06     LDA B 6,X
17830 0D7A 5A        DEC B

```

```

17840 0D7B 2A 06      BPL  PRISC2
17850 0D7D 86 2D      LDA  #'-
17860 0D7F BD 0CF6     JSR  OUTCHA
17870 0D82 50          NFG  B
17880 0D83 4F          PRISC2 CLR A
17890 0D84 4C          PRISC3 INC A
17900 0D85 00 0A      SUB  R #10
17910 0D87 2A FB      BPL  PRISC3
17920 0D89 4A          DEC  A
17930 0D8A CB 0A      ADD  R #10
17940 0D8C BD 0CF2     JSR  OUTDIG
17950 0D8F 17          TRA
17960 0D90 BD 0CF2     JSR  OUTDIG
17970 0D93 20 8A      BRA  PRNO

```

```

17990 0D95 8D 0D      PRTAIL BSR  TSTZER
18000 0D97 24 0A      BCC  PRTAFX
18010 0D99 A6 00      LDA  A 0,X
18020 0D9B BD 0CF2     JSR  OUTDIG
18030 0D9E BD 0552     JSR  ALLFFT
18040 0DA1 20 F2      BRA  PRTAIL
18050 0DA3 39          PRTAFX RTS

```

```

18070 0DA4 BD 02B2     TSTZER JSR  PUSHX
18080 0DA7 DE 5D          LDX  AESTK
18090 0DA9 37          PSH  B
18100 0DAA C6 06      LDA  R #6
18110 0DAC 6D 00      TSTZ1 TST 0,X
18120 0DAE 26 07      RNF  TSTZ3
18130 0DB0 08          INX
18140 0DB1 5A          DFC  B
18150 0DB2 26 F8      BNF  TSTZ1
18160 0DB4 0C          CLC
18170 0DB5 20 01      RRA  TSTZ3+1
18180 0DB7 0D          TSTZ3 SFC
18190 0DB8 33          PIUL B
18200 0DB9 7E 02C7     JMP  PULLX

```

```

18220 0DBC DF 34      LET  LDX  BASPNT
18230 0DBE BD 03C1     JSR  TSTV
18240 0DC1 24 05      RCC  **+7
18250 0DC3 C6 12      LET0  LDA  R #612
18260 0DC5 7F 0828     LET00 JMP  ERROR
18270 0DC8 BD 0744     JSR  SKIPSP
18280 0DCB 08          INX
18290 0DCC 81 3D      CMP  A #'=
18300 0DCE 27 04      RFQ  **+6
18310 0DD0 C6 06      LET2  LDA  B #6
18320 0DD2 20 F1      BRA  LET00
18330 0DD4 BD 04F6     JSR  EXPR
18340 0DD7 81 1F      CMP  A #61F
18350 0DD9 26 F5      RNF  *-9
18360 0DDB BD 02F0     JSR  STORF

```

```

18380 0DDE BD 077D     REMARK JSR  NXTLIN
18390 0DE1 7F 0A00     JMP  BASIC

```

```

18410 0DE4 BD 02B2     GETPAR JSR  PUSHX
18420 0DE7 DF 34          LDX  BASPNT
18430 0DE9 BD 04F6     JSR  EXPR
18440 0DEC 08          INX
18450 0DED 81 2C      CMP  A #'
18460 0DEF 27 0B      RFQ  GETPA2
18470 0DF1 81 29      CMP  A #'
18480 0DF3 26 1A      RNF  DIMFRR
18490 0DF5 DF 34          STX  RASPNT
18500 0DF7 BD 02C7     JSR  PULLX
18510 0DFA 0C          CLC
18520 0DFB 39          RTS
18530 0DFC BD 04B1     GETPA2 JSR  PARFXP
18540 0DFF DF 34          STX  RASPNT
18550 0E01 BD 02C7     JSR  PULLX
18560 0E04 0D          SFC
18570 0E05 39          RTS

```

```

18590 0E06 DF 34      DIM  LDX  RASPNT
18600 0E08 BD 0358     JSR  FINVAR
18610 0E0B 29 02      RVS  DIMFRR
18620 0E0D 25 05      BCS  DIM1
18630 0E0F C6 05      DIMFRR LDA  B #5
18640 0E11 7F 0828     JMP  ERROR
18650 0E14 81 28      DIM1  CMP  A #'
18660 0E16 26 F7      RNF  DIMFRR
18670 0E18 DF 34          STX  BASPNT
18680 0E1A DE 63          LDX  VARPNT
18690 0E1C F7 00      STA  R 0,X
18700 0E1E A7 01      STA  A 1,X
18710 0E20 8D C2      BSR  GETPAR
18720 0E22 24 05      BCC  DIM3
18730 0E24 BD 0467     JSR  FTX
18740 0E27 20 01      RRA  DIM4
18750 0E29 5F          DIM3 CLR  R
18760 0E2A F7 03      DIM4 STA  B 3,X
18770 0E2C BD 0467     JSR  FTX
18780 0E2F 17          TRA
18790 0E30 A7 02      STA  A 2,X
18800 0E32 8D 0D      RSR  DIMCAL
18810 0E34 DF 34          LDX  RASPNT
18820 0E36 BD 0744     JSR  SKIPSP
18830 0E39 08          INX
18840 0E3A 81 2C      CMP  A #'
18850 0E3C 27 CA      RFQ  DIM+2
18860 0E3E 09          DFX
18870 0E3F 20 9D      BRA  REMARK

```

```

18890 0F41 BD 02B2     DIMCAL JSR  PUSHX
18900 0F44 F6 03      LDA  B 3,X
18910 0F46 8D 11      RSR  SJRCAL
18920 0F48 BD 0459     JSR  INXSIX
18930 0F4B DF 46          STX  ARRTAR
18940 0F4D BD 02C7     JSR  PULLX
18950 0F50 96 46      LDA  A ARRTAR
18960 0F52 A7 04      STA  A 4,X
18970 0F54 96 47      LDA  A ARRTAR+1
18980 0F56 A7 05      STA  A 5,X
18990 0F58 39          RTS

```

19010	0E59	4D	SURCAL	TST	A		
19020	0E5A	26 05		RNF		SURC1	
19030	0E5C	C6 15	SURFRR	LDA	B	#S15	
19040	0E5E	7F 0828		JMP		ERROR	
19050	0E61	A1 02	SURC1	CMP	A	2.X	
19060	0E63	22 F7		BHI		SURFRR	
19070	0E65	E1 03		CMP	R	3.X	
19080	0E67	22 F3		RHI		SURFRR	
19090	0E69	36		PSH	A		
19100	0E6A	A6 02		LDA	A	2.X	
19110	0E6C	97 51		STA	A	ANUMP	
19120	0E6E	5D		TST	B		
19130	0E6F	27 0D		RFO		SURC6	
19140	0E71	5A	SUBC4	DFC	R		
19150	0E72	27 0A		RFO		SURC6	
19160	0E74	96 51		LDA	A	ANUMP	
19170	0E76	RD 0459	SURC5	JSR		INXSIX	
19180	0E79	4A		DFC	A		
19190	0E7A	26 FA		RNF		SURC5	
19200	0E7C	20 F3		RRA		SURC4	
19210	0E7E	32	SUBC6	PUL	A		
19220	0E7F	RD 0459	SURC7	JSR		INXSIX	
19230	0E82	4A		DFC	A		
19240	0E83	26 FA		RNF		SURC7	
19250	0E85	39		RTS			
19270	0E86	DE 34	FOR	LDX		BASPNT	
19280	0E88	BD 03C1		JSR		TSTV	
19290	0E8B	24 03		BCC		F0R1	
19300	0E8D	7F 0DC3		JMP		LFT0	ERROR JUMP
19310	0E90	DF 34	FOR1	STX		RASPNT	
19320	0E92	DF 5D		LDX		AESTK	
19330	0E94	RD 044F		JSR		DEXSFV	
19340	0E97	A6 00		LDA	A	0.X	
19350	0E99	F6 01		LDA	B	1.X	
19360	0E9B	97 51		STA	A	ANUMB	
19370	0E9D	D7 52		STA	R	BNUMB	
19380	0E9F	CF 1180		LDX		#FORSTK	
19390	0EA2	96 51	FOR10	LDA	A	ANUMB	
19400	0EA4	D6 52		LDA	R	BNUMB	
19410	0EA6	9C 5F		CPX		F0RPNT	
19420	0EA8	27 1C		RFO		F0R14	
19430	0EAA	A1 00		CMP	A	0.X	
19440	0EAC	26 10		RNF		F0R12	
19450	0EAE	F1 01		CMP	R	1.X	
19460	0EBO	26 0C		RNF		F0R12	
19470	0EB2	96 36		LDA	A	RASLIN	
19480	0EB4	D6 37		LDA	R	RASLIN+1	
19490	0EB6	A1 0F		CMP	A	14.X	
19500	0EB8	26 04		RNF		F0R12	
19510	0EBA	F1 0F		CMP	B	15.X	
19520	0EBC	27 08		RFO		F0R14	
19530	0EBE	RD 0457	FOR12	JSR		INXATF	
19540	0EC1	BD 0457		JSR		INXATF	
19550	0EC4	20 DC		RRA		F0R10	
19560	0EC6	8C 1200	FOR14	CPX		#FORSTK+128	
19570	0EC9	26 05		RNF		F0R11	
19580	0ECB	C6 10		LDA	B	#16	
19590	0ECD	7F 0828		JMP		ERROR	
19600	0ED0	96 51	FOR11	LDA	A	ANUMP	
19610	0ED2	D6 52		LDA	R	BNIJMP	
19620	0ED4	A7 00		STA	A	0.X	
19630	0ED6	08		INX			

19640	0ED7	E7 00		STA	R	0.X	
19650	0ED9	08		INX			
19660	0EDA	DF 5F		STX		F0RPNT	
19670	0EDC	DE 34		LDX		RASPNT	
19680	0EDE	BD 0744		JSR		SKIPSP	
19690	0EE1	08		INX			
19700	0EE2	81 3D		CMP	A	#/=	
19710	0EE4	27 03		RFO		F0R3	
19720	0EE6	7F 0DD0	FOR2	JMP		LFT2	ERROR JUMP
19730	0EE9	RD 04E6	FOR3	JSR		FXPR	
19740	0EEC	RD 02F0		JSR		STORF	
19750	0EEF	RD 0792		JSR		CCINT	
19760	0EF2	8C 0109		CPX		#TOPNT	
19770	0EF5	26 FF		RNF		F0R2	
19780	0EF7	DE 34		LDX		BASPNT	
19790	0EF9	BD 04F6		JSR		FXPR	
19800	0EFC	DF 34		STX		RASPNT	
19810	0EFE	DE 5F		LDX		F0RPNT	
19820	0F00	BD 02D2		JSR		STORFX	
19830	0F03	RD 0450		JSR		INXSIX	
19840	0F06	DF 5F		STX		F0RPNT	
19850	0F08	DE 34		LDX		RASPNT	
19860	0F0A	BD 0792		JSR		CCINT	
19870	0F0D	8C 0110		CPX		#STFPNT	
19880	0F10	26 09		RNF		F0R4	
19890	0F12	DF 34		LDX		RASPNT	
19900	0F14	BD 04F6		JSR		FXPR	
19910	0F17	DF 34		STX		RASPNT	
19920	0F19	20 0F		RRA		F0R5	
19930	0F1B	DF 5D	FOR4	LDX		AESTK	
19940	0F1D	RD 0BDD		JSR		CLRACC	
19950	0F20	86 01		LDA	A	#S01	
19960	0F22	A7 00		STA	A	0.X	
19970	0F24	A7 06		STA	A	6.X	
19980	0F26	RD 043F		JSR		PUSHAF	
19990	0F29	DF 5F	FOR5	LDX		F0RPNT	
20000	0F2B	RD 02D2		JSR		STORFX	
20010	0F2E	RD 0459		JSR		INXSIX	
20020	0F31	DF 5F		STX		F0RPNT	
20030	0F33	DF 34	FOR7	LDX		RASPNT	
20040	0F35	RD 0744		JSR		SKIPSP	
20050	0F38	81 1F		CMP	A	#S1F	
20060	0F3A	26 AA	FOR8	RNF		F0R2	ERROR JUMP
20070	0F3C	08		INX			
20080	0F3D	DF 36		STX		RASLIN	
20090	0F3F	DF 5F		LDX		F0RPNT	
20100	0F41	D6 36		LDA	B	RASLIN	
20110	0F43	F7 00		STA	B	0.X	
20120	0F45	08		INX			
20130	0F46	D6 37		LDA	R	RASLIN+1	
20140	0F48	E7 00		STA	R	0.X	
20150	0F4A	08		INX			
20160	0F4B	DF 5F		STX		F0RPNT	
20170	0F4D	7E 0A00		JMP		BASIC	
20190	0F50	DE 34	NEXT	LDX		RASPNT	
20200	0F52	BD 03C1		JSR		TSTV	
20210	0F55	24 03		BCC		NEXTI	
20220	0F57	7E 0DC3		JMP		LFT0	ERROR JUMP
20230	0F5A	BD 0744	NEXTI	JSR		SKIPSP	
20240	0F5D	81 1F		CMP	A	#S1F	
20250	0F5F	26 D9		RNF		F0R8	ERROR JUMP
20260	0F61	08		INX			
20270	0F62	DF 36		STX		RASLIN	

```

20280 0F64 RD 0448 JSR PULLAF
20290 0F67 A6 00 LDA A 0,X
20300 0F69 E6 01 LDA R 1,X
20310 0F6B DE 5F LDX FORPNT
20320 0F6D 8C 1180 CPX #FORSTK
20330 0F70 27 66 RFO NEXT6
20340 0F72 DF 5F NEXT2 STX FORPNT
20350 0F74 8C 1180 CPX #FORSTK
20360 0F77 27 5B RFO NEXT5
20370 0F79 BD 044F JSR DEXATF
20380 0F7C RD 044F JSR DEXATF
20390 0F7F A1 00 CMP A 0,X
20400 0F81 26 FF RNF NEXT2
20410 0F83 E1 01 CMP R 1,X
20420 0F85 26 EB RNF NEXT2
20430 0F87 DF 61 STX FORNOW
20440 0F89 EE 00 LDX 0,X
20450 0F8B RD 0302 JSR INDX
20460 0F8E DE 61 LDX FORNOW
20470 0F90 RD 0457 JSR INXATF
20480 0F93 86 02 LDA A #2
20490 0F95 6D 00 TST 0,X
20500 0F97 2A 02 BPL **4
20510 0F99 8B 03 ADD A #3
20520 0F9B 97 4F STA A NCMPR
20530 0F9D RD 0302 JSR INDX
20540 0FA0 RD 05FC JSR ADD
20550 0FA3 CF 006F LDX #FORTMP
20560 0FA6 RD 02D2 JSR STORFX
20570 0FA9 RD 043F JSR PUSHA
20580 0FAC DF 61 LDX FORNOW
20590 0FAE 08 INX
20600 0FAF 08 INX
20610 0FB0 BD 0302 JSR INDX
20620 0FB3 RD 1037 JSR CMPR
20630 0FB6 24 07 RCC NEXT4
20640 0FB8 DE 61 LDX FORNOW
20650 0FBA DF 5F STX FORPNT
20660 0FBC 7F 0A00 NEXT3 JMP BASIC
20670 0FBF CF 006F NEXT4 LDX #FORTMP
20680 0FC2 RD 0302 JSR INDX
20690 0FC5 DE 61 LDX FORNOW
20700 0FC7 FE 00 LDX 0,X
20710 0FC9 BD 02D2 JSR STORFX
20720 0FCC DE 61 LDX FORNOW
20730 0FCE EF 0E LDX 14,X
20740 0FD0 DF 36 STX BASLIN
20750 0FD2 20 F8 BRA NEXT3
20760 0FD4 C6 18 NEXT5 LDA B #S18
20770 0FD6 20 02 BRA NEXT6+2
20780 0FD8 C6 17 NEXT6 LDA R #S17
20790 0FDA 7F 0B28 JMP ERROR

20810 0FDD DE 34 IF LDX RASPNT
20820 0FDF BD 04E6 JSR FXPR
20830 0FE2 8D 17 RSR RELOP
20840 0FE4 97 4F STA A NCMPR
20850 0FE6 RD 04E6 JSR FXPR
20860 0FE9 DF 34 STX BASPNT
20870 0FEB 8D 4A RSR CMPR
20880 0FED 24 03 BCC IF2
20890 0FFF 7E 0DDF JMP REMARK
20900 0FF2 DF 34 IF2 LDX RASPNT
20910 0FF4 BD 079B JSR CODDE
20920 0FF7 FE 00 LDX 0,X
20930 0FF9 6F 00 JMP 0,X
    
```

```

20950 0FFB BD 0744 RELOP JSR SKIPSP
20960 0FFE 08 INX
20970 0FFF 81 3D CMP A #'=
20980 1001 26 03 RNF RELOP0
20990 1003 86 00 LDA A #? =
21000 1005 39 RTS
21010 1006 F6 00 RELOP0 LDA R 0,X
21020 1008 81 3C CMP A #'<
21030 100A 26 13 RNF RELOP4
21040 100C C1 3D CMP R #'=
21050 100E 26 04 RNF RELOP1
21060 1010 08 INX
21070 1011 86 02 LDA A #2 <=
21080 1013 39 RTS
21090 1014 C1 3F RELOP1 CMP R #'>
21100 1016 26 04 BNF RELOP3
21110 1018 08 RELOP2 INX
21120 1019 86 03 LDA A #3 <>
21130 101B 39 RTS
21140 101C 86 01 RELOP3 LDA A #1 <
21150 101E 39 RTS
21160 101F 81 3E RELOP4 CMP A #'>
21170 1021 27 05 BFO REL44
21180 1023 C6 06 LDA B #6
21190 1025 7F 0B28 JMP ERROR
21200 1028 C1 3D REL44 CMP B #'=
21210 102A 26 04 RNF RELOP5
21220 102C 08 INX
21230 102D 86 05 LDA A #5 >=
21240 102F 39 RTS
21250 1030 C1 3C RELOP5 CMP B #'<
21260 1032 27 F4 BFO RELOP2
21270 1034 86 04 LDA A #4 >
21280 1036 39 RTS

21300 1037 96 4F CMPR LDA A NCMPR
21310 1039 48 ASL A
21320 103A 48 ASL A
21330 103B CE 1051 LDX #CMPRI
21340 103E DF 20 STX INDFX1
21350 1040 9B 21 ADD A INDFX1+1
21360 1042 97 21 STA A INDFX1+1
21370 1044 BD 05F0 JSR SUB
21380 1047 RD 0448 JSR PULLAF
21390 104A A6 00 LDA A 0,X
21400 104C DF 20 LDX INDFX1
21410 104E 4D TST A
21420 104F 6F 00 JMP 0,X
21430 * *****FOLLOWING MUST BE ON ONE PAGE*****
21440 1051 27 16 CMPRI RFO OKCMPR
21450 1053 20 12 BRA NOCMPR
21460 1055 2B 12 RMI OKCMPR
21470 1057 20 0E BRA NOCMPR
21480 1059 2B 0F RMI OKCMPR
21490 105B 20 F4 BRA CMPRI
21500 105D 26 0A RNF OKCMPR
21510 105F 20 06 BRA NOCMPR
21520 1061 27 04 BFO NOCMPR
21530 1063 2B 02 RMI NOCMPR
21540 1065 2A 02 RPL OKCMPR
21550 1067 0D NOCMPR SFC
21560 1068 39 RTS
21570 1069 0C OKCMPR CLC
21580 106A 39 RTS
    
```

21600	106B	BD	0448	JSFR	JSR	PULLAF
21610	106F	BD	02R2		JSR	PIISHX
21620	1071	DF	67		LDX	DIMMY
21630	1073	AD	00		JSR	O,X
21640	1075	RD	02C7		JSR	PULLX

21660	1078	39		DUMRTS	RTS	
-------	------	----	--	--------	-----	--

21680	1079	RD	0448	INT	JSR	PULLAF
21690	107C	C6	09	INT1	LDA B	#0
21700	107E	FO	06		SUB B	6,X
21710	1080	5D		INT2	TST R	
21720	1081	2F	08		BLF	INT4
21730	1083	BD	0578		JSR	LRIGHT
21740	1086	6C	06		INC	6,X
21750	1088	5A			DFC R	
21760	1089	20	F5		RRA	INT2
21770	108B	6F	05	INT4	CLR	5,X
21780	108D	BD	059F		JSR	NORMPA
21790	1090	7E	0448		JMP	PULLAF

21810	1093	BD	0448	SGN	JSR	PULLAF
21820	1096	A6	00		LDA A	O,X
21830	1098	27	DF		BFO	DUMRTS
21840	109A	97	4A		STA A	TSIGN
21850	109C	BD	0BDD		JSR	CLRACC
21860	109F	86	01		LDA A	#1
21870	10A1	A7	00		STA A	O,X
21880	10A3	A7	06		STA A	6,X
21890	10A5	7D	004A	SGN3	TST	TSIGN
21900	10A8	2A	CF		BPL	DUMRTS
21910	10AA	7F	0528	SGN33	JMP	NFGACC
21920	10AD	39		SGN4	RTS	

21940	10AE	BD	0448	ARS	JSR	PULLAF
21950	10B1	6D	00		TST	O,X
21960	10B3	2A	C3		RPL	DUMRTS
21970	10B5	20	F3		BRA	SGN33

21990	10B7	BD	0448	RANDOM	JSR	PULLAF
22000	10BA	RD	0DA4		JSR	TSTZFR
22010	10BD	24	05		BCC	RAND1
22020	10BF	RD	043F		JSR	PIISHAF
22030	10C2	20	08		BRA	RAND1+8
22040	10C4	CE	0069	RAND1	LDX	#RANNUM
22050	10C7	6C	00		INC	O,X
22060	10C9	RD	0302		JSR	INDX
22070	10CC	CF	10FF		LDX	#RANMUL
22080	10CF	BD	0302		JSR	INDX
22090	10D2	BD	06D1		JSR	MPY
22100	10D5	RD	0448		JSR	PULLAF
22110	10D8	RD	0552		JSR	ALLFFT
22120	10DB	E6	01		LDA R	1,X
22130	10DD	A6	03		LDA A	3,X
22140	10DF	A7	01		STA A	1,X

22150	10E1	E7	03		STA R	3,X
22160	10F3	6F	06		CLR	6,X
22170	10E5	RD	059F		JSR	NORMPA
22180	10E8	CF	0069		LDX	#RANNUM
22190	10EB	7F	02D2		JMP	STORFX
22200	10EE	08		RANMUL	FCB	\$08,\$37,\$25,\$41,\$69,\$00
	10EF	37				
	10F0	25				
	10F1	41				
	10F2	69				
	10F3	00				
22210		10F4		ROBERT	FOU	*

22230	1102				ORG	\$1102
22240	1102	007F		ASTACK	RMR	19*7
22250	1180	0080		FORSTK	RMB	128
22260	1200				ORG	\$1200
22270		1200		FILF	FOU	*
22280	A048				ORG	\$A048
22290	A048	0100			FDR	PROGM
22300					END	

SYMBOL TABLE

INDEX1	0020	INDEX2	0022	INDEX3	0024	INDEX4	0026	SAVEFP	0028
NEXTBA	002A	WORKBA	002C	SOURCE	002E	PACKLN	0030	HIGHLN	0032
RASPNT	0034	RASLIN	0036	PIUSHX	0038	XSTACK	003A	DATPNT	003C
DIMPNT	003E	PRCNT	0040	MEMEND	0044	ARRTAB	0046	KFYWD	0048
TSIGN	004A	FXP	004B	STGDIG	004C	DFCFLG	004D	TGGFLF	004E
NCMPR	004F	TNUMP	0050	ANUMR	0051	RNUMR	0052	LRNINF	0053
ROUNDV	0054	DNDOP	0055	NORSGN	0056	XLONG	0057	MPYXX	0059
BASNUM	005B	AFSTK	005D	FORPNT	005F	FORNDW	0061	VARPNT	0063
SBRPNT	0065	DUMMY	0067	RANNUM	0069	FORTMP	006F	SRRSTK	0075
LDNFLG	0085	BUFNXT	00AC	FNDBUF	00AF	BUFFER	00B0	PRJGM	0100
INTER	0106	TOPNT	0109	STFPNT	0110	COMMAN	0112	GRLIST	0137
STOMSG	01B3	COMEND	01BA	IMPLFT	01BB	FUNCT	01BD	FUNEND	01FD
RDYMSG	01F0	DFLMSG	01FB	PGCNTL	0200	WHAT	0203	FRMSI	020C
ERRMS2	0214	DEL	021E	KFYRD	0225	KFYRD0	022C	KFYRD1	022F
KEYBD2	0258	CNTLIN	025D	FXFIT	0263	OUTPH	0270	OUT4HS	0273
OUTCH	0276	INCH	027R	BRFAK	0281	BRFAK0	028A	BRFAK1	0290
OUTPUT	0292	OUTNCR	0299	CRLF	02A0	CRLFST	02AA	CRFND	02B1
PUSHX	02B2	PULLX	02C7	STORFX	02D2	STORF	02F0	INDX	0302
IND	0309	IND0	0313	OUTLIN	032A	PRINSP	0350	FINVAR	0358
FINV1	0360	FINV11	0386	FINV2	0388	FINV3	039A	FINV4	039B
FINV5	03A9	FINV6	03B0	FINV7	03BD	TSTV	03C1	TSTV1	03C0
TSTV11	03D4	TSTV2	03D6	TSTV3	03F9	TSTV31	03FE	TSTV33	03FE
TSTV34	03F3	TSTV4	03FD	TSTV44	0417	TSTV5	0422	TSTLTR	042A
TESTND	0432	NONO	043A	YESNO	043C	PIUSHAF	043F	PSHXP	0445
PULLAE	0448	DFXATE	044F	DFXSEV	044F	DFXSIX	0450	DFXFIV	0451
DEXFOR	0452	INXATE	0457	INXSEV	0458	INXSIX	0459	INXFIV	045A
INXFDR	045B	OVRTST	045F	FIX	0467	FIXERR	0470	FIX2	0477
FIXEX	0494	FACT	049A	PARFXP	04R1	FACT2	04RC	FACT3	04C1
TERM	04C5	TFRMO	04C7	TFRM1	04D9	TFRM2	04F5	EXPR	04F6
EXPR1	04FR	NFGDP	0517	NFG	051F	NFGACC	0528	NFGA11	0532
NEGA2	0542	ALLEFT	0552	LLFFT	055D	LLFFT2	0563	LLFFT3	056C
LRIGHT	0578	LRIGHT2	0584	LRIGHT3	058A	NORMPA	059F	NORM	05A1
NORM00	05B5	NORM1	05BB	NORMF2	05C5	NORMFX	05C7	NORM01	05D4
NORMF0	05E4	NORMF1	05EB	NORMF2	05F3	SUR	05F9	AND	05FC
ADDO	0611	ADD1	0622	ADD2	0624	ADD4	0629	SWAP	062C
SWAP1	0632	SWAP2	0640	ADDFR	0645	ADDFR1	064C	DIV	0659
DIV2	0678	DIV3	0679	MDSIGN	06A8	MDS2	06R5	MDS3	06C1
MDS4	06CF	MPY	06D1	MPY4	06F3	MPY5	06FA	MPY6	06F3
MDEXIT	06FF	MDFX2	0713	FINND0	0716	FINND1	0720	FINND	0722
HIBALL	073E	SKIPSP	0744	INXSKP	074A	LINEND	074F	LIN0	0753
NXTLIN	077D	CCDF	0789	CCINT	0792	CCDFE	079B	LINP3	07A4
LOOP5	07R6	CCFXIT	07D3	FUNCFX	07D9	NEWSUR	07DC	START	07FE
READY	07F6	NEWLIN	0807	FRDR	0828	FRDR2	0853	RIN	085R
CLIST	0882	CLIST3	08A8	CLIST4	08AC	CLIST5	08C2	CLFXT	08D0
PATCH	08D3	SAVE	08F5	SAVF1	08F2	SAVF5	0903	LIN0	090D
APPEND	0910	LOAD1	0917	DELAY2	0933	DELAY	0935	DFLAY3	0937
LEADER	093E	NUMRFR	0949	NIMI	094C	DFLRF	0964	RFPLAC	0977

```

NEXT 097R  CAPPEN 097C  DELFTF 0984  DEL2  098F  DEL4  099D
DELEX 09A7  INSERT 09AA  INS2  09D2  RUFWR7 09DC  QVFRFL 09FR
RASIC 0A00  THEN  0A20  DNFR  0A36  DNCG  0A3R  GNSHR  0A5A
GOSUB1 0A6F  GOTO  0A7C  GOTO1  0A81  GOTO2  0A9D  RETIURN 0AA2
STOP  0AB6  INPUT 0AC4  INPERR 0ACA  INPUT1 0AD7  INPIIT2 0AF0
INPUT4 0AF6  INPUT 0B01  INPEX  0B0D  DRLLTR 0B1R  INNUM  0B1D
TSTN  0R28  INNUM0 0R2R  INTST1 0R3F  INTST2 0R53  INTST3 0R6F
INTST4 0R70  INTST5 0R85  INTS55 0R8R  INTST6 0R9F  INTS66 0R9R
MAYEXP 0R9D  MAYEX2 0RAA  INTST8 0RRR  INTST9 0RRF  INNEFX 0RC3
CLRACC 0BDD  FORMFX 0BF2  INNER 0BF0  READ  0C17  READ1 0C10
READ2 0C26  RDRFR  0C32  READ3 0C37  READ33 0C42  READ4 0C4D
READEX 0C5D  RSTOR  0C64  RSTO1  0C6A  PRINT  0C6D  PRINT0 0C6F
PRINT1 0C77  PRINT2 0CR6  PRINT4 0CRA  PRINT5 0C9C  PRINTF 0CA2
CHR  0CA5  CHR2  0CAA  PRINT6 0CAD  TAB  0CR2  PRINT7 0CRR
PRINT8 0CC3  PRIN88 0CC8  PLOOP 0CDD  PRIN99 0CDD  PRI999 0CF2
PREND 0CE5  OUTPNT 0CFD  OUTZFR 0CF1  OUTDIG 0CF2  OUTCHA 0CF6
ENLINE 0CFR  ENLEXT 0D0F  PRN  0D13  PRN0  0D1F  PRN1  0D25
PRN2  0D30  PRN3  0D43  PRN4  0D4D  PRN44 0D52  PRN5  0D63
PRISCI 0D67  PRISC2 0DR3  PRISC3 0DR4  PRTAIL 0DR5  PRTAFX 0DA3
TSTZER 0DA4  TSTZ1  0DAC  TSTZ3 0DR7  LFT  0DR0  LFT0  0DC3
LETOO 0DC5  LET2  0DD0  REMARK 0DDF  GETPAR 0DE4  GETPA2 0DF0
DIM  0F06  DIMERR 0E0F  DIM1  0F14  DIM3  0F20  DIM4  0F2A
DIMCAL 0E41  SURCAL 0E59  SURFRR 0E5C  SURC1  0E61  SURC4  0E71
SURC5 0E76  SURC6 0E7F  SUPC7 0E7F  FOR  0E86  FOR1  0E90
FOR10 0EA2  FOR12 0E8F  FOR14 0E96  FOR11 0E9D  FOR2 0E9F
FOR3 0EF9  FOR4  0F1B  FOR5  0F20  FOR7  0F33  FOR8  0F3A
NEXT  0F50  NEXT1 0F5A  NEXT2 0F72  NEXT3 0FR0  NEXT4 0FRF
NEXT5 0F04  NEXT6 0F08  IF  0FDD  IF2  0FF2  QFLOP 0FFR
RELOPO 1006  RFLDP1 1014  RFLDP2 1018  RFLDP3 101C  RFLDP4 101F
RFL44 1028  RFLDP5 1030  C4PR  1037  C4PR1 1051  NDCMR 1067
DKCMR 1069  USFR  106R  DUMRTS 107R  INT  1070  INT1 107C
INT2 1080  INT4  108R  SGN  1093  SGN3 10A5  SGN33 10AA
SGN4 10AD  ARS  10AF  RANDOM 10B7  RAND1 10C4  RANMUL 10FF
ROBERT 10F4  ASTACK 1102  FORSTK 1180  FILE  1200

```

READY

UNDERSTANDING UITERWYK'S BASIC

Introduction

The following Table 1, SUMMARY OF BASIC FEATURES, explains the characteristics of the 6800 4K BASIC Interpreter program. For convenience Robert Uiterwyk's 4K BASIC also carries the mnemonic name RU4KBASIC.

BASIC Program Format

A BASIC program is comprised of programming statements. These statements tell the computer in a step-by-step sequence how to perform a particular task. The computer has no intelligence of its own, and only understands what it has been programmed to do. The BASIC language has some simple but exacting rules on how to instruct a computer how to do a particular task. For instance, the format of a BASIC programming statement is always a statement number, followed by the statement body, and terminated by a carriage return. The statement body always starts with a keyword that identifies the type of statement.

Statements need not be entered in numerical order, because the BASIC interpreter will automatically sort the statements in ascending order by line numbers. The statement number is also used for reference purposes, as will be seen when discussing the GOTO, GOSUB and IF statements. All line numbers must be between 1 and 9999, and zero may not be used. A programming statement may con-

Table 1. Summary of BASIC Features

COMMANDS	STATEMENTS	FUNCTIONS	
LIST	REM	END	ABS
RUN	DIM	GOTO*	INT
NEW	DATA	ON...GOTO*	RND
SAVE	READ	ON...GOSUB*	SGN
LOAD	RESTORE	IF...THEN*	CHR
PATCH	LET*	INPUT	USER
	FOR	PRINT*	TAB
	NEXT	PATCH*	
	STOP	RETURN	
	GOSUB*		

*Flags statements that may be used in the direct mode (No statement numbers)

MATH OPERATORS	RELATIONAL OPERATORS
- Unary negation	= Equal
* Multiplication	<> NOT equal
/ Division	< Less than
+ Addition	> Greater than
- Subtraction	<= Less than or Equal
	>= Greater than or Equal

Line Numbers may be from 0001 to 9999

Variables	Simple variables	Single alphabetic or Single alphabetic and a single digit Single alphabetic

Arrays:

Backspace	Control-O
Line delete	Control-X
Panic Button	Control-C should bring Basic back to the READY mode regardless of what the BASIC user program is doing.

tain no more than 72 characters including blanks. Unless within a character string and enclosed by quotation marks ("), blanks are *not* processed by BASIC, and their use is optional. With blanks, the statement is more readable, as can be seen in Example 1, but will require a slightly longer time to process. In addition typing blanks in the statements will require more memory for the storage of the program. Keep in mind, though, that program readability is very important; for if you cannot read it, you cannot modify or fix it! The only place in BASIC that blanks may not be used at all, is inside the various keywords and inside numbers. "12 34" is *NOT* the same as "1234".

In any one program, a line number may be used only once. A previously entered line may be changed by entering the same line

Programming Statement	Memory required
110 LET A = B + (3.5*5E2)	23 Bytes
110LETA = B + (3.5*5E2)	16 Bytes

Example 1. Statements with and without imbedded blanks.

number along with the desired new statement. Typing just a line number followed by a carriage return will delete that line.

It is strongly suggested that you use line numbers which are at least ten numbers apart. This will make it easier to add in statements between the original statements in case of omission, or if you desire to add additional features to an existing program. If you do wish to insert a statement between two others, you need only type a line number that falls between the other two. For example consider the following original BASIC program:

```
180 LET P = A*A*3.14
185 PRINT
```

in which it is desired to insert the following addition program statement

```
183 REM INSERT THIS LINE
```

with the resulting program becoming:

```
180 LET P = A*A*3.14
183 REM INSERT THIS LINE
185 PRINT
```

If it is desired to replace a statement, a new statement is typed that has the same line number as the one to be replaced. For example, typing the statement:

```
180P = (A*A)*3.14
```

would cause the example program to become:

```
180P = (A*A)*3.14
183 REM INSERT THIS LINE
185 PRINT
```

Each line is terminated by a carriage return. Only after the carriage return is typed will BASIC store the new statement in the proper sequence in its memory. If after typing a line and before typing the carriage return, you should change your mind, simply enter a "control-X" (Control key depressed and X key struck) to delete the line. Single character typing errors can be corrected by typing a "control-O" for a backspace. The computer will print either a left-arrow (←) or an underline (⎵) for each "control-O" pressed, and will backspace across the characters stored in the input buffer. The character displayed in response to a "control-O" will vary depending on the terminal used. If it is desired to stop a running program, or to terminate a LIST command, you should use the "control-C" character.

Variable Names And Data Formats

Now that you know how to enter a program, correct any mistyped lines, and how to stop a running program, we should probably find out how the "numbers" that control the program flow are referenced.

All numbers which can be changed during the execution of a program are called *variables*. Those numbers whose values cannot be changed other than by retyping a programming statement are called *constants*. Variables are referenced by a unique name given them by the programmer. A simple variable may be given a name of a single alphabetic or a single alphabetic character followed by a single digit 0 thru 9. Thus legal names for a simple variable are: A, B, Z0, Q5. Illegal names are 9, 5A, A34, AB. BASIC also has a type of variable known as an array. An array is defined as an ordered collection of numeric data known to BASIC under a single name. Any array may be given a name of a single alphabetic character, A to Z. BASIC allows the use of both the simple variable A and the array A in the same program.

Arrays are divided into columns (vertical) and rows (horizontal). Arrays may have one or two dimensions. For example:

```
1.01
2.11
3.22
4.34
```

is a one dimensional array while

```
6 5 4
3 2 1
0 9 8
```

is a two dimensional array.

Array elements are referenced by their column and row position. For instance, if the examples above were arrays A and Z respectively, 2.11 would be A(2); similarly, 0 would be Z(3,1). The references to array elements are called subscripts, and set apart with parentheses. For example P(1,5) references the fifth element of the first row of array P; 1 and 5 are the subscripts. In X(M,N), M and N are the subscripts.

The range of numbers that can be stored in a single element of an array, or in a simple variable is 1.0×10^{-99} to $9.99999999 \times 10^{99}$. There are nine digits of significance in this version of BASIC. There will be no radix or base conversion errors in this version of BASIC, because all numbers are stored internally in an exponential (floating point) base 10 format. All numbers are internally truncated to nine digits to fit this precision. Numbers may be entered and displayed in three formats: INTEGER, DECIMAL, and EXPONENTIAL. BASIC will automatically select the appropriate output format when executing a "PRINT" statement. Figure 3 identifies the three formats and gives examples.

INTEGER (NO DECIMAL POINT)

1 998 10 99999999

DECIMAL (FLOATING POINT NUMBERS)

.123456789 3.14 1.2E-99

EXPONENTIAL (SCIENTIFIC NOTATION)

10E99 -1.2345E76 1.2E-99

E99 REPRESENTS 10^{99} AND E-99 IS 10^{-99}
(E STANDS FOR EXPONENT)**Figure 3. Data Formats****Commands**

Although most BASIC statements will have line numbers, it is possible to communicate in BASIC by typing certain types of statements without line numbers. This type of statements is called *commands*. A command is a BASIC statement that causes immediate action. In contrast, however, a typical BASIC program is entered into the computer by typing it a line at a time; later typing a command (RUN) to cause the computer to begin executing the program in accordance with the programming statements previously stored in the computer's memory. When BASIC is ready for you to type a command or is able to accept a programming statement for storage, the word "READY" is displayed on the terminal. After each such entry the system will prompt for an additional command or programming statement by displaying a "#". When a command is typed, the computer will take immediate action on that command, and when the command has completed it will again respond with the word "READY" and a "#".

This version of BASIC will understand seven different commands: NEW LIST RUN SAVE LOAD APPEND PATCH. Each of these seven commands are described in Table 2.

Command**Name****Description of Command****NEW**

The NEW command causes the current program to be erased. All working storage, internal pointers, and all variables will be reset. The effect of this command is to erase all traces of a program from memory and to start over.

LIST

The LIST command has several formats which can be used to cause all or selected portions of the current program to be displayed on your terminal. The lines are displayed in sequence with the lowest number first. The different LIST formats are as follows:

LIST will list the entire program from start to finish.

LIST 100 will list only line 100.
LIST 100,200 will list lines 100 thru and including line 200.
LIST 100,1 will cause the program to begin listing at line 100 and will continue through to the end of the program. This operation will occur any time the second line number is less than the first.

RUN

The RUN command causes the current program in memory to begin execution at the first statement number. RUN always starts with the lowest line number and will proceed in sequential order unless directed to do otherwise with a GOTO, GOSUB, ON, or IF statement.

SAVE

The SAVE command causes the current program to be saved in a reloadable format on either the SWTPC AC-30 cassette interface or on a TELETYPE papertape punch, or an equivalent device. The control characters necessary to control the recording and playback mechanisms are output along with the program. More complete details are given in LOAD.

LOAD

The LOAD command will first erase any program currently in working storage (as in the command NEW) and will then load a previously saved program. Control characters are output to control the reading mechanism. More complete details are given in APPEND.

APPEND

The APPEND command functions exactly like the LOAD command except that memory is not cleared prior to the load function.

PATCH

The PATCH command causes the computer to return to the MIKBUG operating system and outputs a carriage return, line feed, and an "*" on the terminal device. If no memory belonging to BASIC (addresses _____ to _____) and the program counter (addresses A048 and A049) are not changed, typing a "G" will restart BASIC with your program intact. The PATCH command may even be inserted as a programming statement with a line number in your program. When the PATCH statement is encountered, con-

trol is transferred to MIKBUG. Upon typing a "G", control will return back to the line that immediately follows the patch command.

Note: All seven of the control commands described above may also be used as programming statements, if the control statement is typed following a line number. As a BASIC programming statement their action will be suspended until the program executes them in sequence.

Caution: Using the NEW statement as an executable statement will cause your program to self-destruct!

Program Statements

Each program statement line begins with a line number which must be an integer between 1 and 9999. Statements may be entered in any order, but they will be executed in numerical order. Blanks, unless enclosed by quotation marks, are ignored. Program statements are limited to 72 characters including blanks. Nineteen types of program statements are allowed and are described in the following paragraphs. For your convenience the statements are listed in alphabetical order, rather than someone's "easy-to-learn" sequence. This will make it easier to use as a reference manual.

DATA The DATA Program statement causes data to be stored as part of a program. This data will be used by a READ statement. Data statements do not execute; they merely specify data. Multiple data items in a data statement must be separated by commas. Data statements may be placed anywhere in a program, and will be read in sequence as required. When the data is read in a program, the data will be read in sequence from the first to last data statements, and from left to right within each data statement. Each data item may be read only once unless a RESTORE statement is executed. BASIC keeps track of data with a *pointer*. When the first READ statement is encountered, the pointer indicates that the first data item in the first data statement is to be used; the pointer is then moved to the second item of data, and so on. An example of the DATA program statement format is as follows;

```
100 DATA 123, 34.5, 695, 500
```

DIM The DIM Program statement allocates memory space for an array. In this version of BASIC, one or two dimensioned arrays are allowed. Maximum array size is 255 x 255

elements. (This is not as much of a limitation as you might think—you would need 393,216 bytes of memory to contain an array of 65535 elements! and this doesn't allow for any memory to store the program code. . .) All array elements are set to zero by the DIM statement. Dynamic re-dimensioning of arrays is not allowed. Once an array has been defined, it must not be re-defined in the same program. If an array is not explicitly defined by a DIM statement, it is assumed to be defined either as an array of 10 elements or as an array of 10 x 10 upon the first reference to it in a program.

An array can be allocated only once in a given program, implicitly and explicitly. Only the variables A thru Z (not followed by a number) may be dimensioned. This does not prevent the use of a simple variable of the same name.

The working size of an array may be smaller than its physical size. For example, an array declared 5 x 5 in a DIM statement may be used to store fewer than 25 elements; the DIM statement supplies only an upper bound on the number of elements. In the example of A(5,5) the first position of this array is A(1,1) and the last is A(5,5). An example of the DIM Program statement format is as follows;

```
10 DIM A(5,5), Z(100,2)
```

END The END Program statement causes the program to stop executing. BASIC will print the word "READY" and a "#" on the terminal device indicating that it is now able to accept commands. In this version of BASIC, END may appear more than once and need not appear at all. It is recommended, however, that the last statement of a program be an END statement. This clearly shows that a program has been completely loaded or saved. An example of the END Program statement format is as follows;

```
9999 END
```

FOR The FOR Program statement allows repetition of a group of statements within a program for a specified number of times. The variable name that follows the word 'FOR' is used to identify the related NEXT statement. In addition the variable is initially set to the

value of the first expression (expression1). All statements between the FOR and the related NEXT statement are then executed. The named variable in the FOR statement is then incremented by the STEP value (expression3) and compared to the upper limit (expression2). If the increment creates a value that is *greater than* the upper limit (expression2) control of the computer is passed to the statement immediately following the NEXT statement. If the increment operation results in a sum that is equal to or less than the upper limit, the programming statements that fall between the FOR and NEXT statements will be repeated. This looping process will be continued until the upper limit has been exceeded, or as the result of an IF statement, control is passed to a statement outside the scope of the programming loop. If no STEP value is specified, a value of +1 is assumed. The loop will be executed once, regardless of the value of the variable. Although expressions are permitted for the initial, final, and step values in the FOR statement, they will be evaluated only once, the first time the loop is entered. It is not possible to use the same indexed variable in two loops if they are nested. When the statement after the NEXT statement is executed, the variable is equal to the value that caused the loop to terminate, and not the TO value itself. The step size need not be an integer. For instance,

```
100 FOR N = 1 TO 2 STEP .01
```

is a valid statement which will produce exactly 100 loop executions, incrementing N by .01 each time. A negative step size may also be used, as seen below:

```
100 FOR Q1 = 100 TO 50 STEP - 2.5
```

Examples of proper FOR Program statement formats are as follows;

```
320 FOR K = 1 TO 300
```

•
•

```
500 FOR variable = expression1 TO  
expression2 STEP expression3
```

•
•

```
760 FOR L9 = 200 TO 1 STEP - 1
```

GOSUB

See the description of NEXT statement for additional information.

The GOSUB program statement causes the program to transfer control to the subroutine at the specified line address and returns to the line immediately following GOSUB XXXX. A subroutine is a sequence of instructions which performs a task that has use in more than one place in a program. To be able

Place
10¢
postage
here

dilithium Press
P.O. Box 10766
Portland, OR 97210

programming
; the facility
; sequence from
ogram.
sequence of in-
ntrol upon the
ement. Upon
ie, control is
of the program
atement. The
follows the
ll receive con-
uted.
o a level of 8
may call a sub-
outine, . . . to a
s. An example
nent format is

transfers con-
he statement
ntrol will con-
ew statement.
ram statement

nsfers control
he result of a
B.
control the se-
nts to be ex-
; conditions. If
rue, then con-
; or statement
EN. If the rela-
ogram execu-
ment that im-
atement. The

value of the first expression (expression1). All statements between the FOR and the related NEXT statement are then executed. The named variable in the FOR statement is then incremented by the STEP value (expression3) and compared to the upper limit (expression2). If the increment creates a value that is *greater than* the upper limit (expression2) control of the computer is passed to

The Floppy ROM™ is available to you at no charge. Please print or type your name below and return this card to us. Please allow 4 weeks for delivery.

Name _____

Street Address _____

City _____ State _____ Zip _____

See the description of NEXT statement for additional information.

GOSUB

The GOSUB program statement causes the program to transfer control to the subroutine at the specified line address and returns to the line immediately following GOSUB XXXX. A subroutine is a sequence of instructions which performs a task that has use in more than one place in a program. To be able to use such a sequence of programming statements, BASIC provides the facility (GOSUB) to call upon such a sequence from more than one place in the program.

The subroutine is actually a sequence of instructions that will receive control upon the execution of a GOSUB statement. Upon completion of the subroutine, control is returned back to the mainline of the program by execution of a return statement. The statement that immediately follows the original GOSUB statement will receive control when the RETURN is executed.

Subroutines may be nested to a level of 8 deep. That is, one subroutine may call a subroutine, which can call a subroutine, . . . to a nested depth of 8 subroutines. An example of the GOSUB program statement format is as follows;

```
235 GOSUB 9010
```

GOTO

The GOTO Program statement transfers control of the computer to the statement number specified. Program control will continue sequentially from the new statement. An example of the GOTO program statement format is as follows;

```
100 GOTO 375
```

IF

The IF Program statement transfers control to a specified statement if the result of a conditional comparison is true.

The IF statement is used to control the sequence of program statements to be executed, depending on specific conditions. If the relational expression is true, then control is given to the statement or statement number declared after the THEN. If the relational expression is false, program execution continues at the statement that immediately follows the IF statement. The

possible relational operators for use with the IF statement are as follows;

- = Equal
- <> Not equal
- < Less than
- > Greater than
- <= Less than or equal
- >= Greater than or equal

If a BASIC statement is specified after the THEN condition, rather than a statement number, the statement specified will be executed if the conditional expression is true, and control will be passed to the statement immediately following the IF statement.

It is possible to code another IF statement as the statement following the THEN condition—. This will give the equivalent of an "AND" condition.

When evaluating relational expressions, arithmetic operations take precedence in their usual order, and the relational operators are given equal weight and are evaluated last.

Examples of proper IF program statements formats are as follows;

```
190 IF X = 56 THEN 300
```

•
•
•

```
200 IF Z9 D + Y - (7*S) THEN PRINT Z9;
"Is out of range"
```

•
•
•

```
2022 IF expression1 relational-test ex-
pression2 THEN basic-statement
```

•
•
•

```
2022 IF expression1 relational-test ex-
pression2 THEN statement-number
```

An example of A program to illustrate IF statements is as follows:

```
10 INPUT X
20 IF X = 1 THEN PRINT "EQUAL TO 1"
30 IF X <> 1 THEN PRINT "NOT EQUAL
TO 1"
40 IF X < 1 THEN PRINT "LESS THAN
1"
```

```
50 IF X > 1 THEN PRINT "GREATER
THAN 1"
60 IF X <= 1 THEN PRINT "LESS THAN
OR EQUAL TO 1"
70 IF X >= 1 THEN PRINT "GREATER
THAN OR EQUAL TO 1"
80 PRINT
90 GOTO 10
100 END
```

INPUT

The INPUT program statement allows users to enter data from the terminal during program execution and to assign that value to a variable. The variable specified may be either a simple variable or a specified element in an array.

When the program comes to an INPUT statement, a question mark is displayed on the terminal. The user then types in the requested data separated by commas and followed by a carriage return. If no data is entered, or if the data entered is insufficient, the system prompts the user for more data with an additional question mark. Only numerical constants can be given in response to an input statement. Any number of variables may be specified on the INPUT statement, within the confines of the 72 character line, and the user must respond with a value for each specified variable. An example of the INPUT program statement format is as follows;

```
100 INPUT A,C,D
```

LET

The LET program or assignment statement is used to assign or specify the value of a variable. The value may be an expression, a number, or another variable. The keyword LET is optional, and need not be specified. There are four functions which may be used inside a formula on a LET statement. ABS (absolute value), INT (integer value), RND (random number), and SGN (sign *not sine*) may be used inside formulas or LET statements or anywhere else that a mathematical expression is allowed. Examples of proper LET statement formats are as follows;

```
100 X = 123.45
110 LET Z9 = INT(RND(0)*100)
120 T5 = (A*A)*3.14
```

```

•
•
300A(1,1) = X + P(M,N)
•
•
400Z(I) = X
    
```

NEXT

The NEXT program statement is used to define the end of a "FOR...NEXT" loop. The variable specified in the NEXT statement identifies the associated FOR statement. Multiple FOR...NEXT statements may be used in the same program; They may also be nested (placed inside one another.) When nested, the innermost loop or loops must be wholly contained inside the outermost loop. The range must not overlap. An example of the NEXT program statement format is as follows;

```
220 NEXT K
```

Figure 4, illustrates proper and improper nesting.

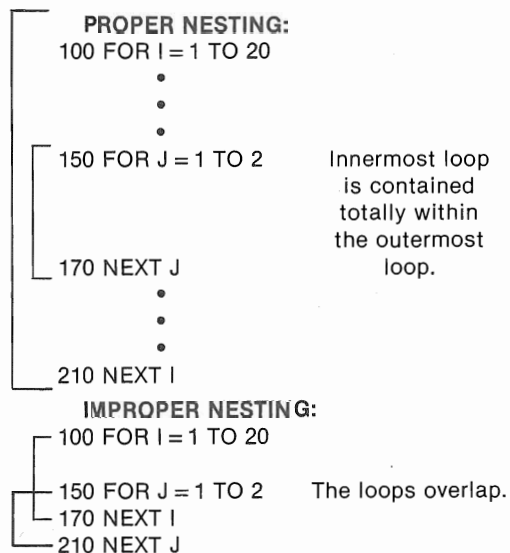


Figure 4. FOR...NEXT NESTING

For additional information see the description of the FOR statement.

ON

The ON program statement provides a computed GOTO or GOSUB mechanism. Instead of specifying the programming sequence of:

```

10 if X = 1 THEN 100
20 if X = 2 THEN 200
30 if X = 3 THEN 300
40 if X = 4 THEN 400
    
```

You could have used the following statement;

```
10 ON X GOTO 100,200,300,400
```

The computer would evaluate X, truncate the resulting value to an integer value, would then select the "Xth" statement number in the list of line numbers, and would use that value as part of a "GOTO" statement. Control of the computer would then be received by the selected line number.

In the example given above if the value of X was between 3 and 3.99999999 then the computer would GOTO 300. If the value of X was 1 then the computer would "GOTO 100". If the value of X is less than 1, or greater than the number of items in the list, the computer will generate an error message, and program execution will stop.

ON X GOSUB 100,200,300,400 works in a similar manner, except that the specified line numbers are assumed to be the start of a subroutine. When a RETURN statement has been executed, control on the computer will be returned to the statement immediately following the "ON X GOSUB" statement.

Note: The value X in the examples above can be any formula or mathematical expression. (Additional note for FORTRAN Programmers or individuals trying to convert FORTRAN to BASIC):

The FORTRAN IF statement can be simulated in BASIC thru the use of the ON...GOTO statement.

```
FORTRAN: IF (X-Y)20,50,10
```

```
BASIC: ON SGN(X-Y)2 GOTO 20,50,10
```

PRINT

The PRINT program statement causes the values of any variables specified in the PRINT statement to be output or printed on the terminal device. The PRINT statement

can also be used to print out text, by enclosing the desired text to be printed in quotation marks (""). The statement

```
100 PRINT "HI THERE, I AM A COMPUTER"
```

would cause the text "HI THERE, I AM A COMPUTER" to be printed on the terminal device (the quotation marks would not be displayed or printed.)

The PRINT statement also makes use of the comma (,) and semicolon (;) to control the formatting of a print line.

In the statement PRINT A,B the numerical value of A will be printed beginning in the left hand margin, and the numerical value of B will be printed in column 18. Basic defines the 72 column line width into 4 zones starting in columns 18,36,48, and 60. The use of a comma in between data items will cause the printing mechanism to be advanced to the start of the next zone. This feature allows you to prepare tabular table data easily. The semicolon disables this "advance to next zone" mechanism and will cause the data items to be printed one after another. Numeric values will be separated by a blank for readability.

In addition if the PRINT statement is terminated by a semicolon, the computer will suppress the normal "carriage return, line feed", that is, automatically output at the end of each line. PRINT statement used with no operations or data items will cause a single blank line to be output.

This version of BASIC supports two special functions (TAB & CHR) that can be used only on the PRINT statement. The TAB function can be used to format items starting in columns other than 18,36,48, and 60. The CHR function is used to create special control characters that may not be generatable from your keyboard.

READ

The READ program statement is similar in function to the INPUT statement except that the next DATA item is read, rather than getting a numeric value from the terminal. READ statements cause values in the DATA buffer to be accessed in a left to right, top to bottom sequential manner and assigned to

the variable named in the read statement. If more than one variable is named in the READ, the values read from the DATA statements will be assigned, in the order read to the variables named on the READ statement. For additional information, see the description of the DATA and RESTORE statements. Examples of proper READ Program statement formats are as follows;

```
100 READ A
200 READ a,q9,B
```

REM

The REM program or remark statement allows insertion of a line of remarks or comments in the listing of a program. REM Lines are saved as a part of a BASIC Program, and are printed when the program is listed, but they are ignored when the program is executing. An example of the REM program statement is as follows;

```
300 REM
```

RESTORE

In the discussion on the DATA and READ statements, it was stated that once a data item was read, it could not be re-read. This is only partially true. The RESTORE program statement is used to reset the internal data pointer that BASIC uses, back to the first data item on the first data statement. Thus the entire data list may be re-read from the beginning.

The only way that you can re-read a particular data item, is to keep track of how far down the data list it is; RESTORE the data pointer, and then READ "n-1" numbers throwing all of the previous numbers away. The data pointer has now been "reset" to a particular data item, so that it can be re-read... clearly not a very practical technique for most programs!!! An example of the RESTORE program statement format is as follows;

```
500 RESTORE
```

RETURN

The RETURN Program statement is used to define the logical end of a subroutine, and when executed will cause control of the computer to be transferred to the statement immediately following the *last* GOSUB executed. A subroutine may call a subroutine, which calls still another subroutine. This

may be done to a nested level of 8 deep. You must take care not to execute a return statement unless you have executed a GOSUB first. An example of the RETURN Program statement format is as follows;

```
1000 RETURN
```

STOP

The STOP program statement operates in a similar fashion to the END statement; both cause the currently running BASIC program to stop running. The STOP statement, however, will identify the line number of the statement that caused the stop; while the END statement merely stops the program without identifying where the "end" occurred. This feature of the STOP statement makes it useful in identifying "error" conditions. For instance, in the following example the computer will print "STOP at 100" if the number typed in response to the INPUT statement is less than zero.

```
90 INPUT X
100 IF X<0 THEN STOP
```

An example of the STOP Program statement format is as follows;

```
900 STOP
```

FUNCTIONS

The following functions are available in Robert Uiterwyk's 4K 6800 BASIC.

ABS(x) The ABS or absolute function will take the absolute value of the expression x. If the value of the expression is a negative number then the value return by the absolute function will be a positive value. A positive value will remain a positive value.

if Z9 is a - 32 then the ABS(Z9) will be a + 32

if Q5 = + 6 then the formula ABS(Q5 + 3) will return a value of +9.

INT(x) The INT or integer function is used to truncate or "chop-off" any decimals in a floating point number. Positive values will be chopped off to the next lower integer, and negative numbers will be rounded up to the next lower value.

INT(9.765) will result in a value of 9.

INT(- 10.2) will result in a value of - 11.

RND(x) The RND or random number generator generates a pseudo-random number ranging

between 0.0 and 1.0. If the argument (x) is not equal to zero, it will be used as a new seed for the random number generator. The value returned as the result of specifying a new seed should be ignored. A new random number will be received when the argument specified is zero. In this version of BASIC it is not necessary to seed the generator in order to start a new random series of numbers at the start of a game.

SGN(x) The SGN Or sign function returns a - 1, 0, or + 1 value depending on the magnitude of the expression (x). If x is less than zero, the SGN will return a value of - 1. If x is equal to zero, SGN will return a value of zero. If x is greater than zero, SGN will return a + 1

if Z9 is a - 45.987 then SGN(Z9) will be a - 1

if Q is a + 34.59 then SGN(Q) will be a + 1

if R is a - 10 and S is a - 5 then SGN (R-2*S) will be 0.

CHR(x) The CHR function is used to convert the value of X (which must be between 2 and 255) to a single ASCII character. Thus CHR(16) will create a "CONTROL-P" and CHR(22) will create a "CONTROL-U". The CHR function may be used only on the PRINT statement and is useful for creating special control characters required for terminal control.

TAB(x) The TAB function can be used only in a PRINT statement. The value of X will cause the print mechanism to be positioned at column X. TAB(10) will cause a tab to column 10, TAB(50) will cause a tab to column 50. The value of the TAB function must be from 1 to 72.

USER The USER function is used to call a user defined assembly language subroutine.

OPERATORS

Symbols used to instruct the microcomputer to perform some operation are called operators. This version of BASIC includes arithmetic and relational operators.

ARITHMETIC OPERATORS—Five standard BASIC arithmetic operators are provided with this versin of BASIC. The arithmetic operators are as follows:

- Unary negation
- * Multiplication
- / Division
- + Addition
- Subtraction

RELATIONAL OPERATORS—Standard BASIC relational operators are provided for comparing the values of integer expressions. The relational operators are as follows:

- = Equal
- <> Not equal
- < Less than
- > Greater than
- <= Less than or equal
- >= Greater than or equal

ERROR MESSAGES

The following error messages can be created by BASIC at any time:

ERROR CODE	MESSAGE DESCRIPTION
1	OVERSIZED VARIABLE. A TAB, CHR, ON or subscript value was greater than 255.
2	INPUT ERROR.
3	ILLEGAL CHARACTER OR VARIABLE. An illegal variable name or an invalid character was found during the evaluation of a mathematical expression.
4	NO ENDING " IN PRINT LITERAL. A missing quotation mark was detected in a print statement. There must be an even number of quotation marks on a print statement. The character (") cannot be output as a text character except by specifying CHR(34).
5	DIMENSIONING ERROR. Only two dimensions are allowed, and must be a value between 1 and 255. Once defined an array may not be defined again. Place all DIM statements at the start of the program, where they can not be accidentally executed a second time. All references to an array which is defined on a DIM statement must occur after the DIM statement has been executed.
6	ILLEGAL ARITHMETIC. The most common cause of "illegal arithmetic" is two arithmetic operators following each other, without a

numeric value separating them. "2+ -3" is illegal arithmetic.

- 7 LINE NUMBER NOT FOUND. The "GOTO", "GOSUB" or "IF...THEN" line number does not exist in the program. If necessary a REM statement may be used as a dummy line to resolve a missing line number.
- 8 DIVIDE BY ZERO ATTEMPTED. You just can't do it! If there is no way to prevent it, place an "IF" statement prior to the divide and branch around the divide statement if a divide by zero might occur.
- 9 EXCESSIVE SUBROUTINE NESTING. The maximum number of subroutines that may be nested is 8. Generally when this occurs it is because a GOSUB is accidentally being executed, or somehow a return statement has not been executed.
- 10 RETURN without PRIOR GOSUB. You executed a RETURN statement without first executing a GOSUB. This is a no-no.
- 11 ILLEGAL VARIABLE. The name for the variable is not correct. An array may only be named with a single alphabetic character. A(1) is correct, A5(1) is not. A simple variable may be named either with a single alphabetic character, OR a single alphabetic character followed by a single numeric character. A, B0, C1 are legal names. 1A, BB, C3A are not legal names. This version of BASIC does not support character string variables (A\$, B\$, etc.)
- 12 UNRECOGNIZABLE STATEMENT. This first word in the statement was not a recognizable COMMAND or BASIC statement name; nor was it a variable (simple or array) name followed by an equal sign (=). This latter case is known as an implied LET statement.
- 13 PARENTHESIS ERROR. Either you have too many opening parenthesis "(" or too few closing ones ")". There must be exactly the same number of opening and closing parenthesis in a programming statement.
- 14 MEMORY FULL. You just ran out of memory. Try to reduce the number of variables used in the program. If arrays are being used make

sure that they are dimensioned to the exact size required. Reduce the amount of remarks and the amount of PRINT statements used. Look and see if any routines are similar enough in nature to be made a subroutine. Look for more efficient algorithms to solve your problem. Split the program up into several different programs. If all else fails, go buy some more memory!

- 15 SUBSCRIPT ERROR. The array was defined as a one dimensional array and the reference contains two dimensions, or vice-versa.
- 16 EXCESSIVE FOR LOOPS ACTIVE. The maximum number of nested FOR...NEXT loops is 8.
- 17 NEXT WITHOUT CORRESPONDING FOR. A NEXT statement has been found without a corresponding FOR statement. The named variable that occurs after the word "FOR" and after the word "NEXT" is used to identify corresponding "FOR" and "NEXT" statements.
- 18 MISNESTED FOR NEXT LOOPS. The computer has detected an improper nesting for multiple "FOR...NEXT" statements. See Figure 3 for an explanation of the proper and improper way to nest "FOR...NEXT" statements.
- 19 READ STATEMENT ERROR. Read statement was executed, and there is not enough data in the data buffer to satisfy the READ request.
- 20 ERROR IN ON STATEMENT. The expression when truncated to an integer value resulted in either a value less than one, or in a value that is greater than the number of statement numbers in the list. In the example below an error 20 will occur if the value of (X-Y) is less than 1 or greater than 5.
100 ON(X-Y) GOTO 1000,235,3000,20,35
- 21 INPUT OVERFLOW. More than 72 characters were typed on the line.

GRAMMAR REFERENCE SUMMARY

Table 3, Quick definition of BASIC Statements, and Table 4, BASIC functions, provide a quick reference for using, Robert Uiterwyk's 4K 6800 BASIC Interpreter.

Table 3. Quick definition of BASIC Statements

STATEMENT	DEFINITION	RELATED STATEMENT(S)
DATA	'STORES' DATA TO BE USED BY READ STATEMENT	READ RESTORE
DIM	DEFINES SIZE OF ARRAYS (1 or 2 dimensions are allowed.)	
END	DEFINES END OF PROGRAM	STOP
FOR	DEFINES START OF PROGRAMMING LOOP	NEXT
GOSUB	TRANSFERS CONTROL TO A SUBROUTINE	RETURN
GOTO	TRANSFERS CONTROL TO OTHER THAN THE NEXT SEQUENTIAL STATEMENT	
IF	CONDITIONAL TEST USUALLY USED TO TRANSFER CONTROL "IF"	THEN
INPUT	GET DATA FROM KEYBOARD OF ATTACHED TERMINAL	
LET	ASSIGNMENT OF CALCULATION STATEMENT	
NEXT	DEFINES END OF PROGRAMMING LOOP	FOR
ON	COMPUTED "GOTO" OR "GOSUB"	GOSUB GOTO
PRINT	PUT DATA TO PRINTER OR CRT OF ATTACHED TERMINAL	
READ	GET DATA FROM "DATA" STATEMENTS	DATA RESTORE
REM	PROGRAMMER'S COMMENT	
RESTORE	RESETS THE DATA POINTER IN "DATA" STATEMENT	DATA READ
RETURN	DEFINES END OF SUBROUTINE — CAUSES CONTROL TO BE TRANSFERRED BACK TO STATEMENT IMMEDIATELY FOLLOWING THE GOSUB STATEMENT	GOSUB

STOP DEFINES THE END OF END
 THE PROGRAM: AND
 IDENTIFIES THE
 STATEMENT NUMBER
 OF THE STOP.

Table 4. BASIC Functions

ABS() ABSOLUTE VALUE
 CHR() CONVERSION OF VALUE TO A SINGLE
 CHARACTER
 INT() INTEGER VALUE (TRUNCATES DECIMAL
 VALUE)
 RND() RANDOM NUMBER GENERATOR; RETURNS
 VALUE FROM .000001 to .999999
 SGN() SIGN OF X; RETURNS A + 1, 0, or - 1
 TAB() USED IN PRINT STATEMENT ONLY;
 CAUSES A TAB TO THE SPECIFIED
 COLUMN
 USER() CAUSES A CALL TO A USER DEFINED
 ASSEMBLY LANGUAGE SUBROUTINE

Appendix A

General Software Index

Compiled by Jim Schreier

The software published in *Interface Age* between December 1976 and December 1978 is rich and varied. Historically, this time period marked the apex of "hobbyist" software and the dawning of "business" programs. Generally, the 1977 pages of *Interface Age* contained assembled programs for the SC/MP, 6800 and 8080. The 1978 edition published software for a wider variety of chips plus BASIC, FORTRAN and PILOT programs. This variety of subject matter will interest most microcomputer users as well as provide a permanent record of excellent software.

This compilation is divided into two sequential sections: Software by High Level Language and Software by CPU chips.

The High Level Languages are in alphabetical order by the name of the BASIC. Most users will want to see what programs were published for their specific microcomputer. The term "GENERAL BASIC" means the documentation failed to identify a specific BASIC dialect. Of course, there is no such thing as a "General BASIC," yet. It is possible to translate one BASIC to another given enough time.

Likewise, the assembled software is arranged by CPU chips in numeric order. Some of the most important published software programs appear in this section. Foremost is the publication in four parts of the Livermore BASIC Interpreter.

ALTAIR BASIC v 4.0 Building and land development program
 BASIC listing November 1978 p 88

ALTAIR Extended BASIC v 3.2 Computer(ese) speech writer
 BASIC listing January 1978 p 150

ALTAIR Extended disk BASIC v 4.0 Number base conversion
 2 BASIC listings for disk and non-disk applications May 1977 p 116

ALTAIR 8K BASIC v 3.2 Star Lanes
 Fixes to the computer game October 1977 p 8