OpenSE BASIC Quick Reference

Overview

OpenSE BASIC is an open source implementation of Sinclair BASIC including many improvements over the original, while retaining a high level of compatibility. Some of the highlights are:

- Overall fastest version of Sinclair BASIC - fully optimized for speed
- Fastest and most user friendly editor - with additional editing commands
- AY support including pseudo-interrupt driven sound
- ULAplus support including a default palette and new commands
- 8-bit character set support including printing characters 24-31
- Direct machine code calls
- BASIC access to LDIR
- Hex and Octal number entry
- Decimal to Hex string conversion
- Intelligent error trapping - OK and STOP are not errors
- More room for BASIC programs and line numbers up to 16383
- Improved SCREEN$ handling for UDGs and 8-bit character sets
- Improved floating-point library - faster and more accurate
- Remains compatible with the majority of Spectrum software and hardware
- Ability to use reserved words as variable names during tokenization

New Command Summary

You will find here a brief description of the 11 new commands in OpenSE BASIC. A single letter is used to represent a numeric expression. Check the given section for a full explanation of the syntax offered.

COPY	A command used to call a machine code routine without returning a value in BC. Defaults to 0 which has no effect.
DELETE f, l	You would use this command to delete a block of program lines, where f is the first line number of the block and l is the last. If the value of f is greater than l then the error message “Integer out of range” is displayed on-screen.
DIR b	A command used to toggle 8-bit character set support on and off where b is 7 or 8. By default 7-bit character sets are used and characters above 127 are displayed as block graphics, UDGs, and tokens.
EDIT l	Use this command to display line l in the input line and activate the line-editor. See The Editor
ERASE	Use this command to reset the default palette. See ULAplus Support
FORMAT p	A command used to set the permanent attribute. See ULAplus Support
MOVE s, d, l	A command to enable access to the Z80’s LDIR (block copy) instruction. A total of l bytes are copied from the source address s to the destination address d. Use with extreme caution as overwriting the system variables or the BASIC program will probably cause a crash.
ON ERR ...	Use ON ERR GOTO n to go to line ‘n’ when an error is trapped. Use ON ERR CONTINUE to continue the program without displaying the error message. Use ON ERR STOP to display the error message. See Error Trapping
PALETTE ...	A command used to set the colours in computers fitted with the ULAplus display chip. See ULAplus Support
RENUM ...	A command used to renumber the current program. See RENUMbering
SOUND ...	A command used to produce sound effects and three channel tunes in computers fitted with an AY-3-8912 sound chip. See Programmable Sound Generator

New Function Summary

The following new function symbols work in much the same way as the BIN function.

&	Used to enter 16-bit hexadecimal positive integers (in upper or lower case), for example 10 PRINT &FFFF
\	Used to enter 16-bit octal positive integers, for example 10 PRINT \177777
˜	Used to convert 16-bit decimal positive integers to a hexadecimal string, for example 10 PRINT ˜65535

A much larger number of functions can be added using the DEF FN command.

Getting Started

OpenSE BASIC is supplied as a 16K ROM file for use with emulators or real machines as a replacement ROM or Interface II cartridge. Please refer to you emulator for instructions on how to use alternate ROMs or Interface II cartridges. OpenSE BASIC is not designed to be used as a replacement for 48 BASIC in computers with 32K or 64K ROMs. In a 32K-ROM computer you should put the original Sinclair ROM in the other 16K. In a 64K-ROM computer you should put OpenSE BASIC and the original Sinclair ROM in one bank of 32K and the UK Sinclair 128 ROMs in the other bank. This will ensure you are able to run the widest range of software.

On a 32K ROM computer:
```
   OUT 32765, 0  = select ROM 0
   OUT 32765, 16 = select ROM 1
```

On a 64K ROM computer:
```
   OUT 8189, 0: OUT 32765, 0  = select ROM 0
   OUT 8189, 0: OUT 32765, 16 = select ROM 1
   OUT 8189, 4: OUT 32765, 0  = select ROM 2
   OUT 8189, 4: OUT 32765, 16 = select ROM 3
```

NOTE: The computer may crash part way through changing ROMs if either OUT instructions causes a ROM other than a version of BASIC to be paged in. When OpenSE BASIC is used on a 128K machine it is effectively in ‘USR 0’ mode.

The Keyboard

When you switch on your computer you will be greeted by the standard copyright message. Try typing a few characters on the keyboard and you will notice that the keys are not producing their usual keywords; instead you see just single characters. From now on, you will have to type out each command in full rather than use the infamous keywords; a facility which transforms your computer keyboard into something approaching that of a ‘normal’ computer.

Although removing the keyword system has many advantages, the change does have a drawback. Certain commands such as ‘PRINT’ could be typed in just by pressing the ‘P’ key, whereas now you will have to type out ‘P’, ‘R’, ‘I’, ‘N’, and ‘T’; for that reason, OpenSE BASIC allows you to abbreviate many of the keywords.

Here follows a complete list of keywords and their new abbreviations; you can assume that keywords omitted from the list cannot be abbreviated and therefore must be typed out in full. Also note that an abbreviated keyword must finish with a full stop; for example, the abbreviation for ‘CONTINUE’ is ‘CON.’.

A.TTR	ED.IT	LO.AD	RA.NDOMIZE
BE.EP	ER.ASE	ME.RGE	RE.AD
B.IN	E.XP	M.OVE	REN.UM
BO.RDER	FL.ASH	NE.XT	RES.TORE
BR.IGHT	F.ORMAT	N.OT	RET.URN
CH.R$	GOS.UB	ON.ERR	R.ND
CI.RCLE	G.OTO	OP.EN#	SA.VE
CLE.AR	I.NK	OV.ER	S.CREEN$
CL.OSE#	INKE.Y$	PA.PER	SO.UND
C.ODE	INP.UT	PAL.ETTE	ST.R$
CON.TINUE	INV.ERSE	PAU.SE	T.AB
DA.TA	L.EN	PE.EK	TH.EN
D.EFFN	LI.NE	PL.OT	U.SR
DEL.ETE	LL.IST	P.OINT	V.AL$
DR.AW	LP.RINT	PR.INT	VE.RIFY

[S]+Q = LOAD	[S]+W = CODE	[S]+E = RUN	[I]+V = SIN
[I]+W = COS	[I]+X = TAN	[I]+Y = ASN	[I]+Z = ACS

If you are going to be typing commands such as ‘GO TO’ and ‘ON ERR’, you do not have to remember to insert the spaces. The commonly used keywords LOAD, CODE, and RUN are available as Symbol Shift and ‘Q’, ‘W’, and ‘E’.

```
+------+------+------+------+------+------+------+------+------+------+
| EDIT | CAPS | TRUE | INV. |  <-  |  \/  |  /\  |  ->  | INS. | DEL. |
| 1 :. | 2 .: | 3 .. | 4 :’ | 5 :  | 6 .’ | 7 .  | 8 :: | 9    | 0    |
| !    | @    | #    | $    | %    | &    | ‘    | (    | )    | _    |
+--+---+--+---+--+---+--+---+--+---+--+---+--+---+--+---+--+---+--+---+--+
   |      | COS  |      |      |      | ASN  |      |      |      |      |
   | Q    | W    | E    | R    | T    | Y    | U    | I    | O    | P    |
   | LOAD | CODE | RUN  | <    | >    | [    | ]    | ©    | ;    | “    |
   +--+---+--+---+--+---+--+---+--+---+--+---+--+---+--+---+--+---+--+---+--+
      |      |      |      |      |      |      |      |      |      |      |
      | A    | S    | D    | F    | G    | H    | J    | K    | L    | ENT. |
      | ~    | |    | \    | {    | }    | ^    | -    | +    | =    |      |
  +---+--+---+--+---+--+---+--+---+--+---+--+---+--+---+--+---+--+---+--+---+
  |      | ACS  | TAN  |      | SIN  |      |      |      |      | BRK. |
  | CAPS | Z    | X    | C    | V    | B    | N    | M    | SYM. | SPA. |
  | SHF. | :    | £    | ?    | /    | *    | ,    | .    | SHF. |      |
  +------+------+------+------+------+------+------+------+------+------+
```

The Editor

The line editing capabilities have been greatly enhanced by OpenSE BASIC. The cursor shows the current mode:

[.] CAPS OFF
['] CAPS ON
[I] INSERT

You will notice that [E] (EXTENDED mode) is missing. You can still use Caps Shift, Symbol Shift and the number keys to insert control codes, but all symbols are now accessed with Symbol Shift and a key, for instance Symbol Shift 'I' produces the copyright symbol. Holding down Caps Shift in caps mode will produce a lower case letter and you can now cursor up and down in an EDIT line.

As well as using Caps Shift '1' to edit the current line it is possible to edit any line in the program by using the 'EDIT' command. This command is followed by a numeric experssion that shows which line is to be edited. If the required line does not exist, then the next program line is used.

In the original ROM it was possible to use keyword names as variable names. This is still supported providing you enter the line in [I] mode, and the variable name contains at least one lowercase character, but remember that variable names are case insensitive. In this mode keywords must be uppercase or they will be ignored by the tokenizer.

The valid line range has been increased from 1-9999 to 1-16383. Programs using line numbers above 9999 will also work with the original ROM.

ULAplus Support

OpenSE BASIC sets a default 64-colour mode palette for ULAplus, although this mode is off by default.

This program tests if ULAplus hardware is present:

```
10 OUT 48955,0: OUT 65339,0: PAUSE 1: LET i = IN 65339
20 IF i = 0 THEN PRINT “ULAplus detected”
```

To switch on 64-colour RGB mode:

```
PALETTE 64,1
```

To switch on 64-colour HSL mode:

```
PALETTE 64,2
```

To switch on 64-colour CMYK mode:

```
PALETTE 64,3
```

To switch off 64-colour mode:

```
PALETTE 64,0
```

The following program will display the default palette.

```
10 FOR x=0 TO 255
20 FORMAT x
30 PRINT CHR$ 131;
40 NEXT x
```

The palette is designed to work well with existing software, and to be easy to use from BASIC. The fourth CLUT (3) is an approximation of a grey scale and has the same colours for PEN and PAPER enabling you to combine any of the colours in a character cell.

To restore the default palette:

```
ERASE
```

To set one of the 64 colours:

```
PALETTE c, BIN gggrrrbb
```

where c is a colour (0-63), and g, r, and b are colour bits for green, red, and blue. For example, bright red is BIN 00011100. The value of c corresponds to the colour values 0-7 in each colour look-up table (CLUT)

```
  0-7  non-bright PEN
 8-15  non-bright PAPER (BORDER in lo-res)
16-23  bright PEN
24-31  bright PAPER (BORDER in hi-res)
32-39  flash PEN
40-47  flash PAPER
48-55  flash/bright PEN
56-63  flash/bright PAPER
```

You may prefer to use hex (&)

```
&00-&07  non-bright PEN
&08-&0f  non-bright PAPER (BORDER in lo-res)
&10-&17  bright PEN
&18-&1f  bright PAPER (BORDER in hi-res)
&20-&27  flash PEN
&28-&2f  flash PAPER
&30-&37  flash/bright PEN
&38-&3f  flash/bright PAPER
```

or octal (\)

```
\00-\07  non-bright PEN
\10-\17  non-bright PAPER (BORDER in lo-res)
\20-\27  bright PEN
\30-\37  bright PAPER (BORDER in hi-res)
\40-\47  flash PEN
\50-\57  flash PAPER
\60-\67  flash/bright PEN
\70-\77  flash/bright PAPER
```

You can set the permanent attributes with a single command using FORMAT n. Using the octal (\) you can set the CLUT, PAPER, and PEN as follows:

```
FORMAT \cpi
```

where c is the CLUT (0-3), p is the PAPER selection (0-7), and i is the PEN selection (0-7).

You may want to set the PAPER colours in the first three CLUTS (0-2) to be the same. While this gives only eight background colours, it enables you to use 24 foreground colours without worrying about what the background colour is. The following command will prevent the background colour being changed when you PRINT or PLOT to the screen:

```
PAPER 8: BRIGHT 8: FLASH
```

When creating your own palettes, you can also use PEN 8: BRIGHT 8: FLASH 8 to set up a palette with 32 PAPERs and 8 PENs if you prefer.

For further information, see: http://sites.google.com/site/ulaplus/

Programmable Sound Generator

The most requested command to add to SE BASIC was PLAY. But there was no room. Instead, the AY is supported by the SOUND command, which enables you to send a set of register pairs to the AY chip. Unlike the PLAY command, the SOUND command will keep playing until an error, or the end of the program, are encountered. When an error report is printed, the AY is silenced. The AY is supported simultaneously on the Spectrum+ 128K and the TS2068 ports. For example, to play the note of A for one second on a 50Hz machine:

```
  10 SOUND 0,124;1,0;8,13;7,62
  20 PAUSE 50
```

The SOUND command allows you to compose music in harmony, with three channels instead of BEEP's one at your disposal. It can also produces some interesting sound effects to add to your programs.

The SOUND command is followed by pairs of numbers, the pairs separates by semicolons and the individual numbers within the pairs by commas. You can include up to 15 pairs of numbers in each SOUND statement. In each pair, the first designates one of fifteen registers—storage locations—within the special sound/music synthesizer chip. These registers control pitch, duration, and volume of the sound being produced. The following examples are from the Timex Sinclair TS2068 User Manual:

Gunshots

```
  10 SOUND 6,15;7,7;8,16;9,16;10,16;12,16;13,0
  20 PAUSE 50
  30 GO TO 10
```

Explosion

```
  10 SOUND 6,6;7,7;8,16;9,16;10,16;12,56;13,8
  20 PAUSE 75
  30 SOUND 8,0;9,0;10,0
```

Whistling Bomb

```
  10 SOUND 7,62;8,15
  20 FOR I=50 TO 100
  30 SOUND 0,I: PAUSE 2.5
  40 NEXT I
```

AY-3-891x Note Tables

Note that the discrepancies in the tables in the TS2068 User Manual and the TS2068 Intermediate/Advanced Guide are due to the former being calculated against a 1.75 Mhz chip with truncation instead of rounding, and the latter being calculated on the TS2068 ROM which contains floating point errors that were present in the original ROM.

1.75000 Mhz
(TC2068, Spectrum 16K/48K with external AY)

		Ideal		Tune Registers		Actual
Note	Octave	Frequency	Period	Coarse	Fine	Frequency
C	1	32.703	3344	13	16	32.708
C♯	1	34.648	3157	12	85	34.645
D	1	36.708	2980	11	164	36.703
D♯	1	38.891	2812	10	252	38.896
E	1	41.203	2655	10	95	41.196
F	1	43.654	2506	9	202	43.645
F♯	1	46.249	2365	9	61	46.247
G	1	48.999	2232	8	184	49.003
G♯	1	51.913	2107	8	59	51.910

A	1	55.000	1989	7	197	54.990
A#	1	58.270	1877	7	85	58.271
B	1	61.735	1772	6	236	61.724
C	2	65.406	1672	6	136	65.416
C#	2	69.296	1578	6	42	69.312
D	2	73.416	1490	5	210	73.406
D#	2	77.782	1406	5	126	77.792
E	2	82.407	1327	5	47	82.423
F	2	87.307	1253	4	229	87.291
F#	2	92.499	1182	4	158	92.534
G	2	97.999	1116	4	92	98.006
G#	2	103.826	1053	4	29	103.870
A	2	110.000	994	3	226	110.035
A#	2	116.541	939	3	171	116.480
B	2	123.471	886	3	118	123.448
C	3	130.813	836	3	68	130.831
C#	3	138.591	789	3	21	138.625
D	3	146.832	745	2	233	146.812
D#	3	155.563	703	2	191	155.583
E	3	164.814	664	2	152	164.721
F	3	174.614	626	2	114	174.720
F#	3	184.997	591	2	79	185.068
G	3	195.998	558	2	46	196.013
G#	3	207.652	527	2	15	207.543
A	3	220.000	497	1	241	220.070
A#	3	233.082	469	1	213	233.209
B	3	246.942	443	1	187	246.896
C	4	261.626	418	1	162	261.663
C#	4	277.183	395	1	139	276.899
D	4	293.665	372	1	116	294.019
D#	4	311.127	352	1	96	310.724
E	4	329.628	332	1	76	329.443
F	4	349.228	313	1	57	349.441
F#	4	369.994	296	1	40	369.510
G	4	391.995	279	1	23	392.025
G#	4	415.305	263	1	7	415.875
A	4	440.000	249	0	249	439.257
A#	4	466.164	235	0	235	465.426
B	4	493.883	221	0	221	494.910
C	5	523.251	209	0	209	523.325
C#	5	554.365	197	0	197	555.203
D	5	587.330	186	0	186	588.038
D#	5	622.254	176	0	176	621.449
E	5	659.255	166	0	166	658.886
F	5	698.456	157	0	157	696.656
F#	5	739.989	148	0	148	739.020
G	5	783.991	140	0	140	781.250
G#	5	830.609	132	0	132	828.598
A	5	880.000	124	0	124	882.056
A#	5	932.328	117	0	117	934.829
B	5	987.767	111	0	111	985.360
C	6	1046.502	105	0	105	1041.667
C#	6	1108.731	99	0	99	1104.798
D	6	1174.659	93	0	93	1176.075
D#	6	1244.508	88	0	88	1242.898
E	6	1318.510	83	0	83	1317.771
F	6	1396.913	78	0	78	1402.244
F#	6	1479.978	74	0	74	1478.041

G	6	1567.982	70	0	70	1562.500
G#	6	1661.219	66	0	66	1657.197
A	6	1760.000	62	0	62	1764.113
A#	6	1864.655	59	0	59	1853.814
B	6	1975.533	55	0	55	1988.636
C	7	2093.005	52	0	52	2103.365
C#	7	2217.461	49	0	49	2232.143
D	7	2349.318	47	0	47	2327.128
D#	7	2489.016	44	0	44	2485.795
E	7	2637.020	41	0	41	2667.683
F	7	2793.826	39	0	39	2804.487
F#	7	2959.955	37	0	37	2956.081
G	7	3135.963	35	0	35	3125.000
G#	7	3322.438	33	0	33	3314.394
A	7	3520.000	31	0	31	3528.226
A#	7	3729.310	29	0	29	3771.552
B	7	3951.066	28	0	28	3906.250
C	8	4186.009	26	0	26	4206.731
C#	8	4434.922	25	0	25	4375.000
D	8	4698.636	23	0	23	4755.435
D#	8	4978.032	22	0	22	4971.591
E	8	5274.041	21	0	21	5208.333
F	8	5587.652	20	0	20	5468.750
F#	8	5919.911	18	0	18	6076.389
G	8	6271.927	17	0	17	6433.824
G#	8	6644.875	16	0	16	6835.938
A	8	7040.000	16	0	16	6835.938
A#	8	7458.620	15	0	15	7291.667
B	8	7902.133	14	0	14	7812.500

1.76400 Mhz
(TS2068)

		Ideal		Tune Registers		Actual
Note	Octave	Frequency	Period	Coarse	Fine	Frequency
C	1	32.703	3371	13	43	32.705
C#	1	34.648	3182	12	110	34.648
D	1	36.708	3003	11	187	36.713
D#	1	38.891	2835	11	19	38.889
E	1	41.203	2676	10	116	41.200
F	1	43.654	2526	9	222	43.646
F#	1	46.249	2384	9	80	46.246
G	1	48.999	2250	8	202	49.000
G#	1	51.913	2124	8	76	51.907
A	1	55.000	2005	7	213	54.988
A#	1	58.270	1892	7	100	58.272
B	1	61.735	1786	6	250	61.730
C	2	65.406	1686	6	150	65.391
C#	2	69.296	1591	6	55	69.296
D	2	73.416	1502	5	222	73.402
D#	2	77.782	1417	5	137	77.805
E	2	82.407	1338	5	58	82.399
F	2	87.307	1263	4	239	87.292
F#	2	92.499	1192	4	168	92.492
G	2	97.999	1125	4	101	98.000
G#	2	103.826	1062	4	38	103.814
A	2	110.000	1002	3	234	110.030
A#	2	116.541	946	3	178	116.543
B	2	123.471	893	3	125	123.460

C	3	130.813	843	3	75	130.783
C#	3	138.591	796	3	28	138.505
D	3	146.832	751	2	239	146.804
D#	3	155.563	709	2	197	155.501
E	3	164.814	669	2	157	164.798
F	3	174.614	631	2	119	174.723
F#	3	184.997	596	2	84	184.983
G	3	195.998	563	2	51	195.826
G#	3	207.652	531	2	19	207.627
A	3	220.000	501	1	245	220.060
A#	3	233.082	473	1	217	233.087
B	3	246.942	446	1	190	247.197
C	4	261.626	421	1	165	261.876
C#	4	277.183	398	1	142	277.010
D	4	293.665	375	1	119	294.000
D#	4	311.127	354	1	98	311.441
E	4	329.628	334	1	78	330.090
F	4	349.228	316	1	60	348.892
F#	4	369.994	298	1	42	369.966
G	4	391.995	281	1	25	392.349
G#	4	415.305	265	1	9	416.038
A	4	440.000	251	0	251	439.243
A#	4	466.164	237	0	237	465.190
B	4	493.883	223	0	223	494.395
C	5	523.251	211	0	211	522.512
C#	5	554.365	199	0	199	554.020
D	5	587.330	188	0	188	586.436
D#	5	622.254	177	0	177	622.881
E	5	659.255	167	0	167	660.180
F	5	698.456	158	0	158	697.785
F#	5	739.989	149	0	149	739.933
G	5	783.991	141	0	141	781.915
G#	5	830.609	133	0	133	828.947
A	5	880.000	125	0	125	882.000
A#	5	932.328	118	0	118	934.322
B	5	987.767	112	0	112	984.375
C	6	1046.502	105	0	105	1050.000
C#	6	1108.731	99	0	99	1113.636
D	6	1174.659	94	0	94	1172.872
D#	6	1244.508	89	0	89	1238.764
E	6	1318.510	84	0	84	1312.500
F	6	1396.913	79	0	79	1395.570
F#	6	1479.978	74	0	74	1489.865
G	6	1567.982	70	0	70	1575.000
G#	6	1661.219	66	0	66	1670.455
A	6	1760.000	63	0	63	1750.000
A#	6	1864.655	59	0	59	1868.644
B	6	1975.533	56	0	56	1968.750
C	7	2093.005	53	0	53	2080.189
C#	7	2217.461	50	0	50	2205.000
D	7	2349.318	47	0	47	2345.745
D#	7	2489.016	44	0	44	2505.682
E	7	2637.020	42	0	42	2625.000
F	7	2793.826	39	0	39	2826.923
F#	7	2959.955	37	0	37	2979.730
G	7	3135.963	35	0	35	3150.000
G#	7	3322.438	33	0	33	3340.909
A	7	3520.000	31	0	31	3556.452

A#	7	3729.310	30	0	30	3675.000
B	7	3951.066	28	0	28	3937.500
C	8	4186.009	26	0	26	4240.385
C#	8	4434.922	25	0	25	4410.000
D	8	4698.636	23	0	23	4793.478
D#	8	4978.032	22	0	22	5011.364
E	8	5274.041	21	0	21	5250.000
F	8	5587.652	20	0	20	5512.500
F#	8	5919.911	19	0	19	5802.632
G	8	6271.927	18	0	18	6125.000
G#	8	6644.875	17	0	17	6485.294
A	8	7040.000	16	0	16	6890.625
A#	8	7458.620	15	0	15	7350.000
B	8	7902.133	14	0	14	7875.000

1.77345 Mhz
(Spectrum 128K)

		Ideal		Tune Registers		Actual
Note	Octave	Frequency	Period	Coarse	Fine	Frequency
C	1	32.703	3389	13	61	32.706
C#	1	34.648	3199	12	127	34.649
D	1	36.708	3020	11	204	36.702
D#	1	38.891	2850	11	34	38.891
E	1	41.203	2690	10	130	41.205
F	1	43.654	2539	9	235	43.655
F#	1	46.249	2397	9	93	46.241
G	1	48.999	2262	8	214	49.001
G#	1	51.913	2135	8	87	51.916
A	1	55.000	2015	7	223	55.008
A#	1	58.270	1902	7	110	58.276
B	1	61.735	1795	7	3	61.750
C	2	65.406	1695	6	159	65.393
C#	2	69.296	1600	6	64	69.275
D	2	73.416	1510	5	230	73.404
D#	2	77.782	1425	5	145	77.783
E	2	82.407	1345	5	65	82.409
F	2	87.307	1270	4	246	87.276
F#	2	92.499	1198	4	174	92.521
G	2	97.999	1131	4	107	98.002
G#	2	103.826	1068	4	44	103.783
A	2	110.000	1008	3	240	109.961
A#	2	116.541	951	3	183	116.552
B	2	123.471	898	3	130	123.431
C	3	130.813	847	3	79	130.863
C#	3	138.591	800	3	32	138.551
D	3	146.832	755	2	243	146.809
D#	3	155.563	713	2	201	155.457
E	3	164.814	673	2	161	164.696
F	3	174.614	635	2	123	174.552
F#	3	184.997	599	2	87	185.043
G	3	195.998	566	2	54	195.831
G#	3	207.652	534	2	22	207.567
A	3	220.000	504	1	248	219.922
A#	3	233.082	476	1	220	232.858
B	3	246.942	449	1	193	246.861
C	4	261.626	424	1	168	261.417
C#	4	277.183	400	1	144	277.102
D	4	293.665	377	1	121	294.007

D#	4	311.127	356	1	100	311.350
E	4	329.628	336	1	80	329.883
F	4	349.228	317	1	61	349.655
F#	4	369.994	300	1	44	369.469
G	4	391.995	283	1	27	391.663
G#	4	415.305	267	1	11	415.133
A	4	440.000	252	0	252	439.844
A#	4	466.164	238	0	238	465.717
B	4	493.883	224	0	224	494.824
C	5	523.251	212	0	212	522.833
C#	5	554.365	200	0	200	554.203
D	5	587.330	189	0	189	586.458
D#	5	622.254	178	0	178	622.700
E	5	659.255	168	0	168	659.766
F	5	698.456	159	0	159	697.111
F#	5	739.989	150	0	150	738.938
G	5	783.991	141	0	141	786.104
G#	5	830.609	133	0	133	833.388
A	5	880.000	126	0	126	879.688
A#	5	932.328	119	0	119	931.434
B	5	987.767	112	0	112	989.648
C	6	1046.502	106	0	106	1045.666
C#	6	1108.731	100	0	100	1108.406
D	6	1174.659	94	0	94	1179.156
D#	6	1244.508	89	0	89	1245.400
E	6	1318.510	84	0	84	1319.531
F	6	1396.913	79	0	79	1403.046
F#	6	1479.978	75	0	75	1477.875
G	6	1567.982	71	0	71	1561.136
G#	6	1661.219	67	0	67	1654.338
A	6	1760.000	63	0	63	1759.375
A#	6	1864.655	59	0	59	1878.655
B	6	1975.533	56	0	56	1979.297
C	7	2093.005	53	0	53	2091.333
C#	7	2217.461	50	0	50	2216.813
D	7	2349.318	47	0	47	2358.311
D#	7	2489.016	45	0	45	2463.125
E	7	2637.020	42	0	42	2639.063
F	7	2793.826	40	0	40	2771.016
F#	7	2959.955	37	0	37	2995.693
G	7	3135.963	35	0	35	3166.875
G#	7	3322.438	33	0	33	3358.807
A	7	3520.000	31	0	31	3575.504
A#	7	3729.310	30	0	30	3694.688
B	7	3951.066	28	0	28	3958.594
C	8	4186.009	26	0	26	4263.101
C#	8	4434.922	25	0	25	4433.625
D	8	4698.636	24	0	24	4618.359
D#	8	4978.032	22	0	22	5038.210
E	8	5274.041	21	0	21	5278.125
F	8	5587.652	20	0	20	5542.031
F#	8	5919.911	19	0	19	5833.717
G	8	6271.927	18	0	18	6157.813
G#	8	6644.875	17	0	17	6520.037
A	8	7040.000	16	0	16	6927.539
A#	8	7458.620	15	0	15	7389.375
B	8	7902.133	14	0	14	7917.188

Advanced Programming

Programs written in OpenSE BASIC will run on the original unmodified ROM providing you restrict yourself to the original commands, although you can safely use line numbers beyond 9999. However, you may want to determine if the SE BASIC ROM is present, either to branch or to inform the user that their ROM is not supported. The following program determines if SE BASIC is present:

```
10 LET r$ = CHR$ (PEEK 43) + CHR$ (PEEK 44)
20 IF r$ = “SE” THEN PRINT “SE BASIC detected”
```

To determine the version number:

```
PRINT CHR$ (PEEK 37) + “.” + CHR$ (PEEK 38) + CHR$ (PEEK 39)
```

Versions prior to 3.00 are not open source.

IF ... ELSE

Although OpenSE BASIC does not include an ELSE command, IF ... ELSE can be constructed as follows:

```
10 IF a = true THEN GO TO lineA
20 IF b = true THEN GO TO lineB
30 IF c = true THEN GO TO lineC
40 GO TO lineD
```

WHILE ... DO

In this kind of loop the test is carried out first. For example:

```
10 IF i =< 100 THEN GO TO 40
20 INPUT “Enter a number above 100: “; i
30 GO TO 10
40 REM END
```

REPEAT ... UNTIL

In this kind of loop the commands are carried out first. For example:

```
10 INPUT “Enter a number above 100: “; i
20 IF i =< 100 THEN GO TO 10
30 REM END
```

NAMED PROCEDURES

Although OpenSE BASIC does not allow you to create named procedures, you can use definitions to make your programs more readable. For example:

```
10 LET HISCORE = 1000
20 GO SUB HISCORE
1000 REM PROC: HISCORE
```

NOTE: If you RENUMber your program you will have to manually change your definitions. Therefore you should use the REM statement to label your procedures.

BOOLEAN LOGIC

OpenSE BASIC provides three boolean operators, AND, OR, and NOT. The result of testing these operators is always 1 (true) or 0 (false). To make programs easier to read it may be worth defining variables for these results as follows:

```
10 LET true = 1 : LET false = 0
```

For example:

```
100 IF a AND b = true THEN GO SUB procedure
```

DPOKE

The double POKE command can be implemented as follows:

```
10 POKE address, number - INT(number/256)*256
20 POKE address + 1, INT(number/256)
```

FREE ()

This will return the same result as DEF FN F()=65536-USR 7962 does on the original ROM:

```
DEF FN F()=(PEEK 23731*256)+PEEK 23730-((PEEK 23654*256)+PEEK 23653)-110
```

Error Trapping

ON ERR can be used to prevent the user BREAKing into a program, or to trap errors. Note, OK and STOP are not treated as errors, but STOP in INPUT is. The following commands are accepted:

```
   ON ERR GO TO n
   ON ERR CONTINUE
   ON ERR STOP
```

These statements allow the progammer to disable automatic program termination upon encountering an error condition. The ON ERR GOTO line number allows the programer to cause the transfer to the specified line number to handle the encountered error. The ON ERR CONTINUE statement causes the program to resume execution at the statement in which the error originally occured. The ON ERR STOP command disables this feature causing the program to report errors and terminate in the usual manner.

The errors 'OK' and 'STOP' are not treated as errors and the program will terminate if they are encountered. 'STOP in INPUT' is. ON ERR CONTINUE has the side effect of preventing a user accidentally BREAKing into a program. However, if the program does not encounter an 'OK' or 'STOP' error, it is possible to get stuck in an infinite loop. The only way to BREAK out of this loop is by triggering a warm restart using the NMI button. To completely prevent the user breaking into the program the NMI BREAK can be disabled by setting the NMIADD system variable to zero.

Renumbering

The following commands change the line numbers of your program:

```
   RENUM
```

This instruction will renumber all your program lines in steps of ten, starting with the first line as 10.

```
   RENUM l
```

makes number ‘l’ the first new line number

```
   RENUM l,s
```

uses numbers in whatever step ‘s’ you instruct.

When RENUMbering, all your instructions like GO TO, GO SUB, RESTORE, RUN, LINE, ON ERR GO TO etc. are dealt with, but any expressions such as GO TO VAL “100”, EDIT 100, DELETE 100,100, and RENUM 100,100,100,100 will be ignored.

Keyword Reference

This reference contains full descriptions of all the keywords available in OpenSE BASIC. Each entry includes:

- abbreviation
- class
- purpose
- use
- format

Keywords fall into one or more of the following classes:

- **Command**
  A keyword which causes an action to occur and can be used to form a direct command. It is carried out on being entered. Examples – RUN, LOAD
- **Statement**
  A keyword which causes an action to occur and which can be used in a program line. It is carried out only when the program is run. Examples – DRAW, INPUT
- **Function**
  A keyword which produces a value of some kind. It forms part of a command or statement. Examples – RND, INT.
- **Logical Operator**
  A keyword which is used to express logic in a statement or command. It can determine or change the truth of certain conditions. OpenSE BASIC has three logical operator keywords – AND, OR and NOT.

Numbers are stored to an accuracy of 9 or 10 digits. The number handling range is about 1038 to 4 * 10-39. Three types of variables are accepted:

- **Number**
  Any length, starting with a letter. Spaces are ignored and all letters are converted to lower-case letters. Capital and lower-case letters are not distinguished. You can use keywords as variables, only if you enter keywords in capitals and variables in lower or mixed case and enter G mode before entering a line.
- **String**
  Any single letter followed by $. Capital and lower-case letters are not distinguished.
- **Array**
  For array variables and subscripts, see DIM.

The following abbreviations are used in the keyword descriptions:

- num-const – a numeric constant, such 24.5.
- num-var – a variable that may contain a numeric constant, such as sum.
- num-expr – any valid combination of numeric constants, variables and keywords that gives a number, such as RND*7.
- int-num-const, int-num-var, int-num-expr – a numeric constant, variable or expression whose value is rounded to the nearest integer.
- string-const – a string constant or string, such as "OpenSE BASIC".
- string-var – a variable that may contain a string, such as a$.
- string-expr – any valid combination of string constants, variables and keywords that gives a string, such as a$(6 TO 8).
- letter – any capital or lower-case letter.
- letter$ – any capital or lower-case letter followed by $.
- cond – a condition or sub-condition within a condition, such as x=10 AND t<10.
- statement – any OpenSE BASIC statement that is valid when used with another statement, such as PRINT PEN 2;x.
- prompt – [string-const][(String-expr)][AT int-num-expr,int-num-expr][statement][:][,][']
- [ ] – an optional item that may be repeated.

The following signs are used in OpenSE BASIC:

- $ string variable.
- ‘ begins new line.
- (open bracket.
-) close bracket.
- <= is less than or equal to.
- <> is not equal to.
- >= is greater than or equal to.
- < is less than.
- > is greater than.
- ^ raise to the power.
- - subtraction or negative.
- + addition, positive, string concatenation .
- = is equal to.
- : separates statements in the program line.
- / division.
- * multiplication.
- . decimal point.
- ; displays at next column, separates statements within a program statement.
- “ open and close string.
- , displays at column 0 or 16, separates values following keywords
- & converts the following four characters from a hex string to decimal
- ~ converts the following positive integer into a hex string
- \ converts the following positive integer from octal to decimal

Keywords

ABS
ABSolute value
Function
ABS num-const ABS num-var ABS (num-expr)

ACS
Arc CoSine
Function
ACS num-const ACS num-var ACS (num-expr)

AND
Logical Operator/Function
cond AND cond num-expr AND num-expr string-expr AND num-expr

ASN
Arc SiNe
Function
ASN num-const ASN num-var ASN (num-expr)

AT
See INPUT, LPRINT, PRINT.

ATN
Arc TaNgent
Function
ATN num-const ATN num-var ATN (num-expr)

ATTR
ATTRibutes
Function
ATTR (num-expr,num-expr)

BEEP
Statement/Command
BEEP num-expr,num-expr

BIN
BINary number
Function
BIN [0][1]

BORDER
Statement/Command
BORDER int-num-expr

BRIGHT
Statement/Command
BRIGHT int-num-expr[;]

CHR$
CHaRacter (string)
CHR$ int-num-const[;][+] CHR$ int-num-var[;][+]CHR$ (int-num-expr)[;][+]

CIRCLE
Statement/Command
CIRCLE [statement;]int-num-expr,int-num-expr,int-num-expr

CLEAR
Statement/Command
CLEAR [num-expr]

CLOSE
Statement/Command
CLOSE #int-num-expr

CLS
Statement/Command
CLS

CODE
Function
CODE string-const CODE string-var CODE (string-expr)

CONTINUE
Command
CONTINUE

COPY
Statement/Command
COPY int-num-const COPY int-num-var COPY (int-num-expr)

COS
COSine
Function
COS num-const COS num-var COS (num-expr)

DATA
Statement
DATA num-expr[,num-expr][,string-expr] DATA string-expr[,num-expr][,string-expr]

DEF FN
DEFine FuNction
Statement
DEF FN letter([letter][,letter]) = num-expr DEF FN letter$([letter$][letter][,letter][,letter$]) = string-expr

DELETE
Command
DELETE int-num-const,int-num-const DELETE int-num-var,int-num-var DELETE (num-expr),(num-expr)

DIM
DIMension array
Statement
DIM letter (num-expr[,num-expr]) DIM letter$ (num-expr[,num-expr])

DIR
DIsplay Rendering
Statement/Command
DIR int-num-const

DRAW
Statement/Command
DRAW [statement;]int-num-expr,int-num-expr[,int-num-expr]

EDIT
Command
EDIT int-num-const EDIT int-num-var EDIT (int-num-expr)

ERASE
Statement/Command
ERASE

EXP
EXPonent
Function
EXP num-const EXP num-var EXP (num-expr)

FLASH
Statement/Command
FLASH int-num-expr[;]

FN
FuNction
FN letter([num-expr][,num-expr]) FN letter$([string-expr][num-expr][,num-expr][,string-expr])

FOR
Statement/Command
FOR letter = num-expr TO num-expr[STEP num-expr]

FORMAT
Statement/Command
FORMAT num-const FORMAT num-var FORMAT (num-expr)

GO SUB
GO to SUBroutine
Statement/Command
GO SUB int-num-expr

GO TO
GO TO line
Statement/Command
GO SUB int-num-expr

IF
Statement/Command
IF num-expr THEN statement[:statement] IF cond THEN statement[:statement]

IN
Function
IN num-const IN num-var IN (num-expr)

INKEY$
INput Key (string)
Function
INKEY$

INPUT
Statement/Command
INPUT [prompt][;][,][']num-var INPUT [prompt][;][,][']string-var INPUT [prompt][;][,][']
LINE string-var

INT
INteger
Function
INT num-const INT num-var INT (num-expr)

INVERSE
Statement/Command
INVERSE int-num-expr

LEN
LENgth of string
Function
LEN string-const LEN string-var LEN (string-expr)

LET
Satement/Command
LET num-var = num-expr LET string-var = string-expr

LINE
See INPUT, SAVE

LIST
Command
LIST [int-num-expr]

LLIST
Line printer LIST
LL. (LIST #3)
Command
LLIST [int-num-expr]

LN
Logarithm (Natural)
Function
LN num-const LN num-var LN (num-expr)

LOAD
Command/Statement
LOAD string-expr LOAD string-expr CODE [int-num-expr][,int-num-expr] LOAD string-expr DATA letter[$]() LOAD string-expr SCREEN$

LPRINT
Line printer PRINT
LP. (PRINT #3)
Statement/Command
LPRINT [TAB int-num-expr;][AT int-num-expr,int-num-expr;][CHR$ (int-num-expr);][statement;][num- expr][string-expr][;][,][']

MERGE
Statement/Command
MERGE string-expr

MOVE
Statement/Command
MOVE int-num-expr,int-num-expr,int-num-expr

NEW
Command
NEW

NEXT
Statement/Command
NEXT letter

NOT
Logical Operator/Function
NOT cond NOT num-expr

ON ERR
Statement/Command
ON ERR CONTINUE ON ERR GO TO num-expr ON ERR STOP

OPEN
Statement/Command
OPEN #int-num-expr

OR
Logical Operator/Function
cond OR cond num-expr OR num-expr

OUT
Statement/Command
OUT int-num-expr,num-expr

OVER
Statement/Command
OVER int-num-expr

PALETTE
Statement/Command
PALETTE num-expr,num-expr

PAPER
Statement/Command
PAPER int-num-expr[;]

PAUSE
Statement/Command
PAUSE int-num-expr

PEEK
Statement/Command
PEEK int-num-const
PEEK int-num-var PEEK (int-num-expr)

PEN
Statement/Command
PEN int-num-expr[;]

PI
Function
PI

PLOT
Statement/Command
PLOT [statement:]int-num-expr,int-num-expr

POINT
Function
POINT (int-num-expr, int-num-expr)

POKE
Statement/Command
POKE int-num-expr, int-num-expr

PRINT
Statement/Command
PRINT [TAB int-num-expr;][AT int-num-expr,int-num-expr;][CHR$ (int-num-expr);][statement;][num- expr][string-expr][;][,][']

RANDOMIZE
Statement/Command
RANDOMIZE [int-num-expr]

READ
Statement/Command
READ num-var[,num-var][,string-var] READ string-var[,num-var][,string-var]

REM
REMark
REM [any characters]

RENUM
RENUMber
Command
RENUM [int-num-expr][,int-num-expr]

RESTORE
Statement/Command
RESTORE int-num-expr

RETURN
Statement/Command
RETURN

RND
RaNDom number
Function
RND

RUN
Statement/Command
RUN [int-num-expr]

SAVE
Statement/Command
SAVE string-expr [LINE int-num-expr] SAVE string-expr CODE int-num-expr,int-num-expr
SAVE string-expr DATA letter[$]() SAVE string-expr SCREEN

SCREEN$
SCREEN (string)
Function
SCREEN$ (int-num-expr,int-num-expr)

SGN
SiGN
Function
SGN num-const SGN num-var SGN (num-expr)

SIN
SINe
Function
SIN num-const SIN num-var SIN (num-expr)

SOUND
Statement/Command
SOUND int-num-expr,int-num-expr[;int-num-expr,int-num-expr]

SQR
SQuare Root
Function
SQR num-const SQR num-var SQR (num-expr)

STEP
See FOR.

STOP
Statement/Command
STOP

STR$
STRing (string)
Function
STR$ num-const STR$ num-var STR$ (num-expr)

TAB
TABulate See LPRINT, PRINT.

TAN
TANgetn
TAN num-const TAN num-var TAN (num-expr)

THEN
See IF.

TO
Function
string-const ([num-expr] TO [num-expr]) string-var ([num-expr] TO [num-expr]) (string-expr)([num-expr] TO [num-expr])

USR
User Sub-Routine
Function
USR int-num-const USR int-num-var USR (int-num-expr) USR string-const USR string-var

VAL
VALue
Function
VAL string-const VAL string-var

VAL$
VALue (string)
Function
VAL$ string-expr

VERIFY
Command/Statement
VERIFY string-expr VERIFY string-expr CODE [int-num-expr][,int-num-expr] VERIFY string-expr DATA letter[$]() VERIFY string-expr SCREEN$

Extended Character Set

Character sets may contain eight additional characters on character codes 24 to 31. No definitions are provided by default but you may use these characters in your own user defined character sets.

8-bit Character Set Support

This is controlled by bit 3 of the system variable FLAGS. You can enable 8-bit character set support from BASIC with POKE 23658,4 and switch it off again with POKE 23658,0. Alternatively you can use the DIR command to toggle support on and off.

When the mode is enabled, instead of printing block graphics, UDGs, and tokens, the print routine will expect to find a further 128 character definitions after the © character (addressed by the CHARS system variable).

Characters 24-255 are printable but the CHARS system variable (23606-23607) should be set to point to the zero character.

Memory Map

```
+-----+---------+------------+---------+-------+------+-----------+...
| ROM | Display | Attributes | Palette | Spare | UDGs | System    |
|     | File    |            |         |       |      | Variables |
+-----+---------+------------+---------+-------+------+-----------+...
^     ^         ^            ^         ^       ^      ^           ^
|     |         |            |         |       |      |           |
0   16384     22528        23296     23360   23384  23552       23734

...+------------+-------------+----+---------+-----------+----+...
   | Microdrive | Channel     | 0x | Basic   | Variables | 0x |
   | Maps       | Information | 80 | Program |           | 80 |
...+------------+-------------+----+---------+-----------+----+...
   ^            ^                  ^         ^                ^
   |            |                  |         |                |
23734        CHANS               PROG      VARS            ELINE

...+--------------------+---+----+-------+---+------------+------------+...
   | Command or program | N | 0x | INPUT | N | Temporary  | Calculator |
   | line being entered | L | 80 | Data  | L | Work Space | Stack      |
...+--------------------+---+----+-------+---+------------+------------+...
   ^                             ^                        ^            ^
   |                             |                        |            |
ELINE                         WORKSP                   STKBOT       STKEND

...+-------+---------+--------+---+----+
   | Spare | Machine | GO SUB | ? | 0x |
   |       | Stack   | Stack  |   | 3e |
...+-------+---------+--------+---+----+
   ^       ^                      ^    ^
   |       |                      |    |
STKEND    SP CHANS           RAMTOP  PRAMT
```

System Variables

KSTATE	23552	(8) Keyboard state.
LASTK	23560	Shift and key code from last key press.
REPDEL	23561	Delay before keys auto-repeat (in 50ths. of a second); normally 25.
REPSPD	23562	Delay between key repeats (in 50ths. of a second); normally 2.
DEFADD	23563	(2) DEF FN address (offset).
KDATA	23565	Used by keyscan.
TVDATA	23566	(2) Used in handling control codes and their parameters.
STREAMS	23568	(38) For streams -3 to 15, a word gives the displacement from the start of the channels area to the assigned channel. If the word is zero, the stream is closed.
CHARS	23606	(2) Address 256 bytes below start of main character set.
ERRSOUND	23608	Length of error sound in 50ths. of a second; normally 60.
CLICK	23609	Length of keyboard click (normally zero).
ERRNR	23610	Error number.
FLAGS	23611	Main flags byte.
DFLAG	23612	Display flags.
ERRSP	23613	(2) SP value to use when an error occurs.
LISTSP	23615	(2) SP value to use when an automatic list fills the screen.
MODE	23617	Cursor mode; L, C, E or G.
NEWPPC	23618	(2) New line to jump to.
NSPPC	23620	New statement to jump to, or FFH.
PPC	23621	(2) Current line number during program execution.
SUBPPC	23623	Current statement number.
BORDCR	23624	Attributes for lower screen except in MODE 2.
EPPC	23625	(2) number of line with > cursor.
VARS	23627	(2) Address of variables.
DEST	23629	(2) Used in variable assignments.
CHANS	23631	(2) start of channels area.
CURCHL	23633	(2) start of current channel.
PROG	23635	(2) Program start (address of line number of first line).
NXTLINE	23637	(2) Address of next line in Basic program.
DATADD	23639	(2) Data address used by READ command.
ELINE	23641	(2) Edit line start.
KCUR	23643	(2) Address of cursor in the edit line.
CHADD	23645	(2) Current character address.
XPTR	23647	(2) Address in the edit line of a syntax error.
STKBOT	23651	(2) Address of bottom of calculator stack.
STKEND	23653	(2) End of floating point calculator stack.
BREG	23655	Calculator's B register.
MEM	23656	(2) Start of calculator's memory area.
KLFLAG	23658	8 if caps lock is on, else zero.
DFSZ	23659	The number of lines (incldui ng one blank line) in the lower part of the screen.
SDTOP	23660	(2) Line number of top line in an automatic listing.
COPPC	23662	(2) Line number that CONTINUE goes to.
COSPCC	23664	Statement number that CONTINUE goes to.
FLAGE	23665	Flags used by INPUT command and the editor.
STRIL	23666	(2) Used when variables are assigned to.
TADDR	23668	(2) Address of next item in syntax table.
SEED	23670	(2) Random number seed. Set by RANDOMIZE.
FRAMES	23672	(3) Frames since machine was switched on (LSB first).
UDG	23675	(2) Address of CHR$ 144.
XCOORD	23677	Current graphics position x coordinate, with 0 at the left. The range is 0-255.
YCOORD	23678	Current graphics position y coordianate, with 175 at the top of the screen and 0 at the bottom.
ERRLN	23679	(2) line to go to ON ERR.
ONERRFLAG	23680	FFH=STOP, FEH=CONTINUE, else GO TO.
USER	23681	Not used.

ECHOE	23682	(2) 33 column number and 24 line number (in lower half) of end of input buffer.
DFCCU	23684	(2) Address in display file of upper window PRINT position.
DFCCL	23686	(2) Address in display file of lower window PRINT position.
SPOSNU	23688	(2) Upper window position as column/row.
SPOSNL	23690	(2) Lower window position as column/row.
SCRCT	23692	(2) Counter used to give “Scroll?” prompt.
ATTRP	23693	Attributes used by mode 0.
MASKP	23694	Mask used by mode 0. Bits which are 1 make the corresponding attribute bit be taken from the screen, not ATTRP.
ATTRT	23695	Temporary version of ATTRP.
MASKT	23696	Temporary versino of MASKP.
WORKSP	23649	(2) workspace start.
PFLAG	23697	Bit 4 and 5 are set for paper 9, bit 6 and 7 for pen 9.
MEMBOT	23698	(30) Calculator’s memory area.
NMIADD	23728	(2) Address to jump to when a peripheral activates the NMI.
RAMPTOP	23730	(2) Address of last byte of BASIC system area.
PRAMT	23732	(2) Address of last byte of physical RAM.

Flags

FLAGS
0 - set to prevent leading space
2 - set if last character detokenized was control code (temporary)
3 - set if 8-bit character set in use
5 - set if a key is pressed
6 - set if numeric result
7 - reset if checking syntax

DFLAG
0 - set when lower screen in use
3 - set if EDIT pressed
4 - set if automatic listing required
5 - set to clear lower screen

KLFLAG
0 - set to clear main screen
3 - set to enable caps lock
4 - set if K channel in use

FLAGE
0 - set if string
1 - set if variable
5 - set if INPUT mode
7 - set if INPUT line

ONERRFLAG
0-7 = set to STOP
1-6 = set to CONTINUE
6-7 = reset to GO TO

PFLAG
4 - set if pen 9
5 - set if pen 9
6 - set if paper 9
7 - set if paper 9

Error Reports

Codes refer to the equivalent SAM BASIC error report.

CODE	ERROR REPORT
0	OK No problems, successful completion, everything is OK.
1	Out of memory There is not enough room in the computer's memory for what you want to do.
2	Undefined variable The computer cannot find a variable, either because it has not yet been loaded, not been assigned or set up, or you have not set its dimensions.
3	End of DATA You are trying to READ past the end of the existing DATA listing.
4	Bad subscript Either the number of subscripts is wrong or the subscript is outside the dimensions of the array.
5	NEXT without FOR Even though there is an ordinary variable with the same name, the control variable has not yet been set up by a FOR statement.
6	FOR without NEXT Even though there is a FOR loop waiting to run, there is no NEXT statement to go with it.
7	Undefined FN A user-defined function is missing.
8	RETURN without GO SUB There is a RETURN statement without a GO SUB to welcome it back.
14	BREAK into program BREAK has been hit in between two statements, and the line and statement number that are shown refer to the statement before BREAK was used. When you CONTINUE, the program goes to the statement that follows and allows for any program jumps that you have made.
15	BREAK, CONTINUE repeats BREAK has been hit while a peripheral operation was taking place, so when you CONTINUE the last statement is repeated.
16	STOP statement When you want to CONTINUE after this, the program will start again at the next statement.
17	STOP in INPUT When you want to CONTINUE after this, the program will start again by repeating the last INPUT statement.
18	Bad filename You are trying to SAVE a file but have forgotten to give it a name, or the name is longer than 10 characters.

19 Loading error
The file you want to LOAD has been found but there is something wrong with it and it refuses to LOAD properly or fails to VERIFY. Check your cables, volume level, cassette tape and dirty play-back heads of the cassette player.

20 Bad device
You are trying to SAVE or LOAD data, but you are using the wrong thing for input/output (such as a disk drive instead of a cassette recorder), or have forgotten to plug it in.

21 Bad stream
You are trying to use a stream number that is inappropriate. Streams 0 to 165 are the paths to the various channels, e.g. 47 "K", "S", "R"; or you are trying to use a stream number that is closed.

22 End of file
The end of a file has been reached, usually a disk file.

23 Bad colour
You have tried to specify a colour with a number that is not appropriate.

26 Parameter error
Either you have used the wrong number of arguments, or the wrong type of argument, like a number instead of a string.

27 Bad argument
You are using an argument that is not suitable for the function you want.

28 Number too large
Your calculations have resulted in a number that is too enormous for the computer to handle.

29 Syntax error
The computer is confused by your (mis)use of BASIC.

30 Integer out of range
A whole number (called an integer) is required, but the argument you are using has been rounded to an integer that is outside of a suitable range.

31 Missing statement
The computer can't make a decision or obey an instruction without the necessary statements. For example, you may have deleted statements after a GO SUB and then RETURNed.

32 Off screen
The graphic requirements that you have asked for cannot fit on the screen.

33 No room for line
There is not enough room in the available memory for the line you are trying to insert, or the line numbering requested in a RENUM is impossible.

48 Bad CLEAR address
You are trying to CLEAR with a number beyond the limits of memory allocated to BASIC