

# Programming SE BASIC

Cordelia 4.2.0

Rob Hagemans

September 17, 2023

## Contents

<b>1</b>	<b>Quick start guide</b>	<b>4</b>
<b>2</b>	<b>User's guide</b>	<b>5</b>
<b>3</b>	<b>Configuration guide</b>	<b>6</b>
<b>4</b>	<b>Language guide</b>	<b>7</b>
<b>5</b>	<b>Language reference</b>	<b>8</b>
5.1	Functions . . . . .	8
5.1.1	ABS . . . . .	8
5.1.2	ACOS . . . . .	8
5.1.3	ASC . . . . .	8
5.1.4	ASIN . . . . .	8
5.1.5	ATAN . . . . .	9
5.1.6	CHR\$ . . . . .	9
5.1.7	COS . . . . .	9
5.1.8	DEEK . . . . .	9
5.1.9	EXP . . . . .	9
5.1.10	FIX . . . . .	10
5.1.11	FN . . . . .	10
5.1.12	INKEY\$ . . . . .	10
5.1.13	INP . . . . .	11
5.1.14	INT . . . . .	11
5.1.15	LEFT\$ . . . . .	11
5.1.16	LEN . . . . .	11
5.1.17	LOG . . . . .	12
5.1.18	MID\$ . . . . .	12
5.1.19	PEEK . . . . .	12
5.1.20	RIGHT\$ . . . . .	12
5.1.21	RND . . . . .	13
5.1.22	SGN . . . . .	13
5.1.23	SIN . . . . .	13
5.1.24	SQR . . . . .	14
5.1.25	STR\$ . . . . .	14
5.1.26	STRING\$ . . . . .	14
5.1.27	TAN . . . . .	14
5.1.28	USR . . . . .	15
5.1.29	VAL . . . . .	15
5.1.30	VAL\$ . . . . .	15
5.2	Statements . . . . .	15

5.2.1	BLOAD	16
5.2.2	BSAVE	16
5.2.3	CALL	16
5.2.4	CHDIR	17
5.2.5	CIRCLE	17
5.2.6	CLEAR	17
5.2.7	CLOSE	18
5.2.8	CLS	18
5.2.9	COLOR	18
5.2.10	CONT	19
5.2.11	COPY	19
5.2.12	DATA	19
5.2.13	DEF FN	20
5.2.14	DELETE	20
5.2.15	DIM	21
5.2.16	DOKE	21
5.2.17	DRAW	21
5.2.18	EDIT	22
5.2.19	ELSE	22
5.2.20	END	22
5.2.21	ERROR	22
5.2.22	FILES	22
5.2.23	FOR	23
5.2.24	GOSUB	23
5.2.25	GOTO	24
5.2.26	IF	24
5.2.27	INPUT	24
5.2.28	KEY (macro definition)	25
5.2.29	KEY (macro list)	25
5.2.30	KEY (macro toggle)	26
5.2.31	KILL	26
5.2.32	LET	26
5.2.33	LINE	26
5.2.34	LIST	26
5.2.35	LOAD	27
5.2.36	LOCATE	27
5.2.37	MERGE	28
5.2.38	MKDIR	28
5.2.39	NAME	28
5.2.40	NEW	29
5.2.41	NEXT	29
5.2.42	OLD	29
5.2.43	ON	29
5.2.44	ON ERROR	30
5.2.45	OPEN	30
5.2.46	OUT	30
5.2.47	PALETTE	30
5.2.48	PLAY	31
5.2.49	Example	31
5.2.50	PLOT	31
5.2.51	POKE	31
5.2.52	PRINT	32
5.2.53	RANDOMIZE	32
5.2.54	READ	33
5.2.55	REM	33
5.2.56	RENUM	33

---

5.2.57 RMDIR . . . . .	34
5.2.58 RUN . . . . .	34
5.2.59 SAVE . . . . .	34
5.2.60 SCREEN . . . . .	35
5.2.61 SOUND . . . . .	35
5.2.62 STOP . . . . .	35
5.2.63 TRACE . . . . .	35
5.2.64 WAIT . . . . .	36
5.2.65 WEND . . . . .	36
5.2.66 WHILE . . . . .	36
<b>6 Technical reference</b>	<b>37</b>
<b>7 Developer's guide</b>	<b>38</b>
<b>8 Acknowledgments</b>	<b>39</b>
<b>9 Licenses</b>	<b>40</b>

# 1 Quick start guide

text

## 2 User's guide

text

## 3 Configuration guide

text

## 4 Language guide

text

## 5 Language reference

---

### 5.1 Functions

---

*Functions* can only be used as part of an expression within a statement; they may take input values between parentheses and produce a return value. For example, in `PRINT ABS(-1)` the `ABS` function is used in an expression within a `PRINT` statement; in `Y = SQR(X) + 2` the `SQR` function is used in an expression within a `LET` statement.

For some function, including those that do not take an input, parentheses are optional. However, for maximum compatibility you may want to include them anyway.

#### 5.1.1 ABS

---

`y = ABS(x)`

Returns the absolute value of `x` if `x` is a number and the value of `x` if `x` is a string.

**5.1.1.1 Parameters** `x` is an expression.

#### 5.1.2 ACOS

---

`y = ACOS(x)`

Returns the inverse cosine of `x`.

**5.1.2.1 Parameters** `x` is a numeric expression that gives the angle in radians.

**5.1.2.2 Errors** `x` has a string value: Type mismatch.

#### 5.1.3 ASC

---

`val = ASC(char)`

Returns the **code point** (ASCII value) for the first character of `char`.

**5.1.3.1 Parameters** `char` is an expression with a string value.

#### 5.1.3.2 Errors

- `char` has a numeric value: Type mismatch.
- `char` equals `""`: Illegal function call.

#### 5.1.4 ASIN

---

`y = ASIN(x)`

Returns the inverse sine of `x`.

**5.1.4.1 Parameters** `x` is a numeric expression that gives the angle in radians.

**5.1.4.2 Errors** `x` has a string value: Type mismatch.



**5.1.5 ATAN**


---

y = ATAN(x)

Returns the inverse tangent of x.

**5.1.5.1 Parameters** x is a numeric expression that gives the angle in radians.

**5.1.5.2 Errors** x has a string value: Type mismatch.

**5.1.6 CHR\$**


---

char = CHR\$(x)

Returns the character with **code point** x.

**5.1.6.1 Parameters** x is a numeric expression in the range [0 to 255].

**5.1.6.2 Errors**

- x has a string value: Type mismatch.
- x is not in [-32768 to 32767]: Overflow.
- x is not in 0 to 255: Illegal function call.

**5.1.7 COS**


---

cosine = COS(angle)

Returns the cosine of angle.

**5.1.7.1 Parameters** angle is a numeric expression that gives the angle in radians.

**5.1.7.2 Errors** angle has a string value: Type mismatch.

**5.1.8 DEEK**


---

value = DEEK(address)

Returns the 16-bit value of the memory at segment \* 16 + address where segment is the current segment set with **DEF SEG**.

**5.1.8.1 Parameters** address is a numeric expression in [-32768 to 65535]. Negative values are interpreted as their two's complement.

**5.1.9 EXP**


---

y = EXP(x)

Returns the exponential of x, that is **e** to the power x.

**5.1.9.1 Parameters** x is a number-valued expression.

**5.1.9.2 Errors**

- x has a string value: Type mismatch.
- x is larger than the natural logarithm of the maximum single-precision value: Overflow.

**5.1.10 FIX**


---

```
whole = FIX(number)
```

Returns a number truncated towards zero.

**5.1.10.1 Parameters** number is a numeric expression.

**5.1.10.2 Notes** FIX truncates towards zero: it removes the fractional part. By contrast, **INT** truncates towards negative infinity.

**5.1.10.3 Errors** number is a string expression: Type mismatch.**5.1.11 FN**


---

```
result = FN[ ]name [(arg_0 [, arg_1] ...)
```

Evaluates the user-defined function previously defined with **DEF FN** name. Spaces between **FN** and name are required.

**5.1.11.1 Parameters**

- name is the name of a previously defined function.
- arg\_0, arg\_1, ... are expressions, given as parameters to the function.

**5.1.11.2 Notes**

- In Microsoft BASIC, spaces between **FN** and name are optional.
- Unlike Microsoft BASIC, in SE Basic IV, functions can be called recursively, albeit without tail call optimization.

**5.1.11.3 Errors**

- No function named name is defined: Undefined user function.
- The number of parameters differs from the function definition: Syntax error.
- The type of one or more parameters differs from the function definition: Type mismatch.
- The return type is incompatible with the function name's sigil: Type mismatch.

**5.1.12 INKEY\$**


---

```
key = INKEY$ [ #file_num]
```

Returns one character from the stream file\_num. If no stream is specified, returns one key-press from the keyboard buffer. If the keyboard buffer is empty, returns the empty string. Otherwise, the return value is a one-character string holding the e-ASCII code of the pressed key.

**5.1.12.1 Notes** When a function key F1 to F15 is pressed, **INKEY\$** will return the letters of the associated macro unless it's been set to empty with the **KEY** statement, in which case it returns the e-ASCII code for the function key.

**5.1.13 INP**


---

```
code = INP(port)
```

Returns the value of a machine port.

**5.1.13.1 Parameters** port is a numeric expression in [0 to 65535].

**5.1.14 INT**


---

```
whole = INT(number)
```

Returns number truncated towards negative infinity.

**5.1.14.1 Parameters** number is a numeric expression.

**5.1.14.2 Notes** **FIX** truncates towards zero: it removes the fractional part. By contrast, **INT** truncates towards negative infinity.

**5.1.14.3 Errors** number is a string expression: Type mismatch.

**5.1.15 LEFT\$**


---

```
child = LEFT$(parent, num_chars)
```

Returns the leftmost num\_chars characters of parent.

**5.1.15.1 Parameters**

- parent is a string expression.
- num\_chars is a numeric expression in [0 to 255].

**5.1.15.2 Notes**

- If num\_chars is zero or parent is empty, **LEFT\$** returns an empty string.
- If num\_chars is greater than the length of parent, returns parent.

**5.1.15.3 Errors**

- parent has a numeric value or num\_chars has a string value: Type mismatch.
- num\_chars is not in [-32768 to 32767]: Overflow.
- num\_chars is not in [0 to 255]: Illegal function call.

**5.1.16 LEN**


---

```
length = LEN(string)
```

Returns the number of characters in string.

**5.1.16.1 Parameters** string is a string expression.

**5.1.16.2 Errors** string has a number value: Type mismatch.

**5.1.17 LOG**


---

`y = LOG(x)`

Returns the natural logarithm of `x`.

**5.1.17.1 Parameters** `x` is a numeric expression greater than zero.

**5.1.17.2 Errors**

- `x` has a string value: Type mismatch.
- `x` is zero or negative: Illegal function call.

**5.1.18 MID\$**


---

`substring = MID$(string, position [, length])`

Returns a substring string starting at position, counting from 1. The substring has length `length` if specified. If `length` is not specified, the substring extends to the end of the string.

**5.1.18.1 Parameters**

- `string` is a string expression.
- `position` is a numeric expression between 1 and the string length, inclusive.
- `length` is a numeric expression in [0 to 255].

**5.1.18.2 Errors**

- `string` has a number value or `position` or `length` have string values: Type mismatch.
- `position` or `length` are not in [-32768 to 32767]: Overflow.
- `position` is not in [1 to 255]: Illegal function call.
- `length` is not in [0 to 255]: Illegal function call.

**5.1.19 PEEK**


---

`value = PEEK(address)`

Returns the value of the memory at segment \* 16 + `address` where segment is the current segment set with **DEF SEG**.

**5.1.19.1 Parameters** `address` is a numeric expression in [-32768 to 65535]. Negative values are interpreted as their two's complement.

**5.1.19.2 Notes** Currently PEEK only accepts values in the range [0 to 65535] and ignores **SEG**, returning values from the 64K address space.

**5.1.19.3 Errors**

- `address` has a string value: Type mismatch.
- `address` is not in [-32768 to 65535]: Overflow.

**5.1.20 RIGHT\$**


---

`child = RIGHT$(parent, num_chars)` Returns the rightmost `num_chars` characters of `parent`.

**5.1.20.1 Parameters**

- parent is a string expression.
- num\_chars is a numeric expression in [0 to 255].

**5.1.20.2 Notes**

- If num\_chars is zero or parent is empty, RIGHT\$ returns an empty string.
- If num\_chars is greater than the length of parent, returns parent.

**5.1.20.3 Errors**

- parent has a numeric value or num\_chars has a string value: Type mismatch.
- num\_chars is not in [-32768 to 32767]: Overflow.
- num\_chars is not in [0 to 255]: Illegal function call.

**5.1.21 RND**


---

random = RND[(x)]

Returns a pseudorandom number.

**5.1.21.1 Parameters** x is a numeric expression. This value is ignored.

**5.1.21.2 Notes**

- SE Basic IV's RND function produces different random numbers from Microsoft BASIC.
- It is a very poor random number generator. RND should not be used for cryptography, scientific simulations or anything else remotely serious.

**5.1.21.3 Errors** x has a string value: Type mismatch.

**5.1.22 SGN**


---

sign = SGN(number)

Returns the sign of number: 1 for positive, 0 for zero and -1 for negative.

**5.1.22.1 Parameters** number is a numeric expression.

**5.1.22.2 Errors** number has a string value: Type mismatch.

**5.1.23 SIN**


---

sine = SIN(angle)

Returns the sine of angle.

**5.1.23.1 Parameters** angle is a numeric expression giving the angle in radians.

**5.1.23.2 Errors** angle has a string value: Type mismatch.

**5.1.24 SQR**


---

```
root = SQR(number)
```

Returns the square root of number.

**5.1.24.1 Parameters** number is a numeric expression.

**5.1.24.2 Errors** number has a string value: Type mismatch

**5.1.25 STR\$**


---

```
repr = STR$(number[,base])
```

Returns the string representation of number.

**5.1.25.1 Parameters**

- number is a numeric expression.
- base is a numeric expression from [2 to 36]. If a base is not provided, defaults to base 10.

**5.1.25.2 Errors** number has a string value: Type mismatch.

**5.1.26 STRING\$**


---

```
string = STRING$(length, char)
```

Returns a string of length times the character char.

**5.1.26.1 Parameters**

- If char is a numeric expression, it must be in [0 to 255] and is interpreted as the **code point** of the character.
- If char is a string expression, its first character is used.

**5.1.26.2 Errors**

- length has a string value: Type mismatch.
- char is the empty string: Illegal function call.
- char or length is not in [-32768 to 32767]: Overflow.
- char or length is not in [0 to 255]: Illegal function call.

**5.1.27 TAN**


---

```
tangent = TAN(angle)
```

Returns the tangent of angle.

**5.1.27.1 Parameters** angle is a numeric expression giving the angle in radians.

**5.1.27.2 Errors** angle has a string value: Type mismatch.

### 5.1.28 USR

---

value = USR[n](expr)

Calls a machine-code function and returns its return value.

#### 5.1.28.1 Parameters

- n is a digit `<b>0</b>` to `<b>9</b>`.
- expr is an expression.

**5.1.28.2 Errors** n is not a digit [0 to 9]: Syntax error.

### 5.1.29 VAL

---

value = VAL(string)

Returns the numeric value of the string expression string. See the section on **numeric literals** for the recognised number formats.

#### 5.1.29.1 Notes

- Spaces before a number are ignored: VAL(" 10") returns 10. But unlike Microsoft BASIC, spaces inside a number are not ignored.
- Unlike Microsoft BASIC, expressions inside the string expression are also evaluated. For example, VAL "5+5" returns 10 and VAL "foo" returns the value of variable foo.
- Expressions between curly braces { and } are not evaluated, but their syntax is checked upon entering. They are interpreted as strings that can be passed to VAL for actual evaluation.

**5.1.29.2 Errors** string has a number value: Type mismatch.

### 5.1.30 VAL\$

---

repr = VAL\$(string)

Evaluates a string as a string expression. For example

```
10 INPUT a$, x$ 20 PRINT VAL$ a$
```

The string value assigned to a\$ should be an expression using x\$. For example, "x\$+x\$". A string value is then assigned to x\$, for example "yo". VAL\$ strips the quotes of the value of a\$ to get x\$+x\$ and evaluates it using the value assigned to x\$ displaying the result yoyo.

#### 5.1.30.1 Notes

- This function is not present in Microsoft BASIC. It is very useful for creating recursive functions, if used together with AND applied to string arguments, allowing for selective evaluation.
- Expressions between curly braces { and } are not evaluated, but their syntax is checked upon entering. They are interpreted as strings that can be passed to VAL\$ for actual evaluation.

## 5.2 Statements

---

A program line is composed of a line number and one or more statements. If multiple statements are put on one line, they must be separated by colons (:). Statements may be empty. Each statement has its own idiosyncratic syntax.

Many reference works distinguish commands and statements; this distinction stems from the original Dartmouth design of the BASIC language, in which commands were not part of the language and could not be used in programs, but were rather used to control the interpreter itself. However, in SE Basic IV this distinction is less useful and therefore this reference includes what is traditionally thought of as commands in the category of statements.

### 5.2.1 BLOAD

---

BLOAD file\_spec, offset

Loads a memory image file into memory.

#### 5.2.1.1 Parameters

- The string expression file\_spec is a valid **filespec** indicating the file to read the memory image from.
- offset is a numeric expression in the range [-32768 to 65535]. It indicates an offset in the current [DEF SEG]

segment where the file is to be stored. If not specified, the offset stored in the **BSAVE** file will be used. If negative, its two's complement will be used.

#### 5.2.1.2 Errors

- The loaded file is not in **BSAVE** format: Bad file mode.
- file\_spec contains disallowed characters: Bad file name.
- file\_spec has a numeric value: Type mismatch.
- offset is not in the range [-32768 to 65535]: Overflow.

### 5.2.2 BSAVE

---

BSAVE file\_spec, offset, length

Saves a region of memory to an image file.

#### 5.2.2.1 Parameters

- The string expression file\_spec is a valid **filespec** indicating the file to write to.
- offset is a numeric expression in the range [-32768 to 65535] indicating the offset into the current **DEF SEG** segment from where to start reading.
- length is a numeric expression in the range [-32768 to 65535] indicating the number of bytes to read.
- If offset or length are negative, their two's complement will be used.

#### 5.2.2.2 Errors

- file\_spec has a numeric value: Type mismatch.
- file\_spec contains disallowed characters: Bad file number (on CAS1:); Bad file name (on disk devices).
- offset is not in the range [-32768 to 65535]: Overflow.
- length is not in the range [-32768 to 65535]: Overflow.

### 5.2.3 CALL

---

CALL address\_var [, p0, p1, ...]

Executes a machine language subroutine.



**5.2.3.1 Parameters**

- address\_var is a numeric variable.
- p0, p1, ... are variables.

**5.2.3.2 Errors**

- address\_var is a string variable: Type mismatch.
- address\_var is a literal: Syntax error.

**5.2.4 CHDIR**


---

CHDIR dir\_spec

Change the current folder on a disk device to dir\_spec. Each disk device has its own current folder.

**5.2.4.1 Parameters** The string expression dir\_spec is a valid **filespec** indicating an existing folder on a disk device.

**5.2.4.2 Errors**

- No matching path is found: Path not found.
- dir\_spec has a numeric value: Type mismatch.
- dir\_spec is empty: Bad file name.

**5.2.5 CIRCLE**


---

CIRCLE x , y , r

Draws a circle where x and y are the center coordinates and r is the radius.

**5.2.6 CLEAR**


---

CLEAR [mem\_limit]

Clears all variables, arrays, **DEF FN** user functions. Closes all files. Turns off all sound. Clears all **ON ERROR** traps. Clears the loop stack.

**5.2.6.1 Parameters** mem\_limit specifies the upper limit of usable memory. Default is previous memory size. Default memory size is 65535.

**5.2.6.2 Notes** If called inside a **FOR to NEXT** or **WHILE to WEND** loop, an error will be raised at the **NEXT** or **WEND** statement, since the loop stacks have been cleared.

**5.2.6.3 Errors**

- Any of the arguments has a string value: Type mismatch.
- mem\_limit is not in [0 to 65535]: Overflow.
- mem\_limit is too low: Address out of range.

### 5.2.7 CLOSE

---

CLOSE [[#] file\_0 [, [#] file\_1] ...]

Closes streams. If no file numbers are specified, all open streams [3 to 15] are closed. The hash (#) is optional and has no effect.

**5.2.7.1 Parameters** file\_1, file\_2, ... are numeric expressions yielding stream numbers.

#### 5.2.7.2 Errors

- file\_1, file\_2, ... are not in [0 to 15]: Bad I/O device.
- file\_1, file\_2, ... are not open streams: Undefined stream.
- file\_1, file\_2, ... have a string value: Type mismatch.
- The statement ends in a comma, Syntax error.
- If an error occurs, only the files before the erratic value are closed.

### 5.2.8 CLS

---

CLS [x]

Clears the screen or part of it. If x is not specified, in SCREEN 0 the text view region is cleared; in other screens, the graphics view region is cleared. The comma is optional and has no effect.

**5.2.8.1 Parameters** x is a numeric valued expression that determines what is cleared:

#### 5.2.8.2 Notes

- If x = 0, the whole screen is cleared.
- If x = 1, the graphics view region is cleared.
- If x = 2, the text view region is cleared.

#### 5.2.8.3 Errors

- x is has a string value: Type mismatch.
- x is not in [-32768 to 32767]: Overflow .
- x is not in [0, 1, 2]: Illegal function call.
- If an error occurs, the screen is not cleared.

### 5.2.9 COLOR

---

COLOR [foreground] [, [background] [, border]]

Changes the current foreground and background attributes. All new characters printed will take the newly set attributes. Existing characters on the screen are not affected.

#### 5.2.9.1 Parameters

- foreground is a numeric expression in [0 to 15]. This specifies the new foreground attribute.
- background is a numeric expression in 0 to 15. This specifies the new background attribute.
- border is a numeric expression in [0 to 15] specifying the border attribute. It is taken MOD 8: Values 8 to 15 produce the same colour as 0 to 7.

**5.2.9.2 Errors**

- Any of the parameters has a string value: Type mismatch.
- Any of the parameters is not in [-32768 to 32767]: Overflow.
- foreground is not in [0 to 31], background is not in [0 to 15] or border is not in [0 to 15]: Illegal function call.
- Statement is used in SCREEN 2: Illegal function call.

**5.2.10 CONT**


---

CONT

Resumes execution of a program that has been halted by **STOP**, **END** or Esc.

**5.2.10.1 Notes**

- Anything after the **CONT** keyword is ignored.
- This statement can only be used in direct mode.
- If a break is encountered in **GOSUB** routine called from a continuing direct line (for example, **GOSUB 100:PRINT A\$**), **CONT** will overwrite the running direct line. As the subroutine **RETURNS** to the position after the **GOSUB** in the old direct line, strange things may happen if commands are given after **CONT**. In Microsoft BASIC, this can lead to strange errors in non-existing program lines as the parser executes bytes that are not part of a program line. In SE Basic IV, if the new direct line is shorter, execution stops after **RETURN**; but if the direct line is extended beyond the old return position, the parser tries to resume at that return position, with strange effects.

**5.2.10.2 Errors**

- No program is loaded, a program has not been run, after a program line has been modified or after **CLEAR**: Can't continue.
- The break occurred in a direct line: Can't continue.
- **CONT** is used in a program: Can't continue.

**5.2.11 COPY**


---

COPY file\_spec\_1 TO file\_spec\_2

Copies the disk file file\_spec\_1 to file\_spec\_2.

**5.2.11.1 Parameters** The string expressions file\_spec\_1 and file\_spec\_2 are valid **file specifications** indicating the source and destination files. The first must point to an existing file on a disk device.

**5.2.11.2 Notes** Typically, this command is not present in Microsoft BASIC.

**5.2.11.3 Errors**

- file\_spec\_1 or file\_spec\_2 have number values: Type mismatch
- file\_spec\_1 does not exist: File not found

**5.2.12 DATA**


---

DATA [const\_0] [, [const\_1]] ...

Specifies data that can be read by a **READ** statement.

**5.2.12.1 Parameters** `const_0, const_1, ...` are string and number literals or may be empty. String literals can be given with or without quotation marks. If quotation marks are omitted, leading and trailing whitespace is ignored and commas or colons will terminate the data statement.

**5.2.12.2 Errors** If the type of the literal does not match that of the corresponding **READ** statement, a Syntax error occurs on the **DATA** statement.

### 5.2.13 DEF FN

---

```
DEF FN[ ]name [( arg_0 [, arg_1] ...)] = expression
```

Defines a function called **FN** name (or FN name: spaces between FN and name are optional). On calling FNname( ... ), expression is evaluated with the supplied parameters substituted. Any variable names used in the function that are not in the argument list refer to the corresponding global variables. The result of the evaluation is the return value of FNname. The type of the return value must be compatible with the type indicated by name.

**5.2.13.1 Example** Create the recursive function FN F(n) to calculate the factorial for n:

```
DEF FN F(N)=VAL (({N*FN F(N-1)} AND N)+({1} AND (N=0)))
```

#### 5.2.13.2 Notes

- This statement may only be used on a program line.
- As the function must be a single expression and SE Basic IV does not have a ternary operator, the only way to define a recursive function that actually terminates is by using **VAL** or **VAL\$**.

#### 5.2.13.3 Parameters

- name must be a legal variable name.
- `arg_0, arg_1, ...` must be legal variable names. These are the parameters of the function. Variables of the same name may or may not exist in the program; their value is not affected or used by the defined function.
- expression must be a legal SE Basic IV expression.

#### 5.2.13.4 Errors

- The statement is executed directly instead of in a program line: Illegal direct.
- If the type of the return value is incompatible with the type of name, no error is raised at the DEF FN statement; however, a Type mismatch will be raised at the first call of FNname.

### 5.2.14 DELETE

---

```
DELETE [line_number_0|.] [-[line_number_1|.]]
```

Deletes a range of lines from the program. Also stops program execution and returns control to the user.

#### 5.2.14.1 Parameters

- `line_number_0` and `line_number_1` are line numbers in the range [0 to 65529], specifying the inclusive range of line numbers to delete.
- A `.` indicates the last line edited.
- If the start point is omitted, the range will start at the start of the program.
- If the end point is omitted, the range will end at the end of the program.
- If no range is specified, the whole program will be deleted.

**5.2.14.2 Errors**

- line\_number\_0 or line\_number\_1 is greater than 65529: Syntax error.
- The range specified does not include any program lines stored: Illegal function call.

**5.2.15 DIM**


---

DIM name {[[*limit\_0* [, *limit\_1*] ... {}]]}

Allocates memory for arrays. The DIM statement also fixes the number of indices of the array. Unlike Microsoft BASIC, an array can be reallocated.

**5.2.15.1 Parameters**

- name is a legal variable name specifying the array to be allocated.
- *limit\_0*, *limit\_1*, ... are numeric expressions that specify the greatest index allowed at that position.

**5.2.15.2 Notes** The size of arrays is limited by the available BASIC memory.

**5.2.15.3 Errors**

- An index is empty: Syntax error.
- An index is missing at the end: Missing operand.
- *limit\_0*, *limit\_1*, ... have a string value: Type mismatch.
- *limit\_0*, *limit\_1*, ... are not within [-32768 to 32767]: Overflow.
- *limit\_0*, *limit\_1*, ... are negative: Illegal function call.
- The array exceeds the size of available variable space: Out of memory.

**5.2.16 DOKE**


---

DOKE address, value

Sets the 16-bit value of the memory byte pair at segment \* 16 + address to value, where segment is the current segment set with DEF SEG.

**5.2.16.1 Parameters**

- address is a numeric expression in [0 to 65535]. Negative values are interpreted as their two's complement.
- value is a numeric expression in [0 to 65535].

**5.2.16.2 Notes**

- DEF SEG is not yet implemented in SE Basic IV.

**5.2.16.3 Errors**

- address or value has a string value: Type mismatch.
- address is not in [-32768 to 65535]: Overflow.
- value is not in [-32768 to 32767]: Overflow.
- value is not in [0 to 65535]: Illegal function call.

**5.2.17 DRAW**


---

DRAW x , y

Draws a line x pixels to the right and y pixels up from the last PLOT location. Accepts negative values.

### 5.2.18 EDIT

---

EDIT {line\_number|.}

Displays the specified program line with the cursor positioned for editing. line\_number must be a line that exists in the program, or a period (.) to indicate the last line stored.

#### 5.2.18.1 Errors

- No line\_number is specified: Undefined line number.
- More characters are written after the line number: Illegal function call.
- line\_number is not in [0 to 65529] Illegal function call.
- The specified line number does not exist: Undefined line number.

### 5.2.19 ELSE

---

ELSE [anything]

Unless part of an **IF** statement on the same line, anything after **ELSE** is ignored in the same way as after ' or **:REM**. Unlike Microsoft BASIC, a colon (:) preceding the **ELSE** statement is required. However, if you enter a space before **ELSE** the tokenizer will add the colon for you. See **IF** for normal usage.

### 5.2.20 END

---

END

Closes all files, stops program execution and returns control to the user. No message is printed. It is possible to resume execution at the next statement using **CONT**.

### 5.2.21 ERROR

---

ERROR error\_number

Raises the error with number error\_number.

#### 5.2.21.1 Parameters

- error\_number is an expression with a numeric value.

#### 5.2.21.2 Errors

- error\_number has a string value: Type mismatch.
- error\_number is not in [-32768 to 32767]: Overflow.
- error\_number is not in [1 to 255]: Illegal function call.

### 5.2.22 FILES

---

FILES [filter\_spec]

Displays the files fitting the specified filter in the specified folder on a disk device. If filter\_spec is not specified, displays all files in the current working folder.

**5.2.22.1 Parameters** filter\_spec is a string expression that is much like a **filespec**, but optionally allows the file name part to contain **wildcards**.

**5.2.22.2 Notes** Wildcards are not currently supported.

### 5.2.22.3 Errors

- filter\_spec has a numeric value: Type mismatch.
- filter\_spec is the empty string: Bad file name.
- The specified filter does not match any files: File not found.

## 5.2.23 FOR

---

FOR loop\_var = start TO stop [STEP step]

Initiates a FOR to NEXT loop.

Initially, loop\_var is set to start. Then, the statements between the FOR statement and the NEXT statement are executed and loop\_var is incremented by step (if step is not specified, by 1). This is repeated until loop\_var has become greater than stop. Execution then continues at the statement following NEXT. The value of loop\_var equals stop+step after the loop.

### 5.2.23.1 Parameters

- loop\_var is a numeric variable.
- start, stop and step are numeric expressions.

### 5.2.23.2 Errors

- No NEXT statement is found to match the FOR statement: FOR without NEXT occurs at the FOR statement.
- loop\_var is a string variable or start, stop, or end has a string value: Type mismatch.
- loop\_var is an array element: Syntax error.
- loop\_var is an integer variable and a start, stop or step is outside the range [-32768, 32767]: Overflow.

## 5.2.24 GOSUB

---

GO [SUB]line\_number [anything]

Jumps to a subroutine at line\_number. The next RETURN statement jumps back to the statement after GOSUB. Anything after line\_number until the end of the statement is ignored. If executed from a direct line, GOSUB runs the subroutine and the following RETURN returns execution to the direct line.

### 5.2.24.1 Parameters

- line\_number is an existing line number literal.
- Further characters on the line are ignored until end of statement.

### 5.2.24.2 Notes

- If no RETURN is encountered, no problem.
- One optional space is allowed between GO and SUB; it will not be retained in the program.

### 5.2.24.3 Errors

- If line\_number does not exist: Undefined line number.
- If line\_number is greater than 65529, only the first four characters are read (for example, 6553).

### 5.2.25 GOTO

---

GOTO line\_number [anything]

Jumps to line\_number. Anything after line\_number until the end of the statement is ignored. If executed from a direct line, GOTO starts execution of the program at the specified line.

#### 5.2.25.1 Parameters

- line\_number is an existing line number literal.
- Further characters on the line are ignored until end of statement.

#### 5.2.25.2 Notes

No spaces are allowed between GO and TO.

#### 5.2.25.3 Errors

- line\_number does not exist: Undefined line number.

### 5.2.26 IF

---

IF truth\_value {THEN|GOTO} [compound\_statement\_true|line\_number\_true [anything]][ELSE [compound\_statement\_false|line\_number\_false [anything]]]

If truth\_value is non-zero, executes compound\_statement\_true or jumps to line\_number\_true. If it is zero, executes compound\_statement\_false or jumps to line\_number\_false.

#### 5.2.26.1 Parameters

- truth\_value is a numeric expression.
- line\_number\_false and line\_number\_true are existing line numbers.
- compound\_statement\_false and compound\_statement\_true are compound statements, consisting of at least one statement, optionally followed by further statements separated by colons (:). The compound statements may contain nested IF to THEN to ELSE statements.

#### 5.2.26.2 Notes

- The comma is optional and ignored.
- ELSE clauses are optional; they are bound to the innermost free IF statement if nested. Additional ELSE clauses that have no matching IF are ignored.
- All clauses must be on the same program line.
- THEN and GOTO are interchangeable; which one is chosen is independent of whether a statement or a line number is given. GOTO PRINT 1' is fine.
- As in GOTO, anything after the line number is ignored.

#### 5.2.26.3 Errors

- If truth\_value has a string value: Type mismatch.
- truth\_value equals 0 and line\_number\_false is a non-existing line number, or truth\_value is nonzero and line\_number\_true is a non-existing line number: Undefined line number.

### 5.2.27 INPUT

---

INPUT [;] [prompt {;|,}] var\_0 [, var\_1] ...

Prints prompt to the screen and waits for the user to input values for the specified variables. The semicolon before the prompt, if present, stops a newline from being printed after the values have been entered. If the prompt is



followed by a semicolon, it is printed with a trailing ?. If the prompt is followed by a comma, no question mark is added.

#### 5.2.27.1 Parameters

- prompt is a string literal.
- var\_0, var\_1, ... are variable names or fully indexed array elements.

#### 5.2.27.2 Notes

- Values entered must be separated by commas. Leading and trailing whitespace is discarded.
- String values can be entered with or without double quotes (").
- If a string with a comma, leading or trailing whitespace is needed, quotes are the only way to enter it.
- Between a closing quote and the comma at the end of the entry, only white- space is allowed.
- If quotes are needed in the string itself, the first character must be neither a quote nor whitespace. It is not possible to enter a string that starts with a quote through INPUT.
- If a given var\_n is a numeric variable, the value entered must be number literal.
- Characters beyond the 255th character of the screen line are discarded.
- If user input is interrupted by Ctrl+Break, **CONT** will re-execute the INPUT statement.

#### 5.2.27.3 Errors

- If the value entered for a numeric variable is not a valid numeric literal, or the number of values entered does not match the number of variables in the statement, ?Redo from start is printed and all values must be entered again.
- A Syntax error that is caused after the prompt is printed is only raised after the value shave been entered. No values are stored.

### 5.2.28 KEY (macro definition)

---

KEY key\_id, string\_value

Defines the string macro for function key **key\_id**. Only the first 15 characters of **string\_value** are stored.

#### 5.2.28.1 Parameters

- key\_id is a numeric expression in the range [1 to 15].
- string\_value is a string expression.

#### 5.2.28.2 Notes

- If key\_id is not in the prescribed range, an error is raised.
- If string\_value is the empty string or the first character of string\_value is CHR\$(0), the function key macro is switched off and subsequent catching of the associated function key with **INKEY\$** is enabled.

#### 5.2.28.3 Errors

- key\_id is not in [-32768 to 32767]: Overflow.
- key\_id is not in [1 to 255]: Illegal function call.
- key\_id has a string value: Type mismatch.

### 5.2.29 KEY (macro list)

---

KEY LIST

Prints a list of the 15 function keys with the function-key macros defined for those keys to the console.

Most characters are represented by their symbol equivalent in the current [codepage](#). However, some characters get a different representation, which is a symbolic representation of the effect as control characters on the screen.

### 5.2.30 KEY (macro toggle)

---

KEY {ON|OFF}

Toggles function-key macros ON or OFF.

### 5.2.31 KILL

---

KILL file\_spec

Deletes a file on a disk device.

### 5.2.32 LET

---

[LET] name = expression

Assigns the value of expression to the variable or array element name.

#### 5.2.32.1 Parameters

- name is a variable that may or may not already exist.
- The type of expression matches that of name: that is, all numeric types can be assigned to each other but strings can only be assigned to strings.

**5.2.32.2 Errors** name and expression are not of matching types: Type mismatch.

### 5.2.33 LINE

---

See **INPUT**.

### 5.2.34 LIST

---

LIST [# file\_num;] [line\_number\_0][, ][line\_number\_1]

Prints the program to the screen or a file, starting with line\_number\_0 up to and including line\_number\_1. Also stops program execution and returns control to the user. In all cases, any further statements in a compound after LIST will be ignored, both in a program and in direct mode.

When listing to the screen, the same control characters are recognised as in the **PRINT** statement.

#### 5.2.34.1 Notes

- In Microsoft BASIC, LIST will not show line numbers 65531 to 65535 inclusive.
- SE Basic IV's line range is currently [0 to 16383].
- There is no LLIST command. Instead, LIST can be directed to the printer stream using LIST #.

**5.2.34.2 Parameters**

- line\_number\_0 and line\_number\_1 are line numbers in the range [0 to 65529] or a . to indicate the last line edited. The line numbers do not need to exist; they specify a range. If the range is empty, nothing is printed.
- The string expression file\_num is a valid stream indicating the file to list to.

**5.2.34.3 Errors**

- A line number is greater than 65529: Syntax error.
- file\_num has a string value: Type mismatch.

**5.2.35 LOAD**


---

LOAD file\_spec [, {"R"|"T"}]

Loads the program stored in a file into memory. Existing variables will be cleared and any program in memory will be erased. LOAD implies a [CLEAR] (CLEAR).

If , "R" is specified, keeps all data files open and runs the specified file. If , "T" is specified, loads a tokenized program.

**5.2.35.1 Parameters** The string expression file\_spec is a valid filespec indicating the file to read the program from.

**5.2.35.2 Notes**

- Unlike Microsoft BASIC, SE Basic IV always expects BASIC programs to be in plain text format.
- Unlike Microsoft BASIC, the R directive must be in quotes. Otherwise SE Basic IV would treat it as a variable.

**5.2.35.3 Errors**

- file\_spec has a numeric value: Type mismatch.
- file\_spec contains disallowed characters: Bad file name.
- The file specified in file\_spec cannot be found: File not found.
- A loaded text file contains lines without line numbers: Direct statement in file.

**5.2.36 LOCATE**


---

LOCATE[row], [col]

Set the next print position to row, col on the screen.

**5.2.36.1 Notes** In Microsoft BASIC, the cursor can be displayed or made invisible and its shape can be changed. This is not supported in SE Basic IV.

**5.2.36.2 Errors**

- Any parameter has a string value: Type mismatch.
- Any parameter is not in [-32768 to 32767]: Overflow.
- row is outside the current view area: Illegal function call.
- col is greater than the current width: Illegal function call.

### 5.2.37 MERGE

---

MERGE file\_spec

Merges the program stored in a file into memory.

**5.2.37.1 Parameters** The string expression file\_spec is a valid **filespec** indicating the file to read the program from.

**5.2.37.2 Notes** Unlike Microsoft BASIC, SE Basic IV always expects BASIC programs to be in plain text format.

#### 5.2.37.3 Errors

- file\_spec has a numeric value: Type mismatch.
- file\_spec contains disallowed characters: Bad file name.
- The file specified in file\_spec cannot be found: File not found.
- A loaded text file contains lines without line numbers: Direct statement in file.

### 5.2.38 MKDIR

---

MKDIR dir\_spec

Creates a new folder on a disk device.

**5.2.38.1 Parameters** The string expression dir\_spec is a valid **filespec** that specifies the path of the new folder on a disk device.

#### 5.2.38.2 Errors

- dir\_spec is not a string: Type mismatch.
- The parent folder does not exist: Path not found.
- The folder name already exists on that path: Path/File access error.
- The user has no write permission: Permission denied.

### 5.2.39 NAME

---

NAME old\_name TO new\_name

Renames the disk file old\_name into new\_name.

**5.2.39.1 Parameters** The string expressions old\_name and new\_name are valid **filespec** file specifications giving the path on a disk device to the old and new filenames, respectively.

**5.2.39.2 Notes** new\_name will be modified into **all-uppercase 8.3 format**.

#### 5.2.39.3 Errors

- old\_name or new\_name have number values: Type mismatch.
- old\_name does not exist: File not found.
- old\_name is open: File already open.
- new\_name exists: File already exists.

**5.2.40 NEW**

---

**NEW**

Stops execution of a program, deletes the program in memory, executes **CLEAR** and **RESTORE** and returns control to the user.

**5.2.41 NEXT**


---

**NEXT** [var\_0 [, var\_1] ...]

Iterates a **FOR** to **NEXT** loop: increments the loop variable and jumps to the **FOR** statement. If no variables are specified, next matches the most recent **FOR** statement. Several nested **NEXT** statements can be consolidated into one by using the variable list. If one or more variables are specified, their order must match the order of earlier **FOR** statements.

**5.2.41.1 Parameters** var\_0, var\_1, ... are numeric variables which are loop counters in a **FOR** statement.

**5.2.41.2 Errors**

- No **FOR** statement is found to match the **NEXT** statement and variables: **NEXT** without **FOR**.
- var\_0, var\_1, ... are string variables: **NEXT** without **FOR**.
- The (implicit or explicit) loop variable is an integer variable and is taken outside the range [-32768, 32767] when incremented after the final iteration: **Overflow**.

**5.2.42 OLD**

---

**OLD**

Loads a backup from disk of the program that was in memory the last time a **NEW** command was issued and returns control to the user.

**5.2.43 ON**


---

**ON** n {**GOTO**|**GOSUB**} line\_number\_0 [, line\_number\_1] ...

Jumps to the nth line number specified in the list. If n is 0 or greater than the number of line numbers in the list, no jump is performed. If **GOTO** is specified, the jump is unconditional; if **GOSUB** is specified, jumps to a subroutine.

**5.2.43.1 Parameters**

- n is a numeric expression in [0 to 255].
- line\_number\_0, line\_number\_1, ... are existing line numbers in the program.

**5.2.43.2 Errors**

- n has a string value: **Type mismatch**.
- n is not in [-32768 to 32767], **Overflow**.
- n is not in [0 to 255]: **Illegal function call**.
- The line number jumped to does not exist: **Undefined line number**.

**5.2.44 ON ERROR**


---

ON ERROR GOTO {line\_number|0}

Turns error trapping on or off. When line\_number is set, any error causes the error handling routine starting at that line number to be called; no message is printed and program execution is not stopped. The error handling routine is ended by a **RESUME** statement. While in an error handling routine, events are paused and error trapping is disabled. After the **RESUME** statement, any triggered events are picked up in the following order: **KEY**, **TIMER**, **PLAY** - the order of the others is unknown. Unlike event trapping, error trapping remains active when no program is running. **ON ERROR CONT** turns off error trapping.

**5.2.44.1 Parameters** line\_number is an existing line number in the program.

**5.2.44.2 Notes** It is not possible to start the error handler at line number 0.

**5.2.44.3 Errors** line\_number does not exist: Undefined line number.

**5.2.45 OPEN**


---

OPEN #file\_num, "mode" [,file\_spec]

**5.2.46 OUT**


---

OUT port, value

Sends a byte to a machine I/O port.

**5.2.46.1 Parameters**

- port is a numeric expression in [0 to 65535].
- value is a numeric expression in [0 to 255].

**5.2.46.2 Errors**

- port or value has a string value: Type mismatch.
- port is not in [0 to 65535]: Overflow.
- value is not in [0 to 32767]: Overflow.
- value is not in [0 to 255]: Illegal function call.

**5.2.47 PALETTE**


---

PALETTE [attrib, color]

Assigns a color to an attribute. All pixels with that attribute will change color immediately. If no parameters are specified, **PALETTE** resets to the initial setting.

**5.2.47.1 Parameters**

- attrib is a numeric expression from [0 to 63].
- color is a numeric expression between [0 and 255]

**5.2.47.2 Notes** Colors are entered in compressed RGB format (lowest to highest bit). The red and green levels are each stored in three bits (0 to 7) while the blue level is stored in two bits (0 to 3). The easiest way to enter values is in octal (@BGR). For example, to set attribute to maximum blue, you would enter: PALETTE attribute, @300.

### 5.2.47.3 Errors

- attrib or colour has a string value: Type mismatch.
- attrib or colour is not in [0 to 32767]: Overflow
- attrib or colour is not in range: Illegal function call

### 5.2.48 PLAY

---

PLAY mml\_string\_1[, mml\_string\_2[, ... mml\_string\_8]

Plays the tune defined by the Music Macro Language (MML) strings mml\_string\_1, ....

### 5.2.49 Example

```
10 a$ = "h1t100o2[[[1CCg-b]] [[-E-E-b-D]] [[FFC-E]] [[GGDF]]]"
20 b$ = "h2o5rrrrrC-bfG]"
30 c$ = "h3o4[[3Cr]Cr1CCDD[3-Er]-Er1-E-EEE[3Fr]Fr1FF-G-G[3Gr]Gr1GG-EC]"
40 d$ = "h4l9rrrrrrrr[9EGF7b5CD]"
50 PLAY a$, b$, c$, d$
```

#### 5.2.49.1 Parameters

- mml\_string\_1...mml\_string\_8 are string expressions in MML.
- At least one parameter must be provided and the statement must not end in a comma.

#### 5.2.49.2 Errors

- mml\_string has a numeric value: Type mismatch.
- mml\_string has errors in the MML: Syntax error.

### 5.2.50 PLOT

---

PLOT x , y

Draws a pixel at coordinates x and y.

### 5.2.51 POKE

---

POKE address, value

Sets the value of the memory byte at segment \* 16 + address to value, where segment is the current segment set with DEF SEG.

#### 5.2.51.1 Parameters

- address is a numeric expression in [-32768 to 65535]. Negative values are interpreted as their two's complement.
- value is a numeric expression in [0 to 255].

#### 5.2.51.2 Notes

- DEF SEG is not yet implemented in SE Basic IV.

**5.2.51.3 Errors**

- address or value has a string value: Type mismatch.
- address is not in [-32768 to 65535]: Overflow.
- value is not in [-32768 to 32767]: Overflow.
- value is not in [0 to 255]: Illegal function call.

**5.2.52 PRINT**


---

PRINT [# stream,] [expr\_0|;|SPC( n)|TAB( n)] ... [USING format; uexpr\_0 [{;|,} uexpr\_1] ... [;|,]]

Writes expressions to the screen, a file or another device. If stream is specified, output goes to the file or device open under that number. ? is a shorthand for PRINT.

When writing a string expression to the screen, the following control characters have special meaning. Other characters are shown as their corresponding glyph in the current [codepage](#).

Code	Chr	Effect
\$07	BEL	Beep the speaker.
\$08	BS	Erase character in previous column and move cursor back.
\$09	HT	Jump to the next 8-cell tab stop.
\$0A	LF	Go to the leftmost column in the next row.
\$0B	VT	Move the cursor to the top left of the screen.
\$0C	FF	Clear the screen.
\$0D	CR	Go to the leftmost column in the next row.
\$1C	FS	Move the cursor one column to the right.
\$1D	GS	Move the cursor one column to the left.
\$1E	RS	Move the cursor one row up.
\$1F	US	Move the cursor one row down.

Note: In SE Basic IV, anything after PRINT CHR\$(12) is not printed.

Expressions can optionally be separated by one or more keywords:

Keyword	Effect
;	Attaches two expressions without any space in between.
SPC(n)	Produces n spaces, where n is a numeric expression.
,	The expression after will be positioned at next available tab stop.
TAB(n)	Moves to column n, where n is a numeric expression.
'	Inserts a newline.

If the print statement does not end in one of these four separation tokens, a newline is printed after the last expression. String expressions can be separated by one or more spaces, which has the same effect as separating by semicolons.

**5.2.53 RANDOMIZE**


---

RANDOMIZE [expr]

Seeds the random number generator with expr.

**5.2.53.1 Parameters** expr is a numeric expression.



**5.2.53.2 Notes**

- For the same seed, SE Basic IV produces different pseudorandom numbers from Microsoft BASIC.
- The random number generator is very poor and should not be used for serious purposes. See **RND** for details.

**5.2.53.3 Errors**

- expr has a string value: Illegal function call.
- The user provides a seed outside [-32768 to 32767] at the prompt: Overflow.

**5.2.54 READ**


---

READ var\_0 [, var\_1] ...

Assigns data from a **DATA** statement to variables. Reading starts at the current **DATA** position, which is the **DATA** entry immediately after the last one read by previous **READ** statements. The **DATA** position is reset to the start by the **RUN** and **RESTORE** statements.

**5.2.54.1 Parameters** var\_0, var\_1 are variables or array elements.

**5.2.54.2 Errors**

- Not enough data is present in **DATA** statements: Out of **DATA**.
- The type of the variable is not compatible with that of the data entry being read: a Syntax error occurs on the **DATA** line.

**5.2.55 REM**


---

{REM|'} [anything]

Ignores everything until the end of the line. The **REM** statement is intended for comments. Everything after **REM** will be stored in the program unaltered and uninterpreted. Apostrophe (') is an alias for **REM**.

Note that a colon (:) does not terminate the **REM** statement; the colon and everything after it will be treated as part of the comment.

**5.2.56 RENUM**


---

RENUM [new|.] [, [old|.] [, increment]]

Replaces the line numbers in the program by a systematic enumeration starting from new and increasing by increment. If old is specified, line numbers less than old remain unchanged. new, old are line numbers; the dot . signifies the last line edited. increment is a line number but must not be a dot or zero.

**5.2.56.1 Notes** The following keywords can reference line numbers, which will be renumbered by **RENUM**: **GOSUB**, **GOTO**, **LIST**, **RESTORE**, **RUN**.

**5.2.56.2 Errors**

- Any of the parameters is not in [0 to 65529]: Syntax error.
- Any of the newly generated line numbers is greater than 65529: Illegal function call. The line numbers up to the error have not been changed.
- increment is empty or zero: Illegal function call.
- old is specified and new is less than or equal to an existing line number less than old: Illegal function call.

**5.2.57 RMDIR**


---

RMDIR dir\_spec

Removes an empty folder on a disk device.

**5.2.57.1 Parameters** The string expression dir\_spec is a valid **filespec** that specifies the path and name of the folder.

**5.2.57.2 Errors**

- dir\_spec has a numeric value: Type mismatch.
- dir\_spec is an empty string: Bad file name .
- No matching path is found: Path not found .

**5.2.58 RUN**


---

RUN [line\_number | app\_name [, p0, p1, ...]]

Executes a program. Existing variables will be cleared and any program in memory will be erased. RUN implies a **CLEAR**. If an app\_name is specified, opens the application.

**5.2.58.1 Parameters**

- line\_number is a valid line number in the current program. If specified, execution starts from this line number. The rest of the RUN statement is ignored in this case.
- The string expression app\_name, if specified, is a valid application name (case-insensitive, truncated to the first 11 characters).
- p0, p1, ... are variables.

**5.2.58.2 Errors**

- line\_number is not a line number in the current program: Undefined line number
- app\_name cannot be found: File not found.

**5.2.59 SAVE**


---

SAVE file\_spec [, "T"]

Stores the current program in a file.

If , "T" is specified, saves a tokenized program.

**5.2.59.1 Parameters** The string expression file\_spec is a valid **filespec** indicating the file to store to.

**5.2.59.2 Notes** In Microsofr BASIC you can append , A to save the file in plain text or , P to save a protected listing, otherwise the file is saved in tokenized format. In SE Basic IV the file is saved in plain text format unless you append , "T".

**5.2.59.3 Errors**

- file\_spec has a number value: Type mismatch.
- file\_spec is an empty string: Bad I/O device.
- file\_spec is too long: Bad I/O device.

**5.2.60 SCREEN**


---

SCREEN [mode] [, [colorburst] [, [apage] [, [vpage] [, erase]]]]

Change the video mode, composite colorburst, active page and visible page.

**5.2.60.1 Parameters** mode is a numeric expression that sets the screen mode.

---

Mode	Notes
0	Text: 80x24 characters. Two attributes picked from 16 colors.
1	Bitmap: 240x192 pixels. 40x24 characters. 8x1 attributes from 16 colors.

---

**5.2.60.2 Notes** The driver for **SCREEN 1** is stored in RAM and can be replaced with a driver for any screen mode supported by the hardware.

**5.2.60.3 Errors**

- No parameters are specified: Missing operand.
- Any parameter has a string value: Type mismatch.
- Any parameter is not in [-32768 to 32767]: Overflow.
- mode is not an available video mode number for your video card setting: Illegal function call.

**5.2.61 SOUND**


---

SOUND frequency, duration

Produces a sound at frequency Hz for duration/18.2 seconds.

**5.2.61.1 Parameters** frequency is a numeric expression in [-60 to 70].

**5.2.61.2 Errors**

- Any argument has a string value: Type mismatch.
- frequency is not in its allowed range, and duration is not zero: Illegal function call.
- duration is zero and more than two arguments are specified: Syntax error.
- frequency is not in [-60 to 70]: Overflow.
- duration is not in [0 to 10]: Illegal function call.

**5.2.62 STOP**


---

STOP

Breaks program execution, prints a Break message on the console and returns control to the user. Files are not closed. It is possible to resume program execution at the next statement using **CONT**.

**5.2.63 TRACE**


---

TRACE {ON|OFF}

Turns line number tracing on or off. If line number tracing is on, BASIC prints a tag [100] to the console when program line 100 is executed, and so forth.

**5.2.63.1 Notes** Tracing is turned off by the **NEW** and **LOAD** statements.

### 5.2.64 WAIT

---

WAIT frames

Pauses for (frames/60) seconds.

**5.2.64.1 Parameters** frames is in [0 to 65535].

**5.2.64.2 Notes** In Microsoft BASIC, WAIT waits for input from a given port.

**5.2.64.3 Errors** Any parameter has a string value: Type mismatch.

### 5.2.65 WEND

---

WEND

Iterates a WHILE-WEND loop: jumps to the matching WHILE statement, where its condition can be checked.

**5.2.65.1 Notes** WHILE-WEND loops can be nested. WEND jumps to the most recent WHILE statement that has not been closed by another WEND.

**5.2.65.2 Errors** All previous WHILE statements have been closed by another WEND or no WHILE statement has been executed before: WEND without WHILE.

### 5.2.66 WHILE

---

WHILE expr

Initiates a WHILE-WEND loop. If expr evaluates to zero, WHILE jumps to the statement immediately after the matching WEND. If not, execution continues.

**5.2.66.1 Parameters** expr is a numeric expression.

**5.2.66.2 Errors**

- No matching WEND is found: WHILE without WEND.
- expr has a string value: Type mismatch.

## 6 Technical reference

text

## 7 Developer's guide

text

## 8 Acknowledgments

---

SE Basic IV 4.2.0 Cordelia. Copyright (C)1981-2023 Source Solutions, Inc.

Based on T/S1000 BASIC by John Grant and Steven Vickers.

Written by Andrew Owen with contributions from Ian Collier, Jonathan Hartson, Richard Kelsh, Slavo Labsky, Jiri LeMac, Ian Logan, Gerton Lunter, Kevin Males, Daniel Nagy, Milan Pikula, Einar Saukas and Andy Wright.

Hardware development by Samuel Baselga, Jordi Bayo, Miguel Angel Rodriguez Jodar, Chris Smith, Don Superfo and Antonio Villena.

Development tools by Juan Carlos Gonzalez Amestoy, Cesar Hernandez Bano, Edouard Berge, Rebecca Bettencourt, Simon Brattel, Miguel Guerreiro, Philip Kendal and Gunter Woigk.

Testing by Eduardo Llana Ugalde and Chris Young.

Documentation adapted from an original work by Rob Hagemans.

Localization with support from Translation Commons and Weblate by Joana Araujo, Mette Hafstad Attar, Baskman, Johanna Behm, Ivan Benishek, Andrea Cebrián, Aiko Chiba, Pia Christensen, Giulia Crotti, Caroline Eira, Arda Erdikmen, Rodrigo Fernandes, Nazar Grymnak, Davide Guida, Ray Juodvirsis, Vessela Kateva, Vasiliki Korfiati, Anestis Koutsoudis, Mike van der Lee, François Lionet, Subi Nga, Wout Plattel, Antonela Prusina, João Diogo Ramos, Alex Semenov, Jad Sinoradzka, Rossella Squillace, Dora Stancu and Tiia Tsurkka.

Patreon support by Sebastian Braunert, Ricardo Calcagno, Gregory Zentkovich, Bruno Silva, Nikolay Ivanchev, John Cook, George Schizas, Thomas Jager, APBDeveloper, John van der Vuurst, Andrew Stephen, Matt Myers, Ramon Martinez Palomares, Benjamin Kleiner, Guy Black, Paul Becker, Robsoft, Manuel Montoto, Gavin Haubelt, Tom Wilson, Charles Stanhope, John Young, Igmarr Palsenberg, Richard Hallas, Alexander Sharikhin, Matthew Butch and Warren Stanley.

## 9 Licenses

text