Title:

RC BASIC
Corrections (no. 1) to the RC BASIC Programming Guide

A/S REGNECENTRALEN

RC SYSTEM LIBRARY: FALKONERALLE 1 DK-2000 COPENHAGEN F

RCSL No: 43-GL 6940
Edition: May 1978
Author: Stig Møllgaard

Keywords:

RC BASIC, corrections, string arrays, LOWBOUND, CALL-routines.

Abstract:

These corrections updates the RC BASIC Programming Guide (RCSL: 42-i 0671) to a point corresponding to rev. 01.11 of the RC BASIC system.

1. INTRODUCTION.

The following pages describes corrections to the first edition of the RC BASIC PROGRAMMING GUIDE (RCSL: 42-i 0671).

Some corrections are due to printing errors, most of the changes are, however, a consequence of new features that have been implemented in the RC BASIC language.

The most important new features are:

- STRING ARRAYS: it is now possible to work with two-dimensional strings.
- LOWBOUND: the lower bound of arrays can be set to zero or one.
- IMPROVED FILE SYSTEM: several users can have write-access to the same logical disc, and one logical disc an be used as a program-library accessable from all terminals.
- CALL ROUTINES: it is now possible to call assembler-coded subroutines from RC BASIC programs.

Most of the corrections are mentioned by a page number and a line-number. A positive line-number means that lines should be counted from the top of the page, a negative line-number means that lines should be counted from the buttom.

With these corrections included, the manual corresponds to rev. 01.11 of the RC BASIC system.

2. CORRECTIONS.

In the list of reserved words on the second page:

word	use	section
CALL	S	App. H
LOWBOUND	C,S	9.10.A

Page 17, line 4:

The lower bound of a dimension is usually 1, it may, however, be set to 0 by means of the LOWBOUND statement (see Ch. 9).

Page 17, line 10:

32 767 $\rightarrow$ 16 380.

Page 22, line -7:

block $\rightarrow$ blocks.

Page 25, line -10, -8, -6:

```
0250   LET MONTHS="OCTOBER"; DAYS=31
0270   LET MONTHS="NOVEMBER"; DAYS=30
0290   LET MONTHS="DECEMBER"; DAYS=31
```

Page 29, line 6 through 9:

2. If the SAVEd program is on disc, the system searches the logical disc to which the terminal is connected for <filename> (see Ch. 8). If <filename>is not found then the system searches for a logical disc called LIB and if this is found then a search is made in this logical disc for <filename>. If <filename> is not found, the system outputs the error message 0100: FILE UNKNOWN.

Page 31, line -3:

more that $\rightarrow$ more than.

Page 33, line -12 through -10:

$$\text{DIM} \left\{ \begin{array}{l} \texttt{<svar>(<l>)} \\ \texttt{<sarray>(<n>,<l>)} \\ \texttt{<array>(<m>)} \\ \texttt{<array>(<row>,<col>)} \end{array} \right\} \left[, \left\{ \begin{array}{l} \texttt{<svar>(<l>)} \\ \texttt{<sarray>(<n>,<l>)} \\ \texttt{<array>(<m>)} \\ \texttt{<array>(<row>,<col>)} \end{array} \right\} \right] \ldots$$

<svar>:	a string variable
<l>:	a numeric expression, which evaluates to the length of a string variable or the length of each element in a string array.
<sarray>:	a string array name.
<n>:	a numeric expression, which evaluates to the number of elements in a string array.
<array>:	an array name.
<m>:	a numeric expression, which evaluates to the number of the last element in a one-dimensional array.
<row>:	a numeric expression, which evaluates to the number of the last row in a two-dimensional array.
<col>:	a numeric expression, which evaluates to the number of the last column in a two-dimensional array.

Page 34, line 1 through 4:

<u>Use</u>

As a statement or command to define explicitly the size of one or more numeric variable arrays, string variables or string arrays. (For the dimensioning of string variables, see Chapter 5).

Page 34, line 15:

1 <= value -> 0 <= value

Page 34, line -8:

Less than 1 -> less than the lower bound
(as defined by means of the LOWBOUND-statement,
see Chapter 9).

Page 35, line -9:

32767 -> 16380

Page 54, insert after line -13:

3.18A: LOWBOUND

For description, see Chapter 9.

Page 92, line -10:

2↑16 -> 2↑15

Page 103, insert after line -13:

5.1.8 String arrays

The array-concept can be used in connection with strings. Dimensioning of a string array is accomplished by means of the DIM statement, (see Chapter 3), for example:

DIM TEXT$(20,40)

The string array TEXT$ consists of 20 string-elements each 40 characters long. The i'th element can be referenced as TEXT$(i). TEXT$(i,j) is a reference to the j'th character in the i'th element, and TEXT$(i,j,k) points out character number j through k in the i'th element of the string array.

The lower bound of a string array (i.e. the number of the first element) is usually 1. By means of the LOWBOUND-statement (see Chapter 9) the lower bound may however be set to zero. The first character in a string element will always be number 1.

Example

```
0010 TAB=4
0020 DIM NAME$(5,8)
0030 LET I=1
0040 REPEAT
0050   READ NAME$(I)
0060   LET I=I+1
0070 UNTIL I>5
0080 FOR I=1 TO 5
0090   PRINT NAME$(I),
0100 NEXT I
0110 DATA "PETER","JOHN","ROBERT","ROBERTA","DIANA"
```

Comment

The 5 names are stored in the string array NAME$

```
PETER   JOHN    ROBERT  ROBERTA DIANA
```

Example | Comment

```
0010 DIM TEXT$(6,5)
0020 FOR I=1 TO 6
0030   LET TEXT$(I)="TEXT",CHR(48+I)
0040 NEXT I
0050 FOR I=6 TO 2 STEP -1
0060   PRINT TEXT$(I)              print string element 6 to 2
0070 NEXT I
0080 FOR I=1 TO 5
0090   PRINT TEXT$(1,I);           print string element 1, cha-
0100 NEXT I                        racter by character
0110 PRINT
0120 PRINT TEXT$(2,3,LEN(TEXT$(2)))      print string ele-
                                   ment 2, from character no. 3.
```

```
TEXT6
TEXT5
TEXT4
TEXT3
TEXT2
TEXT1
XT2
```

Page 108, insert after line 1:

If the lower bound of arrays has been set to zero (by means of the LOWBOUND-statement, see Chapter 9), then MATRIXA will have 11 rows (no. 0 through 10) and 21 columns (0 through 10).

Page 126, line 5:

protection key -> protection key (>0)

Page 126, line -10:

Remarks

1. If the user CONNECT's to a logical disc which has a protection key (< > 0) without specifying this, he may only read from the logical disc.

2. and

3. unchanged.

4. If the protection key of a logical disc is equal to zero, this logical disc can be used by several users at the same time. The users will be able to CREATE, RENAME, read from and write to files on the logical disc, if they do not specify a protection key when CONNECTing. If a file is to be DELETE'd, this can only be done by a user, who is exclusive user of the logical disc. One becomes exclusive user of a logical disc with protection key equal to zero by specifying any protection key not equal to zero when CONNECTing.

Page 127, insert before line 1:

The following table shows which kind of access the user will have depending on the protection key of the logical disc and the key specified in the CONNECT-command.

Key specified in CONNECT-command / Protection key of the logical disc	=0	>0
=0	Write-access: all kind of access is allowed except DELETE	Exclusive use: all kind of access is allowed
>0	Read-access only	Exclusive use: all kind of access is allowed

Page 127, line 14:

user"s -> user's

Page 127, line -11:

1. A file can be copied to a logical disc only if the user has write-access to the files of that logical disc (see sect. 7.2).

Page 130, line -3:

(5-1) x 128 + 80 -> (5-1) x blocksize + 80,
where blocksize is equal to 512 if the device is a moving-head disc file and 128 if the device is a flexible disc.

Page 135, line 13 and
Page 137, line -16 and
Page 149, line 2:

Replace "correctly specified the protection key of the logical disc in the CONNECT command" by
"has write-access to the files of the logical disc to which the terminal is connected"

Page 135:

Replace lines -5 through -3 by
A discfile can only be CREATEd, RENAMEd or written into if the user has write-access to the logical disc (see Chapter 7, CONNECT).
A discfile can only be DELETEd if the user is exclusive user of the logical disc.

Page 138, line -5 through -3:

1. A file can be deleted only if the user is exclusive user of the logical disc containing the file to be deleted (see Chapter 7, CONNECT).

Page 143, line 5:

mode 0 or 3 -> mode 0, 2 or 3.

Page 149, line -13:

mode 0 or 3 -> mode 0, 2 or 3.

Page 160, line -8 and
Page 170, line -3:

disc file -> disc file (only possible if the user has write-access to a logical disc)

Page 161, insert after last line, and
Page 170, insert after line 5:

3. If the SAVEd program is on disc, the system searches the logical disc to which the terminal is connected for <filename> (see Chapter 8). If <filename> is not found then the system searches for a logical disc called LIB and if this is found then a search is made in this logical disc for <filename>. If <filename> is not found, the system outputs the error message 0100: FILE UNKNOWN.

Page 162, insert before line 5:

9.10A LOWBOUND

Format

LOWBOUND=<expr>

<expr>: a numeric expression which evaluates to 0 or 1

Use

As a command or statement to change the lower bound of numeric and string arrays.

Remark

1. The default lower bound is 1 (i.e. the number of the first element in an array is one). By means of the LOWBOUND-statement the lower bound may however be set to zero.
2. If lowerbound is zero, the row and column no. zero is included in all matrix-operations.
3. The LOWBOUND-statement can be placed anywhere in an RC BASIC program. Each time an array element is referenced the current lower bound will be taken as the first element of the array.

Example	Comment
`0010 LOWBOUND =1`	
`0020 DIM A(5)`	
`0030 FOR I=1 TO 5`	
`0040   LET A(I)=I`	
`0050 NEXT I`	
`0060 FOR I=1 TO 5`	lower bound of A-array is 1
`0070   PRINT A(I);`	
`0080 NEXT I`	
`0090 LOWBOUND =0`	
`0100 DIM B(5)`	
`0110 PRINT`	
`0120 FOR I=0 TO 4`	now lower bound of A-array is 0
`0130   PRINT A(I);`	
`0140 NEXT I`	
`0150 PRINT`	
`0160 FOR I=0 TO 5`	B-array has 6 elements (0 to 5)
`0170   PRINT B(I);`	
`0180 NEXT I`	

```
1  2  3  4  5
1  2  3  4  5
0  0  0  0  0  0
```

Page 165, line 3, 8 and 9:

NUL -> STX

New error messages (page 176-187).

0047: PARAMETER ERROR

The actual parameters used in a call of an assembler-coded subroutine does not correspond to the formal parameters as specified in the subroutine.

0090: USER CALL ERROR 1

0091: USER CALL ERROR 2

These messages can be used by the programmer of assembler-coded subroutines if an error is detected during the execution of the subroutine.

0098: PAGING ERROR.

This error will only occur on a system running under the DOMUS-operating system. The reason is, that (part of) the virtual storage can not be read from the disc. The user's current program is destroyed. If the error occurs regularly the reason is probably malfunctioning hardware.

0099: STACK OVERFLOW.

A stack overflow may occur, if the user has recursive functions or procedures in his program, for instance 10 DEF FNA(X) = FNA(X). If this is not the reason, then please contact RC.

Page 189, line 5:

Disc write-protected -> Skip block

Page 211:

Insert LOWBOUND as a reserved word.

Page 212, insert after line 4:

$$\text{CALL} \left\{\begin{array}{l}\text{<svar>}\\ \text{<slit>}\end{array}\right\} \left[, \left\{\begin{array}{l}\text{<var>}\\ \text{<svar>}\\ \text{<mvar>}\\ \text{<sarray>}\\ \text{<slit>}\\ \text{<expr>}\end{array}\right\} \right] \ldots$$

H

STATEMENT

Invokes the execution of an assembler-coded subroutine.

Page 213, line 4 through 8

$$\text{DIM} \left\{\begin{array}{l}\text{<svar>,(<l>)}\\ \text{<sarray> (<n>,<l>)}\\ \text{<array> (<m>)}\\ \text{<array> (<row>,<col>)}\end{array}\right\} \left[, \left\{\begin{array}{l}\text{<svar> (<l>)}\\ \text{<sarray> (<n>,<l>)}\\ \text{<array> (<m>)}\\ \text{<array> (<row>,<col>)}\end{array}\right\} \right] 3.7$$

Defines the size of string variables, string arrays or numeric arrays.

STATEMENT or COMMAND

Page 225, insert after line -5:

LOWBOUND=<expr>

Sets the lower bound of arrays to 0 or 1

9.10A

COMMAND or STATEMENT

Page 229:

H Calling an Assembly Language Subroutine from RC BASIC.

<u>Format</u>

$$\text{CALL <subr.name>} \left[\; , \left\{ \begin{array}{l} \text{<var>} \\ \text{<svar>} \\ \text{<mvar>} \\ \text{<sarray>} \\ \text{<slit>} \\ \text{<expr>} \end{array} \right\} \right] \ldots\ldots$$

<subr.name>:	the name of an assembler coded subroutine expressed as a string literal or by means of a variable.
<var>:	a numeric variable
<svar>:	a string variable
<mvar>:	a matrix variable
<sarray>:	a string array
<slit>:	a string literal
<expr>:	a numeric or relational expression

<u>Use</u>

As a statement to invoke the execution of an assembler-coded subroutine included in the system.

<u>Remarks</u>

1. When the CALL-statement is executed the following happens:
   a. If a module containing user-coded subroutines is present in core, then the subroutine table in this module is searched for the name of the subroutine. If the name is found operation continues at point c.
   b. As a. except that the searching is carried out in the module containing subroutines coded by RC. If the subroutine is not found then the BASIC program is interrupted with error no. 0046: PROCEDURE DOES NOT EXIST.

c. Now the number and the type of the actual parameters are checked against the parameter specifications in the subroutine. If a conflict is found then the BASIC program is interrupted with error no. 0047: PARAMETER ERROR.

d. The actual parameters are organized in a way which makes it possible to access them from the subroutine and then a jump is made to the subroutine.

2. When the subroutine is terminated, return to BASIC can occur in two different ways. In case of normal return execution of the BASIC-program continues from the statement following the CALL-statement. If an error has occured during execution of the subroutine, the BASIC-program will be interrupted (unless an ON ERR-statement has been executed), and an error message is printed.
The seperate publication
   Assembler Coded Subroutines (CALL-routines)
   in RC BASIC (RC3600/RC7000)
   Programmer's Guide
describes how to program subroutines to be called from RC BASIC.

Example

```
0010  DIM A(20), B$(3)
0020  CALL "PUSH", A, 3
0030  B$ = "POP"
0040  CALL B$, A, X
```

The two routines "PUSH" and "POP" are shown as examples in the publication referenced above.

Corrections to the Index (page 230 - 239):

Page 231:

CALL, H

Page 234:

LIB, 3.3, 9.10, 9.16

Logical discs

- write access, 7.2

LOWBOUND, 2.4, 3.7, 5.1, 6.2, 9.10A

Page 238:

Strings

- arrays 3.7, 5.1, 9.10A