

# ColorForth 2.0a TUTORIAL

## Revision 1.0

02/08/08

This tutorial is designed for users with little or no experience with ColorForth and very limited experience with Forth. It describes how to launch and use some of the tools available by walking through examples. It may be helpful to learn some basic editor commands before beginning the tutorial (refer to section 3 of Users' Guide for ColorForth2 *colorForth Human Interface*). In the following examples, commands and editor control panel keys are shown in **bold** while qwerty keys are shown in **bold underline**.

**Warning:** While referring to the examples while in the editor command mode, if a single character is in bold it will sometimes represent a control panel command and typing the qwerty character will not produce the expected result. For example the editor command **I** (left ) is mapped to the qwerty key **j**. This may sound a little confusing but will become clear after doing a few examples.

Below is a copy of the editor command mode key mapping taken from the Users' Guide for ColorForth2. The commands appear in the lower right corner of the screen while in editor command mode.

<b>Q</b>	<b>W</b> s	<b>E</b> c	<b>R</b> t		<b>U</b> y	<b>I</b> r	<b>O</b> g	<b>P</b> *
<b>A</b>	<b>S</b> d	<b>D</b> f	<b>F</b> j		<b>J</b> l	<b>K</b> u	<b>L</b> d	<b>;</b> r
<b>Z</b>	<b>X</b>	<b>C</b>	<b>V</b>	<b>N</b> x	<b>M</b> -	<b>_</b> m	<b>^</b> c	<b>/</b> +
				<b>space</b> .	<b>alt</b> i			

## 1. ColorForth configuration

The following examples show how to use the interpreter, editor, and some configuration utilities. The colorForth editor is a modal editor consisting of a command mode and text entry modes for several colors of text.

### 1.1 First ColorForth function

In Forth, functions are referred to as words. For the first example, the word "2 3 +" will be entered. If a typing mistake is made while in the editor, backspace will remove the text currently being typed. To remove text that has already been entered into a block the editor command, **x** (mapped to **n**), is used. For more editor commands refer to section 3 of the Users' Guide for

ColorForth2. While in the interpreter, backspace will remove the current text and **c** can be used to clear the stack if too much debris collects at the bottom of the screen.

Enter the editor.

**e**

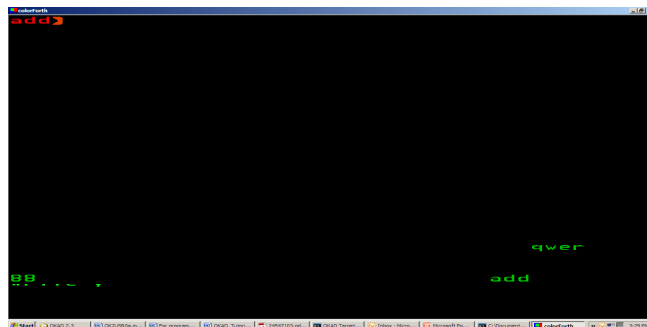


Block 18 is the first block loaded when OKAD is launched. It is also the block that comes up in the editor when no block number is specified. The 18 in the lower left hand corner of the display indicates that the editor is in block 18. The editor control panel map is in the lower right corner and the last word typed, **e**, is displayed just to the left of the control panel map. The last text entered in interpretive mode will be displayed in the same location.

To search for an empty block, use the forward block command **+** ( mapped to **?**). This command is the first character from the right in the third line of the control panel map.

After an empty block is located, select the **r** (mapped to the **i**) on the top of the editor control panel to enter a red word. Red is the color used for word names in colorForth. Enter the name of the word.

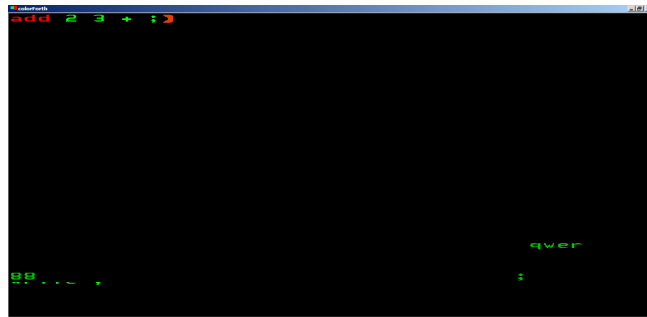
**add**



Notice that after the red word is entered the font changes to green. Green is used to call colorForth words and macros.

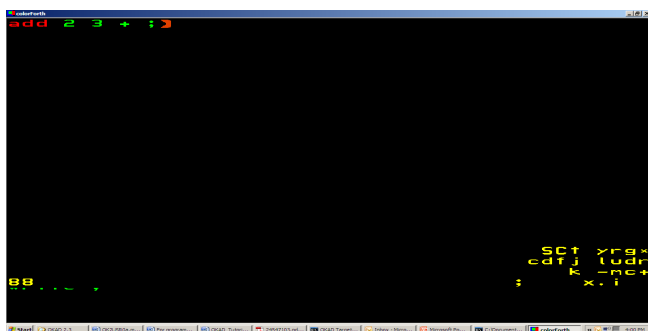
Enter the equation to be executed when the word is called.

**2 3 + ;**



The semicolon is used to end a word. If one is not placed at the end of a word and code exists below, the word will continue to run until a semicolon is reached. If a word has no semicolon at all it will not run. Exit green text entry and return to the editor controls by hitting the escape key.

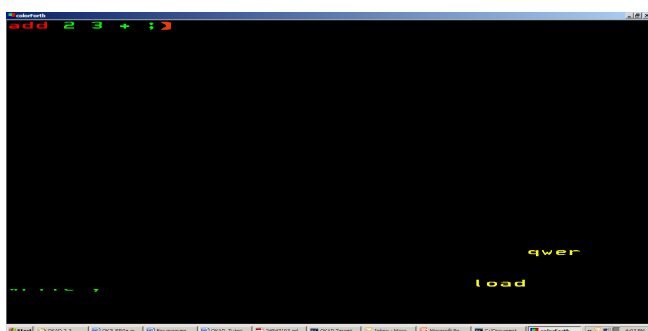
## esc



In order to run this word the block that contains the definition must be loaded. Only Forth words from blocks that have been loaded can be run. In this example the word was placed in block 88. Exit the editor and load the block that contains **add**.

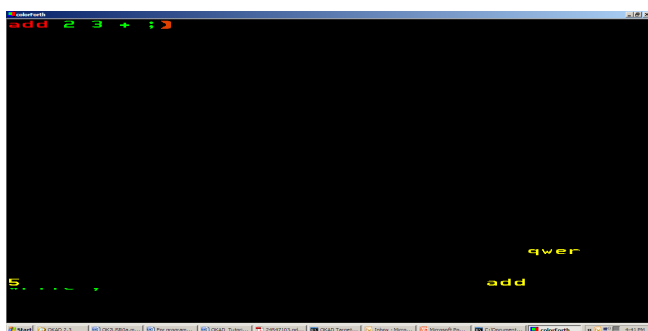
.

## 88 load



Execute the word.

## add



Five appears on the top of the stack in the lower left hand corner.

## 1.2 Accessing memory.

This example shows how to store and retrieve data from memory. In colorForth memory can be accessed by numeric addresses or variable names. ColorForth commands are available to convert blocks and words into cell addresses. A cell is 32 bits long. Variables are displayed as cell addresses as soon as their names are entered on the stack. Magenta is the color used for variable names in colorForth.

An empty block is needed for this example. Use the block from example 1.1 or enter the editor and use **+** to search for another empty block.

**e**

**+** (until empty block is located.)

Once a block is located select text “**t**” (mapped to **e**) from the editor control map. The editor command **t** will place the editor in white text entry mode. In colorForth white words are comments.

**t**

**vtest**

**esc**

Select magenta and enter 2 variables, var1 and var2.

**m**

**var1**

**var2**



As mentioned in the previous example, the block containing a command or variable must be loaded before the command or variable can be used. If we try to access var1 or var2 without loading the block in which it resides, the variable will not be available. Exit the editor and try to store 28 in var1.

**esc**

**.**

**28 var1 !**

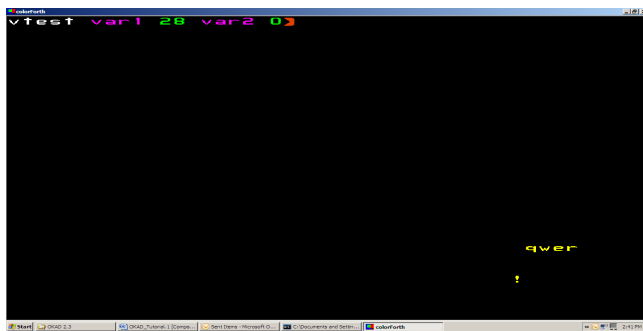


The interpreter will only push valid data onto the stack. “28” is pushed on the stack but “var”1 is not. The question mark following “var1” indicates that it has not been found by the interpreter. This is because block 96 has not been loaded. When “!” is entered a question mark will appear next to it because there is no place to store “28”.

Load the block that contains `var1` and `var2` and try again. When the name `var1` is pushed on the stack, it is converted to a cell address.

**96 load**

**28 var1 !**



As can be seen on the display var1 now contains the value 28.

The address can be fetched from var1 by accessing the variable directly or using cell offsets.

**var1** @

or

**96 block 2 + @**



“28” is displayed on the stack in the lower left of the display.

The colorForth command, **block** converts the block number to a cell address. Two is then added to this memory location and **@** fetches the contents of this address. This shows that the value of **var1** is stored in the second cell address from the start of the block.

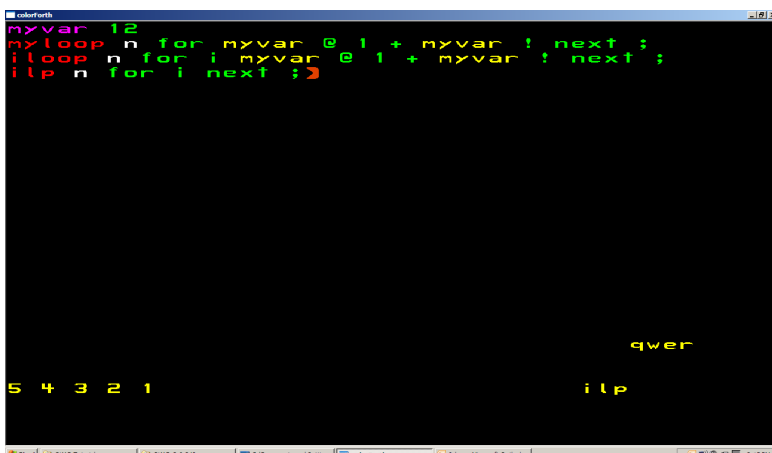
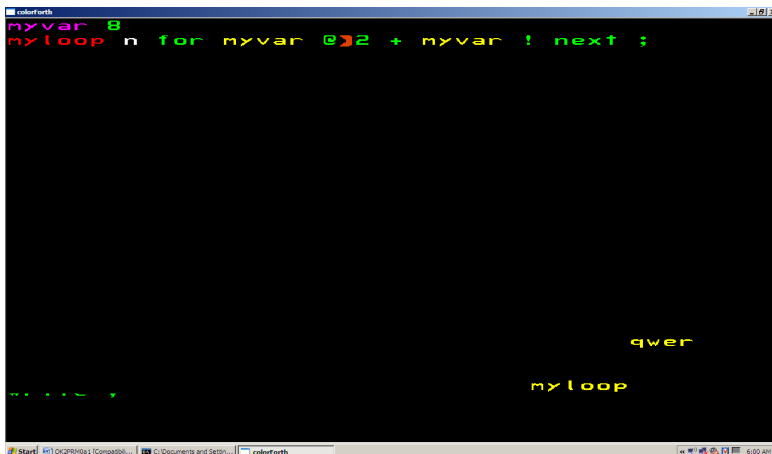
To verify that “**96 block 2 +**” is **var1**, both numbers can be displayed on the stack.

**96 block 2 + var1**



The stack representation in the lower left shows that these locations are identical. You can use similar commands to verify that **var2** is two cells past **var1**.

### 1.3 Loop example (WIP)

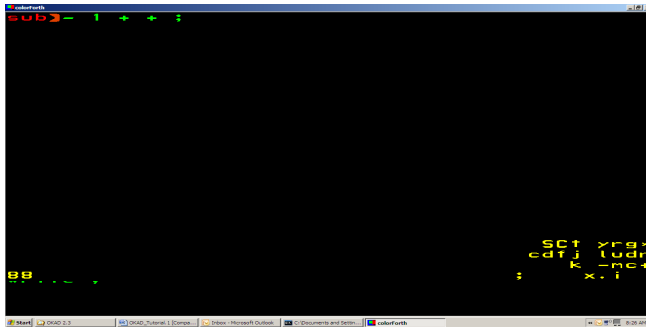


## 1.4 How to modify ColorForth and save the result.

In this example a subtraction word will be written and saved in ColorForth. A load instruction will be placed in block 18 so that the word can be executed whenever ColorForth is launched.

Search for an empty block and enter the following subtraction equation.

**sub - 1 + + ;**



Load the block and test the function.

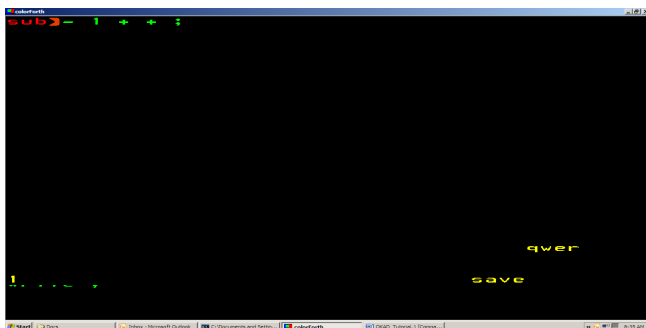
**88 load**

**4 3 sub**



Once you are convinced that the function works save it. If the save command is not executed before closing a ColorForth session, all changes will be lost.

**save**



As mentioned in example 1.1, block 18 is the first block that is loaded when ColorForth is launched. To make **sub** available the next time ColorForth is launched, a load command will be placed in block 18.

Use **d** and **r** from the second row of the editor control panel to position the cursor at the end of the block.

## 18 edit

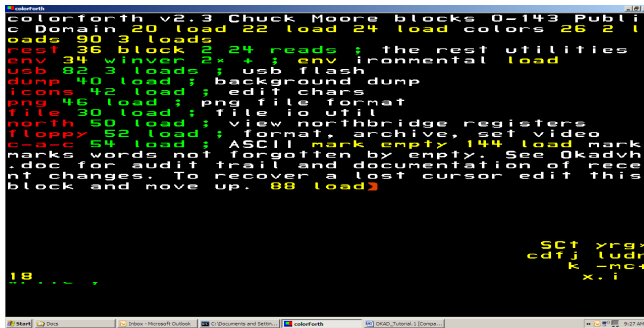
**d** (until bottom is reached)

**r** (until end of line is reached)



Select **y**, yellow, from the editor control panel and load the block containing **sub**.

## 88 load



Exit the editor and type **save**. Exit ColorForth by executing **bye**. When ColorForth is restarted the sub word will be available.

## 1.5 Verifying changes in source code.

Changes in source code can be verified before they are saved to OkadWork.cf. This is done by comparing code in the current ColorForth session to OkadBack.cf. For this example, begin by backing up the current source.

### 0 !back

Add comments to three blocks of your choice. 22, 24, and 26 will be used in the example.

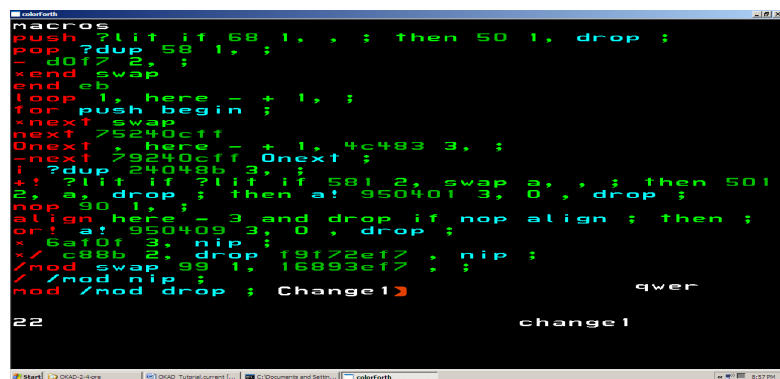
### edit 22

**d** and **l** (until end of block)

**C**

### change1





```
macros
push ?lit if 68 1. . ; then 50 1. drop ;
pop ?dup 58 1. ;
- do f 2. ;
xend swap
end eb
loop 1, here - + 1. ;
for push begin ;
xnext swap
next 75240c1f
Onext , here - + 1. 4c483 3. ;
- next 79240c1f Onext ;
1 ?dup 24048b 3. ;
+ ! ?lit if ?lit if 58 1. 2. swap a. , ; then 50 1
2. a. drop ; then a! 950401 3. 0. drop ;
nop 90 1. ;
align here - 3 and drop if nop align ; then ;
or ! a! 950409 3. 0. drop ;
x 6a10f 3. nip ;
x c88b 2. drop f9f72ef7 . nip ;
/mod swap 99 1. 16893ef7 . ;
/mod nip ;
mod /mod drop ; Change1
qwer

22
change1
```

esc

Continue to make similar changes to the next two blocks.

+

d and l (until end of block)

C

change2

esc

+

d and l (until end of block)

C

Change3

esc

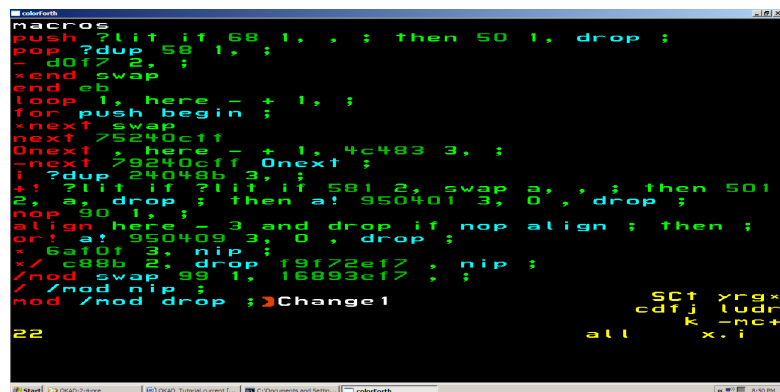
. (exit the editor)

The changes made to ColorForth can now be compared to the backup. Load the disk utility at block 120 and use the **check** function to find changes.

120 load

check

all



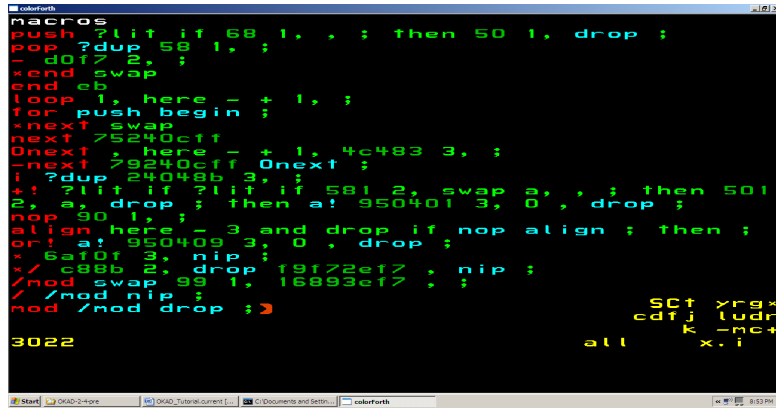
```
macros
push ?lit if 68 1. . ; then 50 1. drop ;
pop ?dup 58 1. ;
- do f 2. ;
xend swap
end eb
loop 1, here - + 1. ;
for push begin ;
xnext swap
next 75240c1f
Onext , here - + 1. 4c483 3. ;
- next 79240c1f Onext ;
1 ?dup 24048b 3. ;
+ ! ?lit if ?lit if 58 1. 2. swap a. , ; then 50 1
2. a. drop ; then a! 950401 3. 0. drop ;
nop 90 1. ;
align here - 3 and drop if nop align ; then ;
or ! a! 950409 3. 0. drop ;
x 6a10f 3. nip ;
x c88b 2. drop f9f72ef7 . nip ;
/mod swap 99 1. 16893ef7 . ;
/mod nip ;
mod /mod drop ; Change1
qwer

22
change1

SCt yrgx
cdf j lldr
k -mc+
all x.i
```

To compare differences, select **j** from the control panel. This will toggle between the current block and the current block + 3000 where the backed up version of the current block is stored. Exit the editor and proceed to the next change by using the **g** command. This command will bring up the editor again. To view the change, select **j** from the editor control panel. When the last change is reached the **g** command will not advance or enter the editor.

**j**



```
macros
push 7lit if 68 1, . ; then 50 1, drop ;
pop ?dup 58 1, ;
-d0f7 2, ;
xend swap
end eb
loop 1, here - + 1, ;
for push begin ;
xnext swap
next 75240c1f
Onext 1, here - + 1, 4c483 3, ;
-nxt 79240c1f Onext ;
i ?dup 24048b 3, ;
i ?lit if 7lit if 58 1, swap a, , ; then 50 1
2, a, drop ; then a! 950401 3, 0', drop ;
nop 90 1, ;
align here - 3 and drop if nop align ; then ;
or! a! 950409 3, 0 , drop ;
x 6a10f 3, nip ;
x/c88b 2, drop 19f72ef7 , nip ;
/mod swap 99 1, 16893ef7 , ;
/mod nip ;
/mod drop ;
SCt yrgx
cdfj lldr
k -mc+
all x.i
3022
```

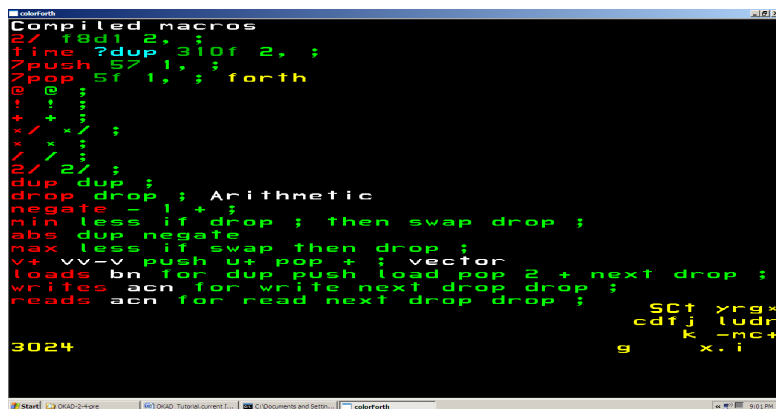
.

**g**



```
Compiled macros
2/ 18d1 2, ;
time ?dup 310f 2, ;
7push 57 1, ;
7pop 5f 1, ; forth
@ 0 ;
t ;
+ ;
x/ x/ ;
x x ;
2/ 2/ ;
dup dup ;
drop drop ; Arithmetic
negate - 1 + ;
min less if drop ; then swap drop ;
abs dup negate
max less if swap then drop ;
v+ vv-v push u+ pop + ; vector
loads bn for dup push load pop 2 + next drop ;
writes acn for write next drop drop ;
reads acn for read next drop drop ; Change2
SCt yrgx
cdfj lldr
k -mc+
g x.i
24
```

**j**



```
Compiled macros
2/ 18d1 2, ;
time ?dup 310f 2, ;
7push 57 1, ;
7pop 5f 1, ; forth
@ 0 ;
t ;
+ ;
x/ x/ ;
x x ;
2/ 2/ ;
dup dup ;
drop drop ; Arithmetic
negate - 1 + ;
min less if drop ; then swap drop ;
abs dup negate
max less if swap then drop ;
v+ vv-v push u+ pop + ; vector
loads bn for dup push load pop 2 + next drop ;
writes acn for write next drop drop ;
reads acn for read next drop drop ;
SCt yrgx
cdfj lldr
k -mc+
g x.i
3024
```

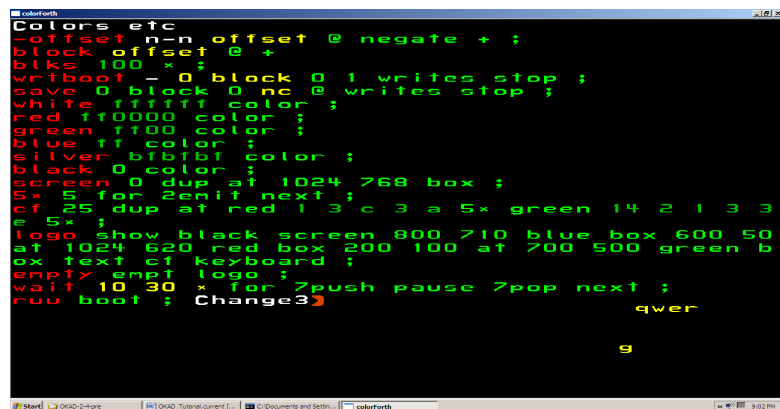
.

**g**

j

.

g



The screenshot shows a terminal window titled 'colorForth'. The code being executed is as follows:

```
Colors etc
-offset n-n offset @ negate + ;
block offset @ +
blks 100 * ;
writeboot - 0 block 0 1 writes stop ;
save 0 block 0 nc @ writes stop ;
white fffffff color ;
red ff0000 color ;
green ff00 color ;
blue ff color ;
silver bfbfbf color ;
black 0 color ;
screen 0 dup at 1024 768 box ;
5* 5 for 2emit next ;
cf 25 dup at red 1 3 c 3 a 5* green 14 2 1 3 3
e 5* ;
logo show black screen 800 710 blue box 600 50
at 1024 620 red box 200 100 at 700 500 green b
ox text cf keyboard ;
empty empty logo ;
wait 10 30 * for 7push pause 7pop next ;
run boot ; Change3
```

The graphical output shows a black screen with a blue box at the bottom right, a red box at the bottom left, and a green box at the bottom center. The text 'qwer' is displayed in the top right corner, and the letter 'g' is displayed in the bottom center.

The changes have all been verified.