Colour-Compiler

Written by Hallo Braun

****

****

APPENDIX:

What is the COLOUR-COMPILER

This manual assumes that you know how to write BASIC programs on your Colour-Genie.

COLOUR-COMPILER is an interactive compiler, i. e. the user can jump between the "source code" (your BASIC program) and the compiled "object code" (machinelanguage). Though it is very simple to change a program and recompile it.
The process of compiling a BASIC program translates your BASIC text into machine language. This is equal to the things your BASIC interpreter does, but it translates one BASIC statement, then executes it. The COLOUR-COMPILER translates the whole program, so it is executed much faster (there is nothing to translate anymore). This compilating increases the speed of your programs 20-200 times.

*** IMPORTANT: ***

COLOUR-COMPILER supports most of the BASIC statements given by Colour-BASIC, but some words are not supported. Please read this manual to know what statements are allowed.

****

Programs compiled by COLOUR-COMPILER don't support "active" commands like BREAK, CLEAR, LIST, RUN, EDIT, AUTO, CONT, LLIST, CLOAD, CSAVE, VERIFY, RENUM, SYSTEM etc. This means: only y o u possess the source code, which makes it easy to change the program, other persons can not change it.

All math operations are performed as integers, though you can only use numbers in the range from -32768 to 32767. This makes programs shorter and much faster.

****
If needed, you can simulate real arithmetic using special machine language subroutines.
****

Attention: Some BASIC statements have different results before and after compiling them (See page 5).
PLEASE READ THIS MANUAL CAREFULLY!!!

COLOUR-COMPILER DOES NOT TEST FOR RUN-TIME-ERRORS.
Because of this: Test your BASIC program carefully before compiling it. If it runs in BASIC it will run after compiling it, too.)
If COLOUR-COMPILER has to find run-time-errors, your programs speed would be decreased and its length increased!!

How to break out of COLOUR-COMPILER programs

The <BREAK> key is not tested by your compiled programs. To leave a compiled program, insert the following statement in your program (in a loop or after an INPUT statement):

```
<lineno.> IF PEEK(-1984)=4 THEN STOP
```

If the <BREAK> key is pressed, when this line is executed, the program will return to the Colour-BASIC READY.

EXAMPLE:

```
10 FOR X=17408 TO 18367:POKE X,65
20 IF PEEK(-1984)=4 THEN STOP
30 NEXT X:END
```

****
At the end of your BASIC program must be an END or STOP statement to return to READY. If there is no such statement, the only way to stop your program is to press both RST keys. This may destroy your program!!!!
****

COLOUR-COMPILER commands

```
**********************************************************************
* Please use this manual in connection with your BASIC               *
* reference manual.                                                  *
**********************************************************************
```

BASIC statement	Description
ABS(argument)	Same as BASIC
AND	Same as BASIC, but works only as maths operator. In IF/THEN statements use IF A=B THEN IF B=C THEN ... instead of IF A=B AND B=C THEN ...
ASC(var$)	Same as BASIC
BGRD	Same as BASIC
CALL hexno.	Same as BASIC
CHAR argument	Same as BASIC
CHR$(argument)	Same as BASIC
CIRCLE x,y,r	Same as BASIC
CLEAR	Is ignored by COLOUR-COMPILER (See page 11, too)
CLS	Same as BASIC
COLOUR argument	Same as BASIC
CPOINT(x,y)	Same as BASIC
DATA number,number	Same as BASIC, but strings are not allowed.
END	Returns to BASIC's READY.
FCLS	Same as BASIC
FCOLOUR argument	Same as BASIC
FGR	Same as BASIC
FILL argument	Same as BASIC
FIX(argument)	Same as BASIC, but unnecessary
FOR	Same as BASIC
GOSUB	Same as BASIC
GOTO	Same as BASIC
INKEY$	Same as BASIC

INP(argument)	Same as BASIC
INPUT var,var INPUT var$	Same as BASIC. String INPUT accepts all characters including hyphens, commas and colons. INPUT A$,B$ is not allowed. A number in the range from 32768 to 65535 is converted into the corresponding negative number: (65536 = -1). Up to 240 characters can be entered at one time.
INT(argument)	Same as BASIC
JOY arg.,direction	Same as BASIC
KEYPAD argument	Same as BASIC
LEN(var$)	Same as BASIC, but LEN(A$+B$) is not allowed
LET	Same as BASIC (See page 10, too)
LGR	Same as BASIC
LPRINT	Same as BASIC
NBGRD	Same as BASIC
NEXT var	Same as BASIC
NOT(argument)	Same as BASIC
NPLOT x,y	Same as BASIC
NSHAPE x,y	Same as BASIC
ON GOSUB	Same as BASIC
ON GOTO	Same as BASIC
OR	Same as BASIC. In IF/THEN, replace IF A=B OR A=C THEN .... by IF A=B THEN ... ELSE IF A=C THEN ...
OUT arg.,arg.	Same as BASIC
PAINT x,y,f1,f2	Same as BASIC
PEEK(argument)	Same as BASIC
PLAY(ch,oct,not,amp)	Same as BASIC
PLOT x,y	Same as BASIC
POKE arg.,arg.	Same as BASIC
POS(dummy)	Same as BASIC
PRINT (PRINT@)	Same as BASIC

RANDOM	Same as BASIC
READ var,var	Same as BASIC. Strings are not allowed
REM (or ')	Same as BASIC, REMs are not compiled
RESTORE	Same as BASIC
RETURN	Same as BASIC
RND(argument)	Same as BASIC
SCALE argument	Same as BASIC
SGN(argument)	Same as BASIC
SHAPE x,y	Same as BASIC
SOUND x,y	Same as BASIC
SQR(argument)	Same as BASIC, result is integer
STEP number	Same as BASIC, variable steps are not allowed
STOP	Same as BASIC
STR$(argument)	Same as BASIC
TO	Same as BASIC
USR(argument)	Same as BASIC
VAL(var$)	Same as BASIC
XSHAPE x,y	Same as BASIC

Math operators

Operator	Description
+ (addition)	1 + 1 = 2
- (subtraction)	1 - 1 = 0
* (multiplication)	2 * 2 = 4
/ (division)	10 / 5 = 2

precedence of the operators:
/, *, +, -, NOT, AND, OR
(Use parentheses for other precendences)

All math operations are performed integer. An overflow is ignored:
16384 + 16384 = -32768

```
  32767 is binary 0111111111111111
 -32768 is binary 1000000000000000

  32767 + 1 = -32768
  65535     = -1
```

(see page 3, too)

IF/THEN logical operators and conditions

Operator	Description
IF/THEN	Same as BASIC
	1. AND and OR are not allowed
Please change --->	IF X=1 AND Y=2 THEN 200
to --->	IF X=1 THEN IF Y=2 THEN 200
Please change --->	IF X=1 OR Y=2 THEN 200
to --->	IF X=1 THEN 200 ELSE IF Y=2 THEN 200
	2. Be careful with logical arguments
Please change --->	IF PEEK(-1984) AND 64 THEN 200
to --->	IF (PEEK(-1984)AND64)=64 THEN 200
In BASIC:	0 = wrong, everything else = true
COLOUR-COMPILER	Operators have to be used
	3. Allowed conditions:
	=, >, <, <=, >=, <>
	Not allowed:
	=>, =<, ><
	4. No math operators in string-IFs:
* WRONG *	IF A$+B$=C$ THEN 200
* OK *	IF A$=B$ THEN 200
* OK *	IF A$>B$ THEN 200
* OK *	IF A$<B$ THEN 200
* OK *	IF "HELLO"=A$ THEN 200
* OK *	IF A$="HELLO" THEN 200
* WRONG *	IF INKEY$=A$ THEN 200
* OK *	B$=INKEY$:IF A$=B$ THEN 200
ELSE	Same as BASIC

COLOUR-COMPILER and stringfunctions

Strings are stored in the following way:
A$="HELLO"
A$ now uses 5 bytes for the text and one termination byte 00H.

For each string 31 text bytes and one termination byte are reserved, but:
If you don't use B$, A$ can use 63 textbytes.
If you don't use B$ and C$, A$ can use 95 textbytes.
If you don't use B$, C$ and D$, A$ can use 127 textbytes
and so on.

Because of this memory management, take care while adding strings:
A$=A$+B$ * OK *
A$=B$+A$ * WRONG !!! *

Compilation of BASIC programs

Turn on your Colour-Genie and press <RETURN>. Insert the tape into your recorder, rewind it if necessary and press PLAY. Enter SYSTEM and press <RETURN>. The computer will print a '*?' and the blinking cursor. Now enter C and press <RETURN> again. When the computer has loaded the whole program, another '*?' will appear. Now enter '/' and press <RETURN>. The screen will be cleared and the computer will print:

```
TCG-COLOUR-BASIC-COMPILER
COPYRIGHT (C) 1982 BY TCG

START COMPILATION WITH "NAME"
(OR PRESS "F3")
```

Now you can load and test your BASIC programs without affecting the compiler. ATTENTION: The compiler will leave 9 kbytes of memory for your programs, longer programs will cause an Out of Memory Error to appear.

If your programs produces no errors while running under the BASIC interpreter (use a DEFINT a-z at the beginning of your program to ensure that the results are the same as the results of the compiled version) and you are sure that you haven't used any prohibited commands (see page 5 ff.) you may compile it!!

```
* Is a 'BREAK' in your program? IF PEEK(-1984)=4 THEN STOP *
-----------------------------------------------------------
Save your BASIC program on tape!
-----------------------------------------------------------
```

In order to compile your program enter NAME and press <RETURN> or simply press the <F3> key.
The COLOUR-COMPILER now translates your BASIC program into machine language, this will take a few seconds.
If the COLOUR-COMPILER finds any errors during compilation, it will generate an errormessage and return to BASICs READY.
If there are no errors found, the COLOUR-COMPILER will print the following message:

```
(S) START COMPILED PROGRAM
(B) BACK TO BASIC
(T) SAVE COMPILED PROGRAM ON TAPE
```

Now press the 'S' key to start your program, the 'B' key to return to BASICs READY or the 'T' key to save the compiled version of your program on tape. The computer will ask you for the name of your program. Enter maximal 6 characters, the first has to be a letter, the others letters or numbers. Then the following message appears:
*** READY TAPE ***. Insert a tape in your recorder, wind it to the right position and press PLAY and RECORD. Now press the <RETURN> key. When your program has been saved, the COLOUR-COMPILER returns to the main menue.

To load a compiled program, enter SYSTEM <RETURN>, then enter the corresponding filename and press <RETURN>.

No BASIC program in memory

Reasons:

The compiler has been started when there is no BASIC program in the memory.

Syntax error in line xxxxx:

Reasons:

The compiler can't understand a word

Examples:

```
10 PRUNT X
10 X=(Y+)2-1)))
10 X="HALLO"
10 X$=5*1
10 X=1+
       2 : REM No linefeeds!!!!!!
10 x=5*2 : REM No lowercase letters !!!!
10 X=5+1 : REM Seems to be ok, but there's a
                 backspace hidden.
10 RUN : REM Not allowed statement
```

Illegal FOR/NEXT nesting in line xxxxx:

Reasons:

FOR without NEXT
NEXT without FOR
NEXT without variable: FOR X=1 TO 10:NEXT <-- X!!
Illegal use of STEP:

```
   Please change ---> FOR X=1 TO 10 STEP Z
   to            ---> FOR X=1 TO 10 STEP 5
```

There must be one and only one NEXT for each FOR:

```
   Please change ---> 10 FOR X=1 TO 10: IF X=9 THEN
                         NEXT X
                      20 NEXT X
   to            ---> 10 FOR X=1 TO 10: IF X=9 THEN
                         20
                      20 NEXT X
```

Undefined lineno. in line xxxxx:

Reasons:

A lineno. 0 in Pass 2:

```
  10 GOTO
  20 STOP
```

causes an 'Undefined lineno. in line 10'

If a lineno. doesn't exist, the error is listed with a wrong lineno. in pass 3:

```
  10 GOTO 50
  20 STOP
```

causes an 'Undefined lineno. in line 4

Check all THENs, ELSEs, ON GOTOs and ON GOSUBs for missing or wrong linenumbers. A good way to do this

is to renumber your program using the RENUM command

Wrong variable in line xxxxx:

Reasons:

A variable named other than A-Z, A1-Z1, A2-Z2 or A$-Z$.
Using A instead of A$ or A$ instead of A.

Out of memory in line xxxxx:

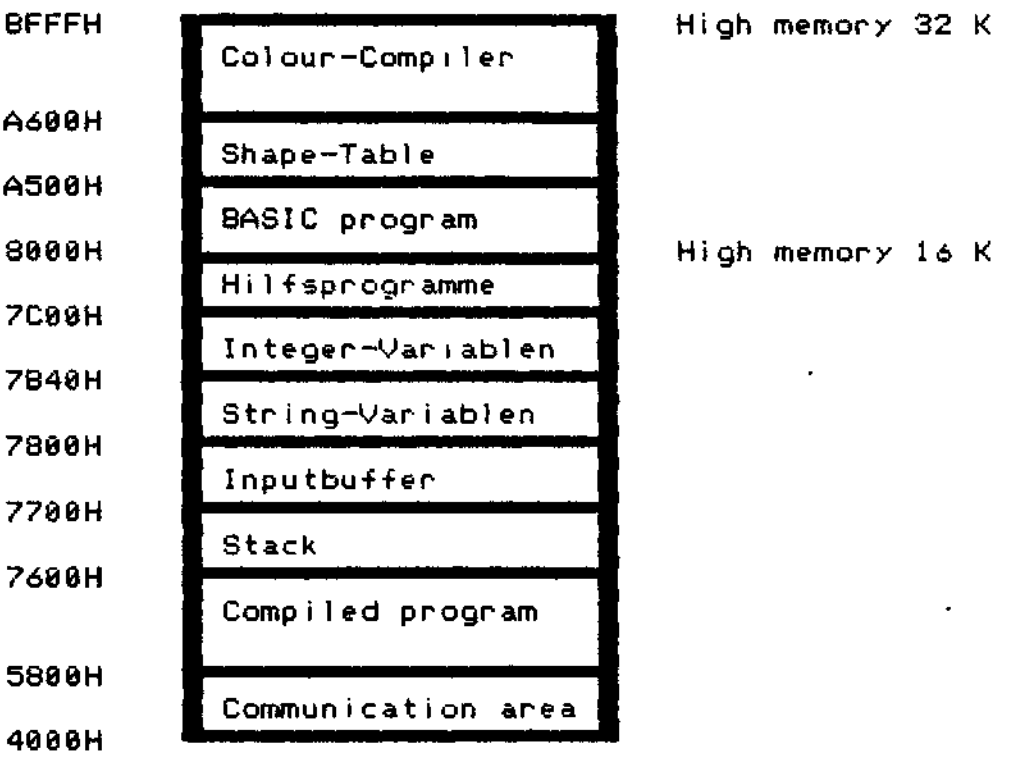
Reasons:

Your BASIC program is too long.
Try the following methods to shorten it:

1. Shorten texts (1 byte per character)
2. Use more subroutines
3. Delete useless lines of your program

The compiler doesn't test for any errors after compiling your program. We recommend to save your program before compiling it. Be careful in using POKE. Be sure, where you POKE!!

Variable	A$-Z$	A-Z	A1-Z1	A2-Z2
A	30720	31552	31616	31680
B	30752	31554	31618	31682
C	30784	31556	31620	31684
D	30816	31558	31622	31686
E	30848	31560	31624	31688
F	30880	31562	31626	31690
G	30912	31564	31628	31692
H	30944	31566	31630	31694
I	30976	31568	31632	31696
J	31008	31570	31634	31698
K	31040	31572	31636	31700
L	31072	31574	31638	31702
M	31104	31576	31640	31704
N	31136	31578	31642	31706
O	31168	31580	31644	31708
P	31200	31582	31646	31710
Q	31232	31584	31648	31712
R	31264	31586	31650	31714
S	31296	31588	31652	31716
T	31328	31590	31654	31718
U	31360	31592	31656	31720
V	31392	31594	31658	31722
W	31424	31596	31660	31724
X	31456	31598	31662	31726
Y	31488	31600	31664	31728
Z	31520	31602	31666	31730


BFFFH
Colour-Compiler
A600H
Shape-Table
A500H
BASIC program
8000H
Hilfsprogramme
7C00H
Integer-Variablen
7B40H
String-Variablen
7800H
Inputbuffer
7700H
Stack
7600H
Compiled program
5800H
Communication area
4000H
High memory 32 K
High memory 16 K

1. Don't add stringvariables to itself:
   A$=A$+B$ * OK *
   A$=B$+A$ * WRONG *
2. AND/OR/NOT are not allowed in IF/THEN statements

3. CLEAR, DEFINT and REM are ignored by the compiler

4. INPUT"text";var does not work. Use PRINT"text";:INPUT var instead

5. INPUT A$,B$ does not work. B$ is ignored.

6. INPUT var$ accepts all characters.

7. READ A$ or DATA "TEST" are not allowed!

8. Variablenames are: A-Z, A1-Z1, A2-Z2 and A$-Z$. All numbers are integer

9. ON ERROR GOTO is not allowed

10. USR(X) passes the value of X to the HL register. It's not necessary to call special ROM routines to do this.

11. VARPTR is not supported. Use the given variable addresses

12. Use INT(argument) to ensure that BASIC produces the same result as the compiled program

13. All numbers are stored as two bytes. The first address contains the remainder of the division by 256, the second address contains the quotient.

14. All strings are stored as characters followed by a byte 00H. Each string may be 31 characters long. If you only use A$, it can consist of up to 831 characters! If you don't use B$, C$ and D$, A$ can be up to 127 characters long!

15. The inputbuffer can contain up to 240 bytes.

16. Errors occuring in pass 3 don't tell you the right lineno.

17. If you don't use some string variables you store little machine language routines at this addresses.

18. Entering numbers bigger than 31767 produces negative numbers: 32768 produces -32768, 65535 produces -1!!!

19. In order to input floating-point-numbers, use the following trick:
    10 INPUT X,Y
    If you enter 1234.56789, 1234 is assigned to X and 56789 is assigned to Y.

```
************************ LEFT$(A$,X) ****************************
After calling this subroutine B$ equals LEFT$(A$,X)

9000 REM M1 is the address of A$: 30720
9010 REM M2 is the address of B$: 30752
9020 L=LEN(A$): IF L<X THEN B$=A$:RETURN
9030 B$="": FOR I=0 TO X-1: POKE M2+I,PEEK(M1+I):NEXT I: POKE
M2+I,0:RETURN

************************ RIGHT$(A$,X) ***************************
After calling this subroutine B$ equals RIGHT$(A$,X)

9000 REM M1 and M2 see above
9010 L=LEN(A$): IF L<X THEN B$=A$:RETURN
9020 B$="": FOR I=0 TO L-X: POKE M2+I,PEEK(M1+L-X+I): NEXT I:
POKE M2+I,0:RETURN

************************ MID$(A$,X,Y) ***************************
After calling this subroutine B$ equals MID$(A$,X,Y)

9000 REM M1 and M2 see above
9010 L=LEN(A$): IF L<X+Y THEN B$=A$:RETURN
9020 B$="":FOR I=Y TO Y+X: POKE M2+I-Y,PEEK(M1+I):NEXT I:POKE
M2+I-Y,0:RETURN

************************ STRING$(X,Y) ***************************
After calling the subroutine A$ equals STRING$(X,Y)

9000 REM X=Length, Y=ASCII code, M=Address of A$ (30720)
9010 FOR I=0 TO X-1: POKE M+I,Y:NEXT I:POKE M+I,0:RETURN

****************** one-dimensional array ************************
One-byte array with 100 elements

9000 REM E=array index, X=value to store
9005 REM V=value found, M=start of array
9010 IF E<0 THEN RETURN ELSE IF E>100 THEN RETURN
9020 V=PEEK(M+E)
9030 POKE M+E,X
9040 RETURN

****************** two-dimensional array ************************
One-byte array with 20 * 20 elements

9000 REM M=start of array, X=first dimension index, Y=second
9005 REM dimension index, Z=value to store, V=value found
9010 IF X>20 THEN RETURN ELSE IF Y>20 THEN RETURN ELSE IF X<0
THEN RETURN ELSE IF Y<0 THEN RETURN
9020 V=PEEK(M+X+Y*20)
9030 POKE M+X+Y*20,Z
9040 RETURN

********************** Cosine or Sine ***************************
This program computes 1000 * SIN(X) or 1000 * COS(X)

5 DEFINT A-Z
10 PRINT"ANGLE";:INPUTX
11 IF PEEK(-1984)=4 THEN STOP
15 Z=X:GOSUB 9010
```

```
20 PRINT "SIN(X)*1000 =";X
25 X=Z:GOSUB 9005
27 PRINT "COS(X)*1000 =";X
30 GOTO 10
9005 X=X+90
9007 IF X>359 THEN X=X-360: GOTO 9007
9010 S=1:IF X>179 THEN X=X-180:S=-1
9030 IF X>89 THEN X=180-X
9040 IF X>45 THEN 9080
9050 X=174*X/10:R=X/10
9060 X=X-R*R/200*R/30+R*R/200*R/100*R/250*R/240
9070 X=X*S:RETURN
9080 X=90-X
9090 X=174*X/10:R=X/10
9100 X=1000-R*R/20+R*R/200*R/10*R/120
9110 X=X-R*R/200*R/100*R/250*R/200*R/720
9120 X=X*S:RETURN
```