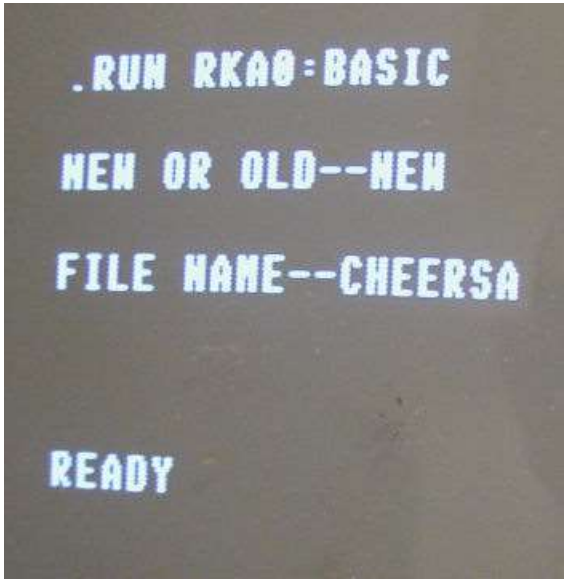
Advanced BASIC



Vintage Computer Festival 9.1

Bill Degnan

Dartmouth College Computation Center

BASIC

Beginners' All-purpose Symbolic Instruction Code

Instruction Manual
May, 1964
First Draft

PREFACE

When plans were made for the Dartmouth College time-sharing system, which will enable 20 or more people to use the computer at the same time, the need arose for a language to meet several requirements:

1./ It should be very easy to learn. This will enable faculty and students to obtain useful information from the computer without an undue investment in learning machine languages.

2./ It should be possible to change programs from this language to the language of the machine ("compile") quickly. This is a necessity when twenty people share the time of the computer.

3./ It should be a stepping-stone for students who may later wish to learn one of the standard languages, such as FORTRAN or ALGOL.

4./ It should be a general purpose language; that is, every kind of machine computation should be programmable in it.

BASIC was constructed to meet these needs. And it has endeavored to stay as close to ordinary English as possible. As evidence for this we present, without any explanation, a program written entirely in BASIC:

```
LET X = (7+8)/3
PRINT X
END
```

Course Outline

- BASIC Overview
- Matricies
- BASIC Timing Comparisons 1977
- Micro-Soft vs. Tiny BASIC
- Micro-Soft BASIC Breakdown using PEEK

BASIC Overview

- BASIC has three classes of capabilities: commands, statements, and functions.

- Commands “part of the operating system or environment” and manipulate global items, such as programs

- Statements are made up of keywords, variables, constants, operators, and functions

- Functions - A user-defined and library functions.

BASIC Overview

• Constants. BASIC programs are made up of statements that contain keywords, variables, operators, and constants

• Numeric constants (Floating point and Integer)
  - Each BASIC version handles numeric constants differently.

• Character String constants
  - signaled by a quote (")

• Variables
  - "names" that may take on different values during a problem.
  - vintage versions of BASIC required variables to start with a letter.

Summary of BASIC Statements

1. Elementary BASIC

	Purpose	*Example*
INPUT	Reads data from teletype	17 INPUT A$, X
READ	Reads data from data block	17 READ X, Y1, M(J + 2, 3), N$
DATA	Storage area for data	17 DATA – 1, 2.07, 31416E – 4, 127829, JONES
PRINT	Types numbers and labels	17 PRINT "ANSWER ="; X, A * B, N$
LET	Computes and assigns value	17 LET X2 = X + Y ↑ 2
GO TO	Transfers control	17 GO TO 175
IF	Conditional transfer	17 IF T(I, J) < = 25 THEN 175
FOR	Sets up and operates a loop	17 FOR N = 10 TO 1 STEP – 1
NEXT	Closes loop	17 NEXT N
END	Final statement in program	17 END

2. Advanced BASIC

INPUT	Reads data from the teletype
LINPUT	Inputs an entire line as a single string
DEF	Defines a function
FNEND	End of a multiple-line DEF
GOSUB	Transfers to a subroutine
RETURN	Returns to statement following GOSUB
RESTORE	Restores data to beginning
REM	Permits comments
DIM	Declares dimensions of lists and tables
STOP	Stops program
CHANGE	Convert string of characters to a vector, vice versa
ON	Multiple way branch
RANDOMIZE	"Randomize" the random number generator

```
17  INPUT X, Y4, Z
17  LINPUT A$
17  DEF FNG (X) = 2 * SIN (X) * EXP (– X)
17  FNEND
17  GOSUB 800
17  RETURN
17  RESTORE
17  REM BEGINNING OF SUBROUTINE
17  DIM A(12), B(3, 5)
17  STOP
17  CHANGE A$ TO V

17  ON X+Z GO TO 200, 400, 750
17  RANDOMIZE
```

3. File instructions

FILES	Specified files	17	FILES DATA; *
FILE	Names or renames a file	17	FILE #2: N$
INPUT	Reads from a teletype file	17	INPUT #1: X, A$, Y
PRINT	"Prints" to a teletype file	17	PRINT #1: A, B, C
READ	Reads from a random file	17	READ #2: X, Y(1), Y(2)
WRITE	Writes to a random file	17	WRITE #2: R, S+T
RESET	Set random file pointer	17	RESET #2: LOC(2) – 1
SCRATCH	Scratch a file	17	SCRATCH #3

4. Matrix instructions

MAT INPUT	Reads data, any number, from teletype	17	MAT INPUT V
MAT READ	Reads a matrix from the data block	17	MAT READ Z(M, N)
MAT PRINT	Types a vector or matrix	17	MAT PRINT A
MAT +	Matrix addition	17	MAT C = A + B
MAT –	Matrix subtraction	17	MAT C = A – B
MAT *	Matrix multiplication	17	MAT C = A * B
MAT ( ) *	Scalar multiplication	17	MAT C = (COS (X)) * A
MAT INV	Matrix inverse	17	MAT C = INV (A)
MAT TRN	Matrix transpose	17	MAT C = TRN (A)
MAT ZER	Matrix of all zeroes	17	MAT C = ZER
MAT CON	Matrix of all ones	17	MAT C = CON (15)
MAT IDN	Identity matrix	17	MAT C = IDN

5. Notes

Variables	X, Y7, A, A(X), B(A(X), 5), A$, B7$, N$(I), T$(A,B)
Operations	+, –, *, /, ↑
Relations	<, < =, =, >, > =, < >
Functions	SQR, SIN, COS, TAN, ATN, LOG, EXP, ABS, SGN, INT, RND, ASC, LOC, LOF, NUM, TAB

Matricies

(1966 Dartmouth BASIC)

- A matrix is simply a rectangular array of numbers
- An array is a set of numbers arranged in rows and columns
- A matrix may also consist of a single row or a single column, also called “row vectors” (lists) and “column vectors”.

```
10 REM MATRICIES USING DARTMOUTH BASIC
20  DIM S(2,2)
30 MAT READ S
...
240 DATA 30, 50
245 DATA 40, 25
250 FOR K=1 TO 2
260 PRINT S(K,1)
270 NEXT K

RUN
[What would be the output??]
```

Matricies

(Digital PDP 11 BASIC)

```
10 DIM A(2,3)
20 FOR I=0 TO 2
30 FOR J=0 TO 3: LET A(I,J) = 0
40 NEXT J
50 NEXT I
60 FOR I = 0 TO 2: LET A(I,0) = I
70 FOR J = 0 TO 3: LET A(0,J) = J
80 PRINT A(I,J);
90 NEXT J
100 PRINT
110 NEXT I
120 END

RUN
0  1  2  3
1  0  0  0
2  0  0  0
STOP AT LINE 120
READY
```

What's different?

The Knight's Tour

• Chess is played on a square board having 64 smaller squares, eight on a side.

• The knight moves in an L-shaped path, moving one square in any direction and two squares in a direction perpendicular to the first move.

• A knight may move to any of eight possible positions (assuming space permits).

• If the knight occupies position (3, 4), then he may move to any one of the following: (4,6), (4, 2), (2, 6), (2, 2), (5, 5), (5, 3), (1, 5) or (1, 3).

• In general, if the knight occupies position *(r,* c), then he may move to any of the following: (r + 1, e + 2), (r + 1, e - 2), (r - 1, e + 2), (r - 1, e - 2), (r + 2, e + 1), (r + 2, e - 1), (r - 2, e + 1), or (r - 2, e - 1), unless the new position is off the board.

• An ancient and intriguing challenge is to move the knight about the board in such a way that it visits all 64 squares of
the chess board exactly once.

```
 94   REM * THIS PRØGRAM CARRIES ØUT A RANDØM
 95   REM  KNIGHT'S TØUR TØ DEAD END
100   DIM B(8,8),T(2,8),U(8,2)
110   MAT B = ZER
120   MAT READ U
130   LET M = 1
140   PRINT  "BEGIN WHERE";
150   INPUT R, C
160   LET B(R,C) = M
170   MAT T = ZER
172
174   REM * K1 CØUNTS THE NUMBER ØF LEGAL MØVES
180   LET K1 = 0
182
184   REM * ENTER ALL LEGAL MØVES IN T ARRAY
190   FØR T = 1 TØ 8
200      LET R1 = R + U(T,1)
210      LET C1 = C + U(T,2)
220         IF INT( (R1-1)/8 ) <> 0 THEN 280
230         IF INT( (C1-1)/8 ) <> 0 THEN 280
240         IF B(R1,C1) <> 0 THEN 280
250      LET K1 = K1+1
260      LET T(1,K1) = R1
270      LET T(2,K1) = C1
280   NEXT T
282
290      IF K1 = 0 THEN 350
292
294   REM * SELECT A LEGAL MØVE AT RANDØM
300   LET T = INT( RND(-1)*K1+1 )
310   LET R = T(1,T)
320   LET C = T(2,T)
330   LET M = M+1
340   GØTØ 160
342
350   PRINT  "GØT TØ"; M
360   PRINT  "PRINT IT";
370   INPUT A$
380      IF A$ <> "YES" THEN 420
390   MAT PRINT B;
392
394   REM
400   DATA  1,2, 1,-2, -1,2, -1,-2
410   DATA  2,1, 2,-1, -2,1, -2,-1
420   END
RUN
TØUR
```

The Knight's Tour

```
BEGIN WHERE?4,4
GØT TØ 41
PRINT IT?YES
 0     8     15    0     0     24    39    0
 14    11    0     7     38    0     0     25
 0     16    9     12    0     26    23    40
 10    13    18    1     6     37    0     0
 17    0     5     36    27    22    41    0
 0     0     2     19    32    35    28    0
 0     0     0     4     0     30    21    34
 0     3     0     31    20    33    0     29
```

BASIC Timing Comparisons

. . . revisited and updated

Tom Rugg
1115 N. Beverly Glen Blvd.
Los Angeles CA 90024

Phil Feldman
1722 Brockton Ave.
Los Angeles CA 90025

Z-80 - More efficient cycles

6800 – A minimum of 2 cycles are required to execute a single byte instruction (2 microseconds)

8080 – A minimum of 4 cycles (each 500 nsec) are required to execute a single byte instruction (2 microseconds)

Z80 – A minimum of 4 cycles are required to execute a single byte instruction (1.6 microseconds)

6502 vs. 8080 vs. Z-80

Benchmark Test in Kilobaud Magazine

Intel 8080 / Zilog Z-80 / MOS 6502

- Thirty-one BASICs tested, the four 6502 versions placed in the top five spots yielding only second place to the Zilog’s new Z80 running at 4 MHz.
- Why so fast? The 6502's many addressing modes make it very efficient and easy to program.

```
300 PRINT"START"
400 FOR K=1 TO 1000
500 NEXT K
700 PRINT"END"
800 END
```

Fig. 1. Benchmark Program 1.

```
300 PRINT"START"
400 K=0
500 K=K+1
600 IF K<1000 THEN 500
700 PRINT"END"
800 END
```

Fig. 2. Benchmark Program 2.

```
300 PRINT"START"
400 K=0
500 K=K+1
510 A=K/K*K+K-K
600 IF K<1000 THEN 500
700 PRINT"END"
800 END
```

Fig. 3. Benchmark Program 3.

```
300 PRINT"START"
400 K=0
500 K=K+1
510 A=K/2*3+4-5
600 IF K<1000 THEN 500
700 PRINT"END"
800 END
```

Fig. 4. Benchmark Program 4.

```
300 PRINT"START"
400 K=0
500 K=K+1
510 A=K/2*3+4-5
520 GOSUB 820
600 IF K<1000 THEN 500
700 PRINT"END"
800 END
820 RETURN
```

Fig. 5. Benchmark Program 5.

```
300 PRINT"START"
400 K=0
430 DIM M(5)
500 K=K+1
510 A=K/2*3+4-5
520 GOSUB 820
530 FOR L=1 TO 5
540 NEXT L
600 IF K<1000 THEN 500
700 PRINT"END"
800 END
820 RETURN
```

Fig. 6. Benchmark Program 6.

```
300 PRINT"START"
400 K=0
430 DIM M(5)
500 K=K+1
510 A=K/2*3+4-5
520 GOSUB 820
530 FOR L=1 TO 5
535 M(L)=A
540 NEXT L
600 IF K<1000 THEN 500
700 PRINT"END"
800 END
820 RETURN
```

Fig. 7. Benchmark Program 7.

	Software/Hardware	Benchmark Number 1	2	3	4	5	6	7
	1. TRW Expanded BASIC/ Control Data Cyber 174	.08	.07	.15	.15	.24	.80	1.04
	*2. 6K Integer BASIC/Apple	1.3	3.1	7.2	7.2	8.8	18.5	28.0
	3. Zapple 8K BASIC (1.1)/ Altair 8800a, TDL ZPU @ 2 MHz	1.7	9.5	20.6	21.7	23.7	36.2	51.8
	4. Altair 8K BASIC (4.0)/ Altair 8800b	1.7	10.2	21.0	22.5	24.3	36.7	52.4
	5. Altair 8K BASIC (3.2)/ Altair 8800a	1.7	10.3	21.4	23.1	24.8	37.3	52.8
	6. Altair 8K BASIC (3.0)/ Imsai 8080	1.6	10.6	22.0	23.7	25.4	38.3	57.1
(tie)	7. Digital Group Z-80 Maxi-BASIC (1.0)/Digital Z-80 @ 2.5 MHz	1.8	7.5	21.2	25.1	26.9	40.3	58.5
(tie)	7. Altair 12K Extended BASIC (4.0)/Altair 8800b	1.9	7.5	20.6	20.9	22.1	37.0	58.5
(tie)	7. Altair Disk Extended BASIC (4.0)/Altair 8800b	1.9	7.5	20.6	20.9	22.1	36.9	58.5
	10. Altair 12K Extended BASIC (3.2)/Altair 8800a	1.9	8.9	21.8	23.0	24.8	39.3	60.7
	11. Altair Disk Extended BASIC (3.4)/Altair 8800b	1.9	8.8	21.7	22.8	24.7	39.6	61.6
	12. Altair 4K BASIC (4.0)/ Altair 8800b	1.9	15.1	26.0	28.9	31.7	44.5	62.1
	13. Compal-80 10K BASIC/ Compal-80	2.0	9.3	23.4	24.6	26.3	42.1	65.7
	14. Process Technology BASIC 5/ Digital Group 8080	3.6	10.5	27.5	30.9	33.2	51.3	67.4
	15. Compucolor 8K BASIC/ Compucolor 8001	2.1	13.1	27.0	29.0	31.3	47.5	67.8
	16. Digital Group 8080 Maxi-BASIC (1.0)/Digital Group 8080	2.2	9.2	26.4	31.2	33.5	49.9	72.3
	17. Altair 680 8K BASIC (3.2)/ Altair 680b	2.5	16.3	30.7	33.4	36.3	55.9	81.8
	*18. Programma TBX (1.1)/ Sphere 330	2.9	34.8	55.6	60.5	73.5	101.1	159.8
	19. Imsai 8K BASIC (1.3)/ Imsai 8080**	7.1	----	44.1	56.2	----	105.7	194.9
	*20. Programma TBX (1.2)/ Sphere 330	3.1	41.2	68.3	72.2	85.3	115.3	202.8
	21. Southwest Tech 8K BASIC (1.0)/ SWTPC 6800	14.9	24.7	96.1	105.3	109.8	174.1	204.5
	***22. Imsai 8K BASIC (1.31)	7.5	28.2	66.4	78.5	88.1	140.1	235.6

*Integer BASIC only.
**Some benchmarks could not be run. Others are incomplete.
***Results for Imsai 8K BASIC, Version 1.31, from tests conducted by Tom Rugg after article submitted for publication.

Fig. 8. Benchmark timings (in seconds).

First Round Conclusions

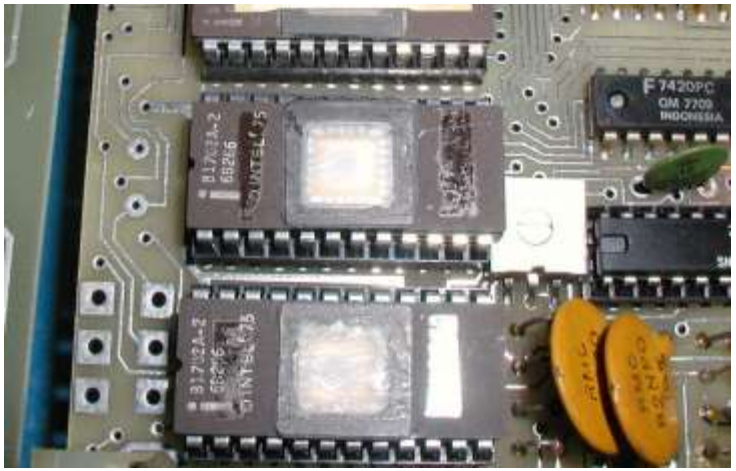
- Integer BASIC has an unfair advantage.
- TDL's Zapple 8K BASIC just barely came out in front of Altair 8K BASIC overall.
- Wide variations of BASIC same processor (eg. Altair vs. IMSAI).
- Not much variation between 8080 and Z80
- 6800-based BASICs are down at the bottom of the list.
- Benchmarks help show which BASICs are faster at specific tasks.

SWTPc BASIC (6800)

- SWTP BASIC generates nine significant digits whereas "everyone else uses six." This additional accuracy contributes to an overhead increase of about 50%.
- BCD arithmetic operations contribute approximately 20% in increased overhead but also provide increased accuracy for those operations.
- The transcendental functions (sine, cosine, tangent, exponents, etc.) should execute twice as fast as before due to modifications found in Version 2

Second Round 10/77 Kb

- Removed Control Data and Apple Integer BASIC
- Only testing BASICs with floating point arithmetic
- Ohio Scientific Instruments 8K BASIC, new and top of list
- North Star BASIC-FPB. That's the one that uses North Star's hardware floating point board. Quite an improvement over North Star BASIC without the FPB for number crunching benchmarks
- Of note - Tektronix 4051 (which uses the 6800 chip). It's not particularly fast, but has 14 digits of precision, graphics, and file I/O handling.
- Micropolis BASIC 1.1 added, not fast
- IBM 5100 not fast, expensive



OSI 500
ROM Pot

Bill Gates' Perspective

- Microsoft wrote Altair BASIC (8080 and 680 versions), OSI BASIC, and PET BASIC
- Altair BASIC can be made into an integer BASIC by using a DEFINT A-Z statement at the start. "..I still think the Apple BASIC would be faster since it has so little complexity to deal with in variable handling, and the 6502 is an inherently faster processor. .."
- Bill was the one to suggest OSI BASIC to be tested.
- Newer versions of Altair BASIC are much faster than v1.0 – "I wish someone still had a version of Altair 1.0 around to do a comparison with, since that would show how much Microsoft has improved Altair BASIC since its inception."
- "Altair 680 BASIC, also written by Microsoft, looks slower than the 8080-based BASICs only because the 680 runs the clock of the 6800 at half its normal speed. Taking the 680 BASIC times and dividing them by two shows
- that the 6800 instruction set is more speed efficient, albeit less byte efficient, than the 8080 instruction set for programs as complex as a BASIC."

Mico-Soft BASIC Stolen?

"Something that should be very surprising about your chart is how close some of the BASICs seemed to be to each other, even some of the non-Altair BASICs. In fact, if you take into account the use of a different processor, the overhead for I/O interrupts and different clock speeds, a number of these independent BASICs take an identical amount of time as one of the versions of Altair BASIC! This becomes more than a coincidence when you consider that a number of signatures have been put into Altair BASIC intentionally and these also appear in the so-called competitive BASICs. Besides comparing apples to oranges and legitimate software to illegitimate software, your article was very worthwhile. I look forward to seeing a complete chart with only real BASICs on it."

Was Bill correct?

Poking around a little at the OSI version of Micro-Soft BASIC I discovered a before-now unknown Easter Egg: Type "A" in response to the question MEMORY SIZE? ….

Richard W. Weiland is the credited author of OSI Micro-soft BASIC. They meant business!

	Software; Hardware	CPU Type	Benchmark Number 1	2	3	4	5	6	7	Notes
	1. OSI 8K BASIC (ver 1.0, rev 3.3); OSI Challenger @ 2 MHz	6502	.9	4.6	8.2	9.3	10.0	14.8	21.6	b
	2. Zapple 8K BASIC (1.1); Altair 8800a, Cromemco Z-80 @ 4 MHz, 1 wait state	Z-80	.9	5.9	13.0	13.5	14.8	22.7	32.7	a
	3. OSI 8K BASIC (ver 1.0, rev 3.2); OSI Challenger @ 1 MHz	6502	1.4	8.6	15.9	17.8	19.3	28.7	42.2	a
	4. OSI 8K BASIC (ver 1.0, rev 3.3); OSI Challenger @ 1 MHz	6502	1.6	8.9	16.2	18.2	19.7	29.2	42.9	b
	5. PET BASIC; Commodore PET 2001 (prototype)	6502	1.7	9.8	18.6	20.4	22.1	32.6	51.3	b
	6. Zapple 8K BASIC (1.1); Altair 8800a, TDL ZPU @ 2 MHz	Z-80	1.7	9.5	20.6	21.7	23.7	36.2	51.8	a
	7. Altair 8K BASIC (4.0); Altair 8800b	8080	1.7	10.2	21.0	22.5	24.3	36.7	52.4	a
	8. Altair 8K BASIC (3.2); Altair 8800a	8080	1.7	10.3	21.4	23.1	24.8	37.3	52.8	a
	9. Altair 8K BASIC (3.0); Imsai I-8080	8080	1.6	10.6	22.0	23.7	25.4	38.3	57.1	a
(tie)	10. Digital Group Z-80 Maxi-BASIC (1.0); Digital Group Z-80 @ 2.5 MHz	Z-80	1.8	7.5	21.2	25.1	26.9	40.3	58.5	a
(tie)	10. Altair 12K Extended BASIC (4.0); Altair 8800b	8080	1.9	7.5	20.6	20.9	22.1	37.0	58.5	a
(tie)	10. Altair Disk Extended BASIC (4.0); Altair 8800b	8080	1.9	7.5	20.6	20.9	22.1	36.9	58.5	a
	13. North Star BASIC-FPB (ver 6); Altair 8800a, North Star Fl. Pt. Board	8080	1.9	9.1	18.4	18.5	20.9	36.1	59.4	b
	14. Altair 12K Extended BASIC (3.2); Altair 8800a	8080	1.9	8.9	21.8	23.0	24.8	39.3	60.7	a
	15. Altair Disk Extended BASIC (3.4); Altair 8800b	8080	1.9	8.8	21.7	22.8	24.7	39.6	61.6	a
	16. Altair 4K BASIC (4.0); Altair 8800b	8080	1.9	15.1	26.0	28.9	31.7	44.5	62.1	a
	17. Compal-80 10K BASIC; Compal-80	8080	2.0	9.3	23.4	24.6	26.3	42.1	65.7	a
	18. Processor Tech BASIC 5; Digital Group 8080	8080	3.6	10.5	27.5	30.9	33.2	51.3	67.4	a
	19. Compucolor 8K BASIC; Compucolor 8001	8080	2.1	13.1	27.0	29.0	31.3	47.5	67.8	a
	20. Digital Group 8080 Maxi-BASIC (1.0); Digital Group 8080	8080	2.2	9.2	26.4	31.2	33.5	49.9	72.3	a
	21. North Star BASIC (ver 6); Altair 8800a	8080	2.3	9.5	26.6	31.3	33.7	50.6	73.8	b
	22. Poly 11K BASIC (ver 9V27); Poly-88 System 16	8080	2.5	10.2	29.0	34.0	36.5	54.0	79.0	b
	23. Altair 680 8K BASIC (3.2); Altair 680b	6800	2.5	16.3	30.7	33.4	36.3	55.9	81.8	a
	24. Poly 11K BASIC (ver A00); Poly-88 System 16	8080	2.5	11.3	31.3	36.1	39.3	58.6	87.6	b
	25. Tektronix Level 5 BASIC; Tektronix 4051	6800	4.8	14.0	33.0	36.2	40.7	68.8	103.8	b
	26. Micropolis BASIC (1.1); Altair 8800a, Mod II Micropolis disk	8080	8.7	19.9	50.4	54.1	58.2	109.9	146.4	a
	27. IBM 5100 BASIC; IBM 5100	??	4.0	20.5	56.5	54.3	58.1	87.0	172.8	a
	28. Southwest Tech 8K BASIC (1.0); SWTPC 6800	6800	14.9	24.7	96.1	105.3	109.8	174.1	204.5	a
	29. Imsai 8K BASIC (1.31); Imsai I-8080	8080	7.5	28.2	66.4	78.5	88.1	140.1	235.6	a,c
	30. Imsai 8K BASIC (1.3); Imsai I-8080, 1702 ROM, 1½ wait states	8080	11.5	39	92	110	121.5	191	320	b
	31. SCELBAL BASIC; MIKE-2	8008	312	369	460	515	760	1814	2151	b

Note a: Timings done by authors.
Note b: Timings were sent in by someone else.
Note c: This is a modified version of 1.3 that works in RAM. Not yet released by Imsai.

Fig. 8 Benchmark Timings (in seconds).

MITS Caravan comes to Ricky's Hyatt House

- Event near the Homebrew Computer Club meeting location early1975
- The Homebrew members all came to see the Altair 8800…and they saw a working BASIC, most for the first time.
- At this time no one who had ordered BASIC received their copy and everyone was pretty impatient about it…
- Someone stole a tape from the session.
- Pretty soon there were dozens of copies distributed on papertape, before the official release of the program.

Gate's Open Letter to Hobbyists

Bill Gates, angry with the people who copied his BASIC wrote a letter to the Homebrew Computer Club, The People's Computer Company, etc. This

"..The feedback we have gotten from the hundreds of
people who say they are using BASIC has all been
positive. Two surprising things are apparent, however.
(1) Most of these "users" never bought BASIC (less than 10% of Altair owners have bought BASIC),
(2) the amount of royalties we have received from
sales to hobbyists makes the time spent of Altair BASIC
worth less than $2 an hour. .."

".. As the majority of hobbyists must be aware, most of you steal your software. Hardware must be paid for, but software is something to share. Who cares if the people who worked on it get paid?…"

-Bill Gates

Gate's Open Letter to Hobbyists

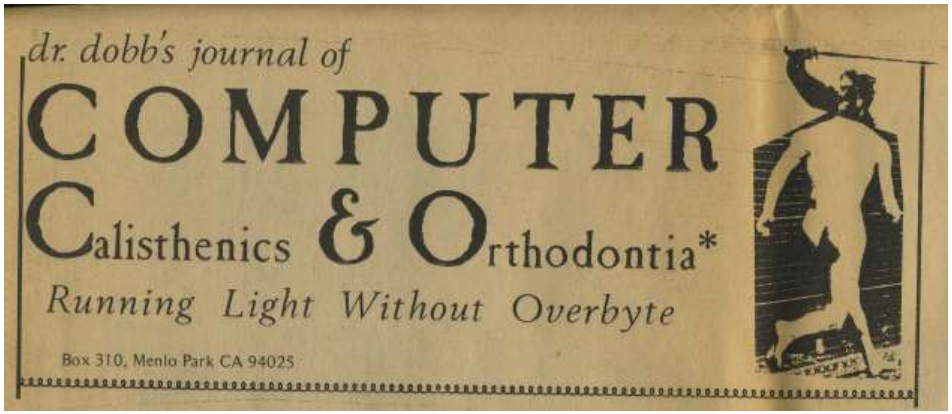
- Gates "Open Letter to Hobbyists" – claimed that copying software was theft. Created quite a controversy, "software flap"
- Hackers: "programs don't belong to anybody"

Who is right?

- Space War – there was no market, therefore not a problem to copy freely.
- Regardless, when other companies needed a BASIC, they went to Micro-soft. It became the standard.

The Tiny BASIC Project

- Tiny BASIC – the People's Computer Company attempt at making an public domain / open source BASIC. Bob Albrecht.
- People immediately started submitting improved versions of the BASIC published for free in the PCC tabloid. Toggling in the code each time they used the computer.
- Bob hired Jim Warren to edit the Dr. Dobbs Journal, the off-shoot publication dedicated at first to writing a Tiny BASIC – intentionally an alternative to Bill Gates and his irate letter to computer hobbyists. The call went out for people to get to work. April 1976
- Tom Pittman – wrote a popular 6800 Tiny BASIC, sold for $5 ea.



The first issue contained:

- Assembler listings of Tiny BASIC versions
- Complete user documentation
- Details for using a calculator chip and other hardware to obtain mathematical and floating-point functions
- Utilities like a 8-bit, binary-to-decimal conversion routine.

"..There is a viable alternative to the problems raised by Bill Gates in his irate letter to computer hobbyists concerning "ripping off" software. When software is free, or so inexpensive that it's easier to pay for it than to duplicate it, then it won't be "stolen." .."

"..those who wish to sell software for significant sums of money must realize that there is only one group that can practically be expected to pay for it:, the hardware manufacturers .."

-Jim C. Warren, April 1976 PCC Newsletter

TINY BASIC

Pretend you are 7 years old and don't care much about floating point arithmetic (what's that?), logarithms, sines, matrix inversion, nuclear reactor calculations and stuff like that.

And . . . your home computer is kinda small, not too much memory. Maybe its a MARK-8 or an ALTAIR 8800 with less than 4K bytes and a TV typewriter for input and output.

You would like to use it for homework, math recreations and games like NUMBER, STARS, TRAP, HURKLE, SNARK, BAGELS, . . .

Consider then, TINY BASIC

- Integer arithmetic only – 8 bits? 16 bits?
- 26 variables: A, B, C, D, . . ., Z
- The RND function – of course!
- Seven BASIC statement types
  - INPUT
  - PRINT
  - LET
  - GO TO
  - IF
  - GOSUB
  - RETURN
- Strings? OK in PRINT statements, not OK otherwise.

TINY BASIC GRAMMAR

The things in **bold face** stand for themselves. The names in lower case represent classes of things. '::=' is read 'is defined as'. The asterisk denotes zero or more occurances of the object to its immediate left. Parenthesis group objects. ε is the empty set. | denotes the alternative (the exclusive-or).

line::= number statement (cr) | statement (cr)
statement::= **PRINT** expr-list
IF expression relop expression **THEN** statement
GOTO expression
INPUT var-list
LET var = expression
GOSUB expression
RETURN
CLEAR
LIST
RUN
END
expr-list::= (string | expression) (, (string | expression) *)
var-list::= var (, var)*
expression::= (+ | − | ε) term ((+ | −) term)*
term::= factor ((* | /) factor)*
factor::= var | number | (expression)
var::= **A** | **B** | **C** ... | **Y** | **Z**
number::= digit digit*
digit::= **0** | **1** | **2** |... | **8** | **9**
relop::= < (> | = | ε) | >(< | = | ε) | =
A BREAK from the console will interrupt execution of the program.

A SECOND AND FINAL LETTER

"..Perhaps the present dilemma has resulted from a failure by many to realize that neither Micro-Soft nor anyone else can develop extensive software without a reasonable return on the huge investment in time that is necessary. .."

- BILL GATES General Partner, Micro-Soft

How to Read a Line of Micro-Soft

The program listing above from the screen display of an Ohio Scientific running MS BASIC. This program is written to PEEK the values in RAM starting from 0769h. MICROSOFT reserves the first three pages of memory for housekeeping duties so the text actually begins at location 0769h…Let's see what happens when you run this program.

How to Read a Line of Micro-Soft

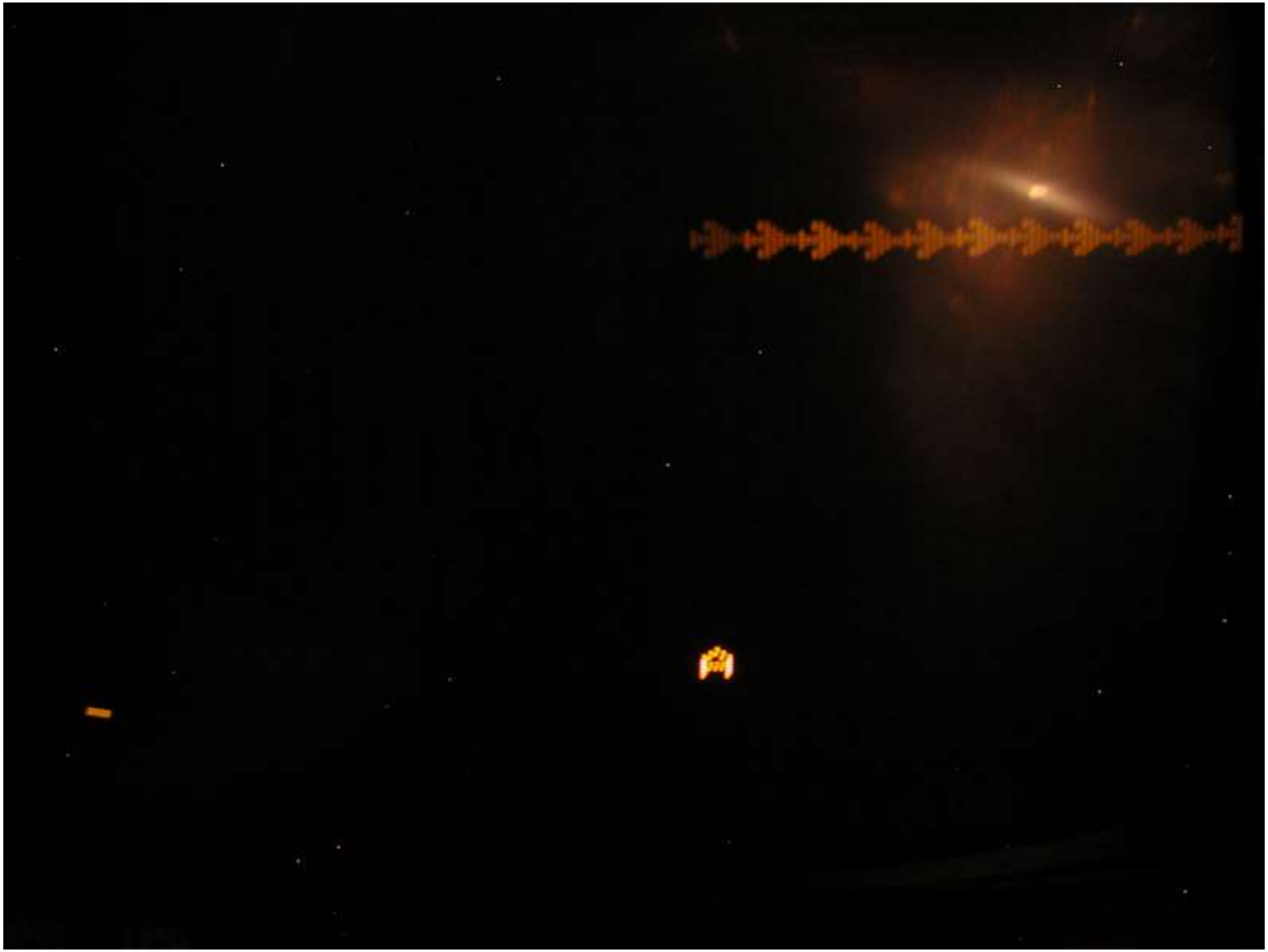
Microsoft Codes

PEEK ram to locate these …

CONT 152 LIST 153 NEW 155 NULL 145 RUN 137
CLEAR 154 DATA 131 DEF 149 DIM 133 END 128
FOR 129 GOTO 136 GOSUB 140 IF ... GOTO 138 **...** 136
IF ... THEN 138 ... 160 INPUT 132 LET 135 NEXT 130
ON...G0T0 144 ...136 ON...GOSUB 144...140 POKE 150
PRINT 151 READ 134 REM 142 RESTORE 139 RETURN 139
STOP 143 - 164 + 163 * 165 / 166
NOT 161 AND 168 OR 169 > 170 < 172
<> 172,170 > =170,171 < = 172,171 =171 ^ *167*
ABS 175 ATN 186 COS 183 EXP 182 FRE 177
LOG 181 PEEK 187 POS 178 RND 180 SGN 173
SIN 184 SPC 159 SQR 179 TAB 156 TAN 185
USR 176 ASC 191 CHR$ 192 LEFT$ 193 LEN 188
MID$, 195 RIGHT$ 164 STR$ 189 VAL 190

6502 Games with BASIC

- 6502 processor well-suited to BASIC graphics.
- Direct Memory Access (DMA)
- POKE and PEEK commands
- polled keyboard or joystick allows for simultaneous commands




```
130 IFPEEK(57100)=255 AND MM=IM THEN POKE MM, 32  :  MM = MM-W

140 IF MM<> IM THEN POKE MM,32 : MM = MM-W : POKE MM,MD

150 IF MM<53248 THEN MM=IM  : POKE MM,MD

160 IF PEEK(MM-W)=237THEN GOSUB300:POKE IM,65
```

Speed Suggestions

- Use variable names rather than actual values in display statements.
- Variable tables are arranged in the order that the variable are seen in the program.
- Avoid remark statements in the middle of display sections.

Speed Suggestions

- Use assumed branches and test as few things as possible.
- Display only what you have to.
- Cheat when it won't be seen.
- Keep explosion and gimmicks short and efficient.
- Don't rewrite more than you have to.

Memory Dump Using BASIC

BASIC program to pull RAM contents in octal format , and format to include leading zeros.

```
10 FOR I=0 TO 32767
20 K=PEEK(I)
30 L$ = OCT$(K)
40 IF LEN(L$)=1 THEN L$="00"+L$
50 IF LEN(L$)=2 THEN L$="0"+L$
80 PRINT L$
90 NEXT I
OK

RUN
363
303
242
013
257
043
343
302
etc.
```

NOTE: After you create the log file, you need to remove all carriage return line feed chars from the text file. You end up with a huge one line text file that can be read into the MITS Turnmon program.