Appendix B
Error messages

When the computer is unable to continue executing a program or a command it will tell you by printing a message on the screen. As shown in the section on error trapping, these error messages can be suppressed provided you have write an alternative routine for the computer to following using ON ERROR . . .

As well as an error message, the computer sets two variables each time an error occurs:

ERR gives the error number.

ERL gives the lines number at which the error was noticed.

The error messages are listed here in alphabetical order, alongside their error numbers:

Accuracy lost **23**

If you try to calculate trigonometric functions with very large angles you will lose a lot of accuracy in reducing the angle to within the range of plus or minus PI radians. When this happens the computer will print the above message, for example:

```
PRINT COS(111111111)
```

Arguments **31**

This indicates that there are too many or too few arguments for a given function of procedure.

Array **14**

This indicates that the computer expects an array, but cannot find it.

Bad call **30**

Incorrect PROC or FN call.

Bad command **254**

Wrongly typed OS command, for example:

```
*FX20,A
```

Bad DIM **10**

Arrays and memory must be dimensioned with a positive number of elements. For example, these will produce errors:

```
DIM array(-10)
DIM P%-2
```

Bad hex **28**

Hex numbers may only consists of 0 to 9 and A to F.

Bad key **251**

Error in *KEY command, including running out of space for key string, and attempting to re-define a key while it is in use.

Bad MODE **25**

You cannot change mode inside a PROC or FN. Nor can you change to a mode which would make HIMEM less than LOMEM.

Bad program

There are a number of occasions when the computer checks a program to see where it starts and ends in memory. The above error means that the computer could not follow the program successfully and that it is therefore aborted. This error is untrappable, which means that you cannot find out at which line it occurred, nor can you retrieve any part of the program! It is caused by part of the program becoming over-written either by a mode change or by another program's BASIC variables.

Block? **218**

This is an error generated by the cassette filing system. It means that the computer found a non-consecutive block number. Rewind the tape a little way and try again.

Byte **2**

Caused in assembly by trying to load a register in immediate mode with a value greater than 255, for example:

```
LDY #266
```

Can't match FOR **33**

The control variable associated with NEXT is different from that associated with FOR.

Channel **222**

This error is generated by the cassette filing system when you try to use a file channel number which has not been opened.

Data? **216**

This is an error generated by the cassette filing system which means that the computer has missed some data from a block. Rewind the tape a little and try again.

DIM space **11**

There is not enough memory for the array to be dimensioned.

Division by zero **18**

You cannot divide by zero.

$range **8**

Strings may not be placed in the zero-page of memory using the indirection operator $. This is illegal:

```
$&70 = "error"
```

Eof **233**

This error is given by the cassette filing system if the end of file is reached.

Escape **17**

The ESCAPE key has been pressed.

Exp range **24**

You cannot exponentiate a number greater than 88. The following is

illegal:

```
A = EXP 90
```

Failed at ... (Line number)

Caused by renumbering a program with a GOTO or GOSUB to a non-existent line number.

```
15 X = 12
34 GOTO200
48 END
```

will give the error message:

```
Failed at line 20
```

File? 219

This error is generated by the cassette filing system and means that the computer was given an unexpected file name.

FOR variable 34

The control variable in a FOR . . . NEXT loop must be numeric, for example:

```
FOR I$ = 0 TO 20
```

is illegal.

Header? 217

This error is generated by the cassette filing system and it means that the computer cannot read the file's header (which contains the name, block number, etc). Rewind the tape a little way and try again.

Index 3

This error occurs during assembly if you use an incorrect index mode, for example:

```
LDA(&70,Y)
```

LINE space

The computer has run out of memory for you to type any extra lines into a program.

Log range **22**

You cannot find the log of a negative or zero number.

Missing , **5**

This means that the computer expected to find a comma in the line, but didn't do so, for example:

```
C$=LEFT$(Z$)
```

Missing " **9**

This means that the computer expected to find a quote, for example:

```
CHAIN"MARSLANDER
```

Missing) **27**

This means that the computer expected to find a closing bracket, for example:

```
PRINT TAB(6,16
```

Missing # **45**

This means that the computer expected to find a hash, for example:

```
?I% = BGET file
```

Mistake **4**

This means that the computer could not understand the instruction, for example:

```
10 PRIT
```

-ve root **21**

You cannot find the square root of a negative number. This may also occur with ASN and ACS.

No GOSUB **38**

The computer encounters RETURN without having passed a GOSUB.

No FN **7**

The computer encounters the end of a function definition without having passed the DEF FN, for example:

=X

No FOR **32**

The computers encounters NEXT without having passed the FOR.

No PROC **13**

The computer encounters ENDPROC without having passed the DEF PROC.

No REPEAT **43**

The computer encounters UNTIL without having passed the REPEAT.

No room **0**

When a program is running, the computer uses the area of memory between LOMEM and HIMEM to store the BASIC variables. If there is insufficient room to store any more of these variables then the above error is given. This most commonly occurs with programs which use arrays in conjunction with a large screen mode (0, 1, 2 or 3)..

No such FN/PROC **29**

The computer encounters an FN or a PROC for which it can find no definition.

No such line **41**

Electron BASIC does not allow you to GOTO or GOSUB a line number which does not exist.

No such variable **26**

All variables must be declared, either globally by assigning them a value or locally by using LOCAL. If the computer encounters an un-declared variable it gives the above error. This error is also given in assembler when the computer encounters a forward reference to a label.

No TO **36**

TO is omitted from the FOR . . . NEXT loop:

```
FOR I=0
```

Not LOCAL **12**

Local variables may be declared only within an FN or a PROC.

ON range

The control variable for ON GOTO or ON GOSUB is either less than 1 or is greater than the number of entries in the list of line numbers. For example, the following will not work if destination = 3.

```
ON destination 60,120
```

because there are only two destinations. This error may be accounted for by using ELSE.

```
ON destination GOSUB 70,90 ELSE . . .
```

ON syntax **39**

The word ON must be followed either by ERROR, or by a numeric variable and GOTO or GOSUB. The following will give an error:

```
ON direction PRINT
```

Out of DATA **42**

The computer encounters a READ instruction for which it cannot find an entry in the DATA list. RESTORE can be used to move the data pointer back to the start of a DATA list.

Out of range **1**

Branch instructions in assembler can access not farther than 127 bytes forwards or 128 bytes backwards. To branch outside these limits you must use JMP or JSR.

Silly **0**

Given by the automatic line numbering system AUTO or the line renumbering system RENUMBER if you attempt to use a step size of less than 1 or more than 255.

String too long **19**

The maximum length of a string is 255 characters.

Subscript **15**

An array subscript is out of range, either less than 0 or greater than the value declared in DIM.

Syntax **220**

Bad syntax in the cassette filing system.

Syntax error **16**

A statement is incorrectly terminated. For example:

```
LIST. 50
```

Too big **20**

The computer calculates a number which is too big or too small to be represented.

Too many FORs **35**

FOR . . . NEXT loops may be nested to a depth of 10, and the control variables must all be different.

Too many GOSUBs **37**

GOSUB . . . RETURN loops may be nested to a depth of 26.

Too many REPEATs **44**

REPEAT . . . UNTIL loops may be nested to a depth of 20.

Type mismatch **6**

You cannot assign a string to a numeric variable or a number to a string variable.